Modèles de visualisation à base d'agents

Arnaud Grignard

UPMC
SORBONNE UNIVERSITÉS

THÈSE DE DOCTORAT DE
L'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité : **Informatique**

Modèles de visualisation à base d'agents

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Arnaud GRIGNARD

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

devant le jury composé de :

M. Guillaume Beslon	Rapporteur	Professeur (INSA Lyon - LIRIS)
M. Guillaume Hutzler	Rapporteur	Maitre de Conférence (Univ Evry-Val d'Essonne)
M. Yves Demazeau	Examinateur	Directeur de Recherche (CNRS - LIG)
M. Matthieu Latapy	Examinateur	Directeur de Recherche (CNRS - LIP6)
Mme. Sophie Lavaud	Examinatrice	Chercheuse associée - Paris 1
M. Michel Morvan	Examinateur	Professeur (École Normale Supérieure de Lyon)
M. Alexis Drogoul	Directeur	Directeur de Recherche (IRD - UMI 209 UMMISCO)
M. Jean-Daniel Zucker	Co-Encadrant	Directeur de Recherche (IRD - UMI 209 UMMISCO)

Résumé

Ce mémoire part du constat que la visualisation est le parent pauvre de l'approche de modélisation à base d'agents : peu citée, peu mise en avant, elle constitue néanmoins, pour beaucoup de modélisateurs, non seulement leur premier point d'entrée vers la construction de modèles, mais aussi une façon de plus en plus pregnante, de concevoir, vérifier, voire valider des modèles de systèmes complexes.

Le domaine de la modélisation à base d'agents, longtemps peu structuré, est en train de progressivement s'organiser en termes de méthodologies de conception, de communication, de description. Le succès du protocole ODD [Grimm et al., 2010] est là pour en témoigner : face à des demandes de représentation de systèmes de plus en plus complexes, les modélisateurs ont besoin de mettre de l'ordre dans leurs façons de modéliser et ils y parviennent. Cependant, il est étonnant de constater qu'aucune place n'est réservée dans ODD, ni dans aucune autre méthodologie approchante, à la façon de visualiser le modèle décrit ou à concevoir. Pour beaucoup de théoriciens de la modélisation, cette étape n'existe tout simplement pas, ou, si elle existe, est considérée comme le lointain produit dérivé d'un modèle déjà conçu, vérifié et validé. Pourtant, l'étude des pratiques de la modélisation à base d'agents révèle tout le contraire : l'aller-retour entre l'écriture et la visualisation d'un modèle fait partie intégrante du quotidien de nombreux chercheurs, comme en témoigne le succès de la plate-forme NetLogo. Cette visualisation, partie d'une démarche intégrée, permet de façon intuitive de vérifier et raffiner aussi bien les comportements individuels des agents que les structures collectives ou émergentes attendues. Poussée à l'extrême, cette pratique se rencontre aussi dans les démarches de modélisation et/ou simulation participative, où la visualisation du modèle sert de médiation entre acteurs et de support aux tâches, collectives, de modélisation.

Absente des propositions méthodologiques, la visualisation de modèles à base d'agents se retrouve donc essentiellement délimitée et structurée par des pratiques individuelles, parfois partagées sous la forme de préceptes [Kornhauser et al., 2009], mais rarement généralisables au-delà de la conception d'un modèle. Il existe pourtant des façons de visualiser, des manières de chercher à faire ressortir une information spécifique, des méthodes à suivre pour étudier visuellement une abstraction.

Ce mémoire a comme objectif, non seulement de mettre en évidence ces méthodes, mais également de les organiser et de les opérationnaliser en utilisant un support et un bagage conceptuel identique à celui utilisé pour construire des modèles à base d'agents, afin de bénéficier des mêmes qualités de flexibilité, de modularité et d'adaptabilité que celles que l'approche à base d'agents a pu apporter à la modélisation.

Je défends dans cette thèse l'idée qu'une approche à base d'agents peut être appliquée au domaine de la visualisation d'informations. En partant de cette idée maintenant communément admise que les modèles à base d'agents offrent des représentations adaptées à la complexité d'un système réel, je propose donc dans mon travail d'adopter une approche fondée sur la définition de modèles de visualisation à base d'agents afin de faciliter la représentation visuelle d'informations et d'offrir un support innovant permettant d'explorer, programmatiquement et visuellement, leurs dynamiques sous-jacentes. Tout comme leurs contreparties logicielles, les modèles de visualisation à base d'agents sont composés d'entités graphiques autonomes, pouvant interagir et s'organiser entre elles, apprendre à partir des informations qu'elles traitent et adapter en conséquence leurs comportements et représentations visuelles. En offrant à un utilisateur la possibilité de décrire des tâches de visualisation sous cette forme, mon objectif est donc de lui permettre de bénéficier des qualités de flexibilité, modularité et adaptabilité inhérentes aux approches à base d'agents. Ces concepts ont été implémentés et expérimentés dans la plate-forme de modélisation et de simulation GAMA, au sein de laquelle j'ai développé un environnement immersif 3D offrant à l'utilisateur une grande liberté de points de vue et d'interaction avec les agents. Leur mise en oeuvre est validée sur des modèles choisis pour leurs propriétés, supports d'une progression linéaire en termes de complexité qui permet de mettre en relief ces notions de flexibilité, de modularité et d'adaptabilité de l'approche visant à améliorer la modélisation des systèmes complexes. Je montre enfin, sur le cas particulier de la visualisation de données, comment mon approche permet, en temps réel, de représenter, d'expliciter, voire de découvrir leurs dynamiques.

Abstract

Information visualization is the study of interactive visual representations of abstract data to reinforce human cognition. It is very closely associated with data mining issues which allow to explore, understand and analyze phenomena, systems or data masses whose complexity continues to grow today. However, most existing visualization techniques are not suited to the exploration and understanding of datasets that consist of a large number of individual data from heterogeneous sources that share many properties with what are commonly called "complex systems". The reason is often the use of monolithic and centralized approaches. This situation is reminiscent of the modeling of complex systems (social sciences, chemistry, ecology, and many other fields) before progress represented by the generalization of agent-based approaches twenty years ago.

In this thesis, I defend the idea that the same approach can be applied with the same success to the field of information visualization. By starting from the now commonly accepted idea that the agent-based models offer appropriate representations the complexity of a real system, I propose to use an approach based on the definition of agent-based visualization models to facilitate visual representation of complex data and to provide innovative support which allows to explore, programmatically and visually, their underlying dynamics. Just like their software counterparts, agent-based visualization models are composed of autonomous graphical entities that can interact and organize themselves, learn from the data they process and as a result adapt their behavior and visual representations. By providing a user the ability to describe visualization tasks in this form, my goal is to allow them to benefit from the flexibility, modularity and adaptability inherent in agent-based approaches.

These concepts have been implemented and experimented on the GAMA modeling and simulation platform in which I developed a 3D immersive environment offering the user different point of views and way to interact with agents. Their implementation is validated on models chosen for their properties, supports a linear progression in terms of complexity, allowing us to highlight the concepts of flexibility, modularity and adaptability. Finally, I demonstrate through the particular case of data visualization, how my approach allows, in real time, to represent, to clarify, or even discover their dynamics and how that progress in terms of visualization can contributing,in turn, to improve the modeling of complex systems.

Remerciements

Je tiens à remercier Guillaume Beslon pour avoir accepté de rapporter cette thèse. Bien que nos disciplines soient éloignées, je lui dois mon goût pour l'outil de découverte scientifique qu'est la modélisation et pour l'interdisciplinarité qu'il a toujours pronée dans ses travaux.

Je tiens également à remercier Guillaume Hutzler de rapporter ce travail. Il est pour moi un des précurseurs dans le domaine de la visualisation à base d'agents. Ses différents travaux notamment sa thèse de doctorat "Du Jardin des Hasards aux Jardins des Données" ont été une source constante d'inspiration pendant ma thèse.

Michel Morvan, en tant que co-fondateur de l'Institut Rhône-Alpin des Systèmes Complexes et de CoSMo Company, deux lieux dans lesquels j'ai travaillé avant de commencer ma thèse, a joué indirectement un grand rôle dans ma fascination pour la modélisation de systèmes complexes.

Je remercie Matthieu Latapy d'avoir accepté de faire partie de mon jury, et j'espère que la visualisation d'informations et le fait de voir des systèmes complexes par une autre approche que l'approche réseaux piqueront sa curiosité. Je remercie également Yves Demazeau d'apporter son experience en tant que membre du jury dans la visualisation à base d'agents, domaine dans lequel il a déjà largement contribué à l'IGN. Je remercie enfin Sophie Lavaud qui en plus d'apporter une touche de féminité dans ce jury saura apporter sa touche artistique en tant que chercheuse en art numérique interactif.

Je tiens à remercier tout particulièrement Alexis Drogoul qui a été un directeur de thèse plus qu'idéal. J'espère avoir l'occasion de partager avec lui des moments aussi privilégiés que ceux que nous avons passés au Vietnam. Il m'a transmis énormément de choses durant ces trois ans, tant sur le plan scientifique que sur le plan humain. Je le remercie pour la confiance et la liberté qu'il m'a offert durant cette thèse. Son soutien a été précieux dans certains moments difficiles et j'espère qu'il sera fier de cet ouvrage. En revanche, il n'est pas impossible qu'il m'ait donné la (mauvaise ?) habitude de considérer que tout peut être vu sous forme d'agents.

Jean-Daniel Zucker, en tant que co-encadrant, a été d'une aide précieuse tout au long de cette thèse. Son approche pluridisciplinaire et les nombreuses voies qu'il m'a permis d'explorer ont été des accélérateurs tout au long de la thèse.

Cette thèse s'est déroulée dans le cadre du Programme Doctoral International « Modélisation des Systèmes Complexes » (PDI MSC), une grande partie de la thèse a été effectuée au sein des laboratoires de l'UMMISCO-Vietnam, l'IFI (MSI) et l'USTH (ICT-Lab) à Hanoï, je remercie donc particulièrement Ho Tuong Vinh, Nguyen Hong Quang et Remi Mullot de m'avoir accueilli au sein de leur laboratoire.

Comme vous le verrez au cours de cette thèse, une grande partie des travaux a été réalisée à l'aide de la plateforme Gama. J'ai eu la chance de participer à cette belle aventure qu'est le développement d'une plateforme de simulation open source dans une équipe de passionnés. Je remercie tout particulièrement Patrick Taillandier, Benoit Gaudou, Nicolas Marilleau, Vo Duc An et Huynh Nghi pour m'avoir donné de précieux coups de pouce dans les phases de développement de GAMA. Cette aventure a été parfois dure notamment lors des interminables sessions de debuggage mais elle a aussi permis de nouer des relations fortes avec toute l'équipe lors de coding camp Vietnamien avec entre autres Javier Gil-Quijano, Samuel Thiriot et Philippe Caillou.

Je tiens à remercier Christophe Cambier et les promotions PDI 2010, 2011, 2012 avec qui nous avons partagé des moments inoubliables lors des doctoriales organisées chaque année au centre IRD Bondy.

Au sein de l'IXXI j'ai aussi eu la chance de travailler avec Eric Boix qui m'a initié à la modélisation des systèmes complexes au sein du projet Dynanets. Je remercie aussi Tri Nguyen-Huu pour son soutien pendant les heures tardives passées au laboratoire lors de la rédaction et pour sa contribution mathématique à la modélisation à base d'agents.

Je dois d'immenses remerciements à mes parents, Pierre et Sylvie, ainsi qu'à mes quatre sœurs Charlotte, Juliette, Margot et Emilie pour leur soutien tout au long de cette thèse.

Je tiens à remercier Marie-Florine qui a su être présente et à l'écoute quotidiennement durant la quasi totalité de cette thèse.

Je tiens à faire savoir à tous mes amis proches l'importance qu'ils ont eu durant cette période et plus particulièrement Timothée, Guillaume, Timat, Tom, Flo et Rémi qui sont les rares personnes extérieures avec qui j'ai pu échanger sur le contenu de ma thèse.

Enfin j'ai une pensée particulière pour mes neveux Pauline, Roméo et Fanette qui sont nés pendant cette thèse et pour mon arrière grand-père, Victor Grignard, prix Nobel de Chimie en 1912.

Table des matières

Table des figures

(Dans certaines figures, le symbole représente un lien vers une version animée.)

Plan du mémoire

La première partie de cette thèse se concentre sur les modèles à base d'agents et leur visualisation. Ces modèles, se caractérisant par leur complexité et leur dynamicité, requièrent une attention particulière dans les phases de conception, d'expérimentation et de communication des résultats. Un des buts de la modélisation à base d'agents est de révéler, par le biais d'une simulation et de sa visualisation, le comportement d'un modèle en représentant les états et comportements de chaque agent, les interactions entre agents et les propriétés émergentes de ces interactions. Nous proposons, grâce à des outils dédiés, une méthodologie adapté à la visualisation d'ABM et qui ne nécessite pas d'introduire de nouveaux paradigmes et permet de conserver les propriétés des approches à base d'agents. Le fait d'introduire des modèles de visualisation utilisant des agents graphiques dédiés permet (1) de clairement séparer les tâches de modélisation d'un phénomène des tâches de visualisation du modèle résultant ; (2) de profiter de la flexibilité de l'approche à base d'agents pour à la fois visualiser des données "réelles" et des données "issues de simulations" et comparer/valider "visuellement" les résultats de simulation aux données ; (3) de profiter de l'aspect modulaire de l'approche afin de réutiliser des modèles de visualisation pour différents modèles.

Le chapitre 1 présente les principes et possibilités de la modélisation à base d'agents afin de fournir le cadre théorique sur lequel est basée notre approche de visualisation. A travers ce chapitre nous définissons l'importance de la visualisation et nous énumérons les différentes pratiques de visualisation dans la modélisation à base d'agents. Cette énumération permet de faire émerger certains besoins auxquels tentera de répondre notre thèse.

Le chapitre 2 présente la plateforme au sein de laquelle est implémentée notre approche, l'environnement 3D immersif et le langage visuel permettant la description des modèles de visualisation développés au cours de cette thèse. Nous définissons ici l'implémentation de notre approche au sein de la plateforme de simulation GAMA. Un intérêt particulier sera porté à l'une des contributions techniques majeures de cette thèse qui est l'intégration d'un environnement 3D immersif au sein de cette plateforme. Enfin nous verrons pas à pas les étapes pour créer un modèle de visualisation à l'aide du langage GAML en introduisant les concepts de représentation d'abstraction et d'interaction nécessaires.

Le chapitre 3 illustre les contributions opérationnelles de notre thèse à travers différents exemples lies à la visualisation de modèle à base d'agents. A travers ces différents exemples nous tenterons de balayer l'ensemble des concepts utilisés en modélisation en insistant plus particulièrement sur le couplage, l'usage et la visualisation des données SIG au sein d'un modèle à base d'agents, la représentation et modélisation multi-niveaux, l'usage d'indicateurs créés à l'aide d'agrégations spatiotemporelles se superposant à la représentation du modèle de référence pour permettre une meilleure compréhension de celui-ci grâce à un retour visuel immédiat. Enfin nous montrerons comment contrôler le modèle de référence à l'aide d'abstractions.

Si la première partie de la thèse se concentre exclusivement sur la visualisation de modèles à base d'agents, nous proposons dans la deuxième partie de cette thèse, une approche plus vaste que nous désignons par le terme de visualisation d'informations.

Le chapitre 4 part de l'hypothèse que si notre approche permet de visualiser des systèmes complexes tels que les modèles à base d'agents elle peut alors s'appliquer à une multitude de domaines, y compris des modèles ou jeux de données non agents, que nous regroupons sous le terme d'informations.

Le chapitre 5 se présente comme une expérimentation de cette approche dans dans deux cas d'applications précis, ARCHEM et DENSEAT. Nous montrerons dans ces deux applications comment les concepts présentés au cours de cette thèse peuvent s'appliquer de manière originale à la visualisation d'informations.

Nous conclurons dans le chapitre 6 sur les perspectives envisagées notamment sur l'intérêt de l'approche agent à laquelle serait ajoutée des capacités d'interaction plus grande.

Contributions théoriques

Les contributions théoriques de cette thèse portent principalement sur la proposition d'une nouvelle approche de visualisation, une approche basée sur le concept de modèle de visualisation à base d'agents. Étonnamment les domaines de la modélisation (au sens de la modélisation des systèmes complexes) et de la visualisation sont deux domaines ayant encore peu d'interaction. Nous souhaitons à travers cette thèse établir un lien entre deux domaines de recherche encore relativement éloignés à savoir la modélisation informatique et la visualisation. La visualisation est souvent considérée comme la dernière étape dans le processus de modélisation pour communiquer un résultat mais elle n'est encore que trop rarement utilisée à des fins prospectives et analytiques. Parallèlement la visualisation de données et d'informations connait un essor assez manifeste ces dernières années mais avec une approche encore relativement monolithique. Le domaine de la visualisation ne semble pas encore avoir intégré les avancées majeures réalisées ces dernières années dans la modélisation des systèmes complexes. Notre approche est donc une des premières tentatives visant à réunir ces deux domaines pour à la fois faire profiter au domaine de la modélisation des avancées récentes en visualisation et inversement de faire profiter au domaine de la visualisation des avancées réalisées en modélisation.

Contributions pratiques

La principale contribution pratique est le développement de la librairie de rendu 3D en OpenGl intégrée au sein de la plateforme Gama [Grignard et al., 2013e] [Grignard et al., 2013d]. Cette librairie offre des outils interactifs pour l'exploration de modèles à base d'agents et ouvre les portes de la troisième dimension à la modélisation à base d'agents. Tout au long de la thèse, les avancées en terme de visualisation ont été appliquées dans différents projets utilisant la plateforme GAMA.

Le projet ARCHIVES [Gasmi et al., 2015], étudie les décisions prises durant les inondations de 1926 dans la ville d'Hanoi et le rôle des différents acteurs politiques impliqués. Un des défis a été la reconstruction historique 3D de la ville d'Hanoi. A la représentation de la ville, s'ajoute un modèle hydrologique permettant de simuler les inondations et une visualisation des différents messages envoyés entre les différents acteurs politiques. Dans cette approche, les données SIG sont couplées à des données historiques qu'il a fallu numériser pour ensuite pouvoir les représenter. Cette représentation intègre à la fois des données SIG, un modèle numérique de terrain, des données démographiques et des agents représentants les acteurs politiques. Un effort particulier a été apporté à la représentation du flux de messages entre acteurs politiques pendant les crues. Différents projets ont été menés en partenariat avec l'université de Can Tho sur des problématiques liées aux changements climatiques et ont aussi pu bénéficier des avancées fournies en terme de rendu réaliste 3D [Truong et al., 2011]. En France, le projet ARCHEM [Grignard et al., 2015], qui sera plus longuement développé dans le chapitre 5, vise à explorer les techniques de visualisation pour illustrer les phénomènes de sédimentation sur la rivière du Rhône. Enfin une contribution que nous développerons dans la partie 5.2.1 concernant le projet DENSEAT exploitant un jeu de données sur la maladie de la dengue avec des données récoltées sur une durée de plus de 30 ans sur toute l'Asie du sud-est.

Chapitre 1

Visualisation à base d'agents

> Nous ne raisonnons que sur des modèles.
>
> Paul Valéry

Introduction

Le chapitre 1 décrit la modélisation et la simulation à base d'agents, la pratique de la visualisation dans la modélisation à base d'agents et les objectifs auxquels cette thèse répond tout au long de ce mémoire. La partie 1.1 présente les concepts et potentialités de la modélisation à base d'agents afin de fournir le cadre théorique sur lequel est basée notre approche de visualisation à base d'agents. La partie 1.2 vise, à partir d'une description du rôle de la visualisation et de ses différentes pratiques dans la modélisation à base d'agents, à montrer qu'il manque à l'heure actuelle de méthodologies et de techniques opérationnelles pour la visualisation à base d'agents. Nous proposons ainsi dans la partie 1.3 de fournir, grâce à des outils dédiés, une méthodologie et des techniques adaptées à la visualisation de modèles à base d'agents ne nécessitant pas d'introduire de nouveaux paradigmes afin de conserver les propriétés "agents". Nous défendons ici l'idée selon laquelle une approche de ce type permet (1) d'améliorer la visualisation des modèles ABM, (2) de générer facilement et naturellement des mécanismes d'abstraction, (3) de favoriser la création d'outils d'analyse en temps réel, (4) de varier les points de vue de façon dynamique et enfin (5) de séparer la tâche de modélisation d'un phénomène à la tâche de visualisation de ce phénomène.

1.1 Modélisation et simulation à base d'agents

Lorsqu'il désire étudier un phénomène particulier, le chercheur dispose de plusieurs stratégies de recherche. Il peut collecter des données sur ce phénomène et les étudier à l'aide d'outils statistiques, il peut mener des expériences "réelles" en laboratoire, et enfin il peut tenter de recréer le phénomène à l'aide d'une simulation informatique. La simulation informatique est une démarche scientifique qui consiste à réaliser une reproduction artificielle, appelée modèle, d'un phénomène réel que l'on désire étudier [Drogoul, 1993]. La simulation est utilisée (1) lorsque la démarche analytique proposée par les modèles statistiques ou mathématiques est insuffisante, (2) lorsque l'on désire imiter les mécanismes d'un processus réel qui sont difficiles à reproduire de façon expérimentales ou à mettre en équation ou (3) lorsque l'on souhaite élaborer une théorie et que l'on ne dispose pas de connaissances assez solides sur celle-ci. L'étape de modélisation, l'étape d'expérimentation et l'étape de validation sont les trois principales étapes de la simulation numérique.

Dans ce mémoire, nous nous intéressons à un type bien particulier de simulation numérique, la simulation à base d'agents. Cette méthode de simulation numérique consiste à représenter de façon explicite des entités et leurs comportements sous une forme informatique. Ces objets informatiques peuvent représenter une multitude de phénomènes allant de la particule, aux cellules, aux individus et enfin aux groupes. Ainsi un modèle à base d'agents offre une grande flexibilité par rapport aux nombreux formalismes qui peuvent co-exister [Varenne, 2011]. Ces entités, possédant leur propre comportement et qui sont appelées agents, forment un modèle qui, une fois simulé, peut conduire à l'émergence d'une structure globale. Les modèles à base d'agents se caractérisent par leur complexité, leur dynamisme, leur hétérogénéité et leur caractère multi-échelles [Railsback and Grimm, 2011].

Nous considérons donc un **modèle** comme un programme informatique et une **simulation** comme l'exécution de ce programme informatique. La conception de ce programme, et donc de ce modèle, est désignée par le terme **modélisation**. Une **expérimentation** désigne un ensemble de simulations. La modélisation à base d'agents sera parfois désignée par le sigle **ABM** (Agent-Based Model), la simulation à base d'agents par le sigle **ABS** (Agent-Based Simulation) et enfin le sigle **ABV** (Agent-Based Visualization) sera utilisé pour la visualisation à base d'agents.

Historique

La modélisation à base d'agents s'est développée en même temps que les notions d'algorithmique distribuée et d'intelligence artificielle [McCarthy and Hayes, 1968] (elle même issue de la cybernétique [Wiener et al., 1948]). De ces racines naissent l'IAD (Intelligence Artificielle Distribuée) et les SMA (Systèmes Multi-Agents) permettant d'étudier, à l'aide d'outils informatiques, les structures émergentes de leurs interactions virtuelles. Les modèles formels de biologie quantitative de la morphogénèse [Hallé et al., 1978] [de Reffye et al., 1988] sont les premiers à utiliser la simulation informatique. [Schelling, 1971] est un des premiers à utiliser des processus générateurs afin d'étudier la ségrégation sociale. Il existe plusieurs façons de nommer cette méthode de modélisation. Parmi elles, les termes "à base d'agents" [Parunak et al., 1998], "orientée-objet et agent" [Uhrmacher et al., 1997], "particulaire" [Schweitzer, 1997]), "modélisation agents" [Macal and North, 2009], "individu-centrée [Railsback et al., 2006], "holonique" [Koestler, 1967] ou enfin "multi-agents" [Drogoul et al., 2003].

Domaines d'applications

La modélisation à base d'agents est utilisée dans de nombreuses disciplines comme l'économie [Farmer and Foley, 2009] [Krugman and Krugman, 1996], la finance [Mathieu et al., 2005], les sciences politiques [Cederman, 2005], la géographie [O'Sullivan, 2008], la criminologie [Birks et al., 2012], l'épidémiologie [Auchincloss and Roux, 2008] [Bouchaud, 2013], la psychologie sociale [Smith and Conrey, 2007], la démographie [Billari and Prskawetz, 2003] ou la biologie [Thorne et al., 2007] [Soula et al., 2005]. Cette multitude de définitions et d'applications utilisant différentes techniques provenant de différentes disciplines en fait un domaine pour lequel il est difficile de fournir une définition universelle.

Définition

Nous rappelons la définition proposée par [Treuil et al., 2008] d'un modèle à base d'agents ainsi que sa représentation graphique dans la figure 1.1.

Tout modèle à base d'agents est un système composé d'entités multiples ou agents qui évoluent dans un environnement, conçu comme une entité particulière, dans lequel ils sont localisés. Ces agents sont dotés d'attributs, de comportements, et de capacités de perception et de communication. L'ensemble des valeurs des attributs d'une entité à un instant donné constitue l'état de cette entité, et la réunion de l'ensemble des états des entités forme l'état microscopique ou - dit plus simplement- l'état du système. Les capacités de perception des entités leur permettent de consulter un sous-ensemble de cet état microscopique, habituellement de façon localisée dans l'environnement. Les comportements sont des règles contrôlant à chaque instant l'évolution de cet état, en intervenant sur les états des entités qui les portent ou sur leur existence même (création et destruction), ainsi que sur les états et existences des autres entités intervenant dans les éventuelles actions, communications ou interactions décrites dans les comportements [Treuil et al., 2008].

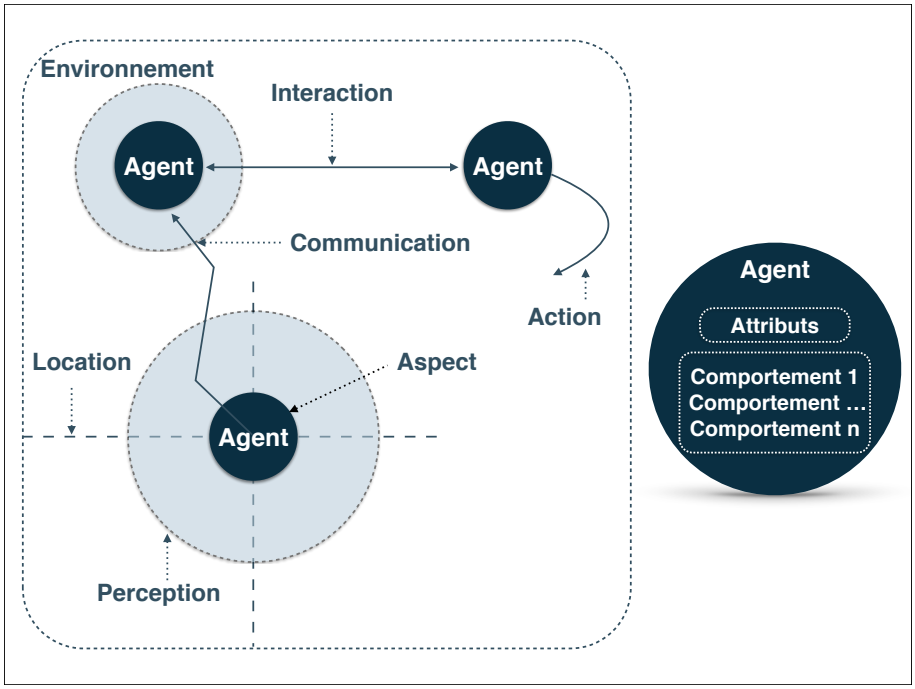

FIGURE 1.1 – Représentation graphique de la définition d'un modèle à base d'agents proposée par [Treuil et al., 2008]

Émergence : du micro au macro

Un des apports majeurs de la modélisation à base d'agents est le potentiel qu'elle offre pour étudier les phénomènes d'émergence. Cette technique de modélisation permet ainsi d'éviter que se crée un décalage entre le niveau d'abstraction utilisé pour décrire globalement un phénomène et sa description atomique. A l'inverse de la modélisation mathématique, se basant sur des grandeurs agrégées, l'approche à base d'agents permet de représenter et de mettre en relation des niveaux de détails et d'analyses beaucoup plus fins. A l'aide de processus consistant à répéter une suite de règles et d'interactions entre entités, la simulation à base d'agents permet, à partir d'une description microscopique, de générer des macro-structures et donc d'expliquer des phénomènes génératifs [Epstein, 2006]. Dans la figure 1.2, le comportement de chaque agent, initialement placés aléatoirement, consistant à rejoindre le centre de l'environnement tout en se repoussant les uns les autres, fait émerger une structure circulaire globale. On parle alors de résultats agrégés, d'émergences ou de macro-structures.

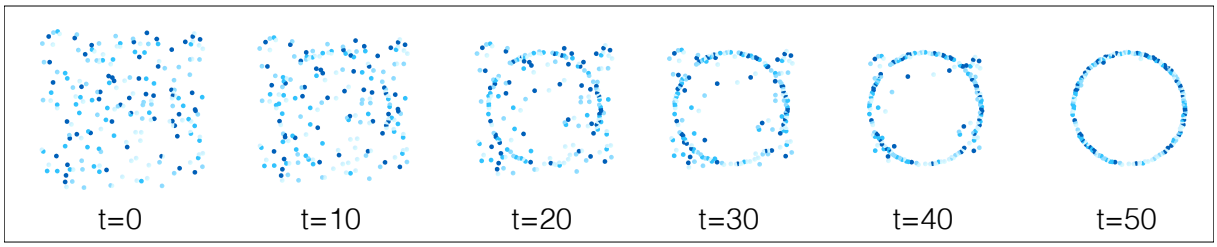

FIGURE 1.2 – Émergence : Les agents placés initialement de façon aléatoire (t=0) créent une forme circulaire au cours de la simulation (t=50).

Simulation et données

La simulation est un instrument de théorisation capable de produire du savoir [Beisbart, 2012] et donc des données. Il est donc nécessaire de mettre en relation simulation et données empiriques afin de valider la robustesse d'un modèle [Manzo, 2014] [Muldoon, 2007] en comparant les données brutes et les données générées par la simulation avec des techniques d'analyses classiques [Hamilton, 1994] [Pandit et al., 1983] [Janssen and Ostrom, 2006]. Une des priorités est de développer de nouveaux outils permettant d'exploiter la richesse de ces données [Venturini et al., 2015].

1.2 La visualisation dans la modélisation à base d'agents

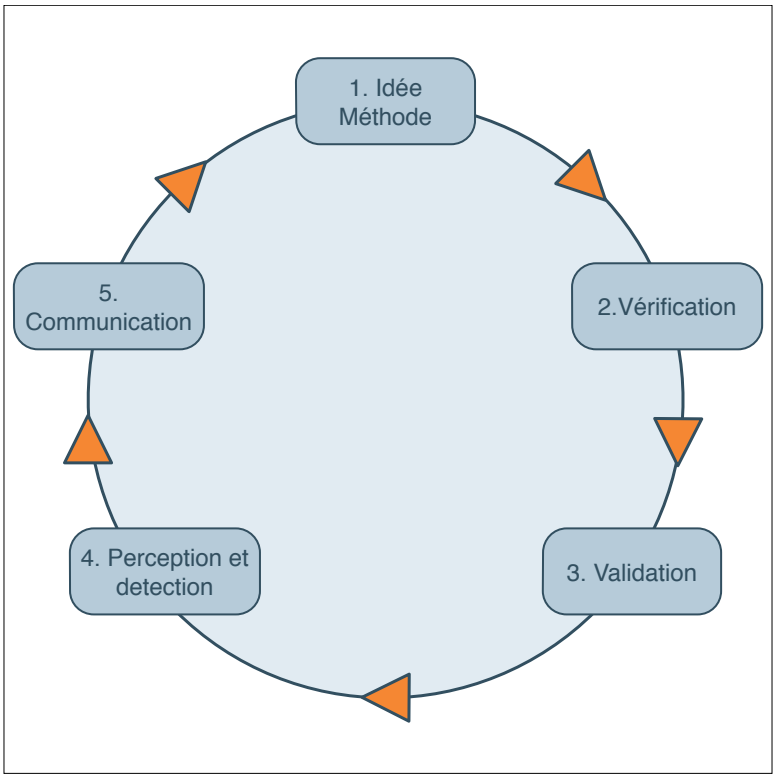

FIGURE 1.3 – Pratique de la visualisation dans la modélisation à base d'agents (inspiré du cycle de modélisation de [Railsback and Grimm, 2011].

La simulation consiste à animer le modèle, à le mettre en action en faisant le passage de la description du modèle au calcul du modèle et à sa visualisation afin de donner une représentation du phénomène modélisé. Les modèles devenant de plus en plus complexes, il est alors nécessaire de représenter et analyser ces derniers à l'aide de nouvelles techniques. La visualisation est identifiée par [Crooks et al., 2008] et [Dorin and Geard, 2014] comme étant un des principaux défis de la recherche en ABM.

Représentation

Un des buts de la modélisation à base d'agents est de montrer par le biais d'une simulation et de sa visualisation le comportement d'un modèle. Comme le montre la figure 1.4, la visualisation permet de représenter les états et comportements d'un agent, les interactions entre agents, les agrégations de population, les propriétés émergentes, les processus et attributs de l'environnement et enfin les interactions entre les agents et l'environnement.

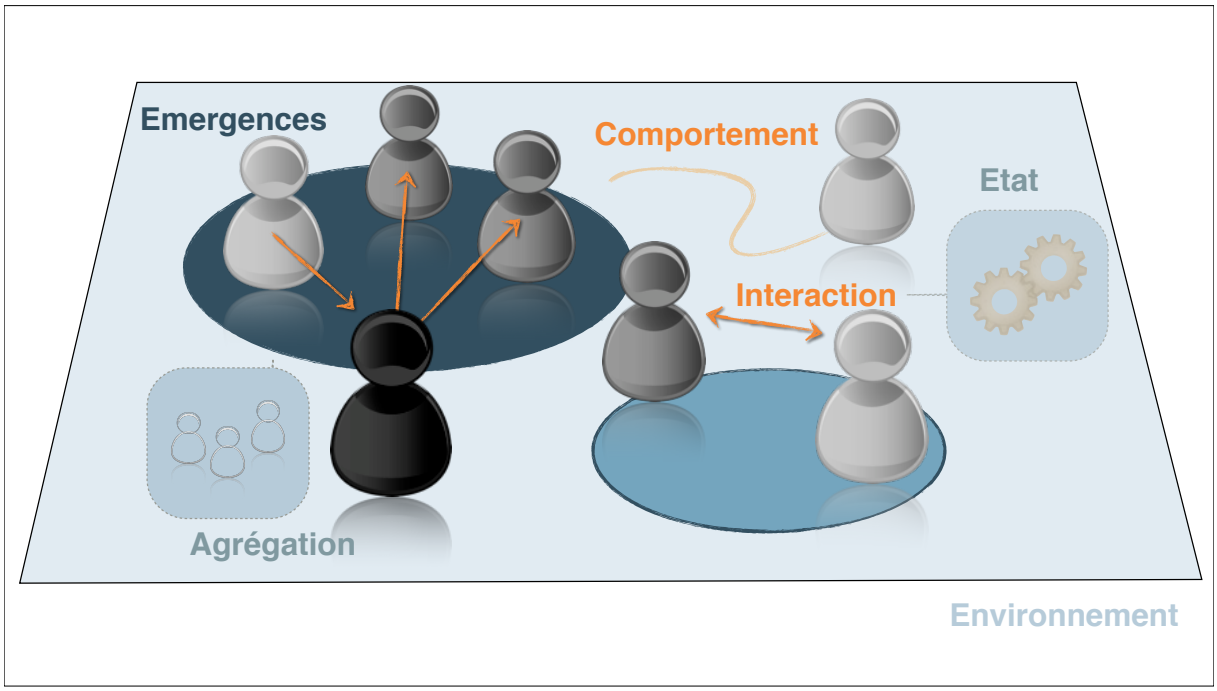

FIGURE 1.4 – Représentation schématique d'un modèle à base d'agents

Les modèles ABM doivent avoir une représentation juste et illustrer clairement le comportement et les dynamiques [Kornhauser et al., 2009]. Un modèle ayant des règles de comportement compliquées peut être facile à comprendre avec une visualisation adaptée. Inversement, un modèle simple peut s'avérer difficile à comprendre à cause d'une visualisation mal définie. La difficulté dans la représentation des modèles à base d'agents outre le fait qu'elle fait appel à des notions d'esthétisme [Pipes, 2008] [Maeda and Foreword By-Antonelli, 2001] et de cognition [Wertheimer, 1938], réside dans le caractère interdisciplinaire et non conventionnel de la gamme des modèles et donc de leurs représentations graphiques.

Vérification et validation de modèles

Certaines propriétés sont simples à valider comme la présence ou non d'un agent au sein d'une simulation, d'autres, plus compliquées, comme la présence ou non d'interactions entre agents sont beaucoup plus faciles à découvrir à l'aide d'une visualisation. La visualisation permet aussi repérer immédiatement des comportements d'agents anormaux se déplaçant de façon incorrecte ou s'écartant de façon anormale de son point initial. Enfin, la visualisation est parfois l'outil le plus efficace pour garantir qu'un algorithme reproduit correctement un phénomène [1].

Perception et détection de patterns dans une simulation

Une des difficultés dans l'observation des modèles à base d'agents est que les entités et comportements émergents (résultant des interactions entre des agents ayant à la base des comportements parfois très simples) ne sont pas définis explicitement dans le modèle. La détection de ces structures émergentes peut avoir lieu de façon *online* (pendant la simulation) ou de façon *offline* (post-simulation). Pour l'analyse *offline*, les techniques de fouille de données permettent d'extraire des connaissances de ces données. A l'exécution, l'utilisateur peut ajuster certains paramètres pouvant favoriser l'émergence de structures pendant la simulation [Sollenberger et al., 2005]. Cette capacité à détecter et à représenter des structures émergentes dans une simulation permet aussi de définir plus facilement différents niveaux d'abstraction dans le cas de la modélisation multi-niveaux [Servat et al., 1998].

Communication et dissémination

Dans beaucoup de cas, quand un modèle est publié, les visualisations ne le sont pas forcément. Elles ne le sont que lorsque la visualisation est elle-même le résultat à communiquer. Comme beaucoup de modèles ABM sont des modèles spatiaux, il est plus naturel de fournir des représentations comme dans le cas des automates cellulaires par exemple. La dynamique d'un comportement peut être transmise en rejouant une simulation ou par le biais d'une vidéo même si ce genre de média deviendra inutilisable en impression.

1. Visualisation de l'algorithme Mitchell's Best-Candidate `https://youtu.be/GAFYMg1VNgI`

Compromis entre représentation et interaction

Représentation

Il est important de trouver la bonne limite entre la représentation stricte du modèle initial et l'image, la vidéo ou l'interaction que l'on propose à l'utilisateur lors de la simulation. Pour [Beslon, 2008], il faut faire attention à ne pas piéger l'utilisateur dans une description phénoménologique pouvant cacher la réalité du phénomène modélisé en conservant dans la mesure du possible la visualisation en tant qu'instrument prolongeant un modèle et se concentrant sur le modèle. Cependant nous pouvons aussi tenir un discours inverse à la manière de Bret Victor dans son approche systémique de la visualisation [Victor, 2012] en soutenant que seules la visualisation et l'interaction permettent une exploration et une compréhension du phénomène modélisé grâce aux outils d'interaction pour le contrôle des paramètres et d'outils visuels pour la construction d'abstraction. Ces différentes propriétés sont parfois regroupées dans le terme de *learnable programming*[2] qui permet de concevoir de façon plus naturelle et interactive des algorithmes. Cette approche permet d'appréhender la visualisation de façon dynamique[3] et d'utiliser ces outils visuels pour "penser l'impensable" [4]

Interaction

Dans une approche purement expérimentale de modélisation il peut paraitre dangereux, voire interdit de modifier, à l'aide d'une interaction homme-machine, le modèle de référence. En revanche, il nous parait important de préciser que cette interaction n'est pas interdite dans certains cas très particuliers comme la simulation participative où l'objectif n'est pas d'obtenir un modèle mais d'utiliser le modèle pour supporter des négociations, des réflexions, etc. Nous ne prendrons pas position quant au mauvais ou bon usage de l'interaction sur un modèle de référence, mais ce mémoire, en considérant les différents usages des modèles à base d'agents, s'interroge essentiellement sur les défis qu'ils posent en terme de visualisation. Ainsi notre approche vise à expliciter ce qui est "complexe" dans l'appréhension des modèles par différentes catégories d'acteurs et pourquoi des outils d'aide à la visualisation sont nécessaires.

2. http://worrydream.com/LearnableProgramming/
3. https://vimeo.com/66085662
4. https://vimeo.com/67076984

1.3 Objectifs : Vers des simulations augmentées

1.3.1 Séparer l'exécution de la représentation

Dans une simulation, on cherche souvent à obtenir des structures de plus haut niveau et on cherche parfois à les obtenir sans les créer explicitement dans le modèle de référence. Différencier le modèle de référence du modèle de visualisation permet alors de "montrer " ces structures de plus haut niveau sans les mettre dans le modèle de référence. Représenter des structures émergentes au cours de la simulation à l'aide de modèles de visualisation nécessite de clairement séparer l'exécution d'une simulation de ses visualisations. Certaines techniques ont été développées en se basant sur la notion de séparation des préoccupations (*Separation of Concern*) préconisée par [Pfaff, 1985] [5]. Ce procédé assure que la visualisation d'une simulation et l'interaction que peut avoir l'utilisateur sont des processus indépendants ne modifiant pas et n'influant pas le modèle de référence [Grignard et al., 2013b].

A l'heure actuelle les modèles de visualisation, lorsqu'ils existent, sont dépendants des modèles qu'ils représentent. Il est difficile de créer des agents spécifiquement dédiés à des tâches de visualisation. Or la tâche de visualisation est indépendante du modèle de référence. Dans un modèle où des agents représentent des points sur une carte, dans certains cas, ces points peuvent se superposer et il peut être intéressant de fusionner ces points pour une meilleure visibilité. Dans le modèle de référence décrire un comportement consistant à faire fusionner les points se superposant s'éloignerait de la description du modèle de référence. L'ajout de modèle de visualisation permet alors de décrire des comportements graphiques visant à reproduire visuellement le modèle étudié ou le rendre plus compréhensible.

