Étude des concepts logiciels et matériels pour un système interactif d'animation en temps réel

Sylvère Bruneaux

THESE

présentée par

Sylvère BRUNEAU

POUR OBTENIR LE TITRE DE DOCTEUR INGÉNIEUR

A L'UNIVERSITE JOSEPH FOURIER - GRENOBLE 1

Spécialité Informatique

ETUDE DES CONCEPTS LOGICIELS ET MATERIELS POUR UN SYSTEME INTERACTIF D'ANIMATION EN TEMPS REEL.

Thèse soutenue le 26 octobre 1989 devant la commission d'examen :

Mr J. Courtin	Président	Professeur à l'IUT2-Grenoble
Mr M. Mériaux	Rapporteur	Chargé de recherche au CNRS
Mr F. Martinez	Directeur	Professeur à l'UJF-Grenoble I
Mme N. Thalmann	Examinateur	Professeur à l'université de Genève
Mr P. Coudsi	Examinateur	Réalisateur de dessins animés

Laboratoire ARTEMIS-IMAG : BP 53X 38041 GRENOBLE CEDEX

"There is no particular mystery in animation... it's really very simple, and like anything that is simple, it is about the hardest thing in the world to do."

Bill Tytla au studio Walt Disney, le 28 juin 1937.

Remerciements

Je tiens à remercier,

Monsieur J. Courtin, qui m'a fait l'honneur d'accepter la présidence du jury de soutenance.

Monsieur M. Mériaux, pour sa disponibilité et les remarques judicieuses dont il m'a fait part pour l'amélioration de mon rapport.

Madame N. Thalmann, pour l'attention qu'elle a manifestée en acceptant de siéger au jury. J'espère qu'elle appréciera cette approche originale de l'animation.

Monsieur P. Coudsi, qui m'a fait profiter de son expérience de réalisateur lors de l'élaboration du système, avec lequel il a réalisé de superbes animations.

Monsieur F. Martinez, qui a déterminé l'orientation de mes recherches tout au long de cette étude et m'a fait part de sa grande expérience dans le domaine de la synthèse d'images.

Je n'oublierai pas la sympathique équipe graphique du laboratoire Artémis, ni la société Gétris Images, qui m'a permis de concilier recherche et responsabilités au sein d'une équipe dynamique.

Cette thèse a été réalisée dans le cadre de la convention CIFRE n° 65/86 établissant une collaboration entre le laboratoire ARTEMIS-IMAG et la société GETRIS IMAGES.

TABLE DES MATIERES

INTRODUCTION

S'il existe un domaine où l'avènement de l'ordinateur a été chaudement combattu, c'est bien celui de l'art! Comment ce symbole de logique et de raison, l'ordinateur, pouvait-il jouer un rôle dans la création artistique, où la liberté d'action et l'intuition ont la plus grande part? L'apparition d'une nouvelle technique ou d'un nouveau mode d'expression ne suscite-t-elle pas toujours le sceptiscisme et ne provoque-t-elle pas une certaine réticence à mettre en pratique ces nouvelles méthodes?

Cependant, aussitôt après l'apparition de l'ordinateur, quelques pionniers comme Ken Knowlton [Kno 64] aux Bell Telephone Laboratories commencèrent à utiliser ces machines pour créer des dessins et même les animer. Ils étaient alors plus informaticiens qu'artistes et programmaient leurs images plutôt qu'ils ne les dessinaient.

Aujourd'hui, les ordinateurs sont largement utilisés pour la peinture, l'architecture et même la musique. Leurs travaux sont reconnus et appréciés par les spécialistes. Il suffit pour s'en persuader d'assister à une des nombreuses expositions d'art moderne mettant en vedette l'ordinateur. Celui-ci permet de concrétiser la pensée de l'artiste, qui dispose d'outils beaucoup plus puissants que ses traditionnels pinceaux et tubes de gouaches. Le choix des couleurs est en effet interactif, les modifications et retouches des images instantanées...

Mais si l'ordinateur joue désormais un grand rôle dans la création des images, il est un domaine dans lequel il commence à faire une percée: l'animation, et plus particulièrement l'aide à la création de films d'animation. Déjà en 1974, le hongrois Peter Foldes gagnait le *Prix du Jury* au Festival du Film de Cannes pour son film *La Faim*, réalisé entièrement par ordinateur.

Cependant cette percée dans l'animation ne saurait se poursuivre sans le soin apporté à l'environnement proposé par l'outil informatique. Pour se faire accepter de l'artiste, l'informaticien, dans la spécification de son système d'animation, doit se familiariser avec les principes de base de l'animation "traditionnelle" et tenir compte de ses habitudes de travail. Pour consolider sa présence, le technicien doit tendre, en apparence tout au moins, à s'effacer devant le créatif.

Rendre l'outil convivial est une première étape, mais l'investissement informatique doit aussi se justifier par une amélioration des performances et donc une baisse des coûts de production.

Ainsi, ergonomie, performance et rentabiblité pourront faire de l'ordinateur un outil précieux dans la réalisation d'animations. C'est en tenant compte de ces critères qu'a été conçu le système d'animation faisant l'objet de cette thèse.

Ce système informatique original se base sur les principes d'animation traditionnelle et offre des possibilités temps réel de description et de visualisation de films d'animation.

Dans le premier chapitre nous présenterons les bases de l'animation par ordinateur et nous passerons en revue les différents systèmes existants à ce jour. Ces systèmes sont essentiellement orientés vers la génération des images.

Nous formaliserons, dans le second chapitre, les différentes étapes de la réalisation d'un film d'animation et nous définirons le vocabulaire propre à chacune d'elles. Nous déterminerons les phases critiques, incompressibles, et nous mettrons l'accent sur les étapes à optimiser.

L'analyse de la décomposition de la réalisation d'une animation nous conduira, dans le troisième chapitre, à établir des concepts d'animation originaux et adaptés.

L'implémentation de ces concepts au niveau matériel sera présentée au quatrième chapitre, notamment du point de vue temps réel.

Nous aborderons, au cinquième chapitre, l'environnement d'animation réalisé dans le cadre de cette thèse. Nous y présenterons les fonctionnalités des différents modules composant les bibliothèques et applications, puis les structures informatiques propres à chacun.

Les aspects ergonomie et dialogue seront présentés dans le dernier chapitre, des points de vue du concepteur, pour la réalisation d'outils de dialogue puissants, et de l'utilisateur, pour la manipulation de ces outils dans les applications.

CHAPITRE 1 : ETAT DE L'ART

Introduction

Avant d'aborder précisément le sujet de la thèse, il est nécessaire d'établir brièvement les bases de l'animation par ordinateur puis de passer en revue les différentes techniques d'animation par ordinateur connues à ce jour.

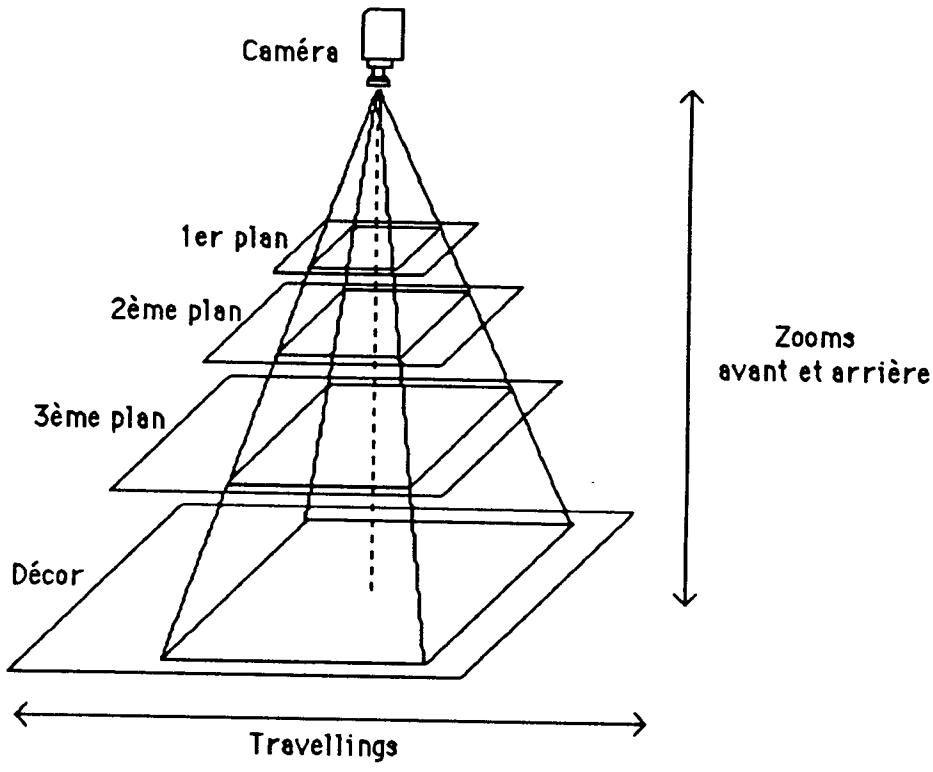
1. LES BASES DE L'ANIMATION PAR ORDINATEUR

1.1 Qu'est-ce-que l'animation?

Le terme animation provient étymologiquement du mot latin "animare" qui signifie "donner vie à".

Dès le début du 19ème siècle, de petits gadgets, aux noms barbares (Thaumatrope, Phenakistoscope...), faisaient leur apparition afin de créer l'illusion du mouvement. Ces gadgets rudimentaires n'avaient pour but que de distraire. A la fin du 19ème siècle, des mécanismes beaucoup plus perfectionnés (Praxinoscope) furent mis au point pour la projection d'images animées. L'apparition du cinématographe allait conduire à la création d'une nouvelle discipline, l'animation.

Les techniques d'animation traditionnelle allaient naître, et sont d'ailleurs toujours pratiquées. Une machine de prises de vue extrêmement perfectionnée, encombrante et coûteuse allait voir le jour : la multiplane.



Schématisation d'une multiplane.

Les dessins, peints sur des feuilles d'acétates, appelées encore celluloïdes, sont placés sur des niveaux transparents (plaques de cristal) à des distances bien calculées d'une caméra afin de réaliser la prise de vue de l'animation. Les plans de cristal peuvent se déplacer verticalement pour réaliser des travellings en profondeur avant et arrière (zooms) et horizontalement pour les déroulements de décors et translations de personnages.

L'ordinateur allait donner une troisième dimension à l'animation. S'il s'avère indispensable pour la réalisation d'animations 3D, il commence seulement à remplacer les techniques d'animation traditionnelle. Mais qu'entend-on par animation par ordinateur?

Une première approche peut être énoncée: l'animation est la présentation d'informations dynamiques, c'est-à-dire évoluant pendant un certain laps de temps.
Plus généralement, l'animation par ordinateur englobe la construction, l'enregistrement et la visualisation d'images graphiques exprimant un mouvement pendant un intervalle de temps. Il est cependant trop limitatif d'assimiler animation et mouvement, car l'animation peut exister sans mouvement. Par exemple, dans les métamorphoses, où un objet se transforme en un autre, dans des changements de couleurs, où un personnage rougit sous l'émotion, ou bien encore dans des changements d'intensité d'éclairage, où la scène s'assombrit lors d'un coucher de soleil.

L'animation est un art graphique, puisqu'il manipule un ensemble de vues 2D, enregistrées sur film ou support vidéo. L'ordinateur intervient non seulement dans la phase d'enregistrement de ces vues, mais aussi dans leur génération. Il permet en effet la création d'images 2D (graphiques, dessins...), mais simule également le processus de représentation de scènes 3D, qu'il projette dans le plan de l'écran, en respectant les lois "naturelles" de la géométrie et de l'optique, ce qui autorise beaucoup plus de réalisme que l'animation traditionnelle, dessinée entièrement sur des celluloïdes.

L'animation est également un art temporel, puisqu'elle implique constamment la notion de temps, qui en définit la 4ème dimension. L'animation regroupe, en effet, la manipulation de paramètres au cours du temps, leur évolution en un ensemble d'instants précis et l'enregistrement des résultats sous la forme d'images. Inversement, l'animation peut être également définie comme un ensemble d'images pour lequel un ou plusieurs paramètres changent à chaque instant. Ces paramètres peuvent être la position des pieds d'un personnage en marche, ou bien encore l'angle de rotation d'une planète en orbite autour d'une autre.

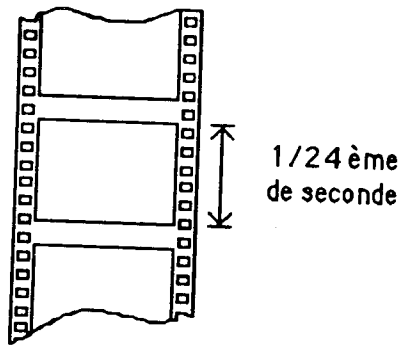
1.2 Le temps en animation

Le temps est le concept de base sur lequel l'animation est fondée. Comme le souligne Reynolds [Rey 87], le temps est une idée tellement implicite, axiomatique, qu'il est presque impossible d'en donner une définition abstraite courante.

Mathématiquement, le temps peut être assimilé à un espace à 1 dimension, ordonné et continu. Il est en effet souvent représenté sous la forme d'une droite, appelée chronogramme. Le choix d'une origine sur cette droite et la mise en correspondance entre ses points et l'ensemble des réels (ces 2 espaces sont isomorphes) définit la graduation du temps. Un instant est alors défini comme un point sur cette droite, un intervalle de temps comme un segment entre 2 points de la droite et la durée d'un intervalle comme la distance entre les 2 points extrêmes de cet intervalle.

D'un point de vue informatique, le temps est synonyme d'horloge, qui a pour rôle de générer des cycles de base. Le temps se trouve ainsi explicitement discrétisé et l'unité de temps en est l'instruction, c'est-à-dire le temps de traitement d'une commande élémentaire.

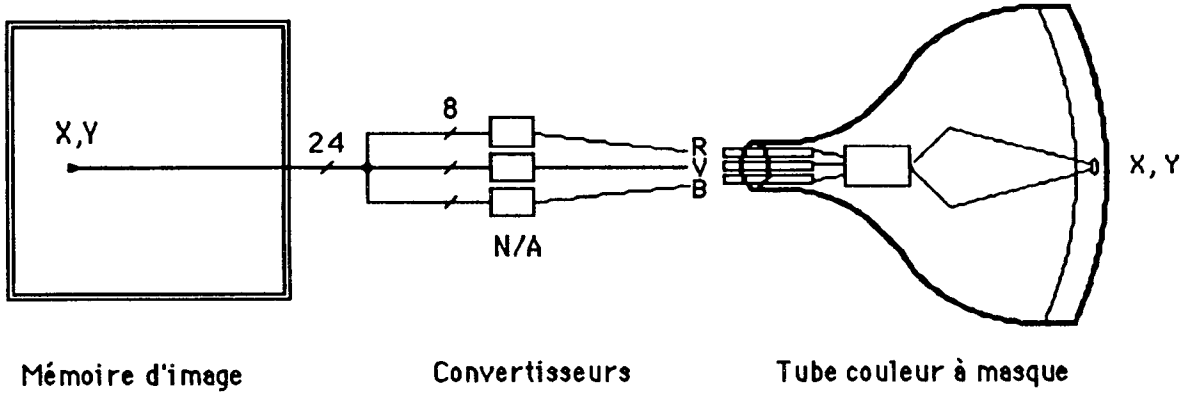
L'animation, prise dans le sens général du terme, suit ce même principe de discrétisation. Au cinéma, un film est composé d'une série d'images successivement photographiées. L'illusion du mouvement continu est produite par la projection du film à une vitesse régulière (24 images par seconde), conjuguée au phénomène de persistence de la rétine. L'oeil humain possède en effet une rémanence : deux images présentées successivement en un intervalle de temps inférieur à 1/15ème de seconde apparaissent liées.



Les images d'un film 35 mm.

1.3 Les caractéristiques vidéo de l'animation

Dans le domaine de la vidéo, il existe plusieurs techniques de visualisation (balayage cavalier, tube mémoire, balayage de trame) [Zar 85]. Nous allons nous limiter volontairement au principe de balayage de trame pour la visualisation d'images. Cette technique est en effet la seule adaptée à la visualisation d'images réalistes. Elle peut en effet être couplée à une mémoire d'image transmettant les informations à visualiser (les 3 composantes Rouge, Vert, Bleu) à la fréquence du balayage vidéo.

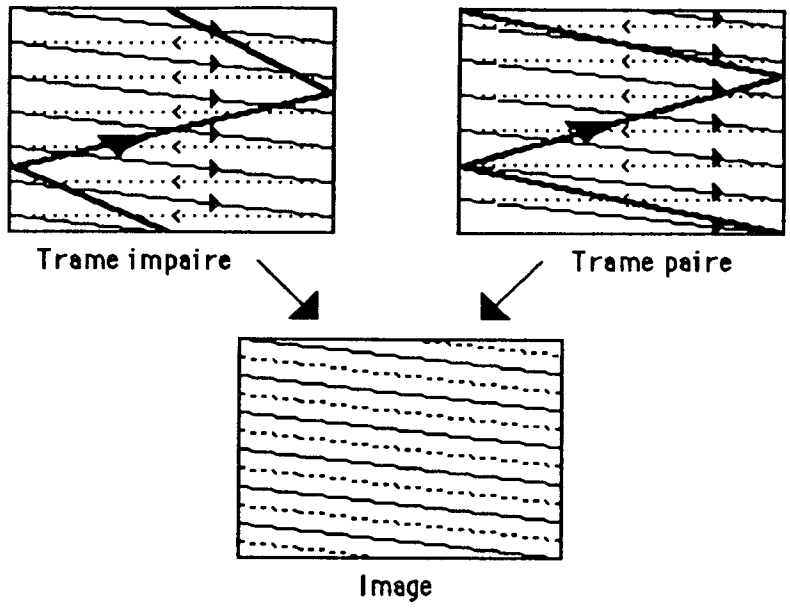


Le balayage de trame.

L'unité de temps dans un tel système est l'**image**. Ce terme englobe les 2 aspects temporel et spatial, puisqu'il représente aussi bien une durée égale à 1/25ème de seconde (1/30ème de seconde aux USA) que le résultat visualisé à l'écran.

Chaque image est balayée par une succession de lignes légèrement obliques (525 dans un système NTSC pour les USA, 625 dans un système PAL ou SECAM pour l'Europe). En fait pour éviter les phénomènes de papillotement, liés au fait que l'image est reconstruite sous nos yeux, on réalise un balayage dit entrelacé (ou interligné). On utilise également la propriété de rémanence des écrans.

Pendant 1/50ème de seconde (20 ms), on explore et transmet les lignes de rang impair (constituant ce qu'on appelle la **trame** impaire), puis pendant le 1/50ème de seconde suivant, les lignes de rang pair (constituant la trame paire). L'ensemble de l'image est bien transmis en 1/25ème de seconde et la rémanence visuelle reconstitue cette image à partir de l'enchevêtrement des 2 trames.



Les 2 trames composant une image.

A la fin de chaque trame, le faisceau d'électrons est coupé et effectue un retour vertical (en zig-zag car le balayage horizontal n'est pas bloqué) jusqu'à la 1ère ligne de la trame suivante. Le temps nécessaire à ce retour vertical est appelé **retour de trame**. La durée du retour de trame est équivalente au balayage de 25 lignes soit environ 2 ms.

Pour éviter les décrochements d'images, obtenus lors de modifications de la mémoire de trame pendant son balayage, toutes les modifications de la trame doivent impérativement survenir pendant le retour de trame.

Un système d'animation est dit **temps réel**, lorsqu'il est susceptible d'afficher les images de l'animation pendant la durée normale de cette animation. Ceci signifie qu'un tel système est capable de modifier les caractéristiques de l'image, dans la mémoire de trame, pendant chaque retour de trame de l'animation (système d'animation au 50ème de seconde) ou tous les 2 retours de trame (système d'animation au 25ème de seconde).

Un système d'animation ne pouvant pas mettre à jour les informations contenues dans la mémoire de trame au rythme imposé par l'animation, est dit système **image par image**, puisqu'il lui faut une durée équivalente à plusieurs images (durée) pour constituer une image (mémoire de trame) de la séquence d'animation.

La plupart des systèmes qui vont être présentés dans le paragraphe suivant sont des systèmes image par image.

2. L'ETAT DE L'ART

Le but de ce paragraphe est de présenter rapidement les divers systèmes d'animation par ordinateur développés dans les dix dernières années. Deux types d'utilisation de l'ordinateur pour la production d'animation peuvent être distingués, l'animation assistée par ordinateur et l'animation modélisée et produite par ordinateur [MTh 85a] [Per 88].

2.1 L'animation assistée par ordinateur

L'ordinateur constitue une aide et un support pour la création d'animation. Plutôt spécialisé dans le domaine de l'animation bidimensionnelle, l'ordinateur permet alors d'optimiser certaines phases de la production des dessins animés traditionnels, telles que l'acquisition de dessins ou la génération d'images, l'interpolation de dessins, l'interpolation de trajectoires...

2.1.1 L'animation traditionnelle assistée par ordinateur

Le travail d'un animateur traditionnel se décompose en 5 phases successives:

- la création des décors et dessins clés de l'animation: les dessins sont réalisés au crayon sur papier, à partir des informations regroupées dans un storyboard. Les dessins clés représentent les attitudes caractéristiques des personnages de l'animation et donnent une idée précise du déroulement de l'animation.

- la génération des dessins intermédiaires: à partir des dessins clés et des indications qui s'y rapportent, des "intervallistes" réalisent les dessins intermédiaires nécessaires à la continuité de l'animation.

- la validation de l'animation: avant de poursuivre la réalisation du dessin animé, l'animateur doit apprécier le rythme de l'animation et les mouvements des personnages. Pour ce faire, il utilise souvent la technique du "flip-book" : les dessins sur papier sont empilés (le premier en fond de pile) et l'animateur fait défiler rapidement les différentes feuilles, ce qui lui permet de réaliser un "line-test" de son animation. Cette méthode est très mal adaptée; il ne peut en effet visualiser que très peu d'images à la fois.

- le gouachage des dessins: tous les dessins sont ensuite recopiés à l'encre de chine sur des feuilles d'acétate (celluloïdes). Ils sont ensuite peints à la gouache (du coté opposé à l'encre de chine). Cette opération est délicate, la moindre variation de couleur est en effet perceptible lors de la visualisation. De plus, les celluloïdes ne sont pas totalement transparents, leur superposition entraine des dégradations de couleur. Le changement de l'ordre de superposition de différents celluloïdes au cours de l'animation entraine nécessairement leur regouachage.

- la prise de vue: elle est réalisée sur multiplane et doit suivre à la lettre les indications de la feuille de prise de vue, regroupant des informations de travelling (éloignement de la caméra), de déroulement (déplacement d'un plan horizontal) et surtout les indications temporelles de l'animation.

Outre l'informatisation des feuilles de storyboard et de prise de vue [Moi84], l'ordinateur fournit une aide précieuse aux animateurs traditionnels. Il n'intervient cependant qu'à partir de la phase de création des dessins intermédiaires. Les animateurs créent toujours leurs dessins-clés sur papier et ne commencent à utiliser l'ordinateur que lorsque une maquette complète de l'animation est réalisée. Les dessins clés sont alors "récupérés" par l'ordinateur à l'aide d'un système d'acquisition vidéo, qui transforme les dessins sur papier en images numériques, dont les contours sont extraits et éventuellement fermés automatiquement. L'ordinateur dispose alors de la structure de tous les dessins et peut générer les dessins intermédiaires par des méthodes d'interpolation [Bur 76] pouvant intégrer la notion de trajectoire [Ree 81].

Cette méthode n'est réellement efficace que lorsqu'elle génère peu de dessins intermédiaires, et a peu d'intérêt lorsque les changements de dessins sont importants. Le cas d'un personnage qui plie le bras, par exemple, doit faire intervenir beaucoup de dessins-clés, pour respecter les mouvements et les dimensions du bras. En outre, l'interpolation de 2 dessins-clés nécessite, entre autres, que ceux-ci soient composés d'un même nombre de contours polygônaux mis en correspondance 2 à 2 et possèdant idéalement le même nombre de points. La non vérification de ces contraintes entraine nécessairement la définition de classes d'interpolation et leur subdivision en un même nombre de points [Mar 77].

Après la génération de tous les dessins intermédiaires, l'ordinateur, à partir des informations temporelles de l'animation, peut générer une animation en line-test, qui permet aux animateurs de juger les mouvements de chacun des personnages, ainsi que leurs déformations. Lorsque l'animation n'apparait pas naturelle, de nouveaux dessins-clés peuvent être introduits afin d'augmenter la précision des dessins intermédiaires. Une animation en line-test se déroule en général en temps réel, donc sans l'obligation d'enregistrement image par image sur support vidéo des dessins au trait. L'ordinateur remplace avantageusement la méthode du flip-book de l'animateur traditionnel.

Lorsque les mouvements de chacun des personnages sont mis au point, un deuxième niveau de line-test réalise ensuite l'animation de tous les personnages de l'animation. Cette animation se déroule également en temps réel puisqu'il ne s'agit là encore que de l'affichage de quelques segments de droite, composant les contours des personnages et décors de l'animation. L'animateur peut ainsi vérifier immédiatement la bonne synchronisation de son animation, par exemple les points de rencontre de différents personnages.

Les dessins sont ensuite gouachés, soit manuellement par utilisation de palettes électroniques, soit automatiquement par des logiciels de gouachage capables de déterminer les couleurs à associer aux différentes parties des dessins clés ou intermédiaires. L'ordinateur remplace ainsi le gouachage manuel des dessins sur celluloïdes et supprime, entre autres, les problèmes d'altération des couleurs engendrés par la superposition des différents celluloïdes. De plus, le gouachage automatique permet également de générer des effets de couleurs impossibles à réaliser en animation traditionnelle, tels que des dégradés horizontaux, verticaux ou encore bilinéaires, etc...

Enfin, à partir de la feuille de prise de vue, le système génère automatiquement toutes les images de l'animation, en simulant par logiciel la superposition des différents celluloïdes [Ste 79]. Ces images sont enregistrées les unes après les autres sur un support vidéo, ou bien directement sur film 35mm.

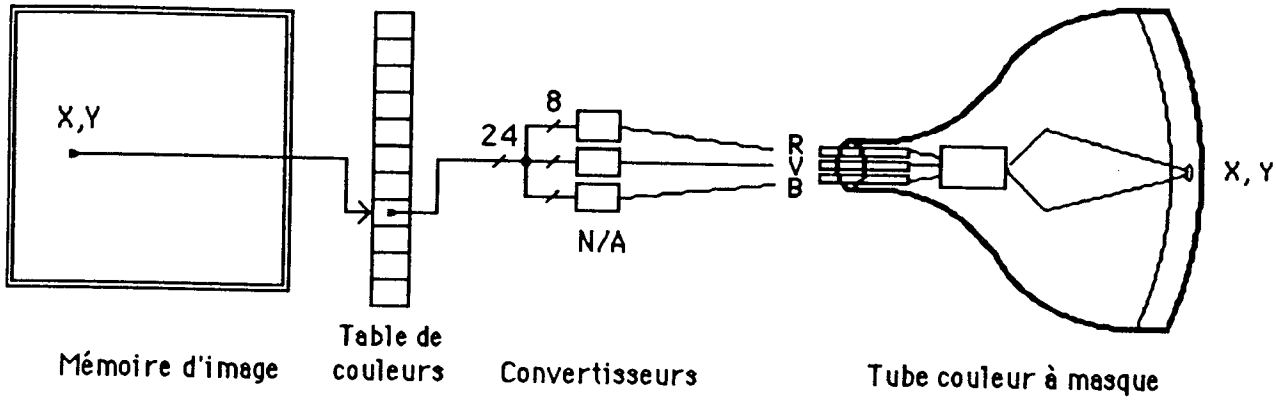
Pour que la qualité de l'animation par ordinateur soit comparable à l'animation traditionnelle, il est nécessaire d'intégrer un algorithme d'antialiasing dans la méthode de génération des images [Kor 83]. En effet, du fait de la résolution des écrans (720 * 576 pixels par exemple) des marches d'escalier, encore appelées effet de crénelage, ont tendance à apparaitre lors de la génération des dessins. Plus la résolution est grande plus ce phénomène est atténué. Il est donc nécessaire de considérer qu'un pixel est composé d'un certain nombre de sous-pixels. Tous les calculs sont effectués en tenant compte de ce suréchantillonage. L'affichage des pixels à l'écran, quant à lui, intègre toutes les informations contenues dans les sous-pixels de l'image.

Comme on peut le voir, le DAAO (Dessin Animé Assisté par Ordinateur) regroupe essentiellement des techniques d'automatisation de l'animation traditionnelle. Il laisse peu de place à la créativité puisque l'animation est totalement corrélée aux dessins et respecte à la lettre les indications contenues dans le storyboard.

2.1.2 L'animation par table de couleurs

L'animation par table de couleurs est un moyen de produire des animations limitées sur une image statique [Sho 79]. Cette technique permet "d'illuminer" successivement des formes prédéfinies dans l'image et est particulièrement adaptée à la réalisation de graphiques ou encore de jeux vidéo.

A la mémoire de trame du système d'animation est associée une table de couleurs, également appelée table de correspondance. L'image n'est alors pas constituée de couleurs mais d'adresses pointant sur les éléments de la table des couleurs. La couleur d'un pixel visible à l'écran est en fait la couleur contenue dans la case mémoire de la table de couleurs indexée par la valeur du pixel de la mémoire de trame. Ainsi la réecriture de quelques valeurs dans la table de couleurs permet de changer instantanément à l'écran la couleur de tous les pixels pointant sur les cases correspondantes.

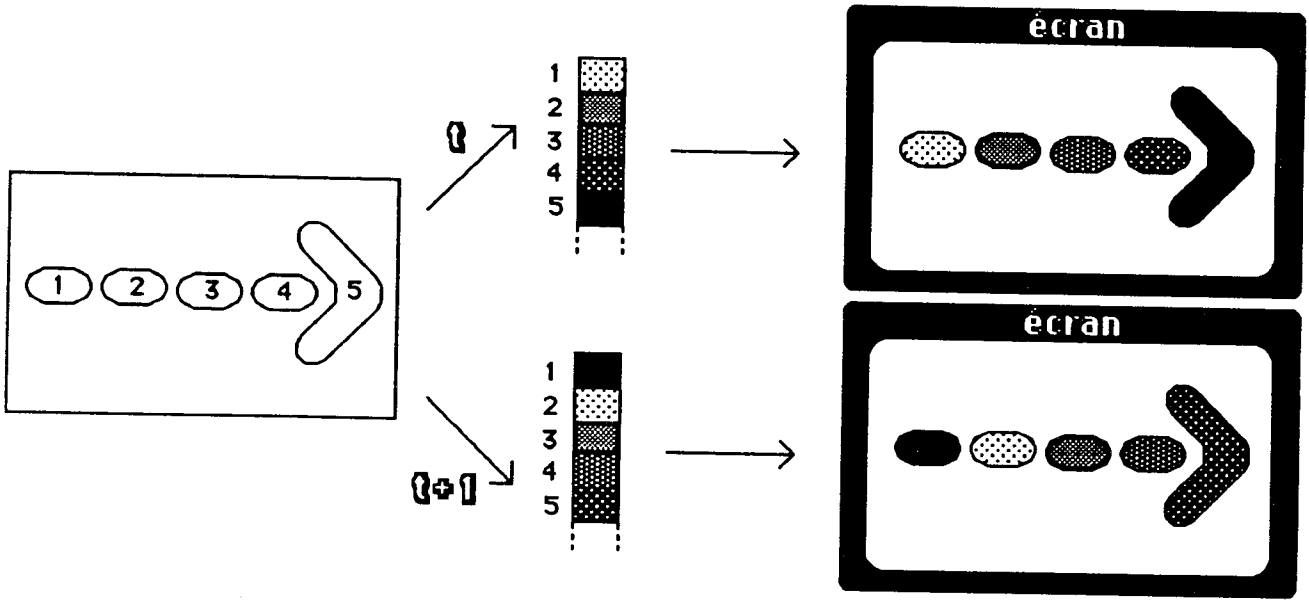


Indirection sur une table de couleurs.

Les animations par table de couleurs peuvent être réalisées en temps réel, puisque les écritures mémoire peuvent être aisément effectuées pendant chaque retour de trame de l'animation.

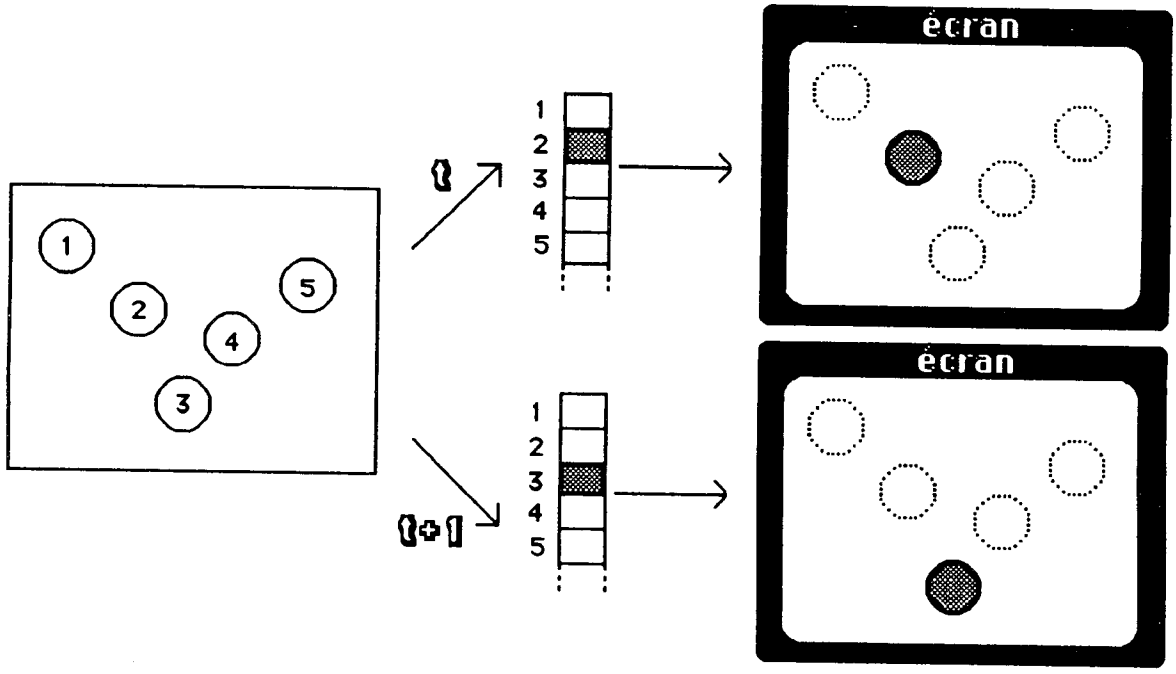
Mais si le procédé de visualisation est simple à mettre en oeuvre, la génération des images est loin d'être évidente. Ces images doivent être construites en fonction de l'animation et demandent des outils performants. On peut distinguer 3 types d'animation par table de couleurs :

- la permutation cyclique: elle permet de créer le mouvement de fluides. Ce style d'animation est réalisé par décalages successifs de la table des couleurs. Les pixels de l'image prennent successivement les mêmes couleurs.



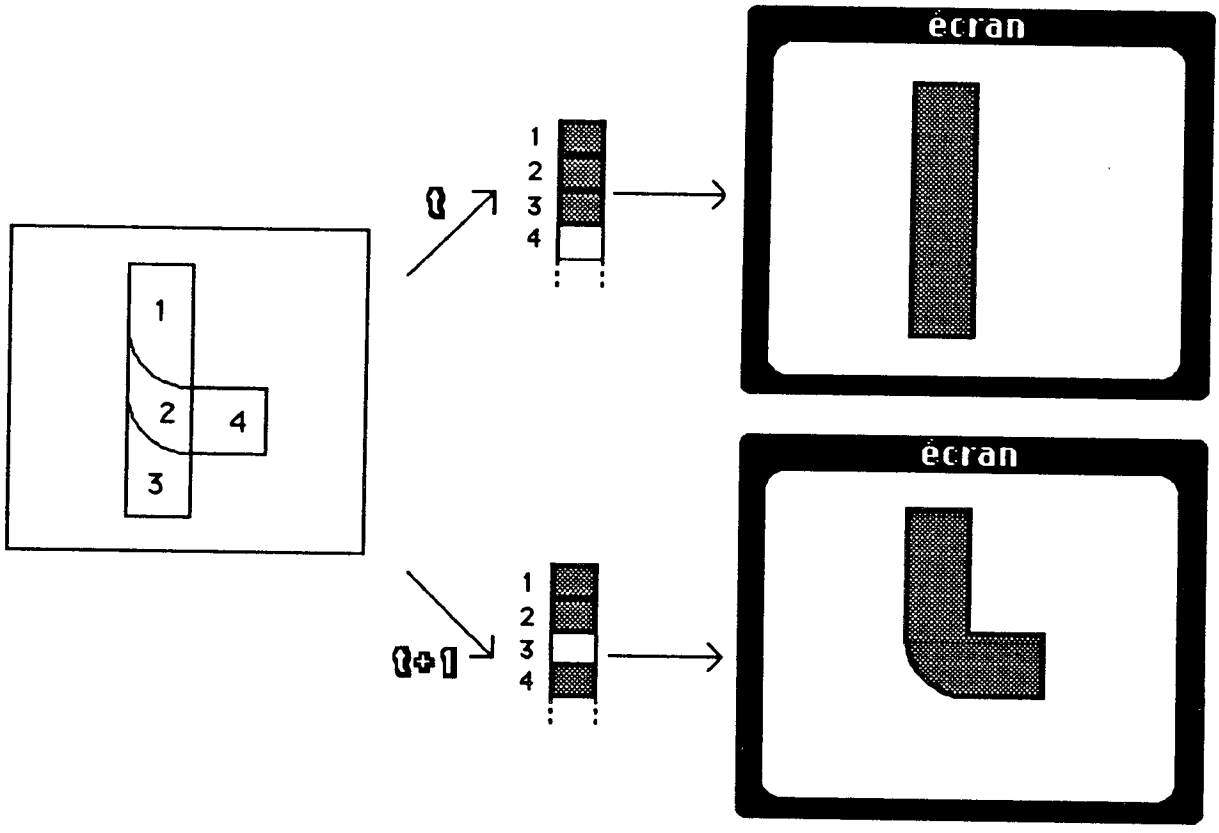
Permutation cyclique.

- l'alternance de couleurs: plusieurs positions d'un même objet sont présentes dans l'image, chaque position ayant un index de couleur différent. Ces positions doivent être disjointes. Au départ, la couleur de fond est associée à tous les index. L'animation va consister à associer une couleur à l'index correspondant à la position courante et la couleur de fond à tous les autres index. Le principal inconvénient de cette méthode réside dans la limitation du nombre d'objets manipulables ainsi que de leur complexité.



Alternance de couleurs.

- la généralisation: les 2 méthodes d'animation présentées ci-dessus ne permettent pas de générer des mouvements généraux ou des déformations complexes. La prise en compte de telles animations nécessitent de combiner plusieurs positions successives d'un objet dans l'image, en gérant éventuellement les intersections entre ces positions. La même zone peut donc être visualisée à des instants différents de l'animation et avec des couleurs différentes. L'animation consiste donc à associer à tout instant des couleurs à chacune des zones de l'image, la seule limitation résidant non plus dans le nombre d'objets mais uniquement dans le nombre de couleurs affichables simultanément.

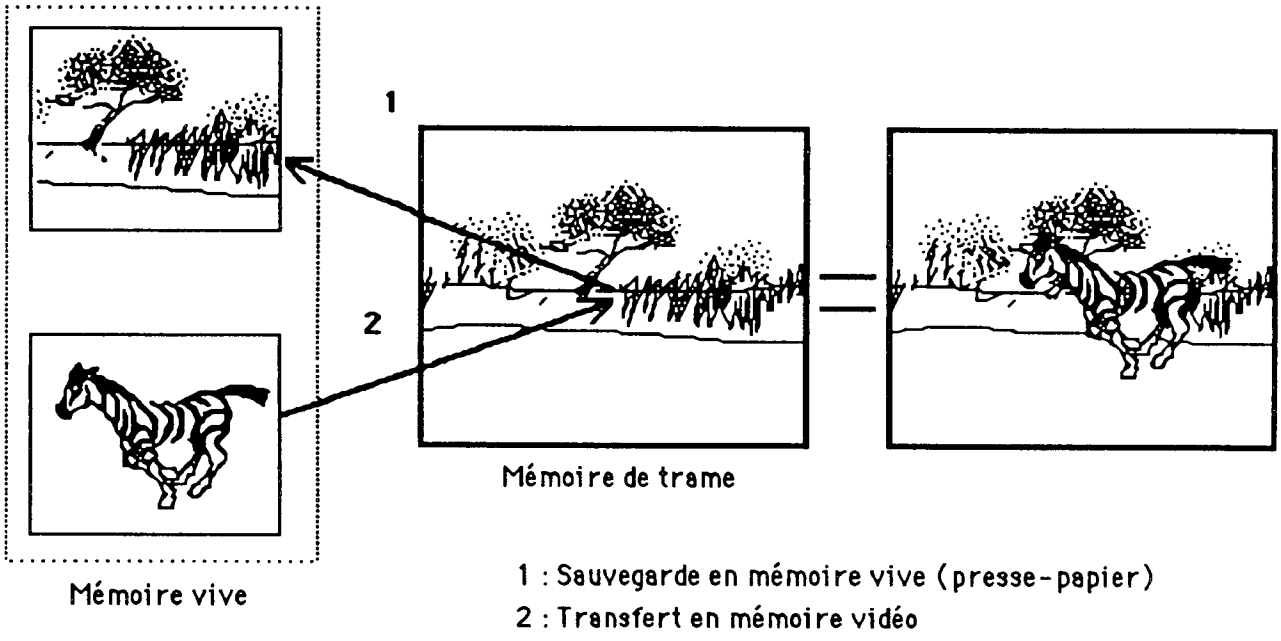


Généralisation.

La technique d'animation par table de couleurs exploite les possibilités temps réel du matériel de visualisation. Son point faible réside essentiellement dans la corrélation étroite entre l'image à animer et l'animation proprement dite. La mise au point de cette animation ne peut en effet jouer que sur le contenu de la table de couleurs ou sur les temps d'animation mais pas sur le contenu de l'image animée.

2.1.3 L'animation par recopie de blocs

La réalisation d'une animation est basée, dans ce cas, sur le principe de recopie de portions d'images rectangulaires, appelées blocs. Toutes les vues composant un objet sont précalculées et stockées en mémoire vive. L'animation d'un objet sur un décor, par exemple, est réalisée par des copies successives de l'objet en différents lieux du décor (transfert mémoire vive vers la mémoire de trame). La portion d'image située sous l'objet est sauvegardée avant la copie de l'objet (transfert mémoire de trame vers mémoire vive), et l'effacement de l'objet est réalisé automatiquement par restitution de la portion d'image altérée.



Animation par transfert de blocs.

Il apparait clairement que le temps de transfert d'un bloc est proportionnel à la taille de ce bloc, ce qui constitue la principale faiblesse d'un tel système. Une vitesse de transfert équivalente à la fréquence du balayage vidéo (13,5 millions de points par seconde) permettrait de ne transférer qu'un demi-écran (en tenant compte de l'effacement préalable) pendant la durée d'une trame complète (20 ms). Malgré l'existence de processeurs spécialisés opérant à de telles vitesses, on est encore loin des performances "temps réel" requises pour la modification de la mémoire d'image durant le retour de trame (2 ms). Il est donc nécessaire de coupler deux mémoires de trame visualisées alternativement. Pendant la visualisation d'une mémoire, une nouvelle image est générée dans la seconde mémoire, qui est elle-même visualisée à l'image suivante, pendant que la première est modifiée, et ainsi de suite. Cette technique est appelée flip-flop, pour exprimer le basculement systématique d'une mémoire à une autre.

Si, par ce biais, l'animation peut s'effectuer en temps réel, la génération des images à animer est très pénalisante. Toutes les vues des objets animés doivent être calculées et stockées en mémoire vive, ce qui pose un sérieux problème de capacité de stockage. Ce problème peut être contourné en ne stockant que les portions de mémoire de trame qui changent d'une image à une autre. Ceci nécessite une phase de compilation de toutes les images de l'animation [Den 86]. Le principal inconvénient d'une telle méthode, hormis le temps d'analyse, réside dans le fait que la moindre modification de l'animation entraine irrémédiablement une nouvelle compilation et une nouvelle génération des blocs.

On voit donc apparaitre des systèmes d'animation par blocs couplés aves des processeurs très rapides spécialisés dans le traitement d'images. Ces processeurs préparent des blocs, calculés à partir des images stockées en mémoire vive, et copiés dans la mémoire par le module de transfert rapide. Ils réalisent notamment

des rotations, des zooms ou encore des mises en perspective. Ainsi seules les vues caractéristiques des objets doivent être générées et stockées en mémoire, et la mise au point de l'animation peut être réalisée interactivement.

L'animation par blocs est la seule technique d'animation 2D évoquée dans ce chapitre permettant de visualiser une animation indépendamment des images animées. Contrairement aux autres systèmes, l'animateur peut décrire interactivement une animation, après la phase de génération des images à animer. Le système exploite en effet des possibilités matérielles temps réel indépendantes du contenu des images.

2.2 L'animation modélisée

Les systèmes d'animation 3D ont pour vocation la création de scènes tridimensionelles plus ou moins réalistes et peuvent être classés dans la catégorie des systèmes d'animation modélisée. L'ordinateur fournit une aide indispensable pour la création de l'univers 3D. Il serait, en effet difficile voire même impossible de dessiner avec précision toutes les images d'un objet 3D en rotation autour d'un point donné de l'espace 3D. L'ordinateur intervient à 3 stades particuliers dans la création d'une animation:

- la modélisation des objets, qui consiste à décrire et construire des objets 3D, à partir de méthode de description telles que rotation d'une courbe 2D autour d'un axe, ou propagation d'une courbe 2D, fermée ou non, le long d'une autre courbe, ou bien par déformations d'un objet de base [Bar 84] et lissage par des splines [HCF 83], ou bien encore à partir d'une bibliothèque de programmation par opérations élémentaires sur des objets de base [Req 80]. L'assemblage des différents objets de base 3D, par fusion des axes de rotation par exemple, permet ensuite d'engendrer des objets 3D complexes.

- la spécification du mouvement et la synchronisation des objets. Les mouvements 3D, plus complexes que les mouvements plans, regroupent non seulement les rotations autour d'un axe ou translations le long d'un axe, mais aussi les transformations ou déformations d'objets (forme ou couleurs), et également les mouvements de la caméra.

- le rendu des images 3D. L'application d'algorithmes de parties cachées [Bou 80], d'ombres portées, de lancer de rayons [Ama 87a] [BWM 87], application de texture, la prise en compte de plusieurs sources lumineuses [Cur 87] et de modèles de réflexion et de transparence, la création de flou temporel et d'antialiasing [KBa 83] comptent parmi les quelques facteurs nécessaires à la génération d'images réalistes [Ama 87b].

Nous allons décrire, dans la fin de ce chapitre, quelques méthodes de réalisation d'animations 3D par ordinateur.

2.2.1 Les langages de programmation

A l'origine de l'animation par ordinateur, les systèmes étaient tous basés sur un langage de programmation classique et permettaient la description d'objets 3D structurés et de contraintes mécaniques entre ces objets. Mais les utilisateurs de ces systèmes devaient alors être plus informaticiens qu'artistes, ce qui explique le désinteressement initial des animateurs envers ces techniques.

Dans un tel système, une animation est décrite par un programme en langage évolué (Fortran, Pascal, C, ADA, LISP...), dans lequel ont été implémentés des types de données particuliers (polygône, vecteur, matrice, couleur ...), ainsi que des procédures de traitement d'animation (lois de déplacement, collision...). Le langage d'animation peut ainsi être vu comme une extension d'un langage de programmation constituant son noyau et peu de développements sont donc nécessaires au niveau de l'environnement du langage.

L'animateur réalise un programme dans lequel il commence par définir les types de données complexes manipulés dans l'animation (par exemple, un tétraèdre, déterminé à partir de 4 sommets formels définissant 4 faces triangulaires, ou encore toute autre forme composée d'objets de types prédéfinis et paramétrés dans la définition du type). Il déclare ensuite des variables appartenant aux classes définies et jouant le rôle d'acteurs de l'animation (par exemple, plusieurs tétraèdres peuvent être déclarés comme des instances du type préalablement défini). Il définit ensuite les mouvements des acteurs comme des procédures réalisant les calculs de la position ou des angles de rotation pour chaque trame de l'animation (un tétraèdre peut se déplacer le long d'une spirale, tout en tournant sur lui-même de plus en plus rapidement). Les mouvements de la caméra, représentée par le point de l'observateur et le point de visée, sont définis de la même manière. Le programme principal, quant à lui, réalise la mise en scène des acteurs de l'animation; il initialise les paramètres des acteurs et de la prise de vue et régit le rythme global de l'animation. Mettre au point une animation revient donc à mettre au point le programme informatique correspondant (paramètres des acteurs, lois de déplacement...).

La programmation, orientée objet, d'animations 3D permet la réalisation de films réalistes mettant en scène de nombreux acteurs, évoluant de façon complexe, aussi bien du point de vue de leur forme intrinsèque que de leur cinématique. L'utilisation, par exemple, des propriétés des Béta-Splines [BBe 83] permet de déformer dans le temps de façon continue avec plus ou moins de rigidité un acteur défini à partir de telles courbes.

Citons, à titre d'exemple, le système ASAS [Rey 82] et implémenté dans un environnement LISP, ou encore le système MIRA [MTh 83] [MTh 85b], extension du langage Pascal.

Utilisé comme tel, un tel système demande toujours une étroite collaboration entre l'informaticien et l'artiste, pour la programmation des acteurs et des

mouvements de l'animation. Le système est ouvert et l'informaticien peut facilement ajouter une nouvelle fonction au système pour répondre à la demande de l'artiste.

De plus, afin de réaliser des animations réalistes, l'animateur doit pouvoir se représenter exactement l'espace 3D dans lequel évoluent les acteurs et la caméra. Le passage de l'animation intuitive à l'animation programmée constitue là encore une barrière difficile à franchir, surtout lorsqu'il n'y a aucun contrôle visuel. Les systèmes basés sur un langage de programmation classique tendent donc de plus en plus à devenir le noyau de systèmes interactifs accessibles directement à l'animateur.

2.2.2 Les éditeurs de scripts

Les éditeurs de scripts ont été developpés afin de fournir une méthode d'animation plus flexible et plus adaptée que la programmation d'un langage classique. Ils se présentent sous la forme de langages de programmation spécialisés de haut niveau, incluant toutes les possibilités d'un langage évolué (notions de procédures, structuration d'objets). Leur spécification nécessite le développement d'un environnement d'animation incluant:

- la définition d'une syntaxe,
- un éditeur de texte adapté à cette syntaxe,
- un compilateur de fichier d'animation en langage "objet", le plus proche possible des caractéristiques matérielles du système,
- un interpréteur de fichier en langage "objet" pour la visualisation de l'animation proprement dite.

Citons, par exemple, l'éditeur de scripts, SCRIPT, développé par Edwards [Edw 87].

La programmation est plus adaptée que pour un langage classique et nécessite moins de compétences informatiques. De plus les fichiers manipulés sont de petite taille, puisqu'ils sont adaptés au système d'animation.

En revanche, l'ajout d'une nouvelle fonction au système nécessite en général l'extension ou la révision de la syntaxe initiale et du même coup de l'environnement d'animation. Ces systèmes sont donc plutôt des systèmes fermés.

2.2.3 Les systèmes interactifs

Les systèmes interactifs sont des applications offrant de nombreux outils de dialogue (menus, souris, tablette...).

Ils permettent la description interactive d'objets 3D, à partir par exemple d'une courbe 2D en rotation autour d'un axe, ou bien encore par élévation en prisme d'une courbe 2D. La forme 3D ainsi créée peut être ensuite tordue, cisaillée, vrillée ou coudée à volonté. Toutes ces manipulations interactives des

objets dans l'espace peuvent être réalisées par exemple en utilisant les vues de l'espace projeté suivant les 3 axes du repère cannonique.

Le choix interactif dans une palette permet ensuite d'associer une couleur à un objet 3D ou à un sous-ensemble de facettes. De même, une image créée par exemple par un logiciel de palette peut aisément définir une texture appliquée automatiquement à l'objet.

Le comportement des objets ainsi créés est décrit explicitement par l'animateur, leurs mouvements 3D (position, angles de rotation) sont définis à l'aide de positions-clés à des instants précis de l'animation. De plus, à tout instant, l'animateur peut modifier la forme 3D d'un objet, afin de générer des anamorphoses.

Plusieurs sources lumineuses peuvent être introduites. Leur mouvement est décrit de la même manière que pour les objets 3D (position, direction). Enfin, la caméra peut être déplacée dans l'espace pour la réalisation de la prise de vue.

Le système est ensuite capable d'interpoler pour chaque image de l'animation toutes les informations décrites, puis de visualiser, en dessin au trait, les objets 3D dans l'attitude correspondante (line-test). Lorsque l'animation filaire paraît satisfaisante, l'animateur peut enfin lancer les calculs de rendu pour l'enregistrement de l'animation réaliste.

Pour de tels systèmes, le résultat visuel prime sur les lois mathématiques. Plus le système est ergonomique et simple à utiliser, plus le noyau du système est perfectionné.

