Contribution à l'étude de l'apprentissage de la programmation en grande section et en cours préparatoire, à travers le logiciel ScratchJr : une approche didactique exploratoire

Sevastiani Touloupaki

Université Paris Cité

En cotutelle avec Université de Patras

École doctorale ED623 Savoirs, Sciences, Education

Laboratoire EDA-Éducation Discours et Apprentissages (EA 4071)

Contribution à l'étude de l'apprentissage de la programmation en grande section et en cours préparatoire, à travers le logiciel ScratchJr : Une approche didactique exploratoire

Par **Sevastiani Touloupaki**

Thèse de doctorat de Sciences de l'Éducation et de la Formation

Dirigée par **Georges-Louis Baron**

Et par **Vassilis Komis**

Présentée et soutenue publiquement le 12 juillet 2023

Devant un jury composé de :

Georges-Louis Baron, professeur émérite, Université Paris Cité (Directeur)

Marina Bers, professor, Boston College Chestnutt Hill, MA (Examinatrice)

Eric Bruillard, professeur, Université Paris Cité (Examinateur)

Angelique Dimitracopoulou, professor, University of the Aegean (Rapportrice)

Béatrice Drot-Delange, professeure, Université Clermont Auvergne (Examinatrice)

Vassilis Komis, professor, University of Patras (Co-Directeur)

Konstantinos Ravanis, professor, University of Patras (Examinateur)

Margarida Romero, professeure, Université Côte d'Azur (Rapportrice)

This research is co-financed by Greece and the European Union (European Social Fund-ESF) through the Operational Programme «Human Resources Development, Education and Lifelong Learning» in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research – 2nd Cycle" (MIS-5000432), implemented by the State Scholarships Foundation (IKY).

Résumé :

Cette thèse de didactique de l'informatique porte sur l'apprentissage de la programmation par de très jeunes élèves (5 à 7 ans). L'objectif de ce travail à visée exploratoire est d'apporter une contribution à l'étude de l'apprentissage de la programmation dans le milieu de la petite enfance. Nous avons réalisé cette recherche dans des conditions réelles de classe pour comprendre ce que les jeunes élèves sont capables de faire seuls ou aidés en programmation sur ScratchJr et quelles sont leurs difficultés. Nous avons en particulier étudié les stratégies de programmation et le débogage qu'ils mettent en œuvre. Il s'agit d'une étude préparatoire auprès de deux classes de grande section de maternelle en Grèce et d'une étude principale auprès de deux classes de cours préparatoire (CP) en France.

L'étude préparatoire en Grèce a permis de tester le scénario pédagogique que nous avons conçu pour enseigner la programmation sur ScratchJr, afin de l'améliorer pour l'étude principale en France où nous avons formé les enseignantes des classes pour qu'elles puissent l'enseigner. Grâce à cette étude préparatoire, nous avons obtenu une vision globale sur ce que les élèves de grande section de maternelle peuvent faire en programmation sur ScratchJr et les difficultés qu'ils risquent de rencontrer.

Nos résultats montrent qu'ils ont rencontré des difficultés avec l'initialisation de la position des personnages, l'estimation d'une distance pour le déplacement de leurs personnages, la « répétition prédéfinie » et les « messages ». Ils préfèrent créer plusieurs fois le même motif des commandes au lieu d'utiliser la commande de « répétition » et ils arrivent mieux à décrire ce que fait une « répétition prédéfinie d'une seule commande » qu'une « répétition prédéfinie d'une séquence des commandes ». Nous avons également repéré qu'il existe un écart entre l'expérience physique et la programmation sur ScratchJr et cela peut constituer un obstacle pour la compréhension de la fonctionnalité de « messages ».

Suite à une analyse fine de l'activité de programmation de quatre élèves de CP, nous avons notamment mis en évidence cinq stratégies de programmation différentes (« pas à pas », « sous-programme », « planification partielle », « planification complète », « recommencer au début ») et des capacités de débogage de types différents (« optimal », « efficace », « non efficace ») selon différents patterns. Nous avons aussi repéré des difficultés dans l'utilisation de la répétition et des messages, liés à des connaissances encore peu maîtrisées comme la gestion du temps, le dénombrement, la causalité, etc. Au-delà de ces résultats, une méthodologie d'analyse de l'activité de programmation des jeunes élèves sur un environnement de programmation visuelle tel que ScratchJr a également été conçue et validée.

Mots-clefs : Didactique de l'informatique, Programmation, Apprentissage de la programmation, Enseignement de la programmation, ScratchJr, École primaire, Programmation visuelle, Cycle 1, Cycle 2, Scénario pédagogique.

Abstract :

This thesis focuses on the learning of programming by very young children (5 to 7 years old). The aim of this exploratory work is to make a contribution to the study of programming learning in early childhood. We conducted this research in real classroom conditions in order to understand what students are able to do, alone or assisted, in programming with ScratchJr, as well as the difficulties they face. Their programming and debugging strategies were notably studied. To begin with, a preparatory study was carried out in two classes of kindergarten in Greece, followed by a main study in two classes of primary schools (CP) in France.

The preparatory study in Greece allowed us to test the designed educational scenario for teaching programming with ScratchJr. Its subsequent improved version was applied in the main study in France, in which case teachers were trained to teach it. Thanks to this preparatory study, we have obtained a global view on what kindergarten students can do in programming with ScratchJr and the difficulties they are likely to encounter. Our results show that kindergarten students struggle with initializing the position of the characters, with estimating the distances for moving their characters, as well as with the use of « predefined repetition » and « messages ». They prefer to create the repeated pattern of commands multiple times instead of using the « repeat » command, and they are better at describing what a « predefined repeat of a single command » does than a « predefined repeat of a sequence of commands ». We have also identified that there is a gap between physical experience and programming with ScratchJr and this can be an obstacle to understanding the « messages » functionality.

A methodology for analysing young students' programming activity on a graphical programming environment such as ScratchJr was also developped. Following a detailed analysis of the programming activity of four first graders, we have highlighted five different programming strategies (i.e. « step by step », « sub-program », « partial planning » and « complete planning » et « start over ») and different types of debugging abilities (i.e. « optimal », « efficient » and « non-efficient ») according to different patterns. We have also identified difficulties in the use of « repetition » and « messages », linked to, still, poorly mastered knowledge such as time management, enumeration, causality, etc.

Keywords: Computer science, Computer programming, Programming learning, Programming teaching, ScratchJr, Early childhood education, Visual programming environnements, Educationl scenarios.

Περίληψη

Η διατριβή εντάσσεται στο πεδίο της Διδακτικής της Πληροφορικής και έχει ως αντικείμενο μελέτης την εκμάθηση του προγραμματισμού από μαθητές προσχολικής και πρώτης σχολικής ηλικίας μέσω της χρήσης του περιβάλλοντος οπτικού προγραμματισμού ScratchJr σε πραγματικές συνθήκες τάξης, σε δυο διαφορετικά σχολικά πλαίσια της Ελλάδας και της Γαλλίας. Η διατριβή διενεργείται μεταξύ του Πανεπιστημίου Πατρών και του Université Paris Cité της Γαλλίας. Στόχος της διατριβής είναι να μελετήσει και να περιγράψει τί κάνουν οι μαθητές νηπίου και πρώτης δημοτικού, μόνοι τους ή με βοήθεια, όταν προγραμματίζουν κάνοντας χρήση του λογισμικού ScratchJr και ποιες δυσκολίες αντιμετωπίζουν. Πιο συγκεκριμένα, η παρούσα διατριβή εξετάζει τις προγραμματιστικές στρατηγικές που χρησιμοποιούν οι μαθητές αλλά και τον τρόπο με τον οποίο πραγματοποιούν την εκσφαλμάτωση των προγραμμάτων τους.

Από τα μέσα της δεκαετίας 2000, παρατηρείται μια αναζωπύρωση του ενδιαφέροντος γύρω από την εισαγωγή της πληροφορικής και συγκεκριμένα μιας εκ των πτυχών της, του προγραμματισμού στα αναλυτικά προγράμματα σπουδών πολλών χωρών σε όλο τον κόσμο. Ενδεικτικά στην Γαλλία, ο προγραμματισμός εισήχθη στο αναλυτικό πρόγραμμα σπουδών του δημοτικού το 2016. Εντούτοις, τα αναλυτικά προγραμματα του 2020 αναφέρονται στον προγραμματισμό αλλά σε πολύ μικρότερο βαθμό απο πρίν. Στην Ελλάδα, ο προγραμματισμός έχει ενταχθεί στο αναλυτικό προγραμμα σπουδών του δημοτικού από το 2003 και στο αναλυτικό πρόγραμμα σπουδών του νηπιαγωγείου από το 2011. Στα νέα αναλυτικά προγράμματα σπουδών του 2021 για το δημοτικό και το νηπιαγωγείο, ο προγραμματισμός κατέχει περίοπτη θέση ενώ γίνεται λόγος και για την ανάπτυξη της πληροφορικής σκέψης. Παράλληλα το ερευνητικό ενδιαφέρον γύρω από την εισαγωγή των παιδιών στον προγραμματισμό και την ψηφιακή δημιουργία έχει ενταθεί.

Η αναζωπήρωση του ενδιαφέροντος που παρατηρείται τα τελευταία χρόνια γύρω από την εισαγωγή των μαθητών στον προγραμματισμό ήδη από την προσχολική ηλικία τροφοδοτείται κατά κύριο λόγο από την εμφάνιση αναπτυξιακά κατάλληλων περιβαλλόντων οπτικού προγραμματισμού. Τα συγκεκριμένα περιβάλλοντα και ειδικότερα οι οπτικές γλώσσες προγραμματισμού εξελίχθηκαν προκειμένου να ανταποκριθούν καλύτερα στα χαρακτηριστικά και τις ανάγκες των μαθητών νηπιαγωγείου και πρώτης δημοτικού. Ακολούθησε μια άνθιση στο συγκεκριμένο τομέα, με αποτέλεσμα σήμερα να συναντάμε πληθώρα επιλογών σε ότι αφορά τις γλώσσες προγραμματισμού για παιδιά. Οι περισσότερες μάλιστα έχουν την μορφή εφαρμογών που υποστηρίζονται από φορητές συσκευές. Η εμφάνιση τέτοιων περιβαλλόντων προγραμματισμού διεύρυνε σημαντικά το πεδίο έρευνας γύρω από την μελέτη της εκμάθησης του προγραμματισμού από τα παιδιά.

Η επισκόπηση της διαθέσιμης βιβλιογραφίας ανέδειξε ότι η εκμάθηση του προγραμματισμού από τα παιδιά είχε απασχολήσει πολύ την ερευνητική κοινότητα στο παρελθόν. Συγκεκριμένα η πλειονότητα των ερευνών γύρω από το θέμα αυτό τοποθετείται στην δεκαετία του 1980 και το προγραμματιστικό περιβάλλον που χρησιμοποιείται περισσότερο εκείνη την περίοδο είναι αυτό της LOGO. Οι συγκεκριμένες έρευνες έχουν κατά κύριο λόγο πραγματοποιηθεί με

μαθητές μεγαλύτερων ηλικιών και ασχολούνται με τις προγραμματιστικές στρατηγικές που χρησιμοποιούν οι μαθητές, τα στάδια από τα οποία περνούν καθώς μαθαίνουν να προγραμματιζουν στο περιβάλλον της LOGO, τις δυσκολίες που αντιμετωπίζουν, τα λάθη που κάνουν, αλλά και τον τρόπο με τον οποίο εκσφαλματώνουν τα προγραμματά τους. Η επισκόπηση της βιβλιογραφίας ανέδειξε επιπλέον μια σειρά ερευνών που έχουν πραγματοποιηθεί τα τελευταία χρόνια και μελετούν την εκμάθηση του προγραμματισμού από παιδιά προσχολικής και πρώτης σχολικής ηλικίας με την βοήθεια αναπτυξιακά κατάλληλων περιβαλλόντων οπτικού προγραμματισμού.

Το ScratchJr αποτελεί ένα από τα πιο χαρακτηριστικά παραδείγματα περιβαλλόντων οπτικού προγραμματισμού. Στο περιβάλλον αυτό, τα παιδιά έχουν τη δυνατότητα να προγραμματίζουν διαδραστικές ιστορίες σε ένα πλαίσιο ανάπτυξης της δημιουργικής τους έκφρασης. Η γλώσσα προγραμματισμού, το περιβάλλον εργασίας, καθώς και οι χαρακτήρες διαμορφώθηκαν με σκοπό να ανταποκρίνονται αναπτυξιακά στις ανάγκες των παιδιών ηλικίας από 5 έως 7 ετών. Εξετάζοντας την διαθέσιμη βιβλιογραφία στην οποία γίνεται χρήση του εν λόγω λογισμικού, διαπιστώθηκε ότι δεν παρέχονται αρκετές πληροφοριές αναφορικά με το τί κάνουν οι μαθητές 5 έως 7 ετών όταν προγραμματίζουν με το λογισμικό, τί είδους δυσκολίες αντιμετωπίζουν, τί είδους προγραμματιστικές στρατηγικές χρησιμοποποιούν και πώς εκσφαλματώνουν τα προγραμματά τους. Για το λόγο αυτό η παρούσα διατριβή επιδιώκει να δώσει απαντήσεις σε αυτά τα ερωτήματα και να ενημερώσει κατά συνέπεια την υπάρχουσα βιβλιογραφία.

Η δραστηριότητα του προγραμματισμού είναι άμεσα συνδεδεμένη με την διαδικασία επίλυσης προβλημάτων. Εξαιτίας της πολυπλοκότητας που χαρακτηρίζει την δραστηριότητα αυτή δεν εντοπίστηκε ένα θεωρητικό μοντέλο που να επιτρέπει την μελέτη όλων των πτυχών του φαινομένου αυτού, κυρίως όσον αφορά την προσχολική και πρώτη σχολική ηλικία. Για το λόγο αυτό επιλέχθηκε συνδυασμός θεωρητικών μοντέλων ώστε να μελετηθεί αποτελεσματικά η προβληματική της έρευνας. Ειδικότερα ο συνδυασμός θεωρητικών μοντέλων που χρησιμοποιήθηκε συνίσταται από τις φάσεις που απαρτίζουν την δραστηριότητα του προγραμματισμού, τις στρατηγικές προγραμματισμού καθώς και την διαδικασία εκσφαλμάτωσης.

Για την πραγματοποίηση της παρούσας μελέτης επιλέχθηκε η ποιοτική μεθοδολογία έρευνας προκειμένου να μελετηθεί και να κατανοηθεί σε βάθος το υπό μελέτη φαινόμενο. Ειδικότερα επιλέχθηκε η μελέτη περίπτωσης ως η καταλληλότερη ερευνητική στρατηγική. Η έρευνα διεξήχθη σε δυο κύκλους. Μια πιλοτική έρευνα πραγματοποίηθηκε στην Ελλάδα σε δυο τάξεις νηπιαγωγείου, στην Πάτρα, ενώ η κύρια έρευνα πραγματοποιήθηκε στην Γαλλία σε δυο τάξεις πρώτης δημοτικού, στα περίχωρα του Παρισιού. Κάθε μία από τις τάξεις που επιλέχθηκαν αποτελεί και μια διαφορετική περίπτωση υπό μελέτη.

Η πιλοτική έρευνα διήρκησε δυο μήνες και σε αυτήν συμμετείχαν 22 μαθητές νηπίου, 10 μαθητές από την πρώτη τάξη (GS1) και 12 μαθητές από την δεύτερη (GS2). Η κύρια έρευνα διήρκησε τέσσερις μήνες και σε αυτήν συμμετείχαν 50 μαθητές πρώτης δημοτικού, 26 μαθητές από την πρώτη τάξη (CP1) και 24 μαθητές από την δεύτερη τάξη (CP2).

Δύο εκπαιδευτικά σενάρια, ένα στα ελληνικά και ένα στα γαλλικά, αναπτύχθηκαν για τις ανάγκες της παρούσας έρευνας. Στην πιλοτική έρευνα αναλάβαμε να εφαρμόσουμε το εκπαιδευτικό σενάριο ενώ στην κύρια έρευνα επιμορφώσαμε τις εκπαιδευτικούς καθώς επιθυμούσαν να αναλάβουν την εφαρμογή του. Κατά την εφαρμογή του εκπαιδευτικού σεναρίου στην κύρια έρευνα συμμετείχαμε ενεργά στην διαδικασία παρέχοντας βοήθεια στις εκπαιδευτικούς, όποτε την χρειάζονταν. Τα εκπαιδευτικά σενάρια ήταν σχεδιασμένα με τέτοιο τρόπο ώστε σε κάθε δραστηριότητα να προτείνεται στους μαθητές ένα πρόβλημα προς επίλυση προκειμένου να έχουν την δυνατότητα να χρησιμοποιήσουν τις εντολές που έμαθαν, κατανοώντας έτσι καλύτερα την λειτουργία τους.

Στο τέλος κάθε σεναρίου οι μαθητές συμμετείχαν σε δυο δραστηριότητες αξιολογήσης. Οι δραστηριότητες αυτές περιλάμβαναν δυο προγραμματιστικά προβλήματα προς επίλυση και μια ημιδομημένη συνέντευξη. Ειδικότερα, οι μαθητές καλούνταν να επιλύσουν μόνοι ή με βοήθεια δυο προγραμματιστικά προβλήματα στο ScratchJr, ενώ η ημιδομημένη συνέντευξη είχε στόχο να τους δώσει την δυνατότητα να εκφραστούν προφορικά γύρω από την διεπιφάνεια χρήσης του λογισμικού και τις προγραμματιστικές εντολές που είχαν διδαχτεί, ώστε να εντοπιστούν τυχόν δυσκολίες και να ληφθούν υπόψη για την βελτίωση του εκπαιδευτικού σεναρίου.

Κατά την πιλοτική έρευνα λόγω καθυστερημένης πρόσβασης στα ελληνικά σχολεία χρειάστηκε να γίνουν μια σειρά από προσαρμογές στο εκπαιδευτικό σενάριο, στο δείγμα υπο μελέτη και κατά συνέπεια στην συλλογή δεδομένων. Ενδεικτικά η δραστηριότητα αξιολόγησης που αφορούσε την επίλυση των δυο προγραμματιστικών προβλημάτων δεν πραγματοποιήθηκε ατομικά με κάθε μαθητή, όπως ήταν ο αρχικός σχεδιασμός, αλλά με τέσσερις μαθητές κάθε φορά. Αντιθέτως, στην κύρια έρευνα η συγκεκριμένη δραστηριότητα πραγματοποίηθηκε ατομικά για τον κάθε μαθητή. Επιπλέον στην κύρια έρευνα είχαμε την δυνατότητα να εφαρμόσουμε ένα σενάριο 13 παρεμβάσεων, αφιερώνοντας έτσι μια επιπλέον δραστηριότητα στις εντολές που δημιουργούσαν πιο πολλά προβλήματα στους μαθητές όπως η επανάληψη και τα μηνύματα.

Χάρη στην πιλοτική έρευνα είχαμε την δυνατότητα να εφαρμόσουμε το εκπαιδευτικό σενάριο σε πραγματικές συνθήκες τάξης με μαθητές νηπίου εντοπίζοντας έτσι τα δυνατά και τα αδύναμα σημεία του, προκειμένου να το βελτιώσουμε για τις ανάγκες της κύριας έρευνας. Επίσης, η πιλοτική έρευνα επέτρεψε να εντοπιστούν τυχόν προβλήματα στη μεθοδολογία διεξαγωγής της έρευνας προκειμένου αυτά να αποφευχθούν κατά την διεξαγωγή της κύριας έρευνας.

Αναλύοντας τις σημειώσεις από την παρατήρηση στο πεδίο της έρευνας κατά την διάρκεια εφαρμογής του εκπαιδευτικού σεναρίου, τα τελικά προγράμματα αλλά και τις απαντήσεις που έδωσαν οι 12 μαθητές της τάξης GS2 στις ερωτήσεις που αφορούσαν τις εντολές της επανάληψης και των μηνυμάτων κατορθώσαμε να αποκτήσουμε μια ολοκληρωμένη εικόνα σχετικά με το τί κάνουν οι μαθητές νηπιαγωγείου όταν προγραμματίζουν μόνοι τους ή με βοήθεια στο λογισμικό ScratchJr και τί δυσκολίες αντιμετωπίζουν.

Ειδικότερα εντοπίστηκε ότι η αρχικοποίηση θέσης χαρακτήρα, η εκτίμηση της απόστασης που πρέπει να μετακινηθούν οι χαρακτήρες στην σκηνή, η επανάληψη και τα μηνύματα δημιουργούν τα περισσότερα προβλήματα στους μαθητές νηπίου. Τα αποτελέσματα της πιλοτική έρευνας στην Ελλάδα αναδεικνύουν επίσης ότι οι μαθητές νηπίου προτιμούν να δημιουργήσουν τρεις φορές το μοτίβο εντολών που επαναλαμβάνεται αντί να δημιουργήσουν μια φορά το μοτίβο εντολών και να χρησιμοποίησουν την εντολή της επανάληψης. Επίσης εντοπίστηκε ότι οι μαθητές νηπίου καταφέρνουν καλύτερα να περιγράψουν την λειτουργία της εντολής της επανάληψης όταν επαναλαμβάνεται μια εντολή απ'ότι όταν επαναλαμβάνεται ένα μοτιβο εντολών. Αναφορικά με τις εντολές των μηνυμάτων διαπίστωθηκε ότι υπάρχει μια απόσταση ανάμεσα στις εμπειρίες των μαθητών με τα μηνύματα από την καθημερινή ζωή και στα μηνύματα στον προγραμματισμό στο ScratchJr και αυτό μπορεί να λειτουργήσει ως εμπόδιο στην κατανόηση της λειτουργίας των μηνυμάτων στον προγραμματισμό.

Τα δεδομένα που συλλέχθηκαν κατά την διάρκεια της κύριας έρευνας στην Γαλλία μας επέτρεψαν να μελετήσουμε όχι μόνο τί μπορούν να κάνουν οι μαθητές πρώτης δημοτικού όταν προγραμματίζουν μόνοι τους ή με βοήθεια στο ScratchJr και τι δυσκολίες αντιμετωπίζουν, αλλά και την διαδικασία επίλυσης προβλήματος προκειμένου να εντοπίσουμε τις προγραμματιστικές στρατηγικές που χρησιμοποιούν αλλά και τον τρόπο με τον οποίο πραγματοποιούν την εκσφαλμάτωση των προγραμμάτων τους.

Τέσσερις μαθητές πρώτης δημοτικού με όσο το δυνατόν διαφορετικές προγραμματιστικές συμπεριφορές επιλέχθηκαν από την τάξη CP1 προκειμένου να πραγματοποιηθεί μια σε βάθος ανάλυση της προγραμματιστικής τους δραστηριότητας. Συγκεκριμένα τέθηκαν υπο ανάλυση οι βιντεοσκοπήσεις της δραστηριότητας αξιολόγησης και τα τελικά προγραμματα των 4 μαθητών.

Στη συνέχεια αναπτύχθηκε πρωτότυπη μεθοδολογία για την ανάλυση της προγραμματιστικής δραστηριότητας των παιδιών. Επιλέξαμε αρχικά να επικεντρωθούμε στις ενέργειες που πραγματοποιούν οι μαθητές κατά την διάρκεια της επίλυσης των προβλημάτων αξιολόγησης όπως αυτές αποτυπώνονταν μέσα στις βιντεοσκοπήσεις. Για να μπορέσουμε να αναλύσουμε αποτελεσματικά τις ενέργειες των μαθητών έπρεπε σε ένα πρώτο στάδιο να τις παρουσιάσουμε με κατάλληλο τρόπο. Ειδικότερα καταγράψαμε αναλυτικά τις ενέργειες των μαθητών χρησιμοποιώντας τις βιντεοσκοπήσεις και δημιούργησαμε ένα αρχείο κειμένου στο οποίο παράλληλα ενσωμάτωσαμε στιγμιότυπα οθόνης του τάμπλετ των μαθητων. Με αυτόν τον τρόπο κατορθώσαμε να μελετήσουμε αποτελεσματικά την πορεία από την οποία περνούν οι μαθητές για να επιλύσουν τα δύο προβλήματα αξιολόγησης.

Έπειτα αποφάσισαμε να δημιουργήσουμε διαγράμματα προκειμένου να αναπαραστήσουμε την αλληλουχία ενεργειών που πραγματοποιούν οι μαθητές για να επιλύσουν καθένα από τα δυο προβλήματα. Για να δημιουργήσουμε τα διαγράμματα ενεργειών πραγματοποιήσαμε μια πρώτη κωδικοποίηση των ενεργειών των μαθητών (επαγωγική κωδικοποίηση) αναπτύσσοντας παράλληλα μια σειρά από λειτουργικούς ορισμούς για τους κωδικούς που εντόπιστηκαν. Συγκεκριμένα εντοπίστηκαν πέντε κατηγορίες ενεργειών : διεπιφάνεια χρήσης,

εντολές, διορθώσεις, αρχικοποίηση, εκτέλεση. Κάθε κατηγορία ενεργειών περιλαμβάνει μια σειρά από παραδείγματα που εντοπίστηκαν μέσα στις βιντεοσκοπήσεις.

Επιπλέον καταμετρήθηκε ο αριθμός ενεργειών που πραγματοποίησαν οι μαθητές στην κάθε κατηγορία ενεργειών καθώς και ο συνολικός αριθμός ενεργειών για την επίλυση καθενός από τα δυο προβλήματα. Επίσης, καταμετρήθηκε ο χρόνος που αφιέρωσαν οι μαθητές στην επίλυση καθενός από τα δύο προβλήματα. Ακόμα καταγράφηκε ο αριθμός και ο τύπος βοηθειών που δόθηκαν κατά την διάρκεια της αξιολόγησης και μελετήθηκε η συμπεριφορά των μαθητών μετά την παροχή βοήθειας προκειμένου να εντοπιστεί αν αυτή διαφοροποιείται ή όχι.

Έπειτα, συγκρίναμε, με την βοήθεια των διαγραμμάτων ενεργειών, την αλληλουχία ενεργειών που ακολούθησαν οι μαθητές για να επιλύσουν τα δυο προβλήματα προκειμένου να εντοπίσουμε τυχόν ομοιότητες, διαφορές ή επαναλαμβανόμενα μοτίβα. Επίσης, αναπτύχθηκε κατάλληλο λεξιλόγιο τυπολογιών για την ανάλυση των δεδομένων της έρευνας το οποίο προέκυψε από συνδυασμό θεωρητικών μοντέλων, εξαιτίας της πολυπλοκότητας που χαρακτηρίζει την δραστηριότητα του προγραμματισμού.

Ακολούθησε η προσπάθεια εφαρμογής του λεξιλογίου της βιβλιογραφίας στα δεδομένα της έρευνας. Ειδικότερα τα θεωρητικά μοντέλα πρότειναν μεν σειρά τυπολογιών αλλά δεν πρότειναν, με εξαίρεση το θεωρητικό μοντέλο για την εκσφαλμάτωση, μια μέθοδο για την αναγνώριση των τυπολογιών μέσα στα δεδομένα. Κατορθώσαμε να επιλύσουμε το συγκεκριμένο πρόβλημα εντοπίζοντας μια σειρά στοιχείων κλειδιά που μας βοήθησαν να χαρακτηρήσουμε τον τρόπο με τον οποίο προγραμματίζουν οι μαθητές. Παράλληλα δημιουργήθηκαν νέες τυπολογίες όταν οι τυπολογίες της υπάρχουσας βιβλιογραφίας δεν προσφέρονταν για την περιγραφή των δεδομένων της έρευνας. Εφαρμόσαμε κατ’αυτόν τον τρόπο επαγωγική και παραγωγική κωδικοποίηση.

Μελετήθηκαν επίσης τα είδη λαθών που πραγματοποίησαν οι μαθητές, το αν είχαν επίγνωση του λάθους που είχαν κάνει, το αν υπήρχαν στιγμές κατά την διάρκεια των βιντεοσκοπήσεων στις οποίες φαινόταν να αντιμετωπίζουν δυσκολία αλλά και η διαδικασία εκσφαλμάτωσης. Έγινε επίσης διαφοροποίηση ανάμεσα στα λάθη που κατορθώνουν να διορθώσουν οι μαθητές κατά την διάρκεια επίλυσης των προβλημάτων αξιολόγησης και σε αυτά που παραμένουν μέσα στα προγράμματα τους μέχρι το τέλος, καθώς αδυνατούν να τα διορθώσουν. Τα διαγράμματα ενεργειών έπαιξαν πολύ σημαντικό ρόλο στον εντοπισμό των μοτίβων εκσφαλμάτωσης που χρησιμοποιούσαν οι μαθητές.

Μέσω της ποιοτικής ανάλυσης της προγραμματιστικής δραστηριότητας των 4 μαθητών πρώτης δημοτικού της τάξης CP1, κατορθώσαμε να κατανοήσουμε σε βάθος και να περιγράψουμε αποτελεσματικά τί μπορούν να κάνουν οι μαθητές 6 έως 7 ετών μόνοι τους ή με βοήθεια όταν προγραμματίζουν στο λογισμικό ScratchJr, τί δυσκολίες αντιμετωπίζουν αλλά και τί είδους προγραμματιστικές στρατηγικές χρησιμοποιούν και πώς πραγματοποιούν την εκσφαλμάτωση των προγραμμάτων τους.

Ειδικότερα εντοπίστηκαν 5 διαφορετικές προγραμματιστικές στρατηγικές και συνδυασμοί αυτών. Επίσης εντοπίστηκαν 4 στρατηγικές μετακίνησης χαρακτήρων, δυο εκ των οποίων δημιουργήθηκαν απο τους ίδιους τους μαθητές. Ακόμα τα αποτελέσματα της κύριας έρευνας ανέδειξαν 3 διαφορετικούς τύπους εκσφαλμάτωσης και 3 διαφορετικά μοτίβα εκσφαλμάτωσης. Επίσης εντοπίστηκαν διαφορετικές συμπεριφορές εκσφαλμάτωσης.

Επιπροσθέτως διαπιστώσαμε ότι η αρχικοποίηση θέσης χαρακτήρα, η εκτίμηση της απόστασης που πρέπει να μετακινηθούν οι χαρακτήρες στην σκηνή, όπως επίσης οι εντολές της επανάληψης και των μηνυμάτων δημιούργησαν δυσκολίες στους μαθητές πρώτης δημοτικού. Τα αποτελέσματα της κύριας έρευνας ανέδειξαν επίσης ότι 2 μαθητές πρώτης δημοτικού προτίμησαν να δημιουργήσουν τρεις φορές το μοτίβο εντολών που επαναλαμβάνεται αντί να δημιουργήσουν μια φορά το μοτίβο εντολών και να χρησιμοποίησουν την εντολή της επανάληψης.

Τέλος, συγκρίνοντας την προγραμματιστική δραστηριότητα των 4 μαθητών πρώτης δημοτικού εντοπίσαμε μια σειρά απο ομοιότητες και διαφορές και βασιζόμενη σε αυτές πρότειναμε μια σειρά από χαρακτηριστικά τα οποία δύνανται να παρουσιάσουν οι μαθητές που έχουν κατανοήσει πώς λειτουργεί ο προγραμματισμός στο ScratchJr και μια σειρά από χαρακτηριστικά τα οποία δύνανται να παρουσιάσουν οι μαθητές που αντιμετωπίζουν δυσκολίες με τον προγραμματισμό στο ScratchJr.

Remerciements

Cette thèse était pour moi un vrai marathon avec beaucoup d'obstacles qui m'ont parfois semblé infranchissables, mais en arrivant à la fin j'ai l'impression que j'ai appris énormément de choses à la fois sur comment mener une recherche mais aussi sur moi-même.

Je tiens avant tout à remercier mes directeurs Georges-Louis Baron et Vassilis Komis pour leur accompagnement, leurs conseils et leur soutien tout au long de cette expérience. Je les remercie également de m'avoir fait confiance pour réaliser ce travail de recherche.

Je remercie chaleureusement les membres du jury Marina Bers, Eric Bruillard, Angelique Dimitrakopoulou, Béatrice Drot-Delange, Konstantinos Ravanis et Margarida Romero de m'avoir fait l'honneur de lire et d'évaluer ma thèse.

J'adresse aussi mes remerciements à Eric Bruillard et Laetitia Boulc'h pour leurs conseils et leurs remarques en tant que membres du comité de suivi de ma thèse. Je remercie également Konstantinos Ravanis de m'avoir fait l'honneur de faire partie de ma Τριμελή συμβουλευτική επιτροπή en Grèce.

Je remercie également les membres du laboratoire EDA de m'avoir accueillie avec tant de bienveillance au sein de leur équipe. Merci à Marianne Doury, Mariam Haspekian, Mehdi Khaneboubi, Pascale Kummer et Eric Roditi pour nos échanges scientifiques.

J'adresse un grand merci aux enseignantes en Grèce et en France qui m'ont fait confiance en m'acceptant dans leurs classes et en me permettant de mettre en place ma recherche. Je les remercie également pour leurs idées, leurs conseils et les échanges intéressants que nous avons eus. Je remercie également Florence Force pour son aide et son soutien dans la réalisation de cette recherche.

Je souhaite également remercier Andromahi Filipidi à la fois pour son accueil au sein du laboratoire de didactique des sciences, de mathématiques et de TIC et pour ses conseils sur l'élaboration du scénario pédagogique en Grèce. Merci à Aurélie Beauné, Christelle Combemorel-Pauty et Solène Zablot de m'avoir accueillie au laboratoire EDA au tout début de cette thèse.

Je remercie également toutes les doctorantes qui ont croisé mon chemin pendant le parcours du doctorat aussi bien dans l'organisation du séminaire doctoral, des ateliers doctoraux que de la Journée d'étude des jeunes chercheurs et chercheuses du laboratoire.

Je remercie également Marie Clement pour son soutien tout au long de cette thèse. Nos discussions chaque lundi pendant les confinements étaient une aide précieuse.

Merci également à Mathilde Lefevre pour son soutien, nos discussions du matin et son professionnalisme qui a grandement facilité l'organisation de ma soutenance.

Une pensée particulière à mes ami-e-s du laboratoire EDA : Emma Archimbaud, Charlotte Barbier, Chloé Benoit, Alice Cesbron, Sarah Halimi, Anna Khalonina, Mathilde Lefevre, Diane Liberatore, Marion Mathieu, Vanessa Münch, Céline Robillard, Théophile Robineau, Martina Ronci et Julie Wasilewski. Pour leurs relectures attentives, un grand merci à Emma Archimbaud, Manon Boutin-Charles, Alice Cesbron, Sarah Halimi, Anna Khalonina, Julie Wasilewski et Céline Robillard. L'écriture de ma thèse aurait été beaucoup plus difficile sans le soutien et la présence quotidienne de mes amies de l'équipe de Zoom et Igor et Dahlia pour leurs drôles de promenades devant l'écran.

Plus généralement merci à tous mes ami-e-s d'être restés mes ami-e-s même si je ne viens plus aux soirées depuis quelque temps maintenant (j'arrive !).

Τέλος, θα ήθελα να ευχαριστήσω κάποιους ανθρώπους στην γλώσσα μου. Ενα μεγάλο ευχαριστώ στον Μάνο, που μου έλεγε πάντα ότι θα τα καταφέρω, αλλα και στον φίλο μου τον Μάριο για την υποστήριξη του. Ένα μεγάλο ευχαριστώ στην φίλη μου την Ευγενία για την υποστήριξή της. Δεν θα μπορούσα να ξεχάσω τη Νατάσσα μου, που ήταν για μένα καταφύγιο στις δυσκολες στιγμες. Ενα μεγάλο ευχαριστώ στον Κώστα μου για την αγάπη του, την βοήθεια του και την υπομονή του. Τέλος, θα ήθελα ευχαριστήσω την οικογένεια μου που ήταν πάντα στο πλάι μου σε όλη αυτήν την δοκιμασία, πιστεύοντας σε εμένα και δίνοντας μου κουράγιο.

Sommaire

Introduction

C'est la rédaction de notre mémoire de fin d'études pour obtenir le diplôme de licence du département Enseignement et Éducation en âge préscolaire de l'université d'Athènes et ainsi devenir professeure des écoles en Grèce, spécialisée pour l'âge préscolaire, qui nous a donné le goût de la recherche, notamment à travers son processus : identification d'un problème, définition d'une question précise et entreprise de recherche.

Les discussions que nous avons pu avoir avec nos professeurs de licence concernant la poursuite de nos études nous ont amenée à envisager de poursuivre nos études en France. Nous avons donc réalisé un Master Recherche en Sciences de l'Éducation, Spécialité Éducation et Formation à l'Université Paris Descartes.

Ayant travaillé en tant que professeure des écoles en Grèce, nous avions déjà repéré l'importance des technologies informatiques en classe. N'étant pas précédemment familiarisée avec l'enseignement de l'informatique à l'école primaire pendant nos études à Athènes, nous avons choisi de suivre le cours de Master intitulé « De l'audiovisuel au numérique », du professeur Georges-Louis Baron.

Lors de ce cours, nous avons entendu pour la première fois qu'il était possible d'enseigner la programmation, une activité si complexe, à des très jeunes élèves. Nous avons également eu la possibilité de connaître les travaux de Georges-Louis Baron et Eric Bruillard sur la didactique de l'informatique et le travail d'Anastasia Misirli et Vassilis Komis sur la robotique pédagogique, qui nous ont particulièrement inspirée dans l'élaboration de notre projet de recherche.

À partir de ce moment-là, l'apprentissage de la programmation par les jeunes enfants a commencé à nous préoccuper en tant que sujet de recherche et nous avons commencé à réfléchir à une série des questions : *Comment enseigner la programmation à des élèves si jeunes ? Avec quel environnement de programmation et quel scénario pédagogique ? Quelles commandes enseigner ? Par où commencer ? Comment accompagner des élèves si jeunes ?*

Le début de notre Master a aussi été contemporain d'un regain d'intérêt international envers l'enseignement de l'informatique et surtout un de ses aspects, l'apprentissage de la

programmation auprès de très jeunes enfants. En France, le rapport de l'Académie des Sciences se positionnant en faveur d'un enseignement de l'informatique dès l'école primaire venait d'être publié et le Ministère de l'Éducation Nationale français était prêt à introduire l'informatique et en particulier la programmation dans les nouveaux programmes scolaires dès le cycle 2. De plus, une large gamme d'environnements de programmation visuel ou graphique adaptés à l'âge de très jeunes élèves, tels que ScratchJr, venaient d'apparaître en mettant ainsi la petite enfance au centre d'expérimentations scientifiques.

Dans ce contexte, des projets de recherche ont démarré dans le domaine des sciences de l'éducation pour étudier cette question et essayer d'apporter des réponses utiles aussi bien pour les politiques éducatives que pour le corps enseignant. DALIE[1] ou *Didactique et Apprentissage de l'Informatique à l'École* a été l'un d'eux, financé par l'Agence Nationale de la Recherche (ANR) en France. L'Université Paris Descartes et l'Université de Patras en faisaient partie.

Lors de notre mémoire de Master, qui s'est inscrit dans le cadre de ce projet, nous nous sommes intéressée à l'apprentissage de la programmation par des élèves de cours préparatoire (CP) à travers ScratchJr. Les résultats préliminaires de ce terrain ont montré qu'il était possible d'initier des élèves de CP à la programmation sur ScratchJr et en particulier aux commandes de « messages ». Nous avons alors entrepris de poursuivre ces recherches dans le cadre d'une thèse en cotutelle entre l'Université Paris Descartes, actuellement Université Paris Cité, et l'Université de Patras.

Dans le cadre de ce travail de thèse, nous avons réalisé une recherche exploratoire pour comprendre ce que les élèves de 5 à 7 ans sont capables de faire seuls ou aidés en programmation sur ScratchJr et quelles sont les difficultés qu'ils risquent de rencontrer. Nous avons voulu en particulier étudier les stratégies de programmation, le débogage et les patterns de débogage qu'ils mettent en œuvre.

Pour cela, nous avons d'abord mené une étude préparatoire dans deux classes de deux écoles maternelles en Grèce et une étude principale dans deux classes de CP de deux écoles primaires en France. Nous avons conçu deux scénarios pédagogiques pour enseigner la programmation aux élèves, un pour l'étude préparatoire en Grèce et un pour l'étude principale en France.

1 Dans la Partie 1 de cette thèse, nous présenterons plus en détail les objectifs de ce projet de recherche.

En Grèce, nous avons nous-même enseigné le scénario, alors qu'en France nous avons formé les enseignantes des classes pour qu'elles puissent l'enseigner. Ce travail s'inscrit également dans le cadre d'un autre projet ANR qui a suivi DALIE, IE-CARE[2] ou *Informatique à l'école : conceptualisations, accompagnement, ressources*.

Cette thèse est divisée en quatre parties. La première, composée de deux chapitres, portera sur le contexte dans lequel s'inscrit cette recherche. La deuxième, composée aussi de deux chapitres, est dédiée à la définition de l'activité de programmation et à la compréhension de ce que font les élèves lorsqu'ils programment. Il s'agissait d'identifier les choix théoriques nous permettant de définir notre problématique répondant à un manque d'informations repéré dans la littérature.

La troisième partie constituée de trois chapitres est consacrée à la présentation de notre méthodologie générale, à la stratégie de recherche utilisée, au modèle de conception des scénarios utilisé, ainsi qu'à l'explicitation des deux scénarios pédagogiques conçus et à la méthodologie de collecte et d'analyses des données.

La quatrième partie est dédiée à la présentation des résultats de l'étude préparatoire en Grèce et de l'étude principale en France. Enfin, nous discutons les principaux résultats de cette recherche en montrant les points communs et les différences avec les résultats des études précédemment présentées. Cette discussion générale nous permettra de présenter les limites de cette recherche et d'en proposer des perspectives à la fois pour l'enseignement de la programmation à des très jeunes élèves sur ScratchJr et pour des futures recherches.

2 Dans la Partie 1 de cette thèse, nous présenterons plus en détail les objectifs de ce projet de recherche.

Partie 1 : Éléments de contexte

Chapitre 1. L'informatique à l'école primaire : un renouveau de l'intérêt pour l'apprentissage de la programmation depuis les années 2000

Dans ce chapitre, nous présentons le renouvellement d'intérêt qui s'est manifesté ces dernières années pour l'enseignement de l'informatique et surtout un de ses aspects, l'apprentissage de la programmation auprès de très jeunes enfants. Plusieurs pays dans le monde s'intéressent à introduire la programmation pour les très jeunes enfants dans leurs programmes scolaires. Dans un premier temps, nous examinons les arguments qui justifient ce regain d'intérêt. Ces arguments ont mis une pression sur les décideurs politiques, qui ont essayé de mettre à jour les programmes scolaires de leurs pays en leur ajoutant un enseignement de l'informatique dès l'école élémentaire, afin de les adapter aux nouveaux besoins de la société pour son industrie, mais aussi à ceux des futurs citoyens. Dans un deuxième temps, nous exposons les modalités d'introduction de l'informatique et de la programmation dans les différents systèmes éducatifs dans le monde, et en particulier, en France et en Grèce.

1. Un regain d'intérêt pour une problématique ancienne

Depuis la seconde moitié de la décennie 2000-2010, il y a eu un renouvellement d'intérêt pour l'introduction de l'informatique et surtout pour l'un de ses aspects, la programmation, dans les programmes scolaires. La programmation est une activité spécifique qui est directement liée à la résolution de problèmes. Elle est généralement reconnue comme une compétence nécessaire pour le XXIème siècle. Plusieurs pays dans le monde l'ont introduite dans leurs programmes, en mettant en place diverses approches (Heintz et al., 2016). Des organisations promouvant l'apprentissage de la programmation comme Code.org[3], Code to Learn

3 http://www.code.org

Foundation[4], Girls who code[5], Code Academy[6] ont aussi participé à ce mouvement que la littérature principalement anglaise, appelle « learn to code mouvement » (Kazakoff, 2014; Portelance, 2015; Vivian, Falkner, & Szabo, 2014)

En particulier Kazakoff(2014) explique dans sa thèse que ce mouvement s'est manifesté à l'époque à travers des articles de presse traitant de la nécessité d'enseigner la programmation, des fondations dévouées à la promotion de cet enseignement, des discussions hebdomadaires sur Twitter, mais aussi à travers le programme « Une heure de code »[7], réalisé pendant la Semaine de l'Informatique[8]. Ce programme de 2013 a attiré plus de 23 millions d'enfants selon le site. La semaine européenne du code[9], l'équivalent de la semaine de l'informatique aux Etats-Unis, a attiré lors de sa première édition en novembre 2013 plus de 10.000 personnes de 26 pays en Europe. En 2021, 4 millions de personnes de plus de 80 pays dans le monde ont participé à cet événement.

Pourquoi un tel intérêt pour l'enseignement de l'informatique dès l'école primaire et surtout pour la programmation pour les enfants, une activité longtemps perçue comme difficile et réservée à une élite, celle des informaticiens ?

Plusieurs arguments ont été mis en avant pour justifier cette initiative. Un des principaux a été la nécessité absolue, pour la société de demain, de disposer d'une main-d'œuvre formée à l'informatique. En effet, la crainte derrière cet argument était qu'il y ait plus de postes à pourvoir que des personnes qualifiées susceptibles de les assurer (Pila et al., 2019).

En juillet 2014, le vice-président de la Commission Européenne a envoyé une lettre[10] à tous les ministres de l'éducation de l'Union Européenne afin de les inviter à introduire l'activité de programmation dans les écoles, pour lutter contre le chômage des jeunes et pour permettre à l'Europe de rester compétitive. Dans cette lettre, il mettait l'accent sur le fait que, d'ici la fin de l'année 2020, il y aurait : un manque de 900 mille professionnels en informatique sur le marché du travail en Europe et que cela pourrait nuire à sa compétitivité dans l'avenir.

4 http://codetolearn.org
5 https://girlswhocode.com
6 https://www.codecademy.com
7 https://hourofcode.com/fr/fr
8 https://www.csedweek.org
9 https://codeweek.eu
10 https://digital-strategy.ec.europa.eu/en/news/promoting-coding-skills-europe-part-solution-youth-unemployment

En France, le rapport de l'Académie des sciences (2013), recommande également l'introduction de l'informatique à l'école élémentaire. Le sous-titre de ce rapport *Il est urgent de ne plus attendre,* suggère le retard de l'Europe en général et de la France en particulier, dans le domaine de l'informatique. Il témoigne aussi du fait que la prise de conscience de la nécessité de son enseignement va croissant. Ce rapport propose de mettre en place cet enseignement de l'école primaire jusqu'au lycée.

Un autre argument utilisé a été l'incapacité de la plupart des systèmes éducatifs actuels à préparer suffisamment les jeunes générations pour vivre et travailler dans un monde de plus en plus informatisé comme le nôtre. Le rapport ACM, CSTA (2010) intitulé *Running on Empty : The Failure to Teach K-12 Computer Science in the Digital Age*, souligne le fait que l'éducation obligatoire des Etats-Unis ou *K-12 education* ne réussit pas à bien préparer les élèves pour leur avenir, en leur donnant les connaissances et compétences fondamentales en informatique. Brian Kelly (Kelly, 2012, 27 avril) dans un article de presse intitulé *What STEM*[11] *Is--and Why We Care,* explique qu'il n'y a pas de continuité entre les compétences transmises par l'école et les compétences nécessaires, pour travailler dans un monde de plus en plus informatisé.

Les jeunes générations sont souvent considérées comme des « natifs du numérique » ou « digital natives » en anglais, car elles sont immergées dans ce monde numérique, ce qui les rendrait, selon le discours ambiant, à l'aise avec celui-ci (Resnick et al., 2009). Les travaux de Baron et Bruillard (2008) montrent cependant que les compétences des jeunes en informatique sont très limitées et qu'ils n'arrivent pas à prendre du recul par rapport à ces objets et leur fonctionnement.

Kafai et Burke (2013) ne sont pas de cet avis non plus. Ils expliquent que les jeunes sont capables de manipuler les machines, mais ils sont peu capables d'exercer un pouvoir sur celles-ci de façon critique, créative et sélective. Les résultats d'un projet européen étudiant les représentations des jeunes élèves de fin de l'école élémentaire par rapport à Internet et aux ordinateurs mettent aussi au jour un déficit de compréhension du fonctionnement de ces outils (Dansac et al., 2000). Le discours autour des « natifs du numérique » est problématique également en ce qu'il présuppose que les pratiques des jeunes générations sont homogènes,

11 STEM c'est un acronyme de *science, technology, engineering and mathematics* en anglais. L'équivalent en français c'est STIM de *science, technologie, ingénierie et mathématiques*, mais il s'agit pas d'un acronyme très utilisé.

tandis que la recherche montre qu'elles sont liées au milieu social d'appartenance (Rinaudo, 2017). Ce dernier cite à ce sujet les travaux de Rideout et al., (2010)[12] démontrant que les jeunes ou les enfants qui viennent d'un milieu défavorisé privilégient les usages à caractère divertissant aux usages à caractère éducatif.

Malgré les doutes sur la capacité des systèmes éducatifs à bien former les nouvelles générations, certains auteurs misent sur le fait qu'un tel enseignement pourrait contribuer à un objectif de citoyenneté accrue pour les élèves. L'omniprésence de l'informatique dans notre vie quotidienne donne encore plus de sens à son enseignement. Bruillard souligne à ce sujet :

> « Dans un monde où les machines vont dialoguer beaucoup entre elles (ce que l'on appelle l'Internet des objets) et un peu avec les humains, avoir un minimum de connaissance des données qu'elles échangent et de compréhension de ce qu'elles traitent, comment et pour qui, semble loin d'être un luxe réservé à un petit nombre de personnes » (Bruillard, 2012, p. 3).

Selon Bruillard (2014, p. 2), pour devenir pleinement citoyen à l'ère actuelle, il faut « aller au-delà de la surface des dispositifs pour en comprendre les concepts sous-jacents et les manipuler ». En effet, vivant à une époque où l'évolution des technologies est rapide, les élèves doivent comprendre le monde numérique qui les entoure, maîtriser les bases de ces techniques et avoir accès aux nouvelles formes de pensée qui accompagnent cette évolution (Institut de France. Académie des Sciences, 2013). Ainsi, Bruillard (2012) considère que pour naviguer dans cet environnement numérique, il faut revoir les compétences de base traditionnelles « lire-écrire-compter » en prenant en compte les compétences liées à l'informatique, qui sont d'ailleurs nécessaires pour le XXIème siècle. Il suggère de remplacer « compter » par « computer ». L'Académie des sciences propose pour sa part le slogan qui suit : « Apprendre à lire, écrire, compter, raisonner et programmer » (Institut de France. Académie des sciences, 2013, p. 14).

Pourtant, les jeunes générations n'ont pas pleinement conscience de la nécessité d'apprendre à commander les machines, car ils ont le sentiment que les machines répondent directement à leurs besoins, sans aucun effort de leur part (Bruillard, 2012). Pour Bruillard, le problème réside dans le fait que les gens pensent qu'ils programment la machine, alors qu'ils sont programmés par elle. L'enjeu essentiel d'après Luehrmann (1972) est de former suffisamment les jeunes en informatique, pour leur donner le pouvoir sur les machines. Douglas Rushkoff

12 Rideout, V. J., Foehr, U. G., & Roberts, D. F. (2010). Generation M2 : Media in the Lives of 8- to 18-Year-Olds. Henry J. Kaiser Family Foundation. https://eric.ed.gov/?id=ED527859

(2010) souligne, dans son ouvrage *Program or to be programmed,* que pour émanciper les humains à l'ère du numérique, il est nécessaire de leur apprendre à programmer, car la programmation est derrière tout objet informatisé. Dans le cas contraire, ils seront dépendants de ceux qui disposent de cette compétence ou de ceux qui les payent. Seymour Papert (1980) était un des premiers à exprimer cette idée.

Ainsi, le développement d'une culture informatique paraît indispensable (Bruillard, 2012). Drot-Delange et Bruillard (2012, p.70) notent qu'historiquement, « la promotion de la culture informatique est née des mouvements sociaux visant à émanciper l'individu et à lui donner la maîtrise de son environnement ». Selon ces auteurs, la culture informatique est constituée des trois éléments complémentaires : « une « pensée » informatique, la maîtrise d'objets informatiques et la participation à des activités sociales dans un monde en réseau » (Drot-Delange & Bruillard, 2012, p. 72).

À l'argument d'une citoyenneté accrue s'ajoute un autre argument fréquemment utilisé en faveur d'une initiation des jeunes élèves à l'informatique et en programmation : celui qui fait état des bénéfices potentiels liés au développement d'une pensée informatique chez les élèves. Les débats autour de cette notion s'appuient sur des arguments de transférabilité qui accompagnaient l'enseignement de l'informatique depuis les années 1980 (Drot-Delange et al., 2019). En 2006, l'article de Jeanette Wing intitulé *Computational Thinking* ou *Pensée Informatique* en français, a bouleversé la communauté scientifique et éducative, car elle a mis à l'ordre du jour une notion longtemps oubliée, en lui donnant une nouvelle définition. Wing est désormais devenue l'ambassadrice contemporaine de la pensée informatique et son article a suscité nombreuses interprétations.

Wing (2006) a défini la pensée informatique comme une compétence fondamentale pour tout le monde et pas seulement pour les informaticiens. D'après Wing, la pensée informatique, en s'appuyant sur des concepts de la science informatique, implique la résolution de problèmes, la conception des systèmes et la compréhension du comportement humain. Wing propose de mettre à jour la trilogie « lire-écrire-compter » en lui ajoutant la pensée informatique. Cette auteure considère également que la pensée informatique influe aussi sur d'autres disciplines. Dans un article de 2008, *Computational thinking and thinking about computing*, l'auteure souligne que ce type de pensée analytique a des liens forts avec la pensée mathématique, la pensée de l'ingénierie et la pensée scientifique (Wing, 2008).

Nous allons voir dans la Partie 2 de cette thèse, que Wing n'était ni la première, ni la dernière à écrire autour de la pensée informatique. L'enjeu derrière cette pensée si célèbre était de renforcer les arguments en faveur de la nécessité d'un enseignement de l'informatique dans l'enseignement obligatoire ; et d'une certaine façon, cet article est arrivé au bon moment pour accompagner le mouvement international prônant l'apprentissage de la programmation. Dans un texte qui rend plus claire cette notion, Drot-Delange et ses co-auteurs constatent que :

> « la « pensée informatique », telle qu'on la décrit aujourd'hui, n'est pas un concept informatique. C'est un concept lié à l'enseignement de l'informatique (et de thématiques connexes), à sa potentielle transversalité, aux compétences générales développées, et aux potentiels transferts que sa maîtrise ou sa connaissance impliquerait » (Drot-Delange et al., 2019, p. 7).

Au-delà des visées d'employabilité, de citoyenneté et de pensée informatique, il y en avait d'autres, toujours en lien avec cette logique de transferabilité accompagnant, les nouvelles disciplines, telle que l'informatique. En effet, très souvent quand une discipline veut se faire reconnaître et acquérir ou préserver sa place dans les programmes, aux côtés de disciplines déjà installées comme le français ou les mathématiques etc, elle cherche à établir que son impact va au-delà d'elle-même, car elle est utile ailleurs. Les défenseurs de l'enseignement de l'informatique à travers la programmation affirment que la pratique de cette activité pourrait avoir des effets positifs vers d'autres domaines. Pour cela, ils s'appuient principalement sur les travaux des années 1980, portant sur des expériences d'enseignement de l'informatique utilisant le langage de programmation LOGO. Dans ces travaux, l'apprentissage de la programmation n'était le plus souvent pas l'objectif principal, mais un outil aidant au développement d'autres compétences.

Dans ce qui suit, nous citons quelques exemples de la littérature illustrant cela. Clements et Gullo (1984) ont ainsi par exemple montré que les élèves de cours préparatoire qui ont appris à utiliser le langage de programmation LOGO à l'aide d'un scénario pédagogique de 12 semaines, ont amélioré leur capacité à décrire des directions ; cela n'a pas été le cas du groupe contrôle qui a participé à cette recherche. Degelman et al. (1986) observent que des élèves de maternelle qui ont utilisé le langage LOGO pendant 5 semaines ont eu une meilleure performance face à la résolution de deux problèmes que le groupe contrôle.

Une autre étude sur quarante-huit élèves de CE2, distribués à un groupe LOGO et un groupe contrôle, montre que ceux qui ont suivi le cours sur LOGO ont réalisé une performance de conceptualisation des angles et des formes meilleure que ceux qui faisaient partie du groupe

de contrôle (Clements & Battista, 1989). Clements et Meredith (1993) remarquent que l'apprentissage de la programmation dans un environnement LOGO renforce également le développement des compétences en mathématiques, en résolution de problèmes et en langage.

Les travaux de recherche menés au départ sur des expérimentations utilisant des langages de programmation textuels comme LOGO ont montré que leur syntaxe posait souvent problème aux élèves. Ce qui a changé ces dernières années, et qui constitue d'ailleurs un des facteurs décisifs du regain d'intérêt pour l'enseignement de la programmation aux enfants, est l'apparition d'une nouvelle forme de programmation à destination des élèves : la programmation visuelle ou graphique. Ainsi, la nature des langages a changé, et est désormais mieux adaptée aux besoins des élèves. Ce type de langage donne la possibilité aux plus jeunes élèves de programmer en utilisant des blocs représentant des commandes de programmation, sans avoir besoin de lire ni écrire, compétences en cours d'élaboration à cet âge-là, ni même de connaître la syntaxe (Duncan et al., 2014).

En effet, les langages de programmation graphiques n'ont pas de règles de syntaxe strictes contrairement aux langages traditionnels. Les commandes sont des blocs graphiques dont la forme est telle qu'elle oblige l'utilisateur à les assembler correctement pour créer des programmes. Toutefois, malgré leur syntaxe simplifiée, les langages graphiques contiennent encore beaucoup de texte, ce qui compromet leur utilisation dans le milieu de la petite enfance. Les langages graphiques ont donc dû évoluer davantage afin de devenir accessibles même aux plus jeunes élèves.

La création des langages de programmation visuels ou graphiques destinés aux jeunes enfants, comme ScratchJr[13], Hopscotch[14], ou Alice[15] a ouvert une période florissante en termes d'expérimentations scientifiques dans le milieu de la petite enfance, groupe d'âge qui n'était jusqu'alors souvent concerné que par des robots programmables.

Au-delà de ces arguments visant à établir la nécessité d'enseigner l'informatique via la programmation aux enfants, certains travaux insistent sur la nécessité d'introduire cet enseignement le plus tôt possible. Selon Kazakoff (2014), il est bénéfique d'enseigner la programmation aux enfants dès la maternelle, car de cette manière, on crée très tôt une gamme d'expériences préalables en informatique chez les élèves.

13 https://www.scratchjr.org/
14 https://www.gethopscotch.com/
15 https://www.alice.org/

Pour l'activité de programmation, l'introduction précoce de telles activités permet de donner très tôt aux filles des expériences positives dans ce domaine. En effet, l'Académie des sciences souligne en 2013 qu' « il est essentiel que tous les citoyennes et citoyens soient égaux dans leur compréhension de l'informatique et du monde numérique » (p.15). Or, Margolis et Fisher (2002) ont observé une chute importante de confiance en soi des filles en informatique à partir d'environ 12 ans. Ils ont aussi remarqué que l'existence d'expériences antérieures à l'école élémentaire les aidait à maintenir leur confiance dans cette matière. En effet, sur le site internet de l'organisation Girls Who Code[16], nous pouvons observer un graphique qui présente la baisse du pourcentage des femmes impliquées en informatique, de 37 % à 1995 au 24 % aujourd'hui.

L'organisation souligne que la plus grande chute du nombre de filles pratiquant l'informatique se situe entre 13 et 17 ans ; elle œuvre à réduire les inégalités femmes-hommes en informatique avant 2030. Le Douarin (2004) montre que les femmes accordent par défaut une expertise face à l'ordinateur et la technique en général aux hommes, en accord avec les stéréotypes de genre encore très vivaces dans notre société.

Tous ces travaux mettant en évidence la nécessité d'un enseignement de l'informatique, et spécifiquement de la programmation, aux jeunes enfants, ont trouvé des échos chez les décideurs politiques, qui ont entrepris de mettre à jour les programmes scolaires de leurs pays en y introduisant un enseignement de l'informatique dès l'école élémentaire, afin de les adapter aux nouveaux besoins de la société, dans le double objectif de former une main-d'œuvre répondant aux exigences de l'industrie, et de former des individus à même d'exercer pleinement leur statut de citoyens. L'objectif affiché de ces initiatives a été de donner aux gens un pouvoir d'agir, afin de les rendre concepteurs et pas seulement consommateurs et simple utilisateurs de ces technologies (Vivian et al., 2014).

Nous allons à présent observer les modalités d'introduction de l'informatique et de la programmation dans les différents systèmes éducatifs dans le monde, et plus particulièrement en Grèce et en France.

16 https://girlswhocode.com

2. Des curricula repensés au XXIe siècle

La dernière décennie a été marquée par une série d'initiatives autour de l'introduction de l'informatique et surtout de l'activité de programmation dans les différents systèmes éducatifs dans le monde. La reconnaissance de l'importance de l'informatique et de la programmation pour les futurs citoyens a convaincu les pouvoirs politiques des Etats-Unis, de l'Australie et des plusieurs pays de l'Europe de les intégrer dans les programmes scolaires.

Nous allons voir dans ce qui suit qu'il existe pléthore de termes utilisés dans les programmes scolaires des différents pays pour parler de l'introduction de cet enseignement. Nous trouvons par exemple le terme « Computer science » aux Etats-Unis, « Digital technologies » en Australie, « Computing » au Royaume-Uni et « Informatics » ou « Informatique » en français, en Europe. Nous repérons aussi très souvent dans les textes officiels le terme « numérique ». Cette pluralité de termes pour décrire la discipline informatique pourrait poser des problèmes à la discipline, aux chercheurs, aux décideurs politiques et au corps enseignant, car il est difficile de communiquer et de collaborer s'il n'est pas clair pour chacun que ces différents termes renvoient tous à une même réalité.

2.1. Des tendances différentes à travers le monde

Les Etats-Unis ont été parmi les premiers à reconnaître la nécessité d'un enseignement de l'informatique et de la programmation à l'école. Au-delà de Code.org, les travaux de ACM, « Association of Computing Machinery » et de CSTA, « Computer Science Teachers Association », associations de professionnels rassemblant des chercheurs et des enseignants en informatique, ont joué un rôle important dans la reconnaissance de ce besoin. En 2003, CSTA a publié un curriculum modèle pour la scolarité obligatoire, dit K-12, qui intégrait des compétences en informatique pour les Etats-Unis, mais aussi pour le reste du monde. À travers ce rapport, l'association a entrepris de montrer que le curriculum actuel ne faisait pas assez de place à l'informatique et à ses concepts et que, par conséquent, il ne formait pas suffisamment les gens à cette discipline. L'objectif de ce curriculum était d'aider à résoudre ces problèmes (ACM CSTA, 2003).

Sept ans plus tard, CSTA a publié un rapport intitulé *Running on empty : The failure to teach K-12 computer science in the digital age.* Ce texte présente les résultats d'une étude effectuée

sur deux ans, dans le cadre de laquelle les auteurs ont comparé le type de prise en compte de l'informatique dans l'éducation obligatoire des 50 états d'Amérique du Nord et ont identifié ceux qui considéraient l'informatique en tant qu'enseignement fondamental (core subject en anglais) à la fin du lycée. Le résultat, considéré comme inquiétant, de ce rapport a été le suivant : les deux tiers des 50 états n'avaient pas de place pour l'informatique dans leur éducation obligatoire et quand il y en avait, souvent il était réduit à l'utilisation d'applications (ACM CSTA, 2010).

En 2011, CSTA a proposé une série des standards d'apprentissages en informatique pour l'éducation obligatoire, K-12, afin de clarifier les missions de l'enseignement de l'informatique et répondre au besoin de continuité entre les compétences transmises par l'école et les compétences nécessaires pour travailler dans la société actuelle (ACM CSTA, 2011). En 2017, l'association a présenté une nouvelle édition de ces standards, qui sont toujours en vigueur (ACM CSTA, 2017).

Parallèlement, aux Etats-Unis, d'autres initiatives ont marqué ce mouvement pour l'innovation dans l'éducation obligatoire. Le ministère fédéral américain de l'éducation et le President Barack Obama[17] ont mis l'accent sur l'importance de l'intégration de l'informatique dans l'éducation et de l'apprentissage de la programmation par les élèves (U.S. Departement of Education, Office of Educational Technology, 2010). En 2016, 4 milliards de dollars ont été investis pour les États et 100 millions pour les « school districts », qui correspondent aux secteurs scolaires en France, afin de financer l'initiative Computer Science for All[18], dont l'objectif est d'émanciper tous les Américains en leur donnant accès à une formation à l'informatique dès la maternelle et jusqu'au lycée.

Dans ce contexte, les organisations ACM, CSTA, Code.org, Cyber Innovation Center et National Math and Science Initiative ont collaboré pour développer et diffuser en 2016 un cadre[19] conceptuel pour l'informatique. Ce cadre pose une série de concepts et de pratiques nécessaires pour répondre aux besoins du XXIème siècle (K-12 Computer Science Framework, 2016). Un chapitre entier de ce texte est dédié à l'apprentissage de l'informatique dès la petite enfance et souligne que ce dernier pourrait être réalisé en respectant et en mettant

17 https://obamawhitehouse.archives.gov/blog/2013/12/09/don-t-just-play-your-phone-program-it
18 https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all
19 https://k12cs.org/

en valeur les apprentissages sociaux-émotionnels et le jeu, qui caractérisent le contexte de la maternelle.

L'Australie fait aussi partie des pays qui ont reconnu l'importance d'une éducation à l'informatique et à la littératie numérique dès la petite enfance (Falkner et al., 2014). Un nouveau programme scolaire national a été élaboré en 2014 par l'Australian Curriculum Assessment and Reporting Authority (ACARA)[20]. Les processus de renouvellement du curriculum national ont débuté en 2009. ACARA suit 4 phases assez rigoureuses pour l'élaboration de chaque nouveau curriculum : « curriculum shaping », « curriculum writing », « preparation for implementation » et « curriculum monitoring, évaluation and review » (Australian Curriculum Assessment and Reporting Authority [ACARA], 2012a april). Le nouveau programme national introduit, d'une part, une compétence liée aux Technologies de l'Information et de la Communication (TIC) dans les compétences de base et d'autre part, une nouvelle matière, « Technologies, qui englobe deux sous-unités : « Design and Technologies » et « Digital Technologies (DT) » (Australian Curriculum Assessment and Reporting Authority [ACARA], 2012b october). Le développement d'une pensée informatique est un des principaux objectifs de l'unité DT. Outre la pensée informatique, cette unité concerne aussi l'utilisation des systèmes numériques et la manipulation de l'information (Falkner et al., 2014).

Pour le groupe d'âges qui nous concerne, c'est-à-dire de 5 à 7 ans (« Foundation to Year 2 » d'après le système éducatif de l'Australie), le programme scolaire prescrit l'utilisation de robots programmables et l'apprentissage par le jeu afin d'aider les élèves à créer une série de solutions numériques (Australian Curriculum, Assessment and Reporting Authority [ACARA], 2015). Dans ce contexte, les élèves seront aussi familiarisés avec le concept de l'algorithme, nécessaire pour résoudre des problèmes simples. Le traitement et l'organisation des données font aussi partie des objectifs visés pour ce groupe d'âge. En discutant avec les enseignants, les élèves auront également la possibilité d'apprendre à se protéger et protéger les autres quand ils interagissent sur internet. La reconnaissance d'une séquence d'instructions ou d'événements fait aussi partie des processus et des compétences à développer, d'après les prescriptions du programme (Australian Curriculum, Assessment and Reporting Authority [ACARA], 2015).

20 https://www.acara.edu.au/curriculum

Selon Vivian, Falkner et Falkner (2014), la majorité des enseignants du premier degré en Australie enseignent toutes les différentes matières du programme. Les auteurs soulignent que certains établissements ont la chance d'avoir dans leur équipe des enseignants formés aux TIC car leur effectif est assez restreint. Pour aider les enseignants à se former à l'informatique, un MOOC leur a été proposé afin de leur transmettre des ressources et des solutions pédagogiques pour la mise en place de cette nouvelle matière (Vivian, Falkner, et Falkner, 2014). En même temps, une revue des ressources disponibles sur internet, en lien avec l'enseignement et l'apprentissage de l'informatique, a été réalisée afin d'évaluer leur qualité et d'identifier d'éventuels besoins en prenant en compte le nouveau curriculum de l'Australie (Falkner & Vivian, 2015).

Plus près de l'Europe, le Royaume-Uni a fait le même constat que les États-Unis : l'informatique n'était pas suffisamment présente dans ses écoles non plus. En 2008, Computing at schools[21] (CAS), une communauté des bénévoles passionnés de l'informatique, a été fondée afin de lutter pour l'introduction de l'informatique à l'école. En quatre ans, la petite communauté a attiré 2000 membres (Brown, Kölling, et al., 2013).

En 2012, la publication du rapport de British Royal Society, *Shutdown or restart,* a mis à l'ordre du jour l'introduction de l'informatique à l'école obligatoire. Selon ce rapport, c'était le bon moment, en cette période de réforme des programmes scolaires nationaux, pour mettre en place des initiatives permettant de renouveler la matière Information and communication technologies (ICT) ou Technologies de l'information et de la communication (TIC) en français. Il propose de remplacer les activités actuelles autour des TIC par des activités rigoureuses et créatives en informatique (The Royal Society, 2012). La parution de ce rapport a donné encore plus de sens à l'agenda de CAS, qui a vu ses activités s'accroitre à une échelle plus large. Son rôle a alors doublé, puisque CAS a dû soutenir à la fois les enseignants qui mettaient en place ce nouvel enseignement, en les aidant avec le contenu à enseigner, mais aussi les pouvoirs politiques, afin d'améliorer le contexte dans lequel cet enseignement prendre place (Brown, Kölling, et al., 2013).

Le Royaume-Uni a annoncé en 2013 l'introduction de l'informatique (computing education) à la place des TIC, dans le nouveau programme scolaire national (Department for Education, 2013). Cette nouvelle matière comprend trois composantes : litteratie numérique (digital

21 https://www.computingatschool.org.uk/

literacy), technologie informationnelle (information technology) et la science informatique (computer science). La pensée informatique fait aussi partie de ce nouveau programme, car elle est présentée comme l'objectif principal d'une éducation de qualité en informatique (Department for Education, 2013).

Ce programme national est organisé en cinq niveaux différents (Key Stages), en fonction de l'âge des élèves (Brown, Sentance, et al., 2013). Le groupe d'âge qui nous concerne, de 5 à 7 ans, correspond au niveau 1 (Key stage 1-Foundation year and years 1 to 2). En examinant le contenu du programme pour le niveau 1, nous pouvons remarquer une liste d'objectifs qui sont définis pour les élèves. Ces derniers doivent se familiariser avec la notion de l'algorithme, comprendre comment un algorithme fonctionne et de quelle manière il est implémenté et exécuté en tant que programme derrière les différents systèmes techniques de leur environnement. Les élèves doivent aussi créer des petits programmes, les déboguer et mettre en œuvre un raisonnement logique afin de prévoir leur comportement. En outre, les élèves doivent pouvoir utiliser la technologie pour gérer le contenu numérique et être capables d'identifier les utilisations de la technologie informationnelle au-delà de la sphère de l'école. Le dernier objectif présenté concerne l'utilisation sécurisée de la technologie par les élèves (Department for Education, 2013).

Dans ce contexte, Informatics Europe et ACM Europe Working Group on Informatics Education ont publié en 2013 un rapport intitulé *Informatics Education:Europe cannot afford to miss the boat.* La première phrase de ce rapport : « All of Europe's citizens need to be educated in both digital literacy and informatics »[22], illustre bien son objectif. Le rapport souligne que la litteratie numérique et l'informatique sont des composantes essentielles pour l'éducation à l'heure actuelle. Les groupes d'experts, après étude de l'état actuel de l'éducation en Europe et des futurs besoins de la société, ont fait une série des recommandations relatives à l'intégration de ces deux composantes aux curricula existants. La pensée informatique de Jeannette Wing a eu aussi une place dans ce texte (Informatics Europe & ACM Europe Working Group, 2013).

Après le Royaume-Uni, plusieurs pays de l'Europe ont ainsi commencé à introduire l'informatique, la programmation et la pensée informatique dans leurs programmes. European

22 Notre traduction en français : Tous les citoyens de l'Europe doivent être formés à une culture numérique et à l'informatique.

Schoolnet[23], a commencé depuis 2014 à suivre les évolutions de ce mouvement en réalisant des études de la situation dans les pays européens et en publiant plusieurs rapports.

En 2015, European Schoolnet a publié une version actualisée du rapport *Computing our future,* dans laquelle les auteurs ont présenté les résultats d'une étude effectuée auprès de 20 pays européens et Israel, sur l'intégration de la programmation dans les programmes scolaires (European Schoolnet, 2015). Ce rapport relève que 15 pays européens (Autriche, Bulgarie, République Tchèque, Danemark, Estonie, France, Hongrie, Irlande, Lituanie, Malte, Espagne, Pologne, Portugal, Slovakie, Royaume-Uni) et Israel ont déjà introduit en 2015 de la programmation dans leurs programmes scolaires à des degrés et selon des modalités divers. Le développement d'une pensée informatique est présent dans les programmes de 4 pays (République Tchèque, Irlande, Malte et Pologne).

Ce rapport présente aussi les modalités d'introduction de la programmation dans les différents systèmes éducatifs en Europe (European Schoolnet, 2015). Certains pays préfèrent la prendre en compte dans le secondaire, mais de plus en plus nombreux sont ceux qui la prennent en compte dès l'école élémentaire. Plusieurs pays l'ont rendue obligatoire pour certains niveaux d'éducation, d'autres pour l'école primaire ou même pour toute la scolarité obligatoire. Certains qui l'ont introduit dans les programmes en tant que matière « informatique », d'autres l'ont introduit en lien avec d'autres matières (European Schoolnet, 2015).

D'après Heintz et al. (2016), il existe plusieurs raisons derrière le choix de certains pays d'opter pour une intégration de l'informatique à d'autres matières, même si une reconnaissance en tant que matière à part entière aurait pu être plus bénéfique pour cet enseignement. Ces auteurs citent le manque de temps disponible dans le système scolaire actuel, la valeur de la pensée informatique pour tous les citoyens et son apport à la résolution des problèmes complexes à l'aide d'un ordinateur, mais aussi le fait qu'une telle intégration permet d'amener à l'informatique des personnes qui n'auraient pas été intéressées a priori par l'activité de programmation. La nécessité d'une formation longue pour les enseignants qui devront prendre en charge cet enseignement constitue aussi un énorme frein à la création de cette nouvelle matière.

Heintz et al. (2016) présentent les résultats d'une étude portant sur 10 pays différents et leur façon d'introduire l'informatique dans les programmes scolaires. Selon les résultats de cette

23 http://www.eun.org/

étude, l'introduction de l'informatique à l'école primaire se fait souvent sous la forme de pensée informatique, de programmation ou des compétences numériques, alors que la tendance pour le secondaire est d'introduire des cours sur l'informatique et son impact sur la société (Heintz et al., 2016).

Nous présentons ici à titre indicatif le format mis en place pour l'introduction de l'informatique dans les programmes par trois pays européens différents, la Finlande, la Slovaquie et la Suède.

La Finlande fait partie des pays qui n'avaient jusqu'alors pas de matière « informatique » dans leurs programmes scolaires. Selon Kurhila et Vihavainen (2015), le programme scolaire national de 2004 ne faisait pas de place à l'informatique mais seulement aux TIC en lien aux autres matières scolaires. Pour régler ce problème, le Departement of Computer Science de l'Université d'Helsinki a construit et proposé un MOOC (Massive Open Online courses) à tous les élèves de toutes les écoles de Finlande. Ces dernières ont très bien reçu cette initiative, mais ont choisi d'offrir ce cours à leurs élèves comme option. En 2016, selon Heintz et al. (2016), un nouveau programme scolaire était prévu en Finlande pour toute la scolarité obligatoire. En ce qui concerne l'informatique, le nouveau programme fera une place à la programmation dans le premier degré, mais pas dans le second degré. Une série d'initiatives ont été mises en place de la part de l'État, mais aussi de la part des universités et des organisations pour soutenir les enseignants face à ce nouvel enseignement (Heintz et al., 2016).

La situation en Slovaquie est différente de celle de la Finlande. En effet, l'informatique est enseignée en Slovaquie depuis des années en tant que matière spécifique séparée des autres. Un système de formation d'enseignants est aussi mis en place (Kabátová et al., 2016) Ce pays pourrait servir de modèle à ceux qui commencent maintenant à aborder la question de l'introduction de l'informatique dans leurs programmes scolaires. Les programmes scolaires de Slovaquie sont assez complets, car ils comportent des concepts en informatique, mais aussi des compétences liées à l'utilisation des TIC. Il s'agit d'un des rares pays à proposer un curriculum national pour l'informatique (Kabátová et al., 2016).

L'enseignement de l'informatique a une longue histoire en Slovaquie qui remonte aux années 1970, où certaines écoles professionnelles ont commencé à préparer les élèves à utiliser les ordinateurs pour la fabrication mécanique. En même temps, des nouveaux programmes de

formation en informatique se créent au sein des universités (Kabátová et al., 2016). Dans les années 1980, la plupart des établissements secondaires offraient déjà à leurs élèves des classes spécifiques destinées à l'apprentissage de l'informatique. Au début des années 1990, la majorité de ces établissements enseignaient l'informatique. Vers la fin des années 1990, le nouveau curriculum national a pris en compte cinq thématiques : Information autour de nous, Communication à travers les TIC, Resolution des problèmes et pensée algorithmique, Principes des TIC et Société Informationnelle (Kabátová et al., 2016).

En 2008, le nouveau programme scolaire a rendu l'enseignement de l'informatique obligatoire de l'âge de 7-8 ans à l'âge de 16-17 ans (du CE1 à la première d'après le système éducatif français). Les cinq thématiques mentionnées auparavant sont toujours d'actualité dans toute la scolarité obligatoire, mais elles sont enseignées différemment en fonction du niveau scolaire. La Slovaquie a étendu son intérêt pour l'enseignement de l'informatique à la maternelle (Kabátová et al., 2016). Une série d'utilisations pédagogiques du robot programmable Bee-Bot ont été mis en place par des chercheurs en Slovaquie afin de collecter des données pour aider les enseignants de la maternelle à intégrer l'informatique dans leurs classes (Pekárová, 2008).

En 2017, le projet *Computing with Emil*, dont l'objectif a été de proposer de nouveaux environnements de programmation et des ressources pour les élèves et les enseignants de l'école élémentaire, a été lancé (Blaho & Kalaš, 2020). Emil[24] a été créé. Il s'agit d'une méthode d'enseignement et d'apprentissage de l'informatique à l'école primaire. L'équipe des chercheurs derrière la création d'Emil est constituée de Ivan Kalas, Blaho Andrej et Milan Moravcik. Emil est un logiciel disponible sur tablettes, ordinateur portable ou fixe, pour Android, Windows et MacOs.

Emil est actuellement disponible pour les élèves de 8 à 9 ans (CE2) et ceux de 9 à 10 ans (CM1). Des ressources pédagogiques pour les élèves de 8 à 9 ans ont été mises à la disposition des enseignants intéressés sur le site d'Emil en 2019. D'autres ressources pour les élèves de 9 à 10 ans ont complété l'offre du site en 2021. Depuis sa création, le logiciel et la méthode d'enseignement ont attiré l'attention de plusieurs chercheurs et enseignants de différents pays, comme la France, la République Tchèque, le Royaume-Uni, la Pologne,

24 https://www.robotemil.com/

l'Hongrie et la Norvège. L'équipe d'Emil a établi aussi une collaboration avec la Haute école pédagogique de Vaud à Lausanne, Suisse.[25]

En Suède la situation est différente. Heintz et al. (2015) rappellent que le système éducatif y est composé de l'école obligatoire (compulsory school) de 7 à 16 ans et de l'école secondaire supérieure (upper secondary school) de 16 à 19 ans. L'informatique et la programmation ne font pas partie du programme scolaire pour l'école obligatoire. Un seul programme de l'enseignement général vise la technologie en proposant des cours qui traitent différents aspects de l'informatique. Par conséquent, peu d'élèves suivent ce cours à l'école secondaire supérieure. L'informatique reste donc une matière réservée à un petit nombre.

Heintz et al. (2015) soulignent qu'en 2012 le gouvernement de Suède a créé un comité scientifique dont son objectif a été d'étudier comment le pays pourrait tirer avantage du numérique et proposer une série de recommandations et des directives pour cela. Deux ans plus tard, le comité a publié un rapport qui met l'accent sur la nécessité d'investir dans le développement des compétences numériques, en les introduisant en tant qu'objectifs dans les programmes scolaires. Entre autres préconisations, ils ont proposé d'introduire la programmation dans les disciplines existantes (cross-curricular approach) (Heintz et al., 2015).

En 2015, les Suédois ont annoncé l'élaboration d'un nouveau curriculum national pour l'école obligatoire et une révision du curriculum de l'école secondaire, afin de prendre en compte l'apprentissage de la programmation et le développement des compétences numériques chez les élèves (Heintz et al., 2016). L'Agence Nationale pour l'Education (National Agency for Education) a été chargée de mettre cette initiative en place. Sous réserve de l'acceptation par le gouvernement de la proposition de l'Agence Nationale pour l'Education, ces nouveaux programmes devaient entrer en vigueur à l'automne 2017 (Heintz et al., 2016). La nouveauté de cette proposition est l'introduction d'une section sur les compétences numériques et la compréhension de la technologie numérique par tous les élèves. Elle suggère aussi de changer l'organisation des cours de Maths, de technologie et de sciences sociales. La programmation est introduite dans le curriculum de mathématiques. L'Agence Nationale pour l'Éducation a été à cette époque chargée de la formation des enseignants. En 2016, la formation initiale des enseignants ne prévoyait pas des cours en informatique (Heintz et al., 2016).

25 Informations disponibles sur le site : https://www.robotemil.com/

Le Centre Commun de recherche (Joint Research Center[26], [JRC]), le service scientifique interne de la Commission Européenne, responsable de la réalisation des recherches pour contribuer à l'élaboration des politiques de l'Union Européenne, a organisé et financé une recherche sur la pensée informatique et son rôle potentiel dans l'éducation obligatoire. La recherche intitulée CompuThink[27], a été menée par l'Institut de Technologie Éducative du Conseil National de la Recherche italienne[28] et European Schoolnet[29]. Suite à cette recherche un rapport a été publié en 2016 (European Commission. Joint Research Centre, 2016). Celui-ci a donné un aperçu des initiatives mises en place concernant le développement de la pensée informatique dans l'éducation obligatoire en Europe, mais aussi des conséquences de ces initiatives sur les politiques éducatives et la pratique des enseignants. Ce rapport a également présenté les concepts et les caractéristiques principaux de la pensée informatique, ainsi que les compétences potentielles à developper chez les élèves (European Commission. Joint Research Centre, 2016).

En janvier 2018, la Commission Européenne a annoncé le nouveau Plan d'Action pour l'Education Numérique (European Commission [COM], 2018) (European Commission [COM], 2018). Son objectif a été de montrer comment les systèmes d'éducation et de formation pourraient mieux utiliser les innovations et les technologies numériques, afin d'émanciper les humains en les aidant à développer les compétences numériques nécessaires pour tout citoyen à l'ère actuelle (COM, 2018).

Ce plan insiste sur trois priorités : une meilleure utilisation de la technologie numérique pour l'enseignement et l'apprentissage, le développement des compétences numériques pour la transformation numérique et « une amélioration de l'éducation à travers une meilleure prévoyance et analyse des données » (COM, 2018). Parallèlement, European Schoolnet et National Research Council of Italy, Institute for Educational Technology ont publié un autre rapport portant sur la pensée informatique, mettant cette fois-ci l'accent sur les pays nordiques. Il s'agit d'une mise à jour du rapport de 2016 (European Commission. Joint Research Centre, 2016). Ce rapport décrit l'état actuel de l'introduction de la pensée informatique et de la programmation dans les programmes scolaires de 4 pays nordiques : Danemark, Finlande, Norvège et Suède (European Schoolnet & National Research Council of

26 https://ec.europa.eu/info/departments/joint-research-centre_en
27 https://joint-research-centre.ec.europa.eu/computational-thinking-study_en
28 https://www.itd.cnr.it/en/
29 http://www.eun.org/

Italy, Institute for Educational Technology, 2018). Selon les résultats de ce rapport, ces pays optent pour une approche qui combine la pensée informatique et la programmation. Ces dernières sont vues comme parties intégrantes d'une définition de la compétence numérique (European Schoolnet & National Research Council of Italy, Institute for Educational Technology, 2018). La pensée informatique n'est pas présente en tant que telle dans les programmes scolaires des pays nordiques, mais les idées principales qui la sous-tendent y figurent.

Un problème, qui est d'ailleurs souvent détecté dans les textes officiels concernant l'informatique, est la pléthore des définitions et des termes utilisés autour de la pensée informatique par les pays nordiques (European Schoolnet & National Research Council of Italy, Institute for Educational Technology, 2018). Les auteurs du rapport soulignent la nécessité d'avoir une base commune de définitions pour éviter les malentendus et faciliter la collaboration entre les différents acteurs de l'éducation (enseignants, chercheurs, décideurs politiques).

En juillet 2019, la présidente de la Commission Européenne Ursula von der Leyen a posé la nécessité de mettre en place un nouveau Plan d'Action pour l'Education Numérique, fondé sur le premier (2018-2020). Un nouveau Plan d'Action pour l'Éducation Numérique a été lancé avec une plus longue durée cette fois-ci, de 2021 à 2027[30]. L'objectif de cela était d'assurer le passage de l'éducation et des systèmes de formation de l'Europe à « l'heure du numérique ». Ce plan a deux priorités majeures : renforcer le développement d'une éducation numérique haute performance et améliorer les « compétences numériques » pour la « transformation numérique ».

Cette action est plus que jamais d'actualité aujourd'hui, car la pandémie COVID-19 a entraîné un développement de l'usage du numérique dans l'éducation et en formation. Au-delà de la mise en évidence d'un réel besoin, pour l'éducation, d'une amélioration des compétences numériques, la pandémie a aussi rendu plus claires les inégalités entre ceux qui ont accès aux technologies numériques et ceux qui n'y ont pas accès. Dans ce contexte, le Centre Commun de recherche (JRC) a publié un nouveau rapport sur la pensée informatique, sur la base de l'étude CompuThink II, afin de compléter les travaux réalisés jusqu'ici sur la question (European Commission. Joint Research Centre., 2022). Ce rapport s'appuie sur les données

30 https://education.ec.europa.eu/focus-topics/digital/education-action-plan

des 22 pays européens et 8 non européens ; il récapitule tous les avancements au niveau de la pensée informatique et de son intégration dans les programmes scolaires de l'Europe, de 2016 à 2021. Nous observons donc que l'éducation à l'informatique et la pensée informatique sont toujours un des axes prioritaires de l'agenda de l'Europe, puisque cette dernière est pour « la transformation numérique ».

Après avoir présenté les modalités d'introduction de l'informatique et un de ses aspects la programmation dans différents systèmes éducatifs dans le monde, il convient maintenant de nous focaliser sur les cas de la France et de la Grèce, pour voir comment ces pays ont mis en place cet enseignement.

2.2. Le cas de la France

L'enseignement de l'informatique à l'école primaire en France a une longue histoire, caractérisée par différents mouvements avec des objectifs variés, souvent liés aux priorités des décideurs politiques de chaque période (Baron & Drot-Delange, 2016). Nous présentons ici un bref point historique regroupant une série d'événements qui ont marqué cette histoire, pour arriver à la situation d'aujourd'hui.

Des années 1970 aux années 1980, nous observons un désintérêt pour l'enseignement de l'informatique à l'école primaire en France (Baron & Drot-Delange, 2016), alors que dès 1966 aux États-Unis, le premier langage de programmation destiné aux enfants, LOGO, était créé par Seymour Papert, Daniel Bobrow, Richard Grant, Cynthia Solomon, Frank Frazier et Wallace Feurzeig (2010). LOGO n'était alors pas encore très connu en France. Au début des années 1980, Seymour Papert publie l'ouvrage intitulé *Jaillissement de l'esprit : ordinateurs et apprentissage.* Papert (1980) y résumait ses idées autour du « constructionnisme », le cadre conceptuel qu'il avait conçu pour aider les enfants à mieux apprendre et mieux réfléchir, à l'aide de l'activité de programmation en LOGO. L'apparition de cet ouvrage a popularisé LOGO et a considérablement bouleversé les idées de l'époque sur ce qu'il était possible de faire avec les ordinateurs et les élèves.

Suite à ces premières réflexions, les textes de l'éducation nationale de 1983 mettent l'accent sur l'initiation des jeunes élèves à l'informatique (Baron & Bruillard, 1996). Baron et Bruillard expliquent que l'objectif principal au cours de cette période a été d'apprendre aux élèves à programmer et à développer une « pensée algorithmique ». En 1985, les politiques

éducatives en France decident de lancer le Plan Informatique pour Tous. Il s'agissait d'un projet assez ambitieux et très bien doté en matière de financement national (Baron, 1989). Son objectif était :

> « [...]d'équiper en 1985 la totalité des écoles de tous les niveaux avec des « nano-réseaux » de micro-ordinateurs, et d'installer ainsi 11 000 « ateliers informatiques » dans les établissements scolaires, en les ouvrant au public en dehors des heures d'enseignement. Une initiation à l'informatique d'une semaine était prévue pour plus de 100 000 enseignants » (Baron, 1989, p. 76).

Le Plan Informatique pour Tous concernait principalement les établissements du premier degré et l'approche dominante était celle de LOGO (Baron & Drot-Delange, 2016). Les auteurs notent qu'après un changement politique en 1986, cette initiative prometteuse prit fin, un an seulement après sa mise en place. Les nouvelles priorités politiques tournent autour de l'informatique conçue en tant qu'outil (Baron & Drot-Delange, 2016). Ils se réfèrent à la circulaire 87 n° 87-160 du juin 1987, intitulée *Utilisations des équipements informatiques des écoles élémentaires,* en y soulignant l'absence de l'activité de programmation. L'utilisation des logiciels comme ceux de traitement texte caractérise l'informatique de la fin des années 1980 jusqu'au début des années 2000 (Baron & Drot-Delange, 2016).

Certains extraits de la rubrique *maniement des ordinateurs* du bulletin officiel (ci-après BO) spécial du n° 7 du 26 août 1999 présentés par Baron et Bruillard (2001), permettent de comprendre la vision adoptée pour l'informatique à cette période : « Écriture de textes et utilisation de quelques logiciels spécifiques à l'école primaire (dessin etc), de CD Rom » (p. 167). Une telle vision rend inutile l'organisation des formations exhaustives, car la manipulation des logiciels semble pouvoir être facilement acquise par les enseignants chargés de cet enseignement (Baron & Drot-Delange, 2016).

Au début du XXIème siècle, l'image d'une informatique au service des autres disciplines est toujours au centre des réflexions politiques (Baron & Drot-Delange, 2016). L'enseignement de l'informatique prend désormais la forme d'une liste des compétences et de savoir-faire à acquérir (Baron & Bruillard, 2011). Le ministère de l'Éducation Nationale met en place le B2i (brevet informatique et internet)[31], un système de certification des compétences en informatique. Lors de sa création en 2000, le B2i comportait deux niveaux : l'école primaire et le collège (BO n° 42 du 23 novembre 2000). Les nouvelles directives de 2006 ont permis

31 https://www.education.gouv.fr/bo/2006/29/MENE0601490A.htm

l'ajout d'un troisième niveau à obtenir à la fin du lycée (BO n° 42 du 16 novembre 2006). Les textes officiels nous aident à préciser les objectifs de ce certificat :

> « Il appartient à l'école de faire acquérir, par chaque élève, les compétences lui permettant d'utiliser de façon réfléchie et efficace ces technologies[32] et de contribuer à former ainsi des citoyens autonomes, responsables, doués d'esprit critique. […] Le B2i atteste l'acquisition d'un ensemble de compétences développées par mes élèves ou les apprentis, tout au long de leur cursus [...] » (BO n° 42 du 16 novembre 2006).

Nous observons dans cet extrait que les usages des technologies tiennent toujours une place importante dans les préoccupations des pouvoirs publics. Pourtant, dans ces programmes il n'y a aucune directive concernant la transmission de ces savoir-faire aux élèves. En 2008, le B2i est devenu obligatoire à la fin du collège pour obtenir le diplôme du brevet (Baron & Bruillard, 2011). En 2016, il a été supprimé et à la rentrée 2017, a été remplacé par PIX [33]. Baron et Drot-Delange (2016) remarquent que malgré ses limites, le B2i peut être considéré comme un pas en avant « vers la reconnaissance du fait qu'il y a, malgré tout, quelque chose à acquérir en informatique » (p. 54).

La publication du rapport de l'Académie des Sciences en 2013 marque une nouvelle phase dans l'histoire de l'introduction de l'informatique à l'éducation obligatoire en France. Ce rapport est inscrit dans le mouvement international de prise en compte de l'informatique de l'école primaire. Il propose :

> « […] de mettre en place un enseignement de science informatique depuis le primaire jusqu'au lycée, orienté vers la compréhension et la maîtrise de l'informatique, et dépassant donc largement les seuls usages des matériels et logiciels. Cette mise en place ne doit plus être différée. » (Institut de France. Académie des sciences, 2013, p. 6).

Nous remarquons que ce rapport prend un positionnement totalement opposé à ce qui a été mis en place dans le pays en informatique, de la fin des années 1980 et jusqu'en 2013. La simple utilisation de certains outils informatiques ne constitue plus un objectif suffisant, et un apprentissage de la science informatique est prescrit. Un autre point qui attire notre attention dans ce rapport est le fait qu'il propose enfin un enseignement de l'informatique dès l'école primaire (approche mise en évidence seulement dans les années 1980). Cette proposition est harmonisée avec la tendance exprimée dans plusieurs rapports du début des années 2010, préconisant une initiation précoce des élèves à l'informatique, à ses concepts et à ses pratiques. Le rapport recommande aussi un curriculum possible avec trois modes d'apprentissages différents : « la découverte, l'acquisition d'autonomie et la maîtrise des

32 Ici les technologies de l'information et de la communication (TIC).
33 https://pix.fr/

concepts » (Institut de France. Académie des Sciences, 2013, p. 20). « La découverte » serait l'objectif visé en maternelle et en primaire, « l'acquisition d'autonomie », au collège, et enfin « la maîtrise des concepts » au lycée et dans le supérieur.

En ce qui concerne le groupe d'âge qui nous intéresse, de 5 à 7 ans, l'Académie des Sciences souligne dans ce texte que « la découverte » pourrait se faire : « avec des langages de programmation adaptés à [l'] âge [des élèves] ou en mode débranché » (Institut de France. Académie des Sciences, 2013, p. 21). Le rapport met l'accent sur la complémentarité entre les activités avec un ordinateur et les activités « débranchées »[34] et sur l'importance de donner la possibilité aux élèves de tester les deux (Institut de France. Académie des Sciences, 2013). Les activités avec l'ordinateur visent à amener les élèves à poser des questions sur le fonctionnement des objets informatiques, et ainsi à découvrir certains concepts informatiques. Les activités dites « débranchées », cherchent à les familiariser avec des notions clés de l'informatique telles que le langage, l'information et l'algorithme (Institut de France. Académie des Sciences, 2013).

À la rentrée 2016, la programmation fait son retour dans les programmes scolaires. Il s'agit d'un retour très attendu de la part des chercheurs en informatique et en sciences de l'éducation, mais pas très souhaitée par la majorité des enseignants chargés de cet enseignement. Avant de présenter le contexte de ce retour, nous allons faire un petit rappel de l'organisation du système d'éducation nationale français.

La scolarité obligatoire (de 6 à 16 ans) en France est organisée en 4 cycles. Les cycles qui nous intéressent pour ce travail de thèse sont le cycle 1 ou *cycle des apprentissages premiers,* qui concerne la maternelle (petite, moyenne, grande section) et le cycle 2 ou *cycle des apprentissages fondamentaux,* qui concerne les classes de CP, CE1 et CE2.

Le socle commun de connaissances, de compétences et de culture[35] se trouve au centre de ce système et il couvre les cycles 2, 3 et 4 (BO n° 17 du 23 avril 2015). Il s'agit de la référence principale des acteurs du système éducatif. (BO n° 17 du 23 avril 2015). Selon les textes officiels, « [l]e socle commun identifie les connaissances et compétences qui doivent être acquises à l'issue de la scolarité obligatoire » (BO n° 17 du 23 avril 2015, p. 5). De plus, « la maîtrise des acquis du socle commun doit se concevoir dans le cadre du parcours scolaire de

34 Les activités sans ordinateur.
35 https://www.education.gouv.fr/le-socle-commun-de-connaissances-de-competences-et-de-culture-12512

l'élève et en référence aux attendus et objectifs de formation présentés par les programmes de chaque cycle » (BO n° 17 du 23 avril 2015, p. 5). Mis en place depuis la rentrée 2016, ce socle est réparti en cinq domaines différents.

En analysant le socle, nous distinguons des contenus différents en lien avec l'informatique dans trois de ces cinq domaines. Les aspects de l'informatique qui nous intéressent le plus dans le contexte de cette recherche, c'est-à-dire la programmation et l'algorithmique, font partie du premier domaine, intitulé *Les langages pour penser et communiquer*. L'élève doit « comprendre, s'exprimer en utilisant les langages mathématiques, scientifiques et informatiques » (BO n° 17 du 23 avril 2015, p. 6). En outre, le texte propose que les élèves acquièrent les différentes connaissances et compétences suivantes :

> « [L'élève] sait que des langages informatiques sont utilisés pour programmer des outils numériques et réaliser des traitements automatiques de données. Il connaît les principes de base de l'algorithmique et de la conception des programmes informatiques. Il les met en œuvre pour créer des applications simples » (BO n° 17 du 23 avril 2015, p. 6).

En plus de cette nouveauté, la rentrée 2016 a été aussi caractérisée par l'arrivée des nouveaux programmes scolaires pour l'école maternelle[36], l'école primaire et le collège[37]. Les programmes scolaires en France permettent la mise en œuvre du socle commun des connaissances et de compétences, car ils précisent en effet « les spécificités du cycle », « les contributions des enseignements au socle commun » et « les attendus de fin de cycle dans chaque enseignement » (BO n° 11 du 26 novembre 2015).

En analysant les programmes scolaires Vandevelde et Fluckiger (2020) ont repéré trois types de contenus différents en lien avec l'informatique : « l'apprentissage du fonctionnement des technologies », « l'apprentissage de l'algorithmique et de la programmation » et « l'apprentissage de l'utilisation des outils informatisés » (p.3). Cette analyse a aussi montré que les programmes ont été conçus de façon à promouvoir une évolution des connaissances et des compétences à acquérir par les élèves entre les cycles.

En ce qui concerne l'apprentissage de la programmation et de l'algorithmique, les auteurs notent qu'au cycle 2, les programmes scolaires s'intéressent au codage des déplacements, au

36 https://www.education.gouv.fr/au-bo-special-du-26-mars-2015-programme-d-enseignement-de-l-ecole-maternelle-3413

37 https://www.education.gouv.fr/au-bo-special-du-26-novembre-2015-programmes-d-enseignement-de-l-ecole-elementaire-et-du-college-3737

cycle 3 l'accent est mis sur l'acquisition des notions et au cycle 4, il y a une mise en application des concepts vus précédemment (Vandevelde & Fluckiger, 2020).

En ce qui concerne le cycle 2, qui nous intéresse le plus, nous observons que la programmation est mentionnée dans l'onglet *Espace et Géométrie* de l'enseignement de Mathématiques. Plus précisément, elle est associée à un des attendus de fin de cycle *(Se) repérer et (se) déplacer en utilisant des repères*. Le Tableau 1 ci-dessus présente comment la programmation est intégrée dans les programmes scolaires du cycle 2, parus au BO n° 17 du 26 novembre 2015.

Connaissances et Compétences associées	Exemples de situations, d'activités et de ressources pour l'élève
(Se) repérer et (se) déplacer en utilisant des repères	
S'orienter et se déplacer en utilisant des repères. Coder et décoder pour prévoir, représenter et réaliser des déplacements dans des espaces familiers, sur un quadrillage, sur un écran. Repères spatiaux. Relations entre l'espace dans lequel on se déplace et ses représentations.	Parcours de découverte et d'orientation pour identifier des éléments, les situer les uns par rapport aux autres, anticiper et effectuer un déplacement, le coder. Réaliser des déplacements dans l'espace et les coder pour qu'un autre élève puisse les reproduire. Produire des représentations d'un espace restreint et s'en servir pour communiquer des positions. Programmer les déplacements d'un robot ou ceux d'un personnage sur un écran.

Tableau 1: Extrait du programme d'enseignement du cycle 2 (BO n° 11 du 26 novembre 2015, p. 84)

Dans le Tableau 1, nous pouvons observer que l'apprentissage de la programmation à travers l'utilisation de robots et de logiciels de programmation, comme Scratch ou ScratchJr[38], a été prescrit pour aider les élèves à se repérer dans l'espace. Nous remarquons ici que l'activité de programmation est intégrée dans le programme du cycle 2 en raison des effets positifs qu'elle pourrait avoir sur un autre domaine, celui du repérage dans l'espace et non pour ce qu'elle pourrait apporter en elle-même.

Nous avons aussi identifié une référence à la programmation dans *Repères de progressivité,* toujours dans l'onglet *Espace et Géométrie* de l'enseignement de Mathématiques, où il est écrit :

> « Il est possible, lors de la résolution de problèmes, d'aller au-delà des repères de progressivité identifiés pour chaque niveau. Au CP, la représentation des lieux et le codage des déplacements se situent dans la classe ou dans l'école, puis dans le quartier proche, et

38 L'extrait du programme nous fait penser à ces logiciels, car ils parlent des déplacements d'un personnage d'un personnage sur un écran. Ces logiciels permettent un tel déplacement.

> au CE2 dans un quartier étendu ou le village. Dès le CE1, les élèves peuvent coder des déplacements à l'aide d'un logiciel de programmation adapté, ce qui les amènera au CE2 à la compréhension, et la production d'algorithmes simples »(BO n° 11 du 26 novembre 2015, p. 86).

Dans cet extrait, nous repérons une proposition de progression possible en ce qui concerne l'apprentissage de la programmation dans le cadre du cycle 2.

Les notions d'algorithme et d'objet programmable apparaissent aussi dans le programme du cycle 3, dans l'onglet *Matériaux et Objets techniques* de l'enseignement Science et Technologie. La programmation des déplacements d'un robot ou d'un personnage sur un écran se trouve aussi dans l'onglet *Espace et Géométrie* de l'enseignement de Mathématiques du cycle 3. Dans le cycle 4, la programmation est présente dans les enseignements de Mathématiques et de Technologie. Le développement d'une pensée algorithmique, la résolution de problèmes et l'émancipation des élèves face au monde numérique qui les entoure, figurent parmi les objectifs cités dans les programmes de ce cycle.

Vandevelde et Fluckiger (2020) soulignent que l'informatique est présente dans les programmes scolaires de 2015 sous une forme transdisciplinaire, c'est-à-dire qu'elle est intégrée aux autres disciplines et non comme une discipline reconnue, indépendante des autres.

Des nouveaux programmes scolaires ont été mis en application pour l'école maternelle, l'école primaire et le collège à la rentrée 2021 (BO n° 31 du 30 juillet 2020). Nous pouvons donc identifier des similarités et des différences par rapport aux programmes précédents en ce qui concerne la prise en compte de la programmation.

Pour le cycle 2, cycle où l'initiation des élèves à la programmation débutait dans les programmes de 2015, il existe toujours, dans l'onglet *Espace et géométrie* de l'enseignement des Mathématiques, un point sur la programmation de déplacements de robots ou d'un personnage sur un écran, associé à l'attendu de fin de cycle *(Se) repérer et (se) déplacer en utilisant des repères et des représentations* cette fois-ci. Toujours au cycle 2, les attendus de fin de cycle pour l'enseignement des mathématiques ne sont plus présentés, comme c'était le cas dans les programmes de 2015, à l'aide de ces deux onglets : C*onnaissances et compétences associées* et *Exemples de situations, d'activités et de ressources pour l'élève*. En ce qui concerne l'attendu de fin de cycle (*Se) repérer et (se) déplacer en utilisant des repères et des représentations* et la référence à l'activité de programmation, nous retrouvons des

compétences provenant d'une fusion des éléments pré-existants dans les onglets supprimés des programmes de 2015.

Certains éléments sont identiques d'un programme à l'autre, d'autres ont disparu en 2020. Le Tableau 2 qui suit présente une analyse comparative de l'introduction de la programmation dans les programmes du cycle 2 de 2020 (BO n°31 du 30 juillet 2020, p.91) par rapport aux programmes de 2015 (BO n° 11 du 26 novembre 2015, p. 84), afin d'identifier les similarités et les différences existantes.

(Se) repérer et (se) déplacer en utilisant des repères et des représentations		
Ce qui est identique	Ce qui a été fusionné	Ce qui a été disparu
S'orienter et se déplacer en utilisant des repères. Réaliser des déplacements dans l'espace et les coder pour qu'un autre élève puisse les reproduire.	Programmer les déplacements d'un robot ou ceux d'un personnage sur un écran. Repères spatiaux. Relations entre l'espace dans lequel on se déplace et ses représentations.	Coder et décoder pour prévoir, représenter et réaliser des déplacements dans des espaces familiers, sur un quadrillage, sur un écran. Parcours de découverte et d'orientation pour identifier des éléments, les situer les uns par rapport aux autres, anticiper et effectuer un déplacement, le coder.

Tableau 2: Analyse comparative de l'introduction de la programmation dans les programmes de 2020, par rapport aux programmes de 2015 (cycle 2)

En plus des éléments mentionnés dans le tableau 2, un autre élément qui a disparu dans les programmes de 2020 est la section *Repères de progressivité* que nous avons citée auparavant. Il y avait dans cette section une proposition explicite pour une initiation des élèves à la programmation avec une progression possible de CP à CE2.

Cette comparaison permet de voir que la programmation apparaît toujours dans les programmes de 2020, mais qu'elle y est beaucoup moins présente qu'en 2015. L'aspect « codage des déplacements d'un robot ou d'un personnage sur un écran » est toujours d'actualité. L'approche adoptée est la même qu'en 2015, c'est-à-dire que la programmation fait toujours partie de l'enseignement de mathématiques. En ce qui concerne les programmes de 2020 pour le cycle 3, nous repérons qu'il manque la section *Repères de progressivité,* de l'enseignement de mathématiques, dans laquelle figurait en 2015 un point explicite pour une préconisant une initiation à la programmation. Concernant les programmes du cycle 4, nous observons qu'ils sont identiques à ceux de 2015, à l'exception d'un paragraphe dédié à

l'enseignement de l'informatique qui disparaît dans la description de l'enseignement de mathématiques du cycle 4 en 2020.

Nous constatons que, malgré un regain d'intérêt à l'échelle internationale pour la transmission d'une culture informatique dès l'école élémentaire et malgré les recommandations du rapport de l'Académie des Sciences publié en 2013, les décideurs politiques ont choisi en 2015 d'introduire l'informatique dans les programmes scolaires à travers les autres disciplines et pas en tant qu'une discipline à part entière. En effet, l'activité de programmation, qui nous intéresse particulièrement dans cette thèse, a été introduite dans les programmes du cycle 2, alors que d'autres pays comme le Royaume-Uni ont choisi de la mettre en place dès l'école maternelle. Cette activité est intégrée de façon très restreinte au cycle 2, dans le cadre de l'enseignement des mathématiques, en lien avec des activités de repérage et de déplacement dans l'espace. Elle est un peu plus présente dans le cycle 3 et beaucoup plus présente dans le cycle 4.

Nous avons aussi identifié, dans les programmes, une série d'éléments qui pourraient contribuer à rendre difficile l'intégration de l'informatique et par conséquent de la programmation, dans le système éducatif français. L'existence d'une multitude de termes différents référant à cet enseignement dans les programmes, l'absence d'effort pour désambiguïser ces termes, ainsi que l'hétérogénéité des contenus associés à cet enseignement, ont pu poser problème aux enseignants qui en étaient chargés. De plus, l'informatique et en particulier la programmation sont moins présents dans les programmes de 2020, élément qui indique un recul de la part des pouvoirs politiques par rapport aux programmes de 2015. On constate malheureusement que le risque indiqué par Baron et Drot-Delange (2016), concernant la délégation de l'enseignement de l'informatique au périscolaire, devient de plus en plus une réalité, dès lors que l'école semble ne pas avoir les moyens nécessaires pour le prendre en charge.

2.3. Le cas de la Grèce

C'est en 1997 qu'on peut observer la première introduction de l'informatique à l'école primaire en Grèce dans le cadre du programme scolaire destiné à l'informatique, intitulé *Cadre unifié des programmes d'études en Informatique* [notre traduction][39] (Υπουργείο

39 Notre traduction du grec : *Ενιαίο Πλαίσιο Προγραμματος Σπουδών Πληροφορικής*.

Εθνικής Παιδείας και Θρησκευμάτων. Παιδαγωγικό Ινστιτούτο, 1997). Il s'agissait alors d'un enseignement optionnel puisque l'État grec n'était pas en mesure de mettre en place un plan national pour équiper les écoles primaires du pays en matériel informatique nécessaire. Cet enseignement concernait toutes les classes[40] de l'école primaire, mais surtout les E΄ et ΣΤ΄, l'équivalent dans le système éducatif français des classes de CM2 et 6e. L'activité de programmation ne faisait pas partie des objectifs indiqués dans ce programme. Son objectif général était alors de faire découvrir les différentes utilisations des technologies aux élèves, à l'aide d'activités organisées dans le cadre d'autres matières. La maternelle n'était pas encore entrée dans le paysage des textes officiels à ce moment-là.

En 2003, les nouveaux programmes scolaires font une place à l'informatique à l'école primaire (*Διαθεματικό Ενιαίο Πλαίσιο Προγράμματος Σπουδών Πληροφορικής [ΔΕΠΠΣ Πληροφορικής]*, 2003) et pour la première fois aussi à l'école maternelle (*Διαθεματικό Ενιαίο Πλαίσιο Προγραμμάτων Σπουδών για το Νηπιαγωγείο [ΔΕΠΠΣ Νηπιαγωγείο]*, 2003). En ce qui concerne les objectifs en lien avec l'informatique pour le primaire, nous pouvons identifier la découverte des fonctionnalités de l'ordinateur, le développement d'une litteratie numérique, ainsi que la reconnaissance des technologies de l'information et de la communication (ΔΕΠΠΣ Πληροφορικής, 2003).

Ce programme prescrit une intégration de l'informatique dans d'autres disciplines, c'est-à-dire une approche transdisciplinaire. Il souligne d'ailleurs que l'informatique ne devrait pas être perçue en tant qu'objet d'enseignement, au regard du fait que le temps scolaire est limité, qu'il n'offre pas la possibilité d'introduire un nouveau créneau horaire. L'activité de programmation apparaît pour la première fois dans ce programme sous l'axe *Contrôler et Programmer* [notre traduction][41] et elle concerne les deux dernières classes du primaire, E΄(CM2) et ΣΤ΄(6e)(ΔΕΠΠΣ Πληροφορικής, 2003). En effet, il est proposé d'utiliser un langage de programmation de type LOGO pour créer des formes géométriques simples. Selon ce programme, les élèves devraient ainsi comprendre que l'ordinateur est contrôlé par l'humain à l'aide des instructions écrites dans un langage de programmation précis.

40 L'enseignement primaire en Grèce est organisé en classes. Nous les présentons ici, en ajoutant entre parenthèses le niveau équivalent selon l'organisation du système éducatif français : Α΄(CP), Β΄(CE1), Γ΄(CE2), Δ΄(CM1), Ε΄(CM2), ΣΤ΄(6e).

41 Notre traduction du grec : Ελεγχω και Προγραμματίζω.

Dans le programme scolaire conçu pour la maternelle et intitulé *Cadre Unifié interdisciplinaire des programmes d'études pour l'école maternelle* [notre traduction][42], l'informatique est présentée comme partie intégrante des activités de toutes les matières (ΔΕΠΠΣ Νηπιαγωγείο, 2003). L'enjeu de cet enseignement est la familiarisation des élèves avec les fonctionnalités de l'ordinateur et ses potentielles utilisations dans l'éducation. Le rôle de l'enseignant dans ce processus de découverte est jugé primordial. L'activité de programmation ne fait pas encore partie des objectifs de l'enseignement de l'informatique en maternelle.

En 2011, de nouveaux programmes complémentaires à ceux de 2003 arrivent pour l'école primaire et la maternelle. Selon le programme du primaire pour l'informatique, la mission de l'école au XXIème siècle est de donner au futur citoyen les bagages nécessaires pour pouvoir vivre dans un environnement de plus en plus informatisé (Παιδαγωγικό Ινστιτούτο, 2011a). Ce programme met en évidence l'importance de la litteratie informationnelle et les nouveaux types de compétences à acquérir par les élèves à l'ère actuelle.

Par rapport aux programmes précédents, l'activité de programmation trouve dans celui de 2011 une place plus importante. Elle fait partie de l'axe *Rechercher, découvrir et résoudre des problèmes avec les TIC* [notre traduction][43]. Elle concerne à nouveau les deux dernières classes de l'école primaire, Ε΄(CM2) et ΣΤ΄(6^{e}). Le programme propose de réaliser 10 heures sur la programmation en Ε΄(CM2) et 12 heures en ΣΤ΄(6^{e}). Il présente plusieurs nouveautés par rapport aux précédents. Par exemple, les objectifs d'apprentissage en lien avec la programmation sont beaucoup plus nombreux, détaillés et précis qu'en 2003. Ils spécifient que l'élève doit, entre autres, savoir utiliser un logiciel de programmation visuel, exécuter des programmes, décrire à l'oral les étapes d'un algorithme, écrire un programme pour obtenir un résultat spécifique, décomposer un programme, identifier des événements en programmation et déboguer. La répétition et les procédures font partie des objectifs d'apprentissage pour la dernière classe du primaire (ΣΤ΄) (Παιδαγωγικό Ινστιτούτο, 2011a).

Une autre nouveauté du programme de 2011 est qu'il propose des activités détaillées guidant l'enseignant du primaire dans ce qu'il pourrait mettre en place avec ses élèves afin d'atteindre les objectifs d'apprentissage spécifiés plus haut. Nous pouvons citer le jeu d'enfant robot, où un élève représente le robot et un autre le programmeur, la création de petits projets à l'aide

42Notre traduction du grec : *Διαθεματικό Ενιαίο Πλαίσιο Προγραμματος Σπουδών για το Νηπιαγωγείο.*
43Notre traduction du grec : Διερευνώ, ανακαλύπτω και λύνω προβλήματα με ΤΠΕ.

d'un logiciel de programmation visuel, la création de formes géométriques simples ou plus complexes, mais aussi l'introduction de la répétition et des procédures dans des programmes qu'ils ont déjà réalisés. Une gamme assez large de logiciels de programmation visuels, ainsi que des robots programmables, sont proposés aux enseignants comme matériel à utiliser. Les premières classes de l'école primaire ne sont pas jugées prioritaires pour la réalisation d'expérimentations en programmation (Παιδαγωγικό Ινστιτούτο, 2011a).

Concernant la maternelle, le programme de 2011 prescrit pour la première fois une initiation des jeunes élèves à la programmation. Ce nouveau programme sépare l'informatique en quatre axes différents et l'activité de programmation appartient à l'axe *Rechercher, s'expérimenter, découvrir et résoudre des problèmes avec les TIC* [notre traduction][44] (Παιδαγωγικό Ινστιτούτο, 2011b). L'objectif principal cité dans le texte officiel en lien avec la programmation est de donner la possibilité aux élèves : « de développer leur pensée critique, leur capacité à prendre des décisions, à résoudre des problèmes et à modéliser le savoir à l'aide des jouets programmables »[notre traduction] (Παιδαγωγικό Ινστιτούτο, 2011b, p. 141).

Une série d'activités est proposée aux enseignants de maternelle afin de les aider à réaliser l'objectif fixé. Il est conseillé de faire découvrir aux élèves les robots programmables de type Bee-Bot et d'organiser des activités de résolution des problèmes à l'aide de ces objets afin de familiariser les élèves avec le repérage dans l'espace, le déplacement, le dénombrement, la mesure d'une distance, etc. Le programme propose également aux enseignants de créer, avec leurs élèves, des tapis comportant des repères spatiaux (les prénoms des élèves, des lettres, des formes géométriques, etc.) sur lesquels les robots programmables se déplacent.

La résolution de problèmes à l'aide des robots programmables pourrait aussi servir, d'après le programme, à amener les élèves à parler de la démarche qu'ils ont mise en place pour les résoudre et à justifier leurs choix. Enfin, le rôle de l'enseignant dans ce processus et les aides didactiques utilisées sont également mis en avant dans le nouveau programme pour l'informatique en maternelle (Παιδαγωγικό Ινστιτούτο, 2011b).

Plus près de nous, en 2021, de nouveaux programmes font leur apparition pour le primaire et la maternelle. Les textes officiels pour le primaire mettent l'accent pour la première fois sur le développement d'une pensée informatique par les élèves (Ινστιτούτο Εκπαιδευτικής Πολιτικής [ΙΕΠ], 2021b). L'acquisition d'une littératie numérique et informationnelle fait

44Notre traduction du grec : Διερευνώ, πειραματίζομαι, ανακαλύπτω και λύνω προβλήματα με τις ΤΠΕ.

aussi partie des objectifs visés pour ce groupe d'âge. L'algorithmique et l'activité de programmation constituent un des cinq axes principaux autour desquels est organisé l'enseignement de l'informatique à l'école primaire.

C'est la première fois que la programmation occupe une place si importante dans les programmes du primaire. De plus, il existe une évolution des objectifs à atteindre en lien avec cet axe dans toutes les classes du primaire, et plus seulement dans les deux dernières, comme c'était le cas dans les programmes précédents qui se focalisaient uniquement sur ces deux dernières années. Il est par exemple prescrit que « [l]es élèves devraient, à la fin du primaire, pouvoir créer leurs propres programmes en utilisant un langage de programmation éducative »[notre traduction] (ΙΕΠ, 2021b, p. 5). Il est également proposé aux enseignants d'utiliser un scénario pédagogique pour la mise en place de cet enseignement dans leurs classes.

Le programme pour le primaire englobe une large gamme des notions en lien avec la programmation, dont les plus importantes sont l'algorithme, le programme, la séquence, la répétition, les événements, l'expression conditionnelle, la répétition conditionnelle, les variables et le parallélisme (ΙΕΠ, 2021b). Ces notions sont intégrées progressivement dans les classes de l'école primaire, en respectant les caractéristiques d'âge des élèves de chaque niveau.

Par exemple les élèves de la classe A΄(CP) seront initiés à des notions plus simples, comme la séquence, le programme et l'algorithme, tandis que les élèves de la classe ΣΤ΄(6^{e}) seront initiés à des notions plus difficiles, comme les variables, la répétition conditionnelle ou les répétitions imbriquées. En plus des notions à découvrir, le programme prescrit leur utilisation opérationnelle par l'élève dans le cadre de la résolution des problèmes, ainsi que le développement des compétences liées à la conception, l'exécution et au débogage d'un programme (ΙΕΠ, 2021b). En poursuivant la logique des programmes de 2011, une série d'activités est proposée aux enseignants de primaire afin de les aider à mettre en place cet enseignement avec leurs élèves. Nous pouvons citer comme exemples le jeu d'enfant-robot, l'identification d'un motif qui est répété dans la vie quotidienne, puis dans une séquence de commandes, la représentation graphique de conditions Si...alors ou l'identification et la correction d'erreurs dans un programme donné.

Les programmes de 2021 prescrivent aussi l'initiation des élèves à la litteratie numérique dès la maternelle (Ινστιτούτο Εκπαιδευτικής Πολιτικής [ΙΕΠ], 2021a). Dans les textes officiels pour ce groupe d'âge, l'informatique est associée au langage dans la thématique *Enfant et communication* [notre traduction][45] . Elle est présentée en même temps en tant qu'objet et outil d'apprentissage et son enseignement est reparti sur trois axes différents, dont l'axe *Découverte, Programmation et Jeu Numérique* [notre traduction][46], qui porte, entre autres, sur la programmation.

Deux sections de ce dernier axe concernent l'initiation des jeunes élèves à la programmation tactile et visuelle, ainsi qu'à la robotique pédagogique. Les objectifs à atteindre en lien avec la programmation sont organisés en connaissances, compétences et postures.

Les élèves de maternelle doivent par exemple connaître les commandes de déplacement d'un robot programmable et les notions de base de programmation, c'est-à-dire, la séquence, la répétition et l'instruction conditionnelle (ΙΕΠ, 2021a). En termes de compétences, ils doivent par exemple savoir utiliser les commandes de déplacement, la séquence et la répétition afin de résoudre des problèmes de programmation. Ils doivent aussi être capables d'exécuter et déboger leurs programmes créés à l'aide d'un objet programmable ou d'un logiciel de programmation tactile.

Concernant les postures, le programme prescrit une collaboration entre élèves pour résoudre les problèmes de programmation. En comparant les objectifs du programme de 2021 à ceux de 2011, nous pouvons remarquer que dans le programme le plus récent, les objectifs sont présentés de façon plus organisée et développent encore davantage les connaissances et compétences à faire acquérir aux élèves. C'est la première fois que les notions de séquence, de répétition et d'instruction conditionnelle font partie des programmes de maternelle en Grèce. Le texte officiel pour la maternelle propose également des activités que les élèves peuvent faire pour développer les objectifs préconisés (ΙΕΠ, 2021a).

Ce retour historique sur la prise en compte de l'informatique et surtout de l'activité de programmation dans les programmes scolaires de l'école maternelle et de l'école primaire en Grèce nous a permis de comprendre que le pays a reconnu très tôt leur importance et reste constant dans ses efforts pour les intégrer au mieux dans la scolarité obligatoire. La première

45 Notre traduction du grec : *Παιδι και επικοινωνία*.
46 Notre traduction du grec : *Ανακάλυψη, Προγραμματισμός και ψηφιακό παιχνίδι*.

intégration de la programmation dans les programmes scolaires a été réalisée en 2003 pour l'école primaire et en 2011 pour l'école maternelle.

Nous remarquons qu'il existe depuis plusieurs années un intérêt particulier des décideurs politiques pour améliorer la mise en place de cet enseignement dans les programmes scolaires. L'enseignement de l'informatique et notamment de la programmation semble devenir de plus en plus important et organisé au cours des années. Les objectifs visés deviennent plus nombreux, détaillés et précis. Ainsi, en 2021 des notions comme celles d'algorithme, de programme, de séquence et de répétition font partie des programmes scolaires dès la maternelle.

Chapitre 2. L'enseignement de l'informatique à l'épreuve de la réalité

Dans ce chapitre, nous commençons par souligner l'importance de la formation des enseignants pour une introduction efficace d'un enseignement de la programmation en école élémentaire. Nous présentons brièvement les démarches en lien avec la formation mises en place par les différents pays qui ont opté pour cette introduction dans leurs programmes. Nous détaillons ensuite le problème de la formation des enseignants en France. Nous consultons le site d'Eduscol et nous montrons l'offre de formation et les ressources proposées aux enseignants français, en lien avec l'enseignement de l'informatique et surtout d'un de ses aspects, la programmation. Nous présentons le plan national de formation en informatique pour les enseignants du premier et du second degré en Grèce. Dans le cadre du regain d'intérêt international pour la didactique de l'informatique et en particulier, pour l'apprentissage de la programmation, des projets de recherche émergent dans le domaine des sciences de l'éducation pour étudier la question et essayer d'apporter des réponses utiles aussi bien pour les politiques éducatives que pour le corps enseignant. Nous mettons en avant en particulier deux projets de recherche soutenus par l'Agence Nationale de la Recherche (ANR) en France, dans lesquels s'inscrit ce travail de thèse. Il s'agit de DALIE et IE-CARE. À la fin de ce chapitre, nous présenterons la question initiale de cette recherche.

1. La formation des enseignants : une difficulté majeure

Pour toute réforme des programmes scolaires, la formation des enseignants est un élément clé. Bozat et al. (2014) soulignent que les enseignants devraient compléter au fur et à mesure leur formation initiale au cours de leur carrière, afin de pouvoir s'adapter aux nouveautés émergeant dans le domaine de l'éducation.

L'émergence d'un mouvement international favorable à l'introduction de l'informatique à l'école primaire a fait apparaître de nouvelles possibilités, mais aussi de nouveaux défis à relever pour les enseignants. La plupart des pays qui ont procédé à cette introduction ont mis

en place des démarches pour former leur corps enseignant afin de pouvoir le soutenir dans la mise en place de ce nouvel enseignement. Par exemple, des MOOCs ont été mis en place pour les enseignants en Australie (Vivian, Falkner, & Falkner, 2014) et en Norvège (Heintz et al., 2016), le CAS a pris en charge cette responsabilité au Royaume-Uni (Brown, Sentance, et al., 2013) et l'Université de Turku en Finlande offre un programme de formation sur l'enseignement de l'informatique aux enseignants (Heintz et al., 2016). Dans le cas où certains enseignants souhaiteraient mettre en place un enseignement à l'informatique, une absence de formation pourrait conduire à la mise en place d'activités souffrant de défauts de conceptualisation et cela pourrait influencer les apprentissages des élèves (Spach, 2017). Vandeput (2017) explique que « la richesse des usages est directement liée à la connaissance profonde des outils et que cette dernière est intimement liée à une perception correcte de la manière dont s'effectue le traitement de l'information » (p. 47).

Très souvent, les enseignants n'ont pas conscience du fait qu'ils enseignent déjà l'informatique, ou du moins des savoirs et compétences utilisés aussi en informatique. Aussi, il est essentiel que les experts en didactique de l'informatique et les formateurs les aident à voir les liens entre ce qu'ils enseignent déjà et l'informatique (Sabitzer et al., 2014). La formation leur permettra ainsi de conscientiser ce qu'ils font déjà mais aussi de développer des compétences pour l'enseigner de la meilleure manière. Afin d'illustrer ce lien entre enseignements ordinaires et enseignements de l'informatique, Bruillard, lors d'une interview réalisée en 2018 par Archiprod[47], donne l'exemple des tableaux qu'on retrouve dans bon nombre d'écoles maternelles pour présenter les noms des élèves, les taches à réaliser, etc. Il explique qu'il s'agit de tableaux à deux dimensions, que l'on utilise aussi en informatique, où ils peuvent avoir trois voire quatre dimensions. D'après Bruillard, faire le lien entre l'utilisation de ce tableau en classe et en informatique pourrait être une première ouverture au monde de l'informatique et à ses différentes possibilités.

Vivian et al. (2014) relèvent que le défaut de préparation des enseignants pour l'enseignement de l'informatique, surtout lors de ce « momentum » global autour de son introduction à l'école primaire, risque de mettre en danger cette initiative, car rien ne garantit alors que cet enseignement sera mis en place de telle sorte qu'il attire l'attention des élèves et qu'il produise les effets souhaités sur leurs apprentissages.

47 Interview disponible ici : https://edu.ge.ch/site/archiprod/pensee-informatique-a-lecole-interview-deric-bruillard/

Les résultats d'une recherche réalisée par Yadav et al. (2014), qui a entrepris une analyse comparative entre deux groupes d'enseignants, les uns ayant reçu une formation à la pensée informatique, les autres n'ayant pas reçu cette formation, montrent que les enseignants formés ont une meilleure compréhension de cette notion et une attitude favorable à son introduction dans la classe. À l'inverse, le groupe contrôle a mal compris cette notion et les enseignants de ce groupe sont beaucoup plus réservés quant à son intégration dans la classe. Duncan et al. (2014) remarquent que les enseignants qui ont confiance en eux sont les mieux placés pour mettre en place des expérimentations réussies en lien avec la programmation pour les élèves. Si les enseignants ne souhaitent pas mettre en place cet enseignement ou ne se sentent pas à l'aise vis-à-vis de celui-ci, il est préférable de ne pas l'enseigner du tout, sans quoi l'expérience risque d'être plus négative que bénéfique.

Le rapport de l'Académie des Sciences (2013) a émis une série de recommandations pour la formation des enseignants, considérée comme « une priorité absolue » (p. 5). Pour les enseignants du premier degré, le rapport propose qu'un cours destiné à la science informatique fasse partie du cursus des ESPE (Écoles Supérieures du Professorat et de l'Éducation), récemment devenus INSPE (Instituts Nationaux Supérieures du Professorat et de l'Éducation)[48]. Dans ce rapport, il est aussi préconisé que les enseignants du second degré chargés de l'enseignement de l'informatique soient soumis aux mêmes processus de recrutement que les enseignants des autres disciplines.

L'Académie met aussi l'accent sur le fait que tous les enseignants doivent être capables de reconnaître l'impact potentiel de la science informatique sur leurs disciplines. Pour les enseignants en activité, le rapport suggère une formation « par un développement professionnel volontariste, afin que tous puissent initier leurs élèves à cette discipline » (Institut de France. Académie des Sciences, 2013, p. 7). L'Académie souligne aussi que « nous avons besoin d'un véritable plan de formation national pour l'informatique si nous voulons sortir notre pays de l'illettrisme informatique dans lequel il se trouve aujourd'hui » (Institut de France. Académie des Sciences, 2013, p. 29).

Toujours dans le contexte français, Baron et Drot-Delange (2016) mettent en avant une série d'éléments qui éclairent un peu plus le problème de la formation des enseignants. Tout d'abord, les auteurs soulignent qu'il s'agit d'une opération coûteuse dont les effets ne sont

48 Les ESPE sont devenues INSPE dans le cadre de la loi n°2019-791 du 28 juillet 2019 « Pour une École de la confiance » disponible ici https://www.legifrance.gouv.fr/loda/id/JORFTEXT000038829065/.

visibles qu'après un certain temps. Ils rappellent également que la dernière opération de formation importante au niveau national a été réalisée dans les années 1980. Ici, les auteurs se réfèrent au plan informatique pour tous, mis en place en 1985 et pour lequel l'État français avait consacré un budget de deux milliards de francs sur une année pour former les enseignants et équiper les écoles en matériel informatique. Le ministère prévoyait à l'époque que plus de 100 000 enseignants suivraient une formation rémunérée de cinquante heures pendant les vacances de Pâques (Fabius, 1985).

Baron et Drot-Delange (2016) identifient un autre obstacle de l'ordre des représentations : pendant longtemps, l'enseignement de l'informatique n'était pas considéré comme appelant une formation préalable, dans la mesure où l'informatique était perçue comme un ensemble d'outils, faciles à prendre à main par les enseignants. Ils relèvent également la difficulté des institutions de formation des maitres à faire le nécessaire pour bien préparer les enseignants en informatique. Les auteurs se réfèrent aussi au Certificat informatique et internet (C2i)[49], institué en 2002, qui devait alors être obtenu par la majorité des nouveaux enseignants. Pour eux, l'obtention de ce certificat ne garantit pas une expertise suffisante pour enseigner l'informatique et surtout la programmation car le C2i est centré sur l'évaluation des compétences.

En ce qui concerne le point de vue des enseignants français du primaire et du secondaire sur la réforme de 2016, les échanges, formels ou informels, que nous avons pu avoir avec eux dans le cadre des projets de recherche[50] auxquels nous avons participé, nous ont permis de nous faire une idée de la façon dont ils appréhendaient l'introduction de l'informatique et surtout de la programmation dans les programmes scolaires de 2016. Les enseignantes du primaire avec lesquelles nous avons pu discuter dans un cadre informel étaient intéressées par la mise en place d'un « projet informatique » dans leurs classes, mais ne se sentaient pas à l'aise à l'idée de le mettre en place toutes seules, sans formation préalable à l'enseignement de l'informatique.

Les enseignants du second degré que nous avons pu interviewer et dont nous avons enregistré le témoignage dans le cadre du projet ReVEA (Ressources vivantes pour l'enseignement et

49 https://www.education.gouv.fr/node/284519

50 Ici nous faisons référence à trois projets de recherche soutenus par l'Agence nationale de la recherche : ReVEA (Ressources vivantes pour l'enseignement et l'apprentissage), DALIE (Didactique et apprentissage de l'informatique à l'école), IE-CARE (Une recherche pluridisciplinaire au service de l'informatique scolaire).

l'apprentissage) afin de savoir quels types de ressources ils utilisent pour l'introduction des contenus informatiques en lien avec la programmation dans leur cours, ont tous exprimé le besoin d'une auto-formation pour pouvoir mettre en place ce qui est prescrit dans les programmes de 2016. Ils ont aussi trouvé insuffisante la formation d'une demi-journée qui leur a été proposée par l'académie et qui visait essentiellement à présenter la réforme. Les interviewés se sont référés également au site Eduscol[51] en critiquant son contenu, jugeant que ce qui y était proposé à ce moment précis (fin décembre 2016) n'était ni clair, ni suffisant pour eux.

Dans ce contexte, nous avons choisi de consulter[52] le site Eduscol afin d'identifier l'offre de formation et les ressources offertes aux enseignants français en lien avec l'enseignement de l'informatique et surtout d'un de ses aspects, la programmation. Ce site comprend deux onglets qui concernent particulièrement l'enseignant : *j'enseigne* et *je me forme*. L'onglet *je me forme ne propose* aucune formation à l'informatique où à la programmation pour les enseignants.

Dans l'onglet *j'enseigne,* il existe un sous-onglet intitulé *j'enseigne avec le numérique*[53]. Le titre choisi pour l'onglet témoigne de l'orientation essentielle du ministère, qui conçoit l' informatique comme outil au service des autres disciplines. Ce sous-onglet comprend quatre sections ; nous avons identifié des ressources en lien avec l'enseignement de l'informatique et en particulier de la programmation dans trois d'entre elles : *Pratiques pédagogiques*, *Ressources numériques pour l'école*, *Culture numérique*.

Tout d'abord, dans la section *Pratiques pédagogiques*[54], trois rubriques contiennent des ressources en lien avec l'enseignement de l'informatique et surtout l'un de ses aspects, la programmation : Prim à bord, Édubase et La CodeWeek Europe. Prim à bord[55] est « un portail à destination des professeurs des écoles et des formateurs du premier degré. Lancé en janvier 2016 à l'initiative du ministère, il propose chaque semaine des scénarios, des services et des retours d'usage du numérique à l'école primaire »[56]. Il s'agit d'une base de données où tout enseignant, formateur ou inspecteur peut déposer une ressource, qui sera ensuite évaluée par

51 https://eduscol.education.fr/
52 Date de consultation du site : le 12 avril 2022.
53 https://eduscol.education.fr/103/j-enseigne-avec-le-numerique
54 https://eduscol.education.fr/157/pratiques-pedagogiques-avec-le-numerique
55 https://primabord.eduscol.education.fr/
56 https://eduscol.education.fr/196/prim-bord

le comité éditorial de Prim à bord et, le cas échéant, publiée sur le portail. Les ressources sont répertoriées à l'aide des filtres. Nous avons repéré deux filtres en lien avec l'apprentissage de la programmation : *codage/programmation* et *robotique*. En réalisant des recherches sur la base de données grâce aux filtres *codage/programmation*, *robotique*, *cycle 1, cycle 2* et *se former,* nous avons compté quatorze articles qui pourraient servir à former les enseignants sur l'informatique. Dans Prim à bord, les enseignants peuvent aussi trouver des ressources pour se former en informatique. Pour finir, les académies proposent également leurs ressources sur ce portail.

Edubase[57] est quant à elle « une base nationale de scénarios pédagogiques opérée par la direction du numérique pour l'éducation (DNE). Elle permet, à partir d'une interface unique, de rechercher un scénario pédagogique élaboré en académie illustrant un thème de programme en lien avec le numérique éducatif »[58]. Elle est destinée aux enseignants, formateurs et inspecteurs de l'école primaire et secondaire. Les scénarios disponibles sont créés par des enseignants en académie, puis ils sont évalués par les inspecteurs et enfin, ils sont ajoutés dans la base de données. Les ressources sont aussi classifiées à l'aide des filtres. Cette base de données propose également des fils RSS pré-enregistrés pour suivre toutes les nouvelles publications d'une académie ou d'une discipline en particulier. Elle donne aussi la possibilité de créer ses propres fils RSS. Nous avons repéré deux filtres en lien avec la programmation dans la discipline des mathématiques : *algorithmique* et *programmation*. Toutefois, il est à noter que le filtre *algorithmique* ne mène à aucune ressource pour les cycles 1 et 2 et qu'il n'y a qu'une seule ressource pour l'école maternelle pour le filtre *programmation*.

Enfin, la CodeWeek Europe[59] ou Semaine européenne du code est aussi présente dans la section *Pratiques pédagogiques*. D'après Eduscol, il s'agit d'« une initiative citoyenne soutenue par la commission européenne qui encourage l'apprentissage de la programmation informatique »[60]. L'objectif de la CodeWeek présenté sur le site est de faire découvrir la programmation aux élèves lors de cet événement qui dure une semaine et qui est organisée une fois par an, au mois d'octobre. Il existe deux manières de participer à cet événement. Les enseignants peuvent soit organiser cet événement et le partager à l'aide de la carte interactive

57 https://edubase.eduscol.education.fr/
58 https://eduscol.education.fr/162/edubase
59 https://codeweek.eu/
60 https://eduscol.education.fr/1446/la-codeweek-europe

de la CodeWeek que les intéressés consultent pour s'informer des événements organisés, soit participer à un événement déjà organisé. Les enseignants sont aussi invités à suivre un MOOC en anglais[61] (ce qui est probablement d'une efficacité assez limitée en France), organisé par European Schoolnet, afin de se préparer à participer à la CodeWeek avec leur classe.

Dans la section *Ressources numériques,* une seule rubrique est dédiée à l'enseignement de l'informatique et surtout de la programmation. La rubrique s'intitule *Premières activités de programmation pour comprendre le numérique*[62]. Il s'agit d'un parcours de formation[63] créé en 2018 par les équipes Class'Code[64] et D-Clics numériques[65] pour aider les enseignants à initier leurs élèves à la programmation, à la pensée informatique et à la culture numérique. Ce parcours de formation est disponible via la plateforme M@gistère[66]. Il est destiné aux enseignants des cycles 2 et 3 et il dure quatre heures, à distance. La dernière mise à jour[67] de ce parcours de formation remonte à 2020, mais ce dernier reste toujours disponible en ligne sur la plateforme M@gistère.

La section *Culture Numérique* comprend pour sa part cinq rubriques, dont une seule dédiée à l'enseignement de l'informatique et de la programmation en particulier. Il s'agit de la rubrique *Programmation et Culture Numérique*[68]. Cette rubrique est consacrée à la présentation des projets et des actions qui ont été mis en place, avec le soutien du ministère, afin d'aider les enseignants à développer chez leurs élèves des compétences en programmation. Il est également précisé sur le site d'Eduscol que ces initiatives se déroulent souvent dans le cadre du périscolaire. Les projets présentés sont : Class'Code, D-Clics numériques, Code-Decode, Capprio, Declick et Vittascience.

Class'Code[69] est un projet de formation lancé en 2016. Il s'adresse aux acteurs de l'éducation (enseignants, animateurs, éducateurs), qui souhaitent initier leurs élèves à la programmation et à la pensée informatique. Il s'agit d'un projet financé par l'État à travers le Programme d'Investissements d'Avenir (PIA). Il propose des ressources éducatives, des parcours de

61 On peut d'interroger sur l'efficacité pratique en France d'un tel dispositif.
62 https://eduscol.education.fr/205/premieres-activites-de-programmation-pour-comprendre-le-numerique
63 https://magistere.education.fr/local/magistere_offers/index.php#offer=83
64 https://project.inria.fr/classcode/
65 http://www.d-clicsnumeriques.org/
66 Il s'agit d'une plateforme de formation continue mise en place par le ministère de l'éducation nationale. Lien vers la plateforme : https://magistere.education.fr/
67 Date de consultation du site : le 24 janvier 2023.
68 https://eduscol.education.fr/1824/programmation-et-culture-numerique
69 https://pixees.fr/classcode-v2

formation, mais aussi des rencontres avec d'autres acteurs de l'éducation. Class'Code offre aussi différents modules de formation qui sont disponibles sur les plateformes de ses partenaires.

Nous avons déjà mentionné le parcours de formation disponible sur la plateforme institutionnelle M@gistère, conçu pour permettre aux enseignants une première initiation à la programmation. Les parcours originaux de Class'Code sont disponibles sur la plateforme OpenClassrooms[70]. Deux cours sont aussi accessibles à travers la plateforme FUN-MOOC : *Se former pour l'ICN Informatique et Création Numérique*[71] et Class'Code IAI[72]. L'Inria, l'Institut national de recherche en sciences et technologies du numérique, pilote ce projet.

D-Clics numériques[73] est un autre projet lancé en 2015. Son chef de file est la Ligue de l'Enseignement[74], et il est destiné aux citoyens qui souhaitent se former aux usages du numérique. Plus précisément, ce projet s'adresse aux enfants de 8 à 14 ans, mais aussi aux enseignants et animateurs des collectivités et associations chargés des activités du périscolaire. Les personnes intéressées, en s'inscrivant à ce projet, pouvaient avoir accès à des parcours éducatifs et activités originales. Ce projet a pris fin en 2018, mais grâce à lui, la Ligue de l'enseignement a pu créer un réseau de plus de 300 référents numériques et a formé près de 10 000 acteurs éducatifs.

Code-Decode[75] est un projet qui donne la possibilité aux élèves de l'école primaire et du collège de se familiariser avec la programmation, la litteratie numérique et la culture du code. Les enfants du primaire participent à ce projet dans le cadre d'activités périscolaires, tandis que ceux du collège peuvent y participer au cours du temps scolaire. Le projet en question est porté par la Bibliothèque sans frontière.

Capprio[76] est un projet pilote mis en place par Simplon.co, dont l'objectif est de former les jeunes décrocheurs au numérique en les aidant à développer les compétences nécessaires pour pouvoir ensuite envisager une carrière professionnelle dans ce domaine. Ce projet est lancé en

70 https://openclassrooms.com/fr/partners/class-code
71 https://www.fun-mooc.fr/fr/cours/se-former-pour-licn-informatique-et-creation-numerique/
72 https://www.fun-mooc.fr/fr/cours/lintelligence-artificielle-avec-intelligence/
73 https://numerique.laligue.org/projets/d-clics-numeriques
74 https://laligue.org/
75 https://www.code-decode.net/
76 https://simplon.co/blog/actualites/retour-sur-capprio-le-programme-pilote-a-l-attention-des-jeunes-eloignes-de-l-emploi.html

2013 et en trois ans il a pu former, d'après les données disponibles sur leur site, 167 jeunes en Île-de-France, dont 40 % de femmes.

Declick[77] est un projet soutenu par le dispositif Edu-up[78] qui permet l'initiation des élèves de 8 à 15 ans à la programmation, grâce à un langage de programmation textuel. Declick se présente comme un dispositif intermédiaire entre les logiciels de programmation visuels, tel que Scratch, et les « vrais » environnements de programmation. Il s'agit d'un logiciel gratuit. En allant sur le site de Déclick, les élèves peuvent consulter des ressources afin de se familiariser progressivement avec des notions de programmation, créer leurs propres jeux et applications et même utiliser des jeux ou des applications déjà existantes dans la communauté de Declick. Ce projet est porté par l'association Colombbus en partenariat avec France IOI.

Vittascience[79] est le dernier projet présenté dans la rubrique *Programmation et Culture Numérique* d'Eduscol. Il s'agit aussi d'une plate-forme en ligne, soutenue par le dispositif Edu-up, qui vise à initier les élèves de collège et de lycée à la programmation en leur donnant la possibilité de choisir, entre 10 interfaces différentes, celle qui leur convient le mieux en fonction de leur niveau de maîtrise. Les élèves peuvent donc programmer à l'aide de blocks, en mode hybride avec une compilation, en temps réel, du langage en blocks en langage textuel ou même directement en langage textuel. La plateforme offre aussi une bibliothèque de ressources gratuites et sous licence libre. Les élèves peuvent, comme sur Scratch, sauvegarder leurs projets sur le cloud de la plateforme ou sur leur propre ordinateur. La version premium de la plateforme propose également aux professeurs des écoles l'espace « Classe », où ils peuvent créer des activités, les attribuer à leurs élèves individuellement ou en groupe et évaluer leurs résultats et même les commenter. Un tableau de bord qui permet aux enseignants de suivre la progression de leurs élèves est aussi disponible sur Vittascience premium.