Notre approche dépasse la simple visualisation de simulation puisque l'information affichée est le résultat d'algorithmes d'extraction effectués sur le flux de données à partir de la simulation de référence. Par analogie avec la réalité augmentée, consistant à superposer un modèle virtuel à la réalité [Furht, 2011], la notion de simulation augmentée consiste à superposer un modèle virtuel au modèle de référence comme le montre la figure 1.5.

5. Parmi eux, le modèle MVC (Modèle, Vue, Contrôleur [Goldberg, 1984]). Dans le monde agent, les modèles PAC [Coutaz, 1987], ALV [Hill, 1992], CNUCE [Paterno et al., 1995] et YORK [Duke and Harrison, 1993]

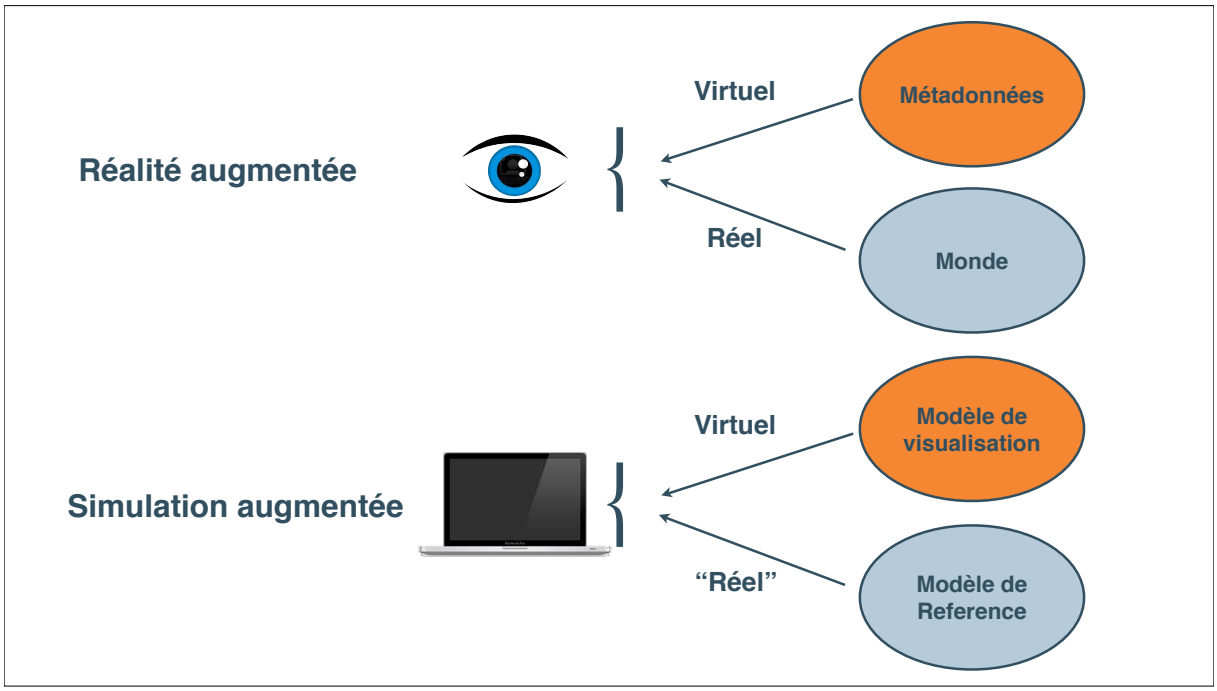

FIGURE 1.5 – Analogie entre réalité augmentée et simulation augmentée

Ces simulations augmentées offrent un retour visuel sur le modèle de référence à l'aide d'agents graphiques se superposant à la représentation du modèle de référence. Une vue est alors considérée comme un nouveau modèle avec lequel l'utilisateur peut interagir graphiquement. Le modèle de référence n'est pas conscient que plusieurs modèles de visualisation peuvent co-exister à différents niveaux d'abstraction au dessus de lui [Grignard et al., 2013c].

1.3.2 Le modèle de visualisation dans le cycle de modélisation

Comme le montre la figure 1.6, la tâche de modélisation, qui consiste à créer un ou plusieurs modèle(s) afin de répondre à une question précise, passe par la création de différents type de modèles (modèle conceptuel, modèle de données, modèle de calcul, modèle de connaissance, modèle de développement, modèle simulable, modèle statistique, etc.). Pour répondre à cette question, différentes couches de modèles sont développées, ces modèles peuvent interagir entre eux et le modélisateur ne cesse de passer d'un modèle à l'autre.

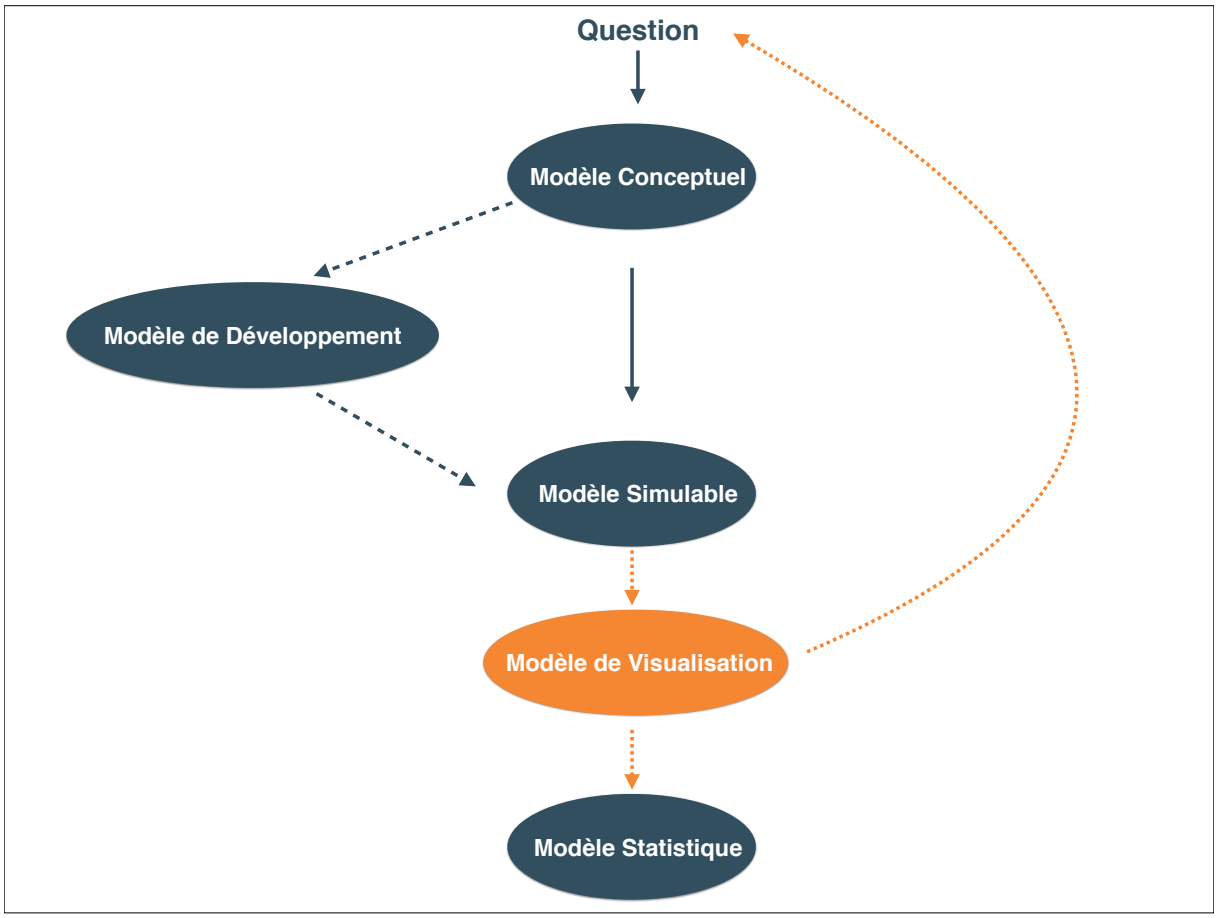

FIGURE 1.6 – Au sein du processus global de modélisation le modèle de visualisation et ses différentes interactions sont représentés en orange. Le modèle de visualisation peut se suffire à lui même, ou alors, comme le montre les flèches en pointillé, nourrir un modèle statistique ou le modèle de référence si la visualisation a permis d'extraire de la connaissance.

Le modèle de visualisation permet de comprendre et de représenter le modèle de référence et d'interagir avec. Cet outil d'exploration du comportement du modèle permet d'extraire visuellement les variables pertinentes pour éventuellement nourrir des modèles statistiques ou extraire une certaine connaissance permettant d'améliorer la description du modèle de référence. On se place alors dans une dynamique de *data mining* [Han et al., 2006] à l'aide d'outils de visualisation où le modèle de visualisation joue son rôle de médiateur entre un objet et une théorie et s'inscrit aussi dans la définition suivante de [Minsky, 1965] : " *To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A*".

Un modèle de visualisation à base d'agents est composé d'entités discrètes, des agents graphiques, manipulant des règles de visualisation et d'interaction et utilisant le même paradigme que des modèles à base d'agents classiques. Grâce à leur versatilité, il est possible d'envisager la création d'agents permettant d'abstraire un certain nombre de caractéristiques à différents niveaux. La construction de modèles de visualisation permet de déclarer facilement (par des règles individuelles) des choix de visualisation, tout en laissant une latitude au système d'organiser au mieux (selon des règles d'interactions entre représentations) la représentation globale. Dans la définition proposée dans la section 1.1, les notions de visualisation et de représentation ne sont pas mentionnées, car un modèle à base d'agents classique n'impose pas nécessairement que les agents soient des entités visuelles. Nous proposons donc une modification de certains points de la définition initiale.

- *Tout modèle de visualisation à base d'agents est un système composé d'entités multiples ou agents qui évoluent dans un environnement* ***graphique*** *dans lequel ils sont localisés et* ***représentés****.*
- *Ces agents sont dotés d'attributs* ***graphiques****, de méthodes* ***graphiques****.*
- *Un agent peut être représenté de plusieurs façons à l'aide de plusieurs aspects déclarés dans le même environnement graphique ou dans différents environnements graphiques.*

La représentation d'un agent peut alors être modifiée au cours de la simulation de la même manière que dans les techniques de visualisation agile [Heer and Bostock, 2010] [Meyer et al., 2006]. A la différence d'un modèle agents classique, lorsque que l'on "lance" un modèle ou qu'on le simule, on ne cherche pas forcément à recréer un phénomène réel (le modèle de référence) mais à accomplir une tâche de visualisation à l'aide d'une simulation à base d'agents. L'utilisation de modèle de visualisation à base d'agents permet de créer/supprimer au cours de la simulation des agents graphiques, de définir plusieurs aspects pour un même agent, plusieurs affichages et donc plusieurs points de vue sur une simulation ou même sur plusieurs simulations en parallèle.

Conclusion

Ce chapitre nous a permis de conceptualiser la notion de modèles de visualisation à base d'agents. Nous avons introduit la modélisation à base d'agents en insistant sur les points clés comme l'émergence et le lien fort existant entre simulation et données. En nous appuyant sur la littérature existante, nous avons ensuite identifié les différentes phases au cours desquelles la visualisation joue un rôle important dans le cycle de modélisation. A travers ces différents exemples, il apparait clairement qu'il manque à l'heure actuelle d'outils permettant de remédier à ces différents besoins. Nous avons donc exposé l'objectif auquel nous répondrons au cours de ce mémoire, à savoir proposer une approche basée sur la création de modèles de visualisation composés d'entités discrètes que nous appelons agents graphiques, avec lesquels l'utilisateur peut interagir sans modifier le modèle de référence. Il est important de proposer différents points de vue lorsqu'un un système ne peut être observé de façon exhaustive. Ainsi, contrairement à l'approche classique de la modélisation à base d'agents dans laquelle l'aspect de l'agent est unique et intrinsèquement lié à la définition de l'agent, dans un modèle de visualisation à base d'agents, la notion d'aspect est indépendante du modèle de référence et ainsi un agent d'un modèle ABV peut prendre différentes formes graphiques au cours de la simulation. Le fait d'introduire des modèles de visualisation utilisant des agents graphiques dédiés, permet de profiter pleinement de leur capacité à percevoir le monde dans lequel ils évoluent et donc leur capacité à adapter leur représentation en fonction de cette perception. Ces agents modulaires peuvent apparaitre et disparaitre au cours de la simulation et prendre des décisions en fonction de la perception qu'ils ont de leur environnement. Nous proposons de donner, grâce à des outils dédiés, un contenu adapté à la visualisation d'ABM ne nécessitant pas d'introduire de nouveaux paradigmes et permettant de conserver les propriétés "agents". Ainsi, de la même manière que la modélisation à base d'agents a contribué à l'amélioration de la modélisation des systèmes complexes, la visualisation à base d'agents contribue elle aussi à l'amélioration de la visualisation de systèmes complexes.

Chapitre 2

Implémentation

> The difference between theory and practice is that in theory, there is no difference between theory and practice.
>
> Richard Moore

Introduction

Le chapitre 2 décrit une implémentation de l'approche proposée dans le chapitre précédent dans GAMA, une plateforme dédiée à la modélisation à base d'agents au sein d'un environnement 3D immersif. Ce chapitre expose de façon générique les concepts fondamentaux pour créer un modèle de visualisation en utilisant la syntaxe développée au cours de la thèse dans le langage GAML. La partie 2.1 étudie les différentes plateformes existantes et explique pourquoi la plateforme GAMA a été choisie. Nous définissons ici l'implémentation de notre approche et présentons en détails cette plateforme et la façon de décrire un modèle de visualisation à l'aide du langage GAML. Dans la partie 2.2, nous nous intéressons à l'une des contributions techniques majeures de cette thèse, à savoir l'intégration d'un environnement 3D immersif au sein de la plateforme. Nous développerons les notions liées à la représentation 3D, indispensables à la création de modèles de visualisation dans la partie 2.3. Enfin, dans la partie 2.4, le langage visuel, exprimé en GAML, introduira les concepts d'abstraction et d'interaction rendus possibles par l'utilisation modèle de visualisation.

2.1 Présentation de la plateforme GAMA

La création de modèles de visualisation passe par l'utilisation d'une plateforme de simulation et d'un langage informatique. Plusieurs plateformes de simulation existent [1]. Parmi ces plateformes, il est étonnant de voir à quel point la visualisation est rarement mentionnée dans les caractéristiques techniques de ces dernières. Une simple recherche du mot "*visualization*" dans un des articles les plus récents ([Kravari and Bassiliades, 2015]) regroupant la plupart des plateformes multi-agents, ne fait ressortir que 3 plateformes (MASON [Luke et al., 2005], SeSam [Klügl, 2009] et GAMA [Grignard et al., 2013e]) sur 23 plateformes proposées. Netlogo [Tisue and Wilensky, 2004] est une des plateformes les plus connues ; cependant elle reste une plateforme se limitant à la construction de modèles simples dans laquelle la visualisation joue un rôle purement représentatif et dans laquelle il est difficile de séparer le modèle de sa visualisation. La plateforme MASON [Luke et al., 2005] est une des seules plateformes proposant une séparation claire entre le modèle et la visualisation. MASON n'est cependant pas une plateforme offrant un langage dédié puisqu'il s'agit d'une librairie au même titre que Repast [North et al., 2013]. BREVE [Klein, 2003] et FRAMSTICKS [Komosiński and Ulatowski, 1999] sont deux plateformes performantes en termes de rendu 3D réaliste de simulation mais elles ne sont actuellement plus développées. FLAMEGPU [Chin et al., 2012] propose des outils de visualisation mais elle est dédiée aux applications parallèles sur carte graphique. Enfin AnyLogic [Borshchev, 2013] possède de très bonnes capacités en termes de visualisation mais cette plateforme est une plateforme propriétaire principalement dédiée à des applications industrielles rendant impossible l'intégration de nouvelles fonctionnalités dans le cadre d'un travail universitaire comme une thèse.

Le choix a été fait de développer notre approche sur la plateforme GAMA. Cette plateforme est développée depuis 2007 par plusieurs équipes de recherche sous l'égide de l'Unité Mixte Internationale de Modélisation Mathématique et Informatique des Systèmes Complexes (UMMISCO). Gama permet la construction de modèles au sein d'un environnement de développement intégré (IDE) intégrant un éditeur de code GAML (GAMA Modeling Language), langage de modélisation orienté agent. La version 1.6.1 est disponible depuis Juin 2014 et est utilisée dans de nombreux projets dont [Gaudou et al., 2013]

1. Pour un état de l'art complet sur les différentes plateformes se référer à [Kravari and Bassiliades, 2015] [Allan, 2010] et [Nikolai and Madey, 2009]

[Gasmi et al., 2015] [Drogoul et al., 2014]. Un des points forts de cette plateforme est la capacité qu'elle offre de définir des modèles à base d'agents spatialement explicites à l'aide du langage dédié GAML. Ce langage a été notre base de départ et nous avons pu le modifier pour permettre la définition de modèles de visualisation. Ainsi il a été possible d'enrichir la plateforme et son langage tout au long de la thèse. Nous tenons à préciser qu'au début de la thèse la visualisation des modèles à base d'agents était déjà possible en GAMA mais de façon relativement classique dans un environnement 2D. La plupart des notions décrites dans les parties suivantes ont pris forme au cours de la thèse. Afin de faciliter l'utilisation de modèles de visualisation à base d'agents, le choix technique a été d'intégrer totalement l'affichage au sein de la plateforme GAMA. Ainsi un modèle de visualisation doit être vu comme un des nombreux composants constituant la plateforme GAMA. Comme le montre, de manière schématique, les figures 2.1 et 2.2, la plateforme GAMA est composée de différents composants dont une fenêtre d'affichage dans laquelle pourront être placé un (ou plusieurs) modèle(s) de visualisation.

2.1.1 GAML (GAMA Modeling Language)

GAML [2] est un langage orienté agent dédié à la définition de simulation à base d'agents. Il possède beaucoup de points communs avec les langages orientés objets comme le C++, Java ou Smalltalk mais étend le concept du langage orienté objet avec d'autres concepts comme la notion de compétences (*skills*) et la possibilité de faire migrer un objet d'une classe à une autre dynamiquement. Un des points particulièrement intéressant de ce langage est la très faible différence existant entre le formalisme utilisé pour définir le modèle d'étude (à priori représenté essentiellement par des agents) et le processus expérimental mis en place autour de la simulation. C'est précisément cette caractéristique qui va nous permettre de facilement décrire des modèles de visualisation au dessus d'un modèle de référence. En général, les processus liés à la simulation ou l'expérimentation ne sont pas exprimés à l'aide d'agents et n'utilise pas le même paradigme. En GAML, une expérience, tout comme un modèle, peuvent être considérés comme des agents à part entière. Chaque simulation (ou expérimentation) est représentée par un agent, ce qui permet la définition de modèles composés eux-mêmes de modèles représentés par des micro-agents.

2. Pour plus de détails sur l'origine du langage GAML se référer au papier fondateur [Drogoul et al., 2003]

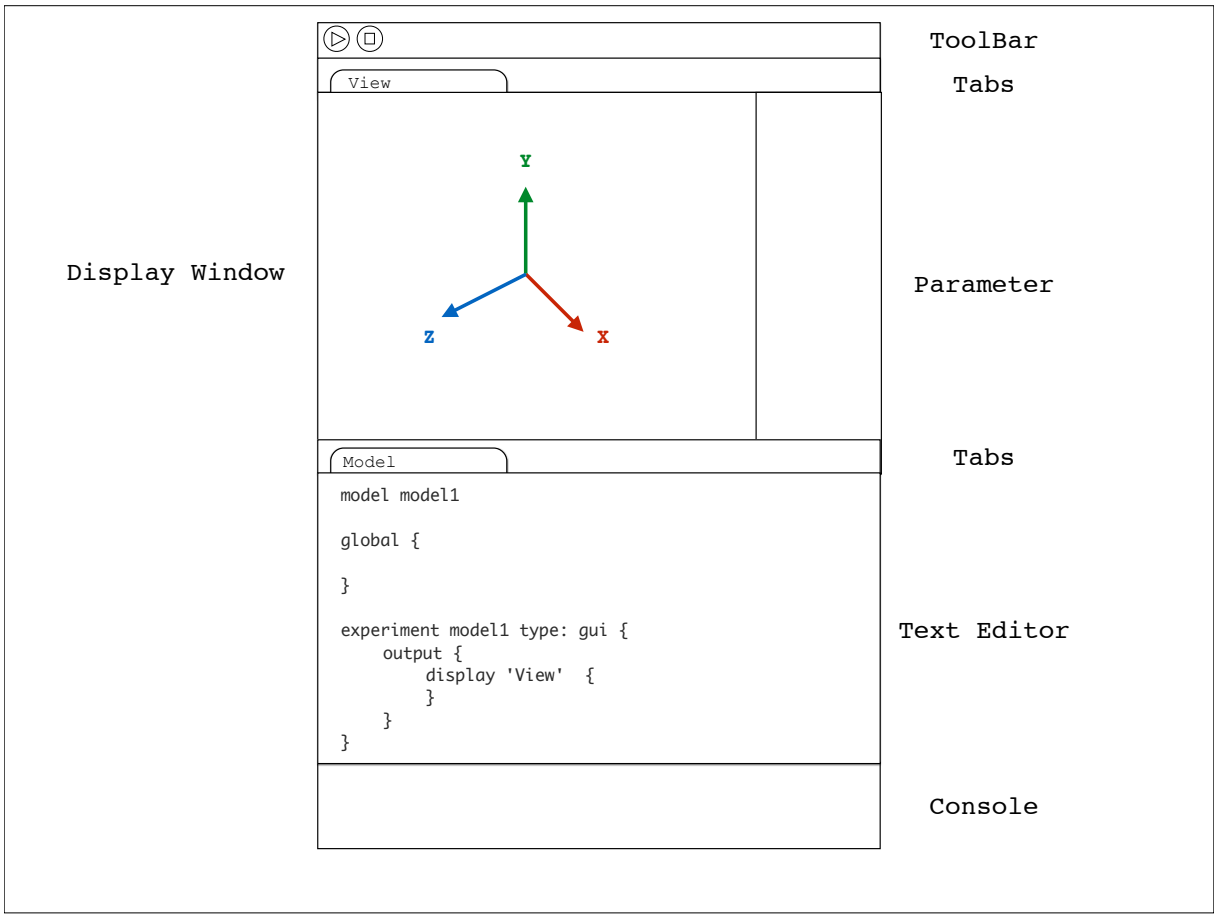

FIGURE 2.1 – Environnement schématisé de GAMA. L'environnement de développement (IDE) permet d'éditer le (ou les) modèle(s) et de le (ou les) visualisé(s) dans une ou plusieurs fenêtres d'affichage.

Un modèle décrit en GAML, représente à la fois les connaissances sur un phénomène particulier qu'un utilisateur veut simuler et la manière de le simuler. Un modèle est un fichier texte (ou un ensemble de fichiers texte se référant les uns aux autres), qui contient des instructions écrites en GAML. Un fichier écrit en langage GAML porte le nom d'extension *.gaml* et est directement interprété par GAMA. Un modèle peut alors être théoriquement édité utilisant n'importe quel logiciel de traitement de texte et plus tard chargé dans GAMA pour exécuter des expériences. Toutefois, en raison de la richesse du langage, utiliser un outil dédié (avec l'aide en ligne, la validation en direct) est clairement la meilleure façon d'écrire des modèles en GAML. L'environnement de développement intégré est composé d'un ensemble d'outils pour soutenir le modélisateur dans l'édition, la validation et la gestion de ses modèles.

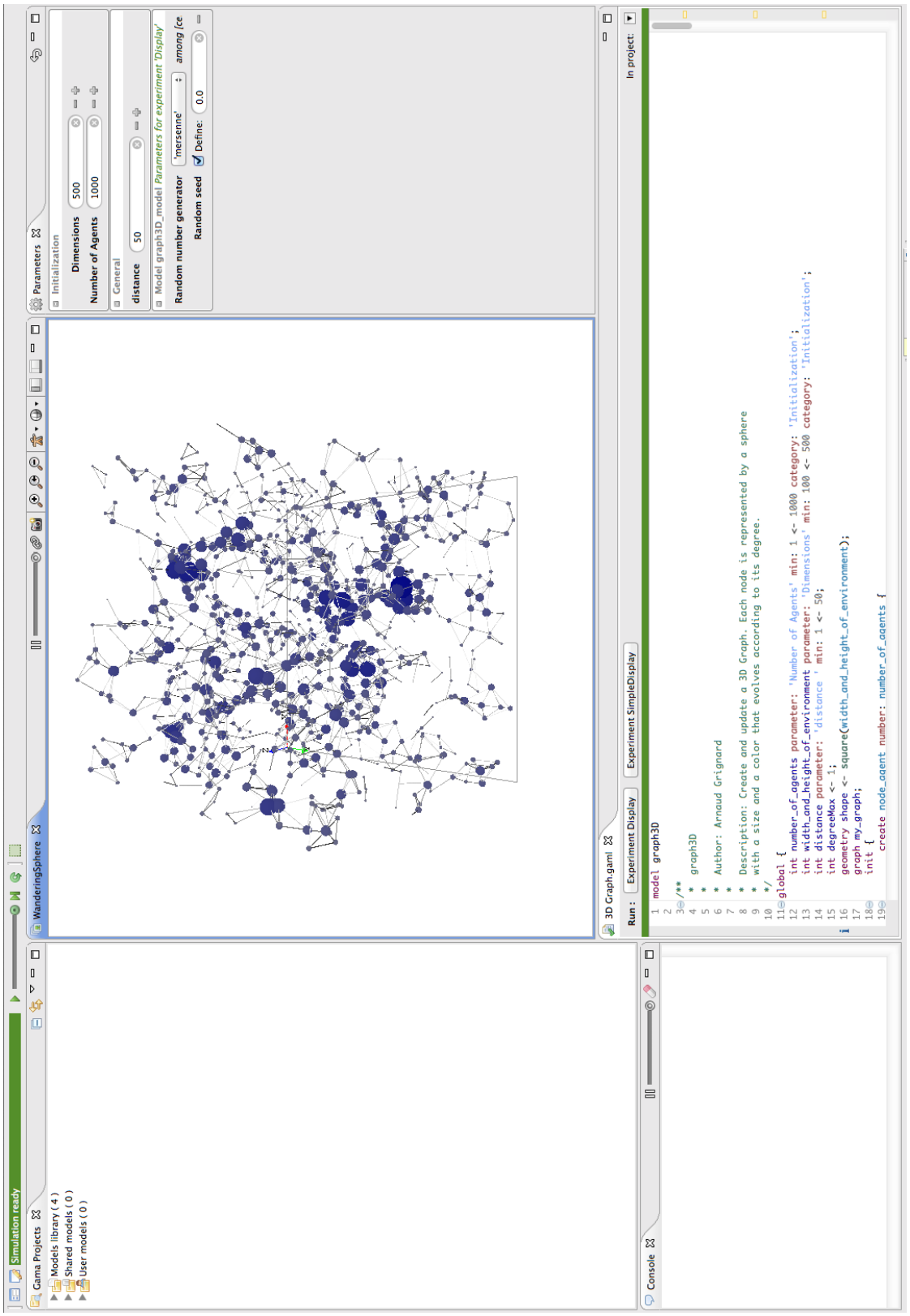

FIGURE 2.2 – Environnement de GAMA 1.6.1.

Meta-Modèle

GAML[3] permet à l'utilisateur d'écrire des modèles (`model`) étant vus comme des spécifications de simulations (`simulations`) pouvant être exécutées et contrôlées lors d'expérimentations (`experiments`).

De la même façon que dans le paradigme orienté objet, où la notion de classe est utilisée pour spécifier un objet, les agents en GAML sont spécifiés par leur espèce (`species`) ayant un ensemble d'attributs (ce qu'ils savent), un ensemble d' actions (`action`) (ce qu'ils peuvent faire) et un ensemble de comportements (`reflex`) (ce qu'ils font vraiment). Une espèce spécifie aussi des propriétés liées à sa `population` par exemple la topologie (`topology`) (la façon dont les agents sont connectés) et l'ordonnanceur (`scheduler`) (dans quel ordre et quand les instances de la population doivent s'exécuter).

Une espèce peut être imbriquée dans une autre espèce (que l'on appellera alors sa macro espèce (`macro-species`)). Dans ce cas les populations de ces instances seront automatiquement hébergées dans une instance de la macro espèce correspondante. Une espèce peut aussi hériter des propriétés d'une autre espèce (que l'on appellera alors une espèce parent (`parent species`)). Enfin une espèce peut être construite avec la notion de `skills` embarquant des attributs et actions pouvant être partagés par différentes espèces et hérités par plusieurs enfants.

Les relations entre espèce, modèle et expérimentation sont représentées dans le meta-modèle de GAML représenté en figure 2.3 composé de trois espèces abstraites appelé `agent`, `model` et `experiment`.

Syntaxe

La syntaxe GAML ressemble à celle des langages de programmation traditionnels comme Java, tout en réutilisant certaines structures de Smalltalk. Une comparaison entre GAML, Java et Netlogo est proposée dans la table 2.1.

- Un **model** est composé d'un en-tête, dans lequel il peut se référer à d'autres modèles, et une séquence de déclaration d'espèces et des expériences, sous la forme d'instructions déclaratives particulières de la langue.
- Un **statement** peut être une déclaration ou une commande. Il est toujours composé

3. Les termes entre parenthèses (`model`, `experiment`, etc) représentent la syntaxe GAML

d'un mot clé (keyword) suivi d'une expression en option, suivie d'une séquence de **facet**, chacune d'elles composée d'un mot-clé et d'une expression.

- Une **facet** permet de passer des arguments aux **statement**. Leur valeur est une expression d'un type donné. Une expression peut être une constante, le nom d'un attribut, une variable, le nom d'une unité ou d'une constante du langage.
- Un **type** peut être un type primitif, un type d'espèce ou un type paramétrique (une composition de types).
- Une déclaration d'espèce (**species**) ne peut inclure que six types de déclarations déclaratives : **attributs**, **actions**, **comportements**, **aspects**, des **équations** et des espèces imbriquées (**micro species**).

TABLE 2.1 – Comparaison des éléments de langage GAML, Java et Netlogo. Le langage GAML intègre des notions provenant de la synataxe Java et Netlogo.

GAML	**Java**	**Netlogo**
species	class	breed
micro-species	nested classes	-
parent species	superclass	-
child species	subclass	- (only from 'turtle')
model	program	model
experiment	(main) class	observer
agent	object	turtle/observer
attribute	member	'breed'-own
action	method	global function
behavior	collection of methods	collection of global functions
aspect	-	only one, mixed with the behavior
skill	interface	model
statement	statement	primitive
type	type	type
parametric type	generics	-

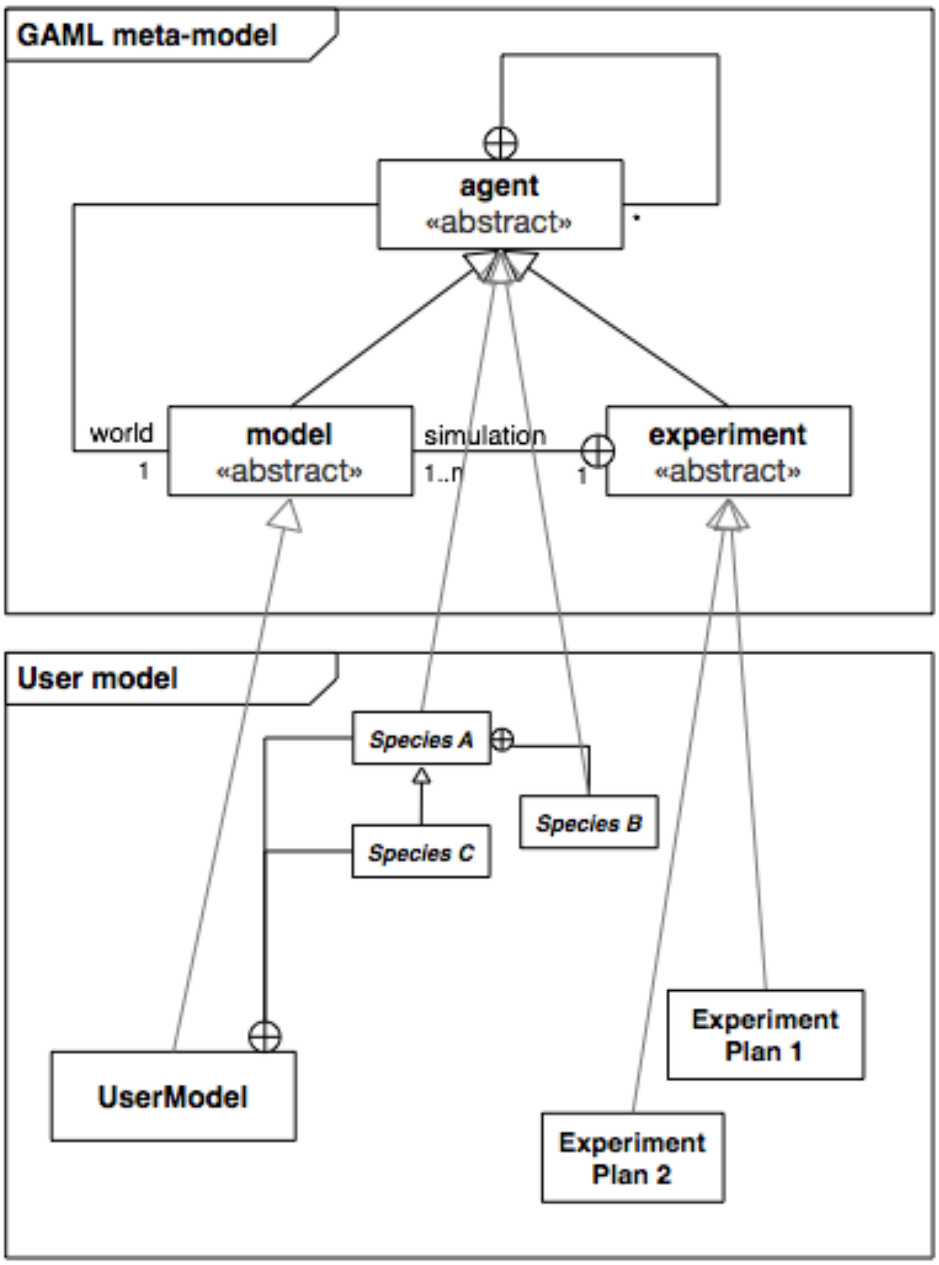

FIGURE 2.3 – Écrire un modèle en GAML consiste à définir une espèce qui hérite de modèle, dans lequel d'autres espèces, héritant (directement ou non) de l'agent, représentent les entités qui peuplent ce modèle, lui-même imbriqué dans un ou plusieurs plans d'expériences parmi lesquels un utilisateur sera en mesure de choisir l'expérience qu'il veut exécuter.

2.1.2 Organisation d'un modèle

Définir un modèle en GAML revient à définir une espèce modèle, ce qui permet ensuite d'instancier un agent modèle (une simulation), qui peut ou non contenir des micro-espèces, et qui peut contenir des plans d'expérience pouvant être simulés. Cette structure conceptuelle est respectée dans la définition de fichiers de modèles, qui suit un schéma similaire :

1. Définition des espèces globales, précédée par un en-tête, afin de représenter les espèces du modèle.
2. Définition des différentes micro-espèces (soit imbriquées dans les espèces globales ou au même niveau).
3. Définition des différents plans d'expérimentation qui ciblent ce modèle.

En-Tête

L'en-tête (header) d'un fichier GAML commence avec la déclaration obligatoire du nom du modèle qui servira aussi de nom pour définir l'espèce du modèle. Un modèle est lui même considéré comme étant un agent. La déclaration du modèle est éventuellement suivie de fichiers pointant vers d'autres modèles. Les variables et actions globales y sont définies. Un bloc d'initialisation permet d'instancier les agents et d'initialiser les variables globales.

Species (Entité)

L'en-tête est suivi par la déclaration des différentes espèces qui seront présentes dans le modèle. Une espèce est un objet représentant une catégorie, qui est instanciée en un ou plusieurs agents qui hériteront de ses attributs et de ses propriétés. Cela rappelle la conception de programmation orientée objet mais la différence la plus notable réside dans la définition des réflexes. On définit un comportement que chaque agent instancié va pouvoir exécuter, que ce soit indépendamment ou en relation avec les autres.

Une espèce peut contenir plusieurs éléments :

- **Attributs** : définit l'état des agents.
- **Action** : définit une capacité des agents. Une action est un bloc d'instructions exécutées lorsque l'action est appelée.

- **Reflex** : définit le comportement par défaut des agents. Ces deux déclarations contiennent des instructions qui sont exécutées, respectivement, une fois lorsque l'agent est créé pour les états d'initialisation, et à chaque étape de simulation ou selon une condition facultative.
- **Aspect** : définit la manière dont les agents peuvent être affichés.
- **Equation** : définit un ensemble d'équations différentielles qui peuvent être intégrées à l'espèce.
- **Micro-species** : espèces imbriquées pouvant être décrites dans une espèce.

Parmi ces éléments, la notion d'**aspect** est particulièrement importante, puisque c'est grâce à cet élément que l'on va définir les différentes façons dont vont s'afficher l'agent et ce dans différents affichages.

Experiment

Une expérimentation (`experiment`) permet d'afficher une représentation graphique d'un modèle. Elle se compose d'entrées (`parameter`) et de sorties (`display`, `file`, `monitor`). Chacune d'entre elles correspond à une simulation qui va être lancée à partir du modèle.

- **Entrées**

  A tout moment l'utilisateur peut définir des entrées sous formes de paramètres dans son expérience. L'utilisation de paramètres permet de définir la valeur d'une variable globale (donc un attribut de l'agent simulation) à l'aide d'une interface graphique. La métaphore de la console a été gardée afin de disposer d'un ensemble de contrôles dans un coin de l'écran. Les contrôleurs sont liés à des variables du programme et permettent de changer le comportement du programme sans toucher au code et sans interrompre le programme.
- **Sorties** Les sorties d'une simulation peuvent être des sorties graphiques. Il est aussi possible d'écrire directement dans un fichier à l'aide du mot clé *file* et de définir des moniteurs permettant d'observer un agent ou un paramètre en particulier.

La syntaxe générique d'un modèle est représentée dans la figure 2.4. La prochaine partie portera une attention particulière à la notion d'affichage et d'aspect qui sont des points fondamentaux pour la création de modèles de visualisation.

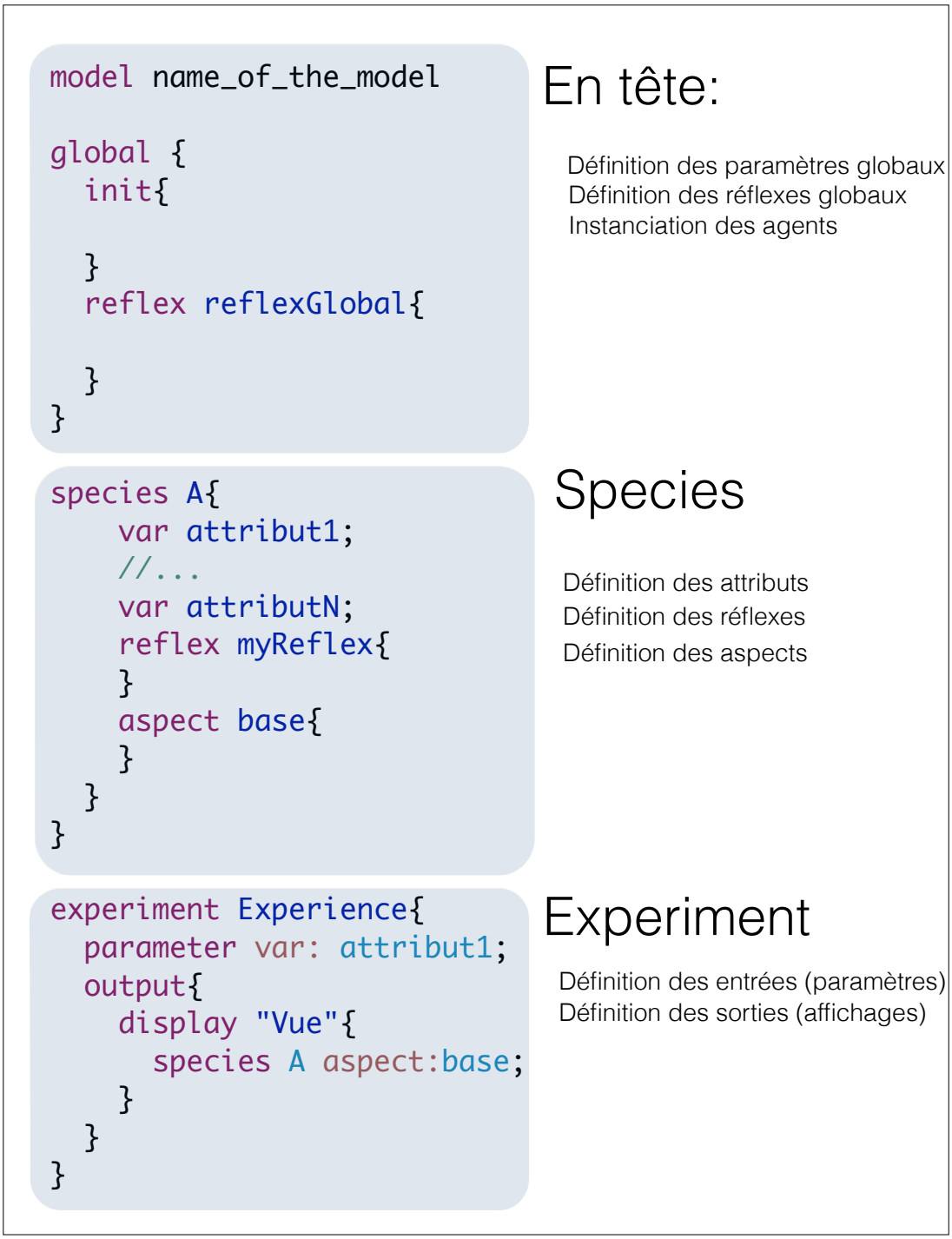

FIGURE 2.4 – Syntaxe GAML pour définir un modèle. L'en-tête permet de définir les paramètres globaux et d'instancier les agents. Suit alors la définition des espèces et enfin des différentes expérimentations.