Citons, par exemple, le système MUTAN [FLT 86], qui constitue la partie interactive, dédiée aux animateurs, des systèmes MIRA évoqués ci-dessus.

2.2.4 Les simulateurs

La principale particularité des simulateurs réside dans leurs calculs 3D en temps réel. Les animations sont moins réalistes que pour tous les systèmes cités ci-dessus. La complexité des objets est en effet réduite à quelques centaines de facettes et les modèles des objets sont beaucoup moins poussés. Les objets manipulés sont décrits dans des bases de données adaptées (sommets et couleur de chaque face) et les processeurs utilisés sont spécialisés dans l'affichage de polygônes.

L'augmentation constante de la puissance de calcul des ordinateurs, notamment avec l'emploi de transputers ou de multiprocesseurs, afin de paralléliser au maximum le traitement des images, tend à réduire de plus en plus l'écart entre simulateurs et systèmes réalistes.

2.2.5 Les systèmes orientés par les buts

L'intelligence artificielle est apparue au début de années 70, pour permettre, entre autres, la manipulation de blocs par des robots. Un double but motivait cette technique dans ce cadre précis: la sélection des blocs à transporter et l'exécution des mouvements du robot permettant leur déplacement.

Des recherches dans le domaine de l'animation utilisant ces idées ont été également menées [Zel 82] [Zel 86]. Ces méthodes, proposées par Zelter, consiste à pourvoir les objets à animer de connaissances, aussi bien sur leur environnement que sur leur propre constitution. L'acquisition de ces informations leur permet d'agir de leur propre chef lorsqu'ils se trouvent confrontés à des situations particulières. C'est le cas du système SAS (Skeleton Animation System) du même auteur, qui permet, entre autres, à un squelette humain de marcher sur un terrain accidenté ou d'éviter des obstacles.

2.2.6 Les systèmes spécifiques

Bien que ces systèmes couvrent un très large éventail d'applications, on peut quand même en distinguer 2 classes principales :

- les systèmes de modélisation de phénomènes, naturels ou scientifiques. La modélisation des vagues [Fou 86] [Pea 86] [Mal 87] en constitue par exemple une application. D'autres s'intéressent au comportement de groupe [Rey 87], ou bien encore à des phénomènes physiques [Gla 87].

- les systèmes de modélisation du corps humain. Certains travaux se portent sur la visualisation des expressions du visage [Pla 81] [Wat 87]. Les muscles du visage sont modélisés à l'aide d'un maillage; à chacun des noeuds sont associées des propriétés, telles que l'élasticité, ainsi que des contraintes telles que l'attachement à la peau. D'autres se portent sur la modélisation et l'animation du corps humain [MTh 88]. Une arborescence du corps est réalisée, reflétant précisément son squelette, et chaque partie du corps est modélisée séparément, à partir de clichées réels ou de techniques spécifiques. L'animation est essentiellement basée sur les techniques d'interpolation [GMa 85].

Conclusion

Les systèmes de production d'animations par ordinateur sont très nombreux et très variés. Nous n'en avons en fait présenté qu'une partie représentative. On peut, en outre, évoquer l'apparition de nouvelles techniques vidéo couplées aux systèmes de synthèse d'images, et qui améliorent sensiblement la condition des animateurs. Citons notamment les vidéo-disques, les disques durs numériques ou

les régies numériques, qui autorisent des actions temps réel sur les images. Ainsi, il devient de plus en plus difficile de faire la part des choses et de cerner qui fait quoi dans les systèmes complexes d'animation. Aussi la spécification d'un nouveau système d'animation doit-elle impérativement tenir compte de ces nouvelles techniques dans les différentes phases de réalisation d'un film d'animation.

Comme on a pu le voir dans ce chapitre, la plupart des systèmes lient totalement l'animation à la génération des images, et dans la majorité des cas, cette génération ne peut être réalisée en temps réel. C'est pourquoi la plupart des systèmes d'animation sont des systèmes image par image.

Notre approche va donc consister à séparer les images à animer de leur animation propre. Les seules propriétés temps réel accessibles pour un système de gamme moyenne sont des fonctions 2D matérielles utilisées pour la manipulation d'images. Nous allons donc, dans la mesure du possible, ramener les animations au niveau de l'animation d'images 2D et exploiter des fonctionnalités matérielles temps réel pour la réalisation d'animations 2D et 3D. Les chapitres 2 et 3 font état des concepts manipulés au niveau des images 2D. Le chapitre 4, quant à lui, révèle les fonctionnalités requises par un tel système d'animation.

CHAPITRE 2 : LA REALISATION D'UN FILM D'ANIMATION

Introduction

Les différents systèmes présentés dans le chapitre précédent ont tous des caractéristiques différentes, mais un but commun : la réalisation de films d'animation les plus attractifs possible. Cette réalisation nécessite souvent de gros moyens : temps importants de création et de mise au point.

Après avoir énoncé les différentes étapes intervenant dans la réalisation d'un film d'animation, nous essaierons d'en extraire des concepts généraux et nous aboutirons à la proposition d'un système original, basé sur ces concepts.

1. LA REALISATION D'UN FILM D'ANIMATION

On peut distinguer quatre phases principales dans la réalisation d'un film d'animation :

- la création des images,
- la visualisation des images,
- l'enregistrement des images,
- la postproduction.

Nous allons reprendre une à une ces différentes étapes, en relevant les limites des principaux systèmes présentés au premier chapitre et en déterminant les points critiques sur lesquels les travaux doivent dorénavant porter afin de faciliter la tâche de l'artiste et de minimiser les coûts de production d'un film d'animation.

1.1 La création des images

Il existe globalement 4 méthodes de génération d'une image :

- la création artistique, à l'aide d'une palette électronique offrant de nombreuses fonctionnalités telles que le tracé de figures géométriques ou de courbes mathématiques, le dessin au pinceau ou à l'aérographe, la génération de dégradé de couleurs. Afin d'éviter les sauts de couleur perceptibles à l'oeil humain, les nuances doivent être nombreuses. Un codage sur 24 bits (8 bits par primaire, soit 256 niveaux pour le rouge, le vert et le bleu) permet la génération d'images "réalistes" avec 16,7 millions de couleurs différentes.

- l'acquisition vidéo, où l'image provient d'une source vidéo externe au système, une caméra, un magnétoscope ou encore un vidéo-disque. Cette source vidéo peut être utilisée en direct ou bien gelée. Dans le premier cas, le lecteur vidéo est en "play" (défilement au rythme vidéo) pendant que le système visualise

l'animation. L'image vidéo peut, par exemple, jouer le rôle de fond de l'animation. Dans le second cas, l'image en provenance de la source vidéo est recopiée instantanément (en 1 ou 2 trames) dans une mémoire vidéo du système. L'image ainsi "gelée" peut alors être éventuellement retouchée dans la palette électronique. La source vidéo externe peut fournir des informations analogiques ou numériques. Dans le cas d'informations analogiques, une conversion des signaux de la source en signaux numériques internes est nécessaire; la qualité des images est nécessairement altérée. En revanche avec des signaux numériques, l'image est visualisée sans aucune dégradation, ce qui autorise des générations multiples (visualisation, enregistrement puis acquisistion...).

- l'utilisation de primitives d'une bibliothèque graphique de haut niveau à travers un programme, réalisant par exemple l'interface avec une base de données scientifique pour la visualisation d'un phénomène physique.

- l'utilisation d'un modeleur d'objets tridimensionnels et d'un module de rendu réaliste. Afin d'améliorer la qualité des images de nombreux développements ont été menés sur les techniques sophistiquées de rendu, de lancer de rayons, ou encore d'application de texture.

Très peu de systèmes 2D sont susceptibles de générer les images d'une animation en temps réel. En fait, seuls des processeurs spécialisés dans l'affichage de polygônes permettent de générer en temps réel les images successives réalisant une animation. Les autres systèmes d'animation en temps réel correspondent à la visualisation d'images à partir d'éléments déjà générés plutôt qu'à la génération proprement dite de ces images. Dans le cas d'animation par table de couleurs, la génération de l'image n'est pas réalisée en temps réel, seule la visualisation est effectuée en temps réel après les écritures dans la table de couleurs. De même, dans le cas de la recopie de blocs, les portions d'images sont générées à l'avance; seul leur transfert a lieu en temps réel.

Quant au 3D, la génération d'une image réaliste pouvant demander plusieurs heures de calculs à l'unité centrale de l'ordinateur, il est nécessaire de stocker une à une les images dans une mémoire, qui peut être ou non un support vidéo. La sauvegarde sur disque dur s'effectue par l'intermédiaire d'un fichier, qui contient toutes les informations de couleurs de l'image, compactées ou non. Tout fichier image peut être interprété afin de visualiser à nouveau l'image. La sauvegarde d'une image sur disque dur vidéo est, quant à elle, totalement gérée par le matériel vidéo, et s'effectue en temps réel.

1.2 La visualisation des images

Les images de l'animation peuvent être visualisées après leur génération. Comme on vient de le voir, les temps de calcul des images ne permettent pas à l'animateur de juger de son animation lors de la génération des images.

Les images sont donc visualisées après avoir été toutes sauvegardées une à une. Cette visualisation constitue un **play-back** de l'animation, c'est-à-dire une

animation différée (nous y reviendrons lors de l'étude matérielle, au chapitre 4). L'animateur peut alors visualiser les unes après les autres les images de l'animation, par restitution des fichiers images stockés sur disque dur ou utilisation d'une carte d'acquisition vidéo couplée à un matériel vidéo. Dans le premier cas, la restitution des images ne peut pas s'effectuer en temps réel, et l'animateur doit accepter des temps de chargement non négligeables lors du défilement des images. En revanche, l'utilisation d'un disque dur vidéo, par exemple, permet de visualiser l'animation en temps réel. On parle alors de **play-back temps réel** (cf chapitre 4).

Il est essentiel que l'animateur visualise en temps réel son animation, avant son enregistrement final. Les temps de génération des images étant en général très supérieurs à la durée de visualisation d'une image, la technique de play-back temps réel s'avère être un atout important pour la réalisation d'une animation.

1.3 L'enregistrement des images

On entend par là enregistrement final de la séquence d'animation, et non pas stockage intermédiaire dû à des temps de génération trop importants.

L'image visualisée doit être enregistrée sur support vidéo. Différents appareils sont utilisés par les professionnels :

- les magnétoscopes, de type U-Matic ou BVU (signal analogique composite), de type Bétacam et 1 pouce (signaux analogiques en composantes), ou encore de type numérique (standard 4.2.2). Le principal inconvénient de ces matériels est leur temps d'accès à une image en lecture et les temps d'enregistrement non négligeables. En effet, l'accès aux images d'une bande vidéo est séquentiel. Le déplacement d'une image à une autre s'effectue nécessairement par défilement de la bande; le temps d'accès à une image est donc directement lié à sa situation sur la bande et à la vitesse de recherche de l'appareil. En outre, lors d'un enregistrement, il est nécessaire de se positionner plusieurs secondes avant le début de l'enregistrement (au minimum 5 secondes), de laisser défiler le magnétoscope en lecture afin de bien le synchroniser sur les signaux vidéo, puis de commuter en enregistrement lors du passage sur l'image. Cette manipulation est appelée **préroll**.
- les disques durs analogiques ou numériques, qui présentent l'avantage par rapport aux magnétoscopes de temps d'accès très rapides à une image. En revanche, leur capacité est encore réduite (de l'ordre d'une minute d'enregistrement).

Tous ces matériels sont directement pilotables par les systèmes d'animation par le biais d'une liaison série, à l'aide d'un protocole propre à chaque constructeur.

Le type de l'enregistrement d'une animation est directement lié à la technique de visualisation de cette animation. L'enregistrement s'effectue image par image dans les cas où la génération d'une image ne peut être réalisée pendant le retour de

trame. Pour chacune des images, il est alors nécessaire d'effectuer un préroll, d'enregistrer cette image, puis de faire progresser d'une unité le point courant d'enregistrement. Cette opération répétée conduit au **montage** du film d'animation, chaque image nécessitant un **point de montage**, c'est-à-dire un point d'arrêt et de reprise de l'enregistrement. En revanche, l'enregistrement est réalisé en continu lorsque les images du film peuvent être visualisées en temps réel. C'est le cas par exemple lors d'un transfert entre un disque dur temps réel et un magnétoscope via une carte d'acquisition vidéo.

Si le temps de génération d'une image est directement lié à l'architecture du calculateur (processeur rapide, transputers...) et à la complexité des algorithmes utilisés, les temps de montage sont eux liés aux contraintes propres aux appareils d'enregistrement et à l'architecture du système de visualisation des images. En considérant, par exemple, un temps de préroll de 5 secondes et un temps de recherche d'image de 1 seconde, le montage d'une seconde de film d'animation nécessite 2 minutes et demie, sans compter bien sûr les temps de génération des images.

L'enregistrement en continu apparait comme un facteur essentiel pour minimiser les coûts de production des animations.

1.4 La postproduction

La postproduction est la dernière étape dans la réalisation d'un film d'animation. Il s'agit d'effectuer le montage des différentes séquences en intégrant éventuellement des effets de régie tels que **fondu enchainé** ou **fondu au noir** entre les séquences; ces fondus réalisent la disparition progressive d'une séquence et l'apparition d'une autre séquence, soit simultanément (fondu enchainé), soit par passage intermédiaire à un fond noir (fondu au noir). Il s'agit également de synchroniser la bande sonore avec les images du film.

Cette étape est souvent réalisée manuellement à l'aide de régies ou de tables de montage. On commence seulement à voir des systèmes informatiques spécialisés dans le montage de film d'animation. Ces systèmes pilotent totalement plusieurs sources vidéo, enchainent automatiquement les séquences, synchronisent la bande sonore et enregistrent le montage sur support vidéo voire même sur film [Kun 89].

La phase de montage pose, en outre, le délicat problème de génération des bandes vidéo. Afin de ne pas dégrader la qualité des images, les séquences doivent être enregistrées sur du matériel très coûteux, l'idéal étant un matériel numérique avec interface 4.2.2.

L'intégration de fonctions de montage dans le même environnement graphique que la création des images et la réalisation des séquences, améliore sensiblement la convivialité de l'ensemble et contribue sérieusement à la diminution des coûts de production.

1.5 Les conséquences

Les contraintes évoquées pour les différentes étapes de la réalisation d'un film d'animation nous conduisent à la constatation suivante : plus on avance dans la réalisation d'un film d'animation, plus on est susceptible de tout reprendre depuis le début. L'enregistrement d'un mauvais mouvement nécessite une nouvelle génération d'images et un nouvel enregistrement de la séquence, mais un problème d'ajustement de temps au niveau du film global peut impliquer la modification d'une ou plusieurs séquences et donc à nouveau une génération des images constituant ces séquences et leur enregistrement image par image.

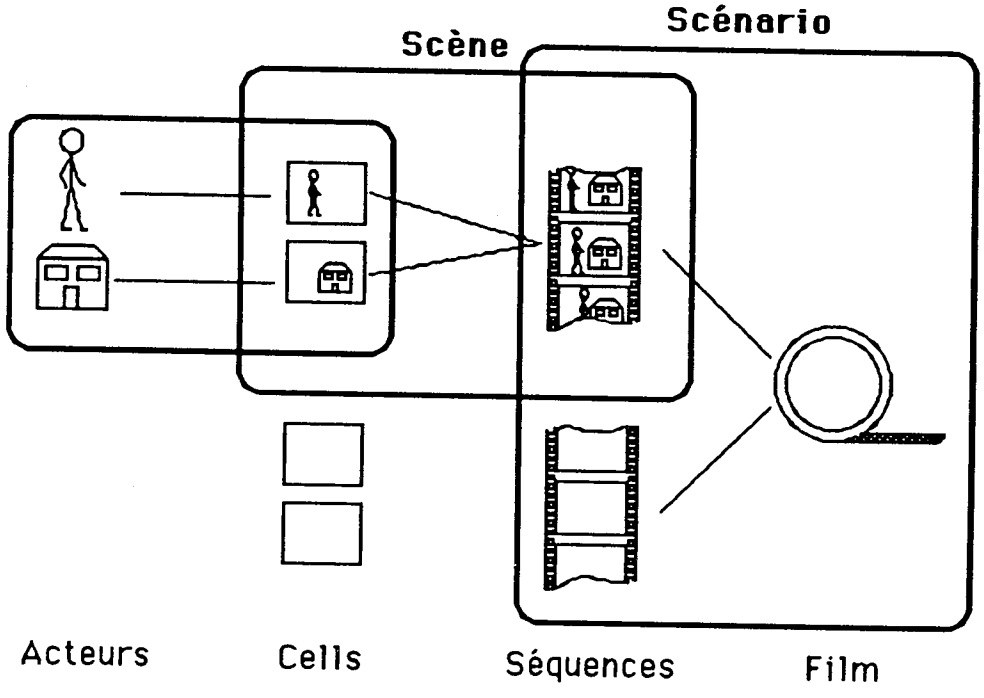
Il apparait donc indispensable de retarder le plus possible l'enregistrement du film d'animation, et de se donner les moyens d'intervenir facilement à tous les niveaux de la réalisation du film sans remettre en cause les niveaux inférieurs.

2. FORMALISATION

2.1 Présentation

L'étude des différents systèmes d'animation et les étapes de production d'un film nous conduisent à extraire des concepts généraux et à formaliser trois différentes couches dans la réalisation d'un film d'animation, présentées dans les paragraphes suivants :

- l'animation d'acteurs et la génération de cells,
- la description de scène et la génération de séquence,
- la description de scénario et la génération de film.



Les 3 phases de réalisation d'un film d'animation.

L'intégration des nouveaux concepts propres à ces 3 couches dans un système informatique contribue à l'optimisation des méthodes de travail des animateurs ainsi que des temps de production des films d'animation.

2.2 L'animation d'acteurs et la génération de cells

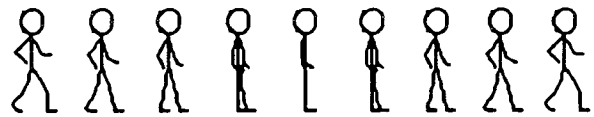
2.2.1 La notion d'acteur

Un **acteur** peut être défini [Mar 77] comme une entité destinée à s'animer, dans le sens théatral du terme, et susceptible à ce titre :

- de jouer un rôle imposé par l'animateur,
- de possèder sa propre animation,
- de réagir aux sollicitations du monde extérieur,
- de conserver certaines propriétés inhérentes à sa constitution.

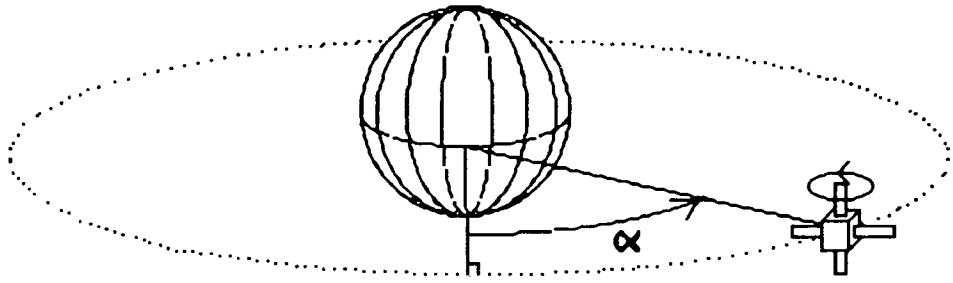
Un acteur 2D peut être représenté sous la forme de dessins. Ces dessins peuvent être structurés, c'est-à-dire représentés sous la forme de polygônes auxquels sont associées des informations de couleur, ou non structurés, c'est-à-dire représentés par un ensemble de points de couleur, issus d'un fichier image par exemple. L'enchainement des différents dessins permet dans la majorité des cas de traduire l'animation d'un acteur 2D. Mais cet enchainement peut découler de la constitution de l'acteur ou être une réaction à une sollicitation extérieure.

C'est le cas, par exemple, de "l'homme sachant marcher". Le seul fait de le déplacer provoque automatiquement sa marche à la vitesse correspondante au déplacement.



L'homme sachant marcher.

Un acteur 3D, quant à lui, peut être représenté par un ensemble de facettes ou bien encore par un ensemble de sommets et de liens entre sommets, auquel sont ajoutées des informations, couleur intrinsèque, modèle de réflexion... Il peut également contenir son propre modèle de comportement. Par exemple, le satellite en rotation autour d'une planète; il peut être asservi à un déplacement sur une orbite circulaire de rayon donné, peut tourner sur lui-même indépendamment de sa position sur cette orbite et réagir à une donnée externe qui peut être l'angle de rotation autour de la planète.



Satellite en orbite.

La prise en compte, à un instant donné, des informations relatives à tous les acteurs d'une animation permet la génération d'une **image**, chargée dans la mémoire de trame du système, et ainsi visualisée à l'écran.

2.2.2 La notion de cell

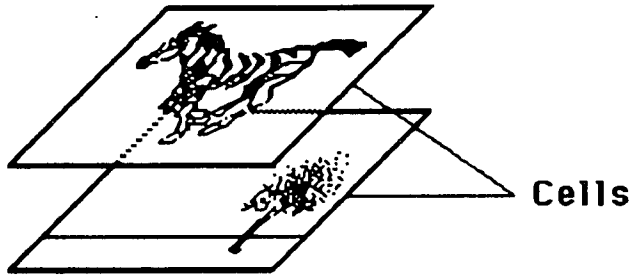
Si la description de l'animation d'un acteur n'est pas indépendante de l'animation de ses semblables (réaction d'un acteur à la sollicitation d'un autre, synchronisation), la visualisation d'un acteur peut, dans la majorité des cas, être rendue indépendante de la visualisation des autres acteurs.

C'est toujours le cas en animation 2D. La visualisation de chaque dessin peut être effectuée indépendamment des autres dessins de l'animation. En revanche, en 3D, ceci n'est exact que si l'on se limite dans le réalisme de l'image. En effet, un calcul 3D réaliste doit tenir compte de tous les acteurs 3D de la scène pour générer une image réaliste avec des ombres portées, des effets de transparence, ou encore de réflexion. Toutefois, les calculs de rendu, appliqués à chaque acteur séparément donnent souvent des résultats satisfaisants.

La visualisation à l'écran d'une animation peut donc être considérée comme la superposition de différentes couches logiques, appelées **cells**. Cette notion est directement inspirée des techniques d'animation traditionnelle, manipulant différentes couches de celluloïdes.

Un cell est la représentation plane d'un acteur 2D ou 3D à un instant donné de l'animation. Ainsi la projection d'un acteur 3D dans le plan de l'écran pour chaque 25ème de seconde du film détermine un ensemble de cells, qui, visualisés les uns à la suite des autres, représentent l'animation de l'acteur. Il en est de même pour un acteur 2D, dont l'ensemble des dessins (de base ou calculés par interpolation à chaque 25ème de seconde) constitue un ensemble de cells représentatifs de l'animation de l'acteur.

Un cell peut être stocké, en mémoire vive, sur fichier ou encore en mémoire vidéo. La combinaison logicielle ou matérielle des différents cells et la prise en compte des priorités associées à chaque cell permet de générer à chaque instant une **image** de l'animation.



Superposition de 2 cells composant une image.

Lorsque la décomposition d'une animation en plusieurs cells est impossible, comme c'est le cas dans les systèmes d'animation 3D réalistes existants, l'animation se réduit à une seule couche logique, donc à un seul cell à un instant donné. Chaque cell représente alors à lui-seul une image du film d'animation.

Plus généralement, un cell peut être le résultat de l'interprétation de l'animation d'un ou plusieurs acteurs, ce qui permet de traiter des cas délicats tels qu'interpénétration d'acteurs. Mais, la mise au point de l'animation est d'autant plus aisée que le nombre de cells constituant une image est important. La correction de l'animation d'un acteur ne nécessite alors que la génération des cells associés à cet acteur, et la recombinaison avec les autres cells qui, eux, peuvent être conservés.

La décomposition en cells permet de combiner différents styles d'animation. Un personnage de dessin animé peut facilement évoluer sur un décor 3D. En effet un cell, quelle que soit son origine, peut être assimilé à un dessin et peut donc, à ce titre, intervenir dans une animation.

Le cell constitue ainsi le point de raccord entre toutes les techniques d'animation par ordinateur existantes.

2.3 La description de scène et la génération de séquence

2.3.1 La notion de scène

La notion de **scène** est directement inspirée de celle utilisée dans le monde cinématographique. Elle est déterminée par trois facteurs fondamentaux :

- le cadre de l'action,
- l'ensemble des cells des acteurs,
- la mise en scène.

Le cadre de l'action évoque une représentation spatiale de la scène. Les cells étant des entités planes, l'ordre d'affichage de ces différents cells permet de traduire la notion de profondeur de la scène. L'arrière plan de la scène (le cell affiché en premier) peut matérialiser un décor, qui représente alors l'unité de lieu de l'action.

La mise en scène, quant à elle, concerne toutes les informations propres à l'animation, telles que les enchaînements temporels des différents cells des acteurs ou encore l'ordre de superposition des cells.

Dans le cas de l'aide à l'animation traditionnelle, la scène est strictement équivalente au storyboard de l'animateur traditionnel et comprend par exemple l'enchaînement des différents cells de la marche d'un personnage évoluant sur un décor.

Dans le cas de l'animation par table de couleurs, la scène d'animation est constituée de l'image à animer (1 seul cell) et des différents cycles de couleurs à appliquer à la table de couleurs.

Dans le cas d'un système d'animation 3D réaliste, la scène manipule un ensemble de cells représentant chacun une image du film. La mise en scène consiste alors à faire défiler successivement les unes après les autres toutes ces images.

2.3.2 La notion de séquence

La **séquence** peut être définie comme la représentation de la scène, dans le sens théatral du terme.

La composition des différents cells à un instant donné permet de construire une image de l'animation à cet instant. L'ensemble de ces images représente la séquence d'animation.

La séquence constitue souvent le point final de la contribution informatique à la réalisation d'une animation.

La séquence est en général enregistrée sur support vidéo. Lorsque la représentation de la scène peut avoir lieu en temps réel, l'enregistrement s'effectue en continu. C'est le cas par exemple de l'animation par table de couleurs. En revanche, lorsque la composition des différents cells de la scène ne peut avoir lieu en temps réel, l'enregistrement de la séquence s'effectue image par image et peut nécessiter un point de montage tous les 25ème de seconde. C'est le cas, par exemple, lorsque les cells doivent être chargés dans la mémoire de trame à partir d'un disque dur.

Le constat d'une imperfection dans la séquence nécessite dans tous les cas un nouvel enregistrement de la séquence, mais peut également obliger l'animateur à retoucher la scène, voire même à regénérer les cells associés à un acteur de l'animation. Il est donc essentiel de visualiser en temps réel une séquence d'animation et de donner des outils interactifs de description et de mise ou point de la scène.

2.4 La description de scénario et la génération de film

2.4.1 La notion de scénario

Le **scénario** manipule deux types d'informations :

- les séquences,
- la bande sonore.

Les séquences sont telles qu'on les a définies ci-dessus. Elles sont, en général stockées sur bandes vidéo ou disques durs numériques.

La bande sonore intervient à plusieurs niveaux : les voix intervenant dans les dialogues de l'animation, les bruitages et le fond musical.

Le scénario contient toutes les informations de montage utilisées pour le raccord des différentes séquences et leur synchronisation avec la bande sonore. Les enchainements de séquences peuvent être réalisés sans transition (en "cut"), par un fondu enchainé, par un fondu au noir ou encore par tout effet de régie permettant de passer d'une séquence à une autre.

2.4.2 La notion de film

Le **film** est le résultat de l'interprétation du scénario. La réalisation d'un film nécessite le pilotage synchrone de matériels vidéo (magnétoscope, vidéo-disques, disques durs numériques), en lecture (éventuellement au travers une régie d'effets) et en enregistrement.

La mauvaise intégration d'une séquence dans le film d'animation (durée inadaptée, rythme de l'animation incohérent...) nécessite la regénération de cette séquence, avec toutes les conséquences évoquées au paragraphe précédent. La prise en compte de cette nouvelle séquence aboutit généralement au montage complet du film d'animation.

3. PROPOSITION D'UN SYSTEME ORIGINAL

Les limites des différents systèmes d'animation évoqués dans les paragraphes précédents, nous conduisent à la conception d'un système original d'animation en temps réel, G_ANIM, visant à minimiser les manipulations des animateurs et à optimiser les temps de production de films d'animation.

Le but recherché par cette étude est l'informatisation complète de la chaine de production d'un film d'animation, depuis la génération des cells jusqu'au montage final du film. Sa contribution se limite volontairement aux couches 2 et 3 évoquées dans les paragraphes précédents, à savoir scène/séquence et scénario/film.

L'étude de ce système menée conjointement par la société Gétris Images et le laboratoire Artémis, dans le cadre d'une convention CIFRE, a conduit à la réalisation d'une architecture originale commercialisée par la société Gétris Images.

3.1 Le cadre du système

L'intérêt principal relevé jusqu'à présent dans la décomposition d'une animation en différents cells concerne l'indépendance de ces cells lors de leur génération. La modification de l'animation d'un acteur ne demande en effet que la regénération des cells qui lui sont associés (cf §2.2.2).

Le système proposé se base essentiellement sur cette propriété. Il est en effet composé de plusieurs mémoires vidéo superposables à la manière des celluloïdes de l'animation traditionnelle et à chaque acteur est associée une mémoire spécifique destinée à recevoir successivement tous les cells de l'acteur. La composition des différentes mémoires est réalisée en temps réel par le matériel, à la fréquence du balayage vidéo. La visualisation d'une image d'une séquence est donc le résultat du chargement des cells correspondants des acteurs dans les mémoires de trame qui leur sont attribuées.

Le deuxième objectif recherché concerne l'optimisation des temps d'enregistrement d'une animation. A cet effet, les mémoires de trame sont dotées de fonctionnalités matérielles permettant le stockage de plusieurs cells à la fois, ainsi que leur défilement en temps réel. Le système ainsi défini correspond à un système play-back en temps réel; il réalise en effet la visualisation et l'enregistrement en continu d'une séquence complète d'animation dont tous les cells ont été générés à l'avance. Le pilotage d'appareils vidéo en lecture et en enregistrement est totalement intégré au système.

Enfin une dernière optimisation consiste à minimiser la phase de génération des cells d'une animation, en exploitant d'autres fonctionnalités matérielles au niveau des mémoires vidéo; ces propriétés 2D peuvent être directement utilisées pour la réalisation des animations d'acteurs 2D ou 3D. Afin de satisfaire notre 2ème objectif, ces fonctionnalités matérielles opèrent pour la plupart en temps réel.

3.2 Acteur et cells

L'étape de description de l'animation des acteurs n'est pas abordée dans cette thèse. Cependant, une approche empirique sur la méthode de génération des cells à partir de l'animation d'acteurs est proposée dans le chapitre suivant.

Dans la suite de la thèse, nous considèrerons que l'animation d'acteurs se ramène à la manipulation de cells, et que l'on dispose des cells sous la forme de fichiers image, stockés dans le disque dur du système.

3.3 Scène et séquence

L'intérêt principal du système réside dans la manipulation des cells en temps réel et dans la description interactive de scènes d'animation.

Le système est totalement interactif et autorise l'enregistrement en continu de séquences d'animation.

De plus, les structures informatiques d'une scène sont conservées et sont directement exploitables lors de la génération du film d'animation. L'enregistrement de la séquence n'est donc pas indispensable au montage du film.

3.4 Scénario et film

Le système est également spécialisé dans le montage de films d'animation, en se limitant uniquement au domaine graphique (le montage sonore n'est pas abordé). Il permet la manipulation en temps réel de séquences d'animation. Des effets de régie sont également intégrés pour la réalisation des transitions entre les séquences.

De plus, le pilotage de matériels vidéo permet à ce niveau d'enregistrer plusieurs minutes d'animation en continu et donc de générer le film complet d'animation, en réalisant automatiquement la phase de montage.

3.5 Les contraintes du système

Le système proposé possède des contraintes dûes aux propres limites du matériel. La capacité de stockage des mémoires vidéo n'est pas infinie et il arrive parfois qu'une mémoire de trame ne puisse contenir tous les cells d'un acteur. Il est nécessaire dans ce cas de charger en cours d'animation les cells de l'acteur couramment utilisés dans la scène. Les fonctionnalités matérielles sont également limitées. Toutes les fonctions requises par le système ne peuvent pas être toutes câblées. Ces fonctions doivent alors être simulées par logiciel et n'opèrent donc pas en temps réel. Le système proposé gère automatiquement les ressources et les fonctionnalités matérielles dont il dispose. Il optimise la phase d'enregistrement d'une animation au maximum; il l'enregistre en continu dans la mesure des possibilités temps réel qui lui sont offertes. En revanche, à la rencontre d'une action non temps réel, telle que le chargement de nouveaux cells dans une mémoire de trame ou l'exécution d'une opération non câblée, deux réactions sont possibles :

- si ces actions doivent apparaitre instantanées du point de vue de l'enregistrement, le système réalise automatiquement un point de montage. La durée totale de la vidéo coïncide exactement avec la durée théorique de l'animation. On dit que le système fonctionne en mode **synchrone**.

- si ces actions doivent être enregistrées, aucun point de montage n'est réalisé. L'enregistrement se poursuit en effet pendant l'exécution de l'action non temps réel, qui peut nécessiter plusieurs trames pendant lesquelles l'animation est arrêtée. A la fin de l'action, l'animation reprend normallement son cours. La durée de l'enregistrement ne coïncide donc pas nécessairement avec la durée théorique de l'animation. On dit que le système fonctionne en mode **asynchrone.**

Le nombre des points de montage d'une animation est totalement fonction de la complexité de l'animation (nombre de cells générés, taille des cells, type des actions à exécuter sur les cells). Le cas le plus défavorable correspond au chargement d'un cell ou à la réalisation d'une action non temps réel à chaque image du film d'animation. On se ramène alors à un enregistrement de type image par image. En revanche, le cas le plus favorable correspond au chargement initial de tous les cells des acteurs dans les mémoires vidéo et à aucune réalisation d'action non temps réel. L'enregistrement est alors intégralement réalisé en continu.

Conclusion

La décomposition de la réalisation d'un film d'animation en différentes couches nous a permis de mettre en évidence les limites des différents systèmes d'animation actuels ainsi que les diverses étapes à optimiser afin d'améliorer sensiblement leur productivité. Il apparait alors clairement que la technique dite de playback en temps réel constitue un atout fondamental dans les phases de description, de visualisation et d'enregistrement des animations. Ceci nous a conduit à la spécification de nouveaux concepts d'animation, tels que le cell, et à la proposition d'un système original basé sur ces concepts. La suite de l'étude concerne l'implémentation de ces concepts dans le système, aussi bien au niveau logiciel que matériel.

CHAPITRE 3 : LES CONCEPTS DU SYSTEME

Introduction

Ce chapitre a pour but la description des concepts manipulés par le système d'animation G_ANIM. Ces concepts découlent directement du mécanisme de décomposition d'un film d'animation en différentes couches logiques, les cells, comme on l'a présenté au chapitre précédent, et de l'analyse des caractéristiques propres à ces cells.

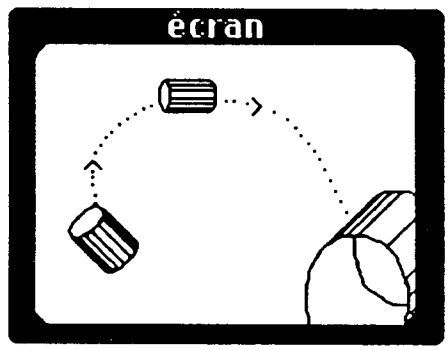
Après analyse du processus de génération d'un cell, nous aborderons l'interprétation des concepts liés à la scène d'animation, puis des concepts propres au scénario d'animation.

1. CONSTITUTION D'UNE SCENE

Après l'illustration d'une animation 3D à travers un exemple, nous évoquerons toutes les notions manipulées au niveau de la scène.

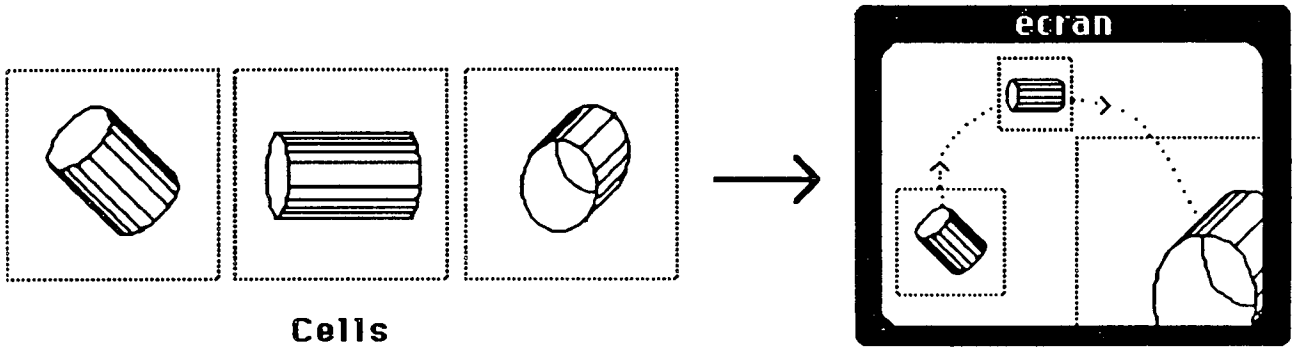
1.1 Exemple d'animation

Soit un objet 3D ne subissant pas de déformation dans le temps et amené à se déplacer le long d'une trajectoire 3D. La visualisation de l'animation de cet objet pose le problème suivant: pour éviter la génération de l'animation image par image, sur quels critères peut-on se baser pour générer un ensemble de cells représentatifs de l'animation chargés successivement dans une mémoire vidéo du système et quelles sont les fonctionnalités matérielles exploitables pour la réalisation de cette animation?



Un exemple d'animation 3D.

Une approche intuitive peut consister à générer un ensemble de cells de l'objet 3D, effectuant des rotations autour de son centre par exemple, indépendamment de la trajectoire 3D à laquelle il est asservi, puis à positionner ces différents cells relativement à l'écran en leur appliquant également un facteur de zoom.



Décomposition d'un animation au niveau de cells.

Cet exemple illustre bien l'exploitation des fonctionnalités 2D pour la réalisation d'animations 3D.

Il nous permet d'introduire la notion d'attribut d'acteur 2D ou 3D, caractérisant leur comportement au cours de l'animation. Parallèlement, les possibilités matérielles des mémoires vidéo du système sont exploitables pour la visualisation de l'animation de ces acteurs. Notre approche consiste donc à introduire la notion d'acteur plan disposant également d'attributs spécifiques 2D corrélés avec le matériel et à établir la correspondance entre l'animation d'un acteur 2D ou 3D et l'animation équivalente d'un acteur plan.

1.2 Les attributs d'acteur

Un acteur d'une animation est avant tout représenté par une structure informatique regroupant plusieurs classes d'attributs [Mar 82] : morphologie, géométrie et aspect.

La morphologie exprime la forme intrinsèque de l'acteur indépendamment de son attitude et des conditions de prise de vue. Elle est caractérisée par les arêtes et les liaisons entre ces arêtes pour un acteur 3D, par les contours polygônaux d'un acteur 2D ou encore par l'ensemble des dessins d'un acteur 2D. La morphologie d'un acteur peut évoluer au cours du temps. C'est le cas, par exemple, lors de l'interpolation des dessins d'un acteur 2D ou des anamorphoses d'un acteur 3D changeant de forme au cours du temps.

La géométrie concerne le placement de l'acteur dans l'espace qui le supporte. Ce placement est caractérisé par 2 attributs particuliers :

- la position d'un point particulier de l'acteur dans l'espace,
- les 3 angles de rotation de l'acteur par rapport aux axes du repère de l'espace.

L'aspect exprime l'apparence intrinsèque de l'acteur indépendamment des conditions d'éclairage. Il s'agit des informations relatives au matériau dont l'acteur est constitué. On y trouve des notions telles que couleur, transparence, texture...

En outre, la visualisation d'un acteur à un temps précis de l'animation nécessite des informations telles que:

La géométrie de prise de vue, qui regroupe des informations telles que, point de vue, direction de visée, focale... Ces informations se composent aux attributs géométriques de l'acteur.

L'éclairage, qui prend en compte les informations liées aux sources lumineuses telles que, direction, intensité, couleur, et qui se composent avec les attributs géométriques et d'aspect de l'acteur.

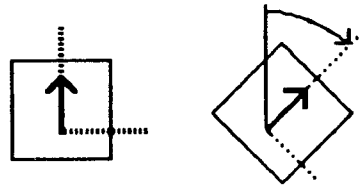
1.3 La notion d'acteur plan

Comme on l'a vu au second chapitre, l'animation d'un acteur 2D ou 3D peut être réalisée par l'animation dans le plan de l'écran d'un ensemble de cells. Cette remarque nous permet d'introduire la notion d'**acteur plan**, défini comme la projection dans le plan de l'écran d'un acteur 2D ou 3D.

Cette notion correspond à la définition générale énoncée au §2.2 du 2ème chapitre. Un acteur plan est en effet caractérisé par un ensemble de cells et par des fonctions qui leur sont appliquées lors de la visualisation (ces fonctions sont totalement liées aux fonctionnalités matérielles du système d'animation). Il peut en outre disposer de sa propre animation et réagir à des sollicitations extérieures.

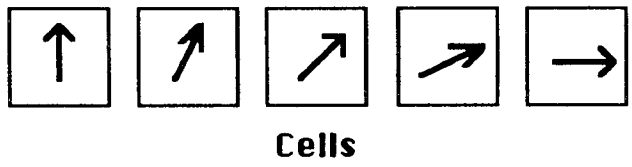
Prenons, par exemple, le cas de l'acteur "horloge" qui réagit à une information extérieure, l'heure. A la donnée d'une nouvelle heure, l'horloge est capable de s'afficher dans la position adéquate.

Pour un système d'animation autorisant les rotations 2D en temps réel, l'acteur plan "horloge" peut se ramèner à un seul cell représentant l'aiguille dans une position quelconque et l'heure à un angle de rotation du cell autour de l'axe de l'horloge. La visualisation de l'acteur revient donc à visualiser le cell avec l'angle de rotation correspondant à l'heure courante.



Rotation d'un acteur "horloge".

En revanche, pour un système ne permettant pas les rotations 2D en temps réel, l'acteur plan "horloge" doit être défini par l'ensemble des cells représentant toutes les attitudes de l'aiguille, et à une heure correspond alors un cell, qui peut être repéré par un numéro. La visualisation de l'acteur est alors réalisée par l'affichage du cell correspondant à l'heure courante.



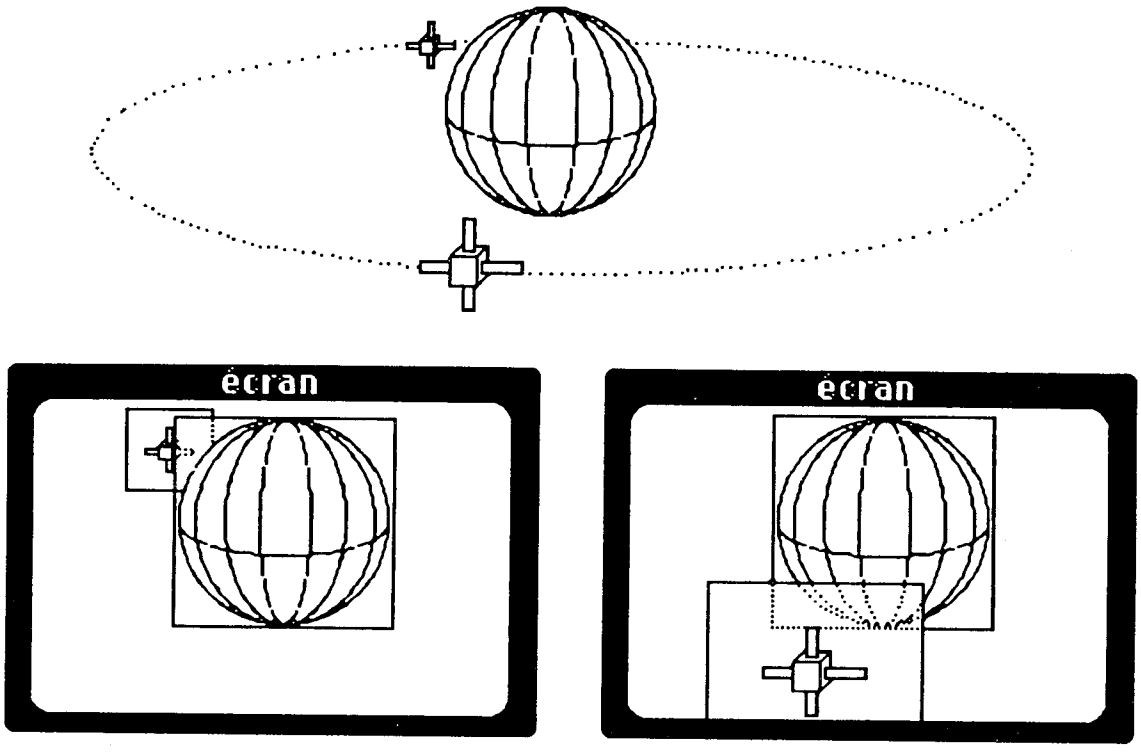
Ensemble des cells d'un acteur "horloge".

1.4 Les attributs d'acteur plan

L'angle de rotation dans le premier cas et le numéro du cell dans le second caractérise l'attitude de l'acteur plan "horloge", pris en exemple ci-dessus. Ces paramètres représentent des attributs de l'acteur plan.

Les attributs peuvent être liés à l'acteur plan, c'est-à-dire inhérents à la constitution de l'acteur. Dans ce cas, ils reflètent le comportement intrinsèque de l'acteur. La visualisation de l'"acteur sachant marcher", par exemple, est réalisée en affichant successivement les cells qui reflètent sa marche. Le déplacement de cet acteur plan conduit implicitement à la détermination du cell correspondant à sa position courante.

Plusieurs attributs peuvent être liés dans une même animation. Par exemple, dans le cas du satellite en orbite autour d'une planète, l'angle de rotation 3D autour de cette planète conduit à la liaison des attributs de l'acteur plan correspondant suivants : position du cell du satellite à l'écran, facteur de zoom et profondeur du cell relativement à celui de la planète.



Satellite en orbite autour d'une planète.

En revanche, un attribut est dit libre, lorsqu'il est manipulable directement par l'animateur, indépendamment des autres attributs.

L'implémentation d'attributs sophistiqués au niveau de l'acteur plan permet de minimiser l'étape de génération des cells, l'animation d'un acteur 2D ou 3D pouvant alors être plus facilement transcrite au niveau de l'animation d'un acteur plan. Cette implémentation dépend totalement des possibilités matérielles offertes par le système d'animation.

Dans la suite de cette étude, nous ne nous préoccuperons plus des méthodes de description d'animation d'acteurs 2D ou 3D ni de leur incidence sur la liaison des attributs des acteurs plans correspondants. Tous les attributs des acteurs plans seront considérés comme libres.

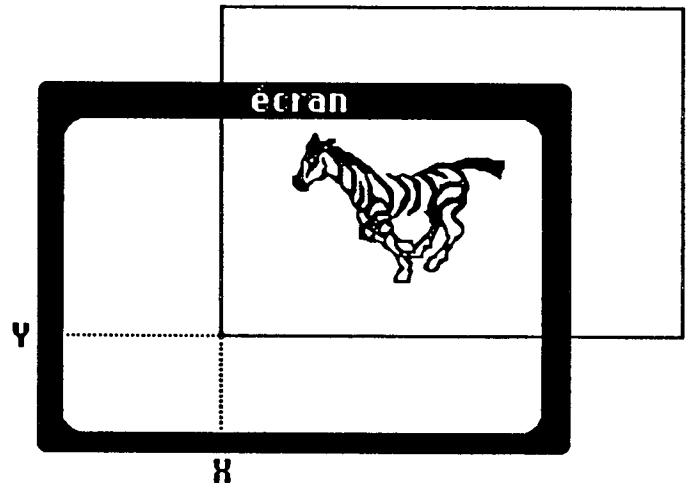
De même que pour l'acteur en général, on peut caractériser le comportement de l'acteur plan à l'aide des 3 classes d'attributs spécifiques : morphologie, géométrie et aspect.

La morphologie exprime la forme intrinsèque de l'acteur plan. Elle est représentée par l'ensemble des cells qui lui sont associés. L'évolution de la morphologie de l'acteur plan coïncide ainsi avec les changements du cell couramment visualisé. A cet effet, un cell peut être désigné par un numéro. On peut ainsi considérer que le numéro de cell, et non le cell lui-même, constitue un attribut morphologique de l'acteur plan. De plus, toutes les fonctions de

déformation s'appliquant aux cells, telles que mapping sur un cylindre, compression, dilatation, etc..., modifient le résultat de la visualisation des cells et peuvent, à ce titre, être vues comme des attributs morphologiques de l'acteur plan.

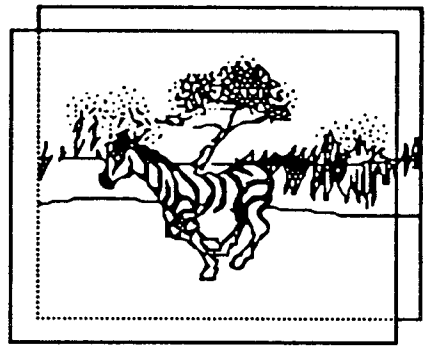
La géométrie permet de réaliser le placement de l'acteur plan à l'écran, c'est-à-dire du cell couramment visualisé. Elle regroupe les attributs suivants:

- position du cell par rapport à l'écran,



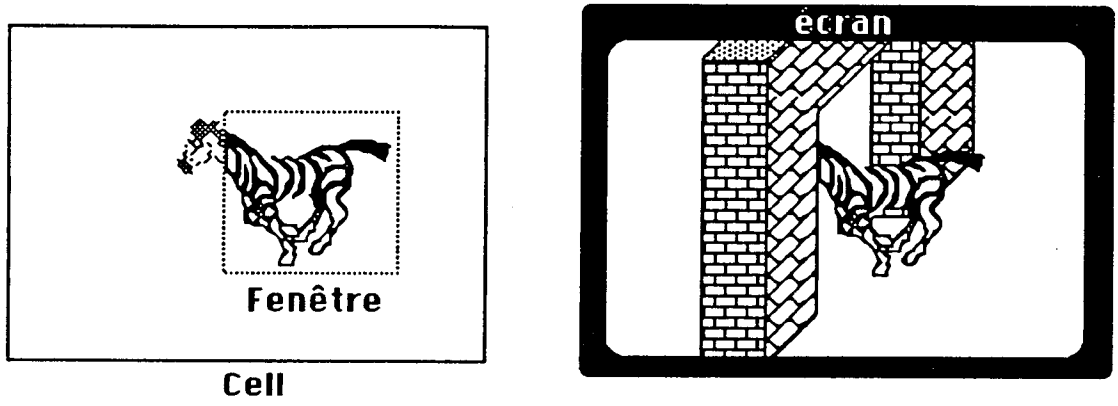
Translation d'un cell à l'écran.

- priorités du cell, définissant globalement l'ordre de superposition des cells,



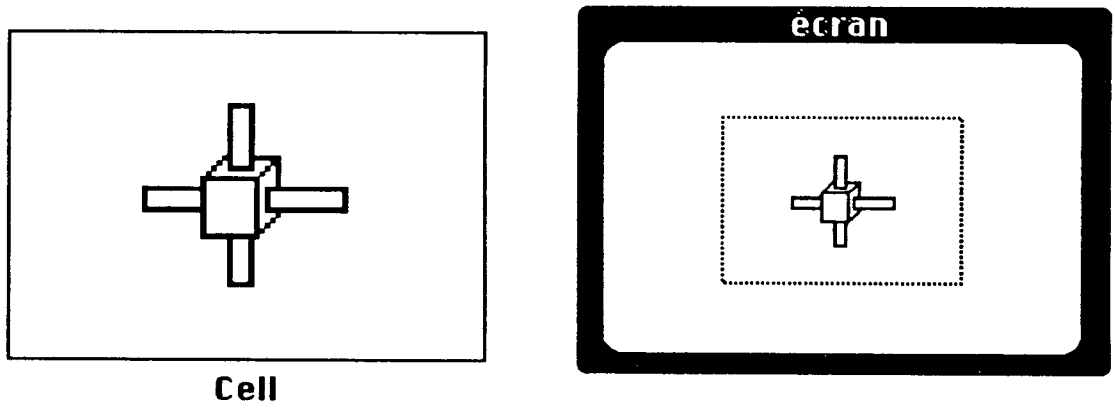
Superposition de cells.