Le réseau Canopé[80] est un opérateur du ministère de l'Éducation nationale, chargé de la formation continue des enseignants. Ce réseau aide les enseignants à l'appropriation des outils et des environnements numériques. Selon la mission qui lui a été confiée par Jean-Michel Blanquer en 2020, le réseau s'occupe de la formation des enseignants au numérique et par le

77 https://declick.net.
78 Edu-up c'est un dispositif créé par la direction du numérique pour l'éducation pour soutenir la production des ressources numériques pour l'école. Pour plus d'informations voir : https://eduscol.education.fr/1603/le-dispositif-edu
79 https://fr.vittascience.com/code
80 https://www.reseau-canope.fr/qui-sommes-nous.html

numérique. Canopé propose « une offre nationale de ressources et de formations – en présentiel et à distance – permettant aux enseignants de développer leurs compétences, leurs savoirs et pratiques professionnelles »[81]. En consultant[82] le site de Canopé, nous avons pu repérer six formations proposées aux enseignants du premier degré, en lien avec l'initiation à la programmation. Les formations proposées pour la maternelle et l'école élémentaire sont les suivantes : *Initiation à l'algorithmique avec Bee-bot et Blue-bot en classe ou à distance*[83], *Programmer un robot*[84], *Robotique et programmation*[85]. Les formations qui ne sont proposées qu'en école élémentaire sont celles qui suivent : *Codage déconnecté : le code sans écran*[86], *Des robots pour s'initier à la programmation*[87] et *Initiation à la programmation visuelle avec le robot Thymio en classe ou à distance*[88].

En recherchant des ressources et des formations disponibles sur le site de Canopé en lien avec l'enseignement de la programmation, nous n'avons pas pu trouver de lien vers le projet Créatice de Canopé[89], des académies de Créteil, de Paris et de Versailles. Ce projet lancé en 2011, permet pourtant aux enseignants d'emprunter du matériel auprès des Ateliers Canopé Île-de-France, sous la condition d'un abonnement, pour mettre en place des séquences pédagogiques avec leurs élèves. L'enseignant a même la possibilité de demander de l'aide sur la manipulation du matériel lors du rendez-vous d'enlèvement de celui-ci.

Le site de Créatice regroupe la liste du matériel numérique disponible en prêt (par exemple des tapis robotiques, robots et automates), des retours d'usage de ces matériels par les enseignants (par exemple des robots programmables ou des tablettes tactiles pour la programmation sur ScratchJr), ainsi qu'une liste de tutoriels. Ces retours d'usage rédigés par les enseignants pourraient servir au titre de formation par les pairs et ainsi orienter la pratique de leurs collègues.

81 Mission Canopé disponible ici : https://www.reseau-canope.fr/fileadmin/user_upload/espace_institutionnel/MISSION2021.pdf.
82 Date de consultation du site : le 25 avril 2022.
83 https://www.reseau-canope.fr/service/initiation-a-lalgorithmique-avec-beebot-et-bluebot-en-classe-ou-a-distance.html
84 https://www.reseau-canope.fr/service/programmer-un-robot.html
85 https://www.reseau-canope.fr/service/robotique-et-programmation_24631.html
86 https://www.reseau-canope.fr/service/codage-deconnecte-le-code-sans-ecran.html
87 https://www.reseau-canope.fr/service/des-robots-pour-sinitier-a-la-programmation.html
88 https://www.reseau-canope.fr/service/initiation-a-la-programmation-visuelle-avec-le-robot-thymio-en-classe-ou-a-distance.html
89 http://www.reseau-canope.fr/creatice/

Après avoir étudié l'offre des ressources et des formations disponibles sur le site d'Eduscol en lien avec l'enseignement de l'informatique et en particulier la programmation, nous pouvons constater qu'il n'existe pas de plan national de formation des enseignants en informatique et en particulier en programmation, comme c'était pourtant la recommandation de l'Académie des Sciences. En effet, aucune de ces recommandations n'a été prise en compte par l'État. Le coût de leur mise en place peut en être une explication plausible. La plupart des ressources proposées sont des activités portant sur l'apprentissage de la programmation à l'aide de robots programmables ou de logiciels de programmation visuels. Ces activités sont mises en place par d'autres enseignants, probablement autoformés. Nous n'avons pas pu repérer de formations ou de ressources mettant en évidence les liens possibles entre les contenus enseignés actuellement par les enseignants et l'informatique.

Quant au besoin des enseignants concernant la maîtrise conceptuelle de l'informatique, à part le parcours de formation *Premières activités de programmation pour comprendre le numérique* disponible sur M@gistère (d'une durée de quatre heures à distance), nous n'avons trouvé que peu de ressources traitant de cette question sur les différentes bases de données disponibles sur Eduscol. Il semble que l'État a consacré le budget réservé à cet enseignement en faisant principalement appel au secteur privé pour financer à la fois des projets de formation en ligne pour les enseignants et des projets destinés à la familiarisation des élèves avec la programmation, déléguant ainsi l'enseignement de la programmation de l'école primaire au périscolaire.

En Grèce, la formation des enseignants en informatique est considérée comme une priorité de la part du ministère de l'Education Nationale. En effet, il existe un plan national de formation en informatique pour les enseignants du premier et du second degré. Ce plan de formation est composé de deux niveaux, A et B. Un site[90] internet a été créé afin de regrouper toutes les informations nécessaires pour la formation des enseignants et faciliter le déroulement de ce processus. L'État promet aux enseignants souhaitant participer à cette formation des avantages en lien avec leur développement personnel et professionnel.

La formation de niveau A porte sur l'acquisition des compétences de base en informatique de la part des enseignants. Ce cycle de formation s'est déroulé entre 2000 et 2004. Son objectif a été de former les enseignants à l'utilisation de l'ordinateur, des logiciels de traitement texte,

90 https://e-pimorfosi.cti.gr/

de tableur, de présentation, ainsi que l'utilisation d'internet. Le contenu de cette formation est toujours disponible sur le site de formation des enseignants. Les enseignants qui n'avaient pas eu la possibilité de valider la formation de niveau A pendant la période de 2000 à 2004 ont eu la possibilité de la passer en mai 2022, car sa validation est nécessaire pour pouvoir suivre la formation de niveau B.

La formation de niveau B comprend deux sous-niveaux, B1 et B2. L'élaboration de ce niveau de formation est le résultat d'un projet co-financé par l'État et l'Union Européenne. Cette formation de trois heures hebdomadaires est réalisée par des formateurs spécialistes en didactique de l'informatique auprès d'enseignants répartis en groupes de 10 à 15 personnes. Elle est organisée en dehors du temps scolaire. Le niveau B1 correspond à l'initiation des enseignants à l'utilisation des technologies numériques dans le cadre de leurs pratiques d'enseignement. Il s'agit d'une formation d'une durée de 36 heures. Tous les enseignants doivent savoir utiliser trois logiciels de base : traitement texte, pdf et tableur. Les enseignants du premier degré[91] s'initient aussi à trois autres logiciels : CmapTools [92], TuxPaint [93] et Java [94].

Le niveau B2 correspond à une formation plus avancée, lors de laquelle les enseignants sont censés se familiariser avec une série des logiciels et concevoir des scénarios pédagogiques en les utilisant. Il s'agit d'une formation d'une durée de 42 heures. Nous retrouvons les trois logiciels de base (traitement texte, pdf et tableur) du niveau B1. Les enseignants, et en particulier ceux du premier degré, peuvent s'initier à l'utilisation d'autres logiciels en fonction de la discipline concernée. Dans le cadre de la discipline nommée *TIC et Informatique*[95] , celle qui nous intéresse le plus, ces enseignants ont la possibilité de se former à l'utilisation de logiciels de programmation tels que Scratch et ScratchJr, mais aussi à des robots programmables tels que Bee-Bot, Blue-Bot ou même Thymio.

La formation de niveau B2 inclut aussi la familiarisation avec des robots de type Lego Wedo. La particularité de ce dernier est que l'élève peut construire son robot avant de le programmer. Les enseignants du premier degré ont également l'opportunité de se familiariser avec la plateforme *Photodentro*[96]. Sur celle-ci, toujours dans le cadre de la discipline *TIC et*

91 Le premier degré en Grèce correspond à l'école maternelle (2 ans) et l'école primaire (6 ans).
92 http://cmap.ihmc.us/cmaptools/cmaptools-download/
93 https://tuxpaint.org/
94 https://java.com/en/
95 Notre traduction du grec : *ΤΠΕ και Πληροφορική*.
96 Photodentro est une plateforme qui a été créée dans le cadre de l'École Numérique(Ψηφιακό Σχολείο), afin de regrouper des séquences pédagogiques, pour tous les niveaux scolaires et toutes les disciplines,

Informatique, les enseignants du premier degré peuvent aussi trouver des environnements de programmation[97] inspirés du langage de programmation LOGO, qu'ils peuvent utiliser pour aider leurs élèves à résoudre des problèmes et ainsi développer des compétences en programmation.

Nous pouvons donc constater que l'État grec a créé une formation spécifique en informatique pour son corps enseignant afin de l'aider à mieux enseigner cette discipline, mais aussi à faciliter son usage dans le cadre des autres disciplines. À l'opposé de ce modèle, l'État français a choisi de ne pas créer de formation spécifique en informatique et de financer des initiatives privées, en déléguant ainsi au périscolaire l'enseignement de la programmation de l'école primaire.

2. Des projets de recherche récents (DALIE & IE-CARE)

Dans le cadre du regain d'intérêt international autour de la didactique de l'informatique et notamment de l'apprentissage de la programmation, des projets de recherche émergent dans le domaine des sciences de l'éducation pour étudier la question et essayer d'apporter des réponses utiles aussi bien pour les politiques éducatives que pour le corps enseignant. Nous présentons en particulier deux projets de recherche soutenus par l'Agence Nationale de la Recherche (ANR) en France, dans lesquels s'inscrit ce travail de thèse. Il s'agit de DALIE et IE-CARE.

DALIE[98] ou Didactique et apprentissage de 'informatique à l'école, est un projet de recherche associant quatorze chercheurs de six laboratoires de recherche en France et en Grèce. Les universités de Limoges, de Cergy-Pontoise, de Poitiers, de Paris Cité (ancien Paris-Descartes) et de Clermont-Ferrand constituent l'équipe française et l'université de Patras représente la partie grecque. Cette recherche a démarré en 2015 et s'est achevée en 2017. DALIE associait des chercheurs en sciences humaines et sociales.

disponibles en ligne à travers ce lien : http://photodentro.edu.gr/lor/.

97 Exemple d'environnement de programmation disponible sur la plateforme Photodentro disponible ici : http://photodentro.edu.gr/lor/r/8521/11286?locale=el.

98 https://www.unilim.fr/dalie/le-projet/

L'objectif central de cette recherche a été d'étudier ce qui pourrait être enseigné autour de l'informatique à l'école primaire, mais aussi les effets possibles sur les élèves et les compétences nécessaires pour que les enseignants puissent mettre en place un tel enseignement. L'algorithmique, la programmation et la pensée informatique sont au cœur de ce projet pilote. L'ambition du projet était :

> « de mieux comprendre comment enseigner les éléments permettant l'émergence d'une « culture informatique » à ce niveau[99], en tenant compte des contextes d'apprentissages et des curricula, à un moment où l'on ne peut – plus – séparer les usages d'instruments informatisés de l'appropriation de concepts spécifiques »[100].

L'objectif central présenté ci-dessus se décline en quatre sous-objectifs qui sont cités sur le site du projet :

> « Faire le point sur les représentations que les enseignants, les élèves ont de l'informatique. Voir s'il y a évolution des représentations de l'informatique des enseignants et des élèves suite à l'expérimentation : enseignement du type pédagogie par projet. Si c'est le cas, la mettre en évidence.
>
> Faire le point sur les représentations que les étudiants se préparant à l'enseignement primaire et leurs formateurs se font de la place à donner aux technologies informatisées à l'école.
>
> Produire des résultats de recherche en didactique de l'informatique à l'école primaire et relancer l'intérêt des chercheurs et des politiques via la diffusion large des résultats de ces travaux.
>
> Contribuer à l'amélioration des dispositifs de formation en informatique dans l'enseignement supérieur à destination des maîtres, par une meilleure connaissance des processus d'apprentissage et d'appropriation (construire un certain nombre de représentations adaptées et opératoires concernant l'informatique, savoir décrire et contextualiser le dispositif utilisé, apprendre à programmer en faisant faire à une machine…) »[101].

Nous avons assuré un double rôle au sein de ce projet de recherche. En effet, nous avons eu la possibilité de participer à la conception de l'onglet « formation » disponible sur la plateforme « e-SPACE », dédiée à la formation des enseignants. Nous nous sommes particulièrement chargée de la conception de la formation des enseignants sur le logiciel de programmation visuelle ScratchJr. Nous avons élaboré six courts scénarios pédagogiques, associant différents concepts de programmation différents (la séquence, la répétition, les messages, la synchronisation, etc.), pour que les enseignants aient un point de départ concernant la structure et les objectifs d'un scénario pédagogique créé pour enseigner la programmation sur ScratchJr. Nous avons aussi suivi avec Michel Spach les expérimentations en robotique

99 Note de bas de page introduite par nous, pour clarifier que les chercheurs se réfèrent ici à l'enseignement primaire.

100 Extrait de la présentation du projet DALIE disponible ici : https://www.unilim.fr/dalie/le-projet/.

101 Extrait de la présentation du projet DALIE disponible ici : https://www.unilim.fr/dalie/le-projet/.

pédagogique d'une classe de CE1, d'une école de la banlieue parisienne engagée dans le projet DALIE.

IE-CARE[102] ou Informatique à l'école : conceptualisations, accompagnement, ressources, est un projet de recherche associant vingt-quatre chercheurs de quatre laboratoires de recherche en France et en Grèce. Les universités de Lille, de Paris Cité (ancien Paris-Descartes) et Sorbonne Université constituent l'équipe française et l'université de Patras représente la partie grecque. Il s'agit d'une recherche qui a démarré en 2018 et s'achèvera en 2023.

IE-CARE est un projet pluridisciplinaire qui associe des chercheurs en informatique et en sciences humaines et sociales (sciences de l'éducation, didactiques, psychologie des apprentissages, sociologie). IE-CARE couvre l'ensemble de la scolarité obligatoire et a pour objectif de « produire des connaissances fondamentales et opératoires sur l'informatique, son enseignement et son apprentissage, à l'école obligatoire »[103]. Ce projet envisage également de contribuer « au développement d'une culture informatique et technologique partagée autant par les enseignants, les accompagnateurs que les superviseurs de l'action pédagogique »[104].

Cette recherche est organisée autour de trois axes qui sont reliés entre eux à l'aide d'une tâche transversale. Les axes sont les suivants :

> « Le premier vise à délimiter un ensemble de contenus informatiques enseignables.
>
> Le deuxième, lié au premier, concerne la conception, la modification et l'usage de scénarios pédagogiques et de ressources pour soutenir les pratiques d'enseignement et d'apprentissage de l'informatique à l'école et au collège.
>
> Le troisième axe se centre sur la mise en place d'un cadre d'accompagnement pour les enseignants et les formateurs en informatique »[105].

À travers ce travail de thèse, nous contribuons à apporter des réponses aux questions soulevées dans ce projet de recherche et en particulier à celles qui concernent les axes 1 et 2.

102 http://iecare.lip6.fr/
103 Extrait de la présentation du projet IE-CARE disponible ici : http://iecare.lip6.fr/.
104 Extrait de la présentation du projet IE-CARE disponible ici : http://iecare.lip6.fr/.
105 Extrait de la présentation du projet IE-CARE disponible sur le site d'ANR ici :https://anr.fr/Projet-ANR-18-CE38-0008.

3. D’où la nécessité d’étudier l’apprentissage de la programmation par les très jeunes élèves

Comme nous l’avons montré, il existe un intérêt grandissant autour de l’introduction de l’informatique en tant que discipline à part entière dans l’éducation obligatoire, et surtout de l’apprentissage de la programmation dès l’école primaire. Plusieurs pays dans le monde ont procédé à l’introduction de l’informatique et en particulier de la programmation dans leurs programmes scolaires (Heintz et al., 2016).

En France, l’informatique a été introduite sous une forme transdisciplinaire dans les programmes scolaires du primaire de la rentrée 2016 (Vandevelde & Fluckiger, 2020). L’activité de programmation a été intégrée dans l’enseignement des mathématiques dès le cycle 2, en lien avec le développement des compétences de repérage et de déplacement dans l’espace (BO n° 17 du 26 novembre 2015). En 2020, les programmes scolaires ont changé de nouveau, l’activité de programmation y est toujours, présente, mais beaucoup moins qu’en 2016 (BO n° 31 du 30 juillet 2020).

En Grèce, l’informatique a été introduite pour la première fois à l’école primaire en 1997 (Υπουργείο Εθνικής Παιδείας και Θρησκευμάτων. Παιδαγωγικό Ινστιτούτο, 1997) et à l’école maternelle en 2003 (ΔΕΠΠΣ Νηπιαγωγείο, 2003). L’activité de programmation fait partie des programmes scolaires du primaire depuis 2003 (ΔΕΠΠΣ Πληροφορικής, 2003) et de l’école maternelle depuis 2011 (Παιδαγωγικό Ινστιτούτο, 2011b). Les nouveaux programmes de 2021 pour le primaire mettent l’accent sur développement d’une pensée informatique chez les élèves (ΙΕΠ, 2021b). C’est la première fois que la programmation occupe une place aussi importante dans les programmes du primaire. La programmation fait aussi partie des programmes de la maternelle de 2021 (ΙΕΠ, 2021a). C’est aussi la première fois que des notions comme celles de séquence, de répétition et d’instruction conditionnelle font partie des programmes de maternelle en Grèce.

La plupart des pays qui ont procédé à l’introduction de l’informatique et de la programmation ont mis en place des démarches pour former leur corps enseignant, afin de pouvoir le soutenir dans la mise en place de ce nouvel enseignement. En France, il n’existe pas de plan national de formation des enseignants en informatique et en particulier en programmation, contrairement à la recommandation de l’Académie des Sciences (2013). Il semble que l’État

français mise davantage sur une formation via le secteur privé pour ses projets de formation en ligne des enseignants et de familiarisation des élèves à la programmation. Ainsi, on observe que l'enseignement de la programmation à l'école primaire se fait plutôt au niveau du périscolaire. En Grèce, la situation est différente. L'État grec a créé une formation spécifique en informatique pour son corps enseignant, afin de l'aider à mieux enseigner cette discipline, mais aussi afin de faciliter son usage dans le cadre d'autres disciplines. Ce plan de formation est composé de deux niveaux, A et B.

Dans le contexte du regain d'intérêt international pour l'enseignement de l'informatique et notamment de la programmation, nous avons pu repérer des projets de recherche comme DALIE et IE-CARE, qui ont émergé dans le domaine des sciences de l'éducation, afin d' étudier la question et de tenter d'apporter des réponses utiles aussi bien pour les politiques éducatives que pour le corps enseignant.

Plusieurs arguments ont été présentés pour justifier cet intérêt. Un argument souvent mentionné est le besoin de former de futurs employés compétents en informatique et notamment en programmation pour l'avenir de la société (Pila et al., 2019). L'importance d'une formation précoce à l'informatique et à la programmation est également un argument récurrent (Kazakoff, 2014). Cette formation pourrait être bénéfique pour tous les élèves et en particulier pour les filles qui ont tendance à perdre confiance en elles face à cette matière vers la fin du primaire et au début du collège (Margolis & Fisher, 2002). Enfin, on peut citer l'incapacité de la majorité des systèmes éducatifs actuels à préparer suffisamment les nouvelles générations à vivre et travailler dans un monde de plus en plus informatisé comme le nôtre (Kelly, 2012, 27 avril).

Les résultats des recherches présentées dans cette sous-partie, qui démontrent les compétences limitées et le manque de conceptualisation des jeunes en informatique, renforcent la validité de cet argument. Les défenseurs de cet enseignement soulignent aussi que pour devenir pleinement citoyen à l'heure actuelle, il faut être capable de comprendre le fonctionnement et les concepts sous-jacents aux dispositifs informatiques qui nous entourent, pour pouvoir ensuite mieux les manipuler (Bruillard, 2014). Ainsi, certains ont proposé de revoir la trilogie « lire-écrire-compter » et de lui ajouter des compétences en informatique, pour mieux l'adapter aux besoins de la société actuelle (Bruillard, 2012). L'enjeu essentiel derrière ce mouvement international a été de former les futurs citoyens en informatique et notamment en

programmation pour les émanciper et leur donner le pouvoir sur les machines, en leur permettant de les programmer au lieu d'être programmés par elles.

En plus de la visée citoyenneté, un autre argument fréquemment utilisé en faveur d'une initiation des jeunes élèves à l'informatique et à la programmation a été celui des bénéfices potentiels liés au développement d'une pensée informatique chez les jeunes élèves. Ce n'est pas la première fois que des arguments de transferabilité sont mobilisés en faveur de la discipline informatique (Drot-Delange et al., 2019). Les travaux des années 1980 sur l'apprentissage de la programmation pour les enfants tournaient aussi en partie autour de ce type d'arguments, affirmant que l'apprentissage de la programmation pouvait contribuer au développement de compétences utiles dans d'autres domaines. C'est pourquoi ces travaux ont aussi servi d'argument pour justifier l'intérêt nouveau pour l'enseignement de la programmation aux enfants. Papert et les réflexions qu'il développe autour de LOGO rencontrent un fort engouement. En effet, des nombreuses recherches traitant de l'apprentissage de la programmation par les enfants citaient et citent toujours ses travaux.

Les travaux de recherche menés au départ à l'aide de langages de programmation textuels comme LOGO ont montré que la syntaxe de ce type de langages posait souvent des problèmes aux élèves. Ce qui a changé ces dernières années, et constitue un moteur important de cet intérêt nouveau pour l'enseignement de la programmation aux enfants, est l'apparition d'une nouvelle forme de programmation à destination des élèves, la programmation visuelle ou graphique. Ainsi, la nature des langages a changé, et est désormais mieux adaptée aux besoins des élèves.

En effet, les langages de programmation graphiques n'ont pas de règles de syntaxe strictes, contrairement aux langages de programmation traditionnels. Les commandes sont des blocs graphiques dont la forme est telle qu'elle oblige l'utilisateur à les assembler correctement pour créer des programmes. Malgré leur simplification au niveau de la syntaxe, les langages graphiques contiennent souvent beaucoup de texte, ce qui compromet leur utilisation dans le milieu de la petite enfance.

Les langages graphiques ont donc dû évoluer davantage, pour être accessibles même aux plus jeunes élèves. Pour cela, le texte sur les blocks de programmation a été remplacé par des icônes, afin que ces langages puissent être utilisés même par les élèves les plus jeunes, qui ne savent pas encore lire. Cela a enrichi énormément les possibilités que nous avions pour

travailler avec des enfants des plus en plus jeunes, sur des notions de programmation peu étudiées auparavant pour un public de cet âge-là.

C'est pourquoi nous avons choisi de nous intéresser à l'apprentissage de la programmation par de très jeunes enfants à l'aide d'un langage de programmation graphique adapté à ce groupe d'âge. Ainsi, nous avons choisi de réaliser une revue de la littérature disponible sur l'apprentissage de la programmation par les enfants en nous focalisant sur ce qu'ils sont capables de faire en programmation et leurs difficultés, afin d'explorer ce qui a été déjà fait sur le sujet pour définir la problématique et les questionnements de cette recherche et réaliser les choix théoriques nous permettant l'étudier.

Partie 2. Cadre conceptuel et construction de la problématique

Chapitre 1. Apprentissage de la programmation par les enfants : première fondation

En s'appuyant sur une revue de la littérature disponible sur l'apprentissage de la programmation par les enfants, nous avons constaté que ce sujet a une longue histoire qui remonte à la seconde moitié des années 1960 et la création du premier langage de programmation pour les enfants, LOGO. Dans ce chapitre, nous présenterons les travaux réalisés sur ce sujet pendant cette période de première fondation. Dans un premier temps, nous mettons en avant une série des définitions que nous jugeons importantes pour comprendre la particularité et la complexité de l'activité de programmation. Par la suite, nous présenterons l'environnement de programmation LOGO et les principales idées du groupe de chercheurs qui l'a inventé et notamment de Seymour Papert, le père du constructionnisme. Puis, nous exposons les résultats de travaux de recherche réalisés avec LOGO ayant pour objectif d'étudier la programmation en tant qu'objet d'apprentissage pour les enfants et de façon plus courte les travaux réalisés ayant pour objectif d'étudier les effets que, l'apprentissage de la programmation par les élèves, pourrait avoir sur d'autres domaines. La dernière section de ce chapitre portera sur les prérequis nécessaires pour l'apprentissage de la programmation par les enfants.

1. L'activité de programmation

La programmation est reconnue comme une compétence nécessaire pour le XXIème siècle. Il s'agit d'une activité qui permet d'écrire, dans un langage de programmation précis, une série d'actions à réaliser, pour atteindre un objectif visé. En d'autres mots, il s'agit d'écrire un algorithme, en utilisant un langage de programmation, pour le communiquer à une machine et lui dire de réaliser une tâche spécifique (Dufoyer, 1988). Arsac (1988) note que « programmer n'est pas de résoudre un problème, c'est le faire résoudre par une machine »(p. 14). Nous sommes donc obligés de fournir une méthode de résolution de problème à la machine (Arsac, 1988).

De part son originalité par rapport à d'autres activités, la programmation a suscité depuis son apparition un grand intérêt auprès des psychologues et des pédagogues (Dufoyer, 1988). Cette activité « appartient à la plus vaste catégorie de tâches que les psychologues appellent résolution de problème »(Dufoyer, 1988, p. 46). Les psychologues s'intéressaient notamment à l'apprentissage de la programmation, aux processus cognitifs associés à cette activité, aux stades qui la composent, aux aptitudes nécessaires pour son apprentissage et à ses effets cognitifs. Les pédagogues trouvent que la pratique de la programmation et surtout la phase d'analyse d'un problème avant de passer à l'écriture d'un programme, peuvent donner lieu à l'enseignement d'une pensée logique aux élèves dans le cadre des situations familières (Dufoyer, 1988). Un autre élément qui rend l'activité de programmation intéressante est le fait qu'elle exige des types de réflexion qui sont rarement exercés dans d'autres domaines (Dalbey & Linn, 1985).

Bien que la programmation soit souvent évaluée en tant qu'activité, elle est composée de phases (Clements, 1999). Pea et Kurland (1984) expliquent qu'il s'agit des phases de résolution d'un problème. George Polya (1965) en décrit quatre : comprendre le problème, concevoir un plan, exécution du plan, revenir sur la solution. D'après lui, il faut d'abord comprendre le problème et identifier les éléments que nous avons à notre disposition pour pouvoir le résoudre. Ensuite, il faut réfléchir aux actions que nous mettrons en place pour le résoudre, avoir donc l'idée d'un plan pour le résoudre. La phase qui suit est celle de l'exécution, où nous allons mettre en oeuvre notre plan. La dernière phase est celle du contrôle de la solution, lors de laquelle nous allons revenir à notre solution pour vérifier son exactitude. Il est possible que la solution contient des erreurs. Dans ce cas-là, il faut procéder à la détection de ces erreurs et à leur correction (Polya, 1965).

Pea et Kurland (1984)expliquent que ces quatre phases de résolution d'un problème s'appliquent aussi à la résolution d'un problème en programmation. Ils citent les phases suivantes : comprendre le problème de programmation, planifier une solution pour le résoudre, écrire le programme mettant en œuvre ce plan, comprendre et déboguer le programme écrit. Dufoyer (1988, p. 93) détaille un peu plus les quatre phases nécessaires pour créer un programme :

> « être en mesure de comprendre le problème qui est posé et fait l'objet du programme à écrire (c'est un minimum !) ;

être capable de déterminer, puis de planifier la méthode de résolution avec description de l'algorithme ;

une fois cet algorithme établi, savoir le traduire dans un code de programmation écrit ;

se trouver en mesure de rechercher les erreurs dans le programme qu'on a écrit et déterminer les solutions correspondantes ».

Pour programmer nous avons besoin d'utiliser un langage de programmation, car les machines ne peuvent pas exécuter des algorithmes écrits en langage naturel (Dowek, 2011). L'auteur explique, dans cet article qu'il a écrit sur les quatre concepts de l'informatique, que la notion de programme se trouve à la frontière des notions d'algorithme et de langage. Un langage de programmation est : « un ensemble de commandes et de règles de combinaison des commandes, qui sont utilisées pour commander l'ordinateur à réaliser des opérations spécifiées »[106] (Pea & Kurland, 1984, p. 145).

Pour pouvoir écrire un programme dans un langage de programmation choisi, il faut connaître les règles d'usage de ce langage (Dalbey & Linn, 1985). Comme toute langue naturelle, un langage de programmation a un lexique, une syntaxe et une sémantique spécifique (M. Beaudouin-Lafon, cours magistral, 1996)[107]. Par contre, les langages de programmation sont plus simples et plus précis que les langues naturelles (Dowek, 2011). Le lexique d'un langage de programmation comprend ses éléments de base, les « mots » de son vocabulaire(M. Beaudouin-Lafon, cours magistral, 1996). La syntaxe d'un langage de programmation est la grammaire du langage c'est-à-dire les règles de composition de ses éléments de base en programmes.

Bien que tous les langages de programmation ont les mêmes composantes syntaxiques, le contenu de ces composantes est différent pour chaque langage (Fay & Mayer, 1988).Pour pouvoir programmer il est nécessaire d'écrire des commandes syntaxiquement correctes sinon l'ordinateur ne pourra pas les exécuter (Fay et Mayer, 1988).

La sémantique d'un langage de programmation est plus difficile à comprendre que sa syntaxe (Ben-Ari, 1996). D'après Fay et Mayer(1988), la sémantique des commandes d'un langage de programmation peut être décrite à l'aide des trois composantes : les opérations, les objets et les emplacements. Les opérations sont les actions qui peuvent être réalisées, les objets sont les

106 Notre traduction de l'anglais : A programming language is a set of coommands and rules for command combinations that are used to instruct the computer to perform specified operations.

107 Les notes de ce cours sont disponibles ici :https://www.lri.fr/~mbl/ENS/DEUG/cours/3-lexique-syntaxe.html

entités sur lesquelles sont réalisées ces actions et les emplacements sont les lieux où se passent les actions. Bien que tous les langages de programmation puissent être analysés en termes de ces trois composantes, ils peuvent se différencier en ce qui concerne les opérations, les objets ou les emplacements qu'ils supportent, c'est-à-dire que le contenu de ces composantes peut être différent en fonction du langage de programmation concerné (Fay & Mayer, 1988). Il est également important de connaître la sémantique des commandes d'un langage de programmation, pour pouvoir réfléchir à l'action que ces commandes produiront quand elles seront exécutées (Fay et Mayer, 1988).

Après avoir écrit un programme en respectant les règles d'un langage précis, il faut vérifier son exactitude. Pour le faire, il est nécessaire de passer par le processus du débogage. Cela sert à détecter la présence d'un écart, entre le résultat souhaité et le résultat obtenu suite à l'exécution du programme que nous avons créé, pour ensuite repérer l'erreur provoquant cet écart et la corriger (Klahr & Carver, 1988).

Le débogage est une phase très importante pour la résolution des problèmes en programmation, car des erreurs apparaissent très souvent et il est nécessaire de les surmonter pour arriver à l'objectif visé. Bonar et Soloway (1985) ont étudié ce qu'il se produit lorsqu'une erreur est commise. Les auteurs expliquent que très souvent une partie du problème n'est pas comprise par celui qui essaie de le résoudre et en sachant pas quoi faire, il réalise une action pour surmonter cette partie et de cette manière une erreur ou *bug* en anglais est née (Bonar & Soloway, 1985). La détection et la correction des erreurs sur les programmes sont des entreprises difficiles et chronophages (Dalbey & Linn, 1985).

Khlar et Carver (1988) définissent le débogage comme un processus qui comprend cinq phases : 1) Evaluation du programme, 2) Identification d'erreur, 3) Représentation du programme, 4) Localisation de l'erreur et 5) Correction de l'erreur.

Au cours de la première phase, l'utilisateur exécute le programme et vérifie s'il existe une différence entre le résultat souhaité et le résultat obtenu suite à l'exécution du programme. Si une telle différence existe, il faudra procéder aux phases suivantes. Au cours de la deuxième phase, l'utilisateur essaie de comprendre en quoi le résultat obtenu suite à l'exécution du programme diffère du résultat souhaité. Pendant la troisième phase, l'utilisateur représente la structure du programme pour trouver quelle est la commande qui pose potentiellement problème dans celui-ci. Pendant la quatrième phase de ce processus, l'utilisateur porte une

attention particulière à son programme pour pouvoir localiser l'erreur. Enfin, l'utilisateur procède à la dernière phase où il corrige le programme en remplaçant l'erreur par sa correction et puis, il re-exécute son programme pour vérifier son exactitude.

Nous pouvons donc constater que la programmation est une activité assez complexe, comprenant plusieurs phases différentes et nécessitant des connaissances spécifiques en lien avec le langage de programmation utilisé. De part sa complexité, l'activité de programmation a été longtemps réservée aux informaticiens (Bruillard, 2012), surtout du moment où les langages de programmation existants au début ne permettaient pas une ouverture possible vers le monde éducatif. Une telle ouverture a été rendue possible dès la seconde moitié des années 1960 grâce à la création de LOGO, le premier langage de programmation pour les enfants.

2. Un point de départ : LOGO

Le langage de programmation LOGO a été créé en 1966 par Seymour Papert, Wallace Feurzeig, Daniel Bobrow, Richard Grand, Cynthia Solomon et Franck Frazier au Département de la Technologie Éducative de la firme de recherche de Cambridge Bolt Beranek et Newman[108] (Feurzeig, 2010). L'équipe de chercheurs derrière LOGO avait repéré que les mathématiques posaient souvent problème aux élèves à l'école et ils ont formulé l'hypothèse que cela est dû à la façon dont cette matière est enseignée (Feurzeig et al., 1969).

Pour remédier à ce problème, ils ont réfléchi à la création d'un environnement où les élèves pourraient faire des mathématiques en ayant conscience qu'il fallait qu'ils soient précis dans leur façon de s'exprimer (Feurzeig & Papert, 2011). L'utilisation d'un langage de programmation adapté à l'âge des élèves et la mise en place d'un enseignement appropriée, pourraient être la réponse à cela.

Ayant travaillé cinq ans avec Piaget, Papert (1989), un de leaders du projet LOGO, était particulièrement impressionné par deux principes du constructivisme : le fait que l'élève apprend mieux lorsqu'il est actif et lorsqu'il réfléchit sur ce qu'il fait. En s'appuyant sur ces deux principes, Papert et ses collègues ont créé le langage de programmation LOGO, pour permettre aux élèves de travailler sur des notions de mathématiques, de façon beaucoup plus concrète (Feurzeig & Papert, 2011). Feurzeig(2010) explique que leur objectif en créant

108BBN Technologies.

LOGO a été de permettre aux élèves d'apprendre à mieux réfléchir et à être actifs dans la construction de leurs savoirs.

Le langage LOGO a pris son nom du mot grec *λόγος* (Feurzeig, 2010), qui a plusieurs significations dont les plus adaptées dans ce contexte sont les suivantes : la capacité à communiquer à l'aide d'un langage, les paroles, la logique ou même la réflexion. Pour sa création, ils ont pris en compte trois critères. Le langage devrait : être facilement utilisé par des élèves de 8 à 9 ans pour réaliser des taches simples, prendre en compte des concepts mathématiques importants en impliquant le moins possible les règles de programmation et permettre l'expression des algorithmes numériques et non numériques (Feurzeig et Papert, 2011). LOGO est un dialecte de LISP, parce qu'ils se sont basés sur ce dernier pour créer le nouveau langage (Feurzeig, 2010). Les liens forts existants entre les mathématiques et la programmation, ont rendu les mathématiques, le contexte le plus adéquat, pour développer des séances d'enseignement en LOGO (Feurzeig, 2010).

L'équipe de chercheurs derrière le projet LOGO a choisi d'utiliser la programmation pour enseigner les mathématiques, car ils ont réfléchi à des avantages que cette activité pourrait avoir pour les apprentissages des élèves (Feurzeig et Papert, 2011). Un de plus importants avantages de cette activité est le fait que l'élève a le pouvoir sur la machine (Papert, 1980). En effet, dans l'environnement LOGO il contrôle l'ordinateur en utilisant un langage de programmation formel pour le faire réaliser une tache spécifique et ainsi, il apprend des concepts mathématiques, qui dans d'autres conditions pourraient lui sembler abstraits et par conséquent plus difficiles à comprendre (Feurzeig et Papert, 2011). Papert (1980) pense que le fait de faire une machine faire quelque chose, donne la possibilité aux élèves de penser sur leur propre pensée et cela est une expérience bénéfique pour eux. De plus, le fait de travailler avec l'ordinateur et le langage de programmation formel dans l'environnement de LOGO aide les élèves à avoir conscience de ce qu'ils font pour résoudre un problème et à pouvoir s'exprimer autour de cela, exercice qu'ils ne font pas très souvent dans le contexte scolaire (Feurzeig et Papert, 2011).

Un autre avantage que l'équipe des chercheurs derrière LOGO souligne est la possibilité que cette activité donne aux élèves de concevoir un plan avant de résoudre un problème (Feurzeig et Papert, 2011). Elle leur permet aussi de comprendre que, pour résoudre un problème il est souvent nécessaire de passer par la résolution de sous-problèmes (Feurzeig et Papert, 2011).

Papert (1980) souligne que l'activité de programmation aide également les élèves à reconnaître la valeur de l'erreur. Dans le contexte scolaire, les élèves ont très souvent peur de se tromper. Dans le contexte de programmation en revanche, ils sont très souvent confrontés à l'erreur, car il est très rare d'écrire un programme qui est correct dès la première exécution (Papert, 1980). Un programme qui ne fait pas exactement ce qu'il était censé faire est un objet utile pour l'apprentissage car nous pouvons l'étudier pour comprendre comment on pourrait le corriger. Pour cela, le processus de débogage est essentiel (Feurzeig et Papert, 2011).

D'après les concepteurs de LOGO, « l'activité de programmation favorise donc une approche expérimentale de la résolution de problèmes » (Feurzeig et Papert, 2011, p.490). Ils ont aussi formulé le propos que la programmation sur LOGO pourrait avoir des effets sur le développement des capacités méta-cognitives des élèves comme la compréhension, la gestion du temps, la réflectivité etc (Feurzeig, 2010).

En 1970, Seymour Papert crée le Laboratoire LOGO à MIT (Feurzeig, 2010). La tortue de plancher a été créée en 1971 et la tortue d'écran en 1972. Cette même année la première tortue de plancher sans fil, Irving, a été créée (Feurzeig, 2010). Dans l'environnement LOGO, les élèves peuvent utiliser le langage de programmation LOGO pour communiquer avec la tortue (Papert, 1980). Celle-ci est un mobile qui peut avancer, reculer, tourner à droite ou à gauche, en laissant ou pas une trace sur le plancher. Ce mobile doit être connecté avec l'ordinateur à travers un fil. La tortue d'écran est un curseur triangulaire qui peut se déplacer sur l'écran en laissant une trace qui représente le chemin réalisé (Papert, 1980). Par exemple l'utilisateur peut commander la tortue à dessiner un carré ou une fleur. Les élèves peuvent commencer à s'expérimenter avec la tortue de plancher et puis passer à la tortue d'écran (Papert, 1980).

Papert a créé la tortue, « un animal cybernétique assisté par ordinateur » (Papert, 1980, p. 11), afin de mettre à disposition des élèves un « objet-pour-penser-avec »[notre traduction](Papert, 1980, p.11), qui leur permettait de construire une culture scientifique, à partir de leurs interactions avec celui-ci. Il a associé la tortue à LOGO, pour permettre à des enfants plus jeunes de faire la programmation (Dufoyer, 1988). Pour Papert (1980), la tortue est comme un être humain, car elle a une position et une direction précise et c'est grâce à ces caractéristiques que les élèves peuvent s'identifier à celle-ci et faire appel aux connaissances qu'ils ont de leur corps et son déplacement dans l'espace, pour apprendre la géométrie. Lorsque l'élève

programme la tortue pour réaliser un chemin précis, il doit d'abord réfléchir à la manière dont il ferait le chemin s'il était à sa place (Papert, 1980).

En 1980, Papert publie le livre *Mindstorms : Children, Computers and Powerful Ideas.* L'apparition de ce dernier a marqué les idées de l'époque par rapport à ce qu'il était possible de faire en programmation avec les enfants. Pour Papert (1980), « programmer un ordinateur c'est communiquer avec lui dans un langage, compréhensible aussi bien par celui qui le programme que par l'ordinateur »[notre traduction](Papert, 1980, p. 6). Puisque les élèves apprennent facilement des nouveaux langages, Papert (1980) pensait qu'ils devraient aussi pouvoir communiquer avec l'ordinateur. Il était également le premier à évoquer que les élèves de l'école élémentaire sont capables à programmer dans l'environnement de programmation LOGO et à considérer que cela semblait aussi possible pour les élèves les plus jeunes.

Dans ce livre Papert (1980), revendique le constructionnisme en résumant ses travaux autour de l'utilisation de l'ordinateur en tant qu'instrument permettant aux élèves de construire des « objets-pour-penser-avec »[notre traduction](p.11). Il explique, qu'il s'appuie sur le propos soutenu par Piaget selon lequel les élèves construisent eux-mêmes spontanément leurs propres connaissances, mais il diffère du savant genevois, car il met davantage que lui l'accent sur le rôle que jouent les matériaux de l'environnement dans cette construction.

Il met également en évidence l'importance de l'apprentissage par la découverte ou *discovery learning,* un des points fort de l'environnement de LOGO (Papert, 1980). D'après cet auteur, les élèves apprennent mieux lorsqu'ils apprennent tous seuls, loin des situations didactiques organisées en utilisant les « objets-pour-penser-avec ».

Il considère que lorsqu'un concept semble difficile à comprendre par l'élève, cela veut dire que ce dernier n'a pas été exposé aux matériaux nécessaires, qui auraient pu le rendre plus concret et par conséquent plus simple à comprendre. Il souligne également que les matériaux pourraient même influencer l'ordre selon lequel les élèves développent certaines capacités cognitives. De manière plus précise, il relève que l'ordinateur pourrait, en tant que matériel, avoir une influence sur les apprentissages des élèves et remettre ainsi en question certaines idées de la psychologie développementale, par rapport à ce que nous pouvons faire à chaque tranche d'âge. Pour lui, « l'ordinateur peut rendre concret le formel »[notre traduction] (Papert, 1980, p.21). L'enjeu pour lui a été de profiter de la nouveauté de ces instruments informatiques, pour améliorer l'enseignement traditionnel.

Papert (1980) explique aussi dans cet ouvrage l'idée de micromonde. Pour lui, c'est un environnement permettant à l'élève de construire de façon active ses connaissances en interagissant avec l'ordinateur. Il s'agit d'un environnement « à explorer et à manipuler »[notre traduction](Papert, 1980, p.129). La Tortue géométrique et la dynatortue sont des exemples de micromondes, faisant partie de l'environnement de LOGO, que Papert a créé pour permettre à l'élève de construire, selon un chemin d'apprentissage piagétien, ses connaissances autour de la géométrie formelle et les lois de Newton. Pour lui, un micromonde est aussi un environnement où l'erreur a une place importante, car elle est utile dans le processus de construction des connaissances des élèves.

Papert (1980) cite enfin trois caractéristiques qui rendent le micromonde particulièrement utile en tant qu'environnement d'apprentissage. La première est le fait qu'il donne la possibilité à l'élève d'avoir une expérience directe des notions à acquérir, en le mettant dans un contexte où il a vraiment besoin de les utiliser pour réaliser une tache, qui a du sens pour lui. La deuxième caractéristique est le fait qu'il donne à l'élève les moyens pour conceptualiser le monde. En d'autres mots, un micromonde permet, à travers la programmation d'un objet sur un écran ou d'un objet sur le sol, de passer de quelque chose d'abstrait à quelque chose de concret. La dernière caractéristique est le fait qu'il donne à l'élève la possibilité de vraiment s'approprier ce qu'il construit, de lui donner la forme qu'il lui plait.

Après l'apparition de l'ouvrage *Mindstorms : Children, Computers and Powerful Ideas* de Seymour Papert au début des années 1980, un grand nombre des travaux de recherche ont été menés utilisant le langage de programmation LOGO. Ces travaux avaient pour objectif d'enseigner la programmation aux enfants, ainsi que d'utiliser la programmation en tant qu'outil pour développer des compétences dans d'autres domaines (Mendelsohn et al., 1990). Baron (1987) nous informe que les premières expérimentations avec LOGO en France ont été réalisées vers la fin des années 1970 et ensuite, le langage est devenu très populaire surtout à l'école primaire.

3. La programmation comme objet d'apprentissage pour les enfants : les travaux réalisés sur LOGO

Dans cette section, nous allons nous concentrer aux travaux réalisés avec LOGO ayant pour objectif d'étudier la programmation en tant qu'objet d'apprentissage pour les enfants. Cela nous aidera à prendre du recul du débat actuel autour de l'apprentissage de la programmation à l'école élémentaire, mais aussi à comprendre que ce dernier est ancré dans un passé pas si éloigné. Les travaux de recherche portant sur l'apprentissage de la programmation par les enfants en utilisant LOGO se sont intéressés aux stratégies de programmation utilisées par ces derniers, aux stades par lesquels ils passent lorsqu'ils programment, ainsi qu'aux difficultés et aux erreurs des élèves.

3.1. Les travaux traitant les stratégies de programmation des élèves

Un des premiers travaux décrivant, entre autres, les stratégies de programmation mises en place par les élèves sur LOGO était le projet de Brookline (Papert et al., 1979). Il s'agit d'une recherche menée auprès de 16 élèves de 11 à 12 ans, qui ont travaillé de 20 heures à 40 heures avec LOGO. Deux stratégies ont été identifiées : la stratégie *bottom-up* et la stratégie *top down.* Les élèves mettant en place la stratégie *bottom-up* utilisaient une série de sous-procédures pour créer un programme plus complexe. En revanche, les élèves utilisant la stratégie *top down* commençaient leur programmation en ayant une idée claire du résultat à produire. Ils écrivaient ensuite la procédure principale et puis, ils s'occupaient de l'écriture des sous-procédures. En ayant déjà l'idée du produit à réaliser, il était beaucoup plus facile pour les élèves mettant en place cette stratégie de planifier, lire et déboguer les procédures. Papert et al. (1979) ont aussi repéré des élèves dont les performances présentent des caractéristiques provenant de deux stratégies déjà mentionnées.

Pea (1983) a aussi examiné la capacité de 25 élèves de 8 à 9 ans et de 11 à 12 ans à définir un plan précis avant de passer à la programmation sur LOGO. En effet, ce dernier a peu constaté ce comportement chez les élèves. En revanche, il explique que les élèves préféraient passer

directement à l'écriture des programmes sur LOGO pour ensuite les exécuter, voir s'ils ont aimé ce qu'ils ont produit et après revenir sur leur programme pour réaliser des modifications, s'ils avaient envie. L'écriture d'un plan précis avant de passer à la programmation n'était donc pas, d'après cet auteur, dans les habitudes des élèves observés (Pea, 1983).

Lowenthal et al. (1998)ont analysé les stratégies utilisées par 280 élèves de 9 à 12 ans qui ont travaillé en groupe de deux pendant une année scolaire, afin de résoudre des problèmes sur LOGO. Les auteurs ont établi cinq stratégies en fonction de la complexité des procédures utilisées par les élèves. Ils ont également décrit deux procédés que les élèves ont suivis pour résoudre les problèmes donnés sur LOGO.

En effet, certains ont choisi la stratégie *trial and error* ou essaie-erreur en français, car ils arrivaient à la solution du problème donné en réalisant directement des tests avec les commandes de base disponibles sur LOGO, sans avoir créé auparavant une procédure. Certains de ces élèves étaient satisfaits avec cette solution alors que d'autres procédaient à l'écriture d'une procédure pour résoudre le problème en utilisant des éléments des tests réalisés. Lowenthal et al. (1998) présentent également un deuxième type de comportement celui des élèves qui écrivaient d'abord la procédure et ensuite ils la testaient. Si le résultat obtenu correspondait à ce qu'ils souhaitaient, ils étaient satisfaits. Dans le cas échéant, soit ils revenaient à leur solution pour la corriger, soit ils mettaient en place une stratégie d'essai-erreur, car ils avaient besoin de tester une série d'éléments avant d'écrire une nouvelle procédure.

Kapa (1999) a mené une recherche auprès des 45 élèves de 10 à 11 ans, qui ont travaillé à la fois en groupe de deux ou individuellement sur LOGO. Bien que la programmation comme objet d'apprentissage pour les enfants n'ait pas été l'objectif principal de l'étude, cette dernière cite trois types de planification observés : « essai-erreur, pas à pas et planification holistique » [notre traduction] (Kapa, 1999, p. 81). L'auteur explique que les principaux types de planification sont *top down,* qui correspond à une planification holistique et *bottom-up,* qui correspond à une écriture du programme en mode pas à pas. Le troisième type cité dans cette recherche est *trial and error,* mode de programmation utilisé par les élèves qui résolvent un problème donné sans avoir un plan préalablement défini pour le faire.

3.2. Les travaux portant sur les stades par lesquels passent les élèves lorsqu'ils programment

D'autres chercheurs se sont intéressés aux stades développementaux par lesquels passent les élèves lors de l'apprentissage de la programmation sur LOGO. Howe (1980) décrit trois stades. Pendant le premier, les élèves se concentrent sur ce qu'ils veulent produire et non pas sur la méthode à suivre pour y arriver. C'est pourquoi, Howe (1980) nomme ce stade *product oriented.* Au deuxième, les élèves deviennent *style conscious.* Ils se préoccupent davantage de l'efficacité de la méthode utilisée pour atteindre l'objectif souhaité. Ils adoptent ainsi lors de ce stade une série des bonnes habitudes en programmation. Le comportement des élèves au troisième stade est caractérisé par *creative problem-solving*, car ils sont capables d'utiliser LOGO et les ressources qu'ils ont à leur disposition pour créer leurs propres projets en s'inspirant des travaux des autres (Howe, 1980).

Yelland (1995) cite les travaux de Noss[109] (1984) expliquant que les élèves de 8 à 11 ans utilisent différents modes d'apprentissages lorsqu'ils programment sur LOGO. Noss (1984 cité dans Yelland, 1995), remarque que les élèves ont tendance à passer d'un mode d'apprentissage à un autre pendant leur programmation. Avant de présenter les modes d'apprentissage identifiés par cet auteur, il est important de souligner le besoin des élèves d'avoir le pouvoir sur la machine et sur le langage utilisé pour communiquer avec cette dernière, afin de résoudre des problèmes. C'est ce besoin-là qui a incité les élèves à présenter ces modes d'apprentissage d'après Noss (1984 cité dans Yelland, 1995).

Au cours de la première phase les élèves explorent au maximum des nouvelles idées qu'ils ont (Noss 1984 cité dans Yelland, 1995). Ils testent des éléments, ils essaient de voir jusqu'où ils peuvent aller avec la machine ou le langage de programmation. En allant plus loin dans l'exploration d'une idée les élèves commencent, lors de la deuxième phase, à faire des liens entre celle-ci et des idées qu'ils savent déjà utiliser. Enfin, les élèves entrent dans un mode de résolution des problèmes, car ils sont plus préoccupés par l'objectif à atteindre que par l'exploration des nouvelles idées. À cette phase-là, les élèves sont censés avoir compris une série des concepts de programmation qu'ils peuvent combiner, pour arriver au résultat souhaité. Noss (1984 cité dans Yelland, 1995) explique que le modèle qu'il propose ne suit

109 Noss, R. (1984). *Creating a Mathematical Environnement Through Programming : A study of young children Learning Logo.* London : University of London, Institut of Education.

pas une hiérarchie spécifique c'est-à-dire que les élèves peuvent présenter les différents modes d'apprentissage sans suivre l'ordre selon lequel il les a présentés dans son modèle.

3.3. Les travaux traitant de ce que sont capables de faire les élèves, leurs difficultés et leurs erreurs

Une série des travaux de recherche a aussi mis en évidence ce que les élèves arrivent à faire en utilisant LOGO, leurs difficultés et leurs erreurs. Ainsi, Pea (1983) a réalisé une série d'études sur l'utilisation de LOGO par les élèves, afin de vérifier la validité des propos triomphalistes de Papert (1980) et il n'est pas d'accord avec lui sur deux points. Il trouve d'une part qu'il est utopique de penser que les élèves pourront apprendre, en étant actifs, à utiliser le langage de programmation LOGO uniquement par la découverte, sans avoir besoin d'instruction spécifique. D'autre part, il pense qu'il est nécessaire de former les enseignants pour enseigner la programmation à travers LOGO, car sinon le développement des compétences en programmation et en résolution de problème ne serait pas garanti (Pea, 1983).

Mendelshon (1985) repère deux types d'erreurs que les élèves de 10 à 12 ans font lorsqu'ils se trouvent face à une épreuve les invitant à déplacer une « tortue graphique » dans un quadrillage sur une feuille de papier, en poursuivant un programme donné. Il s'agit des erreurs d'interprétation et des erreurs de coordination. Les élèves réalisent des erreurs d'interprétation lorsqu'ils confondent le sens des commandes. Cette confusion pourrait être liée à une mauvaise compréhension de l'effet de la commande sur la « tortue graphique » ou à un manque d'attention. Les erreurs de coordination se produisent lorsque les élèves n'arrivent pas à respecter l'ordre indiqué par le problème donné. Mendelsohn (1985) identifie aussi deux sous catégories d'erreurs de coordination : les erreurs qui concernent le non-respect de l'ordre d'une liste d'instructions et les erreurs qui concernent le non-respect de l'ordre entre plusieurs listes d'instructions.

Fay et Mayer (1988) expliquent que pour utiliser des langages de programmation textuels tels que LOGO, il est nécessaire de savoir leur syntaxe car, le cas échéant, il sera impossible de communiquer avec la machine si les expressions utilisées ne sont pas correctement rédigées. Les auteurs remarquent également que certains élèves ont des difficultés avec la syntaxe du langage LOGO et ils ont souvent besoin de réaliser beaucoup d'exercices pour la maitriser (Fay & Mayer, 1988).

Mayer et Fay (1987) suggèrent une raison expliquant les difficultés rencontrées face à la sémantique du langage LOGO est les idées qu'ils ont déjà lorsqu'ils arrivent à programmer sur LOGO. D'après ces auteurs, lorsque les idées initiales des élèves sont en conflit avec la sémantique de LOGO, ces derniers peuvent réaliser des erreurs en programmation. Ils citent également deux types d'erreurs récurrents chez les élèves : les erreurs égocentriques et les erreurs d'interprétation. En effet, les élèves réalisent des erreurs égocentriques, car ils n'arrivent pas à prendre en compte le point de vue de la tortue. Les erreurs d'interprétation se produisent lorsque les élèves associent la mauvaise action à une commande du langage.

Robert (1985) décrit les résultats des premières expériences réalisées en France avec LOGO auprès des élèves. Il cite entre autres les résultats des expériences réalisées par l'équipe de l'Institut National de la Recherche Pédagogique (I. N. R. P.) avec des très jeunes élèves de 5 à 6 ans et de 6 à 7 ans. Ils ont donné la possibilité aux très jeunes élèves de programmer la tortue en créant un système similaire à celui de LOGO qui leur permettait de le contrôler en introduisant des commandes ayant la forme des cartes perforées dans un lecteur-boitier. L'équipe de l'I. N. R. P. a utilisé ce système des cartes d'une part, car ils n'avaient pas le matériel nécéssaire pour utiliser LOGO et d'autre part, pour permettre aux plus jeunes n'ayant pas encore la capacité à lire et à écrire d'avoir accès à la programmation. Les résultats de cette recherche montrent que la majorité des élèves avait des difficultés à la fois à gérer l'orientation et la rotation du mobile dans l'espace et la réalisation des plusieurs déplacements. Robert (1985) met aussi en avant la capacité des jeunes à écrire une série des commandes sans les exécuter une par une, pour créer un dessin en utilisant le système avec les cartes perforées. Il remarque également que les élèves de 6 à 7 ans ayant moins des difficultés avec l'orientation dans l'espace ont pu être initiés à la création des figures nécessitant la répétition d'une suite des commandes spécifique.

Kurland et Pea (1985) ont étudié le niveau de compréhension de la notion de *récursion* sur LOGO par sept élèves de 8 à 12 ans, ayant un an d'expérience avec ce langage de programmation. En effet, ils leur ont demandé de lire une série des programmes créés par les chercheurs pour mettre en avant une structure récursive LOGO, et ensuite d'expliquer ce qui va se passer si ces derniers ont été exécutés. Ils ont remarqué par exemple que les élèves avaient tendance à utiliser des structures complexes tels que les variables, ou des commandes simples telles que la répétition pour résoudre des problèmes, mais ils n'étaient pas capables de

décrire leur fonctionnalité. Les auteurs avaient aussi repéré que les élèves confondaient la structure de *récursion* sur LOGO avec la structure de répétition (Kurland & Pea, 1985).

3.4. Les travaux analysant le débogage réalisé par les élèves

Des travaux ont également été réalisés sur la manière dont les élèves mettent en place le processus de débogage. Pea (1983) remarque par exemple qu'ils préféraient recommencer leur programmation au lieu de passer du temps à déboguer leur programme. Il explique que cela est dû à la complexité de ce processus et notamment au fait qu'il nécessite à la fois une bonne compréhension de ce qu'il fait le programme contenant l'erreur et la réalisation des corrections précises. Pea (1983) souligne également que les élèves programmant sur LOGO ont la possibilité de se familiariser avec ce processus mais ce n'est pas certain qu'ils vont développer la capacité de déboguer leur programme juste parce qu'ils ont programmé sur LOGO.

Carver et Klahr (1986) ont aussi mené une étude avec neuf élèves de 7 ans ayant suivi 24 heures de formation sur LOGO, pour examiner si ces derniers ont réussi à comprendre et mettre en place de manière opérationnelle le processus du débogage. Les résultats de cette expérience montrent que les jeunes élèves n'étaient pas en mesure de déboguer et qu'ils préféraient même d'éviter cette étape si cela était possible (Carver & Klahr, 1986). Ce résultat confirme le propos de Pea (1983). Papert (1980) avait aussi observé des comportements similaires chez les enfants en ce qui concerne le processus du débogage. Carver et Klahr (1986) pensent que les élèves évitent le processus de débogage parce qu'ils ne savent pas comment le mettre en place. Les auteurs identifient trois raisons expliquant cela : « la complexité du processus du débogage, le fait qu'il nécessite plus des capacités et il est rarement enseigné directement » [notre traduction] (p. 521).

Après avoir présenté des travaux réalisés avec LOGO ayant pour objectif d'étudier la programmation en tant qu'objet d'apprentissage pour les enfants, nous allons dans la prochaine section résumer des travaux menés autour des effets que la programmation pourrait avoir sur d'autres apprentissages.

4. Quels apports possibles de la programmation à d'autres apprentissages ?

Dans cette section, nous allons présenter quelques travaux réalisés avec LOGO ayant pour objectif d'étudier les effets que, l'apprentissage de la programmation par les élèves, pourrait avoir dans d'autres domaines. Nous n'allons pas élaborer davantage sur ce point-là, car dans ce travail de thèse nous avons choisi de nous focaliser sur l'apprentissage de la programmation par les enfants.

Suite aux propos exprimés par Papert (1980) autour des effets que l'introduction des élèves sur l'environnement LOGO pourrait avoir, un grand nombre de travaux ont été réalisés pour étudier cette question. Clements (1999) identifie des travaux portant sur les effets de l'apprentissage de la programmation sur plusieurs domaines : les mathématiques, la résolution de problèmes, le développement des compétences de réflexion de haut niveau, le langage, la créativité et le développement socio-émotionnel. Liao et Bright (1991) remarquent que les résultats de la recherche sur les effets de la programmation sont contradictoires, certains travaux montrant des effets positifs de la programmation sur d'autres domaines et d'autres, n'allant pas dans ce sens.

Pea et Kurland (1984) par exemple sont peu optimistes en ce qui concerne les effets que la programmation pourrait avoir sur d'autres domaines. Ils remettent même en question l'idée de s'attendre à ce qu'il y a des effets de l'apprentissage d'un domaine sur d'autres. En effet, ils remarquent que la recherche relative à ce sujet ne permet pas de confirmer ce propos. Ils identifient aussi une limite commune à toutes les recherches menées jusqu'au moment de la rédaction de cet article. En effet, Pea et Kurland (1984) mettent en évidence le fait que ces travaux cherchent à identifier des effets cognitifs, qui, d'après le modèle proposé par eux, font leur apparence après un niveau d'expérience important en programmation, alors que les sujets participant à ces études en ont peu. Ils expliquent ainsi qu'il n'y a pas assez de recherches dont les résultats montrent que la programmation favorise le développement d'une rigueur mathématique. La programmation ne promeut non plus l'exploration de mathématiques par les élèves, d'après les travaux existants sur ce sujet (Pea & Kurland, 1984).

Marcel Crahay (1987) exprime aussi ses doutes par rapport à l'apport de l'apprentissage de la programmation sur LOGO à la résolution de problèmes et il recommande d'avoir des attentes

plus modestes. Dans un article publié dans la Revue Française de Pédagogie, l'auteur résume les principales idées de Seymour Papert et en se basant sur une série des travaux américaine, portant sur l'apport de la programmation à la capacité de résolution de problèmes chez les élèves, dont les résultats étaient décevants il souligne :

> « Croire que programmer en langage LOGO peut développer une aptitue générale à resoudre des problèmes, c'est commettre la même erreur que d'attribuer au latin la faculté de développer l'esprit logique et les capacités de raisonnement » (Crahay, 1987, p.43).

Clements et Gullo (1984) ont aussi étudié les effets que l'apprentissage de la programmation par les jeunes élèves pourrait avoir sur leurs capacités cognitives. Les résultats de ce travail valident les propos de Papert (1980). Les auteurs ont mené cette étude auprès de 16 élèves de cours préparatoire. La moitié a suivi 12 semaines de cours sur LOGO et l'autre moitié, a suivi un enseignement assisté par ordinateur (EAO) pendant la même période de temps. À la fin de cette expérience, les chercheurs ont comparé les réponses des élèves de deux groupes au pré-test et au post-test et ils ont constaté que les élèves ayant fait la programmation sur LOGO ont présenté des meilleures capacités de réflexion divergente et creative que le groupe contrôle (Clements & Gullo, 1984). Ils ont aussi remarqué que les élèves ayant fait de la programmation réalisaient moins d'erreurs, ils étaient moins de temps inactifs et ils avaient une meilleure performance face aux épreuves testant les capacités méta-cognitives, que les élèves du groupe contrôle. Ils ont ainsi montré que l'apprentissage de la programmation pourrait favoriser certains aspects du processus de résolution des problèmes.

Clements et Nastasi (1985) ont mené une recherche auprès de 24 élèves de 6 à 7 ans et de 24 élèves de 8 à 9 ans, afin d'étudier les effets, de leur introduction à l'apprentissage de programmation sur LOGO et à l'enseignement assisté par ordinateur, sur leur développement socio-émotionnel. Les résultats de cette recherche montrent que l'introduction à LOGO et à EAO favorise à la fois le travail collaboratif et le désaccord chez les élèves. Ils ont également remarqué que les élèves du groupe LOGO étaient enthousiastes vis-à-vis de la découverte des nouvelles informations.

D'autres études ont été réalisées pour vérifier les effets que pourrait avoir l'apprentissage de la programmation à travers LOGO sur la conceptualisation des concepts en géométrie. Une de ces études, mené auprès de 48 élèves de CE2 qui ont été confiés à un groupe LOGO et un groupe contrôle, montre que ceux exposés à l'environnement LOGO avaient une meilleure

capacité de conceptualisation des angles et des formes que ceux qui ont suivi le groupe contrôle (Clements & Battista, 1989).

En consultant la littérature disponible sur les effets de la programmation à l'aide de LOGO sur d'autres domaines, Clements et Meredith (1993) remarquent que l'apprentissage de la programmation sur LOGO peut favoriser le développement des compétences en mathématiques, la pensée géométrique et les capacités de résolution des problèmes sous la condition que les élèves soient bien accompagnés dans le processus d'apprentissage. En revanche, ces auteurs sont moins optimistes concernant les effets de l'apprentissage de la programmation avec LOGO sur les capacités de lecture (Clements & Meredith, 1993). Bien qu'ils n'existent pas assez des données pour confirmer les effets de l'apprentissage de la programmation en LOGO sur le langage, Clements et Meredith (1993) repèrent que les premiers résultats montrent des effets positifs.

Liao et Bright (1991) ont analysé 65 études portant sur les effets cognitifs de la programmation et ont trouvé que, sur les 58 études les résultats montraient un transfert positif et une meilleure performance de la part du groupe ayant fait de la programmation que celle du groupe de contrôle.

Comme nous venons de le voir, les effets de la programmation sur d'autres apprentissages sont sujets à controverse et les débats autour de cette question ne sont pas clôs. D'autres recherches doivent être réalisées pour élargir nos connaissances sur ce domaine.

5. Autres apprentissages : les prérequis

La question étudiée par les recherches citées dans la section précédente a été aussi retournée dans l'autre sens par les chercheurs souhaitant identifier les prérequis nécessaires pour apprendre la programmation. La présente section portera ainsi sur les prérequis nécessaires pour l'apprentissage de la programmation par les enfants.

Pour pouvoir programmer, il est nécessaire de passer, comme on l'a vu au début de ce chapitre, par des phases spécifiques de résolution de problème. L'activité de programmation nécessite également une série de connaissances et des capacités qui proviennent d'autres domaines (Dufoyer, 1988). Dufoyer (1988, p. 95) explique qu'il est important « de pouvoir

effectuer des transferts d'apprentissage de ces domaines vers celui de la programmation, au moment opportun ».

Les connaissances et les capacités déjà acquises par les élèves au moment de leur introduction à l'apprentissage de la programmation dépendent en grande partie de leur âge et leur niveau développemental. Nous retournerons ainsi souvent vers la psychologie développementale afin de nous renseigner sur ce que les élèves sont capables de faire à différents âges (Duncan et al., 2014). La théorie de Piaget est une référence majeure dans le domaine. Piaget identifie quatre stades de développement par lesquels passent les enfants dans le même ordre pour développer leur intelligence : le stade sensori-moteur (naissance à 2 ans), le stade pré-opératoire (de 2 ans à 7-8 ans), le stade opératoire concret (de 7- 8 ans à 11-12 ans) et le stade opératoire formel (de 11-12 ans à 16 ans) (Piaget & Inhelder, 2012).

Pea et Kurland (1984) sont contre l'idée d'analyser l'apprentissage de la programmation selon les stades de développement proposés par Piaget pour deux raisons. D'une part, parce que, selon ces auteurs, il existe des travaux montrant que le développement de l'intelligence des élèves peut être influencé par les caractéristiques inhérentes de l'individu, le contexte mais aussi le domaine concerné (Pea & Kurland, 1984). D'autre part, parce que l'apprentissage de la programmation n'a pas été, jusqu'à ce moment-là, analysé selon la théorie du développement et ainsi, nous n'avons pas de connaissances par rapport aux capacités qui pourraient émerger par l'apprentissage de la programmation (Pea & Kurland, 1984).

Ces derniers citent six prérequis nécessaires pour apprendre la programmation : la capacité mathématique, la capacité de mémoire, la capacité de raisonnement analogique, la capacité de raisonnement conditionnel, la capacité de pensée procédurale et la capacité de raisonnement temporel. Ils remarquent que les mathématiques et la programmation sont des domaines voisins et qu'ainsi une capacité mathématique est nécessaire pour apprendre à programmer. Ils expliquent également que pour programmer, nous sollicitons une grande partie de notre mémoire à court terme, car nous avons besoin de retenir un certain nombre d'informations afin de résoudre efficacement un problème donné (Pea & Kurland, 1984). Ils soulignent aussi l'importance d'être capable de réfléchir par analogie pour utiliser des connaissances et des compétences développées dans d'autres domaines en programmation et vice versa.

Ils mettent aussi l'accent sur la capacité de raisonnement conditionnel, car c'est un élément indispensable pour comprendre des structures complexes en programmation tels que la

répétition, les tests etc. Les chercheurs soulignent également que les élèves ayant des expériences des procédures séquentielles de la vie quotidienne tels que la définition et l'exécution d'une séquence d'instructions ou l'exécution d'une recette avec une série d'étapes auront plus de facilité à apprendre la programmation (Pea & Kurland, 1984).

Enfin, pour programmer il est nécessaire d'avoir une capacité de raisonnement temporel puisqu'il faut à la fois comprendre l'exécution séquentielle des commandes et être capable d'identifier l'ordre dans lequel devraient se passer les différentes actions dans un programme (Pea & Kurland, 1984).

Le temps est une notion complexe notamment pour les jeunes élèves qui acquièrent progressivement différents aspects de cette notion (Tartas, 2010). L'auteure remarque que le développement des capacités langagières chez l'élève joue un rôle important dans sa capacité à percevoir le temps. Les élèves élaborent, grâce à l'utilisation du langage, un système « avant/après » leur permettant d'ordonner des événements dans une séquence temporelle (Tartas, 2010). En revanche, ils ont plus des difficultés à situer les événements dans le présent, le passé ou le futur. Les élèves forment une meilleure perception du temps à partir de 7 à 8 ans, lorsqu'ils entrent dans le stade opératoire concret (Piaget, 1981).

Ce qui est un prérequis devient aussi un objectif grâce à l'activité de programmation, comme nous venons de le voir dans la section précédente. La multitude de prérequis nécessaires pour l'apprentissage de la programmation par les élèves et la recherche réalisée sur l'introduction des élèves à la programmation sur LOGO ont joué un rôle important dans l'évolution des environnements et des langages de programmation. Suite à cette période de première fondation, des nouveaux robots programmables et des nouveaux langages de programmation graphiques sont apparus, donnant ainsi lieu à une nouvelle période d'expérimentations scientifiques dans le milieu de l'apprentissage de la programmation par les enfants.

Chapitre 2. Apprentissage de la programmation par les enfants : nouvelle ère

Suite à la période de première fondation présentée dans le chapitre précédent, des nouveaux robots programmables et des nouveaux environnements de programmation graphiques ont apparu, donnant ainsi lieu à une nouvelle période d'expérimentations scientifiques dans le milieu de l'apprentissage de la programmation par les enfants. Ce chapitre portera sur cette nouvelle période. Dans un premier temps, nous embarquerons dans une exploration de la pensée informatique en présentant les définitions existantes autour de cette notion et en mettant l'accent sur celle proposée par Wing. Nous expliciterons l'enjeu derrière cette vieille idée ainsi que son lien avec l'apprentissage de la programmation et nous nous positionnerons par rapport aux débats existants autour de cette dernière.

Par la suite, nous nous concentrerons sur la robotique et les langages de programmation visuels ou graphiques, en mettant en avant leurs avantages et les résultats de travaux réalisés avec des élèves. Puis, nous mettrons en évidence les raisons qui ont motivé l'évolution des langages de programmation graphiques, afin de correspondre aux besoins de très jeunes élèves. Nous présenterons également les résultats des travaux réalisés avec des langages de programmation graphiques autres que ScratchJr. La section suivante sera dédiée à la présentation de l'environnement et du langage de programmation ScratchJr, des objectifs derrière sa création et des résultats des travaux réalisés à l'aide de ce dernier sur l'apprentissage de la programmation par des très jeunes élèves. À la fin de ce chapitre, nous définirons la problématique et les questionnements de cette recherche.

1. La pensée informatique : résurgence d'une vieille idée

Dans cette section nous allons traiter la notion de pensée informatique ou *computational thinking* en anglais. Il s'agit d'une notion qui a suscité nombreux débats au sein de la communauté scientifique et plusieurs chercheurs ont proposé dans les années des définitions différentes. Comme nous l'avons vu dans la Partie 1 de cette thèse, les bénéfices potentiels liés au développement d'une pensée

informatique chez les élèves sont un des arguments fréquemment utilisés en faveur d'une initiation à l'informatique et en programmation. La notion a connu une grande célébrité suite à la publication en 2006 de l'article de Jeanette Wing intitulé *Computational Thinking*. La communauté scientifique et éducative a montré un intérêt majeur pour ce type de pensé. La définition proposée par Wing (2006) a suscité de nombreuses interprétations.

Bien que Wing (2006) soit à l'origine de la popularisation de la pensée informatique, cette forme de pensée a un passé riche, que nous présenterons avant de revenir à la définition de Wing. Baron et Drot-Delange (2016) expliquent que l'idée d'une « pensée singulière » (p. 3) remonte aux années 1970. Jacques Arsac (1988) fait référence au colloque de Sèvres organisé en 1970, proposant une introduction de l'informatique dans l'enseignement en France. Arsac (1988) souligne que l'argument mis en avant en faveur de cette introduction était les aptitudes « algorithmiques, organisationnelles et opératoires »(p. 12) développées grâce à l'apprentissage de la programmation.

Papert (1980) met aussi l'accent sur la pensée procédurale ou *procedural thinking* (p.155) en anglais. En effet, Papert remarque qu'en apprenant la programmation dans un environnement de programmation comme LOGO les élèves peuvent développer une pensée procédurale, car ils ont la possibilité d'utiliser les procédures comme des « objets-pour-penser-avec »[notre traduction](p.11). Il souligne également qu'il est important de bien accompagner les élèves dans ce processus de développement de pensée procédurale au sein de l'environnement LOGO (Papert, 1980). Feurzeig (2010) remarque l'émergence d'une pensée logique par des élèves de 7 ans lors d'une des premières expérimentations menées avec LOGO en 1969.

Wing (2006) remet à l'ordre du jour ce type de pensée. Elle définit la pensée informatique comme une compétence fondamentale pour tout le monde et pas seulement pour les informaticiens. D'après elle, la pensée informatique, en s'appuyant sur des concepts de la science informatique, sous-tend la résolution de problèmes, la conception des systèmes et la compréhension du comportement humain. Wing (2006) propose de mettre à jour la trilogie « lire-écrire-compter » en lui ajoutant la pensée informatique. Elle considère également que la pensée informatique influe aussi sur d'autres disciplines.

L'auteur met également l'accent sur le fait que la pensée informatique va au-delà de l'apprentissage de la programmation, car elle nécessite une réflexion à plusieurs niveaux d'abstraction (Wing, 2006). Drot Delange et al. (2019) soulignent le fait que Wing réalise

cette distinction entre pensée informatique et apprentissage de la programmation, alors que dans la majorité des travaux portant sur la pensée informatique les chercheurs ne le font pas. Nous avons fait le même constat que Drot-Delange et al. (2019) et c'est pourquoi nous avons pris la décision de ne pas présenter dans cette section les travaux portant sur l'apprentissage de la programmation dont les chercheurs ne font pas cette distinction.

Wing revient sur la pensée informatique dans un article de 2008, *Computational thinking and thinking about computing.* Dans celui-ci, l'auteure souligne que ce type de pensée analytique a des liens forts avec la pensée mathématique, la pensée de l'ingénierie et la pensée scientifique. Elle met également l'accent sur le rôle central de l'abstraction. Wing (2008) souligne que l'abstraction est présente à la fois au sein de l'algorithme et d'un langage de programmation. Elle explique que les abstractions sont les outils « mentaux » de l'informatique et pose les deux questions suivantes : « Quelle éducation à la pensée informatique ? Quels sont les concepts fondamentaux à enseigner ? ».

En 2017, Wing revient sur la pensée informatique et elle souligne l'influence que cette notion a eue sur la recherche et l'éducation depuis l'apparition de son article en 2006 (Wing, 2017). Elle explique aussi les raisons qui l'ont amené à rédiger cet article qui a suscité des nombreuses interprétations. Ces travaux sont considérés comme fondateurs de la pensée informatique.

Denning (2009) remet très tôt en question la notion de pensée informatique. Il remarque qu'elle limite le terrain de la science informatique, car elle ne correspond qu'à une seule facette de celle-ci. Il craint qu'en se focalisant sur la pensée informatique on risque de perdre l'opportunité de montrer aux autres disciplines le caractère unique de l'informatique et son potentiel. Il considère que la pensée informatique est l'une des pratiques de l'informatique, elle n'est pas propre à cette dernière et ainsi, elle ne peut pas représenter l'intégralité du champ. Il remarque également que la pensée informatique était connue en tant que « pensée algorithmique » (p. 28) dans les années 1950 et 1960.

Denning souligne presque une décennie plus tard, le fait que les promesses irréalistes réalisées par des enthousiastes de l'introduction de l'informatique dans l'enseignement primaire mettent les enseignants dans une position d'incertitude et cela pourrait éventuellement poser des problèmes à cet enseignement (Denning, 2017).

Grover et Pea (2018) associent une série des concepts et des pratiques à la pensée informatique. Pour eux, elle comprend les concepts suivants : la logique et la pensée logique, les algorithmes et la pensée algorithmique, les patterns et la reconnaissance de ces derniers, l'abstraction et la généralisation, l'évaluation et l'automatisation (Grover & Pea, 2018). Les pratiques de la pensée informatique d'après ces auteurs sont : la décomposition de problèmes, la création d'artefacts computationnels, le test et le débogage, le processus d'amélioration répétitif et la collaboration et la créativité. Grover et Pea (2018) ont essayé d'aller un peu plus loin en proposant des exemples d'enseignement de ces concepts et ces pratiques dans les classes.

Barr et Stephenson (2011) en prenant appui sur les travaux de Wing (2006), remarquent que pour une introduction efficace de la pensée informatique à l'école élémentaire il est essentiel de disposer d'une définition opérationnelle regroupant des concepts et des capacités propres à ce type de pensée. Les auteurs mettent aussi l'accent sur le fait que la définition opérationnelle doit être accompagnée d'exemples concrets d'introduction de la pensée informatique dans les classes pour faciliter le travail des enseignants. Barr et Stephenson (2011) soulignent que les élèves doivent commencer à résoudre des problèmes de type algorithmique et employer des méthodes en lien avec la pensée informatique dès l'école primaire. Ils remarquent également un manque de consensus dans la communauté éducative autour de la définition de cette notion.

Comme nous venons de le montrer, la pensée informatique est une vielle idée qui est revenue à l'ordre du jour au début du mouvement international autour de l'introduction de l'informatique et notamment de la programmation à l'école élémentaire. Depuis sa résurgence en 2006 et jusqu'aujourd'hui d'ailleurs, plusieurs définitions de ce type de pensée ont été réalisées. Différents concepts, capacités et modèles d'évaluation ont été proposés pour la rendre moins obscure. Nous jugeons important de souligner que l'enjeu derrière cette notion était de renforcer les arguments en faveur de la nécessité d'un enseignement de l'informatique dans l'enseignement obligatoire. Nous sommes d'accord avec Denning (2017) sur le fait qu'il faut limiter les promesses irréalistes par rapport à l'universalité de cette pensée. Enfin, comme le constate Drot-Delange et al. (2019) « beaucoup regrettent le manque de recherches expérimentales ou quasi-expérimentales dans les classes ainsi que la rareté des évaluations » (p. 14).

2. Robots programmables

Dans cette section nous allons nous concentrer sur une des approches pour l'apprentissage de la programmation pour les enfants, la robotique. Selon Misirli et Komis (2012, p. 1), la robotique « est constituée par l'ensemble des méthodes et des techniques de conception et de mise en œuvre des robots ». Clements (1999) dans un article qu'il a écrit à l'aube du nouveau millénaire, met l'accent sur la robotique et les langages de programmation graphiques en soulignant le besoin de réaliser plus des recherches expérimentales pour explorer les possibilités de ces nouvelles approches très prometteuses.

À la suite des travaux de Papert (1980) sur la tortue de plancher, des nouveaux robots programmables ont été conçus pour donner la possibilité à des élèves plus jeunes de découvrir la programmation. Wyeth (2008) explique que les robots programmables sont efficaces pour l'introduction des jeunes élèves à la programmation, car il s'agit de vrais objets qu'ils peuvent manipuler pour écrire des programmes et voir directement leurs effets sur ces objets. Plusieurs travaux de recherche ont été réalisés au cours du temps pour explorer les potentielles utilisations des robots programmables par des jeunes enfants et les effets de ces derniers sur l'apprentissage de la programmation et sur d'autres apprentissages.

En ce qui concerne les effets que l'apprentissage de programmation à l'aide des robots programmables pourrait avoir sur d'autres apprentissages, nous pouvons en particulier citer le travail de Greff (2000) qui a étudié les effets que l'introduction des élèves de grande section de maternelle à la programmation à travers du robot de plancher pourrait avoir sur la résolution des problèmes et les apprentissages premiers (se repérer dans l'espace, se repérer dans le temps, la construction du nombre, le respect de la consigne, le raisonnement hypothético-déductifs implicites). Nous pouvons également citer les travaux plus récents de Kazakoff et al. (2013) et Kazakoff et Bers (2014) sur les effets qu'a l'apprentissage de la programmation à travers l'environnement de programmation tactile et graphique CHERP et les robots LEGO Mindstorms ou uniquement avec CHERP sur la capacité de très jeunes élèves à mettre dans l'ordre correct une série des images (séquence).

Quant aux travaux explorant les effets que l'utilisation des robots programmables pourrait avoir sur l'apprentissage de programmation, nous pouvons commencer par citer la recherche de Pekarova (2008). Cette auteure a mené une recherche auprès des 24 élèves de 5 à 6 ans

d'une classe de maternelle en Slovaquie, afin de tester une série d'activités conçues en utilisant le robot programmable Bee-Bot et d'observer les réactions de jeunes élèves. Pekarova (2008) remarque que la simple introduction des robots programmables dans la classe ne résulte pas en l'engagement des jeunes élèves. Elle souligne l'importance de l'organisation d'activités robotiques ayant du sens pour les élèves. Elle note que les 24 élèves d'école maternelle ont apprécié jouer avec Bee-Bot, le robot programmable choisi pour cette étude.

Flannery et Bers (2013) ont aussi réalisé une recherche dans leur laboratoire auprès de 29 élèves d'un niveau équivalent à la moyenne et grande section de l'école maternelle. L'objectif de cette recherche a été d'examiner ce que les très jeunes élèves peuvent faire en programmation à l'aide du langage de programmation tangible et graphique CHERP[110] et des robots LEGO Mindstorms et quelles sont les approches de programmation qu'ils utilisent selon leur niveau de développement cognitif. Les chercheurs ont étudié la capacité de très jeunes élèves à programmer leur robot pour danser le « Robot Hokey-Pokey ». Ils ont d'abord, laissé les élèves élaborer leur programme et après un certain temps, ils sont intervenus pour les aider à finir l'épreuve. Flannery et Bers (2013) ont évalué la performance des élèves à la programmation du robot selon deux critères : la correspondance, c'est-à-dire si les élèves avaient utilisé les commandes qu'il fallait pour que le robot réalise correctement les actions de la chanson et la complétude ou *completeness* en anglais, c'est-à-dire si les élèves avaient utilisé les commandes dans le bon ordre en respectant la chanson.

Flannery et Bers (2013) ont classifié la performance des élèves pendant la réalisation de l'activité de programmation selon une adaptation du modèle de Feldman[111] (2004), en trois niveaux de développement cognitif : *late pre-operational*, *transitional* et *early concrete operational.* Ainsi, ces chercheurs ont identifié des caractéristiques différentes dans la performance des élèves en programmation, selon leur niveau du développement.

Huit élèves sur 29 ont été situés dans le niveau de développement *late pre-operational*. Ils ont préféré explorer ce que le robot pourrait faire et ses limites au lieu de s'occuper de l'épreuve donnée. Ils ont principalement utilisé une approche d'essai-erreur en écrivant une ou deux actions mais pas plus. Les élèves de ce groupe avaient des difficultés avec le processus de

110 Creative Hybrid Environement for Robotic Programming (CHERP). CHERP a été créé par l'Université de Tufts.

111 Feldman, D. H. (2004). Piaget's stages ; The unfinished symphony of cignitive development. *New Ideas in Psychology,* 22, 175-231. doi:10.1016/j.newdeapsych.2004.11.005

débogage et en particulier à vérifier l'exactitude de leur programmation et à corriger les erreurs présentes sur leurs programmes. Ils préféraient recommencer du début au lieu de corriger leur programme.

Sept sur 29 ont manifesté des caractéristiques des deux niveaux, *late pre-operations* et *early concrete opérations*, et c'est pourquoi Flannery et Bers (2013) les ont positionnés dans le niveau *transitionnal*. Leurs comportements varient en termes de leur approche systématique de programmation. Certains élèves ont commencé leur programme avec une estimation et ensuite, ils ont été capables de réaliser une série de débogages, mais ils ne sont pas arrivés à la solution. D'autres, ont commencé avec une approche systématique et ensuite, soit ils ont arrêté leur programmation à un point qui les satisfaisait, même si leur programme n'était pas correct, soit ils ont continué et avec aide ils sont arrivés à la correcte solution. Les élèves de niveau *transitional* avaient aussi des difficultés avec le débogage.

Enfin, 14 élèves sur 29 ont présenté, d'après Flannery et Bers (2013), des caractéristiques d'un niveau *early concrete opérational.* Les élèves de ce groupe ont mis en place des approches systématiques pour écrire et corriger leurs programmes. Ils étaient également capables de reconnaître rapidement les erreurs sur leurs programmes et les corriger. Pour élaborer leur programme, ils analysaient petit à petit la chanson en étant ainsi très efficaces.

Komis et Misirli (2013) ont réalisé une recherche, auprès de 108 élèves de 4 à 6 ans issus de sept classes d'école maternelle en Grèce. L'objectif de la recherche a été d'examiner les processus de construction d'algorithmes et de programmes par les très jeunes enfants à l'aide du robot programmable, Bee-Bot. Komis et Misirli (2013) ont étudié les stratégies de programmation utilisées par les élèves et le débogage mis en place. Ils ont conçu un scénario pédagogique sur la robotique pédagogique où Les élèves devraient programmer Bee-Bot afin qu'il réalise un trajet spécifique dans l'espace quadrillé à deux dimensions. Pour résoudre ce problème, ils procèdent à quatre étapes enseignées dans le scénario pédagogique : 1) conception et verbalisation de l'algorithme, 2) écriture de l'algorithme avec des cartes représentant les commandes de programmation, 3) programmation du robot et 4) exécution du programme et correction si besoin (Komis & Misirli, 2013).

Ils ont identifié quatre stratégies de programmation principales, dont « totale », « pas à pas » et « sous-programme » contiennent une verbalisation de l'algorithme et une écriture de l'algorithme à l'aide des cartes représentant les commandes de programmation avant la

programmation du robot. La quatrième stratégie, celle de l'« essai-erreur » n'implique pas ces deux étapes. Les stratégies « pas à pas » et « totale » ont été enseignées dans le cadre du scénario pédagogique. L'élève mettant en place une stratégie « totale », écrit la totalité du programme à l'aide des cartes et ensuite il programme le Bee-Bot, alors que pour la stratégie « pas à pas », l'élève écrit une commande à la fois à l'aide des cartes et ensuite il programme le Bee-Bot. L'élève mettant en place la stratégie « sous-programme », sépare le trajet à réaliser en sous trajets et de la même manière que pour les stratégies « totale » et « pas à pas », il écrit d'abord à l'aide des cartes chaque sous-programme et ensuite il programme le Bee-Bot. Il réalise ce processus pour tous les sous-programmes. Komis et Misirli (2013) ont aussi identifié des hybridations de ces stratégies : « stratégie totale et pas à pas », « stratégie totale et sous-programme » et « stratégie sous-programme et pas à pas ».

Leur intérêt a aussi porté sur la capacité des très jeunes élèves à déboguer le programme créé à l'aide des cartes et du robot programmable. Cinq approches de débogage ont été identifiées par les auteurs : débogage effectif (31%), débogage non effectif (1%), programme correct et non débogage (41%), programme non correct et non débogage (18%) et enfants ne s'exprimant pas (9%). En se basant sur ces résultats, les chercheurs remarquent que la majorité des élèves ne réalise pas d'erreurs dans leurs programmes. Ils notent également que 31 % arrivent à corriger de façon efficace les erreurs sur leurs programmes et 18 % ne procèdent pas au débogage bien que leur programme contient des erreurs. Un seul élève n'arrive pas à corriger son programme et 10 n'ont pas participé à la séance d'évaluation.

Komis et al. (2017) proposent également une taxonomie d'activités robotiques en fonction du niveau de participation des apprenants (de l'école maternelle à l'enseignement secondaire) dans le processus d'apprentissage. Cinq types d'activités sont présentés : exposition passive à la robotique (sans manipulation), discussion autour la robotique (sans manipulation), robotique individuelle ou collaborative en pas à pas (procédurale), robotique orientée-ingénierie (individuelle ou collaborative) et robotique co-créative orienté-projet pour résoudre un problème.

Strawhacker et Bers (2015) ont mené une recherche auprès de 37 élèves des trois écoles maternelles qui ont été exposés à un curriculum de neuf semaines sur l'apprentissage de la programmation à l'aide des robots programmables. Les concepts de programmation étudiés dans ce curriculum sont : la séquence, la répétition et la programmation créative. Chaque

classe d'école maternelle a utilisé une interface différente pour programmer les robots : tangible, graphique et hybride. Les élèves ont utilisé les LEGO Education Wedo kit de construction robotique pour créer leurs robots et le langage de programmation tactile et graphique CHERP pour les programmer. Les élèves (14 au total) de la première classe ont programmé leurs robots en utilisant la version tangible de CHERP avec les blocs en bois. Ceux (7 au total) de la deuxième classe ont utilisé la version graphique de CHERP avec des blocs sous la forme d'icônes dans un environnement de programmation de type drag-and-drop. Dans la troisième classe, les élèves (14 au total) ont utilisé à la fois la version tangible et la version graphique de CHERP pour programmer leurs robots.

Un des résultats importants que Strawhacker et Bers (2015) relèvent de cette étude, est le fait que les élèves ayant travaillé avec l'interface tangible ont présenté face aux épreuves portant sur la répétition une meilleure performance, que les élèves ayant travaillé avec l'interface graphique et que ceux ayant travaillé avec les deux. Les élèves ayant travaillé avec l'interface graphique ont présenté une meilleure performance face à l'épreuve concernant la séquence. Strawhacker et Bers (2015) remarquent également que les élèves d'école maternelle avaient tendance à se référer à leurs constructions parfois en tant que robots et parfois en tant qu'animaux. Les chercheurs mettent en avant le fait que, même si les élèves de maternelle semblaient avoir compris que les robots n'étaient pas des vrais animaux, souvent ils attribuaient à leurs robots des taches anthropomorphiques.

Un des résultats qui a étonné ces auteurs a été la capacité des élèves à représenter leurs robots en tant qu'animaux, car cela nécessite une capacité d'abstraction que Piaget situe après l'âge de 7 ans. En revanche, les chercheurs ont observé que les élèves de maternelle étaient capables de voir leur robot à la fois en tant qu'objet et en tant qu'animal vivant. Ce constat pourrait expliquer, d'après eux, la performance des élèves face un concept nécessitant des capacités d'abstraction, tel que la répétition. Strawhacker et Bers (2015) ont été surpris de voir que, malgré leur jeune âge, les élèves de maternelle de trois classes démontraient une forme de compréhension du concept de répétition.

Dans cette section, nous avons pu mettre en avant le rôle des robots programmables dans l'initiation des très jeunes élèves à l'apprentissage de la programmation. En revanche, la robotique n'est pas la seule approche permettant aux élèves de découvrir ce qui est la programmation. Les langages de programmation graphiques ou visuels permettent aussi une

initiation des jeunes élèves à la programmation. Dans la prochaine section, nous développerons davantage ce point.

3. Environnements et langages de programmation visuels

Dans cette section nous allons nous concentrer à une autre approche souvent utilisée pour l'initiation des élèves à la programmation, les langages de programmation graphiques ou visuels. Comme nous l'avons vu dans la Partie 1 de ce travail de thèse, l'apparition de ce type des langages a joué un rôle important pour le renouvellement d'intérêt autour de l'enseignement de la programmation aux enfants.

3.1. Les avantages de cette approche

La nature des langages a été modifiée au cours du temps afin d'être mieux adaptée aux besoins des élèves. Les langages de programmation visuels ou graphiques donnent la possibilité de programmer en utilisant des blocs représentant des commandes de programmation, sans avoir besoin de connaître une syntaxe complexe (Duncan et al., 2014). Les blocs graphiques représentant les commandes dans ce type de langages ont une forme qui oblige l'utilisateur à les assembler correctement pour créer des programmes (Duncan et al., 2014).

Ces langages de programmation font partie d'environnements de programmation de type *drag and drop*, car les élèves peuvent choisir les commandes sous forme de blocs graphiques et les glisser vers un endroit de l'environnement qui peut être utilisé pour créer des programmes. Puisque les élèves n'ont plus besoin de réfléchir à la question de syntaxe, ils peuvent se focaliser sur l'apprentissage de la programmation et des concepts informatiques (Lye & Koh, 2014). Les langages de programmation graphiques ou visuels peuvent être aussi utilisés pour créer des animations et des produits multimédias en permettant ainsi aux jeunes élèves d'exprimer leur créativité (Lye & Koh, 2014).

3.2. Le cas de Scratch

Scratch est un environnement de programmation visuel ou graphique. Il a été développé par le groupe de recherche *Lifelong Kindergarten* auprès du laboratoire Média du MIT. C'est un logiciel libre conçu pour initier les élèves dès l'âge de 8 ans à des concepts fondamentaux de programmation, grâce à son interface ludique. Il est disponible en ligne et hors ligne. Il facilite la création d'histoires interactives, de dessins animés, de jeux, et peut se partager sur le Web[112]. Le projet autour de Scratch a commencé en 2003 alors que le logiciel Scratch et le site web existent depuis 2007 (Maloney et al., 2010). Le site de Scratch[113] permet d'une part, de partager ses projets sur le Web et d'autre part, d'apporter de l'aide à la mise en œuvre de Scratch (Resnick et al., 2009).

Trois versions du logiciel Scratch sont disponibles à télécharger en ce moment sur internet : la version 1.4, la version 2 et la version 3 qui est la plus récente. L'objectif du projet Scratch a été d'initier à la programmation des personnes n'ayant pas d'expérience préalable dans ce domaine (Maloney et al., 2010). Un des avantages de cet environnement est le fait que l'utilisateur n'a pas besoin de passer par une phase de compilation. Scratch donne également la possibilité aux élèves de voir le programme de l'espace de programmation qui est exécuté chaque fois (Maloney et al., 2010).

Scratch est un langage de programmation qui a été créé afin d'avoir un « plancher bas », un « plafond haut » et des « murs larges ». Un « plancher bas », pour que l'utilisateur puisse créer des petits programmes de débutant, facilement, sans avoir de connaissances préalables. Un « plafond haut », pour que l'utilisateur puisse créer des projets davantage plus complexes au fur et à mesure qu'il avance sur l'utilisation du logiciel et des « murs larges », afin que l'utilisateur puisse créer plusieurs types de projets (Resnick et al., 2009).

3.3. Des expérimentations avec Scratch

Depuis son apparition en 2007, Scratch a suscité beaucoup d'intérêt. Plusieurs travaux de recherche ont été réalisés afin d'explorer les possibilités de cet environnement pour l'apprentissage de la programmation par des élèves, mais pas uniquement. Ce logiciel a été souvent utilisé pour l'initiation des élèves à des concepts informatiques tels que la séquence,

112 Partie d'un article que nous avons déjà publié dans le cadre du colloque éTIC2.
113 https://scratch.mit.edu/

les messages, la répétition etc. Nous pouvons citer par exemple le travail d'Armoni et Ben-Ari (2013) qui ont écrit un livre destiné à des élèves et des enseignants, afin de les aider à prendre en main Scratch, pour apprendre à programmer et ainsi à se familiariser avec l' informatique. Des recherches expérimentales ont aussi été réalisées afin d'étudier l'apport de l'utilisation de Scratch à l'apprentissage de programmation par des élèves.

Wilson et Moffat (2010) ont réalisé une recherche dans une école primaire à Glasgow, en ayant comme objectif d'évaluer le logiciel Scratch dans sa capacité à initier des élèves à la programmation. Ils ont enseigné la programmation pendant huit semaines, auprès de 20 élèves, âgés de 8 à 9 ans et ils ont abouti aux résultats suivants. Grâce à son langage visuel et dynamique, Scratch a limité les possibilités d'erreurs, et diminué le sentiment de frustration qui accompagne souvent la programmation. C'est pourquoi, le logiciel a paru très efficace aux chercheurs pour initier de manière ludique et plaisante des débutants aux notions de programmation. D'après Wilson et Moffat (2010), Scratch peut ainsi permettre aux débutants d'aborder plus tard d'autres langages de programmation plus difficiles.

Meerbaum-Salant et al. (2010), ont mené une étude exploratoire, auprès de deux classes de 3e de 18 et 28 élèves, afin d'étudier la capacité des élèves à apprendre des concepts informatiques à travers Scratch. Les chercheurs ont élaboré du matériel pour l'enseignement de la programmation sur Scratch à des élèves de 3^{e}. Une formation des deux enseignants a été réalisée pour les aider à mettre en place l'enseignement des concepts informatiques avec Scratch. Meerbaum-Salant et al. (2010) mettent l'accent sur le besoin des enseignants d'avoir du matériel prêt à utiliser pour réaliser un tel enseignement et de ne pas leur laisser la charge mentale de la conception des séances à la dernière minute. Il est important de noter ici que les élèves participant à cette étude n'avaient pas d'expérience préalable avec le logiciel Scratch. Les séances proposées par les chercheurs portaient entre autres sur la séquence, l'initialisation de la position de personnages, la concurrence à la fois sous la forme de plusieurs personnages et de plusieurs programmes par personnage, la répétition prédéfinie et la répétition conditionnelle, la communication et synchronisation à travers l'envoi et la réception des messages (Meerbaum-Salant et al., 2010).

Ils ont collecté les réponses des élèves à trois tests, Pré-test, test intermédiaire et Post-test, afin d'examiner le niveau de comprehension des concepts étudiés par les élèves. Ils ont aussi réalisé des entretiens avec les élèves et les enseignants. Ils ont également collecté les projets

créés par les élèves et des notes d'observation du terrain. Les résultats de cette étude montrent que les élèves se sont familiarisés avec une série de concepts informatiques à travers Scratch. Meerbaum-Salant et al. (2010) remarquent également que les élèves ont rencontré des difficultés face à l'initialisation, les variables et la concurrence. Ils supposent que les difficultés des élèves face à ces concepts sont dues au niveau d'abstraction nécessité pour la compréhension de ces concepts par les élèves. Cette recherche montre également que la répétition prédéfinie était moins bien comprise par les élèves que la répétition conditionnelle et les messages.

Maloney et al. (2008) ont mené une recherche hors du contexte scolaire, dans le cadre d'un centre d'activités extrascolaires autour de la technologie, Computer Clubhouse. Les chercheurs ont analysé 536 projets Scratch créés par des élèves de 8 à 18 ans pendant une période de 18 moins, afin d'étudier l'apprentissage des concepts informatiques par les élèves malgré l'absence d'absence d'instruction explicite. Maloney et al. (2008) remarquent que les élèves dont les programmes ont été analysés ont réussi, sans avoir suivi un curriculum spécifique sur la programmation à travers Scratch, à utiliser des commandes faisant appel à des concepts tels que l'interaction avec l'utilisateur, la répétition, la répétition conditionnelle, la communication et la synchronisation. Les auteurs soulignent que les commandes liées à des concepts tels que les variables ou les expressions booléennes étaient moins utilisés par les élèves.

Franklin et al. (2017) ont étudié les activités réalisées par 123 élèves de 9 à 12 ans, l'équivalent de CM1, CM2 et 6e en France, sur Scratch. Cette recherche a été menée dans des conditions réelles de classe. Tous les élèves participant à l'étude ont suivi le même curriculum sur la programmation en Scratch au cours de l'année. En analysant les activités réalisées par les élèves, ils n'ont pas remarqué de différence dans la performance des élèves de trois classes, face aux concepts plutôt simples. En revanche, des différences ont été identifiées concernant les concepts plus complexes. En effet, la séquence et la gestion d'événements simples semblent être facilement comprises par les jeunes élèves.

En ce qui concerne le déplacement des personnages dans l'espace de deux dimensions de Scratch, Franklin et al. (2017) mettent l'accent sur le fait que les élèves présentent des difficultés lorsqu'ils doivent réaliser des déplacements impliquant des connaissances en mathématiques. Ainsi, ils ont constaté que les élèves de 6e ont eu moins des difficultés sur

cela que ceux de CM2 et CM1. D'après les résultats de cette recherche, l'initialisation a aussi posé des difficultés principalement aux élèves de CM1 et CM2, alors que les élèves de 6e ont réussi sans problème à réaliser les taches portant sur cela.

Fatourou et al. (2018) ont aussi étudié l'apprentissage des concepts de programmation concurrente à la fois par des élèves de CM2 sans expérience préalable en programmation et des élèves de 6e, ayant un peu d'expérience dans ce domaine. L'étude a été réalisée auprès de 123 élèves de sept classes d'école primaire en Grèce. Les résultats montrent que la synchronisation de deux personnages à travers l'envoi et la réception des messages a posé la plupart de problèmes aux élèves, puisque seulement un élève sur cinq a réussi à le mettre en place de manière opérationnelle sur son projet final. En revanche, les élèves de 6e présentent une meilleure performance en ce qui concerne la synchronisation des personnages à travers les messages. Ils ont aussi observé que la majorité des élèves de CM2 et 6e ont réussi à mettre en place la concurrence à la fois sous la forme de plusieurs programmes pour le même personnage et plusieurs personnages en même temps.

Comme nous venons de le voir les environnements de programmation graphiques tels que Scratch, offrent de nombreuses possibilités pour initier les élèves de 8 ans à la programmation et aux concepts informatiques.

4. Une évolution nécessaire pour viser les plus jeunes élèves

Malgré leur syntaxe simplifiée, les langages de programmation visuels tels que Scratch, contiennent encore beaucoup de texte et des structures de programmation qui ne sont pas accessibles par les plus jeunes élèves, ce qui compromet leur utilisation dans le milieu de la petite enfance. De plus, l'existence d'études expérimentales, montrant qu'une initiation des élèves à la programmation est possible dès 4 ans, à l'aide des outils adaptés à ce groupe d'âge et à ses caractéristiques, a aussi joué un rôle important en faveur d'une évolution des langages graphiques ou visuels. Ainsi, les environnements et les langages graphiques ont évolué davantage, afin de devenir accessibles même aux plus jeunes élèves. Pour cela, le texte sur les blocs de programmation a été remplacé par des icônes, afin que ces langages puissent être utilisés même par les plus jeunes élèves, qui ne savent pas encore lire.

Plusieurs environnements de programmation ont ainsi été créés en ayant comme objectif d'initier les plus jeunes élèves à la programmation. Nous pouvons citer à titre d'exemple *ScratchJr*[114], *Daisy the dinosaur* et *Kodable*[115].

Duncan et al. (2014) soulignent une série d'éléments à prendre en compte lorsqu'on entreprend une initiation des jeunes élèves à la programmation. En effet, pour choisir ce que nous allons enseigner aux élèves et l'environnement de programmation que nous allons utiliser pour cela, il est important, d'après eux, de considérer entre autres leur âge et leur maturité, leurs capacités mathématiques et linguistiques, l'existence des ressources pour enseigner la programmation avec ce logiciel et les possibilités d'exploration proposées par l'environnement choisi.

L'apparition d'une multitude d'environnements répondant aux besoins des très jeunes élèves a déclenché une période florissante en termes d'expérimentations scientifiques dans le milieu de la petite enfance, groupe d'âge qui n'était jusqu'alors souvent concerné que par des robots programmables.

Nous pouvons citer le travail de Fessakis et al. (2013) qui ont étudié les processus de résolution de problèmes de programmation par 10 élèves de 5 à 6 ans, à l'aide de deux environnements de programmation de type LOGO, *Ladybug leaf* et *Ladybug maze*. Ces deux environnements sont adaptés à l'âge des élèves de maternelle, car les commandes existantes sont limitées à quatre (avancer, reculer, tourner à droite et tourner à gauche à 45° ou 90°) et elles ont la forme des boutons ayant des icônes, afin qu'elles soient plus facilement utilisées par les enfants.

Fessakis et al. (2013) ont montré qu'à l'aide d'un environnement adapté à l'âge des élèves, d4 activités bien pensées, de matériel approprié et d' enseignants formés à l'enseignement de la programmation, les élèves de 5 à 6 ans sont capables de résoudre de problèmes de programmation et ainsi, être initiés à des notions de programmation de base. Les chercheurs remarquent deux stratégies en particulier : *planning* ou « planification » en français et *trial and error* ou « essai-erreur » en français. Fessakis et al. (2013) notent que 6 élèves sur les 10 ont réussi de mettre en place une sorte de « planification », car ils ont écrit jusqu'à deux ou trois commandes maximum avant d'exécuter leur programme. Les élèves présentant une telle

114 https://www.scratchjr.org/
115 https://www.kodable.com/

capacité de « planification » semblaient, d'après les auteurs, concentrés et sûrs de leurs actions. En revanche, 4 élèves sur les 10 mettant en place une stratégie d' « essai-erreur », écrivaient leurs programmes commande par commande et souvent ils ne savaient pas comment procéder pour résoudre les problèmes donnés. Ces élèves avaient souvent besoin d'être assistés par l'enseignante ou par les camarades de classe.

Pila et al. (2019) ont aussi exploré la possibilité d'initier les jeunes élèves à la programmation à travers d'environnements de programmations graphiques, disponibles sous forme d'application pour tablette. Ils ont en particulier choisi deux applications : *Daisy the dinosaur* et *Kodable*. Cette recherche a été menée auprès de 28 élèves, dont l'âge moyen était 5,15 ans, ayant suivi un curriculum d'une semaine sur l'apprentissage de la programmation dans le cadre d'un camp de vacances. Les résultats de cette recherche montrent que les élèves ont été capables de programmer sur *Kodable* et *Daisy the dinosaur*. Cela prouve, d'après Pila et al (2019), d'une part que ces environnements de programmation graphique peuvent vraiment permettre une initiation des très jeunes élèves à la programmation et d'autre part, que la programmation est un type d'activité qui peut convenir à des plus jeunes élèves et ainsi servir à une première initiation à des concepts informatiques.

D'après les résultats de cette étude, les élèves ont présenté une meilleure compréhension de la séquence au Post-test qu'au Pré-test pour les deux environnements de programmation graphique. Pila et al. (2019) mettent l'accent sur l'importance de passer plus de temps avec les élèves sur des concepts plus complexes tels que la condition sur *Daisy the dinosaur*. Ils remarquent également la difficulté des élèves de maternelle à verbaliser leurs connaissances autour de ce qui est la programmation. Ils n'ont pas trouvé des différences dans la performance des élèves en programmation selon leur genre. En revanche, ils ont repéré que la préférence des élèves pour l'environnement de programmation pourrait influencer positivement la performance des élèves en programmation sur cet environnement.

Léonard et al. (2021) ont travaillé sur l'initiation d'élèves de 5 à 6 ans à la notion de la répétition, à travers un dispositif pédagogique combinant des activités débranchées, la résolution des problèmes et des activités sur l'environnement MOTIF ART. Ce dernier est un environnement de programmation adapté à l'âge des élèves, car le langage de programmation est simplifié, comportant uniquement des blocs représentant des couleurs et la commande de répétition. Dans cette étude, Leonard et al. (2021) ont inclus, non seulement des activités sur

tablette, mais aussi des activités débranchées, en se basant ainsi sur des travaux de recherche mettant en avant les bénéfices de cette approche pour l'apprentissage de la programmation par des élèves (Romero et al., 2018).

Les résultats de cette recherche montrent qu'une initiation des élèves de 5 à 7 ans à la notion de répétition est possible à travers l'identification de motifs visuels, sous la condition que le repérage dans l'espace ne soit pas nécessité par les activités réalisées (Léonard et al., 2021). Les chercheurs constatent également que les élèves de 5 à 6 ans ont des difficultés avec la répétition lorsque le motif à identifier comprend plusieurs commandes. Ce résultat revient aussi dans deux recherches précédentes portant sur l'appropriation de la notion de répétition auprès des élèves de 6 à 7 ans (Léonard et al., 2020) et de 8 à 10 ans (Peter et al., 2019).

Un certain nombre de travaux ont aussi été réalisés pour étudier la possibilité d'initier les très jeunes élèves à la programmation à travers le logiciel ScratchJr. La prochaine section sera dédiée à cet environnement de programmation et aux recherches expérimentales utilisant ce dernier.

5. L'environnement et le langage de programmation ScratchJr

Dans cette section, nous allons nous concentrer sur l'environnement de programmation ScratchJr, son langage de programmation et les objectifs derrière sa création. ScratchJr est un environnement de programmation graphique, sous forme d'application gratuite pour *Ipad*, pour tablette *Android, Amazon* et *Chromebook,* ayant comme objectif d'initier les élèves de 5 à 7 ans à la programmation. Ce logiciel est le fruit d'une collaboration entre le *DevTech Research Group* de l'Université de Tuffts sous la direction de Marina Bers, *Lifelong Kindergarden Group* du MIT sous la direction de Mitchel Resnick et *Playful Invention Company,* sous la direction de Paula Bonta et Brian Silverman (Portelance et al., 2015). Pour les créateurs de ScratchJr, la programmation est un nouveau type de littératie qui doit être accessible par tous, car elle permet de mieux organiser sa pensée et exprimer ses idées (Bers & Resnick, 2016).

Pour Marina Bers (Bers, 2018, p. 4), ScratchJr a été développé pour être « a digital playground for coding ». Ce logiciel a été créé en ayant comme objectif d'une part, d'initier

les plus jeunes élèves à la programmation et d'autre part, de permettre le développement de connaissances et de capacités utiles dans d'autres domaines et notamment la littératie ou *literacy* en anglais, les mathématiques etc (Kazakoff, 2014). Ce dernier explique dans sa thèse que ScratchJr vise trois domaines : le développement des apprentissages liés à une discipline particulière notamment les mathématiques et la littératie, le développement des apprentissages fondamentaux tels que la séquence, la causalité, l'estimation, la reconnaissance des patterns, la compréhension des symboles et le développement des capacités des élèves à résoudre de problèmes donnés.

Bers (2021) souligne que ScratchJr a été créé pour initier les élèves, à travers l'apprentissage de la programmation, à une série d'idées puissantes ou *powerful ideas,* concept central dans les travaux de Papert (1980). Bers (2021) cite en particulier : les algorithmes, la modularité, les structures de contrôle, la représentation, hardware/software, la résolution de problèmes ou *design process* et le débogage. La figure 1 présente le Tableau utilisé par Bers (2018) pour illustrer les liens existants entre ces concepts de l'informatique et la petite enfance, et l' implémentation de ces concepts sur ScratchJr.

L'auteure explique qu'en apprenant la programmation sur ScratchJr, les élèves se familiarisent avec la notion de l'algorithme, car ils réfléchissent à une série d'étapes à suivre pour résoudre un problème précis et ainsi, ils développent leur capacité d'abstraction et de séquence. Grâce à l'apprentissage de la programmation sur ScratchJr les élèves se familiarisent également avec les structures de contrôle car ils ont la possibilité de contrôler l'ordre selon lequel les personnages d'un projet déclencheront leurs programmes (événements), le nombre de fois que les personnages vont répéter des actions (répétition prédéfinie, répétition indéfinie), la position initiale d'un personnage dans la scène ou sa vitesse. Ainsi, ils développent d'après Bers (2018) la causalité et la capacité à prendre des décisions selon des conditions spécifiques, éléments utiles pour la petite enfance.

Pour elle, l'apprentissage de la programmation sur ScratchJr permet également la familiarisation des élèves avec le processus de résolution de problèmes et de débogage, des capacités utiles dans d'autres domaines et notamment les mathématiques où les élèves ont besoin de capacités de résolution de problèmes et de vérification de l'exactitude de leur solution. Les élèves ont la possibilité également de comprendre l'idée de modularité à travers la programmation sur ScratchJr, car ils ont la possibilité de séparer un programme en sous

programmes et d'utiliser des morceaux de ces derniers pour la programmer d'autres personnages. Ainsi, ils apprennent à aborder une épreuve difficile en affrontant chaque fois une petite partie de celle-ci, connaissance utile pour la petite enfance d'après Bers (2018).

En programmant sur ScratchJr les élèves se familiarisent aussi avec la représentation, car les commandes qu'ils utilisent pour programmer représentent des actions spécifiques que le personnage va réaliser et ainsi ils travaillent sur une compétence utile pour la petite enfance la reconnaissance des symboles. Les élèves sont également exposés à un logiciel, le ScratchJr, qui peut être disponible sur différents appareils, et ainsi ils ont accès à l'idée qu'un système peut être composé des différents éléments, utile aussi pour la petite enfance (Bers, 2018).

Powerful Idea	Definition	Early Childhood Connections	ScratchJr Application
Algorithm	A series of ordered instructional steps taken in a sequence to solve a problem or achieve an end goal.	Understanding abstraction and sequencing.	When children put together the colorful ScratchJr programming blocks in a logical order and create a sequence of actions (a script) for the chosen characters.
Modularity	The breaking down of tasks or procedures into simpler, manageable units that can be combined or re-used to create a more complex process.	Understanding that a complex task needs to be broken down into smaller tasks.	In ScratchJr, a common practice of modularity is to copy a portion of a script (coding sequence) from one character to another. For example, if a child wants to make a ScratchJr dance party featuring several characters, she can create a chunk of code for a dance move and copy it to multiple characters.
Control Structures	Control structures determine the order (or sequence) in which instructions are followed or executed within an algorithm or program. For example, repeat functions, loops, conditionals, events, and nested structures, are all control structures.	Understanding control structures requires an understanding of patterns and the concept of making decisions based on certain conditions as well as cause and effect.	In ScratchJr, children explore the concept of control structures by utilizing control flow blocks that allow them to create loops and repetitions as well as set different variables such as speed.
Representation	Programming languages represent information through the use of a symbol system. At the same time, computers store and manipulate data and values in a variety of ways. In order to be made available, this data is represented in different ways.	Understanding that concepts can be represented using symbols, and that programming languages are formal constructed symbol systems designed to communicate instructions (an algorithm) to a machine.	ScratchJr uses different forms of representations. Colorful blocks represent different types of commands. For example, blue blocks represent motion.
Hardware/Software	Computing systems need hardware and software to operate. The software provides instructions to the hardware, which might or might not be visible. Hardware and software work together as a system to accomplish tasks, such as receiving, processing, and sending information.	Understanding systems and their components, as well as the complex interplay between "instructions" (code) and "objects that receive those instructions".	ScratchJr is the software, the programming language that runs on different hardware devices.
Design process	This iterative process involves several steps: ask, imagine, plan, create, test, improve, and share. The process is open-ended, in that a problem may have many possible solutions.	Understanding that creating a final product to be shared with others involves several steps and continuing revising of the work.	In ScratchJr the design process starts when a child asks a question that gives birth to an idea and ends with creating a final project that can be shared with others. The design process makes computational thinking visible: coding becomes a tool of expression.
Debugging	Fixing problems through systematic analysis and evaluation, while developing troubleshooting strategies.	Learning how to debug is an important skill that is similar to "checking your work in math" or "editing" in literacy. It teaches the powerful lesson that things do not just happen to work on the first try, and that many iterations are usually necessary to get it right.	When using ScratchJr to create a personally meaningful program children naturally engage in debugging by fixing what doesn't work and problem solving.

Figure 1: Powerful Ideas from Computer Science and Connections to ScratchJr (Bers, 2018, p.5)

ScratchJr est composé d'une bibliothèque de projets, l'interface ou *main project éditor* et l'éditeur graphique offrant aux élèves des outils, à la fois pour modifier les caractéristiques d'un personnage et pour créer des nouveaux personnages et des arrière-plans (Flannery et al., 2013). L'interface (voir figure 2) est organisée en trois parties : l'espace de programmation et la palette des blocs de programmation, la scène et enfin, la partie responsable de la gestion de l'environnement (personnages, pages, mode plein-écran, grille, ajouter du texte, icône

d'initialisation de la position des personnages, drapeau-vert, enregistrement d'un projet) (Komis, Touloupaki, et al., 2017).

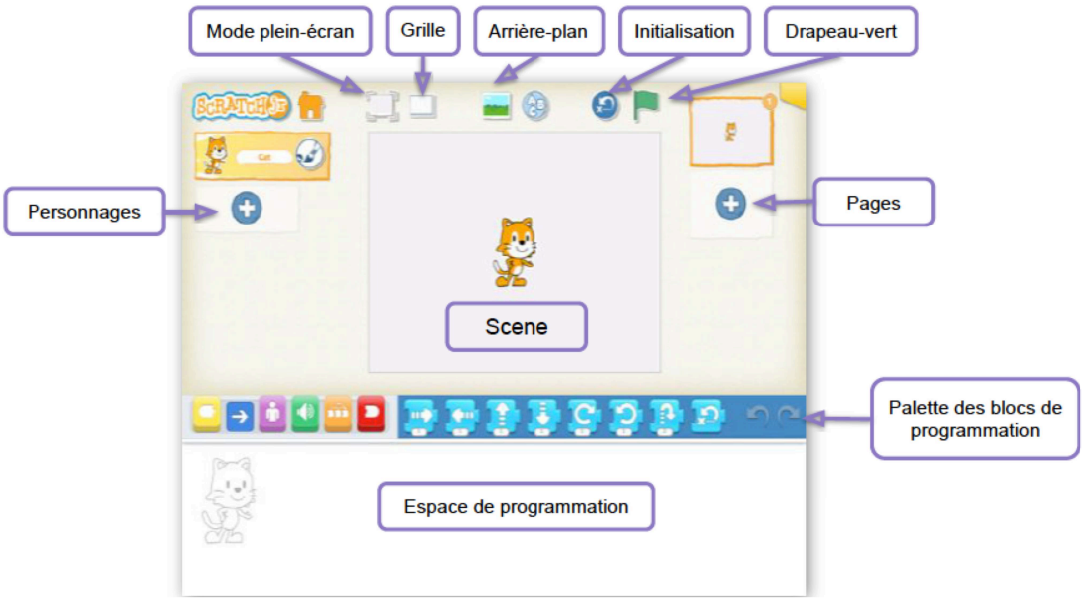


Figure 2: L'interface de ScratchJr

En se basant sur Scratch, l'équipe de ScratchJr a adapté l'interface et le langage de programmation du logiciel, afin de mieux correspondre aux besoins de très jeunes élèves (Flannery et al., 2013). Le langage de programmation est composé de blocs graphiques colorés sur lesquels les élèves peuvent cliquer et qu'ils peuvent faire glisser vers l'espace de programmation pour créer des programmes. Ainsi, l'interface de ScratchJr est de type *drag-and-drop*, comme celui de son ancêtre Scratch.

L'interface dispose une palette de blocs de programmation qui comprend six catégories différentes : les blocs jaunes permettent de décider du mode de départ du programme, les blocs bleus permettant de déplacer le personnage dans la scène, les blocs violets qui gèrent l'apparence, les blocs verts qui sont responsables du son, les blocs orange ou blocs contrôle et enfin, les blocs rouges qui sont les blocs de fin d'un programme (Bers, 2018). Les blocs sont assez larges pour que les jeunes puissent les utiliser facilement (Komis, Touloupaki, et al., 2017).

La forme des blocs de programmation est telle qui oblige l'utilisateur à les assembler correctement pour créer des programmes et ainsi, éviter les erreurs sur la syntaxe (Portelance et al., 2015). Un programme sur ScratchJr est une séquence des commandes assemblées dans

l'espace de programmation et ce dernier est considéré comme complet lorsqu'il dispose une commande de « démarrage » et une commande de « fin ».

Les programmes sur ScratchJr sont écrits en suivant la direction de l'écriture, de gauche à droite. Ce choix a été fait par les créateurs du logiciel afin de renforcer le développement de la littératie. Pour programmer sur ScratchJr, les élèves doivent d'abord, choisir un personnage, ensuite choisir des commandes ayant la forme de blocs de programmation et agencer ces derniers dans le correct ordre séquentiel, afin de pouvoir faire le personnage choisi réaliser les actions qu'ils ont définies dans la scène, pour atteindre un objectif précis (Kazakoff, 2014). Les élèves peuvent ajouter jusqu'à quatre pages différentes à leur animation, sachant que chaque page correspond à une nouvelle scène et à un nouvel espace de programmation pour que les élèves puissent créer un projet avec différentes parties (Flannery et al., 2013). Un projet sur ScratchJr correspond à l'ensemble des pages et personnages faisant partie de ces pages avec leurs programmes.

Pour exécuter une commande sur ScratchJr, il suffit que l'élève appuie sur celle-ci avec son doigt, afin de l'activer. L'exécution d'un programme, une séquence des commandes assemblées, peut aussi se faire en s'appuyant sur celui-ci avec le doigt. L'exécution d'un programme complet, ayant une commande de démarrage à son début, se fait soit avec le doigt soit avec le bouton du drapeau-vert dans la partie haute de l'interface. Les commandes s'exécutent l'une après l'autre selon leur ordre d'apparition sur le programme.

En outre, le logiciel met à la disposition des utilisateurs la « grille » (voir figure 3), afin de les aider à dénombrer et mesurer les distances à réaliser par leurs personnages dans la scène (Flannery et al., 2013). Des ressources et des activités pour une utilisation de ScratchJr dans les classes et à la maison sont également proposées sur le site du logiciel (Flannery et al., 2013).

En utilisant ce logiciel, les jeunes élèves ont la possibilité d'apprendre la programmation à travers la création des projets qui ont du sens pour eux. Ils peuvent même personnaliser leurs animations en créant leurs propres arrière-plans et personnages, en ajoutant leurs propres voix ou des sons, et même en ajoutant des photos d'eux-mêmes (Flannery et al., 2013).

Figure 3: La grille sur ScratchJr

5.1. La programmation comme objet d'apprentissage sur ScratchJr

Depuis son apparition en 2014, ScratchJr a suscité beaucoup d'intérêt. Comme nous l'avons vu plus haut, ce logiciel dispose des structures permettant de travailler avec les plus jeunes élèves sur la séquence, les événements (concurrence et messages), la répétition prédéfinie et indéfinie, mais aussi sur les variables (définition de la position initiale d'un personnage ou de sa vitesse). Plusieurs travaux de recherche ont été réalisés afin d'explorer les possibilités de cet environnement pour l'apprentissage de la programmation par des plus jeunes élèves.

Portelance et al (2015) ont été parmi les premiers à donner des résultats intéressants sur l'introduction des très jeunes élèves à la programmation sur ScratchJr. Leur groupe de chercheurs a mené une recherche auprès de 62 élèves de grande section de l'école maternelle jusqu'au CE1 (2nd grade) d'une école primaire, ayant suivi un curriculum de six semaines sur l'apprentissage de concepts de base de programmation en utilisant le logiciel ScratchJr. L'objectif de cette recherche a été d'identifier les blocs de programmation que les jeunes élèves préfèrent utiliser pour leurs projets, après avoir appris tous les blocs disponibles sur l'application. L'analyse des 977 projets conçus sur ScratchJr montre que les blocs qui sont utilisés le plus souvent par les élèves sont ceux qui sont responsables du déplacement (blocs bleus) du personnage dans la scène. Les chercheurs ont également constaté que les élèves de CE1 utilisaient moins de blocs de fin (blocs rouges) et de blocks de démarrage (blocs jaunes)

que les élèves de maternelle. Portelance et al. (2015) notent que les élèves de CE1 préféraient utiliser plus des blocks de contrôle (blocs orange) que les élèves de maternelle.

Un peu plus tard, Strawhacker et al. (2017) ont aussi réalisé une recherche auprès de 222 élèves de grande section de l'école maternelle jusqu'à CE1, de six écoles primaires (cinq privés et un publique) aux Etats-Unis. Les élèves participant à cette étude ont suivi de deux à sept séances maximum sur l'apprentissage de la programmation à travers ScratchJr. L'objectif principal de cette étude a été d'explorer le lien possible entre le style d'enseignement adapté et les résultats d'apprentissages de la programmation par les élèves. Pour cela, les chercheurs ont évalué le niveau d'appropriation de la programmation par les élèves participant à l'étude en analysant leurs réponses qu'ils ont données à une évaluation, *ScratchJr Solve its*.

Cette évaluation comprend quatre épreuves avec des objectifs différents : *Fix the program*, *Circle the blocks, Match the program et Reverse engineering.* Au cours de ces épreuves, les élèves devraient observer des projets déjà prêts en mode plein-écran et ensuite, réaliser une série d'exercices sur papier, loin du logiciel ScratchJr. Strawhacker et al. (2017) ont étudié en particulier, à travers ces épreuves, les éléments suivants : la compréhension de la séquence, la capacité des élèves à identifier les erreurs sur un programme (débogage), la capacité des élèves à associer une commande à son action (reconnaissance de symbole) et enfin, la capacité des élèves à mettre une séquence des commandes dans le correct ordre, pour atteindre un objectif précis, ce que les auteurs nomment *goal-oriented sequencing*.

Les résultats de cette recherche montrent que les élèves de toutes les écoles ont présenté une bonne performance aux épreuves données. Cela montre d'après eux qu'ils étaient capables de comprendre la séquence, d'associer une commande à son action, de déboguer des programmes et de mettre dans l'ordre correct une séquence des commandes pour atteindre un objectif visé. Les élèves les plus ages de l'échantillon présentent une meilleure performance face aux épreuves, que les élèves les plus jeunes. En se basant su ces résultats, Strawhacker et al. (2017) remarquent que les élèves de trois groupes d'âge (GS, CP, CE1) sont capables d'approprier des concepts de programmation, avec des différences selon leur âge.

Ces chercheurs identifient également trois types d'erreurs qui arrivent de façon récurrente dans leurs données. Les élèves se trompent souvent en utilisant la commande « déplacement d'un pas vers le haut » à la place de la commande « sauter ». Parfois la commande « déplacement d'un pas vers le haut » était aussi accompagnée d'une commande

« déplacement d'un pas vers le bas », mais cela n'était pas obligatoire. Strawhacker et al. (2017) ont aussi observé que la plupart des élèves de maternelle n'arrivaient pas à identifier la commande « tourner à droite » et à sa place ils choisissaient les commandes « tourner à gauche », « sauter » ou « répéter indéfiniment ».

Ils ont également repéré que, bien que les élèves réalisent des erreurs sur la séquence, peu se trompent sur la première commande de cette dernière. Les auteurs soulignent l'importance de la réalisation des recherches plus approfondies pour explorer les erreurs réalisées par les très jeunes élèves pendant la programmation (Strawhacker et al., 2017). En ce qui concerne le lien possible entre le style d'enseignement et les résultats d'apprentissage des élèves, les résultats de cette recherche montrent que la mise en œuvre des certains styles d'enseignement est corrélative d'une haute performance des jeunes élèves à la programmation sur ScratchJr.

Les enseignantes participant à la recherche ont également donné des témoignages intéressants par rapport à ce qu'elles ont observé sur leurs élèves faire. Les enseignants ont en effet remarqué qu'ils étaient tellement attirés par certains éléments de l'environnement de ScratchJr, tels que l'éditeur graphique et la commande d' « enregistrement d'un son », qu'il était parfois difficile de les faire revenir à l'activité en cours. Elles notent par exemple que les élèves passaient beaucoup de temps à modifier les caractéristiques de leurs personnages, sans écrire des programmes pour eux. Une autre enseignante a remarqué que, ses élèves de maternelle avaient des difficultés à faire le passage entre, utiliser des blocs pour voir ce que leur personnage ferra et programmer leur personnage à réaliser une tache spécifique dans la scène (Strawhacker et al., 2017).

Strawhacker et Bers (2019) ont étudié la performance en programmation sur ScratchJr de 57 élèves, de grande section de l'école maternelle jusqu'au CE1 (trois classes au total, une de chaque niveau) d'une école primaire publique, ayant introduit à la programmation sur ScratchJr pendant six semaines. L'objectif de cette recherche a été analysé les erreurs réalisées par les élèves, lors d'une évaluation de leur niveau de compréhension du langage de programmation ScratchJr, afin d'examiner si ces derniers utilisent des connaissances provenant d'autres domaines lorsqu'ils programment. Bien que la programmation en tant qu'objet d'apprentissage par les élèves ne soit pas l'objet principal de cette étude, ses résultats nous informent sur les erreurs que les élèves font lorsqu'ils programment sur ScratchJr. Pour évaluer le niveau d'appropriation de la programmation par les élèves, ces chercheurs ont

utilisé les épreuves *ScratchJr Solve its*. Pour cette étude, deux types d'épreuves ont été donnés aux élèves : *Block recognition,* où ils devraient associer une commande avec son action, et *Block sequencing,* où ils devraient créer un programme en utilisant une séquence des commandes dans le bon ordre, pour atteindre un objectif précis.

Les résultats de cette recherche montrent que les élèves les plus âgés réalisent moins d'erreurs à l'épreuve *Block recognition* que les moins âgés. En effet, les élèves de maternelle se sont trompés plus à ces épreuves que les élèves de CP et les élèves de CP se sont trompés plus à ces epreuves que ceux de CE1. Les chercheurs remarquent que les réponses étaient plus variées face à l'épreuve *Block sequencing*.

En ce qui concerne les erreurs réalisées, Strawhacker et Bers (2019) relèvent que les élèves de tous les groupes d'âges semblent maitriser les commandes de déplacement et pouvoir faire la correspondance entre ces dernières et l'action du personnage dans la scène, sauf la commande « sauter ». Cette dernière pose, d'après eux, la plupart des problèmes aux élèves de maternelle, de CP et de CE1. De plus, les chercheurs indiquent que les élèves de maternelle avaient plus des difficultés avec les commandes de contrôle tels que la « répétition prédéfinie » et « fixer la vitesse ». Ces commandes ainsi que « envoyer un message » et « quand message reçu » sont plus accessibles par les élèves de CP et de CE1. En revanche, ces auteurs soulignent qu'aucun élève de l'échantillon n'a utilisé à la fois la commande « envoyer un message » et la commande « quand message reçu ». Les élèves de CE1 ont toutefois présenté une meilleure performance face aux commandes de « messages » que les élèves de CP.

Les résultats de cette recherche sont similaires à ceux présentés par Flannery et al. (2013), portant sur les premières observations de l'utilisation des versions pilotes du logiciel ScratchJr avec des très jeunes élèves. Flannery et al. (2013) remarquent aussi que les élèves de CP et de CE1 présentent une meilleure performance en programmation sur ScratchJr que les élèves de maternelle. Les chercheurs ont identifié une série d'éléments qui ont posé des difficultés à tous les élèves et notamment ceux de la maternelle. En effet, les éléments qui posent les plus des difficultés d'après eux sont : les commandes qui n'ont pas des effets visibles suite à leur exécution, le changement et la programmation des différents personnages, la gestion des plusieurs programmes pour le même personnage ou pour des personnages différents,

l'utilisation des paramètres pour atteindre un objectif, l'effacement des personnages, l'utilisation d'un vocabulaire précis pour décrire les éléments d'interface.

Nous pouvons également citer une recherche sur les effets de la programmation en ScratchJr dans la consolidation des apprentissages mathématiques chez les élèves de CE1 (7 à 8 ans) dans une école primaire française en Bretagne, proche d'une zone d'éducation prioritaire (Vigot, 2018). Bien qu'elle ne porte pas directement sur l'apprentissage de la programmation par les jeunes élèves, elle met en avant une série d'éléments liés à cette activité que nous jugeons importants à citer. Les 22 élèves participant à cette étude ont découvert ScratchJr dans le cadre d'un projet pluridisciplinaire. Ils devaient travailler en groupe de deux pour résoudre, à la fois trois défis de déplacement d'un personnage sur ScratchJr et une situation problème portant cette fois-ci sur un déplacement à réaliser à l'aide de la « répétition prédéfinie ». Vigot (2018) a voulu observer comment les élèves utilisent leurs connaissances mathématiques pour programmer sur ScratchJr. Les connaissances mathématiques que les élèves doivent utiliser pour réaliser les trois défis sont : l'usage du nombre, la réduction d'une écriture additive, l'addition réitérée et la connaissance du rectangle.

Cinq stratégies en lien avec la programmation du déplacement des personnages sur ScratchJr ont été identifiées. Ce sont les suivantes : « stratégie de segmentation du programme », « stratégie de l'ensemble du programme », « stratégie de l'estimation », « stratégie des petits-pas » et « stratégie des essais-erreurs » ou « stratégie des ajustements ». Vigot (2018) explique que les élèves qui réduisent l'écriture de leur codage en testant une partie de ce dernier mettent en place une « stratégie de segmentation du programme », alors qu'ils existent des élèves qui programment l'ensemble de leur programme. D'après Vigot (2018), il existe des élèves qui commencent leur programmation en faisant une estimation du nombre des pas dont ils auront besoin, en mettant ainsi en place une « stratégie de l'estimation ». D'autres utilisent une stratégie que Vigot (2018) nomme « stratégie des petits pas », car ils programment leur personnage « flèche après flèche » (p.6). L'auteure explique également que les élèves qui commencent leur programme avec une « stratégie de l'estimation » corrigent leur programme en utilisant une « stratégie des essais-erreurs » ou « stratégie des ajustements ».

Une série des difficultés ont été observées chez certains élèves de CE1. Elle remarque des difficultés en ce qui concerne le déplacement d'un personnage à l'aide des paramètres vers un endroit spécifique dans la scène, en raison d'une consolidation partielle de la notion du

nombre. Ces élèves arrivent mieux à déplacer leurs personnages en utilisant plusieurs fois la même « flèche » que d'utiliser un numéro pour représenter le nombre des « fleches » nécessaires. Certains présentent également des difficultés face à l'utilisation des commandes « tourner à droite » et « tourner à gauche ». Elle observe également que les programmes des élèves ont souvent une « flèche en plus » ou une « flèche au moins »et regrette le fait que le temps d'expérimentation au terrain n'ait pas été suffisant pour que les élèves puissent résoudre le problème portant sur la « répétition prédéfinie ». Enfin, elle constate que les connaissances mathématiques acquises préalablement par les élèves sont mobilisées lors de la programmation.

Tsourapi et al. (2018) ont réalisé une étude portant sur l'étayage mis en place par des enseignantes d'école maternelle, afin d'accompagner les très jeunes élèves à l'apprentissage de la programmation sur ScratchJr. Les auteurs remarquent que les élèves de maternelle ont eu besoin d'aide pour résoudre les problèmes de programmation proposés lors de la séance d'évaluation. Ils identifient également une série d'éléments en lien avec la programmation qui nécessitent davantage d'accompagnement de la part des enseignantes, participant à l'étude. D'après eux, les élèves ont eu plus des difficultés à vérifier l'exactitude de leur programmation (étape nécessaire dans le processus du débogage) et à initialiser la position de leur personnage. Ils ont également eu des difficultés à se rappeler d'ajouter la commande « quand drapeau-vert cliqué » et la commande de « fin » à leur programme. Les éléments d'interface n'ont pas nécessité beaucoup de soutien de la part des enseignants (Tsourapi et al., 2018).

6. Problématique et questionnements de recherche

Après avoir identifié un intérêt grandissant autour de l'apprentissage de la programmation par les plus jeunes élèves en raison de l'apparition des langages de programmation graphiques ou visuels adaptés à leur âge, nous nous sommes lancés dans la réalisation d'une revue de la littérature existante sur le sujet.

Nous avons donc constaté que l'apprentissage de la programmation par les enfants était un sujet qui avait beaucoup préoccupé les chercheurs dans le passé. L'activité de programmation

a été pendant longtemps considérée comme très complexe pour être accessible par les enfants. Dès la moitié des années 1960, Seymour Papert Wallace Feurzeig, Daniel Bobrow, Richard Grand, Cynthia Solomon et Franck Frazier au Département de la Technologie Éducative de la firme de recherche de Cambridge Bolt Beranek et Newman n'étaient pas d'accord avec cette idée. Toutefois, reconnaissant que les systèmes disponibles à cette époque n'étaient pas appropriés pour initier les enfants à la programmation, ils ont créé LOGO, le premier langage de programmation destinés aux enfants. L'objectif initial d'environnement LOGO a été d'initier les élèves aux notions de mathématiques de façon plus concrète, à travers l'apprentissage de la programmation.

Seymour Papert et ses collègues ont défendu l'idée que les élèves pourraient apprendre à programmer en utilisant l'environnement LOGO et que cet apprentissage pourrait avoir des effets sur le développement des compétences de haut niveau tels que la résolution des problèmes et la planification, utiles dans d'autres domaines. Ils ont aussi mis l'accent sur l'importance de l'apprentissage par la découverte.

Un grand nombre des travaux de recherche ont été menés utilisant le langage de programmation LOGO. Ces travaux avaient pour objectif d'enseigner la programmation aux enfants, ainsi que d'utiliser la programmation en tant qu'outil pour développer des compétences dans d'autres domaines (Mendelsohn et al., 1990).

En examinant les premiers travaux portant sur l'apprentissage de la programmation en tant qu'objet d'apprentissage pour les élèves, nous pouvons constater qu'elles étaient principalement réalisées avec des élèves plus âgés en raison du caractère textuel du langage LOGO. Les chercheurs se sont intéressés particulièrement aux stratégies de programmation employées par les élèves (Papert et al., 1979), aux stades par lesquels passent les élèves lorsqu'ils apprennent à programmer (Howe, 1980), aux difficultés qu'ils rencontrent (Pea, 1983), aux erreurs qu'ils réalisent (Mendelsohn, 1985) et au débogage qu'ils mettent en place (Carver & Khlar, 1986).

En ce qui concerne les travaux sur les apports de l'apprentissage de la programmation à d'autres domaines, certains chercheurs étaient plus optimistes (Clements & Gullo, 1984) que d'autres (Pea & Kurland, 1985). En résumé, les résultats de ces travaux montraient que les élèves avaient des difficultés à apprendre à programmer à l'aide des langages textuels tels que LOGO en raison de la complexité de leur syntaxe. Peu des travaux, par rapport à la décennie

précédente, ont été réalisés pendant les années 1990 sur des problématiques similaires en utilisant LOGO.

Suite à cette période de première fondation, des nouveaux robots programmables et des nouveaux langages de programmation graphiques ont émergé, donnant ainsi lieu à une nouvelle période d'expérimentations scientifiques dans le milieu de l'apprentissage de la programmation par les enfants. Parallèlement, la publication en 2006 de l'article de Jeanette Wing intitulé *Computational Thinking* ou *Pensée Informatique* en français, a bouleversé la communauté scientifique et éducative, car elle a mis à l'ordre du jour une notion longtemps oubliée, en lui donnant une nouvelle définition. Ainsi, l'apprentissage de la programmation a de nouveau été associé au développement d'un type de pensée utile dans d'autres domaines.

Plusieurs travaux de recherche ont été réalisés au cours du temps pour explorer les potentielles utilisations des robots programmables par des très jeunes enfants et les effets de ces dernières sur l'apprentissage de la programmation et sur d'autres apprentissages. Les objets de recherche de ces travaux étaient similaires que celles de la période de première fondation. En effet, les chercheurs ont étudié ce que les très jeunes élèves peuvent faire en programmation et leurs difficultés (Flannery et Bers, 2013), les stratégies de programmation et le débogage mis en place (Komis & Misirli, 2013) ou la capacité des très jeunes élèves à aborder des concepts de programmation tels que la séquence ou la répétition (Strawhacker & Bers, 2014).

De plus, l'apparition des langages de programmation visuels ou graphiques tels que Scratch ont permis une initiation des élèves de 8 ans à l'apprentissage de la programmation et à des structures de programmation plus complexes. La syntaxe simplifiée et l'environnement de type *drag-and-drop* sont des atouts pour l'initiation des élèves de cet âge-là à la programmation. Des recherches ont été réalisées pour étudier les difficultés rencontrées par les élèves (Fatourou et al., 2018) et les stratégies de programmation utilisées (Meerbaum-Salant et al., 2011). En revanche, malgré leurs avantages, les langages de programmation visuels tels que Scratch, contenaient encore beaucoup de texte et des structures de programmation qui n'étaient pas accessibles par les plus jeunes élèves, ce qui compromettait leur utilisation dans le milieu de la petite enfance.

Pour répondre à ce besoin, les environnements et les langages graphiques ont évolué davantage, afin de devenir accessibles même aux plus jeunes élèves. Pour cela, le texte sur les

blocs de programmation a été remplacé par des icônes, afin que ces langages puissent être utilisés même par les plus jeunes élèves, qui ne savent pas encore lire. Des travaux ont été réalisés sur les stratégies de programmation employés par les très jeunes élèves (Fessakis et al. (2013) ou sur l'initiation des jeunes élèves à la programmation en utilisant aussi des activités débranchées (Leonard et al. 2021).

ScratchJr est un très réussi exemple de cette évolution car, en disposant des structures de programmation adaptées à leur âge, il permet l'initiation de très jeunes élèves à des notions de programmation peu étudiées à cet âge-là. Un certain nombre des travaux de recherche ont été réalisés afin d'explorer les possibilités de cet environnement pour l'apprentissage de la programmation par des plus jeunes élèves. Ces derniers traitent les types commandes qui posent des problèmes aux très jeunes élèves en fonction de leur âge (Portelance et al. 2015), mais elles ne nous disent pas quels types de problèmes elles posent. D'autres recherches montrent que les élèves sont capables de programmer ou même déboguer leur programme sur ScratchJr (Strawhacker et al. 2017), mais elles ne nous donnent pas des informations par rapport à ce que les élèves font lorsqu'ils programment, quelles sont les stratégies de programmation et le débogage qu'ils mettent en place.

Des recherches explorent les commandes utilisées et non utilisées par les très jeunes élèves en fonction de leur âge en examinant parallèlement les connaissances issues d'autres domaines qu'ils mobilisent pour le faire (Strawhacker et al, 2019), mais elles ne nous donnent pas des informations sur ce que les très jeunes élèves font lorsqu'ils les utilisent pour programmer sur ScratchJr. Une recherche étudie principalement les stratégies de déplacement que des élèves plus âgés (CE1) mettent en place lorsqu'ils programment sur ScratchJr (Vigot, 2018), mais nous n'avons pas des informations sur les stratégies de programmation employées par les très jeunes élèves.

Nous pouvons donc constater que nous n'avons pas des informations sur ce que les très jeunes élèves font lorsqu'ils programment sur ScratchJr, les stratégies de programmations, le débogage qu'ils mettent en place et quelles sont leurs difficultés.

C'est pourquoi nous avons choisi de réaliser une recherche exploratoire pour comprendre ce que les élèves de 5 à 7 ans sont capables de faire seuls ou aidés en programmation sur ScratchJr et quelles difficultés ils risquent de rencontrer. Nous avons voulu en particulier étudier les stratégies de programmation et le débogage qu'ils mettent en œuvre. L'originalité

de cette recherche et son apport à la littérature se situent donc dans l'éclairage qu'elle apporte à l'étude de ses différentes questions.

Pour rappel, pour programmer sur ScratchJr, les élèves doivent d'abord, choisir un personnage, ensuite choisir des commandes ayant la forme de blocs de programmation et agencer ces derniers dans l'ordre séquentiel correct. Ces actions sont à réaliser dans le but de faire faire au personnage choisi des actions définies dans la scène, pour atteindre un objectif précis (Kazakoff, 2014).

Puisque nous n'avons pas pu trouver de modèle théorique décrivant toutes les facettes de l'activité de programmation notamment au niveau de la petite enfance, nous avons décidé de combiner plusieurs modèles théoriques pour étudier ce que les jeunes élèves font seuls ou aidés lorsqu'ils programment pour résoudre des problèmes sur ScratchJr.

C'est pourquoi nous avons décidé d'utiliser un modèle théorique portant sur les phases de l'activité de programmation, des modèles théoriques portant sur les stratégies de programmation mises en place par des jeunes élèves et un modèle théorique sur le débogage réalisé par ces derniers.

Pour les phases de programmation, nous avons choisi le modèle proposé par Pea et Kurland (1984). Selon ce dernier, l'activité de programmation comprend quatre phases : comprendre le problème de programmation, planifier une solution pour le résoudre, écrire le programme mettant en œuvre ce plan, comprendre et déboguer le programme écrit. Puisque nous sommes intéressée à ce que font les élèves lorsqu'ils programment, nous avons choisi de nous focaliser sur deux phases en particulier de l'activité de programmation des élèves : l'écriture du programme et le débogage.

Concernant les stratégies de programmation, nous avons fait le choix de combiner trois modèles théoriques : le travail de Komis et Misirli (2013) sur les stratégies utilisées par des élèves de 4 à 6 ans pour la programmation d'un robot programmable, le travail de Fessakis et al. (2013) sur les stratégies de programmation utilisées par des élèves de 5 à 6 ans pour la programmation sur un environnement de programmation graphique de type LOGO adapté à leur âge et le travail de Vigot (2018) sur les stratégies en lien avec la programmation du déplacement des personnages sur ScratchJr utilisées par les élèves de 7 à 8 ans.

Komis et Misirli (2013) ont identifié quatre stratégies de programmation principales, dont les trois « totale », « pas à pas » et « sous-programme » contiennent une verbalisation de l'algorithme et une écriture de l'algorithme à l'aide des cartes représentant les commandes de programmation avant la programmation du robot. La quatrième, celle de l'« essai-erreur » est la stratégie mise en place par l'élève qui ne réalise ni une verbalisation de l'algorithme ni une écriture du programme à l'aide des cartes avant de programmer le robot programmable.