2.2 Environnement 3D immersif

Dans un modèle de visualisation à base d'agents, les agents sont spatialisés et possèdent donc une position dans un monde 3D, utilisant des coordonnées cartésiennes et dans lequel il est possible d'utiliser la plupart des techniques modernes de visualisation réaliste en 3D. Initialement seule la 2D était prise nativement en compte dans la plateforme GAMA. Cet environnement immersif, développé au cours de la thèse, s'articule autour d'une librairie graphique dédiée, OpenGL [4]. OpenGL a été choisie pour différentes raisons notamment son caractère multi-plateformes et non propriétaire. Cette librairie dispose de nombreuses fonctionnalités utiles à l'élaboration d'un laboratoire immersif telles que la possibilité de créer des objets 3D, de définir des caméras, d'utiliser des textures et différentes sources lumineuses [5].

Comme le souligne [Allan, 2010] et [Railsback et al., 2006], l'usage de la 3D est encore peu courant dans le monde de la simulation multi-agent et il existe peu de méthodes pour enrichir visuellement les simulations [Crooks et al., 2011]. Les modélisateurs ont tendance à se concentrer plutôt sur la théorie et la justesse de leur modèle que sur leur représentation souvent à cause d'une technologie existante insuffisante et parfois trop dure à prendre en main. Il y a pourtant maintenant peu de barrières technologiques pour réaliser ces mondes virtuels immersifs fournissant des fenêtres sur la complexité des phénomènes étudiés [Hudson-Smith, 2003], plus particulièrement lorsqu'il s'agit de phénomènes spatialement explicites que se soit en 2D (latitude, longitude), 3D (latitude, longitude, altitude) ou 4D (latitude, longitude, altitude, temps) [6]. De plus, l'utilisation d'une représentation 3D permet de répondre à des questions jusqu'ici impossibles à résoudre en 2D. Par exemple, dans des modèles d'évacuation d'urgence, les piétons peuvent bénéficier de la troisième dimension en évoluant dans des structures complexes formées par des immeubles à plusieurs étages [Castle, 2007] [Batty et al., 1998] [7].

4. Actuellement deux API (Application Program Interface) existent pour exploiter la carte graphique : DirectX (ne fonctionnant que sous Windows) et OpenGL (multi-plateforme).

5. La bibliothèque jogl [5] est la version pour Java que nous utilisons dans la plateforme Gama. Dans la version 1.6.1 de GAMA, la version utilisée est la version 1.1.1 et la version 2.0 sera intégrée à la version 1.7 de GAMA.

6. Se référer à [Crooks et al., 2011] pour un aperçu des différents outils de couplage ABM-3D.

7. Se référer à [Smith and Crooks, 2010] pour une description plus précise de la représentation multi-échelles urbaine.

Introduction à OpenGL

Nous décrivons dans cette partie les possibilités d'OpenGL afin de mieux comprendre le fonctionnement et les capacités des modèles de visualisation que nous proposons. OpenGL (Open Graphics Library) est donc une bibliothèque graphique permettant de développer des applications 2D ou 3D. De nombreux jeux et applications utilisent OpenGL pour leur rendu (Google Earth, Blender, AutoDesk, etc). Cette librairie fournit des fonctions 3D devant être exécutées dans un contexte graphique. Il est donc nécessaire de choisir un système de fenêtrage afin de permettre à l'utilisateur d'interagir via un clavier et un dispositif de pointage comme une souris avec plusieurs applications graphiques visibles simultanément. Une scène 3D est essentiellement formée de polygones, le polygone le plus souvent utilisé est le triangle. Même un modèle 3D extrêmement complexe est souvent composé essentiellement de triangles. Pour dessiner un polygone en OpenGL il faut définir tous les sommets qui le composent et indiquer à OpenGL comment il doit interpréter ces sommets. On parle plus souvent du terme anglais *vertex* (*vertices* au pluriel). Une fois les polygones définis, les transformations vont permettre de placer ces polygones à n'importe quel endroit d'une scène 3D. Parmi ces transformations on trouve la translation, la rotation et le changement d'échelle. Ces transformations sont gérées en OpenGL à l'aide de matrices. L'utilisation géométrique de ces matrices permet de prédire numériquement le résultat d'une transformation. Ces matrices sont rarement manipulées directement mais nous rappelons simplement qu'il en existe trois : la matrice de projection dans laquelle le type de projection est défini (orthogonale ou perspective), la matrice ModelView permettant de positionner les objets dans la scène 3D (caméra, vertices, lumières, etc), et enfin la matrice de texture permettant de faire des textures animées.

Lorsque l'on décrit un ensemble de polygones à l'aide de vertex, on définit la scène 3D "*telle qu'elle est dans l'absolu*". Pour passer du monde "*réel*" à l'écran, il est nécessaire de (1) définir la zone de la fenêtre qui servira pour le rendu, (2) le mode de projection et de (3) placer une caméra dans le monde "*réel*". Ces trois informations vont permettre à OpenGL de calculer les transformations à faire subir aux objets du monde "*réel*" pour qu'ils apparaissent sur un écran à 2 dimensions. En d'autres termes, pour passer des vertices dans le monde "réel" au pixels dans le monde "écran" comme le montre la figure 2.5.

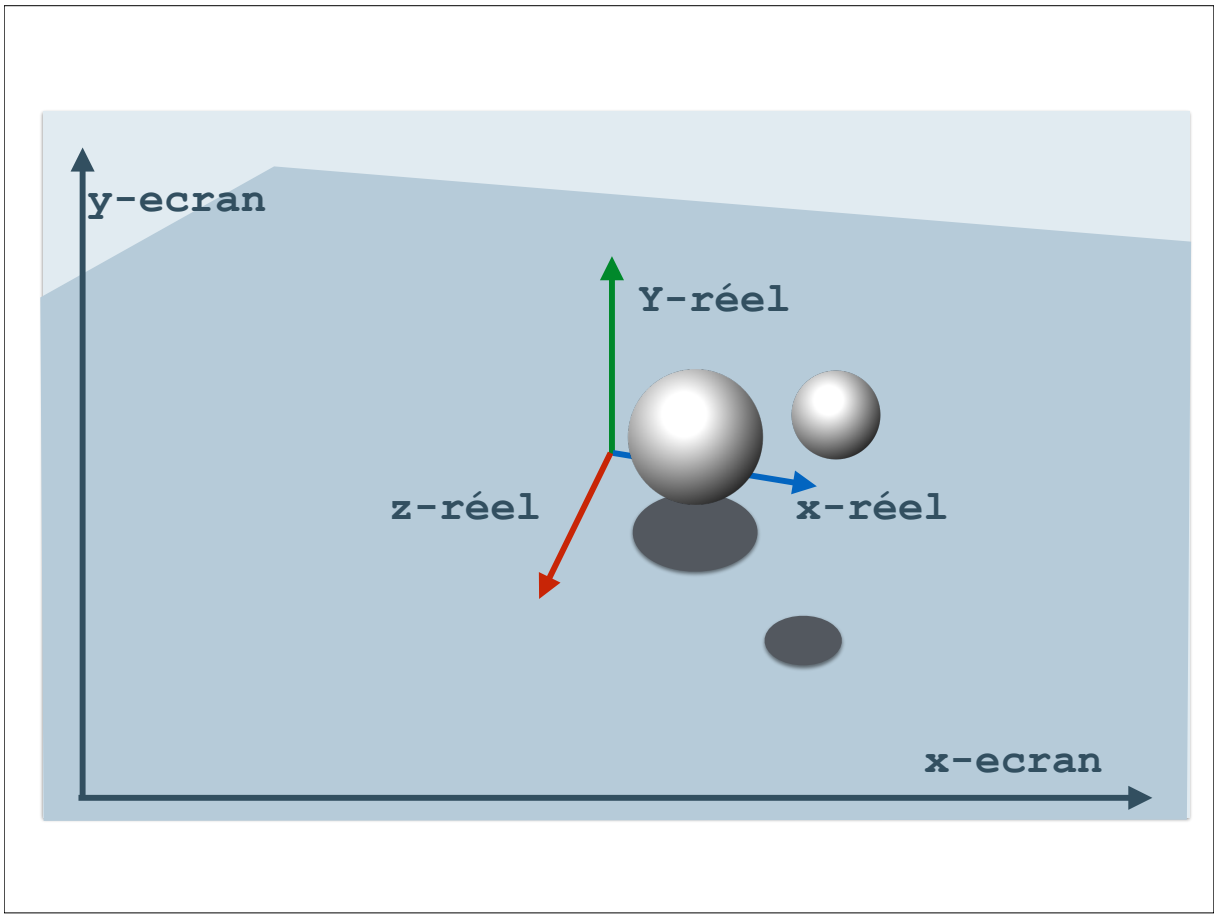

FIGURE 2.5 – Représentation d'une scène 3D. Le monde "réel" (la scène) utilise des vertices dans un repère 3D, le monde écran utilise des pixels dans un monde en 2D.

La pyramide de clipping permet de définir quel objet de la scène 3D sera rendu à l'écran et sa forme permet de définir comment les objets seront projetés à l'écran. Elle est définie par les paramètres *angle*, *ratio*, *near* et *far* de la figure 2.6. L'*angle* est l'angle de vision entre les plans haut et bas de la pyramide. Le *ratio* est le rapport entre la largeur et la hauteur de la fenêtre d'affichage. Les paramètres *near* et *far* représentent les distances minimales et maximales des objets qui seront rendus. Les objets se trouvant en dehors de cet intervalle ne sont pas rendus.

Une fois cette projection définie, il suffit de placer le point de vue à n'importe quel endroit de la scène en faisant appel à une caméra virtuelle représentée en figure 2.7.

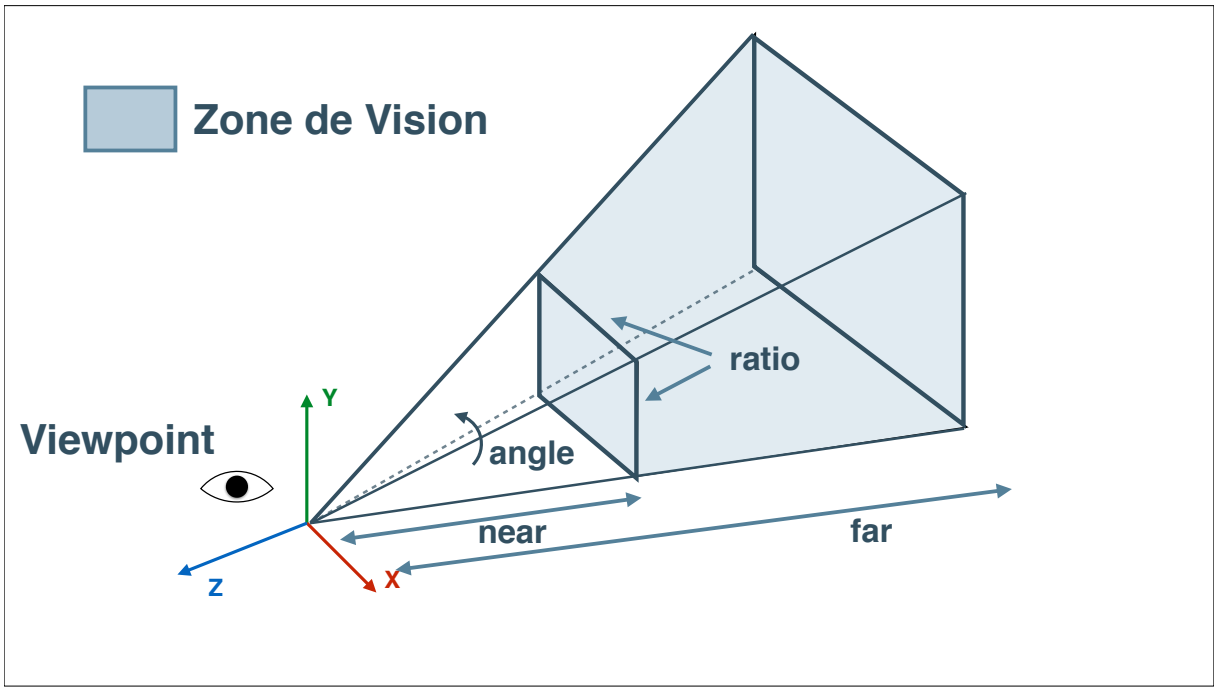

FIGURE 2.6 – Pyramide de clipping.

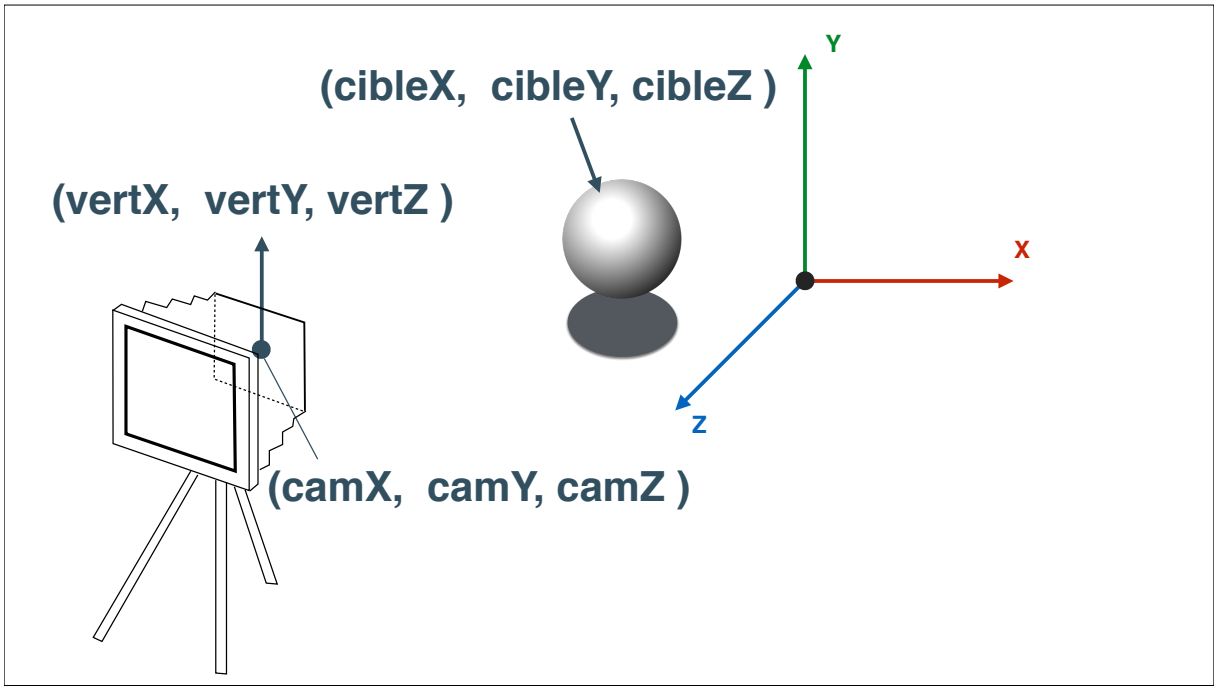

FIGURE 2.7 – Caméra virtuelle. Le vecteur *cam* définit la position de la caméra, le vecteur *cible* définit la position du point que fixe la caméra. Enfin, le vecteur *vert* définit le vecteur vertical de la caméra (son orientation).

2.3 Modèle de visualisation

Cette partie vise à expliciter les concepts exposés dans la partie précédente en les appliquant au modèle de visualisation à base d'agents. Certains concepts seront directement empruntés à la technologie de rendu OpenGL, d'autres seront spécifiques à la méthodologie à base d'agents. Cette syntaxe permet de (1) faciliter la création de scène 3D pour le modélisateur et (2) conserver la même méthodologie orientée agent. Nous rappelons qu'il faut voir un modèle de visualisation à base d'agents comme une collection d'agents ayant chacun un ou plusieurs aspects, chacun de ces aspects pouvant être utilisé dans différents affichages.

2.3.1 Environnement

L'environnement d'un modèle de visualisation est définit par une surface d'affichage. Un affichage (`display`) se réfère à une partie indépendante de l'interface permettant d'afficher des espèces, des images, du texte ou des graphiques. Chaque affichage peut inclure différentes couches (`layer`). Chaque couche possède des options permettant de jouer sur la taille, la transparence et la position des couches comme le montre la figure 2.8. Un affichage se définit à l'intérieur d'une expérimentation `experiment` avec la syntaxe suivante :

```
display my_display [additional options] { ... }
```

Point de vue

Un point de vue peut être statique, dynamique et interactif. Il se contrôle par le biais d'une interface (souris et clavier) ou de façon programmatique en spécifiant à chaque itération de la simulation où doit se positionner la caméra. L'utilisateur a la possibilité de définir dynamiquement le paramètre de la caméra (observateur). Les propriétés de base d'une camera sont sa position (`camera_pos(x,y,z)`), la direction vers laquelle est pointe (`camera_look_pos(x,y,z)`), et son orientation (`camera_up_vector(x,y,z)`). Ces trois paramètres peuvent être définis dynamiquement à chaque itération de la simulation. Par défaut la caméra est placée "au-dessus" de l'environnement afin d'avoir une vision globale du modèle.

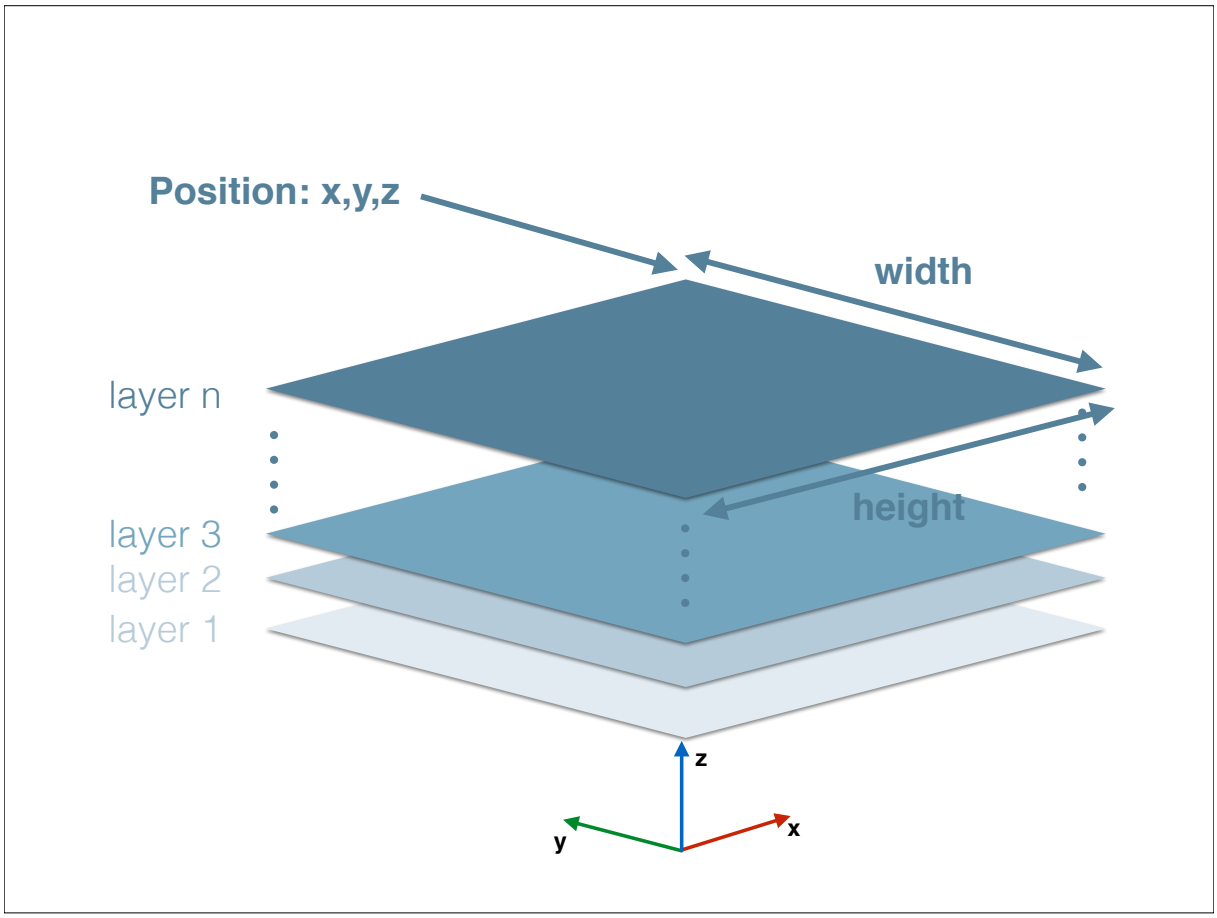

FIGURE 2.8 – Un affichage se compose de différentes couches. Chaque couche possède une position, une largeur (width), une hauteur (height) et une transparence.

Eclairage

La lumière ambiante se définit à l'aide de la facet `ambient_light`, la lumière diffuse à l'aide de la facet `diffuse_light` ainsi que la position de la lumière diffuse à l'aide de la facet `diffuse_light_pos`. Ces différents paramètres sont issus du modèle d'illumination de Phong [Phong, 1975] pour lequel nous ne prenons en compte que la lumière ambiante et diffuse [8].

8. Ce modèle permet de calculer la lumière réfléchie par un point, pour cela il combine trois éléments : la lumière ambiante, la lumière diffuse et la lumière spéculaire.

Variables visuelles

La définition de variables visuelles permet de fournir un langage commun pour la conception graphique. Comme le proposent les travaux en sémiologie et cartographie de [Bertin, 1967] [Tukey, 1977] [Slocum et al., 2009], ces variables visuelles sont la position, la taille, la forme, la couleur, la transparence et l'orientation [9]. Chacune de ces variables sera utilisée selon l'information que l'on désire faire passer. Les variables sélectives permettent à un agent d'être immédiatement distinguable d'un autre agent proche. Les variables associatives permettent à un groupe d'agents d'être immédiatement perçu. Les variables quantitatives permettent une estimation rapide d'un ratio numérique entre agents.

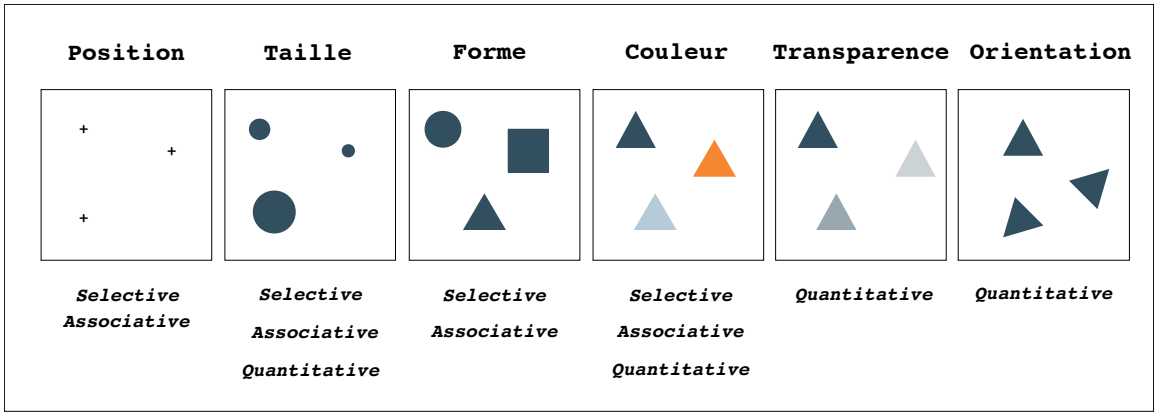

FIGURE 2.9 – Les variables visuelles permettent, selon leur type, une représentation sélective, associative ou quantitative.

Palette de couleur

L'utilisation d'un schéma contre-intuitif de couleur peut perturber la visualisation en représentant une valeur numérique par une couleur inappropriée. Ainsi il est important d'avoir une notion du type d'information que l'on désire faire passer par le biais d'une couleur. Si l'on désire faire passer une information *séquentielle*, comme un gradient de température, on utilisera une palette séquentielle utilisant la même teinte allant du clair au foncé. Lorsque l'on désire représenter des valeurs s'articulant autour d'une valeur moyenne, comme le revenu moyen d'une population, il est plus judicieux d'utiliser une palette *divergente*. Enfin pour des informations purement qualitatives, comme le type de bâtiment dans une ville, l'utilisation d'une palette *qualitative* est plus appropriée.

9. Des variables plus complexes sont définies dans [MacEachren, 2004] et [Ware et al., 2002]

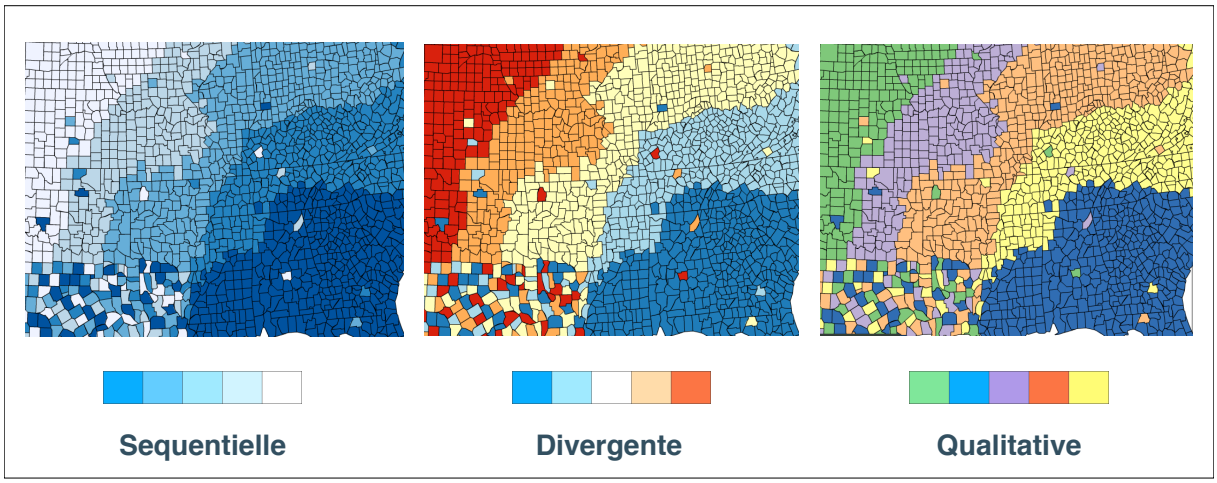

FIGURE 2.10 – Illustration des différents types de palettes. Séquentielle, divergente et qualitative [Harrower and Brewer, 2003].

Animation

L'animation est une composante essentielle de la visualisation de modèles à base d'agents. Les notions de durée, de taux de rafraichissement et de synchronisation sont à prendre en compte dans la définition d'un modèle de visualisation. Ces notions sont liées au fait que lorsqu'une simulation est lancée, une partie du calcul est dédiée au calcul du modèle alors qu'une autre est essentiellement dédiée au rendu graphique. Si la charge de calcul est entièrement dédiée au calcul du modèle, le rendu aura du mal à se mettre à jour ; inversement, si le rendu est trop gourmand en temps de calcul, le temps de calcul dédié au modèle sera faible. Il est donc important de pouvoir, dans certains cas, désynchroniser la fréquence de calcul du modèle de celle de l'affichage. Une mauvaise synchronisation de l'affichage et du modèle pourra mener soit à un *aliasing* temporel [Korein and Badler, 1983], les agents deviennent visuellement difficiles à suivre car ils risquent de changer brutalement de position plutôt que de se déplacer de façon continue, soit à un taux de rafraichissement irrégulier dépendant du temps de calcul du modèle.

En GAML, la facet `synchronized` permet d'éviter les effets indésirables de l'aliasing en spécifiant si l'affichage est synchrone avec le modèle. Dans ce cas, l'affichage se rafraichit lorsque le modèle a terminé ses calculs. Le taux de rafraichissement se définit en spécifiant le nombre de cycles à partir duquel l'affichage se rafraichit grâce à la facet `refresh`. De plus, l'utilisation d'OpenGL permet de libérer du temps de calcul lié à la visualisation consommé par le processeur et de le transférer à la carte graphique.

Couche agent (`agent layer`)

La couche `agents` permet au modélisateur d'afficher uniquement les agents remplissant une condition donnée avec un aspect spécifique. De la façon suivante :

```
display my_display {
   agents layer_name value: expression aspect: aspect_name;
}
```

Couche espèce (`species layer`)

La couche espèces (**species**) permet aux modélisateur d'afficher tous les agents d'une espèce donnée avec un aspect spécifique. La syntaxe est la suivante :

```
display display_name {
   species species_name aspect: aspect_name [additional options];
}
```

Couche image (`image layer`)

Cette couche permet d'afficher une image en utilisant la syntaxe suivante :

```
display display_name {
   image layer_name file: image_file [additional options];
}
```

Couche graphique (`graphics layer`)

La couche graphique permet au modélisateur de dessiner librement formes, géométries, textes sans avoir à définir une espèce. La syntaxe générale est :

```
display my_display {
  graphics "my new layer" {
    draw circle(5) at: {10,10} color: rgb("red");
    draw "test" at: {10,10} size: 20 color: rgb("black");
}
```

2.3.2 Représentation

Dans la plupart des plateformes de simulation existantes, les aspects sont très simples et il est difficile de créer des aspects complexes. En offrant une collection de primitives au modélisateur, il devient alors possible de créer une infinité d'aspects. GAML offre une collection de formes primitives de base comme le montre la figure 2.11 [10]. L'utilisation de structures conditionnelles ou itératives donne une liberté à l'utilisateur pour définir des aspects complexes. Les formes peuvent être fusionnées à l'aide d'opérateurs arithmétiques appliqués aux géométries. La couleur est codée en RGB ou HSB. Enfin, il est possible d'appliquer des transformations de base aux géométries comme le changement de position (translation), d'orientation (rotation) et d'échelles (scaling). Quelques exemples utilisant les différentes notions décrites ici sont présentées dans la figure 2.11 et 2.12. Quelques exemples de modèles développés grâce à l'intégration de la 3D dans GAMA sont représentés dans la figure 2.13.

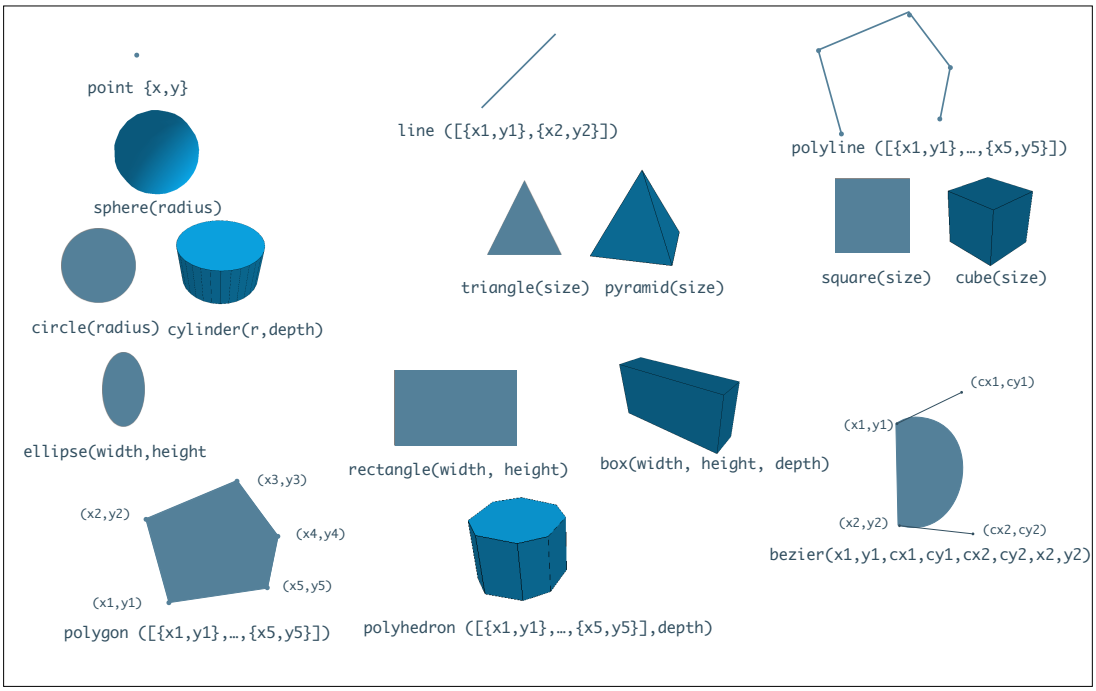

FIGURE 2.11 – GAML propose des primitives 2D et 3D pour assister le modélisateur dans la création de formes simples. La modification des paramètres et la superposition de ces primitives permet de générer un grand nombre de formes.

10. Des aspects plus complexes peuvent être instanciés à l'aide de formats de fichier (.obj, .collada, etc) permettant de stocker des objets en 2 ou 3 dimensions.

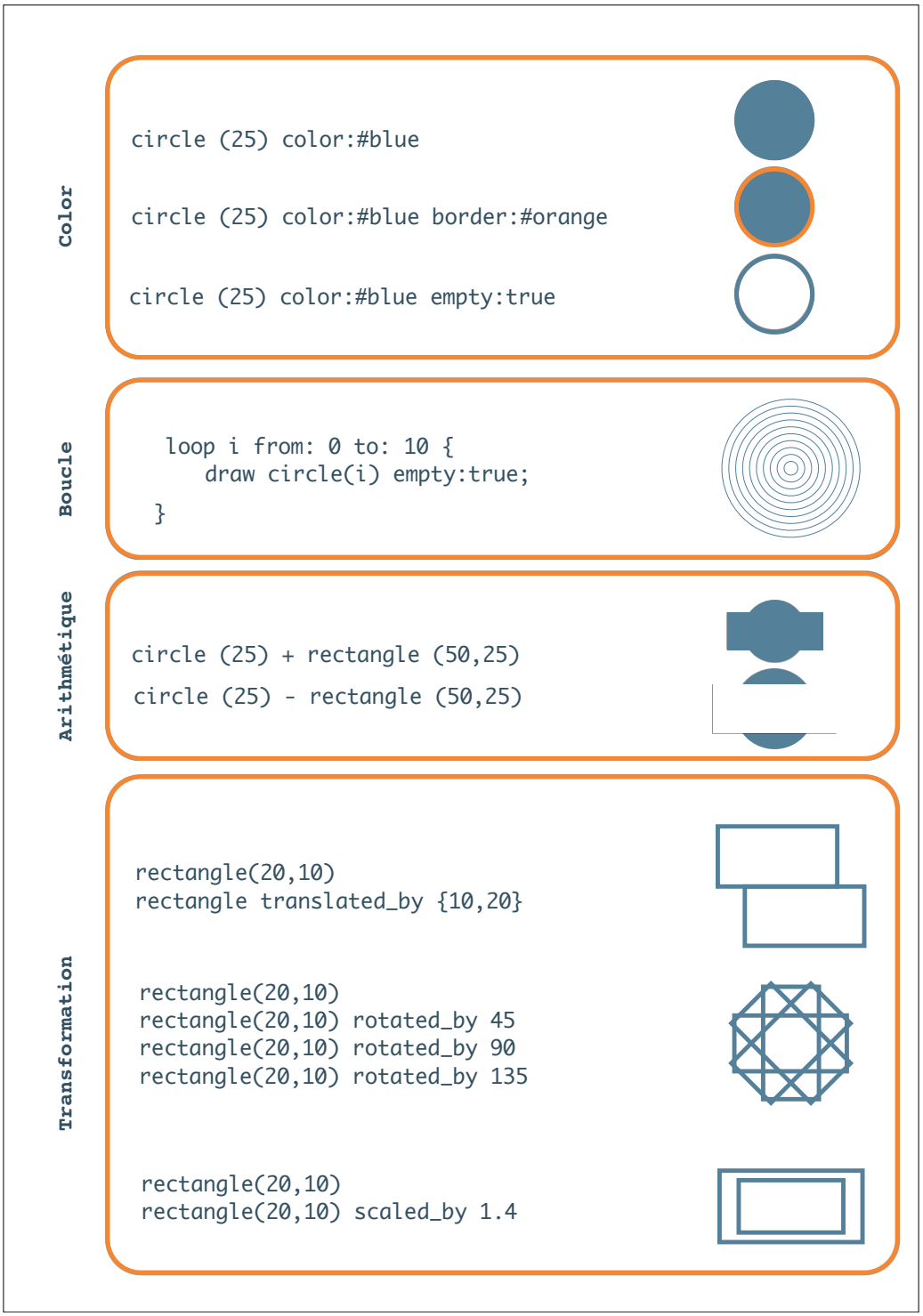

FIGURE 2.12 – Exemple de représentations possibles en utilisant la syntaxe GAML.

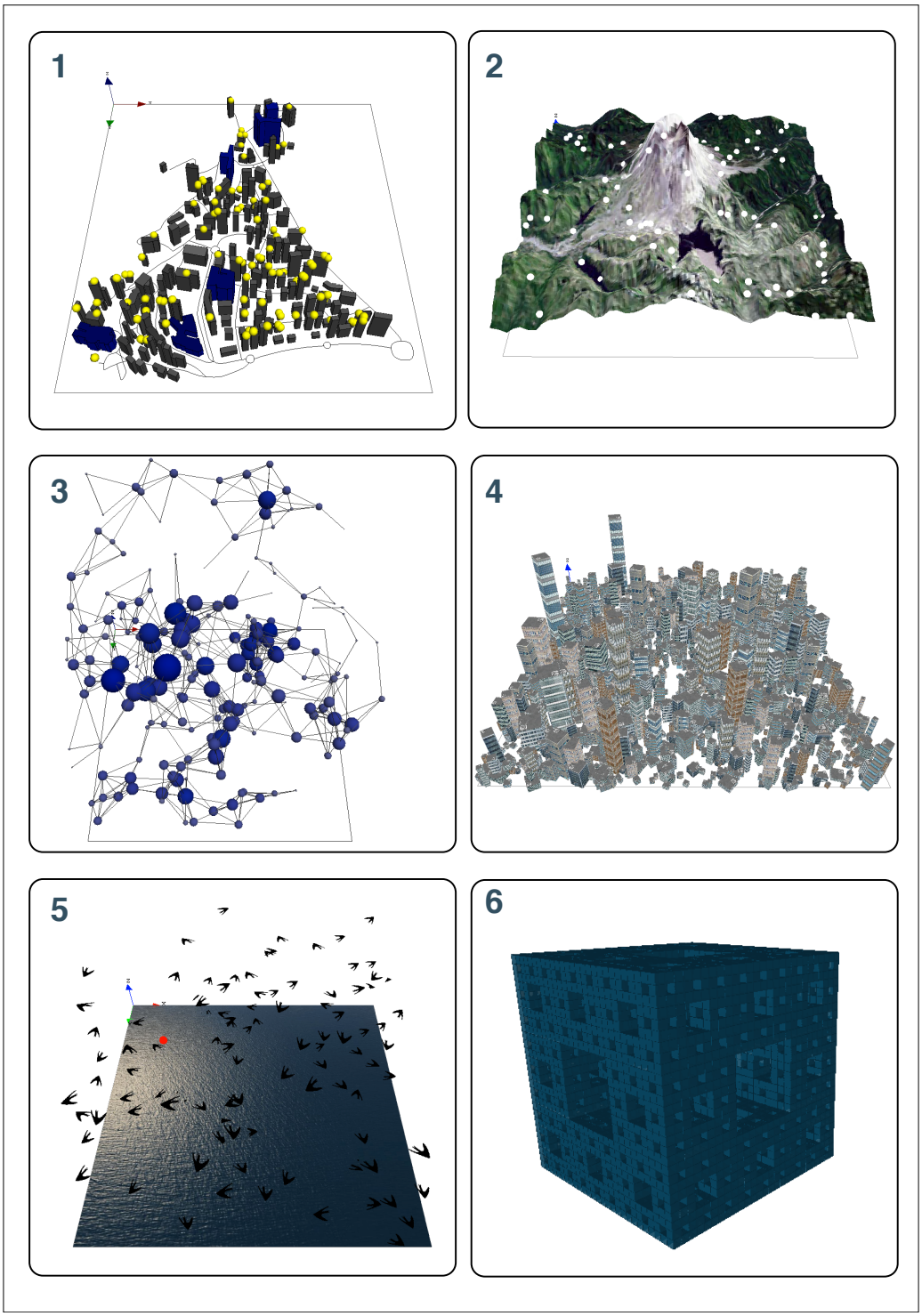

FIGURE 2.13 – Modèle 3D : 1 :Trafic routier. 2 :Modèle Numérique de terrain. 3 :Graph 3D. 4 :Ville procédurale avec textures. 5 :Modèles de nuées d'oiseaux en 3D. 6 : Éponge de Menger.

2.4 Conséquences de l'approche agent

De le même manière qu'en modélisation à base d'agents la programmation individuelle permet de représenter et étudier des phénomènes émergents, on montre dans cette partie que en programmant des agents graphiques dans des modèles de visualisation nous pouvons reproduire de façons simples des processus d'abstraction graphiques, réaliser des activités analytiques de bases et enfin disposer d'une certaine forme d'interactivité avec l'utilisateur.

2.4.1 Abstraction

En visualisation, l'abstraction joue un rôle majeur dans la perception. L'abstraction consiste à réduire la quantité d'information la plupart du temps en cachant des éléments ou en rendant l'information moins détaillée [Saitta and Zucker, 2001]. Kandinsky est un des premiers à proposer une approche scientifique de l'abstraction [Kandinsky, 1991]. Bertin se penche sur la façon de découvrir les relations, l'ordre et la proportionnalité dans les données qualitatives et quantitatives. Dans la sémiologie graphique, qu'il formalise, il étudie comment produire des graphiques utiles en identifiant leurs variables visuelles et en proposant des règles pour les construire [Bertin, 1981]. La figure 2.14 compare deux représentations différentes de données démographiques d'une centaine de régions en France en affichant simplement la valeur de la donnée à gauche et en la représentant avec une forme circulaire, dont le rayon est proportionnelle à la valeur de la donnée, à droite. Dans la représentation abstraite de droite, il est plus facile de voir apparaitre des tendances dans la répartition de la population et d'isoler des zones de forte ou faible densité que dans la représentation de gauche.