- effet de fenêtrage ou de masque d'un cell, pour traduire l'interpénétration de 2 acteurs,



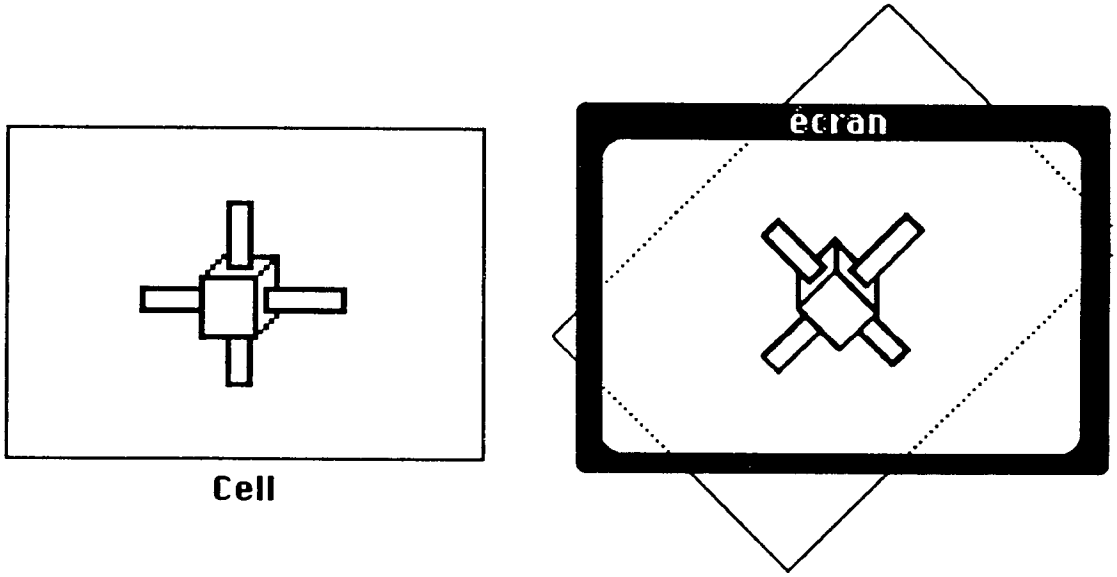
Effet de fenêtrage appliqué à un cell.

- zoom du cell, identique en x et y, pour l'éloignement de la caméra.



Zoom d'un cell à l'écran.

- rotation 2D du cell dans le plan de l'écran,

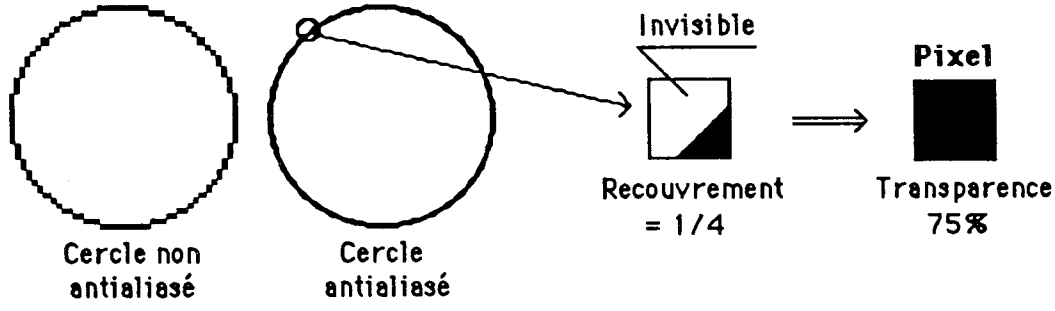


Rotation d'un cell à l'écran.

L'aspect, quant à lui, exprime l'apparence intrinsèque de l'acteur plan, c'est-à-dire du cell couramment visualisé. Il est caractérisé par les 2 attributs suivants:

- la couleur à l'écran des pixels associés au cell de l'acteur plan. Cette couleur peut provenir directement du cell ou bien encore d'une table de couleurs associée à l'acteur plan.

- la transparence des pixels associés aux cells de l'acteur plan. Cette transparence englobe également la notion d'antialiasing des cells des acteurs plans de l'animation les uns par rapport aux autres.
La résolution vidéo conduit à sur-échantillonner les pixels d'une image, afin d'atténuer les effets de crénelage lors de la génération de morphologies (cercle, droite...). Le sur-échantillonage d'un pixel consiste à calculer le % réel de recouvrement de la forme représentée en ce point. Cette notion rejoint totalement l'idée de transparence d'un objet. En effet, si un pixel n'est recouvert qu'à 50 % par exemple, on peut considérer que l'information de couleur intrinsèque à la forme représentée ne doit intervenir qu'à moitié dans la couleur réelle du pixel. Cette couleur peut ainsi être obtenue par mélange, en proportions égales, de la couleur de la morphologie avec la couleur de l'image de fond située sous cette morphologie.



Antialiasing d'une morphologie.

1.5 L'héritage des attributs

Nous nous intéressons ici à la manière de réaliser l'animation de l'acteur plan à partir des informations de l'animation d'un acteur 2D ou 3D. Nous allons, à cet effet, ramener certaines propriétés ou caractéristiques de l'animation de l'acteur 2D ou 3D au niveau de l'animation de l'acteur plan correspondant : on dit que l'acteur plan hérite des attributs de l'acteur 2D ou 3D. Cet héritage est totalement lié à la complexité de l'animation 2D ou 3D. Un attribut de l'acteur 2D ou 3D peut, en effet, avoir une correspondance avec plusieurs attributs de l'acteur plan, qui n'appartiennent pas nécessairement à la même classe. Inversement, plusieurs

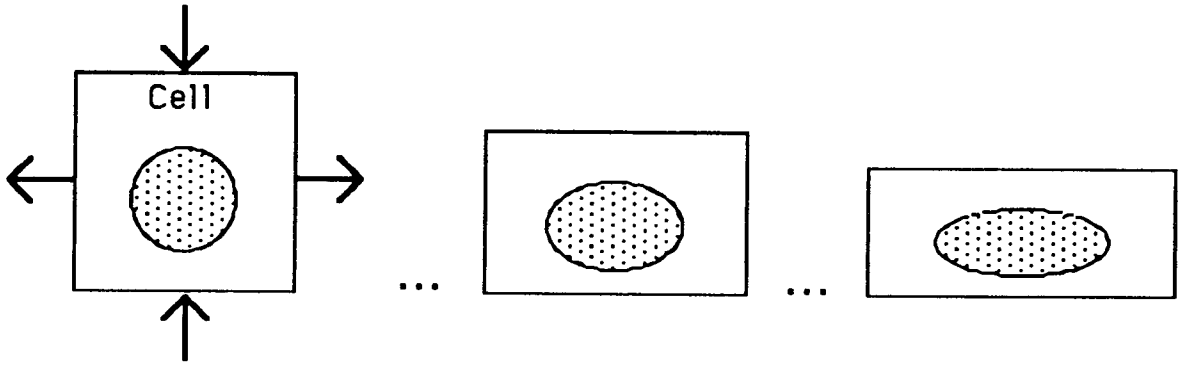
attributs de l'acteur 2D ou 3D peuvent très bien ne définir qu'un attribut de l'acteur plan.

L'héritage de ces attributs est totalement fonction de l'animation et des propriétés propres au système d'animation. Ainsi, les contraintes d'exécution en temps réel des attributs de l'acteur 2D ou 3D vont se retrouver au niveau de l'acteur plan et dépendent totalement des fonctionnalités matérielles du système d'animation (cf chapitre 4). L'implémentation d'attributs sophistiqués au niveau de l'acteur plan permet de minimiser l'étape de génération des cells et facilite la correspondance entre les attributs d'acteur 2D/3D et les attributs de l'acteur plan équivalent. En outre, l'indépendance des fonctionnalités matérielles nous a conduit à rechercher des attributs d'acteur plan indépendants et proches des implémentations matérielles.

1.5.1 Héritage morphologique

L'évolution de la morphologie d'un acteur (déformation, anamorphose) est difficile à ramener au niveau de l'acteur plan. Elle nécessite, dans la majorité des cas, la génération des cells reflétant cette évolution. L'attribut morphologique de l'acteur 2D ou 3D a alors une correspondance directe avec l'attribut morphologique de numéro de cell de l'acteur plan.

Quelques cas particuliers tels que, par exemple, l'aplatissement d'une balle au contact du sol, peuvent cependant être directement traités sans génération supplémentaire de cell. Il suffit dans l'exemple cité de compresser le cell de la balle non déformée dans la direction du choc et de l'étirer dans la direction perpendiculaire. Dans ce cas, la déformation de l'acteur "balle" a une correspondance avec l'attribut morphologique de déformation de l'acteur plan.

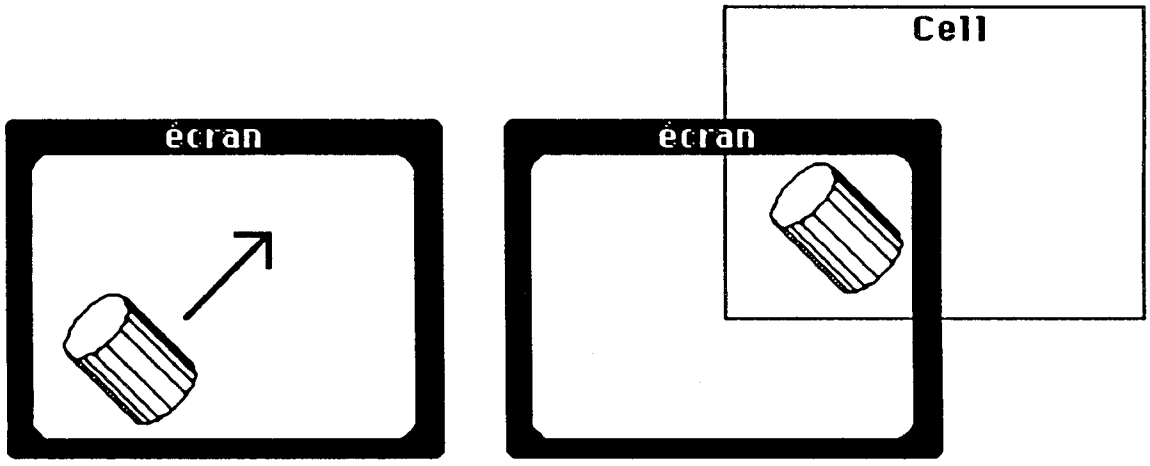


Ecrasement d'une balle au sol.

1.5.2 Héritage géométrique

On parle d'héritage géométrique lorsque l'évolution géométrique d'un acteur 2D/3D au cours d'une animation peut être traitée directement au niveau de l'acteur plan.

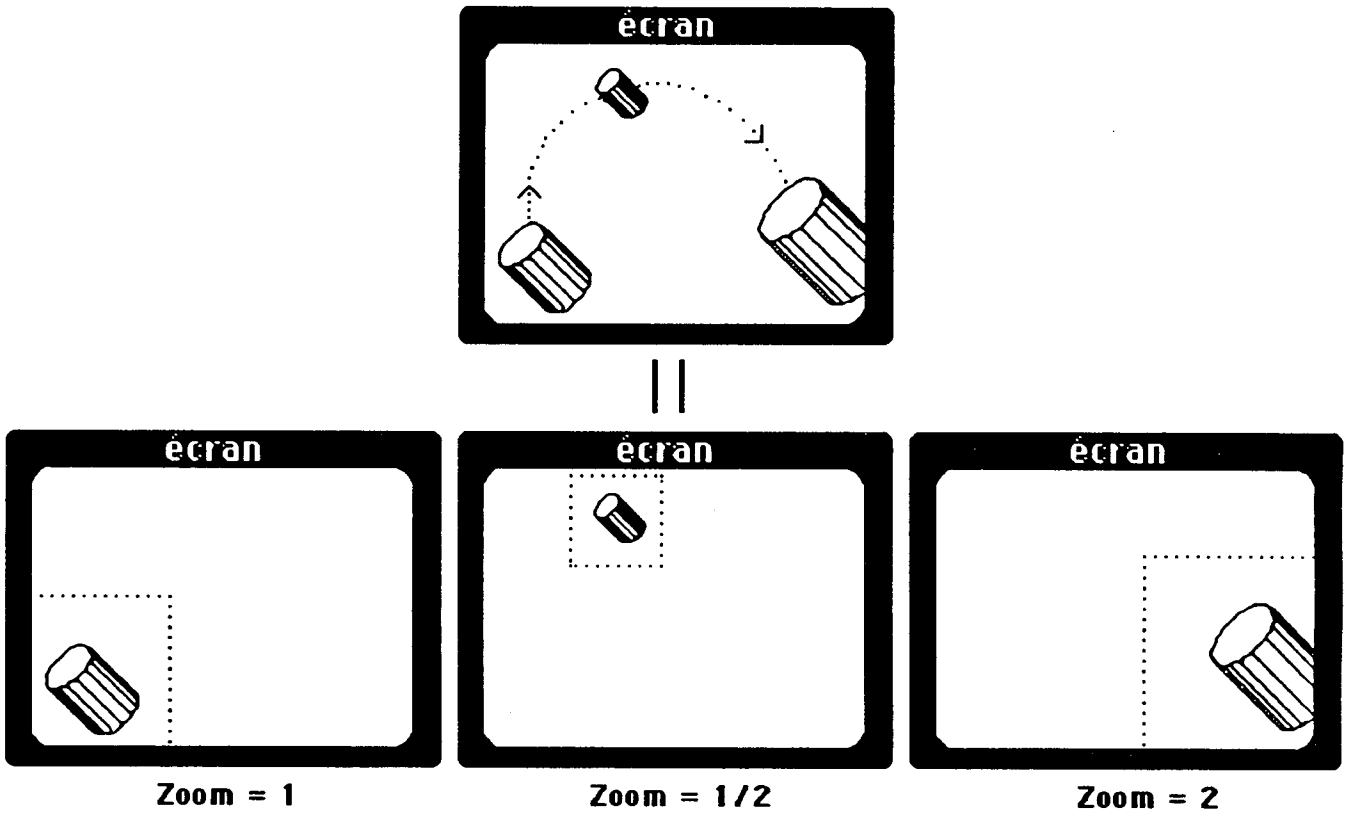
Par exemple, l'animation, qui résulte de la translation d'un acteur 3D, ne subissant aucune autre évolution (morphologie, aspect), dans le plan perpendiculaire à la direction de visée, peut se ramener aisément à la translation d'un cell quelconque de l'acteur plan, dans le repère de l'écran, si l'on admet que l'observateur est situé à l'infini et que la projection dans le plan de l'écran est orthogonale (approximation courante de la prise de vue).



Translation d'un cell par rapport à l'écran.

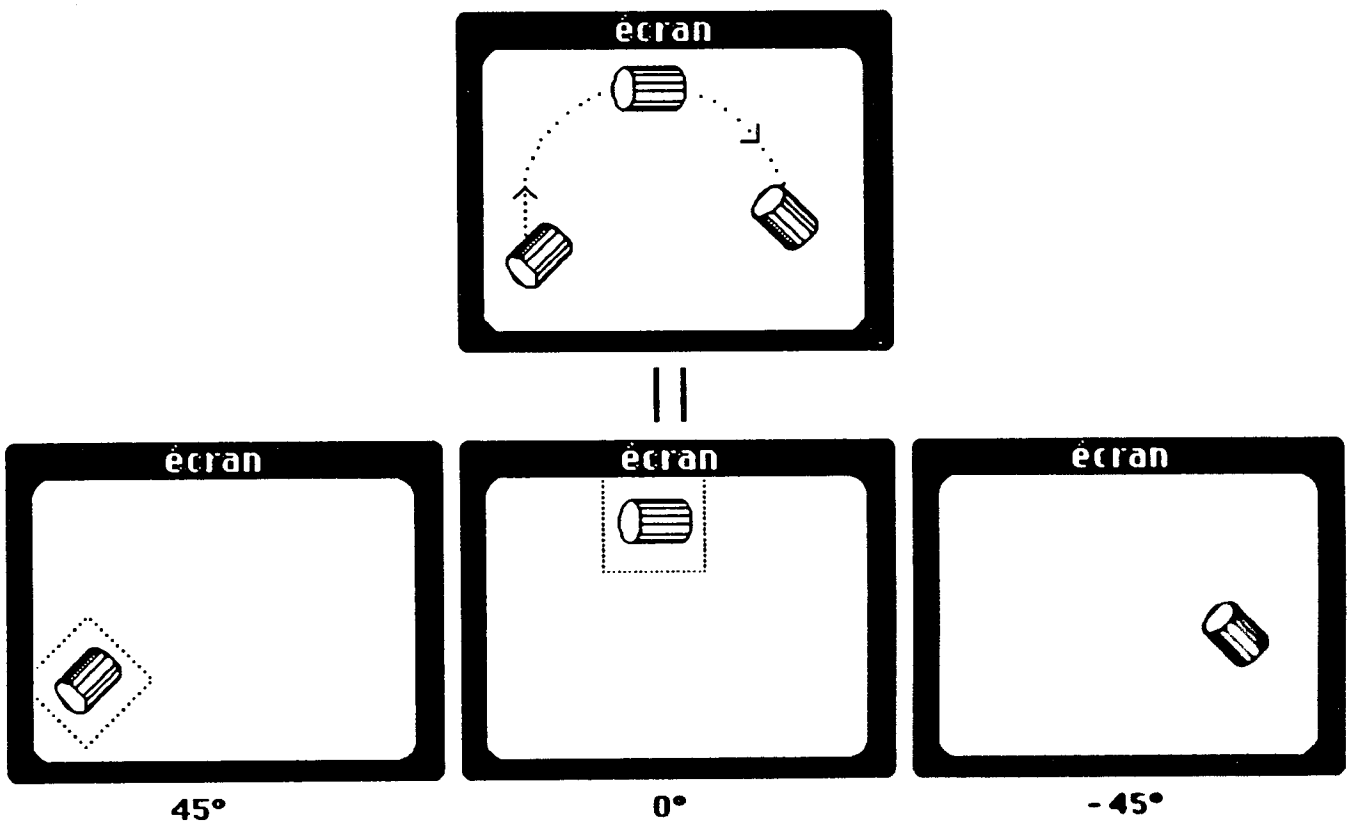
La génération d'un seul cell est alors suffisante. Et le temps de traitement logiciel ou matériel pour réaliser la translation de ce cell, est bien souvent négligeable, comparé au temps de génération de tous les cells de l'animation (pour chaque image de la séquence soit 1/25ème de seconde). L'attribut géométrique de l'acteur 3D a, dans ce cas, une correspondance directe avec l'attribut géométrique de position de l'acteur plan.

De même, le déplacement d'un acteur 3D sur une trajectoire 3D peut souvent être traité au niveau de l'animation de cells, par translation d'un cell de l'acteur dans le repère de l'écran, combinée avec un zoom de ce cell. On suppose alors que l'acteur 3D reste parallèle au plan de l'écran tout au long de l'animation. L'attribut géométrique de position de l'acteur 3D a, dans ce cas, une correspondance avec les 2 attributs géométriques, position et zoom, de l'acteur plan.



Trajectoire 3D et animation plane.

La rotation d'un acteur 3D autour d'un axe perpendiculaire à l'écran a également une correspondance directe avec l'attribut géométrique de rotation de l'acteur plan. En effet la projection à l'écran de la rotation de l'acteur 3D est équivalente, dans ce cas, à la rotation dans le plan de l'écran de la projection de l'acteur 3D, donc d'un cell de l'acteur plan correspondant. Cette rotation peut être composée avec la position du cell et son facteur de zoom.

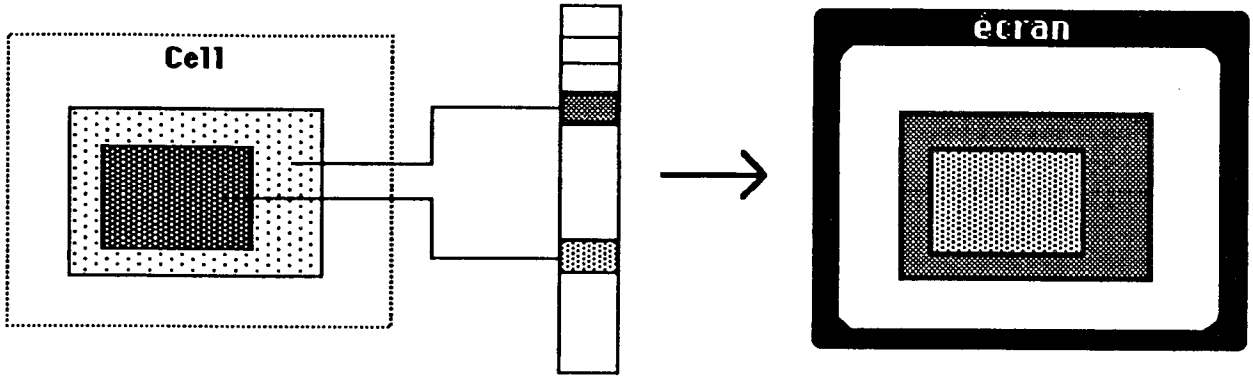


Rotation d'un acteur 3D autour d'un axe perpendiculaire à l'écran.

En revanche, une rotation 3D quelconque nécessite la génération de tous les cells représentant la projection de cette rotation dans le plan de l'écran. L'attribut géométrique de rotation a alors une correspondance avec l'attribut morphologique de numéro de cell de l'acteur plan.

1.5.3 Héritage d'aspect

A quoi bon générer tous les cells d'un acteur 2D dont la couleur change au cours de l'animation, si l'on peut modifier directement les couleurs d'un seul cell de l'acteur plan correspondant? Ces modifications de couleurs peuvent être réalisées par analyse de tous les pixels constituant le cell et réécriture de nouvelles couleurs en remplacement des anciennes. Cette méthode est assez coûteuse puisqu'elle peut nécessiter jusqu'à 720*576 comparaisons et réécritures (résolution de l'écran). En revanche, l'utilisation d'une table de couleurs est très avantageuse, puisque l'animation s'effectue en temps réel, la limite résidant uniquement dans le nombre de couleurs manipulables simultanément (taille de la table). La seule contrainte réside dans le fait qu'une table de couleurs doit pouvoir être associée à chaque acteur plan de l'animation.



Utilisation d'une table de couleurs.

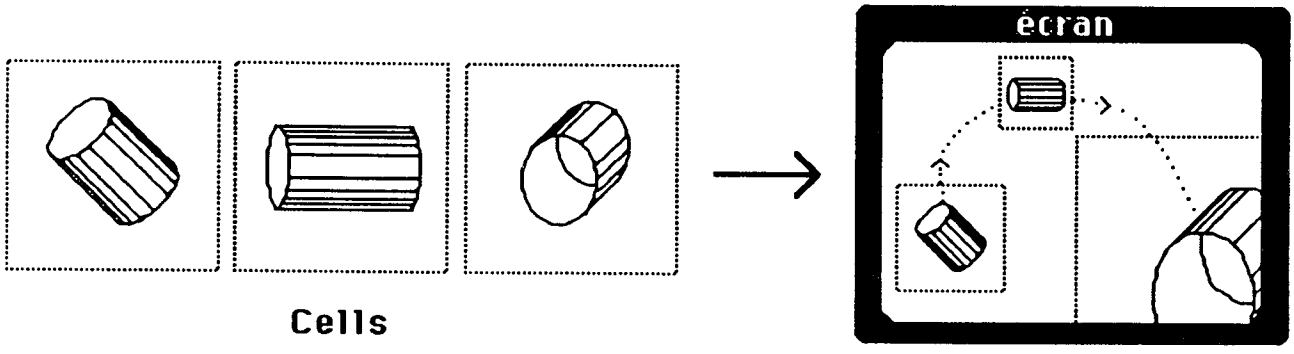
Dans le cas d'un acteur 3D, l'analyse d'un cell en composantes Teinte, Luminance, Saturation peut être également intéressante par rapport au temps nécessaire au rendu du cell. Il faut cependant se limiter au cas, assez fréquent, où la source lumineuse est blanche. La modification de la teinte de l'acteur 3D peut ainsi avoir une correspondance directe avec l'aspect de l'acteur plan équivalent.

En revanche, dans tous les autres cas, l'évolution d'aspect d'un acteur 2D ou 3D doit nécessairement se répercuter sur la morphologie de l'acteur plan, puisque tous les cells correspondant à cette évolution doivent être générés.

1.5.4 Conséquences

Des situations plus complexes que les cas évoqués ci-dessus se présentent lorsque l'acteur subit à la fois des évolutions morphologiques, géométriques et d'aspect.

Cependant une animation d'un acteur 2D ou 3D peut être ramenée, dans la majorité des cas, au niveau du cell. Un ensemble de cells de base, traduisant par exemple une évolution morphologique de l'acteur ou encore sa rotation autour d'un axe quelconque de l'espace, auquel s'appliquent un ensemble d'attributs spécifiques, suffit très souvent à créer l'illusion d'une véritable animation 3D.



Animation plane d'un déplacement 3D de l'observateur.

L'animateur doit savoir apprécier le degré d'indépendance entre la phase de génération des cells et la phase d'animation des cells. Pour des animations trop complexes ou à réaliser trop précisément, tous les cells devront être générés et l'animation plane se réduira à un simple défilement de ces cells. En revanche, pour des animations plus simples, une bonne décomposition impliquera une réduction du temps de génération des cells et une simplification de la description de l'animation de l'acteur 2D ou 3D en vue de cette génération. En contre partie, l'animateur devra décrire une animation de cells, et manipuler les concepts présentés dans ce chapitre.

2. DESCRIPTION D'UNE SCENE

L'animation d'un acteur plan nécessite la maîtrise de tous les attributs cités ci-dessus. La description d'une animation demande aux animateurs une grande rigueur et apparait d'autant plus aisée qu'elle peut être structurée. L'indépendance des fonctionnalités matérielles et donc des attributs de l'acteur plan, non seulement entre les trois classes, mais aussi à l'intérieur de chaque classe, opère dans ce sens.

2.1 Le temps

2.1.1 L'unité de temps

L'unité de temps choisie pour décrire une scène est le 50ème de seconde, c'est-à-dire le retour de trame. En effet, le système proposé est un système manipulant les cells en temps réel, c'est-à-dire qu'il est susceptible, à chaque retour de trame, de mettre à jour les informations nécessaires à la visualisation. La description d'une animation s'effectue également au niveau de la trame, ce qui autorise, dans le cas de la trajectoire par exemple, des déplacements visiblement plus lisses que la même animation définie au niveau de l'image (au 25ème de seconde).

La plupart des actions d'animation s'effectuent en temps réel, c'est-à-dire que le système est capable de visualiser une "nouvelle" image (résultat de la superposition des cells) tous les 50ème de seconde. En revanche, certaines actions, telles que chargement d'un fichier à partir du disque dur de l'ordinateur ne peuvent s'effectuer en temps réel et nécessitent, lors de l'enregistrement, la réalisation d'un point de montage. Le système gère automatiquement ces points de montage en exploitant, quand il le peut, les possibilités temps réel qui lui sont offertes.

2.1.2 Notion d'étape

La manipulation des nombreux attributs d'animation d'un acteur plan peut devenir très lourde s'il est nécessaire pour chaque trame de l'animation de définir chacun de ces attributs. Aussi le système repose-t-il sur le principe d'interpolation,

qui consiste à calculer tous les attributs de l'animation pour chaque trame, à partir d'un ensemble d'informations-clés.

Le lien entre le temps et les attributs de l'animation est réalisé par l'introduction de la notion d'étape. Une **étape** est vue comme un instant précis de l'animation (une trame donnée) auquel sont définis un ou plusieurs attributs. Une étape permet notamment de synchroniser plusieurs attributs de l'animation, d'un même acteur (coïncidence de la morphologie et de la position) ou de plusieurs (collision de 2 acteurs).

2.1.3 Notion d'intervalle

Le principe d'interpolation évite à l'animateur des descriptions inutiles, à condition que les fonctions d'interpolation utilisées correspondent bien à l'effet d'animation souhaité. Il est donc nécessaire de choisir parmi plusieurs types de fonctions d'interpolation et de sélectionner le champ d'application de ces fonctions.

Cette remarque permet d'introduire la notion d'**intervalle**, qui représente la durée de validité d'une fonction d'interpolation. Ainsi, chaque attribut de l'animation possède ses propres intervalles, contigus et dont la durée totale coïncide avec la durée de l'animation.

Les fonctions de calcul associées aux intervalles sont de différents types :

- **discret** : la valeur de l'attribut ne change que sur des étapes,

  soient {t0, t1,...,tn} les instants correspondant à des étapes,
  quel que soit t appartenant à [ti, ti+1[,

  $$f(t) = f(ti);$$

- **linéaire** : il y a interpolation linéaire entre les valeurs des étapes de l'attribut,

  soient {t0, t1,...,tn} les instants correspondant à des étapes,
  quel que soit t appartenant à [ti, ti+1],

  $$f(t) = f(ti) + (t - ti) * (f(ti+1) - f(ti)) / (ti+1 - ti);$$

- **lissé** : les valeurs des étapes appartenant à l'intervalle lissé sont les valeurs-clés d'un calcul de splines. Le calcul par cette méthode est rapporté dans l'annexe I.

- **accéléré**, entre 2 étapes, ti et ti+1 :

  $$f(t) = f(ti) + (f(ti+1) - f(ti)) * (1 - \cos(\pi/2 * (t - ti) / (ti+1 - ti)));$$

- **ralenti**, entre 2 étapes, ti et ti+1 :

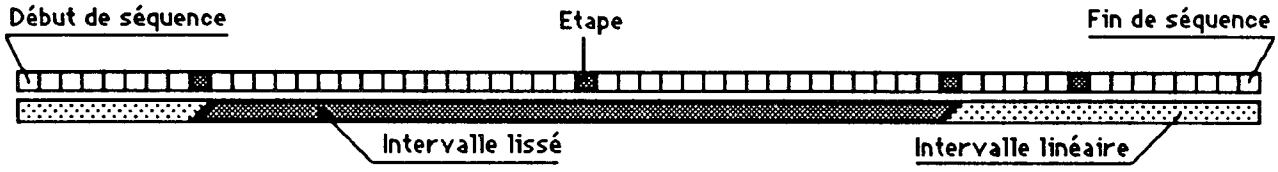
$$f(t) = f(ti) + (f(ti+1) - f(ti)) * \sin(\pi/2 * (t - ti) / (ti+1 - ti));$$

- **accéléré puis ralenti**, entre 2 étapes, ti et ti+1 :

$$f(t) = f(ti) + (f(ti+1) - f(ti)) * (1 - \cos(\pi * (t - ti) / (ti+1 - ti))) / 2;$$

remarque:

Ces 3 dernières fonctions de calcul ne sont utilisables qu'entre 2 valeurs-clés. En effet les dérivées en ti et ti+1 n'étant pas identiques, les calculs successifs entre plusieurs étapes d'un intervalle provoquent des discontinuités de vitesse. Pour des intervalles possèdant plusieurs étapes, il est donc nécessaire de ramener ces fonctions à un calcul de spline tenant compte de contraintes particulières.



Représentation d'étapes et d'intervalles.

2.1.4 Notion d'évolution

Un représentation duale de la précédente consiste à considérer que les attributs évoluent dans un domaine propre et qu'une fonction d'évolution permet de calculer à chaque temps d'animation la valeur courante de l'attribut.

Tout attribut de l'animation trouve en effet ses valeurs dans un ensemble défini appelé **domaine d'évolution** et représentant l'espace d'arrivée de la fonction d'animation associée. Cette fonction, appelée **fonction d'évolution**, établit la correspondance entre le temps de l'animation et un point du domaine d'évolution associé.

Les valeurs à exprimer sont diverses, selon qu'il s'agit de déplacements, de déformations, de modification d'aspect, etc...

exemple : trajectoire

Le domaine d'évolution est constitué de l'ensemble de coordonnées des points de la trajectoire; les déplacements de l'acteur sont asservis à cet ensemble. A chaque point de la trajectoire correspond une abscisse curviligne. Inversement à tout réel appartenant au segment [0,1] correspond un point de la trajectoire et un seul. Il y a donc un isomorphisme entre l'ensemble des points de la trajectoire et le segment [0,1]. La fonction d'évolution associée

établit explicitement à chaque instant la correspondance entre un temps et une abscisse curviligne, et donc un point de la trajectoire.

En fait la représentation par intervalle convient dans la majorité des cas, et simplifie le travail de description de l'animateur, puisque celui-ci choisit parmi des fonctions prédéfinies et n'intervient pas dans la définition de ces fonctions.

En revanche, la représentation sous forme d'évolution apporte plus de liberté à l'animateur qui maîtrise totalement ses effets ainsi que leur cinématique. C'est le cas notamment pour la description des trajectoires 2D des cells dans le repère de l'écran.

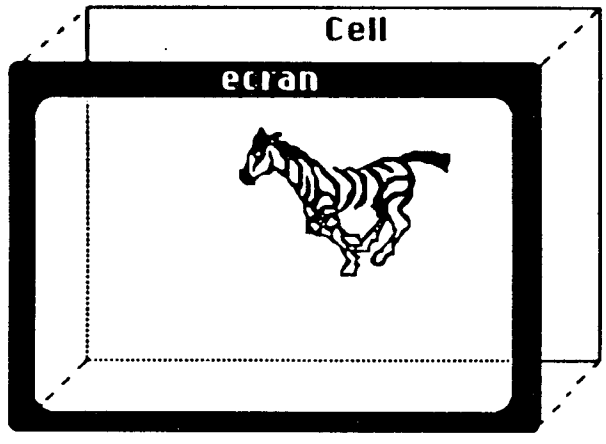
2.2 Evolution morphologique

Cette évolution est caractérisée essentiellement par les modifications du contenu des cells et par l'enchainement des différents cells d'un acteur de l'animation.

2.2.1 Limitation des cells

De la même manière que le celluloïde en animation traditionnelle, le cell est le support des différents dessins d'un acteur.

La génération d'un acteur à un instant de l'animation, conduit à la constitution d'un cell, transparent pour les pixels où l'acteur n'apparait pas et de la taille de l'écran de visualisation. Les quatre coins d'un cell coïncident alors avec les quatre coins de l'écran et la superposition des différents cells aboutit à la visualisation d'une image de la séquence d'animation.

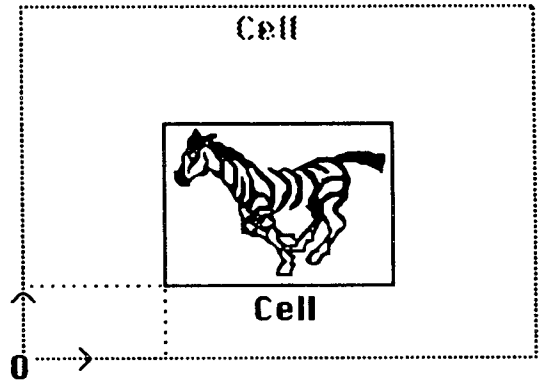


Un cell d'une animation.

L'introduction d'attributs géométriques au niveau du cell, conduit nécessairement à la détermination d'un repère propre à chaque cell. Ce repère peut avoir son origine en n'importe quel point du cell, par exemple son coin inférieur gauche.

En fait, les cells n'ont d'intérêt que pour les dessins qu'ils supportent. Ces dessins peuvent toujours être inscrits dans un rectangle. Si les cellulos de l'animation traditionnelle ont une grandeur fixe et identique pour chacun, les cells de notre système informatique sont réduits aux rectangles englobants les dessins. Toutes les transformations s'appliquant théoriquement au cell de la taille de l'écran peuvent s'appliquer directement sur le rectangle englobant le dessin, dans un repère qui lui est propre.

Dans la suite de cette étude, un cell sera donc assimilé au rectangle englobant un dessin de l'acteur plan.



Réduction d'un cell à sa boite englobante.

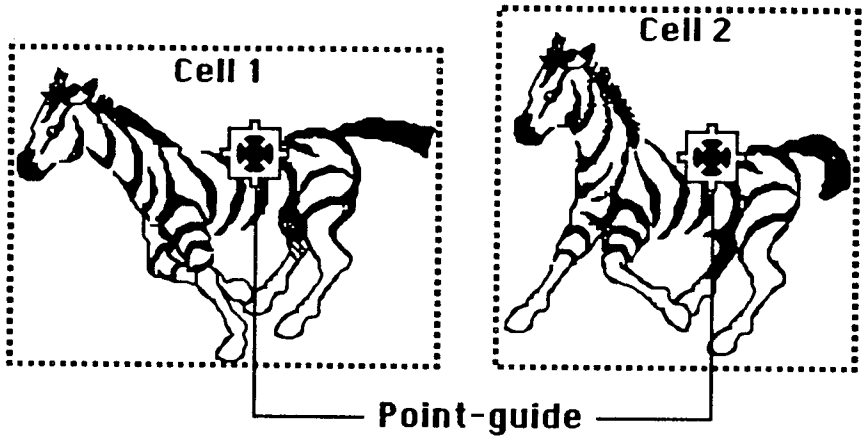
2.2.2 Le point-guide

Comme on a pu le voir précédemment, il apparait nécessaire de séparer nettement la phase de génération des cells, de la phase de description de l'animation. Ainsi, les cells, caractérisant par exemple l'évolution morphologique d'un acteur, sont souvent générés indépendamment des attributs géométriques propres à l'animation de cet acteur.

L'indépendance des attributs de l'acteur plan, notamment entre les numéros de cells et leur position à l'écran, ne peut être respectée qu'après l'introduction d'un point invariant entre tous les cells de l'acteur plan. Ce point peut être positionné manuellement dans le cas d'une animation de dessin 2D ou bien encore être hérité des informations de l'animation d'un acteur 3D projeté à l'écran.

Dans l'exemple évoqué au §1.1, il est apparu nécessaire de générer tous les cells correspondant à la rotation de l'acteur 3D autour de son centre de gravité. Ce point 3D a été projeté dans le plan de l'écran et définit pour chaque cell généré un point 2D exprimé relativement au repère associé au cell. L'animation de l'acteur plan équivalente à l'animation de l'acteur 3D est alors réalisée par positionnements successifs de ce point caractéristique des cells de l'acteur plan sur la trajectoire 2D résultant de la projection dans le plan de l'écran de la trajectoire 3D.

Ce point invariant entre tous les cells d'un acteur plan est appelé **point-guide** de l'acteur plan. C'est non seulement le point du cell amené à se déplacer le long de la trajectoire associée à l'acteur, mais aussi le centre de toutes les transformations géométriques ou morphologiques de l'acteur plan.



Cells d'un acteur avec leur point-guide.

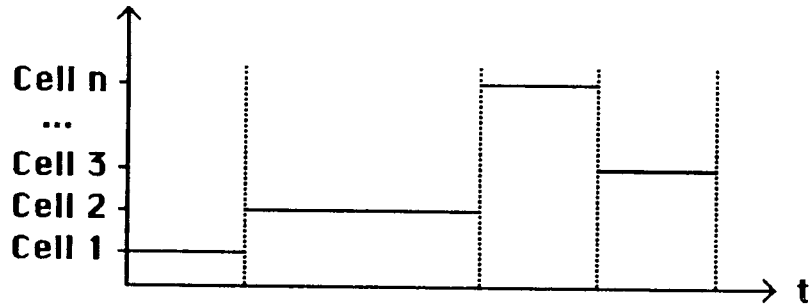
Le choix d'un point-guide inadéquat conduit à rectifier le mouvement de l'acteur par modification de sa trajectoire. On perd alors l'indépendance des attributs de sélection de cell et de position, et la moindre modification de l'un entraine irrémédiablement des modifications de l'autre.

2.2.3 L'album

L'**album** d'un acteur plan regroupe toutes les informations relatives aux cells de cet acteur. Il permet, en outre, d'établir la correspondance entre un cell et son numéro.

La sélection d'un cell peut être assimilée à un attribut de l'animation. Cette sélection s'opère par l'intermédiaire de l'album, qui propose l'ensemble des numéros de cells d'un acteur.

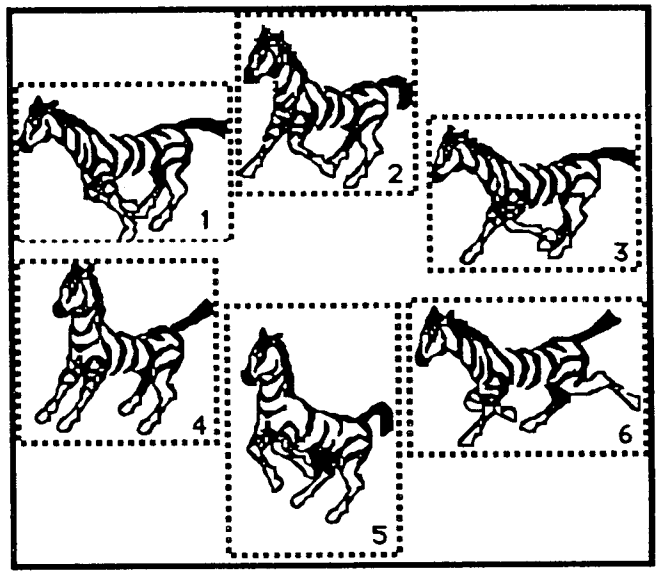
L'album définit ainsi le domaine d'évolution de l'attribut de sélection de cell et la fonction d'évolution propre à cet attribut peut être représentée comme suit:



Graphe de l'évolution de cell.

Cette fonction d'évolution est constante par paliers. L'interpolation entre des numéros de cells n'a, en effet, aucune signification pour des dessins non structurés (ensemble de pixels). Dans le cas de dessins structurés (contours polygônaux), l'interpolation des numéros de cells pourrait conduire à l'interpolation des différents contours et donc à la génération automatique de dessins intermédiaires.

L'album d'un acteur plan a un support physique qui permet le stockage des dessins de tous les cells de cet acteur et possède une représentation informatique pour l'archivage et la gestion des caractéristiques propres aux cells (coordonnées du rectangle englobant et de leurs points-guide dans le support de l'album).



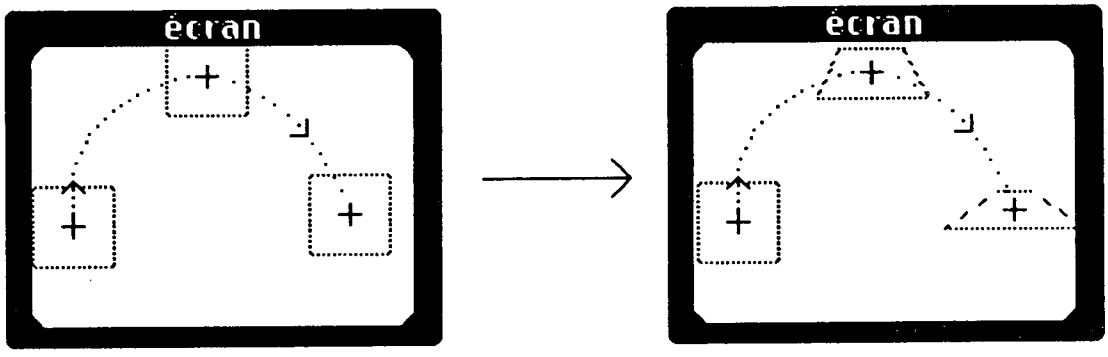
Album proposant 6 cells.

2.2.4 Les déformations de cells

Les déformations de cells regroupent toutes les fonctions de traitement au niveau du pixel susceptibles de modifier la nature du cell (et donc la morphologie de l'acteur).

Pour être exploitables dans le contexte d'évolution de cells présenté ci-dessus, ces déformations doivent respecter la notion de cell et de point-guide, ceci afin de préserver la continuité de l'animation et d'assurer l'indépendance des attributs de l'animation.

Prenons par exemple la mise en perspective d'une animation de cell.



Mise en perspective d'une animation.

Dans la mesure où l'animation de cells non déformés est continue, l'application d'une fonction de déformation sur chacun des cells, autour de leurs points-guide respectifs, devra conserver cette continuité.

2.3 Evolution géométrique

2.3.1 Position

La position d'un cell à l'écran est directement déterminée par la position de son point-guide dans le repère associé à l'écran.

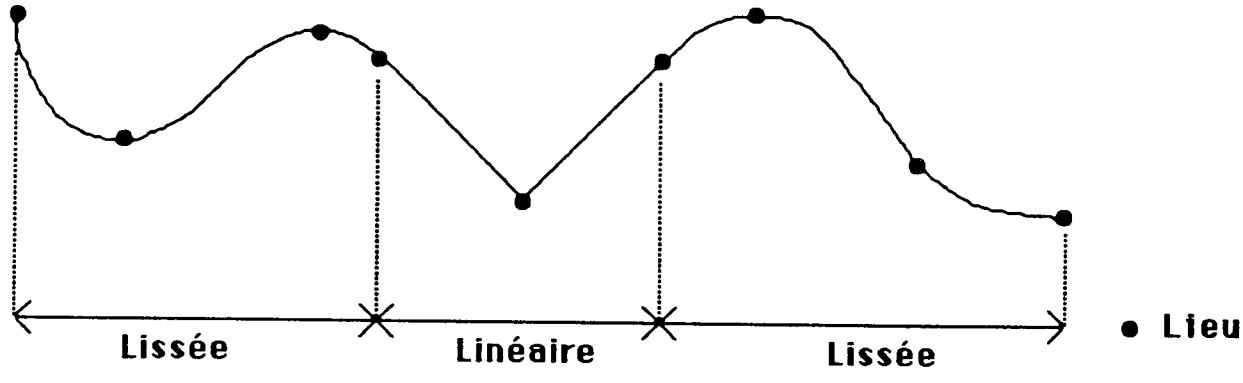
L'attribut de position induit la notion de trajectoire 2D, correlée avec une fonction d'évolution, qui traduit la progression de l'acteur le long de cette trajectoire au cours du temps.

Une trajectoire est un ensemble continu de points 2D (xi, yi). Une méthode de génération de trajectoire consiste à définir un ensemble de points-clés indicés, appelés **lieux** (Xj, Yj), et à appliquer une fonction d'interpolation entre ces points. On peut distinguer 2 types de fonctions d'interpolation :

- fonction linéaire : les points-clés de la trajectoire sont reliés par des segments de droite.

- fonction lissée : les points-clés de la trajectoire sont reliés par des arcs de courbes, issus d'un calcul de lissage par la méthode des splines.

Une trajectoire est nécessairement continue et peut être calculée à partir de ces 2 fonctions d'interpolation.



Une trajectoire composée.

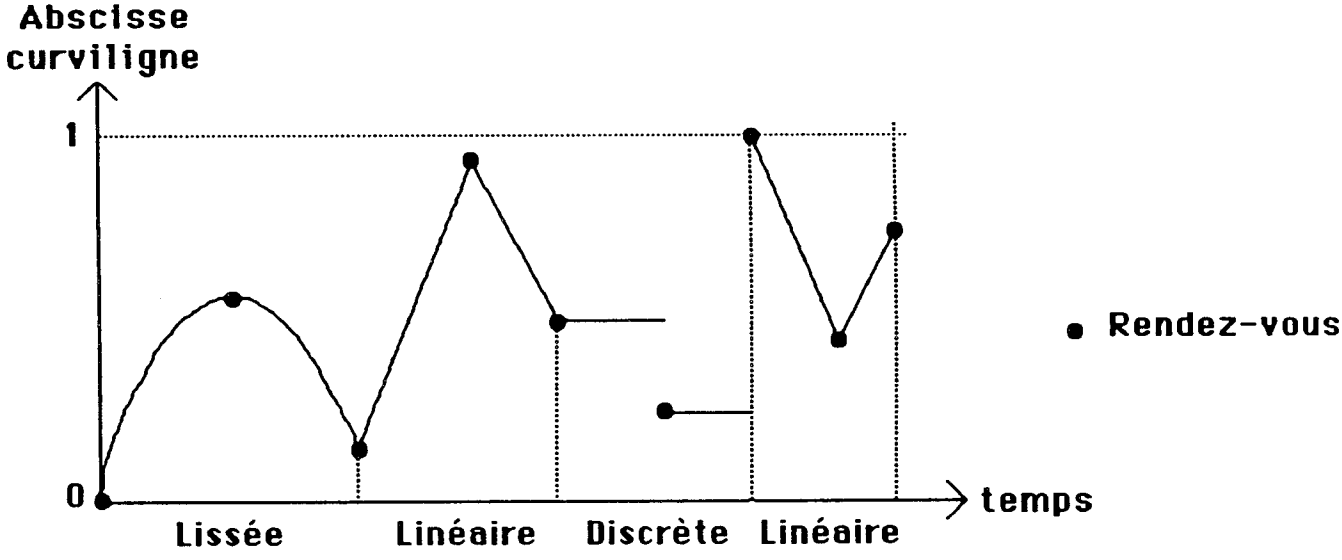
A chaque point de la trajectoire est associée une abscisse curviligne, représentant la distance à parcourir pour atteindre ce point, depuis l'origine de la trajectoire.

Une fonction d'évolution établit la correspondance entre un instant de l'animation et la position du point-guide sur la trajectoire de l'acteur. Cette fonction peut être représentée par un graphe dont l'abscisse est le temps et l'ordonnée l'abscisse curviligne. Ce graphe est déterminé à partir d'un ensemble de points-clés, appelés **rendez-vous**, et d'une fonction d'interpolation entre ces points-clés. La dérivée de cette fonction d'évolution à un temps donné représente la vitesse instantanée de l'acteur le long de la trajectoire.

On peut distinguer trois types de fonctions d'interpolation :

- fonction constante par paliers : L'acteur reste à la position de la trajectoire correspondant à l'ordonnée d'un palier pendant la durée associée à ce palier.
- fonction linéaire : L'acteur se déplace à vitesse constante entre deux rendez-vous. Les rendez-vous représentent les points de discontinuité de la vitesse.
- fonction lissée : La vitesse de déplacement de l'acteur le long de la trajectoire est continue.

Le graphe d'évolution peut être composé de ces 3 fonctions d'interpolation.



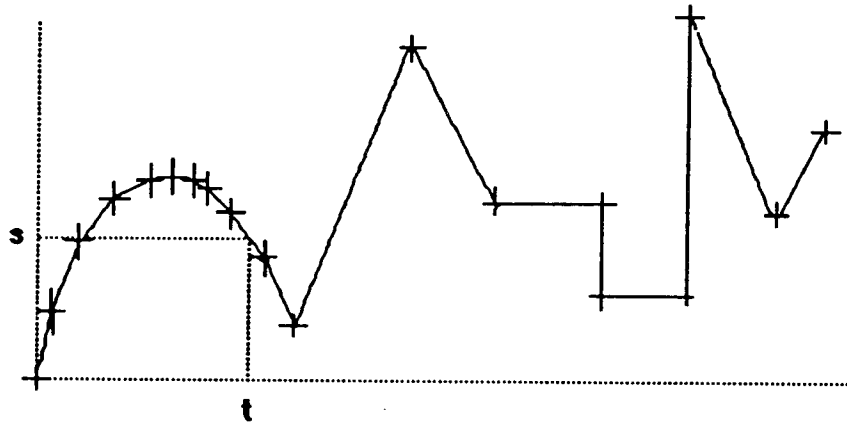
Graphe d'une fonction d'évolution.

remarque : un rendez-vous ne correspond pas forcément à un lieu de la trajectoire.

La détermination, à tout instant de l'animation, de la position du cell courant d'un acteur plan nécessite de nombreux calculs.

Tout d'abord la génération du graphe de la fonction d'évolution : A partir de l'ensemble des rendez-vous et des fonctions de calcul s'appliquant à l'évolution, une ligne polygônale approchant la courbe théorique de l'évolution peut être

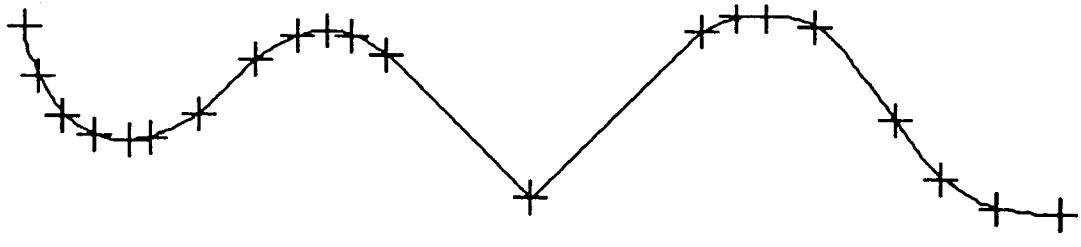
calculée. Une fonction de calcul discrète retourne une ligne brisée, sous la forme de créneaux et une fonction de calcul linéaire retourne directement les sommets que constituent les rendez-vous. En revanche, une fonction de calcul lissée nécessite des calculs basés sur la méthode mathématique des splines déterminant une ligne brisée constituée des sommets de discrétisation de la courbe.



Discrétisation d'une fonction d'évolution.

A chaque temps il est alors possible d'associer une abscisse curviligne, en recherchant dans le graphe d'évolution les 2 sommets encadrant ce temps puis en interpolant les 2 abscisses curvilignes qui leur sont respectivement associées.

Le graphe de la trajectoire doit ensuite être calculé à partir des lieux et des fonctions de calcul qui s'y rapportent. On détermine une ligne polygônale approchant la courbe théorique de la trajectoire.



Discrétisation d'une trajectoire.

On calcule ensuite l'abscisse curviligne de chacun des sommets de la ligne posygônale. On recherche enfin les 2 points de la trajectoire encadrant la valeur de l'abscisse curviligne du point courant, et on calcule les coordonnées de ce point courant par interpolation à partir de ces valeurs et des coordonnées des points associés.

2.3.2 Priorités

Chaque acteur possède à tout instant un rang et un seul dans le mécanisme de superposition des cells d'une animation (l'acteur 2 est associé par exemple au niveau 3). Inversement, à chaque niveau de profondeur est associé un acteur au plus (au niveau 3 est visualisé l'acteur 2). Le domaine d'évolution associé à chaque niveau de profondeur est donc constitué des indices des différents acteurs plan de l'animation.

2.3.3 Fenêtre et masques

L'effet de fenêtrage d'un acteur est réalisé par application d'une zone rectangulaire sur le cell de l'acteur. Les coordonnées de cette zone sont exprimées dans le repère de l'écran. Seuls sont visibles les pixels du cell situés sous la fenêtre écran.