Fessakis et al. (2013) remarquent deux types de stratégies utilisées par des élèves de 5 à 6 ans lors de la programmation sur un environnement de programmation graphique de type LOGO : « planification » et « essaie-erreur ». Fessakis et al. (2013) notent que les élèves mettant en place une stratégie de « planification », arrivent à écrire jusqu'à deux ou trois commandes maximum de leur programme. Les élèves mettant en place une stratégie d' « essai-erreur », écrivent une commande et l'exécute, écrivent une autre commande et l'exécute.

Vigot (2018) identifie cinq stratégies en lien avec la programmation du déplacement des personnages sur ScratchJr mises en place par les élèves de 7 à 8 ans. Ce sont les suivantes : « stratégie de segmentation du programme », « stratégie de l'ensemble du programme », « stratégie de l'estimation », « stratégie des petits-pas » et stratégie des essais-erreurs » ou « stratégie des ajustements ».

Ce travail nous a particulièrement inspirée pour deux raisons. La première était le fait que l'auteur associait les stratégies identifiées à des descriptions détaillées de déroulements d'activité des élèves, un élément qui n'était pas présent dans les travaux consultés auparavant. La deuxième raison était la proximité entre ce travail et ce que nous faisons dans notre étude, car nous n'avons pas pu trouver une autre étude traitant les stratégies mises en place par les élèves sur ScratchJr.

En ce qui concerne le débogage, nous avons décidé d'utiliser le modèle proposé par Klahr et Carver (1988). Les auteurs définissent le débogage en tant que processus qui comprend cinq phases : 1) Evaluation du programme, 2) Identification d'erreur, 3) Représentation du programme, 4) Localisation de l'erreur et 5) Correction de l'erreur.

Dans la Partie 3 qui suit, nous présenterons les choix méthodologiques que nous avons réalisés pour mettre en place cette recherche et pour analyser les données collectées, afin de pouvoir répondre à notre problématique. Nous expliciterons en particulier comment nous nous

sommes servie de cette combinaison de modèles théoriques pour analyser l'activité de programmation des élèves.

Partie 3. Méthodologie

Chapitre 1. Élaboration du programme de recherche

Dans ce chapitre, nous présentons nos choix méthodologiques. Nous décrirons d'abord notre méthodologie générale, ensuite, nous expliciterons la stratégie de recherche que nous avons adoptée et enfin, nous mettrons en évidence le modèle utilisé pour la conception des scénarios pédagogiques.

1. Choix de la méthodologie générale

Notre objectif en réalisant cette recherche a été d'explorer ce qui était possible de faire en programmation sur ScratchJr avec des très jeunes élèves dans des conditions réelles de classe. En effet, nous avons voulu comprendre ce que les élèves de 5 à 7 ans sont capables de faire, seuls ou aidés, les stratégies de programmation qu'ils utilisent, les difficultés qu'ils risquent de rencontrer, le débogage et les patterns de débogage qu'ils mettent en œuvre. Compte tenu de la complexité de l'activité de programmation, une étude approfondie de l'activité de programmation des élèves était indispensable pour répondre à notre problématique.

Ces éléments nous ont conduite à suivre une méthodologie qualitative, car elle nous a paru la démarche la plus adaptée à la nature de notre problème de recherche. En effet, le choix d'une telle méthodologie nous a permis d'explorer et de comprendre profondément le phénomène étudié (Merriam, 2002), ici l'activité de programmation des jeunes élèves sur le langage de programmation ScratchJr.

Il nous a semblé qu'une approche descriptive de ce que font les élèves serait une contribution nouvelle à la littérature sur le sujet. Nous avons commencé notre immersion dans le terrain avec un questionnement général celui de la problématique et des questionnements plus précis. Ce choix méthodologique a ainsi guidé le mode de recueil des données : le processus de collecte, la taille des échantillons ainsi que les méthodes d'analyse des données.

2. Une étude de cas multiples

Après avoir choisi de suivre une méthodologie qualitative, nous avons ensuite cherché l'approche la plus adaptée à notre travail. Nous avons donc opté pour l'étude de cas ou *case study* en anglais.

L'étude de cas est envisagée de différentes manières dans la littérature. Yin (2003) la définit en tant que stratégie de recherche. Stake (2003) indique qu'elle n'est pas un choix méthodologique mais plus tôt un choix de ce qu'on va étudier. Creswel (2007) la perçoit en tant que méthodologie, « a research design » (p. 73), qu'on traduirait en français par « modèle de recherche » [notre traduction]. Merriam (2002), enfin la présente aussi en tant que modèle ou stratégie de recherche.

Dans le contexte de cette thèse, l'étude de cas est une stratégie de recherche. Elle peut être mise en place sur des données qualitatives, quantitatives ou même sur une combinaison des deux (Yin, 2003). La recherche de cette thèse suit une orientation purement qualitative. Merriam et Tisdell (2015) notent que les études de cas ayant une telle orientation ont des caractéristiques en commun avec les autres types de recherches qualitatives. En effet, elles cherchent aussi à comprendre le phénomène étudié et elles suivent une démarche inductive. De plus, le chercheur est chargé du recueil et de l'analyse des données et leurs résultats sont descriptifs.

Pour Yin (2003, p. 13), l'étude de cas est « une recherche empirique qui étudie un phénomène contemporain dans son contexte de vie réelle, en particulier quand les frontières entre le phénomène et son contexte ne sont pas clairement évidentes » [notre traduction]. Selon cet auteur, le cas est l'unité d'analyse et sa définition est directement liée aux questions de recherche. Stake (2003, p. 134) met en avant « ce qui va être étudié, le cas » [notre traduction]. Il explique que l'étude de cas est à la fois la recherche autour du cas mais aussi le fruit de cette recherche. Merriam (2002) définit l'étude de cas comme une description et analyse profonde d'un phénomène délimité.

Dans le contexte de cette thèse, nous nous appuyons sur la définition de Merriam (2002) car pour répondre à notre problématique nous avons besoin d'analyser et comprendre profondément l'activité de programmation des très jeunes élèves sur ScratchJr. Étant donné que l'étude de cas est d'après cet auteur une stratégie de recherche qui permet de comprendre et analyser en profondeur le phénomène étudié dans un contexte délimité, elle nous a paru la plus pertinente pour mener ce travail.

Miles et al. (2014, p. 44) définissent le cas « comme un phénomène qui se produit dans un contexte délimité » [notre traduction]. Un cas peut prendre des formes différentes, il peut par exemple être un élève, une classe, une école, une institution ou une communauté (Merriam &

Tisdell, 2015). Nous pouvons aussi étudier un seul ou plusieurs cas, un cas pouvant même comprendre des sous-cas (Yin, 2003).

De plus, Miles et al. (2014), recommandent de réaliser des études de cas multiples, car cette pratique permet de mieux comprendre et expliquer le phénomène étudié. En effet, lorsque le chercheur travaille avec plusieurs cas en même temps, il a la possibilité d'identifier plus facilement les similarités et les différences existantes entre les cas choisis. De plus, quand un même résultat ressort de deux cas au lieu d'un seul, le chercheur peut l'examiner de façon plus détaillée pour repérer quelles sont les conditions dans lesquelles cela a émergé, mais aussi pourquoi.

En outre, Miles et al. (2014) soulignent qu'en réalisant des études avec de cas multiples, le chercheur peut apporter aux résultats de sa recherche une plus grande « généralisabilité ou transférabilité » (p.101). Les auteurs expliquent que même si ce n'est pas une des priorités des études qualitatives, les chercheurs qui suivent cette méthodologie peuvent toutefois avoir comme objectif que leurs résultats puissent être reproduits dans un contexte similaire à celui dans lequel ils ont émergé. La réalisation des études avec de cas multiples leur donne la possibilité de le faire.

En nous appuyant sur les recommandations de Miles et al. (2014), nous avons choisi d'étudier deux cas provenant de deux écoles maternelles différentes pendant la phase de l'étude préparatoire en Grèce et deux cas provenant de deux écoles primaires différentes pendant la phase de l'étude principale en France. Les deux cas correspondent à deux classes de grande section de maternelle en Grèce et deux classes de cours préparatoire en France. Chacun de ces quatre cas comprend des sous-cas, des élèves, que nous avons étudiés leurs comportements différemment en fonction des phases de l'étude.

En prenant en compte notre objectif d'alimenter la littérature sur la programmation des très jeunes élèves à l'aide d'un langage graphique comme ScratchJr, nous avons opté pour de multiples cas pour plusieurs raisons. La première était d'avoir la possibilité de faire de comparaisons entre les deux cas de chaque contexte et ainsi obtenir une vision plus globale sur le phénomène étudié. La deuxième raison était de pouvoir renforcer la validité de nos résultats et en particulier celle des stratégies de programmation et du débogage réalisé par les élèves. Enfin, la troisième raison était de pouvoir procéder par triangulation avec l'utilisation

des différentes « sources (personnes, temps, lieux etc) »[notre traduction] (Denzin, 2001[116], cité dans Miles et al., 2014, p. 262).

Enfin, l'étude de cas permet aux chercheurs de construire des modèles ou des théories à partir du phénomène étudié, pour aider ensuite la communauté scientifique à comprendre des situations ou des phénomènes qui ressemblent à celui sur lequel ces théories ou modèles ont été basés (Robson, 2002[117] cité dans Cohen et al., 2007).

3. Le modèle suivi pour la conception des scénarios pédagogiques

Pour pouvoir étudier l'activité de programmation des très jeunes élèves sur ScratchJr et ainsi répondre à notre problème de recherche, il a d'abord fallu enseigner la programmation aux élèves. Pour cela, nous avons eu besoin de concevoir deux scénarios pédagogiques. C'est pourquoi nous avons utilisé le modèle de conception des scénarios pédagogiques proposé par Komis et al. (2013). Ce modèle porte sur la conception et l'implémentation des scénarios pédagogiques qui impliquent l'utilisation des technologies éducatives.

Un scénario pédagogique peut enseigner un ou plusieurs concepts d'une ou de plusieurs disciplines différentes. Sa mission est de spécifier l'ensemble des séances à mettre en place, le matériel à utiliser, les stratégies didactiques à employer et les aides didactiques à apporter pour atteindre l'objectif ou les objectifs visés (Komis et al., 2013). Ces auteurs soulignent l'importance d'adapter le contenu du scénario pédagogique à l'âge et aux caractéristiques du public visé. D'après ce modèle, il existe une série des phases (voir figure 4) à suivre pour concevoir un scénario pédagogique (Komis et al., 2013).

116Denzin, N. K. (2001). Interpretive interactionism (2nd ed.). Thousand Oaks, CA: Sage.
117Robson, C. (2002) Real World Research (2nd ed.).Oxford: Blackwell.

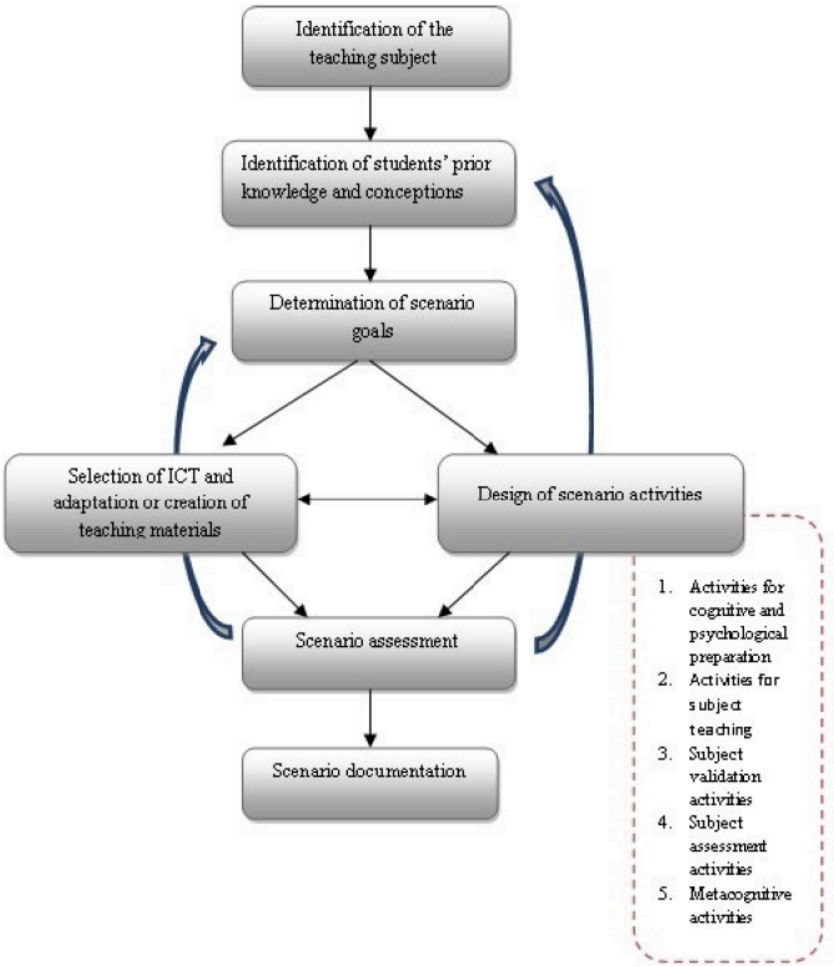


Figure 4: Les phases différentes pour la conception des scénarios pédagogiques en utilisant les TIC (Komis et al., 2013, p.3247)

Dans un premier temps, il convient de choisir le sujet global de ce scénario, le niveau des élèves concernés, la ou les disciplines concernées et les prérequis nécessaires. Ensuite, il est recommandé d'identifier les idées initiales des élèves en ce qui concerne le contenu à enseigner. Après ces deux premières étapes, il faut préciser ce que nous allons enseigner à l'aide de ce scénario, c'est-à-dire les objectifs d'enseignement. Il est important aussi de décrire le matériel nécessaire pour la réalisation du scénario, aussi bien didactique que technologique.

L'étape qui suit est la plus importante dans ce processus de conception, car elle concerne la création des séances, autrement dit le cœur du scénario. Komis et al. (2013) expliquent que les séances d'un scénario peuvent être de cinq types différents : séances de préparation cognitive et psychologique, séances d'enseignement, séances de validation, séances d'évaluation et séances méta-cognitives. La durée de chaque séance, son positionnement dans l'emploi du

temps de l'école, ainsi que l'organisation de la classe au cours de celle-ci sont aussi des éléments à définir à ce moment-là de l'élaboration du scénario (Komis et al., 2013).

Après avoir réfléchi au contenu des séances de notre scénario, il est important de définir les stratégies didactiques à utiliser dans notre scénario (Komis et al., 2013). Les stratégies didactiques peuvent être de différentes sortes. Nous en citons quelques-unes : expérimentation, découverte, résolution de problèmes et résolution collaborative de problèmes, conflit cognitif, etc. Les aides didactiques doivent être aussi précisées dans les séances du scénario. Ces dernières peuvent prendre différentes formes. Elles peuvent être du matériel que l'enseignant a conçu pour enseigner quelque chose ou des questions que l'enseignant pose à ses élèves ou encore des informations que l'enseignant fournit à ses élèves.

L'étape suivante du modèle proposé par Komis et al. (2013) porte sur l'évaluation du scénario créé, après sa mise en place. Enfin, la dernière étape concerne la documentation du scénario, c'est-à-dire des commentaires sur la mise en place du scénario, des suggestions pour l'amélioration de celui-ci ainsi que des conseils pour les enseignants qui auront envie de le mettre en place dans le futur.

Chapitre 2. L'étude préparatoire en Grèce

Dans ce chapitre nous présentons la phase d'étude préparatoire que nous avons réalisée dans deux classes d'écoles maternelles publiques de la ville de Patras, en Grèce en 2017. Cette étude préparatoire nous a permis de tester notre scénario pédagogique auprès d'élèves de grande-section, afin de l'améliorer pour le mettre en œuvre dans des écoles en France pendant l'étude principale. Elle nous a aussi permis de mieux organiser la méthodologie de mise en place de notre recherche pour notre terrain de l'année suivante en France. Grâce à cette étude préparatoire, nous avons également pu obtenir une vision globale sur ce que les élèves de grande section de maternelle peuvent faire seuls ou aidés en programmation sur ScratchJr et les difficultés qu'ils risquent de rencontrer (voir Chapitre 1 de la Partie 4).

L'étude préparatoire a duré deux mois et elle a été mise en place auprès de 22 élèves de grande section, dix élèves de grande section 1 (GS1) et 12 élèves de grande section 2 (GS2). Elle s'est déroulée durant les mois d'avril et mai 2017. Il est important de signaler que trois élèves de la GS1 avaient déjà fait de la programmation sur ScratchJr l'année précédente et deux autres élèves en avaient également fait le semestre précédent notre étude préparatoire. Dans la classe GS2 aucun élève n'avait d'expérience préalable sur ScratchJr mais ils avaient réalisé une séance sur la programmation du robot programmable Thymio.

En raison de difficultés d'accès aux écoles en Grèce, nous avons dû retarder notre intervention et faire en deux mois ce qui était prévu initialement en trois. Il a fallu adapter notre planning et notre scénario pédagogique aux nouvelles disponibilités des enseignants. En effet, nous avons choisi de réaliser un scénario pédagogique de 11 séances au lieu des 13 initialement prévues. Nous avons aussi dû diminuer le nombre des élèves participant à notre étude, pour nous adapter à la contrainte du temps réduit. Le retard pris lors de la mise en place de notre expérience en Grèce a eu des répercussions sur les données que nous avons recueillies pendant l'évaluation car nous n'avons pas eu le temps de la réaliser individuellement pour chaque élève. Nous expliciterons davantage ce point-là dans la section 3 de ce chapitre.

Les enseignantes de deux classes n'ont pas participé à l'étude préparatoire. Puisque nous étions seule sur le terrain nous avons pris en charge : la mise en place du scénario pédagogique dans les classes, l'accompagnement des élèves dans ce processus mais aussi l'enregistrement vidéo des séances. Les parents des élèves ont donné leur accord écrit et signé pour nous autoriser à inclure leurs enfants dans notre projet de recherche et à les enregistrer en vidéo.

1. Présentation du scénario pédagogique mis en place en Grèce

Un scénario pédagogique de 11 séances a été conçu afin de pouvoir initier les élèves de grande section à la programmation sur ScratchJr. Pour le concevoir, nous avons dans un premier temps examiné les possibilités d'exploration proposées par l'environnement choisi, ici l'environnement de programmation ScratchJr. Nous avons également pris en compte les spécificités du groupe d'âge visé, mais aussi les observations préliminaires d'une étude exploratoire que nous avons menée dans le cadre de notre Master 2 sur l'initiation à la programmation des élèves de cours préparatoires à travers ScratchJr (Touloupaki et al., 2018).

Nous détaillons ici l'organisation de deux classes pendant la mise en place du scénario. La première classe a été divisée en trois sous-groupes d'élèves (deux groupes de trois et un groupe de quatre). Cela a permis de travailler en simultané avec l'enseignante, lorsqu'elle faisait cours à deux sous-groupes, nous avons pris en charge le troisième sous-groupe. Nous avons donc travaillé avec les trois sous-groupes à tour de rôle.

Pour la seconde classe, et pour nous adapter aux imprévus liés au temps alloué à cette étude dans la classe, nous avons décidé de travailler avec une partie seulement des élèves de classe. Nous avons aussi divisé les élèves participant à notre étude en trois sous-groupes (quatre élèves par groupe) pour travailler en simultané avec l'enseignante de la classe. Toutefois, il est à noter que dans les deux classes, nous étions dans la même pièce que le reste du groupe et que par conséquent, cela a pu gêner ou distraire certains élèves pendant la réalisation des séances du scénario.

Pour mieux animer les séances, l'écran de notre tablette a été projeté à l'aide d'un mini vidéo-projecteur, pour que les élèves puissent voir ce qui se passe dans notre projet, sur ScratchJr. Chaque élève avait sa propre tablette pour réaliser les activités proposées par le scénario pédagogique élaboré à ce titre. Chaque séance avec un groupe d'élèves a duré 45 minutes.

Ce scénario comprend trois types de séances : les séances de préparation cognitive et psychologique qui servent à familiariser les élèves avec le logiciel et la programmation sur ScratchJr (2), les séances de l'enseignement et de validation (7) et les séances d'évaluation de la performance des élèves (2). Les stratégies didactiques utilisées dans ce scénario sont la

découverte, le tâtonnement expérimental et la résolution de problèmes. Nous avons également encouragé les élèves à interagir lors de la mise en place du scénario, pour échanger des idées et ainsi mieux avancer dans la résolution de problèmes proposés. Les aides didactiques apportées sont de trois types : les cartes utilisées pour l'enseignement de la « répétition » et des « messages », les questions posées par la chercheuse et le rappel de la consigne du problème.

1.1. Séances de préparation cognitive et psychologique

Nous avons commencé notre étude de terrain par un entretien semi-directif (**Pré-test**) réalisé individuellement avec chaque élève. Les objectifs de cet entretien ont été : 1) d'identifier les idées initiales des élèves à propos de l'interface et surtout des commandes que nous souhaitions leur enseigner dans les prochaines séances et 2) de prendre en compte les idées et besoins pour améliorer les séances d'enseignement du scénario.

Nous avons aussi dédié une séance à la **familiarisation** des élèves avec la programmation, à la tablette et à l'interface de ScratchJr. Pendant cette séance, nous avons réalisé une activité débranchée, le jeu d'enfant-robot, afin d'initier les élèves à l'idée de la programmation, du programme, de la commande et de la séquence avant de passer sur le logiciel ScratchJr. Au cours de ce jeu, les élèves ont travaillé en binôme où l'un était le programmeur et l'autre la machine. L'un des élèves devait programmer l'autre en lui donnant des commandes de vive voix afin qu'il se déplace d'une position initiale à une position finale dans un quadrillage sur le sol. Les élèves pouvaient ensuite changer de rôles. Au cours de ce jeu, nous avons utilisé deux cartes, une verte et une rouge, que nous avons données aux élèves pour qu'ils puissent signaler le début et la fin de la programmation à leur camarade enfant-robot.

Nous avons aussi introduit les élèves à l'utilisation de la tablette, en apprenant les processus du verrouillage et du déverrouillage. À la fin de cette séance, nous avons dédié un peu de temps à la manipulation de la tablette et à une première initiation au logiciel ScratchJr.

1.2. Séances d'enseignement et de validation

Ensuite, nous sommes passées aux **séances d'enseignement et de validation**. Au cours de la *première séance* nous avons exploré comment : ouvrir l'application, démarrer un nouveau projet, sélectionner un nouvel arrière-plan, insérer et effacer des commandes de

programmation, assembler les commandes de programmation pour créer des programmes, définir le point de départ d'un personnage, initialiser la position d'un personnage, déplacer le personnage à gauche et à droite dans la scène, mais aussi le processus de l'enregistrement d'un projet. Nous avons expliqué la signification des commandes pour le personnage et nous avons initié les élèves au principe « début-fin » de programmation (séquence). Nous avons également expliqué aux élèves l'exécution séquentielle des commandes sur ScratchJr en dirigeant leur attention au fait que celles-ci clignotent l'une après l'autre lorsqu'elles sont exécutées. Nous avons également présenté aux élèves l'importance de vérification de l'exactitude de leur programmation.

Pendant la *deuxième séance*, nous avons enseigné comment : introduire un nouveau personnage dans la scène, programmer deux personnages, déplacer le personnage en haut et en bas dans la scène, enregistrer un son et l'effacer, aller en plein-écran, utiliser l'icône du drapeau-vert en haut de la scène pour exécuter un projet. Nous avons aussi expliqué aux élèves le lien existant entre le bouton du drapeau-vert en haut de la scène et la commande du « drapeau-vert ».

Au cours de la *troisième séance*, nous avons présenté comment : effacer un personnage et faire sauter un personnage. À la fin de cette séance, nous avons proposé un problème aux élèves, le même pour tous, afin qu'ils puissent utiliser les éléments vus dans ces premières séances.

Les deux séances suivantes ont été dédiées à la commande de « répétition prédéfinie ». Au cours de la *quatrième séance*, nous avons travaillé sur la répétition d'une seule commande un nombre de fois prédéfini et au cours de la *cinquième séance*, nous avons travaillé sur la répétition d'une séquence de commandes un nombre de fois prédéfini. La première partie des séances était dédiée à l'enseignement de la commande de « répétition » (répétition d'une commande et d'une séquence des commandes un nombre de fois prédéfini) à travers un processus de découverte. Une étude pilote, mise en place précédemment, nous a aidée à comprendre que les élèves ont besoin de suivre un raisonnement inductif, pour comprendre la fonctionnalité générale de la commande de « répétition » (Terrain de Master 2). En d'autres termes, ils doivent faire des essais avec des commandes de natures différentes, telles que les commandes de « déplacement » et du « son » afin de pouvoir observer l'effet de la commande de « répétition ». Dans la figure 5, nous présentons les commandes utilisées par les élèves,

afin de découvrir ce que fait la commande de répétition. La deuxième partie était dédiée à la résolution de problèmes en utilisant cette nouvelle commande.

Figure 5: Les commandes utilisées pour aider les élèves à comprendre la fonctionnalité de la commande de répétition

Pour enseigner la répétition d'un motif de commandes, nous avons utilisé un projet prêt à l'emploi avec un motif de commandes répétées trois fois et nous avons demandé aux élèves de l'observer et d'identifier le motif répété. Ensuite, nous leur avons demandé de trouver ce qu'ils devraient faire pour reproduire le même projet mais en utilisant moins de commandes. De cette manière, nous avons réussi à introduire aux élèves pendant cette séance l'utilisation de la commande de « répétition » pour répéter un motif des commandes.

Afin de faciliter la compréhension de cette commande par les élèves, nous avons utilisé une partie du matériel proposé par le site ScratchJr, c'est-à-dire les images des commandes du logiciel à imprimer que nous avons transformées en cartes. Ensuite, nous avons utilisé ces cartes pendant la *quatrième* et la *cinquième séance* d'enseignement, afin d'aider les élèves à identifier le motif des commandes qui a été répétée et compter le nombre de fois que ce motif a été répété sur le programme proposé, pour arriver à construire le même programme à l'aide de la « répétition ».

La *sixième* et la *septième* séance d'enseignement ont été consacrées aux commandes de « messages ». Les observations d'une étude précédente (Touloupaki et al., 2018), ont révélé qu'une utilisation des projets déjà prêts contenant les commandes des « messages », pourrait

contribuer à un meilleur enseignement des « messages ». En effet, nous avons laissé les élèves expérimenter en utilisant ces projets prêts tout en les accompagnant avec des questions dans ce processus de découverte.

Pour aider encore plus les élèves à comprendre le sens des commandes de « messages », nous avons à nouveau utilisé les cartes représentant les commandes du logiciel (voir figure 6). Les élèves ont eu la possibilité de les utiliser pour créer et observer en même temps les programmes de deux personnages communiquant par messages, vu que le logiciel ne permet pas une telle visualisation.

Figure 6: Les cartes utilisées

1.3. Séances d'évaluation

Par la suite, nous sommes passés **aux séances d'évaluation de la performance des élèves**. Nous avons réalisé *deux séances d'évaluation* à la fin du scénario pédagogique. Au cours de la première séance, nous avons de nouveau mené un entretien individuel (**Post-test**) avec les élèves. Le premier objectif de cet entretien était de leur donner la possibilité de s'exprimer à propos des éléments de l'interface et surtout des commandes étudiées dans le scénario, afin de comprendre ce qui leur posait problème. Ceci nous a permis d'améliorer les séances d'enseignement du scénario, afin d'éviter que ces problèmes persistent l'année suivante, au cours de l'étude principale en France.

Pendant la seconde séance d'évaluation, nous avons planifié de donner aux élèves deux problèmes à résoudre individuellement, les mêmes pour tous. En revanche, l'organisation de cette séance d'évaluation a été ajustée suite au retard pris lors de la mise en place de notre

expérience de terrain. En effet, au lieu de la réaliser individuellement pour chaque élève nous avons dû la réaliser avec les élèves en groupe.

L'objectif de cette seconde séance d'évaluation a été de voir comment les élèves de grande section de l'école maternelle étaient capables de résoudre, seuls ou aidés, deux problèmes de programmation en utilisant le langage ScratchJr, après avoir suivi notre scénario pédagogique. Tous les deux problèmes portaient sur la programmation de deux personnages. Le premier problème (PB1) concernait entre autres les commandes de « messages ». Le deuxième problème (PB2) concernait entre autres la commande de « répétition ».

Pour résoudre ces problèmes, les élèves ont été invités à regarder une animation en mode plein-écran, accompagnée de la consigne des problèmes donnés lue par nous, étant donné que les élèves de maternelle ne sont pas encore en mesure de lire, et reproduire le même projet sur leur tablette. En d'autres mots, les élèves devaient observer le projet donné, identifier les commandes appropriées et ensuite, les utiliser dans l'ordre correct, afin de reproduire les comportements observés, pour les personnages participant au projet.

Notre intérêt était de faire de la recherche mais aussi d'intervenir pour faciliter la progression des élèves, car nous avons choisi de mener cette étude dans des contextes dits écologiques, c'est-à-dire des classes réelles au sein des écoles. C'est pourquoi, nous avons choisi d'intervenir aussi pendant la séance de l'évaluation pour aider les élèves à avancer, même si on peut penser a priori que ça pourrait gêner la posture du chercheur. Une autre raison pour laquelle nous avons choisi d'intervenir même pendant l'évaluation était l'impact positif que notre intervention pourrait avoir sur la motivation des élèves à accomplir les épreuves. En effet, sans aide apportée de notre part, il était possible que les élèves ne parviennent pas du tout à poursuivre l'évaluation, ce qui nous aurait privée de résultats. De plus, comme il s'agit des enfants de 5 à 6 ans, le risque aurait été qu'ils perdent confiance en eux voire qu'ils perdent patience et arrêtent l'activité alors même qu'ils étaient capables de la continuer avec de l'aide. Nous nous sommes donc adaptée à notre public. De plus, puisque l'objectif de cette évaluation n'était pas de juger les élèves mais de juger ce qu'ils peuvent faire seuls ou aidés à un moment donné, nous avons jugé que ça serait bénéfique de les aider même pendant l'évaluation.

2. Choix des terrains et échantillon de l'étude préparatoire en Grèce

Comme nous l'avons expliqué auparavant, nous avons réalisé une étude de cas multiples pour répondre à notre problème de recherche. En ce qui concerne l'étude préparatoire en Grèce, nous avons choisi d'étudier deux cas (Miles et al., 2014). Il s'agissait de deux classes de grande section (GS1 et GS2) provenant de deux écoles maternelles publiques de la ville de Patras en Grèce. Nous avons fait un tel choix pour avoir la possibilité de vérifier si nous retrouvons les mêmes résultats d'un cas à l'autre. Cela nous permettrait de renforcer leur validité (Miles et al., 2014). De plus, en étudiant deux cas nous pouvons détecter plus facilement leurs similarités ou leurs différences et de cette manière mieux comprendre et expliquer le phénomène étudié (Miles et al., 2014).

En outre, en étudiant deux classes différentes nous avons procédé par triangulation avec l'utilisation des différentes « sources (personnes, temps, lieux etc) » [notre traduction] (Denzin[118], 2001, cité dans Miles et al., 2014, p. 262). La triangulation est une méthode utilisée pour confirmer les résultats d'une recherche.

Puisqu'il était impossible de trouver des classes déjà équipées de tablettes en Grèce, nous avons dû procéder de la même manière qu'avec le projet DALIE. En effet, nous avons apporté les tablettes de l'Université de Patras.

Par la suite, nous avons précisé ce que nous avons étudié dans chaque cas, c'est-à-dire dans chaque classe, en réalisant ainsi un « within-case sampling » (Miles et al., 2014, p. 47). De cette manière, nous avons défini davantage les limites du contexte autour du phénomène étudié. Puisque nous avons décidé d'utiliser une méthodologie qualitative nous avons choisi de travailler avec des petits échantillons. Nous avons donc mené notre étude auprès de 22 élèves de grande section, dix élèves de GS1 (4 filles et 6 garçons) et 12 élèves de GS2 (6 filles et 6 garçons).

Afin d'obtenir un échantillon représentatif des différents niveaux présents dans les classes, nous avons constitué, à l'aide des enseignantes, des groupes mixtes composés d'élèves en réussite dans la plupart des apprentissages scolaires et d'élèves ayant davantage de difficultés

118 Denzin, N. K. (2001). *Interpretive interactionism* (2nd ed.). Thousand Oaks, CA: Sage.

avec ces mêmes apprentissages. Dans la mesure où nous avons décidé d'encourager les interactions entre les élèves au sein des groupes, nous avons composé ces derniers de façon à éviter des distractions liées aux relations[119] entre les élèves.

Pour conclure, en poursuivant les recommandations de Miles et al., (2014), notre intérêt a été de comprendre ce qui se passe dans chaque cas mais aussi d'aller plus loin dans notre compréhension du phénomène à étudier grâce à la comparaison de deux cas.

3. Collecte des données en Grèce

En ce qui concerne le recueil des données, nous avons dû procéder différemment que ce qu'on avait initialement prévu pour la phase de l'étude préparatoire pour plusieurs raisons. Étant donné que les enseignantes de deux classes n'ont pas participé à notre étude, nous avons dû prendre en charge la mise en place du scénario pédagogique dans les classes, l'accompagnement des élèves dans ce processus mais aussi l'enregistrement vidéo des séances. Nous avons donc opté pour une observation participante (Creswell, 2012).

Par conséquent, ces nombreuses responsabilités nous ont empêchée d'effectuer un protocole d'observation strict. Cela fait partie des contraintes que nous avons eu à respecter pour mener notre étude préparatoire. En revanche, nous avons pris des notes pour garder une trace de ce que les élèves arrivaient à faire et de ce qui leur posait problème. Nous avons aussi pris des notes par rapport au scénario pédagogique et sa mise en place afin de pouvoir l'améliorer pour l'étude principale en France.

Du fait des multiples rôles que nous avons eus pendant la réalisation de l'étude préparatoire, nous avons opté pour la collecte des enregistrements vidéos (Creswel, 2012) des séances d'enseignement et des séances d'évaluation. Cette méthode de collecte des données permet au chercheur d'enregistrer tout ce qui se passe sur le terrain quand ce dernier est chargé de nombreuses responsabilités (Lodico et al., 2006). De cette manière, nous avons pu nous concentrer sur notre terrain et consulter les enregistrements vidéos du processus plus tard sans perdre des informations.

119 Dans le cadre d'une classe, les enseignants remarquent souvent que les relations entre certains élèves ont des effets sur les apprentissages et c'est pourquoi ils choisissent de ne pas les mettre ensemble dans des groupes.

Comme nous l'avons expliqué auparavant, à la fin du scénario nous avons proposé deux problèmes à résoudre aux élèves. L'objectif de cette évaluation a été de voir ce que les élèves de cet âge-là sont capables de faire, seul ou aidés, en programmation sur ScratchJr à la fin du scénario pédagogique.

Le retard pris lors de la mise en place de notre expérience de terrain a également affecté la collecte des données pendant la séance de l'évaluation. Nous avons dû ajuster l'organisation de la dernière séance d'évaluation, car nous n'avons pas eu le temps de la réaliser individuellement pour chaque élève. Nous avons donc fait le choix de la réaliser avec les élèves en groupe. Néanmoins, cette décision a eu des répercussions sur les données que nous avons recueillies pour l'évaluation pour plusieurs raisons.

Tout d'abord, le fait que nous devions passer avec la caméra par chaque élève du groupe pour filmer ses actions, nous a empêchée de capturer en vidéo la totalité de l'activité de programmation des élèves. Nous n'avons donc pu obtenir que des segments de leur activité. Cela ne nous permettait pas d'étudier la succession des actions de programmation de la majorité des élèves de notre échantillon et ainsi d'étudier de façon détaillée leurs stratégies de programmation et leur débogage. Les seuls élèves pour lesquels nous avons eu des informations complètes sont ceux qui ont eu la caméra GoPro accrochée à leur casquette. Toutefois, même pour ces trois derniers, les données collectées ne correspondent pas toujours aux attendus que nous avions avant de commencer l'étude. En effet, les élèves étaient dans la même pièce que leurs camarades et parfois s'arrêtaient de résoudre leur problème pour interagir avec leurs camarades ou étaient interrompus par leurs voisins. Il s'agit d'un comportement normal d'enfant en classe, qui a quand même affecté la qualité des données collectées.

En revanche, nous avons pu collecter les réponses des élèves aux entretiens semi-directifs (Creswel, 2012) réalisés dans le cadre du scénario pédagogique (Pré-test et Post-test). En effet, nous avons collecté les enregistrements audio des Pré-test, parce que nous avons voulu identifier les idées initiales des élèves à propos de l'interface et des commandes étudiées dans le scénario. Notre objectif a été de les prendre en compte pour améliorer les séances d'enseignement du scénario pédagogique. Concernant les enregistrements audios de Post-test, nous avons voulu repérer ce qui posait problème aux élèves afin d'améliorer les séances d'enseignement de l'étude principale en France.

Nous avons aussi recueilli les programmes élaborés par les élèves suite à la réalisation de l'activité d'évaluation, afin de pouvoir étudier leurs productions finales. Nous avons voulu d'une part, savoir si les élèves arrivent ou non à résoudre les deux problèmes et d'autre part, étudier les erreurs existantes sur leurs productions finales.

4. Traitement des données de l'étude préparatoire en Grèce

S'agissant d'une étude préparatoire qui a permis de tester notre scénario pédagogique et la méthodologie de mise en place de notre recherche, les données recueillies en Grèce sont très riches et n'ont pas toutes été utilisées dans la thèse, mais pourront l'être dans des recherches futures.

Nous avons choisi d'analyser des données collectées dans l'une des deux classes de grande section que nous avons observée dans la ville de Patras, en Grèce. Pour rappel, dans la GS1 certains élèves avaient une expérience préalable l'année précédente et l'année en cours avec ScratchJr tandis que dans la GS2 aucun élève n'avait eu une expérience scolaire avec ce logiciel. Nous pensions a priori que les élèves de la GS1 réussiraient mieux l'expérience que ceux de la GS2. Or, après lectures des résultats nous avons constaté que les élèves de la GS2 réussissaient mieux les activités proposées, lors de la mise en place du scénario, que les élèves de GS1. De plus, même si dans les deux classes nous étions dans la même pièce que le reste du groupe, les conditions de mise en place de notre recherche étaient meilleures dans la GS2 que dans la GS1, car nous étions moins interrompus pendant la mise en place du scénario pédagogique. Pour ces raisons, il nous a semblé pertinent de choisir la GS2 pour nos analyses.

Pour comprendre ce que les élèves de GS2 sont capables de faire, seuls ou aidés, en programmation sur ScratchJr et les difficultés qu'ils rencontrent, nous disposions de plusieurs types des données. Nous avons analysé les notes d'observation du terrain pour étudier ce que les élèves arrivaient à faire et ce qui leur posait problème en ce qui concerne la gestion des éléments d'interface et les commandes enseignées. Nous avons aussi analysé les programmes créés par les élèves pendant la séance d'évaluation pour comprendre s'ils arrivaient à utiliser de manière opérationnelle les éléments d'interface et les commandes enseignées.

Pour la commande de « répétition prédéfinie », nous avons également analysé ce que disaient les élèves à la fin du scénario pédagogique, car il s'agit d'une commande qui leur a posé beaucoup de difficultés pendant le scénario pédagogique. Pour cette raison, nous nous sommes penchée sur les réponses que les élèves avaient données aux questions du Post-test qui concernent la commande de « répétition ».

Pour les commandes de « messages », nous avons analysé les notes prises lors de l'observation du terrain et les programmes élaborés par les élèves à la fin du scénario pédagogique. Nous avons décidé de visualiser les enregistrements vidéos des séances d'enseignement de ces commandes pour deux groupes d'élèves de GS2, pour enrichir nos observations. De la même manière que pour la commande de « répétition », nous avons étudié ce que disaient les élèves au sujet des commandes de « messages » à la fin du scénario pédagogique car ce sont des commandes qui leur ont posé beaucoup de difficultés pendant le scénario pédagogique. C'est la raison pour laquelle nous avons aussi pris en compte les réponses que les élèves de GS2 ont données à l'une des questions du Post-test portant sur les « messages ».

Chapitre 3. L’étude principale en France

Dans ce chapitre nous présentons l’étude principale que nous avons réalisée dans deux classes de cours préparatoire (CP) de deux écoles primaires publiques de la ville de Puteaux, en France. Il s’agissait de deux classes déjà équipées en tablettes. L’étude a duré quatre mois et elle a été mise en place auprès de 50 élèves de cours préparatoire, 26 élèves (12 filles et 14 garçons) de cours préparatoire (CP1) et 24 élèves (13 filles et 11 garçons) de cours préparatoire (CP2). Elle s’est déroulée durant les mois de mars, avril, mai et juin 2018.

Contrairement à la phase de l’étude préparatoire en Grèce, nous avons pu bénéficier de plus de temps avec les élèves sur le terrain français. Cela nous a permis de tester une version améliorée du scénario pédagogique que nous avons mis en place en Grèce. En effet, nous avons pu réaliser un scénario pédagogique de 13 séances au lieu de 11 et ainsi dédier une séance de plus à l’enseignement des commandes de « répétition » et de « messages ».

D’après nos observations de terrain (phase de l’étude préparatoire en Grèce et terrain du Master en France), ce sont ces deux commandes qui posent le plus de problèmes aux jeunes élèves et deux séances d’enseignement ne suffisent pas pour leur enseignement. Nous avons également pu réaliser la séance d’évaluation de notre scénario pédagogique individuellement pour chaque élève.

Les enseignantes nous ont reçue dans leurs classes pour mener notre étude à condition qu’on les forme sur l’utilisation du logiciel ScratchJr et qu’on les accompagne à la mise en place de notre scénario pédagogique. Nous avons accepté ces deux conditions et nous avons procédé tout de suite à la formation des enseignantes.

Pour les former, nous leur avons envoyé des ressources sur le logiciel ScratchJr et les commandes de programmation étudiés dans le scénario. Nous avons également fait deux rendez-vous avec elles, pour organiser la mise en place du projet et répondre à leurs questions. Ensuite, nous avons procédé par séance. Avant chaque séance, nous avons prévu du temps pour faire un point avec les enseignantes des classes sur le déroulement de la séance. Les objectifs de ces discussions étaient de répondre aux questions des enseignantes, faire un

rappel des principaux points de chaque séance et les informer des difficultés que les élèves pourraient rencontrer, afin de mieux les préparer pour aider les élèves à les surmonter.

Nous avons aussi fait une série d'adaptations à notre scénario pédagogique afin de le rendre plus utile pour les enseignantes. En effet, nous avons ajouté des captures d'écran de l'interface du logiciel ScratchJr, afin de mieux illustrer les éléments utilisés pour chaque séance. Nous avons aussi séparé les séances du scénario en étapes afin de faciliter leur suivi par les enseignantes. Pour les séances portant sur les commandes de « répétition » et des « messages », nous avons rajouté au scénario une explication détaillée des concepts de programmation associés à celles-ci.

Les enseignantes des deux classes ont suivi le scénario, mais elles ont aussi eu la liberté de réaliser des modifications quand elles les jugeaient nécessaires, car elles connaissaient mieux que nous leurs élèves. Au cours du processus, les enseignantes donnaient leur avis sur les séances, en nous apportant ainsi une aide précieuse.

De plus, étant donné que l'enseignement du scénario pédagogique a été pris en charge par les enseignantes, nous ne pouvions maîtriser complètement la réalisation du scénario pédagogique. Ainsi, nous n'avons pas eu la main sur les modalités d'enseignement.

Pour cette phase de l'étude, nous avons eu à nouveau plusieurs responsabilités. Nous avons pris en charge l'entretien semi-directif (Pré-test et Post-test) réalisé individuellement avec chaque élève, les séances d'évaluation de la performance des élèves, l'accompagnement de la classe et des enseignantes dans ce processus et l'enregistrement vidéo des séances.

Les parents des élèves ont donné leur accord écrit et signé pour nous autoriser à inclure leurs enfants dans notre projet de recherche et à enregistrer les séances.

1. Présentation du scénario pédagogique mis en place en France

Un scénario pédagogique de 13 séances a été conçu selon le modèle proposé par Komis et al. (2013), afin de pouvoir initier les élèves de cours préparatoire à la programmation sur ScratchJr.

Pour concevoir le scénario de l'étude principale en France, nous nous sommes basée sur le scénario que nous avions mis en place en Grèce pendant la phase de la pré-étude. Nous l'avons amélioré en prenant en compte nos observations du terrain en Grèce. Nous l'avons aussi adapté à l'âge des élèves, car pour cette phase de l'étude nous avons travaillé avec des élèves de 6 à 7 ans, tandis qu'en Grèce nous avons travaillé avec des élèves de 5 à 6 ans.

Nous détaillons ici l'organisation de deux classes pendant la mise en place du scénario en France. En effet, nous n'avons pas pu procéder de la même manière dans les deux classes car les deux contextes étaient différents.

Pour la première classe (CP1), nous avons travaillé avec la classe entière en même temps. Pour la deuxième classe (CP2), nous avons dû séparer la classe en deux groupes. Pour cette classe, nous avons fait d'abord la séance avec le premier groupe et ensuite avec le deuxième. Nous avons pris avec l'enseignante la décision de travailler de cette manière, car elle nous a expliqué que cette année elle avait une classe « plutôt difficile ». En effet, cette classe comportait selon elle des élèves « en difficulté » ainsi que des élèves ayant un « comportement inadapté pour l'école ». Elle nous a donc proposé de séparer la classe en deux groupes, pour pouvoir mieux travailler les séances du scénario avec ses élèves. Cette année-là, l'enseignante avait une stagiaire qui a pu prendre en charge l'un des groupes pendant que nous travaillions (l'enseignante et la chercheuse) le scénario pédagogique avec l'autre moitié du groupe.

Nous avons proposé à l'enseignante de CP1 d'adopter la même configuration pour sa classe mais cela n'a pas été possible, cette dernière n'ayant pas de stagiaire ni la possibilité de faire prendre en charge l'autre moitié de sa classe. Nous avons bien sûr respecté sa décision.

Les deux classes étaient équipées d'un tableau numérique interactif (TNI). Pour mieux animer les séances d'enseignement, au lieu d'avoir un mini-vidéoprojecteur comme en Grèce, nous avons connecté la tablette des enseignantes avec le TNI de leur classe. De cette manière, l'écran de la tablette des deux enseignantes a été projeté sur le TNI tout au long des séances du scénario pédagogique. Cela a énormément facilité le déroulement des séances d'enseignement.

Les élèves ont travaillé par groupe de quatre, avec quatre tablettes, pour réaliser les activités proposées par le scénario pédagogique.

Nous avons constitué, à l'aide des enseignantes, des groupes mixtes composés d'élèves en réussite dans la plupart des apprentissages scolaires et d'élèves ayant davantage de difficultés avec ces mêmes apprentissages. Dans la mesure où nous avons décidé d'encourager les interactions entre les élèves au sein des groupes, nous avons composé ces derniers de façon à éviter des distractions liées aux relations[120] entre les élèves. Chaque élève avait sa propre tablette pour réaliser les activités proposées par le scénario pédagogique. Chaque séance a duré de 1 heure à 1 heure et 30 minutes.

Ce scénario pédagogique comprend aussi trois types de séances : les séances de préparation cognitive et psychologique qui servent à familiariser les élèves avec le logiciel et la programmation (2), les séances de l'enseignement et validation (7) et les séances de l'évaluation de la performance des élèves (2). Les stratégies didactiques utilisées dans celui-ci sont la découverte, le tâtonnement expérimental et la résolution de problèmes. Les aides didactiques apportées sont de trois types : les cartes utilisées pour l'enseignement de la répétition et des messages, les questions posées par la chercheuse et le rappel de la consigne du problème.

1.1. Séances de préparation cognitive et psychologique

Nous avons commencé notre étude de terrain par un entretien semi-directif (**Pré-test**) réalisé individuellement avec chaque élève. Les objectifs de cet entretien ont été : 1) d'identifier les idées initiales des élèves à propos de l'interface, notamment sur les commandes que nous souhaitions leur enseigner dans les prochaines séances et 2) d'adapter les séances d'enseignement à ce que nous apprenons des élèves pendant ces entretiens semi-directifs.

Nous avons aussi dédié une **séance à la familiarisation** des élèves avec la programmation, la tablette et l'interface de ScratchJr. Pendant cette séance, nous avons réalisé une activité débranchée, le jeu d'enfant-robot, afin d'initier les élèves à l'idée de la programmation, du programme, de la commande et de la séquence avant de passer sur le logiciel ScratchJr. Au cours de ce jeu, les élèves ont travaillé en binôme où l'un était le programmeur et l'autre la machine. L'un des élèves devait programmer l'autre en lui donnant des commandes de vive voix afin qu'il se déplace d'une position initiale à une position finale dans un quadrillage sur

120 Dans le cadre d'une classe, les enseignants remarquent souvent que les relations entre certains élèves ont des effets sur les apprentissages et c'est pourquoi ils choisissent de ne pas les mettre ensemble dans des groupes.

le sol. Les élèves pouvaient ensuite changer de rôles. Au cours de ce jeu, nous avons utilisé deux cartes, une verte et une rouge, que nous avons données aux élèves pour qu'ils puissent signaler le début et la fin de la programmation à leur camarade enfant-robot.

Pour aider les élèves à distinguer la droite et la gauche, nous avons écrit sur le dos de leur main avec un feutre : un D sur la main droite et un G sur la main gauche. Nous avons aussi introduit aux élèves l'utilisation de la tablette, en leur apprenant les processus de verrouillage et de déverrouillage. À la fin de cette séance, nous avons dédié un peu de temps à la manipulation de la tablette et à une première initiation au logiciel ScratchJr.

1.2. Séances d'enseignement et de validation

La **séance de familiarisation** a été suivie par les **séances d'enseignement et de validation**. Au cours de la *première séance* nous avons exploré comment : ouvrir l'application, démarrer un nouveau projet, sélectionner un nouvel arrière-plan, insérer et effacer des commandes de programmation, assembler les commandes de programmation pour créer des programmes, définir le point de départ d'un personnage, initialiser la position d'un personnage, déplacer le personnage à gauche et à droite le personnage dans la scène, mais aussi le processus de l'enregistrement d'un projet. Nous avons expliqué la signification des commandes pour le personnage et nous avons initié les élèves au principe « début-fin » de programmation (séquence). Nous avons expliqué aux élèves l'exécution séquentielle des commandes sur ScratchJr en dirigeant leur attention sur le fait que celles-ci clignotent les unes après les autres lorsqu'elles sont exécutées. Nous avons également présenté aux élèves l'importance de vérification de l'exactitude de leur programmation.

Pendant la *deuxième séance*, nous avons enseigné comment : introduire un nouveau personnage dans la scène, programmer deux personnages, déplacer le personnage en haut et en bas dans la scène, enregistrer un son et l'effacer, aller en plein-écran, utiliser le bouton du drapeau-vert en haut de la scène pour exécuter un projet. Nous avons aussi expliqué aux élèves le lien existant entre l'icône du drapeau-vert en haut de la scène et la commande du « drapeau-vert ».

Au cours de la *troisième séance*, nous avons présenté comment : effacer un personnage et utiliser les paramètres. À la fin de cette séance, nous avons proposé aux élèves un problème de même nature à résoudre, pour pouvoir utiliser les éléments que nous avions vus jusque-là.

Les trois séances suivantes ont été dédiées à la commande de « répétition prédéfinie ». Au cours de la *quatrième séance*, nous avons travaillé sur la répétition d'une seule commande un nombre de fois prédéfini. Au cours des *cinquième et sixième séances,* nous avons travaillé sur la répétition d'une séquence de commandes un nombre de fois prédéfini. La première partie des séances était dédiée à l'enseignement de la commande de « répétition » (répétition d'une commande et d'une séquence des commandes un nombre de fois prédéfini) à travers un processus de découverte. La deuxième partie était dédiée à la résolution de problèmes en utilisant cette nouvelle commande.

Pour enseigner la « répétition », nous avons procédé de la même manière que pendant la phase de l'étude préparatoire en Grèce, en poursuivant un raisonnement inductif. En effet, nous avons donné aux élèves un projet déjà prêt où ils pouvaient tester des commandes de types différents, afin de pouvoir observer l'effet de la commande de « répétition » sur les actions du personnage. Dans la section dédiée au scénario de la phase d'étude préparatoire en Grèce, nous présentons (voir figure 5) les commandes utilisées par les élèves, afin de découvrir ce que fait la commande de « répétition ».

De plus, dans l'étude principale en France pour mieux aider les élèves à découvrir la fonctionnalité de la commande de « répétition », nous avons décidé de leur présenter la « grille » (voir figure 7). L'enseignante de la classe a utilisé la « grille » dans le processus de découverte de la fonctionnalité de la commande de « répétition ». En effet, la « grille » a servi à accompagner les élèves le temps qu'ils cherchent avec l'aide de leur enseignante la fonctionnalité de la répétition. Elle leur a permis de mieux suivre le déplacement du personnage et ainsi de comprendre la fonctionnalité de la commande de « répétition ». Grâce à la « grille », les élèves ont pu identifier la différence entre le déplacement du personnage sans la « répétition » et le déplacement du même personnage quand on ajoute la « répétition » et qu'on modifie son paramètre. De cette manière, ils ont pu saisir ce que fait la « répétition ».

Figure 7: La grille sur ScratchJr

Pendant la phase de l'étude préparatoire en Grèce, nous avons constaté qu'une des raisons pour lesquelles les élèves avaient du mal à repérer la fonctionnalité de la commande de répétition lors du processus de découverte était la vitesse de déplacement du personnage dans la scène. En effet, le personnage se déplaçait si vite que les élèves n'avaient pas le temps de l'observer et de remarquer qu'il se déplaçait davantage grâce à l'ajout de la « répétition ».

Pour résoudre ce problème, nous avons décidé pour l'étude en France de créer un deuxième programme (voir figure 8) pour le personnage « chat ». Ce programme était caché et il servait à ralentir la vitesse de déplacement du personnage « chat », car il s'exécutait en même temps que le programme du personnage que les élèves pouvaient voir sur leur espace de programmation.

Figure 8: Le programme caché qui sert à ralentir la vitesse de déplacement du chat

Pour enseigner la répétition d'un motif de commandes, nous avons choisi de commencer à nouveau avec un projet déjà prêt. Il s'agissait d'un projet avec un motif de commandes répété trois fois. L'enseignante de la classe a demandé aux élèves de l'observer et d'identifier le motif de commandes qui est répété. Ensuite, elle leur a demandé de trouver ce qu'ils devraient faire pour reproduire le même projet mais en utilisant moins de commandes que celles utilisées dans le projet modèle. De cette manière, nous avons introduit aux élèves l'utilisation de la commande de répétition pour répéter un motif de commandes.

Pour faciliter la compréhension de cette commande par les élèves, nous avons à nouveau utilisé une partie du matériel proposé par le site ScratchJr, c'est-à-dire les images des commandes du logiciel à imprimer et nous avons fabriqué des cartes. Ensuite, nous avons utilisé ces cartes pendant la *cinquième* et la *sixième séance* d'enseignement, afin d'aider les élèves à identifier la séquence de commandes qui a été répétée et compter le nombre de fois que cette séquence a été répétée sur le programme proposé, pour arriver à construire le même programme à l'aide de la répétition.

Les trois séances suivantes d'enseignement ont été consacrées aux commandes de « messages ». La *septième séance* était dédiée à l'enseignement des commandes de « messages » (envoyer un message, quand message reçu) à travers un processus de découverte. Les *huitième* et *neuvième séances* étaient dédiées à la résolution de problèmes en utilisant ces nouvelles commandes.

Pour enseigner la fonctionnalité des commandes de « messages », nous avons à nouveau utilisé des projets déjà prêts contenant ces commandes. En effet, nous avons laissé les élèves expérimenter en utilisant ces projets prêts tout en les accompagnant avec des questions dans ce processus de découverte. En revanche, pour le terrain français, nous n'avons pas utilisé les mêmes projets prêts que ceux que nous avons utilisés pendant la phase de l'étude préparatoire (exemple enfant-ballon de foot). Nous avons fait ce choix, car pendant la phase de l'étude préparatoire nous avons observé que les élèves de grande section avaient du mal à saisir le rôle des « messages » quand ce dernier a été présenté à l'aide de projets qui leur rappelaient une expérience de leur vie quotidienne. Ils pensaient par exemple que le personnage ballon de foot se déplaçait, car le personnage enfant avait exécuté un shoot et non pas parce qu'il avait reçu un message.

Pour aider davantage les élèves, nous avons à nouveau utilisé les cartes représentant les commandes du logiciel (voir figure 6). Les élèves les ont utilisées pour créer et observer en même temps les programmes de deux personnages communiquant par messages, vu que le logiciel ne permet pas une telle visualisation.

1.3. Séances d'évaluation

Les **séances d'enseignement et de validation** ont été suivies par les **séances d'évaluation de la performance des élèves**. Nous avons réalisé *deux séances d'évaluation* à la fin du scénario pédagogique. Au cours de la *première séance d'évaluation* nous avons de nouveau mené un entretien individuel (**Post-test**) avec les élèves. L'objectif de cet entretien était de donner la possibilité aux élèves de s'exprimer à propos des éléments de l'interface et surtout des commandes étudiées dans le scénario, afin de comprendre ce qui leur posait problèmes ou non.

Pendant la *deuxième séance d'évaluation*, nous avons donné aux élèves deux problèmes similaires à résoudre individuellement. L'objectif de cette deuxième séance d'évaluation a été de voir si les élèves de cours préparatoire étaient capables de résoudre, seuls ou aidés, deux problèmes de programmation en utilisant le langage ScratchJr, après avoir suivi notre scénario pédagogique. Tous les deux problèmes portaient sur la programmation de deux personnages. Le premier problème (PB1) concernait entre autres les commandes de « messages ». Le deuxième problème (PB2) concernait entre autres la commande de « répétition ».

Pour résoudre les deux problèmes d'évaluation, les élèves ont été invités à regarder une animation en mode plein-écran, accompagnée de la consigne des problèmes donnés lue par nous, étant donné que les élèves de cours préparatoire ne sont pas tous en mesure de lire. En d'autres mots, les élèves devaient observer le projet donné, identifier les commandes appropriées et ensuite, les utiliser dans l'ordre correct afin de reproduire les comportements observés pour les personnages participant aux deux projets. Les élèves avaient aussi à leur disposition les commandes de ScratchJr qui ont été enseignées pendant le scénario pédagogique, en version papier (cartes) pour les utiliser s'ils le souhaitent, afin de créer leurs programmes avant de passer au logiciel.

Notre intérêt était de faire de la recherche mais aussi d'intervenir pour faciliter la progression des élèves, car nous avons choisi de mener cette étude dans des contextes dits écologiques, c'est-à-dire des classes réelles au sein des écoles. C'est pourquoi, nous avons choisi d'intervenir aussi pendant la séance de l'évaluation pour aider les élèves à avancer, même si on peut penser a priori que ça pourrait gêner la posture du chercheur.

Une autre raison pour laquelle nous avons choisi d'intervenir même pendant l'évaluation était l'impact positif que notre intervention pourrait avoir sur la motivation des élèves à accomplir l'épreuve. En effet, sans aide apportée de notre part, il était possible que les élèves ne parviennent pas du tout à poursuivre l'évaluation, ce qui nous aurait privée de résultats. De plus, comme il s'agit d'enfant de 6 à 7 ans, le risque aurait été qu'ils perdent confiance en eux voire qu'ils perdent patience et arrêtent l'activité alors même qu'ils étaient capables de la continuer avec un peu d'aide. Nous nous sommes donc adaptée à notre public.

De plus, puisque l'objectif de cette évaluation n'était pas de juger les élèves mais de juger ce qu'ils peuvent faire à un moment donné, nous avons jugé que ça serait bénéfique de les aider même pendant l'évaluation, pour observer si grâce à l'apport d'aide ils ont réussi à surmonter les difficultés rencontrées ou non.

2. Choix des terrains et échantillon de l'étude principale en France

Pour rappel, nous avons réalisé une étude de cas multiples pour répondre au problème de recherche. En ce qui concerne l'étude principale en France, nous avons choisi d'étudier à nouveau deux cas (Miles et al., 2014).

Il s'agissait de deux classes, déjà équipées en tablettes, de cours préparatoire (CP1, CP2) de deux écoles primaires publiques de la ville de Puteaux, en France. Nous avons à nouveau fait un tel choix pour avoir la possibilité de vérifier si nous retrouvons les mêmes résultats d'un cas à l'autre. Comme on l'a déjà expliqué plus haut cela nous permettrait de renforcer la validité de nos résultats (Miles et al., 2014). De plus, en étudiant deux cas nous pouvons détecter plus facilement leurs similarités ou leurs différences et de cette manière mieux comprendre et expliquer le phénomène étudié (Miles et al., 2014).

En outre, en étudiant deux classes différentes nous avons procédé par triangulation avec l'utilisation des différentes « sources (personnes, temps, lieux etc.) »[notre traduction] (Denzin, 2001, cité dans Miles et al., 2014, p. 262).

Puisque nous avons décidé d'utiliser une méthodologie qualitative, nous avons choisi de travailler avec des petits échantillons. Nous avons donc mené notre étude auprès de 50 élèves de cours préparatoire, 26 élèves (12 filles et 14 garçons) dans la classe CP1 et 24 élèves (13 filles et 11 garçons) dans la classe CP2.

Pour conclure, en poursuivant les recommandations de Miles et al., (2014), notre intérêt a été de comprendre ce qui se passe dans chaque cas mais aussi d'aller plus loin dans notre compréhension du phénomène à étudier grâce à la comparaison de deux cas.

3. Collecte des données en France

Concernant le recueil des données pendant la phase de l'étude principale en France, nous avons utilisé des méthodes différentes. Les enseignantes des deux classes ont voulu participer à notre étude sous la condition que nous les accompagnions dans la mise en place du scénario pédagogique. Cette responsabilité nous a empêchée d'effectuer un protocole d'observation strict. Cela fait partie des contraintes que nous avons eu à respecter pour mener notre étude

principale en France. Nous avons donc à nouveau choisi de réaliser une observation participante (Creswel, 2012). Toutefois, nous avons pris des notes par rapport au scénario pédagogique et à sa mise en place, afin de comprendre comment on pourrait l'améliorer davantage dans le futur.

Du moment où nous n'avons pas pu adopter un protocole d'observation strict pendant la réalisation de notre étude, nous avons choisi de collecter des enregistrements vidéo (Creswel, 2012) des séances d'enseignement et de l'évaluation. Cette méthode de collecte des données permet au chercheur d'enregistrer tout ce qui se passe sur le terrain lorsque celui-ci est chargé de nombreuses responsabilités (Lodico et al., 2006). De cette manière, nous avons pu nous concentrer sur le terrain et consulter les enregistrements vidéo plus tard sans perdre d'information.

Nous avons décidé de filmer un groupe de quatre élèves pour le CP1 et deux groupes de quatre élèves pour le CP2, car cette classe a été divisée en deux. Une caméra de type GoPro a été fixée sur la casquette d'un des élèves de CP1 et de deux élèves pour le CP2, afin de pouvoir garder une trace de leur progression tout au long du scénario.

Comme nous l'avons expliqué auparavant, à la fin du scénario nous avons proposé deux problèmes à résoudre aux élèves. L'objectif de cette évaluation a été de voir ce que les élèves de cet âge-là sont capables de faire, seul ou aidés, en programmation sur ScratchJr à la fin du scénario pédagogique. Deux problèmes à résoudre ont été présentés aux élèves.

Nous avons décidé d'enregistrer en vidéo l'activité de programmation des élèves au cours de la résolution des deux épreuves, pour pouvoir avoir accès à la succession d'actions réalisées par les élèves pour créer les programmes demandés. Nous avons voulu étudier cela pour pouvoir identifier les stratégies de programmation et le débogage réalisé.

Nous avons aussi collecté les programmes élaborés par les élèves suite à la réalisation de l'activité d'évaluation, afin de pouvoir étudier si ces derniers arrivent ou non à résoudre les deux problèmes. Nous avons aussi voulu étudier les erreurs existantes sur leurs productions finales.

Enfin, nous avons décidé de collecter aussi les enregistrements audio des entretiens semi-directifs (Creswel, 2012) avec les élèves participant à notre étude au début (Pré-test) et à la fin (Post-test) de notre scénario pédagogique. En effet, nous avons collecté les enregistrements

audio des Pré-test parce que nous avons voulu identifier les idées initiales des élèves à propos de l'interface et des commandes étudiées dans le scénario. Notre objectif a été de les prendre en compte pour améliorer les séances d'enseignement du scénario pédagogique de l'étude principale. Concernant les enregistrements audio de Post-test, nous avons voulu repérer ce qui posait problème aux élèves, afin de pouvoir réaliser des recommandations pour la conception des futurs scénarios pédagogiques.

4. Traitement des données de l'étude principale en France

Dans ce qui suit, nous présentons la procédure mise en place pour traiter les données recueillies en France pendant la phase de l'étude principale.

4.1. Première tentative d'analyse des données

Nous avons débuté ce cycle d'analyses en visualisant quelques enregistrements vidéo de la séance d'évaluation réalisée en CP. Suite à cette visualisation, nous avons créé une grille décomposant la grande tâche, c'est-à-dire le problème à résoudre, en sous tâches. Nous avons procédé de la même manière pour les deux problèmes d'évaluation. Pour le PB1 par exemple, nous avons identifié les sous-tâches qui suivent : choisir les éléments de l'interface (personnage, arrière-plan), initialisation de la position des personnages, quand drapeau-vert touché, déplacement réussi pour chaque personnage, déplacement non réussi pour chaque personnage, débogage dans le cas d'un déplacement non réussi pour chaque personnage, exécution du programme de chaque personnage, envoyer un message, quand message reçu, fin de programmation, exécution du projet en plein-écran.

Notre objectif a été d'utiliser cette grille et le logiciel Nvivo pour coder ce que font les élèves de CP dans les enregistrements vidéo d'évaluation, afin de pouvoir identifier les stratégies de programmation et le débogage qu'ils mettent en place.

Lorsque nous avons tenté d'analyser les vidéos à l'aide de la grille, nous avons constaté que celle-ci réduisait beaucoup le taux d'informations recueillies par élève. En effet, les catégories apparaissaient souvent très statiques et ne permettaient pas de rendre compte de la réalité des actions de l'élève. Nous avons ainsi constaté que cette méthode d'analyse ne nous permettait

pas de rendre compte de la succession non linéaire des actions de l'élève et ainsi d'étudier son activité de programmation dans sa totalité, pour répondre à nos questions de recherche. Nous avons donc décidé de changer de direction dans nos analyses des données.

4.2. Choix des données à analyser

Nous avons débuté cette deuxième phase d'analyses en retournant sur les enregistrements vidéo et les programmes élaborés par les élèves de CP pendant la deuxième séance d'évaluation.

Tout d'abord, nous avons visualisé tous les enregistrements vidéo de l'activité de programmation des élèves provenant de l'évaluation réalisée dans les deux classes en France. Ensuite, nous avons décidé de nous focaliser sur le CP1 et en particulier sur quatre élèves de cette classe. De cette manière nous avons défini encore plus les limites du contexte autour du phénomène étudié en réalisant un « within-case sampling » (Miles et al., 2014, p. 47). Toujours dans le contexte d'une méthodologie qualitative, nous avons dû sélectionner un très petit échantillon, pour pouvoir réaliser une analyse fine de l'activité de programmation des élèves. Nous avons choisi des élèves avec des comportements contrastés afin de mieux comprendre le phénomène étudié, en réalisant ainsi un « purposeful sampling » (Creswell, 2012, p. 206).

4.3. Objectif du codage des données

Notre objectif à ce stade-là a été de trouver une méthode d'analyse permettant de mieux comprendre ce que les élèves ont pu faire seuls ou aidés pendant la séance d'évaluation, les stratégies de programmation qu'ils ont utilisées, le débogage qu'ils ont mis en œuvre. Nous avons choisi de rester au plus près de nos données, c'est-à-dire de réaliser « un codage inductif »[notre traduction] (Miles et al., 2014, p. 86) et « descriptif » [notre traduction] (Miles et al., 2014, p. 80). C'est pourquoi, nous avons décidé de nous focaliser sur les actions des élèves au cours de la résolution de deux problèmes de programmation.

4.4. Préparation des données pour l'analyse

Dans un premier temps, nous avons réalisé une transcription détaillée des actions des quatre élèves, seconde par seconde. En relisant les transcriptions, nous avons cependant constaté que

la caractérisation des modes de programmation des élèves n'était pas possible par ce biais-là. Nous avons donc continué à chercher comment faire pour présenter les actions des élèves de façon simple, afin de pouvoir ensuite les étudier et identifier des patterns possibles.

Pour résoudre ce problème, nous avons créé pour chaque élève un document avec deux colonnes contenant d'une part, les captures d'écran du logiciel et d'autre part, le fil d'actions des élèves au cours de la résolution des deux problèmes. Cette méthode de présentation a rendu plus intelligible l'activité de programmation des élèves, mais nous n'étions pas encore en mesure de comparer la succession d'actions des élèves pour repérer des patterns possibles.

C'est pourquoi, nous avons décidé d'utiliser des actigrammes, des barres de couleurs représentant la succession d'actions des élèves, pour chacun des deux problèmes. En effet, nous en avons créé deux par élève, un pour le PB1 et un pour le PB2.

4.5. Premier cycle de codage des données pour la création des actigrammes

Pour créer les actigrammes, nous avons réalisé un premier codage des actions des élèves. En réalisant ce codage, nous avons développé des « définitions opérationnelles » [notre traduction] (Miles et al., 2014, p. 89) pour tous les codes utilisés. Ce choix a facilité notre travail sur le codage des données, car en utilisant les définitions opérationnelles nous avons pu nous appuyer sur les mêmes définitions pour les codes utilisés tout au long de ce processus. De plus, il nous a permis d'être le plus transparente possible dans l'analyse de nos données. Cette transparence peut renforcer la fiabilité de notre recherche, car d'autres chercheurs auront la possibilité de reproduire cette analyse et vérifier s'ils trouvent les mêmes résultats (Miles et al, 2014).

Nous avons décidé d'attribuer une couleur différente à chaque type d'actions réalisé par les élèves. Les cinq types d'actions retenus sont les suivants : interface, commandes, corrections, initialisation et exécution.

Nous avons choisi d'utiliser des couleurs proches pour les commandes et les corrections car en tant que types d'actions ils sont très proches vu que les corrections portent sur les commandes. Nous avons utilisé des couleurs différentes pour représenter l'initialisation et l'exécution dans les actigrammes, même si elles font toutes les deux parties de l'interface.

Nous avons fait un tel choix car pendant notre présence sur le terrain, nous avons remarqué l'importance de ces deux types d'actions dans l'activité de programmation des élèves sur ScratchJr. En lisant nos données, l'intuition qu'on a eue pendant la réalisation du terrain a été renforcée. Dès lors, nous avons jugé qu'il était important de pouvoir les identifier et les différencier facilement dans les actigrammes des élèves, d'où le choix d'une couleur différente.

Chaque catégorie d'actions comprend une liste d'exemples que nous avons repérés dans les enregistrements vidéos. Dans les Tableaux 3, 4, 5, 6, et 7 qui suivent nous présentons de façon détaillée les catégories d'actions et les exemples associés.

Interface	
Description	Il s'agit des actions réalisées par l'élève qui concernent l'interface du logiciel.
Exemples	➢ L'élève efface un personnage. ➢ L'élève insère l'arrière-plan. ➢ L'élève insère un personnage. ➢ L'élève va sur l'espace de programmation du personnage qu'il souhaite programmer. ➢ L'élève va dans la catégorie des commandes qu'il souhaite utiliser. ➢ L'élève va en plein-écran. ➢ L'élève revient à l'espace utilisateur. ➢ L'élève appuie sur l'icône de la grille pour faire apparaître le quadrillage.

Tableau 3: La catégorie d'actions Interface et les exemples associés

Exécution	
Description	Il s'agit des actions réalisées par l'élève qui concernent l'exécution de ses programmes.
Exemples	➢ L'élève exécute le programme de son personnage en appuyant sur n'importe quel endroit de celui-ci. ➢ L'élève exécute son projet, c'est-à-dire les programmes de tous ses personnages, en utilisant le drapeau-vert en haut de la scène.

Tableau 4: La catégorie d'actions Exécution et les exemples associés

Initialisation	
Description	Il s'agit des actions réalisées par l'élève qui concernent l'initialisation de la position des personnages. C'est une spécificité du logiciel ScratchJr. Cette action est une étape obligatoire que l'élève doit réaliser quand il veut évaluer la distance que son personnage doit parcourir, pour se déplacer à un endroit précis dans la scène.
Exemples	➢ L'élève met son personnage au début avec ses doigts. ➢ L'élève met son personnage au début en utilisant l'icône de l'initialisation.

Tableau 5: La catégorie d'actions Initialisation et les exemples associés

Commandes	
Description	Il s'agit des actions réalisées par l'élève qui concernent la manipulation des commandes de programmation.
Exemples	➢ L'élève dépose une commande dans l'espace de programmation. ➢ L'élève efface une commande de l'espace de programmation. ➢ L'élève appuie sur la commande dans l'espace de programmation. ➢ L'élève choisit un chiffre comme le paramètre d'une commande de déplacement ou de la répétition. ➢ L'élève rassemble des commandes pour créer un programme. ➢ L'élève sépare des commandes qui étaient jusqu'à présent rassemblées dans un programme. ➢ L'élève ouvre le programme présent dans l'espace de programmation pour le corriger avant son exécution. ➢ L'élève ajoute une commande dans la commande de « répétition ». ➢ L'élève ajoute un motif des commandes dans la commande de « répétition ».

Tableau 6: La catégorie d'actions Commandes et les exemples associés

Corrections	
Description	Il s'agit des actions de l'élève qui concernent les commandes, qui interviennent suite à un problème constaté sur le programme de l'élève après son exécution.
Exemples	➢ L'élève change le paramètre de la commande de déplacement. ➢ L'élève ouvre le programme présent dans l'espace de programmation pour le corriger après son exécution. ➢ L'élève ajoute une commande dans l'espace de programmation. ➢ L'élève enlève une commande de l'espace de programmation.

	➢ L'élève rassemble des commandes de programmation. ➢ L'élève sépare des commandes de programmation.

Tableau 7: La catégorie d'actions Corrections et les exemples associés

Au cours de ce processus de codage des actions des élèves, nous avons dû prendre plusieurs décisions en ce qui concerne la façon avec laquelle nous allions coder chaque action. Une décision importante a été de ne pas introduire les erreurs réalisées par les élèves au cours de la résolution des deux problèmes dans les actigrammes, parce qu'elles ne sont pas des actions des élèves.

En revanche, nous avons choisi d'indiquer les moments d'interaction entre l'élève et la chercheuse. Nous en avons repéré deux types différents. Le premier type correspond aux moments où l'élève a réclamé de l'aide en nous posant une question. Dans le Tableau 8 qui suit nous présentons les exemples associés à cette catégorie.

Question	
Description	Il s'agit des types de questions que l'élève a posé à la chercheuse pour réclamer son aide.
Exemples	➢ L'élève pose une question par rapport à l'ordre d'exécution des commandes par son personnage. ➢ L'élève pose une question par rapport à l'efficacité de sa programmation. ➢ L'élève pose une question pour recevoir une aide à la résolution du problème.

Tableau 8: La catégorie Question et les exemples associés

Le deuxième type d'interaction identifié correspond aux moments où nous sommes intervenue pour apporter une aide à l'élève. Les aides ont été apportées non seulement lorsque l'élève se retrouvait en blocage mais également lorsqu'il réalisait une erreur.

Nous avons en particulier identifié cinq types d'aides auxquels nous avons attribué des couleurs différentes en fonction de leur nature. Dans le Tableau 9 qui suit, nous présentons les types d'aides apportées par la chercheuse et les exemples associés.

Aides	
Type	**Exemples**
Consigne[121]	➢ La chercheuse rappelle à l'élève la consigne du problème de programmation.

121 Nous avons choisi d'utiliser la même couleur pour ces deux types d'aide (consigne et lancement projet), car nous les considérons comme des aides de la même nature.

Lancement projet	➢ La chercheuse lance le projet modèle, pour que l'élève puisse voir le résultat souhaité qu'il doit reproduire.
Vérification de programmation	➢ La chercheuse pose une question à l'élève pour lui rappeler de vérifier l'exactitude de sa programmation.
Identification de l'erreur	➢ La chercheuse pose une question à l'élève pour le guider dans l'identification du problème existant sur son programme.
Initialisation	➢ La chercheuse aide l'élève à initialiser la position de son personnage.
Effacement d'une commande	➢ La chercheuse aide l'élève à effacer une commande de son espace de programmation pour ne pas rendre plus difficile la correction d'une erreur déjà existante.

Tableau 9: Les types d'aides apportées par la chercheuse et les exemples associés

Ainsi, nous sommes parvenue à la création des actigrammes (voir figure 9), des barres de couleurs représentant la succession des actions des élèves, pour chacun des deux problèmes. Dans la figure 9 qui suit, nous présentons à titre illustratif un exemple d'actigramme provenant de nos données.

Figure 9: Un exemple d'actigramme

Les actigrammes, les captures d'écran du logiciel et la transcription du fil d'actions des élèves nous ont aidée à rendre plus intelligible la visualisation et de ce fait la compréhension de l'activité de programmation des élèves.

Nous avons également décidé d'énumérer le nombre d'actions réalisées par catégorie d'actions, pour repérer celle qui préoccupe le plus les élèves, mais aussi pour comprendre quel était le problème (PB1 ou PB2) qui leur a demandé le plus d'actions. Nous avons aussi énuméré le nombre total d'actions réalisées par les quatre élèves pour la résolution du PB1 et du PB2. L'énumération est une stratégie qui peut aussi être utilisée dans les recherches qualitatives (Miles et al., 2014). De plus, nous avons quantifié le temps passé sur chaque problème pour pouvoir comprendre quel était le problème qui leur a posé la plupart de difficultés.

À partir de ces éléments-là, nous avons créé deux Tableaux (31 et 32) que nous présenterons dans la Partie 4 qui suit. Ces tableaux nous ont permis de faire des comparaisons entre les quatre élèves.

4.6. Deuxième cycle de codage des données

Après avoir représenté de façon visuelle et intelligible l'activité de programmation des élèves, nous avons regardé à la fois si leur succession d'actions était la même ou pas, mais aussi s'il y avait des patterns récurrents. Les patterns sont des actions ou des déroulements d'actions qui sont répétés dans l'activité de programmation des élèves. De cette manière, nous avons rapidement constaté des différences entre les déroulements d'activité des quatre élèves et nous avons utilisé la combinaison de modèles théoriques, que nous avons identifié à la fin de notre cadre conceptuel, pour les caractériser. Nous avons donc opté ici pour « un codage déductif » [notre traduction] (Miles et al., 2014, p.86) provenant de la littérature.

4.7. Adaptation du lexique de la littérature à nos données

Nous avons donc entrepris d'adapter le lexique de la littérature à nos données. Durant ce processus, nous avons aussi eu besoin de créer nos propres typologies quand celles de la littérature ne convenaient pas pour analyser nos données. Nous avons donc combiné les deux approches de codage, en réalisant du codage déductif et inductif (Miles et al., 2014).

La principale difficulté à cette étape était d'identifier les catégories de la littérature dans nos données de corpus. En effet, les recherches existantes avaient fourni un lexique de catégories de stratégies de programmation et de débogage. Toutefois, ces modèles, à l'exception du modèle de Klhar et Carver (1988) sur le débogage, ne proposaient pas de méthode pour les identifier. Cela arrive parce que dans la plupart des articles scientifiques les règles de rédaction sont telles que les chercheurs n'ont pas assez de place pour présenter de façon analytique leur méthodologie.

Cependant, grâce à la représentation simplifiée de l'activité des élèves que nous avons créée et à la suite des lectures consécutives de nos données, nous avons réussi à repérer une série

d'éléments clés qui nous ont aidée à caractériser le mode de programmation des élèves. Nous détaillons ces éléments ci-dessous.

Un des éléments clés, qui provient de la première phase d'analyse des données recueillies en France, a été le constat qu'il faut regarder l'intégralité des actions des élèves pour pouvoir juger leur mode de programmation. En étudiant l'activité de programmation des élèves, un de nos premiers constats a été qu'il n'y avait pas de distinction claire entre l'écriture et le débogage du programme par les élèves. La première question qu'on s'est alors posée alors a été de savoir comment régler ce problème, comment savoir jusqu'où allait la conception du programme et d'où commençait le débogage.

4.8. Exécution, la clé pour caractériser le mode de programmation des élèves

D'après le modèle de Klhar et Carver (1988), le débogage démarre une fois que l'élève a exécuté son programme et qu'il a pu constater qu'il y a un écart entre le résultat obtenu suite à l'exécution et le résultat souhaité. La clé pour clarifier cela a été pour nous l'exécution. Ainsi, nous avons décidé de nous focaliser sur la façon dont les élèves ont utilisé l'exécution dans leur programmation. En effet, nous avons fait l'hypothèse que lorsque nous constatons l'absence de l'exécution par la succession d'actions des élèves, c'est de la « planification » et lorsque nous constatons une utilisation constante, c'est plutôt du « pas à pas ».

Nous avons donc regardé à nouveau la succession d'actions des élèves sous ce nouveau prisme, afin de pouvoir caractériser les stratégies de programmation qu'ils utilisent. Concernant les stratégies de déplacement, nous avons décidé de décrire les différentes manières employées par les élèves dans les vidéos pour déplacer leurs personnages.

4.9. Un focus sur la qualité des actions des élèves

Après avoir étudié les stratégies de programmation et de déplacement utilisées par les élèves, nous avons, dans un deuxième temps, étudié la qualité des actions des élèves. Un de nos objectifs en faisant cette analyse a été de voir s'ils font des erreurs et quelle est la nature de celles-ci. L'erreur est un marqueur de difficulté. Si les élèves arrivent à surmonter des difficultés rencontrées, cela précise ce qu'ils sont capables de faire. Nous avons également

analysé s'ils arrivent à comprendre qu'ils ont réalisé une erreur, s'il y a des moments dans les vidéos où ils sont bloqués, ainsi que le débogage qu'ils mettent en place, s'il y en a. C'est pourquoi, nous avons trouvé intéressant de différencier les erreurs que les élèves arrivent à surmonter et les erreurs qui persistent jusqu'à la fin.

Nous avons aussi cherché dans les enregistrements vidéo des éléments qui pourraient nous aider à identifier quand les élèves se trouvent face à un blocage pendant la résolution des deux problèmes de programmation. Il s'agit d'une suite d'actions de l'élève que l'on interprète comme un blocage. Les actigrammes nous ont beaucoup servi à ce stade-là. Dans le Tableau 10 qui suit nous présentons le catalogue d'exemples que nous avons retrouvés dans les vidéos et nous les avons interprétés en tant que moments de blocage.

Moments de blocage	
Description	Il s'agit d'une suite d'actions de l'élève que l'on interprète comme un blocage. Ce sont des moments dans les vidéos où l'élève présente une certaine incertitude dans ses actions réalisées, il ne réalise pas d'actions ou il réalise des actions qui ne changent pas l'état de son projet.
Exemples	➢ Exécution et initialisation des programmes sans leur apporter de modifications après. ➢ Insertion d'une commande et après effacement. ➢ Temps sans actions sur la tablette. ➢ Changement des catégories de commandes de programmation pendant un certain temps. ➢ Changement des personnages pendant un certain temps. ➢ Exécution d'un programme sans lui apporter des modifications après. ➢ Doigt au-dessus des catégories des commandes mais pas de choix réalisé. ➢ Ouverture du programme après son exécution et déplacement des commandes existantes dans celui-ci à droite et à gauche, sans effectuer une autre action de correction.

Tableau 10 : Les moments de blocage et les exemples associés

Le changement de comportement de l'élève (ou absence de changement de comportement) suite à l'apport d'aide de la chercheuse a également été étudié.

Pour analyser le débogage réalisé par les élèves dans les enregistrements vidéo, nous avons utilisé le modèle proposé par Klahr et Carver (1988), que nous avons brièvement présenté

auparavant dans cette section. En effet, nous avons comparé les cinq étapes présentées dans le modèle avec ce que les élèves font dans les vidéos pour corriger leurs programmes.

Ainsi, nous avons remarqué que pour certains élèves, nous pouvons voir seulement les étapes 1) Évaluation du programme, 2) Identification d'erreur et 5) Correction de l'erreur mais pas les étapes 3) Représentation du programme et 4) Localisation de l'erreur.

Pour d'autres élèves, nous pouvons voir les étapes 1) Évaluation du programme, 2) Identification d'erreur 4) Localisation de l'erreur et 5) Correction de l'erreur, mais nous ne pouvons toujours pas voir l'étape 3) Représentation du programme.

D'après le modèle de Klahr et Carver (1988), le débogage démarre à partir du moment où l'élève exécute son programme et qu'il repère qu'il existe un écart entre le résultat souhaité et le résultat obtenu suite à l'exécution. Pour juger le débogage des élèves, nous avons examiné si le programme contenant l'erreur ou les erreurs est correct ou reste erroné suite à la réalisation de l'action ou des actions de correction par l'élève.

En termes d'actions, nous avons repéré trois types associés au débogage dans les actigrammes des élèves : l'exécution, la correction et l'initialisation. De plus, nous avons observé qu'un débogage pourrait contenir une ou plusieurs actions de correction.

Les actigrammes nous ont particulièrement aidée avec l'identification des patterns de débogage. Dans le contexte de cette thèse, nous définissons en tant que pattern de débogage non pas une succession d'actions spécifiques (comme exécution, exécution, correction, correction, interface, initialisation), mais une succession de type d'actions (comme exécution, correction du code, initialisation). Nous considérons un morceau d'actigramme ou autrement dit une succession de types d'actions en tant que pattern de débogage quand ce dernier se répète au moins deux fois sur les actigrammes d'un élève ou sur les actigrammes de deux élèves différents.

Grace à cette analyse fine des actions de quatre élèves, nous avons pu identifier une série d'éléments qui nous permettent de bien comprendre ce qu'ils font en programmation sur ScratchJr et ainsi répondre à notre problématique. Nous avons décidé de présenter les résultats de cette analyse à l'aide des portraits.

Partie 4. Présentation des résultats de la recherche

Chapitre 1. Étude préparatoire en Grèce

Pour rappel, nous avons mené notre recherche en deux phases : une étude préparatoire réalisée dans deux classes d'écoles maternelles publiques en Grèce et une étude principale réalisée dans deux classes de cours préparatoire publiques en France. Comme nous l'avons expliqué auparavant, nous avons recueilli et traité les données de deux terrains différemment. C'est pourquoi il nous a paru évident de présenter les résultats des deux terrains séparément. Dans ce chapitre, nous nous concentrons sur l'étude préparatoire en Grèce.

Cette phase préparatoire était très importante pour la réalisation de notre recherche car elle nous a permis de tester notre scénario pédagogique auprès d'élèves de grande-section, et de l'améliorer pour l'étude principale dans des écoles primaires en France. La période en Grèce nous a également aidée à mieux organiser la méthodologie de mise en place de notre recherche pour le terrain en France. Malgré les difficultés d'accès aux écoles et les conséquences que ces dernières ont eu sur notre intervention, nous avons pu obtenir une vision globale sur ce que les élèves de grande section de maternelle peuvent faire seuls ou aidés en programmation sur ScratchJr et les difficultés qu'ils risquent de rencontrer.

Nous commençons par présenter des résultats généraux portant sur l'appropriation du logiciel par les élèves de maternelle. Nous détaillons ici ce qu'ils arrivent à découvrir tout seuls, ce qu'il faut leur présenter car ils ne sont pas capables de le découvrir, comment ils gèrent le déplacement de leurs personnages, comment nous avons procédé pour les aider à concrétiser le démarrage et la fin d'un programme, ainsi que le démarrage synchrone des programmes de deux personnages. Dans les résultats généraux, nous présentons également quelques limites du logiciel. Par la suite, nous présentons des résultats en particulier sur le niveau d'appropriation de la « répétition » et des « messages » par les élèves. Nous mettons également en avant les difficultés rencontrées par les élèves de maternelle. Nous clôturons ce chapitre avec une synthèse des résultats de l'étude préparatoire en Grèce.

1. Gestion de l'interface et appropriation des commandes enseignées

Rappelons qu'il s'agit de l'initiation des élèves de GS2 au logiciel ScratchJr et aux commandes de programmation vues dans le scénario. Nous avons décidé de présenter séparément les résultats sur l'appropriation des commandes de « messages » et de « répétition prédéfinie » par les élèves, parce que celles-ci leur posent la plupart des difficultés. Ces résultats proviennent de l'analyse des notes de nos observations du terrain et des programmes élaborés par les élèves suite à la réalisation de l'activité de l'évaluation.

1.1. Éléments de base d'interface et processus d'écriture des programmes

Nous avons observé qu'il existe certains éléments du logiciel ScratchJr que les élèves de GS2 semblent pouvoir découvrir tout seuls et d'autres qu'il faut leur présenter. D'après les notes de nos observations de terrain, les éléments de base de l'interface de ScratchJr ne posent pas de problèmes aux élèves de GS2.

Nous avons par exemple observé qu'ils arrivent à choisir des commandes par la palette des commandes et ils essayent de les poser dans la scène, mais ils comprennent rapidement que cela ne marche pas. Ils testent ensuite de les déposer dans l'espace de programmation et ils découvrent ainsi la logique du *drag-and-drop* de l'interface de ScratchJr, c'est-à-dire qu'ils doivent glisser les commandes et les laisser dans l'espace de programmation afin de les activer. Les élèves de maternelle arrivent sans problème à découvrir qu'ils doivent appuyer sur une commande pour l'exécuter. Ils arrivent aussi à découvrir qu'ils peuvent assembler plusieurs commandes différentes pour créer des programmes. Ils ont facilement découvert qu'ils peuvent exécuter un programme en s'appuyant sur ceci. L'effacement des commandes de l'espace de programmation est aussi un processus qui ne pose pas de problèmes aux élèves de GS2. Pendant la séance de découverte et de familiarisation avec le logiciel, les élèves ont observé, grâce à l'utilisation des commandes de déplacement et accompagnement, qu'ils peuvent dire au personnage « chat » d'aller là où ils veulent.

Ils découvrent également comment choisir des arrière-plans et insérer de nouveaux personnages dans leur projet. Nous avons repéré que les personnages attirent beaucoup l'attention des élèves, qui passent leur temps pendant la séance de familiarisation avec le logiciel à en intégrer de nouveaux, sans les programmer.

Ils arrivent aussi à trouver l'icône leur permettant d'aller en plein-écran. Toutefois, nous avons dû leur présenter le processus d'effacement d'un personnage de la scène, parce qu'ils n'arrivaient pas à le découvrir seuls.

1.2. La gestion du déplacement des personnages par les élèves

1.2.1. Quelques mots sur la fonctionnalité des commandes de déplacement

Le déplacement des personnages a une place importante dans la programmation sur ScratchJr. Les commandes de déplacement servent à déplacer le personnage choisi d'un pas dans une direction spécifique. Toutes les commandes de déplacement ont comme paramètre par défaut le chiffre un à part la commande « sauter » qui a le chiffre deux.

Nous avons décidé de ne pas enseigner aux élèves les paramètres puisque nous n'avons pas voulu leur ajouter une difficulté de plus pour deux raisons. La première parce que nous avons jugé que l'initiation à la programmation graphique était déjà une entreprise difficile. La deuxième parce qu'en maternelle les élèves commencent à être familiarisés avec la notion du nombre, mais ils n'ont pas encore complètement compris que le nombre peut à la fois représenter une quantité (usage cardinal) mais aussi le positionnement dans une liste d'objets (usage ordinal). Certains élèves n'arrivent pas encore à faire la correspondance entre les chiffres et les nombres.

Ce type de commandes a principalement aidé les élèves pendant la séance de préparation psychologique et de familiarisation avec le logiciel ScratchJr à découvrir le fonctionnement de son interface et le processus d'écriture des programmes. Les trois premières séances du scénario ont été dédiées à la résolution de problèmes de déplacement. Les élèves de GS2 arrivent facilement à découvrir la fonctionnalité de ces commandes.

1.2.2. Observation des élèves

Nous avons observé que, pour faire déplacer un personnage, les élèves passent par différentes phases. Au début, ils essaient de déplacer le personnage à l'endroit souhaité avec le doigt. Ensuite, ils choisissent des commandes sans faire attention au problème qu'ils ont à résoudre. En effet, au début les élèves de GS2 ont eu des difficultés à comprendre qu'il ne fallait pas seulement déplacer leur personnage, mais le déplacer vers un endroit spécifique dans la scène pour résoudre le problème donné. Puis, grâce à des aides didactiques apportées par la chercheuse, la plupart d'entre eux semblent comprendre qu'ils doivent chaque fois réfléchir au problème qu'ils ont à résoudre.

Par la suite, ils commencent à utiliser la bonne commande et à cliquer plusieurs fois sur celle-ci pour déplacer le personnage « chat » vers l'endroit demandé. Plus tard, ils utilisent plusieurs fois la même commande afin de résoudre le problème donné. Nous avons également observé qu'à l'issue de la troisième séance du scénario, certains élèves étaient toujours à la phase où ils choisissaient des commandes pour déplacer leur personnage sans réfléchir au problème qu'ils avaient à résoudre.

Un élève assez avancé a découvert les paramètres des commandes de déplacement pendant la première séance d'enseignement et les a utilisées pour déplacer le personnage « chat » au lieu d'utiliser plusieurs fois la même commande, comme le faisaient les autres élèves. Ce même élève a mis en place pendant la deuxième séance d'enseignement une stratégie de réduction d'écriture. Il s'agit de l'élève GS2E7[122]. En effet, au début, GS2E7 utilisait quatre fois la même commande pour déplacer son personnage en bas. Puis il a décidé d'effacer les quatre commandes et d'utiliser une seule commande de « déplacement en bas », dont il a modifié le paramètre en choisissant le chiffre quatre au lieu du chiffre un, qui est le paramètre par défaut. En réalisant cela GS2E7 a dit :« une autre façon de faire »[123].

Une autre élève du même groupe, GS2E8, l'a vu utiliser les paramètres et elle a aussi voulu les utiliser. Au cours des deuxième et troisième séances d'enseignement, GS2E8 utilise en même temps les paramètres et plusieurs fois les mêmes commandes pour déplacer son personnage. À part ces deux élèves, tous les autres utilisent plusieurs fois les mêmes

122 Nous avons décidé de suivre le même système de nomination pour tous les élèves de GS2. GS représente le niveau scolaire de l'élève, le numéro 2 signifie que cet élève vient de l'école maternelle 2, le E signifie l' élève et v signifie le numéro de l'élève en question.

123 Notre traduction du grec : « Αλλος τρόπος ».

commandes pour résoudre les problèmes de déplacement donnés pendant les séances d'enseignement.

1.2.3. Analyse des programmes d'élèves en fin d'intervention

Pour évaluer la capacité des élèves à programmer un déplacement, nous avons intégré des déplacements à réaliser dans les deux problèmes à résoudre de l'évaluation. Le PB1 comportait deux déplacements à réaliser, un pour le personnage « enfant » et un pour le personnage « papillon » et le PB2 comportait un seul déplacement à réaliser, celui du personnage « enfant ». Vu l'âge des élèves de notre public, nous leur avons demandé de réaliser deux déplacements simples vers une seule direction pour le PB1 et un déplacement plus court mais dans deux directions différentes pour le PB2.

En analysant les programmes créés par les élèves pour résoudre les deux problèmes, nous pouvons constater qu'ils utilisent tous plusieurs fois les mêmes commandes pour déplacer leurs personnages. Les élèves GS2E7 et GS2E8 n'ont pas utilisé les paramètres pendant l'évaluation, même s'ils l'ont fait pendant certaines séances d'enseignement.

En analysant les programmes que nous avons collectés pour le PB1, nous pouvons observer que tous les élèves se trompent sur le déplacement. En effet, nous avons remarqué que la majorité se trompe sur le nombre de pas que le personnage concerné doit réaliser dans la scène pour arriver sur l'endroit demandé.

Cinq élèves (GS2E3, GS2E4, GS2E7, GS2E8, GS2E12) sur 12 se sont trompés sur le déplacement de l'un de deux personnages. Nous repérons que dans les programmes des élèves GS2E4, GS2E8 et GS2E12, il manque une commande par rapport au projet modèle pour l'un des deux personnages, tandis que les programmes des élèves GS2E3 et GS2E7 comportent une commande en plus.

Cinq autres élèves (GS2E1, GS2E2, GS2E6, GS2E10, GS2E11) sur 12 se sont trompés de la même manière dans les programmes de leurs deux personnages (soit une commande de déplacement manque dans leur programme, soit leur programme a une commande de déplacement de plus). L'élève GS2E6 se trompe également sur la direction du déplacement car dans le programme du personnage « papillon » elle avait ajouté une commande de

déplacement vers la gauche, tandis que cette dernière ne faisait pas partie du programme du personnage dans le projet modèle.

GS2E5 et GS2E9 sont ceux qui ont eu le plus de difficultés avec le déplacement de leurs personnages au PB1. GS2E5 n'arrive pas à déplacer les deux personnages. Le programme que nous avons trouvé dans sa tablette pour le personnage « enfant », nous l'avons vu l'écrire en regardant le projet de l'élève GS2E7, qui était assis à côté de lui. GS2E9 se trompe sur le déplacement des deux personnages. En effet, pour le déplacement du personnage « papillon » manquent quatre commandes de déplacement vers le bas et par le déplacement du personnage « enfant » il manque deux commandes de déplacement à gauche.

En analysant les programmes que nous avons collectés pour le PB2, nous pouvons observer que tous les élèves utilisent à nouveau plusieurs fois les mêmes commandes pour déplacer leur personnage, même les élèves GS2E7 et GS2E8 qui avaient utilisé les paramètres pendant certaines séances d'enseignement.

Avant d'analyser de manière plus précise les programmes, nous souhaitons rappeler que dans le PB2 les élèves devraient aussi programmer leur personnage à sauter. La commande « sauter » sur ScratchJr est la seule commande de déplacement ayant comme paramètre par défaut le chiffre 2 au lieu du chiffre 1 qui est le paramètre par défaut des commandes de déplacement. Nous avons décidé d'enseigner cette commande aux élèves de maternelle malgré le paramètre différent en leur conseillant de ne pas le changer.

En analysant les programmes pour le PB2, nous pouvons également remarquer que six élèves (GS2E4, GS2E7, GS2E8, GS2E9, GS2E10, GS2E12) sur 12 ont réussi à déplacer le personnage « enfant » correctement vers l'endroit demandé dans la scène. Ces élèves ont utilisé la commande pour faire sauter le personnage sans modifier son paramètre. Deux élèves (GS2E1, GS2E2) se sont trompés sur la commande qui fait sauter le personnage car elles ont modifié son paramètre en choisissant le numéro un au lieu du numéro deux qui est le paramètre par défaut. Trois élèves (GS2E3, GS2E6, GS2E11) sur 12 se sont trompés aussi sur le déplacement car au lieu d'ajouter la commande « sauter » ils ont ajouté une commande pour déplacer le personnage vers le haut dans la scène. L'élève GS2E5 n'est pas arrivé à programmer le déplacement du personnage « enfant ».

1.2.4. Des interprétations

Au vu de ces résultats, nous pouvons constater que les élèves de GS2 présentent une meilleure performance au PB2 qu'au PB1. Nous supposons que cela arrive car dans le second problème les élèves doivent programmer un déplacement court, tandis qu'au premier ils doivent programmer deux déplacements plus longs (5 et 4 commandes de déplacement au total). Ces résultats nous ont fait comprendre que ce type de déplacement était peut-être pas adapté à leur âge pour le moment. Il fallait plutôt leur demander de réaliser des déplacements plus courts si on voulait qu'ils réalisent deux au PB1 et un plus long au PB2, puisqu'il serait le seul à réaliser.

Une autre remarque que nous pouvons faire en analysant la performance des élèves au PB1 c'est que la majorité (dix élèves sur 12) se trompent sur le déplacement de l'un ou des deux personnages en ayant une commande de plus ou de moins dans leur programme. Nous pouvons supposer qu'à l'origine de cette erreur si répandue se trouve l'initialisation de la position des personnages dans la scène. En effet, nous faisons l'hypothèse que les élèves initialisent la position des personnages avec le doigt au lieu d'utiliser l'icône pour l'initialisation et, comme nous l'avons déjà indiqué plus haut, la moindre différence au positionnement pourrait engendrer des erreurs sur la programmation du déplacement de leurs personnages.

Nous pouvons également constater que, malgré ces petites erreurs d'un pas de plus ou un pas de moins au niveau du déplacement, la plupart des élèves de GS2 arrive à utiliser de manière opérationnelle les commandes de « déplacement » pour résoudre le PB1. Ils choisissent les commandes qu'il faut vers la bonne direction pour déplacer leurs personnages selon le projet modèle, même si parfois ils demandent notre aide pour distinguer vers où c'est la droite et vers où c'est la gauche en indiquant avec leur main les directions. En revanche, une fois qu'ils ont cette information ils n'ont pas des difficultés à identifier la commande qui correspond à chaque direction.

En ce qui concerne l'élève GS2E6 qui était la seule à se tromper sur la direction d'une commande du programme du personnage « papillon », nous pouvons faire deux hypothèses pour expliquer cette erreur. Il est possible qu'elle se soit trompée sur le positionnement du personnage « papillon » en le posant avec le doigt un peu plus à droite que ce qu'il fallait et

c’est pourquoi elle a eu besoin d’ajouter une commande de déplacement à gauche dans son programme pour la faire revenir au bon endroit. Une autre hypothèse que nous pouvons faire est qu’elle a confondu le déplacement du personnage « papillon » avec celui du personnage « enfant ».

En ce qui concerne les élèves GS2E5 et GS2E9, nous pouvons repérer que le premier n’est visiblement pas capable de programmer le déplacement de ces personnages ni au PB1, ni au PB2, tandis que GS2E9 se trompe davantage sur le déplacement de ces personnages au PB1, mais elle arrive à programmer correctement le déplacement du personnage au PB2.

Nous pouvons donc réfléchir autour des raisons des difficultés rencontrées par les élèves concernant la programmation du déplacement des personnages. Il est possible qu’ils aient eu des difficultés avec la programmation du déplacement car ils n’ont pas encore bien saisi la logique de la programmation sur ScratchJr et les phases par lesquelles ils doivent passer pour résoudre un tel problème.

Une autre raison expliquant ces difficultés pourrait être la nature de la programmation à réaliser, ici le déplacement. Nous pensons que ces élèves ont des difficultés avec la programmation du déplacement parce qu’ils ne savent pas comment déplacer des personnages.

En analysant le déplacement de personnages sur ScratchJr, nous pouvons observer que ce dernier implique la mobilisation d’une série des connaissances en lien avec l’orientation dans l’espace et les mathématiques. Les élèves doivent pouvoir différencier la gauche, la droite, le dessus et le dessous et associer la flèche existante sur les commandes de déplacement avec la direction indiquée. En ce qui concerne les connaissances mathématiques, les élèves doivent avoir établi la correspondance entre chiffre et nombre mais aussi avoir compris son usage cardinal. Il est important qu’ils sachent dénombrer pour pouvoir trouver le bon nombre de pas pour que leur personnage se déplace d’un endroit spécifique à un autre dans la scène.

Nous pouvons donc supposer que GS2E5 et GS2E9 dominent moins bien ces notions que les autres élèves de notre échantillon. C’est pourquoi, GS2E5 n’arrive pas à déplacer ses personnages ni au PB1 ni au PB2 et GS2E9 arrive moins bien que les autres au PB1 avec deux déplacements longs et mieux au PB2 avec un déplacement court. Nous pouvons également supposer que GS2E7 et GS2E8, qui sont les seuls qui ont utilisé les paramètres

pendant les séances d'enseignement, sont peut-être un peu plus avancés que les autres puisqu'ils dominent mieux ces notions mathématiques et surtout celle du nombre qui est directement liée avec l'utilisation des paramètres.

1.2.5. La commande « sauter » et les problèmes qu'elle pose aux élèves

Nous pouvons également constater que la commande « sauter » a posé des problèmes, puisque cinq élèves sur 12 se trompent sur celle-ci. Une raison de ces difficultés pourrait être le fait que cette commande a le chiffre 2 comme paramètre par défaut, tandis que toutes les autres commandes de déplacement ont le chiffre 1 par défaut.

La commande « sauter » a comme paramètre le chiffre 2 parce qu'elle oblige le personnage à monter une case et ensuite descendre une case sur l'axe des ordonnées. Il s'agit d'une commande qui inclut deux déplacements différents, un vers le haut et un vers le bas. Puisque la réaction du personnage dans la scène suite à l'exécution de cette commande est assez rapide, les élèves n'ont pas beaucoup de temps pour bien comprendre ce qu'elle fait. Par conséquent, ils ont des difficultés surtout avec le sens de son paramètre.

C'est pourquoi lorsque l'énoncé du problème leur demande de faire sauter le personnage une fois, les élèves GS2E1 et GS2E2 ne comprennent pas que cela correspond à une commande « sauter » avec le paramètre 2 par défaut et non pas à une commande « sauter » avec le paramètre 1.

Il est également possible que les élèves GS2E1 et GS2E2 se soient trompés en changeant le paramètre par défaut de la commande « sauter » et en choisissant comme paramètre le numéro un, parce qu'ils étaient plus habitués à avoir des commandes de « déplacement » avec ce paramètre. En outre, il s'agit d'une commande qui a été enseignée en même temps que la commande de « répétition » et qui n'a pas été très utilisée par la suite dans les problèmes à résoudre du scénario pédagogique. Les élèves de GS2 n'avaient pas assez de temps pour se familiariser avec la commande et son sens.

1.3. La nécessité et les difficultés de l'initialisation de la position des personnages

L'initialisation de la position des personnages est une question importante sur le logiciel ScratchJr. Elle est nécessaire car pour programmer le déplacement d'un personnage sur ScratchJr, il faut d'abord choisir sa position initiale avec le doigt et ensuite, utiliser l'icône de l'initialisation pour le remettre au début. ScratchJr propose aussi une commande qui sert à initialiser la position des personnages, mais nous avons décidé de ne pas l'enseigner aux élèves.

L'initialisation fait partie des éléments qu'il faut présenter aux élèves, car ils ne peuvent pas les découvrir seuls. Nous avons observé que les élèves de maternelle avaient des difficultés à retenir qu'il fallait remettre à chaque fois leur personnage à sa position initiale avant de tester la programmation réalisée pour son déplacement dans la scène. Lorsqu'ils oublient d'initialiser la position de leur personnage, ils ne peuvent pas vérifier l'exactitude de leur programmation puisque celui-ci ne commence pas son déplacement du bon endroit dans la scène.

Cela leur pose problème parce qu'il est possible qu'ils aient choisi le bon nombre de pas pour déplacer leur personnage à l'endroit demandé, mais ils ont l'impression qu'ils ont fait une erreur car leur personnage n'arrive pas où il fallait. Certains n'arrivent pas à comprendre que cette erreur provient du fait qu'ils n'ont pas initialisé la position de leur personnage.

D'autres préfèrent mettre le personnage au début avec le doigt au lieu d'utiliser l'icône d'initialisation. Nous avons observé que lorsque les élèves initialisent la position de leur personnage avec le doigt, le risque est de ne pas le remettre chaque fois au même point de départ. Cela pose aussi problème à leur programmation car cela affecte le nombre de pas que le personnage doit faire pour arriver au bon endroit dans la scène.

Pour remédier à ce problème, nous avons eu besoin de leur rappeler l'initialisation de la position des personnages plusieurs fois au cours du scénario, car certains l'oubliaient. À force de se tromper sur le déplacement des personnages en raison de l'oubli de l'initialisation des personnages et grâce à notre accompagnement, certains ont réussi à surmonter cette difficulté. Au bout de la *troisième séance* d'enseignement, certains élèves éprouvaient toujours des difficultés avec l'initialisation de la position des leurs personnages.

Une raison de ces difficultés pourrait être le fait que l'initialisation concerne le positionnement des personnages dans la scène, mais elle influence aussi la vérification de la programmation réalisée et ainsi son efficacité. Cela peut être difficile à comprendre pour les élèves de maternelle. De plus, l'initialisation de la position des personnages n'est pas familière en maternelle puisqu'elle ne ressemble à aucune des activités qu'ils sont habitués à réaliser à l'école.

1.4. Des difficultés face au débogage

Le débogage et surtout sa première étape, la vérification de l'exactitude de la programmation réalisée, se trouvent au cœur de l'activité de programmation et ce sont des éléments clé pour la résolution des problèmes. Pour vérifier l'exactitude de la programmation réalisée, il faut l'exécuter. Il existe deux façons différentes pour le faire sur ScratchJr : en appuyant sur n'importe quelle commande d'un programme avec le doigt ou en cliquant sur le bouton du drapeau-vert en haut de la scène, sous la condition qu'au moins un programme dans le projet contienne la commande « quand drapeau-vert cliqué » à son début.

Le débogage fait partie des éléments qu'il faut présenter aux élèves, car ils ne peuvent pas les découvrir seuls. En effet, nous avons observé pendant notre présence sur le terrain que les élèves de GS2 avaient des difficultés au début à retenir qu'il fallait tester le programme qu'ils ont créé pour voir si cela fonctionne comme ils le souhaitaient ou pas. Ils ne comprenaient pas tout de suite pourquoi la vérification était nécessaire, c'est pourquoi ils oubliaient de le faire surtout au début. Nous avons eu besoin de leur rappeler à plusieurs reprises pendant les trois premières séances d'enseignement la nécessité d'exécuter leurs programmes pour vérifier s'ils l'ont bien programmé, car certains oubliaient de le faire et ainsi ils n'arrivaient pas à résoudre les problèmes.

Une raison de ces difficultés pourrait être le fait que les élèves de maternelle ne sont pas encore totalement familiarisés avec la résolution des problèmes et surtout qu'ils n'ont pas l'habitude de vérifier l'exactitude de ce qu'ils ont produit. C'est sans doute pourquoi il leur paraît difficile de le faire en programmation.

1.5. Les commandes de démarrage et de fin d'un programme

1.5.1. Quelques mots sur la fonctionnalité des commandes

Le démarrage et la fin d'un programme sont aussi des éléments qui ne sont accessibles aux élèves qu'à l'aide de l'enseignante, ici la chercheuse. Il s'agit des commandes « quand drapeau-vert cliqué » et « fin ».

La commande « quand drapeau-vert cliqué » fait partie des commandes de « démarrage » qui servent à définir le mode de démarrage d'un programme. En effet, lorsque l'événement indiqué sur une commande de « démarrage » se déclenche, le programme du personnage ayant celle-ci à son début sera exécuté. La commande « quand drapeau-vert cliqué » sert à démarrer un programme lorsque l'on clique sur le bouton du drapeau-vert en haut de la scène. La forme de cette commande oblige l'utilisateur à la poser au début du programme d'un personnage, avant toute autre commande. Il est obligatoire d'avoir au moins une commande « quand drapeau-vert cliqué » dans un projet de ScratchJr, car cette commande est liée au bouton du « drapeau-vert » en haut de la scène, qui est le seul moyen pour lancer le projet en mode plein-écran.

La commande de « fin » fait partie des commandes de « fin » et elle indique la fin d'un programme, mais elle n'influence ni le programme ni le comportement du personnage concerné.

Pour enseigner ces deux commandes aux élèves, nous nous sommes basés sur le jeu d'enfant robot que nous avons réalisé pendant la séance de familiarisation avec la programmation. Au cours de ce jeu, nous avons utilisé deux cartes, une verte et une rouge, pour signaler le départ et l'arrivée de l'enfant-robot.

Nous avons donc pendant la *première séance* d'enseignement rappelé ces cartes aux élèves en leur disant que comme ils les ont utilisées pendant le jeu pour signaler le début et la fin de déplacement à leur camarade, il fallait de la même manière signaler à leur personnage quand il doit commencer et arrêter son programme. Pour ce faire, il fallait utiliser la commande « quand drapeau-vert cliqué » en tant que début du programme et la commande « fin » en tant que la fin du programme.

Un de trois groupes d'élèves a même fait le lien entre les cartes du jeu et le début et la fin du programme du personnage « chat », sans avoir besoin de l'aide de la chercheuse. Cette stratégie didactique a beaucoup aidé les élèves à concrétiser le démarrage et la fin d'un programme. Néanmoins, nous avons eu besoin de leur rappeler ces éléments plusieurs fois au cours du scénario, car il y avait certains élèves qui parfois les oubliaient.

1.5.2. Analyse des programmes d'élèves en fin d'intervention

En analysant les programmes créés par ces derniers pour résoudre le PB1 pendant l'évaluation, nous observons que neuf (GS2E1, GS2E2, GS2E3, GS2E4, GS2E6, GS2E7, GS2E8, GS2E9, GS2E11) sur 12 utilisent de façon opérationnelle la commande « quand drapeau-vert cliqué » au début du programme du personnage « enfant ». Deux (GS2E10 et GS2E12) sur 12 ne l'utilisent pas. Le programme collecté pour l'élève GS2E5 comprend la commande en question, mais nous avons vu que pendant l'évaluation, il écrivait son programme en regardant le programme de l'élève GS2E7. C'est pourquoi, nous avons décidé de ne pas l'inclure dans la liste des élèves ayant utilisé de manière opérationnelle cette commande au PB1.

En analysant les programmes créés par les élèves pour résoudre le PB2, nous pouvons relever que neuf élèves (GS2E1, GS2E2, GS2E3, GS2E4, GS2E6, GS2E7, GS2E8, GS2E9, GS2E11) sur 12 arrivent à utiliser de manière opérationnelle la commande « quand drapeau-vert est cliqué » au début du programme des personnages « enfant » et « lapin ». GS2E10 ne l'utilise de façon opérationnelle qu'au début du personnage « lapin ». Deux élèves (GS2E5 et GS2E12) sur 12 ne l'utilisent pas dans les programmes de leurs personnages.

1.5.3. Des interprétations

Suite à la description précédemment faite des actions des élèves, nous pouvons donc supposer que la majorité des élèves de GS2 ont réussi à comprendre la fonctionnalité de la commande « quand drapeau-vert cliqué », puisqu'ils arrivent à l'utiliser de manière opérationnelle pour résoudre les PB1 et PB2. Nous pouvons également constater que les élèves GS2E5, GS2E10 et GS2E12 sont ceux qui se trompent sur la commande « quand drapeau-vert cliqué » aussi bien sur le PB1 que sur le PB2.

Nous pouvons donc faire l'hypothèse que ces trois élèves ont le plus des difficultés avec cette commande. En analysant la sémantique de la commande « quand drapeau-vert cliqué » nous observons qu'elle est liée à la compréhension de la notion de séquence, car les élèves doivent réfléchir à l'ordre de réalisation des actions de leur personnage dans la scène pour comprendre que la commande « quand drapeau-vert cliqué » représente toujours le début d'un programme.

La compréhension du temps paraît également essentielle. En prenant en compte ces éléments, nous pouvons supposer que ces élèves dominent moins bien que les autres les notions de la séquence et de temps et c'est pourquoi ils ont des difficultés avec l'utilisation de la commande « quand drapeau-vert cliqué ».

De plus, le fait que les programmes puissent être exécutés sur ScratchJr même lorsque la commande « quand drapeau-vert cliqué » n'en fait pas partie, empêche les élèves de comprendre la nécessite de son utilisation dans leurs programmes. Pour surmonter cet obstacle nous avons demandé aux élèves, pendant l'enseignement, de présenter leurs projets en mode plein-écran à la fin de la résolution de chaque problème donné. Le mode plein-écran oblige à considérer l'utilisation de la commande « quand drapeau-vert cliqué » dans sa programmation, puisque la seule manière d'exécuter un projet en mode plein-écran est à l'aide du bouton « drapeau-vert » en haut de la scène et cela ne sera pas possible s'il n'a pas au moins une commande « quand drapeau-vert cliqué » dans un des programmes du projet.

Il est donc possible que ces trois élèves oublient de passer par cette étape ou qu'ils n'aient pas encore compris que la commande « quand drapeau-vert cliqué » est nécessaire pour qu'ils puissent exécuter leur projet en mode plein-écran.

1.5.4. Analyse des programmes d'élèves en fin d'intervention

En ce qui concerne la commande de « fin », nous observons que 10 élèves (GS2E1, GS2E2, GS2E3, GS2E4, GS2E6, GS2E7, GS2E8, GS2E10, GS2E11, GS2E12) sur 12 l'utilisent de façon opérationnelle dans les programmes des personnages « enfant » et « papillon » au PB1. GS2E9 ne l'utilise que dans le programme de l'un de ses deux personnages au PB1. Le programme collecté pour l'élève GS2E5 comprend la commande en question, mais nous avons vu que pendant l'évaluation il écrivait son programme en regardant le programme de l'élève GS2E7. C'est pourquoi, nous avons décidé de ne pas l'inclure dans la liste des élèves ayant utilisé de manière opérationnelle la commande de « fin » au PB1.

En analysant les programmes créés par les élèves pour résoudre le PB2, nous repérons que six élèves (GS2E1, GS2E2, GS2E3, GS2E4, GS2E8, GS2E9) sur 12 l'utilisent à la fin du programme des personnages « enfant » et « lapin », trois élèves (GS2E6, GS2E7, GS2E11) l'utilisent à la fin du programme de l'un des deux personnages et trois élèves (GS2E05, GS2E10, GS2E12) ne l'utilisent pas.

1.5.5. Des interprétations

Nous pouvons supposer que la majorité des élèves ont compris la sémantique de la commande de « fin » puisqu'ils arrivent à l'utiliser de manière opérationnelle surtout dans le PB1 mais aussi au PB2. Nous observons également que plus d'élèves de GS2 utilisent de manière opérationnelle la commande de « fin » au PB1 qu'au PB2.

Une première hypothèse pourrait être que les élèves de maternelle sont plus fatigués lorsqu'ils arrivent à résoudre le PB2 et c'est pourquoi ils oublient plus d'utiliser la commande de « fin » au PB2 qu'au PB1. De plus, puisqu'il s'agit d'une commande qui n'influence pas le comportement du personnage concerné, nous supposons que c'est plus facile pour les élèves de l'oublier, même si nous avons fait le choix de l'enseigner en tant que groupe « début-fin » avec la commande « quand drapeau-vert cliqué », pour faciliter la compréhension de la séquence.

De la même manière que la commande « quand drapeau-vert cliqué », la sémantique de « fin » est liée à la compréhension de la notion de séquence par les élèves, car ces derniers doivent réfléchir à l'ordre de réalisation des actions par leur personnage dans la scène pour comprendre que la commande « fin » représente toujours la fin d'un programme. La compréhension donc du temps paraît également essentielle. En prenant en compte ces éléments, nous pouvons supposer que les élèves qui n'arrivent pas à utiliser cette commande ont plus de difficultés avec ces notions que les autres.

1.6. La gestion du démarrage synchrone de deux personnages par les élèves

1.6.1. Quelques mots sur comment faire démarrer deux personnages en même temps sur ScratchJr

Un autre élément qu'il fallait présenter aux élèves car ces derniers n'arrivaient pas à le découvrir tout seuls est le démarrage synchrone de deux personnages (concurrence), à l'aide du bouton du drapeau-vert en haut de la scène.

En effet, ScratchJr nous donne la possibilité de faire réagir plusieurs personnages au même événement. Vu l'âge des élèves nous avons décidé de ne leur enseigner que le démarrage synchrone de deux personnages, mais pas plus. Nous avons enseigné comment faire pour programmer deux personnages à démarrer leur programme lorsque le bouton du drapeau-vert en haut de la scène est cliqué. Nous avons choisi d'enseigner cela pendant la *deuxième séance* d'enseignement, après avoir enseigné la commande « quand drapeau-vert cliqué ».

Pour rappel, cette commande sert à démarrer le programme d'un personnage lorsque le bouton du drapeau-vert en haut de la scène est cliqué, c'est pourquoi elle nous a paru un prérequis nécessaire pour la compréhension du démarrage synchrone de deux personnages. En effet, quand deux personnages ont la commande « quand drapeau-vert cliqué » au début de leur programme, ces derniers seront déclenchés en même temps si le bouton du drapeau-vert en haut de la scène est cliqué.

1.6.2. Enseignement du démarrage synchrone

Pour enseigner le démarrage synchrone de deux personnages, nous avons d'abord demandé aux élèves de chercher comment ils peuvent démarrer les programmes de deux personnages en même temps. Ils nous ont proposé d'utiliser la commande « Start on bumb » ou « Démarrer lorsque je suis touché » en français. Cette commande a deux bonhommes sur son bloc l'un à côté de l'autre. Ils ont donc repéré une commande ayant deux personnages sur son bloc et ils ont pensé qu'elle sert à faire lancer les programmes de deux personnages en même temps. C'était une idée intéressante même si ce n'était pas correct.

Ensuite, nous leur avons présenté le bouton du drapeau-vert en haut de la scène et nous les avons invités à l'utiliser pour voir ce qu'elle fait. Les élèves ont repéré qu'en appuyant sur celui-ci, les programmes de leurs personnages ne se lançaient pas en même temps. Nous avons donc rappelé aux élèves qu'il fallait ajouter les commandes « quand drapeau-vert est cliqué » et « fin » dans le programme de deux personnages, pour réussir à les lancer en même temps à l'aide du bouton du drapeau-vert en haut de la scène.

1.6.3. Observation des élèves

Un des élèves nous a répondu pendant la séance d'enseignement ce qui suit : « il peut aussi commencer avec les bleus »[124]. Il s'agit d'une remarque importante d'autant plus parce qu'elle vient d'un élève de grande section de maternelle. Ce dernier se réfère ici aux commandes de déplacement qui sont de couleur bleue et il a raison, parce qu'un programme peut être exécuté sur ScratchJr même lorsqu'il ne comprend que des commandes de « déplacement » et pas de « drapeau-vert » et de « fin ».

Les créateurs du logiciel ont voulu faciliter l'accès des jeunes élèves à la programmation au point où ils ont fait en sorte que le programme d'un personnage peut être déclenché de deux façons possibles. La première c'est en cliquant sur le drapeau-vert en haut de la scène après avoir ajouté la commande du « quand drapeau-vert cliqué » au début du programme du personnage que nous souhaitons lancer. La deuxième façon c'est en appuyant avec le doigt sur le programme du personnage dont nous souhaitons lancer le programme, même lorsque celui-ci ne contient pas la commande « quand drapeau-vert cliqué » à son début. L'élève a signalé ici la possibilité de lancer le programme du personnage même quand ce dernier ne contient que des commandes de déplacement.

Cela est en même temps un des avantages et une des limites du logiciel ScratchJr. Un avantage, parce que l'exécution des programmes est très simple à faire, les élèves y ont facilement accès et ainsi, ils arrivent à comprendre la logique de la programmation sur ScratchJr. Une limite, parce que cela empêche les élèves de comprendre le besoin d'avoir une commande pour signaler le début et une pour la fin de leur programme, éléments essentiels notamment si nous souhaitons aller plus loin en programmation avec eux et leur faire

124 Notre traduction du grec : « Και με τα μπλέ ξεκινάει ».

découvrir le démarrage synchrone de deux personnages (programmation concurrente) ou les messages.

À l'issue de la troisième séance d'enseignement nous avons eu besoin de rappeler aux élèves le démarrage synchrone de deux personnages puisque seulement deux élèves avaient retenu ce processus de la séance précédente.

1.6.4. Analyse des programmes d'élèves en fin d'intervention

À la fin du scénario pédagogique, nous avons évalué la capacité des élèves à réaliser le démarrage synchrone de deux personnages à travers la résolution du PB2. Ils devaient programmer les personnages « enfant » et « lapin » pour démarrer leur programme en même temps. En regardant les programmes que ces derniers ont créés pendant la séance d'évaluation pour résoudre le PB2, nous observons que neuf élèves (GS2E1, GS2E2, GS2E3, GS2E4, GS2E6, GS2E7, GS2E8, GS2E9, GS2E11) sur 12 sont arrivés à réaliser le démarrage synchrone de deux personnages.

1.6.5. Des interprétations

Nous pouvons donc supposer qu'ils ont compris comment faire démarrer le programme de deux personnages en même temps sur ScratchJr, puisqu'ils ont réussi à le faire pendant l'évaluation pour résoudre le PB2. Trois élèves (GS2E5, GS2E10 et GS2E12) sur 12, n'ont pas pu le mettre en place. Nous avons repéré qu'il s'agit de ceux qui avaient des difficultés avec l'utilisation opérationnelle de la commande « quand drapeau-vert cliqué ». Nous pouvons donc faire l'hypothèse qu'ils ont des difficultés avec le démarrage synchrone de deux personnages, parce qu'ils n'ont pas encore bien saisi non plus le démarrage du programme d'un personnage avec le bouton du « drapeau-vert » en haut de la scène.

1.7. La commande préférée par les élèves : l'enregistrement d'un son

1.7.1. Quelques mots sur la fonctionnalité de la commande

Nous avons aussi dû présenter aux élèves la commande « d'enregistrement d'un son », parce qu'ils n'arrivaient pas à le découvrir sans notre aide. Cette commande appartient à la catégorie

des commandes de « son » et elle sert à jouer le son que l'utilisateur a enregistré. L'utilisateur doit d'abord enregistrer un son (processus d'enregistrement) en utilisant le menu qui apparaît lorsqu'il appuie sur la commande, ensuite il doit déposer la commande qu'il a créée dans l'espace de programmation et l'ajouter dans le programme de son personnage pour pouvoir faire jouer le son qu'il a enregistré par le personnage concerné.

1.7.2. Observation des élèves

Nous avons aussi observé que les élèves cherchent parfois la commande « quand drapeau-vert cliqué » dans la catégorie des commandes de « son », parce que cette dernière est aussi de couleur verte. En effet, ils confondent la couleur du drapeau-vert avec la couleur de la catégorie des commandes du « son », parce qu'elles sont toutes les deux vertes.

Le processus d'enregistrement d'un son pose occasionnellement problème aux élèves parce qu'ils se trompent avec les différents boutons du menu d'enregistrement et au lieu d'enregistrer le son souhaité ils recommencent du début l'enregistrement. Nous avons aussi observé que les élèves s'amusent tellement lorsqu'ils écoutent l'enregistrement qu'ils ont produit en étant dans la catégorie des commandes de « son », qu'ils en oublient de le déposer dans l'espace de programmation.

De plus, pendant la séance d'évaluation nous avons remarqué que certains élèves s'amusaient aussi tellement avec l'enregistrement d'un son qu'ils créaient plusieurs fois l'enregistrement demandé et ainsi, ils utilisaient plusieurs commandes différentes d'enregistrement (par exemple enregistrement 1, enregistrement 2, enregistrement 3, etc.) au lieu d'utiliser la même commande plusieurs fois.

1.7.3. Analyse des programmes d'élèves en fin d'intervention

En analysant les programmes créés par les élèves au PB2, nous pouvons repérer que sept élèves (GS2E1, GS2E2, GS2E3, GS2E4, GS2E9, GS2E11, GS2E12) sur 12 ont créé de deux à quatre commandes d'enregistrements différentes. Les élèves GS2E7, GS2E8 et GS2E10 sont les seuls à avoir créé une seule commande d'enregistrement.

Nous avons décidé d'évaluer la capacité des élèves à utiliser la commande d' « enregistrement d'un son » à l'aide du PB2. Puisque nous n'avons accès qu'aux captures d'écran des projets des élèves de GS2, nous ne pouvons pas savoir si le son enregistré par les élèves est correct

d'après l'énoncé du PB2. C'est pourquoi, nous pouvons seulement étudier si les élèves utilisent ou pas la commande de l' « enregistrement d'un son » dans leurs programmes.

En analysant les programmes créés pour le PB2, nous repérons que dix élèves (GS2E1, GS2E2, GS2E3, GS2E4, GS2E5, GS2E7, GS2E8, GS2E9, GS2E10, GS2E11, GS2E12) sur 12 ont utilisé la commande d' « enregistrement d'un son » dans le programme du personnage « lapin ». Deux (GS2E5 et GS2E6) sur 12 ne l'utilisent pas dans le programme de leur personnage.

1.7.4. Des interprétations

Suite à nos observations, nous pouvons donc supposer que la majorité des élèves de notre échantillon avait compris comment enregistrer un son à travers cette commande. Nous pouvons également faire l'hypothèse que les deux qui n'arrivent pas à l'utiliser avaient des difficultés à retenir les étapes qu'il fallait suivre pour enregistrer un son à l'aide du menu qui apparaît lorsque nous appuyons sur cette commande sur la palette des commandes.

Pour conclure sur la gestion de l'interface et l'appropriation des commandes enseignées par les élèves, nous remarquons que la majorité des élèves du GS2 arrive sans problème à gérer les éléments d'interface, l'écriture des programmes, le démarrage et la fin d'un programme ainsi que le démarrage synchrone de deux personnages et l'enregistrement d'un son. En revanche, ils ont des difficultés avec le déplacement des personnages et en particulier avec la commande « sauter », avec l'initialisation de la position des personnages et le débogage. Dans la section qui suit, nous allons étudier la « répétition prédéfinie », une des commandes qui posent la plupart des problèmes aux élèves.

2. Répétition prédéfinie : une commande difficile à maitriser

Dans cette section, nous allons nous concentrer sur l'initiation des élèves de GS2 à la commande de « répétition ». Les résultats qui suivent proviennent comme ceux de la section précédente de nos notes des observations du terrain et de l'analyse des programmes élaborés par les élèves pendant la séance d'évaluation. Puisqu'il s'agit d'une commande jusqu'ici peu étudié au niveau de la petite enfance, nous avons aussi voulu étudier ce que disent les élèves

autour de celle-ci à la fin du scénario pédagogique. C'est pourquoi, nous avons décidé de regarder aussi les réponses que les élèves ont données aux questions du Post-test qui concernent la commande de « répétition ».

2.1. Quelques mots sur la fonctionnalité de la commande

La répétition est une des structures fondamentales en programmation. Le logiciel ScratchJr nous donne la possibilité de travailler avec les très jeunes élèves sur deux types de répétition : la répétition d'un nombre de fois prédéfini et la répétition indéfinie. Il s'agit des commandes « répéter » et « répéter indéfiniment ». En regardant l'interface de ScratchJr, nous repérons que ces deux commandes ne font pas partie de la même catégorie. La commande « répéter indéfiniment » est un bloc qui devrait être placé à la fin d'un programme, d'où son positionnement dans la catégorie de commandes de « fin », qui sont d'une couleur rouge. La commande « répéter » est un bloc incomplet qui pourrait, comme les commandes de déplacement d'ailleurs, être placé à n'importe quel endroit dans un programme. Cette commande permet de répéter un nombre de fois prédéfini la ou les commandes qui s'y trouvent.

Pour le présent travail de thèse, nous avons choisi de n'étudier que la commande « répéter » qui sert à répéter la commande ou les commandes qui s'y trouvent à son intérieur un nombre de fois prédéfini. Pour plus de facilité, nous allons aussi nous référer à celle-ci en tant que « répétition prédéfinie » ou « répétition ».

2.2. Enseigner la répétition : un défi à relever

La « répétition prédéfinie » est une des commandes qu'il faut absolument présenter aux élèves, car ils ne peuvent pas découvrir seuls sa fonctionnalité. Son enseignement était même un défi pour nous. Nous rappelons que pour faciliter la compréhension de cette commande par les élèves nous avons dédié la *quatrième séance* du scénario à l'enseignement de la répétition d'une seule commande et la *cinquième séance* à la répétition d'un motif des commandes.

Nos observations du terrain montrent que les commandes qui aident le plus les élèves à découvrir la fonctionnalité de la « répétition prédéfinie » sont celles de « son ». En effet, les élèves comparent, à l'aide de la chercheuse, le nombre de fois qu'ils écoutent le son quand ils exécutent sa commande et le nombre de fois qu'ils l'écoutent quand sa commande est ajoutée

dans la répétition et ils arrivent à repérer qu'il y a une différence. Ensuite, ils arrivent à voir que cette différence provient de la présence de la commande de « répétition prédéfinie » puisqu'ils font le lien entre le nombre de fois qu'ils ont écouté le son quand celui-ci était dans la répétition et le paramètre de cette dernière. De cette façon, les élèves arrivent à découvrir la fonctionnalité de cette commande. Ainsi, ils arrivent pendant la séance d'enseignement à saisir que le paramètre de la « répétition prédéfinie » signifie le nombre des fois que la commande qui s'y trouve sera exécutée par le personnage concerné.

Pour aider les élèves à découvrir la fonctionnalité de la commande de « répétition prédéfinie » nous avons aussi utilisé les commandes de « déplacement ». Néanmoins, les élèves n'ont pas réagi aussi bien qu'avec la commande de « son ». En effet, nous avons observé que c'était plus difficile pour eux de repérer la différence entre le nombre de pas que leur personnage réalisait dans la scène après avoir exécuté la commande de « déplacement » et le nombre de pas qu'il réalisait lorsque nous l'avons ajoutée dans la commande de « répétition ». La raison de ces difficultés pourrait être la vitesse par défaut de l'exécution des commandes de « déplacement » sur ScratchJr. Les élèves n'arrivaient pas à compter le nombre de pas que le personnage réalisait dans la scène quand la commande de « déplacement » était à l'intérieur de la « répétition prédéfinie ».

Ce qui a aidé beaucoup des élèves dans ce processus était la « grille ». Il s'agit d'un bouton de l'interface qui active et désactive le quadrillage avec les axes x et y de la scène. Nous avons remarqué que la « grille » a beaucoup aidé les élèves à voir que leur personnage se déplaçait de plus de pas quand la commande de « déplacement » était ajoutée dans celle de la « répétition ». De cette manière, ils ont pu faire le lien entre le paramètre de la « répétition » et le nombre des pas que leur personnage réalisait lorsque la commande de « déplacement » était dans celle de « répétition » et ainsi découvrir ce que cette nouvelle commande fait.

2.3. Des difficultés rencontrées face au nombre de répétitions à réaliser

La répétition d'un déplacement a également posé problème aux élèves pendant la séance d'enseignement. En effet, nous avons observé que la présence d'une commande de « déplacement » ayant un paramètre dans une commande de « répétition prédéfinie » qui possède aussi un paramètre fixé à quatre par défaut, perturbait beaucoup les élèves de GS2.

Malgré l'effort que nous avons fait pendant l'enseignement à leur expliquer que lorsque nous ajoutons une commande de « déplacement » dans la « répétition prédéfinie » nous ne modifierons que le paramètre de la dernière, certains élèves continuaient, pendant les deux séances que nous avons dédiées à cette notion, à modifier les paramètres de deux commandes.

Cela posait ensuite des problèmes à leur programmation car le résultat obtenu après l'exécution correspondait à la multiplication de deux paramètres et ce n'était donc pas ce qui était souhaité. Les élèves ne comprenaient pas pourquoi leur programme n'était pas le programme souhaité. Il est possible que la difficulté de ces derniers relève du fait que la commande de « répétition » représente l'opération de la multiplication, qui n'est pas encore maitrisée à cet âge-là, car elle dépasse le niveau cognitif et développemental des élèves de maternelle.

De plus, nous avons remarqué que les élèves de GS2 avaient aussi des difficultés avec la répétition d'un enregistrement de son. En effet, dans le problème que nous leur avons présenté pendant la quatrième séance du scénario, il fallait programmer l'un des deux personnages à répéter le mot « bonjour » trois fois. Etant donné qu'on venait de présenter la fonctionnalité de cette commande aux élèves, nous supposons qu'ils n'avaient pas encore bien saisi ce qu'elle faisait. Nous avons donc observé que lorsque les élèves créaient la commande du « son », ils enregistraient le mot « bonjour » trois fois dans le même enregistrement. Ensuite, ils l'ajoutaient dans la commande de « répétition » et leur personnage disait « bonjour » neuf fois au lieu de trois, suite à l'exécution de la commande. Le comportement des élèves nous montre également qu'ils n'avaient non plus compris le rôle du paramètre de la « répétition ».

Nous avons également remarqué que les élèves étaient en difficulté lorsque, dans le problème à résoudre, le personnage réalisait une autre action après la commande de « répétition ». Nous leur avons proposé un tel problème pendant la quatrième séance et nous avons rapidement compris que c'était très difficile pour eux. À partir de ce moment-là, nous ne leur avons donné que des problèmes où le personnage concerné devrait seulement répéter une action ou une suite d'actions.

2.4. Des difficultés rencontrées face à l'identification et l'écriture du motif des commandes à répéter

Au cours de la cinquième séance du scénario, nous avons enseigné aux élèves la répétition d'une séquence d'actions un nombre de fois prédéfini. Peu d'élèves se souvenaient de ce que fait la commande de « répétition prédéfinie », même si nous avions présenté sa fonctionnalité pendant la quatrième séance du scénario. Une élève, GS2E9, nous a dit qu'avec cette commande, le personnage « peut se déplacer ou parler le nombre de fois que le numéro ici dit »[125].

Nos observations de terrain montrent que la répétition d'un motif des commandes a posé beaucoup de problèmes aux élèves. Tout d'abord, nous avons repéré qu'ils ne connaissaient pas la signification du mot motif, il fallait donc commencer l'expliquer. Ensuite, nous leur avons présenté un projet déjà prêt où le personnage réalisait trois fois le même motif des commandes et nous leur avons demandé de trouver ce qui se répétait dans son programme. Nous avons observé que les élèves se concentraient individuellement sur chaque commande qui était répétée mais sur le motif des commandes. Ils avaient donc du mal à identifier la suite d'actions qui se répétait dans le programme donné.

Les cartes que nous avons créées pour représenter les commandes du logiciel ont beaucoup aidé les élèves à repérer le motif des commandes qui se répétait. En effet, nous leur avons demandé de les utiliser pour construire le programme donné et ensuite, nous leur avons demandé à nouveau d'identifier ce qui se répétait et cette fois-ci, ils arrivaient beaucoup mieux qu'avant à le faire.

Nous avons néanmoins repéré que lorsqu'il fallait reproduire le programme donné en utilisant la commande de « répétition » certains élèves avaient des problèmes à définir le motif des commandes à ajouter dans la « répétition », même s'ils l'avaient déjà identifié à l'aide des cartes. En effet, ils ajoutaient les trois motifs des commandes dans la commande de « répétition » au lieu d'ajouter un seul motif. Cela posait problème à leur programme puisque le résultat obtenu suite à l'exécution n'était pas le résultat souhaité, mais les élèves n'arrivaient pas à comprendre pourquoi. Nous supposons donc qu'ils n'avaient toujours pas bien compris la fonctionnalité de la « répétition ».

125 Notre traduction du grec :« Μπορεί να περπατάει και να μιλάει όσες φορές λέει ο αριθμός εδώ ».

2.5. Analyse des programmes d'élèves en fin d'intervention

Pour évaluer la capacité des élèves à utiliser la commande de « répétition prédéfinie », nous leur avons donné un problème à résoudre. Pour les aider à résoudre le PB2, nous leur avons donné les cartes correspondant aux commandes du logiciel que nous avions utilisées pendant les séances d'enseignement pour découvrir la fonctionnalité de la commande. Les élèves ont choisi de ne pas utiliser les cartes pour préparer la résolution du problème. Ils sont passés directement au logiciel pour programmer leurs personnages.

Avant d'aller plus en avant dans l'analyse, nous souhaitons rappeler que nous sommes à plusieurs reprises intervenue pendant la séance d'évaluation pour aider les élèves à avancer. Notre intervention consistait à : rappeler la consigne du problème à résoudre, lancer le projet modèle en plein écran pour que les élèves puissent voir ce qu'ils devaient reproduire, rappeler la position initiale des personnages et poser des questions pour aider les élèves à surmonter une difficulté rencontrée.

En analysant les programmes que les élèves ont élaborés pendant la séance d'évaluation pour résoudre le PB2, nous observons que seulement un élève sur 12 (GS2E7) a pu utiliser la commande de « répétition prédéfinie » de façon opérationnelle. Huit élèves sur 12 (GS2E1, GS2E2, GS2E3, GS2E4, GS2E8, GS2E9, GS2E10, GS2E11) ont réussi à résoudre le PB2, en utilisant trois fois le même motif des commandes au lieu d'utiliser la commande de « répétition ». Un élève (GS2E12) sur 12 n'a pas réussi à résoudre le problème donné, car il a créé le motif des commandes qui se répétait deux fois au lieu de trois et deux élèves sur 12 (GS2E5, GS2E6) n'ont pas réussi à résoudre le problème donné puisqu'ils n'ont pas pu créer le motif des commandes qui se répétait.

En ce qui concerne la répétition d'un enregistrement sonore, nous avons repéré que même pendant la séance d'évaluation certains élèves continuaient à se tromper en enregistrant trois fois la phrase demandée dans la même commande d'« enregistrement d'un son » et en l'ajoutant ensuite trois fois dans le programme de leur personnage. Ainsi, leur programme était erroné puisque leur personnage répétait la phrase demandée neuf fois au lieu de trois. Nous pouvons donc constater que certains élèves de maternelle n'ont pas réussi jusqu'à la fin du scénario à surmonter la difficulté qu'ils ont eue pendant les séances d'enseignement avec

la répétition d'enregistrement sonore. Nous supposons qu'ils n'avaient toujours pas compris à quoi ça sert la commande de « répétition ». Nous ne sommes malheureusement pas en mesure de savoir combien d'élèves ont réussi à enregistrer correctement la commande de « son » pour résoudre le PB2.

Au vu de ces résultats, nous pouvons constater que la majorité des élèves (8 sur 12) de notre échantillon a réussi à créer trois fois le motif des commandes pour résoudre le PB2. Nous pouvons supposer que l'élève GS2E7 est le seul qui a compris la fonctionnalité de la commande de « répétition prédéfinie » puisqu'il est seul qui arrive à l'utiliser pour résoudre le PB2.

Nous pouvons également faire l'hypothèse que les huit élèves qui ont choisi de créer trois fois le même motif des commandes au lieu d'utiliser la commande de « répétition » n'ont pas compris l'organisation de l'écriture de la commande et c'est pourquoi ils ont préféré éviter son utilisation. Il est également possible que ces derniers aient compris à quoi elle sert mais son utilisation leur paraissait difficile et c'est pourquoi ils l'ont évitée. Il est néanmoins important de signaler qu'ils ont réussi à trouver une autre façon de programmer le comportement du personnage « lapin ».

L'élève GS2E12 a eu plus de difficultés que les autres avec la résolution de ce PB2, parce qu'il a commencé à créer le motif et à le répéter mais il n'est pas arrivé à le finir. Les élèves GS2E5 et GS2E6 sont ceux qui ont eu les plus des difficultés, car ils n'ont pas réussi à créer le motif qui se répétait.

Une des raisons de ces difficultés est la complexité de la nature de la commande. La « répétition prédéfinie » est une commande de nature très différente des autres commandes de l'univers de programmation de ScratchJr. Pour utiliser la « répétition prédéfinie » de façon opérationnelle, les élèves doivent avant tout avoir compris qu'elle sert à faire faire au personnage choisi une action de façon répétée ou une suite d'actions un nombre prédéfini de fois par son paramètre. Ils doivent ensuite identifier le type d'action ou d'actions que le personnage choisi répétera dans la scène. S'il s'agit d'une suite d'actions qui est répétée, ils doivent également réfléchir à l'ordre temporel d'exécution de ces actions par le personnage choisi étant donné que le positionnement des commandes dans un programme sur ScratchJr définit l'ordre d'exécution de ces derniers par le personnage choisi. La compréhension de l'exécution séquentielle des commandes sur ScratchJr paraît donc essentielle. Enfin, les élèves

doivent penser au nombre de fois que cette action ou cette suite d'actions devrait être répétée, élément qui est défini par le paramètre de la « répétition ». Il est possible que la difficulté des élèves relève du fait que la commande de « répétition » représente l'opération de la multiplication, qui n'est pas encore maitrisée à cet âge-là, car elle dépasse le niveau cognitif et développemental des élèves de maternelle.

Nous supposons également que ces difficultés sont dues à la distance existante entre l'écriture de la commande « répétition prédéfinie » et ce qu'elle fait. En effet, ce n'est pas l'action de répétition qui pose problème aux élèves mais la forme d'écriture de celle-ci à l'aide de la commande fournie par ScratchJr. Ils n'arrivent pas à prendre en compte tous ces éléments lorsqu'ils doivent écrire une « répétition prédéfinie » et c'est pourquoi il est difficile pour eux de la mettre en place de manière opérationnelle. C'est un défi qu'ils n'arrivent pas à surmonter.

Une autre raison pour laquelle la commande de « répétition prédéfinie » paraît difficile aux élèves est le fait que suite à son exécution, la réaction du personnage concerné dans la scène varie en fonction du contenu de la commande qui n'est pas spécifique. Nous supposons que ce manque de stabilité dans la réaction du personnage que les élèves observent dans la scène, pourrait les empêcher de saisir la fonctionnalité de cette commande, surtout parce que c'est la seule des commandes enseignées dans notre scénario ayant cette particularité. Toutes les autres commandes obligent le personnage concerné à réaliser la même action spécifique chaque fois qu'ils sont exécutés et ainsi leur sémantique est fixe.