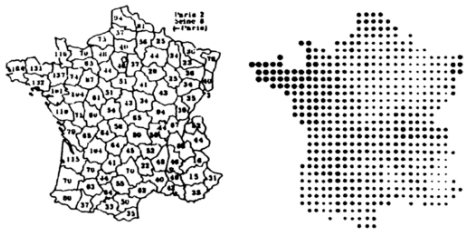

FIGURE 2.14 – L'utilisation de formes abstraites à droite permet d'identifier plus facilement la structure des données que la représentation textuelle de gauche [Bertin, 1983].

Cette capacité à créer différents niveaux d'abstraction sur un même modèle est une des conséquences de l'utilisation de modèles de visualisation à base d'agents grâce à la multiplication des points de vue. En s'inspirant de [Saitta and Zucker, 2013], nous proposons une classification des opérateurs en trois catégories distinctes.

– **Opérateurs cachants**

La classe des opérateurs *cachants* se subdivise en fonction des composants sur lesquels ils agissent, à savoir, les *éléments* ou les *attributs*.

– Les opérateurs consistant à cacher des éléments du système (figure 2.15).

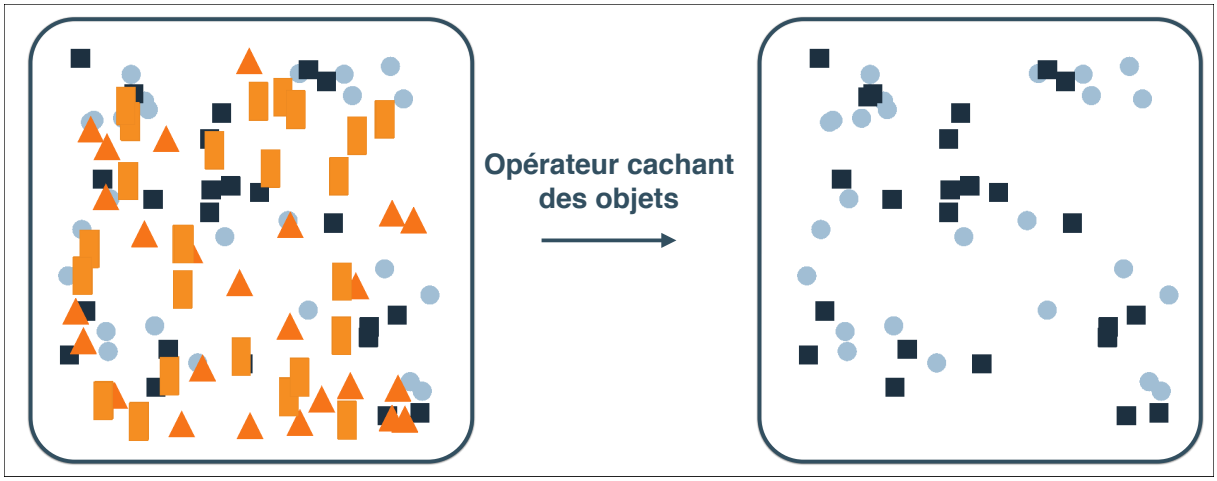

FIGURE 2.15 – Opérateur cachant des objets

– Les opérateurs consistant à cacher des attributs d'agents (figure 2.16)

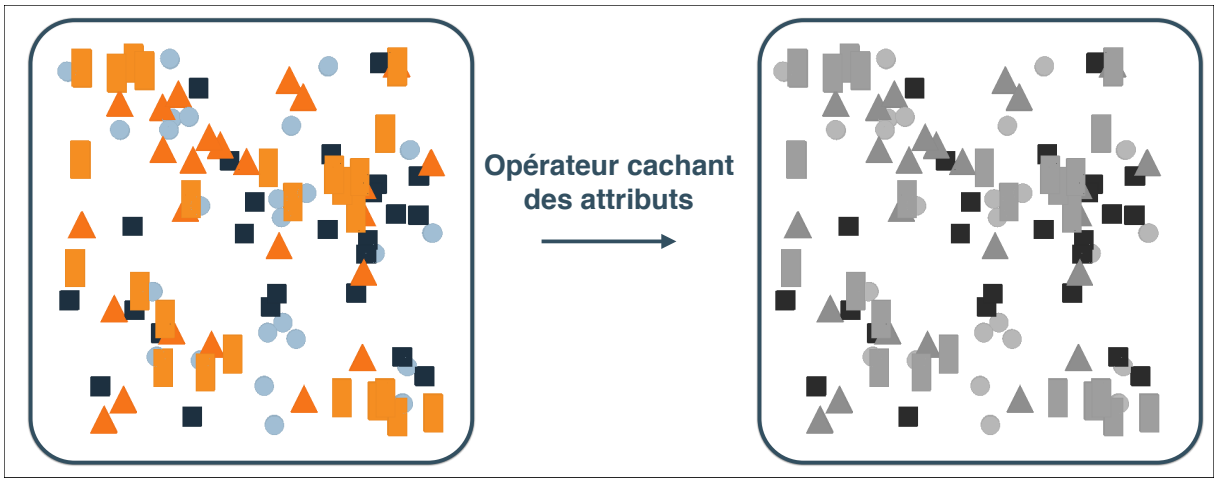

FIGURE 2.16 – Opérateur cachant des attributs

– **Opérateurs d'équivalence** Les opérateurs consistant à construire des équivalences d'éléments (figure 2.17). Il permettent de réduire le niveau de détail de la description d'un système. Ces opérateurs groupent des éléments au sein de classes équivalentes sur des *élements*, sur des *valeurs* ou sur des *arguments*.

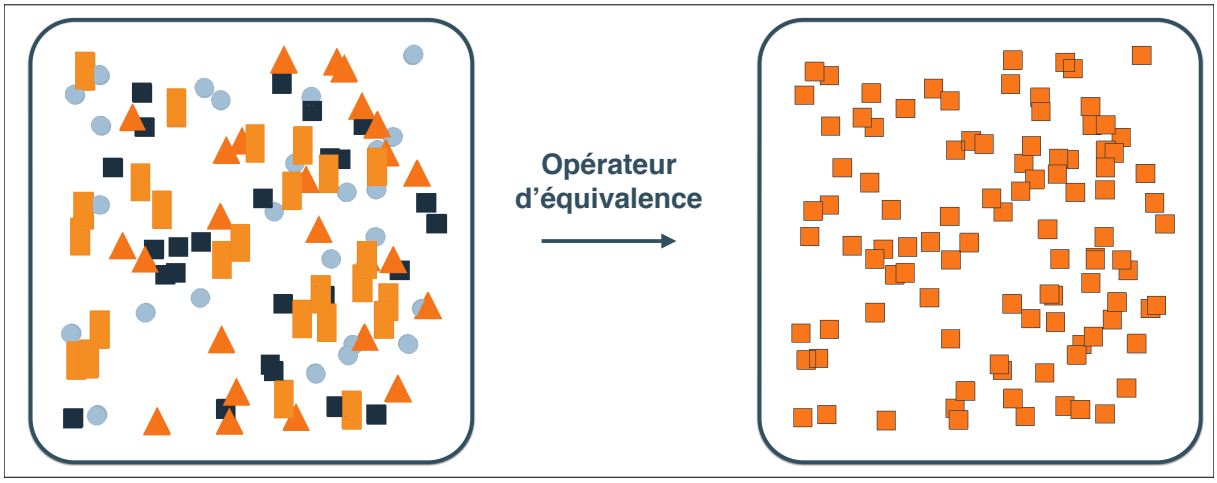

FIGURE 2.17 – Opérateur d'équivalence

– **Opérateurs d'agrégation** Les opérateurs consistant à construire des agrégations d'agents en combinant des éléments existants (figure 2.18). Ces opérateurs remplacent certains éléments par des éléments plus généraux et visent aussi à réduire le niveau de détail en générant des hiérarchies.

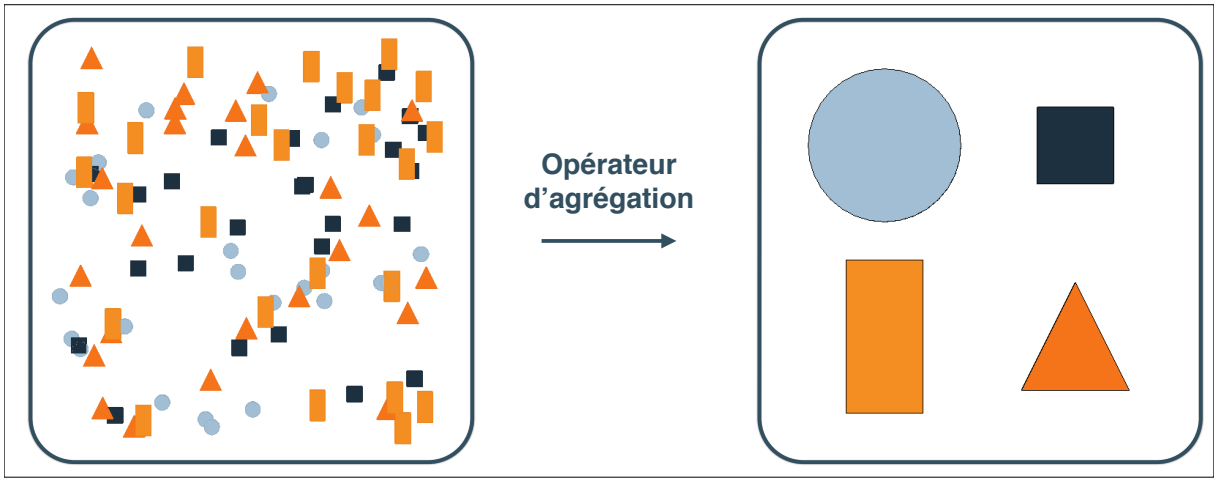

FIGURE 2.18 – Opérateur d'agrégation

2.4.2 Activité analytique

En s'inspirant des travaux de [Amar et al., 2005] présentant un ensemble de tâches analytiques de bas niveau, nous avons retranscrit certaines activités analytiques en partant d'un ensemble de données initial représenté dans la figure 2.19. Chaque tâche est représentée en utilisant une définition générale, une définition abstraite issue de [Newman, 1994] et un exemple de question. Pour chaque cas une figure représente le résultat visuel ainsi que le code GAML utilisé.

FIGURE 2.19 – Le Jeu de données initial est composé de 10 agents `cell` placés aléatoirement dont la taille (`size`) varie entre 1 et 10.

Filtrer

- **Description générale :** Compte tenu des conditions concrètes sur les valeurs d'attributs, trouver des cas de données répondant à ces conditions.
- **Forme abstraite :** Quelles données satisfont les conditions A, B, C... ?
- **Exemple :** Quelles sont les cellules qui ont une taille supérieure à 5.

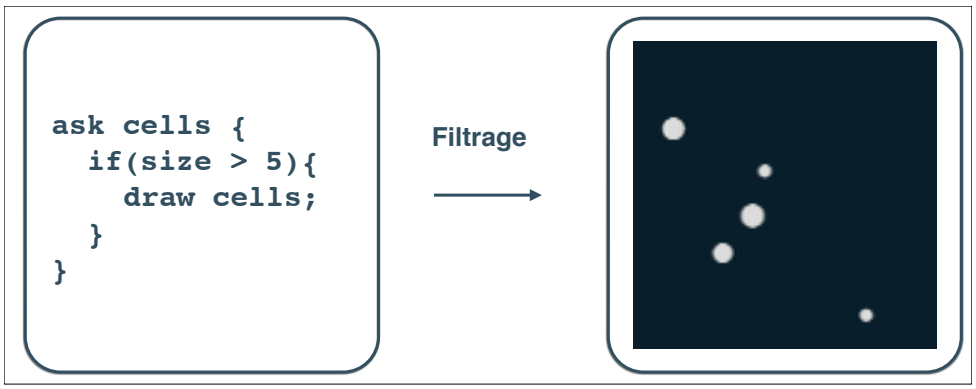

FIGURE 2.20 – Le filtrage consiste à ne garder que les données correspondant à certaines conditions, ici seules les cellules dont la taille est supérieure à 5 sont affichées.

Agréger

- **Description générale :** Étant donné un ensemble de cas de données, calculer une représentation numérique globale de ces cas de données.
- **Forme abstraite :** Quelle est la valeur de la fonction F sur un ensemble de données S ?
- **Exemple :** Représenter une macro-cellule dont la taille est proportionnelle au nombre de cellules présentes.

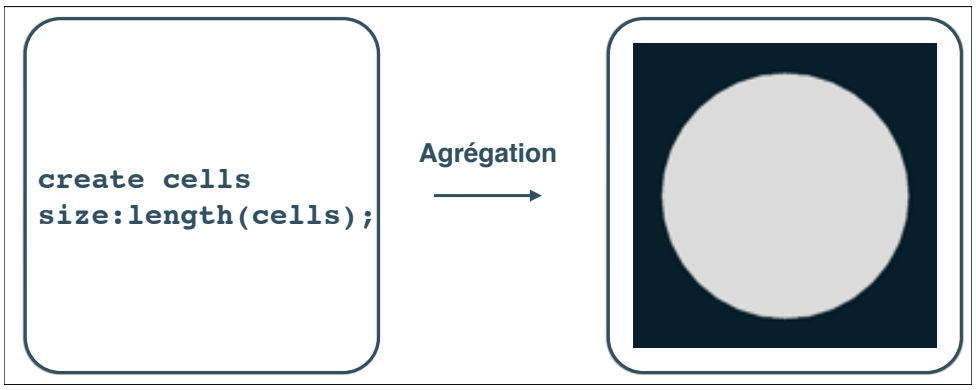

FIGURE 2.21 – L'agrégation consiste à donner une représentation globale d'un ensemble de données, ici le rayon de la macro-cellule représente le nombre de cellules.

Trier

- **Description générale :** Étant donné un ensemble de cas de données, les classer selon certains métriques.
- **Forme abstraite :** Quel est l'ordre de tri d'un ensemble S de cas de données en fonction de leur valeur de l'attribut A ?
- **Exemple :** Représenter les cellules à une position proportionnelle à leur taille sur l'axe des x.

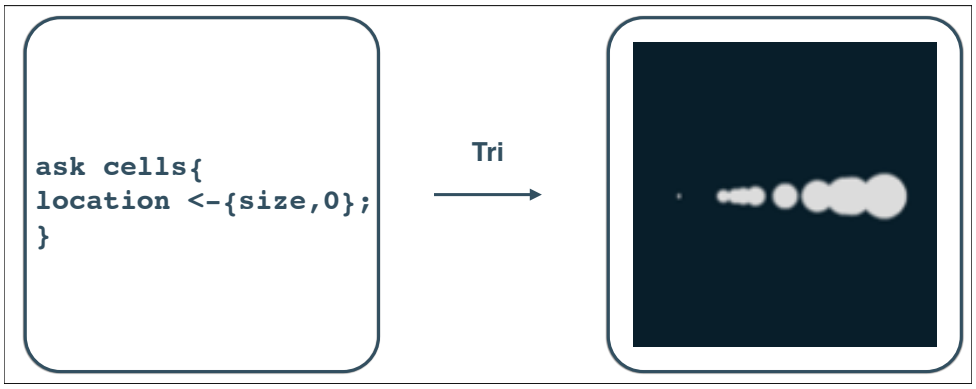

FIGURE 2.22 – Le tri consiste à classer les données selon certains métriques, ici les cellules sont triées en fonction de leur taille.

Regrouper

- **Description générale :** Étant donné un ensemble de cas de données, trouver des groupes de valeurs d'attributs similaires.
- **Forme abstraite :** Quelles données dans un ensemble S sont similaires en valeur pour les attributs X, Y, Z, ... ?
- **Exemple :** Représenter une première cellule dont la taille est proportionnelle au nombre de cellules de taille inférieure à 5. Représenter une deuxième cellule dont la taille est proportionnelle au nombre de cellules de taille supérieure à 5.

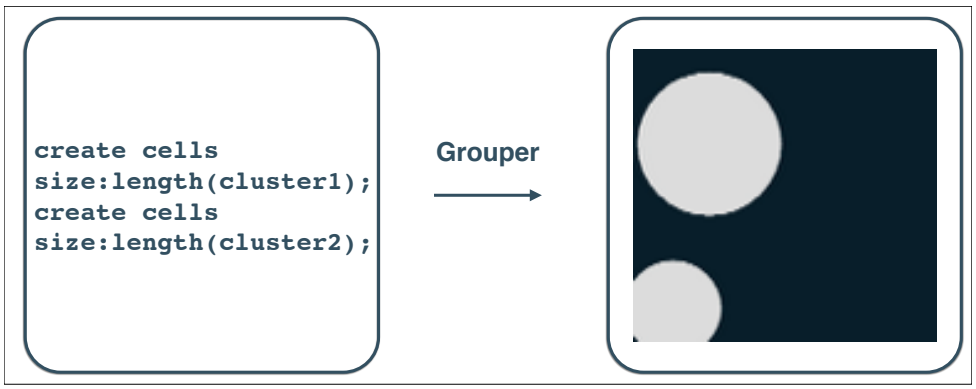

FIGURE 2.23 – Le regroupement consiste à isoler les groupes ayant des valeurs d'attributs similaires, ici on regroupe les cellules dont la taille est inférieure à 5 dans le groupe 1 et les cellules dont la taille est supérieure à 5 dans le groupe 2.

2.4.3 Interaction

L'interaction est "un moyen d'utiliser un dispositif physique pour effectuer une tâche générique dans un dialogue humain-ordinateur" [Foley et al., 1997] et se définit comme la communication entre l'utilisateur et le système [Dix, 2009]. Nous illustrons ici comment les modèles de visualisation à base d'agents permettent d'utiliser l'environnement 3D immersif et les agents graphiques comme support de l'interactivité. En effet dans les modèles ABM, les agents sont des entités actives et interactives, ainsi un agent est sensé pouvoir interagir avec d'autres entités et en particulier l'utilisateur. Nous montrons ici comment l'environnement 3D immersif décrit dans la partie 2.2 répond aux règles d'interaction simple et enfin comment, à l'aide de commande GAML , l'utilisateur peut modifier interactivement la visualisation au cours d'une simulation.

Interaction homme-machine

L'exploration visuelle de données suit généralement les 3 règles décrites dans le Seeking Mantra [Shneiderman, 1996], vue d'ensemble, zoom et filtre et enfin détails sur demande. L'utilisateur doit avoir une vue générale sur les données. Ceci est rendu possible grâce à la définition d'une caméra correctement positionnée. Afin d'analyser ces motifs il est nécessaire d'avoir une représentation plus détaillée. De même ceci est rendu possible grâce à la définition de différentes caméras au sein de différents affichages. Il est alors soit possible de proposer plusieurs représentations (une vue d'ensemble et une plusieurs vues détaillées). Il est parfois nécessaire de réduire le nombre de données à afficher par des techniques de filtrage de données comme celle illustrées dans la parte précédente.

Enfin, à tout moment un agent est "sondable". Cette fonctionnalité est disponible dans GAMA en faisant un clic droit sur un agent. Dans ce cas une fenêtre de dialogue apparait et fournit les trois fonctions suivantes :

– `Inspect` : Cette fonction permet d'afficher dans une nouvelle fenêtre toutes les propriétés de l'agent comme son nom, son identifiant, sa géométrie, sa position, et toutes les valeurs d'attributs définies dans l'espèce. Cette fonction d'inspection est essentielle pour connaitre l'état d'un agent au cours d'une simulation. En plus d'inspecter un agent, cette fonction permet aussi de changer dynamiquement les attributs, règles et aspects d'un agent.

- `Highlight` : Cette fonction permet de rapidement identifier un agent. Lorsque que le modélisateur est en présence d'un grand nombre d'agents affichés à l'écran et qu'il veut étudier un agent en particulier, il est nécessaire de pouvoir reconnaitre rapidement cet agent parmi la multitude d'autres agents. Ainsi, la fonction *highlight* colore cet agent avec une certaine couleur paramétrable.
- `Focus` : Lorsque l'utilisateur désire se concentrer sur un agent en particulier au cours d'une simulation, il doit pouvoir le suivre. Si la fonction *highlight* permet de le colorer sans changer le point de vue, la fonction *focus* va positionner la camera au dessus de l'agent concerné. Lorsque l'agent bouge, la position de la caméra est remise à jour.

Le modèle support de l'interaction

Le rôle de l'interactivité dans un modèle de visualisation est de permettre à l'utilisateur de modifier la visualisation en cours de simulation pour mieux voir certains phénomènes. L'interactivité doit donc être vue ici comme une technique permettant d'améliorer la visualisation au cours de la simulation. Les outils pour définir les interactions avec l'utilisateur lors de la simulation permettent de donner plus de flexibilité dans le contrôle des agents, la création ou suppression d'agents ou l'appel d'actions spécifiques au cours de la simulation.

Événement

Les événements (`event`) permettent d'interagir avec la simulation en capturant les événements de la souris et faisant appel à une action spécifique dans un affichage. Grâce a cette commande, l'utilisateur peut interagir avec les agents du modèle de visualisation. Les commandes peuvent être des commandes simples comme changer de couleur ou de forme ou des commandes plus complexes comme la mise à jour de la taille en fonction d'une certaine propriété de l'environnement globale ou locale. La syntaxe est la suivante :

```
display View_change_color {
  species cell;
  event [mouse_down] action: action_name;
}
```

Commande utilisateur

A la différence d'un `event`, ne pouvant être créée qu'au sein d'un affichage, une commande utilisateur `user command` peut être définie dans le bloc global, dans une espèce ou dans une (GUI) expérience. Elle peut soit appeler directement une action existante ou être suivie par un bloc qui décrit ce qu'il faut faire lorsque cette commande est exécutée.

Les commandes utilisateurs ne sont pas exécutées quand un agent s'exécute. Elles sont recueillies et utilisées de la façon suivante :

- Lorsque définie dans une expérience de GUI, elle apparait sous forme de boutons au-dessus des paramètres de la simulation.
- Lorsque définie dans la séquence globale ou dans n'importe quelle espèce, lorsque l'agent est inspecté, elle apparait comme un bouton situé en dessous des attributs des agents.

```
user_command cmd_name action: action_name;
```

Entrée utilisateur

Il est parfois nécessaire de demander à l'utilisateur certaines valeurs au cours de la simulation, l'opérateur `user input` affiche alors une boîte de dialogue demandant à l'utilisateur d'entrer des valeurs.

```
global {
   init {
      map values <- user_input(["Number" :: 100]);
      create my_species number : int(values at "Number");
   }
}
```

Conclusion

Ce chapitre nous a permis de concrétiser notre approche en proposant une méthodologie fonctionnelle et implémentée au sein de la plateforme GAMA. L'expérience acquise au cours de cette thèse montre qu'il est loin d'être naturel pour tous les utilisateurs de créer une représentation visuelle pertinente à l'aide d'un langage informatique. Ceci est certainement dû au manque de méthodologies et d'exemples d'application. Ce chapitre se donne donc pour but de regrouper l'essentiel des concepts nécessaires à la création de modèles de visualisation. Après avoir présenté l'architecture d'un modèle à base d'agents dans la plateforme GAMA ainsi que langage utilisé, GAML, nous avons illustré à travers différents exemples atomiques les différents éléments de langages disponibles en GAML permettant la représentation, l'abstraction et l'interaction. La sémantique lexicale ainsi que la syntaxe du langage GAML ont été détaillées de manière générique afin de mettre en relief les différents concepts développés au cours de cette thèse. Le cœur de l'approche est basé sur l'usage d'agents graphiques évoluant dans un environnement graphique immersif. Ces agents, une fois instanciés dans un modèle de visualisation, forment une représentation qu'il est possible de modifier à l'aide de la multiplication des aspects, des affichages, des points de vue, d'opérateurs d'abstraction et de techniques d'interaction. A la différence de certaines approches se concentrant essentiellement sur le cadre théorique, nous avons choisi ici de systématiquement relier les concepts à leur implémentation opérationnelle. Dans le chapitre suivant, ces différents concepts sont mis en application à travers divers exemples de modèles ABM implémentés dans la plateforme GAMA.

Chapitre 3

Exemples de mise en oeuvre

Introduction

Ce chapitre illustre les contributions opérationnelles de notre approche illustrées à travers différents exemples implémentés au sein de la plateforme GAMA. La partie 3.1, se basant sur un tutoriel visant à introduire les différents concepts de la modélisation à base d'agents, montre l'apport de notre approche en termes de visualisation dans certaines phases de ce tutoriel. Le couplage avec des données SIG est de plus en plus pratiqué dans la modélisation à base d'agents. Si certaines plateformes permettent d'afficher des données SIG et de faire évoluer des agents dessus, peu permettent de donner un comportement à ces données. Ainsi, la partie 3.2 introduit l'usage des systèmes d'informations géographiques dans la modélisation à base d'agents. La partie 3.3 décrit la notion de modélisation multi-niveaux où nous montrons une application de modèle à trois niveaux. La partie 3.4 vise à illustrer la notion de simulation augmentée et ce, dans une perspective d'analyse en temps réel. Nous verrons que cette analyse est à la fois visuelle à l'aide d'un retour immédiat sur la simulation mais aussi programmatique puisque les agents graphiques ont la capacité de fournir des mesures liées à la leur représentation spatialisé. Enfin, la partie 3.5 illustre le côté interactif de l'approche dans laquelle les agents ABV sont utilisés comme outils pour interagir (lorsque cela est permis) avec le modèle de référence.

3.1 Exemples sur un modèle-jouet

Les travaux de [Railsback et al., 2005] proposent un tutoriel conçu pour introduire la modélisation à base d'agents. Ce tutoriel, composé de 16 étapes, permet d'ajouter de manière incrémentale des fonctionnalités couramment utilisées en ABM. Le modèle de base est une grille de dimensions 100 x 100 sur laquelle 100 agents "bug" (ou insecte) se déplacent de manière aléatoire. Au cours des 16 différents étapes le comportement du modèle est amélioré. Nous proposons, à l'aide de techniques décrites dans le chapitre 2, de rendre la simulation plus intelligible dans 3 des 16 étapes du tutoriel. Nous verrons comment visualiser plusieurs attributs simultanément, comment suivre un agent, comment représenter les interactions entre agents et enfin comment ajouter des informations supplémentaires ne faisant pas forcément partie du modèle de référence.

3.1.1 Visualisation de plusieurs attributs

Dans l'étape 2 du modèle, à chaque pas de simulation, l'agent "bug" grandit et sa couleur est mise à jour en fonction de son âge. Imaginons maintenant que les "bugs" possèdent d'autres attributs tels que le poids, le type d'insecte, etc. Dans une utilisation classique de visualisation, on peut jouer sur la forme, la couleur et la taille d'un agent mais lorsque l'on est en présence de plus de trois attributs à représenter en même temps, il est difficile de tout faire apparaître sur un même affichage. Dans la figure 3.1, trois différents aspects sont mis en œuvre dans trois différents affichages pour représenter trois attributs différents. Cette utilisation donne un retour immédiat à l'utilisateur sur différents attributs de l'agent. Il est surprenant de constater que dans des plateformes comme Netlogo, il est impossible de multiplier les affichages de la sorte.

3.1.2 Suivi d'un agent

L'étape 4 du "Stupide Modèle" consiste à fournir les outils nécessaires pour sonder un agent. Dans le tutoriel original, un agent est "sondable" lorsque que l'on peut le sélectionner et inspecter ses paramètres. Dans le modèle de visualisation proposé dans la figure 3.2, l'affichage 1 est une vue globale du modèle dans lequel un agent en particulier est mis en valeur avec une couleur particulière à l'aide de la fonction `highlight`. L'affichage 2 utilise la fonction `focus` décrites dans la section 2.4.3, il est alors centré sur l'agent que

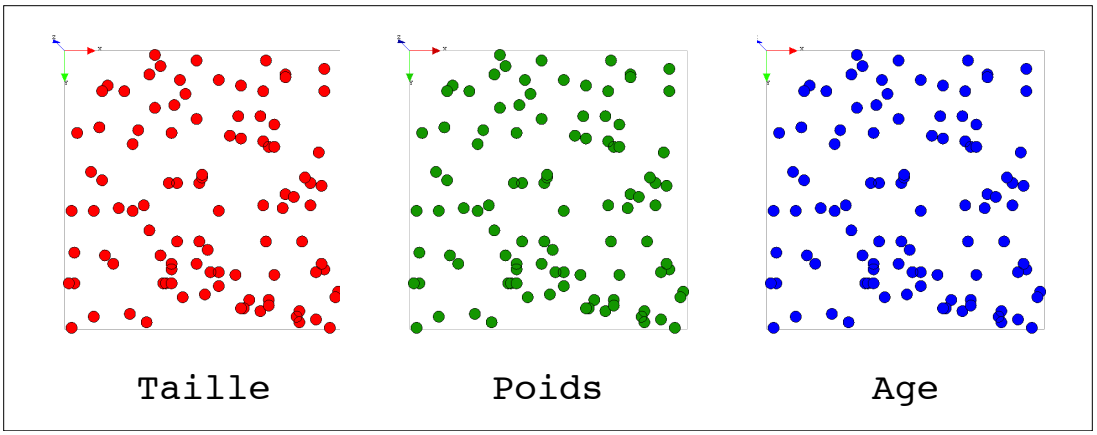

FIGURE 3.1 – La taille, le poids et l'âge sont représentés simultanément dans trois différents affichages grâce à la multiplication des aspects

nous souhaitons sonder. L'aspect de l'agent reste le même, en revanche le point de vue est différent. L'affichage 3 permet, à l'aide d'un aspect dynamique, de garder une trace de son parcours en affichant sa trajectoire au cours de la simulation. L'aspect est alors composé de l'aspect initial (`cercle()`) auquel est ajoutée sa trajectoire. La trajectoire est construite à l'aide d'une liste dynamique dans laquelle est insérée à chaque itération la position courante de l'agent. Cette liste dynamique de points est ensuite affichée comme une ligne à l'aide de la commande `polyline()` décrite dans la section 2.3.2.

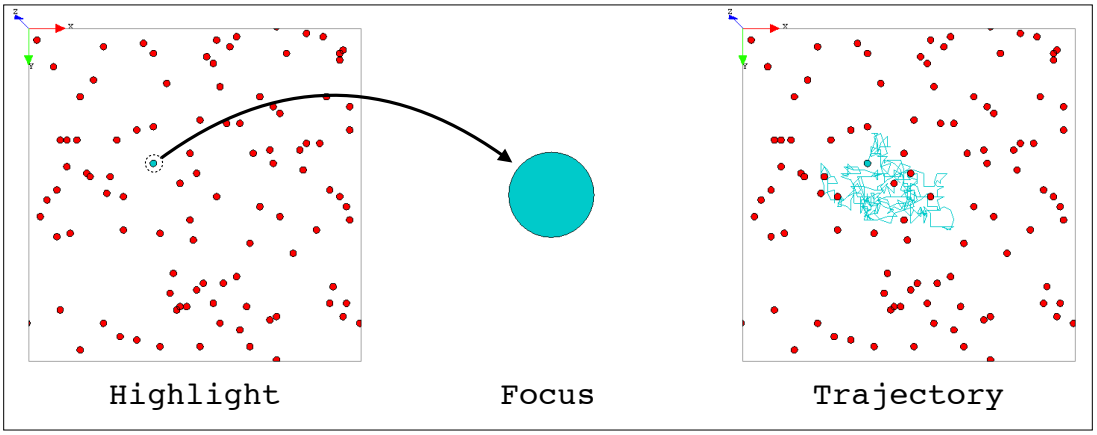

FIGURE 3.2 – Suivi d'un agent à l'aide des fonctions `highlight`, `focus` et de l'utilisation d'un aspect dynamique pour représenter la trajectoire d'un agent particulier.

3.1.3 Visualisation des interactions entre agents

Il est souvent utile de visualiser les interactions existant entre les agents. Deux agents peuvent être en interaction pour différentes raisons. Un exemple classique consiste à relier les agents se situant dans une même zone. Les deux agents sont alors en interaction si la distance qui les sépare est inférieure à un certain seuil. A l'aide d'un modèle de visualisation il est alors facile de visualiser ces interactions en changeant simplement l'aspect de l'agent et en rajoutant, en plus de son aspect initial, une ligne le reliant avec son voisin. Il est aussi possible d'afficher différents graphes d'interactions selon la valeur du seuil permettant de relier différents agents comme le montre la figure 3.3 où chaque insecte est représenté par un cercle dont le rayon est proportionnel au nombre de connections de l'insecte avec ses voisins. Pour obtenir une telle représentation on rajoute à l'aspect initial de l'agent une fonction permettant d'identifier les insectes se situant à une distance inférieure à un certain seuil. Il reste alors à créer au sein de l'aspect une ligne reliant chaque insecte identifié dans le voisinage à l'insecte concerné et de mettre à jour la taille du cercle représentant l'agent en fonction du nombre d'insectes identifiés dans le voisinage.

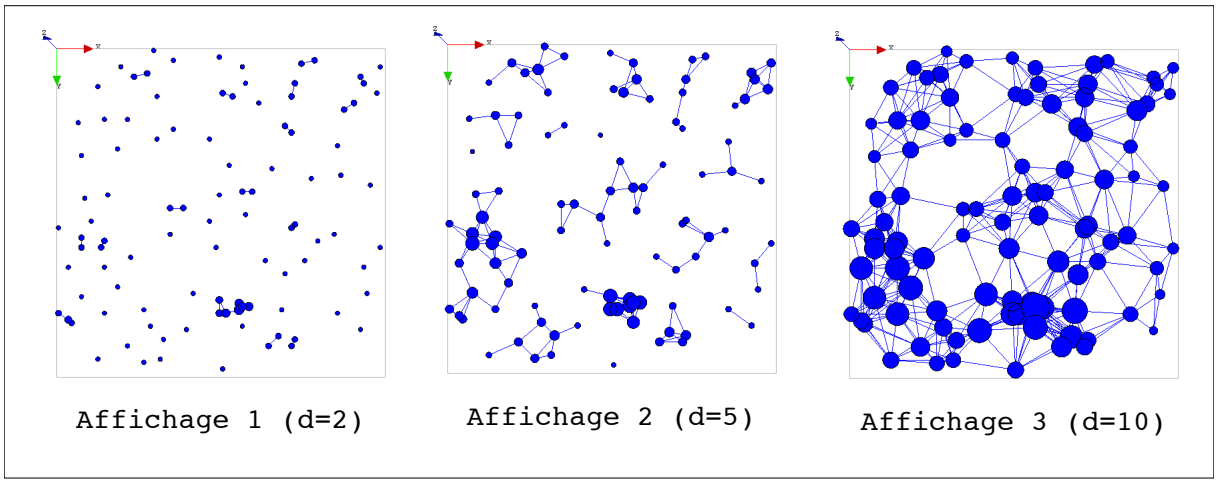

FIGURE 3.3 – Si l'on définit la notion d'interaction entre agent comme résultant d'une distance inférieure à un seuil alors on peut en modifiant ce seuil (respectivement 2, 5 et 10) visualiser le graphe d'interaction et différencier la visualisation des agents de celle de leurs interactions

3.1.4 Superposition d'informations

Comme nous l'avons montré dans la partie 1.2, il est important que l'utilisateur puisse visualiser certaines informations ne faisant pas forcément partie du modèle initial de référence pour différentes raisons, notamment pour vérifier le bon comportement d'un agent ou d'un algorithme. Dans l'étape 11 du tutoriel du "stupide modèle", le comportement de l'insecte est amélioré afin qu'il choisisse la cellule qui possède le plus de nourriture dans son voisinage. Ainsi, il est intéressant de pouvoir afficher la case que l'insecte va choisir afin de vérifier qu'il s'agisse bien de la case ayant le plus de nourriture possible. Dans la figure 3.4, nous superposons à l'affichage original, la cellule la plus fournie que l'agent choisira comme destination et nous l'affichons en orange dans l'affichage 1. Dans l'affichage 2, le voisinage de chaque agent est représenté par un carré blanc afin d'aider le modélisateur à identifier les cellules présentes au sein du voisinage d'un insecte.

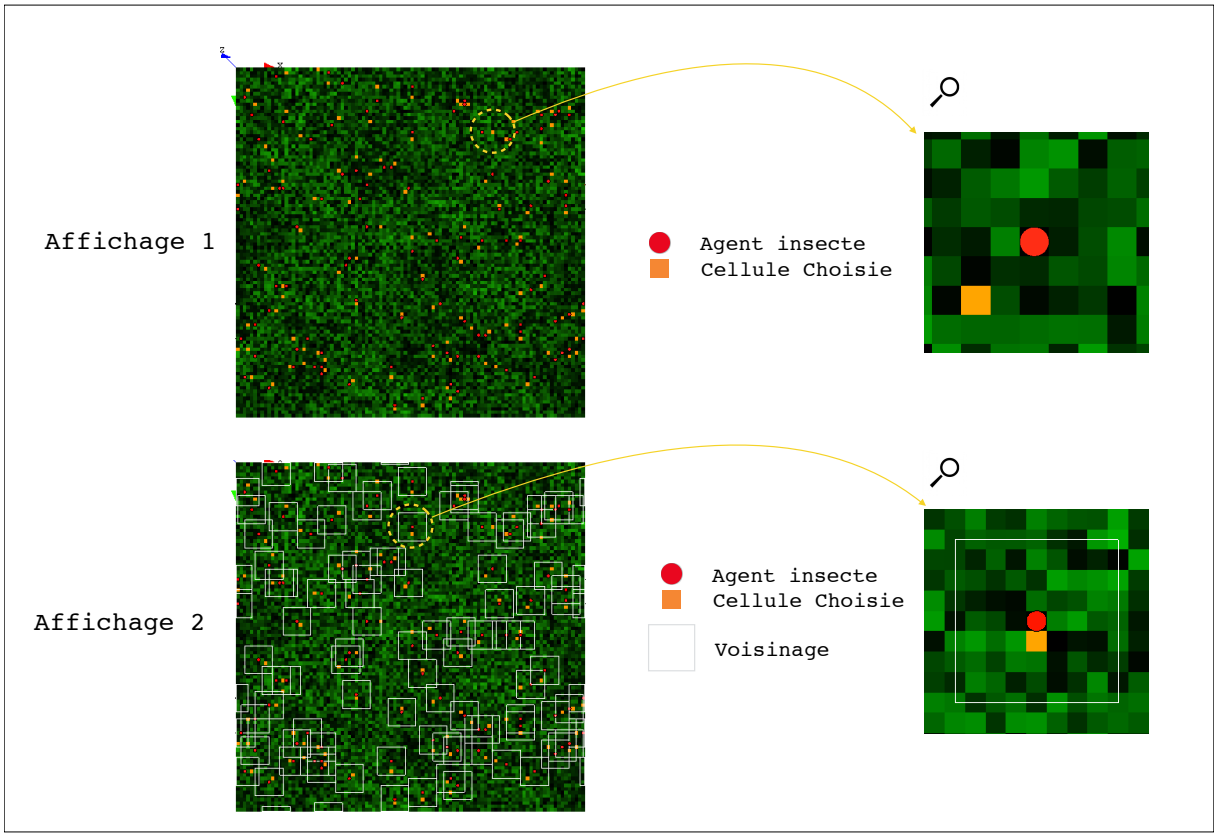

FIGURE 3.4 – Superposition d'informations. Dans l'affichage 1, la cellule choisie par l'algorithme est identifié à l'aide de la couleur orange. Dans l'affichage 2, le voisinage de chaque insecte est représenté par un cadre gris.

3.2 Systèmes d'Informations Géographiques

Les données géographiques vectorielles à grande échelle sont aujourd'hui de plus en plus nombreuses et ce type de données couplé à des outils de simulation permet (1) de créer des modèles proches de la réalité et (2) de bénéficier des outils d'analyses spatiales issus des Systèmes d'Informations Géographiques [Taillandier et al., 2014].

3.2.1 Intégration de données SIG dans un modèle

Le couplage consiste à intégrer des données SIG au sein d'un modèle à base d'agents. A partir d'un fichier SIG, une couche de fond permet de placer les agents sur des objets géographiques jouant le rôle de structure. Les données attributaires des données géographiques peuvent aussi être assignées aux attributs des agents instanciés. Il existe donc une certaine équivalence entre un objet SIG et un agent. Un objet SIG possède une géométrie et des données attributaires de la même façon qu'un agent possède une géométrie (un aspect) et des attributs. En GAML, La commande `create` permet d'instancier les agents à partir des objets SIG en spécifiant le fichier à l'aide de la facet `from`. Les données attributaires peuvent être assignées à des attributs de l'agent à l'aide de la facet *with*, permettant de faire correspondre le nom de l'attribut de l'agent et le nom de la table attributaire du fichier SIG.

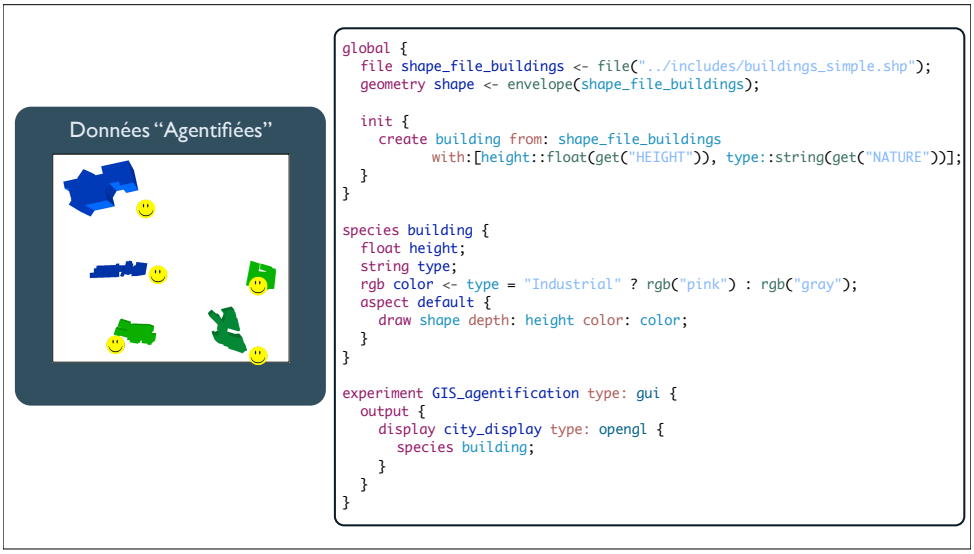

FIGURE 3.5 – A partir du fichier SIG (buildings.shp) nous instancions les espèces *building* et initialisons les attributs *height* et *type* depuis les données attributaires SIG.