Inversement, une anti-fenêtre permet de ne visualiser que les pixels du cell situés en dehors de la zone délimitée par la fenêtre.

Une généralisation de l'effet de fenêtrage conduit à la notion de masque, de forme quelconque, exprimé par rapport à l'écran. Les pixels du cell situés sous le masque peuvent être, au choix, rendus visibles ou invisibles.

2.3.4 Zoom

L'attribut de zoom doit être indépendant des autres attributs de l'animation. Afin d'assurer la continuité d'une animation lors de l'introduction du facteur de zoom, cet attribut doit être assimilé à une dilatation dont le centre est le point-guide du cell courant de l'acteur. En tant qu'opérateur géométrique, le zoom permet de traduire l'éloignement de la caméra. Il doit donc être impérativement identique suivant les directions x et y du cell.

Une trajectoire plane sur laquelle évolue un acteur plan peut facilement refléter une trajectoire 3D par application d'un facteur de zoom à l'acteur plan, indépendamment des valeurs des autres attributs.

Le domaine d'évolution de l'attribut de zoom est l'ensemble des réels et sa fonction d'évolution permet de retrouver à chaque instant le facteur appliqué au cell courant.

2.3.5 Rotation

De même que le facteur de zoom, l'attribut de rotation doit être indépendant des autres attributs de l'animation. L'introduction d'un angle de rotation ne doit pas casser le rythme de l'animation, et notamment la trajectoire de l'acteur. Il est donc nécessaire d'effectuer cette rotation autour du point-guide du cell courant de l'acteur plan.

Le domaine d'évolution de l'attribut de rotation est l'ensemble des réels et la fonction d'évolution permet de retrouver à chque instant l'angle de rotation applikqué au cell courant.

2.4 Evolution d'aspect

2.4.1 Les couleurs

L'attribut de couleur a pour fonction d'établir la correspondance entre des couleurs réelles (inscrites dans le cell) et des fausses couleurs (inscrites dans une table). A chaque instant de l'animation, il est possible d'obtenir les couleurs courantes d'un acteur en consultant des graphes, définis pour une couleur réelle ou un intervalle de couleurs réelles, représentés de la manière suivante :

En fait, il est plus cohérent de considérer à chaque fois trois graphes associés à chacune des composantes couleur R, V, B ou T, L, S.

2.4.2 La transparence

La notion de transparence correspond à l'introduction de niveaux pondérés dans le mécanisme de profondeur évoqué ci-dessus. Le domaine associé à cet attribut est le segment de droite [0, 1], dont les extrêmités correspondent respectivement à une totale opacité et à une complète transparence.

3. CONSTITUTION D'UN SCENARIO

Comme on l'a défini au §2.4 du second chapitre, le scénario décrit l'"animation" des séquences et conduit à la réalisation du film d'animation. Après avoir défini les différentes séquences manipulées par le scénario, nous évoquerons les paramètres propres à la description de ces séquences.

3.1 Les séquences

Une séquence présente les mêmes caractéristiques que l'acteur énoncées au §2 du second chapitre :

- elle joue un rôle imposé par le réalisateur du film d'animation,

- elle possède sa propre animation, constituée de la suite de ses images,

- elle réagit aux sollicitation du monde extérieur; le réalisateur communique par exemple la vitesse de défilement à la séquence, qui se synchronise automatiquement sur l'animation,

- elle conserve certaines propriétés inhérentes à sa constitution, par exemple le défilement de la séquence doit être continu.

On peut distinguer trois catégories de séquences :

- les séquences internes,
- les séquences externes,
- les séquences dégénérées.

3.1.1 Séquences internes

Ce sont des séquences décrites dans le système d'animation de scènes présenté au paragraphe 3 de ce chapitre. Leurs structures informatiques ont été conservées et leur interprétation conduit à la génération interactive de la séquence en interne du système d'animation. Toutes les informations relatives aux albums des acteurs de la séquence (cells et points-guide) sont également conservées et accessibles au système d'animation.

3.1.2 Séquences externes

Ce type de séquences nécessite le pilotage d'un appareil vidéo externe au système d'animation, à travers une carte d'acquisition vidéo. La suite des images fournies par le système externe détermine une séquence d'animation. La source de ces images peut être de différentes natures :

- caméra vidéo,
- magnétoscope,
- vidéo-disque,
- disque dur numérique ou analogique,
- station de synthèse d'images.

Chacune de ces sources vidéo dispose d'un protocole de communication spécifique et de fonctions de pilotage propres.

3.1.3 Séquences dégénérées

Ces séquences interviennent en fait comme un niveau d'image et leur animation est élémentaire. C'est par exemple le cas d'une image jouant le rôle du décor, ou bien encore d'une image de couleur unie, jouant le rôle de couleur de fond du film d'animation.

3.2 Les attributs de séquence

La manipulation des séquences s'effectue par la description de paramètres spécifiques caractérisant le comportement de chacune d'elles. Ces paramètres

constituent, à part entière, des attributs et peuvent être classifiés de la même manière que les attributs des acteurs plans, à savoir :

- attributs morphologiques,
- attributs géométriques,
- attributs d'aspect.

Ces attributs caractérisent le comportement de chaque séquence du film d'animation.

3.2.1 Les attributs morphologiques

La morphologie d'une séquence peut être assimilée à la suite des images qui la composent. L'attribut morphologique qui en découle est le choix du numéro d'une image de la séquence.

Cet attribut s'applique aussi bien aux séquences internes qu'aux séquences externes (excepté la caméra vidéo).

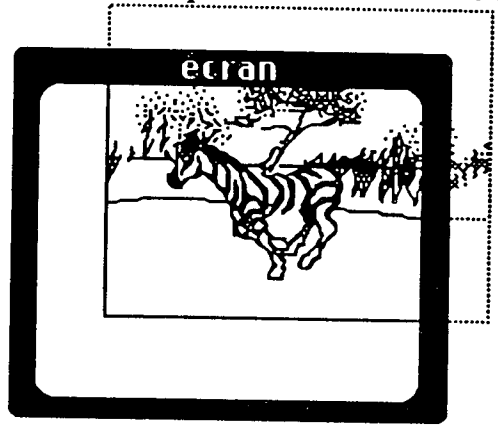
En interne, l'indication d'un numéro d'image provoque la reconstitution de cette image en direct (par superposition des cells correspondants). En externe, l'indication d'une image conduit à la recherche de cette image sur le support vidéo externe.

3.2.2 Les attributs géométriques

De la même manière que l'on applique des opérateurs géométriques aux cells d'une scène, on peut appliquer des opérateurs géométriques aux images d'une séquence. Par défaut, une image d'une séquence coïncide exactement avec l'écran de visualisation. L'application d'opérateurs géométriques permet de réaliser le placement des séquences les unes par rapport aux autres ainsi que la prise de vue de ces séquences.

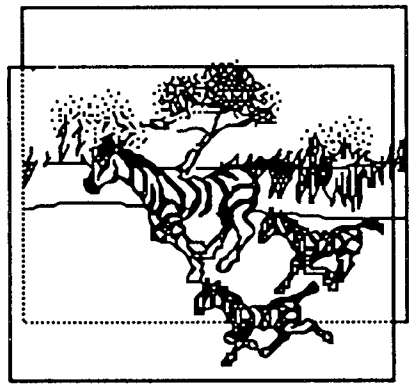
On peut introduire les attributs suivants :

- translation 2D de la séquence relativement à l'écran,



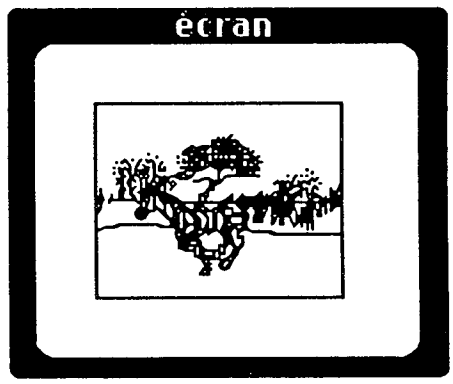
Translation d'une séquence.

- priorité de la séquence, définissant globalement l'ordre de superposition des séquences du film.



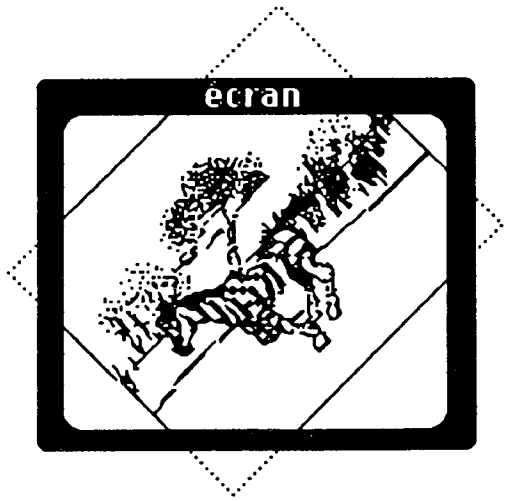
Superposition de 2 séquences.

- zoom de la séquence, identique en x et y.



Zoom d'une séquence.

- rotation 2D de la séquence dans le plan de l'écran.



Rotation d'une séquence.

3.2.3 Les attributs d'aspect

Cette notion intègre essentiellement l'idée de transparence d'une séquence, qui permet, entre autres, d'effectuer les effets de fondus enchainés d'une séquence à une autre.

Il s'agit également de la couleur de séquence unie, qui joue le rôle de la couleur de fond du film, et qui permet, entre autres, d'effectuer les effets de fondus à une couleur pour réaliser l'enchainement de plusieurs séquences.

4. DESCRIPTION D'UN SCENARIO

Le scénario d'un film d'animation correspond à la description temporelle des attributs de tous les acteurs séquences du film.

4.1 Le temps

Le système proposé est un système temps réel, qui autorise la manipulation interactive des séquences, créées par le système de description de scènes.

Il intègre le pilotage d'appareils vidéo et gère les points de montage dûs aux contraintes du matériel vidéo (prérolls, recherche...) et aux caractéristiques propres au système.

L'unité temporelle est l'image, c'est-à-dire le 25ème de seconde, et le réalisateur peut manipuler tous les attributs des séquences à chaque image du film d'animation. Le temps de description représente le temps absolu du film.

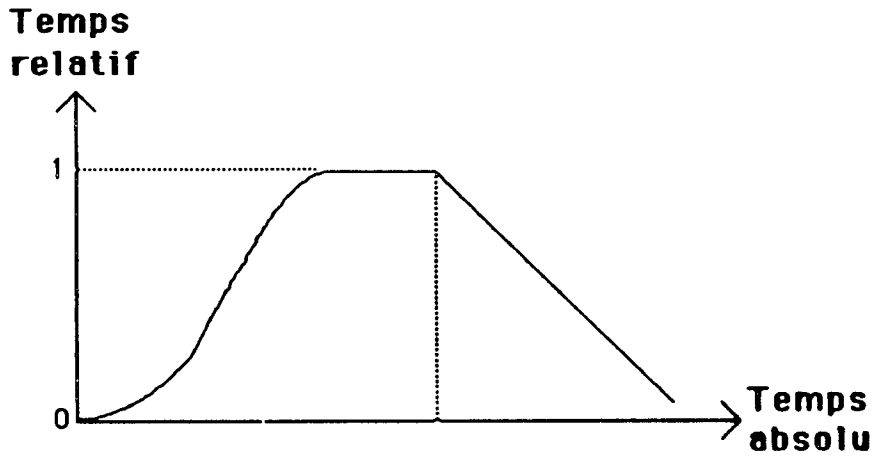
La manipulation des entités temporelles que constituent les séquences nécessite l'introduction d'un temps relatif à la séquence. Pour une séquence interne, ce temps correspond au temps de description de la scène qui a conduit à la séquence. Pour une séquence externe, le temps relatif correspond aux numéros des images de la bande vidéo contenant la séquence. Ce numéro peut être déterminé en comptant les signaux de synchros lors des défilements de la bande relativement à une origine, appelée encore point de calage. Etant données les erreurs que peuvent engendrer les glissements de bande ou les défauts de celle-ci, il est beaucoup plus sûr de lire directement sur la bande le numéro de chacune des images. Le temps relatif de la séquence correspond alors à ce que l'on appelle le temps codé sur la bande vidéo.

4.2 Evolution morphologique

Cette évolution traduit l'enchainement des différentes images composant la séquence d'animation.

Elle peut être représentée sous la forme d'un graphe établissant une correspondance entre le temps absolu du film et le temps relatif à la séquence. A tout instant, les système est alors capable de retrouver l'image qui correspond à chacune des séquences du scénario.

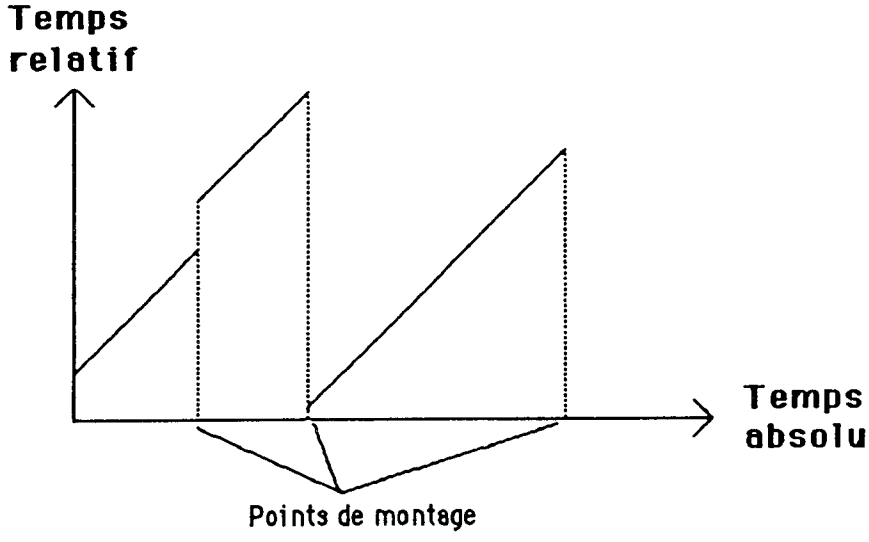
Pour une séquence interne, le graphe d'évolution peut prendre l'allure d'une courbe quelconque, intégrant par exemple des portions discrètes, linéaires ou même lissées.



Evolution d'une séquence interne.

En revanche, pour une séquence externe, le graphe est directement lié aux possibilités du matériel vidéo constituant la source de la séquence.

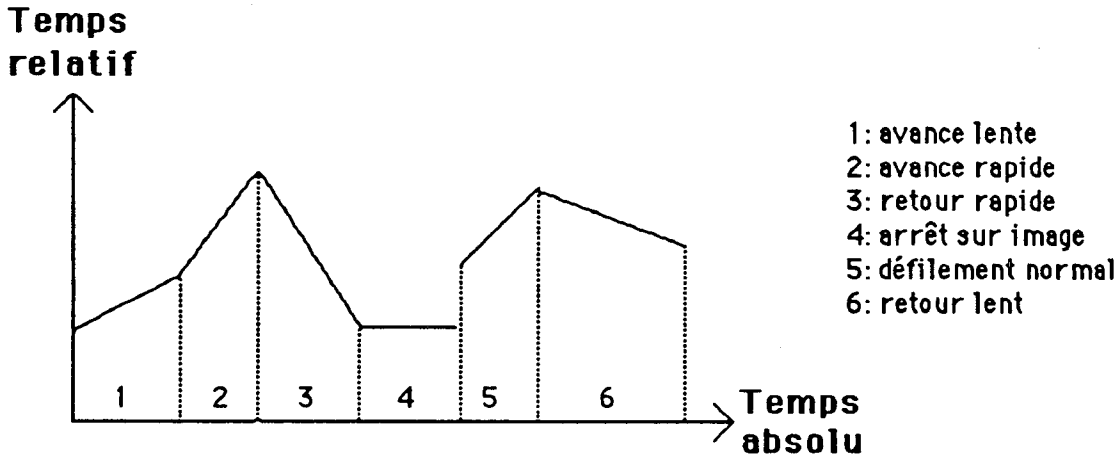
Ainsi pour un magnétoscope, la seule fonction viable étant le défilement de la bande à vitesse normale, le graphe devra être impérativement composé de segments de droite de pente 1. L'origine de chaque segment de droite détermine un point de synchronisation, qui correspond nécessairement à un point montage lors de l'enregistrement.



Evolution d'un magnétoscope.

En revanche, pour un vidéo-disque ou un disque dur vidéo, l'arrêt sur image, les ralentis, les accélérés et les défilements arrière à vitesse variable, ne dégradent

pas la qualité de l'image. Le graphe d'un tel appareil peut alors être constitué d'un ensemble de segments de droite quelconque. Lorsque l'accès à une image ne peut s'effectuer en temps réel, il est nécessaire de réaliser un point de montage. Ceci dépend totalement des caractéristiques du matériel vidéo.



Evolution d'un acteur vidéo-disque.

4.3 Evolution géométrique

Les attributs géométriques d'une séquence externe s'appliquent directement sur l'image vidéo provenant de la source extérieure.

En revanche, une séquence interne est le résultat de la superposition de plusieurs niveaux de cells possèdant leur propre animation.

Le cell est la seule entité du système d'animation à avoir une correspondance directe avec les ressources matérielles. La gestion des séquences d'animation s'effectue donc nécessairement par le contrôle de l'animation de tous les cells de ces séquences. Il est donc nécessaire de ramener l'animation des séquences au niveau des acteurs plans intervenant dans leur description.

Après l'introduction d'un repère propre à chaque séquence, nous étudierons donc l'évolution de chaque attribut géométrique des séquences ainsi que leur répercussion au niveau de l'animation des acteurs plans.

4.3.1 Repère associé à une séquence

La manipulation des images des séquences conduit à la définition d'un repère propre à chaque séquence. L'origine de ce repère doit être choisi, pour faciliter les traitements géométriques sur les images (centre de rotation de l'image, centre de dilatation...), et pour rendre indépendante la description de ces attributs.

Contrairement aux cells manipulés pour la description de scène, les images d'une séquence ont toutes la même taille, la taille de l'écran, et possèdent donc toutes la même origine. Les coordonnées de ce point sont exprimées une fois pour toutes par rapport au repère des images.

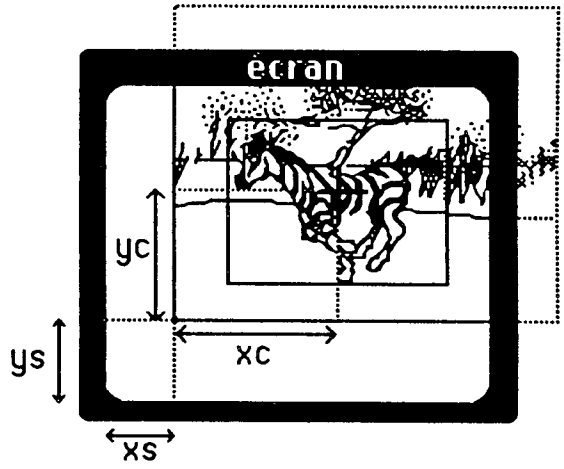
4.3.2 Position

La position d'une séquence est déterminée par les coordonnées de l'origine du repère associé aux images dans le repère associé à l'écran.

De la même manière que l'attribut de position des cells, l'attribut de position de séquence induit la notion de trajectoire 2D, correlée avec une fonction d'évolution, qui traduit la progression de la séquence le long de cette trajectoire.

Cet attribut se compose directement avec les attributs de position de tous les cells de la séquence. En fait la translation à appliquer à un cell d'une séquence est égale à la translation nécessaire au positionnement du cell relativement au repère de la séquence, à laquelle on ajoute le déplacement induit par le positionnement de la séquence relativement à l'écran:

cell/écran = cell/séquence + séquence/écran.



Positionnement d'un cell d'une séquence en translation.

Soient (xc,yc) les coordonnées du point-guide d'un cell relativement à l'origine de la séquence et (xs,ys) les coordonnées de la séquence relativement au repère de l'écran.

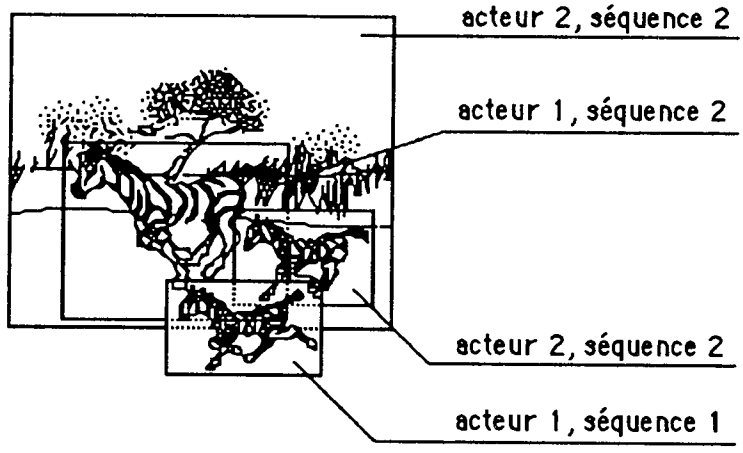
Les coordonnées (Xc,Yc) d'un cell relativement au repère de l'écran sont déterminées comme suit :

Xc = xs + xc;
Yc = ys + xc;

4.3.3 Priorités

A chaque séquence est attribué un rang dans le mécanisme de superposition des séquences.

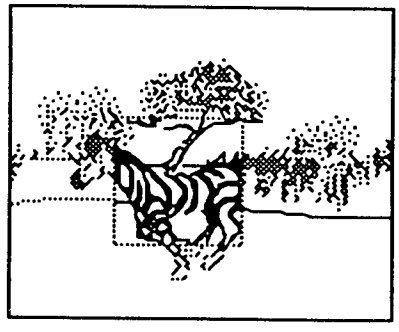
La connaissance de ce rang ainsi que l'ordre de superposition des cells à l'intérieur de la séquence, permet de déterminer globalement un ordre de superposition de tous les cells de l'animation, indépendamment de la séquence dans laquelle ils sont définis.



Superposition des cells de 2 séquences.

4.3.4 Volet

L'attribut de volet peut être vu comme un effet de fenêtrage appliqué à tous les cells de la séquence. Cet attribut se compose directement avec l'effet de fenêtrage propre à chaque cell de la séquence. Seuls sont visibles les pixels d'un cell appartenant à la fois au cell courant, à la fenêtre courante et au volet global.



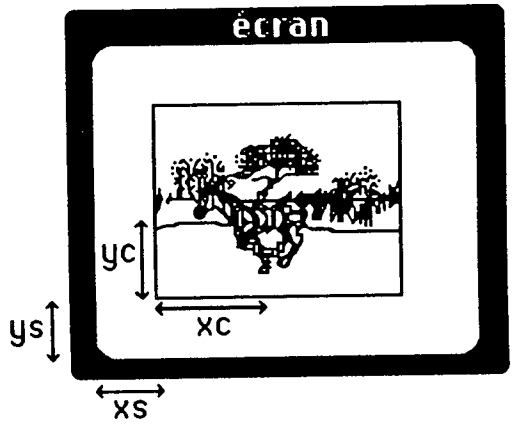
Effet de volet sur une séquence.

L'extension de l'effet de volet correspond à la définition d'un masque, permettant de ne visualiser une séquence qu'au travers un motif, ou en dehors de ce motif.

4.3.5 Zoom

L'attribut de zoom permet de traduire l'éloignement de la caméra par rapport à la séquence. Un zoom correspond à une dilatation autour de l'origine du repère de la séquence.

Cet attribut ne se compose pas seulement avec l'attribut de zoom du cell. En effet, les coordonnées du point-guide du cell et de l'origine du repère de la séquence n'ayant aucune corrélation, l'introduction d'une dilatation au niveau de la séquence se traduit par une dilatation du cell autour de son point-guide, d'un facteur égal au produit du facteur de zoom de la séquence par le facteur de zoom du cell, composée avec une translation de ce cell.



Zoom et translation d'une séquence réduite.

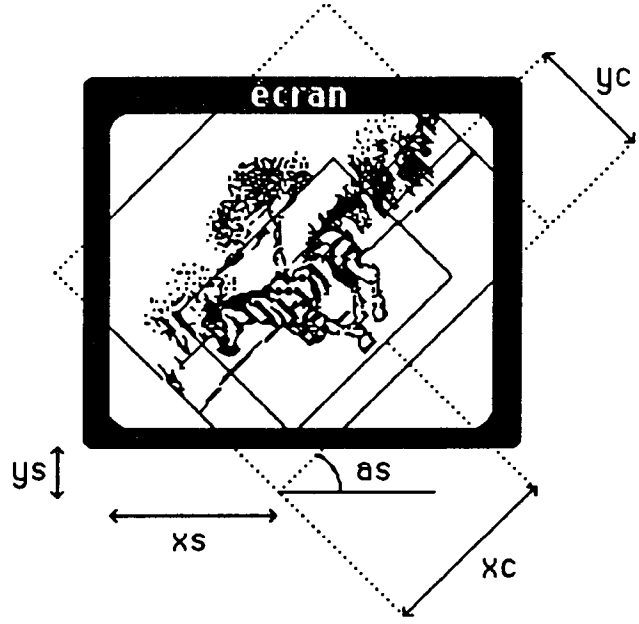
Soient (xc,yc) les coordonnées du point-guide d'un cell relativement à l'origine de la séquence, (xs,ys) les coordonnées de la séquence relativement au repère de l'écran et kx,ky les facteurs de zoom de la séquence.

Les coordonnées (Xc,Yc) d'un cell relativement au repère de l'écran sont déterminées comme suit :

$$Xc = xs + kx*xc;$$
$$Yc = ys + ky*yc;$$

4.3.6 Rotation

L'introduction d'un angle de rotation au niveau de la séquence se traduit au niveau de chaque cell par une rotation, dont l'angle est égal à la somme des angles de rotation de la séquence et de l'angle de rotation du cell, composée avec une translation du cell.



Rotation et translation des cells d'une séquence en rotation.

Soient (xc,yc) les coordonnées du point-guide d'un cell relativement à l'origine de la séquence, (xs,ys) les coordonnées de la séquence relativement au repère de l'écran et as l'angle de rotation de la séquence autour de son origine.

Les coordonnées (Xc,Yc) d'un cell relativement au repère de l'écran sont déterminées comme suit :

```
Xc = xs + xc * cos(as) - yc * sin(as);
Yc = ys + xc * sin(as) + yc * cos(as);
```

4.4 Evolution d'aspect

4.4.1 Transparence

L'évolution du coefficient de transparence des séquences d'animation permet de traduire des effets tels que fondu enchainé. Il suffit, en effet, de disposer de 2 séquences superposées, la seconde séquence étant initialement transparente.

Le passage du coefficient de transparence de 0 à 1 de la 1ère séquence, correlé avec le passage du coefficient de transparence de 1 à 0 de la seconde, permet de passer de la 1ère à la seconde par un fondu enchainé.

4.4.2 Couleur

La disposition d'un fond de couleur (séquence dégénérée) permet de traduire des effets de fondu à une couleur quelconque. Il suffit en effet de disposer de 2 séquences superposées sur la séquence de fond, la seconde séquence étant initialement transparente. Le passage du coefficient de transparence de 0 à 1 de séquence fait disparaitre progressivement cette séquence sur le fond d'animation. Puis le passage du coefficient de transparence de 1 à 0 de l' autre séquence permet de passer progressivement de la couleur de fond à cette nouvelle séquence.

Cet effet n'est pas limité au noir et la couleur de fond peut même évoluer pendant le fondu.

Conclusion

Ce chapitre nous a conduit à l'élaboration des concepts manipulés par le système d'animation proposé dans cette étude. Le principal objectif était de séparer la génération des images de leur animation, afin de décrire interactivement l'animation et de la visualiser en temps réel. L'implémentation du concept de base que constitue le cell et sa dotation d'attributs 2D temps réel permet la réalisation interactive d'animations 2D ou 3D. L'indépendance de ces attributs permet en outre à l'animateur de décrire et de mettre au point des séquences d'animation de façon structurée et progressive, en gardant toujours le contrôle de tous les paramètres de l'animation. La décomposition des attributs de la séquence au niveau même du cell permet également de réaliser le montage du film d'animation interactivement et en temps réel.

La suite de l'étude va présenter l'implémentation matérielle des concepts énoncés dans ce chapitre ainsi que les réalisations logicielles fondées sur ces principes.

CHAPITRE 4 : LES CONCEPTS MATERIELS

Introduction

Sans matériel spécifique il ne peut y avoir d'animations, et encore moins d'animations en temps réel. Les sociétés spécialisées dans la synthèse d'images sont amenées de plus en plus à développer du matériel spécifique, adapté à leurs besoins, surtout pour l'animation.

Après avoir défini les objectifs d'un système dédié à l'animation et passé en revue les différents concepts abordés au chapitre précédent, nous décrirons la solution adoptée par la société Gétris Images.

1. CONTEXTE ET OBJECTIFS

1.1 Présentation

La puissance d'un système d'animation peut s'exprimer à partir de 2 grandeurs complémentaires, la puissance de calcul et la puissance de visualisation. Les différentes classes de systèmes d'animation sont représentées dans le tableau suivant :

Visualisation / Calcul	Modification d'une image <= 2ms	Modification d'une image > 2ms
Calcul d'une image <= 20ms	Animation Directe en Temps Réel	Animation Différée Image par Image
Calcul d'une image > 20ms	Animation Différée en Temps Réel	Animation Différée Image par Image

Classification des systèmes d'animation.

Rappelons que 20 ms correspond à la durée d'une trame et 2 ms à la durée d'un retour de trame.

Deux classes principales se distinguent, l'animation en direct et l'animation différée, encore appelée play-back.

L'animation en direct n'a d'intérêt que si elle a lieu en temps réel. En revanche, l'animation différée trouve 2 styles d'application, le temps réel ou l'image par image.

1.2 Animation en direct

Très peu de systèmes sont susceptibles d'éxecuter une animation en direct et en temps réel. Ce type d'animation nécessite en effet des moyens de calcul considérables qui ne sont réellement justifiés que pour la simulation de conduite, pour laquelle l'interaction doit être puissante et immédiate.

Toutefois, des systèmes 3D temps réel, commercialisés par exemple par les sociétés TELMAT ou APOLLO, proposent aux graphistes des outils de description interactifs d'objets 3D très puissants. Cette interaction sur l'image 3D réaliste constitue un atout essentiel pour les sociétés spécialisées dans la production d'animations 3D.

Ces systèmes sont cependant trop spécialisés dans l'animation 3D, une animation 2D devant en effet être décrite en 3 dimensions. De plus, l'investissement nécessaire à l'acquisition d'un tel système est à l'image de sa complexité et repousse de nombreuses sociétés, qui trouvent un compromis dans le choix de systèmes d'animation différée, proposant par exemple une animation temps réel en fil de fer lors de la description de l'animation.

Enfin, les systèmes d'animation 2D en temps réel sont également trop limités et d'une qualité trop discutable (tracé de polygônes unis).

1.3 Animation différée

Le principe même de l'animation différée consiste en la séparation de la génération des images à animer et de leur propre animation.

1.3.1 Animation différée image par image

Cette technique consiste à prendre le temps nécessaire au calcul de chaque image et à sa visualisation sur une console, puis à enregistrer celle-ci au moyen d'une caméra, si le support désiré est le film, ou d'un vidéo-disque ou magnétoscope, si le support désiré est la vidéo. Cet enregistrement nécessite en général un point de montage (cas de magnétoscope...) et le temps nécessaire à l'enregistrement du film d'animation peut être considérable. Par exemple, pour un magnétoscope dont la durée de préroll est de 10 secondes, le montage d'une seconde de film (soit 25 images), nécessite 4 minutes de montage, si l'on ne tient pas compte des temps de positionnement ni de calcul.

L'animateur ne peut juger de l'animation qu'après le montage total de la séquence, et n'a surtout aucun moyen d'interagir sur cette animation. La moindre modification conduit irrémédiablement à la regénération de la séquence complète (calcul des images et enregistrement).

1.3.2 Animation différée en temps réel

Encore appelée play-back temps réel, cette technique nécessite une technologie susceptible de pouvoir modifier le contenu de la mémoire de trame ou le résultat de la visualisation en temps réel, c'est-à-dire pendant chaque retour de trame de l'animation. Les images constituant le film sont précalculées au rythme imposé par les calculs et sont ensuite mémorisées, avant d'être visualisées en temps réel.

Cette technique se rapproche de l'animation différée image par image dans le cas précis d'utilisation d'un matériel de stockage vidéo très performant, tel qu'un disque dur vidéo, couplé avec une carte d'acquisition vidéo. Ce matériel dispose de fonctionnalités interessantes telles que ralentis, arrêts sur image et retours arrière sans dégradation de l'image. La seule interaction possible s'effectue donc sur le rythme global de la séquence mais pas sur son contenu.

Les images peuvent également être stockées sur une unité de disque rapide. Cette unité doit être capable dans un deuxième temps d'alimenter la mémoire de trame à une vitesse suffisamment rapide pour assurer la continuité de l'animation. De tels systèmes utilisent en général des algorithmes de compression des informations pour accélérer le transfert. Il va de soi que le taux de compression est inversement proportionnel au réalisme de l'image.

Une autre solution consiste à interagir directement sur la visualisation des images indépendamment de leur contenu. Ceci nécessite un processus de visualisation sophistiqué intégrant des possibilités temps réel telles que, par exemple, le zoom.

1.4 Objectifs

L'objectif principal de cette étude est la proposition d'un système d'animation différée en temps réel basé sur le principe de décomposition d'une animation en différents cells. Ces cells sont tous générés à l'avance, en vue de leur manipulation et de leur animation en temps réel.

Le système proposé permet donc une description interactive de scène avec contrôle immédiat de l'attitude des acteurs de l'animation. La visualisation de la séquence s'effectue en direct et en temps réel. Ce système permet de plus une description interactive du scénario d'animation avec contrôle immédiat des différentes séquences. L'animation du film s'effectue également en direct et en temps réel.

Nous allons reprendre les concepts élaborés au chapitre précédent afin de déterminer une architecture originale disposant de fonctionnalités temps réel adaptées.

2. IMPLEMENTATION DES CONCEPTS DE LA SCENE

Une scène regroupe plusieurs acteurs plans susceptibles de disposer de leur propre animation. Nous allons établir dans les paragraphes suivants une correspondance directe entre les différentes classes d'attributs de ces acteurs et les fonctionnalités matérielles idéales d'un système d'animation de cells en temps réel.

2.1 Attributs morphologiques

Un acteur plan est caractérisé par un ensemble de cells, constituant sa morphologie. La composition, à un instant de l'animation, des cells de tous les acteurs plans de l'animation détermine une image de la séquence. Différentes techniques de composition vont être présentées. Elles sont directement liées au procédé de stockage des cells de l'animation.

2.1.1 Stockage des cells

On peut distinguer 4 grands principes de stockage des cells d'une animation:

- les cells peuvent être stockés sur disque dur (non vidéo). Les chargements successifs des cells des différents acteurs plans de l'animation, dans l'ordre de superposition correspondant à la scène (le premier chargé est l'acteur de fond, le dernier est l'acteur situé au premier plan) permet de composer les cells et donc de générer l'image correspondante. Cette composition ne peut s'effectuer en temps réel que dans des cas bien précis : nombre d'acteurs limités, compression importante des cells...

- les cells peuvent être stockés dans une mémoire vive (non vidéo) dans un format image, et transférés successivement dans une mémoire de trame. Cette solution, présentée par la société américaine QUANTEL, nécessite un certain nombre de traitements très rapides sur les cells (rotation 2D, distortion...) ainsi que des opérateurs spécialisés pour le transfert dans la mémoire de trame. Dans le cas où la technique ne permet pas le transfert des cells pendant le retour de trame, 2 mémoires de trame sont utilisées alternativement (cf §2.1.3 du 1er chapitre). Les cells sont transférés dans la mémoire de trame non active pendant la durée d'une trame complète. Le principal inconvénient d'une telle technique réside dans le fait que le rythme de l'animation est directement lié à la taille des cells ainsi qu'à la complexité des transformations qui s'y rapportent. La fréquence d'animation peut être ramenée, par exemple, à 10 images par seconde dans le cas où celle-ci est trop complexe.

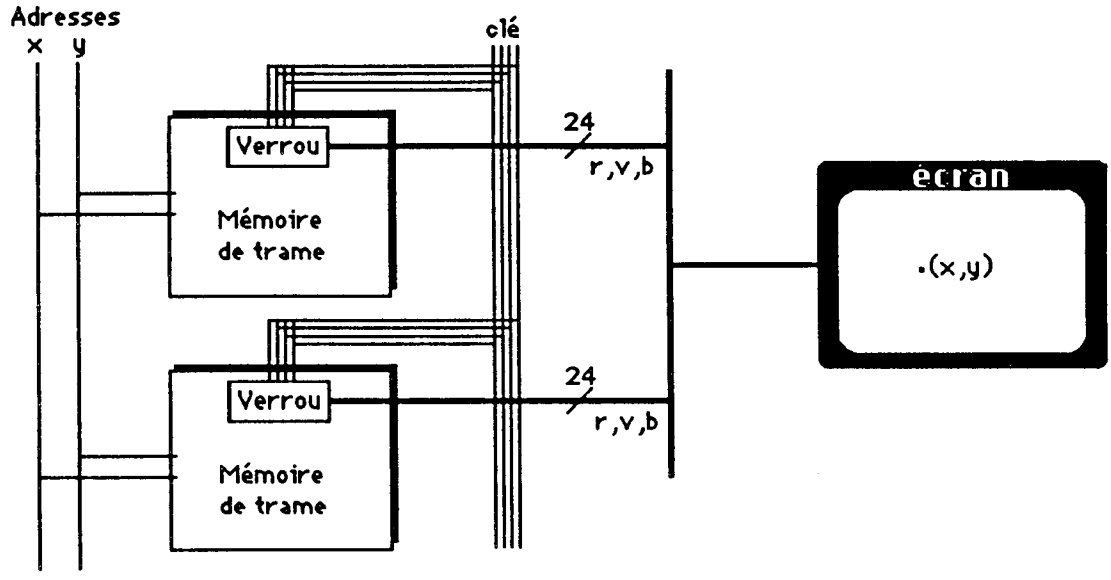
-l'apparition de matériels vidéo très performants permet d'utiliser ces supports vidéo, non plus seulement dans la phase d'enregistrement final du film d'animation, mais aussi dans la phase de stockage des différents cells en vue de leur animation. C'est le cas notamment des disques durs vidéo numériques de la société ABEKAS, qui propose des fonctionnalités temps réel très interessantes. En plus des fonctions de pilotage temps réel ou presque (temps d'accès à une image de 0 à 5 trames suivant l'éloignement des têtes), ce type de matériel permet la combinaison de plusieurs images en temps réel avec des possibilités d'effets (transparence, translation...). Le principal inconvénient de ces disques (hormis le prix) réside dans leur capacité d'enregistrement (en général 1 minute d'images, soit 15 secondes pour 4 couches de cells).

- une autre solution, adoptée par les sociétés PIXAR et GETRIS IMAGES, consiste à stocker les cells de l'animation directement dans la mémoire vidéo du système, qui se décompose alors en plusieurs mémoires de trame superposées. Cette technique va totalement dans le sens de la séparation de la génération des images de leur visualisation, ces 2 processus étant alors totalement indépendants. Le contenu des mémoires de trame n'est en effet jamais modifié. C'est le balayage de ces mémoires à la fréquence vidéo qui modifie l'affichage à l'écran.
La suite de notre étude se place dans le cadre d'un tel système, qui a 2 corollaires immédiats, d'une part la gestion des profondeurs associées aux mémoires de trame, et d'autre part le partage d'une mémoire par plusieurs cells d'un même acteur.

2.1.2 Gestion des profondeurs

La visualisation de plusieurs mémoires de trame nécessite l'introduction d'un mécanisme de **gestion des profondeurs** de ces différentes mémoires. Ce mécanisme doit être capable de déterminer en tout pixel de l'écran la mémoire de trame de laquelle doit provenir l'information de couleur à visualiser.

La détermination de la provenance d'un pixel d'une ligne de l'écran s'effectue en 2 étapes. Tout d'abord, chaque mémoire de trame indique si elle dispose d'une information visible ou non. L'ensemble de ces indications détermine une combinaison clé, qui est alors communiquée à toutes les mémoires de trame. Chacune de ces mémoires dispose d'un code spécifique et unique, jouant le rôle d'un verrou, et la mémoire, dont le code correspond à la combinaison clé, transmet la couleur du pixel traité. Ce traitement est effectué au rythme imposé par le balayage vidéo (13,5 Mhz), ce qui signifie que la combinaison des mémoires de trame est réalisée en temps réel, indépendamment du contenu de ces mémoires.

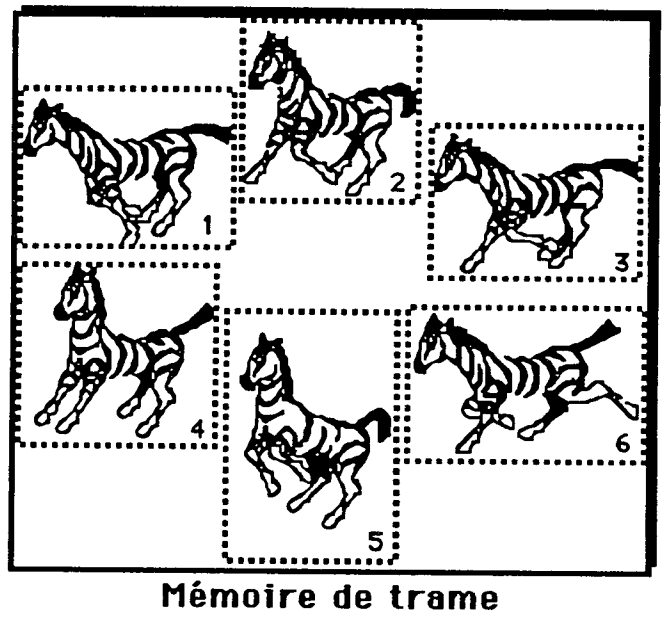


Superposition des mémoires de trame.

2.1.3 Stockage de l'album

Comme on l'a vu au §2.2.3 du chapitre précédent, l'album d'un acteur plan regroupe toutes les informations concernant ses cells. Il est clair que le nombre des mémoires de trame d'un système d'animation est limité. Une mémoire ne peut donc servir au stockage d'un seul cell mais doit être partagée par plusieurs cells de l'animation.

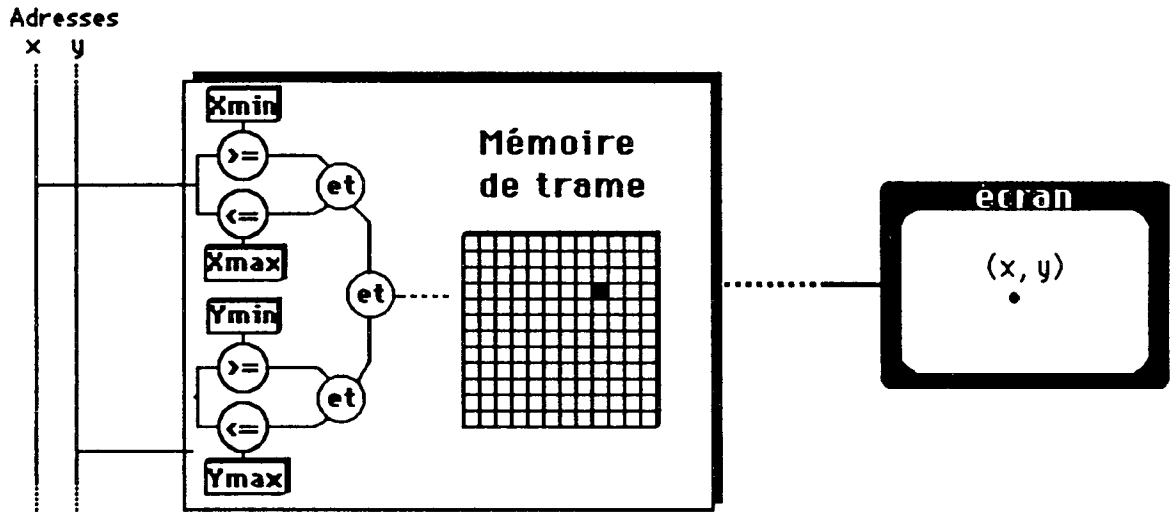
A un instant de l'animation un acteur plan est représenté par un cell et un seul, ce qui nous conduit à particulariser chaque mémoire de trame pour un acteur plan de l'animation. Cette mémoire de trame constitue alors le support physique de l'album de cet acteur. Les cells ont alors une réalité physique, en ce sens qu'ils représentent chacun un dessin sur une portion de mémoire de trame. Les caractéristiques de chaque cell sont exprimées dans le repère propre à la mémoire de trame.



L'album d'un acteur chargé dans une mémoire de trame.

D'un point de vue matériel, il est nécessaire de ne visualiser qu'un seul cell d'une mémoire de trame à un instant de l'animation. Un mécanisme câblé doit donc permettre, au cours de l'animation d'un acteur plan, de rendre invisible le cell précédemment visualisé et de rendre visible un nouveau cell.

A chaque mémoire de trame est donc associé un **effet de fenêtrage** matériel (clipping) permettant de ne visualiser qu'une portion de cette mémoire. L'implémentation de quatre registres de fenêtrage (Xmin, Xmax, Ymin, Ymax) couplés avec des comparateurs en x et en y au niveau de chaque mémoire de trame permet facilement de ne visualiser qu'une portion rectangulaire de chaque mémoire. Le balayage horizontal de la mémoire ne délivre alors d'informations valides que si l'indice de la ligne parcourue est compris dans l'intervalle défini par les registres de lignes. De plus, pour chaque ligne valide, seules les informations d'abscisses comprises dans l'intervalle défini par les registres de colonnes sont valides, à condition bien sûr qu'elles correspondent à un pixel visible du cell délimité par le fenêtrage matériel.



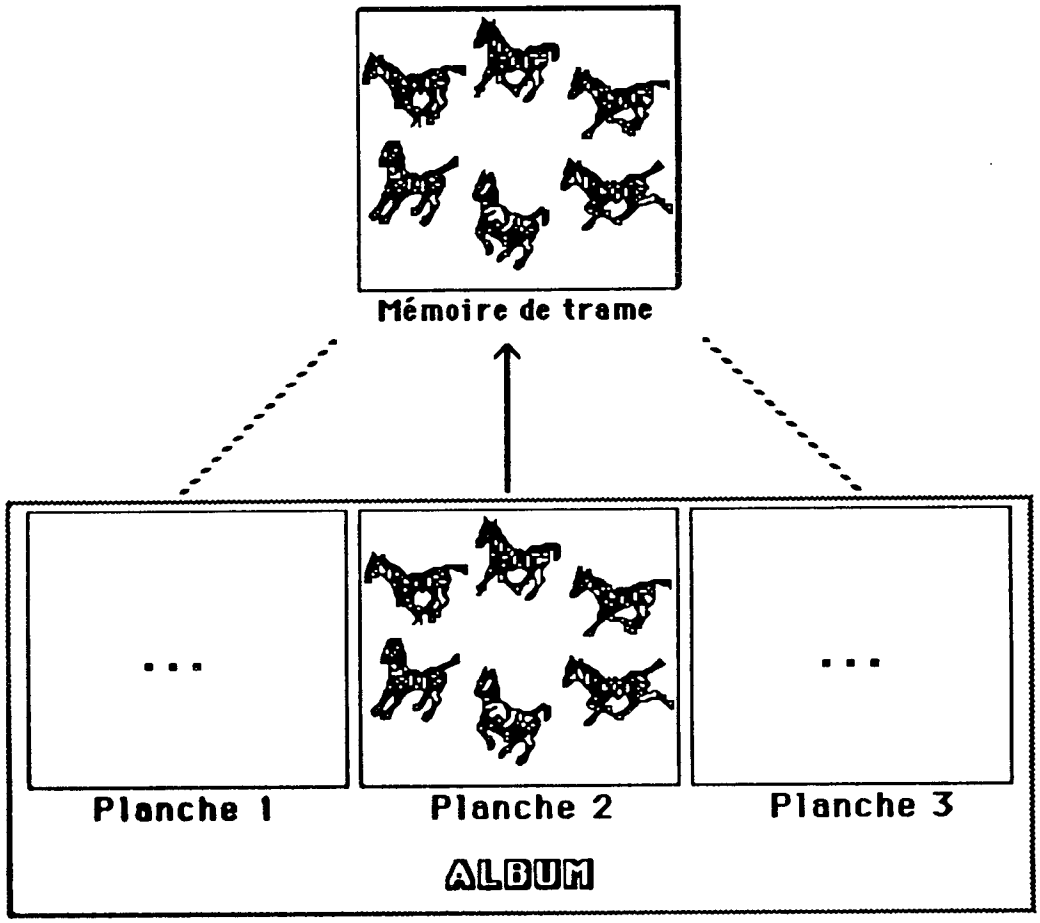
Implémentation de registres de fenêtrage.

L'écriture des quatre registres de fenêtrage est réalisé sans peine pendant chaque retour de trame. Il donc possible de "passer" d'un cell à un autre en temps réel.

Cet effet de fenêtrage est indépendant du mécanisme de gestion des profondeurs, et est appliqué indépendamment sur chacune des mémoires de trame avant leur combinaison en temps réel.

2.1.4 Les planches d'animation

Une mémoire de trame a une capacité de stockage limitée (dans notre cas 1024*1024 pixels). Il peut donc arriver de ne pouvoir charger qu'une partie de l'album dans une mémoire de trame (le nombre des cells est trop important ou les cells ont une trop grande taille). Cette remarque nous conduit ainsi à définir la notion de **planche** d'animation, regroupant une partie des cells d'un acteur plan et coïncidant exactement avec la capacité d'une mémoire de trame. L'ensemble des cells d'un acteur plan peut donc être réparti sur plusieurs planches d'animation.



Les 3 planches de l'album d'un acteur.

Les planches sont sauvegardées sur disque dur et chargées dans la mémoire de trame associée à l'acteur, lorsque le cell courant de l'acteur n'appartient pas à la planche couramment chargée. Ce chargement ne peut pas s'effectuer en temps réel et nécessite, s'il survient en cours d'animation, un arrêt momentané de cette animation et donc la réalisation d'un point de montage lors de l'enregistrement.

2.2 Attributs géométriques

La manipulation en temps réel des attributs géométriques associés aux acteurs plans nécessitent que ces attributs soient implémentés directement au niveau des mémoires vidéo supportant les cells. Nous allons déterminer les fonctionnalités matérielles spécifiques à chacun de ces attributs géométriques.

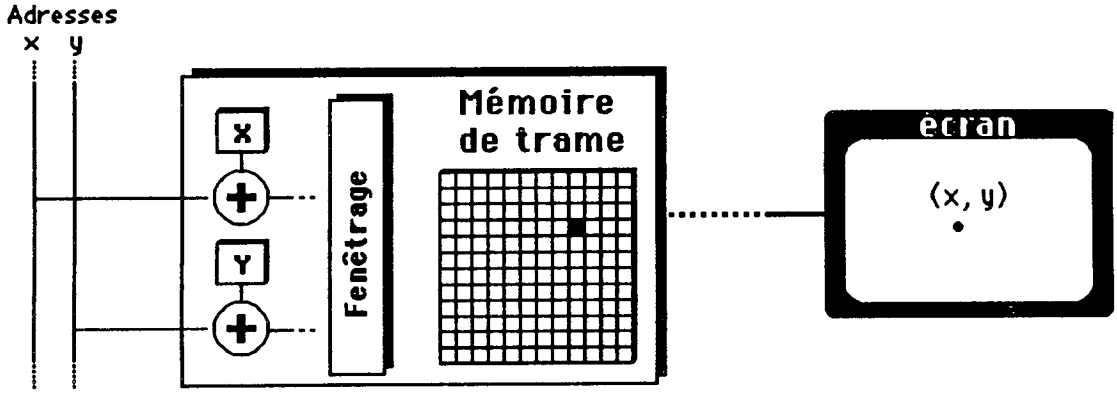
2.2.1 Position

L'attribut de position réalise le placement d'un acteur plan à l'écran, c'est-à-dire le placement du cell courant de cet acteur dans le repère associé à l'écran de visualisation.

Les cells d'un acteur sont stockés dans une mémoire de trame et leurs coordonnées sont exprimées dans le repère associé à cette mémoire. Pour les systèmes de visualisation classiques, ces 2 repères coïncident, c'est-à-dire que le pixel de coordonnées (x, y) dans la mémoire de trame est nécessairement affiché au point de coordonnées (x, y) dans l'écran. Pour le système étudié, les cells d'un acteur plan sont placés dans la mémoire de trame indépendamment de leur position à l'écran (suivant les informations de l'album). Il est donc nécessaire d'introduire un facteur de translation de la mémoire de trame dans le processus de visualisation (même si l'acteur plan ne subit pas de translation par rapport à l'écran).

Soient (xe, ye) les coordonnées du point-guide dans le repère de l'écran,
soient (xi, yi) les coordonnées du point-guide du cell courant dans le repère de la mémoire de trame,
un point de coordonnées (x, y) dans le repère de l'écran aura la couleur du pixel de coordonnées (x + (xi - xe), y + (yi - ye)) dans le repère de la mémoire de trame (en supposant qu'il n'y ait qu'une seule mémoire de trame, intégralement visible).