De plus, la commande de « répétition prédéfinie » sert d'une certaine manière à raccourcir une programmation qui pourrait sinon être longue. Le choix des élèves à créer trois fois le motif des commandes au lieu d'utiliser la « répétition » nous montre que ces derniers ont besoin de voir la version longue de la programmation, c'est-à-dire le motif qui est répété et le nombre de fois qui est répété. De cette façon, ils arrivent à mieux gérer ce que fait leur personnage.

Enfin, il est possible que ces difficultés soient dues à une mauvaise compréhension de la consigne. Il existe une certaine ambiguïté dans le discours de la consigne du problème. Nous supposons que les élèves n'ont pas bien compris dans la consigne ce qui se répétait.

2.6. Analyse des réponses d'élèves au Post-test

À part le problème à résoudre que nous avons donné aux élèves pendant l'évaluation, nous leur avons aussi posé un certain nombre des questions sur la commande de « répétition prédéfinie » lors de l'entretien individuel (Post-test). En consultant les réponses des élèves aux questions posées sur la commande de « répétition prédéfinie » pendant le Post-test, nous pouvons noter une série d'éléments.

Tout d'abord, la majorité des élèves, huit sur 12 (GS2E2, GS2E3, GS2E4, GS2E7, GS2E8, GS2E9, GS2E10, GS2E12) ont répondu correctement que si on appuyait[126] sur la commande de « répétition » pendant qu'elle était vide, celle-ci n'aurait aucun effet sur le personnage choisi. Trois élèves (GS2E1, GS2E5, GS2E6) ont répondu qu'ils ne savaient pas ce qui allait se passer si la commande était vide et une élève (GS2E11) a répondu :

> GS2E11 : « Si le personnage choisi était vers ce sens, il réaliserait quatre pas vers l'autre sens ».

Au vu de ces réponses, nous pouvons supposer que la majorité des élèves de notre échantillon a compris qu'il faut ajouter au moins une commande dans la « répétition » pour que cette dernière fonctionne. Les élèves GS2E1, GS2E5, GS2E6 et GS2E11 semblent ne pas avoir compris cet aspect de la sémantique de la commande de « répétition prédéfinie ».

De plus, nous constatons que les élèves de GS2 arrivent mieux à expliquer ce que fait la « répétition » lorsque c'était une seule commande qui se répétait que lorsque c'était une séquence des commandes qui se répétait. En effet, sept élèves (GS2E2, GS2E4, GS2E6, GS2E7, GS2E8, GS2E9, GS2E12) sur 12 ont répondu correctement à la question sur la « répétition prédéfinie d'une seule commande » (voir figure 10), tandis qu'un seul élève sur 12 (GS2E7) a répondu correctement à la question sur la « répétition d'une séquence des commandes » (voir figure 11). Les élèves GS2E1, GS2E3, GS2E5, GS2E10 et GS2E11 n'ont pas répondu correctement ni à la question sur la répétition d'une commande, ni à celle sur la « répétition prédéfinie d'une séquence des commandes ». Nous pouvons donc supposer que ces cinq élèves ont plus des difficultés que les autres à comprendre ce que fait la commande.

126 Pour rappel, l'exécution des commandes et des programmes sur ScratchJr se fait d'habitude en appuyant sur celles-ci avec le doigt.

Figure 10: Les répétitions prédéfinies d'une commande que nous avons présentée aux élèves pendant le Post-test

Figure 11: La répétition prédéfinie d'une séquence des commandes que nous avons présentée aux élèves pendant le Post-test

Les élèves de GS2 ont donné des réponses différentes à la question sur la « répétition prédéfinie d'une séquence des commandes ». GS2E7 a répondu correctement à la question sur la « répétition prédéfinie d'une séquence des commandes » en disant :

> GS2E7 : « Il va marcher une fois et il va parler, il va marcher une fois et il va parler, il va marcher une fois et il va parler et il va s'arrêter »[127].

Cet élève semble avoir compris la sémantique de la commande de « répétition prédéfinie », puisqu'il arrive à décrire sans problème les trois motifs des commandes que le personnage concerné réalisera si on exécute la commande de « répétition » que nous lui avons présentée. Il est donc capable de réaliser une réflexion d'un niveau d'abstraction élevé puisqu'il voit la commande de « répétition » et il arrive à imaginer ce qu'elle représente en termes d'actions pour le personnage choisi. Il est aussi capable de réfléchir à la succession d'actions que son personnage ferait dans le temps sans les voir.

Quatre élèves sur 12 (GS2E2, GS2E4, GS2E6, GS2E12) ont répondu que le personnage choisi va marcher trois pas et qu'il va parler trois fois. Une de ces élèves (GS2E2) a aussi donné une deuxième réponse :

> GS2E2 : « D'abord, il va se déplacer d'un pas vers là-bas et ensuite, il va parler trois fois »[128].

Les autres élèves ont répondu :

127 Notre traduction du grec : « Θα προχωρήσει μια φορά και θα μιλήσει, θα προχωρήσει μια φορά και θα μιλήσει, θα προχωρήσει μια φορά και θα μιλήσει και μετά θα σταματήσει ».

128 Notre traduction du grec : « Πρώτα θα παει ένα βήμα προς τα εδώ και μετά θα μιλήσει τρεις φορές ».

GS2E8 : « Il va faire un pas vers là-bas et il va peut-être chanter trois fois »[129].

GS2E9 : « Il va marcher trois fois et il va chanter une chanson »[130].

GS2E3 : « Il va se déplacer trois fois à droite et il va reculer de trois pas »[131].

GS2E11 : « Il va marcher de trois pas, puis encore un et ensuite il va dire quelque chose »[132].

GS2E10 : « Il va parler deux fois et il va aller un pas à droite »[133].

Enfin, deux élèves (GS2E1 et GS2E5) n'ont pas pu répondre à la question. En nous focalisant sur les réponses des élèves à la question sur la « répétition d'une séquence des commandes », nous pouvons en tirer une série des constats. Tout d'abord, nous repérons qu'à part les deux élèves (GS2E5, GS2E1) qui n'ont pas répondu à la question et celui (GS2E10) qui a dit que le personnage va parler deux fois et qu'il va se déplacer une fois, tous les autres élèves ont d'abord parlé du déplacement que le personnage ferait et ensuite de ce qu'il allait dire. Cela nous amène à supposer que la majorité des élèves de maternelle ont compris à la fois que le positionnement des commandes dans un programme sur ScratchJr définit l'ordre d'exécution de celles-ci, mais aussi que les programmes s'exécutent de la gauche vers la droite sur ce logiciel.

Nous ne pouvons pas dire la même chose pour l'élève GS2E10, puisque sa réponse indique le contraire. De plus, nous observons qu'il ne se rappelle pas que lorsqu'on ajoute une commande de « déplacement » dans la « répétition » nous ne considérons plus son paramètre car c'est celui de la « répétition » qui compte. Puis, il dit que le personnage choisi parlera deux fois même si le chiffre deux n'apparaît nulle part sur la commande de « répétition » que nous lui avons présentée.

Nous pouvons également relever dans les réponses des élèves GS2E2, GS2E4, GS2E6 et GS2E12 qu'ils se concentrent sur chaque commande individuellement et non pas sur la séquence des commandes. En effet, ils associent le paramètre de la « répétition » avec chacune de deux commandes de la séquence (par exemple il va faire trois pas et il va parler trois fois) et non pas avec la séquence des commandes (par exemple il va faire un pas et il va parler, il va faire un pas et il va parler et il va faire un pas et il va parler), comme GS2E7 fait.

129 Notre traduction du grec : « Θα κάνει ενα βήμα απο εδώ, θα μιλήσει τρεις φορές μαλλον ».
130 Notre traduction du grec : « Θα περπατήσει τρεις φορές και θα πει τραγούδι ».
131 Notre traduction du grec : « Θα μετακινηθεί τρεις φορές δεξιά και θα γυρίσει τρία βήματα πίσω ».
132 Notre traduction du grec : « Θα προχωρήσει τρία βήματα, μετά ενα ακόμα και μετά θα πει κάτι ».
133 Notre traduction du grec : « Θα μιλήσει δυο φορές και θα πάει ένα βήμα δεξιά ».

Nous avons observé le même comportement pendant la séance d'enseignement quand les élèves se concentraient individuellement sur chaque commande du motif qui se répétait et non pas sur le motif entier. Nous supposons qu'il est possible qu'ils n'arrivent pas à réfléchir au motif des commandes qui se répétait parce que penser en même temps à deux commandes peut être difficile pour eux (par rapport à leur âge). Nous imaginons aussi que c'est difficile pour les élèves de maternelle de voir la « répétition d'une séquence des commandes » et de comprendre que celle-ci représente trois motifs des commandes. Cela demande une réflexion d'un niveau d'abstraction assez élevé qui n'est pas facile à cet âge-là.

Il est également possible que les élèves GS2E2, GS2E4, GS2E6 et GS2E12 répondent d'une telle manière parce qu'ils ont peut-être compris que la question posée portait à la fois sur le nombre total de pas que le personnage concerné ferait mais aussi sur le nombre total de fois que ce dernier parlerait, si on exécutait la commande de « répétition ». En relisant les réponses concernant la « répétition d'une séquence de commandes » des élèves sous ce nouveau prisme, nous pouvons dire qu'elles pourront être considérées comme correctes.

Etant donné que GS2E2 a donné deux réponses, nous observons que l'une de deux est la même que celle de l'élève GS2E8. En regardant les réponses de ces deux élèves, nous observons qu'ils associent le paramètre de la « répétition » seulement à l'une de deux commandes du motif, celle du son, puisqu'elles disent que leur personnage fera un pas et qu'il parlera trois fois. Nous supposons qu'elles se trompent en voyant le paramètre de la commande de « déplacement », car elles oublient la règle que nous leur avons apprise selon laquelle lorsque nous ajoutons une commande de « déplacement » dans la « répétition prédéfinie », le seul paramètre qui compte est celui de la « répétition prédéfinie ».

La réponse de l'élève GS2E11 indique que ce dernier a compris que le personnage concerné ferait le nombre de pas qui correspond à l'addition du paramètre de la « répétition » et celui de la commande du « déplacement ». Il est possible que cet élève ait aussi oublié que le paramètre de la commande de déplacement ne compte pas à partir du moment où celle-ci est introduite dans la commande de « répétition ». En plus, GS2E11 n'arrive pas à associer le paramètre de la « répétition » à la commande de l' « enregistrement d'un son ». La réponse donnée par GS2E9 nous montre aussi que cet élève arrive à associer le paramètre de la « répétition » seulement à l'une de deux commandes du motif, celle du « déplacement ».

La réponse donnée par l'élève GS2E3 nous montre qu'il associe aussi le paramètre de la « répétition prédéfinie » seulement avec la commande de « déplacement » et non pas avec les deux commandes du motif qui est répété. Nous supposons qu'il se trompe avec la flèche existant sur la commande de « répétition », car il dit que le personnage concerné reculera de trois pas. En effet, la commande de « répétition prédéfinie » a une flèche sur celle-ci, qui est censée indiquer sa fonctionnalité, mais l'élève a interprété cette flèche comme s'il indiquait que la commande sert à faire reculer le personnage concerné.

Les résultats que nous venons de présenter montrent que la « répétition prédéfinie » est une commande difficile à aborder par les élèves de la maternelle. Elle semble accessible seulement par les plus avancés. Son écriture est très complexe et c'est pourquoi son enseignement est un vrai défi. Les réponses des élèves au Post-test nous aident à comprendre qu'à la fin du scénario pédagogique, la signification de la commande n'est toujours pas comprise par la majorité les élèves. Un seul élève arrive à utiliser de façon opérationnelle la commande de « répétition » pour résoudre le PB2. Cependant, la majorité des élèves arrive à faire répéter par leur personnage la séquence d'actions trois fois sans utiliser la commande de « répétition prédéfinie ». Dans la prochaine section, nous présenterons des résultats sur les « messages », deux commandes qui posent également problème aux élèves de GS2.

3. Les messages : une forme de programmation événementielle qui pose problème

Dans cette section, nous allons nous concentrer sur l'initiation des élèves de GS2 aux commandes de « messages ». Les résultats qui suivent proviennent de nos notes des observations au terrain et de l'analyse des programmes élaborés par les élèves pendant la séance d'évaluation. Nous avons décidé de visualiser aussi les enregistrements vidéo des séances d'enseignement de ces commandes pour deux groupes d'élèves de GS2, pour enrichir nos observations. De plus, puisque les commandes de « messages » sont peu étudiées au niveau de la petite enfance, nous avons voulu étudier ce que disent les élèves de maternelle autour de celles-ci à la fin du scénario pédagogique. C'est pourquoi, nous avons aussi étudié les réponses que les élèves de GS2 ont données à une des questions du Post-test portant sur les « messages ».

3.1. Quelques mots sur la fonctionnalité des commandes

Le logiciel ScratchJr nous donne la possibilité de travailler avec de très jeunes élèves sur différentes formes de programmation événementielle. Comme nous l'avons déjà expliqué auparavant, ce logiciel comporte une catégorie des commandes de démarrage qui servent à définir l'événement auquel le personnage choisi réagira. En effet, lorsque l'événement indiqué sur la commande se déclenche, le programme ayant celle-ci à son début sera exécuté. La commande « quand drapeau-vert cliqué » par exemple sert à démarrer un programme lorsque le bouton du drapeau-vert en haut de la scène est cliqué.

Les commandes de « messages » font partie de cette catégorie de commandes et représentent un autre type d'événement, l'envoi et la réception d'un message. Il s'agit des commandes : « envoyer un message » et la commande « quand message reçu ».

La commande « envoyer un message » ne fonctionne pas de la même manière que les autres commandes de « démarrage ». Au lieu de démarrer le programme du personnage dont elle fait partie, elle sert à le suspendre et à envoyer un message de couleur spécifique à travers duquel elle influence le démarrage des programmes des personnages qui le reçoivent. Elle influence ainsi en même temps le comportement de deux personnages. Étant donné l'âge des élèves et le niveau de difficulté de cette commande, nous avons fait le choix à ne pas mettre l'accent sur la suspension du programme du personnage « expéditeur », parce que cela pourrait compliquer encore plus l'enseignement de la commande à notre public. La forme de cette dernière permet de l'introduire à n'importe quel endroit dans un programme. Comme c'est le cas pour toutes les commandes sur ScratchJr, son positionnement dans le programme définit l'ordre de son exécution.

La commande « quand message reçu » sert à démarrer un programme lorsque le personnage reçoit un message de couleur spécifique. Cette commande est donc associée à un événement déclenché à l'aide d'une autre commande, celle responsable de l'expédition d'un message de couleur spécifique. Le démarrage du programme du personnage « destinataire » ayant au début la commande « quand message reçu » dépend du moment où le personnage « expéditeur » enverra le message. Cela signifie que le démarrage du programme du destinataire dépend du positionnement de la commande « envoyer un message » dans le

programme du personnage « expéditeur », puisque sur ScratchJr les commandes s'exécutent de façon séquentielle de la gauche vers la droite.

3.2. Enseigner les messages : un défi à relever

Les « messages » font partie des commandes qu'il faut absolument présenter aux élèves, car ils ne peuvent pas les découvrir seuls. Leur enseignement était un défi pour nous. Pour rappel, nous avons dédié les deux dernières séances du scénario à leur enseignement. Pendant la *sixième séance*, nous avons enseigné les commandes des « messages » et pendant la *septième* séance, nous avons demandé aux élèves de détecter l'erreur sur un projet déjà prêt contenant ces commandes.

Pour aider les élèves à découvrir la fonctionnalité des commandes de « messages », nous les avons laissés expérimenter en utilisant des projets prêts tout en les accompagnant avec des questions dans ce processus de découverte. De plus, nous avons à nouveau utilisé les cartes représentant les commandes du logiciel que nous avions déjà utilisées pour l'enseignement de la « répétition prédéfinie ». Les cartes ont aidé les élèves de GS2 à créer les programmes de deux personnages et à les visualiser en même temps. Grace à cette visualisation parallèle, les élèves ont pu repérer que ces derniers communiquent grâce à l'envoi et la réception d'un message de couleur spécifique.

Pendant la séance d'enseignement, nous avons observé que certains élèves de GS2 étaient déjà un peu familiarisés avec la réception et l'envoi de courriers dans leur vie quotidienne. Ils avaient par exemple déjà vu leurs parents réceptionner, ouvrir ou envoyer une lettre. Ils savaient qu'une lettre est fermée lorsqu'on veut l'envoyer et une lettre est ouverte lorsqu'on veut la lire. C'est pourquoi, certains élèves arrivaient à reconnaître non seulement les symboles présents sur les commandes des « messages », c'est-à-dire le message fermé et ouverte, mais aussi d'en déduire une signification en se basant sur leurs expériences précédentes.

Cela peut à la fois être un avantage et un obstacle à surmonter pour l'enseignement des commandes des « messages » aux élèves de maternelle. Un avantage parce qu'on pourrait éventuellement utiliser cette connaissance pour aider les élèves à découvrir la fonctionnalité des commandes de « messages ». Un obstacle parce qu'on doit les aider à comprendre que les messages qu'ils connaissaient par leur vie quotidienne sont différents que des « messages »

dans la programmation sur ScratchJr. Nous avons essayé d'utiliser cette connaissance préalable des élèves pendant la séance d'enseignement pour les aider à découvrir ce que font les commandes des « messages » sur ScratchJr.

3.3. Les difficultés rencontrées pendant l'enseignement

En analysant les vidéos de la séance d'enseignement, nous avons également remarqué une difficulté que les élèves ont rencontrée avec les commandes des « messages ». Pour rappel, pendant la sixième séance du scénario, nous avons présenté aux élèves un projet où un personnage « chat » envoyait un message au personnage « fusée » et un projet où un personnage « enfant » envoyait un message au personnage « ballon de foot ». En effet, certains élèves de GS2 avaient des difficultés à saisir le rôle des commandes des « messages » quand ces derniers ont été présentés à l'aide des projets qui leur rappelaient une expérience physique de leur vie quotidienne. Ils pensaient par exemple que le personnage « ballon de foot » se déplaçait, car le personnage « enfant » avait exécuté un shoot et non pas parce qu'il avait reçu un message. Nous présentons ci-dessous deux extraits (1) et (2), provenant de deux groupes d'élèves différents de GS2, qui illustrent ce problème :

Extrait (1) :

ST : « Qu'est-ce qui se passe ici ? »

GS2E6 : « D'abord, l'enfant et puis le ballon. »

ST : « Vous êtes d'accord avec GS2E6 ? »

GS2E7 : « Oui parce que l'enfant court et il shoote dans le ballon de foot et le ballon de foot commence. »

GS1E7 : « Mais maitresse comment ? Le ballon ne peut pas lire ! »

ST : « Exactement. Ces messages sont des commandes, elles ne sont pas des vrais messages ! Le chat ne peut pas écrire un message. »

GS2E6 : « Oui le chat ne peut pas écrire à ma mère par exemple, ni le ballon. »

GS2E7 : « Oui, hahahaha. »

ST : « Les commandes disent aux personnages ce qu'ils doivent faire. »

GS2E7 : « Aaa ! »

Extrait (2) :

ST : « Cela veut dire que la fusée lit le message du chat ? »

GS2E10 : « Non. La fusée ne peut pas lire le message. »

ST : « Pourquoi ? »

GS2E10 : « Non. Parce que c'est une fusée. »

GS2E11 : « Quelqu'un ouvrira la fenêtre de la fusée. »

GS2E12 : « Il va lire le message par la fenêtre de la fusée. »

Ces extraits nous permettent de constater qu'il existe un écart entre l'expérience physique et la programmation sur ScratchJr et cela peut constituer un obstacle à surmonter pour les élèves de maternelle.

Après avoir aidé les élèves à découvrir la fonctionnalité des « messages » sur ScratchJr, nous leur avons demandé de reproduire un des deux projets que nous leur avons présentés. Les élèves avaient tendance à utiliser la commande « quand drapeau-vert cliqué » au début du programme de leurs deux personnages, au lieu d'utiliser la commande « quand drapeau-vert cliqué » au début du programme de l'un personnage et la commande « quand message reçu » au début du programme de l'autre. Cela ne nous a pas surprise puisque jusqu'à ce stade du scénario, ils n'avaient appris à démarrer un programme sur ScratchJr qu'avec la commande « quand drapeau-vert cliqué ». L'extrait (3) qui suit provient de l'élève GS2E7 et il illustre ce propos :

Extrait (3)

ST : « Qui va alors commencer en premier et pourquoi ? »

GS2E7 : « Le chat va commencer en premier. »

ST : « Pourquoi ? »

GS2E7 : « Parce qu'il a un début et une fin ? »

ST : « Et la fusée n'a pas un début et une fin ? »

GS2E7 regarde le programme de la fusée et il répond : « La fusée n'a pas de début et de fin. »

ST : « Et le message ouvert c'est quoi ? Ce n'est pas un début ? »

GS2E7 : « Ah oui ! »

3.4. Analyse des programmes d'élèves en fin d'intervention

Pour évaluer la capacité des élèves à utiliser les commandes de « messages » nous leur avons donné un problème à résoudre à la fin du scénario pédagogique. Pour les aider à résoudre le PB2, nous leur avons donné les cartes correspondant aux commandes du logiciel que nous avions utilisées pendant les séances d'enseignement pour visualiser en même temps les programmes des deux personnages. Les élèves ne les ont pas utilisées pour préparer la résolution de leur problème. Ils sont passés directement au logiciel pour programmer leurs personnages.

Avant de poursuivre l'analyse des programmes créés par les élèves, nous souhaitons rappeler que nous sommes à plusieurs reprises intervenue pendant la séance d'évaluation pour aider les élèves à avancer. Notre intervention consistait à : rappeler la consigne du problème à résoudre, lancer le projet modèle en plein écran pour que les élèves puissent voir ce qu'ils doivent reproduire, rappeler la position initiale des personnages et poser des questions pour aider les élèves à surmonter une difficulté rencontrée.

Nous avons analysé les programmes élaborés par les élèves de GS2 pour résoudre le PB1 sur les « messages » en fonction de deux critères. Nous avons d'abord étudié s'ils ont utilisé les deux commandes de messages « envoyer un message » et « quand message reçu », pour faire communiquer les personnages. Ensuite, nous avons examiné s'ils les ont utilisés dans le bon ordre dans le programme de deux personnages. La figure 12 qui suit présente l'ordre correct des commandes de « messages » selon la consigne du problème donné. Pour rappel, en raison du bas âge des élèves, nous avons décidé de ne pas enseigner l'envoi et la réception des messages des couleurs différentes.

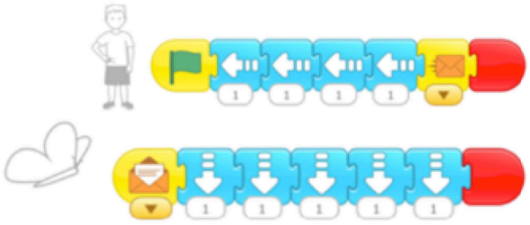


Figure 12: Ordre correct des commandes des messages dans le programme des deux personnages

L'analyse des programmes des élèves selon ces deux critères nous a permis de constater trois types de performance différents. Six élèves sur 12 (GS2E1, GS2E2, GS2E6, GS2E7, GS2E8, GS2E11) ont réussi à utiliser de manière opérationnelle les commandes de « messages ». Ces élèves ont utilisé les deux commandes dans l'ordre correct sur le programme de deux personnages.

Trois élèves sur 12 (GS2E3, GS2E4, GS2E12) sont arrivés à une « utilisation partielle » des commandes des « messages ». En effet, ils ont utilisé de façon opérationnelle la commande « quand message reçu », mais ils se sont trompés sur le positionnement de la commande « envoyer un message » dans le programme de leur personnage « enfant ».

Trois élèves sur 12 (GS2E5, GS2E9, GS2E10) n'ont pas utilisé les commandes des « messages ». GS2E10 a créé une commande d'enregistrement au lieu d'utiliser la commande « envoyer un message » et il l'a ajoutée dans le programme du personnage « papillon », au lieu du programme du personnage « enfant ».

Ces résultats nous font penser que la moitié de notre échantillon s'est approprié la fonctionnalité des commandes de « messages » puisqu'elle a réussi à les utiliser de manière opérationnelle pour résoudre le PB1. Nous pouvons également constater que l'utilisation de la commande « envoyer un message » pose plus de difficultés aux élèves que l'utilisation de la commande « quand message reçu ». En effet, six élèves sur 12 utilisent correctement la commande « envoyer message », tandis que neuf élèves sur 12 utilisent correctement la commande « quand message reçu ». Nous observons que les élèves ont des difficultés en particulier avec le positionnement de la commande « envoyer un message » dans le programme du personnage « enfant ».

Une raison de ces difficultés pourrait être la forme de la commande. Puisqu'il s'agit d'un bloc classique, la commande « envoyer un message » peut être ajoutée à n'importe quel endroit d'un programme. Cela pourrait poser des problèmes aux élèves de maternelle car ils peuvent plus facilement se tromper avec son positionnement qu'avec le positionnement d'un autre type de commande permettant de déclencher ou arrêter un programme.

De plus, pour bien positionner la commande « envoyer un message », les élèves doivent avant tout comprendre ce qu'elle fait. Comme nous l'avons déjà expliqué avant, cette commande sert à envoyer un message de couleur spécifique et cet envoi est un événement provoquant le démarrage du personnage recevant ce message. Nous supposons que l'absence d'un lien visible entre la commande « envoyer un message » et la réaction du personnage « expéditeur » dans la scène rend la compréhension de la sémantique de la commande plus difficile pour les élèves. En revanche, son effet est plus visible sur le comportement du personnage « destinataire » et cela pourrait aussi poser problème aux élèves.

Une autre raison de ces difficultés pourrait être le fait que la commande « envoyer un message » représente, comme on l'a vu ci-dessus, un événement et les élèves de maternelle ne sont pas suffisamment familiarisés avec cette notion. Il est également possible qu'ils aient des difficultés avec la commande « envoyer un message » à cause du fait que le jeu d'enfant-robot que nous avons utilisé pendant la séance de familiarisation avec le logiciel ScratchJr facilitait la compréhension de la programmation du démarrage, du déplacement et de la fin d'un programme par les élèves, mais pas celle de l'envoi et de la réception d'un message. En effet, ce type de jeu donne la possibilité aux élèves de commander leurs camarades pour démarrer, se déplacer sur un quadrillage et de s'arrêter et ainsi de découvrir le principe de commander un personnage pour qu'il se déplace dans la scène. En revanche, ce jeu ne les prépare pas à un autre type de programmation événementielle, l'envoi et la réception d'un message. Pour le faire, il fallait créer une autre activité débranchée et avoir deux enfants « à programmer » sur le quadrillage et un troisième pour les programmer.

De plus, la programmation d'un déplacement constitue la transmission directe d'un ordre dans la mesure où les élèves commandent un personnage à se déplacer dans la scène, alors que la programmation d'un événement, ici l'envoi d'un message, constitue une transmission d'un ordre via un intermédiaire puisque les élèves commandent à un personnage d'envoyer un message pour dire à un autre personnage quand il doit commencer son programme.

Mis à part des difficultés liées à la signification de la commande, nous pouvons réfléchir à d'autres raisons pour lesquelles le positionnement de cette commande pose problème aux élèves. Tout d'abord, comme on l'a vu plus haut, le positionnement de n'importe quelle commande dans le programme d'un personnage sur ScratchJr définit l'ordre d'exécution de celle-ci. La compréhension de l'exécution séquentielle des programmes sur ScratchJr est ainsi un prérequis nécessaire pour l'utilisation opérationnelle de la commande « envoyer un message ». L'ordre d'exécution de cette commande dépend du problème donné. Étant donné que le programme du personnage « destinataire » démarre au moment où le message est envoyé par le personnage « expéditeur », pour bien positionner la commande « envoyer un message » l'élève doit repérer à quel moment de l'exécution du programme de l'« expéditeur » le « destinataire » commence son programme. Pour le faire, il faut soit observer attentivement le projet modèle en plein-écran, soit faire attention à la consigne du problème donné.

Par conséquent, nous pouvons remarquer que le positionnement de la commande « envoyer un message » est un processus difficile à réaliser notamment pour les élèves de maternelle, car il implique une série des connaissances sur la causalité et l'ordre temporel qui ne sont pas encore acquis à cet âge-là. La causalité, parce que l'élève doit comprendre que l'action du premier personnage a un effet sur le démarrage du deuxième. L'ordre temporel, parce qu'un personnage déclenche son programme avant l'autre mais aussi parce que le deuxième personnage démarre son programme au moment où le premier lui envoie le message. Les élèves de maternelle ne sont pas encore capables de dominer ces notions, c'est pourquoi ces dernières constituent un obstacle pour l'utilisation des commandes de « messages » en programmation.

En ce qui concerne la commande « quand message reçu », nous observons que la majorité des élèves (9 sur 12) de notre échantillon arrive à l'utiliser de manière opérationnelle pour résoudre le PB1. Nous pouvons donc supposer que ces élèves ont compris que cette commande peut aussi constituer le début d'un programme sur ScratchJr. Ainsi, nous pouvons constater qu'ils ont surmonté la difficulté qu'ils ont eue, pendant l'enseignement, à reconnaître que cette commande peut aussi servir en tant que début d'un programme et à l'utiliser en même temps que la commande « quand drapeau-vert cliqué » dans un même projet. De plus, nous pouvons faire l'hypothèse que la plupart des élèves a compris que le

deuxième personnage commence son programme une fois qu'il a reçu le message envoyé par le personnage qui a commencé son programme en premier.

Une autre raison pour laquelle la commande « quand message reçu » pose moins de problèmes que la commande « envoyer un message » aux élèves est sa forme. En tant que bloc de « démarrage », la commande « quand message reçu » a une forme qui oblige l'élève à ne la poser qu'au début du programme des personnages. Sa sémantique semble plus facile à comprendre par les élèves, même si elle implique aussi des connaissances sur la causalité et l'ordre temporel, peut-être parce qu'ils arrivent à bien voir la réaction du personnage « destinataire » dans la scène suite à son exécution.

Nous pouvons donc supposer que les six élèves (GS2E1, GS2E2, GS2E6, GS2E7, GS2E8, GS2E11) de notre échantillon qui arrivent à une « utilisation opérationnelle » des commandes des « messages », sont plus avancés que les autres en termes des connaissances sur la causalité et l'ordre temporel et ainsi ils arrivent à une telle performance. Nous pouvons également émettre l'hypothèse que les trois élèves (GS2E3, GS2E4, GS2E12) qui arrivent à une « utilisation partielle » ont compris au moins dans un premier temps, que les deux personnages communiquent à travers l'échange des messages. Néanmoins, ils sont peut-être moins à l'aise avec les notions de causalité et de l'ordre temporel et c'est pourquoi, même avec l'apport de l'aide de la part de la chercheuse, ils ne sont pas arrivés à positionner correctement la commande « envoyer un message ».

Les trois élèves (GS2E5, GS2E9, GS2E10) qui n'ont pas utilisé les commandes des « messages » n'étaient pas capables de saisir ni dans un premier temps ce que font ces commandes, peut-être parce qu'ils ne dominaient pas encore les notions de causalité et d' ordre temporel. L'aide de la chercheuse n'était pas suffisante pour les aider à surmonter cette difficulté. En revanche, nous pouvons supposer que GS2E10 avait compris qu'il y avait une sorte de communication entre les deux personnages et il n'a pu la représenter qu'à travers une commande de « son », même s'il l'a utilisée dans le programme du mauvais personnage.

3.5. Analyse des réponses des élèves au Post-test

En plus du problème à résoudre que nous avons donné aux élèves pendant l'évaluation, nous leur avons aussi posé un certain nombre des questions sur les commandes des « messages » lors de l'entretien individuel (Post-test). Nous avons choisi de traiter une de ces questions

parce que cette dernière portait en même temps aussi bien sur la commande « envoyer un message » que sur la commande « quand message reçu ». En effet, nous leur avons présenté, à l'aide des cartes, en même temps les programmes (voir figure 13) des deux personnages contenant les commandes de « messages » et nous leur avons demandé de nous décrire ce qu'il se passerait si on appuie sur le bouton du drapeau-vert en haut de la scène.

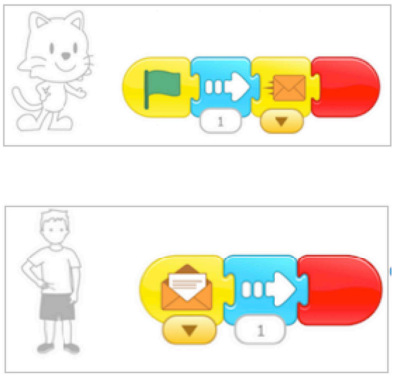


Figure 13: Les programmes présentés aux élèves pendant le Post-test

Nous avons décidé de poser cette question d'une telle manière parce que pendant notre présence sur le terrain, nous avons observé que la visualisation parallèle des programmes de deux personnages avait aidé les élèves à saisir la fonctionnalité des commandes de « messages ». Vu que les programmes présentés aux élèves contenaient aussi d'autres commandes que celles de « messages », il est intéressant d'analyser les réponses données par les élèves pour voir combien de ces derniers ont finalement choisi de décrire la fonctionnalité des commandes de « messages ».

Tout d'abord, nous pouvons observer que cinq élèves sur 12 se sont référés aux commandes de « messages » dans les réponses qu'ils ont données. Sur ces cinq élèves, deux (GS2E7, GS2E11) ont parlé des deux commandes de « messages » et trois (GS2E4, GS2E5, GS2E10) ont parlé de l'une de deux commandes. GS2E7 et GS2E11 ont donné les réponses le plus complètes en ce qui concerne les commandes de « messages ».

> GS2E7 : « Le chat va commencer, il va aller une fois vers là-bas, ensuite il va envoyer la lettre, l'enfant va le recevoir, il va aller vers là-bas et il va s'arrêter »[134].

134 Notre traduction du grec : « Θα ξεκινήσει ο γάτος θα πάει μια φορά προσ τα εκεί και μετά θα στείλει το γράμμα, το παιδί θα το λάβει, θα παέι απο εκεί και θα σταματήσει ».

> GS2E11 : « Le chat va se déplacer, il va envoyer le message à l'enfant pour commencer, ici c'est la fin. Dès que l'enfant reçoit le message, il saura quand il doit commencer et il va aller à la fin »[135].

En étudiant les réponses de ces derniers, nous pouvons constater que celle de GS2E7 est plus courte et contient moins d'informations que celle de GS2E11. GS2E7 se limite à la description de l'action de l'envoi par l' « expéditeur » et de la réception de la lettre par le « destinataire » sans expliquer la raison derrière cet envoi, mais en indiquant que suite à cette réception, le destinataire va se déplacer.

GS2E11 au contraire donne une réponse beaucoup plus détaillée. Pour la commande « envoyer un message » par exemple, l'élève se réfère à l' « expéditeur », l'action de l'envoi du message, mais aussi au « destinataire » et à la raison derrière cet envoi, qui est le démarrage de celui-ci. Sa réponse pour la commande « quand message reçu » est aussi intéressante parce que nous pouvons y repérer le « destinataire », la dimension de l'événement « Dès que l'enfant reçoit le message », celle du temps « il saura quand il doit commencer » mais aussi ce que le « destinataire » fera suite à la réception du message. Ainsi, nous pouvons constater que la réponse de GS2E11 est plus complète que celle du GS2E7.

En analysant les réponses de trois élèves (GS2E4, GS2E5, GS2E10) se référant à l'une de deux commandes de « messages », nous pouvons observer que deux choisissent de parler de la commande « envoyer un message » et un seul de la commande « quand message reçu ». GS2E4 et GS2E10 se concentrent à la commande « envoyer un message ». GS2E4 se réfère à l'expéditeur, l'action de l'envoi du message et au destinataire :

> GS2E4 : « […]. Le chat va commencer, il va faire un pas, il va envoyer le message à l'enfant et ensuite l'enfant va faire un pas à droite et après il va finir »[136].

GS2E10 se réfère seulement à l' « expéditeur » et à l'envoi de la lettre. GS2E5 se limite à une description de l'icône de la commande « quand message reçu » sans parler de la réception du message, ni de l'action associée à cette réception.

> GS2E5 : « L'enfant va ouvrir la lettre » [137]

Ainsi, nous pouvons constater que sur ces trois élèves GS2E4 donne la meilleure réponse parce qu'il se réfère à plus d'éléments que les autres. De plus, nous observons que par les

135 Notre traduction du grec : « Ο γάτος θα μετακινηθεί, θα στείλει το μήνυμα στο παιδί για να ξεκινήσει και εδώ είναι το τέλος. Μόλις το παιδί λάβει το μήνυμα, θα ξέρει πότε να ξεκινήσει και θα πάει στο τέλος ».

136 Notre traduction du grec : « Ο γάτος θα ξεκινήσει, θα κάνει ενα βήμα και θα στείλει το μήνυμα στο παιδί, και μετά θα κάνει ένα βήμα προσ τα δεξιά το παιδί και μετά θα τελειώσει ».

137 Notre traduction du grec : « Θ' ανοίξει το γράμμα το παιδάκι ».

réponses des trois élèves manquent la dimension du temps, la raison derrière l'envoi du message et l'action du « destinataire » suite à la réception du message.

Nous pouvons également noter le fait que sur les cinq élèves qui parlent des commandes de « messages » dans leur réponse, trois (GS2E5, GS2E7, GS2E10) se réfèrent à l'envoi et la réception d'une « lettre » tandis que les deux (GS2E4, GS2E11) autres se réfèrent à l'envoi ou la réception d'un message. Il est important de signaler ici que pendant la mise en place du scénario pédagogique, les deux mots ont été employés, même si nous avons essayé de privilégier le mot « message ».

En ayant pris du recul par rapport à l'expérience menée en Grèce, nous pouvons dire qu'il serait mieux d'employer uniquement le mot « message » pendant l'enseignement, pour pouvoir aider les élèves à différencier l'envoi et la réception des lettres de la vie quotidienne et l'envoi et la réception des « messages » en programmation.

En analysant les réponses de sept élèves restants, nous pouvons repérer que deux des élèves (GS2E2, GS2E12) se réfèrent au démarrage différencié dans le temps du programme de deux personnages, même s'ils ne parlent pas des commandes de « messages ». Ils répondent par exemple :

> GS2E2 :« Le chat va se déplacer et ensuite l'enfant »[138]
> GS2E12 : « Le chat va démarrer en premier et ensuite l'enfant va démarrer »[139].

Nous pouvons donc supposer qu'ils n'ont réussi à retenir que cette partie de la sémantique des « messages », c'est-à-dire le fait que lorsqu'on les ajoute dans les programmes de deux personnages les derniers démarrent leur programme l'un après l'autre et non pas en même temps. Ils n'ont pu retenir que la réaction de deux personnages dans la scène suite à l'exécution de leurs programmes.

La réponse de GS2E6 nous montre que cet élève n'a pas encore compris que la commande « quand message reçu » peut aussi constituer le début d'un programme sur ScratchJr. En effet, l'élève dit :

> GS2E6 : « Seulement le chat va démarrer puisque l'enfant n'a pas de drapeau vert » [140].

138 Notre traduction du grec : « Θα προχωρήσει ο γατούλης και μετά το παιδάκι ».
139 Notre traduction du grec : « Θα ξεκινήσει πρώτος ο γατούλης και μετά θα ξεκινήσει το παιδάκι ».
140 Notre traduction du grec : « Μόνο ο γάτος θα ξεκινήσει ».

Nous pouvons donc supposer que GS2E6 n'a pas surmonté la difficulté que certains élèves ont eue pendant la séance d'enseignement des commandes de « messages » à reconnaître la commande « quand message reçu » en tant que commande de « démarrage ». GS2E3 et GS2E9 se limitent au déplacement des deux personnages, GS2E8 se limite au démarrage du personnage chat et GS2E1 ne sait pas répondre à la question posée.

Au vu de ces résultats, nous pouvons constater que la majorité des élèves de GS2 n'arrive pas à décrire ce que font les commandes de « messages », puisque sept élèves (GS2E1, GS2E2, GS2E3, GS2E6, GS2E8, GS2E9, GS2E12) sur 12 ne parlent pas de ces derniers dans leur réponse. Trois élèves (GS2E4, GS2E5, GS2E10) sur 12 se réfèrent seulement à l'une des deux commandes en restant sur la description de son icône et sur les deux élèves (GS2E7, GS2E11) qui parlent de deux commandes seulement l'un des deux (GS2E11) cite la raison derrière l'envoi et la réception du message ainsi que la dimension de l'événement et du temps qui accompagne le passage de messages.

En faisant une comparaison entre la performance des élèves du GS2 au Post-test et leur performance à la résolution du PB1, nous observons qu'ils arrivent mieux à utiliser les commandes de « messages » pour résoudre un problème donné qu'à décrire ce que font ces commandes. Nous observons que les deux élèves (GS2E7 et GS2E11) qui donnent les meilleures réponses au Post-test font partie des élèves qui utilisent de manière opérationnelle les commandes de « messages » pour résoudre le PB1.

Les quatre autres élèves (GS2E1, GS2E2, GS2E6, GS2E8) qui arrivent à une utilisation opérationnelle des commandes de « messages » font partie des élèves qui ne parlent pas dans leurs réponses des « messages ». Nous pouvons donc supposer qu'il est possible qu'un élève puisse être capable d'utiliser les commandes de « messages » de manière opérationnelle pour résoudre un problème donné sans pouvoir décrire ce que fait la commande.

Sur les trois élèves (GS2E3, GS2E4, GS2E12) qui se trompent sur la commande « envoyer un message » pendant la résolution du PB1, deux (GS2E3, GS2E12) ne parlent pas des « messages » dans leur réponse et le dernier (GS2E4) des trois parle de la sémantique de l'une des deux commandes en décrivant seulement son icône. Nous avons donc des élèves qui ou n'arrivent pas à décrire la sémantique des commandes de « messages » ou décrivent seulement l'icône de l'une des deux commandes et ils arrivent à utiliser de manière

opérationnelle l'une des deux commandes tout en se trompant sur l'autre, pendant la résolution du problème donné.

Enfin, sur les trois élèves (GS2E5, GS2E9, GS2E10) qui n'utilisent pas les commandes de « messages » pour résoudre le PB1, nous repérons que deux élèves (GS2E10, GS2E5) ont parlé de l'une de deux commandes en décrivant son icône et que le dernier (GS2E9) n'a rien dit sur la sémantique des commandes. Nous avons donc des élèves qui n'utilisent pas les commandes de « messages » pour résoudre le problème donné et en même temps ils décrivent que l'icône de l'une des deux commandes ou qui ne parlent pas du tout des « messages » dans leur réponse.

Les résultats que nous venons de présenter montrent que les « messages » sont des commandes difficiles à aborder par les élèves de la maternelle. Leur signification est complexe et c'est pourquoi son enseignement est un vrai défi. La moitié des élèves de notre échantillon a réussi à les utiliser de manière opérationnelle pour résoudre le PB1. La commande « envoyer un message » pose plus des difficultés aux élèves que la commande « quand message reçu ». Les réponses des élèves au Post-test nous aident à comprendre qu'à la fin du scénario pédagogique la sémantique des commandes ne reste accessible qu'à une minorité des élèves de GS2.

4. Synthèse des résultats de l'étude préparatoire en Grèce

L'objectif de cette recherche a été de comprendre ce que les très jeunes élèves sont capables de faire seuls ou aidés en programmation sur ScratchJr dans des conditions réelles de classe et quelles sont les difficultés qu'ils rencontrent. Dans ce chapitre, nous avons présenté les résultats de l'étude préparatoire menée dans le GS2 en Grèce, auprès de 12 élèves de grande section de maternelle.

Tout d'abord, nous avons observé que les éléments de base de l'interface de ScratchJr et l'écriture des programmes n'ont pas posé des problèmes aux élèves du GS2. À part celle qui fait « sauter », toutes les commandes de « déplacement » ne posent pas de problème aux élèves de GS2 car ils arrivent à les utiliser de manière opérationnelle. À la fin de l'intervention, tous les élèves de GS2 utilisent « plusieurs fois les mêmes commandes » en

tant que stratégie pour déplacer leurs personnages. Face aux déplacements longs, ils se trompent sur le nombre de pas que leur personnage doit réaliser pour arriver sur l'endroit demandé de la scène, en ayant une commande de plus ou de moins dans leur programme. Nous supposons que l'origine du problème se trouve dans l'initialisation de la position des personnages. Pour les élèves qui ont les plus de difficultés avec la programmation de déplacement, nous supposons qu'ils ne dominent pas suffisamment bien des connaissances en lien avec l'orientation dans l'espace, la correspondance entre chiffres et nombres et leur usage cardinal c'est-à-dire que les nombres peuvent exprimer une quantité.

L'initialisation de la position pose des problèmes aux élèves de GS2. Ils oublient souvent d'initialiser la position de leurs personnages avant de tester leurs programmes. D'autres élèves préfèrent initialiser la position des personnages avec le doigt et non pas avec l'icône. Ainsi, ils ne mettent pas les personnages tout le temps sur le même début et c'est pourquoi ils se trompent sur le nombre de pas que le personnage a besoin de réaliser pour arriver au bon endroit dans la scène.

Les élèves ont également des difficultés avec le débogage et surtout sa première étape, évaluation du programme, où ils doivent vérifier l'exactitude de la programmation réalisée. Ils oublient souvent de penser à tester le programme qu'ils ont créé pour voir si cela fonctionne comme ils le souhaitaient ou pas. Nous avons eu besoin de leur rappeler à plusieurs reprises, surtout pendant les trois premières séances d'enseignement, la nécessité d'exécuter leurs programmes pour vérifier s'ils ont bien programmé. Nous supposons qu'à cet âge-là, ils ne sont pas encore totalement familiarisés avec la résolution des problèmes et surtout ils n'ont pas l'habitude de vérifier l'exactitude de ce qu'ils ont produit.

Le démarrage et la fin d'un programme ne leur posent pas de difficultés. Ils arrivent à utiliser de manière opérationnelle les commandes « quand drapeau-vert cliqué » et la commande de « fin » pour résoudre les problèmes donnés. Certains élèves oublient de les ajouter dans les programmes de leurs personnages. Le fait que les programmes peuvent être exécutés sur ScratchJr même lorsque les commandes « quand drapeau-vert cliqué » et « fin » n'en font pas partie empêche les élèves de comprendre la nécessité de son utilisation dans leurs programmes. Nous supposons que les élèves qui ont des difficultés avec ces deux commandes dominent moins bien que les autres les notions de la séquence et du temps.

En ce qui concerne le démarrage synchrone de deux personnages, nous avons observé que la majorité des élèves est arrivée à le réaliser pour résoudre le PB2. Les élèves qui avaient des difficultés avec la réalisation du démarrage synchrone de deux personnages avaient aussi des difficultés avec le démarrage du programme d'un personnage avec le bouton du drapeau-vert en haut de la scène.

La commande d' « enregistrement d'un son » est peut-être la commande préférée des élèves de maternelle. Parfois, ils ont des difficultés avec le menu d'enregistrement, car ils se trompent sur l'ordre d'utilisation des boutons pour enregistrer un son. La majorité des élèves de GS2 arrive à utiliser de manière opérationnelle la commande d' « enregistrement d'un son ».

La « répétition prédéfinie » est une des commandes qui pose la plupart de difficultés aux élèves de maternelle. Ces derniers se trompent autant sur la répétition prédéfinie d'une commande que sur la répétition d'une séquence de commandes. À l'exception de l'un d'entre eux, tous les élèves n'ont pas utilisé la commande de « répétition prédéfinie » de manière opérationnelle pour résoudre le PB2. La majorité d'entre eux a préféré créer trois fois le motif des commandes qui se répétait, mais certains continuaient à se tromper sur la commande d' « enregistrement du son », en enregistrant trois fois le son demandé dans la même commande d' « enregistrement ». Nous supposons qu'ils n'avaient toujours pas compris la fonctionnalité de la « répétition prédéfinie ». Une des raisons de ces difficultés pourrait être la complexité de l'écriture de la commande ou le fait qu'elle représente l'opération de la multiplication qui n'est pas encore maitrisée à cet âge-là.

Les élèves arrivent mieux à décrire ce que fait une « répétition prédéfinie d'une seule commande » qu'une « répétition prédéfinie d'une séquence des commandes ». À l'exception du même élève qu'avant, les réponses de ces derniers montrent que la signification de la commande n'est toujours pas comprise par les élèves de GS2. Nous imaginons que c'est dur pour les élèves de maternelle de voir la « répétition d'une séquence des commandes » et de comprendre que celle-ci représente trois motifs des commandes. Cela demande une réflexion d'un niveau d'abstraction assez élevé qui n'est généralement pas facile à cet âge-là.

Les « messages » sont aussi des commandes qui posent des difficultés aux élèves de GS2. Pendant l'enseignement, nous avons observé que quelques élèves étaient déjà un peu familiarisés avec la réception et l'envoi de courriers dans leur vie quotidienne. Certains élèves

avaient des difficultés à saisir le rôle des commandes des « messages » quand ces derniers ont été présentés à l'aide des projets qui leur rappelaient une expérience physique de leur vie quotidienne. Nous avons donc repéré qu'il existe un écart entre l'expérience physique et la programmation sur ScratchJr et cela peut constituer un obstacle pour les élèves.

La moitié des élèves de GS2 a réussi à utiliser de manière opérationnelle les commandes de « messages » pour résoudre le PB1. L'utilisation de la commande « envoyer un message » pose plus de problèmes aux élèves que celle de la commande « quand message reçu ». Les difficultés des élèves portent notamment sur le positionnement de la commande « envoyer un message » dans le programme de leur personnage. En effet, le positionnement de cette commande paraît difficile aux élèves de maternelle, car il implique une série des connaissances sur la causalité et l'ordre temporel qui ne sont pas encore acquis à cet-âge là.

La majorité des élèves de GS2 arrive à utiliser de manière opérationnelle la commande « quand message reçu » pour résoudre le PB1. Nous supposons que les élèves de GS2 ont moins de problèmes avec cette commande, d'une part parce que la forme d'un bloc de démarrage oblige l'utilisateur à la poser au début d'un programme et d'autre part, parce qu'il y a un lien plus visible entre l'exécution de celle-ci et la réaction du personnage dans la scène. La majorité des élèves de GS2 n'arrive pas à décrire ce que font les commandes de « messages » au Post-test. Nous avons également observé qu'ils arrivent mieux à utiliser les commandes de « messages » pour résoudre un problème qu'à décrire ce qu'elles font.

Ce chapitre nous a permis de présenter ce que les élèves de grande section de maternelle ont pu faire, seuls ou aidés, en programmation sur ScratchJr, dans des conditions réelles de classe pendant la phase de la pré-étude en Grèce. Dans le chapitre 2 qui suit, nous allons étudier ce que font les élèves de cours préparatoire en programmation sur ScratchJr.

Chapitre 2. Étude principale en France

Suite à l'étude préparatoire en Grèce, nous avons pu obtenir une vision globale sur ce que les élèves de grande section de maternelle peuvent faire seuls ou aidés en programmation sur ScratchJr et les difficultés qu'ils risquent de rencontrer. Nous avons également pu améliorer notre scénario pédagogique et l'organisation de la méthodologie de mise en place de notre recherche pour le terrain en France. Pour des raisons que nous avons déjà expliquées plus haut liées aux contraintes du terrain en Grèce, nous avons dû adapter notre problématique à ces dernières et ainsi nous n'avons pu étudier seulement une partie de notre problématique initiale.

Pendant l'étude principale en France nous avons eu la possibilité d'étudier notre problématique initiale dans son intégralité. En effet, nous avons pu collecter des données qui nous permettaient d'étudier à la fois ce que les élèves de cours préparatoires sont capables de faire seuls ou aidés en programmation sur ScratchJr, leurs difficultés, mais aussi leur processus de résolution de problèmes pour identifier les stratégies de programmation utilisées et le débogage mis en place.

Dans ce chapitre, nous présentons les résultats provenant de l'étude principale réalisée en France, auprès de deux classes de cours préparatoires (CP1, CP2) déjà équipées en tablettes de deux écoles primaires publiques de la ville de Puteaux.

Dans un premier temps, nous avons analysé l'activité de programmation de quatre élèves du CP1 et nous avons choisi de présenter les résultats de cette analyse à l'aide des portraits. Nous décrirons d'abord les choix que nous avons faits concernant le codage de l'activité de programmation de quatre élèves de CP1 et ensuite, nous présenterons les portraits. Après les portraits, nous mettrons en avant une synthèse comparative de quatre élèves de CP1 pour ressortir les éléments principaux qui émergent de ces derniers. Compte tenu de ce qu'on a vu sur les quatre élèves que nous avons étudiés de manière très fine, nous présenterons à la fin de ce chapitre une série des caractéristiques générales de bonne et de moins bonne performance.

1. Choix terminologiques pour le codage des données

Dans cette section nous présenterons les choix terminologiques que nous avons faits pour coder l'activité de programmation de quatre élèves de CP1. Pour analyser nos données, nous avons décidé de combiner les deux approches de codage, en réalisant du codage à la fois déductif et inductif (Miles et al., 2014). Ainsi, nous avons utilisé à la fois des termes venant de la littérature mais aussi nos propres typologies quand celles de la littérature ne nous convenaient pas. Dans ce qui suit, nous décrirons ces choix en expliquant pourquoi nous avons privilégié ce terme-là plutôt qu'un autre et pourquoi nous avons eu parfois besoin de créer nos propres termes.

1.1. Les stratégies de programmation

En ce qui concerne les stratégies de programmation utilisées, nous avons dans un premier temps repéré deux types de comportements différents en analysant l'activité de programmation des élèves : certains programmaient plusieurs actions ensemble sans exécuter leur programme et certains programmaient une action et l'exécutaient, puis, ils programmaient une autre action et l'exécutaient etc. Ensuite, à force d'observer les actions des élèves appartenant à la première catégorie, nous avons remarqué qu'il y avait ceux qui programmaient l'intégralité du programme d'un personnage avant l'exécution et ceux qui programmaient l'intégralité du programme des deux personnages avant l'exécution. C'est pourquoi, nous avons décidé d'inclure deux types de planification dans notre codage : une « planification partielle », pour les élèves qui programmaient l'intégralité du programme d'un personnage avant l'exécution et une « planification complète », pour les élèves qui programmaient l'intégralité du programme de deux personnages avant l'exécution.

Pour le deuxième comportement repéré, c'est-à-dire les élèves qui programmaient une action et l'exécutaient, puis, programmaient une autre action et l'exécutaient etc, nous avons au début hésité en ce qui concerne sa caractérisation. En consultant les travaux de Fessakis et al. (2013), Komis et Misirli (2013) et (Vigot, 2018), nous avons remarqué que ce comportement a été souvent défini en tant que stratégie d' « essai-erreur » ou de « pas à pas ». En étudiant les comportements que ces travaux de recherche associent à ces stratégies, nous jugeons qu'il n'y

a pas assez d'éléments pour les différencier. C'est pourquoi, nous avons décidé qu'il n'est pas pertinent d'avoir les deux termes dans notre codage. En effet, nous avons privilégié l'utilisation du terme « pas à pas » plutôt que celui de l' « essai-erreur », car nous le trouvons plus adapté à nos données et en particulier à la façon dont les élèves procèdent lorsqu'ils programment. De plus, nous pensons que la dénomination de stratégie « essai-erreur » donne l'impression que l'action de l'élève serait de facto une erreur, tandis que cela n'est pas toujours vrai.

Plus tard dans notre analyse, nous avons identifié un troisième type de comportement. Les élèves ayant ce comportement programmaient une partie du programme de leur personnage avant l'exécution. Il s'agit d'après nous d'une évolution de la stratégie « pas à pas ». Nous avons choisi d'appeler cette stratégie « sous-programme », en s'inspirant du travail de Komis et Misirli (2013). De plus, nous avons détecté un autre comportement celui d'un élève qui effaçait tout le programme de son personnage et il recommençait sa programmation. Nous avons choisi de nommer cette stratégie « recommencer au début ». C'est une stratégie que nous avons repérée une seule fois dans notre corpus. Nous avons aussi identifié des hybridations de ces stratégies, les suivantes : « planification partielle et pas à pas » et « sous-programme et pas à pas ». Ce sont des combinaisons des stratégies déjà repérées.

1.2. Les stratégies de déplacement

Concernant les stratégies de déplacement employées par les élèves nous en avons repéré trois dans notre corpus. Cette catégorie de codage porte sur les différentes manières qu'ont eu les élèves de programmer un déplacement. En consultant le modèle théorique de Vigot (2018), nous avons constaté que les stratégies de déplacement proposées par l'auteure étaient davantage orientées vers les connaissances mathématiques mobilisées par les élèves de 7 à 8 ans. Pour les typologies que nous avons utilisées, nous avons décidé de ne pas mettre en avant les connaissances mathématiques mobilisées. C'est pourquoi nous n'avons pas utilisé ce modèle.

De plus, puisque dans notre scénario pédagogique deux types de déplacements différents ont été enseignés aux élèves (déplacement en utilisant les paramètres et déplacement en utilisant plusieurs fois les mêmes commandes), il était nécessaire de prendre en compte ces types de déplacements dans nos typologies.

La première stratégie est celle des élèves qui choisissaient un paramètre pour définir le nombre de pas dont leur personnage doit effectuer pour arriver au bon endroit dans la scène. Nous l'avons appelé « paramètres ». La deuxième stratégie repérée est celle des élèves qui utilisaient plusieurs fois le même type de commande pour déplacer leur personnage au lieu d'utiliser les paramètres. Nous avons nommé cette dernière « plusieurs fois la même commande ».

La dernière stratégie de déplacement que nous avons identifié a été inventée par une élève du CP1. Il s'agit de la stratégie « grille et paramètres ». L'élève utilisant cette stratégie suivait une série d'étapes pour déplacer son personnage dans la scène. Tout d'abord, elle faisait apparaître la « grille » et ensuite, elle déposait la commande de « déplacement » dans l'espace de programmation. Puis, elle appuyait sur cette commande plusieurs fois et, en regardant la position du personnage concerné sur la « grille », elle comptait le nombre de pas que le personnage devait faire pour arriver sur le bon endroit dans la scène. Enfin, elle ajoutait comme paramètre de la commande de « déplacement », le numéro correspondant au nombre de pas qu'elle a compté auparavant, pour que son personnage arrive au bon endroit.

Pour les stratégies de déplacement, tout comme les stratégies de programmation, nous avons aussi identifié des hybridations. Les hybridations sont les suivantes :« plusieurs fois la même commande et paramètres », « plusieurs fois la même commande et puis, réduction de l'écriture du codage en utilisant les paramètres ».

La dernière hybridation nous rappelle un peu la stratégie « grille et paramètres » car elle suit une logique similaire, mais l'élève n'utilise pas la « grille ». L'élève qui employait cette stratégie déposait dans un premier temps la commande de « déplacement » dans l'espace de programmation. Ensuite, il appuyait sur cette commande plusieurs fois et en regardant la position du personnage concerné sur la scène, il comptait le nombre de pas que le personnage doit faire pour arriver au bon endroit. Enfin, il ajoutait comme paramètre de la commande de « déplacement » le numéro correspondant au nombre de pas qu'il a compté auparavant, pour que le personnage arrive au bon endroit.

1.3. Le débogage

En ce qui concerne le débogage réalisé par les élèves de notre corpus, nous en avons identifié trois types : « optimal », « efficace » et « non efficace ». Nous avons caractérisé « optimal » le

débogage d'un élève qui arrive à corriger l'erreur présente dans son programme le plus vite possible en termes de temps ou de nombre d'essais. Nous avons nommé « efficace », le débogage d'un élève qui arrive à corriger l'erreur présente dans son programme et « non efficace », le débogage de l'élève qui n'arrive pas à la corriger.

Quant aux patterns de débogage, nous avons eu besoin de créer nos propres typologies parce que la littérature ne faisait pas référence à cela. Nous avons défini en tant que pattern de débogage non pas une succession d'actions spécifiques (comme « exécution, exécution, correction, correction, interface, initialisation »), mais une succession de changements d'actions (comme « exécution, correction du code, initialisation »). Nous avons décidé de considérer une succession de changements d'actions en tant que pattern de débogage quand ce dernier se répète au moins deux fois sur les actigrammes d'un élève ou deux élèves différents. De cette façon nous avons identifié trois patterns différents : « pattern de débogage où l'initialisation est après la correction ou les corrections », « pattern de débogage où l'initialisation est avant la correction ou les corrections » et « pattern de débogage sans initialisation avec une seule correction ou plusieurs ».

Cette section a eu pour objectif d'aider le lecteur à comprendre les choix terminologiques que nous avons faits pour décrire nos données, afin qu'il puisse au mieux suivre la section suivante qui porte sur les portraits de quatre élèves de CP1.

2. Portraits de quatre élèves de CP1

Pour rappel, notre objectif en réalisant l'étude principale en France a été de comprendre ce que les élèves de cours préparatoire sont capables de faire, seuls ou aidés, en programmation sur ScratchJr, quelles sont les stratégies de programmation qu'ils mettent en œuvre, quelles sont leurs difficultés et leur capacité de débogage. Nous souhaitons répéter ici que, comme nous l'avons fait pour l'étude préparatoire en Grèce, pour l'étude principale en France nous sommes à plusieurs reprises intervenue pendant la séance d'évaluation pour aider[141] les élèves à avancer.

Comme nous avons déjà expliqué auparavant, nous avons sélectionné quatre élèves de CP1 avec des comportements contrastés, afin de réaliser une analyse fine aussi bien de leur activité

141 Voir Tableau 9 dans la section 4.5, du chapitre 3 de la Partie 3 pour un récapitulatif des types d'aides que nous avons apportés.

de programmation que des programmes qu'ils ont élaborés pour résoudre deux problèmes de programmation pendant la séance d'évaluation. Grace à cette analyse, nous avons pu identifier une série d'éléments qui nous ont permis de répondre à notre problématique. Nous avons décidé de présenter les résultats de cette analyse à l'aide des portraits. Ces derniers regroupent les éléments qui nous permettent de bien comprendre ce que font les quatre élèves choisis en programmation sur ScratchJr.

Chaque portrait a pour ambition de restituer les éléments suivants : la résolution des problèmes par l'élève, la conception des programmes, la gestion de l'interface, la gestion du déplacement des personnages, les messages, la répétition prédéfinie, la gestion du démarrage synchrone, les erreurs et le débogage mis en place et les aides apportées.

L'objectif des portraits est d'obtenir une vue sur la globalité de l'activité de programmation de quatre élèves et de faire émerger les particularités de celle-ci. Pour mieux décrire ce que font les quatre élèves, nous avons utilisé des qualificatifs mettant en avant les caractéristiques de chacun. Dans les sections qui suivent, nous présentons les portraits de quatre élèves du CP1.

2.1. Élève 1 : rapide, précise et efficace

2.1.1. La résolution des problèmes (PB1 et PB2)

L'élève 1 arrive, avec une seule aide, à résoudre en temps limité le PB1. En analysant ses programmes finaux pour le PB1 (voir Tableau 11), nous observons qu'elle ne poursuit pas la suite d'actions de déplacement proposée dans le projet modèle. En effet, elle programme le personnage « papillon » à se déplacer d'abord en bas et puis à droite, tandis que dans le projet modèle le personnage se déplace d'abord à droite et puis en bas.

L'élève 1 arrive avec huit aides à résoudre le PB2. En analysant ses programmes finaux pour le PB2 (voir Tableau 12), nous observons qu'elle ne poursuit de nouveau pas la suite d'actions de déplacement proposée dans le projet modèle. En effet, elle programme son personnage « papillon » de se déplacer d'abord à droite, puis en bas et enfin à droite, alors que dans le projet modèle ce personnage se déplace d'abord à droite et puis en bas.

Nous n'accordons pas beaucoup d'importance au fait que l'élève 1 ne poursuit pas la suite d'actions de déplacement proposée dans le projet modèle, car nous avons choisi de laisser une

marge aux élèves pour expérimenter sur cela, à condition que leur projet final comprenne les commandes nécessaires et que leurs personnages arrivent aux endroits demandés dans la scène. Les programmes finaux de l'élève 1 pour le PB1 et le PB2 sont corrects.

Sa performance au PB1 est assez similaire à celle observée sur le PB2. La seule différence est le fait que pendant la réalisation de son débogage au PB1, elle réussit à mesurer de façon plus efficace la distance dont ses personnages doivent réaliser pour arriver au bon endroit dans la scène, alors qu'au PB2, elle a besoin de faire plus d'estimations à cause du fait qu'elle a introduit dans le programme du « papillon » deux mêmes commandes de déplacement à droite.

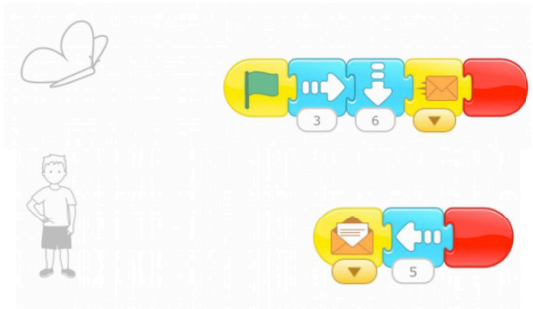
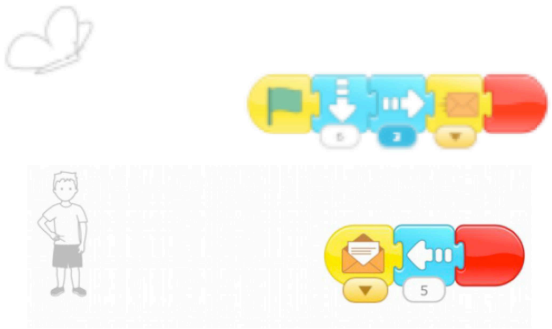
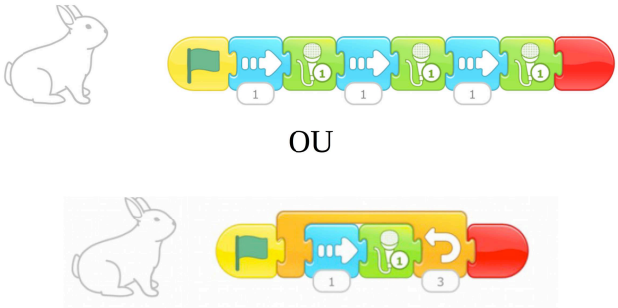
PB1	
Projet modèle	
Élève 1	

Tableau 11: Programmes finaux de l'Élève 1 PB1

PB2	
Projet Modèle	 OU

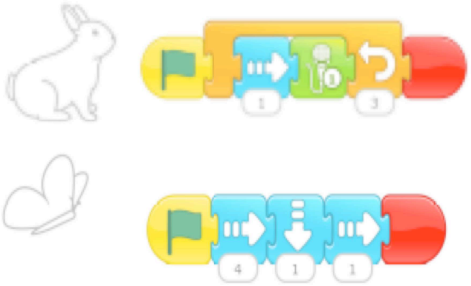
	
Élève 1	

Tableau 12: Programmes finaux de l'Élève 1 PB2

Comme nous pouvons voir dans le Tableau 13, elle a eu besoin à peu près du même nombre d'actions pour résoudre les deux problèmes, tandis que le PB2 demandait souvent beaucoup plus d'efforts aux élèves. En effet, elle a eu besoin de 43 actions pour résoudre le PB1 et 46 actions pour résoudre le PB2. De plus, elle résout le PB1 en 3min22s et le PB2 en 4min35s. Nous repérons que la plupart de ses actions concernent l'interface et ensuite, les commandes (voir Tableau 13).

Élève 1		
	PB1	PB2
Interface	20	17
Commandes	15	14
Corrections	2	3
Initialisation	3	6
Exécution	3	6
Question posée par l'élève	0	0
Total d'actions par élève	43	46
Temps de résolution	3min22s	4min35s
Aides	1	8
Problème réussi	Oui	Oui

Tableau 13: Récapitulatif d'actions de l'Élève 1 PB1 & PB2

2.1.2. La conception des programmes

Dès le début de la résolution des problèmes, l'élève 1 parait sûre d'elle quant à ses actions. Sa programmation est caractérisée par la précision et la rapidité, car elle choisit directement les commandes dont elle a besoin sans hésitation.

Elle commence par définir les éléments d'interface, puis elle passe à l'écriture de ses programmes et enfin, elle les exécute pour vérifier sa programmation. Nous observons également qu'elle termine l'écriture du programme de l'un personnage pour avancer à l'écriture du programme de l'autre. Elle ne réalise pas des allers-retours entre les deux personnages pendant l'écriture de ses programmes.

Pour la conception de ses programmes, elle met en place une stratégie de « planification complète » pour les deux problèmes, car elle écrit l'intégralité des commandes des programmes de ses deux personnages avant la première exécution (voir Tableau 14). C'est pourquoi nous observons dans ses actigrammes que la partie de l'alternance entre l'interface et les commandes est plus longue que la partie dédiée au débogage des erreurs présentes sur ses programmes.

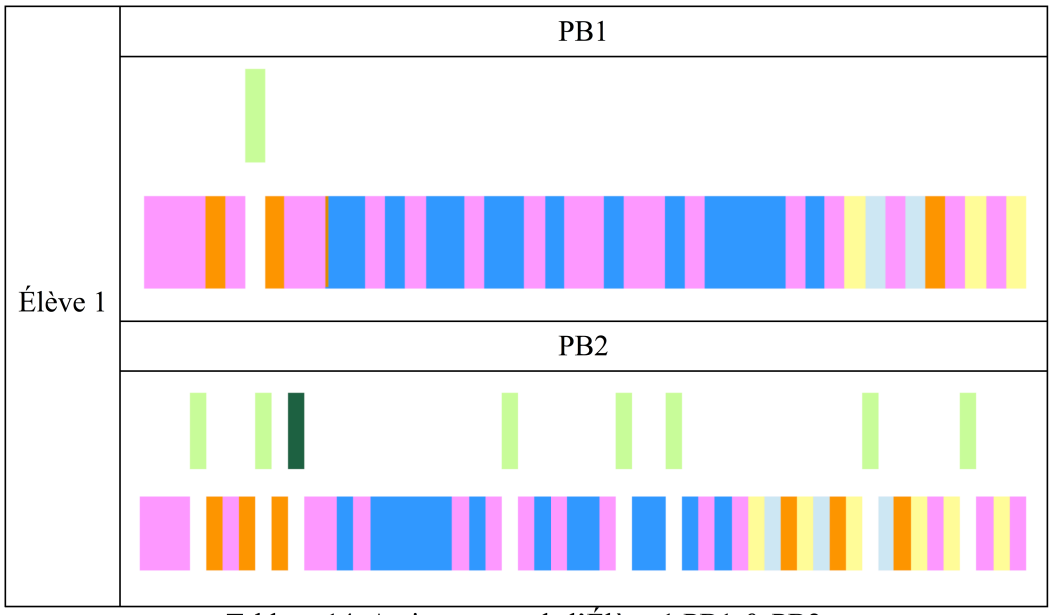


Tableau 14: Actigrammes de l'Élève 1 PB1 & PB2
Rappel de catégories d'actions : interface, commandes, corrections, initialisation, exécution
Aides : Consigne , Lancement projet, Initialisation

2.1.3. La gestion de l'interface

En ce qui concerne les éléments d'interface, nous observons que l'élève 1 arrive très bien à les gérer au PB1. Elle insère les personnages et l'arrière-plan sans problème. L'initialisation de la position des personnages, qui est très souvent un obstacle pour les élèves, lui en pose peu. Au début du PB1, nous lui avons rappelé une fois la consigne du problème pour qu'elle puisse positionner correctement ses personnages avant de commencer sa programmation.

De la même manière, nous lui avons de nouveau rappelé la consigne du problème au début du PB2, pour l'aider avec le premier positionnement de ses personnages dans la scène. L'élève 1 s'est trompée sur le positionnement de l'un des deux personnages et ensuite nous sommes intervenue deux fois pour l'aider. La première pour lui rappeler encore une fois la consigne du problème et la deuxième en l'aidant à initialiser son personnage. À part ce problème de positionnement au début de la résolution du PB2, elle n'a pas eu d'autres difficultés avec l'initialisation de la position de ses personnages. En effet, par la suite elle initialise correctement la position de ses personnages en utilisant toujours l'icône d'initialisation en haut de la scène et elle n'oublie pas de le faire avant chaque exécution. C'est pourquoi le nombre d'actions pour l'initialisation est égal au nombre d'actions pour l'exécution pour les deux problèmes.

2.1.4. La gestion du déplacement des personnages

En ce qui concerne le déplacement de ses personnages, l'élève 1 utilise les paramètres en essayant d'estimer à chaque fois celui qui est nécessaire pour que son personnage arrive au bon endroit dans la scène. Parfois, elle commence en utilisant plusieurs fois la même commande et ensuite, elle choisit de réduire son écriture en utilisant les paramètres. Pour le PB1, elle utilise que les « paramètres » en tant que stratégie de déplacement. Au cours du PB2, elle choisit d'utiliser à la fois « les paramètres et plusieurs fois le même type de commande ».

2.1.5. Les messages

L'élève 1 utilise les commandes de « messages »[142] de manière opérationnelle pour résoudre le PB1 (voir Tableau 11). Elle utilise les deux commandes « envoyer un message » et « quand message reçu » dans l'ordre correct sur le programme de deux personnages.

2.1.6. La répétition prédéfinie

En ce qui concerne la commande de « répétition prédéfinie », l'élève 1 est une des seuls à l'utiliser pour réaliser le PB2 (voir Tableaux 12). Elle n'a pas de difficultés lors de la création du motif des commandes à répéter. Toutefois, nous sommes intervenue trois fois pendant

142 Pour rappel, en raison du bas âge des élèves, nous avons décidé de ne pas enseigner l'envoie et la réception des messages des couleurs différentes.

l'écriture de la « répétition prédéfinie d'une séquence des commandes » en lui rappelant la consigne du problème.

2.1.7. La gestion du démarrage synchrone de deux personnages

L'élève 1 arrive aussi à réaliser le démarrage synchrone de programmes de deux personnages pour résoudre le PB2, en utilisant de manière opérationnelle la commande « quand drapeau vert cliqué » au début du programme de deux personnages et le bouton du drapeau-vert en haut de la scène (voir Tableau 12).

2.1.8. Les erreurs réalisées et le débogage mis en place

L'élève 1 arrive à identifier des erreurs sur son programme avant même de l'exécuter. Plus précisément, pendant la résolution du PB1 après avoir terminé le programme du personnage « papillon », elle passe directement au personnage « enfant » pour ensuite, revenir au programme du « papillon » parce qu'elle avait repéré avoir oublié la commande « envoyer un message ». Elle l'ajoute donc à son programme.

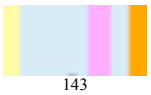
	Pattern de débogage	Suite d'actions	Morceau d'actigramme
Élève 1	Pattern de débogage où l'initialisation est après la correction ou les corrections	Exécution→ Correction → Initialisation Exécution→Correction →Correction→Correction →Interface→Correction →Initialisation	 

Tableau 15: Types de patterns de débogage identifiés sur les actigrammes de l'Élève 1

Les erreurs existantes après la première exécution concernent le déplacement des personnages et plus précisément l'évaluation de la distance réalisée par ces personnages aussi bien au PB1 qu'au PB2. En analysant sa programmation, nous avons pu identifier trois types de débogage différents (efficace, non efficace, optimal).

143 Ce morceau d'actigramme est utilisé comme un exemple d'un débogage avec plusieurs actions de correction et l'initialisation après les corrections. Ce n'est pas un pattern qui est retrouvé plusieurs fois dans les actigrammes des élèves. Ce pattern pourrait avoir des formes différentes. Ces formes contiennent toujours des actions d'exécution, de correction et d'initialisation et ce qui change chaque fois c'est l'existence des actions d'interface, des questions posées par l'élève ou des aides apportées.

Pour le PB1, elle avait deux erreurs de déplacement à corriger, une sur chaque personnage et elle a donc mis en place un débogage « optimal », car elle a corrigé les deux en un seul essai. En analysant son actigramme pour le PB1 (voir Tableau 14), nous avons pu repérer un pattern de débogage où l'initialisation est après les actions de correction (voir Tableau 15).

Pour le PB2, elle a corrigé le déplacement du personnage « papillon », car ce dernier se déplaçait trop à droite. Pour corriger son programme, elle a eu besoin de réaliser trois débogages consécutifs, deux « non efficaces » et un « efficace ». Nous sommes intervenue deux fois pendant les débogages pour lancer le projet modèle et aider l'élève à vérifier si son personnage arrive au bon endroit dans la scène après l'exécution de son programme ou si elle doit continuer à déboguer ses programmes. En analysant son actigramme pour le PB2 (voir Tableau 14), nous avons identifié trois fois le même pattern, où l'initialisation est après l'action de correction (voir Tableau 15).

Nous observons ainsi que tous les patterns de débogage identifiés sur les actigrammes de l'élève 1 suivent la même suite d'actions : *Exécution-Correction-Initialisation* (voir Tableau 14).

De plus, les corrections apportées démontrent également une réflexion préalable sur le résultat obtenu après l'exécution de ses programmes. L'élève 1 réalise peu de corrections aussi bien au PB1 qu'au PB2. Elle n'a pas eu des moments de blocage[144] pendant la résolution des deux problèmes et elle ne nous a pas posé de question.

2.1.9. Les aides apportées

Pour récapituler les aides qu'elle a reçues, nous avons noté une aide au PB1 et huit aides au PB2. Au PB1, elle a reçu au début un rappel de la consigne pour bien positionner ses personnages avant de commencer sa programmation. Au PB2, elle a reçu deux rappels de consigne et une aide sur l'initialisation pour bien positionner ses personnages avant de commencer sa programmation. Ensuite, elle a reçu trois rappels de la consigne pour l'aider à l'écriture de la répétition et deux aides de type lancement du projet modèle pour l'aider à vérifier si son personnage arrive au bon endroit dans la scène ou pas.

144 Pour rappel, nous définissons en tant que moment de blocage une suite d'actions pendant laquelle l'élève présente une certaine incertitude, il ne réalise pas d'actions ou il réalise d'actions qui ne changent pas l'état de son projet. Voir sous section *Un focus sur la qualité des actions des élèves* de la section 4., Chapitre 3 dans la Partie 3. sur la Méthodologie, pour plus d'informations.

2.2. Élève 2 : inventive et précautionneuse

2.2.1. Résolution des problèmes (PB1 et PB2)

L'élève 2 arrive, avec deux aides, à résoudre en temps limité le PB1. Elle arrive aussi avec dix aides à résoudre le PB2. En analysant ses programmes finaux (voir Tableau 16 et17), nous repérons que ces derniers ne différent pas du projet modèle. En effet, l'élève 2 a respecté la contrainte concernant le point d'arrivée des personnages dans la scène. Elle a aussi choisi de suivre l'ordre des commandes de « déplacement » présenté dans le projet modèle alors que cela n'était pas nécessaire. Les programmes finaux de l'élève 2 pour le PB1 et le PB2 sont corrects.

En faisant une comparaison de sa performance aux deux problèmes d'évaluation, nous observons qu'elle est meilleure au PB1 qu'au PB2. Au PB1, l'élève n'a même pas besoin de procéder au débogage, car elle ne réalise aucune erreur. Au PB2, elle a besoin de revenir sur sa programmation pour l'améliorer.

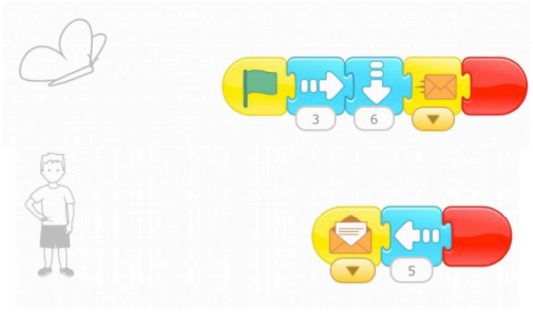
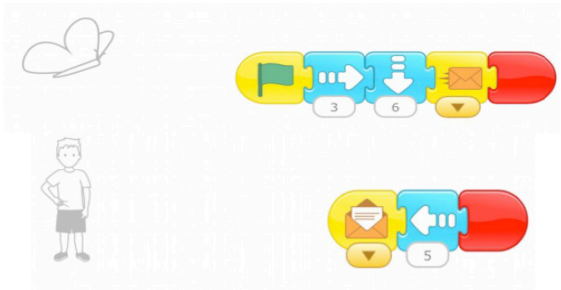
PB1	
Projet modèle	
Élève 2	

Tableau 16: Programmes finaux de l'Élève 2 PB1

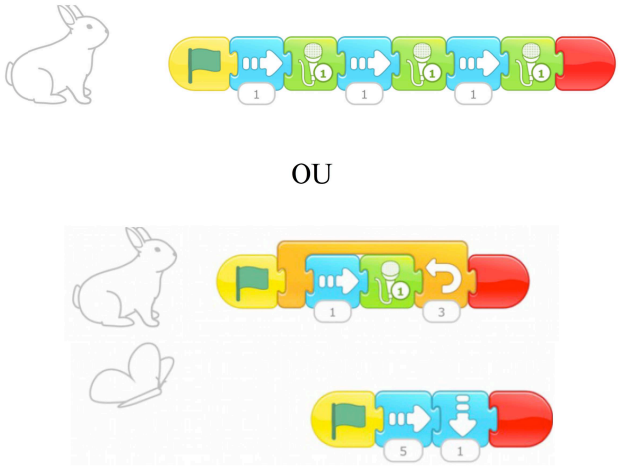
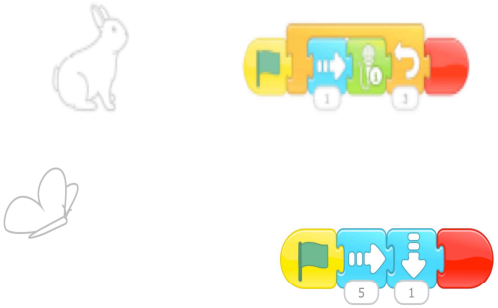
PB2	
Projet Modèle	
Élève 2	

Tableau 17: Programmes finaux de l'Élève 2 PB2

Comme nous pouvons le voir dans le Tableau 18, l'élève 2 a eu besoin de plus d'actions pour résoudre le PB2 que pour le PB1. Le PB2 demandait souvent plus d'efforts aux élèves. En effet, elle a eu besoin de 51 actions pour résoudre le PB1 et 62 actions pour résoudre le PB2. Elle a résolu le PB1 en 3min21s et le PB2 en 5min02s. Nous repérons que la plupart de ses actions concernent l'interface et ensuite, les commandes (voir Tableau 18).

Élève 2		
	PB1	PB2
Interface	15	14
Commandes	29	27
Corrections	0	4
Initialisation	5	9
Exécution	2	7
Question posée par l'élève	0	1
Total d'actions par élève	51	62
Temps de résolution	3min21s	5min02s
Aides	2	10
Problème réussi	Oui	Oui

Tableau 18: Récapitulatif d'actions de l'Élève 2 PB1 & PB2

2.2.2. La conception des programmes

En étudiant son activité de programmation pendant la résolution des problèmes, nous observons que l'élève 2 sait où elle va lorsqu'elle écrit ses programmes et elle préfère ne pas prendre des risques surtout en ce qui concerne le déplacement de ses personnages. Pour cela elle est prête à inventer une stratégie de déplacement qui pourrait lui garantir que le paramètre qu'elle va utiliser sera le bon.

Elle commence sa programmation par définir les éléments d'interface, puis elle passe à l'écriture de ses programmes et enfin, elle les exécute pour vérifier sa programmation. Nous observons également qu'elle termine l'écriture du programme de l'un personnage pour avancer à l'écriture du programme de l'autre. Elle ne réalise pas des allers-retours entre les deux personnages pendant l'écriture de ses programmes.

En ce qui concerne la conception de ses programmes, nous repérons des petites différences entre le PB1 et le PB2. Pour le personnage « papillon » du PB1, elle met en place une stratégie de « sous-programme et pas à pas », car elle écrit les commandes « quand personnage touché », « envoyer un message », « déplacement à droite » et elle s'arrête pour exécuter que la commande de « déplacement ». Ensuite, elle programme le reste du programme du personnage « papillon », c'est-à-dire le « déplacement vers le bas » et la « fin ». Pour écrire le programme du personnage « enfant » du PB1, elle met en place une stratégie de « planification partielle », car elle écrit l'intégralité des commandes de son

programme avant la première exécution. L'élève 2 n'utilise que deux fois l'exécution pendant la résolution du PB1 (voir Tableau 18).

Pour écrire le programme du personnage « lapin » au PB2, elle combine les stratégies « pas à pas » et « planification partielle », car elle exécute séparément la commande de « son » pour vérifier qu'elle a bien enregistré la phrase à répéter et ensuite, elle écrit l'intégralité des commandes du programme de ce personnage avant l'exécution. Toujours au PB2, pour écrire le programme du personnage « papillon », elle met en place une stratégie de « planification partielle et sous-programme », parce qu'elle écrit toutes les commandes du programme et elle exécute en « sous-programme » seulement les commandes de « déplacement ». Ensuite, elle teste le programme en entier. C'est pourquoi, nous observons dans ses actigrammes que la partie de l'alternance entre l'interface et les commandes est plus longue que la partie dédiée au débogage des erreurs présentes sur ses programmes.

Nous repérons également une similarité entre la conception de ses programmes au PB1 et au PB2. L'élève 2 choisit d'exécuter les parties de ces dernières qui lui paraissent susceptibles de contenir des erreurs. Cela arrive plus souvent au PB2 qu'au PB1. Pendant la résolution du PB1, elle exécute la commande de « déplacement » séparément du reste de son programme, pour vérifier si elle avait bien évalué la distance que son personnage devrait réaliser dans la scène. Pendant la résolution du PB2, elle exécute la commande du « son » qu'elle venait d'enregistrer en disant « je vais essayer d'abord », pour vérifier qu'elle l'a bien fait et elle exécute aussi les deux commandes de « déplacement », pour vérifier si elle a bien évalué la distance.

2.2.3. La gestion de l'interface

Les éléments d'interface ne lui posent aucun problème à part l'initialisation. Au début du PB1, nous l'avons aidé à initialiser correctement la position de son personnage « enfant » avant de commencer sa programmation. Suite à cela, l'élève 2 n'a plus eu d'autres difficultés sur l'initialisation. Au cours du PB2, nous l'avons de nouveau aidé à initialiser correctement la position de ses personnages avant de passer à leur programmation. Un peu plus tard dans la vidéo, elle se trompe deux fois sur l'initialisation de la position du personnage « papillon » et nous intervenons deux fois pour l'aider à initialiser sa position. Elle oublie également une fois d'initialiser la position de son personnage « papillon », mais elle l'aperçoit rapidement, elle

dit « j'ai pas initialisé » et elle retourne initialiser la position de son personnage. De plus, en visualisant son enregistrement vidéo nous avons observé qu'elle réalise parfois l'initialisation de la position de ses personnages avec l'icône et parfois avec son doigt.

2.2.4. La gestion du déplacement des personnages

Pour le déplacement de ses personnages, l'élève 2 utilise une stratégie qu'elle a inventée toute seule en utilisant la « grille[145] et les paramètres », pour tous les déplacements effectués au cours de la résolution des deux problèmes. Cette stratégie de déplacement lui permet de valider le déplacement sans exécuter le programme entier. La « grille » est un bouton de l'interface de ScratchJr qui fait apparaître une grille numérotée sur la scène avec les axes x et y.

> L'élève active d'abord la grille. Elle appuie ensuite plusieurs fois sur la même commande de déplacement en regardant la grille, pour voir quand le personnage arrive au bon endroit dans la scène. Après, elle utilise comme paramètre de la commande de déplacement le numéro correspondant au nombre de fois qu'elle a eu besoin d'appuyer sur celle-ci, pour que son personnage arrive au bon endroit dans la scène.

La mise en place de cette stratégie permet à l'élève 2 une vitesse et une meilleure efficacité à la résolution du problème, malgré le fait qu'elle l'oblige à réaliser plus d'actions, car elle vérifie l'exactitude de sa programmation pour le déplacement du personnage avant l'exécution de son programme. Cela peut être vu aussi sur le Tableau 18.

2.2.5. Les messages

L'élève 2 utilise les commandes de « messages » de manière opérationnelle pour le PB1 (voir Tableau 16). En effet, elle utilise les deux commandes, « envoyer un message » et « quand message reçu », dans l'ordre correct sur le programme de deux personnages.

2.2.6. La répétition prédéfinie

En ce qui concerne la répétition prédéfinie, l'élève 2 est une de seules à utiliser de manière opérationnelle la commande de « répétition » pour réaliser le PB2 (voir Tableaux 17). Toutefois, elle se trompe pendant l'écriture de son programme sur l'ordre des commandes à répéter. C'est pourquoi, nous sommes intervenue trois fois en lui rappelant la consigne du problème. La quatrième intervention sous forme de rappel de la consigne a été réalisée suite à

145 Il s'agit d'un bouton de l'interface de ScratchJr qui fait apparaître une grille numérotée sur la scène avec les axes x et y.

la demande de l'élève. Nous préciserons plus tard dans ce portrait les conditions sous lesquelles cet élève a réclamé notre intervention.

2.2.7. La gestion du démarrage synchrone de deux personnages

Elle arrive aussi à réaliser le démarrage synchrone de deux personnages pour résoudre le PB2, en utilisant de manière opérationnelle la commande « quand drapeau-vert cliqué » au début du programme de deux personnages et le bouton du drapeau-vert en haut de la scène (voir Tableau 17).

2.2.8. Les erreurs réalisées et le débogage mis en place

L'élève 2 arrive à identifier des erreurs sur son programme avant de l'exécuter et elle le fait deux fois au cours de la vidéo. Pendant la résolution du PB1, elle repère qu'elle a utilisé la commande « quand personnage touché » au lieu de la commande « quand drapeau-vert touché » pour commencer le programme du personnage « papillon ». Elle dit de vive voix dans la vidéo « Ce n'est pas ça ! » et puis, elle la corrige. Au cours de la résolution du PB2, elle nous pose une question très précise, pour vérifier si elle a bien créé le motif des commandes à répéter du personnage « lapin ». Elle dit donc à voix haute : « Il dit d'abord bonjour le papillon ? » en se référant au personnage « lapin ». De cette manière, l'élève 2 arrive à corriger l'ordre des commandes de son motif avant d'exécuter le programme du « lapin », en évitant ainsi la réalisation d'une erreur.

L'élève 2 ne réalise aucune erreur au PB1. Au PB2, les erreurs existantes après la première exécution concernent le déplacement du personnage « papillon ». Chaque fois que son programme ne correspondait pas à ce qu'elle souhaitait après l'exécution, nous observons dans la vidéo qu'elle s'adressait à elle-même en disant « Ah ! » ou « Non ! » et puis procédait à son débogage. En analysant sa programmation au PB2, nous avons identifié deux types de débogage (non efficace et efficace).

Pour corriger ces erreurs au PB2, elle a eu besoin de réaliser trois débogages, deux « non efficaces » et un « efficace ». Nous sommes aussi intervenue trois fois pendant la réalisation de ces débogages. Nous avons donné deux fois la consigne du problème, afin de lui rappeler l'endroit où ses personnages doivent arriver dans la scène. Pour la même raison, nous avons également lancé une fois le projet modèle.

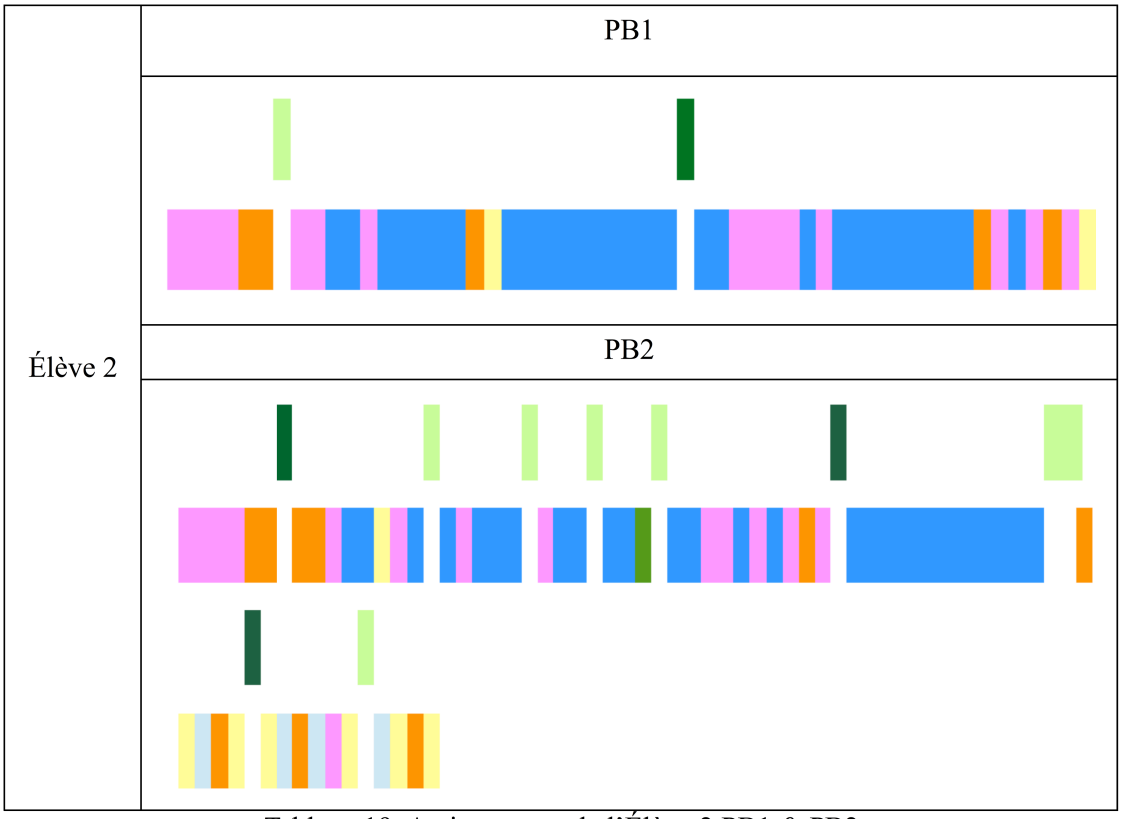


Tableau 19: Actigrammes de l'Élève 2 PB1 & PB2
Rappel de catégories d'actions : interface, commandes, corrections, initialisation, exécution
Aides: Consigne, Lancement projet, Initialisation

En analysant son actigramme pour le PB2 (voir Tableau 19), nous avons repéré un pattern de débogage où l'initialisation est après l'action de correction. Le pattern de débogage identifié est ce qui suit : *Exécution-Correction-Initialisation* (voir Tableau 20). Nous avons aussi identifié un pattern de débogage sans initialisation : *Exécution-Correction* (voir Tableau 20). Nous avons aussi repéré un autre débogage avec une double correction sur le même personnage : *Exécution-Correction-Initialisation-Correction-Interface*. Ce dernier n'est pas considéré en tant que pattern car il a été retrouvé une seule fois dans les actigrammes d'un seul élève.

	Pattern de débogage	Suite d'actions	Morceau d'actigramme
Élève 2	Pattern de débogage où l'initialisation est après la correction	Exécution→ Correction → Initialisation	
	Pattern de débogage sans initialisation avec une seule correction	Exécution→ Correction	

Tableau 20: Types de patterns de débogage identifiés sur les actigrammes de l'Élève 2

Toutes les corrections effectuées au PB2, à part la première réalisée sur la commande de « déplacement », démontrent une réflexion préalable par rapport au résultat obtenu après l'exécution de ses programmes. Il est possible que l'élève 2 ait procédé à une correction qui ne correspondait pas au résultat obtenu après l'exécution, parce que, d'après ce qu'on voit sur l'enregistrement vidéo, le personnage « papillon » cachait l'endroit demandé et elle ne pouvait pas bien voir si ce dernier arrivait bien sur l'endroit demandé. À la fin, aucune erreur ne reste dans ses programmes (voir Tableau 16 et 17). L'élève 2 n'a pas eu de moments de blocage pendant la résolution de deux problèmes.

2.2.9. Les aides apportées

Pour récapituler les aides qu'elle a reçues, nous en avons noté deux au PB1 et dix au PB2. Au PB1, elle a reçu au début une aide sur l'initialisation pour bien positionner son personnage « enfant » avant de commencer sa programmation et une deuxième pour effacer une commande de l'espace de programmation, parce que sans le faire exprès elle avait effacé tout son programme au lieu d'effacer uniquement celle-ci. Au PB2, en plus de trois aides sur l'initialisation et les quatre aides de rappel de la consigne pour l'écriture de la « répétition prédéfinie », elle a reçu trois autres aides. Il s'agit de deux rappels de la consigne et un lancement du projet modèle, pour lui rappeler où il devrait arriver son personnage « papillon ».

2.3. Élève 3 : silencieux et bloqué face à l'erreur

2.3.1. Résolution des problèmes (PB1 et PB2)

L'élève 3 arrive, avec beaucoup d'aides, à résoudre les deux problèmes de programmation. En analysant sa performance aux PB1 et PB2, nous observons que malgré le fait qu'il réalise plus d'erreurs pendant le PB1, il a plus de mal à résoudre le PB2.

En analysant ses programmes finaux pour le PB1 (voir Tableau 21) et le PB2 (voir Tableau 22), nous observons qu'il poursuit la suite d'actions de déplacement proposée dans les projets modèles. La seule erreur restante sur les projets de l'élève 3 est le fait que le personnage « enfant » arrive presque à l'endroit demandé au cours du PB1, car sa position initiale est erronée. Malgré les aides apportées sur l'initialisation de la position de ses personnages

l'élève en question n'arrive pas à repérer cette erreur et la corriger tout seul. Pour le PB2, l'élève 3 a réussi de trouver une autre façon pour faire répéter trois fois le motif des commandes au personnage « lapin » sans utiliser la commande de « répétition ». Nous considérons cette solution correcte car l'élève a réussi à reproduire le projet modèle en respectant la consigne du problème donné.

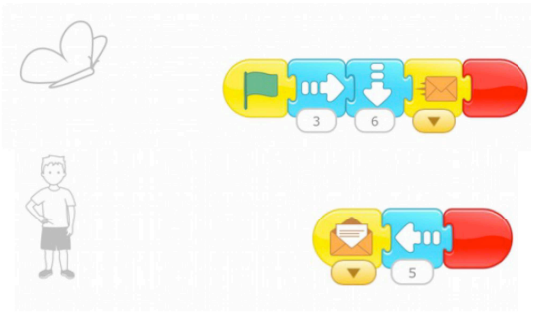
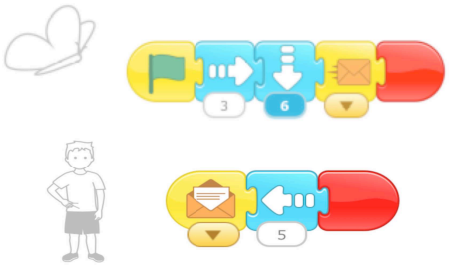
PB1	
Projet modèle	
Élève 3	

Tableau 21: Programmes finaux de l'Élève 3 PB1

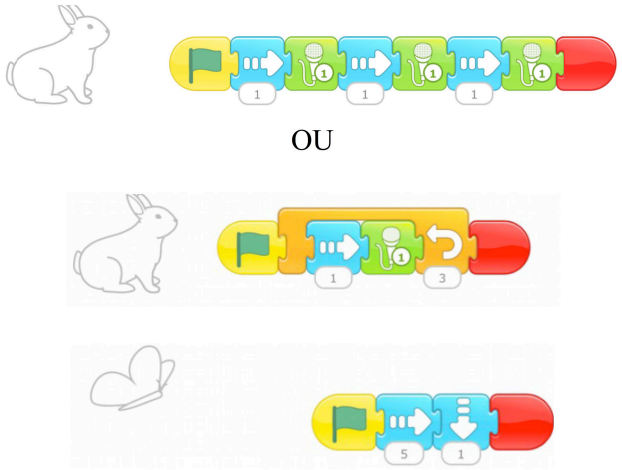
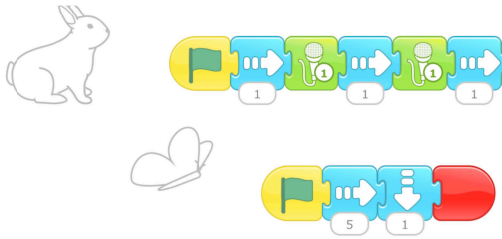
PB2	
Projet Modèle	
Élève 3	

Tableau 22: Programmes finaux de l'Élève 3 PB2

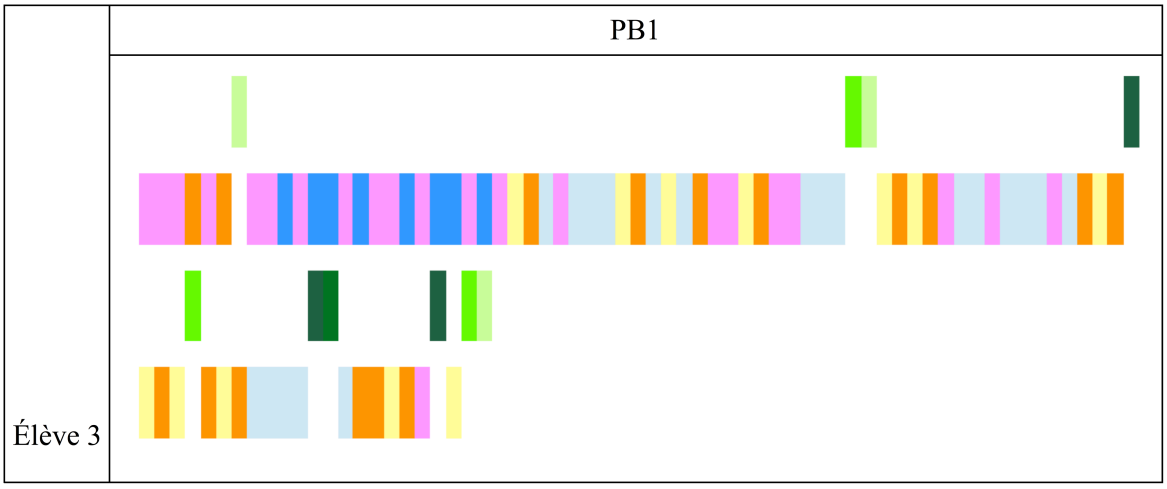
Comme nous pouvons le voir dans le Tableau 23, il passe presque le même temps sur les deux problèmes, mais il réalise un peu plus d'actions sur le PB2. En effet, il a eu besoin de 79 actions et 6min02s pour résoudre le PB1 et 87 actions et 6min34s pour résoudre le PB2. La performance de l'élève 3 au cours du PB2 est intéressante, parce qu'elle change au cours de la vidéo. Nous pouvons remarquer que la plupart de ses actions concernent l'interface et les corrections sur le PB1 et l'interface et les commandes sur le PB2 (voir Tableau 23). De plus, le nombre d'actions sur l'interface est égal au nombre d'actions sur les commandes pour le PB2.

Élève 3		
	PB1	PB2
Interface	22	35
Commandes	8	35
Corrections	20	6
Initialisation	16	9
Exécution	12	4
Question posée par l'élève	0	0
Total d'actions par élève	78	89
Temps de résolution	6min02s	6min34s
Aides	10	19
Problème réussi	Oui	Oui

Tableau 23: Récapitulatif d'actions de l'Élève 3 PB1 & PB2

2.3.2. La conception des programmes

En étudiant le comportement de l'élève 3 pendant le PB1, nous pouvons remarquer qu'il prend son temps pour écrire les programmes de ses personnages. Il commence en écrivant une partie du programme d'un personnage, ensuite il passe à l'autre, il écrit une partie du programme de celui-ci, puis il exécute le programme du premier personnage et il procède de la même manière jusqu'à la fin de la résolution du problème. Il réalise donc plusieurs allers-retours entre les deux personnages pendant l'écriture de ses programmes. Au PB2, il nous donne l'impression qu'il n'est pas sûr de ce qu'il fait, notamment pour la programmation du personnage « papillon ». En effet, il choisit des commandes et hésite à les ajouter dans ses programmes. Toutefois, nous observons qu'au PB2 il écrit d'abord entièrement le programme de l'un personnage et ensuite il passe à l'écriture du programme de l'autre.



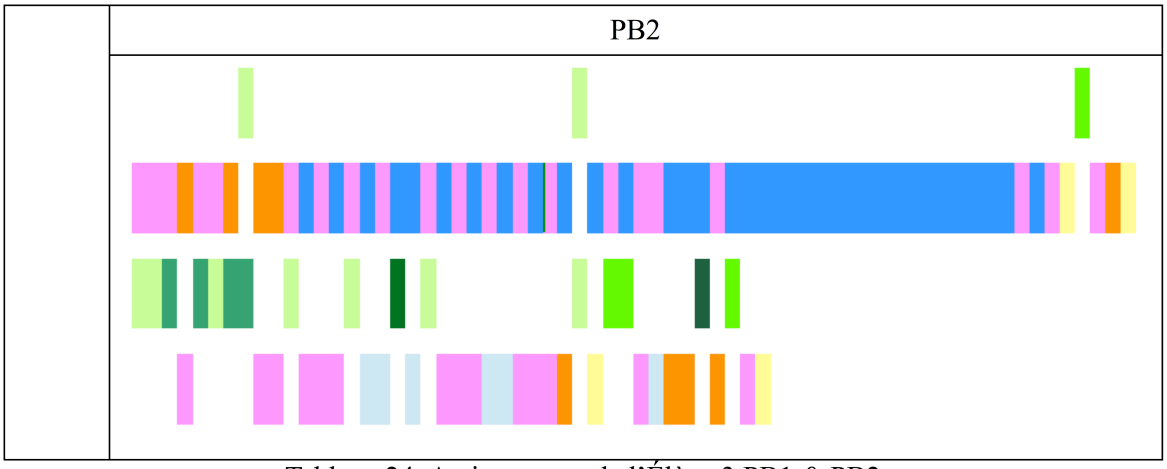


Tableau 24: Actigrammes de l'Élève 3 PB1 & PB2
Rappel de catégories d'actions : interface, commandes, corrections, initialisation, exécution
Aides : Consigne, Lancement projet, Initialisation, Vérification de programmation, Identification de l'erreur , Effacement d'une commande

Pour la conception de ses programmes au PB1, l'élève 3 met en place une stratégie de « sous-programme », car il écrit une partie du programme de ses deux personnages avant la première exécution. Pour le personnage « papillon », il programme les commandes « quand drapeau vert cliqué », « déplacement à droite » et « fin » et pour le personnage « enfant », il programme les commandes « quand message reçu », « déplacement à droite » et « fin ». Pour programmer le reste de ses programmes, il met en place plusieurs fois une stratégie de « pas à pas ». Pour la conception de ses programmes au PB2, l'élève met en place une stratégie de « planification complète », car il écrit l'intégralité des commandes des programmes de ses personnages avant la première exécution.

En faisant une comparaison entre son actigramme de couleurs pour le PB1 et le PB2 (voir Tableau 24), nous observons directement une différence majeure en ce qui concerne la conception de ses programmes, car dans le PB1 la partie de l'alternance entre l'interface et les commandes s'arrête très tôt, tandis que sur le PB2 elle dure plus longtemps. Cela peut être aussi justifié par le Tableau 23 où nous observons que sur le PB1 l'élève 3 réalise plus d'actions sur les corrections, tandis que sur le PB2 il réalise plus d'actions sur les commandes.

2.3.3. La gestion de l'interface

Concernant la gestion de l'interface, l'élève 3 a beaucoup de difficultés face à l'initialisation de la position de ses personnages au cours de la résolution de deux problèmes. Il préfère utiliser son doigt pour initialiser la position de ses personnages. En effet, sur les 16

initialisations qu'il fait au PB1, il n'en fait que 2 avec l'icône et sur les 9 initialisations qu'il fait au PB2, il n'en fait qu'une avec l'icône d'initialisation.

Au début du PB1, nous avons rappelé une fois la consigne du problème à l'élève 3 pour qu'il puisse positionner correctement ses personnages avant de commencer sa programmation. Suite à cela, la position initiale de son personnage « enfant » reste erronée pendant 11 exécutions au cours de la vidéo. L'élève a même eu un moment de blocage lié à l'initialisation de la position de ses personnages au cours de ce problème. En analysant son actigramme pour le PB1 (voir Tableau 24), nous repérons 4 fois consécutives le pattern « initialisation-exécution » (voir Tableau 25) sans corrections, car pendant ce temps-là l'élève 3 initialisait correctement seulement la position de l'un des deux personnages et cela l'empêchait d'identifier les erreurs et de corriger ses programmes.

Voyant qu'il était bloqué, nous l'avons assisté trois fois sur l'initialisation, en lui montrant la position initiale correcte. L'élève 3 a continué à se tromper sur cela même après l'apport d'aide. Sa difficulté au cours du PB1 peut aussi être identifiée si on regarde le nombre d'actions qu'il réalise sur l'initialisation au cours du PB1 (voir Tableau 23). En effet, il réalise 17 actions sur l'initialisation au PB1.

Au début du PB2, nous lui avons de nouveau rappelé une fois la consigne du problème à l'élève 3 pour qu'il puisse positionner correctement ses personnages avant de commencer sa programmation. Suite à notre intervention, il se trompe une fois sur l'initialisation de la position du personnage « papillon », mais après l'aide apportée là-dessus il arrive à corriger son erreur. Cela se voit aussi sur le Tableau 23, où le nombre d'actions sur l'initialisation de la position des personnages au PB2 est beaucoup moins élevé que celui au PB1.

2.3.4. La gestion du déplacement des personnages

Pour le déplacement de ses personnages, il utilise la stratégie « paramètres ». Une fois seulement il programme le déplacement en utilisant plusieurs fois la même commande et puis, il réduit l'écriture du codage en utilisant une seule commande et une stratégie de déplacement similaire à celle de l'élève 2, mais sans la « grille ».

> L'élève 3 appuie plusieurs fois sur la même commande de « déplacement » jusqu'à ce que son personnage arrive sur le bon endroit dans la scène. Ensuite, il utilise comme paramètre de la commande de « déplacement » le numéro correspondant au nombre de fois qu'il a eu besoin d'appuyer sur celle-ci, pour que son personnage arrive au bon endroit dans la scène.

Ses principales erreurs en lien avec le déplacement aux PB1 et PB2, concernent l'évaluation de la distance réalisée par le personnage et la direction du déplacement. En analysant son activité de programmation au PB1, nous pouvons remarquer qu'il arrive avec deux aides à corriger les erreurs sur le déplacement du personnage « papillon ». En effet, nous lui avons rappelé la consigne du problème et nous lui avons proposé de vérifier l'exactitude de sa programmation.

L'élève 3 a beaucoup plus de difficultés à corriger les erreurs sur le déplacement du personnage « enfant » au PB1 et en particulier, celles qui sont dues à une mauvaise initialisation de la position de ce personnage. Pour corriger ces dernières, à part les trois aides sur l'initialisation que nous avons déjà mentionnées, il a besoin de quatre aides. Nous l'avons aidé en lui rappelant deux fois de vérifier l'exactitude de sa programmation, en effaçant une commande qu'il pensait qu'il devrait ajouter à son programme parce qu'il avait mal initialisé la position du personnage « enfant » et en lançant le projet modèle pour qu'il puisse vérifier s'il a bien programmé ou pas.

En analysant son activité de programmation au PB2, nous observons qu'il arrive à corriger les erreurs sur le déplacement du personnage « papillon » avec cinq aides. À part l'aide sur l'initialisation que nous avons déjà présentée, nous l'avons assisté en lançant le projet modèle pour qu'il puisse vérifier s'il a bien programmé, en lui rappelant trois fois de vérifier l'exactitude de sa programmation.

2.3.5. Les messages et la gestion du démarrage synchrone de deux personnages

En ce qui concerne les commandes de « messages », l'élève 3 arrive, sans avoir besoin d'aide, à les utiliser de manière opérationnelle au PB1 (voir Tableau 21). En effet, il utilise les deux commandes, « envoyer un message » et « quand message reçu », dans l'ordre correct sur le programme de deux personnages. Il n'a pas de moment de blocage là-dessus au PB1. Pourtant, il oublie d'ajouter la commande « envoyer un message » sur le programme du personnage « papillon » pendant 4 exécutions.

Nous observons également qu'il utilise les commandes de « messages » aussi sur le PB2, même s'il n'en a pas besoin. Il essaie d'utiliser la commande « envoyer un message » au programme du personnage « lapin », mais il l'efface avant de l'y intégrer une fois que nous

lui rappelons la consigne du PB2. Dans la vidéo, on le voit aussi hésiter entre les commandes « quand drapeau-vert cliqué » et « quand message reçu », pour déclencher le programme du personnage « papillon ». Il choisit finalement la commande « quand message reçu » en réalisant une erreur sur le démarrage synchrone de deux personnages.

Après l'exécution de ses programmes, nous sommes intervenue pour l'aider à surmonter cette difficulté. Ainsi, une série d'aides ont été apportées à l'élève 3. Nous avons commencé par lui proposer de vérifier l'exactitude de sa programmation. Ensuite, nous lui avons rappelé la consigne du problème donné. Nous avons aussi lancé le projet modèle pour qu'il puisse se rappeler du résultat souhaité à reproduire. Puis, nous avons commencé à lui poser des questions pour l'aider à identifier l'erreur sur son programme. Nous avons fait cela deux fois. Puis, nous avons lancé le projet de l'élève pour l'aider à le comparer avec le résultat souhaité et nous lui avons encore posé deux questions pour l'aider à identifier son erreur. Par la suite, nous avons rappelé à nouveau la consigne du problème donné.

Malgré les aides apportées, l'élève se trouve devant plusieurs moments de blocage, face à cette erreur sur le démarrage du programme du personnage « papillon ». Il ne procède pas à des corrections peut-être parce qu'il n'arrive pas à identifier son erreur. Il est donc toujours en difficulté. Cela peut être vu aussi sur son actigramme, où après les deux exécutions consécutives il y a une pause sur ces actions et beaucoup d'aides qui sont apportées par nous (voir Tableau 24).

Il réalise ensuite un débogage « non efficace », car il essaie de corriger le programme du « papillon » en utilisant la commande de « répétition ». Cette correction n'est pas adaptée au problème rencontré. C'est pourquoi, nous l'avons interrompu afin qu'il ne réalise pas une mauvaise correction qui pourrait ralentir encore plus son processus de débogage de l'erreur sur le démarrage du personnage « papillon ». L'élève efface par la suite la commande de « répétition » et corrige l'erreur sur le démarrage du programme du « papillon », en réalisant un débogage « efficace » cette fois-ci. Il met donc la commande « quand drapeau-vert cliqué » à la place de la commande « quand message reçu » au début du programme du personnage « papillon », en réalisant ainsi correctement le démarrage synchrone de deux personnages (voir Tableau 22).

Les actions de l'élève 3 concernant les commandes de « messages » ont marqué notre attention et c'est pourquoi nous avons fait le choix d'en discuter avec lui après la fin de sa

programmation. Nous lui avons donc demandé de nous expliquer pourquoi il a utilisé la commande « quand message reçu » au début du programme du personnage « papillon », mais l'élève n'a pas répondu. Ensuite, nous lui avons demandé de nous rappeler quand on utilise cette commande et l'élève nous a donné une mauvaise réponse : « on l'utilise quand on veut que le personnage « papillon » commence son script en premier ». Nous observons donc qu'il a confondu le sens de la commande « quand message reçu » avec celle de « quand drapeau-vert cliqué ». La dernière question concernait l'utilisation de « messages » au PB1 et là, l'élève 3 a donné une réponse qui décrivait une sorte de communication entre les deux personnages, mais elle n'était pas très complète puisque l'envoie du message n'en faisait pas partie. L'élève a répondu qu'il a utilisé ces commandes « parce que le papillon a dit au garçon qu'il peut commencer ».

2.3.6. La répétition prédéfinie

En ce qui concerne le motif des commandes à répéter par le personnage « lapin » au PB2, l'élève 3 arrive à reproduire le résultat souhaité en utilisant trois fois le même motif des commandes, au lieu d'utiliser la commande de « répétition » (voir Tableau 22). Il s'agit d'un choix fait par la plupart des élèves de deux classes de CP étudiées. Comme nous avons vu au Chapitre 1 de cette Partie, la majorité des élèves de grande section de maternelle en Grèce a aussi préféré de créer trois fois le même motif des commandes plutôt que d'utiliser la commande de « répétition », pour résoudre le PB2.

2.3.7. Les erreurs réalisées et le débogage mis en place

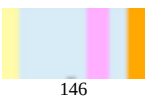
Par rapport aux erreurs réalisées, nous avons déjà repéré au PB1 celles sur l'évaluation de la distance à réaliser par ses personnages, la direction du déplacement, la mauvaise initialisation de la position des personnages et l'oubli de la commande « envoyer un message ». Les erreurs réalisées sur le PB2 concernent à nouveau l'évaluation de la distance à réaliser par son personnage et le démarrage du programme du personnage « papillon » avec la commande « quand message reçu » au lieu de « quand drapeau-vert cliqué ». Il se trompe aussi une seule fois sur l'initialisation de la position du personnage « papillon ». Ces erreurs apparaissent plusieurs fois pendant la résolution de deux problèmes d'évaluation.

L'élève 3 n'arrive pas à repérer les erreurs sur ses programmes avant de les exécuter. Il n'identifie pas tout de suite toutes les erreurs présentes sur ses programmes même après leur

exécution et leur correction lui prend du temps. Il se concentre chaque fois sur la correction d'une erreur. Il a eu besoin d'exécuter 12 fois ses programmes pour corriger toutes ses erreurs au cours du PB1.

Nous observons que pour corriger ses erreurs il réalise plusieurs débogages « non efficaces » avant d'arriver au débogage « efficace ». Plus précisément, l'élève 3 a eu besoin de réaliser 6 débogages au PB1 et 2 au PB2. Ce qui reste erroné jusqu'à la fin de sa programmation, même si les programmes de l'élève ne contiennent pas d'erreur, c'est la position initiale de son personnage « enfant » au PB1.

En analysant son actigramme pour le PB1 (voir Tableau 24), nous repérons un pattern de débogage où l'action d'initialisation arrive avant la ou les corrections (voir Tableau 25). Le pattern qui suit a été identifié 3 fois sur l'actigramme de l'élève 3 : *Exécution-Initialisation-Correction.* Nous avons aussi observé que la plupart des fois que ce pattern apparait sur les actigrammes de l'élève, il contient plusieurs actions de corrections. Nous avons aussi pu repérer un autre pattern de débogage où l'action d'initialisation est après la correction, une fois au cours du PB1 et deux fois au cours du PB2 (voir Tableau 25). Nous avons enfin pu observer une autre succession de type d'actions, où il y a une action d'initialisation avant et une après les actions de correction. Ce dernier n'est pas considéré en tant que pattern, car il a été retrouvé une seule fois dans les actigrammes d'un seul élève et il n'a non plus été retrouvé dans les actigrammes des autres élèves.

	Pattern de débogage	Suite d'actions	Morceau d'actigramme
Élève 3	Pattern de débogage où l'initialisation est après la correction ou les corrections	Exécution→ Correction → Initialisation Exécution→ Correction→ Correction→ Correction→ Interface→ Correction →Initialisation	  146
	Pattern de débogage où l'initialisation est avant la	Exécution→ Initialisation→ Correction	

146 Ce morceau d'actigramme est utilisé comme un exemple d'un débogage avec plusieurs actions de correction et l'initialisation après les corrections. Ce n'est pas un pattern qui est retrouvé plusieurs fois dans les actigrammes des élèves. Ce pattern pourrait avoir des formes différentes. Ces formes contiennent toujours des actions d'exécution, de correction et d'initialisation et ce qui change chaque fois c'est l'existence des actions d'interface, des questions posées par l'élève ou des aides apportées.

	correction ou les corrections	 Exécution→ Initialisation→ Correction→ Interface→ Correction→ Correction→ Correction	 147 

Tableau 25: Types de patterns de débogage identifiés sur les actigrammes de l'Élève 3

Les corrections effectuées sur ses programmes correspondent au résultat obtenu après l'exécution, à part celles sur la correction de la commande « quand message reçu » au cours du PB2. L'élève 3 ne nous pose aucune question.

L'élève 3 a eu un seul moment de blocage au PB1 en lien avec l'initialisation de la position du personnage « enfant ». Toutefois, au PB2 il a eu 10 moments de blocage en lien avec le démarrage du programme du personnage « papillon », l'erreur sur le démarrage du programme de ce dernier, l'utilisation de la commande « envoyer un message » au programme du personnage « enfant » et enfin, avec la correction du déplacement du personnage « papillon ».

2.3.8. Les aides apportées

Pour récapituler les aides reçues par l'élève 3, nous avons noté 10 au PB1 et 19 au PB2. Au cours du PB1, il a été assisté au début sur le positionnement de ses personnages, ensuite sur l'initialisation de la position du personnage « enfant » et sur le débogage du déplacement de ses deux personnages. Plus précisément, il a reçu un rappel de la consigne pour bien positionner ses personnages au début, trois aides sur l'initialisation de la position du personnage « enfant », une aide sur l'effacement d'une commande liée à la mauvaise initialisation du personnage « enfant », un rappel de la consigne pour le débogage du déplacement du « papillon », une aide sur la vérification de l'exactitude de sa programmation pour le débogage du déplacement du « papillon », deux aides sur la vérification de l'exactitude de sa programmation pour le débogage du déplacement de l' « enfant » et un lancement du projet modèle pour qu'il puisse vérifier s'il a bien programmé ou pas.

Au cours du PB2, l'élève 3 a été assisté sur le positionnement de ses personnages, sur le démarrage synchrone de deux personnages, sur le débogage de l'erreur portant sur le démarrage du personnage « papillon », sur le débogage du déplacement du personnage « papillon » et une fois sur l'initialisation de la position du même personnage. Plus

147 Voir note 146.

précisément, il a reçu un rappel de la consigne pour bien positionner ses personnages au début et un rappel de la consigne sur le démarrage synchrone de deux personnages. Il a également reçu quatre rappels de la consigne, deux lancements du projet modèle, une aide sur la vérification de l'exactitude de sa programmation, quatre aides sur l'identification de l'erreur et une aide sur l'effacement d'une mauvaise correction pour le débogage de l'erreur sur le démarrage du programme du « papillon ». Il a aussi reçu quatre aides pendant le débogage du déplacement du personnage « papillon » : un lancement du projet modèle et trois aides sur la vérification de l'exactitude de sa programmation. Enfin, il a également été assisté une fois sur l'initialisation de la position du personnage « papillon ».

En analysant ses programmes finaux pour le PB1 (voir Tableau 21) et le PB2 (voir Tableau 22), nous observons qu'il poursuit la suite d'actions de déplacement proposée dans les projets modèles. La seule erreur restante sur les projets de l'élève 3 est le fait que le personnage « enfant » arrive presque à l'endroit demandé au cours du PB1, car sa position initiale est erronée. Malgré les aides apportées sur l'initialisation de la position de ses personnages l'élève en question n'arrive pas à repérer cette erreur et à la corriger tout seul. Pour le PB2, l'élève 3 a réussi de trouver une autre façon pour faire répéter trois fois le motif des commandes au personnage « lapin » sans utiliser la commande de « répétition ». Nous considérons cette solution correcte car l'élève a réussi à reproduire le projet modèle en respectant la consigne du problème donné.

2.4. Élève 4 : obstiné à corriger, mais en échec

2.4.1. Résolution des problèmes (PB1 et PB2)

La performance de l'élève 4 est très différente entre le PB1 et le PB2. L'élève 4 arrive avec de l'aide à résoudre le PB1 (voir Tableau 26), mais pas le PB2 (voir Tableau 27).En analysant ses programmes finaux pour le PB1 (voir Tableau 26), nous observons qu'ils ne contiennent pas d'erreurs. Nous repérons également qu'il poursuit la suite d'actions de déplacement proposée dans le projet modèle. En analysant ses programmes finaux pour le PB2 (voir Tableau 27), nous observons que le programme du personnage « lapin » est erroné, puisque le personnage réalise « un pas à droite et il dit *trois fois* la phrase demandée » et ces deux commandes il le fait trois fois, au lieu de réaliser « un pas à droite et dire *une fois* la phrase demandée » et faire

cela trois fois. Nous remarquons que pour le programme du personnage « papillon » l'élève 4 poursuit la suite d'actions de déplacement proposé dans le projet modèle.

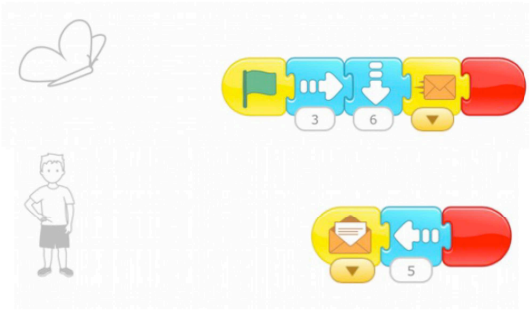
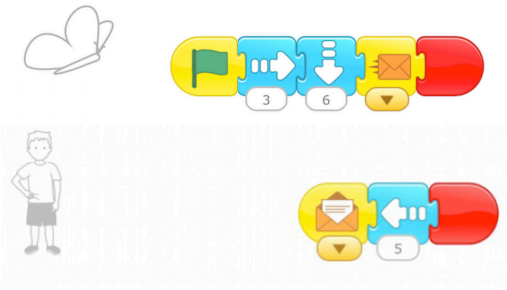
PB1	
Projet modèle	
Élève 4	

Tableau 26: Programmes finaux de l'Élève 4 PB1

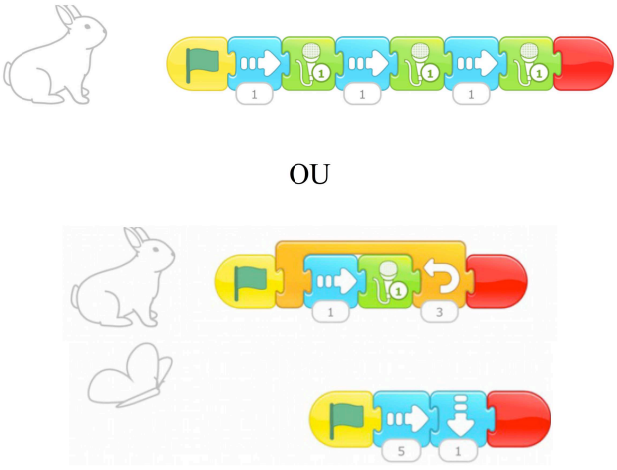
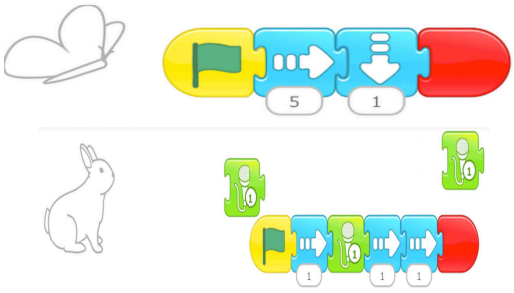
PB2	
Projet Modèle	
Élève 4	

Tableau 27: Programmes finaux de l'Élève 4 PB2

En analysant le Tableau 28 nous repérons qu'il a eu besoin de réaliser 39 actions et 3min10s et 8 aides pour résoudre correctement le PB1, et 146 actions et 12min16s et 28 aides sans pour autant arriver à résoudre le PB2. Nous observons donc qu'il a eu besoin de plus de temps pour résoudre le PB2 que pour résoudre le PB1 (voir Tableau 28).Nous repérons aussi que la plupart de ses actions sur le PB1 concernent l'interface et ensuite, les commandes, tandis que sur le PB2 c'est les corrections et ensuite l'interface. Nous pouvons également observer qu'il réalise le même nombre d'actions sur les corrections et l'exécution au cours du PB1. Le nombre d'actions sur l'initialisation est aussi très proche du nombre d'actions sur les corrections et l'exécution.

Élève 4		
	PB1	PB2
Interface	13	26
Commandes	12	23
Corrections	5	53
Initialisation	4	20
Exécution	5	22
Question posée par l'élève	0	2
Total d'actions par élève	39	146
Temps de résolution	3min10s	12min16s
Aides	8	28
Problème réussi	Oui	Non

Tableau 28: Récapitulatif d'actions de l'Élève 4 PB1 & PB2

2.4.2. La conception des programmes

Les actions de l'élève 4 au cours du PB1 nous montrent qu'il sait comment le résoudre. Pour être plus précis, on voit un élève qui n'hésite pas avant chaque action et il a une progression fluide dans sa programmation. En général, il réalise des bonnes estimations en ce qui concerne le nombre de pas dont ses personnages doivent réaliser dans la scène pour arriver sur l'endroit demandé.

Au cours du PB2, nous observons une insécurité importante aussi bien sur ses actions que sur ses réponses aux questions posées. En effet, l'élève 3 est beaucoup moins sûr de ce qu'il fait surtout pendant l'écriture du programme du personnage « lapin ». Il nous donne l'impression de comprendre qu'il y a un problème avec le programme du personnage « lapin », mais de ne

pas réussir à identifier lequel, malgré les aides reçues. Nous remarquons toutefois qu'il n'hésite pas face à la programmation du personnage « papillon ».

Pour la conception de ses programmes au cours du PB1, nous observons qu'il traite d'abord le programme de l'un des personnages et vérifie sa programmation avant de procéder au deuxième personnage. Il met en place une approche de « sous-programme » pour ses deux personnages, car il écrit une partie de ses programmes puis les exécute pour vérifier sa programmation. Pour programmer le reste de ses programmes, il met en place plusieurs fois une stratégie de « pas à pas ».

Pour la conception de ses programmes au PB2, nous observons qu'il ne procède pas de la même manière qu'au PB1, où il traitait d'abord le premier personnage, vérifiait sa programmation, puis s'occupait du deuxième. Au PB2, nous pouvons noter beaucoup d'aller-retours entre les deux personnages, c'est-à-dire que l'élève crée une partie du programme d'un personnage et ensuite il crée une partie du programme de l'autre personnage, puis il passe au premier, etc.

Pour la programmation de ses personnages au PB2, il met en place une stratégie de « sous-programme », car il écrit une partie de son programme et puis l'exécute pour voir s'il l'a bien programmé. Il procède de la même manière pour le reste du programme du « papillon ». En revanche, pour le personnage « lapin », il met aussi en place une stratégie de « Recommencer au début » (ou « Start over » en anglais), car il efface tout le programme du personnage et il recommence sa programmation en réalisant une stratégie de « planification partielle » cette fois-ci. Suite à la mise en place de cette stratégie, l'élève 4 a produit le programme le plus proche au résultat souhaité de toute son activité de programmation sur le PB2. De plus, pendant la résolution du PB2, il choisit souvent de segmenter le programme du « lapin » ou d'unifier à nouveau ses commandes, pour le tester en entier.

2.4.3. La gestion de l'interface

En ce qui concerne les éléments d'interface, nous observons qu'il les gère très bien, mis à part l'initialisation qu'il lui pose parfois des problèmes. Pour réaliser l'initialisation de ses personnages, l'élève 4 préfère, aussi bien au PB1 qu'au PB2, à utiliser son doigt plutôt que l'icône d'initialisation. Il se trompe une fois sur celle-ci pendant la résolution du PB1. Par la suite de sa programmation, il initialise correctement la position de ses personnages avant

chaque exécution. Nous sommes intervenue une fois au début du PB1 en lui apportant une aide sur l'initialisation afin de l'aider à bien positionner son personnage « enfant ».

Pendant la résolution du PB2, il se trompe davantage sur l'initialisation de la position de ses personnages. Sur les 20 initialisations qu'il réalise au total, les huit sont erronées. Au PB2, nous lui avons rappelé une fois au début la consigne du problème pour l'aider à bien positionner son personnage « lapin ». Nous sommes également intervenue deux fois pendant la résolution du PB2 en lui apportant une aide sur l'initialisation, afin de l'aider à bien positionner son personnage « papillon ».

2.4.4. La gestion du déplacement des personnages

Pour réaliser le déplacement de ses personnages au PB1, l'élève 4 utilise la stratégie « paramètres ». Les erreurs existantes au PB1 sur le « déplacement » concernent l'évaluation de la distance réalisée par les deux personnages, le « papillon » et l' « enfant ». L'élève arrive avec une aide de notre part à corriger l'erreur sur la programmation du déplacement du personnage « papillon ». En effet, nous lui avons rappelé la consigne du problème donné. Il a eu toutefois plus de difficultés à repérer l'erreur sur le déplacement du personnage « enfant ». C'est pourquoi nous lui avons apporté trois aides là-dessus. En effet, nous lui avons rappelé deux fois de vérifier l'exactitude de sa programmation et la troisième fois nous lui avons posé une question plus précise pour l'aider à identifier l'erreur sur son programme. L'élève a réussi finalement à corriger l'erreur sur le programme du personnage « enfant ».

Pour le déplacement de ses personnages au PB2, l'élève 4 utilise à nouveau les « paramètres et plusieurs fois le même type de commande ». Sa principale erreur au PB2 en lien avec le « déplacement » concerne l'évaluation de la distance réalisée par le personnage « papillon ». L'élève a du mal à identifier cette erreur car il est longtemps préoccupé par la correction d'une autre erreur sur le programme du personnage « lapin ». C'est pourquoi, il prend beaucoup plus de temps et d'essais à la corriger que ce qu'il prend pour corriger le même type d'erreur au PB1.

Pour corriger cette erreur il a eu besoin d'être assisté neuf fois. Nous avons d'abord lancé le projet modèle pour l'aider à repérer s'il avait bien programmé le déplacement du personnage « papillon ». Ensuite, nous lui avons rappelé deux fois de vérifier l'exactitude de sa programmation. Nous lui avons aussi posé trois fois une question plus précise pour l'aider à

identifier l'erreur sur son programme. Nous sommes aussi intervenue deux fois pour l'aider à bien positionner le personnage « papillon », parce que, comme on l'a vu plus haut, il s'est trompé sur l'initialisation de la position de ce personnage. Nous lui avons à nouveau rappelé de vérifier l'exactitude de sa programmation pour le déplacement du personnage « papillon ». L'élève 4 a enfin réussi de corriger l'erreur sur le déplacement du personnage « papillon ».

2.4.5. Les messages

En ce qui concerne les commandes des « messages », nous observons qu'il les utilise de manière opérationnelle (voir Tableau 26), au cours du PB1, sans aide, malgré le fait qu'il oublie la commande « envoyer un message » au début. Pourtant, nous observons qu'il essaie, d'utiliser la commande « envoyer un message » aussi dans le programme du personnage « lapin », alors que ce n'était pas nécessaire, pour l'effacer juste après.

2.4.6. La répétition prédéfinie

Au sujet du motif à répéter pour le personnage « lapin » au PB2, l'élève 4 n'utilise pas de manière opérationnelle la commande de « répétition » (voir Tableau 27). Il essaie de créer trois fois le motif des commandes mais il n'arrive pas à le faire même après plusieurs tentatives de lui venir en aide. En effet, l'identification du motif des commandes à répéter pose la plupart des difficultés à cet élève. Nous avons essayé au début de l'aider à la création du motif à répéter en lui rappelant la consigne du problème. L'élève se trompe très tôt pendant la résolution du problème sur la création de la commande d'enregistrement, car il enregistre trois fois la phrase demandée dans la même commande d'enregistrement. Il s'agit d'une erreur qui ressort très souvent dans nos données aussi bien de cours préparatoire que de grande section de l'école maternelle.

Nous sommes intervenue plusieurs fois pour l'aider à l'identifier cette erreur. Malgré nos aides, l'élève se retrouve face à des nombreux moments de blocage, aussi bien avant la première exécution de son programme qu'après chaque exécution de celui-ci. Nous avons noté sept moments de blocage en lien avec l'identification et la création du motif à répéter. Après plusieurs essais de débogage infructueux, l'élève nous demande de l'aide sur cela en disant : « Comment il arrive (le lapin) à faire ça ? » ou « Mais comment il arrive à faire de petits pas comme ça ? ». L'élève 4 n'arrive pas à identifier l'erreur sur le motif des

commandes à répéter et sur l'enregistrement du son pour le personnage « lapin », malgré les aides apportées.

Il réalise donc une série des débogages « non efficaces », parce que les corrections qu'il met en place ne sont pas adaptées à l'erreur. Plusieurs fois il choisit de segmenter le programme du « lapin » ou d'unifier à nouveau ses commandes, pour le tester en entier. Ce type de corrections n'aide pas l'élève à corriger le programme du « lapin ». Une seule fois, il utilise la commande de « répétition », mais il le fait d'une mauvaise manière (voir Figure 14).

Suite à cela, l'élève 4 a choisi d'effacer la totalité du programme du personnage « lapin » et de recommencer au début. Ce choix l'a guidé vers la création d'un programme, qui était le plus proche du résultat souhaité que tous les autres qu'il avait créés jusqu'ici, au cours du PB2 (voir Figure 15). Malgré cette intéressante surprise, l'élève 4 n'a pas pu repérer que la commande du « son » était toujours erronée et il a donc consacré le reste de sa programmation à la réalisation de débogages « non efficaces ». Le programme final du personnage « lapin » est donc erroné (voir Tableau 27).

Figure 14: État du programme de l'élève 4 après avoir ajouté la commande de répétition

Figure 15: État du programme de l'élève 4 suite à l'approche "Recommencer au début"

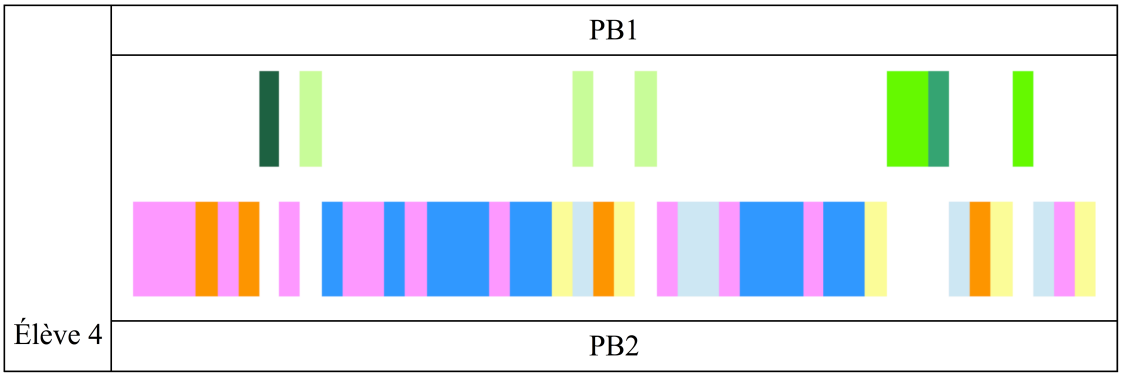
2.4.7. La gestion du démarrage synchrone de deux personnages

À propos du démarrage synchrone de deux personnages, l'élève 4 arrive à le mettre en place de manière opérationnelle sans problème (voir Tableau 27). En effet, il utilise de manière opérationnelle la commande « quand drapeau-vert cliqué » au début du programme de deux personnages et le bouton du drapeau-vert en haut de la scène.

2.4.8. Les erreurs réalisées et le débogage mis en place

L'élève 4 n'arrive pas à identifier les erreurs présentes sur ses programmes avant de les exécuter. Il a également besoin d'aide pour les identifier même après avoir exécuté ses programmes. En ce qui concerne les erreurs, nous avons déjà repéré celles sur l'initialisation, le déplacement, l'oubli de la commande « envoyer un message » au cours du PB1. L'élève 4 a aussi oublié d'ajouter la commande de « fin » au programme du personnage « enfant », mais après une aide sur la vérification de l'exactitude de sa programmation il a corrigé cette erreur.

Pendant un moment au début de la résolution du PB2 l'élève pensait que le personnage qui parlait était le « papillon » et non pas le « lapin ». Il a même failli enregistrer un son pour le personnage « papillon » mais nous l'avons interrompu en lui rappelant la consigne du problème et en l'aidant à effacer le menu d'enregistrement d'un son. À part de cette erreur, au cours du PB2 nous avons identifié d'autres sur le « déplacement », la « répétition », l'identification et la création du motif à répéter, l' « enregistrement d'un son » et l' « initialisation ». Certaines de ces erreurs apparaissant plusieurs fois au cours de la résolution du PB2. L'élève arrive avec de l'aide à corriger les erreurs sur le « déplacement » et l' « initialisation », mais il n'arrive pas à corriger celle sur la « répétition » et sur l'« enregistrement d'un son ».



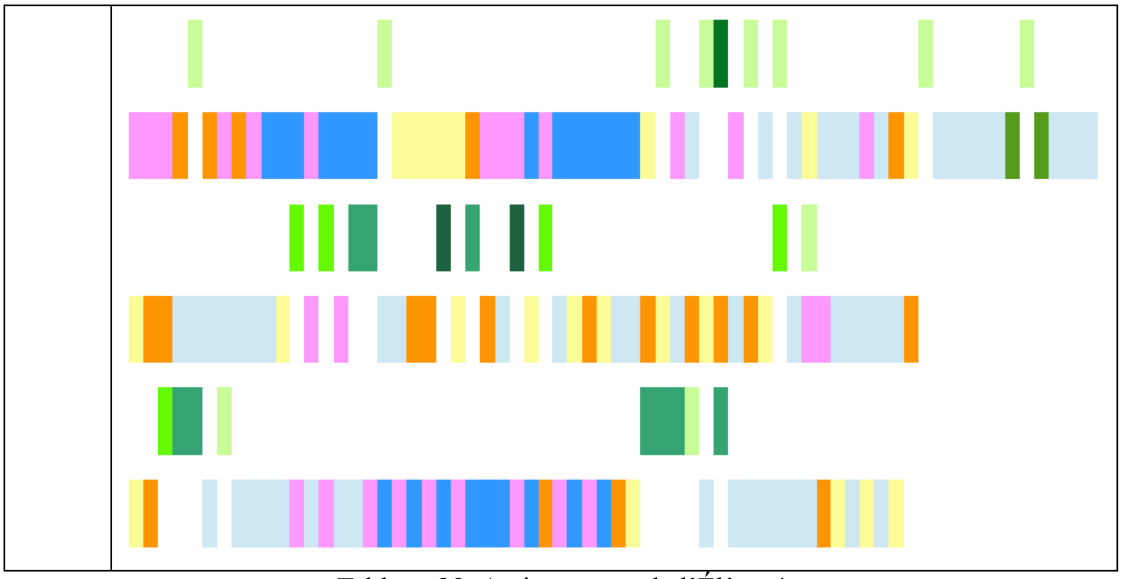


Tableau 29: Actigrammes de l'Élève 4

Rappel de catégories d'actions : interface, commandes, corrections, initialisation, exécution
Aides : Consigne, Lancement projet, Initialisation, Vérification de programmation, Identification de l'erreur , Effacement d'une commande

En analysant son actigramme pour le PB1 (voir Tableau 29), nous pouvons repérer deux fois un pattern de débogage où l'action d'initialisation se trouve après la correction. Nous pouvons aussi repérer deux fois un pattern de débogage avec une ou plusieurs corrections, mais sans initialisation. En ce qui concerne le dernier débogage, l'initialisation n'était pas nécessaire, parce que l'élève a exécuté son programme en plein-écran et là-bas les personnages exécutent leur programme à partir de l'endroit qui a été choisi la dernière fois comme le début du personnage. Il s'agit encore d'une spécificité du logiciel qui affecte le pattern de débogage de l'élève.

En analysant son actigramme pour le PB2 (voir Tableau 29), nous pouvons repérer que ses débogages contiennent souvent de nombreuses actions de correction et non pas une à deux actions. Nous observons que pour la majorité de ses débogages, six fois en particulier, l'élève 4 suit le pattern de débogage où l'initialisation de la position des personnages arrive après les corrections. Nous pouvons aussi repérer une fois le même pattern, mais avec une action de correction. Nous observons également deux fois le pattern où l'initialisation apparait avant les corrections et une fois le même pattern, mais avec une action de correction. Nous pouvons aussi remarquer trois fois le pattern de débogage sans initialisation avec une action de correction et 1 fois avec plusieurs actions de correction. Ces types de débogage sont présentés dans le Tableau 30 qui suit.

	Pattern de débogage	Suite d'actions	Morceau d'actigramme
Élève 4	Pattern de débogage où l'initialisation est après la correction ou Pattern de débogage où l'initialisation est après les corrections	Exécution→ Correction → Initialisation Exécution→ Correction→ Correction→ Correction→ Interface→ Correction →Initialisation	  148
	Pattern de débogage où l'initialisation est avant la correction ou les corrections	Exécution→ Initialisation→ Correction Exécution→ Initialisation→ Correction→ Interface→ Correction→ Correction→ Correction	  149
	Pattern de débogage sans initialisation avec une seule correction ou plusieurs corrections	Exécution→ Correction	  150

Tableau 30: Types de patterns de débogage identifiés sur les actigrammes de l'Élève 4

En ce qui concerne les corrections effectuées au cours du PB1, nous pouvons remarquer qu'elles correspondent au résultat obtenu après l'exécution des programmes. Les corrections qu'il réalise au cours du PB2 qui sont liées au motif des commandes à répéter ne correspondent pas au résultat obtenu après l'exécution de son programme. En ce qui concerne les corrections qu'il réalise sur le déplacement du personnage « papillon », il y en a qui correspondent au résultat obtenu après l'exécution et d'autres qui ne correspondent pas du tout.