Dans la figure 3.6, le modèle est composé de quatre espèces différentes. Les espèces *route* et *rail* sont instanciées à partir de données SIG. Les espèces *voiture* et *train* sont des agents génériques pouvant se déplacer respectivement sur les réseaux routiers et ferroviaires. Une fois les routes et rails instanciés deux graphes sont alors créés afin de permettre le déplacement des agents *voiture* et *train* sur ces deux graphes. L'utilisation de modèles de visualisation permet de représenter facilement ces réseaux sur une même carte, de les superposer dans des plans différents, de les représenter dans différents affichages, etc. Pour les agents provenant de données SIG, la visualisation n'offre rien de plus que ce que peut produire un logiciel SIG. En revanche le fait de pouvoir instancier des agents (voiture, train, etc) et les représenter dynamiquement sur un fond de carte représente une approche nouvelle que peu de plateformes de modélisation offrent aujourd'hui.

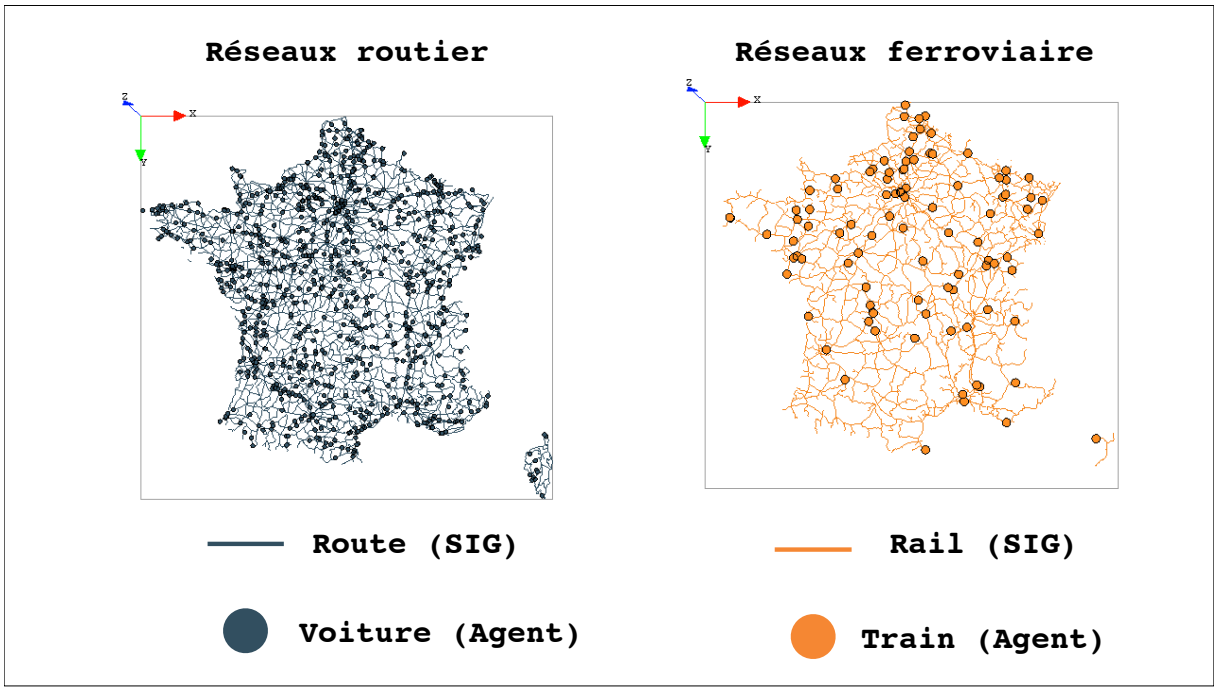

FIGURE 3.6 – Couplage de données SIG et d'agents. Les routes et les rails sont issus de données SIG sur lesquelles sont créés des agents voitures et trains pouvant évoluer sur les objets SIG.

3.2.2 Agentification de données SIG

L'exemple précédent illustrait le couplage d'agents se déplaçant sur un fond de carte SIG. Or considérer chaque objet géographique comme un agent permet de donner du comportement à ces objets. Il est donc envisageable de donner un comportement aux données SIG qui, une fois agentifiées, peuvent s'organiser spatialement pour fournir une nouvelle représentation. De même que dans la visualisation de graphes, où ils existent différentes méthodes pour spatialiser les nœuds et arêtes d'un graphe en fonction de certaines de leurs propriétés, l'utilisation de modèles de visualisation appliqués à des données SIG permet d'imaginer le même genre d'applications mais sur des données SIG. Un modélisateur peut ainsi expérimenter différents algorithmes de disposition en fonction des paramètres qu'il souhaite mette en valeur ou de la question à laquelle il veut répondre.

Le travail présenté dans la figure 3.7 consiste à déconstruire des données spatiales en unités de base pour les réordonner spatialement selon différents critères provenant des données attributaires ou étant calculés à l'aide d'opérateurs spatiaux.

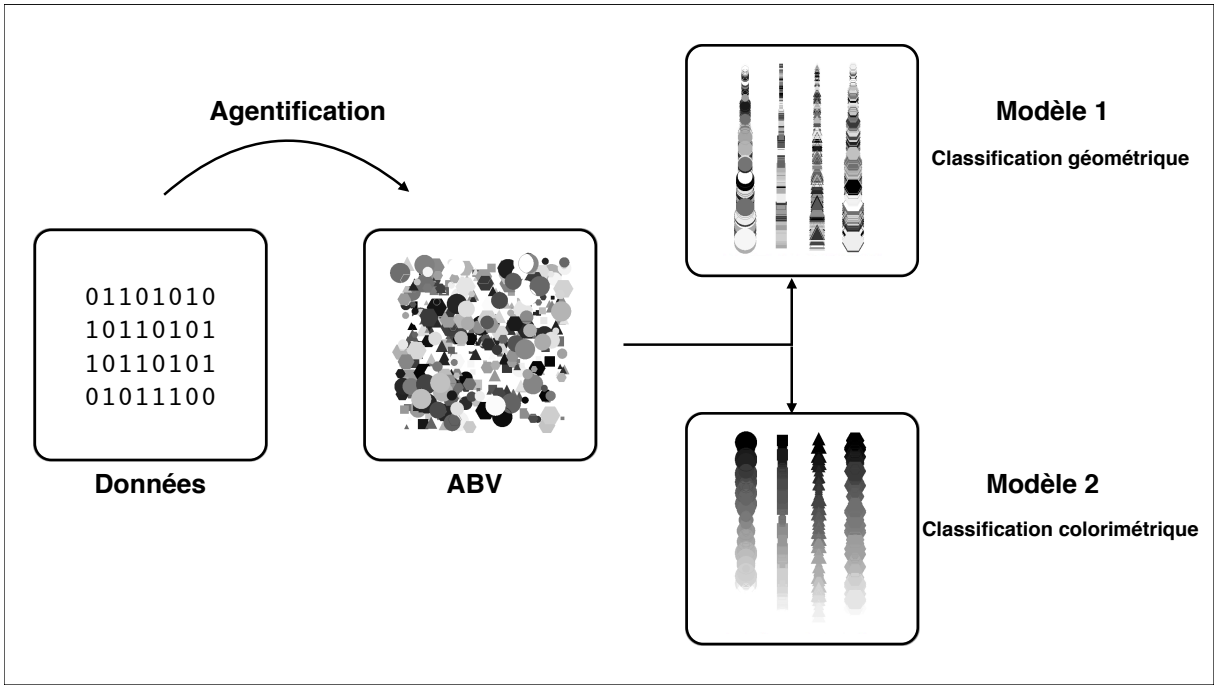

FIGURE 3.7 – Processus de classification `http://youtu.be/Vt0ascqdjL4`. Dans la classification géométrique les agents s'organisent de façon hiérarchique en fonction de leur taille. Dans la classification colorimétrique les agents s'organisent de façon hiérarchique en fonction de leur couleur.

Cette technique est appliquée à un ensemble de données spatialisées représentant des immeubles et des routes. Dans le modèle de référence, les routes et les immeubles sont représentés en fonction de leurs propriétés spatiales de la même façon que dans un SIG. Chacune des espèces a un comportement simple consistant à réorganiser la représentation spatiale en fonction de certains critères spatiaux (surface, volume, etc).

Ce modèle de visualisation est obtenu en appliquant un algorithme de tri appliqué à la fois aux immeubles (en fonction de leur surface) et aux routes (en fonction de leur longueur). Pour les immeubles, le classement est fait de façon verticale. L'immeuble ayant la surface la plus élevée se retrouve à la base de la pile alors que celui ayant la surface la plus petite se retrouvera au dessus de la pile. En ce qui concerne les routes, elles sont triées horizontalement en fonction de leur longueur. La route la plus courte sera affichée à gauche de la pile alors que la route la plus longue sera affichée à droite de la pile. Le modèle de visualisation est représenté dans la figure 3.8.

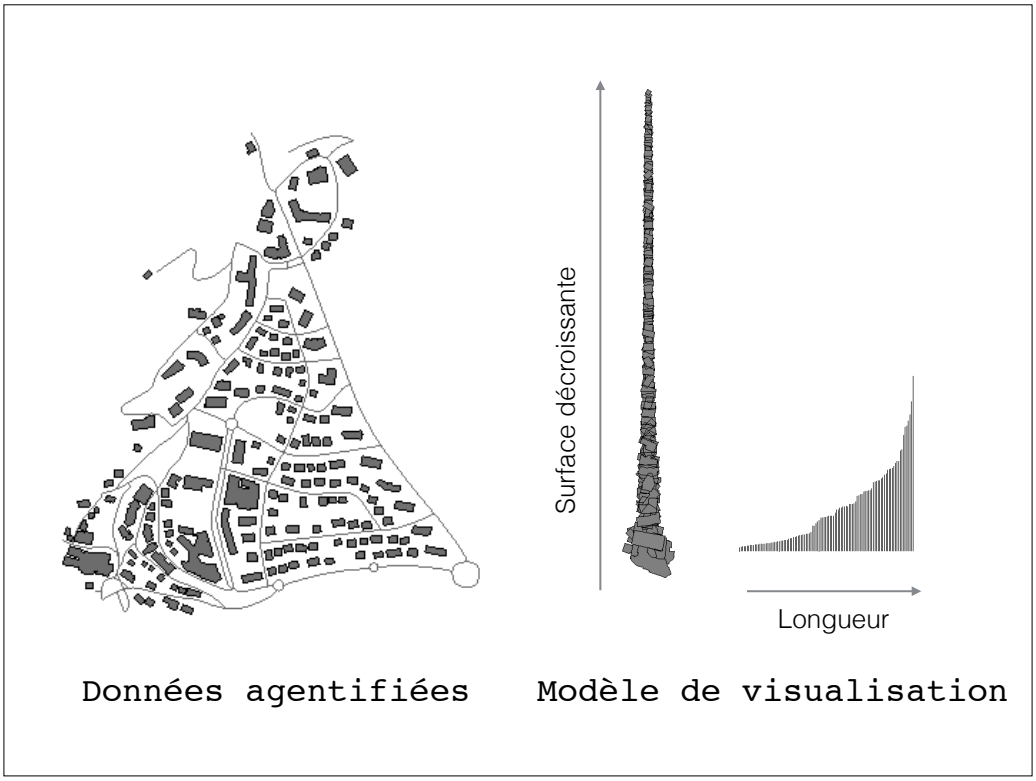

FIGURE 3.8 – La déconstruction spatiale consiste à réorganiser la position des objets SIG selon certaines conditions. Dans ce modèle ABV, les immeubles et les routes se réorganisent spatialement en fonction de leur surface (pour les immeubles) et de leur longueur (pour les routes) `http://youtu.be/FWCHvKGNcq4`.

3.3 Représentation et Modélisation multi-niveaux

Nous avons montré dans la section 1.1 le potentiel qu'offre la modélisation à base d'agents pour étudier les transitions "micro-macro"' ou phénomènes émergents mais qu'il était difficile d'observer cette émergence puisque les entités émergentes n'étaient pas initialement définies dans le modèle de référence. Plusieurs études se sont penchées sur la modélisation multi-niveaux [Gil-Quijano et al., 2012] [Vo et al., 2012] qui consiste à représenter des agents à différents niveaux d'abstraction autant en terme d'abstraction temporelle, que spatiale ou comportementale. Dans cette partie nous montrons comment (1) représenter à l'aide d'agents graphiques ces structures émergentes et (2) comment donner du comportement à ces structures émergentes grâce à l'architecture multi-niveaux offerte par GAMA [Vo, 2012]. Bien que ces deux notions soient proches, nous tenons à les distinguer. Le premier cas, visant à proposer une représentation multi-niveaux, se concentre sur la création de macro-agents graphiques dont l'aspect évolue dynamiquement au cours de la simulation mais n'ayant pas de comportement. Dans le deuxième cas il s'agit d'une modélisation multi-niveaux où les structures émergentes, en plus d'être représentées, ont aussi leur propre comportement. Dans ce cas le passage du micro au macro ne se fait pas simplement de façon visuelle mais au sein d'un modèle de référence intrinsèquement multi-niveaux.

3.3.1 Représentation multi-niveaux

Le modèle de référence est composé d'une population d'agents représentant le niveau micro du modèle de référence. Le modèle de visualisation est quant à lui formé de macro-agents responsables de la détection et de la représentation des groupes d'agents. Cette détection de groupe passe par l'utilisation de techniques de partitionnement des données (*data clustering*), permettant de diviser un ensemble de données en différents groupes [1]. Un opérateur de clustering renvoie une liste de groupes d'agents, eux même représentés comme une liste d'agents. Dans un premier temps, une façon simple de représenter les clusters est de mettre à jour la couleur de chaque agent en fonction de son cluster (deux agents faisant partie du même cluster auront la même couleur). Dans ce cas, il n'est pas

1. GAMA propose différents algorithmes de clustering (k-means [Hartigan and Wong, 1979], dbscan [Ester et al., 1996], spatial, etc.) que nous ne détaillerons pas puisque l'intérêt est ici de pouvoir les représenter sans se soucier de la façon dont ils ont été créés.

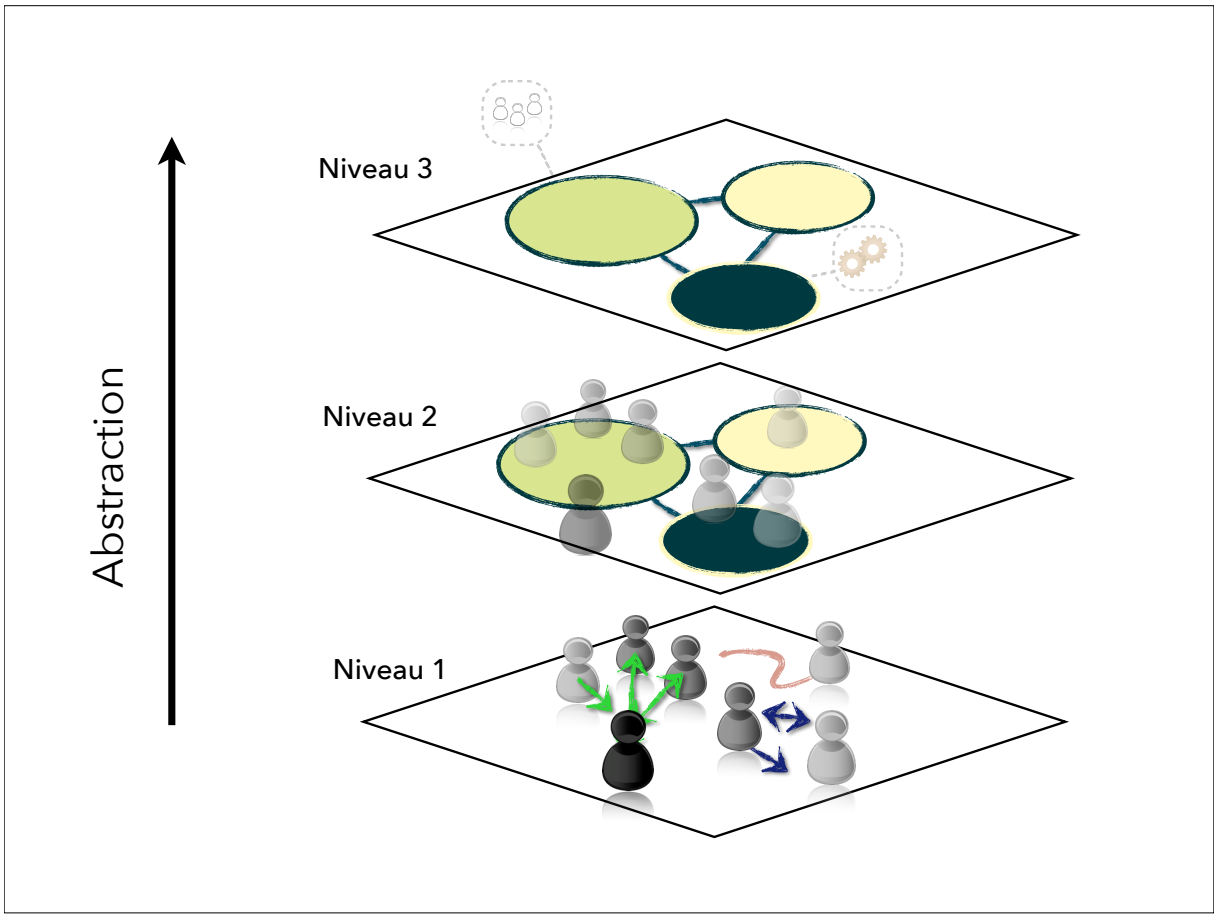

FIGURE 3.9 – Représentation multi-niveaux.

nécessaire de passer par la création de macro-agents. En revanche, si l'on veut identifier et donner un comportement à ces macro-structures, il est nécessaire d'instancier des macro-agents au cours de la simulation. Le modèle de visualisation a donc pour tâche, à chaque itération, de créer la liste de groupes et d'instancier les macro-agents afin de les représenter à l'aide d'un aspect correspondant à l'enveloppe de chacun de ses micro-agents comme le montre la figure 3.10.

3.3.2 Modélisation multi-niveaux

Le langage GAML supporte explicitement la représentation de modèles multi-niveaux, il est alors relativement aisé de produire des outils d'abstraction et d'offrir des modélisations comme celles de la figure 3.11 représentant un modèle à trois niveaux d'organisation. Ce modèle démontre un cas d'utilisation de la fonctionnalité de modélisation multi-niveaux dans lequel des agents *balles* évoluent dans un espace a deux dimensions, ces balles sont

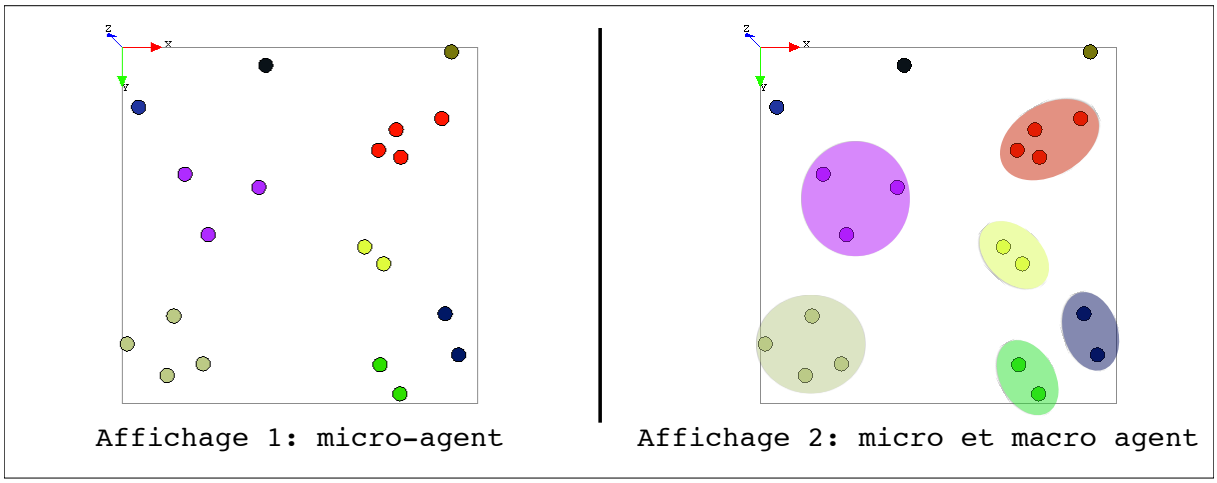

FIGURE 3.10 – Représentation et instanciation de clusters.

dynamiquement agrégées dans des agents *groupes* pouvant eux même être agrégées dans des agents *clouds* selon certaines conditions. Les agents peuvent changer de représentation et de niveau d'organisation grâce aux fonctions *capture*, *release* et *migrate*.

Les balles sont représentées à trois différents niveaux d'organisation : L'espèce *ball* lorsque les balles sont seules. L'espèce *ball in group* lorsque les balles sont membres d'un agent *group*. L'espèce *ball in cloud* lorsque les balles sont membres d'un agent *group* qui est à son tour membre d'un agent *cloud*. Un agent *group* est formé d'un ensemble d'agents *ball* proches. Les groupes sont représentés à deux niveaux différents d'organisation : L'espèce *group* lorsque les groupe sont seuls. L'espèce *group delegation* lorsque les balles sont membres d'un agent *cloud*. Enfin, un agent *cloud* est formé d'un ensemble d'agents *group* proches.

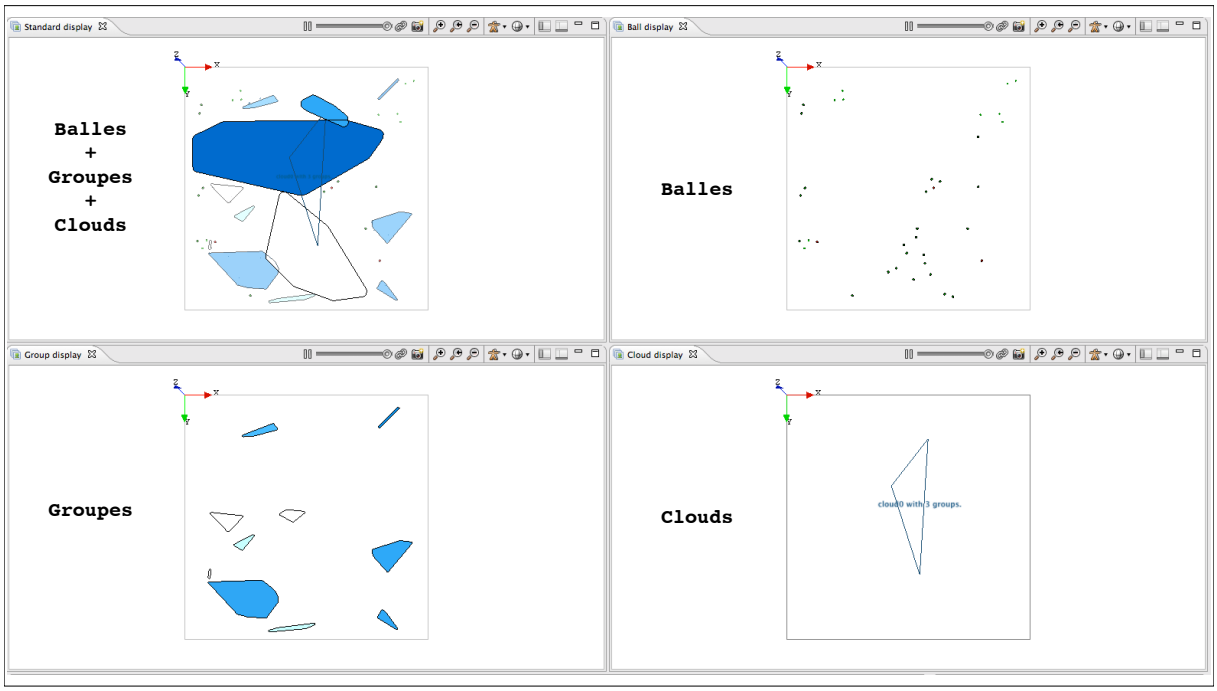

FIGURE 3.11 – Usage des modèles multi-niveaux. Modèle à trois niveaux d'organisation.
https://youtu.be/K5WIw0NzhaE

3.4 Analyse à l'aide d'agrégations spatiotemporelles

La question de la réversibilité d'une simulation pose souvent problème. Il est cependant important, dans certains cas, d'obtenir une trace persistante d'une simulation et d'envisager sa réversibilité. L'utilisation d'agrégation temporelle, si elle ne résout pas le caractère réversible d'une simulation, permet en revanche d'en garder une trace visuelle. Nous présentons ici une façon simple d'abstraire le temps permettant de passer d'une classique à une représentation abstraite représentant le système sur l'ensemble de la période de simulation. En visualisation classique, les agents sont représentés à chaque itération à leur position courante. A chaque itération, la position de l'agent est mise à jour et redessinée à l'étape suivante. Cependant pour bien comprendre le comportement d'un modèle il peut parfois être nécessaire de garder une trace des itérations précédentes. Dans un affichage agrégé, à chaque itération, la position de l'agent est remise à jour sans effacer la précédente. Une telle technique permet un retour d'information immédiat et visuel sur des processus dynamiques ayant lieu au cours d'une simulation et ne pouvant être capturés sur une représentation instantanée classique [Grignard et al., 2013a].

La figure 3.12 montre le résultat d'un affichage agrégé sur un modèle classique de nuées d'oiseaux [Reynolds, 1987] où l'on affiche à chaque itération la position du barycentre de tous les oiseaux (en bleu) afin de visualiser et analyser leur trajectoire par rapport à celle de la cible (représentée par un triangle dont la couleur évolue en fonction de l'angle de rotation de la cible). Cette technique permet à l'utilisateur de (1) rapidement ajuster et de valider la précision de ce modèle en comparant les deux trajectoires de façon visuelle et (2) d'évaluer la qualité de son modèle à l'aide d'une mesure fournie par les agents graphiques à chaque itération en calculant par exemple la distance séparant le point bleu de la cible.

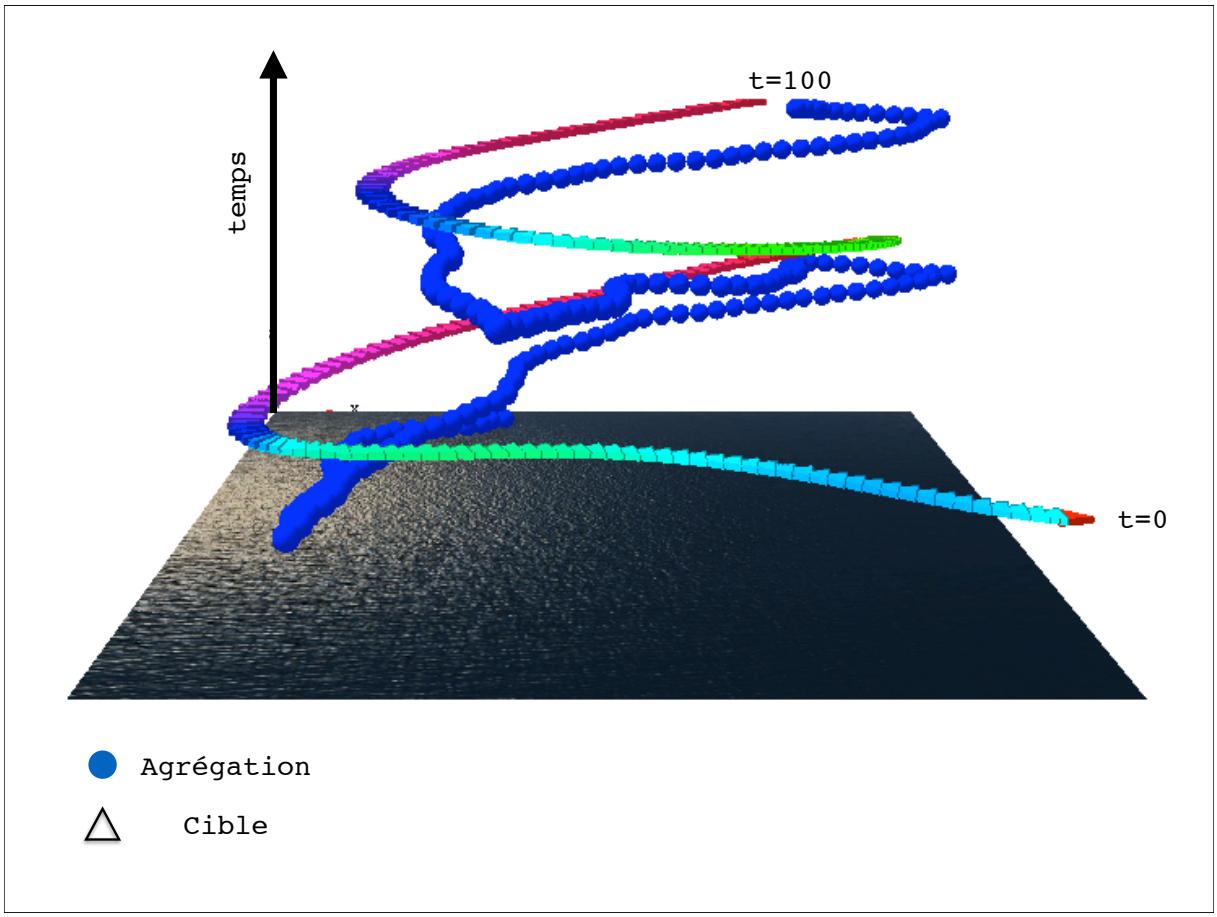

FIGURE 3.12 – Agrégation temporelle. http://youtu.be/mGkWWjP3zr4

Dans la figure 3.13, sur un modèle épidémiologique SIR classique, nous superposons le graphe de contact à l'aide de la troisième dimension. A chaque fois qu'un agent devient infecté, un agent graphique est créé à la position x,y de l'agent infecté et à une position z reflétant le temps à laquelle l'individu a été infecté. Ainsi, de la même manière que ce qui a été proposé dans la partie 3.3.1, en utilisant des algorithmes de clustering spatiaux en 3D sur les agents graphiques, il est alors possible d'isoler spatialement les zones d'infection en les représentant par une sphère englobant la zone.

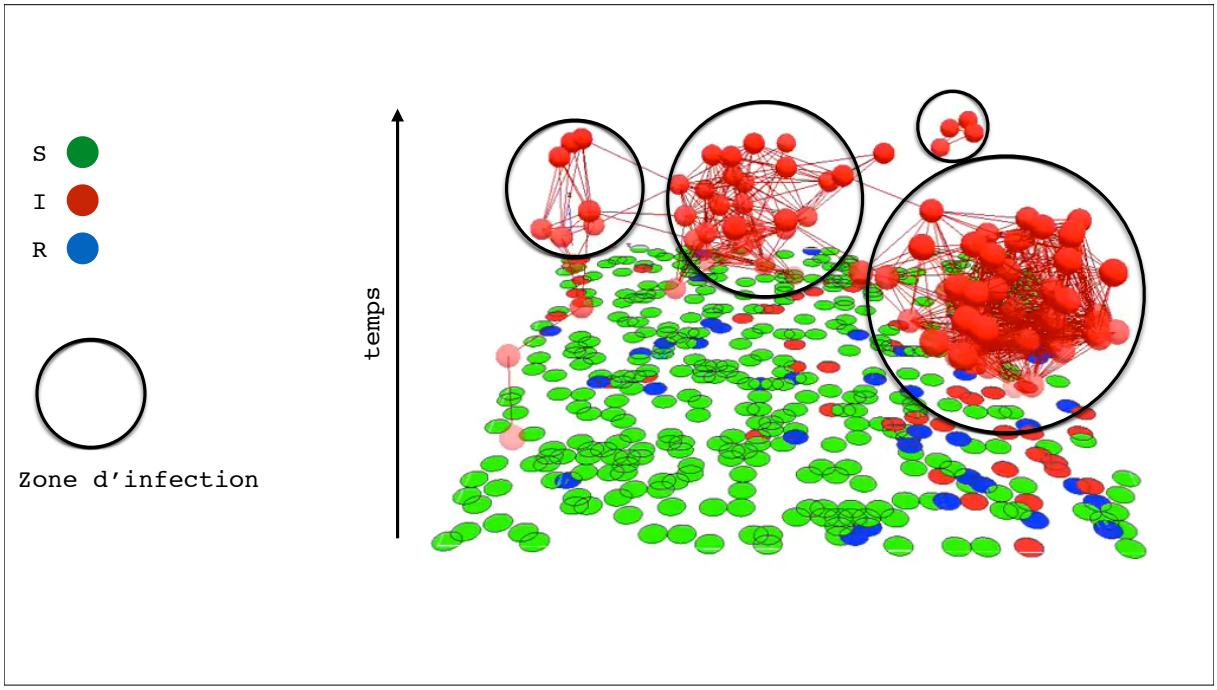

FIGURE 3.13 – Détection de zones d'infection et visualisation agrégée d'un modèle SIR
http://youtu.be/bFBk8mzhtGI

3.5 Contrôle du modèle à l'aide d'abstractions

Dans la figure 3.14, le modèle de référence sur la gauche représente des agents reliés entre eux en fonction de la distance qui les sépare. Chaque agent est représenté par une couleur représentant sa classe. La vue de droite est une vue abstraite du modèle de référence. Il s'agit d'un réseau agrégé crée à l'aide des opérateurs décrits dans la partie 2.4.1. Dans le modèle agrégé, les agents d'une classe donnée (donc d'une certaine couleur) sont agrégés dans un macro nœud représentant cette classe dont le rayon évolue en fonction du nombre d'agents qu'il agrège. Le macro lien créé entre un macro agent A et un macro agent B correspond au nombre total de liens entre les agents de classe A et les agents de classe B dans le modèle de référence. L'épaisseur du lien évolue en fonction du nombre de connections entre les agents du modèle de référence. Dans une telle approche, les macro nœuds peuvent agir sur les agents qu'ils agrègent. Un contrôleur est attaché à chaque macro nœud et macro arête pour contrôler tous les micro agents agrégés (à l'aide des fonctions définies dans la partie 2.4.3) . Une abstraction, en tant qu'agent, n'est plus vue comme un simple indicateur permettant de mettre en valeur des structures émergentes mais aussi en tant qu'objet permettant de contrôler le modèle de référence.

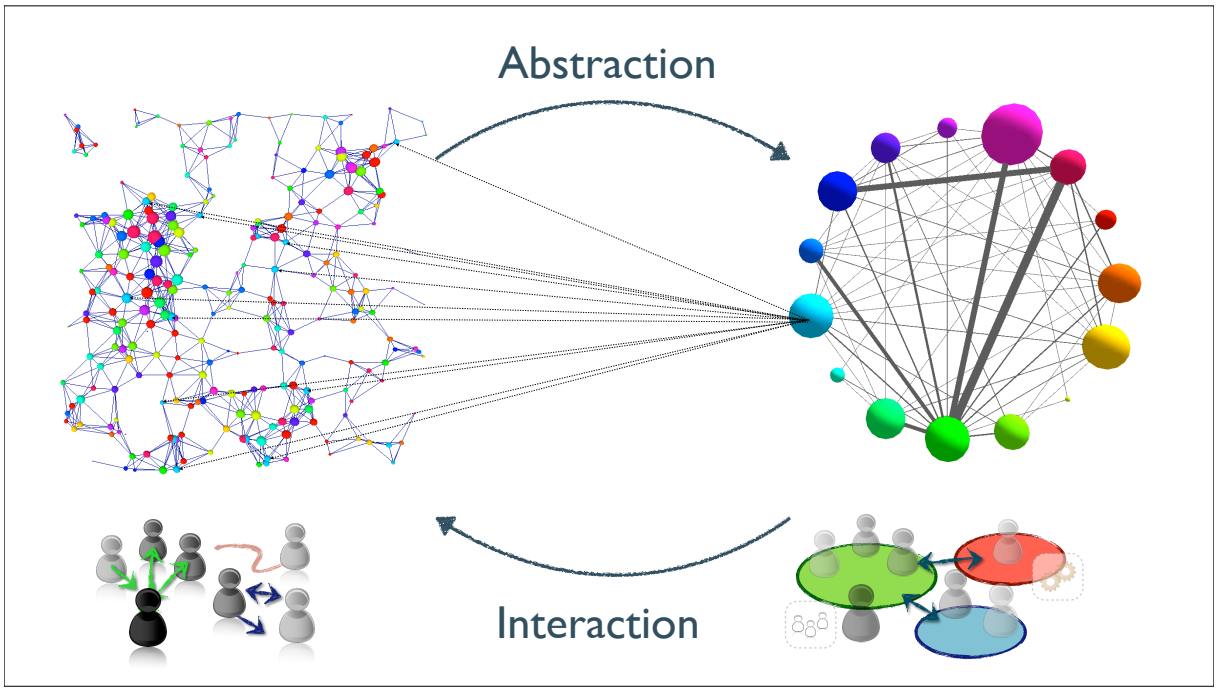

FIGURE 3.14 – Contrôle à l'aide d'abstraction

Conclusion

Au cours de ce chapitre nous avons cherché à illustrer à l'aide de différents exemples, que se soit à l'aide d'exemples génériques ou de modèles classiques et connus de la communauté agent, l'intérêt que peut offrir notre approche en termes de représentation, d'abstraction et de contrôle. Nous avons choisi de rester dans le domaine du modèle-jouet afin de bien mettre en valeur les concepts défendus. Évidemment le lecteur doit garder en tête que ces exemples constituent une panoplie d'exemples joués et qu'ils ne sont pas une fin en soi mais une façon d'illustrer les concepts. Libre à lui de réutiliser ces concepts pour une application en particulier. Nous ne cherchons pas a donner une liste exhaustive des possibilités offertes par cette approche, car ceci reviendrait à vouloir donner une liste exhaustive de ce qu'il est possible de faire en modélisation à base d'agents. Nous tenons simplement a rappeler que l'un des problèmes pour l'expérimentateur réside dans le fait que les mesures qu'il essaie de faire sur un modèle deviennent très rapidement abstraites. Or, dans notre proposition, notamment dans la partie 3.4 sur l'analyse en temps réel de simulation, ces mesures restent concrétisées tout au long de la simulation. Cette "concrétisation" visuelle de la mesure permet de comparer rapidement ce qu'il se passe dans un modèle et aussi, comme nous le verrons dans la partie suivante, sur un jeu de données. Cette visualisation permet aussi d'affiner la façon dont on fait une mesure, permettant ainsi d'extraire visuellement le bon opérateur de calcul et d'éventuellement le réinjecter dans le modèle de référence. La visualisation ne doit pas seulement se voir comme une fin en soi mais comme un outil permettant d'améliorer le modèle de référence. Ainsi l'intérêt d'avoir des agents de visualisation, outre le fait qu'ils rendent la représentation pertinente, permet d'interagir visuellement de façon à créer dynamiquement le processus d'agrégation le plus efficace, ou en tout cas, qui semble le plus efficace. C'est cette généricité qui va devenir le cœur de l'approche proposée dans la deuxième partie de cette thèse où nous allons appliquer les différents concepts proposés ici à un domaine plus général à savoir la visualisation de données ou de manière encore plus générale la visualisation d'informations.

Chapitre 4

Visualisation d'informations

> Because of its ability to process enormous amounts of data, the human visual system lends itself as an exceptional tool to aid in the understanding of complex subjects.
>
> Ben Fry

Introduction

Nous avons montré comment l'approche de visualisation à base d'agents pouvait bénéficier à l'activité de modélisation, offrant des possibilités d'aller-retour inédites entre les abstractions présentes dans les modèles et les abstractions graphiques de visualisation, mais la visualisation de modèles peut être vue aussi comme une spécialisation du domaine plus vaste de la **visualisation d'informations**. Celui-ci regroupe toutes les approches visant, par l'emploi de métaphores visuelles et de mécanismes d'interaction homme-machine, à donner du sens à un ensemble de données, en particulier [pour] mieux rendre compte de sa structure et de son éventuelle dynamique [Card et al., 1999] [Lau and Vande Moere, 2007] [Pettersson, 2002] [Ware, 2012] [Judelman, 2004] [1]. Ce domaine en plein essor propose ainsi différentes façons d'analyser ces données, de les classifier, de les résumer et de les caractériser visuellement à l'aide de techniques de fouille de données [Andrienko et al., 2003] [Compieta et al., 2007] telles que les statistiques ou l'ap-

1. http ://design.osu.edu/carlson/history/ : une histoire critique de l'infographie et de l'animation

prentissage automatique [Han et al., 2006] [Keim et al., 2008], mais aussi à l'aide d'outils de visualisation interactifs. Cependant, comme nous le décrivons dans la partie suivante, les méthodes actuelles de visualisation d'informations, trop souvent basées sur des solutions purement technologiques, manquent encore de flexibilité, d'adaptabilité ou de modularité.

Notre approche de visualisation à base d'agents, même si nous ne l'avons pour l'instant envisagée et présentée que dans le cadre de la visualisation de modèles, s'appuie sur la conception d'agents graphiques potentiellement autonomes, qui peuvent interagir et s'organiser entre eux, et qui ont à ce titre la capacité d'apprendre à partir des données qu'ils traitent et d'adapter en conséquence leurs comportements et leurs représentations visuelles.

Ce chapitre a pour objectif de montrer comment et sous quelles conditions cette approche peut être généralisée à des données arbitraires (et non plus, simplement, à des sorties de modèles) et permettre ainsi à l'utilisateur, par la conception de modèles de visualisation à base d'agents [filtrant] ces données, de bénéficier des qualités précédemment citées pour représenter et explorer, programmatiquement et visuellement, leurs dynamiques sous-jacentes. Nous montrons en particulier que notre approche remplit parfaitement le cahier des charges d'une représentation visuelle correcte des diverses formes que peut prendre une information structurée, condition nécessaire pour pouvoir prétendre à s'appliquer à tout type de données.

4.1 Limites des méthodes et outils existants

Il manque à l'heure actuelle un système suffisamment **flexible** en terme de rendu graphique, **modulaire** pour permettre la réutilisation de composants sur différentes applications et enfin **adaptable** à la variété et la dynamicité des données à traiter. La plupart des solutions existantes manquent de méthodologie permettant à un utilisateur novice de s'approprier un jeu de données et de le représenter visuellement par une approche incrémentale et interactive.