L'implémentation de l'attribut de translation peut être réalisée facilement par ajout de 2 registres X, Y et de 2 additionneurs au niveau de chaque mémoire de trame. Le balayage d'un point d'indice j sur une ligne d'indice i de l'écran correspond alors à la lecture du point d'indice (j - X) de la ligne d'indice (i - Y) dans le repère de la mémoire, si X et Y expriment les translations verticale et horizontale de cette mémoire par rapport à l'écran.
L'écriture de ces 2 registres peut avoir lieu très largement pendant le retour de trame. La translation d'un cell est un donc réalisée en temps réel.



Translation (-X, -Y) d'une mémoire de trame relativement à l'écran.

Comme on peut le voir sur le schéma, la logique câblée propre au fenêtrage doit intervenir après le traitement de la fonction de translation, puisque celui-ci est exprimé relativement au repère de la mémoire de trame.

2.2.2 Profondeurs

L'ordre de superposition des acteurs plans de l'animation, détermine un rang pour chaque mémoire de trame dans le mécanisme de gestion des profondeurs. Ainsi la modification du rang d'un acteur implique une modification de l'ordre de superposition des mémoires de trame.

Ceci nécessite la détermination de nouveaux codes de priorité spécifiques, puis une écriture dans le registre de verrou de chaque mémoire de trame. Ces écritures peuvent être réalisées pendant un retour de trame, donc en temps réel.

2.2.3 Fenêtre

L'attribut de fenêtrage permet de ne visualiser qu'une portion d'un cell de l'animation, pour traduire par exemple l'interpénétration de 2 acteurs.

Cet attribut a une correspondance directe avec l'effet de fenêtrage matériel évoqué au paragraphe 2.1.3, pour la visualisation d'un cell de l'album. En effet, il correspond à la sélection d'une portion rectangulaire à l'intérieur du cell courant. Il suffit donc de positionner le fenêtrage matériel aux intersections des zones délimitées d'une part par la fenêtre et d'autre part par le cell. Ces calculs d'intersection sont très simples à traiter, et puisque les modifications de la fenêtre matérielle peuvent être réalisées pendant le retour de trame, l'attribut de fenêtre est également un attribut temps réel.

2.2.4 Zoom et boites

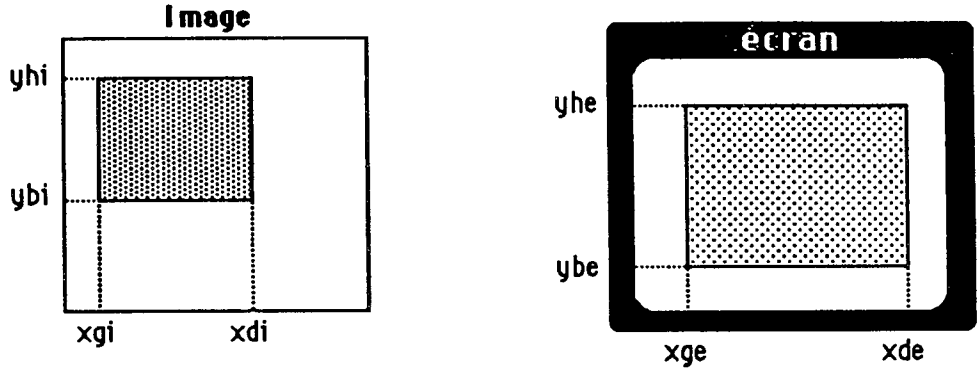
L'attribut de zoom permet de simuler l'éloignement d'un acteur, et évite la génération de cells identiques à un rapport de taille près.

L'implémentation d'un tel attribut au niveau du matériel nécessite le rajout d'une logique câblée pour le balayage des lignes d'une mémoire de trame et pour chaque point de cette ligne. Cette logique doit être capable de répéter plusieurs fois la même ligne et pour chaque ligne balayée de répéter plusieurs fois les mêmes points (cas d'un agrandissement), ou bien au contraire de sauter plusieurs lignes dans le balayage et pour chaque ligne balayée de sauter plusieurs points de cette ligne (cas d'un rétrécissement).

La plupart des sytèmes d'animation disposant d'un zoom câblé propose des facteurs de zoom entiers. Si l'image mémorisée dans la mémoire de trame correspond à un facteur 1, les facteurs applicables à cette image sont des entiers (2,3,4...) voire des puissances de 2 (2,4,8...). Le zoom câblé est en effet réalisé à l'aide de registres de répétition. Ce type de zoom n'est donc pas continu et le rétrécissement d'une image implique que celle-ci soit stockée déjà réduite, d'où une perte de résolution évidente lors d'un agrandissement (effet mosaïque). Un tel zoom ne peut donc être utilisé que pour la mise au point interactive des animations et non pour leur visualisation.

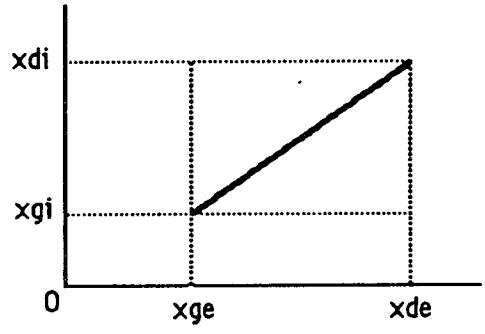
L'utilisation d'un zoom câblé en phase de visualisation nécessite que celui-ci soit continu, c'est-à-dire que le facteur de zoom soit un réel. Il est donc nécessaire de considérer que zoomer un cell à l'écran consiste à remplir au mieux une zone rectangulaire de l'écran avec les pixels de ce cell.

Ceci nous conduit à la notion de **boite**, correspondant à une zone rectangulaire dans un repère propre. Ainsi, un cell est une boite définissant le domaine de la mémoire d'image à visualiser et la visualisation à l'écran du cell courant est elle-même réalisée à l'intérieur d'une boite. Les coordonnées de ces 2 boites dans leur repère propre suffisent à la visualisation.



Boite image et boite écran.

Pour réaliser le mapping, ou encore la projection, de la boite image dans la boite écran, nous avons implémenté 2 processus câblés de répartition [Bre 65] [Bre 82], utilisé pour la génération de droites. L'avantage d'un tel algorithme, rappelé dans l'annexe II, réside dans le fait qu'il ne demande que des comparaisons et des additions. Son câblage demande donc peu de moyens.



Mise en correspondance de l'abscisse des 2 boites.

Les facteurs de zoom en x et y sont donnés directement par les formules :

xde - xge = **kx** * (xdi - xgi) et yhe - ybe = **ky** * (yhi - ybi)

Un processus est chargé de délivrer pour chaque ligne balayée de l'écran, la ligne correspondante dans la mémoire de trame, et l'autre est chargé pour chaque ligne balayée de la mémoire de délivrer les indices des points qui la composent. Ce

traitement a lieu à la fréquence vidéo (13,5 Mhz) et permet de réaliser des zooms continus en temps réel. Les seules écritures à réaliser concernent l'initialisation à chaque retour de trame de l'erreur et des incréments du zoom vertical, et à chaque retour de ligne de l'erreur et des incréments du zoom horizontal, facilement calculés à partir des informations de boites.

L'utilisation de ce style de zoom câblé permet de traiter les symétries par rapport aux 2 axes, conjuguées ou non, ce qui contribue également à limiter le nombre de cells à générer initialement.

Cependant, ce zoom n'est pas antialiasé. Il est en effet basé sur des calculs entiers. Une solution à ce problème consisterait à sur-échantillonner les pixels de l'écran et des mémoires d'image, de calculer les couleurs de chaque sous-pixel de l'écran à partir des sous-pixels de l'image, puis d'effectuer la moyenne des couleurs pour chaque pixel de l'écran. Ceci pose des problèmes délicats tels que l'augmentation de la fréquence interne de balayage vidéo en proportion avec le sur-échantillonnage et l'implémentation d'additionneurs pour la réalisation des calculs. En fait, l'expérience montre que les défauts du zoom n'apparaissent réellement que pour des images tramées ou disposant de motifs très réguliers.

Enfin, l'implémentation de ce type de zoom câblé intègre nécessairement l'attribut de translation, puisque les boites image et écran sont définies en coordonnées absolues, et supprime du même coup la logique câblée évoquée au paragraphe 2.2.1. En revanche, l'effet de fenêtrage peut être conservé tel qu'il a été présenté au paragraphe 2.2.3, à condition qu'il intervienne après le zoom. Celui-ci délivre en effet des indices de lignes et colonnes dans le repère de la mémoire de trame. La comparaison avec les registres de fenêtre permet alors de ne visualiser qu'une portion du cell zoomé.

2.2.5 Rotation

Actuellement peu de systèmes proposent des rotations d'images 2D en temps réel. Les problèmes techniques sont en effet très nombreux et nécessitent l'utilisation d'une électronique volumineuse (et donc coûteuse).

Contrairement aux autres attributs, la rotation ne respecte pas le principe de balayage vertical et horizontal des mémoires de trame. Pour implémenter une solution matérielle, il faudrait en effet gérer le fait qu'une ligne de l'écran peut être constituée d'un ensemble de points appartenant à des lignes différentes de l'image. La détermination de ces points ne pose d'ailleurs aucun problème, puisqu'il suffit d'une multiplication par la matrice de rotation inverse. La contrainte principale réside dans l'accès des points de l'image à la fréquence de 13,5 Mhz. Les mémoires utilisées doivent donc être à accès très rapide et le stockage des cells dans une mémoire vidéo n'est donc pas adapté à ce type de transformation.

Nous n'avons donc pas implémenté l'attribut de rotation dans le système d'animation proposé. Cet attribut doit être traité intégralement par logiciel, et donc pas en temps réel. Dans ce cas, il est beaucoup plus profitable d'intégrer cet

attribut directement dans l'attribut morphologique de changement de cells, c'est-à-dire de générer automatiquement en amont de l'animation tous les cells correspondant à la rotation de l'acteur. Si cette manipulation peut augmenter considérablement le nombre des cells de l'acteur plan, au risque de devoir générer plusieurs planches de cells, la manipulation de ces cells peut ainsi être réalisée en temps réel. Le contrôle de l'animation peut donc avoir lieu en continu et notre 1er objectif se trouve ainsi respecté, à savoir l'animation différée en temps réel.

2.3 Attributs d'aspect

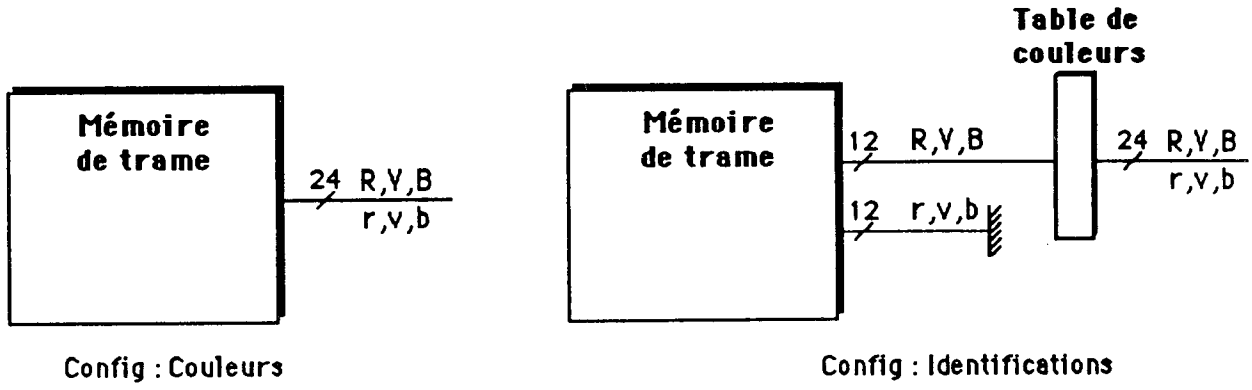
Ces attributs regroupent toutes les informations de couleur et de transparence (% de visibilité) intervenant dans le processus de synthèse des couleurs visibles à l'écran.

2.3.1 Couleur

Les mémoires de trame servent au stockage des informations de couleur des cells des différents acteurs plans de l'animation. Ces informations sont codées sur 24 bits (16,7 millions de couleurs affichables simultanément), soit encore 8 bits (256 niveaux) pour chaque primaire Rouge, Vert et Bleu.

Afin de générer des animations de couleurs en temps réel, il est nécessaire d'introduire une table de couleurs au niveau de chaque mémoire de trame. Chacune de ces mémoires peut alors être vue comme une mémoire de couleurs, visualisable directement à l'écran, ou bien encore comme une mémoire d'identifications permettant d'établir une indirection sur la table de couleurs, qui délivre alors des informations codées sur 24 bits. Il est évident que cette table ne peut disposer de 16,7 millions d'entrées, ce qui corresponderait à une capacité énorme de stockage (l'équivalent de 16 mémoires de trame). Nous avons donc considéré que seuls les 12 bits de poids fort sont significatifs dans la configuration d'identifications, ce qui ne nécessite qu'une table de couleurs à 4096 entrées. Ceci permet toutefois de visualiser 4096 nuances que l'on peut choisir parmi 16,7 millions de couleurs.

L'indépendance des acteurs plans suppose que chacun de ces acteurs dispose d'une table de couleurs spécifique, ce qui revient à associer à chaque mémoire de trame du système une table particulière. En fait la limite du nombre des tables de couleurs peut conduire à partager une même table entre plusieurs mémoires vidéo. Si les poids forts des pixels de ces mémoires sont différents, la table pourra être utilisée de façon indépendante par plusieurs mémoires vidéo, le nombre d'indirections étant bien sûr restreint pour chacune. Sinon, les changements de couleurs opéreront systématiquement sur toutes les mémoires vidéo se partageant la table.



Configurations couleurs et identifications.

2.3.2 Transparence

Comme on l'a vu au §1.4 du 3ème chapitre, la transparence intègre, non seulement les informations relatives au matériau (verre, eau...), mais aussi l'antialiasing indispensable à une bonne qualité d'image.

L'introduction de l'information de transparence nécessite l'extension de la capacité des mémoires de trame du système étudié jusque là. 8 bits de transparence, définissant ce qu'on appelle le **canal alpha**, permettent par exemple de coder 256 niveaux, soit un sur-échantillonage de 16 par 16 pour chaque pixel d'une mémoire.

Ceci demande en outre que le mécanisme de profondeur évoqué ci-dessus soit révisé pour intégrer la notion de transparence, puisque les informations étaient jusque là intégralement visibles ou invisibles. Il est donc nécessaire d'introduire un mécanisme de gestion de profondeurs pondérées, ainsi qu'un opérateur câblé de mélange de couleurs en temps réel.

Considérons un pixel d'une mémoire de trame située en avant de toutes les autres. Soit Ri,Vi,Bi la couleur de ce pixel et Pi le coefficient de transparence qui lui est associé. Soit Rf,Vf,Bf la couleur du pixel de la mémoire située immédiatement en dessous. Les équations de mélange des couleurs sont les suivantes :

$$R = Pi * Ri + (1 - Pi) * Rf;$$
$$V = Pi * Vi + (1 - Pi) * Vf;$$
$$B = Pi * Bi + (1 - Pi) * Bf;$$

En fait, le pixel de couleur Rf,Vf,Bf peut ne pas être complètement opaque et laisser apparaitre à son tour les pixels situés immédiatement en dessous. Nous avons donc étendu le mécanisme de gestion de profondeurs pondérées à 4 niveaux d'images, fournissant chacun une couleur et un coefficient de transparence. Toutes ces informations sont traitées en temps réel par un opérateur câblé de mélange des couleurs, délivrant pour chaque pixel, à la fréquence du balayage vidéo, la couleur à afficher.

Cette architecture permet de générer en temps réel des séquences animation d'une qualité irréprochable et permet la réalisation d'effets temps réel inconnus à ce jour.

3. IMPLEMENTATION DES CONCEPTS DU SCENARIO

Un scénario met en scène plusieurs acteurs, les séquences. Ces séquences sont caractérisées par des attributs spécifiques. Ces attributs ont été directement implémentés au niveau matériel.

Comme on a pu le voir au chapitre 3, la morphologie d'une séquence est l'ensemble des images la constituant. Pour une séquence interne, une image de la séquence est le résultat de la superposition de plusieurs cells d'acteurs. En revanche, pour une séquence externe, une image provient d'une source vidéo extérieure au système. Ceci nous conduit à définir des cartes d'entrée vidéo spécialisées dans l'acquisition de ces séquences et disposant de fonctionnalités temps réel propres.

Les attributs géométriques des séquences internes au système ont une correspondance directe avec les attributs des cells constituant ces séquences. Leur implémentation a été évoqué au paragraphe 2.2 de ce chapitre. Nous n'y reviendrons donc pas.

En revanche, les attributs d'aspect associés aux séquences nous permettent d'introduire de nouvelles fonctionnalités matérielles temps réel.

3.1 Entrée vidéo

La gestion de séquences externes suppose la mise en place de cartes vidéo spécifiques et l'utilisation d'un protocole de communication pour chaque appareil vidéo.

Cette communication s'effectue par une liaison série rapide (38400 bauds) de type RS422. La gestion de plusieurs appareils vidéo en lecture et en écriture nous a conduit à concevoir une carte de liaison série présentant 4 ports.

La carte d'acquisition vidéo permet d'acquérir en temps réel, c'est-à-dire à la fréquence de balayage vidéo, l'image en provenance d'une source extérieure. Deux types d'interface sont autorisés :

- interface analogique R, V, B. Des convertisseurs transforment les signaux analogiques en informations numériques (8 bits par primaire). La manipulation d'informations analogiques pose des problèmes de génération de films, dont la qualité se dégrade progressivement. Un film peut difficilement être enregistré sur support analogique puis réinjecté en fond d'une nouvelle animation.

- interface 4.2.2 numérique. Les informations de couleur sont codées dans le standard Y, U, V. Il est donc nécessaire de convertir ces informations dans le format interne de la machine, c'est-à-dire suivant les 3 primaires R, V, B. Les équations de passage d'un format à l'autre sont facilement calculables à partir des relations suivantes :

$$Y = 0.299 * R + 0.587 * V + 0.114 * B$$
$$U = 0.713 * (R - Y)$$
$$V = 0.564 * (B - Y)$$

et peuvent être directement câblées. Il n'y a dans ce cas aucune dégradation des signaux, et donc aucun problème de génération de films.

Une telle carte ne dispose d'aucune mémoire et les informations sont acquises au fur et à mesure du balayage vidéo. Il est donc impossible, sous cette forme, d'effectuer des transformations géométriques (translation, zoom...) sur l'acquisition vidéo. Seul l'effet de fenêtrage a une réelle signification et permet de ne visualiser qu'une portion de l'image vidéo. Cet effet est réalisé, de la même manière que pour les mémoires de trame, à l'aide de 4 registres et 4 comparateurs.

L'implémentation d'attributs géométriques, pour une séquence externe, nécessite le stockage des informations vidéo. Le couplage du processus d'acquisition vidéo à une mémoire de trame, alimentée en permanence, permet de traiter la source vidéo de la même manière que toute mémoire de trame du système, et autorise donc les mêmes transformations géométriques.

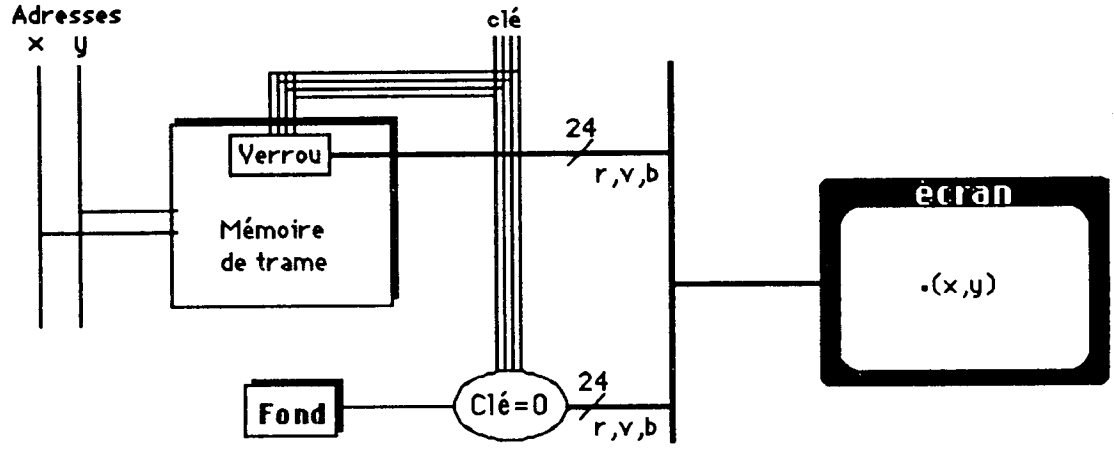
3.2 Attributs d'aspect

3.2.1 Couleur

Nous avons établi au chapitre 3 l'existence de séquences dégénérées. Parmi elles figure la séquence de couleur unie, qui permet, entre autres, d'effectuer les effets de fondu à cette couleur.

En fait, cette séquence peut jouer le rôle de couleur de fond du film. Elle correspond à la couleur affichée à l'écran lorsqu'aucune mémoire de trame ne délivre d'information de couleur. Un simple registre 24 bits suffit à définir cette couleur.

L'introduction de pondération de couleur nécessite que la couleur de fond soit prise en compte dans le mécanisme de gestion des profondeurs pondérées.

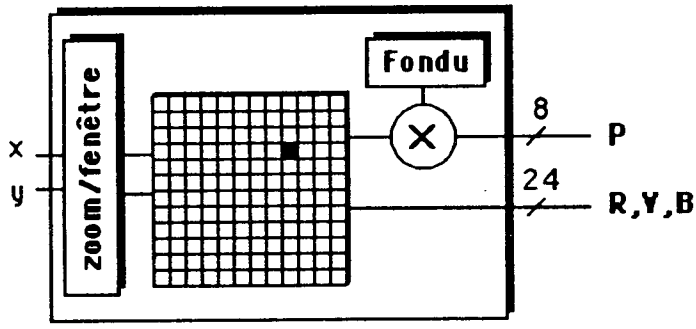


Implémentation d'un registre de couleur de fond.

3.2.2 Transparence

Le passage d'une séquence à une autre par un fondu enchainé consiste tout simplement à rendre progressivement invisible (ou transparente) la séquence située en avant plan. Il en est de même pour le passage d'une séquence à une couleur de fond unie. La diminution progressive de tous les coefficients des mémoires de trame associées à cette séquence permet de réaliser ces effets.

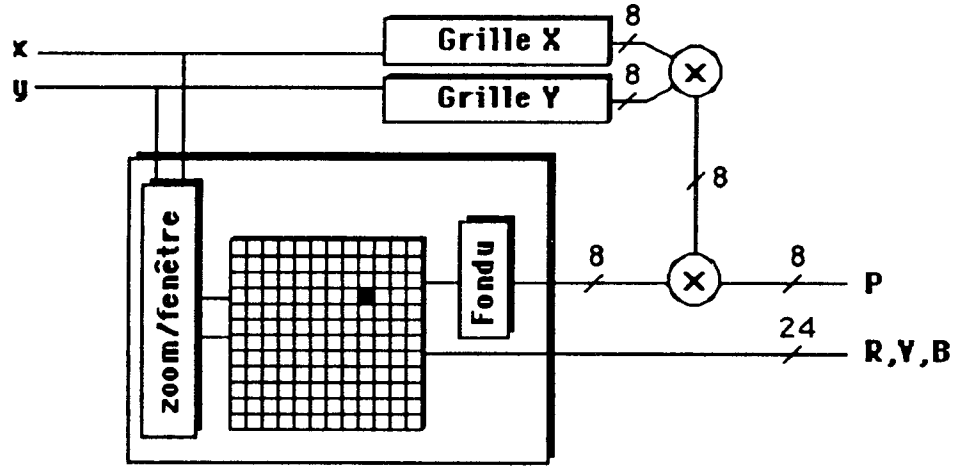
Il est inconcevable de réécrire toutes les valeurs du canal alpha d'une mémoire (cf §2.3.2) pendant un retour de trame. En revanche, l'introduction d'un registre de fondu et d'un multiplieur en sortie de canal de chaque mémoire, pris en compte avant le mécanisme de gestion de profondeur, permet de faire varier en proportion tous les coefficients de cette mémoire, en respectant l'antialiasing et la transparence intrinsèques au cell contenu de la mémoire.



Implémentation d'un registre de fondu.

D'autres transitions peuvent être réalisées par l'implémentation de 2 tables de pondération en X et Y (de 8 bits chacune, soit 256 valeurs), associées aux lignes et aux colonnes de l'écran, et de 2 multiplieurs en sortie du bus de gestion des profondeurs. La pondération de chaque pixel en provenance d'une mémoire de trame est ainsi multipliée successivement par les 2 coefficients contenus dans les tables, pour l'abscisse et l'ordonnée du pixel traité.

L'écriture de motifs particuliers dans les tables permet de réaliser des effets de transition, allant du simple fondu (motifs unis) à un effet de moirage circulaire (motifs sinusoïdaux), etc...



Implémentation de tables de grille.

3.3 Sortie vidéo

Le système de synthèse d'images présenté est destiné à la production de films d'animation de très grande qualité. L'image vidéo finale doit pouvoir être enregistrée sur des supports analogiques ou numériques.

Une carte de sortie vidéo propose donc des sorties analogiques, conversions des primaires R, V, B numériques en signaux analogiques, pour l'enregistrement sur du matériel analogique (magnétoscopes U-Matic, BVU, Bétacam...), et des sorties numériques dans le standard 4.2.2, issues donc de la conversion des primaires R, V, B numériques en composantes Y, U, V numériques, pour l'enregistrement sur du matériel numérique (magnétoscope D1 de Sony, disque dur Abekas...).

Le système s'intégrant dans une chaine d'appareils vidéo, il est nécessaire de synchroniser l'ensemble sur un même signal. Le système doit donc être synchronisé sur un signal externe ou bien fournir lui-même un signal de synchronisation (Genlock).

4. LES SYSTEMES GETRIS

La société Gétris Images est née de l'industrialisation d'un système de synthèse d'images issu de recherches universitaires [Mar 82]. Ce système, alors appelé GETRIS pour "génération en temps réel d'images synthétiques", a eu pour principal objectif la recherche du "temps réel" sur un maximum de fonctions, en particulier pour l'animation.

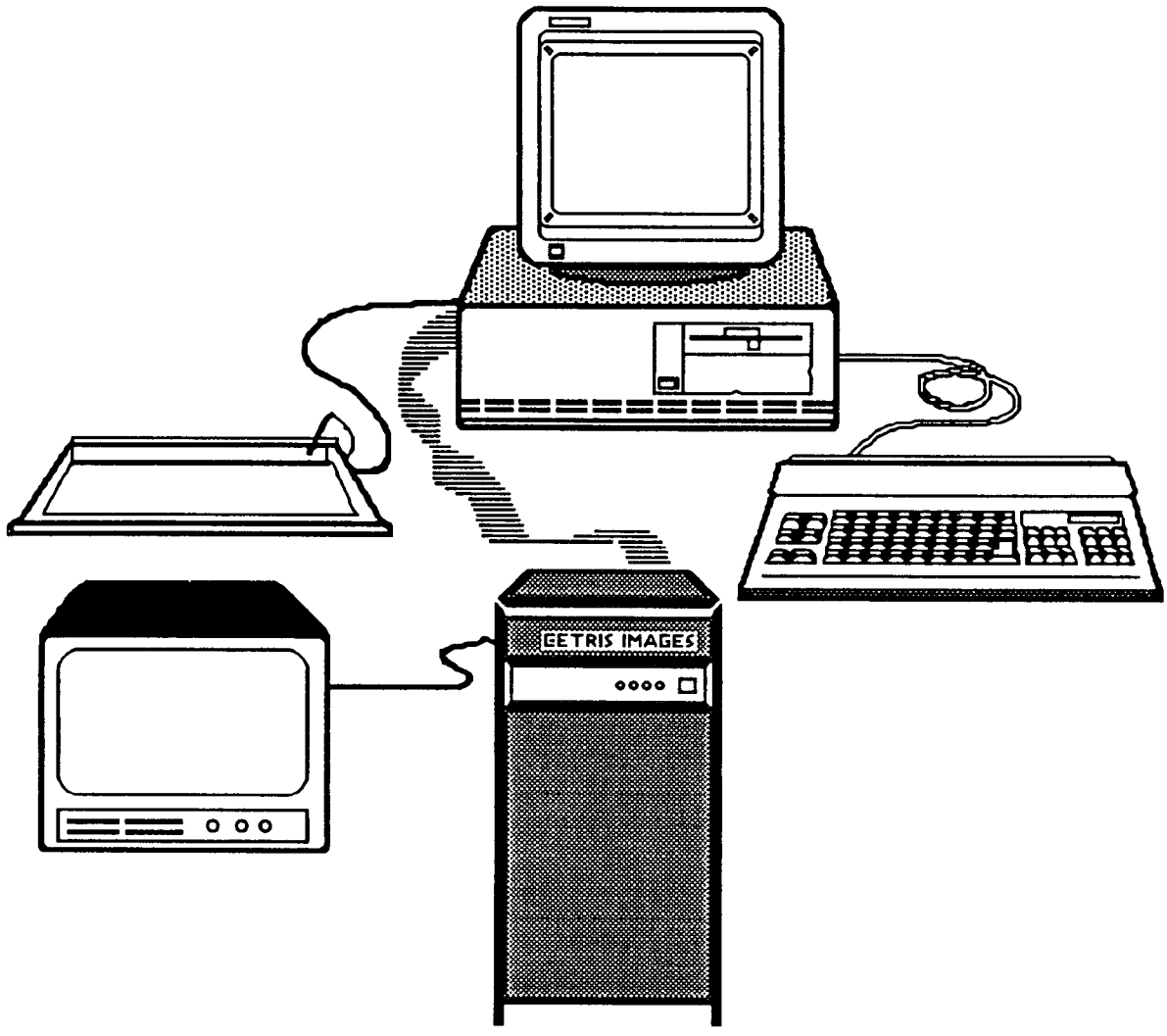
L'analyse du marché audiovisuel, ainsi que l'étude des concepts liés à l'animation 2D en temps réel, ont conduit à la réalisation d'un logiciel d'animation professionnel sur les systèmes ATALIS, puis à la conception d'une nouvelle architecture, VENICE, conservant les principes de base des systèmes précédents mais disposant de nouvelles fonctionnalités temps réel.

4.1 Station de travail

Les sytèmes GETRIS sont constitués principalement d'une partie spécialisée pour mémoriser et traiter des images (la partie de synthèse d'images), d'un ordinateur (le calculateur pilote), de logiciels (les applications), et de dispositifs de dialogue entre l'application est l'utilisateur (tablette, souris...).

L'environnement PC a été choisi comme environnement de travail pour son rapport qualité/prix interessant. Bien que la puissance des micro-ordinateurs ne cesse d'augmenter, un PC dispose de peu de moyens de communication interne et n'a pas été conçu pour gérer et traiter l'énorme masse de données et d'informations nécessaires en synthèse et traitement d'images.

Il a donc été nécessaire de développer une partie matérielle indépendante et taillée à la mesure des besoins. Entre autres, la vitesse du bus du rack de synthèse est plus de 5 fois supérieure à celle des bus d'un PC/AT et la communication interne entre ses modules spécialisés est assurée par des bus de données de 32 bits permettant de traiter instantanément 5 fois plus de données que ne l'autorise un PC. La partie matérielle se comporte comme un automate câblé, exécutant au rythme de la vidéo des traitements systématiques sur le balayage des mémoires de trame (zoom, fenêtrage, superposition...).



Les éléments d'une configuration.

4.2 L'architecture ATALIS

La partie de synthèse d'images se présente sous la forme d'un coffret séparé, pouvant recevoir jusqu'à 11 modules spécialisés.

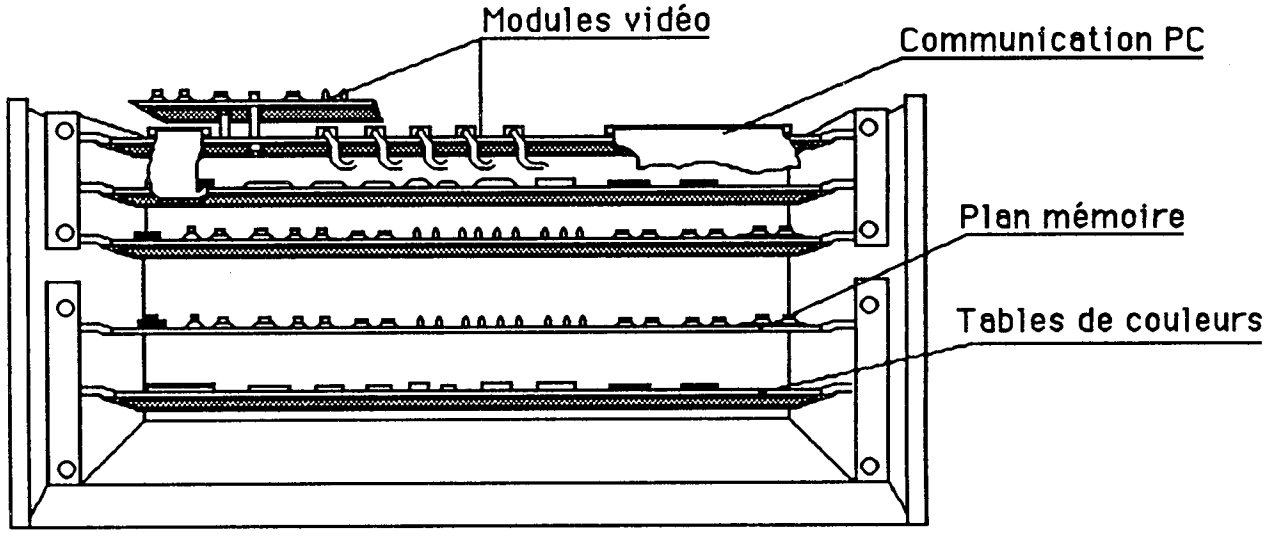
Des modules de 12 bits constituent les différentes mémoires de trame du système, configurable au choix en 12 ou 24 bits, soit respectivement 4096 ou 16,7 millions de couleurs. Dans ce cas 2 modules 12 bits sont nécessaires au stockage d'une image. Les modules mémoires d'image disposent des fonctionnalités temps réel suivantes : translation X et Y, fenêtrage et visibilité/invisibilité de couleurs. Ils communiquent entre eux par différents bus de données de 12 ou 24 bits.

Une carte d'acquisition vidéo analogique permet de visualiser une image extérieure au système en 16,7 millions de couleurs.

Deux tables de couleurs sont utilisables par les mémoires de trame et la carte d'acquisition vidéo et permettent de modifier en temps réel les couleurs de ces images en choisissant 4096 couleurs parmi 16,7 millions (indirection sur 12 bits).

Un bus de profondeur gère la superposition de 5 images simultanément, la dernière étant intégralement visible, et fournit une couleur de fond sur 24 bits dans le cas où aucune image n'est visible.

La liaison entre le calculateur pilote et la partie de synthèse est assurée par une interface parallèle rapide. Cette interface réalise l'écriture directe dans les registres de la logique câblée. Le calculateur doit donc avoir décomposé tous les ordres évolués d'animation en une suite de commandes élémentaires d'écriture. Le volume des informations à traiter étant considérable, l'interface a été bufferisée, afin d'éviter au calculateur des attentes excessives d'acquittement de la logique câblée.



Le rack de synthèse ATALIS.

Le synoptique de l'architecture de la gamme ATALIS est proposé dans l'annexe IV

4.3 L'architecture VENICE

L'architecture ATALIS est basée sur un principe de "tout ou rien", c'est-à-dire qu'un pixel est, soit visible dans son intégralité, soit invisible totalement. Ce principe souffre d'une qualité d'images moyenne en ce qui concerne l'antialiasing et interdit tout effet de transparence en temps réel.

En revanche, l'architecture VENICE est totalement pondérée et constitue une réelle avancée dans la qualité des images et des animations.

Tous les modules du systèmes sont des modules 32 bits, soit 24 bits de couleurs et 8 bits de transparence.

Les mémoires d'images possèdent un registre de pondération générale permettant de moduler globalement toutes les pondérations d'un même facteur. Elles disposent également des fonctionnalités temps réel suivantes : translations X

et Y, effet de fenêtrage, zoom X et Y, effet de déformation (mapping sur un cylindre, effet mosaïque, déroulement d'une portion d'image...).

Une carte d'acquisition vidéo permet de visualiser une image extérieure en 16,7 millions de couleurs. Cette carte admet au choix des signaux analogiques ou numériques au format 4.2.2. Elle fournit en outre un coefficient de pondération, de provenance externe au système (signal d'incrustation) ou bien fonction des couleurs de la vidéo (table de correspondance couleur -> pondération).

A chaque module, mémoire ou vidéo, peut être associée une table de couleurs à 4096 entrées, permettant les modifications en temps réel de couleurs.

Le bus de profondeur pondéré gère jusqu'à 11 niveaux d'images et autorise les effets de transparence sur les 4 premiers niveaux. L'introduction de tables de pondération en X et Y autorise également de nombreux effets de fondus et de masques, encore appelés effets de grille.

La communication entre le calculateur et la partie de synthèse est assuré par une liaison série à très grand débit (100 Mbits / seconde). De plus, deux processeurs de traitement du signal d'une puissance de 10 Mips chacun (ADSP 2100) ont été implémentés dans la partie de synthèse. Ceci afin de décharger le calculateur de la phase de compilation des ordres d'animation en commandes élémentaires. La communication se trouve donc réduite à des commandes évoluées et le parallélisme entre les 3 processeurs du système (PC + 2 ADSP) autorise des traitements plus complexes au niveau de chacun de ces processeurs.

Conclusion

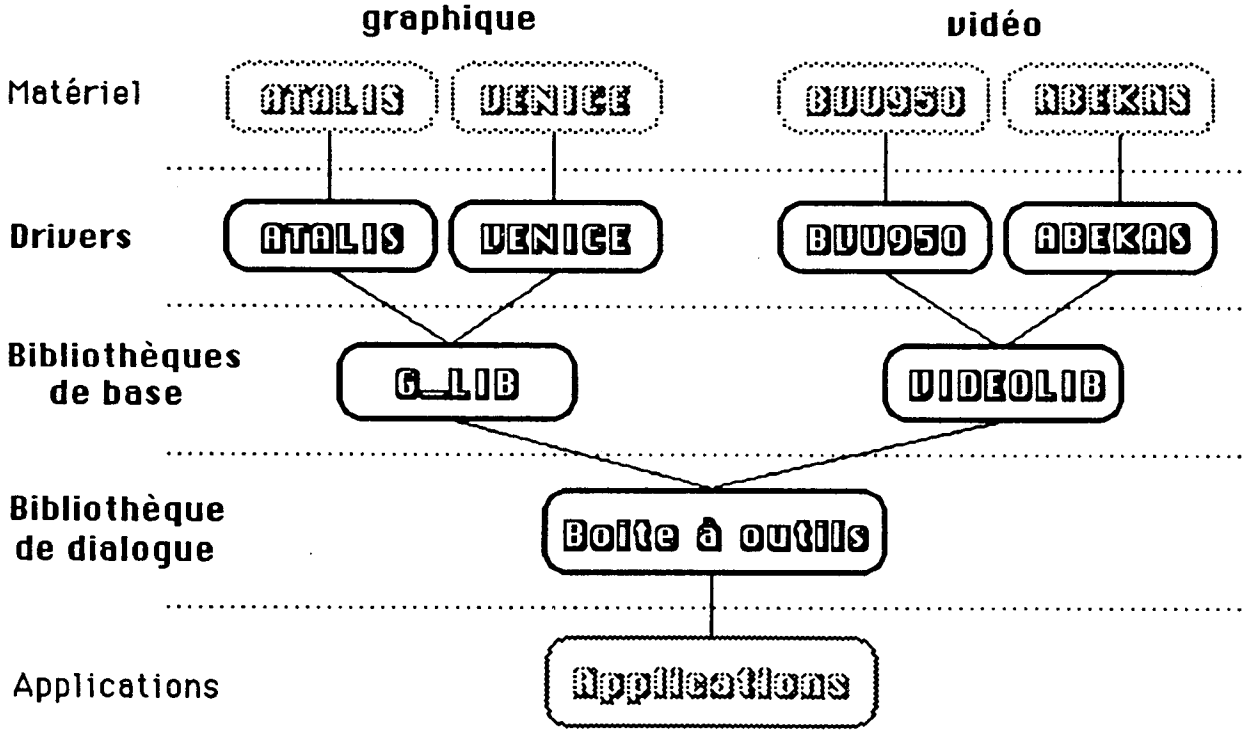
Ce chapitre nous a conduit à la spécification d'une architecture dédiée à l'animation 2D en temps réel, basée sur le principe d'interaction directe sur la visualisation des mémoires de trame et non sur leur contenu. Les fonctionnalités matérielles du système sont très proches des concepts originaux d'animation établis dans les 3 premiers chapitres de cette étude, à savoir le cell et ses attributs d'animation. L'indépendance de ces fonctionnalités (zoom, fenêtre, profondeur, couleurs...) permet une description interactive de chacun des attributs de l'animation indépendamment des autres, qu'il se rapporte à un cell ou bien à une séquence. Ceci facilite considérablement le travail de l'animateur. En outre, l'interface entre la partie de synthèse et les logiciels d'animation implémentés dans l'environnement PC se réduit ainsi à un ensemble de commandes bien déterminé, pour le pilotage de la logique câblée associée à chacune des fonctionnalités du système.

CHAPITRE 5 : LE LOGICIEL DE BASE

Introduction

Nous avons dégagé dans le chapitre précédent les fonctionnalités matérielles d'une machine de synthèse dédiée à l'animation en temps réel. Ces caractéristiques peuvent être partagées entre plusieurs machines (cas d'ATALIS et VENICE par exemple) et on peut alors parler d'une gamme de produits. Leur implémentation matérielle reste cependant entièrement spécifique à la machine proposée. Afin de ne pas remettre en cause les développements passés, lors de la sortie d'une nouvelle machine d'une gamme existante, il est nécessaire d'assurer une totale indépendance entre les fonctionnalités d'une gamme et leur implémentation matérielle. De plus, pour éviter les développements redondants d'une application à une autre et assurer une forte cohérence entre celles-ci, il apparait nécessaire de développer des outils logiciels de haut niveau directement utilisés par les applications.

Les différents développements logiciels se ramènent aux 3 couches présentées dans le schéma suivant:



Les couches logicielles de base.

Nous débuterons donc ce chapitre avec la présentation de la couche logicielle de plus bas niveau, c'est-à-dire les différents drivers, réalisant le pilotage des

différents matériels graphiques, proposés au chapitre précédent, et des matériels vidéo, couplés nécessairement à ces derniers. Nous aborderons ensuite la présentation des bibliothèques de commandes, regroupant notamment toutes les primitives évoluées d'animation. Ces deux couches, drivers et bibliothèques, ont pour but d'assurer l'indépendance entre les fonctionnalités matérielles spécifiques aux appareils d'une même gamme et les couches logicielles supérieures correspondant aux différents modules et applications d'animation.

Nous présenterons enfin la couche la plus proche des applications, à savoir une bibliothèque regroupant tous les outils de dialogue nécessaires à la réalisation d'applications interactives. Cette bibliothèque assure une cohérence de programmation et surtout d'utilisation entre toutes ces applications et décharge le concepteur d'applications interactives des fortes contraintes de gestion de dialogue.

1. LES DRIVERS

Tout appareil graphique ou vidéo dispose de caractéristiques matérielles particulières. Afin de rendre ces caractéristiques transparentes à l'utilisateur, le pilotage d'un tel appareil doit être réalisé par un module de gestion spécifique, appelé driver.

Ce driver a la charge de réaliser l'interface entre le matériel et les fonctions évoluées de commande de la bibliothèque correspondante. Il contrôle notamment toutes les communications avec le matériel et connait toutes ses caractéristiques (adresses des registres, temps de réponse...).

1.1 Les drivers GETRIS

Un driver graphique a la charge d'interpréter toutes les commandes évoluées qui lui parviennent et doit notamment réaliser l'écriture des commandes d'animation dans la logique câblée du système. La donnée des 4 boites d'une image lui permet, par exemple, de calculer les coefficients à affecter au processus câblé du zoom (écriture de l'erreur et des incréments), et de réaliser les écritures dans les registres de fenêtrage.

Les calculs peuvent s'effectuer dès la réception des commandes d'animation. En revanche, les écritures registres doivent impérativement survenir pendant les retours de trame (durée de 2 ms). Il est donc nécessaire que le driver mémorise les valeurs des registres issues des calculs et synchronise les écritures sur le retour de trame.

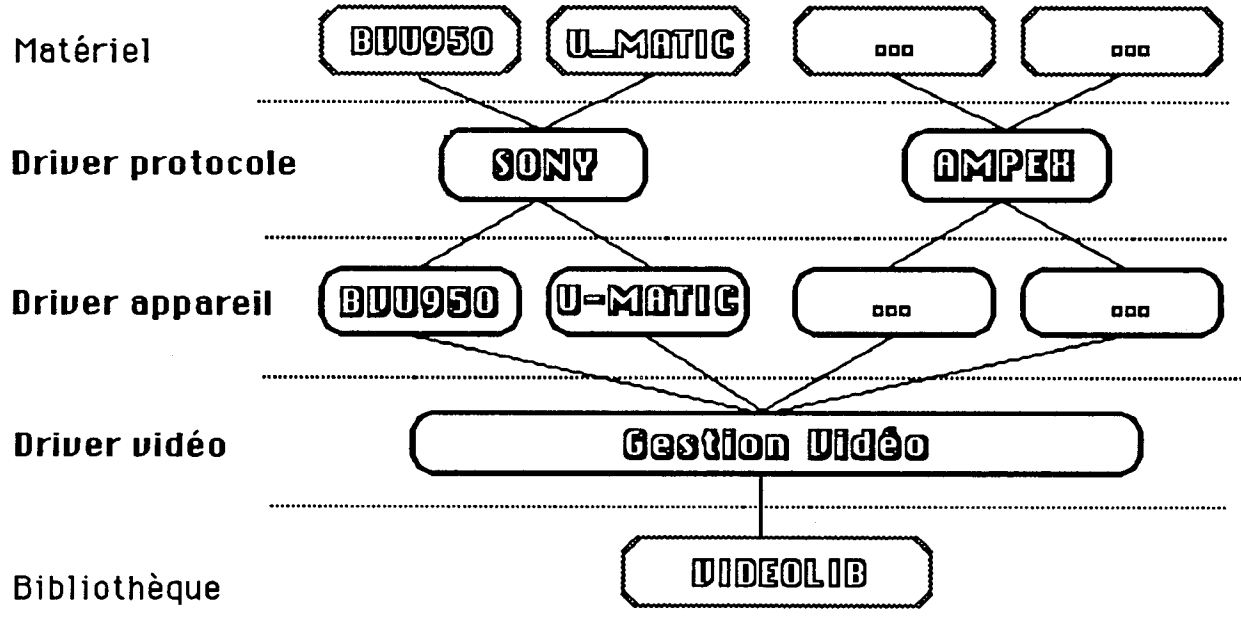
Deux drivers ont été successivement développés au sein de la société Gétris Images.

Le driver ATALIS est implémenté directement sur le PC. C'est donc le même processeur qui est chargé de la gestion des commandes de la bibliothèque et des accès au matériel. Aucun parallélisme n'est donc autorisé.

Le driver VENICE, quant à lui, est implémenté sur 2 processeurs ADSP, chargés de l'interprétation des commandes évoluées en provenance du PC, et des accès à la logique câblée de la machine VENICE. Cette découpe autorise un fort parallélisme et du même coup décuple la puissance du système d'animation.

1.2 Les drivers vidéo

Tous les appareils vidéo (magnétoscopes, vidéo-disques...) proposent en général les mêmes fonctions de base, mais sont fondamentalement différents d'un point de vue matériel. Plusieurs appareils vidéo de types différents peuvent être pilotés simultanément en lecture ou en enregistrement. Le pilotage de ces divers appareils vidéo nécessite donc la réalisation d'un driver vidéo spécifique, chargé de leur gestion et notamment de leur synchronisation. En fait, 3 couches de drivers apparaissent nécessaires au pilotage de ces appareils, comme l'indique la figure suivante.



Représentation des 3 couches de drivers vidéo.

1.2.1 Le driver vidéo

Le driver vidéo est chargé de la gestion globale des appareils vidéo et de la répartition des commandes entre les différents drivers associés aux appareils vidéo. Ce driver est unique et doit pouvoir traiter toutes les configurations ou commandes qui peuvent se présenter dans le système d'animation, entre autres, la synchronisation de plusieurs appareils, indépendamment du type des appareils pilotés. On peut, par exemple, enregistrer sur magnétoscope une animation synchronisée elle-même sur la vidéo d'un disque dur numérique.

1.2.2 Le driver appareil

Le driver appareil est chargé quant à lui de l'envoi des commandes à l'appareil vidéo qu'il pilote et de la transmission au driver vidéo des informations relatives à cet appareil (flag d'état, position courante...). Chaque appareil vidéo dispose d'un driver spécifique.

En fait chaque constructeur de matériel vidéo propose un protocole de communication commun à tous ses appareils, certaines fonctions n'étant pas implémentées sur tous les types d'appareils. Afin de ne pas dupliquer les commandes communes à plusieurs appareils d'une même gamme, il apparait intéressant d'introduire un troisième niveau, appelé driver protocole. Le driver vidéo fournit à ce dernier toutes les commandes interprétables directement par l'appareil concerné. Dans le cas d'un appareil sophistiqué, le driver vidéo se contente de la communication d'ordres évolués au driver protocole. En revanche, pour un appareil bas de gamme, certaines fonctions doivent être simulées et donc compilées en commandes élémentaires compréhensibles par l'appareil concerné. Un magnétoscope 1 pouce, par exemple, dispose en interne d'une commande d'enregistrement entre les points d'entrée et de sortie précisés. L'envoi du code de cette commande seule suffit à déclencher l'enregistrement. En revanche, un magnétoscope U-Matic de même marque ne dispose pas de cette fonction. Il est donc nécessaire de simuler cette fonction au niveau du driver matériel associé en utilisant un ensemble de commandes moins évoluées.

1.3.3 Le driver protocole

Le driver protocole est chargé de communiquer les commandes élémentaires à tous les appareils vidéo obéissant au même protocole. Nous pouvons citer par exemple les protocoles SONY ou bien encore AMPEX, qui sont intégralement respectés par tous les appareils fabriqués par ces sociétés.

L'implémentation de toutes les commandes d'un protocole dans un tel driver ouvre la voie sur le pilotage de tous les appareils obéissant à ce protocole. La prise en compte d'un nouvel appareil d'une gamme nécessite uniquement la réalisation du driver appareil correspondant.

2. LES BIBLIOTHEQUES DE COMMANDES

Une bibliothèque regroupe les fonctions communes à toutes les machines d'une même gamme, indépendamment de l'implémentation de ces fonctions au niveau matériel. Nous avons distingué 2 bibliothèques principales, G_LIB, pour la gestion de la partie de synthèse, et VIDEOLIB, pour la gestion de tous les appareils vidéo en lecture et en enregistrement.

2.2 La bibliothèque G_LIB

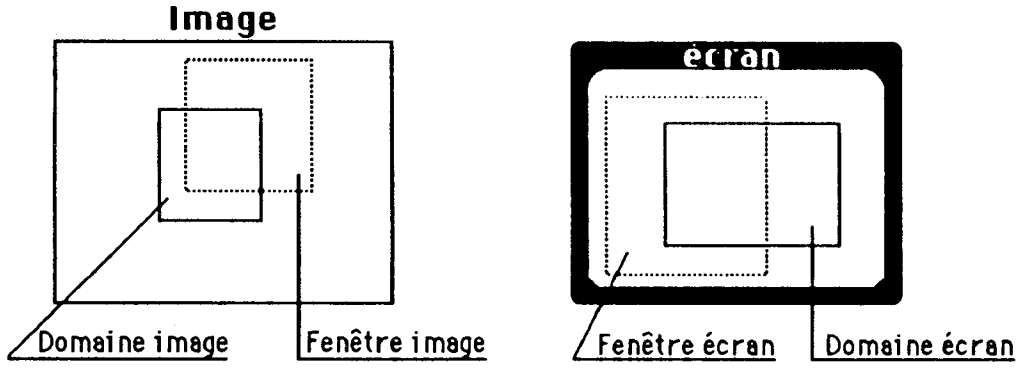
La bibliothèque G_LIB est la bibliothèque de base du système d'animation. Elle regroupe toutes les informations relatives à sa configuration matérielle et logicielle (nombre de mémoires d'images, configuration couleur ou identification...), ainsi que les primitives de commande de la partie de synthèse, et plus particulièrement les commandes d'animation. Ces dernières reflètent exactement les concepts matériels évoqués dans le quatrième chapitre de cette étude. Quelques unes d'entre elles sont présentées ci-dessous :

BoiteImage(N°image, type_boite, xg,yb,xd,yh);

qui permet de préciser les différentes zones d'images ou d'écran (de coordonnées *xg, yb, xd, yh*) intervenant dans le processus de visualisation de l'image de numéro *N°image*. Le paramètre *type_boite* correspond suivant le cas:

- au domaine image, correspondant au cell,
- au domaine écran, correspondant à la zone de visualisation à l'écran,
- à la fenêtre image, permettant de ne visualiser qu'une portion de cell,
- à la fenêtre écran, permettant de ne visualiser qu'une portion de cell zoomée à l'écran.