148 Ce morceau d'actigramme est utilisé comme un exemple d'un débogage avec plusieurs actions de correction et l'initialisation après les corrections. Ce n'est pas un pattern qui est retrouvé plusieurs fois dans les actigrammes des élèves. Ce pattern pourrait avoir des formes différentes. Ces formes contiennent toujours des actions d'exécution, de correction et d'initialisation et ce qui change chaque fois c'est l'existence des actions d'interface, des questions posées par l'élève ou des aides apportées.

149 Voir note 148.

150 Voir note 148.

L'élève 4 n'a pas eu des moments de blocage au PB1. Au PB2, nous en avons noté huit. Sur les huit, les sept concernaient la répétition du motif des commandes pour le personnage « lapin » et un seul concernait l'insertion et puis l'effacement de la commande « envoyer un message » du programme du personnage « lapin ». Cet élève a aussi réclamé de l'aide, pendant la résolution du PB2, en nous posant deux questions que nous avons déjà présentées auparavant.

2.4.9. Les aides apportées

Pour récapituler les aides qu'il a reçues, nous avons noté 8 au PB1 et 28 au PB2. Au cours du PB1, il a été assisté sur l'initialisation de la position du personnage « enfant », sur le débogage du déplacement de ses personnages et enfin sur l'oubli des commandes « envoyer un message » et « fin ». Plus précisément, il a reçu une aide sur l'initialisation du personnage « enfant », un rappel de la consigne du problème à résoudre, un rappel de la consigne pour le débogage du déplacement de « papillon », deux aides sur la vérification de l'exactitude de sa programmation et une aide sur l'identification de l'erreur pour le débogage du déplacement de l' « enfant ». Il a aussi reçu un rappel de la consigne pour l'oubli de la commande « envoyer un message » et une aide sur la vérification de l'exactitude de sa programmation pour l'oubli de la commande de « fin ».

Au cours du PB2, l'élève 4 a été assisté sur le positionnement du personnage « lapin », le débogage du déplacement du personnage « papillon », le fait que le « papillon » ne parle pas, l'identification et la création du motif des commandes à répéter pour le personnage « lapin », l'enregistrement du son pour le personnage « lapin », l'initialisation de la position du personnage « papillon ». Plus précisément, il a reçu à 15 reprises de l'aide des différentes natures (5 fois consigne, 3 fois lancement, 1 fois vérification, 6 fois identification) sur la création et l'identification du motif des commandes à répéter. Il a également reçu huit aides (1 fois lancement, 4 fois vérification, 3 fois identification) de différents types sur le débogage du déplacement du personnage « papillon ». Il a reçu deux aides (1 fois consigne et 1 fois effacement d'une commande) sur le fait qu'il faut pas enregistrer de son pour le « papillon ». Il a enfin reçu une aide sur le premier positionnement du personnage « lapin » et deux aides sur l'initialisation de la position du personnage « papillon ».

Dans la section qui suit, nous allons interpréter ce que les quatre élèves de CP1 semblent être capables de faire.

3. Points de vue interprétatifs

Après avoir présenté les portraits de quatre élèves du CP1, nous allons dans cette section nous concentrer sur les interprétations de ce que font ces élèves en programmation. Pour chacun des élèves, nous allons essayer de comprendre les éléments suivants : la résolution de problèmes par l'élève, la conception des programmes, la gestion de l'interface, la gestion du déplacement des personnages, l'utilisation de « messages », l'utilisation ou non de la « répétition prédéfinie », la gestion du démarrage synchrone, les erreurs, le débogage mis en place et les aides apportées.

3.1. Comment comprendre ce que fait l'Élève 1 ?

3.1.1. Résolution de problèmes (PB1 et PB2)

Ses programmes finaux nous montrent que l'élève 1 est capable, avec peu d'aides, de résoudre deux problèmes de programmation portant entre autres sur l'utilisation opérationnelle des commandes de « messages » et de « répétition ». Cela nous fait également penser que cette élève a compris l'exécution séquentielle des commandes sur ScratchJr, puisqu'elle l'a pris en compte dans la création de ses programmes.

Au vu des éléments présents au portrait de l'élève 1 (voir section 2.1 dans ce chapitre), nous pouvons supposer qu'elle a compris la logique de la programmation sur ScratchJr et elle est à l'aise avec la résolution de problèmes en programmation, puisqu'elle est rapide, précise et sûre de ses actions. Sa facilité se confirme aussi par le fait qu'elle ne pose pas de questions pendant la résolution de deux problèmes et elle ne présente pas de moments de blocage. Un autre indice est le fait qu'elle a eu besoin de moins d'actions que tous les élèves pour résoudre les deux problèmes.

Le fait que la plupart de ses actions dans les deux problèmes concernent l'interface et les commandes nous montrent qu'elle passe plus de temps à écrire ses programmes qu'à les corriger. En effet, les actions réalisées sur les corrections sont limitées dans les deux

problèmes. Cela semble logique puisqu'elle réalise peu d'erreurs pendant la résolution de deux problèmes.

3.1.2. La conception des programmes

De plus, nous pouvons faire l'hypothèse qu'elle est organisée en ce qui concerne l'écriture de ses programmes, puisqu'elle réalise les différentes tâches dans l'ordre. En effet, elle termine avec la programmation d'un personnage pour passer à l'autre, sans faire d'allers-retours entre les deux personnages.

Nous pouvons également supposer qu'elle est capable de savoir en avance ce que ses deux personnages feront sans avoir besoin d'exécuter leur programme, puisqu'elle arrive à mettre en place une stratégie de « planification complète », c'est-à-dire écrire l'intégralité des commandes de programmes de ses personnages avant la première exécution dans les deux problèmes. Cela nous montre que cette élève a une capacité d'abstraction, une bonne compréhension du résultat à reproduire, une bonne connaissance du sens des commandes enseignées par le scénario pédagogique et de l'exécution séquentielle des commandes sur ScratchJr.

3.1.3. La gestion de l'interface

Concernant la gestion de l'interface, nous pouvons supposer que l'élève 1 a compris la fonctionnalité des éléments d'interface, puisqu'elle arrive à les utiliser de manière opérationnelle. Nous pouvons également constater qu'elle a compris la nécessité d'initialiser la position des personnages avant d'exécuter leur programme, puisqu'elle le fait chaque fois lors de la résolution de deux problèmes de programmation. Cela se confirme aussi par le nombre égal d'actions sur l'initialisation et l'exécution dans les deux problèmes (voir Tableau 13). Un autre indice qui nous fait penser qu'elle a compris comment réaliser l'initialisation de ses personnages c'est le fait qu'elle les met la première fois à leur position initiale avec le doigt et ensuite, elle initialise leur position avec l'icône d'initialisation. De cette manière elle évite les erreurs engendrées par l'initialisation avec le doigt.

Malgré sa capacité à réaliser parfaitement le processus de l'initialisation sur ScratchJr, l'élève 1 se trompe sur le premier positionnement de ses personnages. Cela nous montre que la première étape du processus d'initialisation de la position des personnages peut poser de

problèmes même aux élèves qui ont bien compris comment réaliser l'initialisation des personnages sur ScratchJr.

3.1.4. Gestion du déplacement des personnages

En ce qui concerne le déplacement, nous pouvons supposer qu'elle a compris le fonctionnement général de ScratchJr et notamment celui des paramètres, puisqu'elle les utilise pour estimer le nombre de pas que ses personnages doivent réaliser pour se déplacer vers un endroit spécifique de la scène. Elle lui arrive aussi de commencer par utiliser plusieurs fois le même type de commande puis passer aux paramètres. Au PB2 elle utilise les paramètres et plusieurs fois le même type de commandes. Nous pouvons donc faire l'hypothèse qu'elle a compris les deux modes de déplacement présentés dans le scénario pédagogique enseigné. Ses actions nous font penser qu'elle domine bien une série des connaissances en lien avec l'orientation dans l'espace et les mathématiques, puisqu'elle arrive à reconnaître la direction suivie par le personnage dans le projet modèle et à choisir la commande de déplacement correspondante à celle-ci, à établir la correspondance entre chiffre et nombre mais aussi à en faire un usage cardinal.

3.1.5. Les messages

Nous pouvons faire l'hypothèse qu'elle s'est approprié la fonctionnalité des commandes « envoyer un message » et « quand message reçu », puisqu'elle arrive à les utiliser de manière opérationnelle sans aide. Puisque le positionnement de la commande « envoyer un message » ne lui a pas posé des difficultés, nous pouvons supposer qu'elle avait compris son sens ainsi que l'exécution séquentielle des commandes sur ScratchJr.

3.1.6. La répétition prédéfinie

De plus, nous pouvons imaginer qu'elle a compris la fonctionnalité de la commande de « répétition prédéfinie », puisqu'elle arrive à l'utiliser de manière opérationnelle pour résoudre le PB2. La création du motif des commandes et le nombre de fois que cela se répète ne lui posent pas de problème. La complexité de la nature de la commande et la distance entre son écriture et ce qu'elle fait ne lui ont pas posé de difficultés non plus.

3.1.7. La gestion du démarrage synchrone de deux personnages

Ses actions nous montrent aussi qu'elle a compris comment réaliser le démarrage synchrone de deux personnages puisqu'elle utilise de manière opérationnelle la commande « quand drapeau vert cliqué » au début du programme de deux personnages et le bouton du « drapeau-vert » en haut de la scène.

3.1.8. Les erreurs réalisées et le débogage mis en place

Vu la capacité de l'élève 1 à identifier des erreurs existantes sur son programme même avant son exécution, nous pouvons imaginer qu'elle a une bonne connaissance de l'effet que les commandes utilisées auront sur le comportement de ses personnages dans la scène.

Les erreurs existantes sur ses programmes après la première exécution aussi bien au PB1 qu'au PB2, nous montrent que l'élève 1 se trompe que sur l'évaluation de la distance réalisée par ses personnages pour arriver au bon endroit dans la scène. Nous pouvons donc comprendre que l'élève 1 n'a pas des difficultés à programmer le déplacement sur ScratchJr, mais elle a des difficultés à évaluer la distance que ses personnages doivent réaliser dans la scène. Cela ne nous étonne pas puisqu'il est difficile d'évaluer une distance surtout lorsque la scène de ScratchJr n'a pas (par défaut) de repères. Nous observons que cette difficulté ne concerne pas l'activité de programmation mais la nature de la programmation à réaliser.

En termes de débogage, nous pouvons supposer qu'elle sait où se trouve l'erreur et elle sait comment la corriger, puisqu'elle arrive à corriger les erreurs présentes dans ses programmes au PB1 avec un seul essai en réalisant ainsi un débogage « optimal » et au PB2 avec trois essais en réalisant ainsi deux débogages « non efficace » et un « efficace ». Elle a réalisé trois essais au PB2, non pas parce qu'elle ne savait pas où se trouvait l'erreur mais parce qu'elle avait utilisé deux commandes de « déplacement à droite » et il fallait corriger les deux. De plus, cela nous fait comprendre qu'elle arrive à bien mémoriser le résultat à reproduire, à le comparer avec ce qu'elle produit pour identifier les différences et ainsi arriver à réaliser ses débogages efficacement.

Au niveau des patterns de débogage, nous pouvons faire l'hypothèse qu'elle a compris que c'est plus facile pour elle de corriger les erreurs sur ces programmes lorsqu'elle procède d'abord à la correction et ensuite à l'initialisation de la position de son personnage, puisque de

cette manière elle peut prendre le temps d'étudier l'état de la scène après l'exécution de son programme, la comparer avec l'état de la scène après l'exécution du projet modèle et ainsi réaliser des corrections adaptées. Cela se confirme par le fait qu'elle utilise toujours le même pattern *Exécution-Correction-Initialisation*, pour tous les débogages réalisés pendant la résolution de deux problèmes.

3.1.9. Les aides apportées

En prenant en compte à la fois le peu nombre d'aides qu'elle a reçues et la nature de ces dernières (premier niveau d'intervention à part celle sur l'initialisation), nous pouvons supposer qu'elle a eu peu de difficultés et elle était assez indépendante lors de la résolution des deux problèmes. Compte tenu des types d'aides qu'elle a reçues, nous pouvons d'abord comprendre que le premier positionnement des personnages est difficile à réaliser pour l'élève 1 d'une part, parce qu'il lui demande beaucoup de précision afin de bien choisir le point de départ de son personnage et d'autre part, parce que la scène de ScratchJr n'a pas de repères, pour l'aider avec le positionnement. Nous pouvons aussi faire l'hypothèse que pour le débogage du déplacement, il est nécessaire d'insister au moins une fois, même les élèves qui ont compris comment réaliser cela, sur le point d'arrivée demandé dans la scène. Etant donné les trois aides qu'elle a reçues sur la « répétition prédéfinie », même si elle ne semblait pas avoir besoin autant, et le nombre d'informations dont l'élève a besoin pour la réaliser, nous pouvons comprendre qu'au moins un rappel de la consigne est nécessaire.

3.2. Comment on comprend ce que fait l'Élève 2 ?

3.2.1. Résolution de problèmes (PB1 et PB2)

Ses programmes finaux nous montrent que l'élève 2 est capable, avec peu d'aides, de résoudre deux problèmes de programmation portant entre autres sur l'utilisation opérationnelle des commandes de « messages » et de la « répétition ». Cela nous fait également penser que cette élève a compris l'exécution séquentielle des commandes sur ScratchJr, puisqu'elle l'a pris en compte dans la création de ses programmes.

En tenant compte des éléments présents dans le portrait de l'élève 2, nous pouvons supposer qu'elle a compris le fonctionnement général de la programmation sur ScratchJr et elle est à

l'aise avec la résolution de deux problèmes de programmation. Cela se confirme par sa rapidité, sa capacité à poser des questions précises et à inventer des stratégies pour être sûre de l'exactitude de sa programmation, mais aussi le fait qu'elle n'a pas de moments de blocage. De plus, ses programmes finaux respectent la consigne donnée, puisque ses personnages arrivent au bon endroit dans la scène et commencent leur programme de la même manière que dans le projet modèle.

Compte tenu du fait que la plupart des actions de l'élève 2 dans les deux problèmes concernent l'interface et les commandes, nous pouvons émettre l'hypothèse qu'elle passe plus de temps à écrire ses programmes qu'à les corriger. Cela se confirme aussi en analysant ses actigrammes, où la partie de l'alternance entre l'interface et les commandes est plus longue que la partie dédiée au débogage des erreurs présentes sur ses programmes.

3.2.2. La conception des programmes

De plus, nous pouvons dire qu'elle est organisée en ce qui concerne l'écriture de ses programmes, puisqu'elle réalise les différentes tâches dans l'ordre. En effet, elle termine avec la programmation d'un personnage pour passer à l'autre, sans faire des allers-retours entre les deux personnages.

En termes de conception de programmes, nous supposons que la plupart des fois elle est capable de savoir en avance toutes les actions que l'un de ses deux personnages fera sans avoir besoin d'exécuter son programme puisqu'elle met en place une stratégie de « planification partielle », mais elle préfère exécuter séparément une partie ou une seule commande de celui-ci en mettant en place une stratégie de « sous-programme » ou « pas à pas » afin d'être sûre qu'elle a bien programmé. Elle lui arrive une fois de savoir en avance une partie des actions que l'un de deux personnages fera sans avoir besoin d'exécuter son programme puisqu'elle met en place une stratégie de « sous-programme », mais elle préfère exécuter l'une de ses commandes en mettant en place une stratégie de « pas à pas », afin d' être sûre qu'elle a bien programmé et ensuite programmer le reste de son programme.

Ses actions nous font penser qu'elle a peut-être capté que certaines commandes sur ScratchJr sont plus susceptibles de créer d'erreurs que d'autres et c'est pourquoi elle a développé des stratégies afin de l'éviter. Cette hypothèse est confirmée par le fait qu'elle exécute séparément du reste de ses programmes les parties contenant potentiellement des erreurs et en particulier

les commandes de « déplacement » et de l' « enregistrement sonore ». Un autre indice illustrant son intention est le fait que pendant la vidéo elle dit « je vais essayer d'abord » avant d'exécuter séparément la commande du déplacement.

D'après ce qui ressort de notre corpus de données récoltées en Grèce et en France, l'élève 2 a bien fait de tester ces types de commandes indépendamment du reste, car ces dernières sont génératrices d'erreurs. Nous jugeons sa façon de procéder très efficace et cela constitue pour nous un indice fort de sa compréhension de la programmation sur ScratchJr.

La volonté de l'élève 2 à ne pas prendre des risques avec le « déplacement » se voit aussi sur le fait qu'elle a inventé une stratégie pour déplacer ses personnages en utilisant la « grille » qui lui permet de valider leur déplacement, sans exécuter leurs programmes en entier. L'efficacité de cette stratégie peut être constatée en regardant le faible nombre de corrections qu'elle réalise aux deux problèmes.

3.2.3. La gestion de l'interface

Concernant la gestion de l'interface, nous pouvons supposer que l'élève 2 a compris la fonctionnalité des éléments d'interface, puisqu'elle arrive à les utiliser de manière opérationnelle, à part l'initialisation qui lui pose problème. Nous pouvons également faire l'hypothèse qu'elle a compris la nécessité d'initialiser la position des personnages avant d'exécuter leur programme, puisqu'elle le fait chaque fois lors de la résolution de deux problèmes de programmation et la seule fois qu'elle oublie, elle s'en rend compte directement en disant « j'ai pas initialisé » et elle le fait. En analysant le nombre d'actions qu'elle réalise sur l'initialisation, nous observons qu'il n'est pas égal au nombre d'actions sur l'exécution, parce que la stratégie de déplacement qu'elle a inventé l'oblige à initialiser plus de fois la position de ses personnages avant de lancer leur programme.

Ses actions nous font également penser que, même si elle comprend la nécessité d'initialiser la position des personnages sur ScratchJr, elle n'est pas encore arrivée à la mettre en place de la manière qui lui permettrait de ne pas faire des erreurs liées au positionnement. En effet, suite au premier positionnement qui se fait obligatoirement avec le doigt, elle la réalise parfois avec le doigt et parfois avec l'icône d'initialisation et pas uniquement avec l'icône. C'est pourquoi elle se trompe deux fois sur l'initialisation de la position de ses personnages au PB2. Le fait qu'elle se trompe sur le premier positionnement de ses personnages au début de la

résolution de deux problèmes, nous montre que la première étape du processus d'initialisation de la position des personnages peut aussi poser de problèmes aux élèves, car elle demande beaucoup de précision de leur part.

3.2.4. La gestion du déplacement des personnages

En ce qui concerne le déplacement, nous pouvons supposer qu'elle est assez avancée puisqu'elle a non seulement compris comment déplacer des personnages sur ScratchJr mais elle a pu, comme on a vu plus haut, développer une stratégie en utilisant la « grille et les paramètres », qui lui permettait une vitesse et une meilleure efficacité à la résolution du problème. Puisqu'elle utilise cette stratégie pour tous les déplacements qu'elle programme aussi bien au PB1 qu'au PB2, nous pourrons être sûre que c'est le mode de déplacement qu'elle préfère au moment où nous avons réalisé l'évaluation et ce n'est pas dû au hasard. De plus, le fait qu'elle est capable de mettre en place une stratégie si complexe à cet âge-là nous montre qu'elle a appris un certain nombre des choses au niveau de la programmation. Compte tenu de sa performance, nous pouvons également déduire qu'elle domine bien une série des connaissances en lien avec l'orientation dans l'espace et les mathématiques, puisqu'elle arrive à reconnaître la direction suivie par le personnage dans le projet modèle et choisir la commande de déplacement correspondante à celle-ci, à établir la correspondance entre chiffre et nombre mais aussi à en faire un usage cardinal.

3.2.5. Les messages

La performance de l'élève 2 nous amène également à faire l'hypothèse qu'elle a compris la fonctionnalité des commandes « envoyer un message » et « quand message reçu », puisqu'elle arrive à les utiliser de manière opérationnelle sans avoir besoin d'aide. Puisque le positionnement de la commande « envoyer un message » ne lui a pas posé des difficultés, nous pouvons supposer qu'elle avait compris le sens de la commande, ainsi que l'exécution séquentielle des commandes sur ScratchJr.

3.2.6. La répétition prédéfinie

De plus, ses actions nous montrent qu'elle s'est approprié la fonctionnalité de la commande de « répétition prédéfinie », puisqu'elle arrive à l'utiliser de manière opérationnelle pour résoudre le PB2. La création du motif des commandes lui pose toutefois des difficultés

puisqu'elle se trompe sur l'ordre des commandes à répéter. La façon avec laquelle elle réagit face à cette erreur nous montre ce qu'elle est capable de faire. En effet, nous supposons qu'elle n'était pas sûre si elle avait bien programmé le motif des commandes à répéter du personnage « lapin » et c'est pourquoi elle nous a posé une question précise « il dit d'abord « bonjour le papillon ? », afin de vérifier si l'ordre des commandes était celui qu'elle avait programmé ou pas. Dans la mesure où elle pose une question précise et non pas une question du style « je ne comprends pas ce qui se passe » ou « comment faire ça ? », nous imaginons qu'elle a compris un certain nombre des choses sur la « répétition ». Sa capacité à poser une question si précise nous fait penser qu'elle a compris à la fois le fonctionnement de l'écriture de la commande de « répétition prédéfinie » et l'exécution séquentielle des commandes sur ScratchJr. Suite à l'apport de l'aide, l'élève 2 a réussi de corriger cette erreur avant même de passer à son exécution.

3.2.7. La gestion du démarrage synchrone de deux personnages

Nous pouvons également supposer qu'elle a compris comment faire démarrer le programme de deux personnages en même temps sur ScratchJr puisqu'elle arrive à le reproduire dans son projet en utilisant la commande « quand drapeau-vert cliqué » au début du programme de deux personnages et le bouton du drapeau-vert en haut de la scène.

3.2.8. Les erreurs réalisées et le débogage mis en place

Vu la capacité de l'élève 2 à identifier des erreurs existantes sur son programme même avant son exécution, nous pouvons imaginer qu'elle a une bonne connaissance de l'effet que les commandes utilisées auront sur le comportement de ses personnages dans la scène ainsi qu'une bonne connaissance du résultat à produire. Nous pouvons également faire l'hypothèse qu'elle a compris la logique de la programmation.

Les erreurs existantes sur ses programmes après la première exécution au PB2, nous montrent que l'élève 2 se trompe aussi sur l'évaluation de la distance réalisée par son personnage pour arriver au bon endroit dans la scène. Nous pouvons donc supposer qu'elle n'a pas des difficultés à programmer le déplacement sur ScratchJr, mais elle a des difficultés à évaluer la distance que ses personnages doivent réaliser dans la scène. Cela ne nous étonne pas puisque c'est difficile d'évaluer une distance surtout lorsque la scène de ScratchJr n'a pas (par défaut)

de repères. Nous observons que cette difficulté ne concerne pas l'activité de programmation mais la nature de la programmation à réaliser.

Nous pouvons également faire l'hypothèse qu'elle a compris comment déboguer ses programmes puisqu'elle arrive à corriger les erreurs présentes sur son programme. Cela se confirme aussi par le fait qu'elle réalise des corrections qui tiennent compte du résultat obtenu après l'exécution de son programme. Le fait qu'elle réagit en disant « Ah ! » ou « Non ! » lorsque le résultat qu'elle obtient suite à l'exécution de son programme n'est pas celui qu'elle aimerait avoir, nous montre qu'elle est capable de repérer rapidement les erreurs existantes sur son programme. Cela nous fait également comprendre qu'elle est capable de bien mémoriser le résultat à reproduire pour le comparer avec ce que son programme produit et identifier les différences. Nous pouvons également émettre l'hypothèse qu'elle sait comment corriger les erreurs sur le déplacement de son personnage, puisqu'elle le fait avec trois essais, en réalisant ainsi deux débogages « non efficaces » et un « efficace ».

Etant donné que nous avons repéré différentes suites d'actions et aucune de celles-ci ne se répète dans l'actigramme du PB2 de l'élève 2, nous pouvons supposer qu'elle ne suit pas la même suite d'actions chaque fois qu'elle réalise un débogage. Cela montre qu'elle n'a pas encore développé un pattern de débogage stable comme elle a fait pour la stratégie de déplacement à l'aide de la « grille et paramètres ». Compte tenu du fait que dans les suites d'actions qu'elle réalise pour le débogage nous repérons que l'action d'initialisation arrive après les actions de correction, nous pouvons supposer que la façon avec laquelle elle procède pour ses débogages est assez efficace.

3.2.9. Les aides apportées

En prenant en compte à la fois le peu nombre d'aides qu'elle a reçues et la nature de ces dernières (premier niveau d'intervention à part celle sur l'initialisation), nous pouvons supposer qu'elle a eu peu de difficultés face à la résolution des deux problèmes. Compte tenu des types d'aides qu'elle a reçues, nous pouvons d'abord comprendre que le premier positionnement des personnages et l'initialisation de leur position sont difficiles à réaliser pour l'élève 2. Nous pouvons réfléchir à deux raisons expliquant cette difficulté : d'une part, parce qu'ils nécessitent beaucoup de précision de la part de l'élève, afin de bien choisir le point de départ de son personnage et d'autre part, parce que la scène de ScratchJr n'a pas de

repères, pour l'aider avec le positionnement. Nous pouvons aussi faire l'hypothèse que pour le débogage du déplacement il est nécessaire d'assister au moins une fois, même les élèves qui ont compris comment réaliser cela, sur le point d'arrivée demandé dans la scène. Etant donné les quatre aides qu'elle a reçues sur la « répétition prédéfinie » et le nombre d'informations dont l'élève a besoin pour la réaliser, nous pouvons comprendre que des rappels de la consigne sont nécessaires.

3.3. Comment comprendre ce que fait l'Élève 3 ?

3.3.1. Résolution de problèmes (PB1 et PB2)

Vu les éléments présents dans le portrait de l'élève 3, nous pouvons supposer qu'il a beaucoup de difficultés face à la résolution de deux problèmes de programmation. Cela se confirme par le nombre de moments de blocage que nous avons noté, le nombre d'aides qu'il a reçu et le nombre d'actions qu'il a réalisées. Bien que ses programmes finaux soient corrects, l'élève n'a toutefois pas réussi à surmonter sa difficulté avec l'initialisation de la position du personnage « enfant », puisque jusqu'à la fin de sa programmation sa position initiale reste erronée.

Compte tenu du fait que la plupart des actions de l'élève 3 au PB1 concernent l'interface et les corrections, nous pouvons supposer qu'il passe plus de temps à corriger ses programmes qu'à les écrire, alors qu'au PB2 où la plupart des actions de l'élève 3 concernent l'interface et les commandes, nous pouvons supposer qu'il passe plus de temps à écrire ses programmes qu'à les corriger. Cela se confirme aussi en analysant ses actigrammes, où la partie de l'alternance entre l'interface et les commandes est plus longue au PB2 qu'au PB1 et la partie de l'alternance entre l'interface et les corrections est plus longue au PB1 qu'au PB2. Cela ne nous semble pas étonnant si on réfléchit au fait qu'au PB2 l'élève 3 a des difficultés à corriger les erreurs présentes sur ses programmes.

3.3.2. La conception des programmes

Nous pouvons également supposer qu'il n'a pas encore une façon stable de procéder pour créer ses programmes, puisque dans le PB1 il réalise plusieurs allers-retours entre les deux personnages en écrivant chaque fois un bout de leur programme, alors qu'au PB1 il écrit

d'abord le programme de l'un personnage et ensuite il fait la même chose pour l'autre. Nous pouvons faire l'hypothèse qu'au moment de l'évaluation cet élève était entre ces deux modes de fonctionnement pour l'écriture de ses programmes.

En termes de conception de programmes, nous supposons qu'au PB1 l'élève 3 sait en avance une partie des actions que ses deux personnages feront sans avoir besoin d'exécuter leur programme, puisqu'il met en place une stratégie de « sous-programme ». Pour le reste de leurs programmes nous supposons qu'il a besoin de procéder action par action en mettant en place plusieurs fois une stratégie de « pas à pas ». En revanche au PB2, nous supposons qu'il sait en avance toutes les actions que ses deux personnages feront sans avoir besoin d'exécuter leur programme, puisqu'il est capable de mettre en place une stratégie de « planification complète ». Cela se confirme aussi à travers ses actigrammes ou la partie de l'alternance entre les commandes et l'interface s'arrête très tôt alors qu'au PB2 elle dure plus longtemps.

3.3.3. La gestion de l'interface

Concernant la gestion de l'interface, nous pouvons supposer que l'élève 3 s'est approprié la fonctionnalité des éléments d'interface, puisqu'il arrive à les utiliser de manière opérationnelle, à part l'initialisation qui lui pose beaucoup de difficultés surtout au PB1. Plusieurs éléments illustrent ses difficultés au PB1 : le fait que la position initiale de son personnage reste erronée pendant 11 exécutions, le fait qu'il a eu un blocage lié à l'initialisation de la position de son personnage, le fait qu'il a été assisté quatre fois sur l'initialisation dont les trois nous sommes intervenue pour corriger la position initiale de son personnage, ainsi que le fait qu'il continue à se tromper sur l'initialisation même après l'apport de l'aide. Au PB2, il se trompe à nouveau deux fois sur l'initialisation et après l'apport de deux aides il arrive à corriger son erreur. Malgré ces difficultés, nous observons qu'il pense à initialiser la position de ses personnages pendant la résolution de deux problèmes. Cela nous amène à supposer qu'il a compris qu'il est nécessaire de le faire.

Une raison de ces difficultés pourrait être le fait qu'après le premier positionnement des personnages qui se fait obligatoirement avec le doigt, l'élève 3 privilégie l'utilisation de son doigt pour l'initialisation plutôt que l'icône d'initialisation. Comme nous avons déjà vu pendant l'étude préparatoire en Grèce le risque de l'initialisation avec le doigt est de ne pas remettre le personnage chaque fois au même endroit. Cela pose ensuite problème à la

programmation du déplacement des personnages, car il affecte le nombre des pas dont ces derniers auront besoin de réaliser pour arriver au bon endroit dans la scène. Cela arrive plusieurs fois à l'élève 3 pendant l'évaluation.

Une autre raison expliquant ces difficultés est le fait que l'élève 3 réalise l'initialisation rapidement après l'exécution du programme de son personnage, sans faire attention où il le pose dans la scène avec son doigt. En effet, même quand nous essayons de l'aider en lui présentant le projet modèle nous observons qu'il ne fait pas attention au positionnement des personnages. Nous pouvons donc supposer que l'élève 3 n'a pas bien compris que le positionnement des personnages dans la scène influence énormément aussi bien la programmation du déplacement des personnages que la vérification de la programmation réalisée. L'initialisation de la position des personnages n'est pas facile à comprendre surtout parce qu'elle ne rassemble pas à aucune des activités auxquels ils sont habitués à réaliser à l'école.

3.3.4. La gestion du déplacement des personnages

En ce qui concerne le déplacement de ses personnages, nous pouvons supposer qu'il a compris la fonctionnalité des commandes et celle des paramètres, puisqu'il les utilise de manière opérationnelle lors de la résolution de deux problèmes. L'élève 3 est également capable de mettre en place une stratégie de déplacement plus efficace que celle de la simple utilisation des paramètres, puisqu'il commence en utilisant « plusieurs fois la même commande et ensuite il réduit son écriture » en utilisant une seule commande et en appuyant sur celle-ci plusieurs fois pour trouver le nombre de pas qu'il lui faut pour réaliser le déplacement de son personnage à l'aide des paramètres. Puisqu'il utilise cette stratégie pour un seul déplacement alors que pour tous les autres il utilise toujours les « paramètres », nous pouvons supposer qu'il venait de la découvrir et il n'avait pas encore remplacé la stratégie précédemment utilisée. Compte tenu du fait qu'il utilise les paramètres, nous pouvons faire l'hypothèse qu'il domine bien certaines notions mathématiques, car il arrive à établir la correspondance entre chiffre et nombre mais aussi à en faire un usage cardinal.

L'élève 3 se trompe une fois sur la direction du déplacement et plusieurs fois sur l'évaluation de la distance réalisée par ses personnages pour arriver au bon endroit dans la scène. Puisqu'il se trompe une seule fois sur la direction et il arrive à corriger cette erreur rapidement, nous

pouvons imaginer que c'est plus une erreur liée à un manque d'attention au problème donné qu'une erreur liée à une difficulté sur l'orientation dans l'espace. Vu les erreurs réalisées sur l'évaluation de la distance réalisée par ses personnages, nous pouvons supposer qu'il n'a pas des difficultés à comprendre la fonctionnalité des commandes de « déplacement » ou à les utiliser, mais à estimer le nombre de pas que ses personnages doivent réaliser pour arriver au bon endroit dans la scène. En effet, il est difficile d'estimer une distance et notamment sur ScratchJr où la scène n'a pas (par défaut) des repères. Cette difficulté ne concerne pas l'activité de programmation mais la nature de la programmation à réaliser.

Ses actions suite à la réalisation des erreurs sur l'évaluation de la distance réalisée par ses personnages nous font comprendre que l'élève 3 a des difficultés avec la programmation du déplacement, car il a besoin de beaucoup de temps, beaucoup d'essais de débogage et beaucoup d'aides, pour les corriger. Nous pouvons remarquer plusieurs éléments qui se trouvent à l'origine de ces difficultés.

Tout d'abord, nous pouvons repérer les difficultés de l'élève 3 face la réalisation de l'initialisation de la position des personnages. Comme nous l'avons expliqué plus haut l'initialisation est très importante pour la programmation d'un déplacement sur ScratchJr, car une mauvaise position initiale peut influencer aussi bien l'évaluation de la distance à réaliser pour que le personnage arrive au bon endroit dans la scène que la vérification de l'exactitude de la programmation réalisée.

Une autre raison expliquant son problème avec la programmation du « déplacement » est les difficultés qu'il a avec le processus de débogage. Nous supposons que l'élève 3 a des difficultés en particulier avec la première phase de ce processus : l'évaluation du programme. En effet, l'élève exécute ses programmes, mais il ne pense pas toujours à comparer le résultat obtenu après l'exécution avec le résultat souhaité. Il a besoin d'être assisté sur la vérification de sa programmation plusieurs fois aussi bien au PB1 qu'au PB2. De plus, nous observons que suite à l'exécution du programme il procède rapidement à l'initialisation de la position du personnage dont le programme il venait d'exécuter et par conséquent, il ne peut pas comparer le résultat obtenu avec le résultat souhaité puisqu'il n'a plus accès au résultat obtenu après l'exécution. Une autre raison pour laquelle nous supposons qu'il a des difficultés avec la programmation du déplacement c'est le mode de programmation qu'il utilise surtout au PB1, car en réalisant plusieurs allers-retours entre les deux personnages il oublie le résultat obtenu

suite à l'exécution et ainsi il lui prend plus de temps à corriger les erreurs sur l'évaluation de distance. C'est pourquoi, il a plus de difficultés à corriger les erreurs de déplacement au PB1 qu'au PB2.

3.3.5. Les messages

La performance de l'élève 3 aux deux problèmes en ce qui concerne les commandes de « messages » est telle que nous ne pouvons pas être sûre s'il a compris ou pas leur fonctionnalité, puisqu'il les utilise de manière opérationnelle au PB1, mais il essaie de les utiliser au PB2, où il n'en avait pas besoin. En effet, il hésite d'utiliser la commande « envoyer un message » au programme du personnage « lapin » pour l'effacer après, une fois qu'il a attendu la consigne du problème. Il fait la même chose pour le personnage « papillon » avec la commande « quand message reçu », mais cette fois-ci il la laisse au début de son programme en réalisant une erreur sur le démarrage synchrone de deux personnages. Malgré l'apport des nombreuses aides de notre part, il n'arrive pas à identifier cette erreur et il se trouve face à plusieurs moments de blocage pendant lesquels il ne procède pas à des corrections. Suite à un long moment de silence et une série d'actions sur la tablette n'ayant aucun impact sur son programme, il essaye de corriger cette erreur en utilisant la commande de « répétition ». Suite à notre intervention il efface cette commande et ensuite, il corrige l'erreur en utilisant la commande « quand drapeau-vert cliqué » à la place de la commande « quand message reçu ».

Ses actions nous font toutefois penser qu'il n'a pas corrigé l'erreur parce qu'il a compris pourquoi cela posait problème à son programme, mais par déduction, puisque c'était la dernière partie du programme qui pourrait contenir l'erreur. Notre hypothèse se confirme aussi par les réponses qu'il nous donne pendant la discussion réalisée après la séance d'évaluation, car, d'une part, il confond toujours le sens de la commande « quand message reçu » avec celui de la commande « quand drapeau-vert cliqué » et d'autre part, il explique que l'un personnage commence son programme au PB1 parce que l'autre lui a dit, mais il ne se réfère pas à l'envoi et la réception du message.

Nous pouvons faire plusieurs hypothèses pour expliquer ses actions. Une raison pourrait être le fait qu'il a confondu les commandes « quand drapeau-vert cliqué » et « quand message reçu » et il n'avait pas fait attention à son programme pour repérer son erreur. Nous pouvons

également supposer que le grand nombre d'aides que nous lui avons apporté, l'ont angoissé et ainsi l'ont bloqué, au lieu de l'aider. Une autre raison expliquant ses actions pourrait être les difficultés qu'il semble avoir avec le processus de débogage et en particulier, avec la vérification de sa programmation. En effet, lorsqu'on l'invite à comparer son programme avec le projet modèle en lui posant même des questions pour l'aider à identifier son erreur, soit il ne répond pas, soit il ne répond pas correctement. Il est également possible qu'il s'est approprié partiellement la fonctionnalité des commandes de « messages », puisqu'il n'arrive pas à bien saisir la notion de l'événement, directement liée avec celles-ci. En effet, la commande « envoyer un message » sert à envoyer un message de couleur spécifique et cet envoi est un événement provoquant le démarrage du personnage recevant ce message à l'aide de la commande « quand message reçu ». L'élève 3, en tant que l'élève de cours préparatoire, n'est pas suffisamment familiarisé avec la notion de l'événement. Nous supposons que l'absence d'un lien visible entre la commande « envoyer un message » et la réaction du personnage « expéditeur » dans la scène rend la compréhension de la fonctionnalité de la commande et ainsi la compréhension de l'existence d'un événement plus difficile pour l'élève. En revanche, l'effet de l'envoi du message ou de l'événement est plus visible sur le comportement du personnage « destinataire » et cela pourrait aussi poser problème à l'élève.

En ce qui concerne la correction qu'il réalise en utilisant la commande de « répétition », nous pouvons supposer qu'il se souvenait du fait que cette commande a aussi été enseignée dans le scénario et puisqu'il ne l'avait pas utilisée dans ses programmes jusqu'à ce moment-là, il a pensé que s'il utilisait le problème serait résolu. Nous pouvons aussi faire l'hypothèse qu'il a réalisé cette correction pour nous faire plaisir. Puisque cette correction ne tient pas compte du résultat obtenu suite à l'exécution de son programme, nous pouvons faire l'hypothèse qu'en réalisant cette correction il n'avait pas étudié son programme et il ne l'avait pas comparé avec le résultat souhaité présenté dans le projet modèle. Ces actions nous montrent également que l'élève 3 n'avait peut-être pas compris la fonctionnalité de la commande de « répétition prédéfinie », puisqu'il l'a utilisé dans un contexte pas approprié.

3.3.6. La répétition prédéfinie

En ce qui concerne la commande de « répétition prédéfinie », les actions de l'élève 3 nous font penser, dans un premier temps, qu'il n'a pas compris sa fonctionnalité, puisqu'il choisit de ne pas l'utiliser pour résoudre le PB2. En effet, il a réussi de reproduire le résultat souhaité

en créant trois fois le motif des commandes qui se répète par le personnage « lapin », au lieu de créer un seul motif et utiliser la « répétition prédéfinie » pour le répéter le nombre de fois qu'il fallait. Il est important de signaler que le choix réalisé par l'élève 3 est aussi le choix de la majorité des élèves de deux classes de CP et de GS2.

Nous pouvons donc réfléchir autour du pourquoi l'élève 3 a fait ce choix. Il est possible que l'élève 3 ait eu des difficultés à comprendre l'organisation de l'écriture de la commande de « répétition prédéfinie » et a préféré éviter son utilisation. Ses difficultés peuvent être dues à la distance existante entre l'écriture de la commande « répétition prédéfinie » et ce qu'elle fait. Nous pouvons donc faire l'hypothèse que ce n'est pas l'action de répétition qui lui pose problème, puisqu'il arrive à répéter le motif des commandes demandé sans la « répétition prédéfinie », mais la forme d'écriture de celle-ci à l'aide de la commande fournie par ScratchJr.

La commande de « répétition prédéfinie » sert d'une certaine manière à raccourcir une programmation qui pourrait sinon être longue. Le choix de l'élève 3 à créer trois fois le motif des commandes au lieu d'utiliser la « répétition prédéfinie » nous montre aussi que ce dernier a besoin de voir la version longue de la programmation, c'est-à-dire le motif qui est répété et le nombre des fois qui est répété. De cette façon, il arrive peut-être à mieux gérer ce que fait son personnage.

De plus, les discussions que nous avons eues avec les élèves de CP lors de la séance d'évaluation, nous ont fait part d'une autre hypothèse qui pourrait expliquer le choix de l'élève 3 et de la majorité des élèves de notre corpus en lien avec la commande de « répétition prédéfinie ». En effet, il est possible qu'il n'utilise pas la « répétition », d'une part, parce qu'il ne l'aime pas puisqu'il ne lui permet pas de programmer autant qu'il veut et d'autre part, parce qu'il l'a simplement oubliée.

3.3.7. La gestion du démarrage synchrone de deux personnages

En ce qui concerne le démarrage synchrone de deux personnages, nous pouvons supposer qu'il a partiellement compris comment le réaliser, puisqu'il se trompe sur cela, comme on l'a vu plus haut, pendant un long moment de l'enregistrement vidéo mais il arrive à surmonter cette difficulté grâce aux nombreuses aides qu'il a reçues de notre part. En effet, au lieu d'utiliser la commande « quand drapeau-vert cliqué » au début du programme de ses deux

personnages, il utilise la commande « quand drapeau-vert cliqué » au début du programme de l'un de deux personnages et la commande « quand message reçu » au début du programme de l'autre.

Compte tenu du fait qu'il utilise de manière opérationnelle la commande « quand drapeau-vert cliqué » au début du programme du personnage « papillon » au PB1 et du « lapin » au PB2, nous pouvons supposer qu'il a bien saisi le démarrage du programme d'un personnage avec le bouton du « drapeau-vert » en haut de la scène. Nous pouvons ainsi faire l'hypothèse que les difficultés qu'il a eues au PB2 à réaliser le démarrage synchrone de ses deux personnages sont potentiellement dues au fait qu'il confond le sens de la commande « quand drapeau-vert cliqué » avec celui de la commande « quand message reçu ». Cela se confirme par la réponse qu'il nous donne après la résolution du PB2, puisqu'il confond toujours le sens de la commande « quand message reçu » avec celui de la commande « quand drapeau-vert cliqué ». Nous avons déjà expliqué auparavant cette erreur dans la section 3.2.5. dans ce chapitre.

3.3.8. Les erreurs réalisées et le débogage mis en place

En analysant le comportement de l'élève 3 au PB1, nous pouvons supposer qu'il sait comment procéder pour corriger les erreurs qu'il réalise, même s'il n'arrive pas à les identifier rapidement suite à l'exécution de ses programmes. Au PB2, nous pouvons faire l'hypothèse que soit il n'arrive pas à comprendre où se trouvent les erreurs, soit il ne sait pas comment procéder pour les résoudre, soit il est simplement fatigué puisque c'est le deuxième problème qu'il a résolu.

Nous avons déjà repéré plus haut que l'élève 3 a des difficultés avec le processus de débogage. Plusieurs éléments nous amènent à ce constat. En effet, l'élève n'arrive pas à identifier toutes les erreurs existantes sur son programme suite à son exécution. Il se concentre chaque fois à la correction d'une seule erreur. Il prend plus de temps (12 exécutions au PB1 et 6 exécutions au PB2) et il a besoin de réaliser beaucoup de débogages « non efficaces » pour arriver au débogage « efficace ». Il a reçu sept fois de l'aide sur la vérification de sa programmation au cours de la résolution de deux problèmes.

Nous pouvons donc supposer qu'il a un peu des difficultés avec le débogage car il ne fait pas très attention quand il met en place la première phase de ce processus : l'évaluation de son programme. En effet, nous avons observé aussi bien pendant la correction des erreurs sur

l'évaluation de la distance réalisée par ses personnages dans les deux problèmes que pendant la correction des erreurs sur le démarrage du personnage « papillon » au PB1, que l'élève 3 exécutait ses programmes, mais lorsqu'il comparait le résultat obtenu suite à l'exécution avec le résultat souhaité il le faisait rapidement sans faire très attention, ni à son programme, ni au projet modèle. Cela explique le nombre d'aides qu'il a reçu sur la vérification de sa programmation ainsi que le temps et le nombre d'essais qu'il a besoin de réaliser pour corriger les erreurs existantes sur ses programmes, au cours de la résolution de deux problèmes.

Une autre hypothèse expliquant ses difficultés avec le débogage est le choix de l'élève 3 à initialiser rapidement la position de ses personnages, car de cette façon il ne peut pas réaliser efficacement des débogages, puisqu'il n'a plus accès à l'état de sa scène suite à l'exécution de la dernière version de son programme. Nous pouvons faire l'hypothèse que cet élève a compris qu'il fallait tester son programme, puisqu'il le fait chaque fois après avoir fini avec ce qu'il voulait programmer, mais il n'a pas totalement compris comment il devrait s'y prendre suite à l'exécution pour déboguer de façon efficace ses programmes. Il est également possible qu'il a des difficultés avec le débogage parce qu'en étant en CP il commence à résoudre des problèmes, mais il n'a pas encore l'habitude de vérifier l'exactitude de ce qu'il a produit par rapport à ce qu'il a été demandé.

En analysant les patterns de débogage identifiés sur les actigrammes de l'élève 3, nous pouvons remarquer deux types : celui où l'action d'initialisation arrive avant la ou les corrections (3 fois au PB1) et celui où l'action d'initialisation arrive après la ou les corrections (1 fois au PB1 et 2 fois au PB2). Le premier est un pattern qui rend plus facile la réalisation des débogages efficaces que le deuxième, puisque l'action d'initialisation arrive après la ou les actions de correction et ainsi l'élève a plus de temps à étudier l'état de la scène suite à l'exécution de son programme que lorsque l'action d'initialisation arrive juste après celle de l'exécution. Nous pouvons donc supposer qu'au moment de l'évaluation l'élève 3 se trouve entre un pattern de débogage plus efficace et un pattern de débogage moins efficace. Nous pouvons faire l'hypothèse qu'il est dans une phase intermédiaire. Il est probablement en train de découvrir une nouvelle façon de faire le débogage, mais il n'a pas encore privilégié celle-ci au lieu de l'autre. Nous ne pouvons malheureusement pas savoir lequel de deux patterns est arrivé plus récemment dans ses actions, puisqu'on n'a pas accès à cette information.

Puisque la majorité des corrections qu'il réalise aux deux problèmes tiennent compte du résultat obtenu suite à l'exécution de ses programmes, nous pouvons imaginer que, même s'il a besoin de prendre plus de temps et de réaliser plusieurs débogages pour corriger les erreurs présentes sur ses programmes, il a compris le principe du débogage des programmes sur ScratchJr.

3.3.9. Les aides apportées

Vu le nombre (10 aides) et la nature des aides (consigne, lancement projet, vérification, initialisation, effacement commande) qu'il a reçu au PB1, nous pouvons supposer qu'il a eu une série des difficultés face à la résolution de ce problème, mais en général il a pu les surmonter. En nous focalisant sur le type d'aides qu'il a reçu au PB1, nous pouvons comprendre que les éléments qui lui posent des difficultés au PB1 étaient le premier positionnement de ses personnages, l'initialisation de la position du personnage « enfant » et le débogage du déplacement de ses personnages. Nous avons présenté plus haut nos hypothèses expliquant ces difficultés. Nous pouvons ainsi faire l'hypothèse que l'élève 3 a réussi avec notre soutien à surmonter les difficultés rencontrées et à résoudre le PB1.

En prenant en compte le nombre (19 aides) et la nature des aides (consigne, lancement projet, vérification, identification de l'erreur, initialisation, effacement commande) qu'il a reçues au PB2, nous pouvons supposer qu'il a beaucoup de difficultés à le résoudre. En analysant les types d'aides qu'il a reçus, nous pouvons supposer qu'il a des difficultés avec le positionnement de ses personnages, le démarrage du personnage « papillon », le débogage du déplacement du personnage « papillon » et l'initialisation de la position du personnage. Les éléments qui lui posent la plupart des difficultés sont le démarrage du personnage « papillon » (12 aides) et le débogage du déplacement du personnage « papillon » (5 aides), puisqu'il reçoit le plus d'aides là-dessus. Nous avons essayé d'expliquer plus haut ces difficultés.

3.4. Comment comprendre ce que fait l'Élève 4 ?

3.4.1. Résolution de problèmes (PB1 et PB2)

Vu les éléments présents dans le portrait de l'élève 4, nous pouvons supposer qu'il a peu des difficultés face à la résolution du PB1, car il arrive avec une série d'aides (8 aides) de notre

part à les surmonter et ainsi à le résoudre. Nous pouvons aussi faire l'hypothèse qu'il a plus de difficultés face à la résolution du PB2, puisque, malgré les nombreuses aides (28 aides) que nous lui avons apportées, il n'arrive pas à le résoudre. Cela se confirme aussi par le fait qu'au PB1 nous n'avons pas repéré des moments de blocage, alors qu'au PB2 nous en avons remarqué plusieurs (8). De plus, il a eu besoin de plus de temps, plus d'actions et plus d'aides pour résoudre le PB2, que pour résoudre le PB1.

Un autre indice confirmant qu'il est plus à l'aise au PB1 qu'au PB2 est le type des catégories qui sont les plus concernées par ses actions. Puisqu'au PB1 il réalise plus d'actions sur l'interface et les commandes, nous pouvons supposer qu'il passe plus de temps à écrire ses programmes qu'à les corriger. En revanche, au PB2 où il réalise plus d'actions sur l'interface et les corrections, nous pouvons faire l'hypothèse qu'il passe plus de temps à corriger ses programmes qu'à les écrire. Cela se voit aussi sur ses actigrammes où au PB1 la partie de l'alternance entre l'interface et les commandes est plus longue que la partie de l'alternance entre l'interface et les corrections alors qu'au PB2 nous repérons le contraire.

Un autre indice est le fait qu'au PB1 il a une progression fluide dans sa programmation, sans hésitation, alors qu'au PB2 il semble moins sûr de ce qu'il fait notamment en ce qui concerne la programmation du personnage « lapin ». Parallèlement, il ne semble pas avoir de problèmes avec la programmation du « papillon ».

Nous pouvons ainsi émettre l'hypothèse que l'élève 4 a plus des difficultés à résoudre le PB2, car il a des difficultés avec la « répétition » du motif des commandes. Nous allons présenter un peu plus tard dans cette section des indices confirmant cela ainsi que nos hypothèses expliquant ces difficultés.

3.4.2. La conception des programmes

Nous pouvons également constater qu'il n'a pas encore une façon stable de procéder pour écrire les programmes de ses personnages. En effet, au PB1 il est plus organisé, car il programme et corrige le programme du premier personnage et ensuite il s'occupe du programme du deuxième, alors qu'au PB2 il ne procède pas de la même manière, car il réalise plusieurs allers-retours entre les deux personnages.

En termes de conception de programmes, nous pouvons comprendre qu'au moment de l'évaluation l'élève 4 se trouve dans une phase où il privilégie les stratégies de « sous-programme » et de « pas à pas », mais parallèlement il est capable de mettre en place une stratégie de « Recommencer au début » suivie par une stratégie de « planification partielle ». En effet, il utilise les stratégies de « sous-programme » et de « pas à pas » pour écrire les programmes de ses personnages au PB1 et il utilise à nouveau la stratégie « sous-programme » pour créer les programmes de ses personnages au PB2. Pour l'un de ses personnages au PB2, il efface tout son programme et il recommence à le programmer en écrivant cette fois-ci l'intégralité de ses commandes sans l'exécuter.

Ses actions nous montrent qu'il a plutôt une bonne connaissance de ce qu'ils font la plupart des commandes enseignées et il comprend le principe d'exécution séquentielle de celles-ci sur ScratchJr. Nous pouvons également faire l'hypothèse qu'il a une assez bonne capacité d'abstraction, puisqu'il arrive à imaginer chaque fois une partie des actions de son personnage sans avoir besoin d'exécuter son programme. Dans la mesure où il arrive aussi une fois à imaginer l'intégralité des actions de l'un de ses personnages avant l'exécution, nous pouvons supposer qu'au moment de l'évaluation il était en train de progresser vers une bonne capacité d'abstraction. Nous pouvons également interpréter son choix, d'effacer le programme entier du personnage « lapin » puisqu'il n'arrivait pas à le déboguer, en tant que très efficace, car il lui a permis de produire le programme le plus proche au résultat souhaite de toute son activité de programmation au PB2.

3.4.3. La gestion de l'interface

En ce qui concerne la gestion de l'interface, nous pouvons remarquer que l'élève 4 s'est approprié la fonctionnalité des éléments d'interface, puisqu'il les utilise de manière opérationnelle pour résoudre les deux problèmes de programmation, à part l'initialisation qui lui pose parfois des difficultés. En effet, il se trompe une fois sur l'initialisation de son personnage au PB1 et huit fois sur l'initialisation de ses personnages au PB2. Il a également eu besoin d'une aide (initialisation) au PB1 et trois aides (consigne, initialisation, initialisation) au PB2. Au vu de ses actions, nous pouvons supposer qu'il a compris la nécessite d'initialiser la position de ses personnages avant d'exécuter leur programme, puisqu'il le fait avant chaque exécution au PB1 et la plupart des fois au PB2.

Une raison de ces difficultés pourrait être la façon avec laquelle l'élève 4 initialise la position de ses personnages. En effet, nous avons remarqué qu'il n'est pas encore arrivé à la mettre en place de la manière qui lui permettrait de ne pas faire des erreurs liées au positionnement, car suite au premier positionnement qui doit être fait obligatoirement avec le doigt, il continue à la réaliser en utilisant son doigt et pas uniquement avec l'icône d'initialisation, pour les deux problèmes. Le risque existant lorsque les élèves initialisent la position de leur personnage avec le doigt est de ne pas le remettre chaque fois au même début. Cela pourrait ensuite poser problème à la programmation du déplacement de ce personnage, car la position initiale influence le nombre des pas dont ce dernier a besoin de réaliser pour arriver au bon endroit dans la scène selon le problème donné.

Une autre raison expliquant ces difficultés avec l'initialisation au PB2 est le fait qu'il réalise deux fois l'exécution de son programme avec le bouton du drapeau-vert en haut de la scène et cette pratique lui permet d'éviter l'étape d'initialisation. Cela pourrait confondre l'élève 4 qui jusqu'ici était habitué à initialiser la position de son personnage avec le doigt avant l'exécution et ainsi l'amener à ne pas la réaliser systématiquement. Nous considérons en tant que limite du logiciel le fait que les deux exécutions possibles ont cette subtile mais très importante différence. À titre de rappel, pour une exécution d'un programme avec le doigt nous avons besoin d'initialiser la position du personnage avant, alors que pour une exécution d'un programme avec le bouton en haut de la scène, nous n'en avons pas besoin d'initialiser la position du personnage.

Nous pouvons également supposer qu'à l'origine de ces difficultés se trouve la particularité de l'initialisation en tant qu'action et le fait qu'elle influence aussi bien la programmation d'un déplacement que la vérification d'une programmation réalisée. Cela peut paraître difficile à comprendre même par les élèves de cours préparatoire. De plus, l'initialisation de la position des personnages n'est pas familière en cours préparatoire, puisqu'elle ne ressemble à aucune des activités qu'ils sont habitués à réaliser à l'école.

3.4.4. La gestion du déplacement des personnages

En ce qui concerne le déplacement de ses personnages, nous pouvons constater que l'élève 4 s'est approprié la fonctionnalité des commandes de « déplacement », puisqu'il les utilise de manière opérationnelle lors de la résolution de deux problèmes. Compte tenu du fait qu'il

utilise les « paramètres » pour déplacer ses personnages, nous pouvons supposer qu'il domine bien une série des connaissances mathématiques. En effet, il arrive à établir la correspondance entre chiffre et nombre, mais aussi à comprendre son usage cardinal. Nous pouvons également faire l'hypothèse qu'il a des connaissances en lien avec l'orientation dans l'espace puisqu'il arrive à différencier la gauche, la droite, le dessus et le dessous et à associer la flèche existante sur les commandes de déplacement avec la direction indiquée. Ses actions nous font comprendre qu'il est également capable d'identifier la commande de déplacement qui va permettre à son personnage de se déplacer pour reproduire le déplacement demandé par la consigne du problème donné.

Puisque les erreurs sur le déplacement de l'élève 4 concernent le nombre de pas que ses personnages doivent réaliser pour arriver sur l'endroit demandé dans la scène, nous pouvons comprendre qu'il n'a pas des difficultés à saisir la fonctionnalité des commandes de déplacement ou à les utiliser de manière opérationnelle, mais il a plutôt des difficultés avec l'évaluation d'une distance. En effet, il est difficile d'estimer une distance surtout lorsque la scène de ScratchJr n'a pas par défaut des repères qui pourraient l'aider. Il s'agit ainsi d'une difficulté qui ne concerne pas l'activité de programmation mais la nature de la programmation à réaliser, ici le déplacement. Nous pouvons également constater que l'élève 4 a des difficultés à corriger ce type d'erreurs aussi bien au PB1 qu'au PB2, puisqu'il a besoin du temps, des débogages et des aides pour le faire. Nous avons observé qu'il a plus des difficultés à corriger ce type d'erreurs au PB2 qu'au PB1. En effet, il prend plus de temps pour le faire au PB2 que ce qu'il prend au PB1, il réalise plus de débogages « non efficaces » avant d'arriver au débogage « efficace » au PB2 qu'au PB1 et il a aussi besoin de plus d'aides de notre part au PB2 que celles dont il a besoin au PB1.

Nous supposons que ces difficultés face à la correction des erreurs sur le déplacement, sont dues aux difficultés qu'il a avec le processus du débogage et en particulier avec la première étape de ce processus : l'évaluation du programme. Nous sommes arrivée à cette hypothèse pour plusieurs raisons. Tout d'abord, en raison des types d'aides qu'il reçoit aussi bien au PB1 qu'au PB2. En effet, l'élève 4 reçoit principalement des aides de type « vérification de son programme » et « identification de l'erreur ». Nous avons également fait cette hypothèse puisque, notamment au PB2, il réalise une série des corrections qui ne tiennent pas compte du résultat obtenu suite à l'exécution. Cela nous montre qu'il ne vérifie pas attentivement son

programme lorsqu'il exécute et il ne le compare pas suffisamment avec le résultat souhaité. C'est peut être la raison pour laquelle il ne réalise pas des corrections adaptées.

Une autre raison expliquant les difficultés de l'élève 4 avec la correction d'erreurs sur le déplacement notamment au PB2 pourrait être les erreurs qu'il réalise sur l'initialisation de la position de ses personnages. En effet, nous avons observé qu'il se trompe six fois sur l'initialisation de la position du personnage « papillon ». Comme nous l'avons expliqué plus haut l'initialisation est très importante pour la programmation d'un déplacement sur ScratchJr, car une mauvaise position initiale peut influencer aussi bien l'évaluation de la distance à réaliser pour que le personnage arrive au bon endroit dans la scène, que la vérification de l'exactitude de la programmation réalisée.

Nous pouvons également réfléchir autour du pourquoi l'élève 4 a plus des difficultés à corriger les erreurs sur le déplacement au PB2 qu'au PB1. Il est possible que le PB2 lui paraisse en général plus difficile que le PB1, potentiellement car il implique la réalisation d'une répétition. Il est également possible qu'il arrive au PB2 en étant fatigué par la résolution du PB1. Une autre raison expliquant sa performance pourrait être le fait qu'il était déjà pendant la résolution du PB2 préoccupé par la correction de l'erreur sur le motif à répéter. Compte tenu de ses actions dans l'enregistrement vidéo, nous supposons qu'il était plus intéressé par la correction de l'erreur sur le motif à répéter, qu'il s'occupait rapidement de la correction de l'erreur de déplacement pour revenir à la correction de celle qu'il intéressait.

3.4.5. Les messages

En ce qui concerne les commandes de « messages », nous pouvons supposer que l'élève 4 a compris leur fonctionnalité puisqu'il arrive à les utiliser de manière opérationnelle pour résoudre le PB1. Nous pouvons ainsi faire l'hypothèse qu'il a à la fois compris que la commande « envoyer un message » sert à envoyer un message de couleur spécifique et que le programme du personnage « destinataire » démarre au moment où le message est envoyé par le personnage « expéditeur ». Puisqu'il arrive à bien positionner la commande « envoyer un message » dans le programme du personnage « papillon », nous pouvons aussi supposer qu'il a réussi, soit en décodant la consigne du problème donné, soit en faisant attention au lancement du projet modèle, à repérer à quel moment de l'exécution du programme de « l'expéditeur », « le destinataire » devait commencer son programme. Cela nous montre

également qu'il a compris l'exécution séquentielle des programmes sur ScratchJr. Nous pouvons aussi émettre l'hypothèse que l'élève 4 a des connaissances à la fois sur la causalité, car il comprend que l'action du premier personnage a un effet sur le démarrage du deuxième, et sur l'ordre temporel, parce que l'un personnage déclenche son programme avant l'autre mais aussi parce que le deuxième personnage démarre son programme au moment où le premier lui enverra le message. L'utilisation opérationnelle de la commande « quand message reçu » nous fait penser que l'élève 4 a compris qu'un programme peut aussi démarrer avec une autre commande que celle de « quand drapeau-vert cliqué ».

Compte tenu du fait qu'il a hésité une seule fois (pendant 8s d'après l'enregistrement vidéo) à utiliser la commande « envoyer un message » au programme du personnage « lapin » pour ensuite l'effacer sans avoir besoin qu'on intervienne, nous pouvons imaginer qu'il a réalisé cette action rapidement sans réfléchir au problème donné, en réitérant l'action qu'il a précédemment faite au PB1.

3.4.6. La répétition prédéfinie

Au vu de ce qui a été présenté dans son portrait en lien avec le motif des commandes à répéter au PB2, nous pouvons constater que l'élève 4 a beaucoup de difficultés à l'identifier et le créer. Grace aux nombreuses aides (15) qu'il a reçues et à sa décision de mettre en place une stratégie de « recommencer au début », il a réussi à créer trois fois un même motif des commandes, mais pas celui qui a été demandé par le projet modèle, puisque la commande d' « enregistrement d'un son » était toujours erronée. En effet, l'élève a enregistré trois fois la phrase demandée « bonjour le papillon » dans une commande d'enregistrement et ainsi son personnage faisait un pas à droite et il disait trois fois « bonjour le papillon » et il répétait cette séquence de commandes trois fois (ce qui parvient à neuf fois), au lieu de faire un pas à droite et dire une fois « bonjour le papillon » et répéter cette séquence de commandes trois fois.

Nous pouvons réfléchir aux raisons pour lesquelles l'élève 4 a des difficultés à repérer le motif des commandes qu'il se répète au PB2 et à le créer. Une raison expliquant ces difficultés pourrait être la mauvaise compréhension de la consigne du problème donné. En effet, puisqu'il existe une certaine ambiguïté dans le discours de la consigne du problème donné, il est possible que celle-ci n'ait pas été claire pour l'élève. Nous supposons ainsi qu'il

n’a pas bien compris dans la consigne ce qui devait se répéter. Il est également possible que ces difficultés avec l’identification du motif des commandes qui se répète sont dues au fait que l’élève 4 n’a pas étudié attentivement ce que fait le personnage « lapin » dans l’animation du projet modèle, pour comprendre ce qu’il doit reproduire dans son projet. Nous pouvons également supposer qu’à l’origine de ce problème se trouvent les difficultés qu’il a avec le processus du débogage et surtout l’étape d’évaluation de son programme pendant lequel il est censé exécuter son programme et comparer le résultat avec le résultat du projet modèle, pour identifier les erreurs existantes.

Une autre raison pour laquelle l’élève 4 a des difficultés avec l’identification du motif des commandes est le fait qu’il n’est pas habitué avec ce type d’exercice, c’est-à-dire à voir une répétition et essayer de comprendre ce qui se répète. Nous supposons également qu’il a des difficultés à identifier le motif des commandes qui se répète peut-être parce que c’est beaucoup pour lui de penser en même temps à deux commandes. De plus, nous pouvons faire l’hypothèse qu’il était difficile pour cet élève de ne plus réfléchir à l’une commande après l’autre mais de réfléchir à un groupe des commandes.

Compte tenu du fait qu’il utilise une seule fois la commande de « répétition prédéfinie » et d’une mauvaise manière, nous pouvons comprendre que l’élève 4 a des difficultés face à l’utilisation de cette commande. Nous pouvons réfléchir à différentes raisons expliquant ces difficultés. Nous pouvons supposer qu’il n’a pas compris la fonctionnalité de celle-ci. Il est également possible qu’il ait compris à quoi elle sert cette commande, mais il n’ait pas compris l’organisation de son écriture et c’est pourquoi il n’arrive pas à l’utiliser de manière opérationnelle.

Une raison de ces difficultés pourrait être la complexité de la nature de la commande. Cette dernière est très différente des autres commandes sur ScratchJr. Pour utiliser la « répétition prédéfinie » de façon opérationnelle les élèves doivent avant tout avoir compris qu’elle sert à faire le personnage choisi répéter une action ou une suite d’actions dans la scène un nombre de fois prédéfini par son paramètre. Ils doivent ensuite identifier le type d’action ou d’actions que le personnage choisi répétera dans la scène. S’il s’agit d’une suite d’actions qui est répétée, ils doivent également réfléchir à l’ordre temporel d’exécution de ces actions par le personnage choisi étant donné que le positionnement des commandes dans un programme sur ScratchJr définit l’ordre d’exécution de ces derniers par le personnage choisi. Enfin, les élèves

doivent penser au nombre des fois que cette action ou cette suite d'actions devra être répétée, élément qui est défini par le paramètre de la « répétition ». Nous pouvons donc voir que son utilisation opérationnelle présuppose une prise en compte de plusieurs éléments, qui pose probablement problème à cet élève, d'autant plus s'il a déjà beaucoup des difficultés avec la reconnaissance du motif à répéter.

De plus, la commande de « répétition prédéfinie » sert d'une certaine manière à raccourcir une programmation qui pourrait sinon être longue. Le choix de l'élève 4 de créer trois fois le motif des commandes, même si ce dernier n'était pas correct, au lieu d'utiliser la « répétition », nous montre qu'il avait besoin de voir la version longue de la programmation, c'est-à-dire le motif qui est répété et le nombre des fois qui est répété. De cette façon, il était peut-être plus facile pour lui de gérer ce que fait son personnage. Il est aussi possible que les difficultés de l'élève 4 avec la « répétition prédéfinie » relèvent du fait que la commande de « répétition » représente d'une certaine manière l'opération mathématique de la multiplication, qui n'est pas encore maitrisée à cet âge-là.

3.4.7. La gestion du démarrage synchrone de deux personnages

En ce qui concerne le démarrage synchrone de deux personnages au PB2, nous pouvons supposer que l'élève 4 a compris comment le réaliser puisqu'il le met en place de manière opérationnelle pour résoudre le PB2. Compte tenu de cela, nous pouvons également supposer qu'il a compris la fonctionnalité de la commande « quand drapeau-vert cliqué », puisqu'il s'agit d'un prérequis pour la réalisation opérationnelle du démarrage synchrone de deux personnages.

Nous pouvons donc imaginer que la notion du démarrage synchrone est probablement familière à l'élève 4, puisqu'elle lui rappelle une action qu'il a déjà réalisée dans le cadre des jeux avec d'autres élèves au sein de l'école ou même dans le cadre extrascolaire. Nous nous référons ici à des jeux de courses où tous les élèves commencent à courir en même temps. Nous pouvons aussi faire l'hypothèse que le jeu d'enfant-robot que nous avons utilisé pendant la séance de familiarisation avec le logiciel ScratchJr a facilité la compréhension du démarrage avec le drapeau-vert. En effet, ce type de jeu donne la possibilité aux élèves de commander leurs camarades, à l'aide de deux cartes une verte et une rouge, pour commencer

à se déplacer sur un quadrillage au moment où ils voient la carte verte et d'arrêter au moment où ils voient la carte rouge.

3.4.8. Les erreurs réalisées et le débogage mis en place

Compte tenu de son comportement face aux erreurs réalisées au cours de la résolution de deux problèmes, nous pouvons constater que l'élève 4 a des difficultés avec le débogage, qui sont plus visibles au PB2 qu'au PB1. Nous pouvons énumérer une série d'indices présents dans son portrait confirmant ce constat. En effet, l'élève a besoin d'aide pour identifier les erreurs sur ses programmes même après les avoir exécutés. Il a besoin de réaliser plusieurs débogages « non efficaces » pour arriver au débogage « efficace ». Il a également reçu 8 fois de l'aide sur la vérification de sa programmation et 10 fois de l'aide sur l'identification de l'erreur dans sa programmation, au cours de la résolution de deux problèmes. Les corrections qu'il réalise au PB1 tiennent compte du résultat obtenu après l'exécution de ses programmes, mais la majorité des corrections qu'il réalise au PB2 pas.

Une raison de ces difficultés pourrait être le fait qu'il ne réalise pas attentivement la première étape de ce processus : l'évaluation de son programme. En effet, nous supposons que, lorsque l'élève 4 exécute son programme pour le comparer avec le résultat souhaité et identifier des erreurs éventuelles, il ne vérifie pas attentivement son programme et il ne le compare pas suffisamment avec le résultat souhaité. Une autre raison expliquant ces difficultés pourrait être un manque des connaissances par rapport au fonctionnement des certaines commandes ou par rapport à leur écriture. Nous aurions pu faire l'hypothèse qu'il a des difficultés avec le débogage, car il n'arrive pas à bien mémoriser ce qu'il doit reproduire pour résoudre un problème. En revanche, vu le nombre des fois où nous sommes intervenue pour lui rappeler la consigne ou pour lancer le projet modèle pour qu'il se rappelle du résultat souhaite, une telle hypothèse nous semble peu probable.

Il est éventuellement possible que ces difficultés soient dues au fait que parfois cet élève initialise trop rapidement la position de ses personnages suite à l'exécution de ses programmes. En faisant cela, il ne peut pas réaliser efficacement des débogages, puisqu'il n'a plus accès à l'état de sa scène suite à l'exécution de la dernière version de son programme et ainsi il ne peut pas comparer cela avec le résultat souhaité et identifier les erreurs. Nous pouvons aussi faire l'hypothèse qu'il a parfois des difficultés avec le débogage, parce qu'en

étant en CP il commence à se familiariser avec le processus de résolution de problème en mathématiques, mais il n'a pas encore pris l'habitude de réfléchir à ce qu'il a produit et de vérifier à son exactitude, par rapport à ce qui lui a été demandé.

En analysant les patterns de débogage identifiés sur les actigrammes de l'élève 4, nous pouvons remarquer trois familles différentes : celle où l'action d'initialisation arrive après la ou les corrections, celle sans action d'initialisation avec une ou plusieurs corrections et celle où l'action d'initialisation arrive avant la ou les corrections. Nous pouvons également constater que la famille de patterns qui est la plus utilisée par l'élève 4 pendant la résolution de deux problèmes est celle où l'action d'initialisation arrive après la ou les corrections (9 fois). Ensuite, la famille la plus utilisée est celle sans initialisation avec une ou plusieurs corrections (6 fois), suivi par celle ou l'action d'initialisation arrive avant la ou les corrections (3 fois).

La famille de patterns e*xécution-correction/corrections-initialisation* facilite d'après nos observations la réalisation des débogages « efficaces », notamment lorsque les erreurs à corriger concernent le déplacement des personnages, puisqu'elle permet à l'élève d'étudier l'état de la scène suite à l'exécution de son programme et ainsi le comparer avec le résultat souhaité. Il est important de souligner que cette famille facilite mais elle ne garantit pas la réalisation de débogages « efficaces ». En revanche, la famille de patterns *exécution-initialisation-correction/corrections* rend plus difficile la réalisation des débogages « efficaces », notamment lorsque les erreurs à corriger concernent le déplacement des personnages, puisqu'en initialisant après l'exécution la position des personnages l'élève n'a plus accès à l'état de la scène suite à l'exécution de son programme et ainsi, il ne peut plus comparer cela avec ce qu'il devrait avoir selon le problème donné, pour identifier les erreurs existantes. La famille de patterns *exécution-correction/corrections* peut aussi poser des problèmes à la réalisation des débogages « efficaces », car il est possible que l'élève réalise des corrections pour améliorer son programme et puisqu'il oublie d'initialiser la position de ses personnages, il ne peut pas vérifier l'exactitude de sa programmation.

Compte tenu de ces observations, nous pouvons faire plusieurs interprétations des choix des patterns de débogages que l'élève 4 fait. Puisque la famille de patterns la plus utilisée (9 fois) par l'élève 4 est celle qui semble, d'après nos observations, favoriser la réalisation des débogages « efficaces », nous pouvons supposer qu'il a peut-être compris les avantages de

celle-ci et c'est pourquoi il la privilégie. Vu le fait que la deuxième famille de patterns la plus utilisée (6 fois) suit le même ordre de type d'actions que la première mais sans l'action d'initialisation, nous pouvons faire l'hypothèse qu'au moment de l'évaluation l'élève 4 était en train d'adopter la famille de patterns qui semble faciliter la réalisation des débogages « efficaces » mais qu'il oubliait parfois l'initialisation ou lançait son projet avec le bouton du drapeau-vert en haut de la scène, et ainsi il n'avait pas besoin d'initialisation. Nous pouvons également penser qu'il avait compris que la famille de patterns où l'action d'initialisation arrive juste après l'exécution rend plus difficile la réalisation des débogages « efficaces », puisqu'il s'agit de la famille de patterns la moins utilisée par l'élève 4. Compte tenu du fait qu'il l'utilise peu (3 fois) pendant la résolution de deux problèmes de programmation, nous pouvons émettre l'hypothèse qu'au moment de l'évaluation, il était en train de la remplacer par la famille la plus efficace.

3.4.9. Les aides apportées

Au vu des erreurs qu'il a réalisées au cours de la résolution du PB1, l'absence des moments de blocage, les aides que nous lui avons apportées, les débogages et les corrections qu'il a faites, nous pouvons constater que l'élève 4 a eu des difficultés avec le « déplacement » de ses personnages, l'initialisation, la commande « envoyer un message » et la commande de « fin ». Nous pouvons également supposer qu'il a eu les plus des difficultés face au déplacement de ses personnages, puisqu'il a reçu les plus d'aides là-dessus. Enfin, nous pouvons supposer qu'il a réussi, avec notre aide, à surmonter toutes ces difficultés puisqu'il est arrivé à corriger toutes les erreurs existantes sur ses programmes et ainsi à produire l'animation demandée.

En prenant en compte les erreurs qu'il a réalisées au cours de la résolution du PB2, les moments de blocage que nous avons notés, les aides que nous lui avons apportées, les aides qu'il a réclamées, les débogages et les corrections qu'il a faites, nous pouvons arriver au constat qu'il a eu des difficultés avec l'initialisation, le « déplacement » de ses personnages, l'enregistrement d'un son pour le motif, le motif à faire répéter et l'enregistrement d'un son pour le personnage « papillon », car il avait mal entendu la consigne du problème donné. Nous pouvons également comprendre qu'il a eu les plus des difficultés face à la programmation du motif à répéter, puisqu'à l'exception d'un seul, tous les moments de blocage qu'il a sont liés au motif à répéter (7 sur 8), il a même réclamé de l'aide (2 fois) pour savoir comment il va le faire, il a reçu le plus d'aides sur ce point (15), il a réalisé les plus des

débogages « non efficaces » là-dessus et à la fin de sa programmation, il n'a toujours pas pu surmonter cette difficulté et ainsi le motif reste erroné même dans ses programmes finaux.

Nous pouvons aussi remarquer que le déplacement de ses personnages lui a posé les plus des difficultés après le motif à répéter, car il a reçu beaucoup d'aides sur ce point (8 fois) et il a réalisé aussi plusieurs débogages « non efficaces ». L'initialisation de la position de ses personnages lui a posé ensuite les plus des difficultés, car il a reçu de l'aide là-dessus mais beaucoup moins que les autres (3 fois). Enfin, nous pouvons supposer qu'il a réussi, avec notre aide, à surmonter toutes les difficultés à part celles sur le motif à répéter, car il est arrivé à corriger toutes les erreurs existantes sur ses programmes à part, comme on l'a vu plus haut, celle sur le motif à répéter.

4. Comparaison des 4 élèves

Après avoir présenté individuellement pour les quatre élèves de CP1 nos hypothèses d'interprétation de ce qu'ils font, notre objectif dans cette section est de comparer leurs comportements, afin d'identifier les similarités ou les différences existantes et ainsi arriver à mieux comprendre ce qu'ils sont capables de faire en programmation sur ScratchJr et ce qu'il leur pose problème.

4.1. Récapitulatif des actions réalisées, du temps passé sur chaque problème et du nombre d'aides par élève

Tout d'abord, nous allons comparer les actions réalisées par les quatre élèves, le temps passé sur chaque problème et le nombre d'aides qu'ils ont reçues pendant la séance d'évaluation. La comparaison de ces éléments nous aidera à comprendre quel était le problème qui leur a posé les plus des difficultés et quels élèves étaient plus à l'aise face à la résolution de ces problèmes. Dans les Tableaux 31 et 32 qui suivent nous regroupons les éléments que nous venons de présenter pour les quatre élèves de CP1 et ils nous aideront à faire les comparaisons souhaitées.

PB1				
	Élève 1	Élève 2	Élève 3	Élève 4
Interface	20	15	22	13
Commandes	15	29	8	12
Corrections	2	0	20	5
Initialisation	3	5	16	4
Exécution	3	2	12	5
Question posée par l'élève	0	0	0	0
Total d'actions par élève	43	51	78	39
Temps de résolution	3min22s	3min21s	6min02s	3min10s
Aides	1	2	10	8
Problème réussi	Oui	Oui	Oui	Oui

Tableau 31 : Récapitulatif d'actions du temps et des aides par élève pour le PB1

PB2				
	Élève 1	Élève 2	Élève 3	Élève 4
Interface	17	14	35	26
Commandes	14	27	35	23
Corrections	3	4	6	53
Initialisation	6	9	9	20
Exécution	6	7	4	22
Question posée par l'élève	0	1	0	2
Total d'actions par élève	46	62	89	146
Temps de résolution	4min35s	5min02s	6min34s	12min16s
Aides	8	10	19	28
Problème réussi	Oui	Oui	Oui	Non

Tableau 32 : Récapitulatif d'actions du temps et des aides par élève pour le PB2

Comme nous pouvons voir dans le Tableau 31, tous les élèves arrivent avec de l'aide à résoudre le PB1. Les élèves 3 et 4 ont eu besoin de plus d'aides que les élèves 1 et 2 pour le résoudre. L'élève 2 a eu besoin du même temps que l'élève 1, mais de plus d'actions que celui-ci, car elle a mis en place une stratégie particulière pour le déplacement de ses personnages, stratégie qu'elle a inventée seule. L'élève 3 a eu besoin de plus de temps, plus d'actions et plus d'aides que tous les élèves pour arriver à la solution.

Nous observons également que l'élève 4 a eu besoin de moins de temps et moins d'actions que les élèves 1 et 2. Cela s'explique par le fait qu'il a réalisé une bonne estimation en ce qui concerne l'évaluation de la distance réalisée par ses personnages et il n'a donc pas eu besoin

d’effectuer beaucoup de corrections et d’actions sur l’interface. Cela se justifie aussi par le fait qu’il a effectué moins d’actions sur les commandes, car il a utilisé directement les paramètres. Toutefois, l’élève 4 a eu besoin de plus d’aides que les élèves 1 et 2 pour résoudre le PB1, mais moins d’aides que l’élève 3.

Le Tableau 31 nous aide aussi à comprendre que les élèves 1 et 2 étaient plus à l’aise face à la résolution du PB1 que les élèves 3 et 4, car elles ont eu besoin de moins d’exécutions pour arriver à la solution. Nous pouvons remarquer que l’élève 3 est celui qui est le plus en difficulté face au PB1, car il réalise plus de corrections que les autres élèves. Il a aussi eu besoin de plus d’exécutions, de plus d’aides et de plus de temps que les autres. Nous pouvons donc constater que les élèves 1 et 2 ont une meilleure performance au PB1 que les élèves 3 et 4 et que l’élève 4 a une meilleure performance que l’élève 3.

En analysant les éléments présents au Tableau 32 nous pouvons observer qu’à l’exception de l’élève 4, tous les élèves arrivent avec de l’aide à résoudre le PB2, portant entre autres sur l’utilisation opérationnelle de la « répétition ». De plus, nous observons que le nombre total d’actions des élèves est proportionnel au temps passé sur la résolution du problème. En faisant une comparaison des Tableaux 31 et 32, nous pouvons constater que tous les élèves ont eu besoin de plus d’actions, de plus de temps et de plus d’aides pour résoudre le PB2 que pour résoudre le PB1. À l’exception de l’élève 3, tous les élèves ont également eu besoin de plus d’exécutions pour arriver à la solution au PB2 qu’au PB1. De plus, nous pouvons voir que pendant la résolution du PB2, il existe deux élèves qui réclament de l’aide, alors qu’au PB1 aucun de quatre élèves ne fait une telle demande. La comparaison de deux Tableaux nous permet également de voir qu’il existe des différences plus importantes entre les élèves concernant le temps de résolution du PB2 et le temps de résolution du PB1. En effet, les temps de résolution du PB1 par les élèves 1, 2 et 4 sont très proches entre eux et celui de l’élève 3 est le seul qui diffère des autres de trois minutes. Au PB2, nous voyons une différence de deux minutes entre le temps de résolution de l’élève 1 et celui de l’élève 3. Il y a également une différence de presque une minute entre le temps de résolution de l’élève 1 et celui de l’élève 2 et une différence beaucoup plus importante entre le temps de résolution de l’élève 4 et ceux des autres élèves, six minutes avec celui de l’élève 3, sept minutes avec celui de l’élève 2 et huit minutes avec celui de l’élève 1. Ces éléments nous font penser que le PB2 est plus difficile à résoudre que le PB1.

En comparant les actions des élèves entre elles au PB2, nous pouvons observer que les élèves 1 et 2 ont eu besoin de moins d'actions, moins de temps et moins d'aides pour résoudre le problème que les élèves 3 et 4. Nous pouvons donc comprendre que les élèves 1 et 2 ont une meilleure performance face au PB2 que les élèves 3 et 4. L'élève 1 a la meilleure performance de tous les élèves face au PB2, puisqu'elle a besoin de moins d'actions, moins de temps et moins d'aides que les trois autres élèves. De plus, cet élève réalise moins de corrections que les autres élèves au PB2. L'élève 2 a eu besoin de plus d'actions, plus de temps et plus d'aides que l'élève 1, mais moins que les élèves 3 et 4. L'élève 2 réclame aussi de l'aide une fois pendant la résolution du PB2, mais la façon dont elle le fait en posant une question précise sur l'ordre des commandes que le personnage « lapin » réalise d'après la consigne du problème nous montre qu'elle a compris l'exécution séquentielle des commandes sur ScratchJr et son lien avec l'utilisation opérationnelle de la commande de « répétition prédéfinie ». L'élève 2 réalise aussi peu de corrections au PB2.

L'élève 3 a aussi des difficultés face au PB2 mais moins que l'élève 4, puisqu'il arrive à le résoudre avec de l'aide. Le Tableau 32 nous permet également de voir que l'élève 3 réalise peu d'exécutions et peu corrections malgré le fait qu'il reçoit un grand nombre d'aides. Cela s'explique par le fait que cet élève se trouve longtemps en blocage au PB2, face à l'erreur sur le démarrage du programme du personnage « papillon ». L'élève 4 semble être celui ayant le plus de difficultés face au PB2, car il n'arrive pas à le résoudre malgré les aides apportées. Cela se confirme par le fait qu'il a besoin de plus d'actions, de plus de temps et de plus d'aides que tous les élèves, mais aussi par le fait qu'il réclame de l'aide deux fois pendant la résolution du PB2 et la façon dont il le fait, en demandant juste comment faire ce qu'il n'arrive pas à faire, nous montre qu'il ne comprend pas comment il doit procéder pour la corriger. En revanche, nous pouvons remarquer sur le Tableau 32 qu'il réalise beaucoup de corrections. Cela nous montre qu'il n'arrête pas d'essayer de trouver la solution à son problème, sans pour autant y arriver. En effet, l'élève 4 réalise plus d'actions de correction au PB2 que tous les élèves au PB1 et au PB2.

4.1.1. La catégorie d'actions la plus sollicitée : l'« interface »

En ce qui concerne les éléments d'interface, nous observons que tous les quatre élèves arrivent à les gérer bien, mis à part l'initialisation de la position des personnages. En revenant sur les Tableaux 31 et 32, nous pouvons remarquer que l'interface est une des catégories

d'actions les plus sollicitées par les élèves au cours de la résolution de deux problèmes de programmation. Cela nous aide à comprendre le rôle fondamental que l'interface joue sur la programmation sur ScratchJr.

4.1.2. Une difficulté commune : l'initialisation

En ce qui concerne le processus d'initialisation de la position des personnages, nous pouvons observer que tous les quatre élèves se trompent plus ou moins sur celui-ci, certains plus et d'autres moins. Tous les élèves reçoivent au moins une aide sur l'initialisation de la position de leurs personnages. En consultant les Tableaux 31 et 32, nous pouvons observer que le seul élève qui réalise le même nombre d'actions sur l'initialisation et sur l'exécution aux deux problèmes est l'élève 1. Cela nous fait comprendre que cette élève initialise la position de ses personnages avant chaque exécution pendant la résolution de deux problèmes.

En faisant une comparaison du nombre des fois total que les quatre élèves se trompent sur l'initialisation de la position des personnages au PB1 et au PB2, nous pouvons constater que les élèves 1 (deux fois) et 2 (quatre fois) se trompent moins sur ce processus que les élèves 3 (17 fois) et 4 (neuf fois). L'élève qui reçoit les moins d'aides (trois aides) là-dessus est l'élève 1, suivi par les élèves 2 et 4 (quatre aides) et ensuite par l'élève 3 qui reçoit les plus d'aides de quatre (sept aides). Nous pouvons donc constater que l'élève 3 se trompe le plus sur l'initialisation de la position des personnages. Cela se confirme aussi en consultant le Tableau 31 où nous pouvons remarquer que l'élève 3 réalise 16 actions d'initialisation au PB1, alors que tous les autres élèves en réalisent maximum 5. Un autre indice de ces difficultés est le fait que l'élève 3 se trompe même après l'apport des aides et que la position initiale de son personnage « enfant » reste erronée jusqu'à la fin de sa programmation. Nous pouvons aussi repérer sur le Tableau 32 que l'élève 4 réalise plus d'actions d'initialisation que tous les élèves au PB2, mais cela s'explique par le fait qu'il passe beaucoup plus de temps que les autres élèves sur ce problème et qu'il réalise ainsi beaucoup plus d'actions au total que les autres élèves.

En analysant la façon dont les quatre élèves réalisent l'initialisation au cours de la résolution de deux problèmes, nous sommes amenée à constater que les élèves 1 et 2, qui ont moins de difficultés face à l'initialisation, sont ceux qui utilisent le plus l'icône d'initialisation. En effet, l'élève 1 qui a la meilleure performance sur le processus d'initialisation est celle qui utilise

que l'icône d'initialisation, mise à part lors du premier positionnement de ses personnages qui se fait obligatoirement avec le doigt. L'élève 2 qui présente la deuxième meilleure performance sur ce processus, utilise parfois l'icône d'initialisation et parfois son doigt, suite au premier positionnement de ses personnages qui se fait obligatoirement avec le doigt. En revanche, les élèves 3 et 4 qui présentent une moins bonne performance sur le processus d'initialisation préfèrent utiliser leur doigt pour initialiser la position de leurs personnages, dans la suite du premier positionnement avec le doigt.

4.1.3. Après l' « interface », les « commandes » et les « corrections »

L'analyse des actions réalisées par les élèves au PB1 (voir Tableau 31), nous permet de remarquer qu'après l'interface, les élèves réalisent les plus d'actions sur les catégories « commandes » et « corrections ». En effet, nous observons qu'à l'exception de l'élève 3, tous les élèves réalisent plus d'actions sur les « commandes » au PB1. Cela se justifie pour les élèves 1 et 2 car elles mettent en place des stratégies de programmation qui leur permettent d'écrire tout le programme de leurs deux personnages ou tout le programme de l'un de deux personnages avant la première exécution. L'élève 2 utilise aussi une stratégie de « déplacement » particulière qui l'oblige à solliciter beaucoup la catégorie « commandes ». Pour l'élève 4, cela s'explique parce qu'il fait une bonne estimation en ce qui concerne le nombre de pas dont ses personnages doivent se déplacer dans la scène pour arriver sur l'endroit demandé et non pas grâce à son mode de programmation. En revanche, l'élève 3 met en place des stratégies au PB1 qui lui permettent d'écrire une partie de son programme beaucoup moins longue avant la première exécution et c'est pourquoi il réalise plus d'actions sur les « corrections » que sur les « commandes ».

En analysant les actions des élèves au PB2 (voir Tableau 32), nous pouvons remarquer qu'à l'exception de l'élève 4, tous les autres élèves réalisent plus d'actions sur les « commandes » que sur les « corrections ». Cela peut être expliqué si on considère le fait que les élèves 1,2 et 3 utilisent tous des stratégies de programmation qui leur permettent d'écrire tout le programme de leurs deux personnages ou tout le programme de l'un des deux personnages avant la première exécution, alors que l'élève 4 écrit une partie du programme de chacun de ses deux personnages avant la première exécution.

4.2. Cinq stratégies de programmation

En étudiant la façon dont les quatre élèves procèdent pour créer leurs programmes, nous pouvons remarquer cinq stratégies de programmation. La stratégie « planification complète », qu'un élève met en place lorsqu'il est capable d'écrire l'intégralité des commandes du programme de ses deux personnages avant l'exécution. La stratégie « planification partielle », qu'un élève met en place lorsqu'il est capable d'écrire l'intégralité des commandes du programme d'un personnage avant l'exécution. La stratégie « sous-programme », qu'un élève met en place lorsqu'il est capable d'écrire une partie du programme de son personnage avant l'exécution et la stratégie de « pas à pas », qu'un élève met en place lorsqu'il écrit une commande et il l'exécute, etc. La stratégie « recommencer au début », qu'un élève met en place lorsqu'il décide d'effacer tout le programme de son personnage et recommence sa programmation du début.

En faisant une comparaison des stratégies de programmation utilisées par les quatre élèves, nous pouvons repérer des similarités et des différences. L'élève 1 est la seule des quatre à réaliser une « planification complète » aussi bien au PB1 qu'au PB2. L'élève 3 arrive aussi à réaliser une « planification complète » mais il le fait qu'au PB2. L'élève 2 est la seule des quatre à utiliser les hybridations « planification partielle et pas à pas » ou « sous-programme et pas à pas » ou « planification partielle et sous programme », car elle est la seule à programmer tout le programme de son personnage ou une partie de celui-ci et à souhaiter tester les parties qui lui paraissent susceptibles de contenir des erreurs, avant de tout tester. L'élève 2 présente ce comportement aussi bien au PB1 qu'au PB2. L'élève 4 est le seul des quatre à utiliser une stratégie « recommencer au début », car, puisqu'il n'arrive pas à repérer l'erreur sur son programme au PB2, il a choisi de tout effacer pour recommencer de zéro.

Nous observons également que les élèves 2 et 4 réalisent tous les deux une « planification partielle », puisqu'ils programment l'intégralité des commandes du programme de l'un de leurs deux personnages. L'élève 2 fait cela au PB1, tandis que l'élève 4 le fait au PB2, après avoir effacé son programme d'avant. Les élèves 3 et 4 utilisent tous les deux une stratégie de « sous-programme » au PB1, car ils écrivent d'abord, une partie du programme de leur personnage avant l'exécution et programment ensuite le reste de son programme en utilisant plusieurs fois une stratégie de « pas à pas ». Ils font cela pour leurs deux personnages au PB1. Contrairement aux élèves 1 et 2, les élèves 3 et 4 changent leur mode de programmation entre

le PB1 et le PB2. En effet, l'élève 3 utilise une « planification complète » au PB2. L'élève 4 procède au PB2 de la même manière qu'au PB1 jusqu'à un moment et ensuite, il réalise une stratégie de « recommencer au début » et enfin, une « planification partielle ».

L'analyse de la manière dont les élèves procèdent pour écrire les programmes de leurs personnages au PB1 et au PB2, nous a également permis de remarquer que les élèves 1 et 2 écrivent d'abord le programme d'un personnage et passent ensuite à l'écriture du programme de l'autre personnage aussi bien au PB1 qu'au PB2. En revanche, les élèves 3 et 4 procèdent de cette façon seulement pendant la résolution de l'un de deux problèmes. Il s'agit, en particulier, du problème où les élèves 3 et 4 réalisent le moins d'erreurs, le PB2 pour l'élève 3 et le PB1 pour l'élève 2. Pour écrire le programme de leurs personnages à l'autre problème, les élèves 3 et 4 réalisent plusieurs allers-retours entre les personnages, c'est-à-dire ils écrivent une partie du programme de l'un des personnages, ensuite ils passent à l'autre ils écrivent une partie de son programme, ensuite il exécute celui du premier, etc.

Ces éléments nous font comprendre que l'élève 1 présente le mode de programmation le plus efficace des quatre élèves, puisqu'elle arrive à écrire l'intégralité des commandes du programme de ses deux personnages avant la première exécution. Le deuxième mode de programmation le plus efficace est d'après nous celui de l'élève 2, car elle est capable d'écrire la plupart du temps l'intégralité des commandes de l'un des deux personnages et ensuite, exécuter uniquement les parties qui lui paraissent susceptibles de contenir des erreurs. L'élève 3 présente le troisième plus efficace mode de programmation puisque, même s'il écrit au PB1 une partie du programme d'un personnage avant l'exécution et le reste de son programme il l'écrit en utilisant plusieurs fois une stratégie de « pas à pas », au PB2, il est capable d'écrire l'intégralité des commandes du programme de deux personnages avant l'exécution. L'élève 4 présente le mode de programmation le moins efficace de quatre élèves, car, aussi bien au PB1 qu'au PB2, il arrive à écrire une partie du programme d'un personnage avant l'exécution et écrire le reste de son programme en utilisant plusieurs fois une stratégie de « pas à pas ». Au cours du PB2 toutefois, il arrive aussi à écrire l'intégralité des commandes du programme de l'un personnage.

4.3. Les stratégies de déplacement

En comparant la façon avec laquelle les quatre élèves programment le déplacement de leurs personnages dans les deux problèmes, nous pouvons constater que les élèves 1, 3 et 4 utilisent les « paramètres » en essayant d'estimer à chaque fois le nombre de pas dont leur personnage a besoin pour arriver au bon endroit dans la scène. L'élève 1 utilise également une fois au PB2 les « paramètres et plusieurs fois les mêmes commandes » en essayant d'estimer le nombre de pas dont son personnage a besoin pour arriver au bon endroit dans la scène. Nous pouvons également remarquer que l'élève 2 utilise une stratégie qu'elle a inventée toute seule en utilisant la « grille[151] et les paramètres », pour tous les déplacements effectués au cours de la résolution des deux problèmes.

L'élève 3 utilise aussi une fois seulement « plusieurs fois la même commande et puis, il réduit l'écriture de son codage en utilisant une seule commande ». Ensuite, il utilise une stratégie de déplacement similaire à celle de l'élève 2 mais sans la « grille », pour choisir le nombre de pas que son personnage devrait faire pour arriver au bon endroit dans la scène.

4.4. La gestion du déplacement

Nous pouvons également constater que tous les quatre élèves se trompent sur le nombre de pas que leurs personnages doivent faire pour arriver au bon endroit dans la scène. En effet, à l'exception de l'élève 2, tous les élèves se trompent sur cela aussi bien au PB1 qu'au PB2. L'élève 2 est le seul qui ne fait l'erreur qu'au PB2. Cela s'explique par le fait qu'elle utilise une stratégie de déplacement qui lui permet d'estimer efficacement la distance à réaliser par ses personnages. L'élève 3 se trompe aussi sur la direction du déplacement de l'un de ses personnages au PB1.

4.5. Entre utilisation opérationnelle et utilisation inutile de « messages »

En faisant une comparaison de la façon avec laquelle les quatre élèves traitent les commandes de « messages », nous pouvons remarquer une série d'éléments. Tout d'abord, nous pouvons

151 Il s'agit d'un bouton de l'interface de ScratchJr qui fait apparaître une grille numérotée sur la scène avec les axes x et y.

observer que les quatre élèves arrivent à utiliser de manière opérationnelle les commandes de « messages » pour résoudre le PB1. Nous pouvons également remarquer que les élèves 1 et 2 n'utilisent pas ces commandes au PB2 où il y avait pas besoin de le faire, alors que les élèves 3 et 4 essaient de le faire. Nous pouvons donc comprendre que les élèves 1 et 2 présentent une meilleure performance que les élèves 3 et 4 en ce qui concerne l'utilisation des commandes de « messages ».

4.6. Entre 3 x motif et utilisation opérationnelle de la « répétition prédéfinie »

En comparant la façon avec laquelle les quatre élèves utilisent la commande de « répétition prédéfinie », nous pouvons noter des similarités et des différences. Tout d'abord, nous pouvons observer qu'à l'exception de l'élève 4, tous les élèves arrivent à répéter le motif des commandes demandé par le PB2, mais ils ne le font pas de la même manière. En effet, nous pouvons remarquer que les élèves 1 et 2 arrivent à utiliser la commande de « répétition prédéfinie » pour le faire, alors que l'élève 3 utilise trois fois le même motif des commandes au lieu d'utiliser un seul et la commande de « répétition ».

Les élèves 1, 2 et 3 n'ont pas eu des difficultés à enregistrer le son demandé, alors que l'élève 4 s'est trompé là-dessus en enregistrant trois fois la phrase demandée dans la même commande d' « enregistrement ».De plus, nous pouvons constater que les élèves 1 et 3 n'ont pas eu de difficultés à créer le motif de commandes qui se répète alors que les élèves 2 et 4 ont eu. L'élève 2 a pu les surmonter alors que l'élève 4 non, malgré les aides apportées.

Nous pouvons donc constater que les élèves 1 et 2 présentent une meilleure performance que les élèves 3 et 4, parce qu'elles arrivent à utiliser de manière opérationnelle la commande de « répétition prédéfinie ». L'élève 3 présente une meilleure performance que l'élève 4 en ce qui concerne le motif à répéter, car, même s'il n'utilise pas la commande de « répétition », il arrive à reproduire différemment ce qui est demandé au projet modèle. Nous pouvons également comprendre que l'élève 4 présente plus des difficultés que tous les élèves face à la répétition d'un motif des commandes.

4.7. Le démarrage synchrone : une source de confusion

En ce qui concerne le démarrage synchrone de deux personnages, nous pouvons constater que tous les quatre élèves arrivent à le mettre en place de manière opérationnelle pour résoudre le PB2. L'élève 3 est le seul qui a des difficultés avec celui-ci car, comme on a vu plus haut, il a utilisé la commande « quand message reçu » à la place de la commande « quand drapeau-vert cliqué » et pendant un moment ses personnages ne déclenchaient pas leur programme en même temps. Après plusieurs moments de blocage, nombreuses aides apportées et un débogage « non efficace », l'élève 3 a réussi à corriger cette erreur. Pendant la discussion réalisée avec lui après la fin de sa programmation, nous constatons toujours une confusion existante entre les commandes « quand message reçu » et « quand drapeau-vert cliqué ».

4.8. Récapitulatif des erreurs réalisées et du débogage mis en place par les 4 élèves

Nous allons maintenant comparer les erreurs que les quatre élèves réalisent au cours de la résolution de deux problèmes, ainsi que leur réaction face à ces erreurs, c'est-à-dire le débogage qu'ils mettent en place. Nous allons également comparer leurs programmes finaux pour voir s'il existe des erreurs qui restent jusqu'à la fin de leur programmation.

4.8.1. Capacité ou non d'identifier des erreurs existantes sur le programme avant de l'exécuter

En faisant une comparaison des erreurs que les quatre élèves font pendant la résolution de deux problèmes et le débogage qu'ils mettent en place, nous pouvons observer une série des similarités et des différences. Tout d'abord, nous pouvons observer que les élèves 1 et 2 présentent une capacité d'identifier certaines erreurs présentes dans leurs programmes avant de les exécuter, alors que les élèves 3 et 4 ne présentent pas une telle capacité. En effet, l'élève 1 fait cela une fois au PB1, car elle avait oublié d'ajouter la commande « envoyer un message » et elle est revenue à son programme avant de l'exécuter pour corriger cette erreur. L'élève 2 présente ce comportement aussi bien au PB1 qu'au PB2. Au PB1, elle se trompe en utilisant la commande « quand personnage touché » au lieu de « quand drapeau-vert cliqué » et elle retourne à son programme en disant « non ce n'est pas ça » et elle corrige cette erreur avant d'exécuter son programme. Au PB2, elle se trompe, comme on a vu plus haut, sur

l'ordre des commandes du motif à répéter mais elle arrive à corriger cette erreur, puisqu'elle nous pose une question précise pour vérifier dans quel ordre chronologique le personnage « lapin » réalise les actions du motif.

4.8.2. Des différences au niveau des types d'erreurs réalisées

Nous faisons une comparaison des types d'erreurs existants dans les programmes des élèves après la première exécution. Nous pouvons observer que les élèves 1 et 2 se trompent uniquement sur l'initialisation et le nombre de pas que leurs personnages doivent réaliser pour se déplacer jusqu'à l'endroit demandé dans la scène. Les élèves 3 et 4 font les mêmes erreurs mais en réalisent aussi d'autres. Les élèves 3 et 4 oublient d'ajouter la commande « envoyer un message » au programme du personnage « papillon » au PB1, et ils essaient d'utiliser les commandes de « messages » au PB2 (l'élève 3 les deux, l'élève 4 uniquement « envoyer un message »), alors qu'il n'y avait pas besoin. L'élève 3 se trompe ainsi sur le démarrage du personnage « papillon » au PB2, car il utilise la commande « quand message reçu » au lieu de la commande « quand drapeau-vert cliqué ». L'élève 4 efface la commande « envoyer un message » juste après l'avoir ajoutée dans l'espace de programmation de son personnage.

À part les erreurs qu'ils réalisent en commun, les élèves 3 et 4 en font aussi d'autres. L'élève 3 se trompe sur la direction du déplacement de l'un de ses personnages. L'élève 4 réalise d'autres erreurs : il essaie d'enregistrer un son pour le « papillon », alors que c'était le personnage « lapin » qu'il devrait dire une phrase, il se trompe sur l'enregistrement du personnage «lapin », il se trompe sur la création du motif à répéter et il oublie d'ajouter la commande de « fin » à la fin du programme des trois de ses personnages.

Nous pouvons donc observer que les élèves 3 et 4 réalisent plus d'erreurs que les élèves 1 et 2. Puisque les erreurs sont des marqueurs des difficultés, nous pouvons comprendre qu'il y a plus d'éléments qui posent problème aux élèves 3 et 4 pendant la résolution de deux problèmes que ceux qui posent problème aux élèves 1 et 2.

4.8.3. Débogage pour écrire une partie du programme ou débogage sur un programme entièrement écrit

En nous focalisant maintenant sur le débogage que les quatre élèves mettent en place pour corriger leurs erreurs, nous pouvons identifier des similarités et des différences et ainsi arriver

à une série des constats. Tout d'abord, nous pouvons remarquer que les élèves 1 et 2 utilisent le processus de débogage pour corriger les erreurs existantes sur le programme qu'ils ont écrit avant, aussi bien au PB1 qu'au PB2. L'élève 3 l'utilise de la même façon uniquement au PB2. En revanche, nous observons qu'au PB1 il l'utilise différemment, c'est-à-dire pour écrire le reste des programmes de ses personnages en faisant du « pas à pas ». L'élève 4 l'utilise aussi de la même manière aussi bien au PB1 qu'au PB2. Pendant la résolution du PB2, il change de comportement et il l'utilise pour corriger les erreurs existantes sur un programme qu'il a écrit avant, comme les élèves 1 et 2 font au PB1 et PB2 et comme l'élève 3 fais au PB2. Nous pouvons donc identifier deux façons différentes pour utiliser le débogage : pour corriger les erreurs existantes sur un programme déjà écrit avant ou pour écrire le reste d'un programme en passant par « pas à pas ».

4.8.4. Des erreurs communes, des réactions différentes

Il existe aussi des différences concernant le débogage mis en place par les élèves. Ces dernières sont plus visibles sur les erreurs du même type. En faisant une comparaison du débogage que les élèves mettent en place pour corriger l'erreur sur le déplacement de leurs personnages, un type d'erreur que tous les élèves font, nous pouvons remarquer que les élèves 1 et 2 présentent une meilleure performance que les élèves 3 et 4. Nous observons en particulier que les élèves 1 et 2 : corrigent directement les commandes qui posent problème dans leur programme, réalisent des corrections qui tiennent toujours compte du résultat obtenu après l'exécution de celui-ci et elles ont besoin de peu de temps, peu d'essais et peu d'aides de type « consigne » et « lancement du projet modèle » pour le faire.

En revanche, les élèves 3 et 4 ont besoin de plus des débogages, plus de temps et plus d'aides pour corriger ces erreurs. À part les aides de type « consigne » et « lancement du projet modèle », l'élève 3 a aussi eu besoin des aides de type « vérification du programme » et l'élève 4 des aides de type « vérification du programme » et « identification de l'erreur ». De plus, ils réalisent à la fois des corrections qui tiennent compte du résultat obtenu après l'exécution et d'autres qui n'en tiennent pas compte. L'étude du débogage réalisé par les élèves sur le déplacement, nous aide à comprendre que, même s'ils ne le mettent pas en place aussi bien que les élèves 1 et 2, ils semblent avoir compris le processus.

Nous pouvons aussi comparer la réaction de deux élèves à une autre erreur commune, celle sur le motif des commandes à faire répéter. L'élève 2 se trompe sur l'ordre des commandes du motif, mais avant d'exécuter son programme elle pose une question précise pour vérifier dans quel ordre chronologique le personnage « lapin » réalise les actions du motif. Après avoir eu cette information, elle corrige directement l'erreur sur son motif. L'élève 4 a aussi des difficultés avec la création du motif à répéter. Il nous a même posé deux fois une question pour savoir comment il doit faire, pour que son personnage se comporte de la même façon que dans le projet modèle.

Nous pouvons donc observer une différence importante ente les deux réactions. D'un côté, nous avons une élève qui nous pose une question précise « le lapin marche d'abord ? », et de l'autre nous avons un élève qui nous pose une question de type « comment il arrive le « lapin » à faire ça ? ». Nous pouvons donc observer d'une part, une élève qui a compris la fonctionnalité de la « répétition prédéfinie » et de l'exécution séquentielle sur ScratchJr et ainsi, elle sait exactement l'information dont elle a besoin pour corriger son erreur, et de l'autre côte, un élève qui ne sait pas comment il doit procéder pour reproduire ce qu'il voit sur le projet modèle et il demande de lui dire comment il doit faire.

4.8.5. Facteurs favorisant l'efficacité du débogage

En observant la réaction des élèves 3 et 4 face aux autres types d'erreurs réalisées, nous pouvons constater que l'efficacité de leur débogage dépend d'une part, de leur difficulté face à la compréhension de la fonctionnalité des commandes en question et de leur utilisation sur ScratchJr, et d'autre part, des aides apportées pour surmonter ces difficultés et corriger les erreurs.

4.8.6. Les moments de blocage : un indicateur des difficultés rencontrées

Les moments de blocage nous permettent de voir l'impact des difficultés que les élèves 3 et 4 ont à comprendre certaines commandes sur leur capacité de déboguer efficacement. Pour rappel, les élèves 1 et 2 ne présentent aucun moment de blocage pendant la résolution de deux problèmes. Il s'agit des moments qui arrivent, dans les vidéos, après l'exécution et lors desquels les élèves ne réalisent pas de corrections ou ils réalisent des actions sur l'interface qui n'affectent pas l'état de leur programme.

Nous pouvons par exemple remarquer que l'élève 3 présente les plus des difficultés face aux commandes « quand message reçu » et « quand drapeau-vert cliqué » et à l'initialisation de la position des personnages. Il présente dix moments de blocage en lien avec le démarrage du personnage « papillon » et un moment de blocage en lien avec l'initialisation. Après une série des débogages et beaucoup d'aides de différents types, l'élève 3 a réussi à corriger l'erreur sur le démarrage du programme de « papillon », mais pas celle sur la position initiale du personnage « enfant ».

L'élève 4 présente plus de difficultés face au motif à répéter. Il réalise sept moments de blocage en lien avec cette erreur, mais malgré des nombreuses aides et essais de débogage il n'a pas réussi à surmonter cette difficulté et le motif répété reste erroné jusqu'à la fin de sa programmation.

4.8.7. Trois familles de patterns de débogage identifiées

Nous allons maintenant comparer les patterns de débogage identifiés sur les actigrammes des quatre élèves. Pour rappel, nous avons défini en tant que pattern de débogage non pas une succession d'actions spécifiques (comme exécution, exécution, correction, correction, interface, initialisation), mais une succession de type d'actions (comme exécution, correction du code, initialisation) qui se répète au moins deux fois sur les actigrammes d'un élève ou deux élèves différents.

En faisant une comparaison des patterns de débogages identifiés sur les actigrammes de quatre élèves, nous pouvons identifier trois familles (voir Tableau 33): « pattern de débogage où l'initialisation est après la correction ou les corrections », « pattern de débogage où l'initialisation est avant la correction ou les corrections » et « pattern de débogage sans initialisation avec une seule ou plusieurs corrections ». Chaque famille comprend les patterns qui suivent la même succession de type d'actions, avec une ou plusieurs actions de correction.

Élève	Pattern de débogage	Suite d'actions	Morceau d'actigramme
1,2,3,4	Pattern de débogage où l'initialisation est après la correction	Exécution→ Correction → Initialisation	
	ou		
1,3,4	les corrections	Exécution→ Correction→ Correction→ Correction→ Interface→ Correction →Initialisation	 152
3,4	Pattern de débogage où l'initialisation est avant la correction	Exécution→ Initialisation→ Correction	
	ou les corrections	Exécution→ Initialisation→ Correction→ Interface→ Correction→ Correction→ Correction	 153
2,4	Pattern de débogage sans initialisation avec une seule correction	Exécution→ Correction	
4	ou plusieurs corrections		 154

Tableau 33: Patterns de débogage identifiés sur les actigrammes des élèves

152 Ce morceau d'actigramme est utilisé comme un exemple d'un débogage avec plusieurs actions de correction et l'initialisation après les corrections. Ce n'est pas un pattern qui est retrouvé plusieurs fois dans les actigrammes des élèves. Ce pattern pourrait avoir des formes différentes. Ces formes contiennent toujours des actions d'exécution, de correction et d'initialisation et ce qui change chaque fois c'est l'existence des actions d'interface, des questions posées par l'élève ou des aides apportées.

153 Voir note 152.

154 Voir note 152.

Pattern de débogage	Morceau d'actigramme	Élève 1	Élève 2	Élève 3	Élève 4	Nombre total d'utilisations
Pattern de débogage avec l'initialisation après la correction		3	1	2	3	16
ou les corrections		1	0	1	5	
Pattern de débogage où l'initialisation est avant la correction		0	0	1	1	6
ou les corrections		0	0	2	2	
Pattern de débogage sans initialisation avec une correction		0	1	0	4	8
ou plusieurs corrections		0	0	0	3	

Tableau 34: Enumération des patterns de débogage identifiés sur les actigrammes des élèves pour les PB1 & PB2

Le Tableau 33 nous permet de repérer que la famille de patterns dominante dans les actigrammes de quatre élèves est celle qui suit la succession de type d'actions : *Exécution-Correction/Corrections-Initialisation*. Il s'agit de la famille la plus utilisée des trois, avec 16 fois utilisations au total (voir Tableau 34). Comme nous l'avons déjà expliqué, cette famille de patterns est plus efficace que les autres, notamment en ce qui concerne la correction des erreurs sur le déplacement des personnages dans la scène. En effet, elle permet aux élèves d'avoir accès à l'état de la scène suite à l'exécution de leur programme, afin de le comparer avec le résultat souhaite et ainsi réaliser des corrections plus adaptées. Il s'agit également de la succession de type d'actions présentée dans le scénario pédagogique.

La deuxième famille la plus utilisée est celle qui suit la succession de type d'actions : *Exécution-Correction/Corrections*. Nous retrouvons les patterns de cette famille huit fois au total dans les actigrammes de quatre élèves (voir Tableau 34). Cette famille est moins efficace que la première, car les élèves la mettant en place ont oublié d'initialiser la position de leurs personnages, étape nécessaire pour vérifier l'exactitude de leur programmation sur ScratchJr.

La famille *Exécution-Initialisation-Correction/Corrections* est la moins répandue des trois, avec six utilisations au total (voir Tableau 34). Elle est aussi la moins efficace des trois en ce qui concerne la correction des erreurs sur le déplacement des personnages, car, en initialisant rapidement la position du personnage suite à l'exécution de leur programme, ils risquent de réaliser des corrections non adaptées, car ils n'ont plus accès à l'état de la scène pour le comparer avec le résultat souhaité et identifier les erreurs existantes.

Nous pouvons donc comprendre que les quatre élèves du CP1 ont tendance à utiliser plus la famille de patterns la plus efficace ou même la famille de patterns qui s'en rapproche et peu la moins efficace.

Après avoir classé les familles de patterns selon leur fréquence d'apparition dans les actigrammes des élèves, nous allons maintenant étudier quels patterns sont les plus utilisés par quels élèves. Le Tableau 34 nous aidera à voir cela de façon plus claire.

Tout d'abord, nous pouvons remarquer que le pattern *Exécution-Correction-Initialisation* est utilisé au moins une fois par tous les élèves. La famille des patterns *Exécution-Initialisation-Correction/Corrections* est utilisée uniquement par les élèves 3 et 4. Ces élèves utilisent aussi souvent des patterns de débogage avec plusieurs actions de correction. L'élève 3 utilise autant des fois des patterns des familles *Exécution-Correction/Corrections-Initialisation* et *Exécution-Initialisation-Correction/Corrections*. L'élève 4 utilise des patterns provenant des trois familles identifiées en privilégiant la famille *Exécution-Correction/Corrections-Initialisation*, ensuite la famille *Exécution-Correction/Corrections* et enfin la famille *Exécution-Initialisation-Correction/Corrections*.

Les élèves 1 et 2 utilisent toujours des patterns avec une seule correction, sauf l'élève 1 qui réalise une fois un débogage avec deux actions de correction. De plus, l'élève 1 utilise uniquement des patterns de la famille Exécution-Correction/Corrections-Initialisation. L'élève

2 utilise aussi bien le pattern Exécution-Correction-Initialisation que le pattern Exécution-Correction.

Compte tenu de ces informations, nous pouvons constater que l'élève 1 semble avoir mieux compris que tous les quatre élèves la succession d'actions à suivre pour réaliser des débogages efficaces, puisqu'elle est la seule à utiliser uniquement celle-ci pour ses débogages. En étudiant les patterns identifiés sur les actigrammes des autres élèves, nous pouvons observer que l'élève 2 présente la deuxième meilleure performance après l'élève 1, puisqu'elle utilise la succession d'actions la plus efficace et celle qui s'en rapproche sans l'action d'initialisation. De plus, elle n'utilise pas la famille de patterns la moins efficace avec l'action d'initialisation avant les actions de corrections. L'élève 4 semble suivre les élèves 1 et 2 en ce qui concerne la compréhension de la succession d'actions à suivre pour réaliser des débogages efficaces, puisqu'il privilégie la famille la plus efficace et celle qui s'en rapproche, mais il utilise aussi la famille de patterns la moins efficace. L'élève 3 semble avoir compris moins que les autres la succession d'actions à suivre pour réaliser des débogages efficaces, puisqu'au moment de l'évaluation il utilisait autant des fois la famille de patterns la plus efficace et la famille de patterns la moins efficace.

5. Vers des caractéristiques générales d'une bonne et d'une moins bonne performance en programmation

La comparaison de ce que font les quatre élèves de CP1 lorsqu'ils se trouvent face à deux problèmes de programmation sur ScratchJr, nous a permis d'identifier une série des similarités et des différences et ainsi mieux comprendre ce qu'ils sont capables de faire les élèves de cet âge-là en programmation sur ScratchJr, quelles sont leurs stratégies, leurs erreurs et leur débogage. Compte tenu de ce qu'on a vu sur les quatre élèves que nous avons étudiés de manière très fine, nous pouvons identifier des caractéristiques différentes.

5.1. Indices de bonne performance en programmation

Nous pouvons donc supposer qu'un élève qui semble avoir bien compris comment programmer sur ScratchJr peut présenter certaines ou toutes ces caractéristiques :

- Il a besoin de peu de temps, peu d'actions et peu d'aides pour résoudre un problème de programmation sur ScratchJr.
- Il gère bien les éléments d'interface.
- Il comprend que les commandes s'exécutent de façon séquentielle dès la gauche vers la droite sur ScratchJr.
- Il comprend que chaque programme doit avoir un début et une fin.

En ce qui concerne l'initialisation de la position des personnages :

- Il initialise la position de son personnage avant chaque exécution.
- Il utilise son doigt pour choisir la position initiale de son personnage dans la scène et ensuite, il utilise uniquement l'icône d'initialisation pour le remettre au début.
- Lorsqu'il oublie d'initialiser la position de son personnage avant d'exécuter son programme, il le repère directement et il revient le mettre à sa position initiale avant de procéder à la prochaine exécution.
- Il se trompe peu de fois sur l'initialisation de la position de ses personnages.

En ce qui concerne l'écriture de ses programmes :

- Il s'occupe d'abord, de l'écriture du programme de l'un personnage et ensuite, il passe à l'écriture du programme de l'autre personnage.
- Il arrive à écrire l'intégralité des commandes du programme de ses deux personnages avant la première exécution (stratégie de « planification complète »).
- Il arrive à écrire l'intégralité des commandes du programme de l'un des deux personnages avant la première exécution (stratégie de « planification partielle »).
- Il arrive à écrire tout le programme d'un personnage ou une partie de celui-ci et il est capable de tester uniquement les parties du programme qui risquent d'avoir une erreur (stratégie « planification partielle et pas à pas » ou stratégie « sous-programme et pas à pas »).

En ce qui concerne le déplacement de ses personnages :

- Il utilise la stratégie « grille et paramètres » afin de choisir de façon plus efficace le nombre de pas que son personnage doit réaliser pour arriver à l'endroit demandé dans la scène.
- Il utilise la stratégie « paramètres et estimation du nombre de pas ».
- Il se trompe sur le nombre des pas que ses personnages doivent réaliser pour arriver au bon endroit dans la scène.

En ce qui concerne les commandes de « messages » :

- Il arrive à utiliser les commandes « envoyer un message » et « quand message reçu » de manière opérationnelle.

En ce qui concerne la commande de « répétition prédéfinie » :

- Il arrive à utiliser la commande « répétition prédéfinie » pour répéter un motif des commandes un nombre des fois prédéfini.
- Il arrive à répéter le motif des commandes trois fois au lieu d'utiliser la commande de « répétition prédéfinie ».

En ce qui concerne le démarrage synchrone de deux personnages :

- Il est capable de programmer deux personnages à démarrer leur programme en même temps grâce à la commande « quand drapeau-vert cliqué » et le bouton du drapeau-vert en haut de la scène.

En ce qui concerne les erreurs et le débogage réalisés :

- Il réalise peu d'erreurs.
- Il est capable d'identifier les erreurs présentes sur ses programmes avant de les exécuter.
- Il est capable de poser une question précise pour obtenir une information spécifique qu'il a besoin pour corriger l'erreur sur son programme.
- Après avoir exécuté son programme, il s'exprime oralement en détectant l'erreur en disant par exemple « Ah ! », « Non ! », « Ce n'est pas ça ! ».
- Il réalise des erreurs uniquement sur, le nombre des pas que ses personnages doivent réaliser pour arriver au bon endroit dans la scène et l'initialisation de la position de ses personnages.
- Il n'a pas de moments de blocage.
- Il met en place le débogage pour corriger les erreurs sur un programme qu'il a entièrement écrit avant.
- Il corrige directement les commandes qui posent problème dans son programme.
- Il a besoin de réaliser peu de débogages pour corriger les erreurs sur son programme.
- Il est même capable de corriger les erreurs présentes sur le programme de deux personnages en réalisant un débogage « optimal ».
- Il a besoin de peu d'aides de type « consigne » et « lancement projet modèle » pour le faire.
- Les corrections qu'il réalise sur son programme tiennent compte du résultat obtenu suite à son exécution.
- Il réalise peu de corrections.
- Les patterns de débogage identifiés sur ses actigrammes suivent la succession de type d'actions la plus efficace : *Exécution-Correction/Corrections-Initialisation*.
- Ses programmes finaux ne contiennent pas d'erreurs.

5.2. Indices de moins bonne performance en programmation

Nous pouvons également supposer qu'un élève qui semble avoir moins bien compris comment programmer sur ScratchJr, peut présenter certaines ou toutes ces caractéristiques :

- Il a besoin de beaucoup de temps, beaucoup d'actions et beaucoup d'aides pour résoudre un problème de programmation sur ScratchJr.
- Il gère bien les éléments d'interface.
- Il comprend que les commandes s'exécutent de façon séquentielle de la gauche vers la droite sur ScratchJr.
- Il comprend que chaque programme doit avoir un début et une fin.

En ce qui concerne l'initialisation de la position des personnages :

- Il initialise la position de ses personnages la plupart des fois avant l'exécution de son programme.
- Il utilise son doigt pour choisir la position initiale de son personnage dans la scène et ensuite, au lieu d'utiliser l'icône d'initialisation, il continue d'utiliser son doigt pour le remettre au début.
- Il se trompe plusieurs fois sur l'initialisation de la position de ses personnages.
- Lorsqu'il se trompe sur l'initialisation de la position de son personnage, il n'arrive pas à repérer son erreur et ainsi, il procède à des corrections sur le déplacement de ses personnages qui ne sont pas adaptées.

En ce qui concerne l'écriture de ses programmes :

- Il réalise plusieurs allers-retours entre les personnages lorsqu'il écrit ses programmes, c'est-à-dire il écrit une partie du programme de l'un personnage, ensuite il passe à l'autre, il écrit une partie de son programme, ensuite il exécute celui du premier, etc.
- Il est capable d'écrire une partie du programme de l'un de deux personnages avant la première exécution (stratégie de « sous-programme »).

En ce qui concerne le déplacement de ses personnages :

- Il utilise plusieurs fois les mêmes commandes et les paramètres, en essayant d'estimer le nombre de pas que son personnage doit réaliser pour arriver au bon endroit dans la scène.
- Il se trompe sur le nombre de pas que son personnage doit réaliser pour arriver à l'endroit demande dans la scène.
- Il se trompe sur la direction du déplacement de son personnage.

En ce qui concerne l'utilisation des commandes de « messages » :

- Il arrive à utiliser de manière opérationnelle les commandes « envoyer un message » et « quand message reçu », mais parallèlement il essaie de les utiliser même lorsque ce n'est pas nécessaire.
- Il confond la fonctionnalité de la commande « quand message reçu » avec celle de la commande « quand drapeau-vert cliqué ».

En ce qui concerne le motif des commandes à répéter trois fois :

- Il n'arrive pas à utiliser la commande « répétition prédéfinie », pour répéter un motif des commandes, un nombre des fois prédéfini.
- Il n'arrive pas à identifier et à créer le motif des commandes qui se répète.
- Il se trompe sur l'enregistrement du son à répéter, car il enregistre trois fois le son demandé dans une commande d'enregistrement et ensuite il répète cette commande trois fois.

En ce qui concerne le démarrage synchrone :

- Il est capable de programmer deux personnages à démarrer leur programme en même temps grâce à la commande « quand drapeau-vert cliqué » et le bouton du drapeau-vert en haut de la scène.

En ce qui concerne les erreurs et le débogage réalisé :

- Il utilise le débogage pour écrire le reste du programme de ses personnages, après avoir écrit une petite partie de celui-ci avant la première exécution.
- Il réalise beaucoup d'erreurs.
- À part les erreurs sur le déplacement et l'initialisation, il peut aussi réaliser des erreurs sur les « messages », la « répétition prédéfinie », le motif à répéter, le démarrage synchrone et la commande de « fin ».
- Il n'est pas capable d'identifier les erreurs sur son programme avant de l'exécuter.
- Il n'est pas capable de poser une question précise pour corriger son erreur. En revanche, il pose des questions de type « comment il fait le lapin pour faire ça ? », démontrant qu'il ne sait pas comment faire pour corriger son erreur.
- Il présente des nombreux moments de blocage lorsqu'il a des difficultés à corriger une erreur. Pendant ces moments : il ne réalise pas d'actions, il réalise d'actions sur différents endroits de l'interface mais pas sur son programme, il re-exécute son programme plusieurs fois sas réaliser des corrections ou il ajoute des commandes dans l'espace de programmation et ensuite il les efface etc.
- Il a besoin de beaucoup de temps, beaucoup d'essais de débogage et beaucoup d'aides pour corriger ses erreurs.

- Parfois, même après l'apport de l'aide il n'arrive pas à corriger l'erreur sur son programme.
- Il réalise des corrections qui tiennent compte du résultat obtenu après l'exécution et d'autres qui n'en tiennent pas compte.
- Il réalise beaucoup de corrections.
- Les patterns de débogage identifiés sur ses actigrammes suivent la succession de type d'actions la plus efficace, *Exécution-Correction/Corrections-Initialisation* mais aussi la succession de type d'actions la moins efficace, *Exécution-Correction/Corrections-Initialisation.*
- Il arrive avec de l'aide à corriger certaines erreurs, mais parallèlement il n'arrive pas à corriger toutes les erreurs existantes dans ses programmes.
- Ses programmes finaux peuvent contenir des erreurs.
- Bien qu'il ne mette pas en place le débogage de façon optimale, il semble avoir compris le processus.

Pour conclure, les caractéristiques présentés ont émergé de notre analyse de la performance des quatre élèves du CP1 face à la résolution de deux problèmes de programmation sur ScratchJr. Ce sont des hypothèses à vérifier en analysant la performance d'un plus grand nombre d'élèves de cet âge-là face à la résolution des problèmes de programmation sur ScratchJr.

Discussion

Dans le cadre de ce travail de thèse, nous souhaitions apporter une contribution à l'étude de l'apprentissage de la programmation dans le milieu de la petite enfance. Dans cette partie nous rappellerons les grandes lignes de cette recherche et nous mettrons en avant ses principaux résultats en montrant les points communs et les différences avec les résultats des études précédemment présentées. Cette discussion générale nous permettra de présenter les limites de cette recherche et d'en proposer des perspectives à la fois pour l'enseignement de la programmation à des très jeunes élèves sur ScratchJr et pour des futures recherches.

1. L'objectif de recherche

Après avoir identifié un intérêt grandissant autour de l'introduction de l'informatique en tant que discipline à part entière dans l'éducation obligatoire, et surtout de l'apprentissage de la programmation par les plus jeunes élèves en raison de l'apparition des langages de programmation graphiques ou visuels adaptés à leur âge, nous avons constaté un manque d'informations dans la littérature sur ce que les très jeunes élèves font lorsqu'ils programment sur ScratchJr, les stratégies de programmations, le débogage qu'ils mettent en place et quelles sont leurs difficultés.

C'est pourquoi nous avons choisi de réaliser une recherche exploratoire pour comprendre ce que les élèves de 5 à 7 ans sont capables de faire seuls ou aidés en programmation sur ScratchJr et quelles sont leurs difficultés. Nous avons voulu en particulier étudier les stratégies de programmation et le débogage qu'ils mettent en œuvre. L'originalité de cette recherche et son apport à la littérature se situent dans l'éclairage qu'elle apporte dans l'étude de ses différentes questions.

Pour répondre à notre problématique, nous avons choisi de suivre une méthodologie qualitative pour explorer et comprendre profondément le phénomène étudié, ici l'activité de programmation des jeunes élèves sur le langage de programmation ScratchJr. Il nous a semblé pouvoir ainsi apporter une contribution nouvelle à la littérature sur le sujet. Nous avons décidé d'utiliser l'étude de cas en tant que stratégie de recherche dans ce travail de thèse. Nous avons

en particulier choisi d'étudier deux classes de grande section de maternelle en Grèce et deux classes de cours préparatoire en France.

2. Pour étudier la programmation sur ScratchJr, il faut d'abord l'enseigner

Pour étudier l'activité de programmation des très jeunes élèves sur ScratchJr, il a d'abord fallu enseigner la programmation aux élèves. Pour cela, nous avons conçu deux scénarios pédagogiques, le premier pour l'étude préparatoire en Grèce et le second pour l'étude principale en France. En Grèce, nous avons enseigné le scénario, alors qu'en France nous avons formé les enseignantes des classes pour qu'elles puissent l'enseigner. Lors des séances d'enseignement des deux scénarios, des problèmes à résoudre ont été donnés aux élèves afin qu'ils puissent utiliser les commandes enseignées en cours et ainsi mieux comprendre leur fonctionnalité. Pour évaluer ce que les élèves peuvent faire seuls ou aidés en programmation à la fin du scénario pédagogique, deux problèmes à résoudre leur ont été donnés. Un entretien individuel a également été réalisé pour comprendre ce qui posait problème ou pas aux élèves afin d'améliorer le scénario pédagogique.

3. Que font les élèves de maternelle en programmation sur ScratchJr ?

À partir de l'analyse de nos notes sur les observations du terrain au cours de la mise en place du scénario pédagogique, des programmes élaborés et des réponses données aux questions posées sur la commande de « répétition prédéfinie » et sur les « messages » par les 12 élèves de la GS2, nous sommes arrivée à une série des résultats qui nous aident à répondre à notre problématique. Nous rappelons que nous sommes intervenue pendant la séance d'évaluation pour aider les élèves à avancer, les raisons de ce choix méthodologique sont détaillées dans la partie 3. Notre intervention consistait à : rappeler la consigne du problème à résoudre, lancer le projet modèle en plein écran pour que les élèves puissent voir ce qu'ils doivent reproduire, rappeler la position initiale des personnages et poser des questions pour aider les élèves à surmonter une difficulté rencontrée.

3.1. Les limites de l'apprentissage par la découverte sur ScratchJr

Tout d'abord, nous avons observé qu'il existe certains éléments du logiciel ScratchJr que les élèves de maternelle semblent pouvoir découvrir tou seuls et d'autres qu'il faut leur présenter. Nous avons par exemple observé qu'ils arrivent sans problème à découvrir à la fois la majorité des éléments de l'interface (drag and drop, effacement des commandes, arrière-plan, personnages etc.) et les principales règles d'écriture et d'exécution des programmes. En revanche, nous avons dû par exemple leur présenter d'autres éléments tels que le processus d'effacement d'un personnage de la scène, l'exécution séquentielle des programmes, le démarrage et la fin d'un programme sur ScratchJr. Ce résultat rejoint les observations de Pea (1983) par rapport aux limites de l'apprentissage par la découverte et l'importance d'un enseignement pour mieux accompagner les élèves dans le processus d'apprentissage de la programmation.

3.2. Difficultés de la commande « sauter »

De la même manière que Tsourapi et al. (2018), nous avons constaté que les éléments de base de l'interface de ScratchJr et l'écriture des programmes n'ont pas posé des problèmes aux élèves de GS2. Parallèlement, les commandes de « déplacement » à part celle qui fait « sauter » un personnage, ne posent pas non plus de problèmes aux élèves de la GS2, car ils arrivent à les utiliser de manière opérationnelle pour résoudre le problème. Certains élèves se sont trompés aussi sur la commande « sauter » car au lieu d'utiliser cette dernière, ils ont utilisé une commande pour déplacer le personnage vers le haut dans la scène. Nos résultats correspondent à ceux de Strawhacker et al. (2017) et Strawhacker et Bers (2019) montrant que beaucoup d'élèves de grande section de l'école maternelle jusqu'à CE1 réalisaient la même erreur que celle identifiée dans nos données.

3.3. Du « jeu » à la programmation : un passage parfois difficile

Nous avons également constaté que les élèves de maternelle passent par différentes phases pour faire déplacer un personnage sur ScratchJr. Au début, les élèves de la GS2 ont eu des

difficultés à comprendre qu'il ne fallait pas seulement déplacer leur personnage, mais le déplacer vers un endroit spécifique dans la scène pour résoudre le problème donné. Une enseignante de maternelle participant à l'étude de Strawhacker et al (2017) avait remarqué le même comportement chez ses élèves. En effet, ces derniers avaient des difficultés à faire le passage entre, utiliser des « blocks » pour voir ce que leur personnage fera et programmer leur personnage à réaliser une tâche spécifique dans la scène (Strawhacker et al., 2017). À la fin de l'intervention, nous avons remarqué que tous les élèves de la GS2, même les deux ayant découvert et utilisé les paramètres pendant le scénario, utilisent « plusieurs fois la même commande » en tant que stratégie pour déplacer leurs personnages. Nous pouvons supposer que c'était peut-être plus simple pour les deux élèves d'utiliser plusieurs fois la même commande que d'utiliser les paramètres pour déplacer leurs personnages, car à cet âge-là la correspondance entre le chiffre et le nombre et la reconnaissance que le nombre peut exprimer une quantité (usage cardinal) ne sont pas complètement consolidés. Vigot (2018) identifie aussi des difficultés avec l'utilisation des paramètres et une préférence pour l'utilisation de la même commande plusieurs fois, même chez certains élèves de CE1 ou 2nd grade.

3.4. Une commande de déplacement de plus ou de moins

Face aux déplacements longs (4 ou 5 commandes), nous avons remarqué que les élèves de maternelle de la GS2 se trompent sur le nombre de pas que leur personnage doit réaliser pour arriver sur l'endroit demandé de la scène, en ayant une commande de plus ou de moins dans leur programme. Nous supposons que l'origine du problème se situe dans l'initialisation de la position des personnages. Ce résultat correspond aux observations de Vigot (2018) qui observe également que les programmes conçus par les élèves de CE1 ont souvent une « flèche en plus » ou une « flèche en moins ». Cette erreur pourrait aussi être liée à un manque des connaissances en lien avec l'orientation dans l'espace, la correspondance entre chiffre et nombre et son usage cardinal. Ce sont des notions qui font partie des programmes scolaires pour la maternelle en Grèce, mais elles ne sont pas complètement acquises à cet âge-là (ΙΕΠ, 2021a).

3.5. Correspondance entre commande et action mais non entre action et direction

À l'exception d'un seul, les 11 élèves de maternelle n'ont pas eu de difficultés à identifier les commandes nécessaires pour déplacer leur personnage vers la direction indiquée dans le projet modèle. Strawhacker et Bers (2019) repèrent aussi que tous les élèves participant à leur étude indépendamment de leur groupe d'âge (de grande section de l'école maternelle jusqu'à CE1) semblent maitriser ce type des commandes et pouvoir faire la correspondance entre celles-ci et l'action du personnage dans la scène. Toutefois, ils avaient des difficultés à associer la consigne orale par rapport à la direction du déplacement avec la direction indiquée. Lorsque les élèves n'avaient pas fait attention au support visuel accompagnant la consigne, il leur arrivait de nous demander de l'aide afin de savoir distinguer la droite de la gauche en posant des questions telles que: « la droite est par là ? ». Cela est un exemple illustrant les effets que l'apport d'aide de l'enseignant, ici de la chercheuse, pourrait avoir sur ce que les élèves peuvent faire pour résoudre un problème donné. Mayer (1988) souligne également l'importance de l'accompagnement dans le processus de l'apprentissage de la programmation.

3.6. Initialisation avec le doigt : un choix risqué

L'initialisation de la position des personnages pose problème aux élèves de maternelle de GS2. Ils oublient souvent d'initialiser la position de leurs personnages avant de tester leurs programmes. Ainsi, ils ne peuvent pas vérifier l'exactitude de leur programmation puisque celui-ci ne commence pas son déplacement du bon endroit dans la scène. D'autres élèves préfèrent initialiser la position des personnages avec le doigt et non pas avec l'icône. Nous avons observé que lorsque les élèves initialisent la position de leur personnage avec le doigt, le risque est de ne pas le remettre chaque fois au même début. Cela pose aussi problème à leur programmation car il affecte le nombre des pas que le personnage a besoin de réaliser pour arriver au bon endroit dans la scène. Tsourapi et al (2017) repèrent aussi des difficultés chez les élèves de maternelle en lien avec l'initialisation de la position des personnages sur ScratchJr. L'initialisation a également posé des problèmes à des élèves de CM1 et CM2 dans la programmation sur Scratch, d'après Franklin et al (2017). Nous pouvons donc constater qu'il s'agit d'une notion assez complexe même pour des élèves plus âgés.

3.7. Vérifier l'exactitude de la programmation : une habitude à développer

Les élèves de maternelle ont également des difficultés avec le débogage et surtout sa première étape, l'évaluation du programme ou ils doivent vérifier l'exactitude de leur programmation. Ils oublient souvent de penser à tester le programme qu'ils ont créé pour voir si cela fonctionne comme ils le souhaitaient ou pas. Nous supposons qu'à cet âge-là, ils commencent à être familiarisés avec la résolution des problèmes et ainsi ils n'ont pas l'habitude de vérifier l'exactitude de ce qu'ils ont produit. Ce résultat rejoint les observations de Tsourapi et al. (2017) qui identifient aussi cette difficulté chez les élèves de maternelle pendant la programmation sur ScratchJr. Strawhacker et al. (2017) observent que les élèves de maternelle avaient une bonne performance face à l'épreuve de débogage sur ScratchJr et ils réussissaient moins bien sur cette dernière que les élèves de CP et CE1, participant également à leur étude. Toutefois, les chercheurs ne nous donnent pas d'informations sur des difficultés éventuelles auxquelles les élèves font face.

3.8. La majorité des élèves de la GS2 utilisent sans problème les commandes de « démarrage et de fin » d'un programme

Le démarrage et la fin d'un programme ne posent pas de difficultés à la majorité des élèves de la GS2. Ils arrivent à utiliser de manière opérationnelle les commandes « quand drapeau-vert cliqué » et « fin » pour résoudre les problèmes donnés pendant l'évaluation. Certains élèves de la GS2 oublient de les ajouter dans les programmes de leurs personnages. Ce résultat rejoint les observations réalisées par Portelance et al (2015) et Strawhacker et al (2017).

Portelance et al (2015) remarquent aussi que les élèves d'école maternelle utilisent plus de blocs de démarrage (blocs jaunes) et de blocs de fin (blocs rouges) que les élèves de CE1 lorsqu'ils programmaient sur ScratchJr. Strawhacker et al (2017) constatent que les élèves de grande section de l'école maternelle participant à leur étude étaient capables à comprendre la séquence sur ScratchJr, mais ils avaient une moins bonne performance que les élèves de CP et CE1.

3.8.1. Présenter en mode plein-écran : une stratégie didactique qui semble aider

Toutefois, notre résultat ne correspond pas aux observations de Tsourapi et al. (2018) qui identifient que les commandes « quand drapeau-vert cliqué » et « fin » posent des problèmes aux élèves de maternelle lorsqu'ils programment sur ScratchJr. Une raison expliquant cette différence pourrait être le scénario pédagogique que nous avons mis en place et notamment les stratégies didactiques que nous avons utilisées. En effet, puisque ScratchJr permet l'exécution d'un programme lorsque les commandes « quand drapeau-vert cliqué » et « fin » n'en font pas partie, les élèves n'arrivent pas à comprendre la nécessité de leur utilisation dans leurs programmes et souvent ils les oublient. Pour surmonter cet obstacle nous avons demandé aux élèves pendant l'enseignement et l'évaluation de présenter leurs projets en mode plein-écran à la fin de la résolution de chaque problème donné, car le mode plein-écran oblige l'utilisateur à considérer l'utilisation de la commande « quand drapeau-vert cliqué » dans sa programmation, puisque la seule manière d'exécuter son projet est à l'aide du bouton « drapeau-vert » en haut de la scène.

3.8.2. Se baser sur une activité débranchée pour enseigner le démarrage et la fin de programme sur ScratchJr

De plus, notre choix de réaliser le jeu d'enfant robot avec les cartes, vertes et rouges, pour signaler le départ et l'arrivée de l'enfant-robot avant de passer à la programmation et ensuite, de se baser sur cette expérience pour enseigner les commandes « quand drapeau-vert cliqué » et « fin » a beaucoup aidé les élèves à comprendre ces deux commandes et à les utiliser ensuite de manière opérationnelle. Cela confirme le propos de Romero et al. (2018) sur les effets positifs que la combinaison des activités débranchées et des activités sur tablette pourrait avoir sur l'apprentissage de la programmation par les élèves.

En ce qui concerne le petit nombre d'élèves de notre échantillon qui oublient d'ajouter ces deux commandes dans les programmes de leurs personnages, nous supposons qu'ils ont des difficultés, car ils dominent moins bien que les autres les notions de la séquence et du temps. Il s'agit des notions qui font partie des programmes scolaires de l'école maternelle (ΙΕΠ,2021a), mais elles ne sont pas encore consolidées à cet âge-là (Piaget, 1981).

3.9. Démarrage synchrone réussi par la majorité des élèves de GS2

Nous avons également observé que la majorité des élèves de GS2 est arrivée à réaliser le démarrage synchrone de deux personnages pour résoudre le PB2. Ce résultat rejoint les observations réalisées par Flannery et al. (2013) montrant que les élèves de l'école maternelle jusqu'au CE1 étaient capables d'utiliser le bouton du drapeau-vert en haut de la scène pour lancer leurs programmes. Nous avons également remarqué que les élèves de maternelle qui avaient des difficultés avec la réalisation du démarrage synchrone de deux personnages avaient aussi des difficultés avec le démarrage du programme d'un personnage avec le bouton du drapeau-vert en haut de la scène. Comme nous l'avons expliqué plus haut, puisque ces élèves dominent moins bien que les autres les notions de séquence et de temps, nous supposons qu'ils ont des difficultés à saisir le démarrage synchrone car ce dernier implique la compréhension de la simultanéité. De plus, le démarrage synchrone de deux personnages grâce au bouton du drapeau-vert en haut de la scène représente une relation entre une cause et son effet. La causalité est une notion sur laquelle les élèves commencent à travailler dès la maternelle (IEП, 2021a), mais elle n'est pas encore consolidée à cet âge-là. Selon Piaget (1981, p. 6), « le temps est inhérent à la causalité ».

3.10. La commande préférée : l'enregistrement d'un son

À l'instar de Strawhacker et al. (2017), nous observons que la commande d' « enregistrement d'un son » est peut-être la commande préférée des élèves de maternelle. Parfois, ils ont des difficultés avec le menu d'enregistrement, car ils se trompent sur l'ordre d'utilisation des boutons pour enregistrer un son. La majorité des élèves de GS2 arrive à utiliser de manière opérationnelle la commande d' « enregistrement d'un son ».

3.11. La « répétition prédéfinie » : une commande difficile

La « répétition prédéfinie » est une des commandes qui pose la plupart des difficultés aux élèves de maternelle d'après nos observations. Ces derniers se trompent autant sur la répétition prédéfinie d'une commande que sur la répétition d'une séquence des commandes.

Les cartes que nous avons créées pour représenter les commandes du logiciel ont beaucoup aidé les élèves à repérer le motif des commandes qui se répétait pendant l'enseignement.

À l'exception de l'un d'entre eux, aucun élève n'a utilisé la commande de « répétition prédéfinie » de manière opérationnelle pour résoudre le problème donné. Nos résultats correspondent à ceux de Strawhacker et Bers (2019) et Portelance et al. (2015) qui montrent que les élèves de maternelle n'utilisent pas les commandes de contrôle telles que la « répétition » et lorsqu'ils en ont besoin pour accomplir une épreuve sur ScratchJr, ils ont des difficultés à les utiliser. Nos résultats rejoignent aussi les observations de Léonard et al. (2021). En ayant enseigné la programmation aux élèves de 5 à 6 ans à travers des activités sur un environnement de programmation dont le langage comporte uniquement des blocs représentant des couleurs et la répétition et des activités débranchées, Léonard et al. (2021) repèrent que les élèves de 5 à 6 ans ont des difficultés avec la répétition lorsque le motif à identifier comprend plusieurs commandes. Nos résultats confirment également le constat de Strawhackert et Bers (2015) que les élèves de maternelle ayant appris la programmation avec une interface tangible présentent une meilleure performance face aux épreuves portant sur la répétition, que les élèves ayant travaillé avec l'interface graphique et les élèves ayant travaillé avec les deux.