La complexité du processus analytique existant dans les systèmes de visualisation d'informations reste encore un obstacle majeur. L'utilisateur a encore du mal à comprendre comment les données brutes sont transformées en représentations visuelles. Il est donc important de proposer des méthodologies liées à la visualisation d'informations.

Il existe de plus en plus d'outils permettant de mener à bien une visualisation sur le plan du contenu (information) et dans la stratégie de mise en forme (graphisme) mais il manque encore clairement d'outils génériques et de méthodologies pour effectuer des tâches de visualisation classiques qui sont trop souvent réalisées de manière *ad-hoc*. Ces techniques peuvent être regroupées en trois catégories ayant chacune des avantages et des inconvénients rendant la tâche de visualisation de données un processus encore compliqué pour la plupart des scientifiques.

La première catégorie consiste à utiliser un tableur grapheur pour manipuler des feuilles de calcul et produire des graphiques pour représenter les jeux de données. On parle alors de graphiques à barres, à courbes ou à secteurs. Les tableurs restent des outils puissants et encore très utilisés pour beaucoup de tâches classiques de visualisation de données. Le problème des tableurs est le peu de contrôle existant sur la forme de la sortie graphique, contrainte par des canevas bien définis. De plus l'utilisation de tableurs implique de traiter des jeux de données classiques pouvant être représentés en lignes et en colonnes ce qui comme nous l'avons vu dans la partie 4.2 est loin d'être le cas pour tous les jeux de données. En ce qui concerne les données spatiotemporelles décrites dans la partie 4.2.5 il reste encore difficile d'animer une représentation graphique dans un tableur.

La deuxième catégorie consiste à dessiner à la main une représentation à l'aide d'un logiciel dédié. Cette technique reste la plus efficace pour obtenir des rendus très graphiques et faire passer un message clair. La plupart des visualisations récentes d'informations sont réalisées à l'aide de ces outils mais ils restent réservés à des spécialistes du design.

Ainsi, sur un point de vue graphique, cette technique permet d'imaginer une infinité de représentations puisque aucune limite n'est imposée par le logiciel. Les seules limites imposées sont liées aux compétences techniques. En revanche, son manque d'adaptabilité est l'une des principales raisons qui en fait un outil peu générique. En effet, si un utilisateur veut produire une visualisation capable de s'adapter ou de représenter un nouveau jeu de données, la seule solution est de redessiner le graphique. En effet il est difficile de créer des formes dynamiques sans utiliser de script ou de code. Les logiciels dans l'esprit de Adobe Flash [1] permettent la manipulation de graphiques vectoriels et la création de scripts en langage ActionScript et permettent de créer facilement des animations et des outils interactifs mais ils ne sont pas dédiés spécifiquement à) la visualisation de données.

La troisième catégorie consiste à écrire du code à l'aide d'une librairie adaptée (D3 [4], threejs [13], Processing [10], OpenGL [8], max[6] ou encore prefuse [9]). Une idée conceptuelle traduite en code et interprétée graphiquement par l'ordinateur permet de faire émerger des images sans avoir à les dessiner à la main. La génération de forme dont on détermine le comportement de façon programmatique est souvent regroupée sous la notion de design génératif [Bohnacker et al., 2012] [Shiffman et al., 2012] [Galanter, 2003]. Cette solution a permis de concrétiser des idées de façon interactive en utilisant des techniques sophistiquées à l'aide de configurations multiples mais elle nécessite de grandes compétences en programmation et manque de cadres méthodologiques bien définis.

4.2 Structures de l'information et approche ABV

La présentation visuelle d'informations fait intervenir une multitude de principes théoriques, analytiques et graphiques. Nous décrivons dans cette partie les notions-clés liées au design de l'information et à la façon de représenter visuellement l'information en nous appuyant sur l'ouvrage récent de [Meirelles, 2013]. Pour chaque type de structure nous détaillerons comment la représenter de façon générique et montrerons comment l'approche ABV permet cette représentation.

4.2.1 Structures Hiérarchiques

Spécifications

On appelle système hiérarchique un "système composé de sous-systèmes reliés entre eux, dont chacun présente à son tour une structure hiérarchique, jusqu'à atteindre quelque niveau inférieur de sous-systèmes élémentaire" [Simon, 1965]. Les surfaces réduites des écrans d'ordinateur nécessitent de nouveaux modes de représentation pour favoriser la lisibilité des représentations hiérarchiques. Souvent les méthodes de représentation hiérarchique font appel à l'interactivité pour naviguer entre les différents niveaux d'arborescence. Différents modes de représentation existent comme les graphes composés de nœuds et d'arêtes, les arbres coniques [Robertson et al., 1991] ou hyperboliques [Lamping and Rao, 1996], les dendogrammes , les visualisations hierarchiques radiales [Keim et al., 2006] ou les treemap [Johnson and Shneiderman, 1991]. Dans ce type de représentation, l'utilisation de l'espace est le plus souvent schématique. Les propriétés géométriques et les relations spatiales reflètent les propriétés et relations des données initiales. Les technologies de visualisation 3D permettent aussi une représentation dans un espace 3D afin d'optimiser l'utilisation de l'espace disponible à l'écran et permettre la visualisation de l'ensemble de la structure.

Approche ABV

La partie 3.3.1 sur la représentation multi-niveaux permet de visualiser des structures hiérarchiques grâce à la définition de micro espèces comme nous l'avons illustré dans la figure 3.10 et 3.11. Dans la figure 4.1, nous illustrons de façon générique une représentation hiérarchique dans un modèle ABV. Nous décrivons ici un modèle à trois niveaux formé d'une espèce **cloud**, d'une espèce **group** et d'une espèce **individual**. Grâce à l'architecture multi-niveaux, la représentation de structures hiérarchiques consiste à définir différents niveaux d'organisation. L'espèce **cloud** contient des **group** qui eux-même contiennent des **individual**. Cette approche est récursive et permet la définition d'une infinité de niveaux. Il est donc tout à fait envisageable de définir une quatrième espèce **cellule** imbriquée dans l'espèce **individual**.

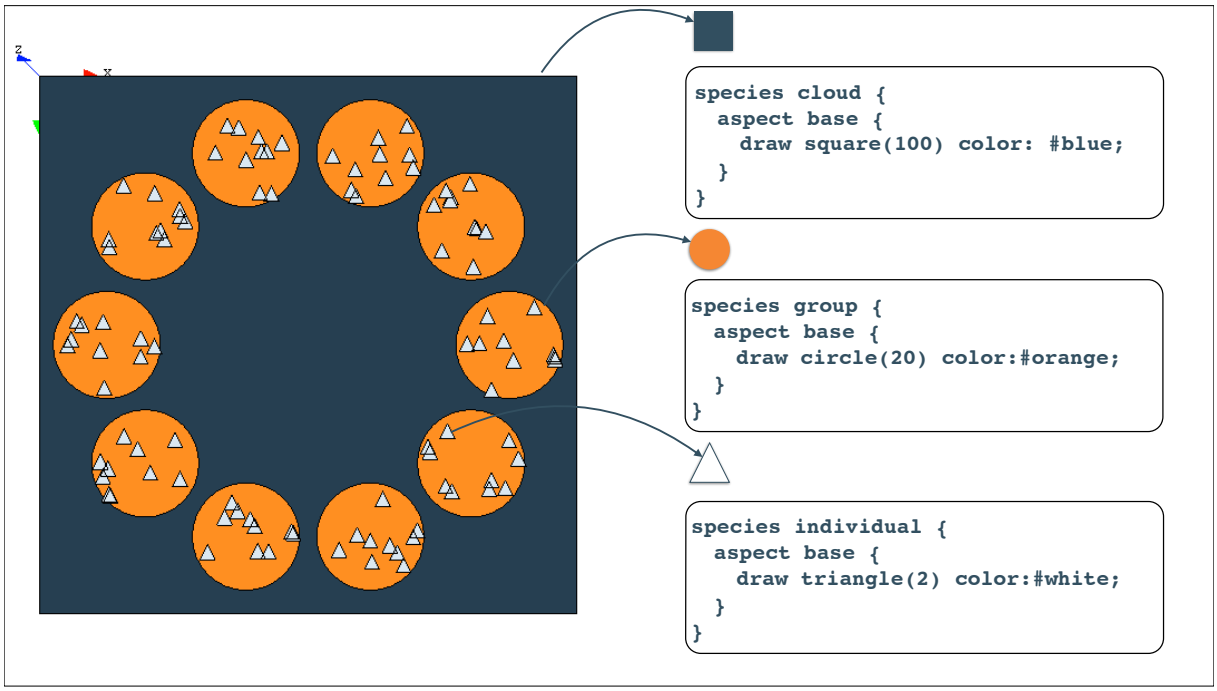

FIGURE 4.1 – Structure hiérarchique à trois niveaux : L'espèce **cloud** contient des **group** qui eux-même contiennent des **individual**

4.2.2 Structures Relationnelles

Spécifications

Les structures relationnelles représentent les données en fonction de leurs connexions. Dans le milieu des sciences sociales, [Shneiderman et al., 2010] donne la définition suivante : "L'objet de l'analyse des réseaux sociaux se situe entre - et non dans - les personnes. Alors que les méthodes traditionnelles de recherche en sciences sociales s'intéressent aux individus et à leurs attributs, la science des réseaux s'intéresse aux connexions entre les individus". Les structures relationnelles, réseaux ou graphes se retrouvent dans différents domaines comme le métabolisme humain, les réseaux sociaux ou encore les réseaux de transports [Albert and Barabási, 2002]. La visualisation joue un rôle clé dans l'étude des réseaux en ajoutant une dimension visuelle et intuitive à l'analyse numérique. Dans un graphe, un nœud peut représenter différentes entités comme une personne ou une cellule, une arête matérialise la relation entre deux nœuds. On parle de réseau monomodal lorsque les nœuds de ce réseau sont tous de même type et de réseau multimodal lorsqu'il y a plusieurs types de nœud. Les liaisons représentent tous les modes d'interaction entre nœuds et peuvent être représentées par des attributs visuels précisant la direction de l'in-

teraction dans les graphes orientés ou son importance dans les graphes pondérés ou par des connexions symétriques dans les graphes non orientés.

Il existe plusieurs manières de représenter les réseaux. Les listes d'adjacence représentent la structure du réseau sous forme textuelle mais sont rarement utilisées. Une matrice d'adjacence est une grille dont les cellules indiquent la présence ou l'absence de liaison entre deux nœuds. Un encodage visuel sur les cellules de la grille permet de représenter des attributs supplémentaires en plus de la simple existence de la liaison. Dans la visualisation de graphes classique les nœuds sont représentés par des symboles, les arêtes par des lignes. Certains graphes comportent une dimension spatiale contraignant la structure du diagramme. Lorsque l'on représente des données abstraites, le positionnement des nœuds n'est pas lié à une propriété spatiale. La visualisation de réseaux est un domaine de recherche très actif et permet de disposer de plusieurs modes de représentation afin de révéler les propriétés significatives d'un réseau [2].

Approche ABV

La partie 3.1.3 a déjà illustré comment représenter les interactions entre agents. Un graphe peut être représenté par un modèle ABV composé d'une espèce **node** et d'une espèce **edge**. L'aspect de chacune de ces espèces permet de mettre en valeur certains paramètres comme le degré du nœud (son nombre de connexion) ou le poids d'une arête comme le montre la figure 4.2. Dans le cas de graphe monomodal, les nœuds auront un aspect similaire, dans le cas de graphe multimodal l'utilisation de variables visuelles comme la forme, la taille ou la couleur permet de différencier le type de nœud. Pour les arêtes, la direction de l'interaction peut être représentée par une flèche, le type d'interaction par l'utilisation de différents couleurs.

2. De nombreux procédés et algorithmes existent pour améliorer la lisibilité des graphiques [Herman et al., 2000] mais nous ne rentrons pas dans leur détails ici.

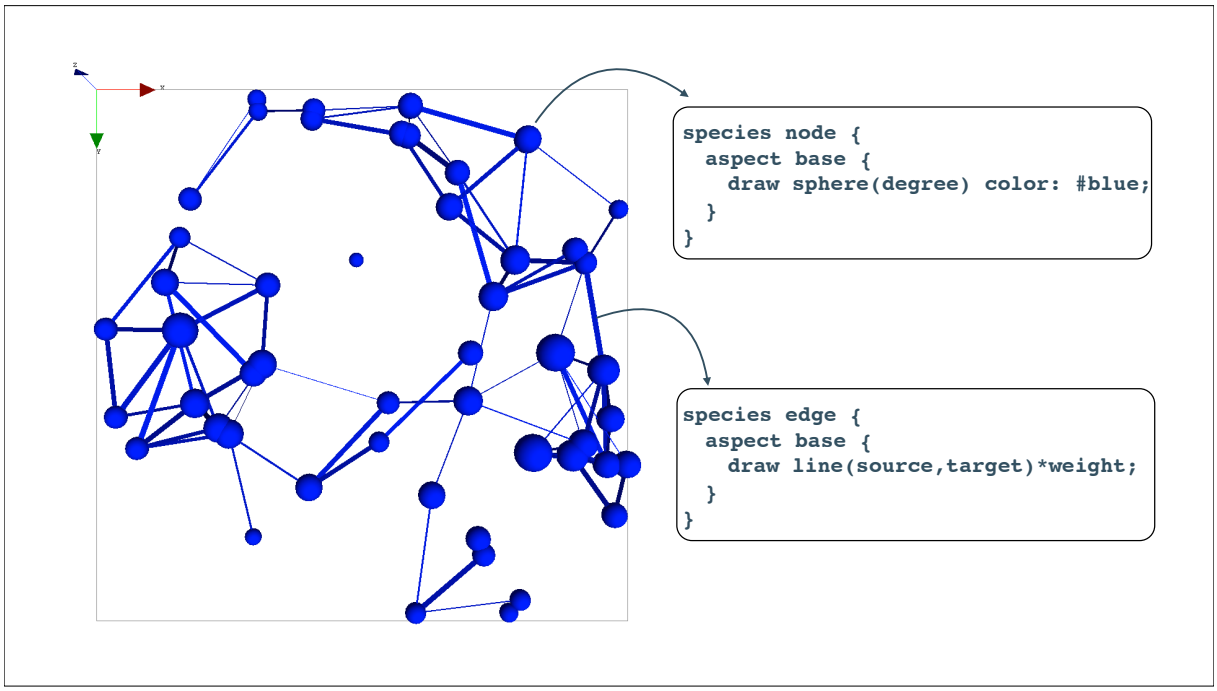

FIGURE 4.2 – Structure relationnelle. La représentation du graphe passe par la création d'une espèce **node** dont l'aspect est une sphère dont la taille est proportionnelle au degré du nœud et d'une espèce **edge** dont l'aspect est une ligne reliant deux nœuds et dont l'épaisseur est proportionnelle au poids de l'arête.

4.2.3 Structures Temporelles

Spécifications

Le temps est un concept abstrait et non visuel par nature [Lakoff and Johnson, 2008]. Cette conception abstraite du temps influe sur la façon de le représenter. Les représentation oscillent entre représentation linéaire et représentation cyclique [Gould, 1987]. Sur une représentation statique, le temps est le plus souvent représenté sous forme de frise ou de ligne du temps, où la position des éléments donne une indication sur le temps qui les sépare et dans lesquelles des données quantitatives remplacent les événements en tant que fonction du temps. Si le monde moderne accorde une prédominance à la représentation linéaire, le modèle cyclique permet en revanche de révéler la périodicité de certaines données. Le format numérique permet lui de naviguer à travers le temps en faisant défiler les structures linéaires ou cycliques. Différentes sources de données peuvent être regrouper thématiquement sur l'axe vertical. Les outils interactifs offerts par le numérique permettent de moduler l'échelle temporelle et ainsi d'explorer

plus en détails des périodes spécifiques. Ce système graphique composé d'échelles chronologiques, d'indicateurs temporels et de sections thématiques est un outil puissant pour la représentation statique de données temporelles. Ces représentations peuvent aussi suivre le modèle de l'arborescence décrit dans le section 4.2.1.

Approche ABV

La figure 4.3 est une manière de représenter sur un support statique une évolution temporelle. C'est l'utilisation des couches, décrite dans la partie 2.3.1 qui permet d'obtenir si facilement une représentation linéaire du temps puisque leur position est simplement décalée à chaque pas de temps par la largeur de l'environnement permettant ainsi une représentation séquentielle. L'utilisation de la facet **trace** permet de garder à l'écran chaque cycle.

Dans un modèle ABV animé, la composante temporelle est intégrée dans la notion de pas de temps de la simulation. L'exécution de la simulation fait apparaitre les éléments correspondants à un pas de temps donné. Le pas de temps de la simulation est alors en lien direct avec l'échelle temporelle du phénomène que l'on désire étudier. L'utilisateur peut spécifier le pas de temps qu'il désire et le faire varier au cours de la simulation. Nous illustrons ces concepts plus en détails dans la partie 5.2.1.

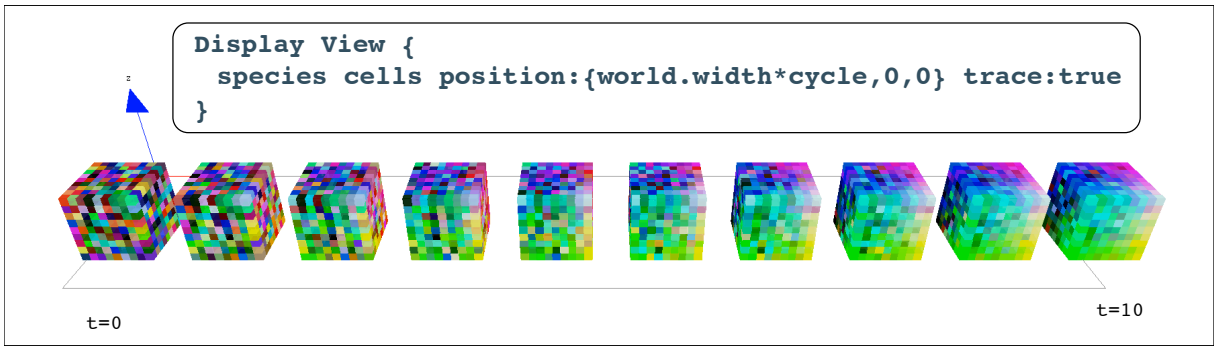

FIGURE 4.3 – Structure temporelle : Représentation linéaire du temps. A chaque itération, la position de la couche représentant les agents est décalée de la taille de l'environnement. La facet `trace` permet de garder la représentation de chaque itération de la simulation afin de garder une trace temporelle de la simulation.

4.2.4 Structures Spatiales

Spécifications

Les données spatiales, aussi connues sous le nom de données géoréférencées, ou de Système d'Information Géographique (SIG) permettent d'associer à chaque entité une structuration des données composée de données spatiales (points, lignes, polygones) [Joliveau, 1996] et de données attributaires. Grâce à l'utilisation d'outils comme GoogleEarth API [Brown, 2006] ou OpenStreetMaps [Haklay and Weber, 2008], de plus en plus d'applications encouragent la spatialisation des données (ArcGIS [Johnston et al., 2001], QGIS [QGis, 2011], R [Bivand et al., 2008]). On regroupe sous le nom de carte thématique [3], les cartes dans lesquelles des données d'attributs à la fois quantitatifs et qualitatifs sont disposées sur un fond topographique. Le fond topographique correspond aux données de localisation représentées à l'aide d'un système de positionnement précis (latitude, longitude, projection, etc). La cartographie fait appel à trois notions principales : la projection, l'échelle et la symbolisation. La projection est une opération mathématique permettant de passer d'une représentation sphérique à une représentation plane, impliquant souvent une perte d'informations se traduisant par une déformation géométrique selon la projection choisie (angles, aires, formes, distances, directions). L'échelle est le rapport entre une distance sur la carte et celle qui lui correspond sur terre. Le choix de l'échelle dépend du type d'information à faire apparaitre (région, pays, monde, etc). La perte de détails qu'implique le changement d'échelle est souvent regroupée sous le terme de généralisation ou de degré de généralisation. Dans ces représentations, les symboles représentent parfois un espace plus grand que ce qu'il couvrent réellement. Le fond de carte utilisé et les données thématiques qui y sont représentées doivent en général utiliser un degré de généralisation proche. L'encodage visuel (ou symbolisation) permet de choisir le mode de représentation le plus approprié selon les phénomènes à faire apparaitre comme nous l'avons mentionné dans la parties 2.3.1.

On distingue plusieurs façons de représenter les données spatiales :

Les *cartes à symboles proportionnels* jouent sur la variable visuelle de la taille pour représenter des rapports de quantités comme le montre la figure 4.4. Les données sont réparties par tranche de valeur, ou discrétisées ou pour les données non classées, ou continues, représentées proportionnellement.

3. Pour un historique de la cartographie, se référer à [Friendly and Denis, 2008]

La *carte choroplèthe* utilise des propriétés graphiques telles que la couleur ou la transparence pour illustrer différentes propriétés comme dans la figure 4.5, représentant la densité de population aux Etats-Unis [Yau, 2011]. Les cartes choroplèthes représentent des données agrégées par secteur, encodées par des aplats de couleur couvrant la surface de chacun de ces secteurs. Comme nous l'avons vu dans la partie 2.3.1, le choix des couleurs dépend beaucoup du type d'information à diffuser.

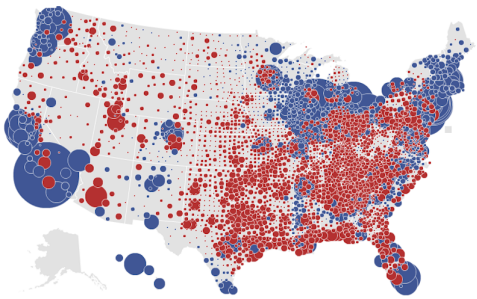

FIGURE 4.4 – Election présidentielle aux Etats-Unis (rouge=démocrates, bleu= républicains) [7]

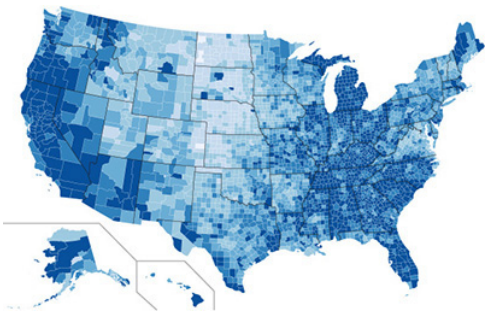

FIGURE 4.5 – Taux de chomâge aux Etats-Unis

Comme le soulignent [Lamarche-Perrin et al., 2012] et [Lamy et al., 1999], certaines techniques de visualisation reposent sur la capacité d'agréger les flux de données. Il est alors possible de représenter les données collectées au cours du temps dans une seule représentation. Cette technique permet de compresser l'information en une seule image, comme dans la figure 4.6 ou dans une animation. On parle alors de *carte de flux ou de réseaux*. Dans ce genre de carte, en général, les lignes suivent les parcours géographiques des voies de communication, la largeur des bandes est proportionnelle aux données quantitatives comme le nombre de passagers et les couleurs permettent de représenter des données quantitatives comme le type de transport. La réalisation de ces cartes nécessite parfois une déformation spatiale du fond de carte pour une meilleure lisibilité. La distorsion géographique est parfois utilisée dans les *cartogrammes* comme dans la figure 4.7 où les régions géographiques n'apparaissent non plus en fonction de leur surface géographique mais en fonction de leur population.

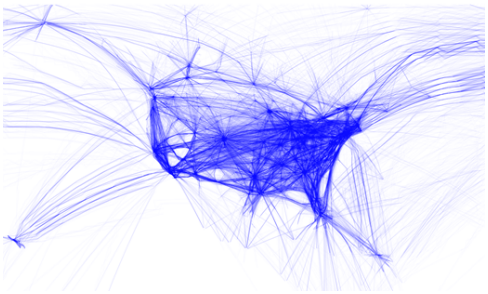

FIGURE 4.6 – Visualisation de données aériennes agrégées [Koblin, 2006].

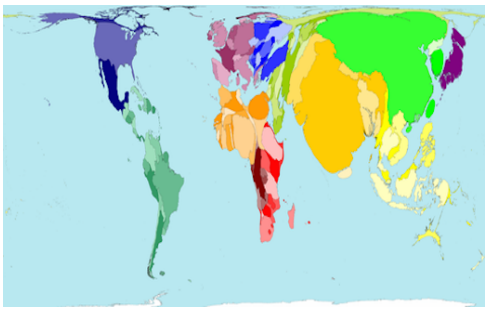

FIGURE 4.7 – Population mondiale sous formes de cartogramme [14].

Approche ABV

Ici notre approche ABV répond à tout les exigences de la représentation spatiales. Le langage présenté dans la partie 2.3.2 permet sans problème de réaliser des cartes à symbole proportionnelles en spécifiant la taille de l'agent en fonction de l'attribut à observer. Les cartes choroplèthe sont réalisables en créant un agent par régions et en lui assignant une couleur ou une transparence en fonction de l'attribut à observer. Pour les cartes de flux de réseaux comme nous l'avons montré dans la figure 3.2, il suffit de mettre à jour la trajectoire de l'agent et de l'afficher dans son aspect et d'utiliser le principes d'agrégation temporelle comme nous l'avons montré dans la figure 3.12. Enfin les opérateurs spatiaux permettent de déformer facilement la géométrie d'un agent pour obtenir des cartogrammes.

4.2.5 Structures Spatiotemporelles

Spécifications

Les données spatiotemporelles sont des données spatiales évoluant dans le temps. Souvent, la nature dynamique de ces données ne peut pas se contenter d'une représentation statique. Ainsi, la conception permettant des rendus dynamiques et interactifs est une discipline en plein essor. Les types de phénomènes spatiotemporels peuvent être existentiels (apparition, disparition, réapparition), spatiaux (changements affectant les propriétés spatiales des objets) ou thématiques (changements s'appliquant aux attributs thématiques des objets). Comme nous l'avons montré dans la partie 3.1.2, lorsqu'il s'agit d'objets en mouvement, la représentation de leur trajectoire, éventuellement ponctuées de repères temporels, est une bonne façon de mettre en évidence la temporalité de ces données. De la même façon que dans la partie 3.4, lorsque les deux dimensions représentent l'espace, l'utilisation de la troisième dimension permet d'introduire la dimension temps. On parle alors de cube spatiotemporel (space-time cube) [Kraak and Koussoulakou, 2005]. Les données spatiotemporelles thématiques sont difficiles à représenter sur un support statique [4] [5]. Les cartes multiples, qui consistent à afficher des séquences de cartes représentant un moment particulier, offrent une vision simultanée des changements. L'animation permet en revanche de déployer différentes séquences d'images. La dimension temporelle réside alors dans la durée réelle de l'animation. L'usage d'outils interactifs permet de moduler l'échelle du temps afin de l'accélérer ou de le ralentir. L'étude de données spatiotemporelles fait donc appel à une échelle temporelle et une échelle spatiale.

Approche ABV

L'approche ABV sera largement détaillée dans le chapitre suivant où nous illustrerons sur deux cas d'applications différents la représentation et l'analyse de données spatiotemporelles.

4. Les cartes multiples permettent à l'utilisateur de se concentrer sur les patterns spatiaux alors que les représentations interactives et animées favorisent plus la concentration sur les patterns temporaux [Andrienko et al., 2010].

5. Dans les cartogrammes de distances, ces deux échelles se confondent puisque les temps de déplacement sont exprimés en termes de distance [Carden, 2006].

Conclusion

Comme nous l'avons montré dans la première partie de cette thèse, la visualisation à base d'agents permet, par le biais des agents graphiques, d'extraire de la connaissance des données qu'ils manipulent et d'adapter leur aspect individuel ou collectif en fonction de celles-ci. Ainsi, nous avons souhaité valider dans cette partie la généralisation de l'usage de modèle de visualisation à base d'agents à la visualisation d'informations. S'appuyant sur un état de l'art de la visualisation d'informations, ce chapitre s'est donné pour tâche de faire émerger un certain nombre de besoins en termes de visualisation et de montrer que l'usage de modèle de visualisation à base d'agents permettait de répondre à la quasi totalité de ces besoins.

Le langage présenté dans la partie 2.3 permet de disposer d'une grande flexibilité graphique et d'exprimer de la façon la plus libre possible le type de représentation souhaitée. Cette flexibilité graphique permet de représenter des données en ayant toute la liberté de jouer sur les formes, les couleurs et les positions des données à visualiser. A travers l'utilisation d'ABV, nous pensons donc pouvoir répondre à plusieurs problèmes liés à la visualisation d'informations en proposant une approche flexible, modulaire et adaptable.

La variété des données imposent de fournir des outils permettant une approche modulaire et suffisamment générique pour être réutilisés dans d'autres domaines et sur d'autres applications. De plus en plus de logiciels sont disponibles et l'approche libre, modulaire et réutilisable a déjà largement fait ses preuves pour faciliter leur conception et maintenance. Ainsi, l'utilisation de modèle de visualisation permet d'envisager cette modularité en développant des agents suffisamment génériques pour être utilisés comme des modules de visualisation. Enfin, le fait que nous considérons les données comme provenant de systèmes complexes dynamiques évoluant parallèlement à différents niveaux impose une approche adaptable pour gérer de façon autonome des données hétérogènes et dynamiques n'utilisant pas forcément les mêmes formalismes.

Ces trois contraintes (flexibilité, modularité, adaptabilité) montrent bien que la visualisation d'informations peut se voir plus comme une construction dynamique décentralisée et itérative plutôt que comme une conception statique centralisée. Dans la lignée de travaux tel que [Hutzler and Renault, 2000], nous pensons donc que cette construction dynamique et itérative est grandement facilitée par l'usage d'objets ayant la faculté de s'auto-organiser, de s'adapter, d'évoluer en fonction de l'information traitée. Nous avons

vu qu'il était possible de doter ces agents de connaissance permettant à la fois de décrire la complexité de la structure de l'information à représenter et d'offrir des moyens de représentation auto-organisés. L'approche ABV permet donc de (1) doter ces agents de connaissances spécifiques permettant d'adapter leur représentation visuelle (leur aspect) en fonction de la structure de l'information et (2) de doter ces agents de propriétés leur permettant de s'adapter aux actions de l'utilisateur.

Après avoir donné une définition des notions clés de la visualisation d'informations, nous avons montré que, face à des données complexes et dynamiques, il n'y avait actuellement peu de méthodologies génériques provenant du fait que les outils actuels manquaient de flexibilité, modularité et adaptabilité. Nous avons proposé une classification des techniques de représentation de l'information en détaillant les différentes structures de l'informations. Nous avons montré comment l'utilisation de modèle de visualisation à base d'agents permettait de fournir une approche générique et capable de répondre à la quasi totalité des spécificités de chaque structure d'informations.

L'usage de techniques de visualisation avancées reste encore un domaine réservé à une communauté d'experts. Pour le grand public la tendance est encore à l'utilisation d'outils relativement basiques, comme les tableurs, inadaptés au traitement de données dynamiques et complexes. La complexité du processus analytique existant dans les systèmes de visualisation reste un obstacle lié au manque de méthodologies en visualisation d'informations. Ainsi, en se basant sur les travaux de [Fry, 2004], nous proposons dans le chapitre suivant une approche incrémentale permettant, à l'aide de modèles de visualisation, de représenter et analyser des jeux de données.

Chapitre 5

Représentation de données spatiotemporelles

Ce chapitre illustre les concepts montrés dans le chapitre 4 appliqués à deux jeux de données spatiotemporelles et vise à exploiter les possibilités de notre approche en termes de visualisation d'informations. Nous décrivons dans la première partie de ce chapitre une approche de visualisation dynamique de sédimentation d'une rivière que nous étudierons comme un ensemble d'agents représentant des secteurs de la rivière. Tout d'abord nous montrons le couplage de données SIG sur lequel un modèle à base d'agents permettra de modéliser la dépose sédimentaire. Puis nous proposons une représentation spatiale d'un modèle numérique morphosédimentaire permettant de visualiser et d'interpréter les sorties de ce modèle. Dans la deuxième partie de ce chapitre, nous disposons d'un jeu de données sur la maladie de la dengue en Asie du Sud-Est. La propagation spatiotemporelle de la maladie sera représentée dans un premier temps puis nous exploiterons la capacité des agents à analyser les données pour détecter les épidémies dans le jeu de données.

5.1 ARCHEM (Action de Recherche Collaborative sur les Hydrosystèmes et les Environnements en Mutation)

Introduction

La modélisation des systèmes complexes comme les interactions hommes-milieux nécessite la mobilisation de concepts et de méthodes permettant d'étudier un territoire à différentes échelles spatiotemporelles. Parmi ces techniques, la modélisation à base d'agents permet d'étudier un système en modélisant les entités qui le composent sous la forme d'agents dont les interactions permettent l'émergence de dynamiques globales. Le projet Archem, financé par le laboratoire d'excellence DRIIHM / IRDHEI (Dispositif de recherche interdisciplinaire sur les Interactions Hommes-Milieux), appartient à ce nouveau domaine de recherche. Ce projet propose une nouvelle méthodologie pour visualiser le transport sédimentaire fluviale à grande échelle (1/5000e) en mettant l'accent sur la visualisation dynamique et 3D du phénomène.

Sur le Rhône, les aménagements anciens, liés à la navigation, ont profondément modifié la géométrie du chenal au cours du XXème siècle. De récentes recherches menées dans par l'OSR (Observatoire des Sédiments du Rhône) ont montré l'intérêt du démentellement de ces anciens aménagements dans les Vieux-Rhône pour une restauration du lit de la rivière. Ces travaux consistent à effectuer des opérations de recharge sédimentaire. La prise de décision autour de ce type d'opération est délicate, tant pour la CNR (Compagnie Nationale du Rhône) que pour les services d'Etat car les gains écologiques escomptés étaient jusqu'à présent complexes à évaluer et à représenter.

L'approche théorique du fonctionnement hydro-sédimentaire permet aujourd'hui de modéliser finement le transport de sédiments à l'échelle d'un tronçon [Lauer and Parker, 2008]. En revanche la communication autour des résultats issus de ces modélisations reste à être améliorée pour être utilisée pour l'aide à la décision. Ainsi la visualisation joue un rôle important dans la capacité d'appréhension des dynamiques présentes dans le processus observé. Nous représentons les dynamiques propres aux données puis tirons profit de l'approche en ajoutant dynamiquement certains indicateurs afin d'augmenter les données disponibles.

5.1.1 Représentation spatiale

Ce premier modèle ABV est une mise en œuvre de la partie 4.2.4. Il consiste à obtenir une représentation spatiale d'un phénomène. Cette représentation est obtenue en agentifiant les différents données SIG et en les représentant dans un espace 3D comme le montre la figure 5.1. Les trois types de données correspondent à trois espèces d'agents distinctes. Ce modèle représente la section de la rivière étudiée (ici le Rhône), le type de terrain aux abords de la rivière et enfin les points kilométriques disposés tous les 500 mètres.

L'espèce **rivière** possède un aspect consistant à prendre la représentation définie dans le shapefile. Cette espèce est instanciée à partir d'un fichier *shapefile*. Le type de terrain est une donnée qualitative représentée à l'aide de l'attribut couleur afin de catégoriser visuellement les différentes zones représenté par l'espèce **terrain**. L'espèce **point** représenté les point kilométriques par un cercle rouge .

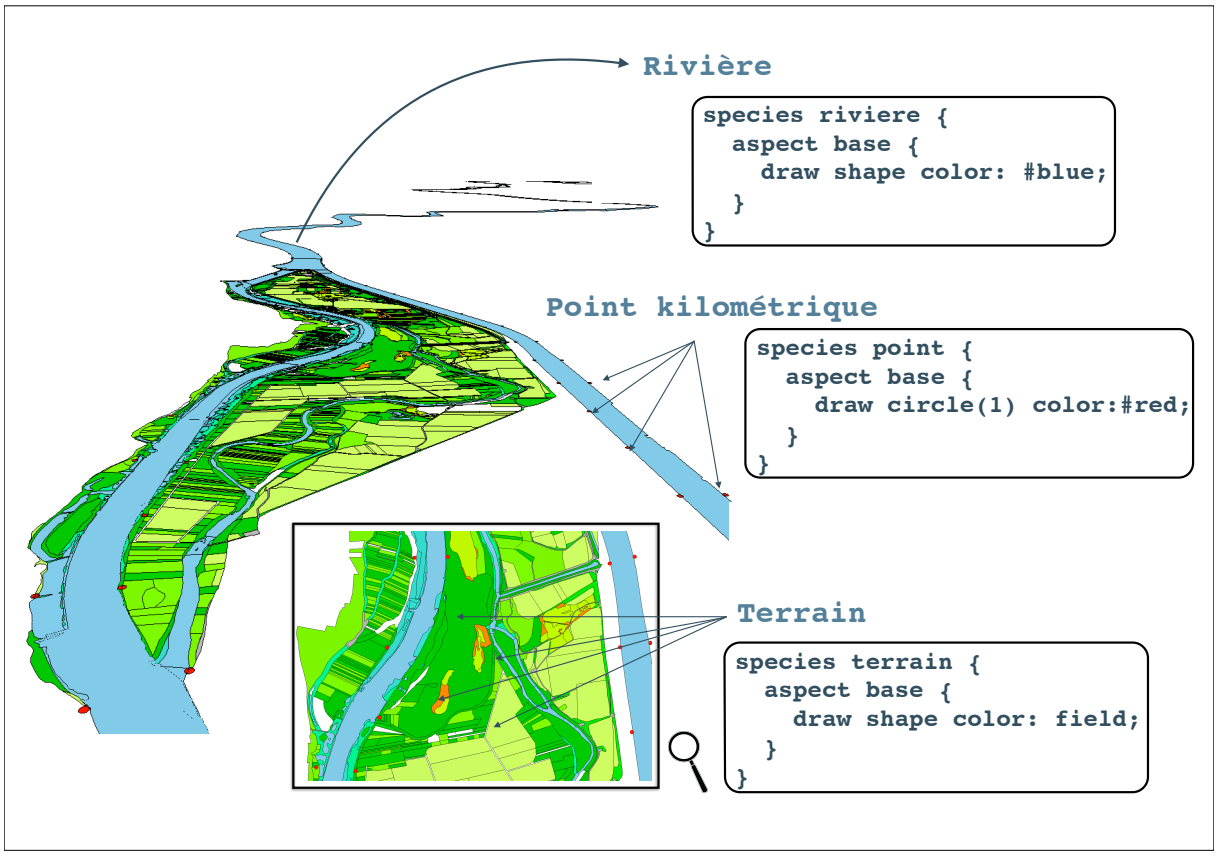

FIGURE 5.1 – Représentation 3D de données SIG.

Reconstitution de données : Calcul du talweg

Les agents graphiques du modèle ABV proposé ici sont exploitées pour, à partir des profils transversaux de la rivière, recréer le talweg (ligne qui rejoint les points les plus bas d'une vallée). Le talweg devient alors un nouvel agent créé au cours de la simulation. La figure 5.2 représente une section de la rivière sur laquelle sont représentés les profils transversaux et le talweg. Dans ce modèle ABV, l'aspect de l'espèce **profil** consiste à dessiner une liste de points qui représente le profil en travers de la rivière. Pour chaque section de la rivière correspond un profil en travers. L'aspect de l'espèce **talweg** consiste dans un premier temps à extraire les valeurs d'altitude minimale de chaque section. La commande `ask` permet d'interroger chaque section et d'identifier l'altitude minimale (Zmin) de chaque section. Le point Zmin est ajouté à la variable de type *list* **channelSlopePoints**. Une fois cette liste de point créée il suffit de dessiner une ligne à partir de cette dernière à l'aide de la fonction `line`.

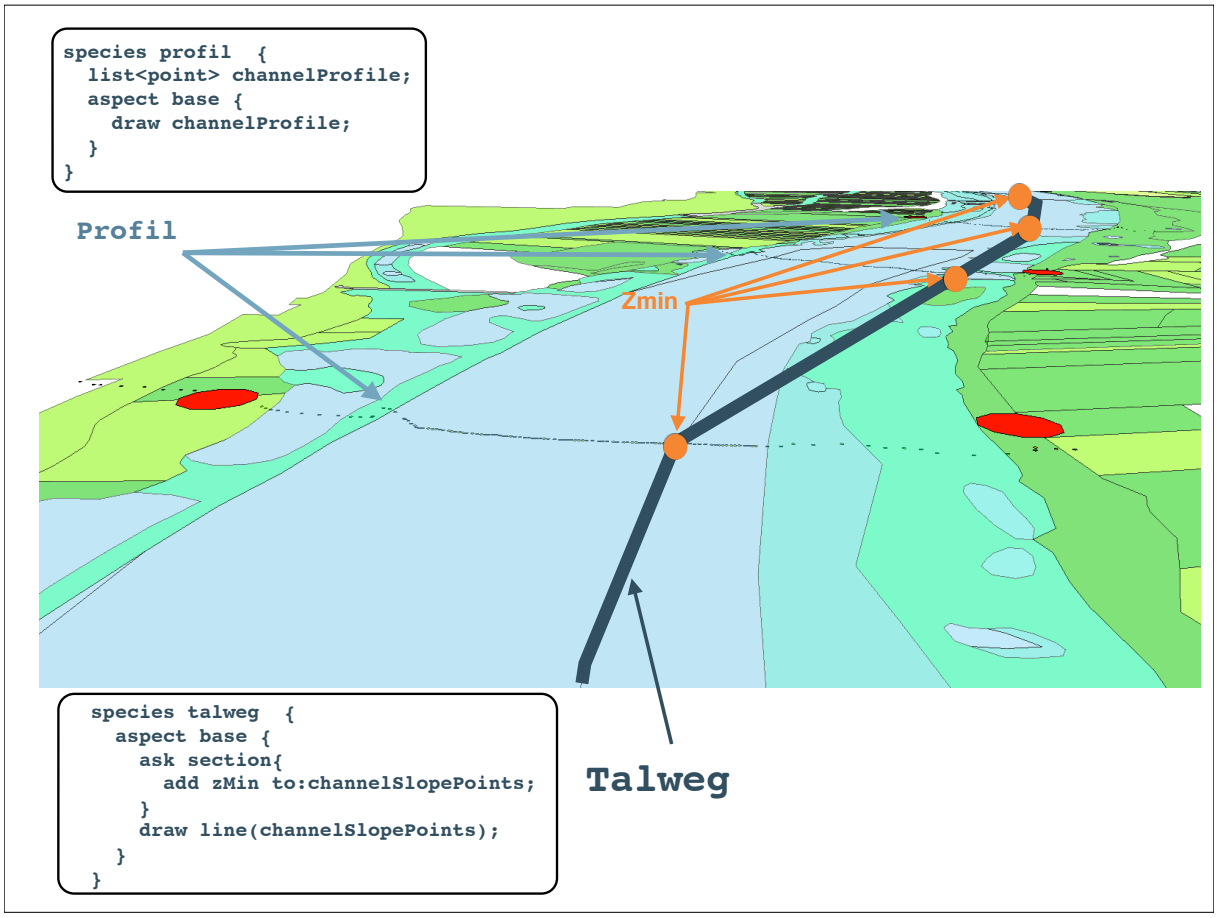

FIGURE 5.2 – Représentation des profils transversaux et reconstitution du talweg.