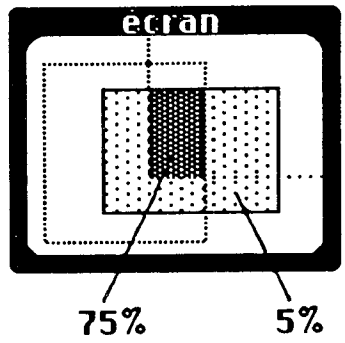
L'indication de ces 4 boites suffit à déterminer les paramètres de visualisation d'un acteur. Rappelons que la boite image est zoomée afin d'être visualisée dans la boite écran.



Les 4 types de boites.

PonderImage(N°image, pint, pext);

qui permet de moduler globalement les pondérations de l'image de numéro *N°image*. La portion d'image comprise dans l'intersection des 4 boites citées ci-dessus est modulée par le coefficient *pint*, le reste de l'image est modulé par le coefficient *pext*. Dans un fonctionnement standard, *pint* correspond à 100% et *pext* à 0%.



Pondération de fenêtre (pint = 75 %, pext = 5%).

ProfImage(N°image, prof);

qui permet d'affecter le rang *prof* à l'image de numéro *N°image*, dans le processus de superposition des mémoires d'images.

FondGetris(Rouge, Vert, Bleu);

qui permet de visualiser la couleur dont les primaires sont donnés par les coefficients *Rouge*, *Vert* et *Bleu*, en fond d'animation.

GrilleGetris(N°motif,p1,p2,...pn);

qui permet de réaliser un effet de fondu entre plusieurs mémoires d'images.

Les primitives de la bibliothèque G_LIB peuvent suivant les cas, soit communiquer au driver les commandes et les paramètres tels qu'ils apparaissent lors de l'appel (ce sont alors des points d'entrée dans le driver), soit décomposer certaines actions complexes en actions élémentaires du driver (elles compilent ces actions en codes interprétables par le driver).

2.3 La bibliothèque VIDEOLIB

Cette bibliothèque regroupe toutes les primitives évoluées de pilotage de matériels vidéo, indépendamment de leurs types ou de leurs marques. Un noyau commun a en effet pu être dégagé à partir des fonctionnalités de base proposées par les différents matériels vidéo (Arrêt, Défilement, Edition, Rendez-vous, Recherche, Rebobinage, Bobinage...).

Cependant, il existe des différences majeures entre les matériels vidéo, notamment dans les vitesses d'accès à une image du support vidéo et dans la qualité des images au cours du traitement des commandes. Un disque dur numérique, par exemple, dispose d'un accès aléatoire aux images, d'une durée variable de 0 à 5 trames, et autorise un défilement rapide des images, sans perte de qualité. En revanche, un magnétoscope dispose d'un accès séquentiel aux images. Les temps

d'accès sont donc proportionnels au nombre de trames séparant l'image couramment visualisée de l'image recherchée. Les défilements rapides, sans perte de qualité, ne sont possibles que sur des appareils très haut de gamme.

Afin de décharger l'utilisateur de toutes ces contraintes matérielles, la bibliothèque VIDEOLIB propose également un certain nombre de fonctions évoluées pour la lecture et l'enregistrement d'animations, notamment pour la synchronisation de plusieurs appareils vidéo. Ces fonctions de haut niveau ont été réalisées indépendamment des caractéristiques spécifiques des appareils vidéo. Les drivers vidéo, quant à eux, ont la charge de réaliser ces fonctions et de communiquer à la bibliothèque les informations propres à leur configuration.

Prenons par exemple le cas d'une synchronisation entre un vidéo-disque et un magnétoscope. La primitive de synchronisation de la bibliothèque interroge tout d'abord les différents drivers et analyse en retour les informations qui lui sont données, en particulier les temps de réponse de chaque appareil. En fonction de ces données elle répartit dans le temps les envois des différentes commandes de synchronisation.

3. LA BOITE A OUTILS

La réalisation d'applications interactives d'animation ne nécessite pas seulement l'existence d'une bibliothèque logicielle de base, mais demande également un très gros investissement au niveau de l'interface homme/machine. Nous allons présenter dans cette partie les outils de dialogue d'un point de vue du concepteur. Ces outils ont été réalisés à partir des 2 bibliothèques G_LIB et VIDEOLIB et constituent la bibliothèque de base pour le développement d'applications interactives d'animation.

3.1 Les bases du dialogue interactif

3.1.1 Le problème de l'interface utilisateur

L'évolution et les progrès constants de l'informatique font que les systèmes actuels évoluent vers une plus grande interactivité et s'adressent de plus en plus à des utilisateurs non spécialistes de l'informatique. Aussi l'interface utilisateur occupe-t-elle une part de plus en plus importante dans les applications.

Pour des systèmes dédiés à la synthèse d'images, cette constatation est encore plus évidente. Toutes les fonctionnalités du systèmes doivent être accessibles simultanément, l'ergonomie du dialogue doit être très poussée et la manipulation des informations la plus adaptée possible. Ainsi l'utilisation du clavier doit-elle être réduite à des cas très particuliers. De plus les fonctions doivent être disponibles directement et séparément à tout instant. Ceci proscrit l'utilisation de menus imbriqués et la saisie systématique de tous les paramètres d'une action.

3.1.2 La notion de bibliothèque de dialogue

Ces considérations ont mené à la conception d'une bibliothèque, G_DIAL [Gen 89], de haut niveau proposant des outils de dialogue graphiques puissants et extensibles. Fondée sur les concepts des langages "**orientés objets**", cette bibliothèque offre aux développeurs des outils permettant de réaliser rapidement des applications à haut degré d'interactivité. L'apprentissage est facilité par la logique de fonctionnement simple s'appliquant sans exception à tous les objets de dialogue.

L'utilisation des primitives de cette bibliothèque de dialogue permet de réaliser rapidement le squelette d'une application interactive et préserve une cohérence certaine entre toutes les applications interactives de la société, minimisant ainsi la phase d'apprentissage de l'utilisateur du système.

3.2 La définition de l'interface utilisateur

Du point de vue du concepteur, la définition de l'interface utilisateur consiste à associer à différentes zones de l'écran :

- une présentation visuelle matérialisant les éléments les composant,
- une description de leur comportement lors des échanges entre le programme d'application et l'utilisateur.

3.2.1 La définition d'objets de dialogue

De cette constatation découle immédiatement la définition du mécanisme de base qui sert à la construction des objets de dialogue : un objet élémentaire de dialogue définit une zone sur une image qui réagit de manière uniforme lors des échanges entre l'application et l'utilisateur.

On peut définir 2 classes principales d'objets de dialogue, basées sur deux représentations de zones de dialogue différentes :

- les domaines, ayant comme support une zone rectangulaire et définis par les coordonnées de leurs coins inférieurs gauches et supérieurs droits,

- les morphologies, ayant comme support un ensemble de points caractéristiques.

Ces objets sont proches de la philosophie "orientée objet". En effet chacun d'eux dispose d'informations propres et est susceptible de réagir spontanément, suivant ses propres fonctionnalités, aux sollicitations extérieures que représentent les actions de dialogue.

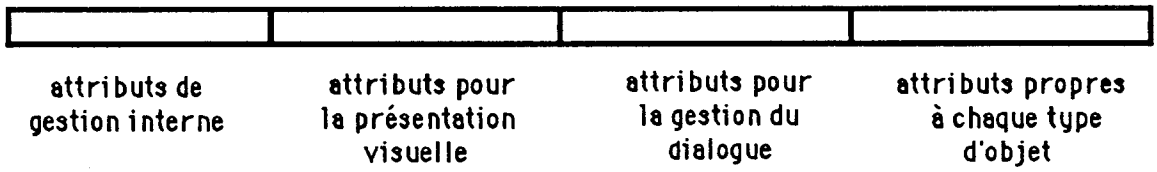
3.2.2 La définition d'attributs de dialogue

Chaque objet élémentaire de dialogue est représenté par un ensemble d'attributs regroupés à l'intérieur d'un seul descripteur.

Certains de ces attributs sont communs à tous les objets de dialogue, indépendamment de leur classe. Il s'agit notamment des attributs définissant le comportement de l'objet de dialogue lors d'opérations de dialogue, c'est-à-dire lors de l'utilisation et de l'activation des dispositifs d'interaction sur l'objet. Ces attributs seront présentés dans le paragraphe suivant, de façon commune aux deux classes d'objets.

D'autres sont spécifiques à une classe d'objets de dialogue. Citons, entre autres, les attributs pour la gestion interne et la structuration des objets élémentaires de dialogue, ainsi que les attributs permettant la matérialisation de l'objet de dialogue lors de son utilisation. Ces attributs seront présentés spécifiquement à chaque classe d'objets.

Aux attributs partagés par tous les objets de dialogue élémentaires s'ajoutent éventuellement d'autres attributs spécifiques à chaque type d'objet élémentaire.



structure générale des descripteurs d'objets élémentaires de dialogue.

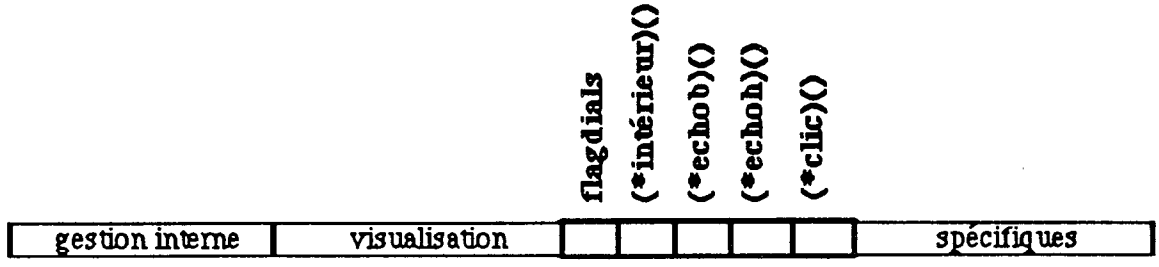
3.2.3 Attributs pour la gestion du dialogue

Les attributs définissant le comportement d'un objet pour le dialogue sont de 3 types :

- des attributs permettant d'activer ou de désactiver l'objet pour le dialogue,

- des attributs définissant l'apparence visuelle du dialogue, c'est-à-dire la forme et la couleur de l'écho du dispositif de dialogue lorsque celui-ci se situe sur la zone utile correspondant à l'objet,

- des attributs définissant l'interprétation des événements du dialogue, intervenant sur l'objet, c'est-à-dire les actions à réaliser lors des changements d'état du dispositif d'interaction.



Attributs pour la gestion du dialogue.

3.2.3.1 la validité du dialogue

(*intérieur)() est l'adresse d'une fonction permettant d'identifier l'intérieur d'un objet. Cette fonction permet la reconnaisance d'une zone rectangulaire (cas de domaines) ou de la région correspondant à une morphologie (par exemple le disque associé à un cercle).

flagsdial est un indicateur qui détermine pour quels états du dispositif d'interaction l'objet intervient lors d'une séance de dialogue.

Dans le cas d'une tablette à digitaliser, qui constitue un support de dialogue privilégié, nous ne considèrerons que 2 états qui correspondent au stylet levé (dialogue en position haute) et au stylet enfoncé (dialogue en position basse). Ces deux états sont étendus aux autres dispositifs tels que la souris, où la position basse est conditionnée par la pression sur le bouton de "clic".

Le positionnement des indicateurs de *flagdials* peut être effectué par l'intermédiaire des constantes prédéfinies :

haut : dialogue valide lorsque le dispositif est en position haute,
bas : dialogue valide lorsque le dispositif est en position basse,
tout : qui correspond à *haut* + *bas* et indique que le dialogue est valide sur l'objet quel que soit l'état du dispositif d'interaction.

3.2.3.2 La présentation du dialogue

Les attributs d'écho lient les déplacements du dispositif d'interaction sur l'objet à un ou plusieurs témoins. Ils permettent de spécifier 2 formes différentes d'écho suivant l'état du dispositif de dialogue (position haute ou basse).

(*echob)() et ***(*echoh)()*** sont respectivement les adresses des 2 fonctions d'affichage de l'écho lorsque le dispositif est en position basse et haute. Le premier paramètre de ces fonctions est l'étape du dialogue, qui permet de distinguer le moment de l'interaction sur l'objet et donc d'effectuer éventuellement un

traitement particulier. Ce paramètre doit être choisi parmi les constantes prédéfinies suivantes :

- ***premier*** qui correspond au début d'une séquence d'interaction sur l'objet,
- ***interm*** lorsqu'il n'y a pas de changement d'état du dispositif, qui est resté sur le même objet,
- ***dernier*** lorsqu'il y a eu un changement d'état du dispositif ou que celui a quitté la région correspondant à l'objet.

Les fonctions d'écho permettent, par exemple, de visualiser des informations en cours de dialogue sur un objet (mise à jour de témoins tels que des compteurs de coordonnées...).

3.2.3.3 L'interprétation du dialogue

L'attribut ***(*clic)()*** est l'adresse d'une fonction qui est invoquée lorsque des événements surviennent sur la région associé à l'objet durant une séance de dialogue, en particulier lorsque le dispositif d'interaction change d'état. Trois types d'évènements de dialogue sont pris en compte par les constantes prédéfinies, passées en premier paramètre de la fonction :

- ***descendant*** qui indique que le dispositif de dialogue est passé de la position haute à la position basse,
- ***montant*** qui indique que le dispositif de dialogue est passé de la position basse à la position haute, en restant sur la même région de dialogue,
- ***annulation*** , active uniquement pour la classe des domaines, qui indique que le dispositif a été déplacé en position basse hors du domaine avant d'être relevé.

3.3 Les morphologies

Cette classe d'objets de dialogue regroupe l'ensemble des objets permettant de saisir interactivement des formes élémentaires avec matérialisation d'un écho temporaire à l'écran.

3.3.1 Présentation visuelle d'une morphologie

Une morphologie est matérialisée par une ligne polygônale ou un ensemblde points. A chaque morphologie est associée implicitement une fonction d'affichage adaptée au type de la morphologie. Cette fonction interprète les différents paramètres apparaissant dans la structure de la morphologie.

La morphologie est affichée suivant l'aspect de ligne qui lui est associé. Cet aspect est caractérisé par trois paramètres :

- la couleur de la ligne,
- la nature de la ligne (continu, pointillé...),
- l'épaisseur de la ligne correspondant à un crayon ou a des pinceaux de différentes largeurs.

3.3.2 La manipulation des morphologies

Des actions communes à toute la classe des morphologies assure une cohérence et une homogénéité de comportement de ce type d'objets de dialogue, non seulement au niveau de la définition, mise à jour et consultation, mais aussi au niveau de l'affichage et du dialogue.

La primitive **refmorph = DefMorph(refpère,typmorph)** permet d'initialiser une structure de morphologie. *Refpère* représente la référence d'un domaine de dialogue (cf § 3.4). *Typmorph* indique le type de la morphologie définie à choisir parmi les suivantes : rectangle, cercle, segment de droite, points, polygône ouvert ou fermé, ellipse, courbe. Cette primitive retourne la référence *refmorph* de la morphologie créée. Cette référence permet l'identification de la morphologie dans toutes ses manipulations.

La primitive **MajMorph(refmorph,nbpts,x1,y1,...,xn,yn)** permet d'initialiser ou de modifier les coordonnées des *nbpts* points caractéristiques de la morphologie référencée par *refmorph.*

La primitive **ConsMorph(refmorph,type,nbpts,adrx,adrxy)** réalise la consultation du type de la morphologie référencée par *refmorph* et retourne dans les tableaux d'adresses *adrx* et *adry* les *nbpts* points caractéristiques de la morphologie.

la primitive **DefFctMorph(refmorph, &fct_trt,...)** réalise l'association de la fonction de traitement *fct_trt* pour les événements de dialogue à la morphologie référencée par *refmorph.*

La primitive **AffMorph(refmorph)** réalise l'affichage de la morphologie référencée par *refmorph* dans le domaine précisé au moment de la définition (celui-ci doit être validé pour le dialogue). Cette action réalise automatiquement la sauvegarde de la portion d'image recouverte par la morphologie.

La primitive **EffMorph(refmorph)** réalise, quant à elle, l'effacement de la morphologie référencée par *refmorph.* La portion d'image sauvegardée lors de l'affichage est alors restituée.

Enfin, la primitive **ReqMorph(refmorph,type)** permet de saisir interactivement la morphologie référencée par *refmorph*. Le paramètre *type* indique s'il s'agit d'une initialisation ou d'une modification de la morphologie.

3.3.3 Exemples de morphologies

Nous allons proposer très brièvement quelques morphologies caractéristiques. Nous schématiserons rapidement la structure de chacune d'entre elles ainsi que le dialogue spécifique qui s'y rapporte.

3.3.3.1 La morphologie rectangle

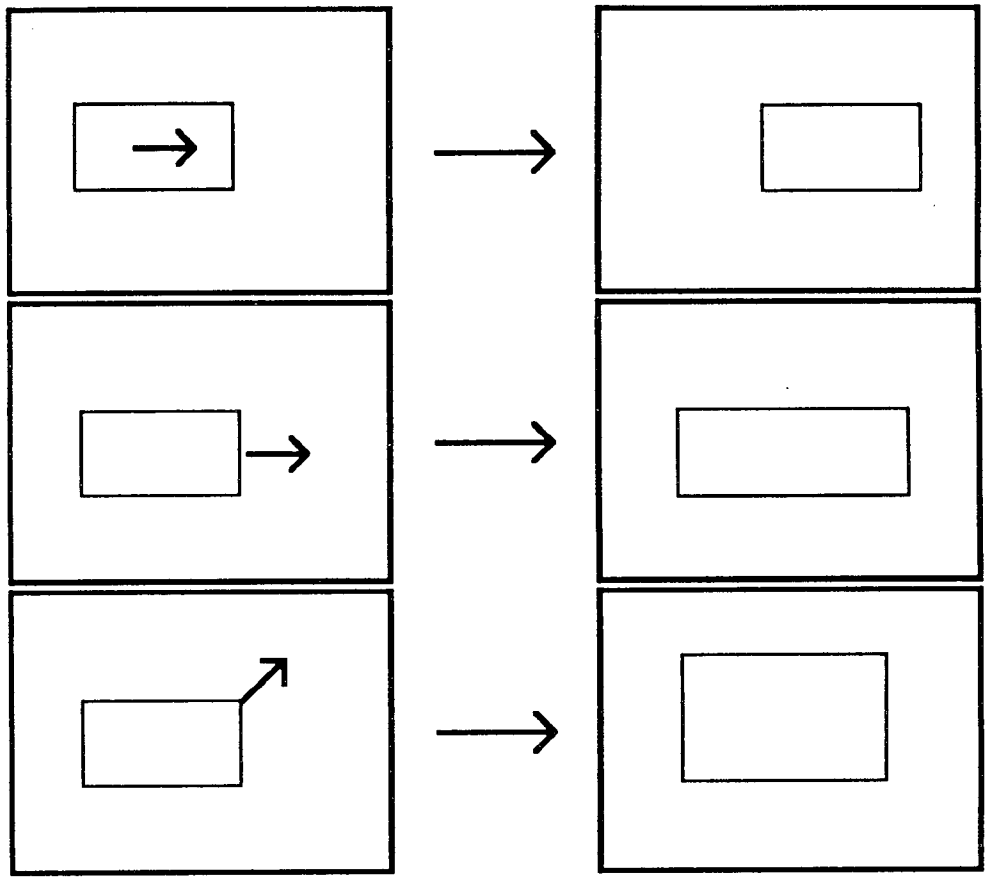
Une morphologie rectangle est caractérisée par les coordonnées de ses coins inférieur gauche et supérieur droit.

La saisie d'un rectangle se déroule suivant le scénario suivant :

1 - Appui du dispositif de dialogue (clic) sur le premier point à saisir,
2 - Déplacement sans relâchement du dispositif de dialogue jusqu'au point suivant. L'affichage de la morphologie est asservi aux déplacements du dispositf. Des touches spéciales du clavier peuvent être utilisées pour l'application de contraintes au rectangle (saisie d'un carré).
3 - Relâchement du dispositif sur le second point.

La modification d'un rectangle peut s'effectuer de trois manières différentes :

1 - par clic à l'intérieur du rectangle, pour la translation de la morphologie jusqu'au relâchement du dispositif,
2 - par clic sur un côté du rectangle, pour étendre ou réduire la dimension du rectangle correspondante,
3 - par clic sur un coin du rectangle, pour étendre ou réduire indépendamment les dimensions des côtés intersectant en ce point.



Modification interactive d'une morphologie de type rectangle.

3.3.3.2 La morphologie cercle

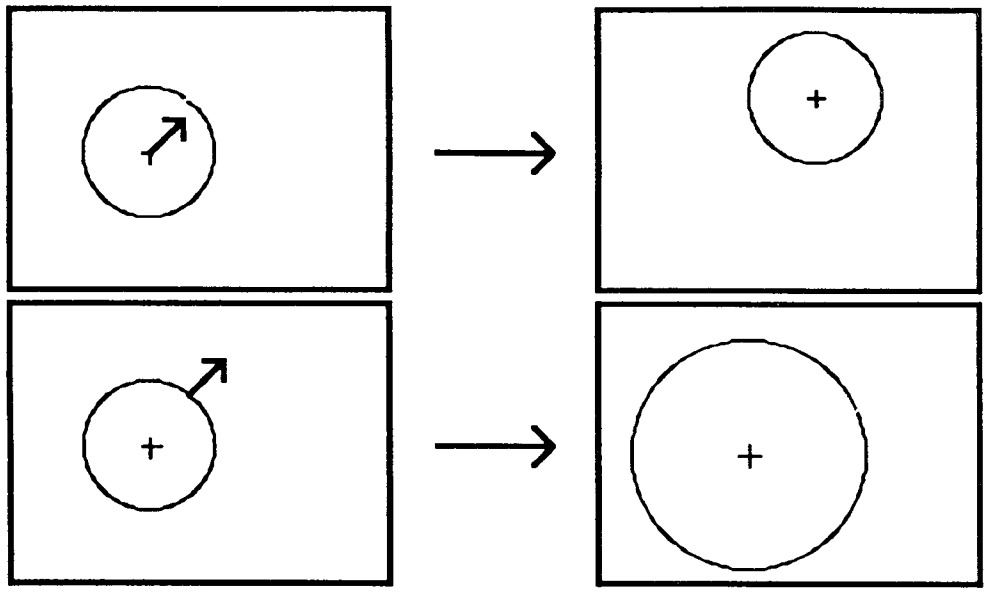
Une morphologie cercle est définie à partir de deux points caractéristiques, le centre du cercle et un point appartenant au cercle. Ce dernier sert uniquement au calcul du rayon du cercle.

La saisie d'un cercle se déroule suivant le scénario suivant :

1 - Appui du dispositif de dialogue (clic) sur le centre du cercle,
2 - Déplacement sans relâcher du dispositif de dialogue jusqu'à un point quelconque du cercle. L'affichage de la morphologie est asservi aux déplacements du dispositf.
3 - Relâchement du dispositif sur le point du cercle désiré.

La modification d'un cercle peut s'effectuer de deux manières différentes :

1 - par clic à l'intérieur du cercle, pour la translation de la morphologie jusqu'au relâchement du dispositif,
2 - par clic en un point du cercle, pour augmenter ou diminuer le rayon du cercle.



Modification interactive d'une morphologie de type cercle.

3.3.3.3 La morphologie spline

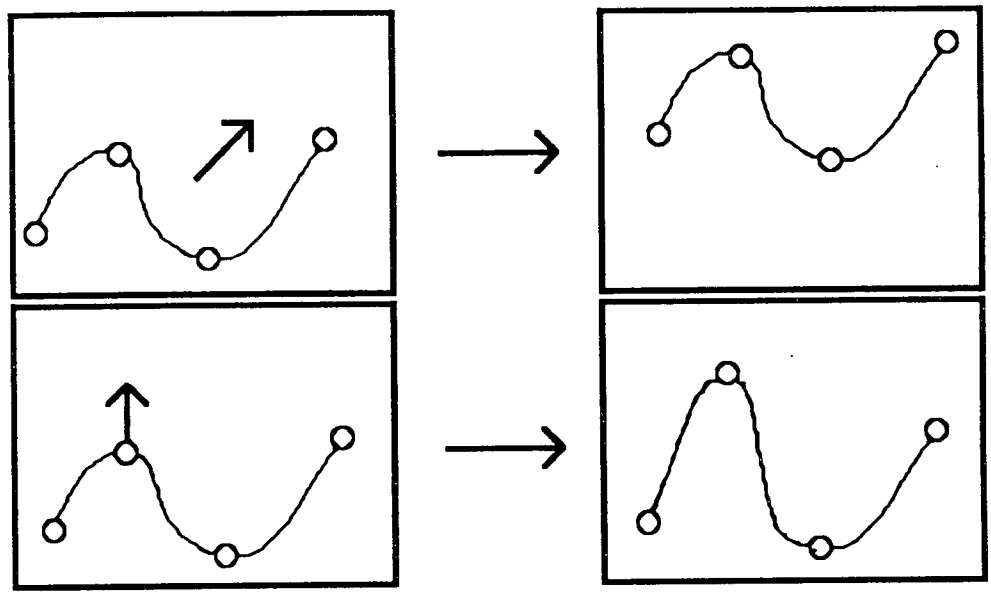
Une morphologie spline est caractérisée par un ensemble de points (limité en l'occurrence à 50) sur lequel s'applique une fonction spéciale de lissage basée sur la méthode mathématique des B-Splines.

La saisie d'une spline se déroule suivant le scénario suivant :

1 - Appui et relâchement du dispositif de dialogue sur le premier point à saisir. Ce point est aussitôt matérialisé.
2 - Appui et relâchement sur tous les autres points de la courbe, aussitôt matérialisés. L'affichage de la courbe est asservie sur les déplacements (état relevé) du dispositif de dialogue.
3 - Appui et relâchement en dehors de la zone de dialogue.

La modification d'une peut s'effectuer de trois manières différentes :

1 - par clic à l'intérieur du rectangle englobant la courbe, pour la translation de la morphologie jusqu'au relâchement du dispositif,
2 - par clic sur un point-clé de la courbe, pour modifier les coordonnées de ce point (la visualisation de la courbe est asservie au déplacement du dispositif dans l'état appuyé), puis relâchement du dispositif,
3 - par clic sur un point quelconque de la courbe, pour ajouter un point-clé à la courbe (le lissage est asservi au déplacement du dispositif dans l'état appuyé), puis relâchement du dispositif.



Modification interactive d'une morphologie de type spline.

3.4 Les domaines

Un domaine définit une zone rectangulaire sur une image.

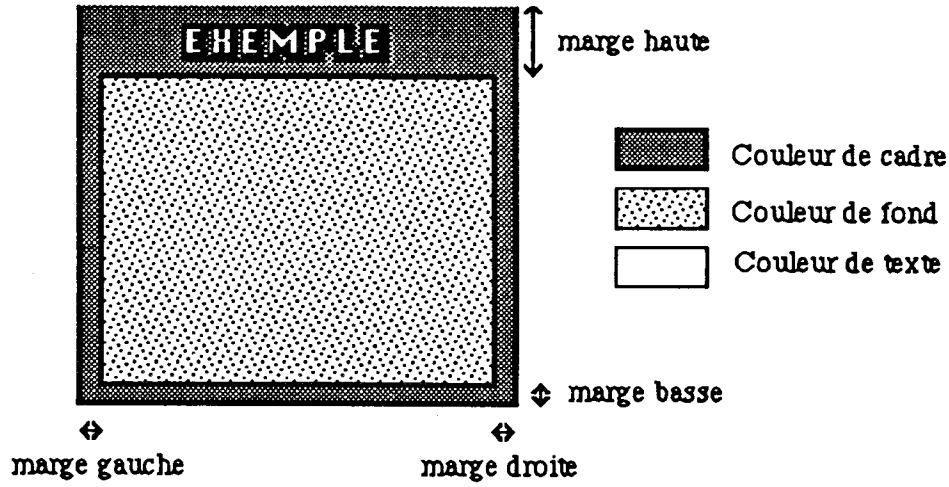
3.4.1 Présentation visuelle d'un domaine

La zone utile d'un domaine est matérialisée par un attribut de type cadre. Celui-ci définit une couleur de fond pour la zone utile et un dessin permettant éventuellement de marquer les limites du domaine, matérialisées par des marges. Dans la marge du haut peut être affiché un titre.

Les cadres sont définis indépendamment des domaines. Seules leurs références (pointeurs sur les structures de cadres) sont conservées dans le descripteur de domaine. Un cadre peut ainsi être partagé par plusieurs domaines à la fois.

La connaissance de différents aspects est nécessaire à la visualisation d'un cadre:

- une expression de couleur pour le fond, qui peut être invisible,
- une expression de couleur pour les marges,
- une expression de couleur pour le titre du cadre.

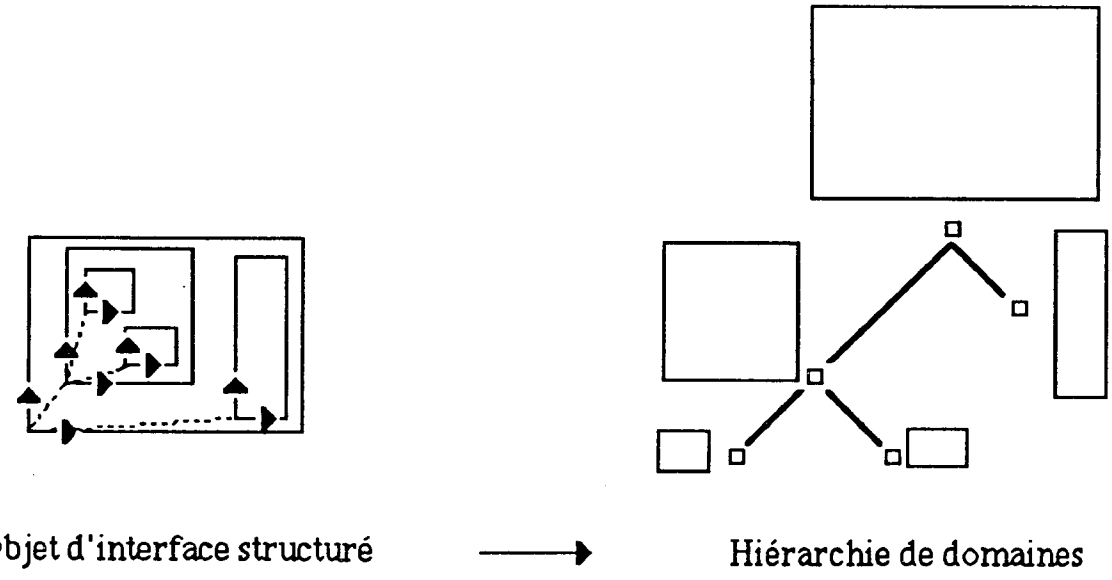


Matérialisation d'un domaine à l'aide d'un attribut cadre.

3.4.2 La structuration arborescente des domaines

Dans ce qui précède, nous avons présenté les objets de dialogue élémentaire ainsi que les attributs qui les caractérisent. En fait, dans la réalité, les objets d'interface de la classe des domaines sont plus complexes. Bien souvent ils sont composés d'un ensemble de zones qui réagissent différemment. Il est donc nécessaire de regrouper en une seule entité un ensemble de domaines qui peuvent être manipulés simultanément.

Un objet d'interface est alors défini par une structure arborescente retraçant une hiérarchie de domaines, où chacun des domaines est défini relativement à un domaine père. Les coordonnées de la zone utile de chaque domaine sont définies relativement au repère du domaine père.



Structuration arborescente de domaines.

Outre le fait qu'elle permet de regrouper en une seule entité un ensemble de régions d'une image, la structuration arborescente des domaines présente plusieurs intérêts :

- elle permet la réutilisation d'objets d'interface précédemment définis pour la construction d'objets plus complexes,
- elle induit un ordre de priorité lors de l'affichage des domaines d'un objet, ainsi qu'un ordre de parcours de la structure lors du dialogue sur l'objet. Ainsi le dernier domaine rattaché à la structure est affiché en dernier. Ce sont ses attributs qui sont pris en compte lors du déplacement du dispositif de dialogue sur la zone correspondante,
- elle permet l'héritage des attributs de dialogue, autorisant une définition simple et puissante d'objets d'interaction complexes.

3.4.3 La manipulation des domaines

L'ensemble des primitives dédiées à la manipulation des domaines de dialogue depuis un programme d'application peut se scinder en deux groupes :

- les primitives dépendantes du type de l'objet de dialogue auquel elles se rapportent. Elles concernent la création, la modification et la consultation des objets d'interface, qui sont bien entendu paramétrés selon leur nature,
- les primitives standards indépendantes du type des objets de dialogue qu'elles manipulent.

3.4.3.1 Primitives spécifiques à chaque domaine

Si à chaque objet de dialogue sont associées ses propres primitives de création, modification et consultation, celles-ci partagent néanmoins la même forme générique. Seul le nombre et la nature des paramètres changent d'un objet à un autre.

La primitive **refdom = Def_*X*(refpère,...)** permet la définition d'un objet de dialogue de type X. Pour respecter la hiérarchie des domaines, l'objet est créé relativement à un domaine père, qui peut être une image, au premier niveau de définition. Cette primitive retourne la référence *refdom* de l'objet créé. Cette référence permet l'identification de l'objet dans toutes les manipulations de l'objet.

Un certain nombre de paramètres propres à chaque type d'objet permettent de particulariser l'objet lors de sa création.

La primitive de déclaration n'a aucune incidence sur sa visualisation, qui doit être demandée explicitement.

La primitive **Maj_*X*(refdom,...)** autorise la modification des paramètres propres à l'objet de référence *refdom*.. L'application de cette primitive entraine la mise à jour automatique de l'image, si l'objet est affiché.

La primitive **Def_Exec_*X*(refdom, &fct_trt,...)** permet la modification des paramètres décrivant le comportement de l'objet lors des séances d'interaction. Elle permet, entre autres, d'associer à l'objet référencé par *refdom* une fonction de traitement pour les événements de dialogue.

Enfin, le programme d'application peut récupérer certains paramètres caractéristiques d'un objet de dialogue à l'aide de la primitive **Cons_*X*(refdom,...)**.

3.4.3.2 Primitives standards

Ces primitives traitent les objets de dialogue de manière banalisée. Elles sont totalement indépendantes du type de l'objet d'interface.

La primitive **Aff_ObjDial(refdom,mode)** affiche un objet de dialogue quelconque référencé par *refdom*. Le paramètre *mode* indique les opérations à effectuer lors de la viusalisation de l'objet et peut être une combinaison des trois constantes suivantes :

- ***cadre*** provoque l'affichage du cadre du domaine,
- ***presse*** indique l'utilisation du presse-papier pour la sauvegarde de la région d'image correspondant au domaine,
- ***entité*** provoque automatiquement l'appel à la fonction d'affichage dont l'adresse est mémorisée dans la structure de l'objet.

La primitive **Def_Aff_ObjDial(refdom,&fct_affich,...)** permet l'association d'une fonction particulière d'affichage du domaine référencé par *refdom*. La fonction d'affichage *fct_affich* permet notamment l'affichage d'objets graphiques à l'intérieur du domaine. Elle est également utilisée pour la visualisation automatique des sous-domaines du domaine référencé par *refdom*.

La primitive **Eff_ObjDial(refdom)** réalise, quant à elle, l'effacement de l'objet de référence *refdom* précédemment affiché. L'objet n'est alors plus "rattaché" à son image support. Le dialogue n'est plus activé sur le domaine correspondant. Si le mode *presse* a été utilisé lors de l'affichage, la portion d'image recouverte par l'objet est restituée. dans le cas contraire l'effacement n'a aucune incidence sur l'image visible à l'écran.

Enfin, l'utilisation d'un objet de dialogue pour une séance d'interaction s'effectue à l'aide de la primitive **Séance(refdom)**. Une séance ne peut avoir lieu que si l'objet référencé est affiché. Lorsqu'un objet de dialogue est activé pour une séance de dialogue, les attributs de la hiérarchie du domaine sont automatiquement pris en compte. Ce sont les fonctions d'écho et de traitements des événements associées à chaque domaine composant l'objet.

La manière dont la séance de dialogue se termine est totalement fonction de la nature de l'objet de dialogue. A cet effet une fonction **FinSéance()** est proposée aux développeurs d'objets de dialogue et doit être insérée dans l'une des fonctions de traitement associées à l'objet.

3.4.4 Le contrôle dynamique du dialogue

3.4.4.1 Contexte de dialogue

Un contexte de dialogue décrit l'environnement d'exécution pour les opérations de dialogue. Il regroupe l'ensemble des paramètres qui le régit.

Un nouveau contexte de dialogue est défini par la primitive **DebDial(refdom)**, où *refdom* désigne le domaine support de l'interaction utilisateur. Cette primitive provoque l'empilement du contexte de dialogue précédent. Tous les domaines qui sont hiérarchiquement dépendants de *refdom* sont également rendus actifs. Cela signifie que les attributs de dialogue de ces différents domaines sont pris en compte automatiquement lors des séances d'interaction, durant toute la durée de ce contexte.

La terminaison d'un contexte de dialogue s'effectue par la primitive **FinDial()**, qui restitue le contexte de dialogue précédent.

3.4.4.2 Les primitives de collecte

A la base de la gestion de dialogue se trouve la primitive **Collecte()** qui provoque la collecte d'informations à partir du dispositif de saisie. Elle doit être invoquée avant tout autre récupération d'informations lors d'une séance d'interaction. Cette fonction retourne 0 si le dispositif de dialogue est en position haute, le numéro de la touche enfoncée dans le cas contraire.

La primitive *Collecte()*, à partir de l'environnement défini par le contexte de dialogue courant, récupère les données du périphérique d'entrée, gère les déplacements de l'écho et assure l'interprétation des événements du dialogue.

En outre, un certain nombre de primitives sont proposées pour récupérer des informations sur l'état courant du dialogue.

CollXY(refdom,x,y) permet de récupérer les coordonnées du point courant de dialogue relativement au repère du domaine de référence *refdom*. ces cooordonnées ont été récupérées par le dernier appel à la primitive *Collecte()* et mémorisées dans la structure de données du contexte de dialogue.

CollVidéo(rouge, vert, bleu) permet de récupérer la couleur finale (au niveau des signaux vidéo) aux coordonnées du point courant de la sénace de dialogue.

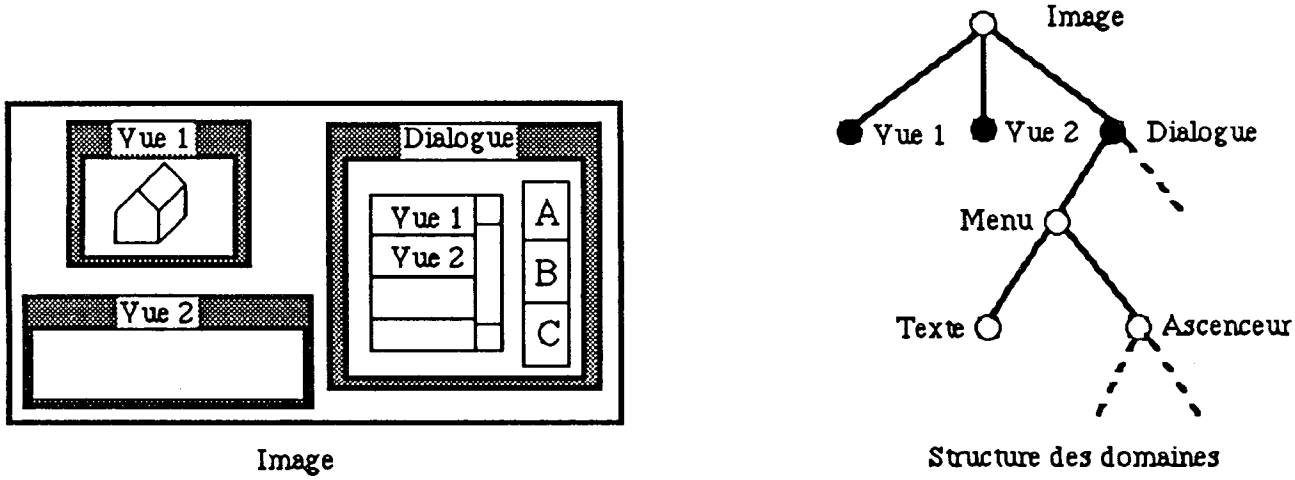
3.4.5 Exemples d'objets d'interface

Nous allons proposer très brièvement quelques exemples d'objets de dialogue de la classe des domaines, appliquant les notions énoncées dans ce dernier chapitre. Pour chaque objet nous schématiserons la structure de chacun de ces objets et nous proposerons quelques primitives utilisées par les applications.

3.4.5.1 Les vues

Les vues sont des objets d'interface uniquement destinés à la visualisation d'autres objets, graphiques ou de dialogue. Elles constituent les objets d'interface de niveau le plus élevé dans la hiérarchie des domaines.

Les vues sont à rapprocher de la notion de fenêtre propre aux systèmes possèdant une interface multifenêtres.



Utilisation des vues dans la structuration des domaines.

La primitive **Def_Vue(refpère,titre,xg,yb,xd,yh)** permet la définition d'une vue rattachée au domaine de référence *refpère*.

La primitive **Maj_Vue(refvue,type,...)** autorise les modifications des coordonnées de la vue (zone utile ou origine du domaine).

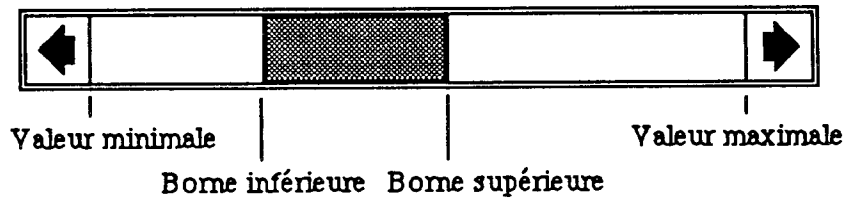
La primitive **Cons_Vue(refvue,type,...)** permet la récupération des coordonnées caractéristiques d'une vue.

3.4.5.2 Les ascenseurs

Un ascenseur est un objet de dialogue destiné à la récupération d'un intervalle de valeurs, déterminé par une borne inférieure et une borne supérieure, comprises entre une valeur minimale et une valeur maximale.

Le domaine de variation de l'ascenseur est symbolisé par une bande de défilement centrale. La valeur de l'intervalle courant est indiquée par la position et la taille d'un rectangle à l'intérieur de cette région.

Enfin, deux flèches situées aux extrêmités du domaine de l'ascenseur permettent de déplacer de façon unitaire l'intervalle courant.



Représentation visuelle d'un ascenseur.

La primitive

refasc=Def_Asc(refpère,xg,yb,xd,yh,valmin,valmax,valinf,valsup)

définit un ascenseur rattaché au domaine père de référence *refpère*. *Xg,yb,xd,yh* indiquent la position et la taille de l'ascenseur relativement au domaine père. *Valmin* et *valmax* sont les bornes du domaine de valeur associé à l'ascenseur, et valsup et valinf les valeurs d'initialisation de l'intervalle.

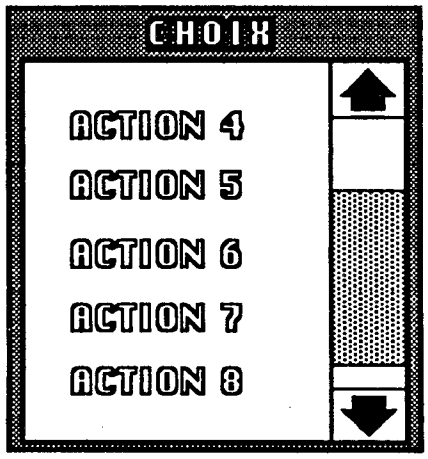
La primitive **Maj_Asc(refasc,inf,sup)** permet de modifier l'intervalle courant de l'ascenseur de référence refasc.

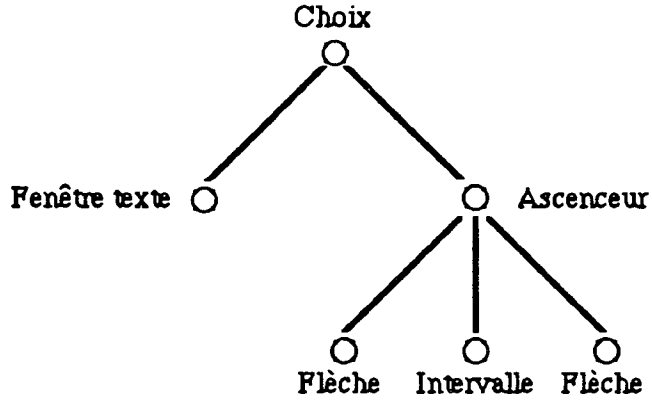
La primitive **Cons_Asc(refasc,&inf,&sup)** permet la récupération des bornes de l'intervalle courant.

Enfin, **DefExecAsc(refasc,&fct_trt)** permet de modifier la fonction de traitement de l'ascenseur référencé par refasc, afin, par exemple, de mettre à jour des compteurs pendant le dialogue sur l'ascenseur.

3.4.5.3 Les choix

Dans l'interface des applications interactives, il est très souvent nécessaire de définir des menus permettant de sélectionner une entité parmi un ensemble possible. L'objet d'interface de type choix a été défini dans ce sens.





Hiérarchie des domaines

Structure d'un domaine de type choix.

Le domaine du choix dans son ensemble est divisé en deux sous-domaines, correspondants respectivement :

- à une fenêtre d'affichage pour le texte identifiant les différentes entités du menu,
- à un ascenseur permettant de déplacer la fenêtre d'affichage du texte sur l'ensemble des entités associées au menu. Ce domaine n'existe en fait que si le nombre d'entités du menu est trop important pour pouvoir être visualisé intégralement dans la fenêtre d'affichage du texte.

la primitive **Def_Choix(refpère,refcadre,xg,yb,xd,yh,descr)** réalise la définition d'un nouvel objet de type choix, rattaché au domaine père *refpère*. Les autres paramètres de définition sont les suivants :

- *xg, yb,xd,yh* qui définissant les coins inférieur gauche et supérieur droit de la zone ocupée par le choix. Ces coordonnées sont exprimées par rapport au repère de refpère.
-*refcadre* aui est la référence de l'attribut cadre attaché à l'objet de type choix.
- *descr* qui est une chaine de caractères contenant la liste des différentes entités destinées à apparaitre dans le menu. Le premier caractère de ce descripteur joue le rôle de séparateur entre les entités.

Ainsi la chaine **"/Fichier1/Fichier2/Fichier3/Fichier4/Fichier5"**

détermine 5 entités qui peuvent représenter des noms de fichiers par exemple.

La primitive **Maj_Choix(refchoix,nument,état,texte)** permet de modifier les caractéristiques d'une entité du choix de référence *refchoix.. Nument* représente le rang de l'entité dans le descripteur, *état* indique si l'entité est valide, invalide ou encore sélectée, et *texte* contient la nouvelle chaine de caractères associée à l'entité.

Enfin, la primitive **Def_Exec_Choix(refchoix,&fct_trt)** permet d'associer une fonction de traitement, d'adresse *fct_trt*, au choix référencé par *refchoix*. Cette fonction est exécutée chaque fois qu'une entité du menu est sélectée.

3.4.5.4 Les valeurs

Les valeurs sont définies comme des domaines chargés de la visualisation et de la communication avec l'utilisateur de chaines de caractères.

Elles sont souvent utilisées en tant que compteur entier. Elles jouent alors le rôle de témoin de dialogue autant que d'outils de saisie de valeurs entières.

Un objet de type valeur.

La primitive **Def_Val(refpère,xg,yb,xd,yh,val)** permet la définition d'un objet de type valeur, attaché au domaine père de référence *refpère*. *Xg,yb,;xd,yh* représentent les coordonnées du domaine associé à la valeur et sont exprimées par rapport au repère du domaine père. *Val* est la chaine de caractère à visualiser dans le domaine.

La primitive **Maj_Val(refval,val)** réalise la modification du contenu de la chaine de caractères associée à la valeur. Si cette dernière est affichée, la modification a une répercussion immédiate sur l'écran.

La primitive **Cons_Val(refval,val)** permet la récupération de la chaine de caractères associée à la valeur.

Enfin, la primitive **Def_Exec_val(refval,&fct_trt)** permet d'associer une fonction de traitement, d'adresse *fct_trt*, à la valeur référencée par *refval*. Cette fonction est appelée à chaque fois qu'une nouvelle valeur est saisie.

Conclusion

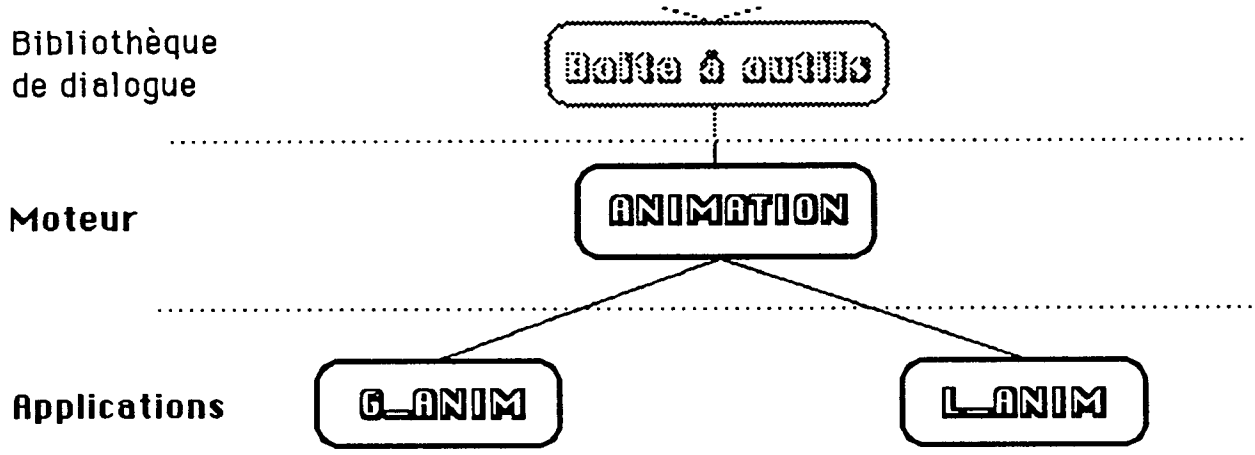
L'organisation du logiciel de base en différents drivers et bibliothèques de commandes est nécessaire pour assurer l'indépendance des fonctionnalités matérielles d'un système et leur implémentation. Elle permet d'étendre le système, aussi bien à une nouvelle machine d'animation respectant les principes de base énoncés dans cette étude, qu'à de nouveaux appareils vidéo, sans jamais remettre en cause l'environnement déjà développé.

D'autre part, un bon logiciel sans dialogue adapté court nécessairement à l'echec. Il est donc fondamental que les outils de dialogue, mis à la disposition du concepteur d'applications interactives, soient puissants et ergonomiques. La mise en place de la biobliothèque de dialogue "orienté objet" G_DIAL a nécessité de très grands moyens de spécification et de développement. Elle permet désormais de mettre rapidement en place le squelette d'applications, en respectant une forte homogénéité dans la présentation des domaines de dialogue et en assurant une totale cohérence dans l'utilisation des objets de dialogue.

CHAPITRE 6 : LES APPLICATIONS

Introduction

Nous allons présenter dans ce chapitre les différentes applications réalisées dans le cadre de notre étude. Ces applications ont été construites à partir du logiciel de base présenté dans le chapitre précédent aussi bien pour l'aspect animation que pour l'aspect dialogue. Les différents développements sont regroupés dans le schéma suivant:



Les différentes réalisations d'animation.

Pour chaque module développé nous présenterons les structures de programmation spécifiques et nous évoquerons les principaux problèmes rencontrés ainsi que les solutions retenues. Nous aborderons également le dialogue du point de vue de l'utilisateur à travers l'exemple de l'application de description de scènes et de visualisation de séquences, G_ANIM.

1. LE MOTEUR D'ANIMATION

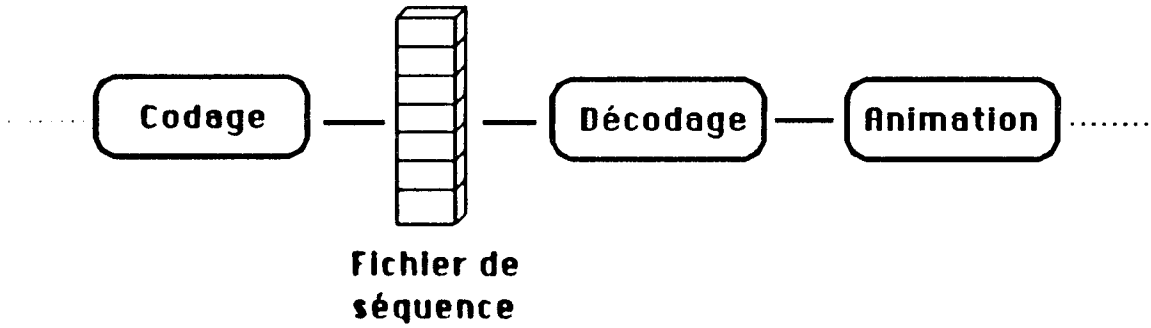
Le rôle du moteur d'animation est d'assurer l'interface entre les applications d'animation et la bibliothèque graphique. Il permet le passage des concepts logiciels d'animation (cf chapitre 3) manipulés par les applications, aux concepts matériels (cf chapitre 4) traités par la bibliothèque G_LIB.