3.11.1. 3 x motif au lieu de la « répétition prédéfinie »

Nous remarquons que la majorité des élèves de maternelle dans notre étude a préféré créer trois fois le motif des commandes qui se répétait au lieu d'utiliser la commande de « répétition prédéfinie », mais certains continuaient à se tromper sur la commande d' « enregistrement du son », en enregistrant trois fois le son demandé dans la même commande d' « enregistrement ».

Pour expliquer cela, nous formulons trois hypothèses. La première est que les élèves n'avaient pas compris la fonctionnalité de la « répétition prédéfinie ». La deuxième est qu'ils avaient compris à quoi elle sert mais son utilisation leur paraissait difficile en raison de la complexité de l'écriture de la commande. De plus, le choix des élèves à créer trois fois le motif des commandes au lieu d'utiliser la « répétition » nous montre que ces derniers ont besoin de voir la version longue de la programmation, c'est-à-dire le motif qui est répété et le nombre de fois qui est répété. De cette façon, ils arrivent à mieux gérer ce que fait leur personnage. Ce

résultat est conforme aux observations de Piaget. Pour lui, les élèves ont toujours des difficultés à cet âge-là à comprendre des concepts qui sont abstraits. Ces derniers apprennent à cet âge-là à représenter un objet ou un événement sans le voir grâce à la fonction sémiotique (Piaget & Inhelder, 2012).

3.11.2. La « répétition prédéfinie d'une séquence des commandes » plus complexe que la « répétition prédéfinie d'une commande »

Les élèves de la GS2 arrivent mieux à décrire ce que fait une « répétition prédéfinie d'une seule commande » qu'une « répétition prédéfinie d'une séquence de commandes ». Nous imaginons que c'est difficile pour les élèves de maternelle de voir la « répétition d'une séquence de commandes » et de comprendre que celle-ci représente trois motifs de commandes. Cela demande une réflexion d'un niveau d'abstraction assez élevé qui n'est pas facile à cet âge-là d'après Piaget (Piaget et Inhelder, 2012).

3.12. Des expériences physiques : obstacle à la compréhension de la fonctionnalité de « messages »

Les « messages » sont aussi des commandes qui posent des difficultés aux élèves de maternelle. Certains élèves avaient des difficultés à saisir le rôle des commandes des « messages » quand ces derniers ont été présentés à l'aide des projets qui leur rappelaient une expérience physique de leur vie quotidienne. Ils pensaient par exemple que le personnage « ballon de foot » se déplaçait, car le personnage « enfant » avait exécuté un shoot et non pas parce qu'il avait reçu un « message ». Nous avons donc repéré qu'il existe un écart entre l'expérience physique et la programmation sur ScratchJr et cela peut constituer un obstacle pour les élèves. Strawhacker et Bers (2015) ont aussi remarqué une tendance chez les élèves de maternelle à attribuer à des robots des taches anthropomorphiques.

3.12.1. Visualisation parallèle des programmes à l'aide des cartes : stratégie didactique qui semble aider les élèves

Les cartes représentant les commandes du logiciel ont aidé les élèves de maternelle à créer les programmes de deux personnages et à les visualiser en même temps et ainsi comprendre que ces derniers communiquent grâce à l'envoi et la réception d'un message de couleur

spécifique. Cet exemple illustre à nouveau les effets positifs que la combinaison des activités débranchées et des activités sur tablette pourrait avoir sur l'apprentissage de la programmation par les élèves (Romero et al., 2018).

3.12.2. « Envoyer un message » plus difficile à utiliser que « Quand message reçu »

La moitié des élèves de maternelle a réussi à utiliser de manière opérationnelle les commandes de « messages » pour résoudre le problème donné. L'utilisation de la commande « envoyer un message » pose plus de problèmes aux élèves que celle de la commande « quand message reçu ». Les difficultés des élèves portent notamment sur le positionnement de la commande « envoyer un message » dans le programme de leur personnage. En effet, le positionnement de cette commande paraît difficile aux élèves de maternelle, car il implique une série des connaissances sur la causalité et l'ordre temporel qui ne sont pas encore acquis à cet âge-là, même si elles font partie des programmes scolaires pour la maternelle (ΙΕΠ, 2021a). La majorité des élèves de maternelle arrive à utiliser de manière opérationnelle la commande « quand message reçu » pour résoudre le PB1. Nous supposons que les élèves ont moins de problèmes avec cette commande d'une part, parce qu'en tant que bloc de démarrage elle oblige l'utilisateur à la poser au début d'un programme et d'autre part, parce qu'il y a un lien plus visible entre l'exécution de celle-ci et la réaction du personnage dans la scène.

Nos résultats concordent avec ceux de Strawhacker et Bers (2019) et de Flannery et al. (2013) qui identifient aussi des difficultés chez les très jeunes élèves face aux commandes de « messages ». Toutefois, nous nuançons ces résultats car la moitié de nos élèves réussit avec de l'aide à utiliser les deux commandes pour résoudre le problème donné. Cela peut s'expliquer à la fois par les choix didactiques mis en place lors de l'enseignement de ces commandes et par le fait que nous avons travaillé avec un petit nombre d'élèves et ainsi nous avons eu la possibilité de bien accompagner ces derniers dans l'apprentissage de la fonctionnalité de ces commandes. De plus, en prenant en compte l'âge des élèves nous avons décidé de ne pas enseigner certains aspects des commandes de « messages », tel que la couleur, pour faciliter la compréhension de la fonctionnalité de ces derniers par les élèves de maternelle.

3.12.3. Description de ce que font les commandes de « messages » plus complexes que leur utilisation

La majorité des élèves de la GS2 n'arrive pas à décrire ce que font les commandes de « messages » au Post-test. En revanche, nous avons observé qu'ils arrivent mieux à utiliser les commandes de « messages » pour résoudre un problème qu'à décrire ce qu'elles font. Ce résultat correspond aux observations réalisées par Kurland et Pea (1985) qui remarquent que les élèves avaient tendance à utiliser des structures complexes telles que les variables ou des commandes simples telles que la répétition pour résoudre des problèmes sur LOGO, mais ils n'étaient pas capables de décrire leur fonctionnalité.

Suite à cette étude préparatoire en Grèce, nous avons pu obtenir une vision globale de ce que les élèves de grande section de maternelle peuvent faire seuls ou aidés en programmation sur ScratchJr et leurs difficultés. Nous avons également pu améliorer notre scénario pédagogique et l'organisation de la méthodologie de mise en place de notre recherche pour le terrain en France.

4. Proposition d'une méthodologie pour l'analyse de l'activité de programmation des jeunes élèves

Les données collectées lors de l'étude principale en France nous ont permis d'étudier à la fois ce que les élèves de CP sont capables de faire seuls ou aidés en programmation sur ScratchJr et leurs difficultés, mais aussi leur processus de résolution de problèmes pour identifier les stratégies de programmation utilisées et le débogage mis en place.

Nous avons sélectionné quatre élèves de CP avec des comportements contrastés afin de réaliser une analyse fine de leur activité de programmation. Nous avons en particulier analysé les enregistrements vidéo et les programmes élaborés par les élèves pendant la deuxième séance d'évaluation portant sur la résolution de deux problèmes en programmation. Puisque nous n'avons pas trouvé une méthodologie pour analyser l'activité de programmation des élèves dans la littérature disponible sur le sujet, nous avons décidé d'en concevoir une.

Pour le faire, nous avons choisi de nous focaliser sur les actions des élèves au cours de la résolution de deux problèmes de programmation. Pour mieux analyser l'activité de programmation des élèves, il a d'abord fallu trouver une façon simple de la présenter. C'est pourquoi nous avons créé pour chaque élève un document avec deux colonnes contenant d'une part, les captures d'écran du logiciel et d'autre part, le fil d'actions des élèves au cours de la résolution des deux problèmes. Nous avons aussi décidé d'utiliser des actigrammes, des barres de couleurs représentant la succession d'actions des élèves, pour chacun des deux problèmes.

Pour créer les actigrammes, nous avons réalisé un premier codage des actions des élèves, en développant des définitions opérationnelles pour tous les codes utilisés. Nous avons décidé d'attribuer une couleur différente à chaque type d'actions réalisé par les élèves. Les cinq types d'actions retenus sont les suivants : interface, commandes, corrections, initialisation et exécution. Chaque catégorie d'actions comprend une liste d'exemples que nous avons repérés dans les enregistrements vidéo. Nous avons choisi d'indiquer les moments d'interaction entre l'élève et la chercheuse. En revanche, nous n'avons pas introduit dans les actigrammes les erreurs réalisées par les élèves au cours de la résolution des deux problèmes. Nous avons également décidé d'énumérer le nombre d'actions réalisées par catégorie d'actions et le nombre total d'actions réalisées par les quatre élèves pour la résolution du PB1 et du PB2. De plus, nous avons quantifié le temps passé sur chaque problème.

Les actigrammes, les captures d'écran du logiciel et la transcription du fil d'actions des élèves nous ont aidée à rendre plus intelligible la visualisation et de ce fait la compréhension de l'activité de programmation des élèves.

Après avoir représenté de façon simplifiée l'activité de programmation des élèves, nous avons étudié à la fois si leur succession d'actions était la même ou non, mais aussi s'il y avait des patterns qui ressortaient. Nous avons également élaboré un lexique adapté pour décrire nos données en combinant plusieurs modèles théoriques. Nous avons fait un tel choix d'une part, en raison de la complexité de l'activité de programmation et d'autre part, parce que nous n'avons pas trouvé de modèle décrivant toutes les facettes de cette activité, surtout au niveau de la petite enfance. Nous avons aussi créé nos propres typologies quand celles de la littérature ne convenaient pas à nos données pour leur description. Nous avons donc combiné les deux approches de codage, en réalisant du codage déductif et inductif (Miles et al., 2014).

À cette fin, nous avons eu besoin d'utiliser des travaux de recherche traitant les phases de résolution d'un problème en programmation (Pea & Kurland, 1984), les stratégies de programmation (Komis & Misirli, 2013, Fessakis et al. 2013, Vigot, 2018) et le débogage (Klahr et Carver, 1988). Ensuite, nous avons donc entrepris d'adapter ce lexique à nos données.

La principale difficulté à cette étape était d'identifier les catégories de la littérature dans nos données de corpus. En effet, les recherches existantes avaient fourni un lexique de catégories de stratégies de programmation et de débogage. Toutefois, ces modèles, à l'exception du modèle de Klhar et Carver (1988) sur le débogage, ne proposaient pas de méthode pour les identifier. Nous avons réussi à repérer une série d'éléments clés qui nous ont aidée à caractériser le mode de programmation des élèves. Ces éléments clés sont : 1) il faut regarder l'intégralité des actions des élèves pour pouvoir juger le mode de programmation des élèves, 2) il n'y a pas une distinction claire entre la conception et le débogage du programme par les élèves, 3) d'après la littérature, le débogage démarre une fois que l'élève a exécuté son programme et il a pu constater qu'il y a un écart entre le résultat obtenu suite à l'exécution et le résultat souhaité, 4) la clé pour clarifier cela a été pour nous l'exécution.

Ainsi, nous avons décidé de nous focaliser sur la façon avec laquelle les élèves ont utilisé l'exécution dans leur programmation pour caractériser les stratégies de programmation qu'ils utilisent. Concernant les stratégies de déplacement, nous avons décidé de décrire les différentes manières employées par les élèves dans les vidéos pour déplacer leurs personnages.

Par la suite, nous avons étudié la qualité des actions des élèves. Nous avons en particulier analysé la nature des erreurs qu'ils réalisent, s'ils arrivent à comprendre qu'ils ont réalisé une erreur, s'il y a des moments dans les vidéos où ils bloquent, ainsi que le débogage qu'ils mettent en place s'il y en a un. Nous avons aussi différencié les erreurs que les élèves arrivent à surmonter et les erreurs qui persistent jusqu'à la fin.

Pour analyser le débogage réalisé par les élèves, nous avons comparé les cinq étapes présentées dans le modèle de Klahr et Carver (1988) avec ce que les élèves font dans les vidéos pour corriger leurs programmes. D'après le modèle de Klahr et Carver (1988), le débogage démarre à partir du moment où l'élève exécute son programme et il repère qu'il existe un écart entre le résultat souhaité et le résultat obtenu suite à l'exécution. Pour juger le

débogage des élèves, nous avons examiné si le programme contenant l'erreur ou les erreurs est correct ou reste erroné, suite à la réalisation de l'action ou des actions de correction par l'élève.

Les actigrammes nous ont particulièrement aidée avec l'identification des patterns de débogage. Nous avons repéré trois types d'actions associées au débogage dans les actigrammes des élèves : l'exécution, la correction et l'initialisation. De plus, nous avons observé qu'un débogage pourrait contenir une ou plusieurs actions de correction. Dans le contexte de cette thèse, nous définissons en tant que pattern de débogage non pas une succession d'actions spécifiques (comme exécution, exécution, correction, correction, interface, initialisation), mais une succession de type d'actions (comme exécution, correction du code, initialisation). Nous considérons un morceau d'actigramme ou autrement dit une succession de types d'actions en tant que pattern de débogage quand ce dernier se répète au moins deux fois sur les actigrammes d'un élève ou sur les actigrammes de deux élèves différents.

5. Que font les élèves de CP en programmation sur ScratchJr ?

À partir de ces analyses, nous sommes arrivées à une série de résultats qui nous aident à répondre à notre problématique. Ces résultats nous permettent ainsi de comprendre ce que les élèves de 6 à 7 ans sont capables de faire seuls ou aidés en programmation sur ScratchJr, quelles sont leurs difficultés, mais aussi quelles sont les stratégies de programmation et le débogage qu'ils mettent en œuvre.

5.1. Des performances différentes face aux deux problèmes d'évaluation

Tout d'abord, nous avons observé que les quatre élèves de CP1 arrivent, avec de l'aide, à résoudre le PB1 et à l'exception d'un élève, tous les élèves arrivent avec de l'aide à résoudre le PB2. Nous pouvons donc constater que certains élèves ont réussi grâce à l'aide apportée à surmonter les difficultés rencontrées et à résoudre le problème donné, alors que d'autres n'ont pas réussi à résoudre le problème donné même après l'apport d'aide.

5.2. PB2 plus complexe que PB1

Le PB2 était plus difficile à résoudre que le PB1, car tous les élèves ont eu besoin de plus d'actions, de plus de temps et de plus d'aides pour le résoudre que pour résoudre le PB1. Nous pouvons donc constater que, de la même manière que les élèves de GS2 en Grèce, les élèves de CP1 en France ont eu plus de difficultés à réaliser l'épreuve portant entre autres sur l'utilisation opérationnelle de la « répétition prédéfinie » qu'à réaliser l'épreuve portant entre autres sur l'utilisation opérationnelle de « messages ». Ce résultat rejoint les observations de Strawhacker et Bers (2019) qui montrent que les commandes de contrôle telles que la « répétition prédéfinie » sont plus accessibles par les élèves de CP et de CE1 que par les élèves de maternelle. Toutefois, nous nuançons ces résultats car les élèves de CP réussissent avec de l'aide à utiliser les deux commandes des « messages » pour résoudre le problème donné, alors qu'aucun élève de l'étude de Strawhacker et Bers (2019) n'utilise les deux commandes de « messages ».

5.3. L'initialisation pose problème aussi en CP

En ce qui concerne les éléments d'interface, nous observons que, comme les élèves de la GS2, les élèves de CP1 ont des difficultés avec l'initialisation de la position des personnages. Ce résultat rejoint les observations de Franklin et al. (2017) qui montrent que l'initialisation a posé des problèmes à des élèves de CM1 et CM2 dans la programmation sur Scratch. En effet, nous remarquons que tous les élèves de CP se trompent et reçoivent de l'aide sur l'initialisation, certains plus et d'autres moins. De la même manière que les élèves de maternelle, certains élèves de CP utilisent l'icône d'initialisation et d'autres initialisent la position de leurs personnages avec le doigt, suite au premier positionnement qui se fait obligatoirement avec le doigt. Nous constatons également que les élèves qui ont moins de difficultés face à l'initialisation sont ceux qui utilisent le plus l'icône d'initialisation, alors que les élèves qui ont plus des difficultés face à l'initialisation préfèrent utiliser leur doigt, suite au premier positionnement qui se fait obligatoirement avec le doigt. Nous pouvons émettre l'hypothèse que les élèves de maternelle et de CP ont des difficultés avec l'initialisation car elle ne ressemble à aucune des activités qu'ils sont habitués à réaliser à l'école.

5.4. Cinq stratégies de programmation identifiées

En étudiant la façon dont les quatre élèves de CP1 procèdent pour créer leurs programmes, nous pouvons remarquer cinq stratégies de programmation : 1) la stratégie « planification complète », 2) la stratégie « planification partielle », 3) la stratégie « sous-programme » 4), la stratégie de « pas à pas » et 5) la stratégie « recommencer au début ». Nous avons aussi identifié des hybridations de ces stratégies : « planification partielle et pas à pas » et « sous-programme et pas à pas ».

D'après nos observations sur les quatre élèves, nous pouvons constater qu'il existe des élèves de 6 à 7 ans qui, non seulement sont capables de programmer sur ScratchJr, mais qu'ils peuvent présenter des capacités impressionnantes de planification en écrivant tout le programme d'un ou même de deux personnages avant l'exécution. Nos résultats montrent également qu'il existe un élève de cet âge-là qui est capable d'écrire tout le programme d'un personnage et de tester uniquement les parties du programme qui risquent d'avoir une erreur (stratégie « planification partielle et pas à pas »). Ce sont des capacités que nous n'attendions pas de voir à cet âge-là.

Ces résultats confirment l'hypothèse que, même les très jeunes élèves sont capables de programmer à l'aide d'un logiciel de programmation graphique adapté à leur âge et d'un scénario pédagogique approprié sur la programmation sur ScratchJr (Flannery et al., 2013 ; Bers, 2018) et de développer des stratégies de programmation. Nous pouvons également constater que l'évolution des langages de programmation graphiques a été efficace, car actuellement même les plus jeunes élèves sont capables de programmer et de se familiariser avec davantage de structures de programmation que celles disponibles sur les robots programmables.

Puisque le logiciel ScratchJr est adapté aux besoins des jeunes élèves et qu'il n'a pas de syntaxe complexe, nous avons pu identifier une large gamme des stratégies de programmation, par rapport aux stratégies identifiées par d'autres chercheurs avec des logiciels moins adaptés à leur âge (Fessakis et al. 2013).

5.5. Une performance en programmation similaire aux deux épreuves ou une performance en programmation qui change d'une épreuve à l'autre

Nous avons également observé que certains élèves de CP présentent une performance de programmation similaire pendant les deux épreuves données tandis que d'autres présentent des différences dans leur performance en programmation d'une épreuve à l'autre. Nos résultats montrent en particulier que certains élèves présentent des caractéristiques de programmation efficace et d'autres présentent des caractéristiques de programmation à la fois efficace et moins efficace. Nous avons par exemple repéré des élèves mettant en place une programmation organisée et des stratégies de programmation plus avancées et d'autres élèves combinant ce dernier avec des stratégies de programmation moins avancées et une programmation moins organisée.

Ce résultat correspond aux observations de Bers (2019) qui explique que les élèves peuvent en même temps présenter des caractéristiques des différents « stades » de programmation. Bien que l'objectif de ce travail de thèse ne soit pas de comprendre les stades de programmation par lesquels les élèves passent lorsqu'ils programment sur ScratchJr, nous observons que certains élèves de CP semblent être plus préoccupés par l'efficacité de la méthode utilisée pour atteindre l'objectif visé, en présentant ainsi des caractéristiques d'un stade *style concious* d'après Howe (1980). D'autres élèves se focalisent davantage sur l'objectif à atteindre en présentant ainsi des caractéristiques d'un stade *product oriented* toujours selon Howe (1980).

5.6. Quatre stratégies de déplacement identifiées

En étudiant la façon avec laquelle les quatre élèves de CP1 procèdent pour déplacer leurs programmes, nous pouvons remarquer les deux stratégies de déplacement que nous avons enseignées dans le scénario : 1) la stratégie de « paramètres », 2) la stratégie « plusieurs fois la même commande ». Nous avons observé que les trois élèves de CP1 utilisent principalement la stratégie de « paramètres ». Vigot (2018) trouve la même stratégie de déplacement pendant la programmation sur ScratchJr que nous, employée par les élèves de CE1, mais elle la nomme « stratégie de l'estimation ».

À l'opposé des élèves de maternelle, nous avons repéré sur deux élèves de CP1 une capacité d'inventer d'autres stratégies de déplacement, leur permettant une vitesse et une meilleure efficacité à la programmation du déplacement des personnages. Nous pouvons par exemple citer la stratégie 3) « grille et paramètres ». La dernière stratégie que nous avons identifiée est similaire que celle de « grille et paramètres », mais l'élève n'utilise pas la « grille ». Ces stratégies de déplacement n'ont pas été identifiées par Vigot (2018).

5.7. Des difficultés face à l'estimation d'une distance

De la même manière que les élèves de maternelle de la GS2, les quatre élèves de CP1 se trompent sur le nombre de pas que leurs personnages doivent faire pour arriver au bon endroit dans la scène. Nous supposons que les élèves de maternelle et de CP font cette erreur non pas parce qu'ils ont des difficultés à comprendre la fonctionnalité des commandes de « déplacement » ou à les utiliser, mais parce qu'il est difficile pour eux d'estimer une distance et notamment sur ScratchJr où la scène n'a pas par défaut des repères. Cette difficulté ne concerne pas l'activité de programmation mais la nature de la programmation à réaliser.

Nous pouvons donc constater que les très jeunes élèves peuvent avoir des difficultés avec la programmation du déplacement de leurs personnages sur ScratchJr en raison d'une consolidation partielle des connaissances mathématiques telles que la correspondance entre chiffre et nombre, l'usage cardinal du nombre et l'estimation d'une distance. Ce sont des notions qui sont au programme dès la maternelle (ΙΕΠ, 2021a) et qui sont approfondies au CP (BO n° 31 du 30 juillet 2020). Toutefois, elles ne sont pas dominées par tous les élèves de la même manière. À l'origine de ce problème pourrait aussi être l'initialisation de la position des personnages. Les quatre élèves de CP1 n'ont pas de difficultés à identifier les commandes nécessaires pour déplacer leur personnage vers la direction indiquée dans le projet modèle.

5.8. La commande préférée : l'enregistrement d'un son

À l'instar des élèves de la GS2 de maternelle, la commande d' « enregistrement d'un son » est aussi la commande préférée des quatre élèves de CP1. Parfois, ils ont parfois des difficultés avec le menu d'enregistrement car ils se trompent sur l'ordre d'utilisation des boutons pour enregistrer un son. Les quatre élèves de CP1 arrivent à l'utiliser de manière opérationnelle.

5.9. Le démarrage synchrone semble maitrisé mais confusion constatée avec « quand message reçu »

En ce qui concerne le démarrage synchrone de deux personnages, nous pouvons constater que les trois élèves de CP1 arrivent à le mettre en place de manière opérationnelle pour résoudre le PB2. De la même manière que la majorité des élèves de la GS2 de maternelle, un élève de CP1 parvient avec de l'aide, à le mettre en place de manière opérationnelle pour résoudre le PB2 de l'évaluation. Ce résultat rejoint les observations réalisées par Flannery et al. (2013) montrant que les élèves de l'école maternelle jusqu'à CE1 étaient capables d'utiliser le bouton du drapeau-vert en haut de la scène pour lancer leurs programmes. Toutefois, nous constatons chez un élève une confusion entre les commandes « quand message reçu » et « quand drapeau vert cliqué ». Le démarrage synchrone des personnages grâce au bouton du drapeau-vert en haut de la scène et le démarrage suite à la réception d'un message représentent une relation entre une cause et son effet. Le démarrage synchrone implique une compréhension de la concurrence et le démarrage suite à la réception d'un message implique quant à lui une compréhension de la causalité. La concurrence et la causalité sont des notions liées à la compréhension du temps, sur lesquelles les élèves commencent à travailler dès la maternelle (BO n° 31 du 30 juillet 2020) et ils continuent de les travailler pendant le CP (BO n° 31 du 30 juillet 2020), mais elles ne sont pas complètement consolidées à cet âge-là. Les élèves forment, selon Piaget (1981) une meilleure perception du temps à partir de 7 à 8 ans.

5.10. Entre 3 x motif et utilisation opérationnelle de « répétition prédéfinie »

Nous avons également observé qu'à l'exception d'un élève, les trois élèves de CP1 arrivent à répéter le motif des commandes demandé par le PB2, mais ils ne le font pas de la même manière. Deux élèves arrivent à utiliser de manière opérationnelle la commande de « répétition prédéfinie », alors que le dernier préfère, de la même manière que les élèves de maternelle, utiliser trois fois le motif des commandes au lieu d'utiliser un seul motif et la commande de « répétition ». Un des quatre élèves de CP1 a des difficultés à créer le motif des commandes qui se répète. Nous avons en particulier repéré qu'il réalise la même erreur que nous avons repérée chez les élèves de maternelle. En effet, il parvient avec de l'aide, à créer trois fois le même motif, mais il se trompe sur la commande d' « enregistrement d'un son »,

car il enregistre trois fois la phrase demandée par le problème donné dans une même commande d' « enregistrement d'un son » et ensuite, en créant trois fois le motif des commandes, son personnage répète 9 fois la phrase au lieu de trois. Malgré les aides apportées de notre part, cet élève n'a pas réussi à surmonter cette difficulté et ainsi à corriger cette erreur.

Ces résultats rejoignent les observations de Strawhacker et Bers (2019) qui montrent que les commandes de contrôle telles que la « répétition prédéfinie » sont plus accessibles aux élèves de CP et de CE1 qu'aux élèves de maternelle. Toutefois, nous nuançons ces résultats car nous remarquons que les élèves de CP peuvent aussi avoir des difficultés face à l'utilisation de la commande de « répétition », qui sont similaires à celles rencontrées par les élèves de maternelle. Nos résultats par rapport aux difficultés des quatre élèves de CP1 face à la répétition d'un motif des commandes correspondent aussi aux observations réalisées par Léonard et al. (2020). Ils indiquent que les élèves de 6 à 7 ans ont des difficultés avec la répétition lorsque le motif à identifier comprend plusieurs commandes.

5.10.1. La complexité de l'écriture de « répétition prédéfinie » : obstacle pour son utilisation

Nous pouvons émettre une série d'hypothèses par rapport aux raisons expliquant les difficultés des élèves de CP sur l'utilisation de la « répétition prédéfinie ». Il est possible que les élèves n'aient pas compris la fonctionnalité de la « répétition prédéfinie ». Nous supposons également qu'ils ont compris à quoi elle sert, mais son utilisation leur paraissait difficile en raison de la complexité de l'écriture de la commande. De plus, le choix des élèves de CP1 de créer trois fois le motif des commandes au lieu d'utiliser la « répétition » nous montre qu'ils ont besoin de voir la version longue de la programmation, c'est-à-dire le motif qui est répété et le nombre de fois qui est répété. De cette façon, ils arrivent à mieux gérer ce que fait leur personnage. Ce résultat rejoint la théorie de Piaget selon laquelle les élèves de cet âge-là ont des difficultés à percevoir des concepts qui sont abstraits (Piaget & Inhelder, 2012). En outre, puisque nous retrouvons une même difficulté chez les élèves de GS2 et de CP1 en ce qui concerne l'identification du motif des commandes qui se répètent, il est possible qu'il y ait eu une certaine ambiguïté dans le discours de la consigne des problèmes donnés.

5.10.2. Les élèves n'aiment pas raccourcir leur programme

Les discussions que nous avons eues avec les élèves de CP lors de la séance d'évaluation, nous ont permis d'émettre deux autres hypothèses. En effet, il est possible que les élèves n'utilisent pas la commande de « répétition » parce qu'ils l'ont oubliée ou parce qu'ils n'aiment pas cette commande puisqu'elle ne leur permet pas de programmer autant qu'ils veulent, car elle sert à raccourcir un programme qui sinon pourrait être long. Pila et al. (2019) repèrent que la préférence des élèves pour l'environnement de programmation pourrait influencer positivement la performance des élèves en programmation sur cet environnement. De la même manière, nous supposons que la « non préférence » des élèves pour la commande de « répétition » pourrait influencer les choix de programmation. Puisque nous retrouvons une même difficulté chez les élèves de maternelle et de CP en ce qui concerne l'identification du motif des commandes qui se répètent, il est possible qu'il y ait eu une certaine ambiguïté dans le discours de la consigne du problème donné.

5.11. Les commandes de « messages » : entre utilisation opérationnelle et utilisation inutile

En ce qui concerne les commandes de « messages », nous observons que les quatre élèves de CP1 arrivent à utiliser de manière opérationnelle les commandes de « messages » pour résoudre le PB1 de l'évaluation. Deux élèves utilisent les commandes de « messages » aussi au PB2, alors que ce n'était pas nécessaire. Nous supposons que ces derniers se sont approprié partiellement la fonctionnalité des commandes de « messages », puisqu'ils n'arrivent pas à bien saisir la notion de causalité, directement liée avec celles-ci. Comme nous l'avons déjà expliqué la causalité est une notion complexe, liée à la compréhension du temps, qui n'est pas encore consolidé chez les très jeunes élèves. Nous supposons que l'absence d'un lien visible entre la commande « envoyer un message » et la réaction du personnage « expéditeur » dans la scène rend la compréhension de la fonctionnalité de la commande et ainsi la compréhension de l'existence d'une cause provoquant un effet plus difficile pour les élèves. Ce résultat rejoigne ceux de Strawhacker et Bers (2019) et de Flannery et al. (2013) qui identifient aussi des difficultés chez les très jeunes élèves face aux commandes de « messages ». Toutefois, nous nuançons ces résultats car les quatre élèves de CP1 réussissent à utiliser les deux commandes pour résoudre le problème donné, alors que dans l'étude de Strawhacker et Bers

(2019) aucun élève de CP ou de CE1 ne le fait. Cela peut s'expliquer par les choix didactiques mis en place lors de l'enseignement de ces commandes et par le fait que les élèves de CP1 étaient aidés pendant l'évaluation.

Nous avons également observé qu'un des quatre élèves de CP1 arrive à utiliser les commandes de « messages » pour résoudre le problème donné, mais pendant une discussion avec nous à la fin de l'épreuve, il ne répond pas correctement sur ce que fait la commande « quand message reçu ». Ce résultat rejoint les observations de Kurland et Pea (1985) qui remarquent que les élèves avaient tendance à utiliser des structures complexes telles que les variables, ou des commandes simples telles que la répétition pour résoudre des problèmes, mais ils n'étaient pas capables de décrire leur fonctionnalité. Pila et al. (2019) trouvent aussi des résultats similaires, car dans leur étude les jeunes élèves étaient capables de programmer sur des environnements de programmation graphique, mais ils n'étaient pas capables de verbaliser leurs connaissances autour de ce qu'est la programmation.

5.12. Des erreurs de types différents

En ce qui concerne les erreurs, nous observons que les quatre élèves de CP1 en réalisent, mais ils ne réalisent pas les mêmes types d'erreurs. Deux élèves se trompent sur le déplacement et l'initialisation et les deux autres se trompent sur ces derniers mais également sur les « messages », la « répétition prédéfinie », le motif à répéter et le démarrage synchrone.

5.13. Des performances de débogage différentes

Nous avons observé que les quatre élèves de CP1 arrivent à mettre en place le processus de débogage pour corriger les erreurs sur leurs programmes. Nous avons toutefois remarqué différentes performances de débogage : deux élèves présentent une bonne performance et les deux autres une moins bonne performance. Ce résultat rejoint les observations de Pea et Kurland (1984) qui soulignent que les programmeurs experts mettent en place différemment le processus de débogage que les programmeurs novices.

5.13.1. Capacité d'identifier des erreurs existantes sur le programme avant de l'exécuter

Deux élèves de CP1 sont capables d'identifier les erreurs existantes sur leurs programmes et de les corriger avant de les exécuter. Une des deux arrive à poser une question précise pour obtenir une information dont elle a besoin pour savoir comment corriger cette erreur.

En effet, l'élève se trompe sur l'ordre des commandes qu'elle a ajoutées dans la commande de « répétition prédéfinie » et elle nous demande de lui rappeler dans quel ordre le personnage réalise ces actions d'après la consigne du problème. Cet exemple illustre les effets positifs que l'apport d'aide orale de l'enseignant, ici de la chercheuse, pourrait avoir sur ce que les élèves peuvent faire pour résoudre un problème donné. Il s'agit de la zone proximale de développement de Vygotski (1978) Selon lui, l'élève peut, à l'aide d'un adulte ou d'un élève plus expérimenté, résoudre des problèmes plus difficiles que ceux qu'il aurait pu résoudre tout seul.

5.13.2. Correction directe des commandes posant problème

Ces élèves arrivent aussi à corriger directement les erreurs présentes sur les programmes de leurs personnages, en ayant besoin ainsi de peu de débogages et peu de temps pour le faire. Elles peuvent également corriger les erreurs sur les programmes de deux personnages avec un seul essai, en réalisant ainsi un débogage de type « optimal ».

5.13.3. Des difficultés d'identification des erreurs sur les programmes après leur exécution

Les deux autres élèves de CP1 ont des difficultés à repérer les erreurs existantes sur leurs programmes même après leur exécution. Ces élèves ont besoin de beaucoup de temps, de beaucoup d'essais de débogage et de beaucoup d'aides pour corriger leurs erreurs. Ils ont plus des difficultés à corriger certaines erreurs notamment celles sur les « messages », la « répétition prédéfinie » et le « démarrage synchrone ». En effet, ils arrivent à identifier qu'il existe une différence entre leur programme et le programme souhaité, mais ils n'arrivent pas à identifier l'erreur qui provoque cette différence. Ainsi, ils arrivent à identifier ce que Carver et Klahr (1986) appellent *discrepancy,* mais ils n'arrivent pas à identifier l'erreur ou *bug* qui la

provoquait. C'est pourquoi ils présentent des moments de blocage au cours de la vidéo et ils réclament de l'aide pour la correction de leur programme.

5.13.4. « Comment je dois faire pour que le lapin fasse des petits pas comme ça ? »

Au lieu de poser une question précise, l'un des deux élèves demande de lui dire comment il doit faire pour que son personnage réalise la même chose que sur le projet modèle. Nous avons également observé qu'après plusieurs essais de débogage infructueux, il choisit d'effacer son programme et de recommencer à nouveau à programmer. Ce résultat correspond aux observations de Pea (1983) qui remarque que les élèves préféraient recommencer leur programmation au lieu de passer du temps à déboguer leur programme. Toutefois, nous nuançons les résultats de Pea (1983), car nous avons observé que l'élève dans notre corpus a effacé son programme pour recommencer à nouveau, après avoir réalisé plusieurs tentatives de débogage infructueuses.

5.13.5. Des erreurs restantes jusqu'à la fin de la programmation

À l'instar des élèves de maternelle, nous avons remarqué que deux élèves de CP1 n'ont pas pu corriger toutes les erreurs existantes sur leurs programmes, car les aides apportées n'étaient pas suffisantes pour qu'ils puissent surmonter les difficultés rencontrées. Nous pouvons donc supposer que des parties de cette épreuve dépassaient la zone proximale de développement de certains élèves (Vygotski, 1978) et c'est pourquoi, même avec de l'aide ils n'ont pas pu les résoudre.

5.13.6. Facteurs favorisant l'efficacité du débogage

Nous avons donc constaté que l'efficacité de débogage des élèves dépend d'une part, de leur difficulté face à la compréhension de la fonctionnalité des commandes et de leur utilisation sur ScratchJr et d'autre part, des aides apportées pour surmonter cette difficulté et corriger l'erreur.

5.13.7. Débogage pour écrire une partie du programme ou débogage sur un programme entièrement écrit

Nous avons également observé que les deux élèves de la CP1 mettent en place le débogage pour corriger les erreurs sur un programme qu'ils ont entièrement écrit avant, alors que les deux autres l'utilisent en faisant du « pas à pas » pour écrire le reste du programme de leurs personnages, après avoir écrit une petite partie de celui-ci avant la première exécution.

5.14. Trois familles de patterns de débogage

Trois familles de patterns de débogage ont été repérées sur les actigrammes des quatre élèves de CP1. Ces familles de patterns sont : 1) « pattern de débogage où l'initialisation est après la correction ou les corrections », 2) « pattern de débogage où l'initialisation est avant la correction ou les corrections » et 3) « pattern de débogage sans initialisation avec une seule ou plusieurs corrections ».

La première famille des patterns de débogage est la plus efficace, notamment en ce qui concerne la correction des erreurs sur le déplacement des personnages dans la scène. La troisième famille de patterns est la moins efficace de trois. Nous observons que les quatre élèves du CP1 ont tendance à utiliser davantage la famille de patterns la plus efficace ou même la famille de patterns qui s'en rapproche et peu la moins efficace.

5.15. Vers des caractéristiques générales des performances des élèves

Enfin, nous proposons, à partir des résultats de nos observations sur les quatre élèves de CP1, une série des caractéristiques générales correspondant à une bonne et à une moins bonne performance que les jeunes élèves peuvent présenter face à la résolution des problèmes de programmation sur ScratchJr.

À travers cette recherche, nous avons pu montrer ce que les élèves de 5 à 7 ans sont capables de faire seuls ou aidés en programmation sur ScratchJr et quelles difficultés ils risquent de rencontrer. Nous avons en particulier identifié les stratégies de programmation, le débogage et les patterns de débogage que les élèves de 6 à 7 ans mettent en œuvre lorsqu'ils résolvent des problèmes de programmation pendant la séance d'évaluation, après avoir suivi un scénario pédagogique.

Perspectives

1. Perspectives pour améliorer le scénario pédagogique

Le scénario pédagogique pourrait être davantage amélioré notamment à partir des résultats de cette recherche. Compte tenu des difficultés des élèves de maternelle et de CP sur l'évaluation de la distance que leurs personnages doivent réaliser pour arriver à un endroit spécifique dans la scène et du fait qu'une élève de CP a inventé une stratégie de déplacement en utilisant la « grille », nous pourrions l'intégrer davantage dans le scénario. Nous pourrions commencer par l'introduire en CP pour le déplacement des personnages, comprendre si cette dernière les aide ou pas dans la réalisation des déplacements, et ensuite si les résultats sont positifs, nous pourrions envisager une introduction de la « grille » dans le scénario pédagogique pour la maternelle.

Les résultats de notre recherche nous font comprendre que le scénario pédagogique doit davantage prendre en compte les connaissances en lien avec les mathématiques, les repères temporels, les repères spatiaux et la causalité, que les élèves auront besoin d'utiliser pour programmer sur ScratchJr. Des séances d'enseignement pourraient être dédiées au processus de débogage en faisant davantage le lien avec les processus que les élèves réalisent déjà dans le cadre de l'école et qui y ressemblent (estimation des quantités et vérification, relire un texte pour vérifier la conformité orthographique). L'initialisation de la position des personnages, qui est d'après nos observations une étape nécessaire pour la réalisation du débogage sur ScratchJr, nécessite aussi une attention particulière.

Les commandes les plus difficiles comme celles de « messages » et de la « répétition prédéfinie » devraient être introduites dans le scénario une fois que les élèves sont bien familiarisés avec le fait que chaque commande a un effet spécifique sur le personnage, l'exécution séquentielle des commandes, le débogage, le démarrage d'un programme à l'aide du bouton du drapeau-vert, le déplacement des personnages et le démarrage synchrone de deux personnages sur ScratchJr. Enfin, pour mieux aider les élèves à comprendre la fonctionnalité des commandes telles que la « répétition prédéfinie » et les « messages »,

davantage d'activités débranchées doivent être proposées aux élèves avant qu'ils commencent à explorer ces commandes sur ScratchJr. Pour l'enseignement des « messages », nous pourrions ajouter une activité similaire au jeu d'enfant-robot avant de passer au logiciel ScratchJr. Au lieu d'un seul élève « à programmer » sur le quadrillage, nous pourrions en avoir deux et un troisième élève à l'extérieur du quadrillage pour les programmer.

Pour l'enseignement de la « répétition prédéfinie », nous pouvons aussi ajouter des activités débranchées avant de présenter la commande sur le logiciel. Nous pouvons par exemple commencer l'initiation des élèves à la « répétition » par une activité de reconnaissance et de répétition des motifs des couleurs avec des briques colorées. Ensuite, nous pourrions réaliser une activité combinant le jeu d'enfant-robot et les cartes représentant les commandes du logiciel lors de laquelle un élève va programmer un autre élève pour qu'il répète le motif des commandes indiquées en étant sur le quadrillage.

2. Perspectives de recherche

Le travail de recherche que nous avons mené dans le cadre de cette thèse avait un caractère exploratoire. Pour comprendre profondément le phénomène étudié, ici l'activité de programmation des élèves, nous avons opté pour une méthodologie qualitative. C'est pourquoi nous avons décidé de travailler avec des petits échantillons. Ainsi, nous ne pouvons pas généraliser les résultats obtenus. En revanche, puisque nous avons réalisé une étude de deux cas en Grèce et une étude de deux cas en France, nous pourrions envisager une analyse des données collectées dans les autres classes afin de vérifier si nous retrouvons les mêmes résultats d'un cas à l'autre. Cela nous permettrait de renforcer leur validité (Miles et al., 2014), mais aussi d'aller plus loin dans notre compréhension de ce que les jeunes élèves sont capables de faire grâce à la comparaison de deux cas. Nous pourrions par exemple vérifier si nous retrouvons les mêmes stratégies de programmation, types et patterns de débogage chez les élèves de CP2.

Puisque nous avons décidé de mettre en place cette recherche dans des contextes dits écologiques, c'est-à-dire des classes réelles au sein des écoles, nous avons eu à respecter un certain nombre des contraintes liées à ce choix. En raison de difficultés d'accès aux écoles en Grèce, nous avons dû retarder notre intervention et faire en deux mois ce qui était prévu initialement en trois. Le retard pris lors de la mise en place de notre expérience en Grèce a eu

des répercussions sur les données que nous avons recueillies pendant l'évaluation car nous n'avons pas eu le temps de la réaliser individuellement pour chaque élève. Par conséquent, nous avons dû nous adapter aux contraintes du terrain et ainsi nous n'avons pu étudier seulement qu'une partie de notre problématique initiale. En revanche, puisque nous avons recueilli des données un peu plus riches pour quelques élèves de maternelle ayant la GoPro, que nous n'avons pas traité, une perspective éventuelle de cette recherche pourrait être d'analyser les stratégies de programmation ou le débogage que ces derniers mettent en place. De plus, l'étude préparatoire en Grèce nous a permis de tester notre scénario et l'organisation de la méthodologie de mise en place de notre recherche et ainsi de les améliorer pour l'étude principale en France.

Une autre contrainte a été le fait qu'au cours de l'étude préparatoire en Grèce nous avons enseigné notre scénario pédagogique, en ayant ainsi le contrôle sur les modalités d'enseignement, alors que pendant la mise en place de l'étude principale en France nous n'avons pas pu maîtriser complètement la réalisation du scénario pédagogique, car ce dernier a été pris en charge par les enseignantes des deux classes. En revanche, nous avons pris en charge la mise en place de la séance d'évaluation pendant le terrain en France.

Nous sommes ainsi consciente que l'enseignement mis en place a pu favoriser ou rendre plus difficile la compréhension de certains éléments de notre scénario pédagogique. Cependant, puisque nous n'avons pas eu le contrôle sur cela, nous ne pouvons pas réaliser de jugements là-dessus. Malgré cette limite, nous pensons qu'il était bénéfique pour nous de pouvoir former les enseignantes des deux classes sur la programmation sur ScratchJr et de les accompagner au cours de la mise en place du scénario, car nous avons pu explorer une autre dimension importante de l'enseignement de l'informatique, la formation des enseignants.

Une autre contrainte liée au choix de travailler dans des conditions réelles de classes a été le fait que nous avons dû intervenir même pendant la séance d'évaluation pour faciliter la progression des élèves sous la responsabilité de l'enseignant, même si on peut penser a priori que cela pourrait gêner la posture du chercheur. Une autre raison pour laquelle nous avons choisi d'intervenir même pendant l'évaluation était l'impact positif que notre intervention pourrait avoir sur la motivation des élèves à accomplir l'épreuve. En effet, sans aide apportée de notre part, il aurait été possible que les élèves ne parviennent pas du tout à poursuivre l'évaluation, ce qui nous aurait privée de résultats. De plus, comme il s'agit d'enfant de 6 à 7

ans, le risque aurait été qu'ils perdent confiance en eux voire qu'ils perdent patience et arrêtent l'activité alors même qu'ils étaient capables de la continuer avec un peu d'aide. Nous nous sommes donc adaptée à notre public. En outre, puisque l'objectif de cette évaluation n'était pas de juger les élèves mais de juger ce qu'ils peuvent faire à un moment donné, nous avons estimé qu'il serait bénéfique de les aider même pendant l'évaluation, pour observer si grâce à l'apport d'aides ils pouvaient surmonter les difficultés rencontrées ou non. Les résultats de notre recherche montrent que nous avons pris la bonne décision.

Hormis les clarifications que ce travail de thèse apporte sur ce que les élèves de 5 à 7 ans peuvent faire seuls ou aidés en programmation sur ScratchJr et sur les difficultés qu'ils risquent d'avoir, une méthodologie d'analyse de l'activité de programmation des jeunes élèves sur un environnement de programmation graphique tel que ScratchJr est aussi proposée. Nous jugeons que cette méthodologie d'analyse pourrait être utilisée par d'autres chercheurs pour analyser l'activité de programmation des jeunes élèves sur ScratchJr mais aussi sur d'autres environnements de programmation graphique. Nous avons également pu repérer une série d'avantages et de limites du logiciel qui ont pu poser des difficultés aux élèves. En outre, les scénarios pédagogiques en grec et en français conçus dans le cadre de ce travail de thèse pour enseigner la programmation à des élèves de 5 à 7 ans, pourront également être utilisés par des enseignants.

D'autres recherches pourraient être réalisées dans la continuité de ce travail. Tout d'abord, les résultats de notre recherche par rapport aux stratégies de programmation et le débogage mis en place par les jeunes élèves pourraient servir en tant qu'hypothèses à vérifier par des recherches quantitatives avec des échantillons plus larges dans des conditions réelles de classes. Il serait également intéressant de réaliser des études similaires à la nôtre dans des contextes hors de l'école, pour examiner ce que les jeunes élèves de 5 à 7 ans pourraient faire seuls en programmation sur ScratchJr et repérer des différences potentielles.

Un autre prolongement possible pourrait être de réaliser une étude longitudinale, pendant deux ans auprès de deux classes de CP à l'aide d'un scénario pédagogique basé sur le nôtre mais avec davantage de séances. L'activité de programmation d'un petit nombre d'élèves de chaque classe pourrait être filmée pendant ce temps-là, afin de pouvoir étudier comment les stratégies de programmation utilisées par les élèves évoluent au cours du temps. Puisque nous avons observé qu'à la fin de notre scénario pédagogique les élèves présentent des

caractéristiques correspondant à des stades différents de programmation d'après Howe (1980), une autre piste possible pourrait être d'étudier les stades de programmation par lesquels passent les élèves pendant ces deux années et leurs caractéristiques et jusqu'à quel stade ils arrivent à la fin de l'étude longitudinale.

Pour poursuivre cette recherche une autre piste pourrait être d'utiliser les caractéristiques générales de la réussite et de l'échec que nous avons identifiées dans ce travail en tant qu'hypothèse à vérifier en analysant la performance, d'un plus grand nombre d'élèves en CP ayant été initié à la programmation à travers un scénario pédagogique différent que le nôtre, face à la résolution de problèmes de programmation sur ScratchJr.

Enfin, un autre prolongement possible de cette recherche pourrait être d'étudier plus profondément le débogage mis en place par les très jeunes élèves en se basant sur les résultats de notre étude. En effet, des séances d'enseignement focalisées uniquement sur le débogage pourraient être ajoutées au scénario pédagogique.

Un petit nombre d'élèves de grande section de maternelle et de CP pourraient être filmés pendant la résolution de problèmes de programmation sur ScratchJr tout au long de la mise en place du scénario pédagogique. Ainsi, le débogage que les élèves mis en oeuvre pourrait être analysé pour comprendre comment il évolue au cours de l'année et comment la façon dont ils le mettent en place change lorsqu'ils sont plus familiarisés avec les commandes enseignées dans le scénario.

Bibliographie

ACM CSTA. (2003). *A Model Curriculum for K--12 Computer Science : Final Report of the ACM K--12 Task Force Curriculum Committee* [Technical Report]. Association for Computing Machinery Computer Science Teachers Association.

ACM CSTA. (2010). *Running on empty : The failure to teach K-12 computer science in the digital age* (p. 76) [Technical Report]. Association for Computing Machinery Computer Science Teachers Association. https://runningonempty.acm.org/fullreport2.pdf

ACM CSTA. (2011). *CSTA K-12 Computer Science Standards Revised 2011* (p. 73) [Technical Report]. Association for Computing Machinery Computer Science Teachers Association. http://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf

ACM CSTA. (2017). *CSTA K-12 Computer Science Standards Revised 2017* (p. 30) [Technical Report]. Association for Computing Machinery Computer Science Teachers Association. https://drive.google.com/file/d/1-dPTAI1yk2HYPKUWZ6DqaM6aVUDa9iby/view

Armoni, M., & Ben-Ari, M. (2013). *Computer Science Concepts in Scratch | Scratch*. Weizmann Institute of Science. https://stwww1.weizmann.ac.il/scratch/scratch_en/

Arsac, J. (1988). *La didactique de l'informatique : Un problème ouvert ?*. 9. https://edutice.archives-ouvertes.fr/edutice-00359090

Australian Curriculum Assessment and Reporting Authority. (2012a). *Curriculum development process version 6* (p. 25). https://urlz.fr/lG4D

Australian Curriculum Assessment and Reporting Authority. (2012b). *The Shape of the Australian Curriculum Version 4.0* (p. 30).https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjHjpuhgfn8AhX9YKQEHfzOBqwQFnoECAsQAQ&url=https%3A%2F

%2Fwww.acara.edu.au%2F_resources%2FThe_Shape_of_the_Australian_Curriculum_v4.pdf&usg=AOvVaw1lUKCIY32z48KzreP-RmRH

Australian Curriculum, Assessment and Reporting Authority. (2015). *F-10 Curriculum : Digital Technologies at the Foundation to year 2 level*. ACARA. https://www.australiancurriculum.edu.au/f-10-curriculum/technologies/digital-technologies/

Baron, G.-L. (1987). *La constitution de l'informatique comme discipline scolaire, le cas des lycées* [Phdthesis, Université René Descartes-Paris V]. https://theses.hal.science/edutice-00000369

Baron, G.-L. (1989). *L'informatique, discipline scolaire ? (Le cas des lycées)*. PUF. http://tel.archives-ouvertes.fr/edutice-00000369

Baron, G.-L., & Bruillard, É. (1996). *L'informatique et ses usagers dans l'éducation* (Vol. 1-1). Presses universitaires de France.

Baron, G.-L., & Bruillard, É. (2001). Une didactique de l'informatique ? *Revue française de pédagogie*, *135*(1), 163-172. https://doi.org/10.3406/rfp.2001.2813

Baron, G.-L., & Bruillard, E. (2008). Technologies de l'information et de la communication et « indigènes numériques » : Quelle situation ? *Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation*, *15*(1), 19-38. https://doi.org/10.3406/stice.2008.976

Baron, G.-L., & Bruillard, É. (2011). L'informatique et son enseignement dans l'enseignement scolaire général français : Enjeux de pouvoir et de savoirs. In *Recherches et expertises pour l'enseignement scientifique: Vol. 1re éd.* (p. 79-90). De Boeck Supérieur. https://doi.org/10.3917/dbu.lebea.2011.01.0079

Baron, G.-L., & Drot-Delange, B. (2016). L'informatique comme objet d'enseignement à l'école primaire française ? Mise en perspective historique. *Revue française de pédagogie*, *195*(2), 51-62. Cairn.info.

Barr, V., & Stephenson, C. (2011). Bringing Computational Thinking to K-12 : What is Involved and What is the Role of the Computer Science Education Community? *ACM Inroads*, *2*(1), 48-54.

Beaudouin-Lafon, M. (1996). Lexique et syntaxe [notes de cours]. Département de l'informatique, Université Paris-Sud. https://www.lri.fr/~mbl/ENS/DEUG/cours/3-lexique-syntaxe.html

Ben-Ari, M. (1996). *Understanding Programming Languages* (1st éd.). John Wiley & Sons, Inc.

Bers, M. U. (2018). Coding and Computational Thinking in Early Childhood : The Impact of ScratchJr in Europe. *European Journal of STEM Education*, *3*(3). https://eric.ed.gov/?id=EJ1190774

Bers, M. U. (2019). Coding as another language : A pedagogical approach for teaching computer science in early childhood. Journal of Computers in Education, 6(4), Article 4.

Bers, M. U. (2021). *Coding as a playground : Programming and computational thinking in the early childhood classroom* (Second edition). Routledge.

Bers, M. U., & Resnick, M. (2016). *The official ScratchJr book : Help your kids learn to code!* No Starch Press.

Blaho, A., & Kalaš, I. (2020). *Programming with Emil in Lower Primary Years : Workshop Part 1 for Year 3, Part 2 for Year 4*. 1-4.https://urlz.fr/lP5Q

Bonar, J., & Soloway, E. (1985). Preprogramming Knowledge : A Major Source of Misconceptions in Novice Programmers. *Human–Computer Interaction*, *1*(2), 133-161. https://doi.org/10.1207/s15327051hci0102$_3$

Bozat, P., Bozat, N., & Hursen, C. (2014). The Evaluation of Competence Perceptions of Primary School Teachers for the Lifelong Learning Approach. *Procedia-Social and Behavioral Sciences*, *140*, 476-482. https://doi.org/10.1016/j.sbspro.2014.04.456

Brown, N. C. C., Kölling, M., Crick, T., Peyton Jones, S., Humphreys, S., & Sentance, S. (2013). Bringing computer science back into schools : Lessons from the UK.

Proceeding of the 44th ACM technical symposium on Computer science education, 269-274. https://doi.org/10.1145/2445196.2445277

Brown, N. C. C., Sentance, S., Crick, T., & Humphreys, S. (2013). Restart : The Resurgence of Computer Science in UK Schools. *ACM Trans. Comput. Educ*, *1*(1), Article 1.

Bruillard, E. (2012). *Lire-écrire-computer : Émanciper les humains, contrôler les machines*. http://www.ina-expert.com/e-dossier-de-l-audiovisuel-l-education-aux-cultures-de-l-information/l-informatique-en-education-quel-s-objet-s-d-enseignement.html

Bruillard, E. (2014). *Une voie pour penser et construire une formation à l'informatique pour les élèves de l'école primaire ?* Site du laboratoire STEF, ENS de Cachan.

Carver, S., & Klahr, D. (1986). Assessing Children's Logo Debugging Skills With A Formal Model. *Journal of Educational Computing Research*, *2*, 487-525. https://doi.org/10.2190/KRD4-YNHH-X283-3P5V

Clements, D. (1999). The Future of Educational Computing Research:The Case of Computer Programming. *Information Technology in Child Education*, *1*.

Clements, D., & Battista, M. (1989). Learning of Geometric Concepts in a Logo Environment. *Journal for Research in Mathematics Education*, *20*(5), 450. https://doi.org/10.2307/749420

Clements, D. H., & Gullo, D. F. (1984). Effects of Computer Programming on Young Children's Cognition. *American Psychological Association Inc.*, *76*(6), 1051-1058.

Clements, D., & Meredith, J. (1993). Research on Logo : Effects and Efficacy. *Journal of Computing in Childhood Education archive*. https://www.semanticscholar.org/paper/Research-on-Logo%3A-Effects-and-Efficacy.-Clements-Meredith/1eabb79c06885e55b4b792200450e5d82fa69e7a

Clements, D., & Nastasi, B. (1985). Effects of Computer Environments on Social-Emotional Development: *Computers in the Schools*, *2*, 11-31. https://doi.org/10.1300/J025v02n02_04

Cohen, L., Manion, L., & Morrison, K. (2007). *Research methods in education* (6th ed). Routledge.

Crahay, M. (1987). Logo, un environnement propice à la pensée procédurale. *Revue française de pédagogie*, *80*(1), 37-56. https://doi.org/10.3406/rfp.1987.1473

Creswell, J. W. (2007). *Qualitative inquiry & research design : Choosing among five approaches* (2nd ed). Sage Publications.

Creswell, J. W. (2012). *Educational research : Planning, conducting, and evaluating quantitative and qualitative research* (4th ed). Pearson.

Dalbey, J., & Linn, M. C. (1985). The Demands and Requirements of Computer Programming : A Literature Review. *Journal of Educational Computing Research*, *1*(3), 253-274. https://doi.org/10.2190/BC76-8479-YM0X-7FUA

Dansac, C., Baron, G.-L., & Bruillard, E. (2000). Pupils'representations of ICT: a Preliminary study in six European countries. *Proceedings of conference on educational uses of information and communication technologies : IFIP, 16th World computer congress 2000, Beijing, China, 21-25 August 2000*.

Degelman, D., Free, J. U., Scarlato, M., Blackburn, J. M., & Golden, T. (1986). Concept Learning in Preschool Children : Effects of a Short-Term Logo Experience. *Journal of Educational Computing Research*, *2*(2), 199-205. https://doi.org/10.2190/RH2K-4AQ7-2598-TVEA

Denning, P. J. (2009). The profession of IT Beyond computational thinking. *Communications of the ACM*, *52*(6), Article 6.

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33-39. https://doi.org/10.1145/2998438

Department for Education. (2013). *The National Curriculum in England. Key Stages 1 and 2 framework document* (p. 201).

Dowek, G. (2011). *Les quatre concepts de l'informatique*. 21. https://edutice.archives-ouvertes.fr/edutice-00676169

Drot-Delange, B., & Bruillard, E. (2012). Éducation aux TIC, cultures informatiques et du numérique : Quelques repères historiques. *Études de communication-Langages, information, médiations*, *38*, 69-80.

Drot-Delange, B., Pellet, J.-P., Delmas-Rigoutsos, Y., & Bruillard, E. (2019). *Pensée informatique : Points de vue contrastés*. *26*(1). https://doi.org/10.23709/STICEF.26.1.1

Dufoyer, J.-P. (1988). *Informatique, éducation et psychologie de l'enfant : Jean-Pierre Dufoyer* (1. éd). Presses universitaires de France.

Duncan, C., Bell, T., & Tanimoto, S. (2014). Should Your 8-year-old Learn Coding? *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, 60-69. https://doi.org/10.1145/2670757.2670774

European Commission. (2018). *Communication from the commission to the european parliament, the council, the european economic and social committee and the committe of the regions on the Digital Education Action Plan* (22 final). https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2018:22:FIN

European Commission. Joint Research Centre. (2016). *Developing computational thinking in compulsory education : Implications for policy and practice.* Publications Office of the European Union. https://data.europa.eu/doi/10.2791/792158

European Commission. Joint Research Centre. (2022). *Reviewing computational thinking in compulsory education : State of play and practices from computing education.* Publications Office. https://data.europa.eu/doi/10.2760/126955

European Schoolnet. (2015). *Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across europe*. https://urlz.fr/lP3T

European Schoolnet, & National Research Council of Italy, Institute for Educational Technology. (2018). *The nordic approach to introducing computational thinking and programming in compulsory education* (p. 42). CNR Edizioni. https://doi.org/10.17471/54007

Fabius, L. (1985). *L'informatique pour tous*. *37*, 23-30.

Falkner, K., & Vivian, R. (2015). A review of Computer Science resources for learning and teaching with K-12 computing curricula : An Australian case study. *Computer Science Education*, *25*(4), 390-429. https://doi.org/10.1080/08993408.2016.1140410

Falkner, K., Vivian, R., & Falkner, N. (2014). The Australian Digital Technologies Curriculum : Challenge and Opportunity. In J. Whalley & D. D'Souza (Éds.), *Proceedings of the 16th Australasian Computing Education Conference (ACE2014)* (Vol. 148, p. 11).

Fatourou, E., Zygouris, N. C., Loukopoulos, T., & Stamoulis, G. I. (2018). Teaching Concurrent Programming Concepts Using Scratch in Primary School : Methodology and Evaluation. *International Journal of Engineering Pedagogy (iJEP)*, *8*(4), 89. https://doi.org/10.3991/ijep.v8i4.8216

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem Solving by 5-6 Years Old Kindergarten Children in a Computer Programming Environment : A Case Study. *Comput. Educ.*, *63*, 87-97. https://doi.org/10.1016/j.compedu.2012.11.016

Fay, A. M., & Mayer, R. E. (1988). Learning LOGO : A cognitive analysis. In R. E. Mayer (Éd.), *Teaching and Learning Computer Programming : Multiple Research Perspectives* (p. 320). L. Erlbaum Associates.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem Solving by 5-6 Years Old Kindergarten Children in a Computer Programming Environment : A Case Study. *Comput. Educ.*, *63*, 87-97. https://doi.org/10.1016/j.compedu.2012.11.016

Feurzeig, W. (2010). Toward a Culture of Creativity : A Personal Perspective on Logo's Early Years and Ongoing Potential. *International Journal of Computers for Mathematical Learning*, *15*(3), 257-265. https://doi.org/10.1007/s10758-010-9168-4

Feurzeig, W., & Papert, S. A. (2011). Programming-Languages as a Conceptual Framework for Teaching Mathematics. *Interactive Learning Environments*, *19*(5), 487-501. https://doi.org/10.1080/10494820903520040

Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Solomon, C. (1969). *Programming-languages as a conceptual framework for teaching mathematics. Final Report on the First Fifteen Months of the LOGO Project.* (R-1889; Numéro R-1889). Bolt Beranek and Newman, Inc., Catridge, Mass.

Flannery, L. P., & Bers, M. U. (2013). Let's Dance the "Robot Hokey-Pokey!" *Journal of Research on Technology in Education*, *46*(1), Article 1. https://doi.org/10.1080/15391523.2013.10782614

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013). Designing ScratchJr : Support for Early Childhood Learning Through Computer Programming. *Proceedings of the 12th International Conference on Interaction Design and Children*, 1-10. https://doi.org/10.1145/2485760.2485785

Franklin, D., Skifstad, G., Rolock, R., Mehrotra, I., Ding, V., Hansen, A., Weintrop, D., & Harlow, D. (2017). Using Upper-Elementary Student Performance to Understand Conceptual Sequencing in a Blocks-based Curriculum. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 231-236. https://doi.org/10.1145/3017680.3017760

Greff, É. (2000). *Résolution de problèmes en GS : Autour des pivotements à l'aide du robot de plancher*. *68*, 7-16.

Grover, S., & Pea, R. (2018). Computational thinking : A competency whose time has come. In S. Sentance, E. Barendsen, & C. Schulte (Éds.), *Computer Science Education : Perspectives on Teaching and Learning in School*. Bloomsbury Publishing.

Heintz, F., Mannila, L., & Färnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education. *2016 IEEE Frontiers in Education Conference (FIE)*, 1-9. https://doi.org/10.1109/FIE.2016.7757410

Heintz, F., Mannila, L., Nygårds, K., Parnes, P., & Regnell, B. (2015). *Computing at School in Sweden – Experiences from Introducing Computer Science within Existing Subjects*. 118-130. https://doi.org/10.1007/978-3-319-25396-1_11

Howe, J. A. M. (1980). Developmental Stages in Learning to Program. In F. Klix & J. Hoffmann (Éds.), *Advances in Psychology* (Vol. 5, p. 253-263). North-Holland. https://doi.org/10.1016/S0166-4115(08)62461-9

Informatics Europe & ACM Europe Working Group. (2013). *Informatics education : Europe cannot afford to miss the boat* [Report of the joint Informatics Europe & ACM Working Group on Informatics Education]. https://urlz.fr/lP46

Institut de France. Académie des Sciences. (2013). *L'enseignement de l'informatique en France. Il est urgent de ne plus attendre.* (p. 34 pages). https://www.academie-sciences.fr/fr/Rapports-ouvrages-avis-et-recommandations-de-l-Academie/l-enseignement-de-l-informatique-en-france-il-est-urgent-de-ne-plus-attendre.html

Istitut de France & Académie des sciences. (2013). *L'enseignement de l'informatique en France, Il est urgent de ne plus attendre*. http://www.academie-sciences.fr/activite/rapport/rads_0513.pdf

K-12 Computer Science Framework. (2016). *K-12 Computer Science Framework*. https://k12cs.org/

Kabátová, M., Kalaš, I., & Tomcsányiová, M. (2016). Programming in Slovak Primary Schools. *Olympiads in Informatics*, *10*, 125-159. https://doi.org/10.15388/ioi.2016.09

Kafai, Y. B., & Burke, Q. (2013). Computer Programming Goes Back to School. *Phi Delta Kappan*, *95*(1), 61-65. https://doi.org/10.1177/003172171309500111

Kapa, E. (1999). Problem solving, planning ability and sharing processes with LOGO. *Journal of Computer Assisted Learning*, *15*(1), 73-84. https://doi.org/10.1046/j.1365-2729.1999.151077.x

Kazakoff, E. R. (2014). *Cats in Space, Pigs that Race : Does self-regulation play a role when kindergartners learn to code?* [PhD Thesis]. Tufts University.

Kazakoff, E. R., & Bers, M. U. (2014). Put Your Robot in, Put Your Robot out : Sequencing through Programming Robots in Early Childhood. *Journal of Educational Computing Research*, *50*(4), 553-573. https://doi.org/10.2190/EC.50.4.f

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The Effect of a Classroom-Based Intensive Robotics and Programming Workshop on Sequencing Ability in Early Childhood. *Early Childhood Education Journal*, *41*(4), 245-255. https://doi.org/10.1007/s10643-012-0554-5

Klahr, D., & Carver, S. M. (1988). Cognitive objectives in a LOGO debugging curriculum : Instruction, learning, and transfer. *Cognitive Psychology*, *20*(3), 362-404. https://doi.org/10.1016/0010-0285(88)90004-7

Kelly, B. (2012, avril 27). *What STEM Is—And Why We Care*. https://www.usnews.com/news/blogs/stem-education/2012/04/27/what-stem-is--and-why-we-care

Komis, V., & Misirli, A. (2013). *Étude des processus de construction d'algorithmes et de programmes par les petits enfants à l'aide de jouets programmables*. Sciences et technologies de l'information et de la communication (STIC) en milieu éducatif. https://edutice.archives-ouvertes.fr/edutice-00875628

Komis, V., Romero, M., & Misirli, A. (2017). A Scenario-Based Approach for Designing Educational Robotics Activities for Co-creative Problem Solving. In D. Alimisis, M. Moro, & E. Menegatti (Éds.), *Educational Robotics in the Makers Era* (p. 158-169). Springer International Publishing. https://doi.org/10.1007/978-3-319-55553-9_12

Komis, V., Tzavara, A., Karsenti, T., Collin, S., & Simard, S. (2013). Educational scenarios with ICT : An operational design and implementation framework. *Proceedings of Society for Information Technology & Teacher Education International Conference*, 3244-3251.

Kurhila, J., & Vihavainen, A. (2015). A Purposeful MOOC to Alleviate Insufficient CS Education in Finnish Schools. *ACM Transactions on Computing Education*, *15*(2), 1-18. https://doi.org/10.1145/2716314

Kurland, D. M., & Pea, R. D. (1985). Children's Mental Models of Recursive Logo Programs. *Journal of Educational Computing Research*, *1*(2), 235-243. https://doi.org/10.2190/JV9Y-5PD0-MX22-9J4Y

Le Douarin, L. (2004). Hommes, femmes et micro-ordinateur : Une idéologie des compétences. *Réseaux*, *123*(1), 149-174.

Léonard, M., Peter, Y., & Secq, Y. (2020, février 5). *Reconnaissance et synthèse de motifs redondants avec des élèves de 6-7 ans MOTIFS.MOTIFS.MOTIFS. ⟺ 3 x MOTIFS. 3 x MOTIFS.* Colloque DIDAPRO 8 -DIDASTIC - L'informatique, objets

d'enseignements enjeux épistémologiques, didactiques et de formation. https://hal.univ-lille.fr/hal-02971775

Léonard, M., Peter, Y., Secq, Y., Alvarez, J., & Fluckiger, C. (2021). MOTIF..MOTIF.. : Initier à la notion de « répétition » en maternelle sans mobiliser de repérage spatial. *Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation*, *28*(3), 39-69. https://doi.org/10.23709/sticef.28.3.4

Liao, Y., & Bright, G. (1991). Effects of Computer Programming on Cognitive Outcomes : A Meta-Analysis. *Journal of Educational Computing Research*, *7*, 251-266. https://doi.org/10.2190/E53G-HH8K-AJRR-K69M

Lodico, M. G., Spaulding, D. T., & Voegtle, K. H. (2006). *Methods in educational research : From theory to practice* (1st ed). Jossey-Bass

Lowenthal, F., Marcourt, C., & Solimando, C. (1998). Cognitive strategies observed during problem solving with LOGO. *Journal of Computer Assisted Learning*, *14*(2), 130-139. https://doi.org/10.1046/j.1365-2729.1998.1420130.x

Luehrmann, A. W. (1972). Should the computer teach the student, or vice versa? *Proceedings of the May 16-18, 1972, spring joint computer conference*, 407-410. https://doi.org/10.1145/1478873.1478925

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming : What is next for K-12? *Computers in Human Behavior*, *41*, 51-61. https://doi.org/10.1016/j.chb.2014.09.012

Maloney, J., Peppler, K., Resnick, M., & Rusk, N. (2008). Programming by choice : Urban Youth Learning Programming with Scratch. *ACM SIGCSE Bulletin*, *40 (1)*, 367-371. https://doi.org/10.1145/1352322.1352260

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education*, *10*(4), 1-15. https://doi.org/10.1145/1868358.1868363

Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse : Women in computing*. MIT Press.

Mayer, R. E., & Fay, A. L. (1987). A chain of cognitive changes with learning to program in Logo. *Journal of Educational Psychology*, *79*, 269-279. https://doi.org/10.1037/0022-0663.79.3.26

Mayer, R. (1988). Introduction to Research on Teachning and Learning Computer Programming. In Teaching and learning computer programming : Multiple research perspectives (p. 320). L. Erlbaum Associates.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2010). Learning Computer Science Concepts with Scratch. *Proceedings of the Sixth International Workshop on Computing Education Research*, *23*, 69-76.

Mendelsohn, P. (1985). L'analyse psychologique des activités de programmation chez l'enfant de CMI et CM2. *Enfance*, *38*(2), 213-221. https://doi.org/10.3406/enfan.1985.2881

Mendelsohn, P., Green, T. R. G., & Brna, P. (1990). Programming languages in education : The search for an easy start. In J.-M. Hoc, T. R. G. Green, R. Samurçay, & D. J. Gilmore (Éds.), *Psychology of programming* (p. 175-200). Academic Press.

Merriam, S. B. (2002). *Qualitative research in practice : Examples for discussion and analysis* (1st ed). Jossey-Bass.

Merriam, S. B., & Tisdell, E. J. (2015). *Qualitative research : A guide to design and implementation* (Fourth edition). John Wiley & Sons.

Miles, M. B., Huberman, A. M., & Saldaña, J. (2014). *Qualitative data analysis : A methods sourcebook* (Third edition). SAGE Publications, Inc.

Papert, S. (1980). *Mindstorms : Children, computers, and powerful ideas*. Basic Books.

Papert, S. (1989). Teachning children thinking. In E. Soloway & J. Spohrer (Éds.), *Studying the Novice programmer* (p. 29-40). Lawrence Erlbaum Associate, Publishers.

Papert, S., Watt, D., DiSessa, A., & Weir, S. (1979). *The Final Report of the Brookline Project Part II : Project Summary and Data Analysis* (Logo Memo No 53; p. 222). Massachusetts Institute of Technology. Artifficial Intelligence Laboratory.

Pea, R. D. (1983). Logo Programming and Problem solving. *Proceedings of the American Educational Research Association Symposium*, 9.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, *2(2)*, 137.

Pekárová, J. (2008). Using a Programmable Toy at Preschool Age : Why and How ? *Workshop Proceedings of SIMPAR 2008*, 112-121.

Peter, Y., Léonard, M., & Secq, Y. (2019, juin 4). Reconnaissance de Motifs et Répétitions : Introduction à la Pensée Informatique. *Actes du Colloque Environnements Informatiques pour l'Apprentissage Humain*. Environnements Informatiques pour l'Apprentissage Humain. https://hal.science/hal-02151035

Piaget, J. (1981). Le développement de la notion de temps chez l'enfant (3e édition). Presses Universitaires de France-PUF.

Piaget, J., & Inhelder, B. (2012). *La psychologie de l'enfant* (3e éd). PUF.

Pila, S., Aladé, F., Sheehan, K. J., Lauricella, A. R., & Wartella, E. A. (2019). Learning to code via tablet applications : An evaluation of Daisy the Dinosaur and Kodable as learning tools for young children. *Computers & Education*, *128*, 52-62. https://doi.org/10.1016/j.compedu.2018.09.006

Polya, G. (1965). *Comment poser et resoudre un problème* (2e edition). Jacques Gabay.

Portelance, D. J. (2015). *Code and Tell : An exploration of peer interviews and computational thinking with ScratchJr in the early childhood classroom* [Tufts University]. https://ase.tufts.edu/DevTech/resources/Theses/DPortelance_2015.pdf

Portelance, D. J., Strawhacker, A. L., & Bers, M. U. (2015). Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education*, 1-16. https://doi.org/10.1007/s10798-015-9325-0

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., & Silverman, B. (2009). Scratch : Programming for all. *Communications of the ACM*, *52*(11), Article 11.

Rinaudo, J.-L. (2017). Donc il y aurait à enseigner l'informatique à l'école…. In G.-L. Baron, J. Béziat, & F. Villemonteix (Éds.), *L'école primaire et les technologies*

informatisées : Des enseignants face aux TICE (p. 39-44). Presses universitaires du Septentrion. http://books.openedition.org/septentrion/15143

Robert, F. (1985). L'utilisation de l'ordinateur dans l'enseignement primaire : L'exemple de la France. *Enfance*, *38*(1), 19-30. https://doi.org/10.3406/enfan.1985.2857

Romero, M., Lille, B., Viéville, T., Duflot-Kremer, M., Smet, C. D., & Belhassein, D. (2018, août 27). *Analyse comparative d'une activité d'apprentissage de la programmation en mode branché et débranché*. Educode-Conférence internationale sur l'enseignement au numérique et par le numérique. https://hal.inria.fr/hal-01861732

Rushkoff, D. (2010). *Program or be programmed : Ten commands for a digital age*. OR Books.

Sabitzer, B., Antonitsch, P. K., & Pasterk, S. (2014). Informatics Concepts for Primary Education : Preparing Children for Computational Thinking. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, 108-111. https://doi.org/10.1145/2670757.2670778

Spach, M. (2017). *Activités robotiques à l'école primaire et apprentissage de concepts informatiques. Quelle place du scénario pédagogique ? Les limites du co-apprentissage*. Université Paris Descartes & Université de Lille 3.

Stake, R. (2003). Case studies. In N. K. Denzin & Y. S. Lincoln (Éds.), *Strategies of Qualitative Inquiry* (2nd éd., p. 134-164). SAGE Publications.

Strawhacker, A., & Bers, M. U. (2015). "I want my robot to look for food" : Comparing Kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces. *International Journal of Technology and Design Education*, *25*(3), 293-319. https://doi.org/10.1007/s10798-014-9287-7

Strawhacker, A., & Bers, M. U. (2019). What they learn when they learn coding : Investigating cognitive domains and computer programming knowledge in young children. *Educational Technology Research and Development*, *67*(3), 541-575. https://doi.org/10.1007/s11423-018-9622-x

Strawhacker, A., Lee, M., & Bers, M. U. (2017). Teaching tools, teachers' rules : Exploring the impact of teaching styles on young children's programming knowledge in ScratchJr. *International Journal of Technology and Design Education*. https://doi.org/10.1007/s10798-017-9400-9

Tartas, V. (2010). Le développement de notions temporelles par l'enfant. *Développements*, *4*(1), 17-26. https://doi.org/10.3917/devel.004.0017

The Royal Society. (2012). *Shut down or restart ? The way forward for computing in UK schools* (p. 121). http://royalsociety.org/uploadedFiles/Royal_Society_Content/education/policy/computing-in-schools/2012-01-12-Computing-in-Schools.pdf

Touloupaki, S., Baron, G.-L., & Komis, V. (2018). *Un apprentissage de la programmation dès l'école primaire : Le concept de message sur ScratchJr*. 303-323. https://hal.science/hal-01753125

Tsourapi, C., Komis, V., & Baron, G.-L. (2018). L'étayage des enseignants de l'école maternelle au cours des activités de programmation avec le logiciel ScratchJr. In G. Parriaux, J.-P. Pellet, G.-L. Baron, E. Bruillard, & V. Komis (Éds.), *De 0 à 1 ou l'heure de l'infromatique à l'école. Actes du colloque Didapro 7-DidaSTIC* (p. 291-302). Peter Lang AG.

U.S. Departement of Education, Office of Educational Technology. (2010). *Transforming American Education : Learning powered by technology* (p. 124). https://urlz.fr/lP5b

Vandeput, É. (2017). Une didactique de l'informatique...Une didactique des STIC. In *L'informatique et le numérique dans la classe : Qui, quoi, comment ?* (p. 37-48). Presses Universitaires de Namur.

Vandevelde, I., & Fluckiger, C. (2020, février 5). *L'informatique prescrite à l'école primaire. Analyse de programmes, ouvrages d'enseignement et discours institutionnels*. Colloque Didapro-Didastic 8. https://hal.univ-lille.fr/hal-02462385

Vigot, N. (2018). *Enseigner l'informatique à l'école élémentaire au cycle deux. Une expérimentation avec Scratch Junior*. *113*, 31-44.

Vivian, R., Falkner, K., & Falkner, N. (2014). Addressing the challenges of a new digital technologies curriculum : MOOCs as a scalable solution for teacher professional development. *Research in Learning Technology*, *22*. https://doi.org/10.3402/rlt.v22.24691

Vivian, R., Falkner, K., & Szabo, C. (2014). Can everybody learn to code? Computer science community perceptions about learning the fundamentals of programming. *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, 41-50. https://doi.org/10.1145/2674683.2674695

Vygotsky, L. (1978). Mind in Society—Development of Higher Psychological Processes (New Ed). Harvard University Press.

Yin, R. K. (2003). *Case Study Research : Design and Methods* (3rd Revised edition). SAGE Publications Ltd.

Wilson, A., & Moffat, D. C. (2010). *Evaluating Scratch to introduce younger schoolchildren to programming*. http://scratched.media.mit.edu/sites/default/files/wilson-moffat-ppig2010-final.pdf

Wing, J. (2006). Computational thinking. *Commun. ACM*, *49*(3), Article 3. https://doi.org/10.1145/1118178.1118215

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, *366*(1881), Article 1881. https://doi.org/10.1098/rsta.2008.0118

Wing, J. (2017). Computational Thinking's Influence on Research and Education for All. *Italian Journal of Educational Technology*, *1*(1). https://doi.org/10.17471/2499-4324/922

Wyeth, P. (2008). How Young Children Learn to Program with Sensor, Action, and Logic Blocks. *The Journal of the Learning Sciences*, *17*(4), 517-550.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational Thinking in Elementary and Secondary Teacher Education. *ACM Transactions on Computing Education*, *14*(1), 5:1-5:16. https://doi.org/10.1145/2576872

Yelland, N. (1995). Mindstorms or a storm in a teacup ? A review of research with Logo. *International Journal of Mathematical Education in Science and Technology*, *26*(6), 853-869. https://doi.org/10.1080/0020739950260607

Διαθεματικό Ενιαίο Πλαίσιο Προγραμμάτων Σπουδών για το Νηπιαγωγείο. (2003) (p. 30).

Διαθεματικό Ενιαίο Πλαίσιο Προγράμματος Σπουδών Πληροφορικής. (2003) (p. 17).

Ινστιτούτο Εκπαιδευτικής Πολιτικής. (2021a). *Πρόγραμμα σπουδών για την προσχολική εκπαίδευση* (Πρώτη Εκδοση; p. 109).

Ινστιτούτο Εκπαιδευτικής Πολιτικής. (2021b). *Πρόγραμμα Σπουδών για το Μάθημα Τεχνολογίες της Πληροφορίας και της Επικοινωνίας (ΤΠΕ) και Πληροφορικής Δημοτικού* (Πρώτη Εκδοση; p. 45).

Παιδαγωγικό Ινστιτούτο. (2011a). *Πρόγραμμα σπουδών για τις Τεχνολογίες Πληροφορίας και Επικοινωνιών (ΤΠΕ) στην Πρωτοβάθμια Εκπαίδευση*.

Παιδαγωγικό Ινστιτούτο. (2011b). *Πρόγραμμα Σπουδών Νηπιαγωγείου*.

Υπουργείο Εθνικής Παιδείας και Θρησκευμάτων. Παιδαγωγικό Ινστιτούτο. (1997). *Ενιαίο Πλαίσιο Προγράμματος Σπουδών Πληροφορικής*. (p. 13).

Παιδαγωγικό Ινστιτούτο. (2011a). *Πρόγραμμα σπουδών για τις Τεχνολογίες Πληροφορίας και Επικοινωνιών (ΤΠΕ) στην Πρωτοβάθμια Εκπαίδευση* .

Παιδαγωγικό Ινστιτούτο. (2011b). *Πρόγραμμα Σπουδών Νηπιαγωγείου*.

Table des matières

Index des figures

Index des tableaux

Université Paris Cité

En cotutelle avec Université de Patras

École doctorale ED623 Savoirs, Sciences, Education

Laboratoire EDA-Éducation Discours et Apprentissages (EA 4071)

Contribution à l'étude de l'apprentissage de la programmation en grande section et en cours préparatoire, à travers le logiciel ScratchJr : Une approche didactique exploratoire

Par **Sevastiani Touloupaki**

Thèse de doctorat de Sciences de l'Éducation et de la Formation

Dirigée par **Georges-Louis Baron**

Et par **Vassilis Komis**

Présentée et soutenue publiquement le 12 juillet 2023

Volume 2 : Annexes

Devant un jury composé de :

Georges-Louis Baron, professeur émérite, Université Paris Cité (Directeur)

Marina Bers, professor, Boston College Chestnutt Hill, MA (Examinatrice)

Eric Bruillard, professeur, Université Paris Cité (Examinateur)

Angelique Dimitracopoulou, professor, University of the Aegean (Rapportrice)

Béatrice Drot-Delange, professeure, Université Clermont Auvergne (Examinatrice)

Vassilis Komis, professor, University of Patras (Co-Directeur)

Konstantinos Ravanis, professor, University of Patras (Examinateur)

Margarida Romero, professeure, Université Côte d'Azur (Rapportrice)

This research is co-financed by Greece and the European Union (European Social Fund-ESF) through the Operational Programme «Human Resources Development, Education and Lifelong Learning» in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research – 2nd Cycle" (MIS-5000432), implemented by the State Scholarships Foundation (IKY).

Sommaire

Annexe 1 : Incidents critiques issus de la 6e séance d'enseignement de « messages »

6G2AGS2GOPRO0129

00 :03 :49 :0-00 :04 :19 :7
ST: La « fusée » a un message ouvert ? qu'est-ce qui se passe alors ?
GS2E7: A! Le « chat » a envoyé un message à la « fusée ».
ST : Et le « chat » lui a dit quoi ?
GS2E7 : Il lui a dit de commencer.

00 : 04 :21 :1- 00 :04 :45 :0
ST : Lorsqu'on avait deux personnages combien des drapeaux-verts on avait ?
GS2E7 : Une.
ST : Deux.

00 :06 :22 :0 - 00 :06 :40 :1
ST : Et maintenant cliquez sur le drapeau-vert pour voir qu'est-ce qui va se passer pour la « fusée ».
GS2E7 : Maitresse la « fusée » n'a pas de drapeau-vert sur son programme.
ST : Je parle du drapeau vert en haut de la scène.
GS2E7 : A ! Ok !

00 :06 :39 :6 – 00 :06 :42 :7
ST : Pourquoi la « fusée » ne bouge pas ?
GS2E7 : Parce qu'on n'a pas mis de début.
GS2E8 : On n'a pas mis le message.

00 :08 :19 :4 – 00 :08 :46 :0
ST : Qui va alors commencer en premier et pourquoi ?
GS2E7 : Le « chat » va commencer en premier.
ST : Pourquoi ?
GS2E7 : Parce qu'il a un début et une fin ?
ST : Et la « fusée » n'a pas un début et une fin ?
GS2E7 regarde le programme de la « fusée » et il répond : La « fusée » n'a pas de début et de fin.
ST : Et le message ouvert c'est quoi ? Ce n'est pas un début ?
GS2E7 : A oui !

00 :09 :12 :9 - 00 :09 :16 :1
ST : Cliquez sur le drapeau-vert en haut de la scène pour voir qu'est-ce qui va se passer.
GS2E7 et GS2E6 et GS2E8 disent : Seul le « chat » commence.
ST : Pourquoi la « fusée » ne commence pas maintenant ?
GS2E6 : Parce que le « chat » ne lui a pas envoyé le message du moment où on l'a enlevé de son programme.

00 :09 :36 :8 - 00 :09 :38 :2
GS2E7 donne un exemple : Disons que j'ai écrit un message pour GS2E8, mais je l'ai pas envoyé.

GS2E8 reprend l'exemple : Disons qu'elle avait oublié de me l'envoyer. Comment pourrais-je le lire ? C'est impossible !

00 :10 :04 :4 – 00 :10 :18 :0
ST leur demande d'ajouter de nouveau la commande « envoyer un message » dans le programme du « chat » et de cliquer sur le drapeau-vert en haut de la scène pour regarder le résultat.
GS2E7 de nouveau confond la commande drapeau-vert avec le bouton du drapeau-vert en haut de la scène.

00 :10 :43 :7 - 00 :10 :58 :9
ST : Du coup le démarrage de la « fusée » dépend de qui ?
GS2E7 : Du « chat » !
ST : Exactement !

00 :13 :04 :4 – 00 :13 :21 :2
ST : qu'est-ce qui se passe ici ?
GS2E6 : D'abord l' « enfant » et puis le « ballon ».
ST : Vous êtes d'accord avec GS2E6 ?
GS2E7 : Oui parce que l' « enfant » cours et il donne un coup au « ballon » et le « ballon » commence.

00 :13 :21 :9-00 :13 :34 :0
ST : Du coup, pourquoi commence le « ballon » ?
GS2E7 : Parce qu'on lui a mis un début et une fin.
ST : Lorsque tu parles du début tu parles de quoi ?
GS2E7 : Du drapeau vert.

00 :14 :45 :3 – 00 :14 :45 :4
GS2E7 : Mais maitresse comment ? Le « ballon » ne peut pas lire !
ST : Exactement. Mais ces messages-là sont des commandes, elles ne sont pas des vrais messages. Le « chat » ne peut pas écrire un message.
GS2E6 : Oui le « chat » ne peut pas écrire à ma mère par exemple ni le « ballon ».
GS2E7 : Oui haha.
ST : Les commandes disent aux personnages qu'est-ce qu'ils doivent faire.
GS2E7 : Aaaa !

00 :17 :03 :2 – 00 :17 :14 :3
GS2E6 réalise le programme du « ballon » à l'aide des cartes.
GS2E6 : Mais au début il n'y aura pas le drapeau vert. Il y aura un message.
ST : Pourquoi GS2E6 ?
GS2E6 : Parce que s'il avait le drapeau vert, il commencerait en même temps que l' « enfant ».
ST : Bravo GS2E6.

00 :17 :24 :7 – 00 :17 :36 :1
ST : Et si on voulait que le « ballon » commence en premier et l' « enfant » après qu'est-ce qu'on devait faire ?

GS2E8 : Alors il fallait qu'on ajoute le drapeau vert au début du « ballon ».
GS2E6 : Et cette fois-ci ça sera le « ballon » qui envoie le message.

00 :17 :51 :8-00 :19 :00 :3
GS2E7 : Maitresse mais le « ballon » a plus de pièces.
ST : Et alors ? Tu veux dire quoi.
GS2E7 : Le « ballon » fait plus de choses.
ST : Oui et alors ?
GS2E7 : Elle commence plus.
ST : Tu veux dire qu'elle commence plutôt ?
GS2E7 : Oui !

00 :19 :03 :0-00 :19 :46 :2
ST : Pour éclairer cela, GS2E7 qu'est-ce que tu dois faire pour déclencher un programme ?
GS2E7 : Je dois commencer par le début et la fin ?
ST : qu'est-ce que tu dois faire pour lancer ton programme final ? Quelle icône tu utilises ?
GS2E7 : Aa le drapeau vert en haut.
ST : Super ! Du coup, si on déclenche nos programmes en utilisant le bouton du drapeau vert en haut de la scène les personnages qui ont le drapeau vert au début commencent leurs programmes en premiers. Du coup, le personnage qui a le drapeau vert au début commence on programme toujours d'abord.

6G2BGS2GP010129

00 :00 :08 :4 – 00 :00 :21 :3
ST : Et après qu'est-ce qu'il fait ?
GS2E6 : Il envoie le message au « ballon » et il lui dit, quand j'aurai fini, tu commences.

00 :00 :39 :0 – 00 :00 :58 :0
ST : Si j'enlève la commande « quand message reçu » par le programme du « ballon », qu'est-ce qui va se passer ?
GS2E6 : L' « enfant » sera le seul qui commence parce que le « ballon » n'a pas lu le message de l' « enfant ».

00 :01 :55 :6 – 00 :02 :10 :0
ST : Le « ballon » commencera maintenant ?
GS2E7 : Oui elle va commencer en premier, parce qu'on lui a mis le drapeau vert.

00 :07 :54 :3 – 00 :08 :16 :8
GS2E7 : Maitresse GS2E5 n'a pas mis le début et la fin. Je les ai mis.

00 :11 :12 :3-00 :11 :45 :0
GS2E7 : Mais maitresse pourquoi sur mon programme le « ballon » commence en même temps que l' « enfant » ? J'avais mis à tous les deux un début et une fin. Qu'est-ce qui se passe ?
GS2E8 : Tu avais mis le mauvais début pour le « ballon ». Tu avais mis le drapeau vert.

00 :14 :24 :2-00 :14 :38 :4

GS2E7 : Maitresse je me suis trompé. Voilà je l'ai fixé.
GS2E7 enlève une de ces commandes de mouvement par le programme du « ballon ».

00 :14 :34 :6-00 :14 :53 :2
GS2E8 corrige GS2E7.
GS2E8 : Tu avais mis quatre ici. Il fallait mettre trois.
GS2E7 : Désolée. Je me suis trompé.

6G3AGS2GOPRO0128

00 :02 :55 :7-00 :03 :01 :5
ST : qu'est-ce qu'on a appris jusqu'à maintenant ? Les deux personnages commençaient ensemble ou l'un après l'autre ?
Élèves : L'un après l'autre.

00 :02 :57 :2-00 :03 :09 :0
ST : Jusqu'à maintenant les personnages avec lesquelles on avait travaillé commençaient leurs programmes en même temps ou l'un après l'autre ?
GS2E9, GS2E10, GS2E11 et GS2E12 : L'un après l'autre.

00 :05 :40 :9-00 :06 :40 :6
ST : qu'est-ce qu'on fait lorsqu'on a une lettre comme ça ?
GS2E11 : On le ferme pour qu'on l'envoie.
GS2E12 : On le ferme pour qu'on l'envoie par la poste.

00 :06 :34 :5-00 :06 :52 :9
ST : qu'est-ce qu'on voit au début du programme de la « fusée » ?
GS2E11 : Cela au lieu de trouver le drapeau-vert.

00 :07 :58 :4-00 :08 :39 :5
ST : Cela veut dire que la « fusée » lit le message du « chat » ?
GS2E10 : Non. Il ne peut pas le lire la « fusée ».
ST : Pourquoi ?
GS2E10 : Non. Parce que c'est une « fusée ».
GS2E11 : Quelqu'un ouvrira la fenêtre de la « fusée »…
GS2E12 : Il va lire le message par la fenêtre de la « fusée ».

00 :14 :40 :3-00 :14 :41 :4
ST : Est-ce que l' « enfant » touche le « ballon » ?
GS2E12 : Non.
ST : Du coup, comment commence le « ballon » ?
GS2E10 : À l'aide des commandes.

00 :19 :28 :6 – 00 :19 :38 :7
ST : Que fait alors le « ballon » lorsqu'il reçoit le message ?
GS2E10 : Il le met dans sa poche.

6G3BGS2GP010128

00 :00 :55 :4-00 :02 :57 :2
ST : Lorsque la commande « quand message reçu » n'existe pas dans le programme du « ballon », alors le « ballon » commence ?
GS2E12 : Non.
ST : Du coup à quoi ça sert la commande « quand message reçu »?
GS2E12 : Il lui dit de commencer.

00 :02 :36 :2-00 :02 :57 :3

ST : Qui dit au ballon quand il doit commencer ?
GS2E10 : Le « chat ».
ST : Nous avons un « chat » ici ?
GS2E10 : Non c'est le message.
ST : Qui envoie le message au ballon ici ?
GS2E10 : L' « enfant ».

00 :11 :06 :3-00 :11 :27 :5

ST : Qu'est-ce qu'on doit faire pour que les deux personnages déclenchent leurs programmes en même temps et pas l'un après l'autre ? Qu'est-ce qu'on doit faire pour que le « ballon » commence en même temps que l' « enfant » ?
GS2E12 : On doit lui donner le message.

Annexe 2: Post-test Επανάληψη

Ερώτηση 1: Τι θα συμβεί αν πατήσουμε αυτή την εντολή;

Απαντήσεις :

GS2E1 : « Δεν ξέρω. »

GS2E2 : « Όχι φυσικά γιατί δεν είχαμε βάλει τίποτα μέσα.»

GS2E3 : « Όχι γιατί είναι σκέτη. »

GS2E4 : « Τίποτα γιατί αν βάλουμε κάτι από αυτά εδώ θα γίνει. »

GS2E5 : « Δεν ξέρω.»

GS2E6 : « Δεν ξέρω. »

GS2E7 : « Δεν θα κουνηθεί καθόλου γιατί δεν είχαμε βάλει κάτι μέσα.»

GS2E8 : « Όχι γιατί δεν έχουμε βάλει εντολές μέσα.»

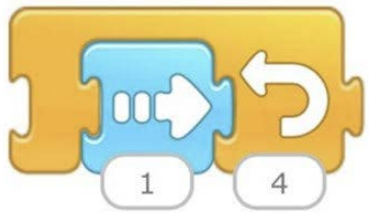
GS2E9 : « Τίποτα γιατί δεν έχουμε βάλει κάτι μέσα.»

GS2E10 : « Τίποτα γιατί δεν έχουμε βάλει κάτι μέσα.»

GS2E11 : « Ας πούμε ότι ο γάτος θα ειναι από εκεί και θα γυρίσει προς την άλλη πλευρά 4 βήματα.»

GS2E12 : « Δεν θα κάνει τίποτα. Γιατί μέσα δεν έχει τίποτα.»

Ερώτηση 2 : Τι θα συμβεί αν πατήσουμε αυτή την εντολή;



Απαντήσεις :

GS2E1 : « Θα πατήσουμε αυτό (πάνω στην εντολή). Δεν θυμάμαι.»

GS2E2 : « Θα πάει 4 φορές προς τα εδώ (δεξιά). »

GS2E3 : « Θα πάει 1 βήμα προς τα δεξιά. »

GS2E4 : « Θα κάνει 4 βήματα προς τα εκεί ο γατούλης. »

GS2E5 : « Νομίζω ο γατούλης θα φύγει και θα πάει στο Χ επειδή το βελάκι μας δείχνει εδώ (δεξιά) 2 φορές. »

GS2E6 : « Ο γατούλης ή ένας άλλος θα πάει από εκεί (δεξιά) 4 βήματα. »

GS2E7 : « Θα προχωρήσει 4 φορές προς τα εκεί (δεξιά). »

GS2E8 : « Θα πάει 4 βήματα από σένα (δεξιά). »

GS2E9 : « Θα πάει 4 βήματα προς τα εκεί (δεξιά). »

GS2E10 : « Θα κάνει 1 βήμα από εδώ (δεξιά). »

GS2E11 : « Θα πάει 5 βήματα προς τα εδώ (δεξιά). Ήταν προς τα εδώ (αριστερά) και θα πάει προς τα εδώ (δεξιά). »

GS2E12 : « Θα προχωρήσει 4 βήματα προς τα εδώ (δεξιά). »

Ερώτηση 3 : Τι θα συμβεί αν πατήσουμε αυτή την εντολή;

Απαντήσεις :

GS2E1 : « Δεν ξέρω.»

GS2E2 : « Θα πει 4 φορές ότι του έχουμε πει 4 φορές. »

GS2E3 : « Θα μιλήσει ο γατούλης 1 φορά. »

GS2E4 : « Θα κάνει 4 φορές ότι πούμε εδώ πέρα. »

GS2E5 : « Θα μιλήσει 1 φορά. »

GS2E6 : « Θα μιλήσει 4 φορές. »

GS2E7 : « Θα μιλήσει 4 φορές με αυτό που θα έχεις πει. »

GS2E8 : « Θα απαντήσει 4 φορές. »

GS2E9 : « Θα πει 4 φορές ένα τραγούδι. »

GS2E10 : « Θα μιλήσει 1 φορά. »

GS2E11 : « Θα πάει 4 βήματα και μετά θα πει κάτι. »

GS2E12 : « Θα μιλήσει 4 φορές. »

Ερώτηση 4 : Τι θα συμβεί αν πατήσουμε αυτή την εντολή;

Απαντήσεις:

GS2E1 : « Δεν ξέρω. »

GS2E2 : « Πρώτα θα πάει ένα βήμα προς τα εδώ και μετά θα μιλήσει 3 φορές. Θα κάνει 3 βήματα και μετά θα μιλήσει τρεις φορές. »

GS2E3 : « Θα μετακινηθεί 3 φορές δεξιά και θα γυρίσει 3 βήματα πίσω. »

GS2E4 : « Θα κάνει 3 βήματα προς τα εκεί ο γατούλης και μιλάει 3 φορές. »

GS2E5 : « Μια…δεν ξέρω. »

GS2E6 : « Θα πάει 3 φορές προς τα εδώ και θα μιλήσει 3 φορές. »

GS2E7 : « Θα προχωρήσει μια φορά και θα μιλήσει, θα προχωρήσει μια φορά και θα μιλήσει, θα προχωρήσει μια φορά και θα μιλήσει και μετά θα σταματήσει . »

GS2E8 : « Θα κάνει ένα βήμα από εδώ, θα μιλήσει 3 φορές μάλλον. »

GS2E9 : « Θα περπατήσει 3 φορές και θα πει τραγούδι (Πρώτα βήματα και μετά τραγούδι 3 φορες). »

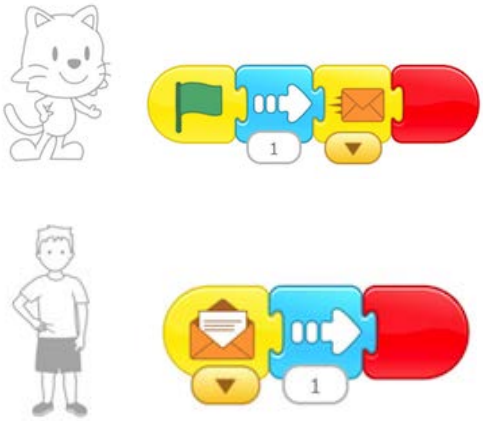
GS2E10 : « Θα μιλήσει 2 φορές και θα πάει ένα βήμα από εκεί (δεξιά). »

GS2E11 : « Θα προχωρήσει 3 βήματα, μετά 1 ακόμα και μετά θα πει κάτι. »

GS2E12 : « Τώρα θα προχωρήσει 3 φορές και θα τραγουδήσει 3 φορές. »

Annexe 3: Post-test Μηνύματα

Ερώτηση : Αν έχουμε αυτούς τους δύο χαρακτήρες, τι θα συμβεί όταν πατήσουμε το εικονίδιο της πράσινης σημαίας ;



Απαντήσεις :

GS2E1 : « Δεν ξέρω. »

GS2E2 : « Θα ξεκινήσει πρώτος ο γατούλης και μετά θα ξεκινήσει το παιδάκι. »

GS2E3 : « Θα μετακινηθούν όλα τα παζλ. Ο γατούλης θα πάει όλα τα βήματα που του έχουμε βάλει και το παιδάκι. »

GS2E4 : « Θα ξεκινήσουν πάλι. Ο γατούλης θα ξεκινήσει να κάνει ένα βήμα, θα στείλει ένα μήνυμα στο παιδί και μετά θα κάνει ένα βήμα προς τα δεξιά το παιδί και μετά θα τελειώσει.»

GS2E5 : « Θα ανοίξει τα γράμμα το παιδάκι. »

GS2E6 : « Θα ξεκινήσει μόνο ο γατούλης γιατί το παιδάκι δεν έχει πράσινη σημαία. »

GS2E7 : « Θα ξεκινήσει ο γάτος, θα πάει μια φορά προς τα εκεί μετά θα στείλει το γράμμα, το παιδί θα το λάβει θα πάει από εκεί και θα σταματήσει. »

GS2E8 : « Θα ξεκινήσει ο γάτος. »

GS2E9 : « Θα προχωρήσει ο γάτος και το παιδάκι. »

GS2E10 : « Θα πατήσουμε την πράσινη σημαία…Δεν ξέρω. Θα προχωρήσει ο γατούλης και θα στείλει το γράμμα και θα σταματήσει. »

GS2E11 : « Θα προχωρήσει ο γάτος, θα στείλει μήνυμα στο παιδί να ξεκινήσει, εδώ είναι ο τερματισμός. Μόλις το παιδί πάρει το μήνυμα θα ξέρει πότε να ξεκινήσει και θα πάει στο τέρμα.»

GS2E12 : « Θα προχωρήσει ο γατούλης και μετά το παιδάκι. »

Annexe 4 : Le scénario pédagogique conçu pour l'étude préparatoire en Grèce

ΕΚΠΑΙΔΕΥΤΙΚΟ ΣΕΝΑΡΙΟ ΕΞΟΙΚΕΙΩΣΗΣ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΜΕ ΤΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΛΟΓΙΣΜΙΚΟ ΟΠΤΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ SCRATCH JUNIOR

1. Γνωστικό αντικείμενο του σεναρίου

Το γνωστικό αντικείμενο του σεναρίου αφορά τις Τεχνολογίες της Πληροφορίας και των Επικοινωνιών και την Πληροφορική στην προσχολική ηλικία. Ειδικότερα συμβάλλει στην εκμάθηση προγραμματισμού από τα παιδιά του νηπιαγωγείου, μέσα από το περιβάλλον του ScratchJr (http ://www.scratchjr.org/), το οποίο έχει αναπτυχθεί ειδικά για αυτές τις ηλικίες. Στο πλαίσιο αυτό το σενάριο περιλαμβάνει ένα σύνολο από διδακτικές δραστηριότητες. Οι δραστηριότητες θα διεξάγονται ατομικά για το κάθε παιδί, αλλά μέσα σε ομάδες προκειμένου εξοικονομήσουμε χρόνο για την διεξαγωγή της δραστηριότητας.

Αρχικά θα διεξαχθούν τρεις **δραστηριότητες ψυχολογικής** (πλαισίωση και κίνητρα) και **γνωστικής** (γνωριμία με ταμπλέτα και ανίχνευση ιδεών) **προετοιμασίας** στο πλαίσιο των οποίων τα παιδιά (στο σύνολο της τάξης) θα ενημερωθούν για το πώς θα δουλέψουμε με το ScratchJr, θα αναπτυχθούν κίνητρα εμπλοκής τους σε αυτή τη διαδικασία, θα διεξαχθεί μια εξοικείωση με την ταμπλέτα και θα δοθεί ένα ατομικό ερωτηματολόγιο (προ-τεστ) με την μορφή συνέντευξης στα παιδιά, με σκοπό την ανίχνευση των πρότερων γνώσεων, σχετικά με το περιβάλλον του ScratchJr, αλλά και την ταμπλέτα.

Στη συνέχεια θα υλοποιηθεί μια σειρά από **δραστηριότητες διδασκαλίας**, οι οποίες θα διεξαχθούν σε ομάδες (τέσσερα ή πέντε παιδιά ανά ομάδα, ανάλογα τον αριθμό των παιδιών της τάξης) αλλά το κάθε παιδί θα έχει το δικό του τάμπλετ. Οι **δραστηριότητες διδασκαλίας** χωρίζονται σε δυο κύκλους. Ο πρώτος κύκλος περιλαμβάνει τις έννοιες που σχετίζονται με την διεπιφάνεια χρήστη, τις εντολές κίνησης και την ηχογράφηση. Μετά την λήξη του πρώτου κύκλου δραστηριοτήτων διδασκαλίας θα υλοποιηθεί **δραστηριότητα εμπέδωσης**, η οποία θα περιλαμβάνει μια δραστηριότητα προγραμματισμού μέσω της επίλυσης ενός προβλημάτων κάνοντας χρήση των εννοιών που έχουν διδαχτεί ως εκείνη τη στιγμή. Ο δεύτερος κύκλος περιλαμβάνει δραστηριότητες

που σχετίζονται με τις προγραμματιστικές έννοιες, επανάληψη και μηνύματα. Σε κάθε μια από αυτές τις εντολές θα αφιερωθούν δυο δραστηριότητες, μια **δραστηριότητα διδασκαλίας και μια εμπέδωσης**.

Στο τέλος θα διεξαχθούν δυο **δραστηριότητες αξιολόγησης**. Η μια θα αφορά δύο ατομικά πρόβληματα προγραμματισμού προς επίλυση, και η άλλη ένα ατομικό ερωτηματολόγιο (μετά-τέστ) με την μορφή ημιδομημένης συνέντευξης, ίδιο επί της ουσίας με το αρχικό ερωτηματολόγιο.

2. Συμβατότητα σεναρίου με το πρόγραμμα σπουδών του νηπιαγωγείου

Οι δραστηριότητες που εμπεριέχονται στο σενάριο είναι απόλυτα συμβατές με το πρόγραμμα σπουδών του 2011 για το νηπιαγωγείο, καθώς εξυπηρετούν τον βασικό σκοπό ένταξης των ΤΠΕ στο νηπιαγωγείο και καλύπτουν τους θεματικούς άξονες « *Γνωρίζω τις ΤΠΕ και Δημιουργώ » και « Διερευνώ, Πειραματίζομαι, Ανακαλύπτω και Λύνω προβλήματα με τις ΤΠΕ ».*

Τα παιδιά εξοικειώνονται με ψηφιακές φορητές συσκευές (ταμπλέτες) και μαθαίνουν να χειρίζονται το περιβάλλον προγραμματισμού ScratchJr, μέσα από το οποίο δημιουργούν και εκφράζουν τις ιδέες τους. Ακόμα, μέσα από το περιβάλλον αυτό πειραματίζονται με σκοπό να ανακαλύψουν τη νέα γνώση, σκέφτονται και προσπαθούν να επιλύσουν προβλήματα κατά τη διάρκεια του προγραμματισμού.

3. Προαπαιτούμενες γνώσεις των μαθητών

Για την υλοποίηση του σεναρίου απαιτείται οι μαθητές να έχουν απλές γνώσεις χειρισμού μιας ψηφιακής φορητής συσκευής (ταμπλέτα) και ειδικότερα να είναι εξοικειωμένοι με τον χειρισμό της οθόνης αφής και της διαδικασίας *drag and drop* (σύρε κι άφησε). Επιπλέον καθώς στο περιβάλλον υπάρχει το προσθετικό σύμβολο (+), θα πρέπει τα παιδιά να έχουν έρθει σε επαφή με απλές πράξεις πρόσθεσης.

4. Σκοπός Σεναρίου

Σκοπός του σεναρίου είναι η εισαγωγή των παιδιών προσχολικής ηλικίας στον προγραμματισμό μέσα από τη γνωριμία και εξοικείωση με το περιβάλλον οπτικού προγραμματισμού ScratchJr.

Ηλικία παιδιών : 5-6 ετών.

Υλικά : τάμπλετες με την εφαρμογή ScratchJr.

5. Διδακτικές προσεγγίσεις- Θεωρητική- Μεθοδολογική προσέγγιση

Σε όλο το εύρος του σεναρίου κυριαρχεί η εποικοδομιστική διδακτική στρατηγική, την οποία και υπηρετεί η γλώσσα προγραμματισμού ScratchJr. Οι μαθητές τίθενται στο επίκεντρο. Θα πρέπει να ενεργήσουν με σκοπό να οδηγηθούν στην **ανακάλυψη του γνωστικού αντικειμένου**. Καλούνται ακόμα να **πειραματιστούν** και να **επιλύσουν προβλήματα.** Οξύνουν έτσι την κριτική τους σκέψη και οδηγούνται στην λήψη αποφάσεων. Μέσα από το σενάριο επομένως γίνεται προσπάθεια να έρθουν σε επαφή με την νέα γνώση, να εξοικειωθούν με αυτή και τέλος να την κατακτήσουν.

6. Οργάνωση Τάξης (οργάνωση ανά είδος δραστηριότητας)

6.1. Οργάνωση δραστηριότητας ψυχολογικής & γνωστικής προετοιμασίας (πλαισίωση, κίνητρα, εξοικείωση με ταμπλέτα) :

Η δραστηριότητα πλαισίωσης διεξάγεται στο σύνολο της τάξης, ενώ η δραστηριότητα εξοικείωσης με την ταμπλέτα διεξάγεται ατομικά (ή σε δυάδες με δύο ταμπλέτες ώστε το ένα παιδί να υποστηρίζει το άλλο) με υποδείξεις της ερευνήτριας.

6.2. Οργάνωση υλοποίησης δραστηριοτήτων γνωστικής προετοιμασίας (ανίχνευσης) και αξιολόγησης :

Η διαδικασία υλοποίησης των συνεντεύξεων οι οποίες αφορούν στην καταγραφή των γνωστικών αναπαραστάσεων (pre-test, post-test) καθώς και της προγραμματιστικής ικανότητας των παιδιών με την επίλυση συγκεκριμένων προβλημάτων θα πραγματοποιηθούν ατομικά.

6.3. Οργάνωση υλοποίησης δραστηριοτήτων διδασκαλίας και εμπέδωσης :

Η ερευνήτρια θα εργάζεται κάθε φορά με 1 ομάδα των 4 ατόμων. Κάθε παιδί θα έχει το δικό του τάμπλετ. Η ταμπλέτα της ερευνήτριας θα είναι συνδεδεμένη με ένα προβολικό έτσι ώστε

οι μαθητές να έχουν πρόσβαση σε ό,τι συμβαίνει σε αυτήν και να συμμετέχουν με μεγαλύτερη άνεση στην δραστηριότητα.

7. Λεξιλόγιο για το Περιβάλλον Προγραμματισμού ScratchJr

Εφαρμογή : το λογισμικό – το περιβάλλον προγραμματισμού ScratchJr.

Εντολές : τα χρωματιστά τουβλάκια, τα οποία μπορούν να ενταχθούν σε ένα σενάριο.

Σενάριο : σειρά εντολών με αρχή και τέλος.

Έργο : το σύνολο των σεναρίων, τα οποία είναι αποθηκευμένα (ή μπορούν να αποθηκευτούν) σε ένα αρχείο.

Παλέτα εντολών : Αποτελείται από 6 χρωματιστές κατηγορίες εντολών, καθεμία από τις οποίες εμπεριέχει τις εντολές προγραμματισμού :

1. **Κατηγορία εκκίνησης** (κίτρινη)
2. Κ**ατηγορία κίνησης** (μπλε)
3. Κ**ατηγορία εμφάνισης** (μωβ)
4. **Κατηγορία ήχου** (πράσινη)
5. **Κατηγορία ελέγχου** (πορτοκαλί)
6. **Κατηγορία τερματισμού** (κόκκινη)

Κίνηση :

- Οριζόντια κίνηση (κίνηση προς τα δεξιά & αριστερά)
- Κάθετη κίνηση (κίνηση προς τα πάνω και κάτω)
- Αναπήδηση

Ηχογράφηση : καταγραφή φωνής και απόδοση ήχου στους χαρακτήρες.

Χώρος προγραμματισμού : Ο χρήστης πρέπει να σύρει και να συνδέσει μεταξύ τους τις εντολές, προκειμένου να προγραμματίσει τον χαρακτήρα που παρουσιάζεται στα αριστερά.

Χαρακτήρες : Οι ήρωες ενός σεναρίου / ενός έργου.

Σκηνικά : Εικόνες που τοποθετούνται στη σκηνή ως υπόβαθρο.

Σκηνή : Ο χώρος δράσης των χαρακτήρων (παραδείγματα καθημερινής ζωής : χαρακτήρες που ανήκουν σε μουσική ή θεατρική σκηνή ή και σε σκηνή κινηματογράφου. Οι χαρακτήρες δρουν ανάλογα με το πραγματολογικό πλαίσιο στο οποίο εντάσσονται και ανήκουν.

Πλήρης οθόνη : Μεγαλώνει η σκηνή και καλύπτει όλη την επιφάνεια της ταμπλέτας.

Πράσινη σημαία :

- Με την επιλογή της πράσινης σημαίας σε ένα σενάριο, ξεκινά εκτέλεσή του.

- Με την επιλογή της πράσινης σημαίας πάνω δεξιά ξεκινά το σύνολο των σεναρίων της ενεργής σκηνής.

Έναρξη έργου : Η πράσινη σημαία (πάνω δεξιά) σηματοδοτεί την εκκίνηση του σεναρίου ή των σεναρίων της ενεργής σκηνής.

Τέλος έργου : Η εντολή αυτή σηματοδοτεί το σταμάτημα του έργου.

Αποθήκευση έργου : Το τρέχον έργο αποθηκεύεται αυτόματα.

8. ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ ΓΝΩΣΤΙΚΗΣ – ΨΥΧΟΛΟΓΙΚΗΣ ΠΡΟΕΤΟΙΜΑΣΙΑΣ (ΑΝΙΧΝΕΥΣΗΣ)

Προτείνονται στους μαθητές τρείς (3) δραστηριότητες.

Στις δραστηριότητες αυτές : α) ανιχνεύονται οι γνώσεις των παιδιών, β) γίνεται η πλαισίωση του σεναρίου, αναπτύσσονται τα κίνητρα εμπλοκής των μαθητών σε αυτό και γ) λαμβάνει χώρα η αρχική εξοικείωση με την ταμπλέτα.

1η Δραστηριότητα : Ανίχνευση πρότερων γνώσεων (Προ-Τεστ)

Εκτιμώμενη διάρκεια : 10 λεπτά ανά παιδί

Η δραστηριότητα αυτή πραγματοποιείται ατομικά. Η ερευνήτρια παρουσιάζει μέσα από την ταμπλέτα της την εφαρμογή ScratchJr, ενώ παράλληλα διατυπώνει ερωτήσεις με σκοπό να εκμαιεύσει τις πρότερες γνώσεις των παιδιών.

2η Δραστηριότητα : Πλαισίωση και ανάπτυξη κινήτρων

Εκτιμώμενη διάρκεια : 30 λεπτά

Προαπαιτούμενες γνώσεις : Γνώση του Δεξιά και του Αριστερά. Δεν απαιτούνται περαιτέρω γνώσεις για την διεξαγωγή αυτής της δραστηριότητας.

Έννοιες : κίνηση στο χώρο, αριθμός βημάτων, αρχή & τέλος προγραμματισμού, προγραμματισμός, αλγόριθμος.

Στόχοι της δραστηριότητας :

- Να έρθουν σε μια πρώτη επαφή με τον προγραμματισμό υπολογιστών αναλαμβάνοντας ρόλους προγραμματιστή-μηχανής σε παιχνίδι κατεύθυνσης στο χώρο.

- Να εξοικειωθούν με την προφορική διατύπωση ενός αλγόριθμου βημάτων (ρόλος προγραμματιστή), προκειμένου να κατευθύνουν τους συμμαθητές τους (ρόλος μηχανής) στο δάπεδο.
- Να πειραματιστούν εκτελώντας ένα προφορικό αλγόριθμο βημάτων (εμπρός, πίσω, δεξιά, αριστερά).

- Να κατορθώσουν να ορίσουν την αρχική και την τελική θέση προγραμματισμού στο χώρο.

- Να αντιληφθούν τον ρόλο του προγραμματιστή στο παιχνίδι και να μεταφέρουν αυτή την γνώση για να κατανοήσουν τον προγραμματισμό υπολογιστών.

Υλικοτεχνική υποδομή : χαρτοταινία, χαρτόνι κόκκινο και πράσινο.

Οργάνωση τάξης : Η δραστηριότητα πλαισίωσης διεξάγεται στο σύνολο της τάξης. Τα παιδιά χωρίζονται σε δυάδες. Πρόκειται για ένα παιχνίδι ρόλων. Στο παιχνίδι συμμετέχει μια δυάδα παιδιών κάθε φορά.

Διδακτικές προσεγγίσεις και στρατηγικές :

Θεωρητική προσέγγιση : Σε αυτή τη δραστηριότητα έχουμε υιοθετήσει την εποικοδομιστική θεωρητική προσέγγιση καθώς οι μαθητές καλούνται να εισαχθούν στον προγραμματισμού μέσω ενός παιχνιδιού ρόλων στο χώρο. Τα παιδιά καλούνται να ανακαλύψουν την έννοια του προγραμματισμού υπολογιστών μέσω ενός βιωματικού παιχνιδιού. Καλούνται ακόμα να πειραματιστούν επιλύοντας προβλήματα που αφορούν την κίνηση στο χώρο από μια αρχική θέση(αφετηρία) προς μια τελική θέση(τέλος) και με αυτόν τον τρόπο εισάγονται στην έννοια του αλγορίθμου βιωματικά.

Διδακτικές στρατηγικές : Ως διδακτικές στρατηγικές θα χρησιμοποιηθούν κατά κύριο λόγο η επίλυση προβλήματος σε ότι αφορά την διατύπωση του αλγόριθμου βημάτων για να φτάσουν από την αρχική στην τελική θέση στο χώρο και διερεύνηση κατά την διάρκεια της δημιουργίας του αλγορίθμου. Θα χρησιμοποιηθεί διδακτική βοήθεια όποτε οι μαθητές το

έχουν ανάγκη για να προχωρήσουν προς την κατάκτηση της γνώσης μέσω της επίλυσης προβλήματος. Πιο συγκεκριμένα ενδέχεται να βοηθήσουμε τους μαθητές να εκφράσουν αποτελεσματικά τον αλγόριθμο βημάτων για την επίλυση του προβλήματος. Η ερευνήτρια θα έχει ρόλο συνερευνητικό στην όλη διαδικασία.

Περιγραφή δραστηριότητας :

I. Αρχικά θα συγκεντρωθούμε στην γωνιά της παρεούλας όπου θα ξεκινήσουμε με ένα παιχνίδι ρόλων. Σε ένα πρώτο στάδιο θα εξηγήσουμε στα παιδιά τους κανόνες του παιχνιδιού αυτού.

II. Τα παιδιά θα χωριστούν σε δυάδες και θα κληθούν να παίξουν ένα παιχνίδι κατεύθυνσης στο χώρο. Πιο συγκεκριμένα το ένα παιδί θα έχει τον ρόλο του προγραμματιστή, ενώ το δεύτερο θα αναλάβει να αναπαραστήσει μια μηχανή. Ο προγραμματιστής θα πρέπει να προγραμματίσει κατάλληλα την μηχανή του, σχηματίζοντας προφορικά έναν αλγόριθμο βημάτων, έτσι ώστε να κινηθεί αποτελεσματικά από μια αρχική θέση σε μια τελική θέση μέσα στον χώρο της παρεούλας. Για να διευκολύνουμε την κίνηση βημάτων στην παρεούλα, θα σχηματίσουμε στην μοκέτα τετραγωνάκια κίνησης, χρησιμοποιώντας χαρτοταινία.

III. Στη συνέχεια, θα δώσουμε στον προγραμματιστή δυο κάρτες, μια πράσινη και μια κόκκινη, οι οποίες θα δηλώνουν την αφετηρία και το τέλος του προγραμματισμού αντίστοιχα. Ο προγραμματιστής θα ορίζει αρχικά στο ταμπλό από χαρτοταινία την αρχική και την τελική θέση της μηχανής του και έπειτα θα διατυπώνει προφορικά οδηγίες, ώστε ο συμμαθητής του να φτάσει από την αφετηρία στην τελική θέση.

IV. Αφού ολοκληρώσουν επιτυχώς ένα γύρο παιχνιδιού τα παιδιά θα αλλάξουν ρόλους, ώστε να βιώσουν εμπειρικά και τους δυο ρόλους και να αντιληφθούν με αυτόν τον τρόπο πώς λειτουργούν ως προγραμματιστές και πώς ως μηχανές.

V. Αρχικά, και μέχρις ότου τα παιδιά να εξοικειωθούν με το παιχνίδι ρόλων, στο ταμπλό θα δουλεύει μια ομάδα κάθε φορά και η επόμενη θα ξεκινάει μόλις τελειώσει η προηγούμενη.

VI. Ανάλογα με το πώς προχωράει η δραστηριότητα, θα μπορούσαμε ενδεχομένως να ζητήσουμε προοδευτικά από δυο ομάδες να παίξουν ταυτόχρονα στο ταμπλό και να προγραμματίσουν κατάλληλα τις μηχανές τους έτσι ώστε να φτάσουν στην επιθυμητή τελική θέση, χωρίς να συγκρουστούν με την μηχανή του άλλου προγραμματιστή.

VII. Όταν όλες οι δυάδες ολοκληρώσουν το παιχνίδι θα αρχίσουμε μια συζήτηση στην παρεούλα σχετικά με το παιχνίδι που παίξαμε. Στόχος αυτής της συζήτησης είναι μέσα από κατάλληλα διατυπωμένες ερωτήσεις, η ερευνήτρια να εισάγει τα παιδιά στην έννοια του προγραμματισμού υπολογιστών αξιοποιώντας την βιωματική τους εμπειρία. Οι ερωτήσεις που χαρακτηρίζονται παρακάτω ως διδακτικές βοήθειες, θα χρησιμοποιηθούν μόνο στην περίπτωση που οι μαθητές δεν θα οδηγηθούν εκεί μέσα από την συζήτηση.

VIII. Οι ερωτήσεις που θα ακολουθήσουν θα είναι τέτοιου τύπου :

- Μόλις καταφέρατε να προγραμματίσετε την μηχανή σας. Τι άλλο θα μπορούσατε να προγραμματίσετε;
- Μήπως το τάμπλετ;(διδακτική βοήθεια)
- Πως νομίζετε ότι μπορείτε να το προγραμματίσετε;
- Τι νομίζετε ότι θα γίνει αν το αφήσουμε εδώ, θα κάνει τίποτα;(διδακτική βοήθεια)
- Τι νομίζετε ότι πρέπει να κάνουμε;
- Μήπως να το ανοίξουμε;(διδακτική βοήθεια)
- Και τώρα τί κάνουμε; Πώς θα το προγραμματίσουμε; Ποιος νομίζετε είναι ο τρόπος να πούμε στο τάμπλετ να κάνει κάτι;
- Ίσως αν πατήσουμε κάποιο από αυτά εδώ;(διδακτική βοήθεια)
- Τώρα που το πατήσαμε τι πιστεύετε ότι κάναμε; Πιστεύετε ότι το προγραμματίσαμε; Του δώσαμε εντολές, κάνε εκείνο κάνε το άλλο;
- Όχι, διότι όλα αυτά τα έκανε πριν από εμάς κάποιος άλλος και εμείς απλά τα χρησιμοποιούμε(διδακτική βοήθεια).
- Τι λέτε να μάθουμε πώς να κάνουμε εμείς κάποιος τα προγραμμάτισε πριν από εμάς;
- Πώς γίνεται αυτό τώρα : Το τάμπλετ και γενικά όλες οι μηχανές καταλαβαίνουν μόνο μια γλώσσα. Για αυτό το λόγο για να μπορέσουμε να τους πούμε τι να κάνουν πρέπει να μάθουμε να γράφουμε την γλώσσα που καταλαβαίνουν. Η γλώσσα που καταλαβαίνουν όμως είναι αρκετά δύσκολη για να την μάθουμε από

τώρα(διδασκαλία). Εμείς τώρα θα μάθουμε μια γλώσσα πιο απλή και είμαι σίγουρη ότι θα καταφέρουμε να προγραμματίσουμε ακόμα πιο ωραία και πιο διασκεδαστικά πράγματα.

- Στη συνέχεια θα διενεργηθεί η εξοικείωση με την ταμπλέτα ανά ομάδα παιδιών.

3η Δραστηριότητα : Εξοικείωση με το περιβάλλον του ScratchJr

Εκτιμώμενη διάρκεια : 15 λεπτά ανά ομάδα παιδιών.
Προτείνεται να έχει εγκατασταθεί η εφαρμογή ScratchJr.

Προαπαιτούμενες γνώσεις : Καλό θα ήταν οι μαθητές να έχουν προηγούμενη εμπειρία χρήσης ταμπλέτας ώστε να έχουν εξοικειωθεί με το περιβάλλον και την οθόνη αφής αυτού του εργαλείου.

Λίστα εννοιών, γνώσεων και δεξιοτήτων :

Έννοιες : ταμπλέτα, εφαρμογή, οθόνη, πρόγραμμα, κλείδωμα-ξεκλείδωμα οθόνης, εικονίδια, επιφάνεια εργασίας.

Γνώσεις & Δεξιότητες : χειρισμός διεπιφάνειας λογισμικού, λεπτή κινητικότητα (σύρε κι άφησε) – οπτικοκινητικός συντονισμός.

Στόχοι της δραστηριότητας :

- Να πειραματιστούν με την χρήση μιας ταμπλέτας.
- Να κατανοήσουν πως μπορούν να κλειδώσουν και να ξεκλειδώσουν την οθόνη της ταμπλέτας.
- Να ανοίγουν ένα πρόγραμμα.
- Να εξοικειωθούν με την λεπτή κινητικότητα.

Υλικοτεχνική υποδομή : ταμπλέτες με εγκατεστημένη την εφαρμογή ScratchJr.

Οργάνωση της τάξης : Η δραστηριότητα εξοικείωσης με την ταμπλέτα διεξάγεται σε ομάδες. Πρόκειται για δραστηριότητα διερεύνησης της χρήσης της ταμπλέτας και του

λογισμικού, αλλά και διδασκαλίας ορισμένων σημαντικών εννοιών σχετικών με την ταμπλέτα.

Διδακτικές προσεγγίσεις και στρατηγικές :

Θεωρητική προσέγγιση : Σε αυτή τη δραστηριότητα έχουμε υιοθετήσει την εποικοδομιστική θεωρητική προσέγγιση καθώς οι μαθητές καλούνται να διερευνήσουν και να ανακαλύψουν τον τρόπο χρήσης της ταμπλέτας αλλά και τον τρόπο λειτουργίας του λογισμικού. Τα παιδιά καλούνται επίσης να χρησιμοποιήσουν την γνώση για την έννοια του προγραμματισμού υπολογιστών, που αποκόμισαν μέσω του βιωματικού παιχνιδιού της δραστηριότητας ψυχολογικής προετοιμασίας, για να κατανοήσουν τον τρόπο λειτουργίας του λογισμικού.

Διδακτικές στρατηγικές : Ως διδακτικές στρατηγικές θα χρησιμοποιηθούν κατά κύριο λόγο η επίλυση προβλήματος, η διερεύνηση και η ανακάλυψη σε ότι αφορά την κατανόηση της λειτουργίας του λογισμικού. Θα χρησιμοποιηθεί επιπλέον διδακτική βοήθεια όποτε οι μαθητές το έχουν ανάγκη για να προχωρήσουν προς την κατάκτηση της γνώσης. Η ερευνήτρια θα έχει ρόλο συνερευνητικό στην όλη διαδικασία.

Περιγραφή :

Ι. Αρχικά, αφού δώσουμε στα παιδιά την ταμπλέτα θέτουμε ορισμένες διερευνητικές ερωτήσεις, προκειμένου να εκμαιεύσουμε τους τρόπους χρήσης που γνωρίζουν. Έπειτα τους εξηγούμε τι είναι αυτό το εργαλείο και ποιες μπορεί να είναι οι πιθανές χρήσεις του (διδασκαλία).

- Τι είναι; Με τι μοιάζει; Έχετε χρησιμοποιήσει κάτι παρόμοιο; (μπορούμε να αντιπαραβάλουμε μ' έναν σταθερό υπολογιστή ή έναν φορητό)

ΙΙ. Στη συνέχεια, ζητάμε από τους μαθητές να ανακαλύψουν πώς ανοίγει η ταμπλέτα. Σε περίπτωση που δυσκολεύονται θα δοθεί διδακτική βοήθεια, αφού θα τους υποδείξουμε το σχετικό κουμπί.

III. Ακολούθως εξηγούμε πως αυτό που βλέπουμε είναι η επιφάνεια εργασίας, η οποία έχει διάφορα εικονίδια, τα περισσότερα από τα οποία είναι προγράμματα (διδασκαλία). Έπειτα μοιράζουμε από μια ταμπλέτα σε κάθε δυάδα παιδιών.

IV. Έπειτα, παρουσιάζουμε στα παιδιά το εικονίδιο του ScratchJr και τα προτρέπουμε να το ανοίξουν. Στο σημείο αυτό θα πρέπει να τονίσουμε την αποκλειστική χρήση δαχτύλων για την χρήση της οθόνης αφής και πιο συγκεκριμένα την χρήση των δεικτών.

V. Στη συνέχεια εξηγούμε στα παιδιά πως με αυτό το πρόγραμμα μπορούμε να φτιάξουμε δικές μας ιστορίες και κινούμενα σχέδια.

VI. Έπειτα, φτάνουμε στην αρχική οθόνη του προγράμματος και εκεί τους ζητάμε να διερευνήσουν που πρέπει να πατήσουμε για να βρούμε τον γατούλη.

VII. Έπειτα τους ζητάμε να μας πουν τι παρατηρούν στην οθόνη της ταμπλέτας.

- Τι παρατηρείτε στην οθόνη της ταμπλέτας τώρα που ανοίξαμε το πρόγραμμα της γάτας ;
- Τι νομίζετε ότι μπορούμε να κάνουμε με την γάτα ;
- Πώς νομίζετε ότι μπορούμε να κουνήσουμε την γάτα ;

VII. Αφού πειραματιστούν με το λογισμικό και μόνο στην περίπτωση που δεν κατορθώνουν να ανακαλύψουν πως μπορούν να χρησιμοποιήσουν τις εντολές κίνησης, θα προταθεί στα παιδιά να τις χρησιμοποιήσουν σύροντας τις στο χώρο προγραμματισμού(διδακτική βοήθεια).

VIII. Για να βγουν έπειτα από το λογισμικό και να επιστρέψουν στην επιφάνεια εργασίας, θα πρέπει να πατήσουν χαμηλά στην οθόνη για να εμφανιστεί μια σειρά επιλογών και να επιλέξουν την επιστροφή στη κεντρική οθόνη. Αν δυσκολεύονται να ανακαλύψουν την εν λόγω διαδικασία θα δοθεί διδακτική βοήθεια.

IX. Μπορούμε επίσης να κλειδώσουμε την οθόνη, εάν θέλουμε να σταματήσουμε για λίγο να χρησιμοποιούμε το τάμπλετ (δείχνουμε το σχετικό κουμπί-διδακτική βοήθεια).

X. Προτρέπουμε να το ξαναπατήσουν για να διαπιστώσουν ότι ξεκλειδώνει πάλι η οθόνη και μάλιστα ότι είμαστε στο ίδιο σημείο που ήμασταν πριν την κλειδώσουμε(διδακτική βοήθεια).

XI. Τέλος, τους ζητάμε να κλείσουν την ταμπλέτα και να την κλειδώσουν.

9. ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ ΔΙΔΑΣΚΑΛΙΑΣ ΓΝΩΣΤΙΚΟΥ ΑΝΤΙΚΕΙΜΕΝΟΥ

Προτείνονται στους μαθητές επτά (7) δραστηριότητες διδασκαλίας.

1η Δραστηριότητα διδασκαλίας : Πρώτη γνωριμία και εξοικείωση την διεπιφάνεια χρήστη και την γλώσσα προγραμματισμού ScratchJr .

Εκτιμώμενη διάρκεια : 45 λεπτά ανά ομάδα παιδιών.

Προαπαιτούμενες γνώσεις : Καλό θα ήταν οι μαθητές να γνωρίζουν πώς να ανοίξουν την ταμπλέτα και το πρόγραμμα ScratchJr.

Έννοιες : εφαρμογή, πρόγραμμα, αρχική οθόνη προγράμματος, νέο έργο, εντολή, ακολουθία, εισαγωγή εντολών στο χώρο προγραμματισμού, εντολές οριζόντιας κίνησης, ένωση εντολών στο χώρο προγραμματισμού, διαγραφή εντολών από τον χώρο προγραμματισμού, αρχή και τέλος προγραμματισμού, σενάριο προγραμματισμού, αρχικοποίηση θέσης χαρακτήρα, αποθήκευση έργου.

Γνώσεις : εφαρμογή, πρόγραμμα, αρχική οθόνη προγράμματος, νέο έργο, εντολή, εντολές οριζόντιας κίνησης, αρχή και τέλος προγραμματισμού, σενάριο προγραμματισμού.

Δεξιότητες : δημιουργία νέου έργου, εισαγωγή νέου σκηνικού, εισαγωγή εντολών στο χώρο προγραμματισμού, ένωση εντολών στο χώρο προγραμματισμού, αρχικοποίηση θέσης χαρακτήρα, διαγραφή εντολών από τον χώρο προγραμματισμού, επαλήθευση προγράμματος, αποθήκευση έργου.

Στόχοι της δραστηριότητας :

- Να γνωρίσουν τα παιδιά την διαδικασία **δημιουργίας νέου έργου.**
- Να γνωρίσουν τα παιδιά την διαδικασία **εισαγωγής νέου σκηνικού**.
- Να γνωρίσουν την διαδικασία **εισαγωγής, εξαγωγής και σύνδεσης εντολών στο χώρο προγραμματισμού.**

- Να κατανοήσουν τα παιδιά τον τρόπο προγραμματισμού κίνησης του χαρακτήρα δεξιά και αριστερά στην σκηνή.
- Να γνωρίσουν τα παιδιά την **αρχή και το τέλος του προγραμματισμού** και να σχηματίσουν ένα ο**λοκληρωμένο σενάριο**.
- Να γνωρίσουν την διαδικασία **επαλήθευσης προγράμματος**.
- Μέσω της επίλυσης ενός προβλήματος, να γνωρίσουν την **έννοια της ακολουθίας**, όπως αποτυπώνεται στο ScratchJr. Ότι δηλαδή οι εντολές ενός σεναρίου εκτελούνται με την σειρά από τα αριστερά προς τα δεξιά, η μια μετά την άλλη.
- Εκμάθηση **αποθήκευσης έργου**.

Υλικοτεχνική υποδομή : Ταμπλέτες με εγκατεστημένη την εφαρμογή ScratchJr, προβολικό.

Οργάνωση της τάξης : Η δραστηριότητα εκτυλίσσεται σε ομάδες. Κάθε παιδί έχει το τάμπλετ του. Το τάμπλετ της ερευνήτριας συνδέεται με προβολικό, ώστε οι μαθητές να βλέπουν καλύτερα τι γίνεται στην εφαρμογή. Οι ομάδες υλοποιούν τις δραστηριότητες η μια μετά την άλλη και όχι ταυτόχρονα.

Διδακτικές προσεγγίσεις & στρατηγικές :

Θεωρητικές προσεγγίσεις : Σε αυτή τη δραστηριότητα έχουμε υιοθετήσει την εποικοδομιστική θεωρητική προσέγγιση, καθώς οι μαθητές καλούνται να επιλύσουν ένα πρόβλημα προγραμματισμού. Επιπλέον, στην παρούσα δραστηριότητα έχουμε υιοθετήσει την κοινωνικοεπικοδομιστική θεωρητική προσέγγιση, καθώς προωθείται η ανταλλαγή γνώσεων μεταξύ των μαθητών όπου είναι δυνατόν, μέσα από την συζήτηση και την αλληλεπίδραση.

Διδακτικές στρατηγικές : Ως διδακτικές στρατηγικές θα χρησιμοποιηθούν κατά κύριο λόγο η επίλυση προβλήματος, η διερεύνηση και η ανακάλυψη σε ότι αφορά την γνωριμία με την σύνταξη και την σημασιολογία της γλώσσας προγραμματισμού ScratchJr. Θα χρησιμοποιηθεί επιπλέον διδακτική βοήθεια όποτε οι μαθητές το έχουν ανάγκη για να προχωρήσουν προς την κατάκτηση της γνώσης. Η ερευνήτρια θα έχει ρόλο συνερευνητικό στην όλη διαδικασία. Ακόμα θα χρησιμοποιηθεί ως διδακτική στρατηγική η συζήτηση και αλληλεπίδραση μεταξύ των παιδιών, που στόχο έχει την ανταλλαγή γνώσεων μεταξύ των μαθητών.

Σύντομη περιγραφή της δραστηριότητας :

Σε αυτή τη δραστηριότητα οι μαθητές καλούνται να λύσουν ένα προγραμματιστικό πρόγραμμα χρησιμοποιώντας τις **εντολές οριζόντιας κίνησης.** Επιπλέον, καλούνται να εξοικειωθούν με την **διεπιφάνεια χρήσης**, την **εισαγωγή νέου έργου**, την **εισαγωγή σκηνικού**, και **την διαδικασία εισαγωγής, σύνδεσης και διαγραφής εντολών** από το χώρο προγραμματισμού. Στη συνέχεια, οι μαθητές θα γνωρίσουν την **αρχή και το τέλος του προγραμματισμού,** την **αρχικοποίηση της θέσης ενός χαρακτήρα και την επαλήθευση προγράμματος**. Επιπλέον, θα γίνει μια παρουσίαση της βασικής ορολογίας προγραμματισμού που συνοδεύει το λογισμικό. Τέλος, θα διδαχτεί η **αποθήκευση έργου**.

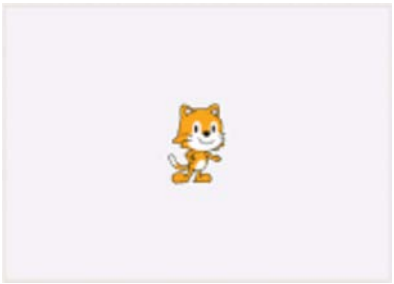
Αναλυτικά τα βήματα της δραστηριότητας :

I. Η δραστηριότητα ξεκινάει με την ερευνήτρια :
« Την προηγούμενη φορά θυμάστε τι κάναμε με το τάμπλετ; Ποιος θυμάται τι ανακαλύψαμε την προηγούμενη φορά; Την προηγούμενη φορά είχαμε μπει στο πρόγραμμα ScratchJr και είχαμε συναντήσει ένα καινούριο φίλο το γατούλη. Πώς το είχαμε κάνει αυτό; Για δοκιμάστε να **ανοίξετε το πρόγραμμα** με τον γατούλη. »
Όπου τα παιδιά δυσκολεύονται θα δίνεται διδακτική βοήθεια από την ερευνήτρια. Αν κάποιος συμμαθητής τους έχει καταφέρει να εκτελέσει μόνος του την οδηγία, θα καλείται να δείξει στους συμμαθητές του πώς το έκανε.

II. «Πολύ ωραία. Φτάσαμε εδώ. Αυτό το μέρος από εδώ και πέρα θα το ονομάζουμε αρχική οθόνη. Πώς μπορούμε εδώ να βρούμε τον γατούλη;»
Τα παιδιά καλούνται να **ανοίξουν ένα νέο έργο**.

III. «Κάθε φορά που πατάμε πάνω στο συν για να βρούμε το γατούλη, ανοίγουμε ένα νέο έργο. Κάθε έργο που θα φτιάχνουμε θα είναι και μια καινούρια ιστορία, την οποία θα σας δείξω την επόμενη φορά πώς μπορούμε να την βλέπουμε σαν κινούμενο σχέδιο στα τάμπλετ μας ».

IV. «Είχαμε ανακαλύψει διαφορετικούς τρόπους για να κινούμε τον γατούλη μέσα σε αυτό εδώ το πλαίσιο. Αυτό το πλαίσιο από εδώ και πέρα θα το λέμε **σκηνή**. Θυμάστε ποιοι ήταν αυτοί οι τρόποι; Θα τους δούμε σήμερα ξανά μαζί ».



V. «Σήμερα θα μάθουμε πώς μπορούμε να φτιάξουμε μια ιστορία με τον γατούλη. Για να το κάνουμε όμως αυτό πρέπει να διαλέξουμε ένα άλλο φόντο για την σκηνή μας. Ποιο κουμπάκι νομίζετε ότι θα μας οδηγήσει κάπου όπου μπορούμε να διαλέξουμε διαφορετικά φόντο για την σκηνή μας; Καμία ιδέα ; »
Αν τα παιδιά δυσκολεύονται θα γίνει υπόδειξη του **εικονιδίου αλλαγής σκηνικού** από την ερευνήτρια.

VI. «Εδώ που είμαστε τώρα βλέπετε πόσα πολλά φόντο υπάρχουν; Σήμερα θα διαλέξουμε ένα απλό, αυτό εδώ. Δοκιμάστε να ανακαλύψετε **πώς μπορούμε να το εισάγουμε στη σκηνή μας**».
Πάλι εδώ αν τα παιδιά δυσκολεύονται, θα δοθεί διδακτική βοήθεια από την ερευνήτρια.

VII. «Εδώ που είμαστε τώρα τι παρατηρείτε στη σκηνή; Θυμάστε την προηγούμενη φορά την δραστηριότητα που είχαμε κάνει στην παρεούλα με την αρχική και την τελική θέση και τις κάρτες; Λοιπόν το **πρόβλημα** έχει ως εξής : Πρέπει να πάρουμε με το δείκτη τον γατούλη και να τον τοποθετήσουμε στο πράσινο κύκλο. Μετά πρέπει να τον προγραμματίσουμε μέχρι να μπει στον κόκκινο κύκλο. Μόλις φτάσει μέσα στο κόκκινο κύκλο πρέπει να σταματήσει. Πώς θα το κάνουμε αυτό;». Αφήνουμε τα παιδιά να πειραματιστούν με το πρόγραμμα.

VIII. Έπειτα τους υπενθυμίζουμε ότι πάντα για να ξεκινήσουμε να προγραμματίζουμε ένα χαρακτήρα πρέπει να ορίσουμε την **αρχική του θέση.** Ορίζουμε την αφετηρία πάντα με το δείκτη. Επίσης η ερευνήτρια τους εξηγεί ότι τα κομμάτια πάζλ ονομάζονται εντολές, καθώς αναγκάζουν τον χαρακτήρα να κάνει κάτι στη σκηνή. Αν δυσκολεύονται και δεν θυμούνται ότι μπορούν να χρησιμοποιήσουν τις εντολές κίνησης τα παροτρύνουμε να τις χρησιμοποιήσουν. Αν έστω και ένας συμμαθητής τους θυμάται να τις χρησιμοποιήσει ζητάμε να το δείξει στους υπόλοιπους.

IX. Όταν αρχίζουν να χρησιμοποιούν τις **εντολές κίνησης** για να λύσουν το πρόβλημα, θα υπάρξει θέμα με τον αριθμό βημάτων, που χρειάζεται να κάνει ο γάτος για να μετακινηθεί από την αφετηρία στο τέλος. Για να είναι σωστός ο προγραμματισμός που έχουν κάνει θα πρέπει **να επαληθεύσουν το πρόγραμμα τους** εκτελώντας το αφού αρχικοποιήσουν την θέση του χαρακτήρα. Η ερευνήτρια επισημαίνει την επαλήθευση του προγράμματος με την αρχική θέση και παρουσιάζει τις **δυο διαδικασίες αρχικοποίησης**, αρχικοποίηση με εικονίδιο και αρχικοποίηση με δείκτη.