5.1.2 Représentation temporelle

Ce modèle illustre le caractère temporelle décrit dans la partie 4.2.3 des données en proposant une représentation temporelle animée du processus de sédimentation. Les données pluviométriques sont couplées au modèle précédent afin de visualiser leurs effets sur le cours d'eau.

Visualisation dynamique du processus de sédimentation

Tout au long de la rivière sont disposés des casiers construits pour la navigation. Ces casiers contiennent un grand nombre de sédiments dont la hauteur varie selon les données pluviométriques. Dans le modèle ABV les casiers sont représentés par leur forme SIG à laquelle on ajoute une hauteur représentant la hauteur des sédiments grâce à la facet `depth`.

L'espèce **water** représente les flux d'eau et permet de modéliser l'influence du flux d'eau sur les casiers. L'espèce *water* possède un attribut **flewValue** caractérisant le valeur de flux. Selon cette valeur un agent water fera plus ou moins augmenter la hauteur des casiers situés dans son voisinage. Ainsi le comportement de l'agent **water** est composé de deux `reflex`. Dans le réflexe **move**, l'agent se déplace le long de la rivière grâce à la commande `follow` et au paramètre `path` contenant la liste des points de la rivière. Le réflexe **updateCasiers** consiste à mettre à jour la valeur de l'attribut **sediment** des casiers situés dans le voisinage (représenté en rouge sur la figure 5.3) de l'agent **water**. Chaque agent *water* se déplace sur le tracé de la rivière à l'aide du **skills** *moving* lui permettant de suivre un chemin (*path*), en l'occurrence ici le tracé de la rivière.

Le lancement de la simulation de ce modèle ABV, permet d'obtenir une représentation animée du processus de sédimentation.

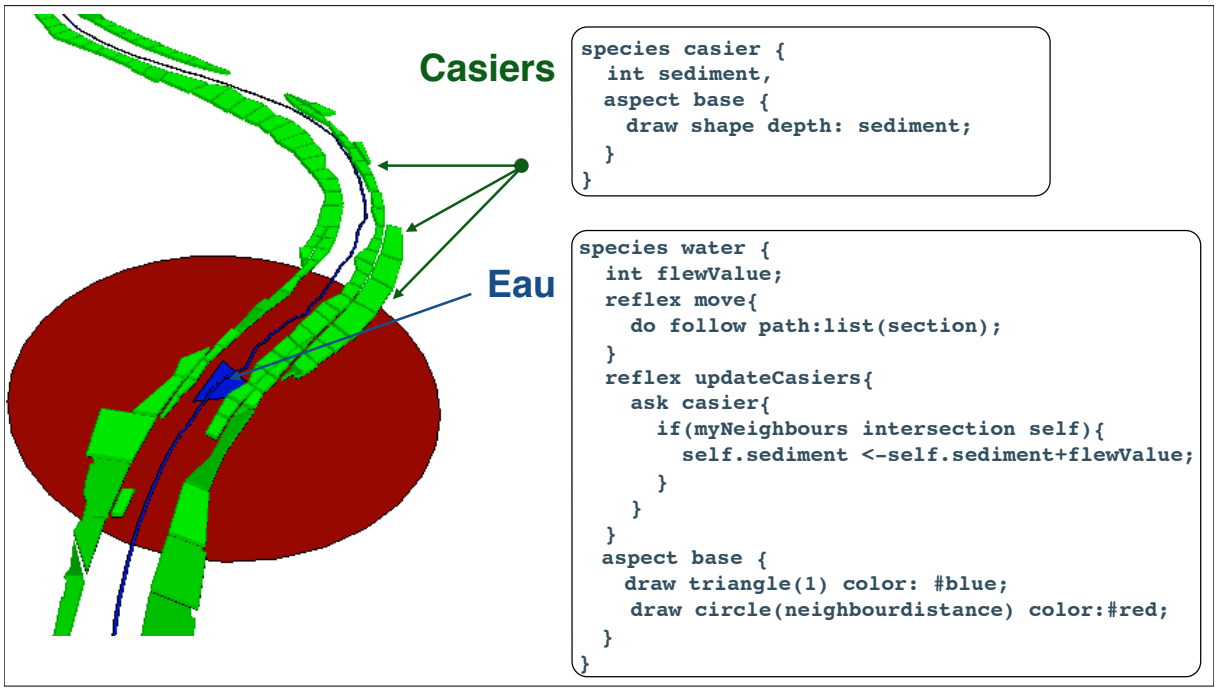

FIGURE 5.3 – Modélisation du flux d'eau. L'agent "**water** se déplace le long de la rivière et met à jour la hauteur des casiers situés dans son voisinage afin de fournir une représentation animée du processus de sédimentation.

Couplage avec des données réelles

Nous proposons dans ce modèle de visualiser un jeu de données sur les débits relevés dans le fleuve. Ces données sont stockées dans un fichier *.csv* où chaque ligne du fichier correspond à la mesure du débit relevé quotidiennement, en m^3/s. A chaque itération, selon la valeur du flux, un agent *water* est créé et la valeur de son attribut **flewValue** est initialisée avec la valeur lue dans le fichier. Par exemple, la première valeur contenue dans le fichier est de 2080 m^3/s, ceci correspond donc à la création d'un agent *water* dont la valeur de l'attribut **flewValue** est de 2080. Le modèle de visualisation proposé reprend le modèle précédent. En revanche deux types de flux sont distingués, si la valeur du flux est inférieure à un certains seuil (**threshold**), la couleur de l'agent sera verte, si la valeur est supérieure à ce seuil la couleur est rouge. La valeur de **flewValue** est aussi affichée textuellement. La simulation du modèle permet de visualiser la dynamique de la rivière sur une année. Les 10 premières itérations de la simulation sont montrées dans la figure 5.4.

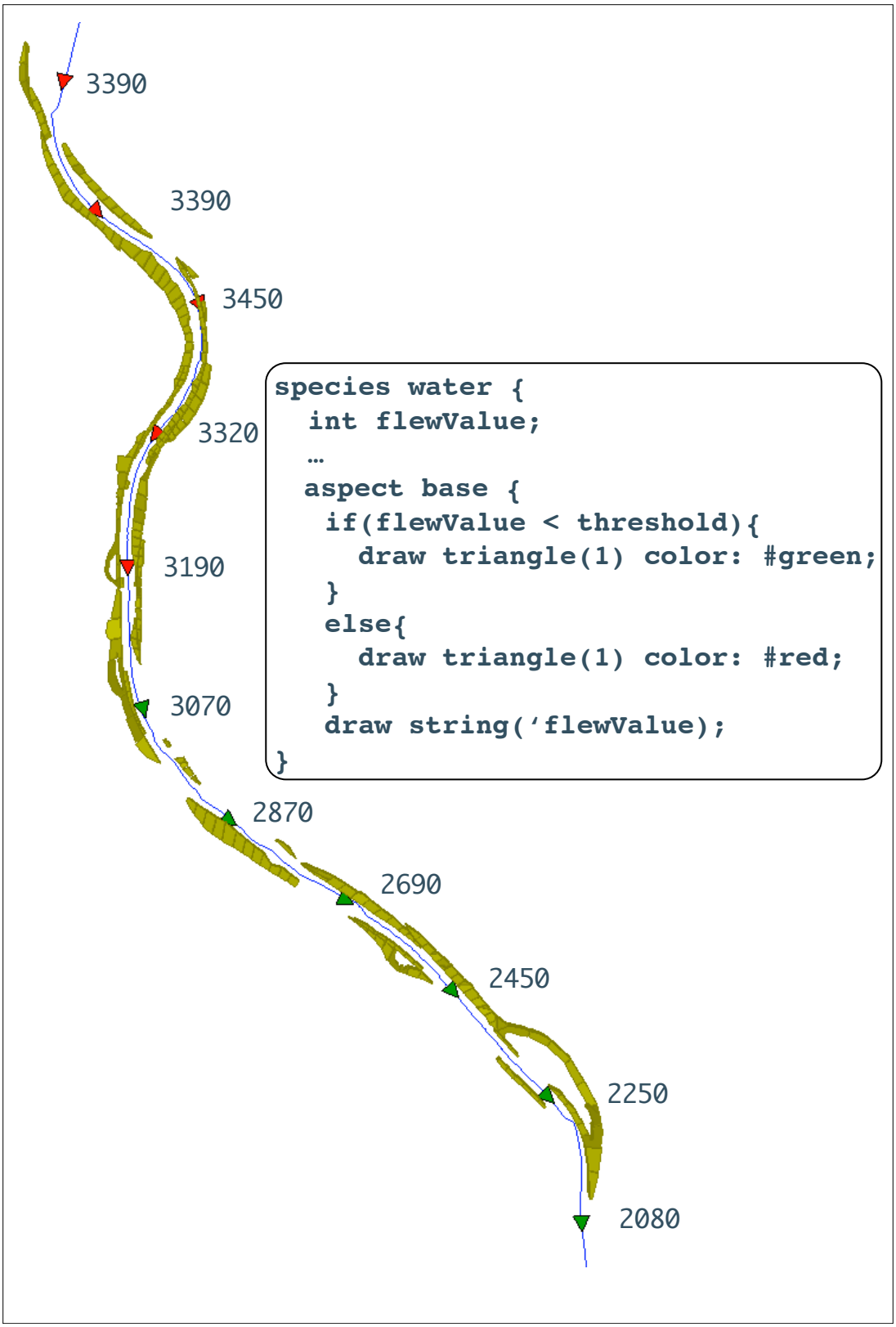

FIGURE 5.4 – Création d'agents *water* à partir de données pluviométriques

5.1.3 Couplage avec un modèle hydro-sédimentaire

Dans le modèle précédent, le flux d'eau était représenté par des agents ayant une influence directe sur la rivière. Cette approche reste cependant une méthode approximative et vise plus à représenter schématiquement le phénomène qu'à réellement le modéliser. Les modèles hydro-sédimentaires comme MAST-1D [Lauer et al., 2014] permettent aujourd'hui de modéliser finement le transport de sédiments à l'échelle d'une rivière. Cependant, ces modèles numériques permettent une analyse fine du comportement de la rivière mais ils manquent d'intelligibilité car ils ne sont pas spatialisés. Les résultats sont stockés dans des tableaux difficilement interprétables. Nous proposons donc une représentation spatialisée sur un SIG correspondant à celui de la rivière étudiée, le Rhône.

Dans un premier temps, nous nous intéressons à une représentation spatiale générique du modèle numérique MAST-1D. Ce modèle initialement non-spatialisé fournit les valeurs des différents paramètres tous les 500 mètres. L'évolution du cours de la rivière est alors représentée dynamiquement en lisant les valeurs des paramètres et en mettant à jour l'aspect de chaque agent correspondant à une section de la rivière et ce, à chaque itération. Comme le montre la figure 5.5, dans ce modèle seront représentés, pour chaque section, le chenal (*Chanel*), l'épaisseur de la bande active (*Active Layer*), l'épaisseur de la plaine alluviale (*Food Plain*) et enfin l'épaisseur de la plaine alluviale distante (*Distal Food Plain*).

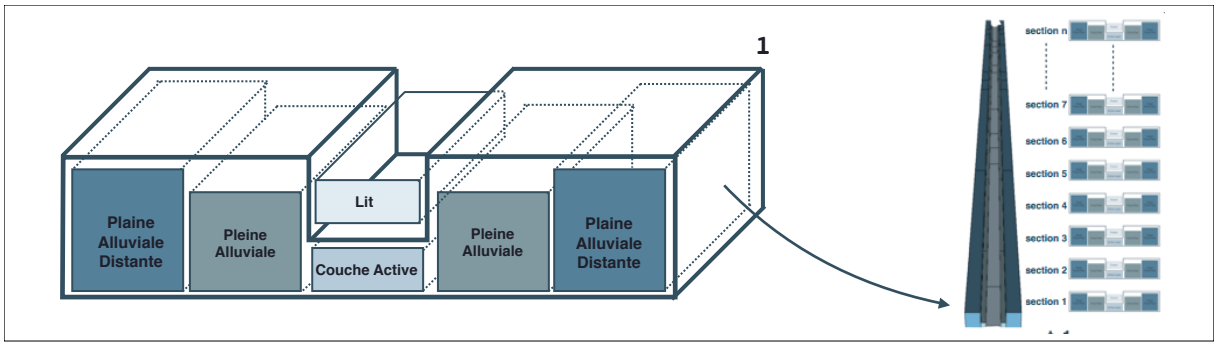

FIGURE 5.5 – Représentation générique statique. Chaque agent possède un aspect complexe représentant le lit de la rivière, la couche active, et la plaine alluviale et la plaine alluviale distante.

Résultats

Nous montrons ici la spatialisation des résultats du modèle MAST-1D. Chaque agent représentant une section possède une position et il est alors possible d'appliquer une translation à une position bien précise pour appliquer le modèle à n'importe quelle rivière. La figure 5.6 montre les résultats d'une simulation du modèle MAST-1D sur la portion de rivière étudiée.

Le travail réalisé peut désormais servir à la visualisation de plusieurs scénarios pour des opérations de restauration. Ces simulations visuelles peuvent servir à présenter, aux gestionnaires mais aussi au grand public, de manière explicite les conséquences de ces opérations en matière de transformation de la géométrie du chenal et de changement de granulométrie.

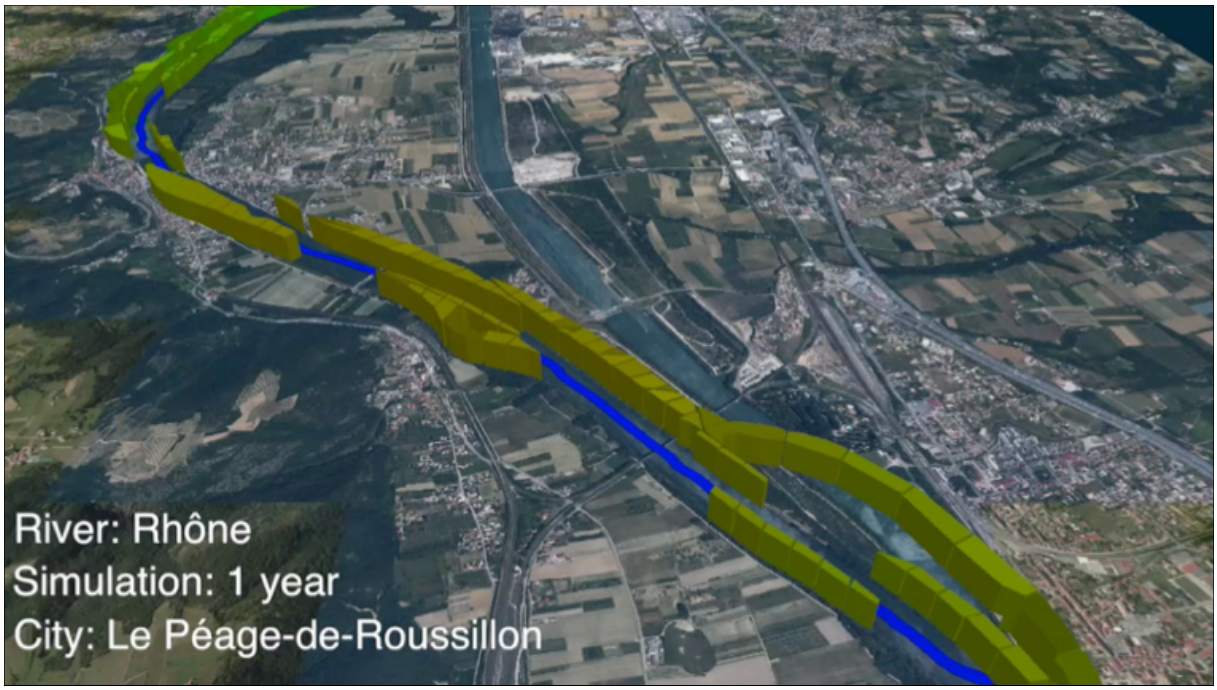

FIGURE 5.6 – Représentation spatialisée du modèle MAST-1D `http://youtu.be/HWctj1ni5Qk`

5.1.4 Conclusion

Les premiers résultats de cette approche sont très encourageants. Ce couplage a aidé à construire une représentation spatiotemporelle de la morphodynamique d'une rivière dans un environnement 3D immersif. La représentation dynamique du processus de sédimentation a été mise en œuvre avec des valeurs simulées et des données historiques. Ainsi, la spatialisation d'un modèle hydro-sédimentaire numérique a été accomplie. A notre connaissance, une telle visualisation de la morphodynamique d'une rivière résultant du couplage de données SIG et d'un modèle mathématique représente une première. Les logiciels de SIG actuels (ArcGIS [3], QGIS [11]), permettent de visualiser un grand nombre de données SIG et de faire de l'analyse spatiale mais il est surprenant de constater les deux points suivants : (1) Aucun d'eux ne gère la 3D de façon native[1]. (2) Aucun d'eux ne permet de donner du comportement à des objets SIG. Notre approche, grâce à l'agentification des ces derniers, permet ainsi d'obtenir une visualisation 3D de la dynamique de la rivière et de lui donner du comportement.

Les travaux futurs sur ce projet consisteront à (1) analyser l'ajustement morphologique du Rhône après le retrait des casiers *Girardon* et (2) évaluer les conséquences de la construction d'un canal de dérivation sur les inondations. Ces simulations seront utilisées pour tester le comportement du modèle MAST-1D dans les configurations de réglages connus et appliquées de manière prospective pour analyser la réponse morphologique du canal en fonction de différents scénarios de recharge sédimentaire artificielle.

1. Une solution consiste à utiliser une extension ArcGis 3D Analyst [2] ou QGIS Planet [12]

5.2 DENSEAT (Dengue South East Asia Timing)

Introduction

La dengue, première maladie virale transmise par des moustiques, a commencé à émerger dans les années 50 en Thaïlande, puis en Asie du sud-est et progresse depuis dans le reste du monde tropical. Les chercheurs en épidémiologie s'intéressent à la dynamique spatiale des maladies infectieuses [Grenfell et al., 2001] [Cummings et al., 2004]. Ces études ont été rendues possibles par la disponibilité de données de surveillance épidémiologique comprenant une information temporelle et spatiale. En termes de visualisation, les résultats actuels sont peu nombreux et essentiellement présentés à l'aide d'outils d'analyses statistiques comme des diagrammes, séries temporelles ou heatmap [2]. Ces outils ont leurs avantages sur un point de vue analytique mais reflètent difficilement le caractère spatiotemporel du jeu de données. Nous illustrons dans cette partie l'intérêt de notre approche pour représenter et analyser la propagation spatiotemporelle de la maladie, puis nous proposons une méthode tirant profit de l'utilisation de modèles de visualisation à base d'agents afin de faire émerger, de façon distribuée, les différentes épidémies présentes dans le jeu de données. Cette approche est utilisée dans le projet *"Modeling of emerging infectious diseases in Vietnam" (2013-2014), Vaccine Modeling Initiative, Bill & Melinda Gates foundation, PI : Dr. Pham Quang Thai"* en collaboration avec l'unité MIVEGEC et l'UMMISCO.

5.2.1 Représentation spatiotemporelle

Le jeu de données regroupe des données sur 8 pays de l'Asie du Sud Est (*Cambodge*, *Timor Oriental*, *Indonésie*, *Laos*, *Malaisie*, *Philippines*, *Taïwan*, *Thaïlande* et *Vietnam*). Chaque pays est découpé à l'échelle provinciale, ce qui représente près de 500 régions distinctes. La fréquence d'échantillonnage est de un mois sur une période de plus de 40 ans (de 1968 à 2010). Les données sont stockées dans le format décrit dans le tableau 5.1 et représentent près de 100 000 entrées [3].

2. On peut tout de même citer les travaux récents de [Brockmann and Helbing, 2013] proposant une visualisation spatiotemporelle de la propagation de la maladie.

3. Le manque de données, notamment pour les périodes avant 1990, fait chuter de 240 000 entrées théoriques à 100 000 entrées en pratique.

Nom	**Description**	**Exemple**
province	nom de la province	Hanoi
cases	nombre de nouveaux cas observés	233
year	année à laquelle a été prise la mesure	2005
month	mois auquel a été prise la mesure	10
time	temps représenté de manière continue	2005.8
country	nom du pays	Vietnam
longitude, latitude	position de la province	-2225, 3916

TABLE 5.1 – Données DENSEAT

Modèle de visualisation spatiale

Dans ce modèle, un seul type d'espèce (que nous appellerons *data*) est utilisé pour représenter les données. Ici, les agents graphiques sont simplement utilisés pour la représentation. Les comportements permettant de remettre à jour les aspects des agents sont gérés au niveau de l'environnement (le monde). Cet exemple illustre l'utilisation d'attributs graphiques combinés à des aspects pour représenter les données de façon statique et dynamique et ce, dans un environnement 2D ou 3D.

Comme le montre la figure 5.7, chaque ligne de l'ensemble de données est agentifiée et chaque colonne est utilisée pour initialiser les attributs de l'agent. Le type *string* représente un ensemble de caractères utilisé pour définir le nom de la province et du pays correspondant. Le type *int* représente un nombre entier utilisé ici pour le nombre de cas, le mois et l'année correspondant. Le type *float* est un nombre avec une ou plusieurs décimales utilisé pour le temps absolu ainsi que pour la localisation (longitude, latitude).

Affichage 2D

Chaque agent *data*, représente une province et possède un attribut *latitude* et *longitude*. Il peut donc être affiché dans un environnement dont la taille a été déterminée en extrayant les bornes minimales et maximales du domaine d'étude, à savoir, la *longitude max*, *longitude min*, *latitude max* et *latitude min*. L'*aspect* de l'espèce *data* est un cercle noir de taille fixe à la position définie par les attributs *latitude* et *longitude*. La longitude

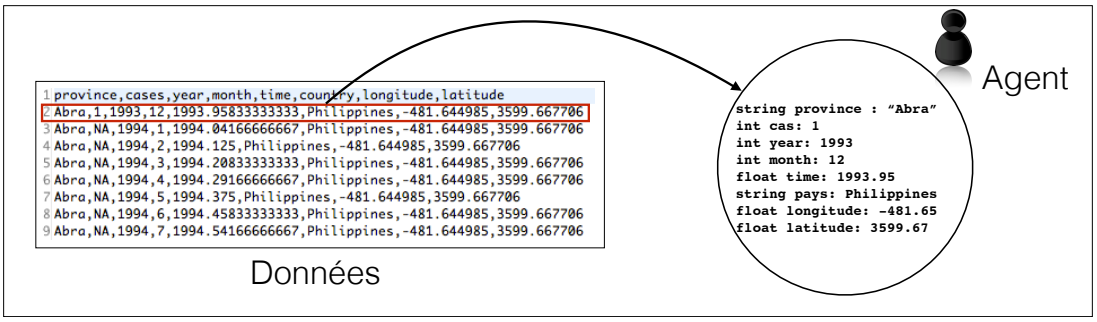

FIGURE 5.7 – Chaque donnée est instanciée dans un agent *data*. Les attributs de chaque agent permettent de stocker les informations dans le format décrit dans le tableau 5.1.

correspond à l'axe des abscisses x et la latitude à l'axe des ordonnées y. Une fois toutes les données agentifiées, elles sont représentées dans un affichage (*display*). Nous obtenons alors la représentation de la figure 5.8 où chaque agent *data* est représenté par un cercle noir à la position correspondante.

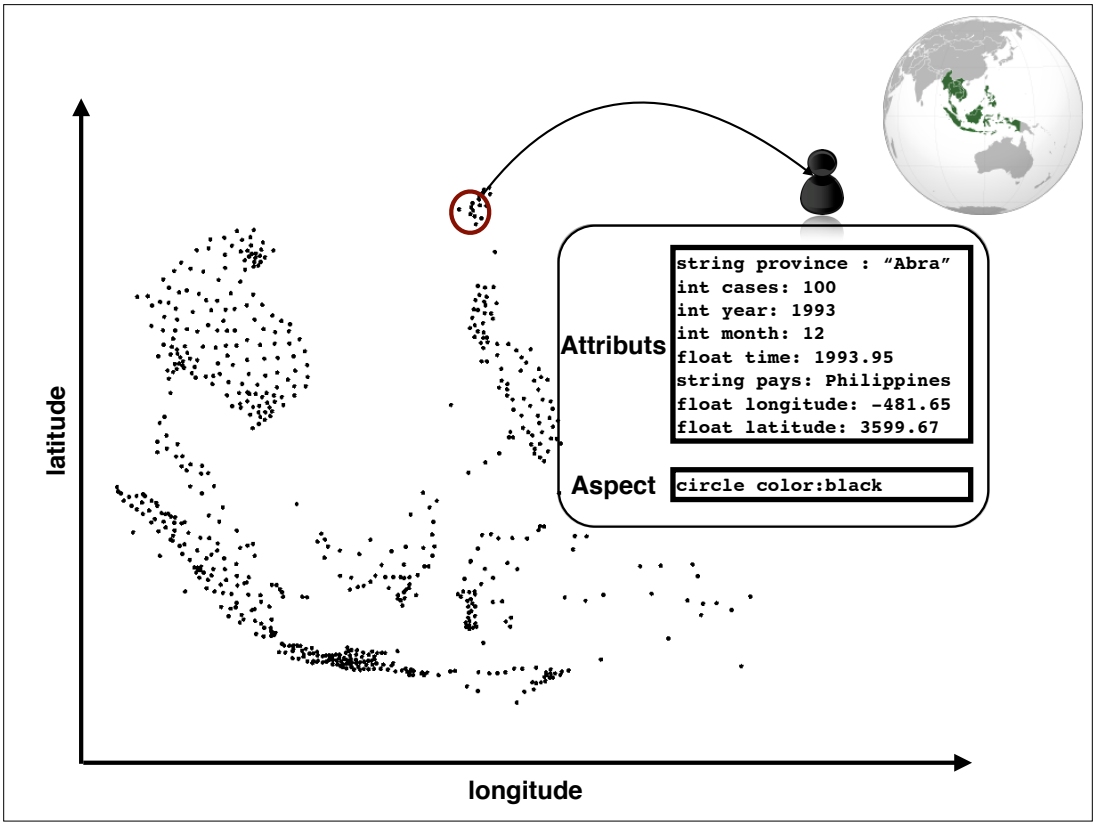

FIGURE 5.8 – Représentation 2D dans laquelle chaque province agentifiée en agent ABV est représenté par un cercle noir.

Affichage 3D

De la même façon que nous avons extrait les valeurs *longitude max*, *longitude min*, *latitude max*, *latitude min* pour délimiter l'espace 2D de notre représentation, nous extrayons maintenant les valeurs temporelles *year_max*, *year_min* afin de borner l'espace 3D. Nous ajoutons à la position 2D, une composante en z représentant la composante temporelle de la donnée en utilisant l'attribut *time*. Les données sont donc affichées dans un système de coordonnées à 3 dimensions. La latitude et la longitude sont représentées sur les axes x,y et le temps correspondant à la date d'acquisition de la donnée est représenté sur l'axe z. Le nombre de cas, pour chaque province, stocké dans la colonne *cases* est mis en évidence grâce à l'utilisation de la couleur. Nous extrayons les valeurs maximales du nombre de cas (*max_cases* et *min_cases*) afin de borner l'espace colorimétrique. Nous utilisons un gradient de couleur allant du jaune au rouge. Dans la figure 5.9, les faibles valeurs de cas sont représentées en jaune et les fortes valeurs en rouge.

Modèle de visualisation spatiotemporel

Dans cette partie, l'utilisation de modèles de visualisation permet de mettre en valeur le caractère temporel des données. Cette dimension temporelle est rendue possible grâce à l'utilisation de la simulation. Une fois le modèle initialisé, la simulation met à jour à chaque itération les attributs et les aspects des agents *data* pour obtenir un affichage dynamique des données. Ici, les agents *data* sont essentiellement utilisés pour représenter les données et ne possèdent pas de méthode graphique. Comme il s'agit ici de résultats animés, nous

Le modèle ABV correspondant à l'animation suivante `http://youtu.be/aVG0vZOKkQc` consiste à n'afficher que les agents dont l'attribut *time* est égal à l'itération courante de la simulation. Ce modèle de visualisation permet, à la manière d'un film, de dérouler le scénario passé. Enfin le modèle ABV de l'animation suivante `http://youtu.be/quVT7jNjhKE` utilise la troisième dimension pour représenter le temps et chaque donnée est affichée à sa position spatiale et à une hauteur correspondant au temps.

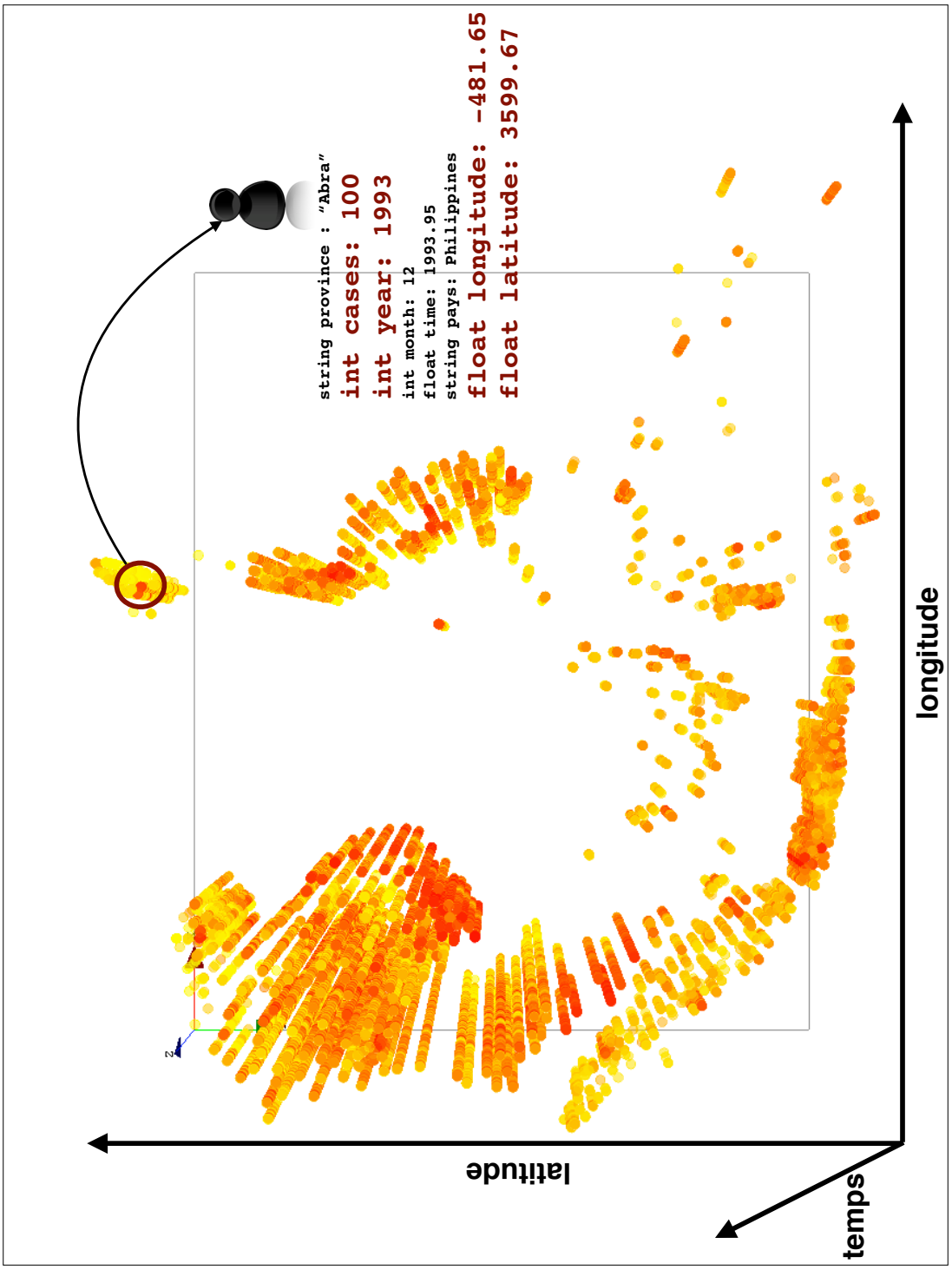

FIGURE 5.9 – Représentation 3D dans laquelle chaque province agentifiée en agent ABV est représenté par un cercle dont la couleur représente le nombre de cas. La troisième dimension est utilisée pour représenter le temps.

5.2.2 Détection d'épidémie

Cette partie illustre l'utilisation de méthodes graphiques pour analyser le jeu de données et adapter la représentation en conséquence. Nous appliquons un comportement à chaque donnée agentifiée afin d'identifier les périodes d'épidémie. Chaque donnée est représentée par un agent dont l'altitude représente la date à laquelle a été produite la donnée et où la couleur représente le nombre de cas. Cette représentation initiale est utilisée afin de détecter les épidémies au sein du jeu de données. Ici, le modèle de visualisation devient un modèle où la spatialisation des données est utilisée à des fins analytiques. Dans cette partie, nous définissons une méthode graphique, attaché à l'espèce *data*, permettant de détecter de façon décentralisée et de proche en proche les épidémies.

Un agent province est considéré viral lorsque le nombre de cas dépasse un certain seuil noté S. Lorsqu'un agent est viral, la valeur de l'attribut *viral* est vraie sinon sa valeur est fausse. Un agent *data* possède aussi les attributs *startTime* et *endTime* qui seront utilisés pour mettre à jour l'aspect de l'agent. Comme le montre la figure 5.10, un agent non viral est représenté en gris, un agent viral est représenté par une couleur correspondante au nombre de cas. L'aspect de l'agent est défini par un cylindre de rayon 1 et de hauteur égale à la différence entre l'attribut *endTime* et l'attribut *startTime*.

La méthode graphique, définie dans la figure 5.10, consiste à comparer les valeurs de cas de proche en proche afin de regrouper les agents similaires pour détecter et agréger les épidémies. A chaque itération, un agent interroge l'agent se situant au dessus de lui. L'agent prendra alors une décision en fonction de la valeur de son attribut *viral* et en fonction de la valeur de l'attribut *viral* de l'agent se situant au dessus de lui. Si la valeur de l'attribut *viral* de l'agent courant est fausse, alors l'agent fusionne avec l'agent du dessus et met à jour la valeur de son attribut *viral* avec la valeur de l'attribut *viral* de l'agent situé au dessus. Si la valeur de l'attribut *viral* de l'agent courant est vraie, alors l'agent fusionne avec l'agent du dessus si ce dernier est viral et met à jour la valeur de son attribut *viral* à vraie. Enfin, lors de la fusion, les valeurs des attributs *startTime* et *endTime* sont aussi mises à jour. Le nombre d'agents et la valeur de leur attribut évoluent au cours de la simulation. Ce comportement, couplé à l'utilisation d'un aspect dynamique, permet de détecter les épidémies en quelques itérations comme le montre la figure 5.11. A la fin du processus, les épidémies sont clairement représentées par des cylindres dont la hauteur correspond à la durée de l'épidémie.

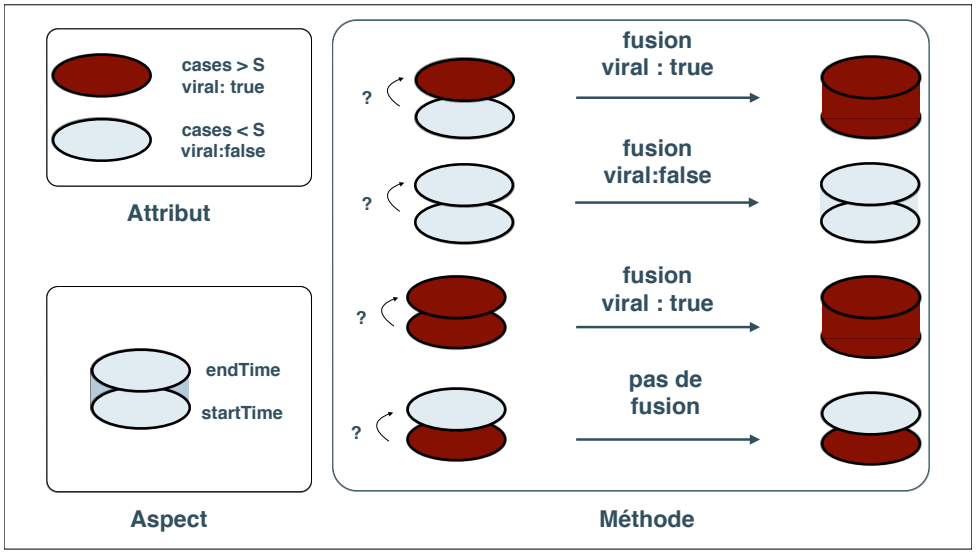

FIGURE 5.10 – Attribut, méthode et aspect de l'agent *data*

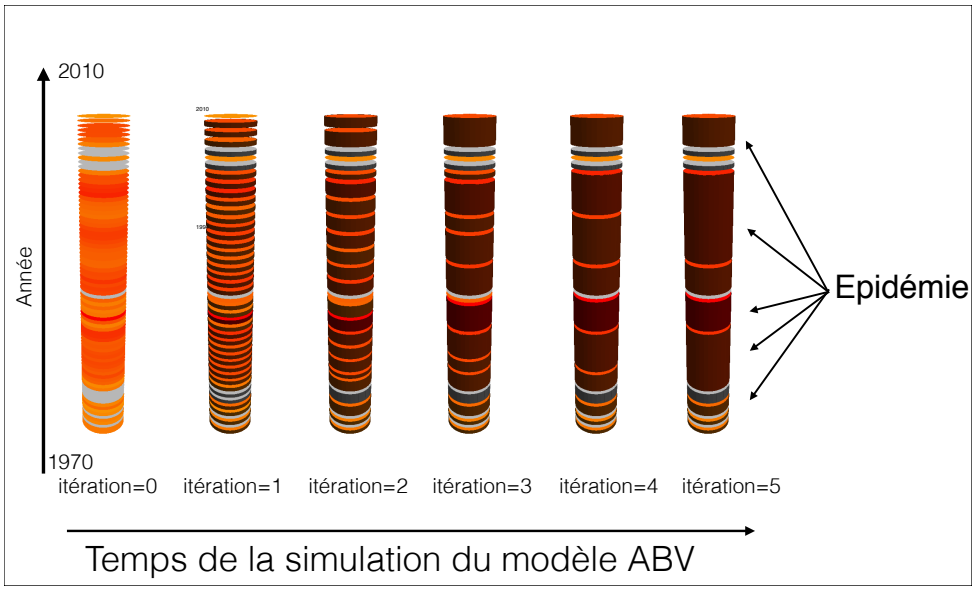

FIGURE 5.11 – Processus de détection d'épidémies pour une province. La simulation du modèle ABV se stabilise dans cet exemple au bout de 5 itérations et permet de détecter les épidémies.

FIGURE 5.12 – Processus d'identification des épidémies appliquées aux jeu de données entier. Cette représentation permet de voir instantanément les différentes épidémies de dingue ayant eu lieu au cours des 40 dernières années. Les épidémies sont représentées par un cylindre rouge dont la hauteur dépend de la durée de l'épidémie. http://youtu.be/Q6EJusctdPY

Applications

Ce modèle de visualisation étant appliqué à chacune des 500 provinces de notre jeu de données, nous obtenons une manière originale de détecter des épidémies dans ce jeu de données. Une représentation animée du processus de détection des épidémies est montrée dans la figure 5.12. Une fois les épidémies détectée il est alors possible de repositionner ces nouveaux agents et de modifier leur aspect en fonction de certains critères. Par exemple en repositionnant les épidémies non plus à leur position géographique mais à une position proportionnelle à la longueur de l'épidémie, nous obtenons un histogramme permettant de rapidement identifier les périodes et régions où les épidémies ont été les plus fortes et les plus longues comme le montrent les figures 5.13 et 5.14.

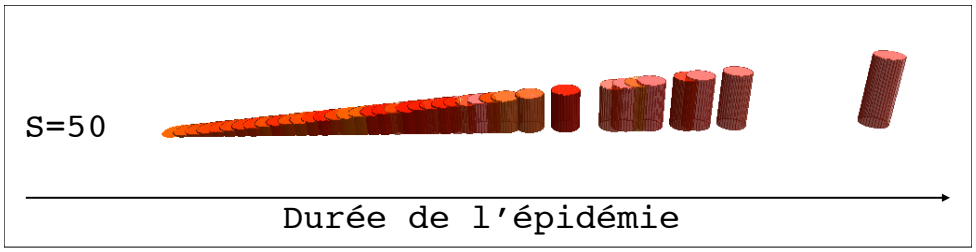

FIGURE 5.13 – Représentation des épidémies en fonction de la durée de l'épidémie.