La réalisation du moteur d'animation confine les applications dans la description des animations et les décharge ainsi de la gestion de l'animation proprement dite. Ceci contribue à l'indépendance des applications et des bibliothèques et constitue un facteur essentiel dans la portabiblité de ces applications sur des systèmes d'animation différents.

Une animation simple peut toujours être réalisée par l'utilisation directe des commandes d'animation de la bibliothèque. Cependant, la réalisation d'une animation complexe nécessite la connaissance pour chaque trame des informations relatives à cette trame et donc le calcul, à la fréquence imposée par l'animation, de ces informations. Dans la majorité des cas, ces calculs ne peuvent pas s'effectuer en temps réel. Il est donc nécessaire de préparer à l'avance toutes les commandes d'animation de chaque trame de l'animation. Le moteur a donc pour fonctions, l'ordonnancement dans le temps des commandes d'animation, et leur interprétation en temps réel.

1.1 Organisation générale

Le moteur d'animation est composé de 3 principaux modules comme l'indique la figure suivante. Le point de raccord entre ces modules est la définition d'un format de commandes d'animation spécifique.



Représentation des 3 modules du moteur d'animation.

1.2 Le module de codage

1.2.1 Le rôle du module de codage

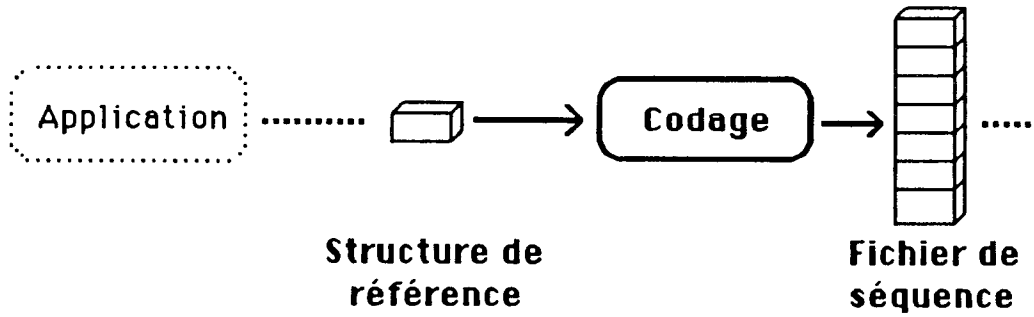
Ce module est chargé de la constitution d'une liste d'informations relatives à chaque trame de l'animation.

Chaque application dispose d'informations temporelles dans un format propre. Ce sont en général des informations clés correspondant à des instants précis de l'animation. A partir de ces informations, l'application est susceptible de déterminer toutes les informations propres à chaque trame de l'animation (par calculs d'interpolation, de splines...).

Pour que le rythme de l'animation soit respecté, toutes les informations doivent impérativement être disponibles en mémoire vive. En effet, un accès à un disque dur prend souvent plus de 20 ms (cas d'un disque déjà performant), soit au moins la durée d'une trame, ce qui interdit tout traitement supplémentaire et écriture dans la logique câblée du système.

Toutes les informations, relatives à chaque trame de l'animation, caractérisant les fonctions temps réel de la machine peuvent être stockées en RAM dans une structure de taille fixe (de l'ordre de 128 octets). Ceci fixe la durée maximale d'animation en temps réel, quelle que soit la complexité de l'animation traitée. Une capacité de 32 Koctets n'autorise ainsi qu'une animation de 256 trames, soit environ 5 secondes. Il est donc apparu nécessaire de déterminer un format de stockage plus avantageux, réalisant un compactage des informations en fonction de la complexité de l'animation traitée.

Le module de codage a donc pour rôle de récupérer les informations fournies par les applications, par le biais d'une **structure de référence**, dont le format est présenté dans le paragraphe suivant, et de les compacter selon le format défini. Cette liste compactée peut également être stockée sur disque dur en vue d'une animation ultérieure. On parle alors de **fichier de séquence**.



Structures manipulées par le module de codage.

Le compactage des informations est spécifique aux actions temps réel de la machine de synthèse. Or une scène d'animation peut nécessiter également des actions non temps réel, telles que chargement d'un fichier, génération d'une morphologie... Toutes ces actions doivent être communiquées, pour chaque trame d'animation, au module de codage, qui se charge alors de leur stockage en mémoire. La multiplication des actions non temps réel diminue donc nécessairement la capacité d'animation globale du système.

1.2.2 Les structures de référence

Le moteur réalise l'interface entre les applications et la bibliothèque d'animation. Les concepts manipulés par les applications étant plus évolués qu'au niveau de la bibliothèque, la communication avec le moteur s'établit à l'aide de structures dont le format est intermédiaire entre les formats des structures des applications et les formats d'animation propres à la bibliothèque. Nous avons vu par exemple que la réalisation matérielle d'un zoom passait par la notion de boite, englobant le cell d'un acteur dans la mémoire d'image, ou encore le cell de l'acteur zoomée à l'écran. Cependant, au niveau d'une application, il est beaucoup plus souple de manipuler directement les facteurs de zoom, qui sont alors indépendants des cells manipulés.

Certains attributs des applications caractérisent directement la mémoire d'image associée à un acteur d'une scène, d'autres en revanche se rapportent à un contexte plus général, tel qu'une séquence. Cette séparation se retrouve au niveau de 2 structures définies comme suit :

```
structure acteur
{ entier xgv,ybv,xdv,yhv; /* coordonnées image du cell d'un acteur */
   entier xi,yi;              /* coordonnées image du point-guide du cell */
   entier xgf,ybf,xdf,yhf;    /* coordonnées écran de la fenêtre */
   entier xe,ye;              /* coordonnées écran du point-guide du cell */
   réel   kx, ky;             /* facteurs de zoom du cell à l'écran */
   entier pint,pext; /* pondérations intérieure et extérieure à la fenêtre */
   entier coulcoul;        /* type de la fonction de traitement de couleurs */
   entier p1,p2,...;          /* paramètres de la fonction de couleur */
}

structure global
{   entier rouge,vert,bleu;            /* couleur de fond de l'animation */
    entier profondeurs[];           /* priorités de chacune des mémoires */
    entier grillex, motifx, ...,pnx;   /* style de fondu en x et paramètres */
    entier grilley, motify, ...,pny;   /* style de fondu en y et paramètres */
}
```

1.2.3 Le codage différentiel

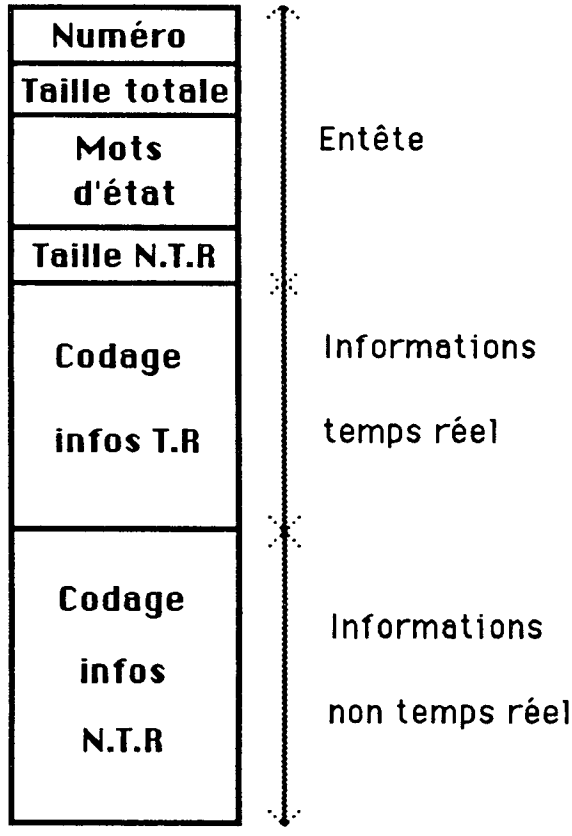
A chaque trame de l'animation, un ensemble de structures (n structures *acteur* et 1 structure *global*) est communiqué au module de codage. Comme on l'a vu au § 1.2.1, il est nécessaire de compacter toutes ces informations.

Un codage optimal consiste à ne stocker, d'une trame à une autre, que les informations ayant changé. Ce codage est réalisé en mémorisant toutes les informations non codées relatives à la trame précédemment traitée, et en les comparant champ par champ avec les informations relatives à la trame couramment traitée. A chacune de ces informations est associé un bit d'état, positionné en fonction du changement ou non de l'information. Le codage de chaque trame aboutit donc à la mémorisation, d'une part de ces bits d'état et d'autre part des informations ayant subi une modification. Il est clair que pour la première trame de l'animation, toutes les informations sont mémorisées et tous les bits d'état sont positionnés.

Le volume des informations codées est fonction de la complexité de l'animation, et l'expérience a montré qu'un tel codage permet en moyenne de réaliser un gain de 90% de la taille du code.

Le module de codage est également chargé du stockage des actions non temps réel relatives aux trames de l'animation. Ces informations sont mémorisées à la suite des informations compactées de la trame.

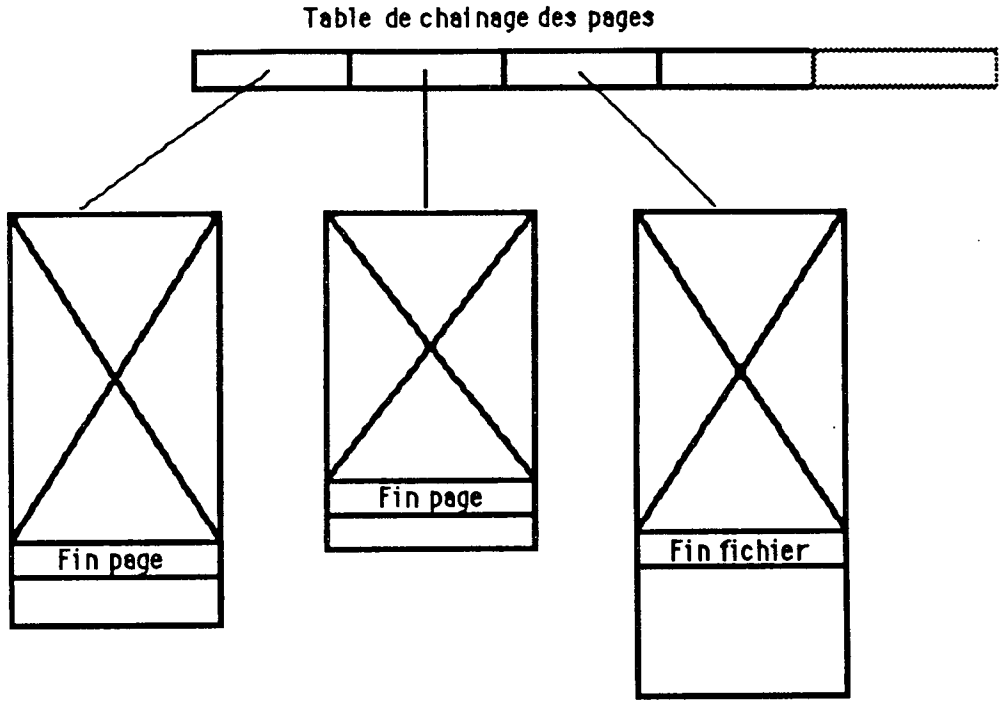
Le numéro de la trame, sa taille totale et la taille de la liste des actions non temps réel permettent au moteur d'animation de parcourir aisément le fichier d'animation.



Structure du codage d'une trame.

Le stockage des informations d'animation peut être réalisé au choix en mémoire vive ou sur disque dur, par l'intermédiaire d'un fichier de séquence.

Dans le cas de codage en RAM d'une animation complexe, il peut être nécessaire de morceller cette mémoire en différentes pages. Le module de codage a donc la charge de la gestion de l'espace disponible dans chaque page mémoire, ainsi que le chainage entre les différentes pages.



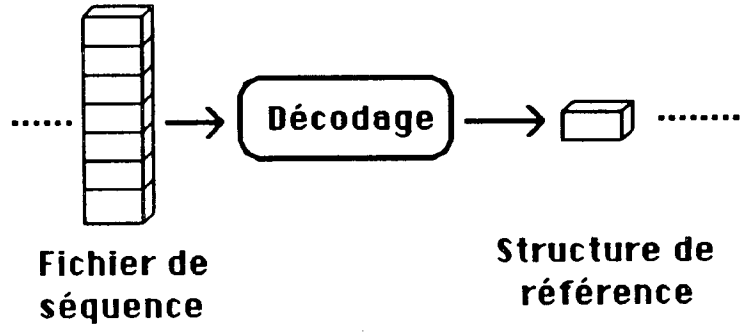
Chainage des pages mémoire pour le codage d'une séquence.

Le stockage sur disque dur, quant à lui, a conduit à la définition d'un format de fichier de séquence. Ce fichier regroupe en entête des informations générales, telles que le nombre des images à animer, leur configuration, la durée totale de l'animation... La suite du fichier regroupe les informations relatives à chaque trame d'animation, de la même manière qu'en RAM.

1.3 Le module de décodage

1.3.1 Le rôle du module de décodage

Le module de décodage, comme son nom l'indique, a pour fonction de décompacter les informations relatives à une animation. A chaque trame décodée, ce module fournit des informations dans le même format que celui établi pour la communication entre les applications et le module de codage (structure de référence).



Structures manipulées par le module de décodage.

Le décodage peut être explicitement demandé par une application, pour réaliser par exemple la fusion de 2 animations. Mais son utilisation la plus fréquente est réalisée par le module d'animation.

1.3.2 Le décompactage

Le module de décodage est chargé du décompactage des informations codées suivant le format établi au § 1.2.3.

A chaque trame, ce module parcourt en parallèle les bits d'état de la trame (à l'aide d'un pointeur d'état) et les informations de la trame (à l'aide d'un pointeur d'infos), dans le même ordre que le codage. Lorsqu'un bit d'état indique un changement, le pointeur d'infos permet la récupération de l'information concernée.

Ces informations sont stockées dans une structure de référence (qui peut être la même que celle utilisée lors du codage). Cette structure contient successivement pour toutes les trames de l'animation, toutes les informations relatives à ces trames.

Certaines informations interviennent dans le calcul des paramètres d'une même ressource matérielle, alors que d'autres sont totalement indépendantes (par exemple l'ordre de superposition n'a aucune incidence sur le zoom d'une mémoire, et inversement). Afin donc d'éviter les traitements systématiques et donc redondants, d'une trame à une autre, le module de décodage a la charge de tenir à jour une nouvelle liste de bits d'états, concernant cette fois-ci chacune des fonctionnalités matérielles du système.

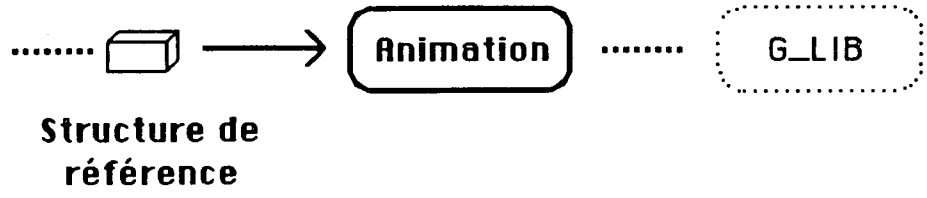
Par exemple les coordonnées d'un cell dans le repère de l'image et le facteur de zoom interviennent dans le processus de génération du zoom, et participent à la détermination de la boite écran. La modification d'un de ces paramètres a donc une incidence directe sur la boite écran. Le bit d'état associé à cette dernière est donc déterminé par un "OU logique" entre les bits associés aux coordonnées du cell et au facteur de zoom (au moins).

Enfin, le module de décodage transmet éventuellement un pointeur sur la liste des actions non temps réel relatives à la trame décodée.

1.4 Le module d'animation

1.4.1 Le rôle du module d'animation

Ce module est chargé de visualiser une animation, dont les informations ont été préalablement compactées et mémorisées en mémoire vive. A partir des informations délivrées par le module de décodage, pour chaque trame de l'animation, ce module génère des commandes d'animation de la bibliothèque G_LIB, synchronisées sur chaque retour de trame.



Structures manipulées par le module d'animation.

Le module d'animation est également chargé du pilotage des appareils vidéo, aussi bien en lecture qu'en enregistrement. A l'issue du décodage d'une trame, le module détermine si l'animation peut s'effectuer en temps réel, dans le cas où seules des informations temps réel ont été mémorisées, ou bien si elle nécessite un point de montage vidéo, dans le cas où des actions non temps réel se rapportent à la trame courante.

Les actions non temps réel sont explicites ou implicites, selon qu'elles interviennent explicitement dans la description de l'animation ou bien qu'elles sont réalisées à l'initiative du moteur pour la réalisation de l'animation. Dans tous les cas, ces actions correspondent à une insuffisance matérielle. Le premier cas correspond, par exemple, à la recopie d'un cell d'un acteur plan sur un décor lorsque cette recopie ne peut s'effectuer à la fréquence du balayage vidéo. Le second cas peut, par exemple, coïncider avec la non implémentation de l'attribut de rotation dans le système (la rotation doit alors intégralement être simulée par logiciel), ou bien encore avec le chargement d'une nouvelle planche d'animation (lorsque l'album d'un acteur ne tient pas dans une seule mémoire de trame).

Pour une animation exclusivement temps réel, l'enregistrement sur support vidéo peut avoir lieu en continu. En revanche, l'apparition d'actions non temps réel pour une trame nécessite d'arrêter tous les appareils vidéo avant la visualisation de cette trame, puis d'exécuter toutes les actions temps réel et non temps réel, avant de resynchroniser tous les appareils vidéo de la configuration. Ces commandes de synchronisation sont réalisées par les fonctions évoluées de la bibliothèque VIDEOLIB.

Enfin, des fonctionnalités, telles que ralenti, accéléré, retour arrière ou encore pas à pas, sont également intégrées au niveau du module d'animation, qui doit alors adapter la fréquence de décodage des trames à la fréquence de visualisation et gérer les appareils vidéo en conséquence.

1.4.2 Les commandes d'animation

A partir des structures de référence et des mots d'état, le module d'animation génère un ensemble minimum de commandes pour la visualisation de l'effet d'animation correspondant à la trame couramment traitée.

Les bits d'état associés à chacune des ressources matérielles déterminent ou non le calcul de nouvelles informations (boites, pondérations, profondeurs...).

De plus, dans le cas où un pointeur sur des actions non temps réel est retourné par le module de décodage, le module d'animation interprète chacune de

ces actions (numéro de code, suivi de paramètres) et appelle les primitives de traitement correspondantes de la bibliothèque G_LIB.

2. LE LANGAGE L_ANIM

L'environnement d'animation L_ANIM, basé sur un langage évolué LA, a été conçu afin de répondre à des besoins spécifiques d'animation. Ce langage autorise le mélange d'actions non temps réel et d'actions temps réel, décrites sous une forme très condensée, en gèrant au mieux les plages temps réel et non temps réel de l'animation. Il constitue ainsi un compromis efficace entre les applications totalement interactives, adaptées au temps réel, et les applications de montage image par image, n'exploitant en général aucune fonctionnalité d'animation du système.

Il permet, entre autres, de répondre à des besoins scientifiques, tels que l'animation de phénomènes physiques (évolution de l'impact d'une balle dans un blindage, perturbations météorologiques...). C'est, en outre, un environnement totalement ouvert à de nouvelles fonctionnalités.

2.1 Les concepts du langage

Les concepts manipulés par le langage d'animation sont proches de ceux établis au chapitre 3 de cette étude. Le mélange d'actions temps réel et non temps réel oblige cependant l'animateur à raisonner directement au niveau des images et non pas au niveau des acteurs. Toutefois, la syntaxe du langage LA est assez souple pour permettre à l'animateur de générer des séquences d'animation sophistiquées. Cette syntaxe est détaillée dans l'annexe III.

2.1.1 Les zones d'image

Les opérateurs du langage se rapportent aux mémoires d'image du système d'animation. Ces mémoires doivent être configurées de la manière suivante :

CONFIG <Idf_image> <Configuration>

avec configuration = COULEUR ou IDENTIFICATION.
L'identificateur d'une image est la chaine "IMAGE", indicée par un numéro (de 0 à 11).

exemple :

```
CONFIG IMAGE0    COULEUR
CONFIG IMAGE1    IDENTIFICATION
```

Afin d'utiliser les ressources matérielles que constituent ces images, un certain nombre de zones rectangulaires peuvent être définies. Ces zones présentent

une totale analogie avec les cells des acteurs manipulés par exemple dans l'application G_ANIM et sont définies comme suit :

ZONE <Idf_zone> <Idf_image> <entier><entier><entier><entier>

où les 4 entiers représentent les coordonnées image de la zone.

exemple :

ZONE DEST IMAGE0 100 100 500 500

2.1.2 Le temps

La syntaxe du langage est basée sur le principe d'association de traitements d'images, dans un sens très large incluant aussi bien les effets temps réel que l'analyse et la modification du contenu des images, à des instants précis de l'animation.

La définition d'intervalles de temps est à la base du langage. Cette notion permet, par exemple, de répéter une même action pendant un certain laps de temps, à partir d'une seule ligne de commandes.

Un intervalle représente un prédicat élémentaire pour le langage. En effet, les actions associées à cet intervalle ne sont effectivement réalisées que pour les trames vérifiant ce prédicat, c'est-à-dire comprises entre les bornes de l'intervalle (des conditions plus sophistiquées pourraient être très facilement implémentées).

-**entier** : dans le cas où l'intervalle est réduit à une seule image,
-**entier..entier** : spécifiant les bornes inférieure et supérieure de l'intervalle,
-**entier..entier PAS entier** : spécifiant les bornes inférieure et supérieure de l'intervalle, ainsi qu'un critère de validité.

exemple :

1..15 PAS 2 pour toutes les images d'indices impairs compris entre 1 et 15.

2.1.3 Les variables d'animation

Afin d'éviter les actions quasi-répétitives, ne changeant d'une trame à l'autre qu'au travers certains paramètres, toutes les instructions du langage peuvent être génériques, c'est-à-dire qu'elles peuvent être répétées avec des paramètres variables au cours du temps. Des variables, dans un sens général de programmation peuvent être définies et utilisées dans les instructions du programme. A chaque variable est associé un mode de variation temporel (constant, linéaire, uniforme...). Les variables permettent de caractériser

l'évolution des attributs des images de l'animation ou des traitements relatifs à ces images. La déclaration d'une variable entière doit respecter la syntaxe suivante :

VARIABLE <Idf_variable>
{ <mode_de_variation> <paramètres>
}

avec <mode_de_variation> parmi :

CONST <val>, où val est une valeur entière,
LINEAIRE <inf> <sup>,où inf et sup sont des valeurs entières représentant les bornes inférieure et supérieure du domaine de variation. La borne supérieure n'est atteinte qu'au dernier pas d'interpolation.
BRESHM <inf> <sup>, où inf et sup sont des valeurs entières représentant les bornes inférieure et supérieure du domaine de variation. Les valeurs comprises entre inf et sup sont réparties au mieux au cours de l'intervalle de temps.
UNIFORME <inf> <sup>, où inf et sup sont des valeurs entières représentant les bornes inférieure et supérieure du domaine de variation. Les valeurs comprises entre inf et sup sont réparties au mieux au cours de l'intervalle de temps.

exemple :

```
VARIABLE PAS
{     LINEAIRE      0     500
}
```

2.1.4 Les instructions du langage

Les instructions caractérisent les attributs associés aux images ou permettent d'effectuer des traitements sur le contenu de ces images. Des opérateurs spécifiques peuvent être employés dans une instruction (chargement d'une image, réalisation d'un filtre, translation d'un cell...).

Ces opérateurs sont de deux types différents. Ce sont des fonctions, dans un sens classique de programmation, lorsqu'ils délivrent un résultat, modifiant explicitement une zone. Ce sont des procédures lorsqu'ils ne délivrent pas de résultat. Les instructions du langage se limitent ainsi à des affectations, dans le cas de fonctions, et à de simples appels, dans le cas de procédures.

On trouve parmi les opérateurs, des procédures de translation, fenêtrage, zoom, profondeur, couleur de fond ou encore d'effet de grille. Ces procédures correspondent aux actions temps réel du système.

D'autres fonctions, telles que filtre, transfert, effacement, seuillage... agissent directement sur le contenu des images, et correspondent donc à des actions différées.

Une instruction doit toujours se rapporter à un intervalle de temps, définissant sa portée, c'est-à-dire l'ensemble des images de l'animation (1/25ème de seconde) pendant lequel l'instruction doit être exécutée. Enfin, plusieurs instructions peuvent être définies dans un même intervalle de temps.

La syntaxe d'une instruction a la forme suivante :

<zone_destination> [<liste_param_entiers>] <=
<nom_fonction>[<liste_param_entiers>](<liste_param_zones>);

dans le cas d'un opérateur délivrant un résultat, c'est-à-dire modifiant le contenu d'une zone donnée d'une image,

<nom_procédure> [<liste_param_entiers>]
(<liste_param_zones>);

dans le cas d'un opérateur ne délivrant pas de résultat.

Plusieurs instructions peuvent être associées au même intervalle de temps, suivant le format indiqué ci-dessous :

<intervalle> | <instruction1>;
<instruction2>;

2.1.5 Les procédures du langage

On entend par là les procédures bâties par l'utilisateur et non les fonctions de base du système. Une procédure regroupe un ensemble d'instructions associées à un ou plusieurs intervalles de temps. Elle évite les duplications de codes d'animation, puisque toutes ses instructions peuvent être exécutées par un simple branchement. Les instructions d'une même procédure peuvent ainsi être exécutées plusieurs fois au cours du temps et avoir même plusieurs occurrences au même temps d'animation, ce qui autorise entre autres la génération de processus parallèles et indépendants. Les ressources nécessaires à l'exécution d'une procédure peuvent être paramétrées, ainsi que les variables entières référencées dans le code de cette procédure. Une procédure peut ainsi être définie à l'aide de paramètres formels.

L'utilisation de procédures permet, d'une part, une programmation structurée et d'autre part la réutilisation d'actions d'un programme à un autre.

Le branchement à une procédure peut être réalisé à l'intérieur même d'une procédure. Cependant, aucune récursivité, directe ou croisée n'est autorisée.

La définition d'une procédure doit respecter la syntaxe suivante :

```
PROCEDURE <Idf_procédure>   [<liste_paramètres_entiers_formels]
                            (<liste_paramètres_zones_formels)
{<intervalle> |   <instruction>;
                  <instruction>;
 <intervalle> |   <instruction>;    ... ;
}
```

L'appel à une procédure s'effectue directement par son nom, suivi éventuellement d'un indicateur de temps et de paramètres entiers et zones.

<Idf_procédure> <temps_appel> [<liste_entiers>] (<liste_zones>);

Le programme principal est une procédure particulière. Il regroupe en effet un ensemble d'instructions associées à des intervalles de temps, mais il constitue le point d'entrée du programme d'animation (pas de branchement).

2.1.6 Exemple de programme

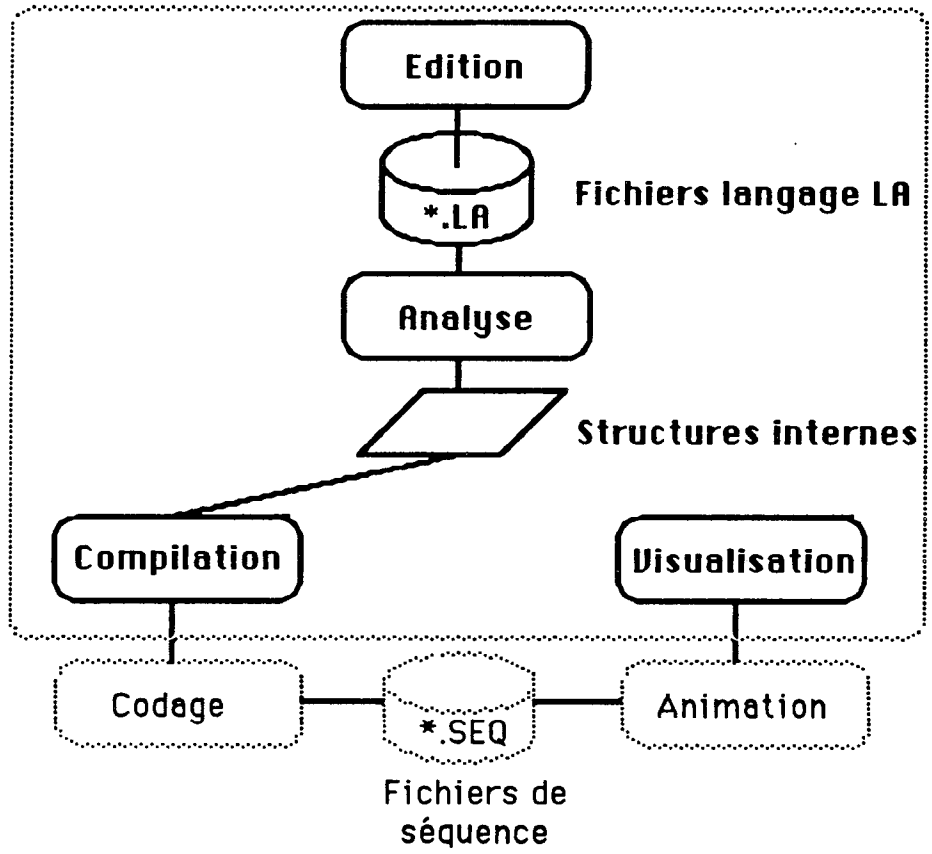
```
                        / Configuration des images /
CONFIG IMAGE0 COULEUR
CONFIG IMAGE1 COULEUR
CONFIG IMAGE2 COULEUR
                    / déclaration de 2 variables entières /
VARIABLE  indice      { LINEAIRE  0   9  }
VARIABLE  indice2     { LINEAIRE  0   99 }
                  / déclaration d'une séquence d'animation /
SEQUENCE 1  "C:\TPSREEL"

/ Procédure de numérotation: tous les fichiers de préfixe "NUMERO" et d'indice donné par la
valeur courante de la variable 'index' sont chargés successivement dans 'image' /
PROCEDURE numéroter [index] (image)
{ 1..100 |  image <= IMA[ABS,"C:\IMA\NUMERO",index];
}

PROGRAMME      / Programme principal d'une durée de 8 secondes /
{                / Numérotation d'une séquence temps réel toutes les 10 images /
   1..100 |   SEQUENCE  1 (IMAGE1,IMAGE2);
              numeroter [indice] (IMAGE0);
               /Numérotation continue de la même séquence temps réel /
 101..200 |  SEQUENCE  1 (IMAGE1,IMAGE2);
              numeroter [indice2] (IMAGE0);
}
```

2.2 L'environnement du langage

Cet environnement se compose de 4 modules principaux comme le fait apparaitre la figure suivante :



Représentation de l'environnement du langage.

- le module d'édition, chargé de la saisie d'un fichier d'animation dans le langage LA,

- le module d'analyse, réalisant les analyses syntaxique et sémantique d'un fichier d'animation,

- le module de compilation, chargé de calculer les informations relatives à toutes les trames de l'animation,

- le module de visualisation, proposant différents modes d'animation.

Une application interactive permet l'enchainement des actions entre les différents modules de l'environnement. Cette application a été exclusivement réalisée sur le PC.

En outre, une instruction particulière du langage permet de faire référence à des procédures externes à l'environnement. Un driver de fonctions de traitement d'images peut à cet effet être installé en résident dans la mémoire vive du PC. Son accès s'effectue par le biais d'un vecteur d'interruption logiciel prédéfini. Le premier paramètre d'appel à ce driver est le numéro de la fonction de traitement à appliquer. Cette notion permet d'étendre à volonté les fonctionnalités du langage et de le particulariser en fonction de besoins spécifiques, sans aucune intervention au niveau de l'environnement (analyse ou compilation).

2.2.1 Edition

Ce module permet de réaliser la saisie d'un fichier ASCII de commandes d'animation. En fait aucun éditeur n'a été développé spécifiquement pour l'environnement. Le module d'édition se contente de réaliser un overlay avec un éditeur déjà existant et disponible sous le système informatique (en l'occurrence MS-DOS).

En fait, le développement d'un éditeur spécifique permettrait d'intégrer une vérification syntaxique au cours de l'édition d'un fichier en langage LA, ainsi que des macros d'édition spécialisées pour le langage d'animation.

2.2.2 Analyseur

Ce module réalise en parallèle les analyses syntaxique et sémantique d'un fichier d'animation écrit en langage LA, et construit une arborescence abstraite de ce fichier. L'originalité de cette arborescence est qu'elle intègre la notion de temps, c'est-à-dire qu'elle regroupe des actions relatives à des instants différents de l'animation. Des prédicats temporels permettent ainsi de réaliser la factorisation de ces actions. Ces prédicats ont été limités à la seule condition d'appartenance à un intervalle de temps (cf § 2.1.2).

2.2.3 Compilateur

Ce module détermine, à partir de l'arbre abstrait construit dans la phase précédente, les informations relatives à toutes les trames de l'animation.

Pour chaque trame, l'arbre est évalué depuis la racine jusqu'aux feuilles. L'évaluation des prédicats temporels de l'arborescence permet d'en déterminer une instance pour chaque trame de l'animation. Le parcours de cette instance permet d'engendrer toutes les informations d'animation relatives à la trame traitée. Ces informations sont alors communiquées au module de codage du moteur d'animation, qui construit progressivement un fichier d'animation compacté.

2.2.4 Visualisation

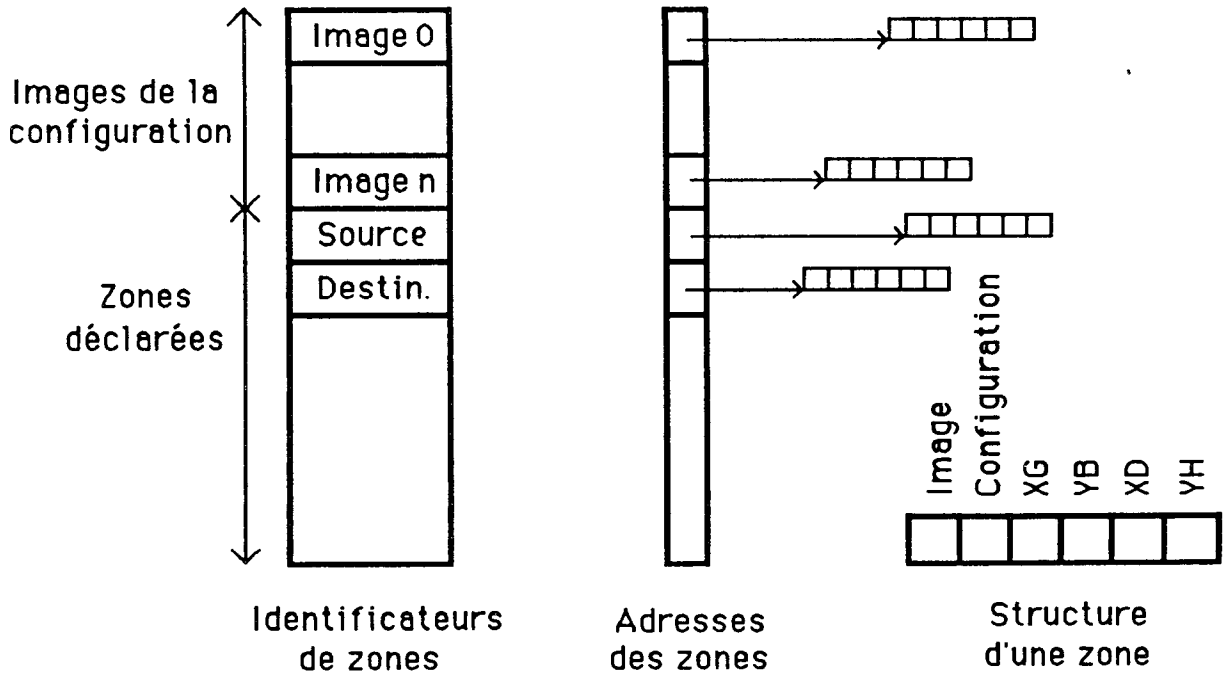
Ce module propose différents modes de visualisation de l'animation précédemment compactée. Ces modes caractérisent le fonctionnement de l'appareil vidéo en enregistrement (Pas d'enregistrement, Enregistrement synchrone, Enregistrement asynchrone), comme on l'a évoqué au § 3.5 du chapitre 2.

2.3 Les structures internes

L'analyse syntaxique et sémantique d'un fichier d'animation écrit en langage LA aboutit à la construction de structures de données et d'une arborescence abstraite. Cette dernière regroupe toutes les instructions du programme d'animation. Ces instructions font référence aux structures de données, que constituent les zones et les variables entières.

2.3.1 La structure de zone

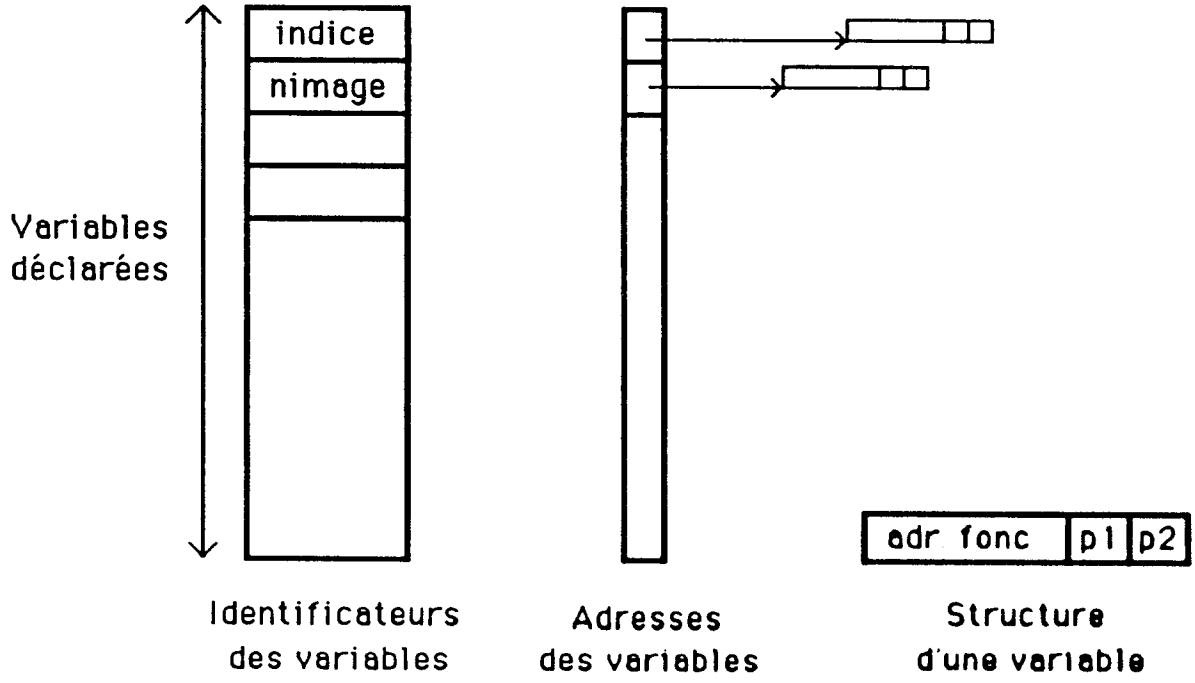
Les zones sont représentées par une structure renfermant les coordonnées de la zone et le numéro de l'image la supportant. Un tableau de chaines de caractères permet d'établir le lien entre un identificateur de zone et sa représentation informatique. Les images de la configuration déterminent implicitement des zones. Leurs structures sont stockées en début de tableau.



Structure et mémorisation de zones.

2.3.2 La structure de variable

Les variables sont représentées sous forme de structures, dont l'adresse est mémorisée dans un tableau de variables. Un tableau de chaines de caractères permet d'établir le lien entre un identificateur de variables et sa description informatique. Afin de banaliser le traitement des variables, chacune d'elles dispose de l'adresse de sa fonction d'évaluation.



Structure et mémorisation de variables.

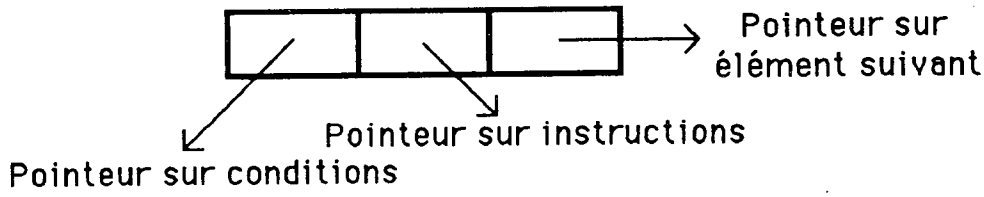
2.3.3 La structure de bloc

La définition de procédures aboutit à la notion de bloc, représenté par une liste chainée permettant de retrouver les prédicats et les instructions figurant dans ce bloc.

Les pointeurs sur les blocs sont stockés dans une table spécifique. Cette table est parcourue lors de la rencontre d'une instruction de branchement sur une procédure. La correspondance entre les identificateurs de procédures et les adresses des blocs est réalisée par un tableau de chaines de caractères. Le programme principal correspond au premier bloc du tableau.

L'analyse d'un bloc nécessite la gestion de variables formelles. Chaque variable (entière ou zone) rencontrée dans un bloc est comparée aux indentificateurs des variables formelles. Dans le cas où les identificateurs sont identiques, le numéro de la variable ainsi que le type "formel" sont mémorisés dans l'instruction où apparait la variable. Sinon, la variable est globale (sauf erreur de déclaration) et le numéro de définition de la variable ainsi que le type "global" sont mémorisés dans l'instruction correspondante.

Chaque élément de la liste constituant un bloc est une structure associée à un intervalle de temps.

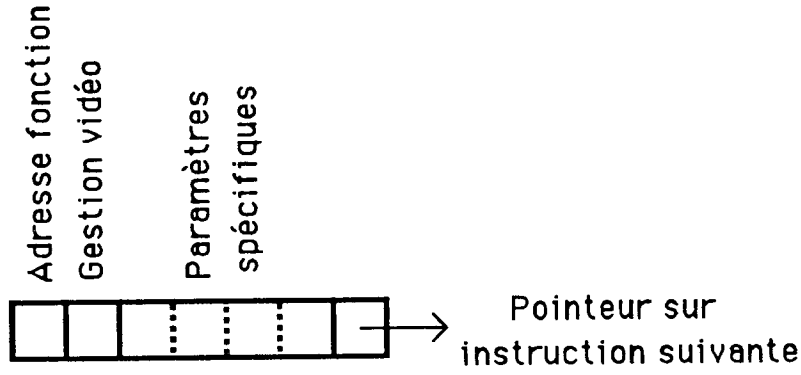


Représentation d'un élément du chainage d'un bloc.

Un prédicat est défini comme une liste chainée de conditions. Chaque élément de cette liste a la structure suivante :

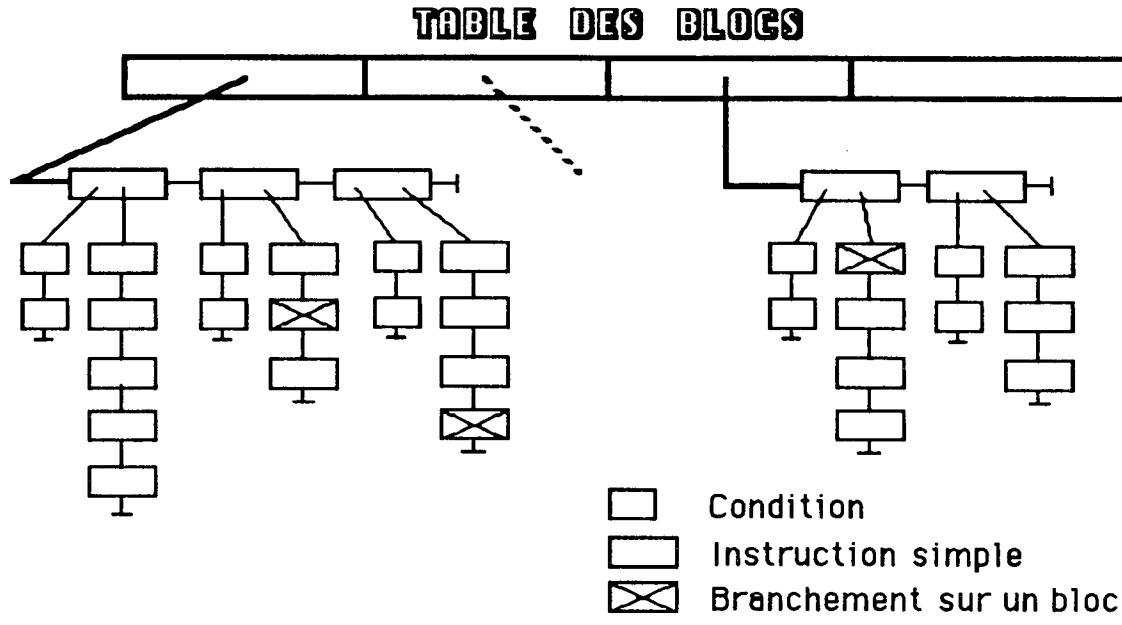
Structure d'une condition.

Les instructions associées à un même intervalle sont également chainées entre elles. Une instruction a la structure suivante :



Structure d'une instruction.

Le tableau des blocs de procédures, les structures chainées internes à ces blocs et les chainages implicites entre blocs (branchement) détermine une arborescence, comme le montre la figure suivante.



Représentation de l'arborescence abstraite d'un programme.

2.4 La compilation

La phase de compilation consiste à parcourir l'arbre abstrait, depuis la racine que constitue le programme principal jusqu'aux opérateurs de base. Ce parcours est réalisé pour chaque trame de l'animation, et permet de recueillir toutes les informations, temps réel ou non, relatives à ces trames. Les informations temps réel sont stockées dans une structure partagée avec le moteur d'animation. Les actions non temps réel, quant à elles, sont stockées dans une file.

2.4.1 Durée d'animation

La durée totale d'une animation est déterminée par les différents intervalles du programme principal. L'animation débute toujours sur la 1ère image du programme principal et se termine avec la plus grande borne rencontrée dans les intervalles.

exemple :

```
PROGRAMME
{
        10 .. 50    |    ...;        / de l'image 10 à l'image 50 /
        25          |    ...;        / à l'image 25 /
        1..5        |    ...;        / de l'image 1 à l'image 5 /
}
```

La durée totale de l'animation est de 50 images, soit 2 secondes.

2.4.2 Le temps de compilation

Pour chaque image du programme d'animation, toutes les instructions du programme principal sont parcourues, dans l'ordre de leur définition, en partant du premier intervalle de temps, jusqu'au dernier.

L'image couramment compilée est appelée temps absolu de compilation.

Lorsque ce temps absolu est compris entre les 2 bornes d'un intervalle (en tenant compte éventuellement du pas), toutes les instructions de l'intervalle sont compilées.

Cette compilation s'effectue toujours relativement au temps de début de l'intervalle où les instructions apparaissent. Ce temps de compilation relatif est appelé temps courant de compilation.

Pour l'exemple précédent, pour un temps de compilation absolu qui vaut 16, seules les instructions du premier intervalle sont compilées. Le temps courant de compilation de ces instructions est 7 (16 - 10 + 1).

Toutes les variables entières, paramètres d'une instruction en cours d'évaluation, sont compilées par rapport au temps courant de compilation, relativement à la durée de l'intervalle dans lequel elles apparaissent et aux bornes entières précisées lors de la déclaration.

2.4.3 La compilation d'une procédure

Lors de la rencontre d'une instruction de branchement à une procédure, le principe de compilation évoqué ci-dessus pour le programme principal est appliqué à la procédure. La structure définie par le bloc de la procédure est parcourue, en partant du pointeur correspondant dans la table des blocs.

Le temps d'appel à une procédure est défini comme le temps auquel la procédure doit être compilée. Ce temps est déterminé à partir du temps courant de compilation, auquel peut être ajouté un entier (1er paramètre d'appel à la procédure), qui joue le rôle de coefficient de déphasage de la procédure.

Chaque procédure a une durée, déterminée de la même manière que pour le programme principal. On définit le temps local de compilation, comme étant le temps d'appel à une procédure, modulo la durée de la procédure. Ceci permet d'exécuter cycliquement une procédure au cours de l'animation.

Ainsi, pour une procédure de durée 45 images et un temps d'appel de 100, le temps local vaut 11 = 100 - (2 * 45) + 1.

Enfin, le temps courant de compilation des instructions d'une procédure est calculé à partir du temps local à cette procédure et des indices de début des intervalles.

La compilation de procédures nécessite la gestion de 2 piles d'évaluation, une spécifique aux variables entières, l'autre aux variables zones.

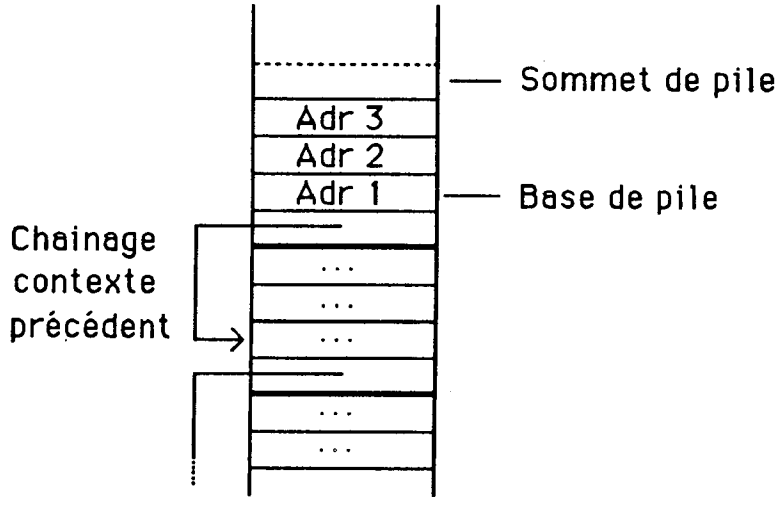
Lors du branchement à un bloc, les adresses des variables globales instanciant les paramètres formels de ce bloc sont empilées dans la pile d'évaluation correspondante, après avoir créé un chainage sur le contexte précédent.

Deux cas peuvent se présenter:

- l'appel a lieu au premier niveau : les variables passées en paramètres sont nécessairement globales (le programme principal n'a pas de paramètres formels). La structure de l'instruction d'appel contient alors les numéros de ces variables globales. On empile donc les adresses des structures correspondantes dans la pile d'évaluation, après avoir créé un chainage sur le programme principal (bp = 0).

- l'appel ne s'effectue pas au premier niveau : les paramètres peuvent être globaux ou formels. Si un paramètre est global, on empile directement l'adresse de sa structure. S'il est formel, on recherche dans le contexte précédent de la pile l'adresse associée au paramètres, et on l'empile dans le nouveau contexte.

La structure d'une instruction d'un bloc contient les adresses des variables globales (résolues à la compilation) et les numéros des paramètres formels. Le numéro d'un tel paramètre permet de retrouver dans le contexte courant d'évaluation, l'adresse de la variable globale qui l'instancie.



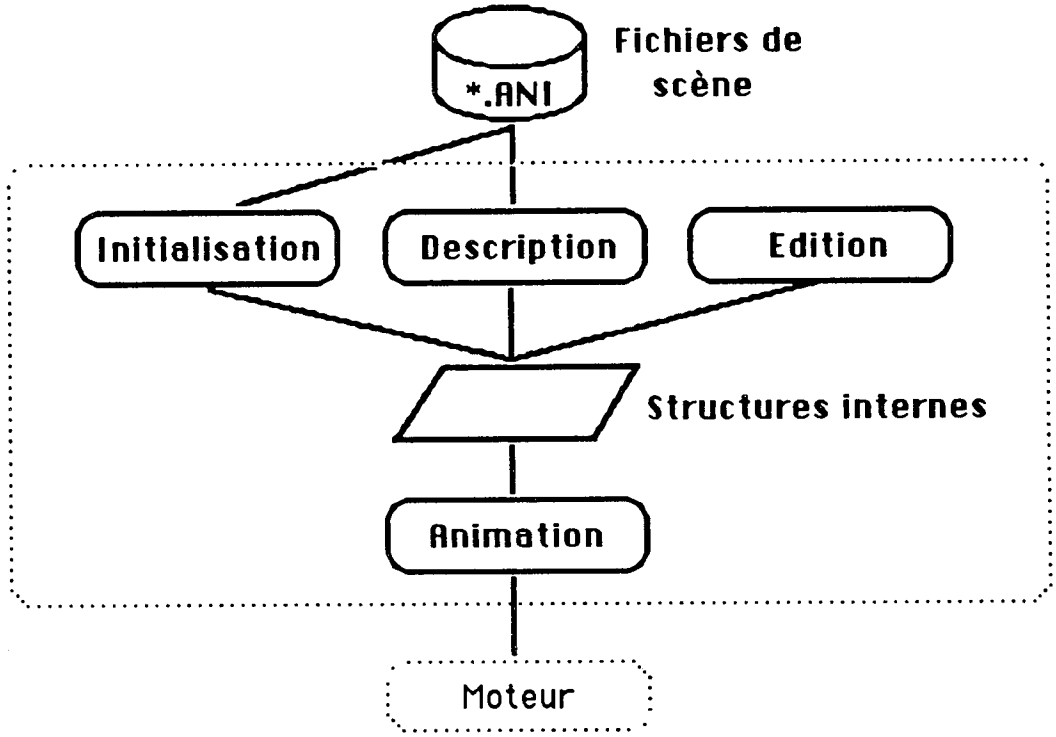
Représentation d'une pile d'évaluation.