X. Στην συνέχεια, αν δεν έχουν χρησιμοποιήσει την πράσινη σημαία και το τέλος μέχρι τώρα, αντιπαραβάλουμε με την δραστηριότητα ψυχολογικής προετοιμασίας και τους ζητάμε να βρουν ένα τρόπο για να ορίσουν **την αρχή και το τέλος του προγραμματισμού**. Αν κάποιος συμμαθητής τους το έχει χρησιμοποιήσει του ζητάμε να το δείξει στους υπόλοιπους και να αιτιολογήσει την επιλογή του. Εάν πάλι κανένα από τα παιδιά δεν σκεφτεί να χρησιμοποιήσει την πράσινη σημαία και το τέλος θα δοθεί διδακτική βοήθεια στα παιδιά, αφού θα παρουσιάσουμε στην ομάδα τις δυο εντολές και την χρήση τους. Έπειτα, θα τους εξηγήσουμε ότι όταν έχουμε ένα ολοκληρωμένο πάζλ εντολών σαν αυτό με την αρχική σημαία και το τέλος θα το ονομάζουμε « **σενάριο ».**

XI. Έπειτα, θα ζητηθεί στα παιδιά να τρέξουν το σενάριο που δημιούργησαν για να δούμε εάν επιλύσαν σωστά το πρόβλημα, ώστε ο γάτος να μετακινηθεί από το πράσινο κύκλο, την αφετηρία, στο τέλος, τον κόκκινο κύκλο. Κατά την διάρκεια της επίλυσης του προβλήματος ενδέχεται οι μαθητές να ανακαλύψουν την **διαγραφή εντολών** από το χώρο προγραμματισμού, αλλά και την **σύνδεση εντολών**. Αν τα παιδιά δεν οδηγηθούν σε αυτή την ανακάλυψη θα δοθεί διδακτική βοήθεια και θα παρουσιαστούν αυτές οι διαδικασίες στους μαθητές. Η ερευνήτρια θα περάσει να δει τα έργα όλων των μαθητών δίνοντας τους μικρές κατευθύνσεις, για να τους βοηθήσει στην κατάκτηση της γνώσης και την επίλυση του προβλήματος.

XII. Έπειτα, θα ζητηθεί στα παιδιά ενώ τρέχουν το σενάριο τους να παρατηρήσουν τις εντολές στο χώρο προγραμματισμού. Στόχος αυτής της οδηγίας είναι οι μαθητές να παρατηρήσουν ότι καθώς εκτελείται το σενάριο **οι εντολές αναβοσβήνουν η μια μετά την άλλη και κατ' επέκταση εκτελούνται με την σειρά (ακολουθία)**.

XIII. Στη συνέχεια, η ερευνήτρια θέτει τον προβληματισμό πώς θα μπορούσαμε να **αποθηκεύσουμε το έργο μας**, την πρώτη μας ιστορία, ώστε να την βρούμε όταν θα ξανά ανοίξουμε την εφαρμογή. Αφήνουμε τους μαθητές να σκεφτούν και να κάνουν δοκιμές στο

περιβάλλον. Αν δεν καταφέρουν να το ανακαλύψουν, τους λέμε ότι πρέπει να πατήσουμε πάνω στο « σπιτάκι » για να επιστρέψουμε στην αρχική οθόνη της εφαρμογής και να ελέγξουμε αν το έργο μας έχει αποθηκευτεί.

XIV. Τέλος, οι μαθητές κλείνουν τα τάμπλετ και η επόμενη ομάδα παίρνει σειρά για την δραστηριότητα.

<u>2η Δραστηριότητα διδασκαλίας :</u> Προγραμματισμός δυο χαρακτήρων, εκτέλεση έργου σε πλήρη οθόνη, ήχος-ηχογράφηση, εντολές κάθετης κίνησης .

Εκτιμώμενη διάρκεια : 45 λεπτά

Προαπαιτούμενες γνώσεις :
Απαιτείται οι μαθητές να έχουν κατανοήσει πώς επιλέγουμε ένα σκηνικό, πώς εισάγουμε εντολές στο χώρο προγραμματισμού και πώς τις διαγράφουμε, πώς ενώνουμε εντολές για να σχηματίσουμε ένα σενάριο. Επίσης απαιτείται να έχουν κατανοήσει την αρχή και το τέλος του προγραμματισμού, αλλά και την διαδικασία αρχικοποίησης με τους δυο τρόπους, δάχτυλο και εικονίδιο αρχικοποίησης.

Έννοιες : χαρακτήρες, εκτέλεση έργου σε πλήρη οθόνη, εντολές κάθετης κίνησης (πάνω-κάτω), ήχος-ηχογράφηση.

Γνώσεις : χαρακτήρας, εικονίδιο πλήρους οθόνης, εντολές κάθετης κίνησης.

Δεξιότητες : εισαγωγή χαρακτήρα, προγραμματισμός δυο χαρακτήρων, εκτέλεση έργου σε πλήρη οθόνη, ηχογράφηση, διαγραφή εντολής ηχογράφησης.

Στόχοι δραστηριότητας :

- Να προγραμματίσουν ένα χαρακτήρα χρησιμοποιώντας τις **εντολές κάθετης κίνησης**.
- Να γνωρίσουν και να εξοικειωθούν τα παιδιά με την **διαδικασία εισαγωγής νέου χαρακτήρα**.
- Να κατορθώσουν **να προγραμματίσουν δυο χαρακτήρες** στο ίδιο έργο.
- Να γνωρίσουν την **εντολή της ηχογράφησης** και να εξοικειωθούν με την **διαδικασία ηχογράφησης μιας εντολής ήχου**.
- Να γνωρίσουν τον τρόπο **διαγραφής μιας εντολής ήχου**.
- Να κατανοήσουν την λειτουργία του **εικονιδίου της πράσινης σημαίας**.
- Να γνωρίσουν το **εικονίδιο της πλήρους οθόνης** και να εξοικειωθούν με την **διαδικασία προβολής ενός έργου σε πλήρη οθόνη**.

Υλικοτεχνική υποδομή : Ταμπλέτες με εγκατεστημένη την εφαρμογή ScratchJr, προβολικό.

Οργάνωση της τάξης : Η δραστηριότητα εκτυλίσσεται σε ομάδες. Κάθε παιδί έχει το δικό του τάμπλετ. Το τάμπλετ της ερευνήτριας συνδέεται με προβολικό, ώστε οι μαθητές να βλέπουν καλύτερα τι γίνεται στην εφαρμογή. Οι ομάδες υλοποιούν τις δραστηριότητες η μια μετά την άλλη και όχι ταυτόχρονα.

Διδακτικές προσεγγίσεις & στρατηγικές :

Θεωρητικές προσεγγίσεις : Σε αυτή τη δραστηριότητα έχουμε υιοθετήσει την εποικοδομιστική θεωρητική προσέγγιση, καθώς οι μαθητές καλούνται να επιλύσουν ένα πρόβλημα προγραμματισμού. Επιπλέον, στην παρούσα δραστηριότητα έχουμε υιοθετήσει την κοινωνικοεπικοδομιστική θεωρητική προσέγγιση, καθώς προωθείται η ανταλλαγή γνώσεων μεταξύ των μαθητών όπου είναι δυνατόν, μέσα από την συζήτηση και την αλληλεπίδραση.

Διδακτικές στρατηγικές : Ως διδακτικές στρατηγικές θα χρησιμοποιηθούν κατά κύριο λόγο η επίλυση προβλήματος, η διερεύνηση και η ανακάλυψη σε ότι αφορά την γνωριμία με την σύνταξη και την σημασιολογία της γλώσσας προγραμματισμού ScratchJr. Θα χρησιμοποιηθεί επιπλέον διδακτική βοήθεια όποτε οι μαθητές το έχουν ανάγκη για να προχωρήσουν προς την κατάκτηση της γνώσης. Η ερευνήτρια θα έχει ρόλο συνερευνητικό στην όλη διαδικασία. Ακόμα θα χρησιμοποιηθεί ως διδακτική στρατηγική η συζήτηση και αλληλεπίδραση μεταξύ των παιδιών, που στόχο έχει την ανταλλαγή γνώσεων μεταξύ των μαθητών.

Σύντομη περιγραφή της δραστηριότητας :

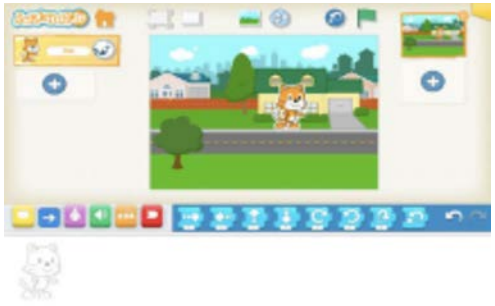
Σε αυτή τη δραστηριότητα, τα παιδιά καλούνται να επιλύσουν ένα προγραμματιστικό πρόβλημα χρησιμοποιώντας τις εντολές κάθετης και οριζόντιας κίνησης. Πιο συγκεκριμένα, καλούνται για πρώτη φορά να εισάγουν και να προγραμματίσουν δυο (2) χαρακτήρες. Επιπλέον, καλούνται να ηχογραφήσουν μια εντολή ήχου και να την εισάγουν στο πρόγραμμά τους. Έπειτα, οι μαθητές θα προβάλουν το έργο τους σε πλήρη οθόνη, κάνοντας χρήση του εικονιδίου της πράσινης σημαίας.

Αναλυτικά τα βήματα της δραστηριότητας :

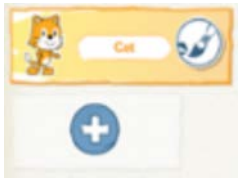
I. Οι μαθητές αρχικά εισάγονται στο πρόγραμμα και έπειτα σε **νέο έργο**.

II. Έπειτα, οι μαθητές καλούνται να εισάγουν ένα **νέο σκηνικό**, το «Suburbs».

III. Στη συνέχεια, ζητείται από τους μαθητές να τοποθετήσουν τον **χαρακτήρα « γάτο »** μπροστά από το σπίτι και να τον προγραμματίσουν κατάλληλα **(εντολές κάθετης κίνησης)** ώστε να περάσει το δρόμο.



IV. Έπειτα, η ερευνήτρια θα ρωτήσει τα παιδιά πως μπορούν να εισάγουν **ένα νέο χαρακτήρα** στη σκηνή για να έχει ο χαρακτήρας « γάτος » παρέα.



V. Έπειτα, θα ζητηθεί από τα παιδιά να **προγραμματίσουν τον νέο χαρακτήρα** ώστε να κάνει κάτι.

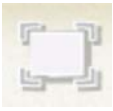
VI. Στη συνέχεια, θα τεθεί από την ερευνήτρια ο ακόλουθος προβληματισμός : « **Πώς μπορούμε να κάνουμε τον νέο χαρακτήρα να μιλήσει στον γάτο ; ».** Εδώ ζητείται από τα παιδιά να ανακαλύψουν την εντολή της ηχογράφησης και να την χρησιμοποιήσουν για να **εισάγουν έναν ήχο** στο σενάριο του νέου χαρακτήρα. Σε περίπτωση που τα παιδιά δυσκολεύονται με την διαδικασία ηχογράφησης θα δοθεί διδακτική βοήθεια από την ερευνήτρια.

VII. Μετά η ερευνήτρια θέτει το εξής ερώτημα στα παιδιά : **«Πώς νομίζετε ότι μπορούμε να διαγράψουμε μια ηχογράφηση που δεν μας αρέσει;».** Σε αυτό το σημείο τα παιδιά θα διατυπώσουν τις απόψεις τους και αν χρειαστεί θα παρουσιαστεί η διαδικασία διαγραφής της εντολής ηχογράφησης.

VIII. Στη συνέχεια, θα τεθεί το ακόλουθο ερώτημα : **«Πως νομίζετε ότι μπορούμε να δούμε τους δυο χαρακτήρες να κινούνται την ίδια στιγμή;»**. Αφήνουμε τα παιδιά να πειραματιστούν με την διεπιφάνεια χρήσης. Σε περίπτωση που τα παιδιά δεν καταλήξουν στην ανακάλυψη του **εικονιδίου της πράσινης σημαίας,** θα γίνει παρουσίαση του από την ερευνήτρια. Σε αυτό το σημείο όσοι μαθητές δεν έχουν σχηματίσει ολοκληρωμένο σενάριο με αρχή και τέλος, τα προγράμματα των χαρακτήρων τους δεν θα τρέξουν καθόλου. Η

ερευνήτρια θα εκμεταλλευτεί αυτήν τη παράλειψη για να εξηγήσει στα παιδιά την **σύνδεση που υπάρχει μεταξύ εντολής πράσινης σημαίας και εικονιδίου πράσινης σημαίας.**

IX. Έπειτα θα ζητηθεί από τα παιδιά να ανακαλύψουν πώς μπορούμε να προβάλλουμε το έργο μας σε πλήρη οθόνη : **«Πώς νομίζετε ότι μπορούμε να μεγαλώσουμε την σκηνή και να δούμε το έργο μας σε μεγαλύτερη οθόνη, σαν κινούμενο σχέδιο;»**. Αν τα παιδιά δυσκολεύονται να εντοπίσουν το εικονίδιο της πλήρους οθόνης θα τους το υποδείξουμε. Αφού τα παιδιά εισαχθούν σε πλήρη οθόνη, τα αφήνουμε να πειραματιστούν μέχρι να ανακαλύψουν **την λειτουργία του εικονιδίου της πράσινης σημαίας.**



X. Τέλος, γίνεται **αποθήκευση έργου** χρησιμοποιώντας το εικονίδιο της αποθήκευσης.

<u>3η Δραστηριότητα διδασκαλίας και εμπέδωσης :</u> Εντολή αναπήδησης, διαγραφή χαρακτήρων, επίλυση προβλήματος ανακεφαλαίωσης.

Εκτιμώμενη διάρκεια : 45 λεπτά.

Προαπαιτούμενες γνώσεις : Για την υλοποίηση αυτής της δραστηριότητας απαιτείται οι μαθητές να έχουν κατανοήσει μια σειρά εννοιών και δεξιοτήτων από προηγούμενες δραστηριότητες διδασκαλίας. Σε περίπτωση που δυσκολεύονται κάπου θα γίνεται υπενθύμιση της διαδικασίας ή της έννοιας από την ερευνήτρια. Πιο συγκεκριμένα, καλό θα ήταν οι μαθητές να μπορούν να εισάγουν ένα νέο χαρακτήρα και να τον προγραμματίσουν, να μπορούν να εισάγουν ένα νέο σκηνικό, να χρησιμοποιήσουν την εντολή της ηχογράφησης, να μπορούν να δημιουργήσουν ένα ολοκληρωμένο σενάριο χρησιμοποιώντας την αρχή και το τέλος προγραμματισμού, καθώς και να προβάλλουν το έργο τους σε πλήρη οθόνη χρησιμοποιώντας το αντίστοιχο εικονίδιο. Επίσης, πολύ σημαντικό για την πορεία του εκπαιδευτικού σεναρίου είναι να έχουν κατανοήσει την συσχέτιση που υπάρχει μεταξύ του εικονιδίου της πράσινης σημαίας και την εντολή της πράσινης σημαίας, που δηλώνει την αρχή του προγραμματισμού.

Έννοιες : εντολή αναπήδησης, εικονίδιο πράσινης σημαίας στην σκηνή.

Γνώσεις : εντολή αναπήδησης, διαγραφή χαρακτήρων.

Δεξιότητες : διαγραφή χαρακτήρων.

Στόχοι της δραστηριότητας :

- Να προγραμματίσουν **δυο χαρακτήρες** χρησιμοποιώντας τις εντολές κίνησης που έχουν διδαχτεί ως τώρα.
- Να γνωρίσουν την **εντολή της αναπήδησης.**
- Να χρησιμοποιήσουν την **εντολή της ηχογράφησης** και να εξοικειωθούν με την **διαδικασία ηχογράφησης μιας εντολής ήχου**.
- Να γνωρίσουν τους **δυο τρόπους διαγραφής ενός χαρακτήρα**.

- Να γνωρίσουν την **λειτουργία του εικονιδίου πράσινης σημαίας** στη σκηνή.
- Να χρησιμοποιήσουν το **εικονίδιο της πλήρους οθόνης** προκειμένου να προβάλλουν το έργο τους σε πλήρη οθόνη.

Υλικοτεχνική υποδομή : Ταμπλέτες με εγκατεστημένη την εφαρμογή ScratchJr, προβολικό.

Οργάνωση της τάξης : Η δραστηριότητα εκτυλίσσεται σε ομάδες. Κάθε παιδί έχει το δικό του τάμπλετ. Το τάμπλετ της ερευνήτριας συνδέεται με προβολικό, ώστε οι μαθητές να βλέπουν καλύτερα τι γίνεται στην εφαρμογή. Οι ομάδες υλοποιούν τις δραστηριότητες η μια μετά την άλλη και όχι ταυτόχρονα.

Διδακτικές προσεγγίσεις και στρατηγικές :

Θεωρητικές προσεγγίσεις : Σε αυτή τη δραστηριότητα έχουμε υιοθετήσει την εποικοδομιστική θεωρητική προσέγγιση, καθώς οι μαθητές καλούνται να επιλύσουν ένα πρόβλημα προγραμματισμού. Επιπλέον, στην παρούσα δραστηριότητα έχουμε υιοθετήσει την κοινωνικοεπικοδομιστική θεωρητική προσέγγιση, καθώς προωθείται η ανταλλαγή γνώσεων μεταξύ των μαθητών όπου είναι δυνατόν, μέσα από την συζήτηση και την αλληλεπίδραση.

Διδακτικές στρατηγικές : Ως διδακτικές στρατηγικές θα χρησιμοποιηθούν κατά κύριο λόγο η επίλυση προβλήματος, η διερεύνηση και η ανακάλυψη σε ότι αφορά την γνωριμία με την σύνταξη και την σημασιολογία της γλώσσας προγραμματισμού ScratchJr. Θα χρησιμοποιηθεί επιπλέον διδακτική βοήθεια όποτε οι μαθητές το έχουν ανάγκη για να προχωρήσουν προς την κατάκτηση της γνώσης. Η ερευνήτρια θα έχει ρόλο συνερευνητικό στην όλη διαδικασία. Ακόμα θα χρησιμοποιηθεί ως διδακτική στρατηγική η συζήτηση και αλληλεπίδραση μεταξύ των παιδιών, που στόχο έχει την ανταλλαγή γνώσεων μεταξύ των μαθητών.

Σύντομη περιγραφή της δραστηριότητας :

Σε αυτή τη δραστηριότητα, τα παιδιά καλούνται να επιλύσουν ένα προγραμματιστικό πρόβλημα χρησιμοποιώντας τις **εντολές κάθετης και οριζόντιας κίνησης**. Επιπλέον καλούνται να γνωρίσουν την **εντολή της αναπήδησης**. Έπειτα, καλούνται **να εισάγουν και**

να προγραμματίσουν δυο (2) χαρακτήρες, καθώς και να γνωρίσουν την **διαδικασία διαγραφής ενός χαρακτήρα** από την σκηνή. Επιπλέον, καλούνται **να ηχογραφήσουν μια εντολή ήχου** και να την εισάγουν στο πρόγραμμά τους. Έπειτα, οι μαθητές θα προβάλουν το έργο τους σε **πλήρη οθόνη**, κάνοντας **χρήση του εικονιδίου της πράσινης σημαίας**.

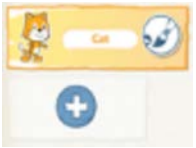
Αναλυτικά τα βήματα της δραστηριότητας :

I. Οι μαθητές αρχικά εισάγονται στο πρόγραμμα και έπειτα **σε νέο έργο**.

II. Έπειτα, οι μαθητές καλούνται να εισάγουν ένα **νέο σκηνικό**, το « Park ».

III. Στη συνέχεια, ζητείται από τους μαθητές να ανακαλύψουν πώς μπορούν να διαγράψουν τον χαρακτήρα « γάτο » από την σκηνή. Με αυτόν τον τρόπο καλούνται να ανακαλύψουν τους δυο τρόπους **διαγραφής χαρακτήρων**. Σε περίπτωση που τα παιδιά δυσκολεύονται με την διαδικασία διαγραφής χαρακτήρων θα δοθεί διδακτική βοήθεια από την ερευνήτρια.

IV. Έπειτα, η ερευνήτρια θα ρωτήσει τα παιδιά πως μπορούν να **εισάγουν ένα νέο χαρακτήρα** στη σκηνή και πιο συγκεκριμένα το χαρακτήρα « πεταλούδα ». Αν χρειαστεί θα γίνει από την ερευνήτρια υπενθύμιση της διαδικασίας εισαγωγής ενός χαρακτήρα στη σκηνή.



V. Έπειτα, θα ζητηθεί από τα παιδιά να **προγραμματίσουν τον νέο χαρακτήρα** ώστε να μετακινηθεί από τον δεύτερο κύκλο στα αριστερά στο δέντρο. Σε αυτό το σημείο να σημειώσουμε ότι θα έχουν τοποθετηθεί στη σκηνή μια πράσινη και μια κόκκινη τελεία, για να

υπενθυμίζουν στους μαθητές την αρχή και το τέλος του προγραμματισμού του χαρακτήρα « πεταλούδα ». Κάθε φορά που οι μαθητές ολοκληρώνουν μια προσπάθεια προγραμματισμού η ερευνήτρια θα τους υπενθυμίζει την **διαδικασία επαλήθευσης του προγράμματος τους** για να ελέγξουν αν έχουν προγραμματίσει σωστά.

VI. Μετά η ερευνήτρια θέτει το εξής ερώτημα στα παιδιά : « **Πώς μπορούμε να εισάγουμε τώρα τον χαρακτήρα « μύγα » στη σκηνή; ».** Στη συνέχεια, θα ζητηθεί από τα παιδιά να προγραμματίσουν τον **χαρακτήρα « μύγα »**, έτσι ώστε να κινηθεί από τον τρίτο κύκλο στα δεξιά στο παγκάκι πάνω δεξιά. Και εδώ θα έχουν τοποθετηθεί δυο τελείες, μια πράσινη και μια κόκκινη, στη σκηνή προκειμένου να υπενθυμίζουν στα παιδιά την αρχή και το τέλος του προγραμματισμού του χαρακτήρα « μύγα ».

VII. Ακόμη, θα ζητηθεί από τα παιδιά να **ηχογραφήσουν ένα ήχο** για την μύγα και να τον εισάγουν στο πρόγραμμα της. Αυτή η οδηγία στοχεύει στο να θυμηθούν οι μαθητές την **διαδικασία ηχογράφησης και εισαγωγής ήχου.**

VIII. Στη συνέχεια, θα τεθεί το ακόλουθο ερώτημα : **«Πως νομίζετε ότι μπορούμε να δούμε τους δυο χαρακτήρες να κινούνται την ίδια στιγμή εδώ που είμαστε;».** Αφήνουμε τα παιδιά να πειραματιστούν με την διεπιφάνεια χρήσης. Σε περίπτωση που τα παιδιά δεν

καταλήξουν **στην ανακάλυψη του εικονιδίου της πράσινης σημαίας**, θα γίνει παρουσίαση του από την ερευνήτρια. Σε αυτό το σημείο όσοι μαθητές δεν έχουν σχηματίσει ολοκληρωμένο σενάριο με αρχή και τέλος, τα προγράμματα των χαρακτήρων τους δεν θα τρέξουν καθόλου. Η ερευνήτρια θα εκμεταλλευτεί αυτήν τη παράλειψη για να εξηγήσει στα παιδιά **την σύνδεση που υπάρχει μεταξύ εντολής πράσινης σημαίας και εικονιδίου πράσινης σημαίας.**

IX. Έπειτα θα ζητηθεί από τα παιδιά να προβάλλουν το έργο μας σε πλήρη οθόνη : **«Πώς νομίζετε ότι μπορούμε να μεγαλώσουμε την σκηνή και να δούμε το έργο μας σε μεγαλύτερη οθόνη, σαν κινούμενο σχέδιο ;».** Αν τα παιδιά δυσκολεύονται να εντοπίσουν **το εικονίδιο της πλήρους οθόνης** θα τους το υποδείξουμε. Αφού τα παιδιά εισαχθούν σε πλήρη οθόνη, τα αφήνουμε να πειραματιστούν μέχρι να ανακαλύψουν την **λειτουργία του εικονιδίου της πράσινης σημαίας**.

X. Τέλος, γίνεται αποθήκευση έργου χρησιμοποιώντας το εικονίδιο της αποθήκευσης.

<u>4η Δραστηριότητα Διδασκαλίας :</u> Εντολή επανάληψης, Επανάληψη μιας εντολής.

Εκτιμώμενη διάρκεια : 45 λεπτά.

Προαπαιτούμενες γνώσεις : Για την υλοποίηση αυτής της δραστηριότητας απαιτείται οι μαθητές να έχουν κατανοήσει μια σειρά εννοιών και δεξιοτήτων από προηγούμενες δραστηριότητες διδασκαλίας. Σε περίπτωση που δυσκολεύονται κάπου θα γίνεται υπενθύμιση της διαδικασίας ή της έννοιας από την ερευνήτρια. Πιο συγκεκριμένα, καλό θα ήταν οι μαθητές να μπορούν να εισάγουν ένα νέο χαρακτήρα και να τον προγραμματίσουν, να

μπορούν να εισάγουν ένα νέο σκηνικό, να χρησιμοποιήσουν την εντολή της ηχογράφησης, να μπορούν να δημιουργήσουν ένα ολοκληρωμένο σενάριο χρησιμοποιώντας την αρχή και το τέλος προγραμματισμού, καθώς και να προβάλλουν το έργο τους σε πλήρη οθόνη χρησιμοποιώντας το αντίστοιχο εικονίδιο. Επίσης, πολύ σημαντικό είναι να έχουν κατανοήσει την συσχέτιση που υπάρχει μεταξύ του εικονιδίου της πράσινης σημαίας και την εντολή της πράσινης σημαίας, που δηλώνει την αρχή του προγραμματισμού.

Έννοιες : Επανάληψη μιας εντολής, παράμετρος εντολής επανάληψης.

Γνώσεις : Επανάληψη μιας εντολής, παράμετρος εντολής επανάληψης.

Δεξιότητες : Εισαγωγή εντολής στο εσωτερικό της εντολής επανάληψης, εξαγωγή εντολής από το εσωτερικό της εντολής επανάληψης, χρήση της παραμέτρου της εντολής επανάληψης.

Στόχοι της δραστηριότητας :

- Να πειραματιστούν οι μαθητές με την **εντολή της επανάληψης** προκειμένου να ανακαλύψουν την λειτουργία της.
- Να διερευνήσουν οι μαθητές την **διαδικασία εισαγωγής και εξαγωγής μιας εντολής από την εντολή της επανάληψης.**
- Να ανακαλύψουν οι μαθητές την **λειτουργία της παραμέτρου** της εντολής της επανάληψης.
- Να μπορέσουν οι μαθητές **να χρησιμοποιήσουν λειτουργικά την εντολή της επανάληψης** για να επιλύσουν ένα προγραμματιστικό πρόβλημα.

Υλικοτεχνική υποδομή : Ταμπλέτες με εγκατεστημένη την εφαρμογή ScratchJr, προβολικό, κάρτες εντολών, μαρκαδόρος για λευκοπίνακα.

Οργάνωση της τάξης : Η δραστηριότητα εκτυλίσσεται σε ομάδες. Κάθε παιδί έχει το δικό του τάμπλετ. Το τάμπλετ της ερευνήτριας συνδέεται με προβολικό, ώστε οι μαθητές να βλέπουν καλύτερα τι γίνεται στην εφαρμογή. Οι ομάδες υλοποιούν τις δραστηριότητες η μια μετά την άλλη και όχι ταυτόχρονα.

Διδακτικές προσεγγίσεις και στρατηγικές :

Θεωρητικές προσεγγίσεις : Σε αυτή τη δραστηριότητα έχουμε υιοθετήσει την εποικοδομιστική θεωρητική προσέγγιση, καθώς οι μαθητές καλούνται να επιλύσουν ένα πρόβλημα προγραμματισμού. Επιπλέον, στην παρούσα δραστηριότητα έχουμε υιοθετήσει την κοινωνικοεπικοδομιστική θεωρητική προσέγγιση, καθώς προωθείται η ανταλλαγή γνώσεων μεταξύ των μαθητών όπου είναι δυνατόν, μέσα από την συζήτηση και την αλληλεπίδραση.

Διδακτικές στρατηγικές : Ως διδακτικές στρατηγικές θα χρησιμοποιηθούν κατά κύριο λόγο η επίλυση προβλήματος, η διερεύνηση και η ανακάλυψη σε ότι αφορά την γνωριμία με την σύνταξη και την σημασιολογία της εντολής της επανάληψης. Θα χρησιμοποιηθεί επιπλέον διδακτική βοήθεια όποτε οι μαθητές το έχουν ανάγκη για να προχωρήσουν προς την κατάκτηση της γνώσης. Η ερευνήτρια θα έχει ρόλο συνερευνητικό στην όλη διαδικασία. Ακόμα θα χρησιμοποιηθεί ως διδακτική στρατηγική η συζήτηση και αλληλεπίδραση μεταξύ των παιδιών, που στόχο έχει την ανταλλαγή γνώσεων μεταξύ των μαθητών.

Σύντομη περιγραφή της δραστηριότητας : Σε αυτή τη δραστηριότητα, τα παιδιά καλούνται να **γνωρίσουν και να διερευνήσουν την λειτουργία της επανάληψης**, που αποτελεί μια έννοια υψηλού επιπέδου στον προγραμματισμό. Καλούνται επιπλέον να **αναπτύξουν δεξιότητες που σχετίζονται με την χρήση της εντολής της επανάληψης**, όπως για παράδειγμα να εισάγουν μια εντολή μέσα στην εντολή της επανάληψης ή να εξάγουν μια εντολή από την εντολή της επανάληψης. Στη συνέχεια, καλούνται **να επιλύσουν ένα προγραμματιστικό πρόβλημα χρησιμοποιώντας την εντολή της επανάληψης και τις κάρτες εντολών**. Έπειτα, οι μαθητές θα προβάλουν το έργο τους σε **πλήρη οθόνη**, κάνοντας χρήση του **εικονιδίου της πράσινης σημαίας** και θα το **αποθηκεύσουν χρησιμοποιώντας το εικονίδιο με το σπιτάκι**.

Αναλυτικά τα βήματα της δραστηριότητας :

I. Αρχικά, η ερευνήτρια ζητά από τους μαθητές **να ανοίξουν το λογισμικό ScratchJr** και **να ανοίξουν το έργο που τους δείχνει στο προβολικό** και προϋπάρχει στην διεπιφάνεια χρήσης.

II. Στη συνέχεια, οι μαθητές καλούνται **να παρατηρήσουν τι υπάρχει στο χώρο προγραμματισμού**.

III. Έπειτα, η ερευνήτρια χρησιμοποιώντας κατάλληλες ερωτήσεις παροτρύνει τους μαθητές **να διερευνήσουν και να ανακαλύψουν ποια είναι η λειτουργία αυτής της νέας εντολής**.

- Τι υπάρχει αυτή τη στιγμή στο χώρο προγραμματισμού; Παρατηρείτε κάτι καινούργιο ;
- Πως μπορούμε να ανακαλύψουμε τι κάνει αυτή η νέα εντολή; Δοκιμάστε να πατήσετε πάνω στην νέα εντολή να δούμε τι κάνει. Τι γίνεται; Κάνει κάτι;(διδακτική βοήθεια)
- Τι πιστεύετε ότι μπορούμε να κάνουμε με αυτές τις εντολές και την καινούρια εντολή; Καμία ιδέα; Μήπως να δοκιμάσουμε να τοποθετήσουμε μια, μια τις εντολές να δούμε τι θα γίνει;
- Ξεκινήστε με την πρώτη εντολή. Πατήστε πάνω στην μπλε εντολή. Τι κάνει ο γάτος; Για δοκιμάστε να τοποθετήσετε την μπλε εντολή μέσα στην πορτοκαλί. Τι συμβαίνει; Τι κάνει ο γάτος τώρα; Τι διαφορετικό παρατηρείτε τώρα στο χώρο προγραμματισμού;
- Ας δοκιμάσουμε τώρα να βγάλουμε την μπλε εντολή και να την τοποθετήσουμε πάλι στο πλάι. Πατήστε τώρα πάνω στην άλλη μπλε εντολή που πάει προς τα πάνω. Τι κάνει ο γάτος τώρα; Ας τοποθετήσουμε τώρα αυτή την εντολή μέσα στην πορτοκαλί εντολή. Τι συμβαίνει τώρα; Τι κάνει ο γάτος αυτή τη φορά; Τι παρατηρείτε τώρα;

- Ας δοκιμάσουμε τώρα να πατήσουμε πάνω στην εντολή ήχου. Τι παρατηρείτε; Τι κάνει ο γάτος; Ας δοκιμάσουμε τώρα να τοποθετήσουμε την ίδια εντολή μέσα στην πορτοκαλί εντολή. Τι συμβαίνει τώρα; Τι κάνει τώρα ο γάτος; (διδακτική βοήθεια)
- Πόσες φορές λέει «γεια» ο γάτος; Μπορείτε να τις μετρήσετε; Γιατί άραγε όταν η πράσινη εντολή είναι μέσα στην πορτοκαλί ο χαρακτήρας μιλάει (4)τέσσερεις φορές ενώ όταν την αφαιρέσουμε μιλάει μια μόνο;(διδακτική βοήθεια)
- Άρα, τι παρατηρούμε; Τι μπορεί να κάνει λοιπόν η πορτοκαλί εντολή; (διδακτική βοήθεια)
- Πώς θα μπορούσαμε να ακούσουμε τον ήχο τρεις φορές αντί για τέσσερεις χρησιμοποιώντας την πορτοκαλί εντολή;

IV. Σε περίπτωση που τα παιδιά δεν κατορθώσουν μέσα από τον πειραματισμό με την εντολή να ανακαλύψουν την λειτουργία της εντολής αλλά και της παραμέτρου θα γίνει διδασκαλία από την ερευνήτρια, καθώς η εντολή της επανάληψης είναι υψηλού επιπέδου και συχνά η διδασκαλία κρίνεται απαραίτητη.

V. Στη συνέχεια, θα δοθεί στους μαθητές **ένα απλό πρόβλημα προς επίλυση** με την οδηγία να χρησιμοποιήσουν την εντολή της επανάληψης για να το επιλύσουν. Πιο συγκεκριμένα, η ερευνήτρια θα προβάλλει στο προβολικό ένα έτοιμο έργο με δυο χαρακτήρες σε πλήρη οθόνη και θα ζητήσει από τους μαθητές **να αναλύσουν την κίνηση των χαρακτήρων χρησιμοποιώντας τις κάρτες εντολών και να δημιουργήσουν το ίδιο έργο στα τάμπλετ τους χρησιμοποιώντας την εντολή της επανάληψης.**

VI. Το έργο που καλούνται να δημιουργήσουν οι μαθητές περιλαμβάνει δυο χαρακτήρες τον «κάβουρα» και τον «αστερία» στο σκηνικό « Beach day ». Οι μαθητές πρέπει **να προγραμματίσουν τον χαρακτήρα «κάβουρα»** ώστε να μετακινηθεί δεξιά τέσσερεις φορές και να χοροπηδήσει μια φορά. Έπειτα, **πρέπει να προγραμματίσουν τον χαρακτήρα «αστερία»**, ώστε να φωνάξει « γεια » τρεις φορές, χρησιμοποιώντας την εντολή της ηχογράφησης, και να προχωρήσει αριστερά δυο βήματα. Όπου οι μαθητές δυσκολεύονται θα δίνεται διδακτική βοήθεια από την ερευνήτρια, η οποία όπου χρειάζεται θα υπενθυμίζει επίσης στους μαθητές την **επαλήθευση του προγράμματος**.

VII. Στη συνέχεια, θα τεθεί το ακόλουθο ερώτημα : « **Πως νομίζετε ότι μπορούμε να δούμε τους δυο χαρακτήρες να κινούνται την ίδια στιγμή εδώ που είμαστε ;»**. Αφήνουμε τα παιδιά να πειραματιστούν με την διεπιφάνεια χρήσης. Σε περίπτωση που τα παιδιά δεν καταλήξουν στην ανακάλυψη **του εικονιδίου της πράσινης σημαίας**, θα γίνει παρουσίαση του από την ερευνήτρια. Σε αυτό το σημείο όσοι μαθητές δεν έχουν σχηματίσει ολοκληρωμένο σενάριο με αρχή και τέλος, τα προγράμματα των χαρακτήρων τους δεν θα τρέξουν καθόλου. Η ερευνήτρια θα εκμεταλλευτεί αυτήν τη παράλειψη για να εξηγήσει στα παιδιά **την σύνδεση που υπάρχει μεταξύ εντολής πράσινης σημαίας και εικονιδίου πράσινης σημαίας**.

VIII. Έπειτα θα ζητηθεί από τα παιδιά να προβάλλουν το έργο μας σε πλήρη οθόνη : **«Πώς νομίζετε ότι μπορούμε να μεγαλώσουμε την σκηνή και να δούμε το έργο μας σε μεγαλύτερη οθόνη, σαν κινούμενο σχέδιο;»**. Αν τα παιδιά δυσκολεύονται να εντοπίσουν **το εικονίδιο της πλήρους οθόνης** θα τους το υποδείξουμε. Αφού τα παιδιά εισαχθούν σε πλήρη

οθόνη, τα αφήνουμε να πειραματιστούν μέχρι να ανακαλύψουν την **λειτουργία του εικονιδίου της πράσινης σημαίας**.

IX. Τέλος, γίνεται **αποθήκευση έργου** χρησιμοποιώντας το εικονίδιο της αποθήκευσης.

5η i. Δραστηριότητα Διδασκαλίας

Εκτιμώμενη διάρκεια : 25 λεπτά

Προαπαιτούμενες γνώσεις : Για την υλοποίηση αυτής της δραστηριότητας απαιτείται οι μαθητές να έχουν κατανοήσει μια σειρά εννοιών και δεξιοτήτων από προηγούμενες δραστηριότητες διδασκαλίας. Σε περίπτωση που δυσκολεύονται κάπου θα γίνεται υπενθύμιση της διαδικασίας ή της έννοιας από την ερευνήτρια. Πιο συγκεκριμένα, καλό θα ήταν οι μαθητές να μπορούν να εισάγουν ένα νέο χαρακτήρα και να τον προγραμματίσουν, να μπορούν να εισάγουν ένα νέο σκηνικό, να χρησιμοποιήσουν την εντολή της ηχογράφησης, να μπορούν να δημιουργήσουν ένα ολοκληρωμένο σενάριο χρησιμοποιώντας την αρχή και το τέλος προγραμματισμού, καθώς και να προβάλλουν το έργο τους σε πλήρη οθόνη χρησιμοποιώντας το αντίστοιχο εικονίδιο. Επίσης, πολύ σημαντικό είναι να έχουν κατανοήσει την συσχέτιση που υπάρχει μεταξύ του εικονιδίου της πράσινης σημαίας και την εντολή της πράσινης σημαίας, που δηλώνει την αρχή του προγραμματισμού. Αναφορικά με την εντολή της επανάληψης καλό θα ήταν οι μαθητές να έχουν κατανοήσει την λειτουργία της επανάληψης.

Έννοιες : επανάληψη ενός μοτίβου εντολών, παράμετρος εντολής επανάληψης για μοτίβο εντολών.

Γνώσεις : επανάληψη ενός μοτίβου εντολών, σημασιολογία της εντολής επανάληψης ενός μοτίβου εντολών, παράμετρος εντολής επανάληψης για μοτίβο εντολών.

Δεξιότητες : Εισαγωγή και εξαγωγή ενός μοτίβου εντολών στην εντολή της επανάληψης, χρήση της παραμέτρου της εντολής επανάληψης για μοτίβο εντολών.

Στόχοι της δραστηριότητας :

- Να πειραματιστούν οι μαθητές με την **επανάληψη ενός μοτίβου εντολών**.
- Να διερευνήσουν οι μαθητές την **διαδικασία εισαγωγής και εξαγωγής ενός μοτίβου εντολών στην εντολή της επανάληψης**.

- Να ανακαλύψουν οι μαθητές την λειτουργία της **παραμέτρου της εντολής της επανάληψης για ένα μοτίβο εντολών.**
- Να μπορέσουν οι μαθητές να **χρησιμοποιήσουν λειτουργικά την εντολή της επανάληψης ενός μοτίβου εντολών** για να επιλύσουν ένα προγραμματιστικό πρόβλημα.

Υλικοτεχνική υποδομή : Ταμπλέτες με εγκατεστημένη την εφαρμογή ScratchJr, προβολικό, κάρτες εντολών, μαρκαδόρος για λευκοπίνακα.

Οργάνωση τάξης : Η δραστηριότητα εκτυλίσσεται σε ομάδες. Κάθε παιδί έχει το δικό του τάμπλετ. Το τάμπλετ της ερευνήτριας συνδέεται με προβολικό, ώστε οι μαθητές να βλέπουν καλύτερα τι γίνεται στην εφαρμογή. Οι ομάδες υλοποιούν τις δραστηριότητες η μια μετά την άλλη και όχι ταυτόχρονα.

Διδακτικές προσεγγίσεις και στρατηγικές :

Θεωρητικές προσεγγίσεις : Σε αυτή τη δραστηριότητα έχουμε υιοθετήσει την εποικοδομιστική θεωρητική προσέγγιση, καθώς οι μαθητές καλούνται να επιλύσουν ένα πρόβλημα προγραμματισμού. Επιπλέον, στην παρούσα δραστηριότητα έχουμε υιοθετήσει την κοινωνικοεπικοδομιστική θεωρητική προσέγγιση, καθώς προωθείται η ανταλλαγή γνώσεων μεταξύ των μαθητών όπου είναι δυνατόν, μέσα από την συζήτηση και την αλληλεπίδραση.

Διδακτικές στρατηγικές : Ως διδακτικές στρατηγικές θα χρησιμοποιηθούν κατά κύριο λόγο η επίλυση προβλήματος, η διερεύνηση και η ανακάλυψη σε ότι αφορά την γνωριμία με την σύνταξη και την σημασιολογία της εντολής της επανάληψης. Θα χρησιμοποιηθεί επιπλέον διδακτική βοήθεια όποτε οι μαθητές το έχουν ανάγκη για να προχωρήσουν προς την κατάκτηση της γνώσης. Η ερευνήτρια θα έχει ρόλο συνερευνητικό στην όλη διαδικασία. Ακόμα θα χρησιμοποιηθεί ως διδακτική στρατηγική η συζήτηση και αλληλεπίδραση μεταξύ των παιδιών, που στόχο έχει την ανταλλαγή γνώσεων μεταξύ των μαθητών.

Σύντομη περιγραφή της δραστηριότητας :

Σε αυτή τη δραστηριότητα, τα παιδιά καλούνται να γνωρίσουν και να διερευνήσουν την λειτουργία της επανάληψης για ένα μοτίβο εντολών. Καλούνται επιπλέον να αναπτύξουν δεξιότητες που σχετίζονται με την χρήση της εντολής της επανάληψης, όπως για παράδειγμα να εισάγουν ένα μοτίβο εντολών μέσα στην εντολή της επανάληψης ή να εξάγουν ένα μοτίβο εντολών από την εντολή της επανάληψης.

Αναλυτικά τα βήματα της δραστηριότητας :

I. Αρχικά, η ερευνήτρια ζητά από τους μαθητές **να ανοίξουν το λογισμικό ScratchJr** και **να ανοίξουν το έργο που τους δείχνει στο προβολικό** και προϋπάρχει στην διεπιφάνεια χρήσης.

II. Στη συνέχεια, οι μαθητές καλούνται να παρατηρήσουν τις εντολές που υπάρχουν στο χώρο προγραμματισμού.

III. Έπειτα, η ερευνήτρια χρησιμοποιώντας κατάλληλες ερωτήσεις παροτρύνει τους μαθητές να διερευνήσουν και να ανακαλύψουν ποια είναι η λειτουργία αυτής της νέας εντολής.

- Τι παρατηρείτε στο χώρο προγραμματισμού; Τι κάνει ο βάτραχος; (διδακτική βοήθεια)
- Τι επαναλαμβάνεται στο σενάριο του βατράχου;
- Μπορείτε να εντοπίσετε τις εντολές χρησιμοποιώντας τις κάρτες;
- Μπορείτε να δημιουργήσετε το σενάριο του βατράχου με τις κάρτες πριν περάσετε στο τάμπλετ;
- Αυτό που επαναλαμβάνεται το ονομάζουμε μοτίβο. Πόσες φορές επαναλαμβάνεται το μοτίβο;

- Πως αλλιώς μπορούμε να επαναλάβουμε το μοτίβο χωρίς να χρησιμοποιήσουμε τόσες εντολές;
- Μήπως να δοκιμάσουμε την εντολή που μάθαμε την προηγούμενη φορά; Την εντολή της επανάληψης; Πώς θα την χρησιμοποιήσουμε; Τι πρέπει να κάνουμε; (διδακτική βοήθεια)

IV. Σε περίπτωση που τα παιδιά δυσκολεύονται με την χρήση της εντολής επανάληψης θα δοθεί διδακτική βοήθεια από την ερευνήτρια όπου χρειάζεται.

5η ii. Δραστηριότητα Εμπέδωσης

Εκτιμώμενη διάρκεια : 20 λεπτά.

Προαπαιτούμενες γνώσεις : Για την υλοποίηση αυτής της δραστηριότητας απαιτείται οι μαθητές να έχουν κατανοήσει μια σειρά εννοιών και δεξιοτήτων από προηγούμενες δραστηριότητες διδασκαλίας. Σε περίπτωση που δυσκολεύονται κάπου θα γίνεται υπενθύμιση της διαδικασίας ή της έννοιας από την ερευνήτρια. Πιο συγκεκριμένα, καλό θα ήταν οι μαθητές να μπορούν να εισάγουν ένα νέο χαρακτήρα και να τον προγραμματίσουν, να μπορούν να εισάγουν ένα νέο σκηνικό, να χρησιμοποιήσουν την εντολή της ηχογράφησης, να μπορούν να δημιουργήσουν ένα ολοκληρωμένο σενάριο χρησιμοποιώντας την αρχή και το τέλος προγραμματισμού, καθώς και να προβάλλουν το έργο τους σε πλήρη οθόνη χρησιμοποιώντας το αντίστοιχο εικονίδιο. Επίσης, πολύ σημαντικό είναι να έχουν κατανοήσει την συσχέτιση που υπάρχει μεταξύ του εικονιδίου της πράσινης σημαίας και την εντολή της πράσινης σημαίας, που δηλώνει την αρχή του προγραμματισμού. Αναφορικά με την εντολή της επανάληψης καλό θα ήταν οι μαθητές να έχουν κατανοήσει την λειτουργία της επανάληψης.

Έννοιες : επανάληψη ενός μοτίβου εντολών, παράμετρος εντολής επανάληψης για μοτίβο εντολών.

Γνώσεις : επανάληψη ενός μοτίβου εντολών, σημασιολογία της εντολής επανάληψης ενός μοτίβου εντολών, παράμετρος εντολής επανάληψης για μοτίβο εντολών.

Δεξιότητες : Εισαγωγή και εξαγωγή ενός μοτίβου εντολών στην εντολή της επανάληψης, χρήση της παραμέτρου της εντολής επανάληψης για μοτίβο εντολών.

Στόχοι της δραστηριότητας :

- Να μπορέσουν οι μαθητές να χρησιμοποιήσουν λειτουργικά την εντολή της επανάληψης ενός μοτίβου εντολών για να επιλύσουν ένα προγραμματιστικό πρόβλημα.

Υλικοτεχνική υποδομή : Ταμπλέτες με εγκατεστημένη την εφαρμογή ScratchJr, προβολικό, κάρτες εντολών, μαρκαδόρος για λευκοπίνακα.

Οργάνωση τάξης : Η δραστηριότητα εκτυλίσσεται σε ομάδες. Κάθε παιδί έχει το δικό του τάμπλετ. Το τάμπλετ της ερευνήτριας συνδέεται με προβολικό, ώστε οι μαθητές να βλέπουν καλύτερα τι γίνεται στην εφαρμογή. Οι ομάδες υλοποιούν τις δραστηριότητες η μια μετά την άλλη και όχι ταυτόχρονα.

Διδακτικές προσεγγίσεις και στρατηγικές :

Θεωρητικές προσεγγίσεις : Σε αυτή τη δραστηριότητα έχουμε υιοθετήσει την εποικοδομιστική θεωρητική προσέγγιση, καθώς οι μαθητές καλούνται να επιλύσουν ένα πρόβλημα προγραμματισμού. Επιπλέον, στην παρούσα δραστηριότητα έχουμε υιοθετήσει την κοινωνικοεπικοδομιστική θεωρητική προσέγγιση, καθώς προωθείται η ανταλλαγή γνώσεων μεταξύ των μαθητών όπου είναι δυνατόν, μέσα από την συζήτηση και την αλληλεπίδραση.

Διδακτικές στρατηγικές : Ως διδακτικές στρατηγικές θα χρησιμοποιηθούν κατά κύριο λόγο η επίλυση προβλήματος. Θα χρησιμοποιηθεί επιπλέον διδακτική βοήθεια όποτε οι μαθητές το έχουν ανάγκη για να προχωρήσουν προς την κατάκτηση της γνώσης. Η ερευνήτρια θα έχει ρόλο συνερευνητικό στην όλη διαδικασία. Ακόμα θα χρησιμοποιηθεί ως διδακτική στρατηγική η συζήτηση και αλληλεπίδραση μεταξύ των παιδιών, που στόχο έχει την ανταλλαγή γνώσεων μεταξύ των μαθητών.

Σύντομη περιγραφή δραστηριότητας :

Οι μαθητές καλούνται **να επιλύσουν ένα προγραμματιστικό πρόβλημα με την βοήθεια των καρτών χρησιμοποιώντας την εντολή της επανάληψης για μοτίβο εντολών.**

Αναλυτικά τα βήματα της δραστηριότητας :

I. Αρχικά, θα δοθεί στους μαθητές **ένα απλό πρόβλημα προς επίλυση** με την οδηγία να χρησιμοποιήσουν την εντολή της επανάληψης για να το επιλύσουν. Πιο συγκεκριμένα, η ερευνήτρια θα προβάλλει στο προβολικό ένα έτοιμο έργο σε πλήρη οθόνη και θα ζητήσει από τους μαθητές **να αναλύσουν την κίνηση του χαρακτήρα χρησιμοποιώντας την ψευδογλώσσα και να δημιουργήσουν το ίδιο έργο στα τάμπλετ τους χρησιμοποιώντας την εντολή της επανάληψης.**

II. Πιο συγκεκριμένα, θα τεθούν μια σειρά από ερωτήσεις στους μαθητές :

- Τι παρατηρείτε στο χώρο προγραμματισμού; Τι κάνει ο γάτος; (διδακτική βοήθεια)
- Ποιες εντολές επαναλαμβάνονται στο σενάριο του γάτου;
- Μπορείτε να εντοπίσετε τις εντολές χρησιμοποιώντας τις κάρτες;
- Μπορείτε να δημιουργήσετε το σενάριο του γάτου με τις κάρτες πριν περάσετε στο τάμπλετ;
- Πόσες φορές επαναλαμβάνεται το μοτίβο εντολών από το γατούλη;
- Πως αλλιώς μπορούμε να επαναλάβουμε το μοτίβο χωρίς να χρησιμοποιήσουμε τόσες εντολές;
- Μήπως να δοκιμάσουμε την εντολή που μάθαμε την προηγούμενη φορά; Την εντολή της επανάληψης; Πώς θα την χρησιμοποιήσουμε; Τι πρέπει να κάνουμε; (διδακτική βοήθεια)

III. Σε περίπτωση που τα παιδιά δυσκολεύονται με την χρήση της εντολής επανάληψης θα δοθεί διδακτική βοήθεια από την ερευνήτρια όπου χρειάζεται.

IV. Στη συνέχεια, θα τεθεί το ακόλουθο ερώτημα : « **Πως νομίζετε ότι μπορούμε να δούμε τον χαρακτήρα να κινείται εδώ που είμαστε;** ». Αφήνουμε τα παιδιά να πειραματιστούν με την διεπιφάνεια χρήσης. Σε περίπτωση που τα παιδιά δεν καταλήξουν

στην ανακάλυψη του **εικονιδίου της πράσινης σημαίας**, θα γίνει παρουσίαση του από την ερευνήτρια. Σε αυτό το σημείο όσοι μαθητές δεν έχουν σχηματίσει ολοκληρωμένο σενάριο με αρχή και τέλος, τα προγράμματα των χαρακτήρων τους δεν θα τρέξουν καθόλου. Η ερευνήτρια θα εκμεταλλευτεί αυτήν τη παράλειψη για να εξηγήσει στα παιδιά την **σύνδεση που υπάρχει μεταξύ εντολής πράσινης σημαίας και εικονιδίου πράσινης σημαίας.**

V. Έπειτα θα ζητηθεί από τα παιδιά να προβάλλουν το έργο μας σε πλήρη οθόνη : « Πώς νομίζετε ότι μπορούμε να μεγαλώσουμε την σκηνή και να δούμε το έργο μας σε μεγαλύτερη οθόνη, σαν κινούμενο σχέδιο; ». Αν τα παιδιά δυσκολεύονται να εντοπίσουν το εικονίδιο της πλήρους οθόνης θα τους το υποδείξουμε. Αφού τα παιδιά εισαχθούν σε πλήρη οθόνη, τα αφήνουμε να πειραματιστούν μέχρι να ανακαλύψουν την λειτουργία του εικονιδίου της πράσινης σημαίας.

VI. Τέλος, γίνεται αποθήκευση έργου χρησιμοποιώντας το εικονίδιο της αποθήκευσης

6η Δραστηριότητα Διδασκαλίας

Εκτιμώμενη διάρκεια : 45 λεπτά

Προαπαιτούμενες γνώσεις : Για την υλοποίηση αυτής της δραστηριότητας απαιτείται οι μαθητές να έχουν κατανοήσει μια σειρά εννοιών και δεξιοτήτων από προηγούμενες δραστηριότητες διδασκαλίας. Σε περίπτωση που δυσκολεύονται κάπου θα γίνεται υπενθύμιση της διαδικασίας ή της έννοιας από την ερευνήτρια. Πιο συγκεκριμένα, καλό θα ήταν οι μαθητές να μπορούν να εισάγουν ένα νέο χαρακτήρα και να τον προγραμματίσουν, να μπορούν να εισάγουν ένα νέο σκηνικό, να χρησιμοποιήσουν την εντολή της ηχογράφησης, να μπορούν να δημιουργήσουν ένα ολοκληρωμένο σενάριο χρησιμοποιώντας την αρχή και το τέλος προγραμματισμού, καθώς και να προβάλλουν το έργο τους σε πλήρη οθόνη χρησιμοποιώντας το αντίστοιχο εικονίδιο. Επίσης, πολύ σημαντικό είναι να έχουν κατανοήσει την συσχέτιση που υπάρχει μεταξύ του εικονιδίου της πράσινης σημαίας και την εντολή της πράσινης σημαίας, που δηλώνει την αρχή του προγραμματισμού.

Έννοιες : στέλνω ένα μήνυμα, λαμβάνω ένα μήνυμα.

Γνώσεις : εντολή «στέλνω ένα μήνυμα», εντολή «λαμβάνω μήνυμα», σημασιολογία εντολής στέλνω ένα μήνυμα για τον χαρακτήρα που στέλνει το μήνυμα, σημασιολογία εντολής λαμβάνω ένα μήνυμα για τον χαρακτήρα που λαμβάνει το μήνυμα, συσχετισμός εντολής λαμβάνω ένα μήνυμα με την ύπαρξη της εντολής στέλνω ένα μήνυμα στον άλλο χαρακτήρα.

Δεξιότητες : χρήση των δυο εντολών μηνυμάτων σε δυο χαρακτήρες.

Στόχοι της δραστηριότητας :

- Να πειραματιστούν οι μαθητές με τις εντολές « **στέλνω ένα μήνυμα » και « ανοίγω ένα μήνυμα ».**
- Να διερευνήσουν οι μαθητές την **λειτουργία της εντολής « στέλνω ένα μήνυμα» και «ανοίγω ένα μήνυμα ».**
- Να ανακαλύψουν οι μαθητές την λειτουργία των εντολών αυτών.

- Να μπορέσουν οι μαθητές να **χρησιμοποιήσουν λειτουργικά τις εντολές των μηνυμάτων σε δυο χαρακτήρες** για να επιλύσουν ένα προγραμματιστικό πρόβλημα.

Υλικοτεχνική υποδομή : Ταμπλέτες με εγκατεστημένη την εφαρμογή ScratchJr, προβολικό, κάρτες εντολών, μαρκαδόρος για λευκοπίνακα.

Οργάνωση τάξης : Η δραστηριότητα εκτυλίσσεται σε ομάδες. Κάθε παιδί έχει το δικό του τάμπλετ. Το τάμπλετ της ερευνήτριας συνδέεται με προβολικό, ώστε οι μαθητές να βλέπουν καλύτερα τι γίνεται στην εφαρμογή. Οι ομάδες υλοποιούν τις δραστηριότητες η μια μετά την άλλη και όχι ταυτόχρονα.

Διδακτικές προσεγγίσεις και στρατηγικές :

Θεωρητικές προσεγγίσεις : Σε αυτή τη δραστηριότητα έχουμε υιοθετήσει την εποικοδομιστική θεωρητική προσέγγιση, καθώς οι μαθητές καλούνται να επιλύσουν ένα πρόβλημα προγραμματισμού. Επιπλέον, στην παρούσα δραστηριότητα έχουμε υιοθετήσει την κοινωνικοεπικοδομιστική θεωρητική προσέγγιση, καθώς προωθείται η ανταλλαγή γνώσεων μεταξύ των μαθητών όπου είναι δυνατόν, μέσα από την συζήτηση και την αλληλεπίδραση.

Διδακτικές στρατηγικές : Ως διδακτικές στρατηγικές θα χρησιμοποιηθούν κατά κύριο λόγο η επίλυση προβλήματος, η διερεύνηση και η ανακάλυψη σε ότι αφορά την γνωριμία με την σύνταξη και την σημασιολογία των εντολών των μηνυμάτων. Θα χρησιμοποιηθεί επιπλέον διδακτική βοήθεια όποτε οι μαθητές το έχουν ανάγκη για να προχωρήσουν προς την κατάκτηση της γνώσης. Η ερευνήτρια θα έχει ρόλο συνερευνητικό στην όλη διαδικασία. Ακόμα θα χρησιμοποιηθεί ως διδακτική στρατηγική η συζήτηση και αλληλεπίδραση μεταξύ των παιδιών, που στόχο έχει την ανταλλαγή γνώσεων μεταξύ των μαθητών.

Σύντομη περιγραφή δραστηριότητας :
Σε αυτή τη δραστηριότητα, τα παιδιά καλούνται **να γνωρίσουν και να διερευνήσουν την λειτουργία των μηνυμάτων**. Καλούνται επιπλέον να **αναπτύξουν δεξιότητες που σχετίζονται με την χρήση των εντολών των μηνυμάτων.**

Αναλυτικά τα βήματα της δραστηριότητας :

I. Αρχικά, η ερευνήτρια ζητά από τα παιδιά να ανοίξουν τα τάμπλετ και το λογισμικό και να εντοπίσουν το έργο που προβάλλεται στο προβολικό.

II. Στη συνέχεια, η ερευνήτρια ζητά από τα παιδιά να παρατηρήσουν τον χώρο προγραμματισμού και να εντοπίσουν αν υπάρχει κάποια εντολή που δεν γνωρίζουν.

III. Έπειτα, η ερευνήτρια ζητά από τα παιδιά να ανοίξουν το έργο σε πλήρη οθόνη να παρατηρήσουν το πρόγραμμα και να ανακαλύψουν πώς λειτουργεί η νέα εντολή και τι διαφορετικό παρατηρούν όταν εκκινούν το πρόγραμμα με την πράσινη σημαία. Σε περίπτωση που δεν θυμούνται πώς εκκινούμε ένα πρόγραμμα θα γίνει υπενθύμιση του αντίστοιχου εικονιδίου από την ερευνήτρια.

IV. Παράλληλα η ερευνήτρια εκτελεί το πρόγραμμα σε πλήρη οθόνη με την βοήθεια του προβολικού, έτσι ώστε οι μαθητές να μπορέσουν να παρατηρήσουν τι γίνεται στα σενάρια των δυο χαρακτήρων. Επιπλέον, σε αυτό το σημείο θα γίνει χρήση της ψευδογλώσσας,

προκειμένου να διευκολυνθούν οι μαθητές στο να παρατηρήσουν ταυτόχρονα τα σενάρια των δυο χαρακτήρων, καθώς το λογισμικό δίνει την δυνατότητα να εξετάσεις το σενάριο του ενός χαρακτήρα και όχι και των δυο ταυτόχρονα.

V. Τα παιδιά πρέπει να εστιάσουν την προσοχή τους στα εξής :

- Στο γεγονός ότι οι δυο χαρακτήρες επικοινωνούν με την βοήθεια των μηνυμάτων.
- Στο γεγονός ότι οι δυο χαρακτήρες δεν ξεκινούν ταυτόχρονα.
- Στο γεγονός ότι ο γάτος ξεκινάει πριν το διαστημόπλοιο.
- Στο γεγονός ότι το διαστημόπλοιο ξεκινάει μόνο μόλις ο γάτος του στείλει το μήνυμα.
- Στο γεγονός ότι το διαστημόπλοιο δεν ξεκινάει αν αφαιρέσουμε είτε την εντολή « στέλνω ένα μήνυμα » από το πρόγραμμα του γάτου, είτε την εντολή « λαμβάνω ένα μήνυμα » από το πρόγραμμα του διαστημόπλοιου.
- Στο γεγονός ότι για να λειτουργήσουν τα μηνύματα πρέπει να χρησιμοποιηθούν και οι δυο εντολές, η εντολή « στέλνω ένα μήνυμα » στο χαρακτήρα που ξεκινάει πρώτος και η εντολή « λαμβάνω ένα μήνυμα » στο χαρακτήρα που ξεκινάει δεύτερος.

VI. Έπειτα, η ερευνήτρια ζητά από τους μαθητές να κλείσουν τα μάτια τους και εκείνη αφαιρεί την εντολή « στέλνω ένα μήνυμα » από το χαρακτήρα γάτο, προκειμένου οι μαθητές να παρατηρήσουν ότι το διαστημόπλοιο δεν ξεκινάει το πρόγραμμα του πια και να εντοπίσουν το γιατί.

VII. Η ερευνήτρια θα επαναλάβει την ίδια διαδικασία και για το διαστημόπλοιο προκείμενου να αντιληφθούν οι μαθητές ότι η εκκίνηση του προγράμματος του διαστημόπλοιου εξαρτάται από τον χαρακτήρα γάτο.

VIII. Η ίδια διαδικασία θα επαναληφθεί και για το δεύτερο έργο με τον χαρακτήρα παιδί και τον χαρακτήρα μπάλα. Δηλαδή η ερευνήτρια θα ζητήσει σε πρώτο επίπεδο από τους μαθητές να ανοίξουν το έργο σε πλήρη οθόνη και να παρατηρήσουν τι γίνεται εκεί.

IX. Τα παιδιά πρέπει να εστιάσουν και πάλι την προσοχή τους στα εξής :

- Στο γεγονός ότι οι δυο χαρακτήρες επικοινωνούν με την βοήθεια των μηνυμάτων.
- Στο γεγονός ότι οι δυο χαρακτήρες δεν ξεκινούν ταυτόχρονα.
- Στο γεγονός ότι το παιδί ξεκινάει πριν την μπάλα.
- Στο γεγονός ότι η μπάλα ξεκινάει μόνο μόλις το παιδί του στείλει το μήνυμα.
- Στο γεγονός ότι η μπάλα δεν ξεκινάει αν αφαιρέσουμε είτε την εντολή « στέλνω ένα μήνυμα » από το πρόγραμμα του παιδιού, είτε την εντολή « λαμβάνω ένα μήνυμα » από το πρόγραμμα της μπάλας.
- Στο γεγονός ότι για να λειτουργήσουν τα μηνύματα πρέπει να χρησιμοποιηθούν και οι δυο εντολές, η εντολή « στέλνω ένα μήνυμα » στο χαρακτήρα που ξεκινάει πρώτος και η εντολή «λαμβάνω ένα μήνυμα» στο χαρακτήρα που ξεκινάει δεύτερος.
- Στο γεγονός ότι το τέρμα έχει διακοσμητικό ρόλο.

X. Τέλος, οι μαθητές καλούνται να δοκιμάσουν να αναπαράγουν το δεύτερο έργο έτσι ώστε να εξοικειωθούν με την χρήση των μηνυμάτων και εμείς να εντοπίσουμε τυχόν προβλήματα ή δυσκολίες που αντιμετωπίζουν.

7η Δραστηριότητα Εμπέδωσης

Εκτιμώμενη διάρκεια : 45 λεπτά

Προαπαιτούμενες γνώσεις : Για την υλοποίηση αυτής της δραστηριότητας απαιτείται οι μαθητές να έχουν κατανοήσει μια σειρά εννοιών και δεξιοτήτων από προηγούμενες δραστηριότητες διδασκαλίας. Σε περίπτωση που δυσκολεύονται κάπου θα γίνεται υπενθύμιση της διαδικασίας ή της έννοιας από την ερευνήτρια. Πιο συγκεκριμένα, καλό θα ήταν οι μαθητές να μπορούν να εισάγουν ένα νέο χαρακτήρα και να τον προγραμματίσουν, να μπορούν να εισάγουν ένα νέο σκηνικό, να χρησιμοποιήσουν την εντολή της ηχογράφησης, να μπορούν να δημιουργήσουν ένα ολοκληρωμένο σενάριο χρησιμοποιώντας την αρχή και το τέλος προγραμματισμού, καθώς και να προβάλλουν το έργο τους σε πλήρη οθόνη χρησιμοποιώντας το αντίστοιχο εικονίδιο. Επίσης, πολύ σημαντικό είναι να έχουν κατανοήσει την συσχέτιση που υπάρχει μεταξύ του εικονιδίου της πράσινης σημαίας και την εντολή της πράσινης σημαίας, που δηλώνει την αρχή του προγραμματισμού. Επιπλέον σημαντικό είναι να έχουν κατανοήσει ότι οι εντολές των μηνυμάτων είναι δυο και ότι πρέπει να υπάρχουν τουλάχιστον δυο χαρακτήρες, ο ένας να στέλνει το μήνυμα και ο άλλος να το λαμβάνει. Επιπλέον, πολύ σημαντικό στην όλη διαδικασία είναι να έχουν καταλάβει ότι οι δυο χαρακτήρες επικοινωνούν μεταξύ τους και δεν ξεκινούν την ίδια στιγμή.

Έννοιες : στέλνω ένα μήνυμα, λαμβάνω ένα μήνυμα.

Γνώσεις : εντολή «στέλνω ένα μήνυμα», εντολή «λαμβάνω μήνυμα», σημασιολογία εντολής στέλνω ένα μήνυμα για τον χαρακτήρα που στέλνει το μήνυμα, σημασιολογία εντολής λαμβάνω ένα μήνυμα για τον χαρακτήρα που λαμβάνει το μήνυμα, συσχετισμός εντολής λαμβάνω ένα μήνυμα με την ύπαρξη της εντολής στέλνω ένα μήνυμα στον άλλο χαρακτήρα.

Δεξιότητες : χρήση των δυο εντολών μηνυμάτων σε δυο χαρακτήρες.

Στόχος της δραστηριότητας :

- Να μπορέσουν οι μαθητές **να χρησιμοποιήσουν λειτουργικά τις εντολές των μηνυμάτων σε δυο χαρακτήρες για να επιλύσουν ένα προγραμματιστικό πρόβλημα**.

Υλικοτεχνική υποδομή : Ταμπλέτες με εγκατεστημένη την εφαρμογή ScratchJr, προβολικό, κάρτες εντολών, μαρκαδόρος για λευκοπίνακα.

Οργάνωση τάξης : Η δραστηριότητα εκτυλίσσεται σε ομάδες. Κάθε παιδί έχει το δικό του τάμπλετ. Το τάμπλετ της ερευνήτριας συνδέεται με προβολικό, ώστε οι μαθητές να βλέπουν καλύτερα τι γίνεται στην εφαρμογή. Οι ομάδες υλοποιούν τις δραστηριότητες η μια μετά την άλλη και όχι ταυτόχρονα.

Διδακτικές προσεγγίσεις και στρατηγικές :

Θεωρητικές προσεγγίσεις : Σε αυτή τη δραστηριότητα έχουμε υιοθετήσει την εποικοδομιστική θεωρητική προσέγγιση, καθώς οι μαθητές καλούνται να επιλύσουν ένα πρόβλημα προγραμματισμού. Επιπλέον, στην παρούσα δραστηριότητα έχουμε υιοθετήσει την κοινωνικοεπικοδομιστική θεωρητική προσέγγιση, καθώς προωθείται η ανταλλαγή γνώσεων μεταξύ των μαθητών όπου είναι δυνατόν, μέσα από την συζήτηση και την αλληλεπίδραση.

Διδακτικές στρατηγικές : Ως διδακτικές στρατηγικές θα χρησιμοποιηθούν κατά κύριο λόγο η επίλυση προβλήματος. Θα χρησιμοποιηθεί επιπλέον διδακτική βοήθεια όποτε οι μαθητές το έχουν ανάγκη για να προχωρήσουν προς την κατάκτηση της γνώσης. Η ερευνήτρια θα έχει ρόλο συνερευνητικό στην όλη διαδικασία. Ακόμα θα χρησιμοποιηθεί ως διδακτική στρατηγική η συζήτηση και αλληλεπίδραση μεταξύ των παιδιών, που στόχο έχει την ανταλλαγή γνώσεων μεταξύ των μαθητών.

Σύντομη περιγραφή δραστηριότητας :

Οι μαθητές καλούνται **να επιλύσουν ένα προγραμματιστικό πρόβλημα με την βοήθεια των καρτών χρησιμοποιώντας τις εντολές των μηνυμάτων.**

Αναλυτικά τα βήματα της δραστηριότητας :

I. Αρχικά, η ερευνήτρια ζητά από τα παιδιά να ανοίξουν τα τάμπλετ και το λογισμικό και να εντοπίσουν το έργο που προβάλλεται στο προβολικό. Η ερευνήτρια παρουσιάζει ένα λανθασμένο έργο στα παιδιά, προκειμένου να εντοπίσουν το λάθος που υπάρχει και με αυτό το τρόπο να κατανοήσουμε τυχόν δυσκολίες ή και παρανοήσεις που είναι δυνατόν να υπάρχουν. Σε όλη την διαδικασία οι μαθητές μπορούν να χρησιμοποιήσουν την ψευδογλώσσα προκειμένου να διευκολυνθούν στην επίλυση του προβλήματος.

II. Στη συνέχεια, η ερευνήτρια αφηγείται στα παιδιά την ιστορία του έργου. Το παιδί κινείται προς τα δεξιά δυο βήματα και έπειτα λέει : « Το λεωφορείο ήρθε ! ». Μόλις το παιδί τελειώνει την φράση του, το λεωφορείο ξεκινά να κινείται προς τα δεξιά, μέχρι να φτάσει μπροστά από το σπίτι. Στο έργο που παρουσιάζουμε λείπει η εντολή « στέλνω ένα μήνυμα » από το σενάριο του χαρακτήρα « παιδί ».

III. Έπειτα, η ερευνήτρια ζητά από τα παιδιά να εστιάσουν την προσοχή τους στα τάμπλετ τους, να παρατηρήσουν το έργο σε πλήρη οθόνη και να εντοπίσουν αν το αποτέλεσμα που βλέπουν ανταποκρίνεται στην ιστορία που τους αφηγήθηκε προηγουμένως και αν όχι, γιατί.

IV. Τα παιδιά καλούνται να εντοπίσουν το λάθος στο πρόγραμμα, να το διορθώσουν και να δικαιολογήσουν τις κινήσεις τους, ενώ παράλληλα η ερευνήτρια συμμετέχει στην όλη διαδικασία συν-ερευνητικά με ερωτήσεις, όπου χρειάζεται. Επιπλέον, αν χρειαστεί η ερευνήτρια θα επαναλάβει την ιστορία του έργου ώστε τα παιδιά να συγκρίνουν τι γίνεται στην πραγματικότητα με αυτό θα έπρεπε να γινόταν αν το πρόγραμμα ήταν σωστό.

V. Τέλος, οι μαθητές καλούνται να αποθηκεύσουν το έργο τους.

10. ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ ΑΞΙΟΛΟΓΗΣΗΣ

1η Δραστηριότητα Αξιολόγησης :

I. Αρχικά, η ερευνήτρια αφηγείται στα παιδιά την ιστορία του έργου που θα κληθούν να δημιουργήσουν στην συνέχεια, ενώ παράλληλα προβάλει το έργο σε πλήρη οθόνη.

II. Το πρόβλημα έχει ως εξής : « Μια φορά ήταν στο δάσος ένα παιδάκι και μια πεταλούδα. Η πεταλούδα καθόταν πάνω στο δέντρο αριστερά και το παιδάκι δίπλα στον κορμό ενός δέντρου. Το παιδάκι ξεκινάει πρώτο να κινείται μέχρι να φτάσει στη μέση των λουλουδιών. Μόλις το παιδάκι φτάσει στη μέση των λουλουδιών η πεταλούδα ξεκινάει να κινείται μέχρι να φτάσει πάνω στο χορτάρι ».

III. Καθ'όλη την διάρκεια επίλυσης του προβλήματος, τα παιδιά **θα έχουν στην διάθεση τους τις κάρτες εντολών**, σε περίπτωση που θέλουν να τις χρησιμοποιήσουν για να βοηθηθούν με την επίλυση του προβλήματος.

IV. Τα παιδιά δεν θα έχουν πρόσβαση στο χώρο προγραμματισμού των δυο χαρακτήρων, καθώς πρέπει να αποκωδικοποιήσουν την κίνηση των χαρακτήρων αφενός από την εκφώνηση του προβλήματος και αφετέρου από την παρατήρηση του έτοιμου έργου σε πλήρη οθόνη και να σχηματίσουν εκείνοι τα προγράμματά τους.

2η Δραστηριότητα Αξιολόγησης

I. Αρχικά, η ερευνήτρια αφηγείται στα παιδιά την ιστορία του έργου που θα κληθούν να δημιουργήσουν στην συνέχεια, ενώ παράλληλα προβάλει το έργο σε πλήρη οθόνη.

II. Το πρόβλημα έχει ως εξής : « Μια φορά ήταν ένα φθινοπωρινό πρωινό σε ένα δάσος ένα παιδάκι και ένα κουνέλι. Το παιδάκι βρίσκεται μπροστά από ένα δέντρο στα αριστερά ενώ το κουνέλι μπροστά από ένα άλλο δέντρο στα δεξιά. Οι δυο χαρακτήρες ξεκινούν ταυτόχρονα. Το παιδάκι κινείται προς τον κουνέλι δυο βήματα και χοροπηδάει μια φορά, ενώ το κουνέλι κινείται προς το παιδάκι και κάθε φορά που κάνει ένα βήμα λέει : « Γειά σου παιδάκι !» και αυτό το κάνει τρείς φορές. »

III. Καθ'όλη την διάρκεια επίλυσης του προβλήματος, τα παιδιά θα έχουν στην διάθεση τους τις κάρτες εντολών, σε περίπτωση που θέλουν να τις χρησιμοποιήσουν για να βοηθηθούν με την επίλυση του προβλήματος.

IV. Τα παιδιά δεν θα έχουν πρόσβαση στο χώρο προγραμματισμού των δυο χαρακτήρων, καθώς πρέπει να αποκωδικοποιήσουν την κίνηση των χαρακτήρων αφενός από την εκφώνηση του προβλήματος και αφετέρου από την παρατήρηση του έτοιμου έργου σε πλήρη οθόνη και να σχηματίσουν εκείνοι τα προγράμματά τους.

Annexe 5 : Les programmes d'élèves du GS2 en fin d'intervention, PB1

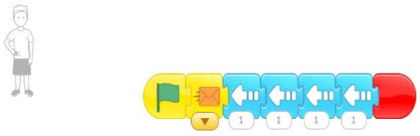
Programmes corrects PB1

Dans la suite, le code est celui de l'élève : GS représente grande section suivi du numéro de l'élève. PB1 réfère au problème 1.

GS2E1, PB1

GS2E2, PB1

GS2E3, PB1



GS2E4, PB1

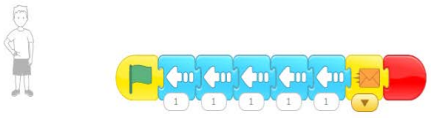

SCRATCHJR
Child
Butterfly
Π4Δ8N2


SCRATCHJR
Child
Butterfly
Π4Δ8N2

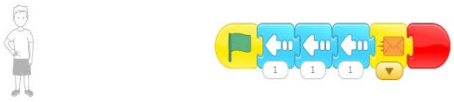
GS2E5, PB1

GS2E6, PB1

GS2E7, PB1

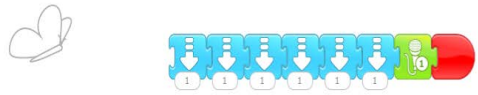


GS2E8, PB1



GS2E9, PB1

GS2E10, PB1



GS2E11, PB1

GS2E12, PB1

Annexe 6 : Les programmes d'élèves du GS2 en fin d'intervention, PB2

Programmes corrects PB2

Dans la suite, le code est celui de l'élève : GS représente grande section suivi du numéro de l'élève. PB2 réfère au problème 2.

GS1E1, PB2

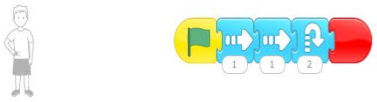
GS2E2, PB2


SCRATCHJR
Child
Rabbit
Π2Δ8N2

SCRATCHJR
Child
Rabbit
Π2Δ8N2

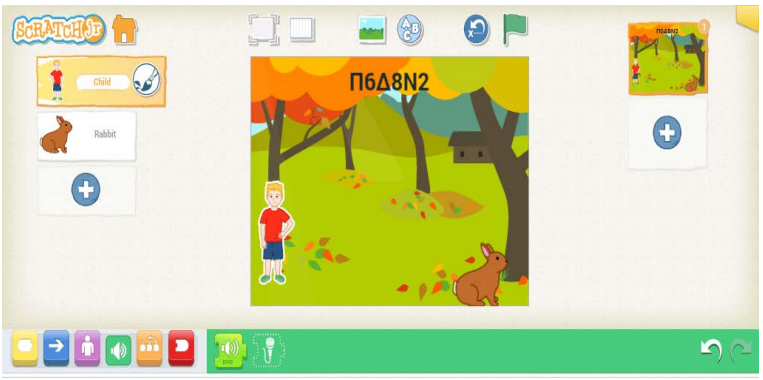
GS2E3, PB2

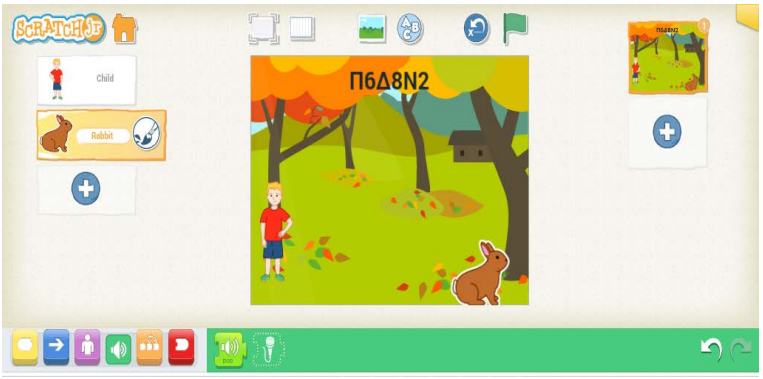
GS2E4, PB2

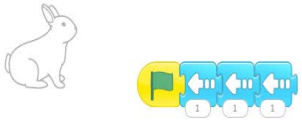


GS2E5, PB2

GS2E6, PB2







GS2E9, PB2

GS2E10, PB2

GS2E11, PB2


SCRATCHJR
Child
Rabbit
Π11Δ8N2


SCRATCHJR
Child
Rabbit
Π11Δ8N2

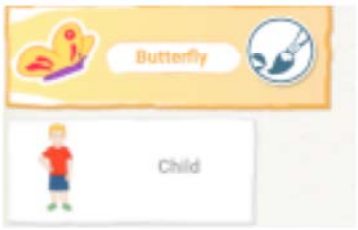
GS2E12, PB2

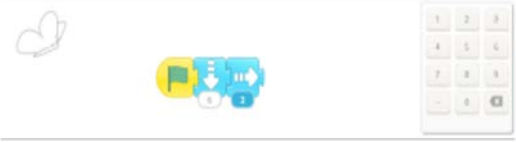
Annexe 7 : Transcription d'actions de l'Élève 1, CP1

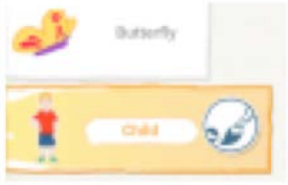
PB1

Déroulé des actions

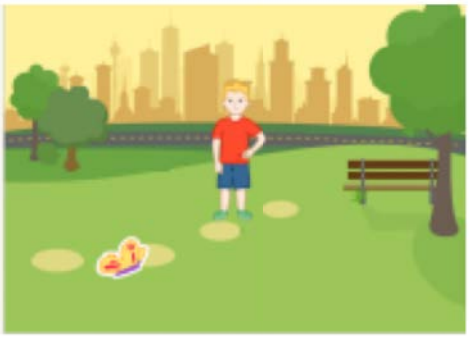
Captures d'écran	Transcription d'actions	Commentaires
	01:36 L'élève 1 efface le personnage « chat ».	Il s'agit d'une contrainte du logiciel ScratchJr, car chaque nouveau projet commence en ayant le personnage « chat » comme personnage par défaut.
	01 :39 L'élève 1 insère correctement l'arrière-plan.	
	01 :46 L'élève 1 insère correctement le personnage « papillon ».	
	01:48 L'élève 1 met le personnage « papillon » en position initiale dans la scène avec son doigt.	L'élève 1 initialise correctement la position de son personnage « papillon ».
	01:52 L'élève 1 insère correctement le personnage « enfant ».	
	02:00 La chercheuse rappelle à l'élève la consigne du problème.	Aide : Consigne

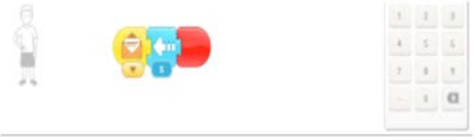
	02 :01 L'élève 1 met le personnage « enfant » en position initiale dans la scène avec son doigt.	L'élève 1 initialise correctement la position de son personnage « enfant ».
	02 :04 L'élève 1 commence par la programmation du personnage « papillon ».	
	02:05 L'élève 1 va dans la catégorie des commandes de « déplacement ».	
	02 :07 L'élève 1 dépose la commande « déplacement d'un pas en bas » dans l'espace de programmation.	
	02:10 L'élève 1 efface ensuite la commande de « déplacement ».	
	02:11 L'élève 1 va dans la catégorie des commandes de « démarrage ».	
	02 :13 L'élève 1 dépose la commande de « quand drapeau-vert cliqué » dans l'espace de programmation du personnage « papillon ».	
	02:14 L'élève 1 va dans la	

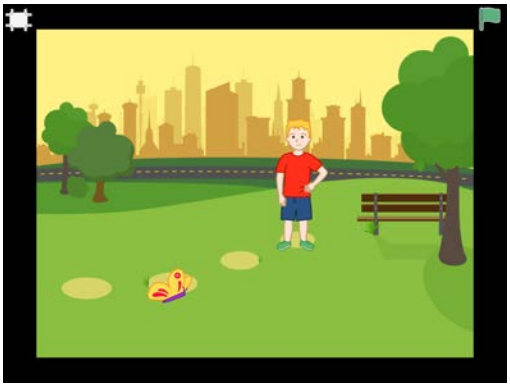
	catégorie des commandes de « déplacement ».	
	02 :16 L'élève 1 dépose la commande de « déplacement d'un pas en bas » une seule fois dans l'espace de programmation.	Paramètre pour le déplacement de son personnage.
	02:21 Elle choisit le chiffre 6 comme paramètre de la commande de « déplacement d'un pas en bas ».	Estimation du paramètre
	02:24 L'élève 1 va dans la catégorie des commandes de « déplacement ».	
	02 :25 L'élève 1 dépose la commande de « déplacement d'un pas à droite » une seule fois dans l'espace de programmation.	
	02:30 Elle utilise le chiffre 2 comme paramètre de la commande de « déplacement ».	Estimation du paramètre
	02:32 L'élève 1 va dans la catégorie des commandes de « fin ».	
	02 :35 L'élève 1 insère la commande de « fin » à la fin du programme du personnage « papillon ».	

	02 :40 L'élève 1 passe au personnage « enfant ».	
	02 :44 L'élève 1 revient au programme du personnage « papillon ».	Ici l'élève 1 revient au programme du personnage « papillon » parce qu'elle se souvient qu'elle a oublié la commande « envoyer un message ».
	02 :46 L'élève 1 ouvre le programme du personnage « papillon ».	
	02:47 L'élève 1 va dans la catégorie de commandes de « démarrage ».	
	02:49 Elle ajoute la commande « envoyer un message ».	Elle utilise la commande « envoyer un message » au bon endroit sur le programme du personnage « papillon ».

	02:58 L'élève 1 passe au personnage « enfant ».	
	02:57 L'élève 1 va dans la catégorie de commandes de « démarrage ».	Elle passe directement à la programmation du personnage « enfant » sans exécuter le programme du personnage « papillon » avant.
	03 :05 L'élève 1 utilise la commande « quand message reçu » au début du programme du personnage « enfant ».	
	03:07 L'élève 1 passe à la catégorie de commandes de « déplacement ».	
	03 :10 L'élève 1 choisit la commande de « déplacement d'un pas à gauche ».	
	03 :16 L'élève 1 choisit une deuxième commande de « déplacement d'un pas à gauche ».	

	03:23 L'élève 1 efface l'une des deux commandes de « déplacement d'un pas à gauche ».	
	03:33 L'élève 1 choisi le chiffre 7 comme le paramètre de la commande de « déplacement d'un pas à gauche ».	Estimation du paramètre.
	03:35 L'élève 1 passe à la catégorie des commandes de « fin ».	
	03:38 L'élève 1 ajoute la commande de « fin » à la fin du programme du personnage « enfant ».	
	03:40 L'élève 1 passe au personnage « papillon ».	
	03 :43 L'élève 1 lance le projet en appuyant sur le programme du personnage « papillon ».	
	03:45	Le personnage « papillon » ne se déplace pas assez vers la droite. Le personnage « enfant » dépasse l'endroit demandé parce qu'il se déplace trop vers la gauche.

	03 :46 L'élève 1 change le paramètre de la commande « déplacement d'un pas à droite » et elle choisit le chiffre 3 au lieu de 2.	L'élève 1 identifie l'erreur sur le programme du personnage « papillon » et elle la corrige. Cette correction correspond au résultat obtenu après l'exécution du programme.
	03:55 Puis, l'élève 1 va dans l'espace de programmation du personnage « enfant ».	
	L'élève 1 change le paramètre de la commande « déplacement d'un pas à gauche » en mettant le chiffre 5 au lieu de 7.	L'élève 1 identifie l'erreur sur le programme du personnage « enfant » et elle la corrige. Cette correction correspond au résultat obtenu par l'exécution du programme.
	04:03 L'élève 1 initialise la position des personnages en utilisant l'icône d'initialisation.	
	04 :05 L'élève 1 revient au programme du personnage « papillon »	
	04:06 L'élève 1 lance le	Les programmes de l'élève 1 sont

	programme du personnage « papillon » et les personnages arrivent au bon endroit dans la scène.	corrects.
	04:10 La chercheuse lui dit d'aller en plein-écran pour mieux voir son projet.	
	04 :15 L'élève 1 va en plein écran.	
	04:18 L'élève 1 lance le projet en plein-écran pour valider si elle a bien programmé. Les personnages arrivent au bon endroit.	

PB2

Déroulé des actions

Captures d'écran	Transcription d'actions	Commentaires
	05:58 L'élève 1 efface le personnage « chat » sans problème.	
	06:03 L'élève 1 insère l'arrière-plan sans problème.	
	06 :07 L'élève 1 insère le	

	personnage « papillon ».	
	06:14 La chercheuse lui rappelle la consigne du problème.	Aide : Consigne
	06:15 L'élève 1 met le personnage « papillon » à sa position initiale avec le doigt.	Le personnage « papillon » n'est pas exactement au bon endroit.
	06:26 L'élève 1 insère le personnage « lapin ».	
	06:31 L'élève 1 met le personnage « lapin » à sa position initiale avec le doigt.	Le personnage « lapin » n'est pas au bon endroit dans la scène.
	06:34 La chercheuse lui rappelle la consigne du problème.	Aide : Consigne
	06:36 L'élève 1 remet le personnage « lapin » à sa position initiale avec le doigt.	L'élève 1 initialise correctement la position du personnage « lapin ».
	06:44 La chercheuse intervient pour l'aider à initialiser la position du personnage « papillon » car elle ne le mettait pas exactement au bon endroit.	Aide : Initialisation
	06:46 L'élève 1 commence la programmation par le personnage « papillon ».	
	06:48 L'élève 1 va dans la	

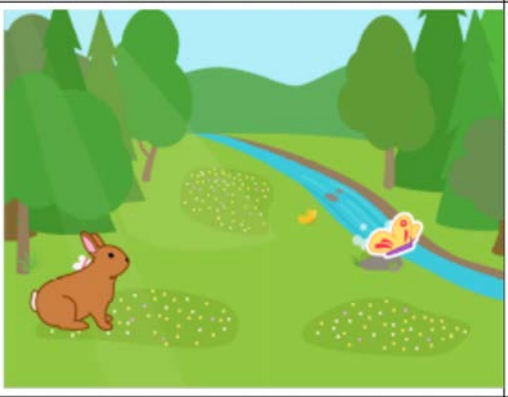
	catégorie de commandes de « démarrage ».	
	06:49 L'élève 1 utilise la commande de « quand drapeau-vert cliqué».	
	06:55 L'élève 1 va dans la catégorie de commandes de « déplacement ».	
	06:58 L'élève 1 dépose la commande de « déplacement d'un pas à droite » dans l'espace de programmation.	
	07:03 L'élève 1 choisit le paramètre 6.	Estimation du paramètre
	07:19 L'élève 1 dépose la commande de « déplacement d'un pas en bas » dans l'espace de programmation.	
	07 :22 L'élève 1 utilise encore une commande de « déplacement d'un pas à droite ».	
	07:26 L'élève 1 choisit le paramètre 3.	Estimation du paramètre
	07:28 L'élève 1 va dans la catégorie de commandes de « fin ».	

	07 :29 L'élève 1 ajoute la commande de « fin » à la fin du programme du personnage « papillon ».	
	07 :29 L'élève 1 passe directement au programme du personnage « lapin ».	L'élève 1 passe directement à la programmation du personnage « lapin » sans exécuter le programme du personnage « papillon » avant.
	07:32 La chercheuse rappelle la consigne du problème.	Aide : Consigne
	07;34 L'élève 1 va dans la catégorie des commandes de « démarrage ».	
	07 :38 L'élève 1 dépose la commande de « quand drapeau vert cliqué » dans l'espace de programmation du personnage « lapin ».	Il est important de noter ici que l'élève 1 programme chaque fois les actions de ses personnages en respectant l'ordre d'exécution de ces actions au projet modèle. Par exemple elle commence chaque fois par la commande « quand drapeau-vert cliqué », ensuite elle passe à la commande de « répétition » etc. Elle programme de la même manière tous ses personnages.

	07 :44 L'élève 1 passe ensuite à la catégorie des commandes de « contrôle ».	
	07:46 L'élève 1 dépose la commande de « répétition » dans l'espace de programmation.	
	07 :49 L'élève 1 ajoute à l'intérieur de la commande de « répétition » la commande de « déplacement d'un pas à droite ».	
	07 :50 L'élève 1 passe à la catégorie des commandes de « son ».	
	07:51 La chercheuse lui rappelle la consigne du problème.	Aide : Consigne
	07:56 L'élève 1 enregistre le son « Bonjour le papillon » sans problème.	
	07:59 L'élève 1 ajoute ensuite la commande du « son » à l'intérieur de la « répétition », après la commande de « déplacement d'un pas à droite ».	Le motif des commandes qui est répété est correct.

	08:02 La chercheuse lui rappelle la consigne du problème.	Aide : Consigne
	08 :02 L'élève 1 change le paramètre de « répétition » en choisissant le chiffre 3.	
	08:11 L'élève 1 passe à la catégorie de commandes de « fin ».	
	08 :14 L'élève 1 ajoute la « fin » à la fin du programme du personnage « lapin ».	
	08 :16 L'élève 1 passe au personnage « papillon ».	
	08:17 L'élève 1 lance le programme du « papillon », pour valider si elle avait bien programmé le personnage « papillon ».	
		08 :21 Le personnage « papillon » dépasse l'endroit demandé dans la scène, parce qu'il se déplace trop vers la droite.

	08:25 L'élève 1 corrige son programme en choisissant le chiffre 4 comme paramètre de la première commande « déplacement d'un pas à droite », au lieu de 6.	L'élève 1 identifie l'erreur sur le programme du personnage « papillon » et elle la corrige. Cette correction correspond au résultat obtenu par l'exécution du programme, mais elle n'est pas suffisante pour la correction de l'erreur.
	08 :29 L'élève 1 initialise la position du personnage « papillon » en utilisant l'icône.	
	08 :31 L'élève 1 lance de nouveau son programme.	
		08 :33 Le personnage « papillon » n'arrive toujours pas au bon endroit dans la scène, car il se déplace encore trop vers la droite.
	08:36 L'élève 1 change le paramètre de la deuxième commande « déplacement d'un pas à droite » en mettant le chiffre 2 au lieu de 3.	L'élève 1 identifie l'erreur encore présente sur le programme du personnage « papillon » et elle la corrige. Cette correction correspond au résultat obtenu par l'exécution du programme, mais elle n'est pas suffisante

		pour la correction de l'erreur.
	08 :42 L'élève 1 remet le personnage « papillon » au début en utilisant l'icône.	
	08 :46 L'élève 1exécute le programme du personnage « papillon » de nouveau pour valider.	
	08 :51 La chercheuse lance le projet modèle.	Aide : Lancement du projet modèle
	08:59 L'élève 1 change de nouveau le paramètre de la deuxième commande « déplacement d'un pas à droite » en mettant le chiffre 1 au lieu de 2.	L'élève 1 identifie l'erreur présente encore sur le programme du personnage « papillon » et elle la corrige. Cette correction correspond au résultat obtenu après l'exécution du programme de « papillon » et elle est suffisante pour la correction de l'erreur.
	09 :03 L'élève 1 remet le personnage « papillon » en position initiale en utilisant	

	l'icône.	
	09 :04 L'élève 1 exécute à nouveau le programme du personnage « papillon ».	
	09 :07 L'élève 1 observe que le personnage « papillon » arrive au bon endroit dans la scène cette fois-ci.	
	09:11 L'élève sort sans le faire exprès de son projet au lieu de sortir du mode plein-écran.	
	09:18 La chercheuse l'aide à revenir à son projet.	
	09:19 L'élève exécute le programme du personnage « lapin ».	
	09:29 La chercheuse lance le projet modèle pour que l'élève vérifie qu'elle a bien programmé.	Aide : Lancement projet modèle.
	09:37 La chercheuse lui dit d'aller en plein-écran.	

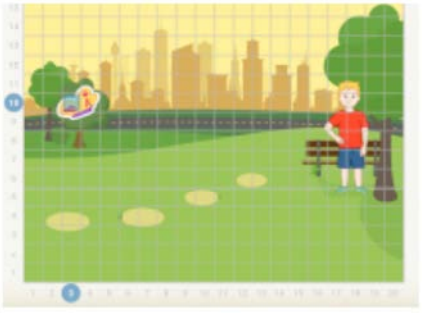
	09 :39 L’élève 1 passe en plein-écran.	
	09:44 L’élève exécute son projet en plein-écran.	Le projet de l’élève 1 est correct.
	09:53 L’élève 1 quitte le plein-écran.	

Annexe 8 : Transcription d'actions de l'Élève 2, CP1

PB1

Déroulé des actions

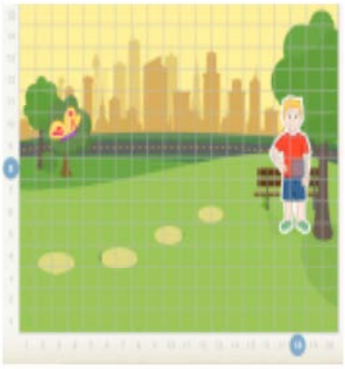
Captures d'écran	Transcription d'actions	Commentaires
	02:34 L'élève 2 efface le personnage « chat ».	Contrainte du logiciel ScratchJr.
	02:41 L'élève 2 insère le personnage « enfant ».	
	02:48 L'élève 2 insère le personnage « papillon ».	
	02:56 L'élève 2 insère correctement l'arrière-plan.	
	03:01 L'élève 2 met le personnage « enfant » en position initiale.	
	03:04 La chercheuse intervient pour aider l'élève à choisir la correcte position initiale pour son personnage « papillon ».	Initialisation
	03:05 L'élève 2 met le personnage « papillon » en position	

	initiale.	
	03:12 L'élève 2 appuie sur l'icône de la « grille » pour faire apparaître le quadrillage.	
	03:14 L'élève 2 va dans la catégorie des commandes de « démarrage ».	
	03:15 L'élève 2 dépose la commande « quand personnage cliqué » .	L'élève 2 fait descendre toutes les commandes dont elle a besoin par la catégorie des commandes de « démarrage ».
	03:16 L'élève 2 dépose la commande « envoyer un message » dans l'espace de programmation.	
	03:23 L'élève 2 va dans la catégorie des commandes de « déplacement ».	
	03:24 L'élève 2 dépose la commande « déplacement d'un pas à droite » dans l'espace de programmation.	Stratégie « grille et paramètres » pour le déplacement du personnage. Cette stratégie a été inventée par l'élève 2.
	03:27 Puis, elle appuie sur la commande « déplacement d'un pas à droite » une fois en regardant la « grille ».	
	L'élève 2 appuie encore une fois sur la commande « déplacement d'un pas à	

	droite ». L'élève 2 appuie encore une fois sur la commande « déplacement d'un pas à droite ». 03:30 Ensuite, elle choisit le chiffre 3 comme paramètre de la commande de « déplacement d'un pas à droite ».	
	03:33 L'élève 2 initialise la position du personnage « papillon » en utilisant l'icône d'initialisation.	
	03 :34 L'élève 2 exécute la commande de déplacement à droite du « papillon » pour vérifier s'il arrive en haut de l'endroit souhaité dans la scène.	L'élève 2 lance que la commande de « déplacement » du personnage, pour vérifier si la programmation réalisée pour le déplacement à droite est correcte.
		Les actions après l'exécution de la commande « déplacement d'un pas à droite » ne sont pas considérées en tant que corrections, parce que l'élève 2 est une exception en raison de la stratégie particulière qu'elle utilise pour l'écriture et l'exécution de ses programmes.

	03:43 L'élève 2 dépose la commande « déplacement d'un pas en bas» dans l'espace de programmation.	Stratégie « grille et paramètres » pour le déplacement du personnage. Cette stratégie a été inventée par l'élève.
	03:45 L'élève 2 appuie sur la commande de « déplacement d'un pas en bas » et en utilisant la « grille » elle compte les pas que le personnage « enfant » doit faire, pour arriver à l'endroit souhaité dans la scène.	
	Puis, elle appuie sur la commande « déplacement d'un pas en bas » une fois en utilisant la « grille ».	
	Puis, elle appuie sur la commande « déplacement d'un pas en bas » une fois en utilisant la « grille ».	
	Puis, elle appuie sur la commande « déplacement d'un pas en bas » une fois en utilisant la « grille ».	
	Puis, elle appuie sur la commande « déplacement d'un pas en bas » une fois en utilisant la « grille ».	
	Puis, elle appuie sur la commande « déplacement d'un pas en bas » une fois en utilisant la « grille ».	
	03:52 Ensuite, elle utilise le chiffre 6 comme paramètre de la commande de « déplacement	

	d'un pas en bas ».	
	04:02 L'élève 2 rassemble toutes les commandes .	
	04:05 Elle dit : « Ce n'est pas ça ! ».	L'élève 2 repère qu'il y a une erreur sur son programme et elle procède tout de suite à sa correction. L'élève 2 arrive à repérer cette erreur sans avoir exécuté son programme. L'élève 2 est une des seules à s'exprimer à voix haute qu'il y a une erreur sur son programme. Aide : Effacement d'une commande
	04:05 L'élève 2 essaie d'effacer la commande de démarrage mais efface tout le programme qu'elle avait écrit jusqu'ici par erreur.	
	04:10 La chercheuse est intervenue pour appuyer sur le bouton « annuler » de l'interface de programmation afin de revenir en arrière.	
	04:14 La chercheuse efface la commande « quand personnage cliqué ».	
	04:18 L'élève 2 remplace la commande « quand personnage cliqué » par la commande « quand drapeau-vert cliqué ».	
	04:20 L'élève 2 ajoute à la fin la commande de « fin ».	
	04:25 L'élève 2 enlève la « grille ».	

	04:27 L'élève 2 remet la « grille ».	
	04:28 L'élève 2 passe au personnage « enfant ».	
	04:31 L'élève 2 déplace le personnage « papillon » vers le centre du cercle jaune avec le doigt.	
	04:38 L'élève 2 va dans la catégorie des commandes de « démarrage ».	
 	04:39 L'élève 2 choisis la commande « quand message reçu » et elle la dépose dans l'espace de programmation.	
	L'élève 2 va dans la catégorie des commandes de « déplacement ».	
  	04:43 L'élève 2 utilise la commande de « déplacement d'un pas à gauche » une seule fois dans l'espace de la programmation.	Stratégie « grille et paramètres » pour le déplacement du personnage. Cette stratégie a été inventée par l'élève.

	04:45 L’élève 2 appuie sur la commande de « déplacement d’un pas à gauche » et en utilisant la « grille », elle compte en même temps les pas que le personnage « enfant » doit faire, pour arriver à l’endroit souhaité dans la scène.	
	L’élève 2 appuie encore une fois sur la commande de « déplacement d’un pas à gauche » et en utilisant la « grille » elle compte les pas que le personnage « enfant » doit faire, pour arriver à l’endroit souhaité dans la scène.	
	L’élève 2 appuie encore une fois sur la commande de « déplacement d’un pas à gauche » et en utilisant la « grille » elle compte les pas que le personnage « enfant » doit faire, pour arriver à l’endroit souhaité dans la scène.	
	L’élève 2 appuie encore une fois sur la commande de « déplacement d’un pas à gauche » et en utilisant la « grille » elle compte les pas que le personnage « enfant » doit faire, pour arriver à l’endroit souhaité dans la scène.	
	L’élève 2 appuie encore une fois sur la commande de « déplacement d’un pas à gauche » et en utilisant la « grille » elle compte les pas	

	que le personnage « enfant » doit faire, pour arriver à l'endroit souhaité dans la scène.	
	04:50 Puis, elle utilise le chiffre 5 comme le paramètre de la commande de « déplacement d'un pas à gauche ».	
	04:55 L'élève 2 ajoute la commande « déplacement d'un pas à gauche » au programme du personnage « enfant » .	
	04:57 L'élève 2 place le personnage « enfant » en position initiale avec l'icône d'initialisation.	
	04:58 L'élève 2 va à la catégorie de commandes de « fin ».	
	05:00 L'élève 2 ajoute la commande de « fin » à la fin du programme du personnage « enfant ».	
	05:05 L'élève 2 enlève la « grille ».	
	05:09 L'élève 2 met le personnage « papillon » en position initiale avec le doigt.	Elle a eu besoin d'utiliser son doigt parce qu'avant elle avait cliqué à nouveau sur le « papillon » avec son doigt, en lui

		modifiant sa position initiale.
	05:12 L'élève 2 passe en plein-écran.	
	05:15 L'élève 2 exécute son projet pour vérifier son exactitude.	
	Le projet est correct.	

PB2

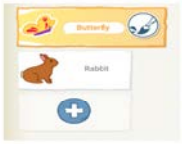
Déroulé des actions

Captures d'écran	Transcription d'actions	Commentaires
	07:25 L'élève 2 efface le personnage « chat » sans problème.	Contrainte de ScratchJr. Le personnage « chat » est le personnage par défaut de tout les nouveaux projets sur ScratchJr.
	07:30 L'élève 2 insère le personnage « papillon » sans problème.	
	07:37 L'élève 2 insère le personnage « lapin » sans problème.	
	07:46 L'élève 2 insère l'arrière-plan sans problème.	
	07:51 L'élève 2 met le personnage « lapin » en position initiale.	La position initiale du personnage « lapin » n'est pas correcte.
	07:53 L'élève 2 met le personnage « papillon » en position initiale.	La position initiale du personnage « lapin » n'est pas correcte.

	07:58 La chercheuse aide l'élève à initialiser la position de ses personnages.	Initialisation
	08:00 L'élève 2 met le personnage « lapin » en position initiale avec son doigt.	La position initiale est correcte.
	08:02 L'élève met le personnage « papillon » en position initiale avec son doigt.	La position initiale est correcte.
	08:10 L'élève 2 va dans la catégorie de commandes de « son ».	
	08:12 L'élève 2 crée l'enregistrement correctement.	
 	08:13 L'élève 2 dépose la commande de « son » dans l'espace de programmation du personnage « lapin ».	
	08:14 L'élève 2 dit : « Je vais essayer d'abord » et exécute la commande de l'enregistrement, pour valider qu'elle a bien enregistré le son.	L'élève 2 ne lance que la commande du « son », pour vérifier si elle a bien enregistré le son pour le personnage « lapin ».
	08:20 L'élève 2 va dans la catégorie de catégories de « démarrage ».	
 	08:23 L'élève 2 insère la commande de « quand drapeau-vert cliqué » au	

	début du programme du personnage « lapin ». (avant la commande d'enregistrement)	
	08 : 29 La chercheuse intervient pour lui rappeler la consigne du problème. L'élève dit « d'accord ».	Aide : Consigne.
	08:32 Ensuite, l'élève 2 sépare les deux commandes (drapeau-vert et enregistrement).	
	08:35 L'élève 2 va dans la catégorie de commandes de « contrôle ».	
	08:37 L'élève 2 insère la commande de « répétition » sur la commande d'enregistrement.	
	08:40 L'élève 2 change le paramètre de la commande de « répétition » et elle choisit le chiffre 3 au lieu de 4.	
	08:47 Elle rassemble la commande de « répétition » avec la commande de «quand drapeau-vert cliqué ».	
	08: 55 L'élève dit « après... L'élève 2 compte avec ses doigts les pas que le personnage « lapin » devrait faire pour arriver à l'endroit demandé dans la scène.	
	08:58	Aide : Consigne

	La chercheuse intervient pour lui rappeler la consigne du problème. L'élève répond « d'accord ».	
	09:00 L'élève 2 va dans la catégorie de commandes de « déplacement ».	
	09:04 L'élève 2 utilise la commande de « déplacement d'un pas à droite » et l'ajoute dans la commande de « répétition » après la commande du « son ».	
	L'élève 2 va dans la catégorie de commandes de « fin ».	
	09:10 La chercheuse intervient pour lui rappeler la consigne du problème. L'élève répond « D'accord ».	Aide : Consigne
	09:12 L'élève 2 ajoute la commande de « fin » à la fin du programme du personnage « lapin ».	
	09:16 L'élève 2 enlève les deux commandes de la commande de « répétition ».	
	09:20 L'élève 2 pose la question suivante : « Il dit d'abord bonjour le papillon ? »	
	09:22 La chercheuse lui répond que le personnage « lapin » marche et puis il dit « Bonjour » et elle continue à lui dire la consigne du	Aide : Consigne

	problème pour le personnage « papillon ». L'élève 2 répond « D'accord ».	
	09:23 L'élève 2 choisit la commande de « déplacement d'un pas à droite » et elle l'ajoute dans la commande de « répétition ».	
	09:26 L'élève 2 ajoute ensuite la commande de « son ».	
	09:28 L'élève 2 passe au personnage « papillon ».	
	09:32 L'élève 2 va dans la catégorie de commandes de « démarrage».	
	09:35 L'élève 2 utilise la commande de « quand drapeau-vert cliqué» et elle la met au début du programme du personnage « papillon ».	
	09:37 L'élève 2 va dans la catégorie de commandes de « fin ».	
	09:38 L'élève 2 utilise ensuite la commande de « fin » et elle la met à la fin du programme du	

	personnage « papillon ».	
	09:42 L'élève 2 ensuite utilise la « grille ».	
	09:43 L'élève change avec son doigt la position initiale du personnage « papillon ».	La position initiale du personnage « papillon » n'est pas correcte.
	09:46 L'élève 2 va dans la catégorie de commandes de « déplacement ».	
	09:47 La chercheuse aide l'élève à choisir la correcte position initiale pour son personnage « papillon ».	Aide : Initialisation
	09:48 L'élève 2 choisit la commande de « déplacement d'un pas à droite » et elle la dépose dans la scène.	Stratégie de « grille et paramètres » pour le déplacement du personnage. Cette stratégie a été inventée par l'élève.
	Le logiciel se met à boguer à ce moment et nous empêche de déplacer la commande de « déplacement ». L'élève est obligée de sélectionner une autre commande de « déplacement d'un pas à droite ».	

	10:27 L’élève 2 appuie sur la commande de « déplacement d’un pas à droite » et en utilisant la « grille » elle compte les pas que le personnage « papillon » doit faire pour arriver au-dessus de l’endroit souhaité dans la scène.	Stratégie « grille et paramètres » pour le déplacement du personnage. Cette stratégie a été inventée par l’élève.
	10:30 L’élève 2 appuie sur la commande de « déplacement d’un pas à droite » et en utilisant la « grille » elle compte les pas que le personnage « papillon » doit faire, pour arriver à l’endroit souhaité dans la scène.	
	L’élève 2 appuie sur la commande de « déplacement d’un pas à droite » et en utilisant la « grille » elle compte les pas que le personnage « papillon » doit faire, pour arriver à l’endroit souhaité dans la scène.	
	L’élève 2 appuie sur la commande de « déplacement d’un pas à droite » et en utilisant la « grille » elle compte les pas que le personnage « papillon » doit faire, pour arriver à l’endroit souhaité dans la scène.	
	L’élève 2 appuie sur la commande de « déplacement d’un pas à droite » et en utilisant la « grille » elle compte les pas que le personnage « papillon » doit faire, pour arriver à l’endroit souhaité dans la scène.	
	10:35 Puis, elle choisit le chiffre 5 comme paramètre de la	

	commande « déplacement d'un pas à droite »	
	10:39 L'élève 2 insère la commande « déplacement d'un pas en bas » dans l'espace de programmation.	Stratégie « grille et paramètres » pour le déplacement du personnage. Cette stratégie a été inventée par l'élève.
	10 :41 L'élève 2 appuie sur la commande de « déplacement d'un pas en bas » et elle compte le nombre des pas que le personnage « papillon » doit faire pour arriver à l'endroit souhaité à l'aide de la « grille ».	
	L'élève 2 appuie à nouveau sur la commande « déplacement d'un pas en bas » en regardant la « grille ».	
	10:43 Puis, elle choisit le chiffre 2 comme paramètre de la commande de « déplacement d'un pas en bas ».	
	10:46 L'élève 2 rassemble les deux commandes de « déplacement ».	
	10:50 La chercheuse intervient pour lui rappeler la consigne du problème et lance le projet modèle en même temps.	Aides : Consigne. Lancement du projet modèle.

	10:51 L'élève 2 met le personnage « papillon » en position initiale avec son doigt.	
	10 :54 L'élève 2 exécute les commandes de déplacement du personnage « papillon » pour vérifier s'il arrive au bon endroit dans la scène.	
		Le personnage « papillon » se déplace trop vers le bas.
	10 :56 L'élève 2 dit : « Ah! ».	Identification de l'erreur : L'élève repère que le personnage « papillon » n'arrive pas au bon endroit dans la scène. L'élève 2 procède tout de suite à la correction de cette erreur.
	10:57 L'élève 2 revient à la commande de « déplacement d'un pas à droite » et elle corrige son paramètre en choisissant le chiffre 6 à la place du chiffre 5.	Cette correction ne correspond pas tout à fait au résultat obtenu après l'exécution des commandes de déplacement, mais il est possible que l'élève a procédé à une telle correction, parce que le personnage

		« papillon » cachait l'endroit demandé et l'élève ne pouvait donc pas bien voir si ce dernier arrivait bien sur l'endroit demandé.
	11:03 L'élève 2 initialise la position du personnage « papillon » avec l'icône d'initialisation.	
	11:04 L'élève 2 n'exécute que les commandes de « déplacement » pour valider si elle a bien programmé.	
	11:06 La chercheuse lui montre la position initiale correcte.	Aide :Initialisation
	11:10 L'élève 2 exécute de nouveau les commandes de « déplacement » pour vérifier si elle a bien programmé le personnage « papillon ».	
		Le personnage « papillon » se déplace trop vers le bas.
		Le personnage « papillon » se déplace trop vers la droite.
	11:13 L'élève 2 dit « Non ! » .	Identification de l'erreur : L'élève 2 repère que le personnage

		« papillon » n'arrive pas à l'endroit demandé.
	11:17 L'élève 2 corrige son programme en choisissant le chiffre 1 à la place du chiffre 2 pour la commande de « déplacement d'un pas en bas ».	Cette correction correspond au résultat obtenu après l'exécution des commandes de « déplacement ».
	11:20 L'élève 2 met le personnage « papillon » au début avec l'icône d'initialisation.	
	11:28 L'élève 2 rassemble toutes les commandes ensemble.	
	11:38 L'élève 2 enlève la « grille ».	
	11:40 L'élève 2 exécute son projet avec le bouton du drapeau-vert en haut de la scène.	
		Le personnage « papillon » se déplace trop vers la droite.
	11:59 L'élève 2 regarde l'endroit où le personnage « papillon » arrive.	
	La chercheuse lui rappelle l'endroit demandé dans la scène pour le personnage « papillon ».	Aide :Consigne

	12:09 L'élève 2 revient à la commande de « déplacement d'un pas à droite » et elle choisit le chiffre 5 à la place du chiffre 6.	Cette correction correspond au résultat obtenu après l'exécution du programme.
	12:12 L'élève 2 exécute le programme du personnage « papillon ».	
	12:14 L'élève dit : « j'ai pas initialisé » en se référant au positionnement du personnage « papillon ».	La position initiale du personnage « papillon » n'est pas correcte.
	12:15 L'élève 2 met de nouveau le personnage « papillon » en position initiale avec son doigt.	
	12:21 L'élève 2 lance de nouveau le programme du « papillon ».	Le personnage « papillon » arrive cette fois-ci au bon endroit dans la scène.
	12:25 La chercheuse lui pose la question « Là est-ce que c'est bon ? »	
	12:27 L'élève 2 répond : « oui ».	

Annexe 9 : Transcriptions d'actions de l'Élève 3, CP1

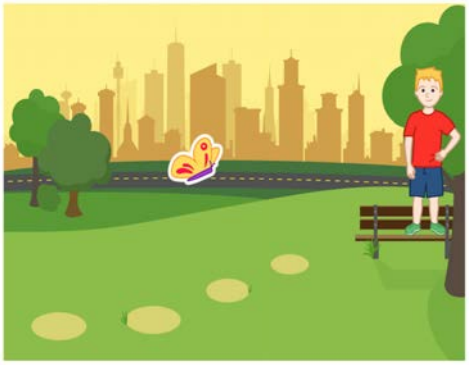
PB1

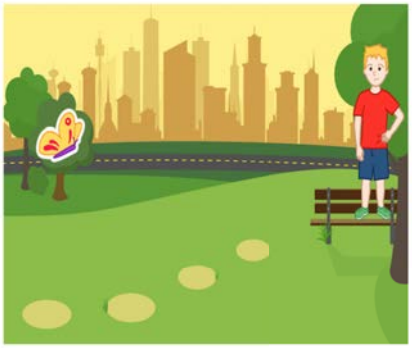
Déroulé des actions

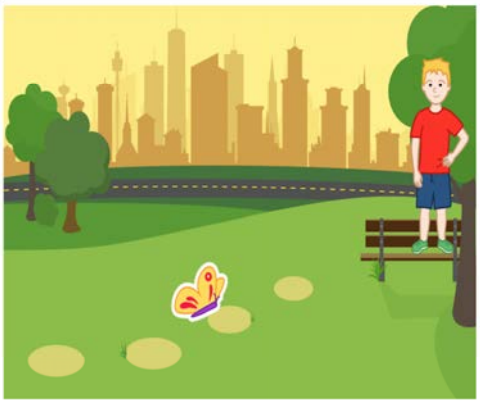
Captures d'écran	Transcription d'actions	Commentaires
	00:08 L'élève 3 efface le personnage « chat ».	Il s'agit d'une contrainte du logiciel ScratchJr, car chaque nouveau projet commence en ayant le personnage « chat » comme personnage par défaut.
	00 :10 L'élève 3 insère correctement l'arrière-plan.	
	00 :30 L'élève 3 insère correctement le personnage « papillon ».	
	00:33 L'élève 3 met le personnage « papillon » en position initiale avec son doigt.	
	00:47 L'élève 3 insère le personnage « enfant ».	
	00 :50 L'élève 3 met le personnage « enfant » en position initiale avec son doigt.	Mauvaise initialisation de la position du personnage « enfant ».
	00:50 La chercheuse intervient pour rappeler la consigne du problème à l'élève et surtout le positionnement de deux	Aide : Consigne

	personnages.	
	00:53 L'élève 3 passe au personnage « papillon ».	
	00:58 L'élève 3 va dans la catégorie des commandes de « démarrage ».	
	00:59 L'élève 3 dépose la commande de « quand drapeau-vert cliqué ».	
	0 1 :04 L'élève 3 va dans la catégorie de s commandes de « déplacement » .	
	01:06 L'élève 3 dépose la commande de «déplacement d'un pas à droite » dans l'espace de programmation.	
	01:10 L'élève 3 choisit le chiffre 5 comme paramètre de la commande « déplacement d'un pas à droite ».	
	01 :23 L'élève 3 va dans la catégorie de commandes de « fin ».	
	01:30 L'élève 3 dépose la commande « fin » à la fin du programme du personnage « papillon ».	
	01 :34 Puis, l'élève 3 passe au personnage « enfant ».	
	01 :43	

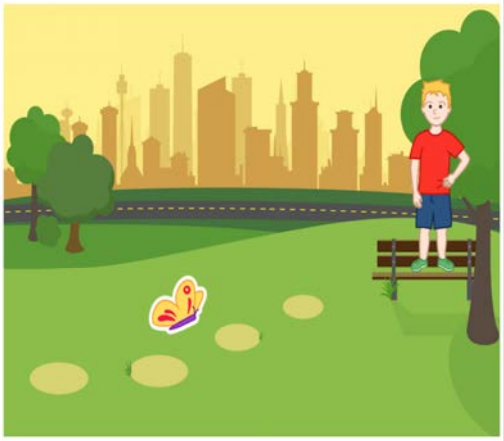
	L'élève 3 va dans la catégorie des commandes de « démarrage ».	
	01:45 L'élève 3 choisit la commande « quand message reçu » et il la met au début du programme du personnage « enfant ».	
	01 :48 L'élève 3 passe à la catégorie de commandes de « déplacement ».	
	01:48 L'élève 3 choisit la commande de « déplacement d'un pas à droite » et il ajoute au programme du personnage « enfant ».	
	02:02 L'élève 3 choisit le chiffre 3 comme paramètre de la commande « déplacement d'un pas à droite ».	
	02 :09 L'élève 3 va dans la catégorie de commandes de « fin » .	
	02:10 L'élève 3 dépose la commande « fin » à la fin du programme du personnage « enfant ».	
	02 :12 L'élève 3 passe au personnage « papillon ».	

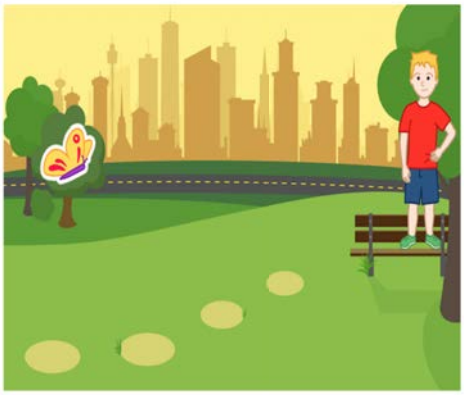
	02:13 L'élève 3 lance le programme du personnage « papillon ».	
	02:14	Erreur : Le personnage « papillon » se déplace trop vers la droite.
		Erreur : Le personnage « papillon » ne se déplace pas vers le bas.
		Erreur : Oubli de la commande « Envoyer un message ».
		Erreur : La position initiale du personnage « enfant » est erronée.
		Erreur : La direction de déplacement du personnage « enfant » est erronée.

	02 :15 L'élève 3 remet le personnage « papillon » en position initiale avec son doigt.	
	02 :19 L'élève 3 ouvre le programme du personnage « papillon » .	
	02:19 L'élève 3 va dans la catégorie de commandes de « déplacement ».	
	02:22 L'élève 3 ajoute la commande de « déplacement d'un pas en bas ».	
	02:25 L'élève 3 choisit le chiffre 4 comme paramètre de la commande « déplacement d'un pas en bas ».	Correction seulement de l'erreur sur le déplacement en bas du «papillon ». Les autres erreurs restent sur les programmes de ses personnages. Il s'agit d'une correction qui correspond au résultat obtenu après l'exécution

		du programme du « papillon », mais elle n'est pas suffisante pour corriger l'erreur choisie.
	02:28 L'élève 3 rassemble la commande de « fin » avec les autres commandes du programme du personnage « papillon ».	
	02 :30 L'élève 3 exécute le programme du personnage « papillon » pour voir si sa programmation est correcte.	
	02:32	Erreur : Le « papillon » se déplace trop à droite.
		Erreur : Le « papillon » ne se déplace pas assez vers le bas.
		Erreur : Oubli de la commande « envoyer un message ».
		Erreur : La position initiale du personnage « enfant » est

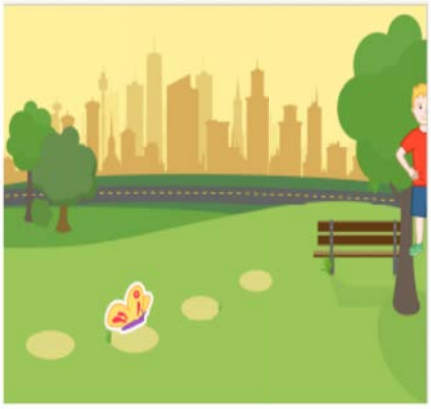
		erronée.
		Erreur : La direction du personnage « enfant » est erronée.
	02 :33 L'élève 3 remet le personnage « papillon » en position initiale avec son doigt.	
	02:43 L'élève 3 change le paramètre de la commande « déplacement d'un pas à droite » en choisissant le chiffre 4 au lieu de 5 .	Correction de l'erreur de « déplacement à droite » pour le « papillon ». Les autres erreurs restent sur les programmes de ces personnages. Il s'agit donc d'une correction qui correspond au résultat obtenu après l'exécution, mais elle n'est pas suffisante pour corriger le programme du personnage « papillon ».

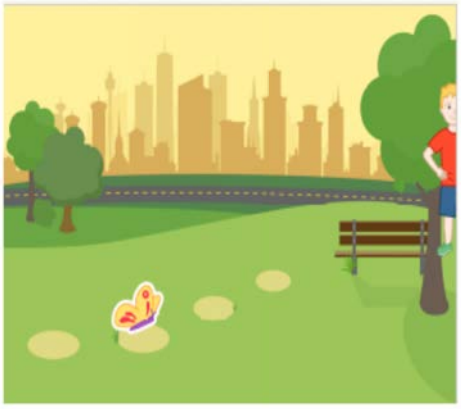
	02 :53 L'élève 3 exécute à nouveau le programme du personnage « papillon » pour voir si le « papillon » arrive au bon endroit dans la scène.	
	02:58	Erreur : Le personnage « papillon » se déplace trop vers la droite.
		Erreur : Le personnage « papillon » ne se déplace pas assez vers le bas.
		Erreur : Oubli de la commande « envoyer un message » par le programme du personnage « papillon ».
		Erreur : La direction du personnage « enfant » est erronée.
		Erreur : La position initiale du personnage

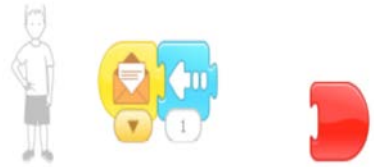
		« enfant »est erronée.
	03 :00 L'élève 3 corrige à nouveau le programme du personnage « papillon » en modifiant le paramètre de la commande « déplacement d'un pas en bas » et en choisissant le chiffre 5 au lieu de 4.	Correction du déplacement à droite du « papillon ». Il s'agit d'une correction qui correspond au résultat obtenu par l'exécution du programme du « papillon », mais elle n'est pas suffisante pour corriger cette erreur. Les autres erreurs restent sur les programmes de ses personnages.
	03 :04 L'élève 3 remet le personnage « papillon » en position initiale avec son doigt.	
	03 :10 L'élève 3 passe au personnage « enfant ».	
	03:22 L'élève 3 revient au personnage « papillon ».	
	03:23 L'élève 3 exécute à nouveau le programme du personnage « papillon » pour voir si le papillon arrive au bon endroit dans la scène.	

	03:25	Erreur : Le papillon se déplace trop vers la droite encore.
		Erreur : Le personnage « papillon » ne se déplace pas assez vers le bas.
		Erreur : Oubli de la commande « envoyer un message » par le programme du personnage « papillon ».
		03:25 Erreur : La position initiale du personnage « enfant » est erronée.
		03:25 Erreur : La direction du personnage « enfant »est erronée.
	03 :28 L'élève 3 met le personnage « papillon » en position initiale avec le doigt.	
	03:32 L'élève 3 va dans la catégorie des commandes de « contrôle ».	
	03:35	

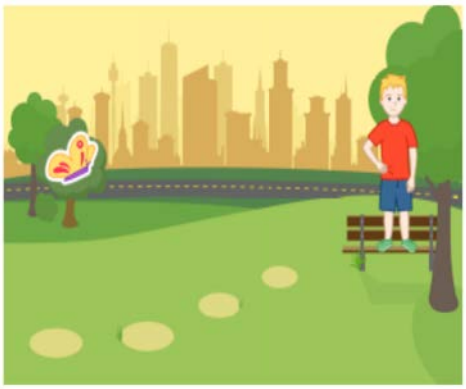
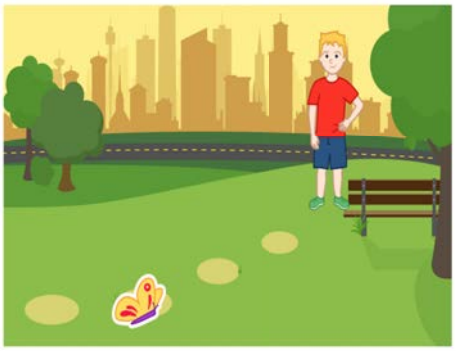
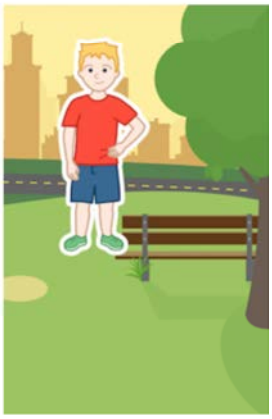
	L'élève 3 va dans la catégorie des commandes de « démarrage ».	
	03:37 L'élève 3 dépose la commande « envoyer un message » dans l'espace de programmation du personnage « papillon ».	
	03:39 L'élève 3 ajoute la commande « envoyer un message » au programme du personnage « papillon ».	Il s'agit d'une correction qui correspond au résultat obtenu après l'exécution du programme du « papillon ».
	03:46 L'élève 3 change le paramètre de la commande de « déplacement d'un pas à droite » et il choisit le chiffre 3 au lieu de 4.	Il s'agit d'une correction qui correspond au résultat obtenu par l'exécution du programme du « papillon ». C'est la seule fois que l'élève corrige deux erreurs avant le même exécution.
	03:51 La chercheuse lui pose la question suivante : « Ton personnage « papillon » est arrivé au bon endroit ou pas ? Regarde le projet modèle »	Aide : Vérification de la programmation
	03:51 La chercheuse lui rappelle la consigne du problème donné.	Aide : Consigne

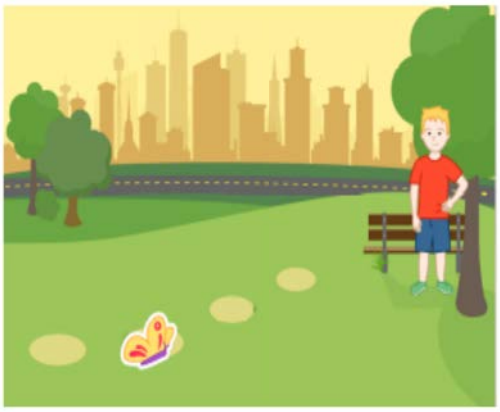
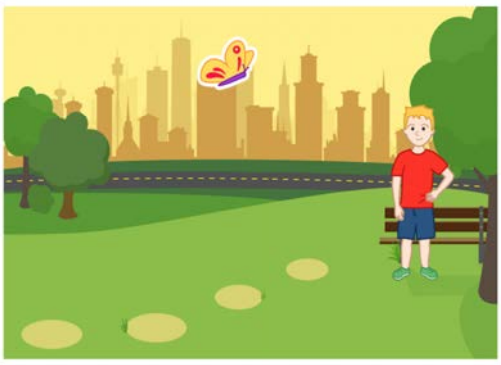
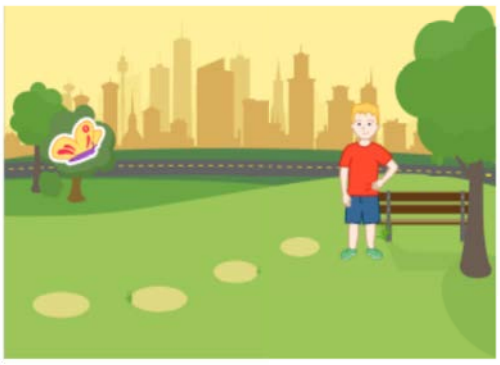
	04 :07 L'élève 3 exécute le programme du « papillon » de nouveau.	
	04:08	Erreur : Le personnage « papillon » ne se déplace pas assez vers le bas .
		Erreur : Le personnage « enfant » se déplace dans la mauvaise direction.
		Erreur : La position initiale du personnage « enfant » est erronée.
	04 :10 L'élève 3 remet le personnage « papillon » en position initiale avec son doigt.	

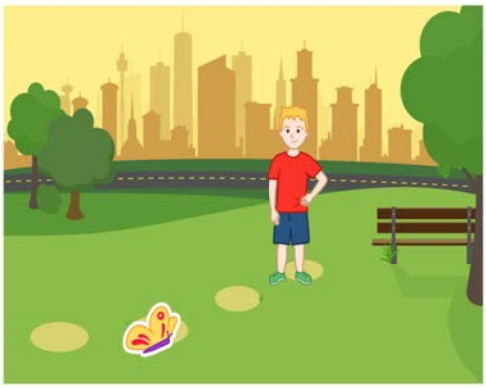
	04:17 L'élève 3 exécute à nouveau le programme du personnage « papillon ».	
	04:17	Erreur : Le personnage « papillon » ne se déplace pas assez vers le bas.
		Erreur : Le personnage « enfant » se déplace dans la mauvaise direction.
		04:17 Erreur : La position initiale du personnage « enfant » est erronée.
	04:19 L'élève 3 remet le personnage « enfant » en position initiale avec son doigt.	Position initiale du personnage « enfant » erronée.
	04:21 L'élève 3 passe au personnage « enfant ».	

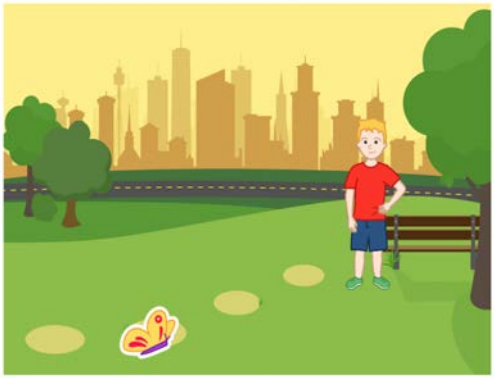
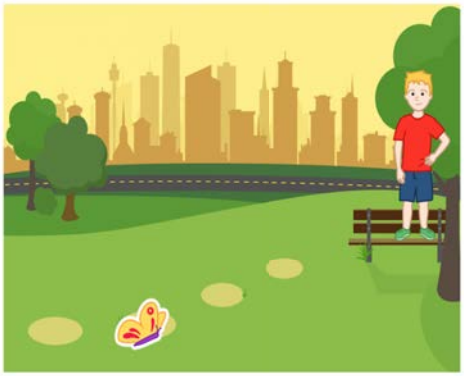
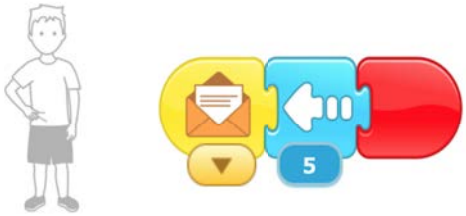
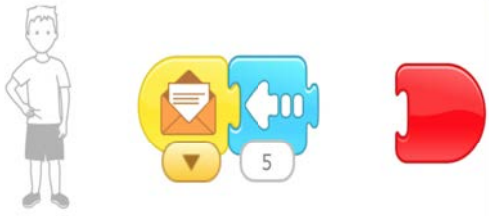
	04:23 L'élève 3 ouvre le programme du personnage « enfant ».	
	04:24 L'élève 3 enlève la commande « déplacement d'un pas à droite ».	Première correction sur le programme du personnage « enfant ». Il s'agit d'une correction qui correspond au résultat obtenu après l'exécution du programme du « papillon », mais elle n'est pas suffisante pour corriger l'erreur sur le déplacement à droite du personnage « enfant ».
	04 :25 L'élève 3 va dans la catégorie des commandes de « déplacement ».	
	04:27 L'élève 3 choisit la commande « déplacement d'un pas à gauche ».	
	04:29 L'élève 3 rassemble la commande de « fin » avec les autres commandes du programme du personnage « enfant ».	

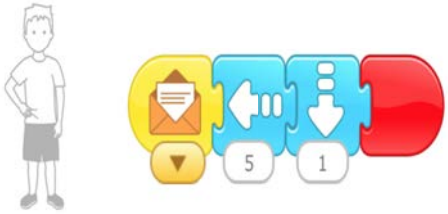
 	04:34 L'élève 3 choisit le chiffre 3 comme le paramètre de la commande « déplacement d'un pas à gauche ».	Cette correction arrive après la 6e exécution du programme de « papillon ». Il s'agit d'une correction qui correspond au résultat obtenu après l'exécution du programme du « papillon », mais elle n'est pas suffisante pour corriger l'erreur du déplacement à gauche de l' « enfant ». Deuxième fois que l'élève 3 corrige deux erreurs avant une exécution.
	04 :40 L'élève 3 passe au programme du personnage « papillon ».	
 	04:45 L'élève 3 change le paramètre de la commande de « déplacement d'un pas en bas » et il choisit le chiffre 6 au lieu du chiffre 5 .	Il s'agit d'une correction qui correspond au résultat obtenu après l'exécution du programme du personnage « papillon ».

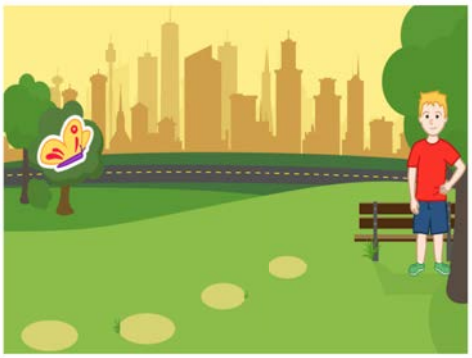
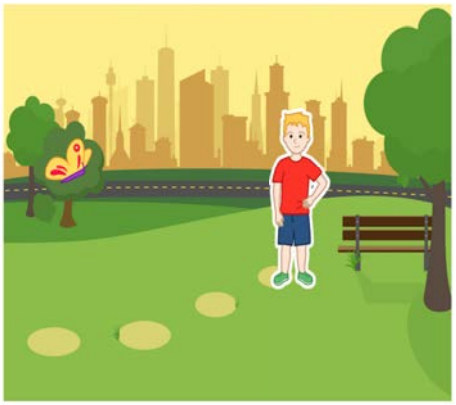
	04:48 L'élève 3 remet le personnage « papillon » en position initiale avec son doigt.	Moment de blocage : L'élève initialise et lance le programme du personnage « papillon » quatre fois consécutives sans apporter des corrections.
	04:50 L'élève 3 lance le programme du personnage « papillon ».	
	04:50	Erreur : La position initiale du personnage « enfant » est erronée.
		04:50 Erreur : Le nombre de pas du personnage « enfant » est erronée.
	04:53 L'élève 3 remet le personnage « enfant » en position initiale avec son doigt.	Erreur : Position initiale du personnage « enfant » erronée.

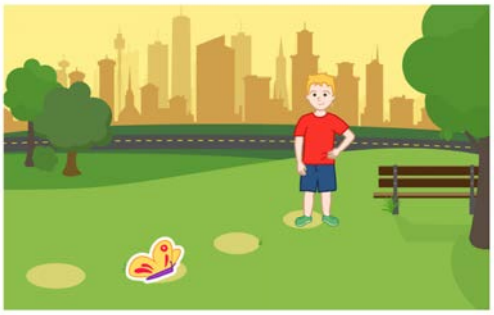
	05:00 La chercheuse l'aide à choisir la correcte position initiale pour le personnage « enfant ».	Aide : Initialisation
	05:08 L'élève 3 exécute à nouveau le programme du personnage « papillon ».	
	05:08	Erreur : La position initiale du personnage « papillon » est erronée.
		Erreur : Le nombre de pas du personnage « enfanr » est erroné.
	05:13 L'élève 3 remet le personnage « papillon » en position initiale avec son doigt.	Position initiale du personnage « enfant » erronée. L'élève se trompe de nouveau après l'apport d'aide.

	05:20 L'élève 3 exécute à nouveau le programme du personnage « papillon ».	
	05:21	Erreur : La position initiale du personnage « enfant » est erronée.
		Erreur : Le nombre de pas du personnage « enfant » est erroné.
	05:25 La chercheuse lui demande « est-ce que c'est bien ? »	Aide : Vérification de programmation
	05 :25 L'élève 3 remet les personnages en position initiale avec l'icône d'initialisation.	

	05:26 L'élève 3 exécute une dernière fois son projet pour valider s'il a bien programmé les deux personnages.	Moment de blocage terminée : Celle-ci est la dernière fois qu'il exécute son programme sans apporter des modifications aux programmes de ces personnages.
	05:28	Erreur : Le nombre de pas du personnage « enfant » est erroné.
	05:29 L'élève 3 remet le personnage « enfant » en position initiale avec son doigt.	Position initiale du personnage « enfant » erronée.
	05 :35 L'élève 3 change le paramètre de la commande de « déplacement d'un pas à gauche » du personnage « enfant » et il choisit le chiffre 5 au lieu de 3.	Il s'agit d'une correction qui correspond au résultat obtenu par l'exécution du projet.
	05:39 L'élève 3 ouvre le programme du personnage « enfant ».	

	05:41 L'élève 3 ajoute la commande « déplacement d'un pas en bas » .	
	05:43 L'élève 3 rassemble la commande de « fin » aux autres commandes du programme du personnage « enfant ».	Il s'agit d'une correction qui correspond au résultat obtenu après l'exécution du projet, mais elle est erronée parce que la position initiale du personnage « enfant » était erronée et cela affecte la programmation de l'élève.
	05:45 La chercheuse lui montre la correcte position initiale. Elle intervient aussi pour effacer la commande de « déplacement d'un pas vers le bas » par le programme du personnage « enfant ».	Aide : Initialisation Effacement d'une commande
	05:48 L'élève 3 rassemble la commande « fin » avec les autres commandes du programme du personnage « enfant ».	
	05:49 L'élève 3 remet le personnage « enfant » en position initiale avec son doigt	
	05:50	

	L'élève 3 remet les personnages en position initiale avec l'icône d'initialisation.	
	05:51 L'élève 3 exécute le programme du personnage « enfant ».	
	05:51	Erreur : La position initiale du personnage « enfant » est erronée.
	05:52 L'élève 3 met en position initiale le personnage « enfant » avec son doigt.	La position initiale du personnage « enfant » n'est pas correcte.
	05:53 L'élève 3 passe au personnage « papillon ».	
	05:54 La chercheuse l'aide encore une fois à initialiser la position du	Aide : Initialisation

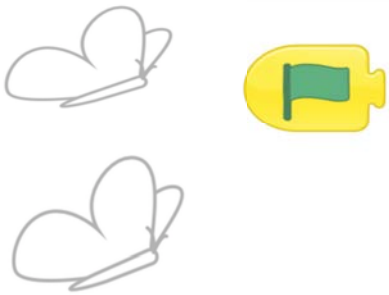
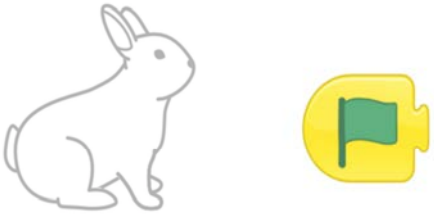
	personnage « enfant ».	
	05 : 55 L'élève 3 exécute le programme du personnage « papillon ».	
	06:02 La chercheuse lui demande « Est-ce que c'est bien maintenant ? ». L'élève répond « oui ». La chercheuse lui dit « je te montre le modèle et tu me diras ».	Aide : Vérification de programmation
	06:03 La chercheuse lui montre le projet modèle pour qu'il puisse vérifier si le programme est correct ou pas.	Aide : Lancement projet modèle
	06:10 L'élève 3 dit : c'est bien.	Le personnage « enfant » n'arrive pas tout à fait sur l'endroit demandé parce la position initiale du personnage « enfant » est toujours erronée, même si le nombre de pas de son déplacement est correct selon la solution proposée par le projet modèle.

PB2

Déroulé des actions

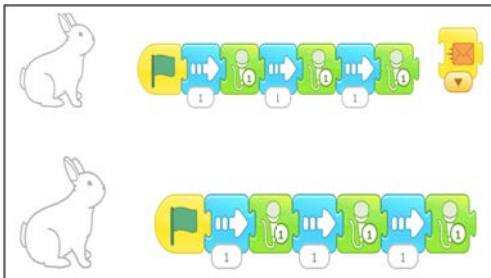
Captures d'écran	Transcription d'actions	Commentaires

	08:24 L'élève 3 efface le personnage sans problème.	Il s'agit d'une contrainte du logiciel ScratchJr, car chaque nouveau projet commence en ayant le personnage « chat » comme personnage par défaut.
	08:34 L'élève 3 insère l'arrière-plan sans problème.	
	0 8:38 L'élève 3 insère le personnage « lapin » sans problème.	
	08:39 L'élève 3 met le personnage « lapin » en position initiale avec son doigt.	
	08:43 L'élève 3 insère le personnage « papillon » sans problème.	
	08:45 L'élève 3 prend le personnage « papillon » avec le doigt et le déplace jusqu'à l'endroit demandé.	
	08:46 L'élève 3 initialise la position du personnage « papillon » avec le doigt.	
	08:49 La chercheuse intervient pour lui rappeler la consigne et	Aide : Consigne

	surtout les positions initiales de deux personnages.	
	08:51 L'élève 3 met le personnage « papillon » en position initiale avec le doigt.	
	08:52 L'élève 3 met le personnage « lapin » en position initiale avec le doigt.	
	La chercheuse demande à l'élève ensuite « Par qui tu veux commencer ? Par le papillon ou par le lapin ? C'est comme tu veux. »	
	08:53 L'élève 3 va dans la catégorie des commandes de « démarrage ».	
	08 :54 L'élève 3 choisit la commande de « quand drapeau-vert cliqué » et avant de la déposer il l'efface.	Moment de blocage : L'élève n'est pas sûr s'il faut utiliser la commande « quand drapeau-vert cliqué ».
	09:04 L'élève 3 passe au personnage « lapin ».	
	09:06 L'élève 3 choisit la commande de « drapeau-vert » et il la dépose dans l'espace de programmation du personnage « lapin ».	

	09:14 L'élève 3 va dans la catégorie des commandes de « déplacement ».	
 	09:16 L'élève 3 choisit la commande « déplacement d'un pas à droite » et il la rassemble avec la commande de « quand drapeau-vert cliqué ».	
	09 :18 L'élève 3 passe à la catégorie des commandes de « son ».	
	09:27 L'élève 3 enregistre correctement le son « Bonjour le papillon »	
 	09:28 L'élève 3 ajoute la commande du « son » au programme du personnage « lapin » juste après la commande de « déplacement d'un pas à droite ».	
	09 :33 L'élève 3 va dans la catégorie des commandes de « déplacement ».	
 	09:34 L'élève 3 choisit une commande de « déplacement d'un pas à droite » et il la dépose dans l'espace de programmation du personnage « lapin » juste après la commande du « son ».	