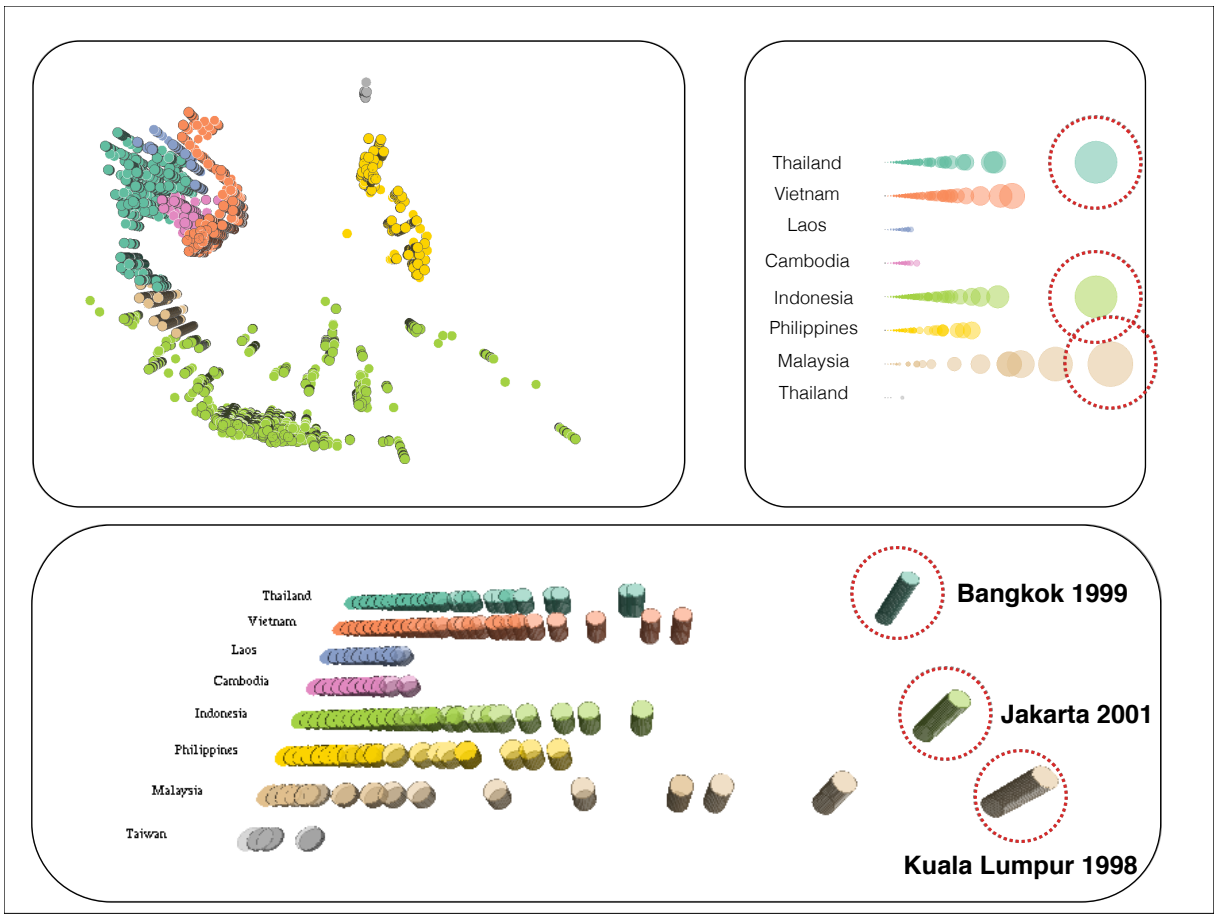

FIGURE 5.14 – Repositionnement regroupé par pays des épidémies en fonction de la durée de l'épidémie. Cette représentation permet d'immédiatement isoler les fortes épidémies selon les pays. Les trois plus grandes épidémies détectées sont entourées par un cercle rouge.

Conclusion

Dans le domaine de l'aide à la décision, les décideurs (acteurs politiques, financiers, etc) manquent d'outils simples pour vérifier et analyser rapidement les informations pour prendre la décision la plus adaptée. Des outils visuels, comme ceux proposés grâce à l'utilisation des modèles ABV, donnent la possibilité de présenter de façon analytique leurs résultats, de trouver des pertinences parmi leurs données et de communiquer certains concepts et hypothèses. C'est dans cette perspective que le projet ARCHEM présenté dans la partie 5.1 se situe.

La partie 5.2.1 met l'accent sur la flexibilité de l'approche pour illustrer la partie liée à la représentation de données spatiotemporelles acquises ou générées. Comme nous l'avons vu, ces représentations peuvent être statiques mais il est aussi possible de faire apparaître le caractère temporel de ce jeu de données grâce à une représentation dynamique. La partie 5.2.2 est une illustration de l'adaptabilité des agents pouvant s'auto-organiser afin de détecter dans notre cas des épidémies par une approche individu-centrée. Dans cette partie nous avons voulu mettre l'accent sur l'intérêt d'avoir un modèle de visualisation pour analyser un jeu de données.

Chapitre 6

Conclusions et Perspectives

Conclusion

Le travail présenté dans le cadre de ce mémoire de thèse représente la synthèse de trois années de recherche en informatique. Comme beaucoup de travaux en informatique, ce travail n'est pas une fin en soi mais une ouverture, ici vers la construction d'une nouvelle méthodologie pour la visualisation d'informations. Nous avons, au cours de ce mémoire, énoncé ses principales caractéristiques, à la fois théoriques et pratiques. Comme nous l'avons vu, la plupart des techniques de visualisation actuelles ne sont pas encore adaptées à l'exploration et la compréhension de modèles ou de données devenant de plus en plus complexes. Inspirée des travaux réalisés dans le domaine de la modélisation des systèmes complexes, en particulier de ceux portant sur la modélisation à base d'agents, notre approche de visualisation est porteuse de la même ambition, à savoir fournir une représentation qui, sans dénaturer la complexité des données initiales, permet à un utilisateur de facilement multiplier les points de vue pour en dégager du sens. En ce sens, elle constitue une rupture par rapport aux approches classiques de visualisation.

Cette rupture est concrétisée par un ensemble de contributions qui nous paraissent importantes, et ce, d'autant plus qu'elles sont immédiatement réutilisables dans le cadre d'une implémentation générique. La première consiste en la possibilité de décrire, au sein d'une plate-forme de modélisation, des agents de visualisation, dotés de propriétés et de comportements graphiques ; la seconde réside dans les possibilités d'organiser dynamiquement ces agents au sein de structures originales baptisées modèle de visualisation à base d'agents, capables de filtrer et représenter n'importe quel type de données en entrée par

le biais des actions, interactions et comportements collectifs de leurs agents ; la troisième, enfin, offre la possibilité de projeter ces modèles et leurs agents dans un environnement immersif en 3 dimensions, qui démultiplie les possibilités de représentation visuelle tout en offrant l'opportunité d'une interaction plus intuitive avec l'utilisateur.

Ces contributions, complètement opérationnelles, ont été réalisées dans le but de faciliter, dans un premier temps, la visualisation des données complexes issues de simulations, et s'adressent ainsi en priorité à la communauté des modélisateurs, et plus particulièrement des modélisateurs à base d'agents, déjà familiers avec les concepts sur lesquels elles reposent. Dans un deuxième temps, nous avons élargi notre cible d'utilisateurs potentiels aux chercheurs, non modélisateurs, qui souhaitent visualiser et interagir avec des données de façon moins contrainte qu'avec les approches existantes. Notre souhait est que cette nouvelle technique, tout comme l'approche à base d'agents a permis à des chercheurs issus de disciplines diverses de se familiariser avec la modélisation des systèmes complexes, permettra à un public plus large de se familiariser avec la visualisation d'informations.

Le besoin d'analyser visuellement des masses croissantes de données n'est pas nouveau, mais la visualisation d'informations reste un domaine récent dans lequel beaucoup de pistes restent encore à explorer. La méthodologie que nous proposons, qui prend appui sur les contributions précédentes, exige néanmoins de l'utilisateur, dès lors que la tâche de visualisation envisagée est un peu complexe, un certain degré de familiarisation avec la programmation, ce qui peut représenter un frein à son adoption par des utilisateurs néophytes. Même si nous avons pris soin de proposer des outils simples pour atténuer cette difficulté, en offrant par exemple des possibilités d'interaction directe avec un modèle de visualisation, il nous semble néanmoins important de souligner que la programmation d'interfaces ou de modèles a toutes les chances de s'imposer comme la piste la plus féconde du domaine de la visualisation d'informations, même si aucune méthodologie n'y a encore été véritablement proposée. Nous estimons avoir contribué, dans ce manuscrit, même modestement, à ce mouvement.

Perspectives

Jamais les données n'ont été produites de façon aussi variée, volumineuse et rapide dans l'histoire. Ce phénomène récent est désigné sous le terme de *Big Data* et il devient de plus en plus difficile, sachant que ces données peuvent être dynamiques et hétérogènes, d'envisager des traitements complètement centralisés afin d'identifier et de classifier les structures qu'elles recèlent (hiérarchiques, relationnelles et spatiotemporelles) ainsi que découvrir leurs éventuelles relations (corrélation, causalité, etc.). Notre approche, du fait de son assise multi-agent, se positionne idéalement pour ouvrir la voie à une forme décentralisée de représentation et d'analyse de données.

Dans cette approche, au-delà des modèles présentés dans le mémoire, les capacités d'action et de communication des agents graphiques permettent d'imaginer que, en plus de prendre une décision autonome pour se représenter individuellement à l'écran, ils coopèrent, également de façon autonome, pour que la représentation des données dont ils ont la charge soit organisée de la façon la plus pertinente ou la plus intelligible pour un utilisateur.

Cette coopération peut prendre plusieurs formes. Guillaume Hutzler a par exemple exploré dans sa thèse ([Hutzler, 2000]) des mécanismes d'auto-organisation proches de la Vie Artificielle pour adapter la visualisation, à la fois aux actions de l'utilisateur et à la dynamique des données (données et utilisateurs constituant l'"environnement" des agents, respectivement en termes de ressources et en termes de "feedback"). Cette approche a donné lieu à de nombreux travaux, qui se sont néanmoins heurtés, dans le cas de données massives, à la nécessité de pouvoir générer des représentations structurées et prédictibles, donc éventuellement dépendantes de règles (de visualisation, d'abstraction, etc.) qui ne soient pas simplement le fruit de l'auto-organisation des agents.

Nos perspectives se situent dans cette direction de recherche, et envisagent donc plutôt de s'appuyer sur les travaux autour des organisations fonctionnelles et hiérarchiques des systèmes multi-agents [Da Silva and Demazeau, 2002] afin de distribuer, au mieux et surtout de façon explicite, la connaissance de visualisation et les besoins de l'utilisateur. Dans cette optique, chaque agent peut posséder sa propre expertise et des capacités de coopération avec d'autres agents. Couplé à la définition récursive autorisée par notre approche (un modèle de visualisation visualisant les sorties d'autres modèles de visualisation), on peut envisager des agents ayant des connaissances graphiques (un agent

intégrant la théorie de la Gestalt, un autre la théorie des couleurs, etc..) pour améliorer leur représentation graphique, chacun d'entre eux ayant la liberté d'adapter sa représentation en fonction des données qu'il traite. A ces agents spécialisés, peuvent être ajoutés des macro-agents permettant de combiner certains agents, de coordonner des groupes d'agents ou éventuellement d'en créer d'autres. Un groupe d'agents peut aussi collaborer pour ne former plus qu'une seule représentation visuelle. Dans cette perspective, la spécialisation des agents n'est pas que visuelle puisque certains agents devront nécessairement être spécialisés dans des tâches liées à la fouille de données pour détecter, par exemple, des patterns spécifiques dans les données ou dans les agents visuels. Enfin, la prise en compte de l'utilisateur peut être envisagée si l'on considère que ses actions peuvent elles aussi être analysées par d'autres agents spécialisés. Ces agents auront donc la capacité de mettre à jour leur représentation, leur connaissance, mais aussi la représentation d'autres agents en fonction des actions effectuées par l'utilisateur, ce qui lui permet d'être directement impliqué dans le processus dynamique de représentation visuelle.

Dans cette perspective, toute la littérature produite sur l'approche multi-agent devient dès lors une source potentielle d'inspiration pour construire des systèmes de visualisation adaptables à des données complexes et aux exigences des utilisateurs. Et notre intime conviction est que c'est par l'intermédiaire de la coopération de multiples agents d'extraction, d'analyse et de visualisation que des connaissances qu'aucune méthode ne pourrait reproduire de manière centralisée seront mises à jour.

Bibliographie

[Albert and Barabási, 2002] Albert, R. and Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1) :47.

[Allan, 2010] Allan, R. J. (2010). *Survey of agent based modelling and simulation tools*. Science & Technology Facilities Council.

[Amar et al., 2005] Amar, R., Eagan, J., and Stasko, J. (2005). Low-level components of analytic activity in information visualization. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pages 111–117. IEEE.

[Andrienko et al., 2010] Andrienko, G., Andrienko, N., Demsar, U., Dransch, D., Dykes, J., Fabrikant, S. I., Jern, M., Kraak, M.-J., Schumann, H., and Tominski, C. (2010). Space, time and visual analytics. *International Journal of Geographical Information Science*, 24(10) :1577–1600.

[Andrienko et al., 2003] Andrienko, N., Andrienko, G., and Gatalsky, P. (2003). Exploratory spatio-temporal visualization : an analytical review. *Journal of Visual Languages & Computing*, 14(6) :503–541.

[Auchincloss and Roux, 2008] Auchincloss, A. H. and Roux, A. V. D. (2008). A new tool for epidemiology : the usefulness of dynamic-agent models in understanding place effects on health. *American journal of epidemiology*, 168(1) :1–8.

[Batty et al., 1998] Batty, M., Conroy, R., Hillier, B., Jiang, B., Desyllas, J., Mottrom, C., Penn, A., Hudson-Smith, A., and Turner, A. (1998). The virtual tate. *Centre for Advanced Spatial Analysis (UCL)*.

[Beisbart, 2012] Beisbart, C. (2012). How can computer simulations produce new knowledge ? *European Journal for Philosophy of Science*, 2(3) :395–434.

[Bertin, 1967] Bertin, J. (1967). Sémiologie graphique. *Paris : Mouton, 1966, 432p.*

[Bertin, 1981] Bertin, J. (1981). *Graphics and graphic information processing*. Walter de Gruyter.

[Bertin, 1983] Bertin, J. (1983). Semiology of graphics : diagrams, networks, maps.

[Beslon, 2008] Beslon, G. (2008). Apprivoiser la vie : Modélisation individu-centrée de systèmes biologiques complexes. *Habilitation à Diriger des Recherches, INSA-Lyon*.

[Billari and Prskawetz, 2003] Billari, F. C. and Prskawetz, A. (2003). *Agent-based computational demography : Using simulation to improve our understanding of demographic behaviour*. Springer Science & Business Media.

[Birks et al., 2012] Birks, D., Townsley, M., and Stewart, A. (2012). Generative explanations of crime : Using simulation to test criminological theory*. *Criminology*, 50(1) :221–254.

[Bivand et al., 2008] Bivand, R. S., Pebesma, E. J., and Gómez-Rubio, V. (2008). *Applied spatial data analysis with R*, volume 747248717. Springer.

[Bohnacker et al., 2012] Bohnacker, H., Gross, B., Laub, J., and Lazzeroni, C. (2012). *Generative Design : Visualize, Program, and Create with Processing*. Princeton Architectural Press.

[Borshchev, 2013] Borshchev, A. (2013). *The Big Book of Simulation Modeling : Multimethod Modeling with AnyLogic 6*. AnyLogic North America.

[Bouchaud, 2013] Bouchaud, J.-P. (2013). Crises and collective socio-economic phenomena : simple models and challenges. *Journal of Statistical Physics*, 151(3-4) :567–606.

[Brockmann and Helbing, 2013] Brockmann, D. and Helbing, D. (2013). The hidden geometry of complex, network-driven contagion phenomena. *Science*, 342(6164) :1337–1342.

[Brown, 2006] Brown, M. C. (2006). *Hacking GoogleMaps and GoogleEarth (Extreme-Tech)*. John Wiley and Sons, Inc.

[Card et al., 1999] Card, S. K., Mackinlay, J. D., and Shneiderman, B. (1999). *Readings in information visualization : using vision to think*. Morgan Kaufmann.

[Carden, 2006] Carden, T. (2006). Travel time tube map.

[Castle, 2007] Castle, C. J. E. (2007). *Agent-Based modelling of pedestrian evacuation : study of London's King's Cross underground station*. PhD thesis, University College London (University of London).

[Cederman, 2005] Cederman, L.-E. (2005). Computational models of social forms : Advancing generative process theory1. *American Journal of Sociology*, 110(4) :864–893.

[Chin et al., 2012] Chin, L., Worth, D., Greenough, C., Coakley, S., Holcombe, M., and Gheorghe, M. (2012). *Flame-ii : a redesign of the flexible large-scale agent-based modelling environment*. STFC.

[Compieta et al., 2007] Compieta, P., Di Martino, S., Bertolotto, M., Ferrucci, F., and Kechadi, T. (2007). Exploratory spatio-temporal data mining and visualization. *Journal of Visual Languages & Computing*, 18(3) :255–279.

[Coutaz, 1987] Coutaz, J. (1987). Pac, an object oriented model for dialog design. In *Human-Computer Interaction-INTERACT*, volume 87, pages 431–436.

[Crooks et al., 2008] Crooks, A., Castle, C., and Batty, M. (2008). Key challenges in agent-based modelling for geo-spatial simulation. *Computers, Environment and Urban Systems*, 32(6) :417–430.

[Crooks et al., 2011] Crooks, A. T., Hudson-Smith, A., and Patel, A. (2011). Advances and techniques for building 3d agent-based models for urban systems. *no*, 1 :49–65.

[Cummings et al., 2004] Cummings, D. A., Irizarry, R. A., Huang, N. E., Endy, T. P., Nisalak, A., Ungchusak, K., and Burke, D. S. (2004). Travelling waves in the occurrence of dengue haemorrhagic fever in thailand. *Nature*, 427(6972) :344–347.

[Da Silva and Demazeau, 2002] Da Silva, J. L. T. and Demazeau, Y. (2002). Vowels co-ordination model. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems : part 3*, pages 1129–1136. ACM.

[de Reffye et al., 1988] de Reffye, P., Edelin, C., Françon, J., Jaeger, M., and Puech, C. (1988). Plant models faithful to botanical structure and development. In *ACM SIGGRAPH Computer Graphics*, volume 22, pages 151–158. ACM.

[Dix, 2009] Dix, A. (2009). *Human-computer interaction*. Springer.

[Dorin and Geard, 2014] Dorin, A. and Geard, N. (2014). The practice of agent-based model visualization. *Artificial life*, 20(2) :271–289.

[Drogoul, 1993] Drogoul, A. (1993). *De la simulation multi-agents à la résolution collective de problèmes*. PhD thesis, Thesis at University of Paris VI.

[Drogoul et al., 2003] Drogoul, A., Vanbergue, D., and Meurisse, T. (2003). Multi-agent based simulation : Where are the agents ? In *Multi-agent-based simulation II*, pages 1–15. Springer.

[Duke and Harrison, 1993] Duke, D. and Harrison, M. (1993). Towards a theory of interactors. *The Amodeus Project, Esprit Basic Research*, 7040.

[Epstein, 2006] Epstein, J. M. (2006). *Generative social science : Studies in agent-based computational modeling*. Princeton University Press.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

[Farmer and Foley, 2009] Farmer, J. D. and Foley, D. (2009). The economy needs agent-based modelling. *Nature*, 460(7256) :685–686.

[Foley et al., 1997] Foley, J. D., Dam, A. v., Feiner, S. K., Hughes, J. F., and Carter, M. P. (1997). Computer graphics : Principles and practice, in c. *Color Research and Application*, 22(1) :65–65.

[Friendly and Denis, 2008] Friendly, M. and Denis, D. J. (2008). Milestones in the history of thematic cartography, statistical graphics, and data visualization. *Seeing Science : Today American Association for the Advancement of Science*.

[Fry, 2004] Fry, B. J. (2004). *Computational information design*. PhD thesis, Massachusetts Institute of Technology.

[Furht, 2011] Furht, B. (2011). *Handbook of augmented reality*. Springer Science & Business Media.

[Galanter, 2003] Galanter, P. (2003). What is generative art. In *Complexity Theory as a Context for Art Theory. Generative Art 2003 Conference*.

[Gaudou et al., 2013] Gaudou, B., Sibertin-Blanc, C., Therond, O., Amblard, F., Arcangeli, J.-P., Balestrat, M., Charron-Moirez, M.-H., Gondet, E., Hong, Y., Louail, T., et al. (2013). The maelia multi-agent platform for integrated assessment of low-water management issues. In *MABS, Multi-Agent-Based Simulation XIV-International Workshop (to appear, 2013)*.

[Gil-Quijano et al., 2012] Gil-Quijano, J., Louail, T., and Hutzler, G. (2012). From biological to urban cells : lessons from three multilevel agent-based models. In *Principles and Practice of Multi-Agent Systems*, pages 620–635. Springer.

[Goldberg, 1984] Goldberg, A. (1984). *SMALLTALK-80 : the interactive programming environment*. Addison-Wesley Longman Publishing Co., Inc.

[Gould, 1987] Gould, S. J. (1987). *Time's arrow, time's cycle : Myth and metaphor in the discovery of geological time*. Harvard University Press.

[Grenfell et al., 2001] Grenfell, B., Bjørnstad, O., and Kappey, J. (2001). Travelling waves and spatial hierarchies in measles epidemics. *Nature*, 414(6865) :716–723.

[Grimm et al., 2010] Grimm, V., Berger, U., DeAngelis, D. L., Polhill, J. G., Giske, J., and Railsback, S. F. (2010). The odd protocol : a review and first update. *Ecological modelling*, 221(23) :2760–2768.

[Haklay and Weber, 2008] Haklay, M. and Weber, P. (2008). Openstreetmap : User-generated street maps. *Pervasive Computing, IEEE*, 7(4) :12–18.

[Hallé et al., 1978] Hallé, F., Oldeman, R. A., Tomlinson, P. B., et al. (1978). *Tropical trees and forests : an architectural analysis*. Springer-Verlag.

[Hamilton, 1994] Hamilton, J. D. (1994). *Time series analysis*, volume 2. Princeton university press Princeton.

[Han et al., 2006] Han, J., Kamber, M., and Pei, J. (2006). *Data mining : concepts and techniques*. Morgan kaufmann.

[Harrower and Brewer, 2003] Harrower, M. and Brewer, C. A. (2003). Colorbrewer. org : an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1) :27–37.

[Hartigan and Wong, 1979] Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136 : A k-means clustering algorithm. *Applied statistics*, pages 100–108.

[Heer and Bostock, 2010] Heer, J. and Bostock, M. (2010). Declarative language design for interactive visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6) :1149–1156.

[Herman et al., 2000] Herman, I., Melançon, G., and Marshall, M. S. (2000). Graph visualization and navigation in information visualization : A survey. *Visualization and Computer Graphics, IEEE Transactions on*, 6(1) :24–43.

[Hill, 1992] Hill, R. D. (1992). The abstraction-link-view paradigm : Using constraints to connect user interfaces to applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 335–342. ACM.

[Hudson-Smith, 2003] Hudson-Smith, A. P. (2003). *Digitally distributed urban environments : The prospects for online planning.* PhD thesis, University College London (University of London).

[Hutzler, 2000] Hutzler, G. (2000). *Du Jardin des Hasards aux Jardins de Données : une approche artistique et multi-agent des interfaces Homme/ Systèmes Complexes.* PhD thesis.

[Hutzler and Renault, 2000] Hutzler, G. and Renault, V. (2000). Les jardins de données : des sociétés d'agents pour la visualisation de données complexes. *D. Scapin et E. Vergison, éditeurs, ERGO-IHM*, pages 94–101.

[Janssen and Ostrom, 2006] Janssen, M. A. and Ostrom, E. (2006). Empirically based, agent-based models. *Ecology and Society*, 11(2) :37.

[Johnson and Shneiderman, 1991] Johnson, B. and Shneiderman, B. (1991). Tree-maps : A space-filling approach to the visualization of hierarchical information structures. In *Visualization, 1991. Visualization'91, Proceedings., IEEE Conference on*, pages 284–291. IEEE.

[Johnston et al., 2001] Johnston, K., Ver Hoef, J. M., Krivoruchko, K., and Lucas, N. (2001). *Using ArcGIS geostatistical analyst*, volume 380. Esri Redlands.

[Joliveau, 1996] Joliveau, T. (1996). Gérer l'environnement avec des sig mais qu'est-ce qu'un sig ? *Revue de géographie de Lyon*, 71(2) :101–110.

[Judelman, 2004] Judelman, G. (2004). Aesthetics and inspiration for visualization design : bridging the gap between art and science. In *Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on*, pages 245–250. IEEE.

[Kandinsky, 1991] Kandinsky, W. (1991). Point et ligne sur plan. *Folio essais, Gallimard.*

[Keim et al., 2006] Keim, D. A., Mansmann, F., Schneidewind, J., and Schreck, T. (2006). Monitoring network traffic with radial traffic analyzer. In *Visual Analytics Science And Technology, 2006 IEEE Symposium On*, pages 123–128. IEEE.

[Keim et al., 2008] Keim, D. A., Mansmann, F., Schneidewind, J., Thomas, J., and Ziegler, H. (2008). *Visual analytics : Scope and challenges.* Springer.

[Klein, 2003] Klein, J. (2003). Breve : a 3d environment for the simulation of decentralized systems and artificial life. In *Proceedings of the eighth international conference on Artificial life*, pages 329–334.

[Klügl, 2009] Klügl, F. (2009). Sesam : Visual programming and participatory simulation for agent-based models. *Published in Multi-Agent Systems : Simulation and Applications, by AM Uhrmacher and D. Weyns*, pages 477–507.

[Koblin, 2006] Koblin, A. (2006). Flight patterns. In *ACM SIGGRAPH 2006 Computer animation festival*, page 203. ACM.

[Koestler, 1967] Koestler, A. (1967). The ghost in the machine. 1967. *London : Hutchinson*.

[Komosiński and Ulatowski, 1999] Komosiński, M. and Ulatowski, S. (1999). Framsticks : Towards a simulation of a nature-like world, creatures and evolution. In *Advances in Artificial Life*, pages 261–265. Springer.

[Korein and Badler, 1983] Korein, J. and Badler, N. (1983). Temporal anti-aliasing in computer generated animation. In *ACM SIGGRAPH Computer Graphics*, volume 17, pages 377–388. ACM.

[Kornhauser et al., 2009] Kornhauser, D., Wilensky, U., and Rand, W. (2009). Design guidelines for agent based model visualization. *Journal of Artificial Societies and Social Simulation*, 12(2) :1.

[Kraak and Koussoulakou, 2005] Kraak, M.-J. and Koussoulakou, A. (2005). A visualization environment for the space-time-cube. In *Developments in Spatial Data Handling*, pages 189–200. Springer.

[Kravari and Bassiliades, 2015] Kravari, K. and Bassiliades, N. (2015). A survey of agent platforms. *Journal of Artificial Societies and Social Simulation*, 18(1) :11.

[Krugman and Krugman, 1996] Krugman, P. R. and Krugman, P. (1996). *The self-organizing economy*. Blackwell Oxford.

[Lakoff and Johnson, 2008] Lakoff, G. and Johnson, M. (2008). *Metaphors we live by*. University of Chicago press.

[Lamarche-Perrin et al., 2012] Lamarche-Perrin, R., Vincent, J.-M., Demazeau, Y., et al. (2012). Informational measures of aggregation for complex systems analysis. *Rapport de recherche RR-LIG-026, Laboratoire d'Informatique de Grenoble, Grenoble, France*.

[Lamping and Rao, 1996] Lamping, J. and Rao, R. (1996). Visualizing large trees using the hyperbolic browser. In *Conference companion on Human factors in computing systems*, pages 388–389. ACM.

[Lamy et al., 1999] Lamy, S., Ruas, A., Demazeau, Y., Jackson, M., Mackaness, W., and Weibel, R. (1999). The application of agents in automated map generalisation. In *19th international cartographic conference*, volume 13.

[Lau and Vande Moere, 2007] Lau, A. and Vande Moere, A. (2007). Towards a model of information aesthetics in information visualization. In *Information Visualization, 2007. IV'07. 11th International Conference*, pages 87–92. IEEE.

[Lauer and Parker, 2008] Lauer, J. and Parker, G. (2008). Modeling framework for sediment deposition, storage, and evacuation in the floodplain of a meandering river : Theory. *Water Resources Research*, 44(4).

[Lauer et al., 2014] Lauer, W., Viparelli, E., and Piegay, H. (2014). A 1-d size specific numerical model for gravel transport that includes sediment exchange with a floodplain. In *EGU General Assembly Conference Abstracts*, volume 16, page 10126.

[Luke et al., 2005] Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and Balan, G. (2005). Mason : A multiagent simulation environment. *Simulation*, 81(7) :517–527.

[Macal and North, 2009] Macal, C. M. and North, M. J. (2009). Agent-based modeling and simulation. In *Winter Simulation Conference*, pages 86–98. Winter Simulation Conference.

[MacEachren, 2004] MacEachren, A. M. (2004). *How maps work : representation, visualization, and design*. Guilford Press.

[Maeda and Foreword By-Antonelli, 2001] Maeda, J. and Foreword By-Antonelli, P. (2001). *Design by numbers*. MIT Press.

[Manzo, 2014] Manzo, G. (2014). Potentialités et limites de la simulation multi-agents : une introduction. *Revue française de sociologie*, 55(4) :653–688.

[Mathieu et al., 2005] Mathieu, P., Beaufils, B., and Brandouy, O. (2005). *Artificial economics : agent-based methods in finance, game theory and their applications*, volume 564. Springer Science & Business Media.

[McCarthy and Hayes, 1968] McCarthy, J. and Hayes, P. (1968). *Some philosophical problems from the standpoint of artificial intelligence*. Stanford University USA.

[Meirelles, 2013] Meirelles, I. (2013). *Design for Information : An Introduction to the Histories, Theories, and Best Practices Behind Effective Information Visualizations.* Rockport publishers.

[Meyer et al., 2006] Meyer, M., Gîrba, T., and Lungu, M. (2006). Mondrian : an agile information visualization framework. In *Proceedings of the 2006 ACM symposium on Software visualization*, pages 135–144. ACM.

[Minsky, 1965] Minsky, M. (1965). Matter, mind and models. *MIT Press.*

[Muldoon, 2007] Muldoon, R. (2007). Robust simulations. *Philosophy of Science*, 74(5) :873–883.

[Newman, 1994] Newman, W. (1994). A preliminary analysis of the products of hci research, using pro forma abstracts. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 278–284. ACM.

[Nikolai and Madey, 2009] Nikolai, C. and Madey, G. (2009). Tools of the trade : A survey of various agent based modeling platforms. *Journal of Artificial Societies and Social Simulation*, 12(2) :2.

[North et al., 2013] North, M. J., Collier, N. T., Ozik, J., Tatara, E. R., Macal, C. M., Bragen, M., and Sydelko, P. (2013). Complex adaptive systems modeling with repast simphony. *Complex adaptive systems modeling*, 1(1) :1–26.

[O'Sullivan, 2008] O'Sullivan, D. (2008). Geographical information science : agent-based models. *Progress in Human Geography.*

[Pandit et al., 1983] Pandit, S. M., Wu, S.-M., et al. (1983). *Time series and system analysis with applications.* Wiley New York.

[Parunak et al., 1998] Parunak, H. V. D., Savit, R., and Riolo, R. L. (1998). Agent-based modeling vs. equation-based modeling : A case study and users' guide. In *Multi-agent systems and agent-based simulation*, pages 10–25. Springer.

[Paterno et al., 1995] Paterno, F., Leonardi, A., and Pangoli, S. (1995). A tool-supported approach to the refinement of interactive systems. In *Interactive Systems : Design, Specification, and Verification*, pages 149–159. Springer.

[Pettersson, 2002] Pettersson, R. (2002). *Information design : An introduction*, volume 3. John Benjamins Publishing.

[Pfaff, 1985] Pfaff, G. E. (1985). *User interface management systems*. Springer-Verlag New York, Inc.

[Phong, 1975] Phong, B. T. (1975). Illumination for computer generated pictures. *Communications of the ACM*, 18(6) :311–317.

[Pipes, 2008] Pipes, A. (2008). *Foundations of art and design*. Laurence King.

[QGis, 2011] QGis, D. (2011). Quantum gis geographic information system. *Open Source Geospatial Foundation Project*.

[Railsback et al., 2005] Railsback, S., Lytinen, S., and Grimm, V. (2005). Stupidmodel and extensions : A template and teaching tool for agent-based modeling platforms. *Swarm Development Group. http ://condor. depaul. edu/˜ slytinen/abm*.

[Railsback and Grimm, 2011] Railsback, S. F. and Grimm, V. (2011). *Agent-based and individual-based modeling : a practical introduction*. Princeton University Press.

[Railsback et al., 2006] Railsback, S. F., Lytinen, S. L., and Jackson, S. K. (2006). Agent-based simulation platforms : Review and development recommendations. *Simulation*, 82(9) :609–623.

[Reynolds, 1987] Reynolds, C. W. (1987). Flocks, herds and schools : A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 25–34. ACM.

[Robertson et al., 1991] Robertson, G. G., Mackinlay, J. D., and Card, S. K. (1991). Cone trees : animated 3d visualizations of hierarchical information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 189–194. ACM.

[Saitta and Zucker, 2001] Saitta, L. and Zucker, J.-D. (2001). A model of abstraction in visual perception. *Applied Artificial Intelligence*, 15(8) :761–776.

[Saitta and Zucker, 2013] Saitta, L. and Zucker, J.-D. (2013). *Abstraction in artificial intelligence and complex systems*. Springer.

[Schelling, 1971] Schelling, T. C. (1971). Dynamic models of segregation†. *Journal of mathematical sociology*, 1(2) :143–186.

[Schweitzer, 1997] Schweitzer, F. (1997). *Self-organization of complex structures : From individual to collective dynamics*. CRC Press.

[Servat et al., 1998] Servat, D., Perrier, E., Treuil, J.-P., and Drogoul, A. (1998). When agents emerge from agents : Introducing multi-scale viewpoints in multi-agent simulations. In *Multi-Agent Systems and Agent-Based Simulation*, pages 183–198. Springer.

[Shiffman et al., 2012] Shiffman, D., Fry, S., and Marsh, Z. (2012). *The nature of code*. D. Shiffman.

[Shneiderman, 1996] Shneiderman, B. (1996). The eyes have it : A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343. IEEE.

[Shneiderman et al., 2010] Shneiderman, B., Pirolli, P., and Preece, J. (2010). Cyberinfrastructure for social action on national priorities. *Computer*, 43(11) :0020–21.

[Simon, 1965] Simon, H. A. (1965). The architecture of complexity. *General systems*, 10(1965) :63–76.

[Slocum et al., 2009] Slocum, T., McMaster, R., Kessler, F., and Howard, H. (2009). Thematic cartography and geovisualisation. *Prentice Hall, New Jersey, NJ*.

[Smith and Crooks, 2010] Smith, D. and Crooks, A. (2010). From buildings to cities : techniques for the multi-scale analysis of urban form and function. *Centre for Advanced Spatial Analysis (UCL)*.

[Smith and Conrey, 2007] Smith, E. R. and Conrey, F. R. (2007). Agent-based modeling : A new approach for theory building in social psychology. *Personality and social psychology review*, 11(1) :87–104.

[Sollenberger et al., 2005] Sollenberger, R. L., Willems, B., Della Rocco, P., Koros, A., and Truitt, T. (2005). Human-in-the-loop simulation evaluating the collocation of the user request evaluation tool, traffic management advisor, and controller-pilot data link communications : Experiment i-tool combinations. *US Department of Transportation, Federal Aviation Administration*.

[Soula et al., 2005] Soula, H., Robardet, C., Perrin, F., Gripon, S., Beslon, G., and Gandrillon, O. (2005). Modeling the emergence of multi-protein dynamic structures by principles of self-organization through the use of 3dspi, a multi-agent-based software. *BMC bioinformatics*, 6(1) :228.

[Thorne et al., 2007] Thorne, B. C., Bailey, A. M., and Peirce, S. M. (2007). Combining experiments with multi-cell agent-based modeling to study biological tissue patterning. *Briefings in bioinformatics*, 8(4) :245–257.

[Tisue and Wilensky, 2004] Tisue, S. and Wilensky, U. (2004). Netlogo : A simple environment for modeling complexity. In *International Conference on Complex Systems*, pages 16–21.

[Treuil et al., 2008] Treuil, J.-P., Drogoul, A., and Zucker, J.-D. (2008). *Modélisation et simulation à base d'agents : exemples commentés, outils informatiques et questions théoriques*. Dunod.

[Truong et al., 2011] Truong, V. X., Drogoul, A., Huynh, H. X., and Le, M. N. (2011). Modeling the brown plant hoppers surveillance network using agent-based model : application for the mekong delta region. In *Proceedings of the Second Symposium on Information and Communication Technology*, pages 127–136. ACM.

[Tukey, 1977] Tukey, J. W. (1977). Exploratory data analysis. *Reading, Ma*, 231 :32.

[Uhrmacher et al., 1997] Uhrmacher, A. M., Tyschler, P., and Tyschler, D. (1997). Concepts of object-and agent-oriented simulation. *Transactions of the Society for Computer Simulation*, 14(2) :59–67.

[Varenne, 2011] Varenne, F. (2011). *Modéliser le social-Méthodes fondatrices et évolutions récentes : Méthodes fondatrices et évolutions récentes*. Dunod.

[Venturini et al., 2015] Venturini, T., Jensen, P., and Latour, B. (2015). Fill in the gap. a new alliance for social and natural sciences. *Journal of Artificial Societies and Social Simulation*, 18(2) :11.

[Victor, 2012] Victor, B. (2012). Inventing on principle.

[Vo, 2012] Vo, D. A. (2012). *An operational architecture to handle multiple levels of representation in agent-based models*. PhD thesis. Thèse de doctorat dirigée par Drogoul, Alexis Informatique Université Pierre et Marie Curie - Paris 6 2012.

[Vo et al., 2012] Vo, D.-A., Drogoul, A., Zucker, J.-D., and Ho, T.-V. (2012). A modelling language to represent and specify emerging structures in agent-based model. In *Principles and Practice of Multi-Agent Systems*, pages 212–227. Springer.

[Ware, 2012] Ware, C. (2012). *Information visualization : perception for design*. Elsevier.

[Ware et al., 2002] Ware, C., Purchase, H., Colpoys, L., and McGill, M. (2002). Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2) :103–110.

[Wertheimer, 1938] Wertheimer, M. (1938). *Gestalt theory*. Hayes Barton Press.

[Wiener et al., 1948] Wiener, N. et al. (1948). *Cybernetics*. Hermann Paris.

[Yau, 2011] Yau, N. (2011). *Visualize this : the FlowingData guide to design, visualization, and statistics*. John Wiley & Sons.

Sites Internet

[1] Adobe flash. http://www.adobe.com/products/flash.html. Accessed : 2014-09-06.

[2] Arcgis 3d analyst. http://www.esri.com/software/arcgis/extensions/3danalyst. Accessed : 2014-08-18.

[3] Arcgis : A platform for designing and managing solutions through the application of geographic knowledge. http://www.esri.com/software/arcgis. Accessed : 2014-08-18.

[4] D3 : turns your social media data into interactive infographics, videos, and more. http://d3js.org/. Accessed : 2014-08-01.

[5] Jogl : Java binding for the opengl api. http://jogamp.org/jogl/www/. Accessed : 2014-09-06.

[6] Max - visual programming. http://cycling74.com/products/max/. Accessed : 2014-07-05.

[7] New york times, election results 2008. http://elections.nytimes.com/2008/results/president/map.html. Accessed : 2014-07-07.

[8] Opengl : The industry's foundation for high performance graphics. http://www.opengl.org/. Accessed : 2014-07-05.

[9] Prefuse - the prefuse visualization toolkit. http://prefuse.org/. Accessed : 2014-07-05.

[10] Processing - programming language, development environment, online community. https://www.processing.org/. Accessed : 2014-07-05.

[11] Qgis : A free and open source geographic information system. http://www.qgis.org. Accessed : 2014-08-18.

[12] Qgis planet : 3d viz with qgis and three.js. http://planet.qgis.org/planet/tag/3d/. Accessed : 2014-08-18.

[13] Threejs : A javascript 3d library. http://threejs.org/. Accessed : 2014-07-05.

[14] World mapper. http://www.worldmapper.org/. Accessed : 2014-09-06.

Publications

[Drogoul et al., 2014] Drogoul, A., Gaudou, B., Gasmi, N., Grignard, A., Taillandier, P., Tessier, O., and Vo, D. A. (2014). Understanding Past Crises to Better Manage the Present : Initiation to Geo-Historical Risk Modelling (The 1926 Red River Swelling). In *Perception and Risk Management - Methodological approaches applied to development*, Regional Social Sciences Summer University, pages 299–336. Journées de Tam Dao.

[Gasmi et al., 2015] Gasmi, N., Grignard, A., Drogoul, A., Gaudou, B., Taillandier, P., Tessier, O., and An, V. D. (2015). Reproducing and exploring past events using agent-based geo-historical models. In *Multi-Agent-Based Simulation XV*, pages 151–163. Springer.

[Grignard et al., 2013a] Grignard, A., Drogoul, A., and Zucker, J. D. (2013a). Complex systems simulation online visual analysis and assessment using dynamic aggregation operators. In *Soft Computing and Pattern Recognition (SoCPaR), 2013 International Conference of*, pages 288–291. IEEE.

[Grignard et al., 2013b] Grignard, A., Drogoul, A., and Zucker, J.-D. (2013b). A model-view/controller approach to support visualization and online data analysis of agent-based simulations. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2013 IEEE RIVF International Conference on*, pages 233–236. IEEE.

[Grignard et al., 2013c] Grignard, A., Drogoul, A., and Zucker, J.-D. (2013c). Online analysis and visualization of agent based models. In *Computational Science and Its Applications–ICCSA 2013*, pages 662–672. Springer.

[Grignard et al., 2013d] Grignard, A., Drogoul, A., and Zucker, J. D. (2013d). Using high-level 3d graphics library in agent-based simulation platform. *Advances in Smart Systems Research*, 3(1) :1–6.

[Grignard et al., 2015] Grignard, A., Fantino, G., Lauer, J. W., Verpeaux, A., and Drogoul, A. (2015). Agent-based visualization : A simulation tool for the analysis of river morphosedimentary adjustments. In *Multi-Agent-Based Simulation XVI*. To appear.

[Grignard et al., 2013e] Grignard, A., Taillandier, P., Gaudou, B., Vo, D. A., Huynh, N. Q., and Drogoul, A. (2013e). Gama 1.6 : Advancing the art of complex agent-based modeling and simulation. In *PRIMA 2013 : Principles and Practice of Multi-Agent Systems*, pages 117–131. Springer.

[Taillandier et al., 2014] Taillandier, P., Grignard, A., Gaudou, B., and Drogoul, A. (2014). Des données géographiques à la simulation à base d'agents : application de la plate-forme gama. *Cybergeo : European Journal of Geography*.