	09 :36 L'élève 3 va dans la catégorie des commandes de « son »	
	09:37 L'élève 3 choisit la commande du « son » et il la dépose dans l'espace de programmation du personnage « lapin ».	
	09:39 L'élève 3 va dans la catégorie des commandes de « déplacement ».	
	09:41 L'élève 3 choisit la commande de « déplacement d'un pas à droite » et il la dépose dans l'espace de programmation du personnage « lapin » juste après la commande du « son ».	
	09:43 L'élève 3 va dans la catégorie des commandes de « son ».	
	09:45 L'élève 3 choisit encore une commande de « son » et il l'ajoute au programme du personnage « lapin » juste après la commande de « déplacement d'un pas à droite ».	
	09:49 L'élève va dans la catégorie des commandes de « démarrage ».	

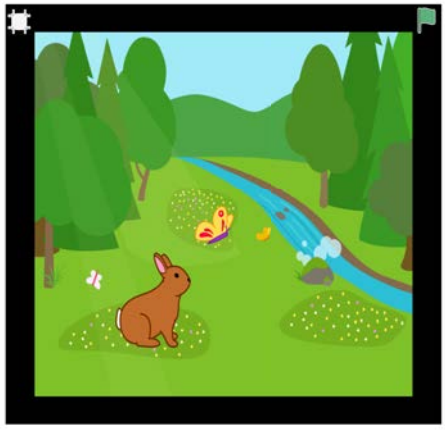
	09:50 L'élève 3 dépose la commande « envoyer un message » juste après la commande du son mais il hésite à l'ajouter au programme du personnage « lapin ».	Moment de blocage : L'élève n'est pas sûr s'il faut utiliser la commande « envoyer un message » sur le programme du personnage «lapin », c'est pourquoi il la garde à côte de son programme avec son doigt. Nous observons que l'élève essaie d'utiliser la commande « envoyer un message » même s'il n'en a pas besoin.
	09:59 La chercheuse lui rappelle la consigne du problème.	Aide : Consigne.
	10:00 L'élève 3 efface la commande « envoyer un message ».	Juste après le rappel de la consigne du problème donné, l'élève efface la commande « envoyer un message » de l'espace de programmation du « lapin ». Moment de blocage terminé.
	10:01 L'élève 3 va dans la catégorie des commandes de « fin ».	
	10:02 L'élève 3 ajoute la commande de « fin » rouge au programme du personnage «lapin ».	
	10 :03 L'élève 3 passe au programme du personnage « papillon ».	

	10 :07 L'élève 3 va dans la catégorie des commandes de « démarrage ».	
	10:09 L'élève 3 ajoute la commande « quand drapeau vert cliqué ».	Moment de blocage : L'élève n'est pas sûr s'il faut utiliser la commande « quand drapeau-vert cliqué ».
	10 :12 L'élève 3 efface la commande « quand drapeau-vert cliqué ».	
	10:13 L'élève 3 ajoute la commande « quand message reçu » à la place de la commande du «quand drapeau-vert cliqué».	Finalement l'élève utilise la commande « quand message reçu » au début du programme du personnage « papillon », alors que ce n'était pas nécessaire.
	10 :21 L'élève 3 va dans la catégorie des commandes de « déplacement ».	
	10:22 L'élève 3 ajoute une commande « déplacement d'un pas à droite » juste après la commande « quand message reçu ».	
	10:23 L'élève 3 ajoute encore une commande « déplacement d'un pas à droite ».	

	10:26 L'élève 3 ajoute encore une commande « déplacement d'un pas à droite ».	
	10:32 L'élève 3 ajoute encore une commande « déplacement d'un pas à droite ».	Plusieurs fois la même commande et réduction de l'écriture.
	10 :35 L'élève 3 enlève toutes les commandes de « déplacement d'un pas à droite ».	
	10:39 L'élève 3 choisit une commande de « déplacement d'un pas à droite » et il la dépose dans l'espace de programmation à côté de la commande « quand message reçu ».	
	10:42 L'élève 3 appuie sur la même commande de « déplacement d'un pas à droite » plusieurs fois jusqu' à ce que le personnage « papillon » se déplace pour arriver en haut de l'endroit demandé et parallèlement il compte le nombre de pas que le personnage « papillon » fait pour arriver là-bas.	Stratégie de déplacement similaire à celle utilisée par l'Élève 2 mais sans la « grille ».
	L'élève 3 appuie sur la même commande encore une fois.	
	L'élève 3 appuie sur la même commande encore une fois.	
	L'élève 3 appuie sur la même commande encore une fois.	

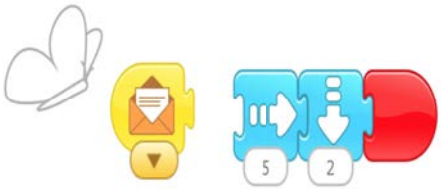
	L'élève 3 appuie sur la même commande encore une fois.	
	10 :46 L'élève 3 ajoute ensuite le chiffre 5 comme paramètre de la commande « déplacement d'un pas à droite ».	
	10 :47 L'élève 3 rassemble la commande « déplacement d'un pas à droite » avec la commande « quand message reçu ».	
	10:51 L'élève 3 choisit la commande « déplacement d'un pas en bas » et il la dépose dans l'espace de programmation du personnage « papillon ».	Stratégie de déplacement similaire que celle utilisée par l'élève 2, mais sans la « grille ».
	10:56 L'élève 3 appuie une fois sur celle-ci.	
	10:57 L'élève 3 appuie encore une fois sur celle-ci.	
	10:58 L'élève 3 choisit le chiffre 2 comme paramètre de la commande de « déplacement d'un pas en bas »	

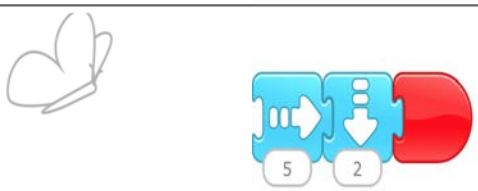
	10:59 L'élève 3 rassemble la commande « déplacement d'un pas en bas » aux deux autres commandes.	
	11 :00 L'élève 3 va dans la catégorie des commandes de « fin ».	
	11:02 L'élève 3 ajoute la commande de « fin » à la fin du programme du personnage « papillon ».	
	11 :05 L'élève 3 revient au personnage « lapin ».	
	11:06 L'élève 3 exécute le programme du personnage « lapin » pour voir s'il a bien programmé.	Le programme du personnage « lapin » est correct dès la première exécution.
	11:18 La chercheuse intervient en invitant l'élève à aller en mode plein-écran pour vérifier s'il a bien programmé.	Aide : Vérification de programmation
	11 :24	

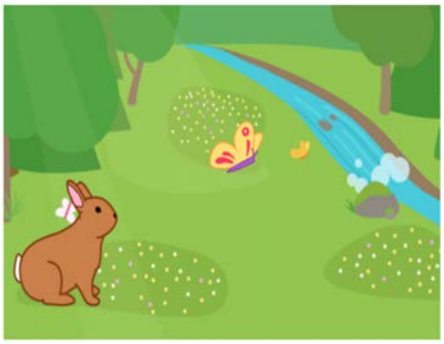
	L'élève 3 va en plein-écran.	
	11:27 L'élève 3 met les personnages en position initiale avec son doigt.	
	11.30 L'élève 3 exécute le projet en plein-écran.	
	11:33	Le personnage « papillon » ne déclenche pas son programme. Le personnage « papillon » se déplace trop vers le bas. Moment de blocage : L'élève regarde sa tablette sans rien faire pendant 9 secondes.
	11 :42 La chercheuse lui rappelle la consigne du problème.	Aide : Consigne
	11:42 La chercheuse exécute le projet modèle.	Aide : Lancement du projet modèle
	11:46 La chercheuse lui demande « Est-ce que ça marche sur le tien ? Qu'est-ce qui se	Aide : Identification de l'erreur

	passe ? »	
	11:49 L'élève 3 quitte le plein-écran.	
	11:54 L'élève 3 regarde l'écran de sa tablette, mais il ne fait rien.	Moment de blocage : L'élève regarde sa tablette sans rien faire pendant 8 secondes.
	11:58 La chercheuse lui pose la question suivante pour l'aider à identifier l'erreur : « Qu'est-ce qui se passe sur ton projet? »	Aide : Identification de l'erreur
	12 :00 L'élève 3 ne répond pas.	
	12:04 La chercheuse exécute de nouveau le projet de l'élève.	Aide : Lancement du projet de l'élève
	12:04 La chercheuse lui pose la question suivante : « qui ne se déplace pas dans ton projet ? »	Aide : Identification de l'erreur
	12:14 L'élève 3 répond silencieusement : « le lapin ».	Mauvaise réponse de l'élève à la question posée, car le personnage qui ne déclenche pas son programme est le « papillon ».
	12:16 La chercheuse lui pose la question suivante en exécutant de nouveau le projet de l'élève : « et le papillon ? Pourquoi le	Aide : Identification de l'erreur

	« papillon » ne se déplace pas ? »	
	12:28 L'élève 3 va dans l'espace de programmation du personnage « papillon ».	Moment de blocage : L'élève change des catégories de commandes pendant 24 secondes. Aide : Consigne
	12:32 L'élève 3 va dans la catégorie des commandes de « déplacement ».	
	12:42 La chercheuse intervient pour lui rappeler la consigne du problème.	
	12:43 L'élève 3 va dans la catégorie des commandes de « contrôle ».	
	12:47 L'élève 3 va dans la catégorie des commandes de l' « apparence ».	
	12:52 L'élève 3 va dans la catégorie des commandes de « démarrage ».	
	13:00 La chercheuse lui rappelle à nouveau la consigne du problème. La chercheuse demande « comment on a appris de faire ça ? »	Aide : Consigne

	13:10 L'élève 3 ouvre le programme du personnage « papillon ».	
	13:15 L'élève 3 choisit la commande de « répétition » et il essaie de l'ajouter au programme du personnage « papillon » avec beaucoup d'hésitation.	Moment de blocage : L'élève essaie en hésitant beaucoup d'ajouter la commande de « répétition » pour corriger le programme du personnage « papillon ». Cette correction ne corresponde pas au résultat obtenu après l'exécution du programme de personnage « papillon » et la consigne du problème donné.
	13:20 La chercheuse l'interrompt et ne le laisse par ajouter la commande de « répétition ». Elle lui dit « fais attention à ce que je te dis parce que ce que tu faisais jusqu'ici était bien ».	Effacement d'une commande
	13:21 L'élève 3 enlève la commande de « répétition » de l'espace de programmation du personnage « papillon ».	Moment de blocage terminé : Après avoir reçu de l'aide là-dessus, il efface la commande de « répétition ».
	11:24 La chercheuse intervient encore une fois pour lui rappeler la consigne du problème.	Consigne

	La chercheuse complète « On veut pas faire quelque chose plusieurs fois, on veut qu'ils commencent leurs programmes en même temps. Je pense qu'à part cela ton programme était correct ».	
	13 :33 L'élève 3 va dans la catégorie de commandes de « déplacement ».	Moment de blocage: Changement des catégories de commandes pendant 6 secondes.
	13:35 L'élève 3 va ensuite dans la catégorie de commandes d' « apparence ».	
	13:39 L'élève 3 va ensuite dans la catégorie de commandes de « démarrage ».	
	13:40 L'élève 3 prend la commande « quand message reçu » avec ces doigts et il la déplace vers la palette des commandes pour l'effacer, mais en hésitant beaucoup.	Cette correction correspond au résultat obtenu après l'exécution de son projet en plein-écran et il arrive donc à corriger l'erreur existante sur le déclenchement du programme du personnage « papillon ».
	13:54 L'élève 3 dépose la commande de « drapeau vert » en position initialet du programme du personnage « papillon ».	
	13:58 L'élève 3 passe au personnage « lapin ».	Moment de blocage : Changement des catégories de commandes pendant

		9 secondes.
	14:00 L'élève 3 passe au personnage « papillon ».	
	14:07 L'élève 3 va en plein-écran.	
	14:10 L'élève 3 remet les personnages en position initiale avec son doigt.	
	14:11 La chercheuse exécute le projet modèle pour aider l'élève à vérifier l'exactitude de sa programmation.	Aide : Lancement projet modèle
	14:23 L'élève 3 exécute son projet.	
	14:25	Le personnage « papillon » se déplace trop vers le bas. Moment de blocage : L'élève regarde sa tablette sans réaliser d'action pendant 12 secondes.
	14:37 La chercheuse lui pose à nouveau la question : « est-ce que tes personnages arrivent	Aide : Vérification de programmation

	au même endroit que les personnages du projet modèle ? »	
	14:40 L'élève ne répond pas à la question posée.	
	14:41 La chercheuse lui pose à nouveau la question : « est-ce que tes personnages arrivent au même endroit que les personnages du projet modèle ? »	Aide :Vérification de programmation
	14:42 L'élève 3 quitte le mode pleine-écran.	
	14:43 L'élève 3 change le paramètre de la commande « déplacement en bas » en choisissant le chiffre 1 au lieu du chiffre 2.	Correction qui correspond au résultat obtenu après l'exécution du projet, car il arrive à corriger la commande « déplacement en bas » pour aider le personnage « papillon » à arriver au bon endroit dans la scène.
	14:48 L'élève 3 prend le personnage « papillon » avec ses doigts pour le mettre en position initiale, mais il laisse le personnage « papillon » sur le point d'arrivée et pas en position initiale.	L'élève ne choisit pas la bonne position avec le doigt pour le personnage « papillon ».
	14:50 L'élève 3 initialise maintenant la position des personnages avec l'icône de	La position du personnage « papillon » est toujours erronée. Cela

	l'initialisation.	arrive parce que l'icône de l'initialisation de la position des personnages sur ScratchJr, sert à faire le personnage revenir à la position qui a été préalablement choisi par l'élève avec son doigt.
	14:54 La chercheuse lui rappelle qu'il faut mettre le personnage « papillon » en position initiale.	Aide : Initialisation
	14:55 L'élève 3 met le personnage « papillon » en position initiale avec son doigt.	
	14:56 La chercheuse intervient en invitant l'élève à aller en mode plein-écran pour vérifier s'il a bien programmé.	Vérification de programmation
	14:57 L'élève 3 va en mode pleine-écran.	
	14:58 L'élève 3 exécute son projet qui est maintenant correct.	
Discussion avec l'Élève 3 après la résolution des problèmes d'évaluation		

	La chercheuse lui demande « Est-ce que c'est bien ? » L'élève 3 répond « oui ».	
	La chercheuse lui pose une question sur l'utilisation de la commande « quand message reçu » pour qu'elle puisse comprendre pourquoi l'élève l'a utilisé auparavant. « Pourquoi tu as utilisé la commande « quand message reçu » au début du programme du personnage « papillon » ?	
	L'élève 3 ne répond pas à la question. Il dit : « Parce que... ».	
	La chercheuse lui montre la commande « quand message reçu » et elle lui pose la question suivante : « Quand on utilise cette commande ? »	
	L'élève 3 répond : « Je sais… Il faut que...on l'utilise quand on veut que le papillon commence en premier. »	La réponse de l'élève est erronée.
	La chercheuse lui montre le projet où il a utilisé les commandes de « messages » et elle lui pose à nouveau la même question : « Ici pourquoi tu as utilisé les messages ? »	
	L'élève répond : « Parce que le papillon dit au garçon qu'il peut commencer. »	Cette réponse est correcte.

Annexe 10 : Transcription d'actions de l'Élève 4, CP1

PB1

Déroulé des actions

Captures d'écran	Transcription d'actions	Commentaires
	0 1 :18 L'élève 4 insère correctement l'arrière-plan.	
	0 1:33 L'élève 4 efface le personnage « chat ».	Il s'agit d'une contrainte du logiciel ScratchJr, car chaque nouveau projet commence en ayant le personnage « chat » comme personnage par défaut.
	01:42 L'élève 4 insère le personnage « papillon ».	
	01:45 L'élève 4 met le personnage « papillon » en position initiale avec son doigt.	L'élève 4 initialise correctement la position de son personnage « papillon ».
	01:50 L'élève 4 insère correctement le personnage « enfant ».	

	01:54 L'élève 4 met le personnage « enfant » en position initiale avec son doigt.	La position du personnage « enfant » est erronée.
	01:58 La chercheuse intervient pour l'aider à initialiser la position du personnage « enfant ».	Aide :Initialisation
	02:03 La chercheuse demande à l'élève 4 : « Par qui tu veux commencer ? Par le papillon ou par le lapin ?	
	02:05 L'élève répond « par le papillon ».	
	02:06 L'élève 4 va sur l'espace de programmation du personnage 180 « papillon ».	
	02:07 La chercheuse rappelle à l'élève 4 la consigne du problème donné.	Aide : Consigne

	02:09 L'élève 4 dépose la commande « déplacement d'un pas à droite » dans l'espace de programmation.	L'élève 4 commence sa programmation par le déplacement.
	02:16 L'élève 4 va dans la catégorie des commandes de « démarrage ».	
	02:18 L'élève 4 va dans la catégorie des commandes de « fin ».	
	02:19 L'élève 4 dépose la commande de « fin » dans l'espace de programmation du personnage « papillon » et la pose à côté de la commande de « déplacement » sans rassembler les deux commandes.	
	02:24 L'élève 4 va dans la catégorie des commandes de « démarrage ».	
	02:25 L'élève 4 choisit la commande « quand drapeau-vert cliqué » et la pose dans l'espace de programmation du personnage « papillon ».	
	02:26 L'élève 4 rassemble la commande « quand drapeau-vert cliqué » avec la commande « déplacement à droite ».	

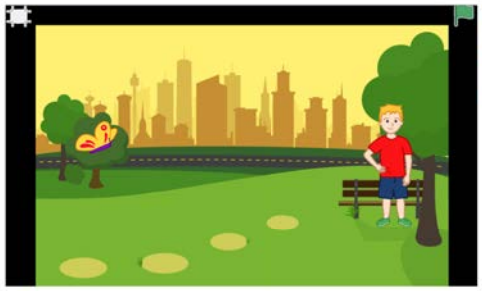
	02:30 L'élève 4 choisit le chiffre 4 comme paramètre de la commande de « déplacement d'un pas à droite ».	Estimation du paramètre de la commande de « déplacement ».
	02:34 L'élève 4 va dans la catégorie des commandes de « déplacement ».	
	02:35 L'élève 4 choisit la commande « déplacement d'un pas en bas » et la rassemble au programme déjà présent dans l'espace de programmation.	
	02:37 L'élève 4 choisit le chiffre 6 comme paramètre de la commande « déplacement en bas ».	Estimation du paramètre de la commande de « déplacement ».
	02:41 L'élève 4 exécute les commandes « quand drapeau-vert cliqué » et celles responsables du « déplacement » en appuyant sur celles-ci avec son doigt.	Erreur : Le personnage « papillon » se déplace trop vers la droite et il dépasse l'endroit demandé dans la scène. L'endroit demandé était le milieu du cercle. Erreur : Oubli de la commande « envoyer un message ». Erreur : Oubli de la commande « fin ».
	02:53 L'élève 4 change le paramètre de la commande « déplacement à droite » et au lieu du chiffre 4 il choisit le chiffre 3.	Aide : Consigne Cette correction démontre une réflexion préalable sur le résultat obtenu après l'exécution de ce

		programme.
	02:56 L'élève 4 initialise la position du personnage « papillon » avec son doigt.	L'élève 4 initialise correctement la position du personnage « papillon ».
	02:58 L'élève 4 exécute le programme du personnage « papillon » en appuyant sur celui-ci avec son doigt.	Le personnage « papillon » arrive maintenant au bon endroit dans la scène. Erreur : Oubli de la commande « envoyer un message ». Erreur : Oubli de la commande « fin ».
	03:00 La chercheuse intervient pour lui rappeler la consigne du problème donné.	Aide : Consigne
	03 :03 L'élève 4 va à la catégorie des commandes de « démarrage ».	
	03 :07 L'élève 4 ajoute la commande « envoyer un message » au programme du personnage « papillon » juste après la commande « déplacement de six pas en bas ».	Cette correction démontre une réflexion préalable sur le résultat obtenu après l'exécution de ce programme.
	03:09 L'élève 4 rassemble ensuite la commande de « fin ».	L'élève 4 termine le programme du personnage « papillon » et puis il avance à celui du

		personnage « enfant ». Cette correction démontre une réflexion préalable sur le résultat obtenu après l'exécution de ce programme.
	03:11 L'élève 4 va sur l'espace de programmation du personnage « enfant ».	
 	03:16 L'élève 4 dépose la commande « quand drapeau-vert cliqué » dans l'espace de programmation du personnage « enfant ».	
	03:18 L'élève 4 efface la commande « quand drapeau-vert cliqué » dans l'espace de programmation du personnage « enfant ».	
 	03:20 L'élève 4 dépose la commande « quand message reçu » dans l'espace de programmation du personnage « enfant ».	
	03:24 L'élève 4 va dans la catégorie des commandes de « déplacement ».	
 	03:28 L'élève 4 dépose la commande « déplacement d'un pas à gauche » dans l'espace de programmation du personnage « enfant » et la rassemble avec l'autre commande.	

 	03:31 Ensuite, il utilise le chiffre 4 comme paramètre de la commande de « déplacement à gauche ».	Estimation du paramètre de la commande de « déplacement ».
	03:32 L'élève 4 exécute le programme de deux commandes en appuyant sur celui-ci avec son doigt. L'élève 4 dit : « C'est bon »	Erreur : Le personnage « enfant » n'arrive pas au bon endroit dans la scène, car il se déplace pas assez vers la gauche. L'endroit demandé était le milieu du cercle comme pour le personnage « papillon ». Erreur : Oubli de la commande « fin ».
	03 : 38 La chercheuse lui pose la question suivante : « Est-ce que ton personnage est arrivé au bon endroit dans la scène ?	Aide : Vérification de programmation
	03:42 L'élève 4 répond : « Oui ? »	
	03:46 La chercheuse lui pose la question suivante : « C'est bon ? »	Aide : Vérification de programmation
	03:47 L'élève 4 répond : « Non. »	

	03:48 La chercheuse lui pose la question suivante : « Pourquoi ? »	Aide : Identification de l'erreur
	03:52 L'élève 4 montre avec ses doigts que le personnage « enfant » doit se déplacer encore un peu à gauche dans la scène, pour arriver sur l'endroit demandé. Parallèlement l'élève 4 dit : « Parce qu'il n'arrive pas au milieu »	
	03:55 L'élève 4 change le paramètre de la commande « déplacement à gauche » et au lieu du chiffre 4 il choisit le chiffre 5.	Cette correction démontre une réflexion préalable sur le résultat obtenu après l'exécution de ce programme.
	03:59 L'élève 4 met le personnage « enfant » en position initiale avec son doigt.	
	04:03 L'élève 4 lance le programme de deux commandes à nouveau.	Le personnage « enfant » arrive exactement sur le bon endroit dans la scène. Erreur : Oubli de la commande « fin ».

	04:06 La chercheuse lui pose la question suivante : « C'est bon ? Tu as tout fini ? »	Aide : Vérification de programmation
 	04:12 L'élève 4 ajoute au programme la commande de « fin ».	Cette correction démontre une réflexion préalable sur le résultat obtenu après l'exécution de ce programme.
Les actions qui suivent ont été demandées par la chercheuse.		
	04:16 L'élève 4 va en plein-écran.	
	04:28 L'élève 4 exécute son projet à l'aide du bouton « drapeau-vert » en haut de la scène.	Le projet est correct.

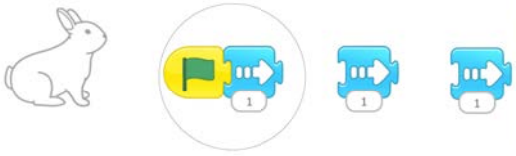
PB2

Déroulé des actions

Captures d'écran	Transcription d'actions	Commentaires
	06:30 L'élève 4 insère l'arrière-plan sans problème.	
	06:43 L'élève 4 efface le personnage « chat ».	Il s'agit d'une contrainte du logiciel ScratchJr, car chaque nouveau projet commence en ayant le personnage « chat » comme personnage par défaut.
	06:49 L'élève 4 insère le personnage « lapin » sans problème.	L'élève 4 initialise correctement la position du personnage « lapin ».
	06:52 L'élève 4 met le personnage « lapin » en position initiale avec son doigt.	La position initiale du personnage « lapin » est erronée.
	06:56 La chercheuse intervient pour rappeler à l'élève la consigne du problème pour l'aider à bien positionner son personnage.	Aide : Consigne
	06:59	

	L'élève 4 initialise correctement cette fois-ci la position de son personnage « lapin » avec le doigt.	
	07:04 L'élève 4 insère le personnage « papillon » .	
	07:11 L'élève 4 met le personnage « papillon » au début avec son doigt.	L'élève 4 initialise correctement la position du personnage « papillon ».
	07:17 L'élève 4 va dans l'espace de programmation du personnage « lapin ».	
 	07:18 L'élève 4 dépose la commande « déplacement d'un pas à droite » dans l'espace de programmation du personnage « lapin ».	
  	07:20 L' élève 4 dépose encore une commande de « déplacement d'un pas à droite » dans l'espace de programmation du personnage « lapin » et la pose à côté de l'autre commande sans la connecter avec elle.	
   	07:22 L'élève 4 dépose encore une commande de « déplacement d'un pas à droite » dans l'espace de programmation du	

	personnage « lapin » et la pose à côté de deux autres commandes.	
	07:31 L'élève 4 va dans la catégorie des commandes de « démarrage ».	
	07:34 L'élève 4 dépose la commande « quand drapeau-vert cliqué » dans l'espace de programmation du personnage « lapin » et il la rassemble avec la commande de « déplacement d'un pas à droite »	
	07:34-07:40 L'élève 4 se promène avec son doigt au-dessus des catégories des commandes sans choisir une catégorie pour utiliser des commandes.	Moment de blocage : Doigt au-dessus des catégories des commandes, mais pas de choix réalisé.
 Ensuite, 	07:42-07:50 L'élève 4 choisit la commande « envoyer un message » et la garde à côté de la commande de « déplacement ». Il la déplace ensuite vers la palette des commandes, puis vers l'espace de programmation à nouveau.	Moment de blocage : Insertion d'une commande et après effacement.
	07:50 L'élève 4 dépose la commande « envoyer un message » dans l'espace de la programmation.	
	07:52	

	L'élève 4 efface la commande « envoyer un message » de l'espace de programmation.	Moment de blocage terminé avec l'effacement de la commande.
	07:53 La chercheuse intervient pour lui rappeler la consigne du problème donné.	Aide : Consigne
	07:58 L'élève 4 exécute le programme de deux commandes en appuyant sur celui-ci.	Erreur : Oubli de la commande de « son ». Erreur : Oubli de la commande de « fin ». Erreur : Oubli de la commande de « répétition ». OU Erreur : Absence du motif de commandes répété 3 fois. Moment de blocage : Exécution d'un programme sans lui apporter des

		modifications après.
	07:59 L'élève 4 exécute le programme de deux commandes en appuyant sur celui-ci.	
	08:00 L'élève 4 exécute pour une troisième fois le programme de deux commandes en appuyant sur celui-ci.	
	08:01 L'élève 4 exécute pour une quatrième fois le programme de deux commandes en appuyant sur celui-ci.	
	08:02 L'élève 4 exécute pour une quatrième fois le programme de deux commandes en appuyant sur celui-ci.	Moment de blocage terminé.
	08: 05 L'élève 4 remet le personnage « lapin » en position initiale avec ses doigts.	L'élève 4 initialise correctement la position du personnage « lapin ».
	08:09 L'élève 4 va sur l'espace de programmation du personnage « papillon ».	
	08:15 L'élève 4 va dans la catégorie des commandes de « déplacement ».	
	08:16	

	L'élève 4 va dans la catégorie de commandes de « démarrage ».	
	08:19 L'élève 4 dépose la commande « quand drapeau-vert cliqué » dans l'espace de programmation du personnage « papillon ».	
	08:23 L'élève 4 va dans la catégorie des commandes de « déplacement ».	
	08:23 L'élève 4 dépose la commande de « déplacement d'un pas à droite » dans l'espace de programmation.	
	08:24 L'élève 4 rassemble la commande « déplacement d'un pas à droite » avec la commande « quand drapeau-vert cliqué ».	
	08:29 L'élève 4 change le paramètre de la commande de « déplacement » et il choisit le chiffre 5 comme son paramètre.	Estimation du paramètre de la commande de « déplacement ».
	08:34 L'élève 4 dépose la commande « déplacement d'un pas en bas » dans l'espace de programmation du personnage « papillon ».	
	08:35 L'élève 4 rassemble la	

	commande « déplacement d'un pas en bas » avec le programme déjà existant dans l'espace de programmation.	
	08:38 L'élève 4 choisit le numéro 2 comme son paramètre.	Estimation du paramètre de la commande de « déplacement ».
	08:40 L'élève 4 exécute le programme de trois commandes présentes dans l'espace de programmation du « papillon » en appuyant sur celui-ci avec ses doigts.	Erreur : Le personnage « papillon » se déplace trop en bas et il dépasse un peu l'endroit demandé dans la scène. L'endroit demandé était sur la pierre. Erreur : Oubli de la commande de « fin ».
	08:43-08:48 L'élève 4 a son doigt au-dessus de l'écran de la tablette, il l'approche un peu comme s'il voulait faire quelque chose et puis, il recule son doigt sans rien faire.	

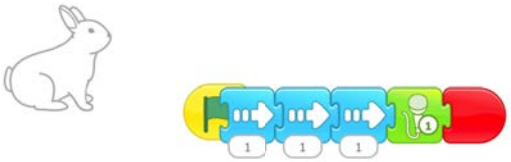
	08:44 La chercheuse intervient en lançant le projet modèle.	Aide : Lancement projet modèle
	08:49 L'élève 4 va dans la catégorie des commandes de « son ».	
	08:51 L'élève 4 appuie sur la commande « enregistrement d'un son » pour créer un son. L'élève 4 se trouve toujours sur l'espace de programmation du personnage « papillon ».	
	08:53 La chercheuse intervient pour lui rappeler la consigne du problème et pour l'arrêter avant d'enregistrer un son pour le personnage « papillon ».	Aide : Consigne Aide : Effacement d'une commande
	08:57 L'élève 4 va ensuite sur l' espace de programmation du personnage « lapin ».	
	09:00 La chercheuse intervient pour lui rappeler la consigne du problème donné.	Aide :Consigne
	09:06 L'élève 4 appuie sur la commande « enregistrement d'un son » pour créer un son. Il enregistre dans la commande de « son » trois fois la phrase demandée « bonjour le papillon ».	La phrase demandée « Bonjour le papillon » devrait être enregistrée une fois seulement dans la commande du « son ».

	09:16-09:23	Moment de blocage Temps sans actions sur la tablette.
	09 :24	Aide : Consigne
	09:38 L'élève 4 dépose la commande de « son » dans l'espace de programmation du personnage « lapin », mais il ne la connecte pas avec les autres commandes. Il la laisse à côté de la commande de « déplacement d'un pas à droite ».	
	09:42 L'élève 4 exécute tous les programmes commençant par la commande « quand drapeau-vert cliqué » en appuyant sur le bouton « drapeau-vert » en haut de la scène. Rappel : En exécutant le projet avec le bouton du drapeau-vert en haut de la scène les personnages commencent leur programme de leur dernière position initiale et non pas de la position où ils se trouvent dans la scène.	Erreur :Le personnage « papillon » dépasse toujours l'endroit demandé, car il se déplace trop vers le bas. Endroit demandé : sur la pierre. Erreur : Oubli de la commande de « fin ». Erreur : Oubli de la commande de « son », car elle

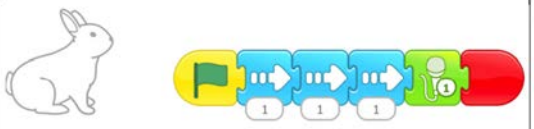
		ne fait pas partie du programme exécuté. Erreur : Absence de la commande de « répétition ». OU Erreur : Absence du motif répété trois fois. Erreur : Oubli de la commande de « fin ».
	09:43-09:46 L'élève 4 regarde sa tablette sans réaliser d'action.	Temps court pour le considérer en tant que moment de blocage.
	09:47 L'élève 4 rassemble la commande de « déplacement d'un pas à droite » avec les autres commandes.	Les actions de correction qui suivent ne démontrent pas une réflexion préalable sur le résultat obtenu après l'exécution de ce programme.

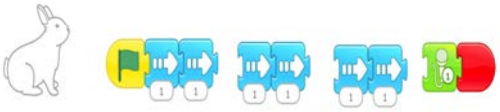
	09:48 L'élève 4 rassemble la commande de « déplacement à droite » avec les autres commandes.	
	09:49 L'élève 4 rassemble aussi la commande de « son » avec le programme présent dans l'espace de programmation du personnage « lapin ».	
	09:50 L'élève 4 va dans la catégorie des commandes de « fin ».	
	09:52 L'élève 4 dépose la commande de « fin » dans l'espace de programmation du personnage « lapin » et il la rassemble avec les autres commandes.	
	09:55 L'élève 4 initialise la position du personnage « papillon » avec le doigt.	La position initiale du personnage « papillon » est erronée.
	09:56 L'élève 4 lance à nouveau le programme du « lapin » en appuyant sur celui-ci avec son doigt.	Erreur : Le « lapin » se déplace trois pas à droite et puis il dit 3 fois « Bonjour le papillon » au lieu de réaliser un pas à droite et puis dire « Bonjour le papillon » et ce motif des commandes le

		répéte trois fois. Les actions de correction qui suivent ne démontrent pas une réflexion préalable sur le résultat obtenu après l'exécution de ce programme.
	10:00-10:10 L'élève 4 regarde sa tablette sans rien faire.	Moment de blocage : Temps sans actions sur la tablette.
	10:10-10:15 L'élève 4 regarde le projet modèle qui est lancé en plein-écran pour se rappeler du résultat attendu.	Aide : Lancement projet.
	10:16 L'élève 4 sépare les deux commandes de « déplacement d'un pas à droite » et il les pose à gauche au-dessus de la commande « quand drapeau-vert cliqué ».	Moment de blocage : Ouverture du programme après son exécution et déplacement des commandes existantes dans

		celui-ci à droite et à gauche, sans effectuer une action de correction.
	10:19 L'élève 4 prend les commandes « déplacement d'un pas à droite », « son » et « fin » et les pose à droite.	
	10:20 L'élève 4 prend les commandes « déplacement d'un pas à droite », « son » et « fin » et les rassemble à nouveau avec les autres commandes.	
	10:21 L'élève 4 déplace à nouveau avec son doigt un peu à droite les commandes « déplacement d'un pas à droite », « son » et « fin ».	
 Puis 	10:24 L'élève 4 choisit une des commandes de « déplacement à droite », il la déplace à droite et puis, la rassemble à nouveau à l'autre commande de « déplacement d'un pas à droite ».	Moment de blocage terminé
	10:34	

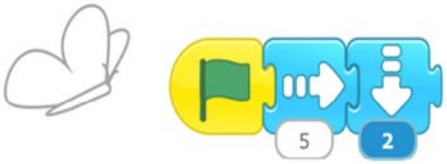
	L'élève 4 demande clairement d'aide par la chercheuse. « Comment il arrive (le lapin) à faire ça ? »	
	10:37 La chercheuse intervient pour rappeler à l'élève 4 la consigne du problème.	Aide : Consigne
	11:02 L'élève 4 pose encore une question à la chercheuse, en lui montrant avec son doigt les petits pas que le lapin doit faire dans la scène. « Mais comment il arrive à faire des petits pas comme ça ? »	
	11:04 La chercheuse lui répond : « C'est à toi de trouver comment il doit faire le « lapin ».	
	11:06 L'élève 4 rassemble les commandes de « déplacement d'un pas à droite » avec la commande « quand drapeau-vert cliqué ».	Moment de blocage : Ouverture du programme après son exécution et déplacement des commandes existantes dans celui-ci à droite et à gauche, sans effectuer une action de

		correction.
	11:10 L'élève 4 rassemble à nouveau les trois commandes « déplacement d'un pas à droite », « son » et « fin » avec les autres commandes pour créer un programme entier.	
	11:12 L'élève 4 sépare à nouveau les trois commandes « déplacement d'un pas à droite », « son » et « fin » et il les déplace à droite.	Moment de blocage terminé.
	11:13 L'élève 4 lance les premiers trois commandes c'est-à-dire la commande « quand drapeau-vert touché » et celles de « déplacement d'un pas à droite » en appuyant sur celles-ci avec son doigt.	Erreur : Oubli de la commande de « répétition ». Erreur : Oubli de la commande de « fin ». Erreur : Oubli de la commande de « son ».
	11:17 L'élève 4 met le personnage « lapin » en position initiale avec ses doigts. 11:18 L'élève 4 met le personnage « papillon » en position initiale avec son doigt.	L'élève 4 initialise correctement la position du personnage « lapin ». La position initiale du personnage « papillon » est erronée.

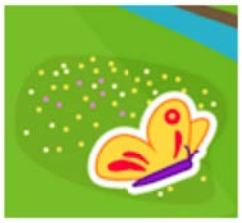
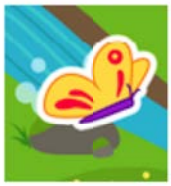
	11.20 L'élève 4 sépare la commande du « son » ainsi que la commande de « fin » par la commande de « déplacement d'un pas à droite » et les déplace à droite.	Les actions de correction qui suivent ne démontrent une réflexion préalable sur le résultat obtenu après l'exécution de ce programme.
	11:23 L'élève 4 ajoute une commande de « déplacement d'un pas à droite » et il la rassemble avec les commandes qu'il avait à droite dans son espace de programmation.	
	11:27 L'élève 4 sépare les deux commandes celles du « son » et de « fin » et il les pose à droite, en laissant les deux commandes de déplacement au milieu.	
	11:29 L'élève 4 ajoute encore une commande de « déplacement d'un pas à droite » et il l'ajoute à côté des autres commandes de déplacement.	
	11:32 L'élève 4 déplace encore plus loin à droite les commandes du « son » et de la « fin ».	
	11:35 L'élève 4 ajoute encore une commande de « déplacement	

	d'un pas à droite » et il la rassemble avec l'autre commande de « déplacement d'un pas à droite ».	
	11:40 L'élève 4 prend les commandes du « son » et de « fin » et il les rassemble avec les nouvelles commandes de « déplacement d'un pas à droite ».	
	11:43 L'élève 4 lance le programme de deux personnages, en utilisant le bouton du « drapeau-vert » en haut de la scène.	Erreur :Pas de commande de « son ». Erreur : Pas de commande de « fin ». Erreur : Pas de commande de « répétition ». OU Erreur : Pas de motif des commandes répété trois fois. Erreur : Le programme du « papillon » est toujours erroné, parce qu'il se déplace trop vers le bas et pas assez vers la droite, pour arriver au bon endroit dans la sècne. Erreur : La position initiale du personnage « papillon » est erronée aussi.

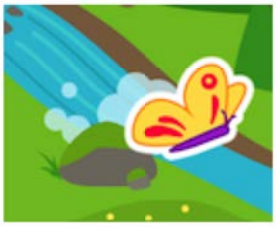
	11:44-11:48 L'élève 4 regarde sa tablette sans rien faire.	Temps court pour le considérer en tant que moment de blocage.
	11:50 La chercheuse lui pose la question suivante : « Le programme du personnage « papillon » est bon ?	Aide : Vérification de programmation
	11:51 L'élève 4 revient à l'espace de programmation du personnage « papillon ».	
	11:53 La chercheuse lui demande : « Est-ce que c'est bon le programme du « papillon » ?	Aide : Vérification de programmation
	11:54 L'élève 4 répond : « Non, parce que je n'ai pas mis le drapeau rouge ».	Quand l'élève parle d'un drapeau-rouge, il se réfère à la commande de « fin ».
	11:56 L'élève 4 va sur la catégorie des commandes de « fin ».	
	11:56 La chercheuse lui pose la question suivante : « Pourquoi ce n'est pas bon ? »	Aide : Identification de l'erreur
	11:58 L'élève 4 répond : « Parce que j'ai pas mis le drapeau rouge ».	Quand l'élève 4 parle d'un drapeau-rouge, il se réfère à la commande de « fin ».
	11:59	

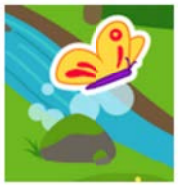
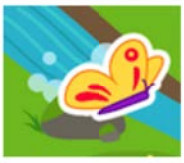
	L'élève 4 a son doigt au-dessus de la commande « fin », mais finalement il ne la choisit pas.	
	12:00 La chercheuse lui pose la question suivante : « Est-ce que ton personnage « papillon » arrive au bon endroit dans la scène ? »	Aide: Identification de l'erreur.
	12:05 L'élève 4 touche avec son doigt la commande de « déplacement en bas » pour lui changer son paramètre, mais finalement il ne le fait pas.	
	12:09 L'élève 4 touche avec son doigt la commande de « déplacement d'un pas à droite » et au lieu du chiffre 5 il choisit le chiffre 6.	L'élève 4 réalise cette correction sans réfléchir au fait que la position initiale du personnage « papillon » n'était pas correcte lorsqu'il a exécuté son programme la dernière fois.
	12:17 L'élève 4 met le personnage « papillon » en position initiale avec ses doigts.	L'élève 4 met le personnage « papillon » à côté de l'endroit demandé en réalisant ainsi une mauvaise initialisation de la position du personnage.

	12:19 L'élève 4 met le personnage « papillon » à un autre endroit avec ses doigts.	L'élève 4 met le personnage « papillon » encore plus loin de l'endroit demandé dans la scène, en réalisant toujours une mauvaise initialisation de la position du personnage.
	12:20 La chercheuse intervient pour l'aider à initialiser correctement la position de ses personnages.	Aide : Initialisation
	12:24 L'élève 4 lance à nouveau le programme du personnage papillon.	Erreur : Le personnage papillon se déplace toujours trop vers le bas et il dépasse l'endroit demandé dans la scène. Endroit demandé : sur la pierre.
	12:28	Aide:

	La chercheuse intervient en lui posant une question pour l'aider à vérifier sa programmation. « Est-ce que le personnage « papillon » arrive au bon endroit dans la scène ? »	Identification de l'erreur.
	12:29 L'élève 4 remet le personnage « papillon » en position initiale avec son doigt.	Position initiale du personnage « papillon » erronée.
	12:31 L'élève 4 change le paramètre de la commande « déplacement en bas » et au lieu du chiffre 2 il choisit le numéro 1.	
	12:34 La chercheuse intervient pour l'aider à initialiser la position de son personnage.	Aide : Initialisation
	12:35 L'élève 4 lance le programme du personnage « papillon » en appuyant sur celui-ci avec son doigt. La chercheuse lui pose la question suivante : « Est-ce que c'est bon ? » 12:40	Erreur : Le personnage « papillon » dépasse un peu l'endroit demandé dans la scène. Aide : Vérification de la programmation

	L'élève 4 répond : « oui ». 12:41 La chercheuse lui demande « Pourquoi ? »	
	12:46 L'élève 4 répond : « Non. Il faut qu'il aille plus loin. »	
	12 :50 L'élève 4 change encore le paramètre de la commande « déplacement à droite et il choisit au lieu du chiffre 6 le chiffre 7.	Cette correction ne démontre pas une réflexion préalable sur le résultat obtenu après l'exécution de ce programme. Cette correction empire l'erreur existante, car il augmente la distance du personnage par l'endroit demandé dans la scène.
	12:56 L'élève 4 lance le programme du personnage « papillon ».	Erreur : Le personnage dépasse trop l'endroit demandé car l'élève n'a pas initialisé sa position avant de lancer son programme.
	13:00 L'élève 4 met le personnage « papillon » en position initiale avec son doigt.	L'élève initialise correctement la position du personnage « papillon ».

	13:02 L'élève 4 exécute à nouveau son programme avec son doigt.	Erreur : Le personnage « papillon » dépasse toujours l'endroit demandé dans la scène, car il se déplace trop à droite.
	13:07 L'élève 4 prend le personnage « papillon » avec son doigt et le pose sur le bon endroit dans la scène.	
	13:10 L'élève 4 change le paramètre de la commande « déplacement à droite » et au lieu du chiffre 7, il choisit le chiffre 6 comme son paramètre.	Cette correction ne démontre pas une réflexion préalable sur le résultat obtenu après l'exécution de ce programme.
	13:15 L'élève 4 change le paramètre de la commande « déplacement en bas » et au lieu du chiffre 1 il choisit le chiffre 0 comme son paramètre.	Cette correction pose problème, car elle augmente la distance du personnage de l'endroit demandé dans la scène.
	13:26 L'élève 4 initialise la position du personnage « papillon » avec son doigt.	L'élève 4 initialise correctement la position du personnage

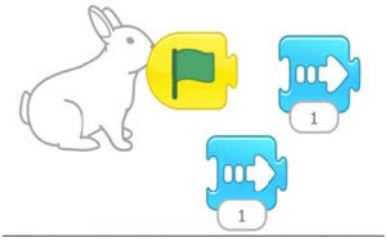
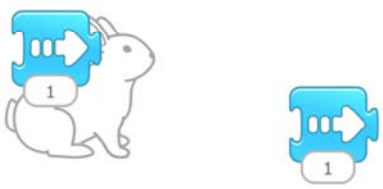
		« papillon ».
	13:29 L'élève 4 lance le programme du personnage « papillon ».	Erreur : Le personnage « papillon » dépasse l'endroit demandé, parce qu'il se déplace trop vers la droite. Erreur : Le personnage « papillon » dépasse l'endroit demandé, car il se déplace pas vers le bas. Endroit demandé : Sur la pierre.
 	13:32 L'élève 4 change le paramètre de la commande « déplacement en bas » et au lieu du 0 il choisit le chiffre 1.	
	13:37 L'élève 4 met le personnage « papillon » en position initiale avec son doigt.	La position du personnage « papillon » est correcte.
	13:38 L'élève 4 exécute le programme du personnage « papillon » en appuyant sur celui-ci avec son doigt.	Erreur : Le personnage « papillon » dépasse l'endroit demandé dans la scène car il se déplace trop vers la droite.
	13:43 L'élève 4 met le personnage	L'élève initialise correctement la

	« papillon » en position initiale avec son doigt.	position du personnage « papillon ».
	13.46 L'élève 4 change le paramètre de la commande « déplacement à droite » et au lieu du numéro 6 il choisit le numéro 5 comme paramètre.	Correction qui démontre une réflexion préalable sur le résultat obtenu après l'exécution de ce programme.
	13:51 L'élève 4 met le personnage « papillon » en position initiale avec son doigt.	La position du personnage « papillon » est correcte.
	13:57 L'élève 4 lance le programme du personnage « papillon ».	Le personnage « papillon » arrive au bon endroit dans la scène.
	13:59 L'élève 4 dit : « C'est bon ! »	
	14:00 La chercheuse lui pose la question suivante « C'est bon maintenant ? ».	Aide : Vérification de programmation
	14:03 L'élève 4 ajoute la commande de « fin » au programme du « papillon ».	
	14:07 L'élève 4 revient à l'espace de programmation du personnage « lapin ».	Aide : Lancement projet
	14:26	

	Ici l'élève 4 dit quelque chose par rapport aux cartes, mais cela n'est pas clair parce que le son de la vidéo est de très mauvaise qualité.	
	14:30 L'élève 4 va dans la catégorie des commandes de « contrôle ».	
	14:33 L'élève 4 ajoute la commande de « répétition » et il la pose sur une commande de « déplacement à droite » de celles qui existaient déjà sur l'espace de programmation.	Les corrections qui suivent ne démontrent pas une réflexion préalable sur le résultat obtenu après l'exécution du programme du personnage « lapin ».
	14:37 L'élève 4 choisit les deux autres commandes de « déplacement » et les rassemble avec les commandes à droite.	
	14:40 L'élève 4 sépare la commande de « fin » et il la pose à droite.	
	14:43 L'élève 4 rassemble les commandes de déplacement avec le programme à gauche.	
	14:45 L'élève 4 rassemble aussi la « fin » avec les autres commandes en créant un programme.	

	14:47 L'élève 4 met le personnage lapin au début avec son doigt.	La position du personnage « lapin » est correcte.
	14:50 L'élève 4 lance le programme du personnage « lapin » et celui du personnage « papillon » en même temps, en utilisant le bouton du « drapeau-vert » en haut de la scène.	Erreur :Le programme du personnage « lapin » est erroné, car il réalise 9 pas à droite et il dit trois fois « Bonjour le papillon », au lieu de réaliser un pas à droite et dire « Bonjour le papillon » et répéter ce motif des commandes trois fois. Erreur : La commande de « répétition » est utilisée de façon erronée. Erreur : Enregistrement erroné, car trois fois « Bonjour le papillon » dans la même commande de « son ».
	14:52 L'élève 4 met le personnage « lapin » en position initiale avec son doigt.	La position du personnage « lapin » est correcte.
	14:53-15:00	

	L'élève 4 regarde sa tablette sans réaliser d'action.	
	15:00 La chercheuse lui pose la question suivante : « C'est bon ? »	Aide : Vérification de programmation
	15:02 L'élève 4 répond : « Non ».	
	15:03 La chercheuse lui demande : « Pourquoi ? »	Aide : Identification de l'erreur
	15:04 L'élève 4 répond : « Parce qu'il n'avance pas. »	
	15:06 La chercheuse lui dit : « Je crois qu'il avance même trop »	Aide : Identification de l'erreur
	15:12 L'élève 4 change le paramètre de la commande de « répétition » et au lieu du paramètre par défaut il utilise le chiffre 2.	
	15:16 La chercheuse intervient en lui disant : « Fais attention au problème » et elle lui rappelle la consigne de celui-ci.	Aide: Consigne
	15:37 L'élève 4 ouvre le programme du personnage « lapin ».	

	15:39 L'élève 4 enlève la commande de « déplacement d'un pas à droite » de la commande de la « répétition » et il la pose à gauche dans l'espace de programmation du personnage « lapin ».	
	15:42 L'élève 4 efface une grande partie du programme du personnage « lapin ».	
	15:43 L'élève 4 prend une commande de « déplacement d'un pas à droite » et il la pose en haut dans l'espace de programmation.	
	15:51 L'élève 4 va dans la catégorie des commandes de « démarrage ».	
	15:53 L'élève 4 efface aussi la commande « quand drapeau-vert touché » par l'espace de programmation du « lapin ».	
	15:55	

	L'élève 4 va dans la catégorie des commandes de « déplacement ».	
	15:57 L'élève 4 efface la commande « déplacement d'un pas à droite ».	
	15:59 L'élève 4 efface la commande « déplacement d'un pas à droite ».	L'élève 4 a effacé tout le programme du personnage « lapin ».
	16:02 L'élève 4 va dans la catégorie des commandes de « son ».	
	16:04 L'élève 4 ajoute la commande de « son » déjà enregistrée dans l'espace de programmation du personnage « lapin ».	
	16:06 L'élève 4 va dans la catégorie des commandes de « déplacement ».	
	16:07 L'élève 4 ajoute la commande « déplacement d'un pas à droite » dans l'espace de programmation et la rassemble avec la commande de « son ».	C'est la première fois que le motif des commandes est correct, mais la commande de « son » reste erronée.
	16:10 L'élève 4 va dans la catégorie des commandes de « son ».	

	16:11 L'élève 4 ajoute encore une commande de « son » dans l'espace de programmation du personnage « lapin » et la pose à droite des autres commandes.	
	16:13 L'élève 4 va dans la catégorie des commandes de « déplacement ».	
	16:14 L'élève 4 ajoute une commande de « déplacement » et la rassemble avec le motif de commandes déjà créé.	
	16:16 L'élève 4 ajoute à ces commandes la commande de « son ».	
	16:18 L'élève 4 ajoute encore une commande de « déplacement d'un pas à droite » et la rassemble avec les autres.	
	16:21 L'élève 4 va dans la catégorie des commandes de « son ».	
	16:23 L'élève 4 ajoute encore une commande de « son » dans l'espace de programmation du personnage « lapin » et la rassemble au programme existant là-bas.	
	16.26 L'élève 4 met le personnage « papillon » en position initiale avec son doigt.	La position initiale du personnage « papillon » est erronée.

	16:31 L'élève 4 va dans la catégorie des commandes de « fin »	
	16:33 L'élève 4 ajoute la commande de « fin », à la fin du programme.	
	16:35 L'élève 4 va dans la catégorie des commandes de « démarrage».	
	16:37 L'élève 4 ajoute la commande « quand drapeau-vert cliqué » au début du programme du « lapin ».	
	16:39 L'élève 4 initialise la position du personnage « papillon » avec son doigt.	La position initiale du personnage « papillon » est erronée.
	16:43 L'élève 4 lance le programme du personnage « lapin » en appuyant sur celui-ci.	Erreur : Le programme du personnage « lapin » est erroné, car le son enregistré par l'élève est erroné. En effet, l'élève a enregistré trois fois « Bonjour le papillon» dans la même commande de « son ».
	17:04 La chercheuse lui pose la question suivante : « Combien de fois il parle le lapin dans la	Aide : Identification de l'erreur

	vidéo ? ».	
	17:07 L'élève 4 répond : « Trois ».	
	17:10 La chercheuse lui pose la question suivante : « Combien des fois tu l'as entendu parler sur ton programme ? »	Aide : Identification de l'erreur
	17:14 L'élève 4 répond : « Quatre ? »	
	17:17 La chercheuse lui pose la question suivante : « Qu'est-ce qu'il faut faire alors ? »	Aide :Identification de l'erreur
	17:20 L'élève 4 répond : « Il faut mettre le micro ? »	
	17:23 La chercheuse intervient pour l'aider en lançant le projet modèle.	Aide : Lancement du projet modèle.
	17:23 L'élève 4 sépare son projet en deux parties.	Moment de blocage Ouverture du programme après son exécution et déplacement des commandes existantes dans celui-ci à droite et à gauche,

		sans effectuer autre action de correction. Les actions de correction qui suivent ne démontrent pas une réflexion préalable sur le résultat obtenu après l'exécution de ce programme.
	17:37 La chercheuse lance le projet modèle et elle l'invite à le regarder. Ensuite, elle lui pose la question suivante : « Combien de fois le « lapin » dit « Bonjour le papillon » dans ton programme ? Qu'est-ce qu'il faut faire alors ? »	Aide : Identification de l'erreur
	17:38 L'élève 4 prend les commandes de « son » et de « fin », il les sépare des autres commandes et il les laisse à droite dans l'espace de programmation.	
	17:42 L'élève 4 prend la commande de « déplacement » et il la rassemble avec les deux commandes en bas.	

	17 : 47 L'élève 4 prend le motif des commandes de « déplacement » et de « son » et il les pose en bas dans l'espace de programmation, à côté des autres commandes.	
	17:48 L'élève 4 prend la commande de « son » et il la pose en haut dans l'espace de programmation.	
	17:49 L'élève 4 prend la commande de « déplacement » qui se trouvait toute seule à gauche et il la rassemble au programme qui se trouve au milieu de l'espace de programmation.	
	17:51 L'élève 4 pose les commandes de « son » et celle de « fin » un peu plus en haut et à droite dans l'espace de programmation du personnage « lapin ».	Moment de blocage terminé.
	17:54 L'élève 4 met le personnage « lapin » en position initiale avec son doigt.	La position initiale du personnage « lapin » est correcte.
	17 :55 L'élève 4 lance le programme du « papillon » en appuyant sur le bouton du « drapeau-vert » en haut de la scène. Le programme du personnage « lapin » n'est pas lancé, car il n'y a pas des commandes connectées à la commande de « quand drapeau-vert cliqué » dans l'espace de programmation.	Le programme du personnage « papillon » est correct. Erreur : Aucune commande à part la commande « quand drapeau-vert cliqué ».

	18:01 L'élève 4 rassemble la commande de « fin » avec les autres commandes.	
	18:04 L'élève 4 lance sans le faire exprès la commande de « son » en s'appuyant sur celle-ci mais ensuite il re-appuie sur celle-ci et il l'arrête.	
	18:05 L'élève 4 ajoute la commande « quand drapeau-vert cliqué » au début du programme.	Les actions de correction qui suivent ne correspondent pas au problème relevé.
	18:06 L'élève 4 lance le nouveau programme du personnage « lapin » en s'appuyant sur celui-ci avec son doigt.	Erreur : Le « lapin » réalise un pas à droite, puis il dit trois fois « Bonjour le papillon », puis il se déplace deux pas à droite et il s'arrête, au lieu de réaliser un pas à droite, puis dire « Bonjour le papillon » et répéter ce motif des commandes seulement trois fois.

Note pour cet élève

Nous avons décidé de ne pas transcrire les actions de l'élève dans le reste de la vidéo parce que son comportement ne change pas et nous avons déjà transcrit une partie assez importante de ce qu'il fait. Ainsi, nous avons assez d'informations pour caractériser son activité de programmation. Toutefois, il est important de noter que dans le reste de la vidéo l'élève 4 n'est toujours pas capable de corriger son erreur et il n'arrive pas à identifier le motif des commandes qui se répète. En se trompant sur les boutons du menu d'enregistrement d'un son il a réussi sans le faire exprès à enregistrer correctement le son demandé. Puisqu'il avait dépassé le temps disponible, nous sommes intervenue à la fin en lui indiquant l'erreur qu'il devrait corriger. Nous considérons donc que cet élève il a reçu beaucoup d'aides pendant le PB2, mais il n'a pas réussi à le résoudre.

Annexe 11 : Le scénario pédagogique conçu pour l'étude principale en France

SCENARIO PEDAGOGIQUE DE FAMILIARISATION ET D'APPRENTISSAGE DE LA PROGRAMMATION À TRAVERS LE LOGICIEL DE PROGRAMMATION VISUEL SCRATCH JUNIOR

1. OBJECTIF PEDAGOGIQUE

Ce scénario pédagogique concerne les technologies de l'information et de la communication et la didactique de l'Informatique en école élémentaire. Son objectif est l'initiation et l'apprentissage de la programmation par les jeunes élèves, à travers l'environnement de ScratchJr (http://www.scratchjr.org/).

ScratchJr est un environnement de programmation audiovisuel qui a été créé par le laboratoire Media du MIT, l'université de Tuffts et PICO, afin d'initier les élèves aux notions de programmation, de la grande section de l'école maternelle jusqu'au CE1. Le code et l'interface ont été simplifiés pour qu'ils puissent correspondre au développement cognitif, social et affectif des jeunes élèves.

Niveau de classe : Cours Préparatoire

Age : 5 à 6 ans

Logiciel utilisé : ScratchJr

Equipement : Tablettes avec l'application ScratchJr téléchargée.

2. APPROCHES DIDACTIQUES – THÉORIQUES – MÉTHODOLOGIQUES

Tout au long de la durée de ce scénario pédagogique, nous adopterons une stratégie didactique inspirée du constructivisme, dont la philosophie suit le langage de programmation ScratchJr. Les élèves se trouvent au centre du processus de l'apprentissage afin de construire, pas à pas, des nouvelles connaissances.

3. MODALITÉS D'ENREGISTREMENT DES PROJETS DES ÉLÈVES

À la fin de chaque activité, l'enseignante demandera aux élèves d'enregistrer leur projet d'une telle manière : AvPré, c'est-à-dire Activité no v et le Prénom de l'élève.

4. LE VOCABULAIRE DE L'ENVIRONNEMENT SCRATCHJR

Application : l'environnement de programmation ScratchJr.

Commandes : Les blocs colorés, qui peuvent constituer un programme.

Programme : Une série des commandes avec un début et une fin rouge.

Projet : Les programmes enregistrés dans un fichier.

Palette des commandes : Cette dernière comprend six catégories des commandes différentes. Chaque catégorie comprend des commandes de programmation :

- Catégorie de démarrage (jaune)
- Catégorie de déplacement (bleu)
- Catégorie d'apparence (violet)
- Catégorie du son (vert)
- Catégorie du contrôle (orange)
- Catégorie de la fin (rouge)

Déplacement :

- Horizontal (vers la gauche, vers la droite)
- Vertical (En haut, En bas)

Enregistrement : enregistrement du son pour faire parler les personnages.

Espace de programmation : L'utilisateur doit cliquer et glisser les commandes vers l'espace de programmation et les connecter afin de programmer le personnage à réaliser une action dans la scène.

Personnages : Les héros d'un projet.

Arrière-Plan : Permet de sélectionner ou créer un arrière-plan.

Scène : C'est l'espace ou l'action se déroule dans le projet.

Mode de présentation : Agrandit l'écran.

Drapeau vert :

- En s'appuyant sur le drapeau vert d'un script, on déclenche ce script.
- En s'appuyant sur le drapeau vert en haut et à droite, on déclenche tous les scripts, qui ont le drapeau vert au début.

Drapeau vert : Permet de démarrer les programmes qui ont le drapeau vert au début.

Fin de programmation : Cette commande signale la fin du script.

Message envoyé : Envoie un message d'une couleur spécifique.

Message reçu : Lance le script chaque fois qu'un message de la couleur spécifiée est envoyé.

Répéter : Exécute les blocs qui se trouvent à son intérieur un certain nombre de fois.

Initialisation : Replace la scène au mode initial.

Enregistrement d'un projet : Le projet en cours s'enregistre automatiquement.

5. SEANCES DU SCÉNARIO PÉDAGOGIQUE

Le scénario pédagogique comprend trois catégories de séances, les séances de l'initiation au projet (catégorie A), les séances de l'enseignement du contenu et de validation (catégorie B) et les séances de l'évaluation du projet (catégorie C).

Dans un premier temps, nous allons réaliser deux **séances de préparation cognitive et psychologique** (détection des idées initiales et familiarisation avec l'utilisation de la tablette) au cours desquelles les élèves seront informés concernant le projet, développeront des motivations pour la suite et auront la possibilité de passer une période de découverte du logiciel ScratchJr. Nous allons également réaliser un entretien semi-directif avec chacun des élèves, pour détecter leurs idées initiales autour les commandes à étudier et l'interface du logiciel.

Dans un deuxième temps, nous allons réaliser neuf **séances d'enseignement et de validation**, au cours desquelles les élèves vont travailler en groupes de quatre, où chaque élève aura sa propre tablette à la main. Les séances d'enseignement et de validation pourront être regroupées en deux cycles. Le premier cycle comprend les concepts de programmation de niveau-bas de difficulté, qui concerne l'interface d'utilisateur, les commandes de déplacement et l'enregistrement du son. Au cours de chaque séance, il y aura un temps prévu pour l'enseignement du concept ou du processus et un temps prévu pour que les élèves se mettent en pratique en utilisant cette commande ou ce processus, afin de résoudre un problème donné. À la fin du premier cycle nous allons mener une séance récapitulative avec un problème à résoudre, pour faire une répétition de tous les commandes et processus vus jusqu'à ce moment. Le deuxième cycle comprend les concepts de programmation de haut-niveau, qui sont les concepts et les processus les plus difficiles à s'approprier par les élèves, comme celui de messages ou de la répétition.

Enfin, nous allons réaliser deux **séances d'évaluation de la performance des élèves**. La première sera dédiée à la résolution de deux problèmes d'évaluation par tous les élèves. Les problèmes seront résolus individuellement par chaque élève. La deuxième séance sera dédiée à la mise en place des entretiens semi-directifs individuels avec les élèves, pour détecter leurs idées à propos de l'interface et des commandes étudiées.

A. ACTIVITES DE PREPARATION COGNITIVE ET PSYCHOLOGIQUE

1re séance : Détection des idées initiales des élèves sur les commandes étudiées

Durée : 10 min par élève.

2ème séance : Contextualisation et développement de motivation

Durée : 30 min.

Connaissances préalables : Savoir la gauche et la droite.

Concepts : déplacement dans l'espace, nombre de pas, programmation, commande, latéralisation/direction/orientation, début et fin de la programmation, algorithme.

Objectifs :

- Initiation des élèves à la programmation à travers d'un jeu de déplacement dans l'espace.

- Se familiariser avec la conception orale d'un algorithme de pas (rôle de programmeur), afin de diriger leurs camarades (rôle de la machine) pour se déplacer dans un quadrillage créé sur le sol.
- S'exercer avec l'exécution d'un algorithme de pas oral (avancer, reculer, tourner à droite, tourner à gauche).
- Définir le début et l'arrivée de leur programmation dans l'espace.
- Comprendre le rôle de programmeur dans le contexte du jeu et transférer cette connaissance, afin de pouvoir comprendre la programmation informatique.

Matériel : scotch, carte vert et carte rouge.

Organisation de la classe : Cette activité se déroulera avec l'ensemble de la classe. Les élèves vont travailler en binômes. L'un élève sera le programmeur et l'autre la machine. Tous les binômes vont jouer le jeu, l'un après l'autre.

Description :

I. Tout d'abord, nous allons commencer la séance par expliquer aux élèves les règles du jeu.

II. Les élèves vont travailler en binômes et ils vont jouer un jeu de déplacement dans l'espace. Plus précisément, l'un élève représentera le programmeur et l'autre, la machine. Le programmeur doit programmer correctement sa machine, en construisant oralement un algorithme de pas, pour que sa machine se déplace d'un début choisi à une fin choisie. Pour faciliter le déplacement des élèves dans l'espace, nous allons créer un quadrillage sur le sol avec du scotch.

III. Ensuite, nous allons donner au programmeur deux cartes, une verte et une rouge, qui signaleront le début et l'arrivée de la programmation de la machine. Le programmeur choisira d'abord le début et l'arrivée de sa machine sur le quadrillage du sol et ensuite il lui donnera des commandes pour arriver à l'endroit choisi.

IV. Lorsque les binômes terminent une fois le jeu et s'il reste du temps, ils jouent une deuxième fois, pour que tous les élèves puissent passer par les deux rôles.

V. Au début et jusqu'à que les élèves arrivent à se familiariser avec le jeu, chaque fois un binôme jouera sur le quadrillage. Ensuite, nous pourrons travailler avec deux binômes

différents en même temps et demander aux programmeurs de programmer leurs machines afin d'éviter des collisions.

VI. Lorsque le jeu sera terminé, nous commencerons une discussion avec les élèves, concernant le jeu que nous avons fini. L'objectif de cette discussion est l'initiation des élèves à l'activité de la programmation, à travers une série de questions bien choisies, en utilisant leurs expériences vécues au cours du jeu. Les questions qui suivent sont des aides didactiques.

VII. Les questions proposées sont les suivantes :

- Vous venez de programmer vos machines. Quoi d'autre pourriez-vous programmer ?
- Peut-être la tablette ?
- Comment vous pourriez la programmer ?
- Si je laisse ma tablette ici, elle fera quelque chose ou pas ? (Lien avec la machine qui fait rien sans le programmeur)
- Qu'est-ce qu'on doit faire pour que notre tablette fasse quelque chose ?
- On peut commencer par l'allumer ?
- Et là ? Qu'est-ce qu'on doit faire maintenant ? Comment on va la programmer ? Si on lui parle, comme à notre machine ça va faire quelque chose ? Comment on peut lui dire de faire quelque chose ?
- Si on touche par exemple sur un de ces icônes ?
- Maintenant qu'on l'a touché qu'est-ce que vous en pensez ? Est-ce qu'on lui a donné des commandes ?
- La réponse est non ! Parce que quelqu'un d'autre avant nous, un programmeur a programmé cette machine et ces applications et nous les utilisons tout simplement !
- Au cours de ce projet-tablette, nous allons devenir programmeurs comme la personne qui a programmé nos tablettes !
- Comment on va faire ça ?
- Les tablettes comme toutes les machines comprennent un seul langage. Comme vous vous comprenez le français par exemple. C'est pourquoi nous allons apprendre leur langue pour leur parler et dans ce cas-là leur écrire car les machines ne parlent pas. Ce ne sont pas des humains ! La langue que les machines comprennent est un peu difficile pour nous maintenant. C'est pourquoi nous allons apprendre un autre langage, beaucoup plus simple, pour commencer. Nous allons apprendre comment créer nos propres dessins animés !

VIII. Par la suite nous allons réaliser la séance de familiarisation avec la tablette.

3ème séance : Familiarisation avec l'interface de ScratchJr

Durée : 15 min.

- **Concepts :** tablette, application, écran, programme, icônes, verrouillage/déverrouillage de la tablette.
- **Connaissances et processus :** manipulation de l'interface, motricité fine (cliquer et glisser), synchronisation vision et déplacement.

Objectifs :

- Expérimenter avec la manipulation de la tablette.
- Familiarisation avec le processus du verrouillage et du déverrouillage d'une tablette.
- Apprendre à déclencher une application.
- Se familiariser avec la motricité fine.

Matériel : Tablettes avec l'application ScratchJr téléchargée.

Organisation de la classe : Chaque élève travaille seul sur sa tablette.

Description :

I. Tout d'abord, nous allons expliquer aux élèves les règles d'utilisation de tablette.

II. Après avoir donné les tablettes aux élèves, nous allons leur poser une série de questions pour découvrir les différentes utilisations que les élèves font de la tablette et on va leur expliquer qu'est-ce que c'est une tablette.

   - Qu'est-ce que c'est une tablette ? À quoi cela ressemble-t-il ? Avez-vous utilisé quelque chose de similaire ? (On peut faire référence à l'ordinateur)

III. Ensuite, nous allons demander aux élèves de déverrouiller la tablette. Si les élèves ne savent pas le faire, nous allons leur montrer comment le faire.

IV. Après, nous allons leur parler des applications disponibles sur la tablette. On va leur expliquer qu'il s'agit de programmes différents déjà prêts à utiliser.

V. Par la suite, nous allons présenter l'application de ScratchJr et nous allons leur demander de l'ouvrir. Nous allons signaler aux élèves qu'on utilise que nos doigts et que la tablette doit être toujours sur la table, pour travailler.

VI. Ensuite, nous allons leur expliquer qu'à l'aide de cette application nous allons devenir programmeurs et nous allons apprendre à créer nos propres histoires et jeux interactifs

VII. Après, nous allons arriver à l'espace d'utilisateur (la maison) et on va leur demander d'aller chercher le « chat ». Pour le faire, il faut qu'ils appuient sur « le plus ».

VIII. Ensuite, on leur posera les questions suivantes, pour les motiver à découvrir qu'est-ce qu'on peut faire avec le personnage « chat ».

- Qu'est-ce que pourriez-vous observer ici ?
- Qu'est-ce qu'on peut faire avec le « chat » ?
- Comment peut-on faire bouger le « chat » ?

IX. Si les élèves n'arrivent pas à découvrir la fonctionnalité des commandes de déplacement, nous allons leur présenter leur fonctionnalité sur notre propre tablette, dont l'écran sera projeté sur le TNI de la classe.

X. Si un élève découvre un processus, nous lui demanderons de partager sa découverte avec toute la classe.

XI. Par la suite, nous allons leur demander de sortir de l'application et revenir aux applications de début, en utilisant le bouton principal.

XII. Après, on leur demandera de découvrir comment on verrouille la tablette. S'ils n'arrivent pas à découvrir le bouton, on va leur le montrer.

XIII. Avant de finir, on leur demande de re-ouvrir l'application pour voir que nous sommes où on s'est arrêté tout-à-l 'heure.

XIV. Enfin, on leur demande de verrouiller la tablette pour finir cette séance de familiarisation.

B. ACTIVITÉS DE L'ENSEIGNEMENT DU CONTENU ET DE VALIDATION

1ère séance : Familiarisation avec l'interface et le langage de programmation ScratchJr

Durée : 45 min.

- **Concepts :** application, programme, espace d'utilisateur, démarrer un nouveau projet, commande, séquence, introduction des commandes à l'espace de programmation, commandes pour déplacer le personnage à droite et à gauche, connexion des commandes dans l'espace de programmation, effacer des commandes par l'espace de programmation, début et fin de la programmation, enregistrement d'un projet.
- **Connaissances :** application, programme, espace d'utilisateur, démarrer un nouveau projet, commande, commandes pour déplacer le personnage à gauche et à droite, début et fin de la programmation.
- **Processus :** démarrer un nouveau projet, introduction des commandes à l'espace de programmation, connexion des commandes dans l'espace de programmation, effacement des commandes de l'espace de programmation, enregistrement d'un projet.

Objectifs :

- Se familiariser avec le processus pour démarrer un nouveau projet.
- Se familiariser avec le processus pour introduire un nouveau arrière-plan.
- Se familiariser avec le processus d'introduction, d'effacement et de connexion des commandes dans l'espace de programmation.
- S'approprier le processus de programmation pour déplacer le personnage à droite et à gauche dans la scène.
- Se familiariser avec le début et la fin de la programmation du personnage et apprendre à créer un programme complet.
- Se familiariser avec la séquence sur ScratchJr, c'est-à-dire, le fait que les commandes s'exécutent d'une manière séquentielle, l'une après l'autre.
- Enregistrement d'un projet.

Matériel: Tablettes avec l'application ScratchJr téléchargée.

Organisation de la classe : La séance sera réalisée avec l'ensemble de la classe. Chaque élève a sa propre tablette. La tablette de l'enseignante sera connectée avec le TNI de la classe, pour que tous les élèves puissent voir ce qui se passe sur la tablette de l'enseignante. Les élèves vont être assis par quatre.

Description :

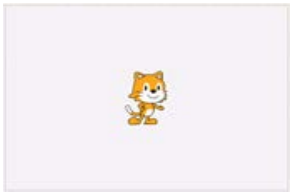
I. Tout d'abord, nous allons demander aux élèves d'allumer la tablette et démarrer l'application ScratchJr. Si les élèves ont des difficultés, l'enseignante apportera de l'aide en présentant le processus aux élèves.

II. Par la suite, l'enseignante demandera aux élèves d'entrer à la maison. Dans la maison l'enseignante leur demandera de **démarrer un nouveau projet** en s'appuyant sur le « **plus** ».

> Pour commencer un nouveau projet, on doit appuyer sur le « plus ». Chaque projet qu'on crée sera une petite histoire avec le personnage « chat » ou d'autres personnages. La prochaine fois je vous montrerai comment projeter notre histoire en plein écran, pour la voir comme un dessin-animé.

III. Ensuite, l'enseignante demandera aux élèves de lui rappeler les différentes façons qu'ils avaient découvertes la dernière fois, **pour faire bouger le « chat »** dans cet espace blanc, qui s'appelle **scène**. Si les élèves ne se souviennent pas, l'enseignante apportera de l'aide en présentant le processus pour introduire des commandes dans l'espace de programmation, connecter des commandes et effacer des commandes de l'espace de programmation.

> Comment peut-on faire bouger le personnage « chat » dans la scène ?



IV. Par la suite, l'enseignante expliquera aux élèves qu'ils vont créer leur première histoire avec le personnage « chat » aujourd'hui. Pour commencer, il est nécessaire de **modifier l'arrière-plan** existant. Si les élèves n'arrivent pas à trouver l'icône pour modifier l'arrière-plan, l'enseignante leur la présentera.

> Comment peut-on modifier l'arrière-plan existant?

V. L'enseignante présentera ensuite aux élèves la banque des arrières-plans disponibles sur le logiciel et elle leur demandera de choisir celui qu'elle a sur sa propre tablette. Ensuite, elle leur demandera de trouver comment l'insérer dans la scène. Si les élèves n'arrivent pas à l'insérer, l'enseignante leur présentera les processus pour le faire.

> **Pour insérer un arrière-plan** dans la scène il y a deux façons différentes. On peut soit cliquer une fois sur l'arrière-plan demandé et après cliquer sur l'icône en haut et à droite pour valider notre choix, soit cliquer deux fois sur le même arrière-plan.

VI. L'enseignante demandera aux élèves d'observer la scène, rappeler du jeu qu'ils ont joué la dernière fois avec les cartes et réfléchir sur l'objectif de ce problème de programmation. Ensuite, elle leur expliquera le problème et elle leur laisse s'expérimenter pour trouver la solution.

> Tout d'abord, on doit cliquer sur le « chat » avec notre doigt et le glisser dans le cercle vert. Ensuite, on doit programmer le « chat » pour qu'il arrive dans le cercle rouge. On doit lui donner les bonnes commandes, pour qu'il se déplace et il arrive dans le cercle rouge. Dès qu'il arrive dans le cercle rouge, le « chat » devra s'arrêter. Comment peut-on faire ça ?

VII. Ensuite, l'enseignante leur rappellera que chaque fois qu'on commence à programmer un personnage, on doit bien définir son début. **Pour définir le début de la programmation d'**un personnage, on utilise notre doigt pour placer le personnage où on veut. Ensuite, l'enseignante expliquera aux élèves que toutes ces pièces de puzzle s'appellent des **commandes**, parce qu'elles obligent le personnage « chat » à réaliser une action dans la scène.

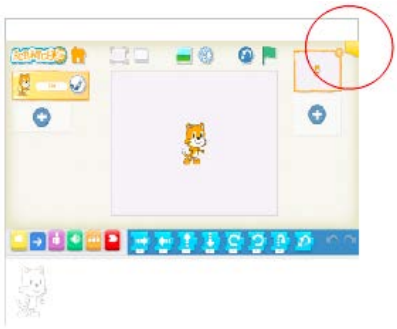
VIII. **Ils existent deux solutions possibles pour résoudre ce problème**. La première solution est qu'ils utilisent plusieurs fois la commande pour déplacer le personnage à droite. La deuxième solution est qu'ils utilisent une commande pour déplacer le personnage à droite et ils modifient son paramètre. Pour trouver le nombre de pas dont le personnage a besoin pour arriver au centre du cercle rouge, ils doivent faire des essais. Par contre, chaque fois qu'ils font un essai, il est nécessaire qu'ils fassent revenir le personnage au début, pour valider s'ils ont bien programmé le personnage ou pas. Pour **initialiser la position du personnage** il y a deux façons à le faire, soit en utilisant le doigt, soit en utilisant l'icône d'initialisation.

IX. Ensuite, si les élèves n'ont pas encore utilisé le drapeau-vert et la fin rouge, l'enseignante leur rappellera le jeu de la semaine dernière avec les cartes et elle leur demandera de trouver une façon pour signaler au personnage le début et la fin de programmation. Si quelqu'un trouve la solution, il pourra la présenter à la classe. Sinon, l'enseignante présentera aux élèves les deux commandes et elle leur expliquera leur fonctionnalité. Un programme avec un début et une fin sera nommé **programme complet**.

X. Par la suite, l'enseignante demandera aux élèves de démarrer leur programme pour vérifier s'ils ont bien programmé le personnage, pour résoudre le problème donné. En plus, elle leur demandera de faire attention à l'exécution du programme du personnage, pour

observer le fait que les commandes qui sont en train d'être exécutées clignotent l'une après l'autre, parce qu'elles s'exécutent de manière séquentielle (**séquence**).

XI. Après, l'enseignante leur présentera **le processus d'enregistrement de leur projet.** Elle leur expliquera qu'il est nécessaire de cliquer sur le coin jaune en haut et à droite. Ensuite, il faut effacer ce qui est déjà écrit et écrire à l'aide du clavier tactile « A1prénom ».



XII. Enfin, l'enseignante leur demandera d'aller à la maison pour vérifier si leur projet a été bien enregistré ou pas.

2ème séance : Programmation de deux personnages, projection d'un projet en plein-écran, enregistrement du son, déplacer un personnage en haut et en bas, résolution d'un problème

Durée : 45 minutes.

- **Concepts :** personnages, exécution d'un projet en plein écran, commandes de déplacement en haut et en bas, enregistrement du son.
- **Connaissances** : personnage, icône pour projeter le projet en plein-écran, commandes de déplacement en haut et en bas.
- **Processus :** démarrer un nouveau projet, introduction des commandes dans l'espace de programmation, connexion des commandes dans l'espace de programmation, effacer des commandes de l'espace de programmation, enregistrement du projet.

Objectifs :

- Se familiariser avec les commandes pour faire **déplacer le personnage en haut et en bas dans la scène.**
- Se familiariser avec le processus pour **introduire un nouveau personnage** dans la scène.
- Se familiariser avec le processus d'**enregistrement d'un son** et son introduction dans le programme d'un personnage.
- Comprendre la **fonctionnalité de l'icône du drapeau vert.**
- Connaître **l'icône pour projeter notre projet en plein-écra**n et se familiariser avec le processus de projection de leur projet en plein-écran.

Matériel : tablettes avec l'application ScratchJr téléchargée.

Organisation de la classe : La séance sera réalisée avec l'ensemble de la classe. Chaque élève a sa propre tablette. La tablette de l'enseignante sera connectée avec le TNI de la classe, pour que tous les élèves puissent voir ce qui se passe sur la tablette de l'enseignante. Les élèves vont être assis par quatre.

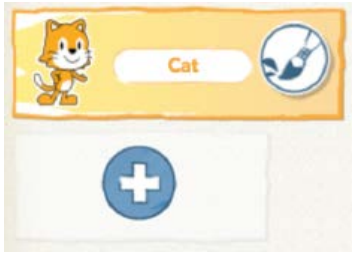
Description :

I. Tout d'abord, l'enseignante demandera aux élèves de démarrer l'application et un **nouveau projet**.

II. Ensuite, l'enseignante leur demandera d'**introduire l'arrière-plan**, «Banlieue ». Il est important de signaler que dans la scène ils existent deux points, une verte et une rouge, pour faciliter les élèves à distinguer le début et l'arrivée de chaque personnage.

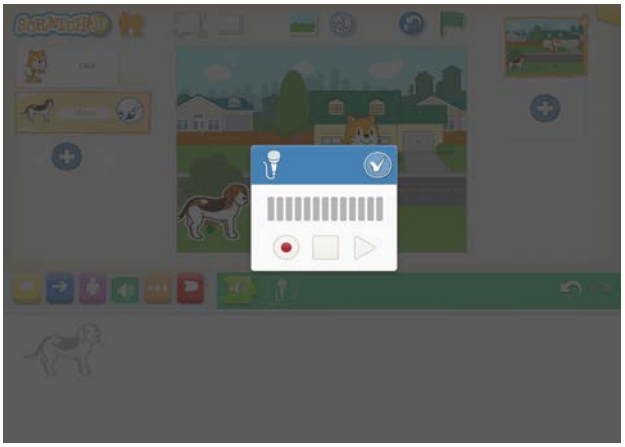
III. Ensuite, l'enseignante demandera aux élèves de **poser le personnage « chat »** devant la porte de la maison et le **programmer pour qu'il traverse la rue.**

IV. Après, l'enseignante demandera aux élèves de découvrir le processus pour **introduire un nouveau personnage** dans la scène et en particulier le personnage « chien ». Pour introduire un nouveau personnage dans la scène, il faut appuyer sur le plus qui se trouve à gauche de la scène pour arriver à la banque de personnages existants sur le logiciel. Il existe deux façons pour le faire, soit en s'appuyant sur le personnage qu'on veut et après valider notre choix, soit en s'appuyant deux fois sur le personnage choisi.



V. Par la suite, l'enseignante demandera aux élèves de **poser le personnage « chien »** sur l'herbe à gauche et **le programmer pour approcher le personnage « chat »** .

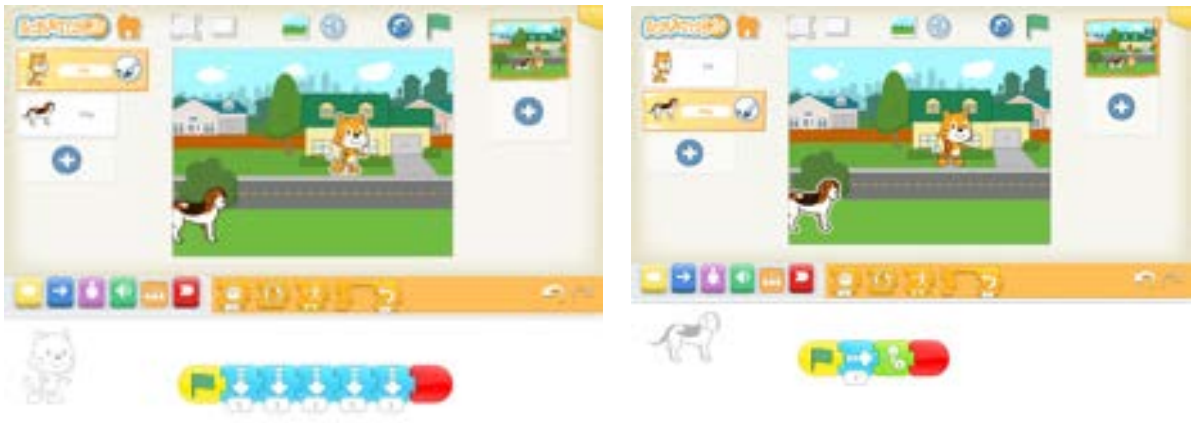
VI. Ensuite, l'enseignante leur demande de découvrir **le processus pour faire parler le personnage « chien ».** Si les élèves ont des difficultés à trouver la commande pour enregistrer un son et le processus pour le faire, l'enseignante leur les présentera. Pour enregistrer un son, il faut d'abord, appuyer sur la commande de l'enregistrement. Ensuite, il y a un carré qui apparait dans la scène. Pour enregistrer un son il faut appuyer sur le bouton rouge, parler et enfin valider.



VII. Après, l'enseignante demandera aux élèves de découvrir comment ils peuvent faire déplacer les deux personnages en même temps. Si les élèves n'arrivent pas à penser à l'icône du drapeau vert en haut de la scène, l'enseignante leur expliquera sa fonctionnalité.

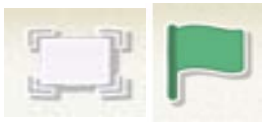
> Lorsqu'on clique sur le drapeau vert en haut de la scène, les deux personnages déclenchent leurs programmes en même temps. Essayez d'appuyer sur le drapeau vert en haut. Est-ce que ça marche ? Les programmes de vos personnages se déclenchent ou pas ?

VIII. Les personnages ne se déclencheront pas, parce qu'ils n'ont pas la commande du drapeau vert au début de leurs programmes. C'est pourquoi l'enseignante demandera aux élèves de réfléchir pour trouver s'il manque quelque chose dans leurs programmes. Si les élèves ne se souviennent pas, l'enseignante fera référence au jeu des cartes et au fait que leur machine avait un départ et une arrivée spécifique, afin de leur rappeler d'ajouter le drapeau-vert et la fin rouge sur les programmes de leurs personnages.

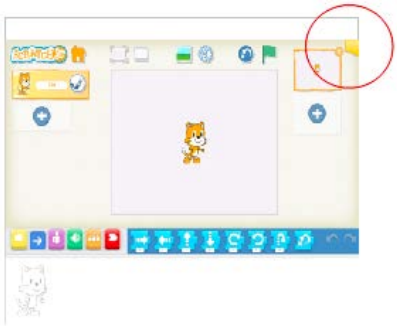


IX. IX. Ensuite, l'enseignante demandera aux élèves de ressayer l'icône du drapeau vert en haut de la scène, pour vérifier si les deux personnages déclencheront ou pas leurs programmes, après avoir ajouté le drapeau vert et la fin rouge dans leurs programmes.

X. Puis, l'enseignante demandera aux élèves de découvrir l'icône pour projeter leur histoire en plein écran, comme un dessin animé. S'ils ont des difficultés à la découvrir, l'enseignante leur la présentera. Pour voir leur projet en plein-écran, il faut appuyer sur le drapeau vert en haut de la scène.



XI. Après, l'enseignante leur demandera d'enregistrer leur projet. Elle leur expliquera qu'il est nécessaire de cliquer sur la coin jaune en haut et à droite. Ensuite, il faut effacer ce qui est déjà écrit et écrire à l'aide du clavier tactile « A2prénom ».



XII. Enfin, l'enseignante leur demandera d'aller à la maison pour vérifier si leur projet a été bien enregistré ou pas.

3ème séance : Effacer des personnages, Paramètres, Résolution d'un problème pour faire un récapitulatif de ce qu'on a appris jusqu'à cette séance

Durée : 45 minutes.

- **Concepts** : icône du drapeau vert en haut de la scène, paramètres de commandes de déplacement.
- **Connaissances** : effacer un personnage, paramètres de commandes de déplacement.
- **Processus** : effacer un personnage, modifier le paramètre d'une commande de déplacement.

Objectifs :

- **Programmer deux personnages** en utilisant les commandes de déplacement vues jusqu'à cette séance.
- Se familiariser avec **la commande et le processus pour enregistrer un son**.
- Connaître les deux processus pour **effacer un personnage** de la scène.
- Se familiariser avec **l'utilisation de paramètres**.
- Comprendre **la fonctionnalité du drapeau vert** dans la scène et en plein-écran.
- Utilisation de l'**icône pour projeter le projet en plein-écran**.

Matériel :tablettes avec l'application ScratchJr téléchargée.

Organisation de la classe : La séance sera réalisée avec l'ensemble de la classe. Chaque élève a sa propre tablette. La tablette de l'enseignante sera connectée avec le TNI de la classe, pour que tous les élèves puissent voir ce qui se passe sur la tablette de l'enseignante. Les élèves vont être assis par quatre.

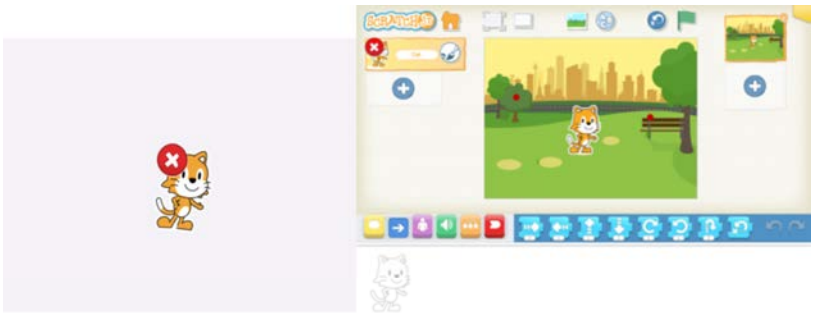
Description :

I. Tout d'abord, l'enseignante demandera aux élèves de démarrer l'application et un nouveau projet.

II. Ensuite, l'enseignante leur demandera d'**introduire l'arrière-plan**, «Parc ». Il est important de signaler que dans la scène ils existent deux points, une verte et une rouge, pour faciliter les élèves à distinguer le début et l'arrivée de chaque personnage.

III. Ensuite, l'enseignante demandera aux élèves de découvrir le processus pour **effacer le personnage** de la scène. Si les élèves n'arrivent pas à le découvrir, l'enseignante le leur présentera.

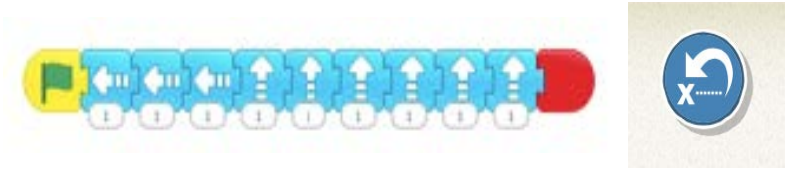
> Pour effacer un personnage, il faut, soit appuyer longtemps sur le personnage, un « X » va apparaître, on clique sur le « X » et le personnage sera enlevé de la scène, soit appuyer longtemps sur le personnage à gauche, un « X » va s'apparaître, on clique sur le « X » et le personnage sera enlevé de la scène.



IV. Après, l'enseignante demandera aux élèves d'**introduire un nouveau personnage** dans la scène, le personnage « papillon ». Si les élèves ne se souviennent pas du processus pour introduire un nouveau personnage, l'enseignante le leur rappellera.

> Pour introduire un nouveau personnage dans la scène, il faut appuyer sur le plus qui se trouve à gauche de la scène, pour arriver à la banque de personnages existants sur le logiciel. Après, ils existe deux façons pour le faire, soit en appuyant sur le personnage « papillon » et après valider notre choix, soit en appuyant deux fois sur le personnage « papillon ».

V. Par la suite, l'enseignante demandera aux élèves de **poser le personnage papillon** dans le deuxième cercle de la gauche à la droite et **le programmer pour arriver sur l'arbre à gauche**. Il est important de signaler que dans la scène il existe deux points, un verte et un rouge, pour faciliter les élèves à distinguer le début et l'arrivée de chaque personnage. Chaque fois que les élèves terminent un essai de programmation, l'enseignante leur rappellera d'**initialiser la position du personnage dans la scène**, pour vérifier s'ils ont bien programmé. L'enseignante leur expliquera ensuite l'**utilisation de paramètres** au lieu d'utiliser plusieurs mêmes commandes.



VI. Après, l'enseignante demandera aux élèves d'**introduire un nouveau personnage** dans la scène, **le personnage « mouche »**. Si les élèves ne se souviennent pas du processus pour introduire un nouveau personnage, l'enseignante le leur rappellera. Par la suite, elle leur demandera de **poser le personnage « mouche »** dans le troisième cercle de la gauche à la droite et **le programmer pour qu'il arrive sur le bang**, au point rouge.

VII. Par la suite, l'enseignante demandera aux élèves d'**enregistrer un son pour le personnage « mouche »**, pour qu'ils se souviennent du processus d'enregistrement d'un

son. L'enseignante leur expliquera également l'utilisation de paramètres au lieu d'utiliser plusieurs mêmes commandes.

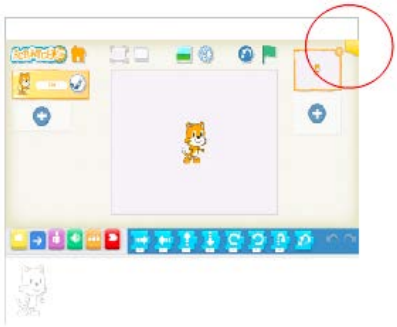
VIII. Ensuite, l'enseignante demandera aux élèves **de déclencher les deux personnages en même temps dans la scène.** Si les élèves ne se souviennent pas de la fonctionnalité de l'**icône du drapeau vert** en haut, l'enseignante leur la rappellera.

Lorsqu'on clique sur le drapeau vert en haut, les deux personnages déclenchent leurs programmes en même temps. Essayez d'appuyer sur le drapeau vert en haut de la scène. Est-ce que ça marche ? Les programmes de vos personnages se déclenchent ou pas ?

IX. Les programmes de leurs personnages ne se déclencheront pas s'ils n'ont pas la commande du drapeau-vert au début de leurs programmes. C'est pourquoi l'enseignante demandera aux élèves de réfléchir pour trouver s'il manque quelque chose dans leurs programmes dans le cas où ils ont oublié d'ajouter ces deux commandes auparavant. Si les élèves ne se souviennent pas, l'enseignante fera référence au jeu des cartes et au fait que leur machine avait un départ et une arrivée spécifique, pour leur rappeler d'**ajouter le drapeau vert et la fin rouge** sur les programmes de leurs personnages.

X. Puis, l'enseignante demandera aux élèves de **projeter leur projet en plein-écran,** comme un dessin-animé. Si les élèves ne se souviennent pas de l'icône pour réaliser cela, l'enseignante la leur rappellera.


XI. Après, l'enseignante leur demandera d'**enregistrer leur projet**. Elle leur expliquera qu'il est nécessaire de cliquer sur le coin jaune en haut et à droite. Ensuite, il faut effacer ce qui est déjà écrit et écrire à l'aide du clavier tactile « A3prénom ».



XII. Enfin, l'enseignante leur demandera d'aller à la maison pour vérifier si leur projet a été bien enregistré ou pas.

4ème séance : Répéter une commande d'un nombre de fois prédéfini

Durée : 45 min.

Concepts : Répétition d'une commande, paramètre de la commande de répétition.

Connaissances : Répétition d'une commande, paramètre de la commande de répétition, signification du paramètre de la commande de la répétition.

Processus : Introduction d'une commande dans la commande de répétition, faire sortir une commande de la commande de la répétition, utilisation du paramètre de la commande de la répétition.

Objectifs :

- S'expérimenter avec **la commande de la répétition** afin de découvrir s**a fonctionnalit**é.
- Découvrir le processus de l'**introduction d'une commande dans la commande de la répétition.**
- Découvrir le processus pour **faire sortir une commande de la commande de la répétition.**
- Comprendre la **signification du paramètre de la commande de la répétition**.
- **Utilisation de la commande de la répétition de manière opérationnelle** pour résoudre un problème de programmation.

Matériel : tablettes avec l'application ScratchJr téléchargée.

Organisation de la classe : La séance sera réalisée avec l'ensemble de la classe. Chaque élève a sa propre tablette. La tablette de l'enseignante sera connectée avec le TNI de la classe, pour que tous les élèves puissent voir ce qui se passe sur la tablette de l'enseignante. Les élèves vont être assis par quatre.

Ressources complémentaires sur la répétition pour l'enseignante :

- **Répétition** : commande qui ressemble à une pièce de puzzle incomplète.

- **Que disent les élèves à propos de cette commande** : les élèves pensent que cette commande fait tourner les personnages à gauche, à cause de la direction de la flèche qui existe sur la commande.

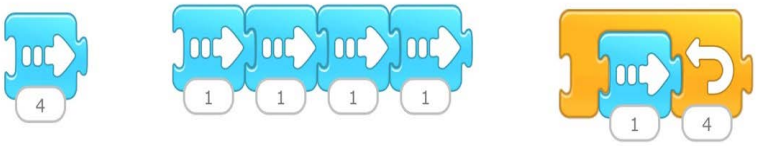
- **Fonctionnalité de la commande** : Cette commande fait répéter la commande ou la séquence des commandes qui se trouvent à son intérieur un nombre de fois prédéfini par son paramètre. Cette commande n'a aucun effet sur le personnage lorsqu'elle est vide.

  Par exemple, lorsque nous ajoutons une commande pour déplacer le personnage dans la scène, le paramètre de la répétition signifiera le nombre de pas que le personnage fera vers la direction choisie. Si nous ajoutons une commande de son pour faire parler le personnage, le paramètre de la répétition signifiera le nombre de fois que notre personnage dira le mot ou la phrase choisie.

  Dans le cas de la répétition d'une commande de déplacement du personnage, la modification du paramètre de la commande de déplacement qui se trouve à l'intérieur de la répétition,

- **Pour l'enseignement de la commande de la répétition aux élèves de CP,** nous avons choisi de limiter la fonctionnalité de la commande en ce qui concerne la répétition de commandes de déplacement, en expliquant aux élèves que quand nous ajoutons une commande de déplacement à l'intérieur de la répétition on ne modifie que le paramètre de la répétition et pas le paramètre de la commande de déplacement. Nous avons fait ce choix, parce que après il s'agit de la multiplication et ça devient compliqué pour les élèves de cet âge-là.

- **Nous avons appris 3 façons pour faire la même chose** :



Description :

I. Tout d'abord, l'enseignante demandera aux élèves de démarrer l'application ScratchJr et ouvrir le projet qu'elle leur présentera sur sa tablette.

II.

Ensuite, l'enseignante demandera aux élèves d'observer les commandes qui existent dans l'espace de programmation et essayer de comprendre quelle est la fonctionnalité de la nouvelle commande orange.

III. Après avoir laissé du temps aux élèves pour découvrir la fonctionnalité de la nouvelle commande, l'enseignante lancera un processus de découverte de celle-ci, à l'aide des questions qui amèneront les élèves à cette découverte.

- Qu'est-ce qu'il existe actuellement sur l'espace de programmation ? Avez-vous repéré quelque chose de nouveau ?
- Est-ce que l'un d'entre vous a déjà découvert la fonctionnalité de la nouvelle commande ?
- Comment peut-on découvrir que fait-elle la nouvelle commande ?
- Allons-y , on commence avec la première commande de déplacement. Que fait le « chat » avec cette commande normalement ?
- Essayez de l'ajouter à l'intérieur de la commande orange. Qu'est-ce que vous observez ? Que fait le « chat » maintenant ?

IV. Pour continuer la séance, l'enseignante présentera aux élèves la « grille ». À l'aide de la grille, ils pourront mieux suivre les déplacements du personnage « chat ». Ensuite, elle continuera avec les questions.

V. Ensuite, elle leur demandera de poser le « chat » sur le (1,1) de la « grille » et relancer le programme du « chat », pour pouvoir compter le nombre de pas qu'il fait sur la « grille ».

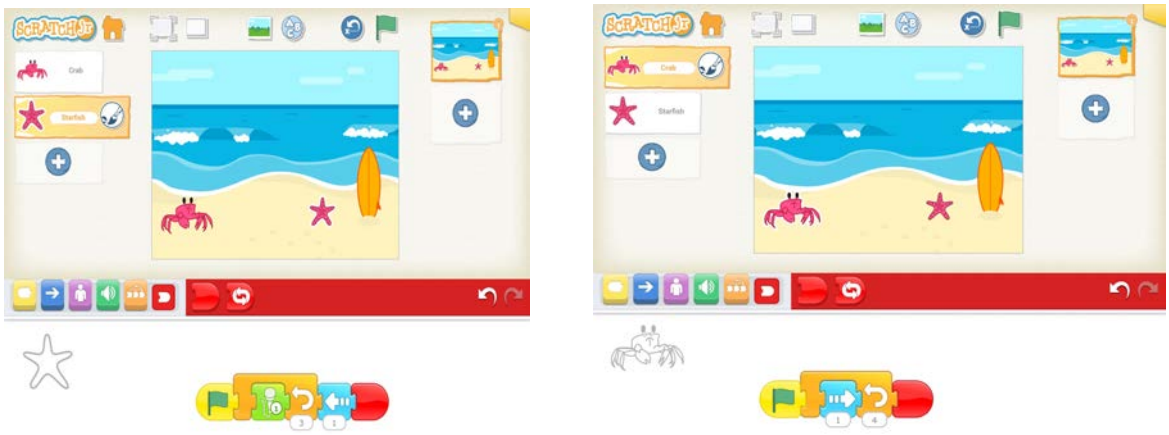
- De combien de pas avance le « chat » sur la « grille » ?
- Modifiez le paramètre de la commande orange. Choisissez le chiffre 3 par exemple. Posez le « chat » sur le (1,1) de la « grille » et lancez le programme du « chat ». Que fait le « chat » maintenant ? De combien de pas avance-t-il ?
- Essayez maintenant la commande qui fait le « chat » aller en haut. Posez le « chat » sur le (1,1). Que fait le « chat » normalement avec cette commande ?
- Mettez la commande à l'intérieur de la commande orange. Posez le « chat » sur le (1,1) de la « grille » et lancez le programme. Combien de pas avance le chat sur la « grille » ?
- Modifiez le paramètre de la commande orange de nouveau. Choisissez le chiffre 4. Posez le « chat » de nouveau sur le (1,1) de la « grille ». Que fait-il le « chat » maintenant ?
- Choisissez le chiffre 2. Posez le « chat » sur le (1,1) de la « grille ». Que fait-il le « chat » maintenant ?
- Du coup quel chiffre compte à chaque fois ? Celui de la commande à l'intérieur de la commande orange ou celui de la commande orange ?

- Essayez maintenant d'introduire la commande du son à l'intérieur de la commande orange. Que fait-il le « chat » maintenant ? Combien de fois parle-t-il ?
- Choisissez le chiffre 4. Lancez le programme de nouveau. Combien de fois il parle maintenant ?
- Choisissez le chiffre 3. Que fait-il le « chat » maintenant ?
- Qu'est-ce que vous observez finalement ? À quoi ça sert finalement la commande orange ?
- Qu'est-ce que signifie le paramètre pour la commande orange ? Quelle information nous donne-t-elle pour le « chat » ?

VI. Si les élèves n'arrivent pas à trouver la fonctionnalité de la nouvelle commande, l'enseignante leur la présentera.

VII. Par la suite, l'enseignante demandera aux élèves de résoudre un problème à l'aide de la commande de répétition.

VIII. L'enseignante présentera un projet déjà prêt aux élèves, qui doivent analyser le programme des personnages et le reconstruire sur leurs tablettes.



IX. Le projet que les élèves doivent reproduire comprend deux personnages, « le crabe » et « l' étoile de mer » qui se trouvent sur l'arrière-plan « plage ».

X. Les élèves doivent programmer « le crabe » pour qu'il se déplace quatre fois à droite.

XI. Ensuite, les élèves doivent programmer l'étoile de mer pour qu'il dise « Salut » trois fois et aller à gauche une fois après.

XII. L'enseignante n'oubliera pas de rappeler aux enfants l'initialisation de la position des personnages.

XIII. Ensuite, l'enseignante demandera aux élèves **de déclencher les deux personnages en même temps dans la scène.** Si les élèves ne se souviennent pas de la fonctionnalité de l'**icône du drapeau vert** en haut de la scène, l'enseignante la leur rappellera.

Lorsqu'on clique sur le drapeau vert en haut, les deux personnages déclenchent leurs programmes en même temps. Essayez d'appuyer sur le drapeau vert en haut. Est-ce que ça marche ? Les programmes de vos personnages se déclenchent ou pas ?

XIV. Les programmes de personnages ne se déclencheront pas, s'ils n'ont pas la commande du drapeau vert au début de leurs programmes. C'est pourquoi l'enseignante demandera aux élèves de réfléchir pour trouver s'il manque quelque chose dans leurs programmes dans le cas où ils ont oublié d'ajouter ces deux commandes auparavant. Si les élèves ne se souviennent pas, l'enseignante fera référence au jeu des cartes et au fait que leur machine avait un départ et une arrivée spécifique, pour leur rappeler d'**ajouter le drapeau vert et la fin rouge** sur les programmes de leurs personnages.

XV. Puis, l'enseignante demandera aux élèves de **projeter leur projet en plein-écran,** comme un dessin-animé. Si les élèves ne se souviennent pas de l'icône pour réaliser cela, l'enseignante la leur rappellera.

XVI. Après, l'enseignante leur demandera d'**enregistrer leur projet**. Elle leur expliquera qu'il est nécessaire de cliquer sur le coin jaune en haut et à droite. Ensuite, il faut effacer ce qui est déjà écrit et écrire à l'aide du clavier tactile « A4prénom ».

XVII. Enfin, l'enseignante leur demandera d'aller à la maison pour vérifier si leur projet a été bien enregistré ou pas.

5ème séance : Répéter un motif des commandes un nombre des fois prédéfini

Durée : 45 min

Concepts : répéter un motif des commandes un nombre de fois prédéfini, paramètre de la commande de répétition d'un motif des commandes.

Connaissances : répéter un motif des commandes un nombre de fois prédéfini, paramètre de la commande de répétition d'un motif des commandes, signification de la commande de la répétition d'un motif des commandes un nombre de fois prédéfini.

Processus : Introduire et faire sortir un motif des commandes de la commande de répétition, utilisation du paramètre de la commande de répétition d'un motif des commandes un nombre de fois prédéfini.

Objectifs :

- Se rappeler de la **commande de répétition** d'une commande un nombre des fois prédéfini.
- S'expérimenter avec **la commande de la répétition** d'un motif des commandes un nombre de fois prédéfini, afin de découvrir s**a fonctionnalit**é.
- Découvrir le processus de l'**introduction d'un motif des commandes dans la commande de la répétition.**

- Découvrir le processus pour **faire sortir un motif des commandes de la commande de la répétition.**
- Comprendre la **signification du paramètre de la commande de la répétition pour le motif des commandes.**
- **Utilisation de la commande de la répétition d'un motif des commandes de manière opérationnelle,** pour résoudre un problème de programmation.

Matériel : tablettes avec l'application ScratchJr téléchargée.

Organisation de la classe : La séance sera réalisée avec l'ensemble de la classe. Chaque élève a sa propre tablette. La tablette de l'enseignante sera connectée avec le TNI de la classe, pour que tous les élèves puissent voir ce qui se passe sur la tablette de l'enseignante. Les élèves vont être assis par quatre.

Description :

I. Tout d'abord, l'enseignante demandera aux élèves de lui rappeler la fonctionnalité de la commande orange et comment elle s'appelle.

II. Si les élèves n'arrivent pas à se rappeler de sa fonctionnalité, l'enseignante leur rappellera leur dernière séance. Si les élèves ont encore des difficultés à se souvenir la fonctionnalité de la répétition, elle leur demandera d'utiliser la commande de la répétition de nouveau pour se souvenir.

III. Ensuite, l'enseignante leur demandera de faire attention au programme de la grenouille et elle leur posera des questions pour les aider à identifier le motif des commandes qui se répètent.

➢ Que fait la grenouille ?

➢ Qu'est-ce qu'il se répète sur le programma de la grenouille ?

➢ Pourriez-vous identifier les commandes qui se répètent ?

➢ Ce qui se répète il s'appelle un « motif ». Combien des fois ce motif est-il répété ?

IV. Après, l'enseignante leur demandera de reproduire le projet qu'elle a sur sa tablette, en utilisant le même arrière-plan, le même personnage et créer le motif des commandes qui se répètent sur son projet, les fois qu'il se répète sur le programme de la grenouille.

V. Par la suite, l'enseignante leur demandera comment ils peuvent créer le même programme pour la grenouille en utilisant la commande de la répétition. À ce moment-là, nous allons laisser du temps aux élèves à s'exercer avec le processus d'introduction d'un motif des commandes à l'intérieur de la répétition, parce que c'est un processus qui pose souvent problème.

VI. Ensuite, l'enseignante demandera aux élèves comment ils sont arrivés à faire la grenouille faire la même chose qu'avant, en utilisant la commande de répétition. Elle va réaliser les propositions des élèves sur sa tablette, en gardant à côté la version initiale du programme de la grenouille sur son espace de programmation, pour pouvoir faire chaque fois la comparaison avec le programme initial de la grenouille et valider si elle a bien programmé. L'objectif de cette stratégie didactique est que les élèves arrivent à comprendre la raison pour laquelle la répétition est utile pour résoudre des problèmes comme celui-ci, à l'aide des questions et des essais différents.

VII. L'enseignante présentera également aux élèves le processus pour introduire un motif des commandes à l'intérieur de la répétition, parce que c'est un processus qui pose souvent plusieurs problèmes aux élèves. Si des élèves arrivent à le faire tout seul, l'enseignante leur demandera de venir présenter le processus à leurs camarades, à l'aide du projecteur.

VIII. Après avoir essayé différentes façons pour réaliser la même chose, l'enseignante va essayer de leur expliquer la solution correcte du problème et la raison pour laquelle ils choisissent d'utiliser la répétition et d'introduire tel motif à l'intérieur de la répétition et tel paramètre.

IX. Si nous avons le temps, nous allons résoudre encore un problème en utilisant la répétition.

X. L'enseignante va démarrer un autre projet et elle leur demandera d'arriver au même point qu'elle se trouve, c'est-à-dire qu'elle va leur demander de choisir le même arrière-plan et le même personnage qu'elle a sur sa tablette.

XI. Ensuite, elle va leur lire l'énoncé du problème et elle leur demandera de le résoudre en utilisant la répétition. Le problème est le suivant : « Un « chat » se trouve sur l'herbe à gauche. Il fait un pas à droite et puis il dit « salut » et cela le fait 5 fois ».

XII. Dans un premier temps, nous allons laisser du temps aux élèves à essayer de résoudre le problème sur leur tablette.

XIII. Ensuite, l'enseignante demandera aux élèves comment ils sont arrivés à résoudre le problème donné et faire faire le personnage « chat » ce qu'il a été demandé par l'énoncé du problème. L'enseignante demandera de lui faire des propositions sur comment résoudre ce problème et elle les réalisera ou elle demandera aux élèves de venir les réaliser sur sa tablette et puis à l'aide des questions appropriées elle leur aidera à comprendre comment utiliser la répétition, quel motif à introduire, quel paramètre et pourquoi. Chaque fois qu'ils font un essai, ils valideront tous ensemble s'ils ont bien programmé ou pas en faisant une comparaison du résultat obtenu et de l'énoncé du problème. L'objectif de cette stratégie didactique est que les élèves arrivent à comprendre la raison pour laquelle la répétition est utile pour résoudre des problèmes comme celui-ci, à l'aide des questions et des essais différents.

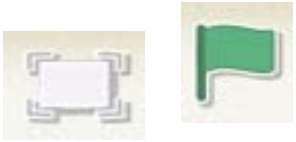
XIV. L'enseignante n'oubliera pas de rappeler aux élèves l'initialisation de la position du personnage.

XV. Ensuite, l'enseignante demandera aux élèves **de déclencher le programme du personnage dans la scène.** Si les élèves ne se souviennent pas de la fonctionnalité de l'**icône du drapeau vert** en haut, l'enseignante la leur rappellera.

Lorsqu'on clique sur le drapeau vert en haut, les personnages qui ont le drapeau vert au début de leur programme déclencheront leurs programmes. Ici nous avons un seul personnage. Essayez d'appuyer sur le drapeau vert en haut de la scène. Est-ce que ça marche ? *Le programme de votre* personnage se déclenche ou pas ?

XVI. Le programme du personnage ne se déclenchera pas, sauf s'il a la commande du drapeau vert au début de son programme. C'est pourquoi l'enseignante demandera aux élèves de réfléchir pour trouver s'il manque quelque chose à son programme dans le cas où ils ont oublié d'ajouter la commande du drapeau vert auparavant. Si les élèves ne se souviennent pas, l'enseignante fera référence au jeu des cartes et au fait que leur machine avait un départ et une arrivée spécifique, pour leur rappeler d'**ajouter le drapeau vert et la fin rouge** sur le programme du personnage.

XVII. Puis, l'enseignante demandera aux élèves de **projeter leur projet en plein-écran,** comme un dessin-animé. Si les élèves ne se souviennent pas de l'icône pour réaliser cela, l'enseignante la leur rappellera.



XVIII. Après, l'enseignante leur demandera d'**enregistrer leur projet**. Elle leur expliquera qu'il est nécessaire de cliquer sur le coin jaune en haut et à droite. Ensuite, il faut effacer ce qui est déjà écrit et écrire à l'aide du clavier tactile « A5prénom ».

XIX. Enfin, l'enseignante leur demandera d'aller à la maison pour vérifier si leur projet a été bien enregistré ou pas.

6ème séance: Répéter un motif des commandes d'un nombre des fois prédéfini

Durée : 45 min.

Concepts : répéter un motif des commandes un nombre de fois prédéfini, paramètre de la commande de répétition d'un motif des commandes.

Connaissances : répéter un motif des commandes un nombre de fois prédéfini, paramètre de la commande de répétition d'un motif des commandes, signification de la commande de la répétition d'un motif des commandes un nombre de fois prédéfini.

Processus : Introduire et faire sortir un motif des commandes de la commande de répétition, utilisation du paramètre de la commande de répétition d'un motif des commandes un nombre de fois prédéfini.

Objectifs :

- S'exercer avec **la commande de la répétition** d'un motif des commandes un nombre de fois prédéfini, afin de découvrir s**a fonctionnalit**é.
- S'entrainer avec le processus de l'**introduction d'un motif des commandes dans la commande de la répétition.**
- S'entrainer avec le processus pour **faire sortir un motif des commandes de la commande de la répétition.**
- Comprendre la **signification du paramètre de la commande de la répétition pour le motif des commandes.**
- **Utilisation de la commande de la répétition d'un motif des commandes de manière opérationnelle,** pour résoudre un problème de programmation.

Matériel : tablettes avec l'application ScratchJr téléchargée.

Organisation de la classe : La séance sera réalisée avec l'ensemble de la classe. Chaque élève a sa propre tablette. La tablette de l'enseignante sera connectée avec le TNI de la classe, pour que tous les élèves puissent voir ce qui se passe sur la tablette de l'enseignante. Les élèves vont être assis par quatre.

Description :

I. Tout d'abord, l'enseignante demandera aux élèves de lui rappeler la fonctionnalité de la commande orange et comment elle s'appelle.

II. L'enseignante va démarrer un autre projet et elle leur demandera d'arriver au même point qu'elle se trouve, c'est-à-dire qu'elle va leur demander de choisir le même arrière-plan et le même personnage qu'elle a sur sa tablette.

III. Ensuite, elle leur lira l'énoncé du problème et elle leur demandera de le résoudre en utilisant la répétition. Le problème est le suivant : « Un lapin et une grenouille se trouvent dans le paysage « Rivière ». Les deux personnages commencent leurs programmes en même temps. Le « lapin » fait un pas à droite et puis il dit « Bonjour la grenouille » et cela le fait 5 fois. La « grenouille » se déplace pour arriver aux fleurs au centre.

IV. Dans un premier temps, nous allons laisser du temps aux élèves d'essayer de résoudre le problème sur leur tablette.

V. Ensuite, l'enseignante demandera aux élèves comment ils sont arrivés à résoudre le problème donné et à faire faire les personnages lapin et grenouille ce qui a été demandé par l'énoncé du problème. L'enseignante demandera de lui faire des propositions sur comment résoudre ce problème et elle les réalisera ou elle demandera aux élèves de venir les réaliser sur sa tablette et puis à l'aide des questions appropriées elle leur aidera à comprendre comment utiliser la répétition, quel motif à introduire, quel paramètre et pourquoi. Chaque fois qu'ils font un essai, ils valideront tous ensemble s'ils ont bien programmé ou pas en faisant une comparaison du résultat obtenu et de l'énoncé du problème. L'objectif de cette stratégie didactique est que les élèves arrivent à comprendre la raison pour laquelle la répétition est utile pour résoudre des problèmes comme celui-ci, à l'aide des questions et des différents essais.

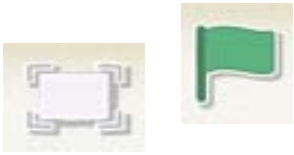
VI. L'enseignante n'oubliera pas de rappeler aux l'initialisation de la position du personnage.

VII. Ensuite, l'enseignante demandera aux élèves **de déclencher les programme des personnages dans la scène.** Si les élèves ne se souviennent pas de la fonctionnalité de l'**icône du drapeau vert** en haut, l'enseignante la leur rappellera.

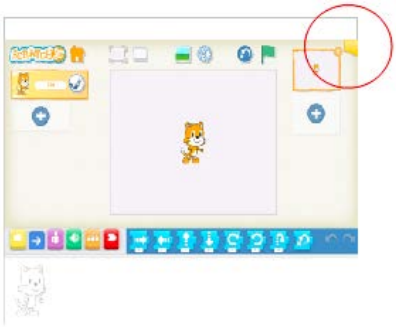
Lorsqu'on clique sur le drapeau vert en haut, les personnages qui ont le drapeau vert au début de leur programme déclencheront leurs programmes. Ici nous avons deux personnages. Essayez d'appuyer sur le drapeau vert en haut. Est-ce que ça marche ? Les programmes de vos personnages se déclenchent en même temps ou pas ?

VIII. Les programmes des personnages ne se déclencheront pas si la commande du drapeau vert ne fait pas partie de leur programme. C'est pourquoi l'enseignante demandera aux élèves de réfléchir pour trouver s'il manque quelque chose de leurs programmes dans le cas où ils ont oublié d'ajouter la commande du drapeau vert auparavant. Si les élèves ne se souviennent pas, l'enseignante fera référence au jeu des cartes et au fait que leur machine avait un départ et une arrivée spécifique, pour leur rappeler d'**ajouter le drapeau vert et la fin rouge** sur les programmes des personnages.

IX. Puis, l'enseignante demandera aux élèves de **projeter leur projet en plein-écran,** comme un dessin-animé. Si les élèves ne se souviennent pas de l'icône pour réaliser cela, l'enseignante la leur rappellera.



X. Après, l'enseignante leur demandera d'**enregistrer leur projet**. Elle leur expliquera qu'il est nécessaire de cliquer sur le coin jaune en haut et à droite. Ensuite, il faut effacer ce qui est déjà écrit et écrire à l'aide du clavier tactile « A6prénom ».



XI. Enfin, l'enseignante leur demandera d'aller à la maison pour vérifier si leur projet a été bien enregistré ou pas.

7ème séance : Messages

Durée : 45 min

Concepts : « envoyer un message », « quand message reçu »

Connaissances : commande « envoyer un message », commande « quand message reçu », fonctionnalité de la commande « envoyer un message » pour le personnage qui envoie le message, fonctionnalité de la commande « quand message reçu » pour le personnage qui reçoit le message, association de la commande « quand message reçu » avec l'existence de la commande envoyer un message dans le script de programmation de l'autre personnage.

Processus : utilisation de deux commandes de messages dans les scripts de programmation de deux personnages.

Objectifs :

➢ S'exercer avec les commandes « envoyer un message » et « quand message reçu ».

- Découvrir la fonctionnalité de la commande « envoyer un message » et « quand message reçu ».
- Utiliser les deux commandes de messages de manière opérationnelle, afin de résoudre un problème de programmation.

Matériel : tablettes avec l'application ScratchJr téléchargée, commandes du logiciel imprimées et plastifiées.

Organisation de la classe : La séance sera réalisée avec l'ensemble de la classe. Chaque élève a sa propre tablette. La tablette de l'enseignante sera connectée avec le TNI de la classe, pour que tous les élèves puissent voir ce qui se passe sur la tablette de l'enseignante. Les élèves vont être assis par quatre.

Ressources complémentaires sur les commandes de messages pour l'enseignante :

- **Messages** : ce concept est implémenté par deux commandes, la commande « envoyer un message » qui envoie un message de couleur spécifique à un personnage et la commande « quand message reçu », avec laquelle le programme du personnage dont la commande du message a la même couleur est déclenché.

- **Que disent les élèves à propos de ces commandes** : les élèves disent que la commande « envoyer un message » sert à envoyer une lettre à quelqu'un et que la commande « quand message reçu » sert à lire une lettre.
- **Fonctionnalité de la commande** : La commande « envoyer message » peut être ajoutée dans un programme, comme toutes les autres commandes sur ScratchJr. Son seul paramètre est la couleur. La commande « quand message reçu » sert à déclencher un événement et elle se place au début d'un programme.

  Ce concept sert à faire communiquer deux ou plusieurs personnages à travers l'envoi et la réception d'un message. En effet, les programmes des personnages sur ScratchJr

s’exécutent de manière concurrente et pas l’un après l’autre. Les commandes de messages permettent de déclencher les programmes de deux personnages l’un après l’autre.

Il est nécessaire d’avoir les deux commandes de messages de la même couleur, dans les programmes des deux ou plusieurs personnages différents. Par exemple, si la commande « envoyer un message » fait partie du programme d’un personnage et la commande « quand message reçu » ne fait pas partie du programme de l’autre personnage et les deux commandes ne sont pas de la même couleur, les deux personnages ne communiqueront pas.

Description :

I. Tout d’abord, l’enseignante présentera aux élèves un projet déjà prêt en pleine écran où les deux commandes de messages ne sont pas présentes et elle leur demandera d'observer ce qui se passe sur son écran. Les tablettes ne vont pas être encore distribuées aux élèves, pour qu'ils soient le plus attentifs possible et observer ce qui se passe au projet présenté en pleine écran.

II. Après avoir laissé du temps aux élèves pour observer le projet en pleine écran, l'enseignante leur posera les questions suivantes pour les aider à découvrir ce qui se passe :

➢ Qu’est-ce qui se passe sur ce projet ?

➢ Est-ce que les deux personnages commencent leurs programmes en même temps ?

➢ Que fait le « chat » ? Que fait la « fusée » ?

III. Par la suite, elle leur demandera d'utiliser les cartes représentant les commandes du logiciel présentes dans une enveloppe devant eux, pour reproduire les programmes de deux personnages. Après, l'enseignante validera le travail fait par les élèves.

IV. Ensuite, elle leur présentera le même projet mais cette fois-ci avec les commandes de messages et elle leur demandera d'observer ce qui se passe sur son écran. Les tablettes ne sont toujours pas distribuées aux élèves, pour qu'ils soient le plus attentifs possible à la projection en plein-écran.

V. Après avoir laissé du temps aux élèves pour observer le projet en plein-écran, l'enseignante leur posera les questions suivantes pour les aider à découvrir ce qui se passe.

- Qu'est-ce qui se passe sur ce projet ?
- Est-ce que les deux personnages commencent leurs programmes en même temps ?
- Qui commence en premier son programme ? Qui en deuxième ?
- Que fait le « chat » ? Que fait la « fusée » ?
- Pourquoi la « fusée » commence son programme en deuxième et pas en même temps comme sur le projet d'avant ?

VI. À ce moment-là, l'enseignante va quitter le mode « plein-écran » pour montrer aux élèves le programme de chaque personnage et elle leur posera les questions suivantes pour les guider à comprendre ce qui se passe sur ce projet.

- On va commencer avec le programme du « chat ». Le programme du « chat » est le même que celui que vous avez reproduit avant sur vos tables ? Si non, pourquoi ? Qu'est-ce que c'est qui a changé ?

- Avez-vous repéré quelque chose de nouveau ?

- Que pourrait-elle faire cette nouvelle commande ?

VII. Par la suite, elle leur demandera de reproduire de nouveau le programme du personnage « chat » avec les cartes et elle passera valider leur travail.

VIII. Après, ils vont s'occuper du programme de la « fusée ». L'enseignante leur montrera le programme de la « fusée » et leur posera les questions suivantes :

- Le programme de la « fusée » est le même que celui que vous avez reproduit avant sur vos tables ?Si non, pourquoi ? Qu'est-ce qui a changé ?
- Avez-vous repéré quelque chose de nouveau ?

- Que pourrait faire cette nouvelle commande ?

- Quand la « fusée » commence son programme ?

- Qui dit à la « fusée » de commencer ?

IX. Par la suite, elle leur demandera de reproduire de nouveau le programme du personnage « fusée » avec leurs commandes imprimées et elle passera valider leur travail.

X. Ensuite, l'enseignante leur demandera de fermer leurs jeux et elle va enlever la commande « quand message reçu » de son projet, elle va revenir en mode pleine-écran et elle lancera son projet de nouveau.

- Qu'est-ce qui se passe maintenant ?

- Pourquoi la « fusée » ne commence pas son programme ?

- Est-ce qu'elle a reçu le message que le « chat » lui a envoyé ?

- Et si on met la commande « quand message reçu » au programme de la « fusée » et on enlève la commande « envoyer un message » du programme du « chat », que va-t-il se passer ?

XI. Si les élèves n'arrivent pas à comprendre la fonctionnalité des commandes de messages elle va la leur expliquer. Il faut que les élèves comprennent que :

- Les deux personnages communiquent à l'aide de messages.
- Les deux personnages ne déclenchent pas leurs programmes en même temps.
- Le personnage qui a le drapeau vert au début de son script déclenche son programme toujours au premier.
- Le personnage qui a la commande « quand message reçu » au début de son programme, déclenche son programme après qu'il aura reçu le message par l'autre personnage.
- Le personnage qui commence son programme avec la commande « quand message reçu » ne déclenchera jamais son programme, si on enlève la commande « envoyer un message » du programme du personnage qui envoie le message.

XII. Après l'enseignante va distribuer les tablettes aux enfants. Si nous avons du temps, les élèves résoudront un problème pour se familiariser avec l'utilisation des commandes. Si nous avons pas de temps, ils reproduiront le projet avec le « chat » et la « fusée ».

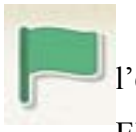
XIII. L'enseignante lira l'énoncé du problème aux élèves. Une étoile de mer et un poisson jaune se trouve dans l'océan. L'étoile de mer commence son programme en premier et il se déplace pour arriver sur l'algue orange en bas. Dès que l'étoile de mer arrive sur l'algue orange le poisson jaune commence son programme pour qu'il arrive à toucher le bateau en haut. L'enseignante leur montrera le projet prêt en plein-écran.

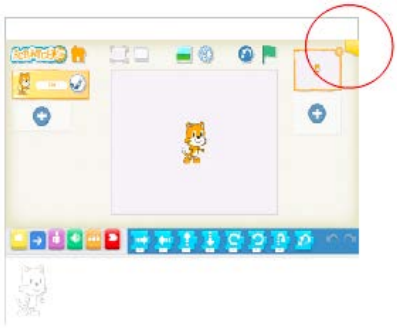
XIV. L'enseignante n'oubliera pas de rappeler aux enfants l'initialisation de la position des personnages.

XV. Ensuite, l'enseignante demandera aux élèves **de déclencher les deux personnages dans la scène.** Si les élèves ne se souviennent pas de la fonctionnalité de l'**icône du drapeau vert** en haut, l'enseignante la leur rappellera.

Lorsqu'on clique sur le drapeau vert en haut, les deux personnages déclenchent leurs programmes en même temps. Essayez d'appuyer sur le drapeau vert en haut. Est-ce que ça marche ? Les programmes de vos personnages se déclenchent ou pas ?

XVI. Puis, l'enseignante demandera aux élèves de **projeter leur projet en plein-écran,** comme un dessin-animé. Si les élèves ne se souviennent pas de l'icône pour réaliser cela, l'enseignante la leur rappellera.

XVII. Après,  l'enseignante leur demandera d'**enregistrer leur projet**. Elle leur expliquera qu'il est nécessaire de cliquer sur le coin jaune en haut et à droite. Ensuite, il faut effacer ce qui est déjà écrit et écrire à l'aide du clavier tactile « A7prénom ».



XVIII. Enfin, l'enseignante leur demandera d'aller à la maison pour vérifier si leur projet a été bien enregistré ou pas.

8ème séance : Messages

Durée : 45 min

Concepts : « envoyer un message », « quand message reçu »

Connaissances : commande « envoyer un message », commande « quand message reçu », fonctionnalité de la commande « envoyer un message » pour le personnage qui envoie le message, fonctionnalité de la commande « quand message reçu » pour le personnage qui reçoit le message, association de la commande « quand message reçu » avec l'existence de la commande envoyer un message dans le script de programmation de l'autre personnage.

Processus : utilisation de deux commandes de messages dans les scripts de programmation de deux personnages.

Objectifs :

- S'exercer avec les commandes « envoyer un message » et « quand message reçu ».
- Rappeler la fonctionnalité de la commande « envoyer un message » et « quand message reçu ».
- Utiliser les deux commandes de messages de manière opérationnelle afin de résoudre un problème de programmation.

Matériel : tablettes avec l'application ScratchJr téléchargée.

Organisation de la classe : La séance sera réalisée avec l'ensemble de la classe. Chaque élève a sa propre tablette. La tablette de l'enseignante sera connectée avec le TNI de la classe, pour que tous les élèves puissent voir ce qui se passe sur la tablette de l'enseignante. Les élèves vont être assis par quatre.

Description :

I. Tout d'abord, l'enseignante présentera aux élèves un projet déjà prêt en pleine-écran et elle leur demandera d'observer ce qui se passe sur son écran. Les tablettes ne vont pas être encore distribuées aux élèves, pour qu'ils soient le plus attentifs possible et qu'ils observent la projection.

- Qu'est-ce qui se passe sur ce projet ?

➢ Est-ce que les deux personnages commencent leurs programmes en même temps ?

➢ Qui commence en premier son programme ? Qui en deuxième ?

➢ Pourquoi le papillon commence son programme en deuxième et pas en même temps que la grenouille ?

II. Après elle lira l'énoncé du problème aux élèves. Une grenouille et un papillon se trouvent dans le parc. La grenouille commence son programme en premier et il se déplace pour arriver sur le banc. Dès que la grenouille arrive sur le banc, le papillon commence son programme pour qu'il arrive dans le deuxième cercle.

III. L'enseignante n'oubliera pas de rappeler aux enfants l'initialisation de la position des personnages.

IV. Ensuite, l'enseignante demandera aux élèves comment ils sont arrivés à résoudre le problème donné et à faire faire aux personnages grenouille et papillon ce qui a été demandé par l'énoncé du problème. L'enseignante demandera de lui faire des propositions sur comment résoudre ce problème et elle les réalisera ou elle demandera aux élèves de venir les réaliser sur sa tablette et puis à l'aide des questions appropriées elle leur aidera à comprendre comment utiliser les messages pour résoudre ce problème. Chaque fois qu'ils font un essai, ils valideront tous ensemble s'ils ont bien programmé ou pas en faisant une comparaison du résultat obtenu et de l'énoncé du problème. L'objectif de cette stratégie didactique est que les élèves arrivent à comprendre la raison pour laquelle les messages sont utiles pour résoudre des problèmes comme celui-ci, à l'aide des questions et des essais différents.

V. Ensuite, l'enseignante demandera aux élèves **de déclencher les deux personnages dans la scène.** Si les élèves ne se souviennent pas de la fonctionnalité de l'**icône du drapeau vert** en haut, l'enseignante la leur rappellera.

Lorsqu'on clique sur le drapeau vert en haut, les deux personnages déclenchent leurs programmes en même temps. Essayez d'appuyer sur le drapeau vert en haut. Est-ce que ça marche ? Les programmes de vos personnages se déclenchent ou pas ?

VI. Puis, l'enseignante demandera aux élèves de **projeter leur projet en plein-écran,** comme un dessin-animé. Si les élèves ne se souviennent pas de l'icône pour réaliser cela, l'enseignante la leur rappellera.

VII. À la fin de cette séance il faut que les élèves comprennent que :

- Les deux personnages communiquent à l'aide de messages.
- Les deux personnages ne déclenchent pas leurs programmes en même temps.
- Le personnage qui a le drapeau vert au début de son script déclenche son programme toujours au premier.
- Le personnage qui a la commande « quand message reçu » au début de son programme, déclenche son programme après qu'il ait reçu le message par l'autre personnage.
- Le personnage qui commence son programme avec la commande « quand message reçu » ne déclenchera jamais son programme, si on enlève la commande « envoyer un message » du programme du personnage qui envoie le message.

VIII. Après, l'enseignante leur demandera d'**enregistrer leur projet**. Elle leur expliquera qu'il est nécessaire de cliquer sur le coin jaune en haut et à droite. Ensuite, il faut effacer ce qui est déjà écrit et écrire à l'aide du clavier tactile « A8prénom ».

IX. Enfin, l'enseignante leur demandera d'aller à la maison pour vérifier si leur projet a été bien enregistré ou pas.

X. Avant de finir la séance l'enseignante leur demandera de fermer leurs jeux et elle va enlever la commande « quand message reçu » de son projet, elle va revenir en mode plein-écran et elle lancera son projet de nouveau.

- Qu'est-ce qui se passe maintenant ?
- Pourquoi le papillon ne commence pas son programme ?

XI. L'objectif de cette stratégie didactique est que les élèves puissent s'exprimer autour des messages, pour qu'on puisse comprendre quelles sont leurs difficultés principales.

9ème séance : Messages

Durée : 45 min

Concepts : « envoyer un message », « quand message reçu »

Connaissances : commande « envoyer un message », commande « quand message reçu », fonctionnalité de la commande « envoyer un message » pour le personnage qui envoie le message, fonctionnalité de la commande « quand message reçu » pour le personnage qui reçoit le message, association de la commande « quand message reçu » avec l'existence de la commande envoyer un message dans le programme de l'autre personnage.

Processus : utilisation de deux commandes de messages dans programmes de deux personnages.

Objectifs :

- S'exercer avec les commandes « envoyer un message » et « quand message reçu ».
- Rappeler la fonctionnalité de la commande « envoyer un message » et « quand message reçu ».
- Utiliser les deux commandes de messages de manière opérationnelle afin de résoudre un problème de programmation.

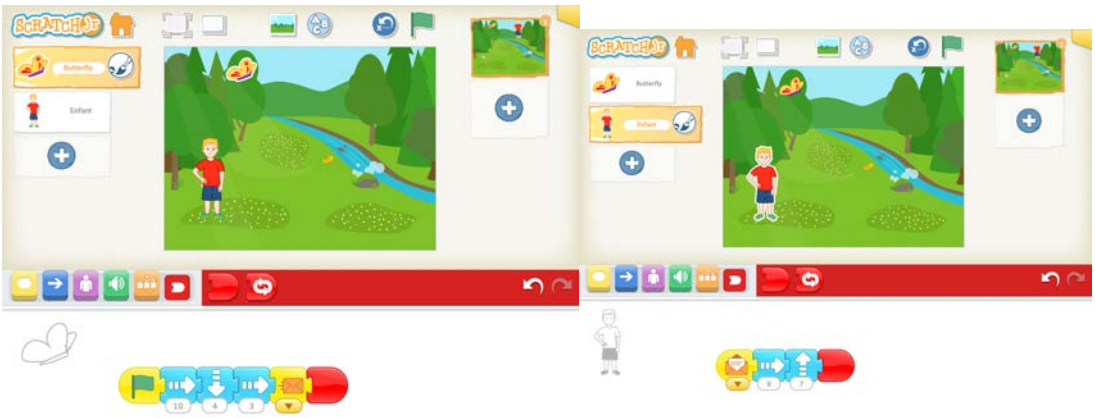
Matériel : tablettes avec l'application ScratchJr téléchargée.

Organisation de la classe : La séance sera réalisée avec l'ensemble de la classe. Chaque élève a sa propre tablette. La tablette de l'enseignante sera connectée avec le TNI de la classe, pour que tous les élèves puissent voir ce qui se passe sur la tablette de l'enseignante. Les élèves vont être assis par quatre.

Description :

I. Tout d'abord, l'enseignante présentera aux élèves un projet déjà prêt en plein-écran et elle leur demandera d'observer ce qui se passe sur son écran. Les tablettes ne vont pas être encore distribuées aux élèves, pour que les élèves puissent observer attentivement la projection.

- Qu'est-ce qui se passe sur ce projet ?
- Est-ce que les deux personnages commencent leurs programmes en même temps ?
- Qui commence en premier son programme ? Qui en deuxième ?
- Pourquoi l'enfant commence son programme en deuxième et pas en même temps que le papillon?



II. Après, elle lira l'énoncé du problème aux élèves. Un papillon et un enfant se trouvent dans le paysage « rivière ». Le papillon commence son programme en premier et il se déplace pour arriver sur les fleurs du milieu. Dès que le papillon arrive sur les fleurs du milieu, l'enfant commence son programme pour qu'il traverse à l'autre côté de la rivière.

III. L'enseignante n'oubliera pas de rappeler aux enfants l'initialisation de la position des personnages.

IV. Ensuite, l'enseignante demandera aux élèves comment ils sont arrivés à résoudre le problème donné et à faire faire aux personnages enfant et papillon ce qui a été demandé par l'énoncé du problème. L'enseignante demandera de lui faire des propositions sur comment résoudre ce problème et elle les réalisera ou elle demandera aux élèves de venir les réaliser sur sa tablette et puis à l'aide des questions appropriées elle leur aidera à comprendre comment utiliser les messages pour résoudre ce problème. Chaque fois qu'ils font un essai, ils valideront tous ensemble s'ils ont bien programmé ou pas en faisant une comparaison du résultat obtenu et de l'énoncé du problème. L'objectif de cette stratégie didactique est que les élèves arrivent à comprendre la raison pour laquelle les messages sont utiles pour résoudre des problèmes comme celui-ci, à l'aide des questions et des essais différents.

V. Ensuite, l'enseignante demandera aux élèves **de déclencher les deux personnages dans la scène.** Si les élèves ne se souviennent pas de la fonctionnalité de l'**icône du drapeau vert** en haut, l'enseignante la leur rappellera.

Lorsqu'on clique sur le drapeau vert en haut de la scène, les deux personnages déclenchent leurs programmes en même temps. Essayez d'appuyer sur le drapeau vert en haut. Est-ce que ça marche ? Les programmes de vos personnages se déclenchent ou pas ?

VI. Puis, l'enseignante demandera aux élèves de **projeter leur projet en plein-écran,** comme un dessin-animé. Si les élèves ne se souviennent pas de l'icône pour réaliser cela, l'enseignante leur la rappellera.


VII. À la fin de cette séance il faut que les élèves comprennent que :

- Les deux personnages communiquent à l'aide de messages.
- Les deux personnages ne déclenchent pas leurs programmes en même temps.
- Le personnage qui a le drapeau vert au début de son script déclenche son programme toujours au premier.
- Le personnage qui a la commande « quand message reçu » au début de son programme, déclenche son programme après qu'il ait reçu le message par l'autre personnage.
- Le personnage qui commence son programme avec la commande « quand message reçu » ne déclenchera jamais son programme, si on enlève la commande « envoyer un message » du programme du personnage qui envoie le message et qui le commande à commencer son programme.

VIII. Après, l'enseignante leur demandera d'**enregistrer leur projet**. Elle leur expliquera qu'il est nécessaire de cliquer sur le coin jaune en haut et à droite. Ensuite, il faut effacer ce qui est déjà écrit et écrire à l'aide du clavier tactile « A9iprénom ».

IX. Enfin, l'enseignante leur demandera d'aller à la maison pour vérifier si leur projet a été bien enregistré ou pas.

X. Avant de finir la séance l'enseignante leur demandera de fermer leurs yeux et elle va enlever la commande « quand message reçu » de son projet, elle va revenir en mode plein-écran et elle lancera son projet de nouveau.

- Qu'est-ce qui se passe maintenant ?
- Pourquoi l'enfant commence son programme en même temps que le papillon?
- Qu'est-ce qu'on voulait au début ?

XI. Ensuite, l'enseignante présentera aux élèves un deuxième problème déjà prêt en plein-écran et elle leur demandera d'observer ce qui se passe sur son écran. Les tablettes ne vont pas être encore distribuées aux élèves, pour qu'ils soient le plus attentifs possible et observer ce qui se passe au projet présenté en plein-écran.

- Qu'est-ce qui se passe sur ce projet ?
- Est-ce que les deux personnages commencent leurs programmes en même temps ?
- Qui commence en premier son programme ? Qui en deuxième ?
- Pourquoi Tac commence son programme en deuxième et pas en même temps que Scratch?

XII. Après elle lira l'énoncé du problème aux élèves. Scratch et Tac se trouvent dans le paysage « printemps ». Scratch commence son programme en premier et il se déplace pour arriver au milieu du parterre des fleurs à droite. Dès que Scratch arrive sur le parterre des fleurs en bas, Tac commence son programme pour arriver sur la branche de l'arbre en haut.

XIII. L'enseignante n'oubliera pas de rappeler aux enfants l'initialisation de la position des personnages.

XIV. Ensuite, l'enseignante demandera aux élèves comment ils sont arrivés à résoudre le problème donné et à faire faire les personnages Scratch et Tac ce qui a été demandé par l'énoncé du problème. L'enseignante demandera de lui faire des propositions sur comment résoudre ce problème et elle les réalisera ou elle demandera aux élèves de venir les réaliser sur sa tablette et puis à l'aide des questions appropriées elle leur aidera à comprendre comment utiliser les messages pour résoudre ce problème. Chaque fois qu'ils font un essai, ils valideront tous ensemble s'ils ont bien programmé ou pas en faisant une comparaison du résultat obtenu et de l'énoncé du problème. L'objectif de cette stratégie didactique est que les élèves arrivent à comprendre la raison pour laquelle les messages sont utiles pour résoudre des problèmes comme celui-ci, à l'aide des questions et des essais différents.

XV. Ensuite, l'enseignante demandera aux élèves **de déclencher les programmes des deux personnages dans la scène.** Si les élèves ne se souviennent pas de la fonctionnalité de **l'icône du drapeau vert** en haut, l'enseignante la leur rappellera.

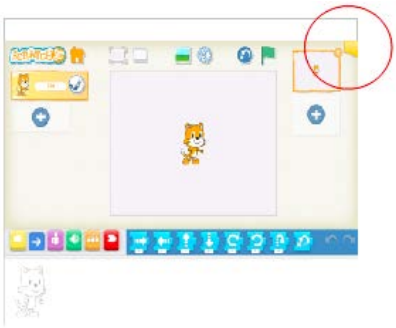
Lorsqu'on clique sur le drapeau vert en haut de la scène, les deux personnages déclenchent leurs programmes en même temps. Essayez d'appuyer sur le drapeau vert en haut. Est-ce que ça marche ? *Les programmes de vos* personnages se déclenchent ou pas ?

XVI. Puis, l'enseignante demandera aux élèves de **projeter leur projet en plein-écran,** comme un dessin-animé. Si les élèves ne se souviennent pas de l'icône pour réaliser cela, l'enseignante la leur rappellera.


XVII. À la fin de cette séance il faut que les élèves comprennent que :

- Les deux personnages communiquent à l'aide de messages.
- Les deux personnages ne déclenchent pas leurs programmes en même temps.
- Le personnage qui a le drapeau vert au début de son script déclenche son programme toujours au premier.
- Le personnage qui a la commande « quand message reçu » au début de son programme, déclenche son programme après qu'il ait reçu le message par l'autre personnage.
- Le personnage qui commence son programme avec la commande « quand message reçu » ne déclenchera jamais son programme, si on enlève la commande « envoyer un message » du programme du personnage qui envoie le message et qui le commande à commencer son programme.

XVIII. Après, l'enseignante leur demandera d'**enregistrer leur projet**. Elle leur expliquera qu'il est nécessaire de cliquer sur le coin jaune en haut et à droite. Ensuite, il faut effacer ce qui est déjà écrit et écrire à l'aide du clavier tactile « A9iiprénom ».



XIX. Enfin, l'enseignante leur demandera d'aller à la maison pour vérifier si leur projet a bien été enregistré ou pas.

C. SEANCES D'ÉVALUATION

1èr problème d'évaluation

I. Tout d’abord, la chercheuse leur présentera le projet prêt en plein-écran et elle va leur demander d’observer ce qui se passe sur le projet.

II. 

III. Ensuite, elle leur lira l’énoncé du problème : Un papillon et un enfant se trouvent dans le paysage « parc ». Le papillon commence son programme en premier pour arriver sur le deuxième rond clair en bas. Dès que le papillon arrive sur le deuxième rond clair en bas, l'enfant commence son programme pour qu'il arrive sur le rond clair à côté de lui.

2ème problème d'évaluation

I. Tout d'abord, la chercheuse leur présentera le projet prêt en plein-écran et elle va leur demander d'observer ce qui se passe sur le projet.

II. Ensuite, elle leur lira l'énoncé du problème : Un lapin et un papillon se trouvent dans le paysage « rivière ». Les deux personnages commencent leurs programmes en même temps. Le lapin fait un pas à droite et puis il dit « Bonjour le papillon » et ces deux commandes il les fait 3 fois. Le papillon se déplace pour arriver sur la pierre à droite.