3. L'APPLICATION G_ANIM

G_ANIM est l'application interactive de création de séquences d'animation. Elle constitue l'application pilote de l'environnement d'animation et met en application tous les concepts de la scène élaborés au second chapitre.

3.1 Organisation générale

Quatre modules principaux se dégagent, comme le montre la figure suivante:



Organisation des modules de G_ANIM.

- le module d'initialisation, chargé de la mémorisation des informations globales relatives aux acteurs de la scène ainsi que de son contexte,

- le module de description, spécialisé dans la saisie interactive des attributs de la scène,

- le module d'édition, regroupant des fonctions de mise au point de la scène d'animation,

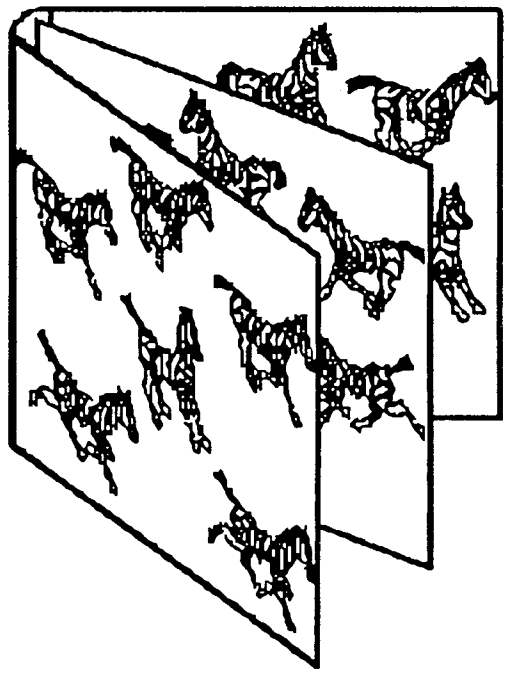
- le module d'animation, chargé du calcul des informations et de la visualisation de la séquence.

3.2 Le module d'initialisation

3.2.1 Le rôle du module d'initialisation

Ce module a la charge de la gestion des informations globales de la scène d'animation. Ceci comprend notamment, le nombre d'acteurs de la scène, et pour chaque acteur, sa configuration (couleur ou identification), le(s) nom(s) de fichier(s) constituant les images de l'acteur, ainsi que l'album de ses cells.

On a vu que l'album regroupe l'ensemble des cells d'un acteur (zone rectangulaire de l'image et point-guide associé). Ces cells peuvent ne pas appartenir à la même image, et on parle alors de plusieurs planches d'animation (cf §2.1.4 du chapitre 4). La consultation de l'album, par les autres modules de G_ANIM, permet de réaliser automatiquement les chargements des planches successives de l'acteur.



Représentation d'un album regroupant 3 planches.

Le module d'initialisation a la charge également du temps global d'animation et permet l'ajustement de cette durée par modification globale du temps ou encore insertion ou suppression de trames dans l'animation.

Il réalise également la sauvegarde de la scène d'animation dans un fichier d'animation, ainsi que la restitution d'une scène sauvegardée dans un tel fichier.

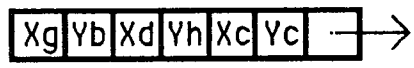
3.2.2 Les structures internes

A chaque acteur de l'animation est associée une structure regroupant toutes les informations générales de cet acteur.

```
char nom[];
char répertoire[];
char suffixe[];
int  typfich;
int  inf, sup;
int  configuration;
```

Ces informations sont gérées par le module d'initialisation.

L'album d'un acteur est représenté par une liste chainée d'éléments caractérisant chacun un cell (coordonnées des coins inférieur gauche et supérieur droit du cell dans le repère image et coordonnées du point-guide du cell relativement à son coin inférieur gauche). Le rang d'un élément dans la liste détermine le numéro du cell correspondant.

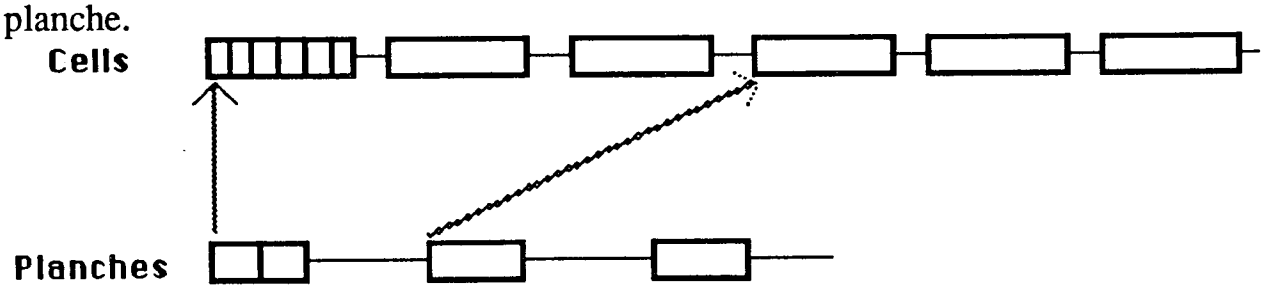


Structure d'un cell.

Les planches de l'acteur sont représentées par une liste chainée d'éléments, déterminant pour chaque planche le nombre de cells qui lui est associé. Le parcours de cette liste permet d'établir les appartenance des cells aux planches de l'acteur.

Structure d'une planche.

Un gestionnaire de cells permet la création, la modification, la suppression ou la consultation de cells de l'acteur indépendamment des planches auxquelles ils appartiennent. Un gestionnaire de planches permet, quant à lui, de créer ou de détruire une planche ou bien encore de modifier le nombre des cells associés à une planche.



Correspondance des cells et des planches.

L'album d'un acteur est initialisé à partir des informations relatives à chaque planche de l'acteur. Ces informations sont lues séparément à partir de chaque fichier image.

3.3 Le module de description

3.3.1 Le rôle du module de description

Le module de description est spécialisé dans la gestion des attributs de la scène. Il établit la correspondance entre les valeurs-clés des attributs et les instants-clés de leur saisie.

Les attributs manipulés sont les suivants :

- numéro de cell, choisie dans l'album de l'acteur,
- position du point-guide du cell à l'écran,
- zoom du cell, autour de son point-guide,
- fenêtrage, relativement à l'écran,
- pondérations associées à la fenêtre,
- profondeur de l'acteur.

Ces attributs constituent un sous-ensemble des attributs d'animation évoqués aux chapitres 2 et 3 de cette étude.

L'attribut de fenêtrage, relativement à l'image, n'a pas été implémenté. Cette fenêtre coïncide dans tous les cas avec le cell courant de l'acteur.
L'attribut de rotation a été implémenté du point de vue du dialogue, mais n'est pas encore pris en compte dans l'animation de la scène.
Enfin les outils de dialogue adaptés à la description de l'attribut de correspondance couleur/couleur n'ont pas encore été intégrés dans l'application. Cette interaction devra porter simultanément sur les couleurs, vraies ou fausses, d'un acteur.

Les attributs sont tous définis dans le temps. Des fonctions de calcul spécifiques permettent l'interpolation des valeurs-clés sur des intervalles de temps définis. Toutes ces informations sont regroupées dans un chronogramme, dont la gestion est intégralement confiée au module de description.

3.3.2 Les structures de description

L'animation de chaque acteur est déterminée par un ensemble d'attributs, coïncidant avec les concepts évoqués au chapitre 2 de cette étude.

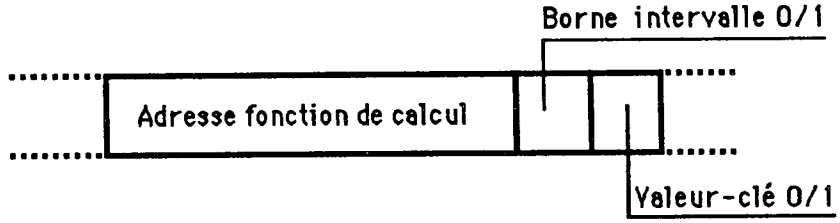
Chaque attribut est géré sous la forme d'une liste chainée. Un tableau regroupe, pour chaque acteur, l'ensemble des pointeurs sur les éléments de tête des listes d'attributs.

Chaque élément d'une liste contient, en standard, un pointeur sur l'élément suivant, une information temporelle et le type des informations qu'il véhicule. Chaque élément mémorise également des informations spécifiques à l'attribut auquel il se rapporte.

Constitution d'un élément d'une liste.

L'information temporelle n'est pas le temps absolu de l'animation, mais un coefficient relatif, dont les bornes de variation sont constantes (on a choisi l'ensemble des entiers positifs représentables sur 16 bits). Une primitive permet de convertir tout temps absolu en un indice temporel. Inversement, tout indice peut être mis en correspondance avec un temps absolu d'animation. Ceci autorise les modifications du temps total de l'animation, sans incidence sur les listes des attributs de l'animation.

Les informations véhiculées par les listes d'attributs sont de 2 types complémentaires. Pour chaque élément d'une liste, correspondant à un instant précis de l'animation, on trouve nécessairement le type de l'intervalle de calcul couramment défini (constant, linéaire, lissé...), ainsi que l'indication de début ou non d'un nouvel intervalle. De plus, tout élément d'une liste peut définir une nouvelle valeur de l'attribut correspondant. Ces deux types d'informations sont codés dans le même descripteur.



Constitution du descripteur d'un élément.

Des attributs globaux ont été intégrés à G_ANIM (couleur de fond, effet de fondu...). Ces attributs sont gérés suivant le principe établi ci-dessus. Un tableau global mémorise les pointeurs sur les listes associées à chacun de ces attributs.

3.3.3 Les fichiers d'animation

Les fichiers d'animation, de suffixe ANI, permettent la mémorisation sur disque dur d'une animation créée par G_ANIM. Ces fichiers regroupent toutes les informations nécessaires à l'élaboration de la scène d'animation.

En entête figurent les informations générales de l'animation, version du fichier, nombre d'acteurs, configurations, noms de fichiers des acteurs, durée

totale d'animation, ainsi que des informations relatives au contexte de travail dans G_ANIM, par exemple, le bloc d'édition courant, le temps de description courant...

L'album de chaque acteur est mémorisé, suivi des listes d'attributs, dans le même format que celui présenté ci-dessus. Un entier spécial symbolise la fin de chacune de ces listes.

Enfin apparaissent les listes d'attributs globaux stockées de la même manière que les acteurs spécifiques aux acteurs.

3.4 Le module d'édition

3.4.1 Le rôle du module d'édition

Le module d'édition a été spécialement conçu pour la mise au point interactive de scènes d'animation. Il propose des macro-commandes destinées, d'une part, à minimiser le travail de description de l'utilisateur, et d'autre part, à contrôler le rythme global de l'animation. Ces macros-commandes permettent, entre autres, de dupliquer des informations, en les répétant éventuellement plusieurs fois (création de cycles), de les supprimer ou encore de leur appliquer une nouvelle fonction de répartition temporelle (accélération d'une portion de scène...).

La manipulation des informations d'une scène conduit à la définition de 2 notions propres à l'édition, le bloc d'édition et le tampon d'édition.

Le bloc d'édition caractérise un intervalle de temps précis de la scène, ainsi qu'un ensemble défini d'attributs. Il n'a pas de signification physique, c'est uniquement un outil de sélection d'informations de la scène. La manipulation de ces informations s'effectue quant à elle par le biais du tampon d'édition, qui réalise la duplication d'un sous-ensemble des informations de la scène d'animation.

3.4.2 Les structures d'édition

Un bloc d'édition minimal permet la sélection d'une portion d'informations de l'animation, relativement à l'attribut courant de l'acteur courant. La représentation de ce bloc se limite à 2 entiers qui jouent le rôle de ses bornes inférieure et supérieure. Le bloc est caractérisé par le type d'information qu'il englobe. En effet, il est souvent souhaitable de ne tenir compte que des valeurs-clés de la liste indépendamment des fonctions de calcul qui s'y rapportent ou bien au contraire de tenir compte de l'intégralité des informations d'un attribut. Le bloc d'édition est donc également caractérisé par un type.

Les actions se rapportant au bloc d'édition sont les suivantes :

- effacer : les informations correspondant au type du bloc d'édition situées entre les bornes du bloc sont effacées.

- copier : les informations correspondant au type du bloc d'édition situées entre les bornes du bloc sont recopiées dans le tampon d'édition. Dans le cas où la valeur de l'attribut n'est pas définie à une borne du bloc, le système calcule automatiquement cette valeur, avant la recopie dans le tampon.

- couper : qui réalise successivement les opérations copier et effacer sur le bloc d'édition.

Ainsi le tampon d'édition permet le stockage temporaire d'informations du chronogramme de l'animation, en vue de leur traitement ou de leur recopie sur ce même chronogramme. Le tampon réalise le stockage d'une portion de liste d'un attribut de l'animation, sous une forme chainée. Les informations véhiculées par le tampon d'édition sont totalement analogues aux informations des listes d'attributs. Au premier élément de cette liste est associé un coefficient de temps, déterminant l'origine du tampon. Tous les coefficients temporels des autres éléments sont relatifs à ce coefficient et calculés à partir des primitives de conversion présentées plus haut.

En outre, le type des informations véhiculées par le tampon est signifié par un indicateur dans la structure associée au tampon.

Les actions de base se rapportant au tampon d'édition sont les suivantes :

- chrono : qui permet une dilatation du tampon. Une modification par un facteur commun de tous les coefficients temporels de la liste suffit à réaliser cette opération.

- coller : qui réalise la recopie des informations du tampon à partir du temps courant d'édition. Les informations du tampon remplacent les informations de l'attribut situées sous le tampon d'édition. Ce collage tient compte du type du tampon (fonctions de calcul, valeurs-clés).

La structure du tampon d'édition est la suivante :

```
structure tampon
{
 entier vide;        /* indicateur de validité des informations */
 entier type;        /* type des infos véhiculées : valeurs, intervalles */
 entier taille;      /* longueur du tampon */
 entier attribut;    /* type de l'attribut : position, cell, fenêtre... */
 char   *info;       /* pointeur sur la liste des informations */
}
```

Ce tampon minimal permet de copier une portion de la liste d'un attribut d'un acteur et de recopier cette liste sur la liste d'un même attribut d'un autre acteur. Il est souvent utile de pouvoir recopier globalement une portion entière de tous les attributs d'un acteur, voire même de recopier une portion de toute l'animation.

Nous avons donc , à partir des primitives de traitement du bloc et du tampon d'édition, créer des macros permettant de réaliser des opérations d'édition non seulement sur un attribut, mais aussi sur tous les attributs d'un acteur ou sur tous les attributs de l'animation.

La nouvelle structure permet donc le stockage de plusieurs attributs :

```
structure tampon
{
 entier vide;          /* indicateur de validité des informations */
 entier type_infos; /* type des infos véhiculées : valeurs, intervalles */
 entier taille;        /* longueur du tampon */
 entier type_liste;  /* 1 attribut, attributs d'1 acteur, attributs de l'anim */
 entier attribut;      /* type de l'attribut : position, cell, fenêtre... */
 entier acteur;        /* numéro de l'acteur recopié */
 char  *info_act[NB_ACT][NB_ATTR];  /*pointeurs sur infos d'acteurs */
 char  *info_glob[NB_GLOB];              /* pointeurs sur infos globales */
}
```

L'entier type_liste indique la complexité des informations du bloc. Dans le cas d'un seul attribut, le numéro est stocké dans le champ attribut. En fonction du type de cet attribut (spécifique à un acteur ou global), la liste est pointée soit par *info_act[acteur][attribut]*, soit par *info_glob[attribut]*.

Dans le cas de tous les attributs d'un acteur, les informations sont accessibles par les pointeurs du tableau *info_act[acteur]*.

Dans le cas d'un animation complète, tous les toutes les informations sont accessibles au travers tous les pointeurs des 2 tableaux *info_act* et *info_glob*.

3.5 Le module d'animation

3.5.1 Le rôle du module d'animation

Le module d'animation, comme son nom l'indique, est chargé de l'animation de la scène décrite et mise au point par les modules définis ci-dessus.

Son rôle est de calculer toutes les informations relatives à chaque trame de la scène d'animation. Ces informations sont communiquées au fur et à mesure au moteur d'animation, qui se charge de les compacter.

Le calcul s'effectue attribut par attribut, en fonction des informations-clés et des intervalles de calcul du chronogramme. Dans le cas où l'intégralité de la séquence ne peut être calculée (défaut de mémoire vive), le module propose ou

réalise automatiquement l'animation par parties de cette séquence, c'est-à-dire les enchainements de calculs et de visualisation.

A l'issue du calcul d'une séquence (ou portion de séquence), différents modes de visualisation sont proposés :

- Standard
- Pas à pas,
- Bouclé,
- Asservi.

Dans le cas d'une animation standard, l'animation se déroule en continu sur l'intervalle d'animation précisé par l'utilisateur.

L'animation pas à pas, permet à l'utilisateur de se déplacer dans la séquence trame à trame, en avant ou en arrière, ou de se rendre directement à une trame particulière de l'animation, s'il le désire. Ce mode facilite considérablement le travail de mise au point de l'animateur.

Le mode bouclé permet de répéter à l'infini la visualisation de la séquence.

Enfin, l'asservissement du temps permet de visualiser la séquence suivant le sens et la vitesse induits par le déplacement d'un dispositif extérieur de dialogue (souris, tablette...). L'asservissement d'un attribut, quant à lui, permet l'interaction sur cet attribut au cours de la visualisation de la séquence, qui demeure inchangée pour les autres attributs.

3.5.2 Les structures de calcul

Il est nécessaire d'effectuer des calculs relatifs à la scène dans 2 cas précis d'utilisation :

- pendant la phase de description, afin de visualiser l'animation à un instant précis. Nous parlerons alors de calcul d'attitude.

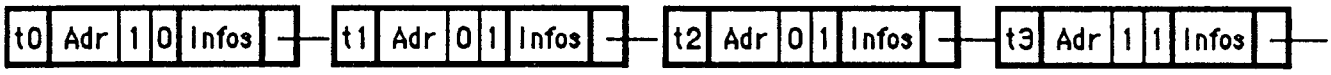
- avant la visualisation complète de la séquence, c'est-à-dire lors du codage des fonctions d'animation. Nous parlerons alors de calcul de séquence.

Le calcul d'attitude

Le calcul d'attitude consiste à déterminer tous les paramètres de l'animation en un instant défini.

Ce calcul est réalisé par le parcours, pour chaque attribut de l'animation, spécifique ou global, de la liste des informations qui lui sont associées. Ce parcours est fonction de l'intervalle de calcul auquel appartient le temps courant de calcul. Comme on l'a vu au § 3.3.2 de ce chapitre, la liste peut regrouper des éléments ne contenant qu'une indication de type d'intervalle de calcul (transition

d'un intervalle à un autre). En revanche, une valeur-clé de l'attribut s'accompagne toujours du type de l'intervalle de calcul dans lequel elle apparait.



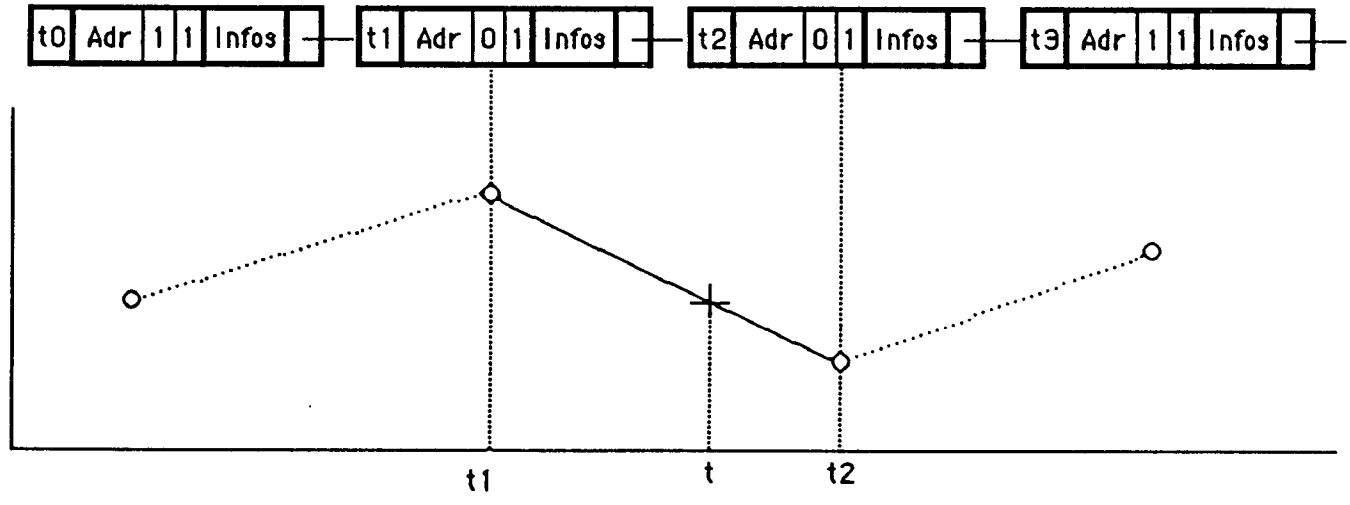
Chainage sur un intervalle possèdant 3 valeurs-clés.

Bien que la valeur courante d'un attribut ne soit pas nécessaire lors de la définition d'un nouvel intervalle, il est indispensable d'associer des valeurs aux bornes de l'intervalle lors du calcul.

Dans le cas où aucune valeur n'est définie pour la borne inférieure d'un intervalle, on recherche dans les intervalles précédents la dernière valeur définie de l'attribut. On trouve nécessairement une valeur, puisque les attributs de la scène sont initialisés par défaut.

Dans le cas où aucune valeur n'est définie pour la borne supérieure d'un intervalle, on recherche dans l'intervalle la dernière valeur définie, ou à défaut, la valeur associée à la borne inférieure.

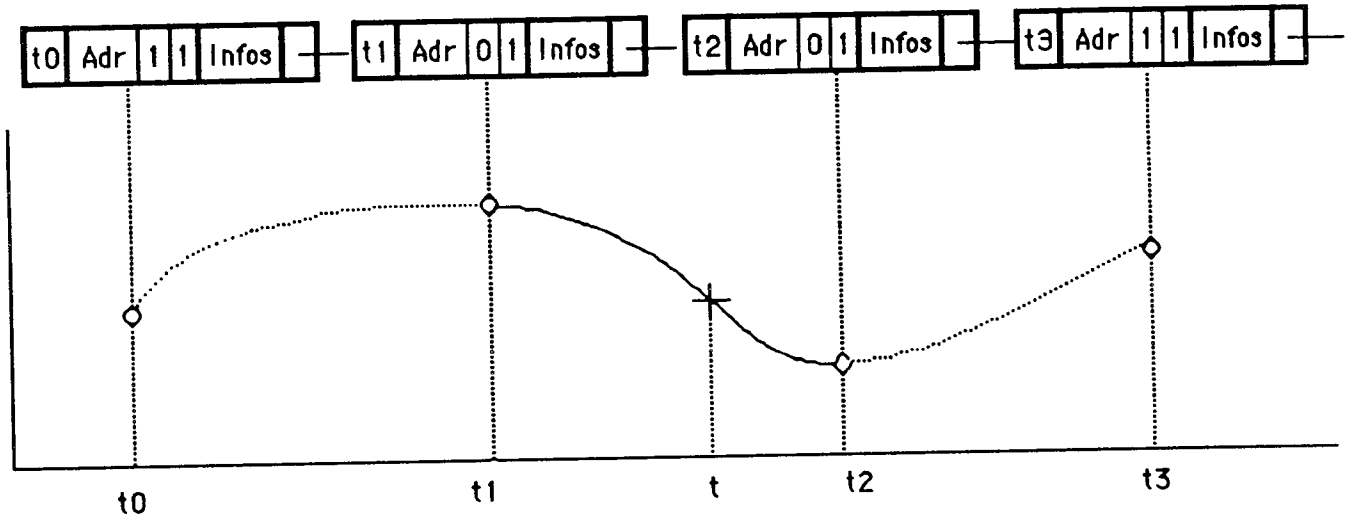
Dans le cas d'une fonction de calcul discrète ou linéaire, la liste de l'attribut est parcourue, jusqu'à la détermination de 2 bornes encadrant le temps courant d'évaluation. La fonction de calcul détermine alors par interpolation linéaire la valeur associée au temps courant.



Calcul sur un intervalle linéaire.

Le cas d'un fonction de calcul lissée, à base de splines, est plus complexe à traiter. En effet, toutes les valeurs (explixites ou implicites) de l'intervalle lissé sont nécessaires à l'évaluation de l'attribut. Le parcours de la liste de l'attribut

conduit à l'initalisation d'un tableau avec toutes les valeurs rencontrées dans l'intervalle, ainsi qu'un tableau de correspondance temporelle (indices des temps des éléments de l'intervalle). Une fonction de lissage réelle réalise le calcul des points de la courbe d'abscisse le temps et d'ordonnée le support des valeurs de l'attribut. On peut alors considérer que la fonction lissée est la suite des segments de droite reliant les points successifs du lissage. Déterminer la valeur courante de l'attribut revient à trouver, dans le tableau d'abscisse, 2 indices de temps encadrant le temps d'évaluation, puis à réaliser une interpolation linéaire entre les valeurs associées à ces indices.



Calcul sur un intervalle lissé.

Remarques :

Il est nécessaire que la courbe de lissage passe exactement par les points de description, afin de conserver un calage strict entre la phase de description et de visualisation.
D'autre part, afin d'obtenir des résultats précis, il est indispensable de normaliser les échelles d'abscisse et d'ordonnées.

Le cas particulier des trajectoires et évolutions se traite encore différemment. La trajectoire est en effet le résultat d'un calcul à partir d'un ensemble de points-clés. La fonction de calcul utilisé pour ce calcul peut être soit linéaire soit lissée. Afin de standardiser le traitement de ces différents cas, la trajectoire se ramène à un tableau de points 2D et à un tableau indiquant l'abscisse curviligne normalisée de chacun de ces points. De la même manière l'évolution le long de cette trajectoire est une suite de points d'abscisse le temps et d'ordonnée l'abscisse curviligne. La fonction de calcul de l'évolution peut être discrète, linéaire, lissée ou bien quelconque, par exemple dessinée. Calculer les coordonnées d'un point 2D à un instant de l'animation s'effectue en 2 étapes. Tout d'abord la détermination de l'abscisse curviligne correspondant au temps d'évaluation. Ensuite, la recherche de 2 valeurs encadrantes dans le tableau d'abscisse curviligne de la trajectoire et le calcul par interpolation linéaire des coordonnées du point recherché.

Le calcul de séquence

Le calcul de la séquence consiste à déterminer toutes les informations relatives à la scène successivement pour chaque trame de l'animation. Ces informations sont communiquées au fur et à mesure au module de codage du moteur d'animation.

Une première approche consiste à considérer le calcul de la séquence comme une suite de calcul d'attitude. Ceci suppose qu'à chaque trame les listes sont parcourues depuis leur premier élément, mais surtout que les calculs d'intervalle, de trajectoires et d'évolution, sont repris à chaque trame, alors qu'ils restent valides sur toute la durée de l'intervalle ou de la fonction d'évolution.

Ceci nous a conduit à déterminer des structures de stockage intermédiaires, en vue de minimiser le parcours des listes d'attributs et de conserver les résultats de calculs complexes. Ces structures contiennent pour chaque attribut, les bornes de l'intervalle couramment traité, les pointeurs sur les éléments appartenant à cet intervalle, ainsi que des pointeurs sur des tableaux de données, telles que graphes, trajectoires, évolutions...

A chaque nouvelle trame, pour chaque attribut de l'animation, le temps courant d'évaluation est comparé aux informations de temps mémorisées dans la structure.

Dans le cas où le temps d'évaluation n'appartient pas à l'intervalle de temps défini, toutes les informations relatives à l'intervalle ou à l'évolution précédents sont effacées. Des nouveaux calculs sont menés sur le nouvel intervalle ou la nouvelle évolution. Les résultats de ces calculs sont mémorisés dans les structures appropriées et on se ramène au cas suivant.

Dans le cas où le temps courant d'évaluation est compris dans les bornes de l'intervalle, une fonction de calcul utilise les informations mémorisées pour évaluer l'attribut. Notamment, dans le cas d'utilisation d'une fonction lissée, les indices ayant permis le calcul de la dernière valeur du graphe associé, peuvent être utilisés comme points de départ dans la recherche du tableau d'indices, plutôt que de reparcourir systématiquement la totalité du tableau. On utilise ainsi sur la propriété de continuité de la fonction manipulée.

4. LE DIALOGUE DE G_ANIM

L'application d'animation G_ANIM a été conçue autour de la bibliothèque G_LIB, pour les actions graphiques de base, et surtout autour de la bibliothèque G_DIAL pour tout l'aspect dialogue.

Différentes actions sont proposées à l'utilisateur à l'aide de plusieurs menus spécifiques. Celui-ci dispose en outre d'outils de dialogue puissants, bâtis sur les concepts évoqués dans la dernière partie du chapitre précédent.

Nous allons présenter rapidement quelques unes des fonctionnalités de G_ANIM, du point de vue de l'ergonomie et du dialogue avec l'animateur.

4.1 Organisation générale

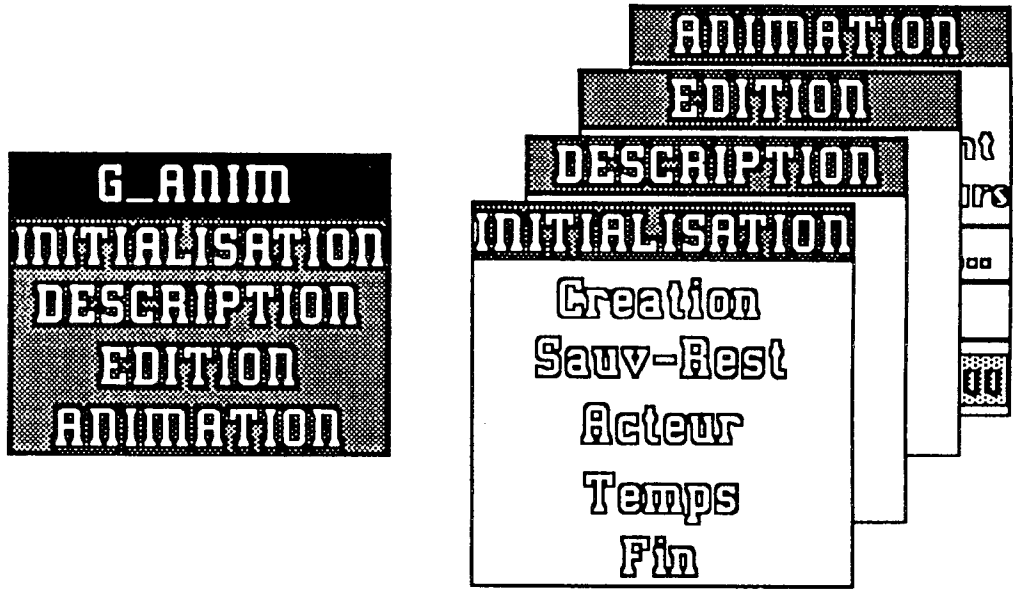
Ce premier paragraphe regroupe les fonctions utilisées les plus couramment et les objets de dialogue communs à toutes les actions d'animation.

4.1.1 Choix du mode d'actions

Etant données la complexité des fonctions d'animation de G_ANIM et les contraintes ergonomiques d'utilisation de ces fonctions (pas de menus hiérarchisés...), toutes les actions d'animation sont réparties en 4 modes spécifiques, suivant d'ailleurs la découpe logicielle évoquée au chapitre précédent.

Ces 4 modes correspondent à des contextes complets de travail, et le passage d'un mode à un autre s'effectue très simplement.

Chacun des 4 modes possède un menu d'actions spécifique. Ce menu a pour titre le contexte courant de travail. Le passage d'un contexte à un autre s'effectue par cliquage sur ce titre. Un menu proposant les 4 contextes est alors affiché. Le choix d'un contexte dans ce menu provoque l'affichage des menus et outils de dialogue correspondants. Cet affichage est réalisé automatiquement par une fonction de traitement spécifique associée au domaine de choix des modes (cf § 3.4.5.3 du chapitre précédent).

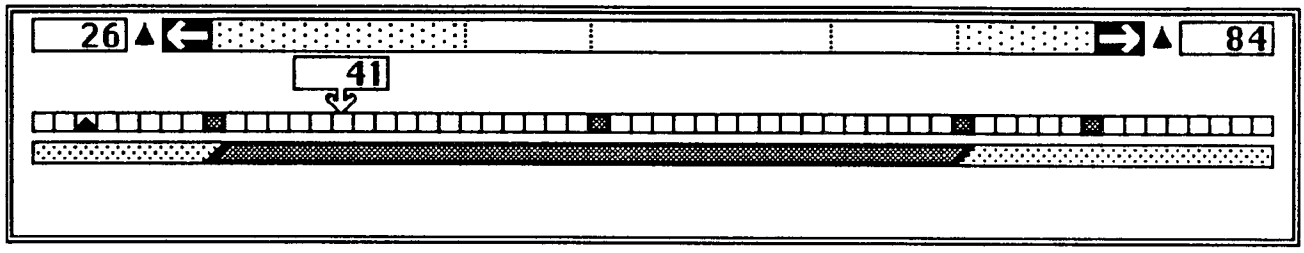


Les 4 contextes de travail et les menus associés.

4.1.2 Le chronogramme

Le chronogramme est une vue chargée de la présentation des informations temporelles de l'animation et de la gestion du dialogue spécifique au temps. Il est

visualisé en bas de l'écran et reste affiché pour les 3 contextes d'initalisation, description et édition.



La représentation du chronogramme.

Dans la partie supérieure est affiché un ascenseur horizontal, chargé de déterminer la portion d'animation visible dans le chronogramme. La partie mobile de l'ascenseur est découpée en trois zones permettant non seulement de déplacer le mobile, mais aussi de modifier sa taille. Les deux compteurs situés à droite et à gauche de l'ascenseur précise l'ensemble des trames simultanément visualisées.

Une fonction de traitement spéciale à été associée à l'ascenseur (cf § 3.4.5.2 du chapitre précédent) afin de réaliser l'affichage des informations temporelles pendant la séance de dialogue sur ce domaine.

Dans la partie centrale figurent une échelle horizontale graduée sur laquelle se déplace un compteur précisant le numéro de la trame courante. Une fonction spéciale d'écho bas (cf §3.2.3.1 du chapitre précédent) réalise le déplacement de ce compteur ainsi que la mise à jour du numéro de la trame.

Des triangles peuvent être positionnés sur cette échelle et joue le rôle de marques pour l'utilisateur. De plus, les graduations foncées symbolisent les instants de définition de valeurs-clés relatives à l'attribut couramment sélecté (cf § 3.1.4). Le ruban coloré affiché sous l'échelle représente les intervalles associés aux différentes fonctions de calcul de l'attribut. A un type de fonction est associée une couleur particulière (discrète, linéaire, lissée...).

La partie basse de la vue chronogramme est réservée au mode d'édition (cf § 3.4).

4.1.3 La vue des acteurs

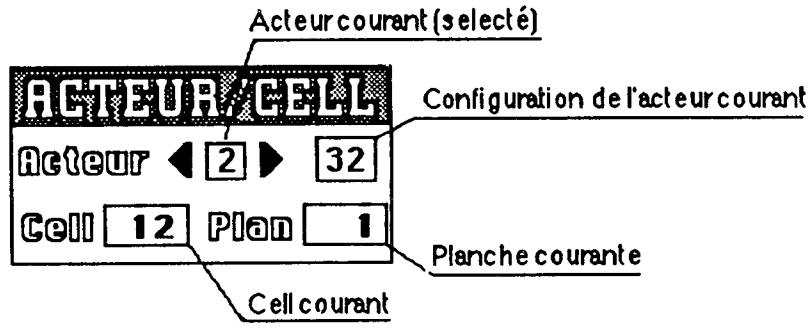
G_ANIM autorise la manipulation de plusieurs acteurs, correspondant aux différents niveaux de cells de l'animation.

La sélection de l'acteur courant s'effectue à l'aide des flèches situées de part et d'autre du numéro de l'acteur précédemment sélecté. Une fonction d'écho bas (cf 3.2.3.2 du chapitre précédent) est associée à chacun des domaines déterminés par les flèches. Cette fonction d'écho réalise automatiquement l'incrémentation ou la décrémentation du numéro de l'acteur courant et visualise ce numéro entre les flèches, ainsi que la configuration de l'acteur concerné.

Il existe donc toujours un acteur sélecté. Toutes les actions de saisie ou d'édition d'attributs spécifiques se rapportent directement à cet acteur.

La vue des acteurs joue également le rôle de témoin des informations particulièrement utiles à l'animateur, telles que le numéro du cell courant et le numéro de la planche à laquelle celui-ci appartient. Ces informations sont mises à jour à chaque changement du temps courant.

Deux objets valeurs jouent le rôle de ces témoins, qui sont également utilisés pour la saisie des informations de cells et de planches.

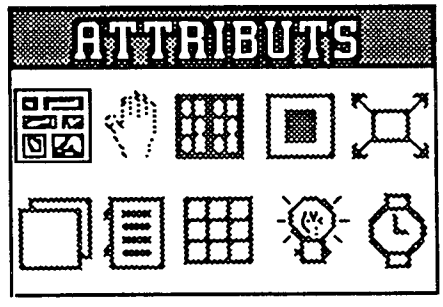


La vue des acteurs.

4.1.4 La vue des attributs

Un menu spécialisé regroupe tous les attributs de l'animation. La sélection d'une entité dans ce menu détermine l'attribut courant de l'animation. Les actions de saisie et d'édition se rapportent toujours à cet attribut.

Une fonction de traitement spécifique est associée au menu des attributs (cf § 3.4.5.3), afin de mettre à jour toutes les informations relatives au nouvel attribut sélecté, notamment dans le domaine du chronogramme (valeurs-clés et fonctions de calcul).



La vue des attributs de l'animation.

Le menu propose 2 rangs d'icônes, associées chacune à un attribut d'animation.

Le premier rang regroupe les attributs spécifiques à l'acteur courant. De la gauche vers la droite, on peut reconnaitre:

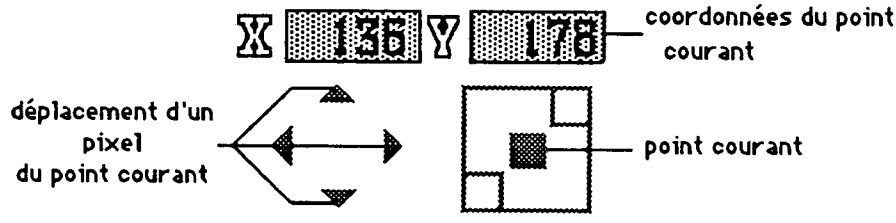
- le numéro de cell de l'acteur courant,
- la trajectoire suivie par l'acteur,
- l'effet de fenêtrage,
- la pondération de fenêtre,
- l'effet de zoom.

Le second rang regroupe les attributs globaux de l'animation. De la gauche vers la droite, on peut reconnaitre :

- l'ordre de superposition des acteurs,
- l'effet de volet ou sur-fenêtrage,
- l'effet de fondu ou de grille,
- la couleur de fond,
- les interruptions ou actions non temps réel.

4.1.5 La vue de travail

Une vue spéciale de travail est partagée entre plusieurs modes. Cette vue propose des outils pour la saisie de coordonnées dans l'écran.



Les différents symboles de la vue de travail

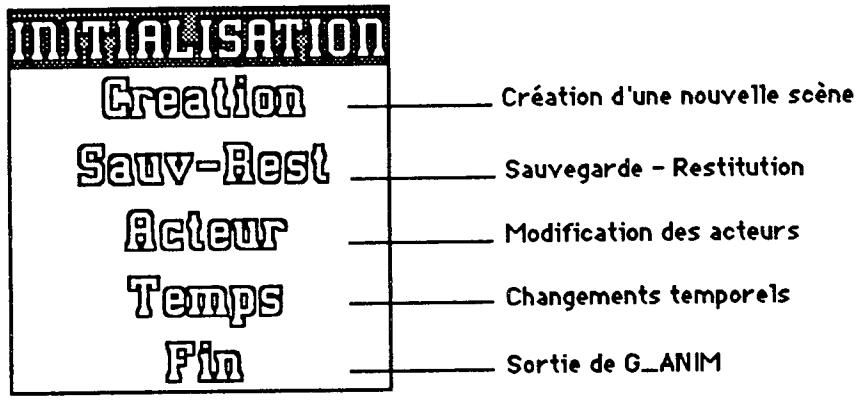
Deux objets valeurs représentent les coordonnées du point couramment traité, et des pictogrammes, en forme de triangles permettent d'ajuster ces coorodonnées. Les flèches horizontales caractérisent les déplacements d'un pixel sur la gauche ou la droite, les flèches verticales les déplacements d'un pixel vers le haut ou le bas.

A chaque pictogramme correspond un domaine, auquel est associé une fonction de traitement des événements du dispositif de dialogue (action de clic).

La vue de travail est souvent utilisée pour la saisie interactive d'objets de dialogue de type morphologie rectangulaire. Cette morphologie est représentée par un carré à l'intérieur duquel figurent 3 petits carrés, permettant de choisir un point caractéristique (coin inférieur gauche, centre, coin supérieur droit).

4.2 Le mode d'initialisation

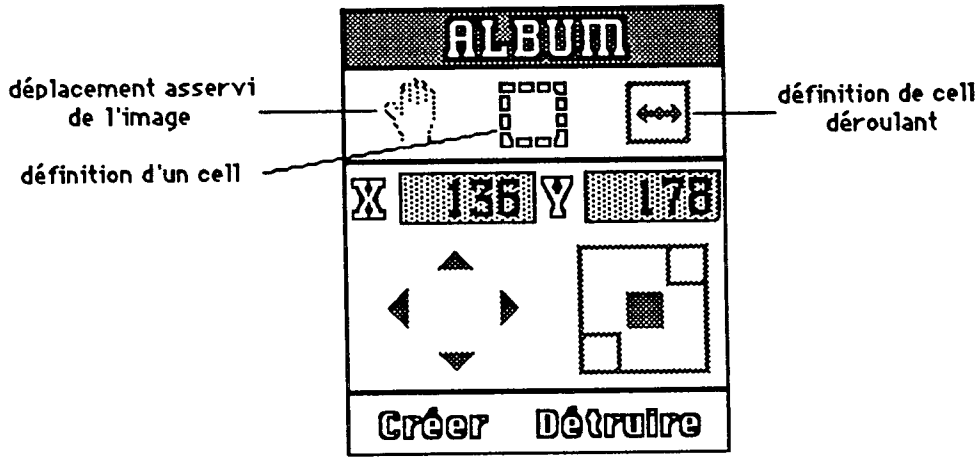
Le mode d'initialisation présente le dialogue spécifique à la gestion des acteurs d'une scène d'animation, ainsi que de l'environnement général de l'animation. Il propose, entre autres, des outils de dialogue pour la création et la mise au point d'un album, ainsi qu'un dialogue spécifique à la sauvegarde et la restitution de scènes d'animation.



Représentation du menu du mode d'initialisation.

4.2.1 L'album

L'album d'un acteur regroupe l'ensemble de ses cells, répartis éventuellement sur plusieurs planches images. Des outils spécifiques sont proposés à l'animateur pour la gestion de l'album d'un acteur.



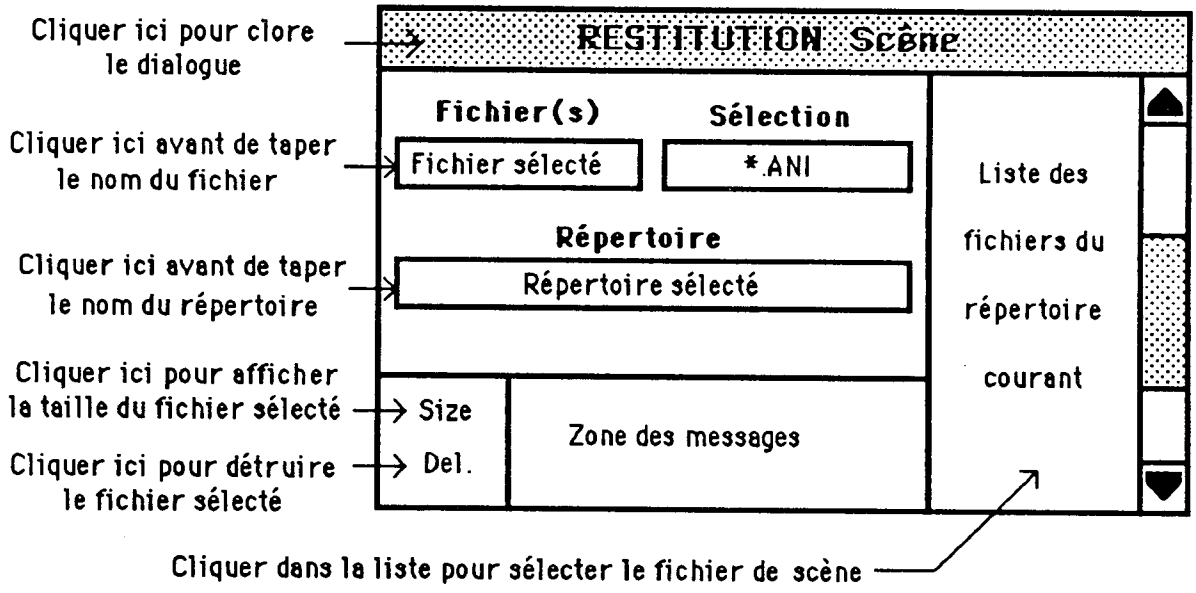
Le dialogue spécifique à l'album.

Un cell est matérialisé par une morphologie de type rectangle dans l'image de l'acteur. Les informations du cell courant sont affichées dans les compteurs dédiées ainsi que dans le domaine "ACTEUR" présenté au § 3.1.3.

L'animateur a la possibilité d'intervenir directement sur la morphologie, ou d'utiliser les outils spécifiques à la saisie de points. Chacune des actions a une incidence directe sur la visualisation à l'écran.

4.2.2 Sauvegarde et restitution

Une vue spécialisée pour la sauvegarde/restitution de scène a été construite, afin de permettre à l'utilisateur d'accéder aux informations du disque dur du système sans quitter l'application. Cette vue met en partique les concepts élaborés dans la première partie de ce chapitre et illustre bien la notion de séance de dialogue sur un objet complexe (cf 3.4.3.2 du chapitre précédent).



Représentation de la vue de dialogue pour les sauvegardes et restitutions.

La sélection de l'entité Sauv-Rest du choix principal du mode d'initialisation provoque l'affichage de la vue de dialogue présentée ci-dessus et ouvre une séance de dialogue sur cet objet complexe. La vue est composée d'un ensemble d'objets élémentaires, de types valeur et choix. Chacun des domaines dispose d'une fonction de traitement particulière. Par exemple, le choix d'un fichier provoque automatiquement l'affichage du nom correspondant dans la valeur de titre "Fichier", ou la saisie directe du nom de fichier détermine ou non l'existence dans le choix de ce fichier...

La séance de dialogue se termine par le clic dans la zone réservée au titre de la vue, qui provoque l'effacement de la vue. L'application peut alors récupérer les informations issues du dialogue.

4.3 Le mode de description

Le mode de description regroupe tous les choix et les outils nécessaires à la description interactive d'une scène d'animation. Nous allons illustrer, dans ce paragraphe, le dialogue spécifique à 3 fonctions particulières de description : la saisie de trajectoires, la saisie de fonctions d'évolution et la saisie de fenêtre.

4.3.1 Trajectoire

Un acteur peut être amené à se déplacer au cours de l'animation. Ce déplacement induit d'une part la notion de trajectoire, représentant le lieu de déplacement de l'acteur, et d'autre part la notion d'évolution, permettant de fixer des rendez-vous précis le long de la trajectoire.

Une trajectoire est nécessairement continue et une fonction de calcul appliquée à tous les points-clés permet de déerminer un ensemble de points, constituant les sommets d'une ligne polygônale, approchant la trajectoire théorique. La fonction de calcul peut être une fonction d'interpolation linéaire ou encore une fonction de lissage, basée sur la méthode mathématique des splines.

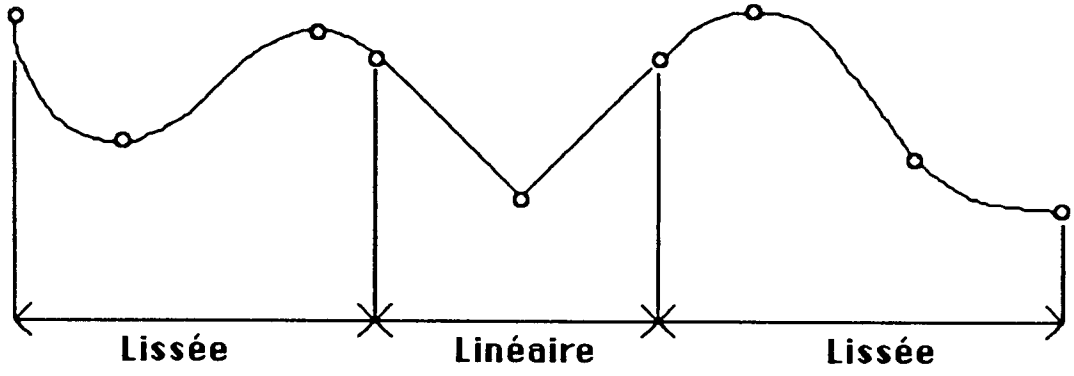
Afin d'autoriser une plus grande souplesse de description de trajectoire, nous avons considéré qu'une trajectoire peut être constituée d'un ensemble de sous-trajectoires, calculées à l'aide des 2 fonctions citées ci-dessus. Un choix propose en permanence le type de saisie de la trajectoire.

Mode de saisie d'une trajectoire.

Une trajectoire peut être représentée à l'aide d'une suite de morphologies de type ligne brisée et spline. Ces morphologies sont jointives; deux morphologies successives se partagent en effet un même point de contrôle. Des contraintes peuvent être ajoutées à la jointure des sous-trajectoires, telles que continuité de la dérivée première...

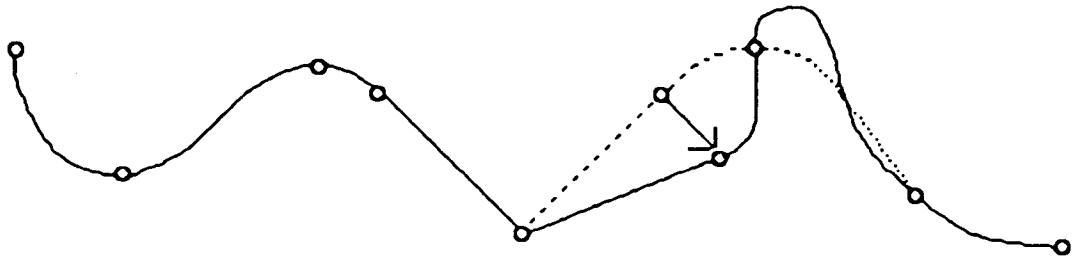
La saisie d'une trajectoire est totalement interactive; l'animateur voit se construire la trajectoire au cours du dialogue. Les points de contrôle de la

trajectoire sont définis par les clics de l'utilisateur et sont matérialisés le long de la trajectoire. Lors des déplacements du dispositif de dialogue, la trajectoire est affichée, en prenant en compte le point courant de dialogue, ce qui autorise un ajustement précis de la courbe. L'utilisateur a la possibilité de changer de fonction de calcul en cours de saisie. La trajectoire n'est enregistrée qu'après sa validation.



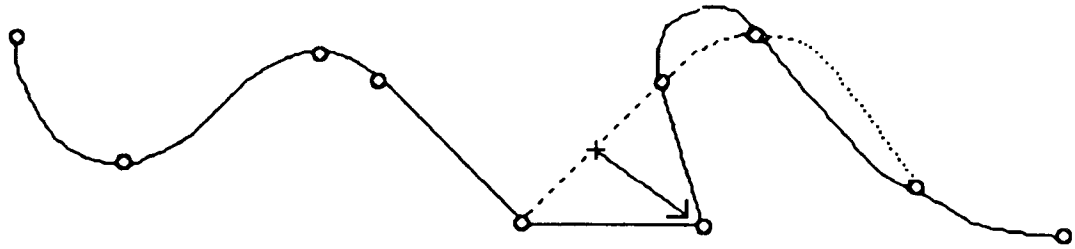
Représentation d'une trajectoire.

L'utilisateur a la possibilité de modifier une trajectoire précédemment saisie. Il lui suffit de cliquer sur un point de contrôle et de déplacer le dispositif de dialogue, resté dans un état bas. Les répercussions sur la trajectoire sont immédiates. L'affichage tient compte de tous les points de contrôle ainsi que des éventuelles contraintes de continuité.



Modification d'un point de contrôle.

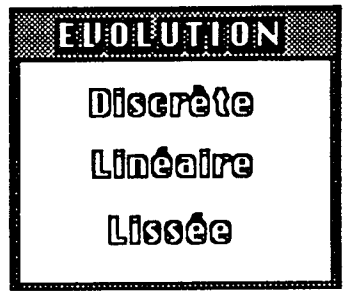
L'utilisateur a également la possibilité d'ajouter un nouveau point de contrôle à une trajectoire déjà définie.



Ajout d'un point de contrôle.

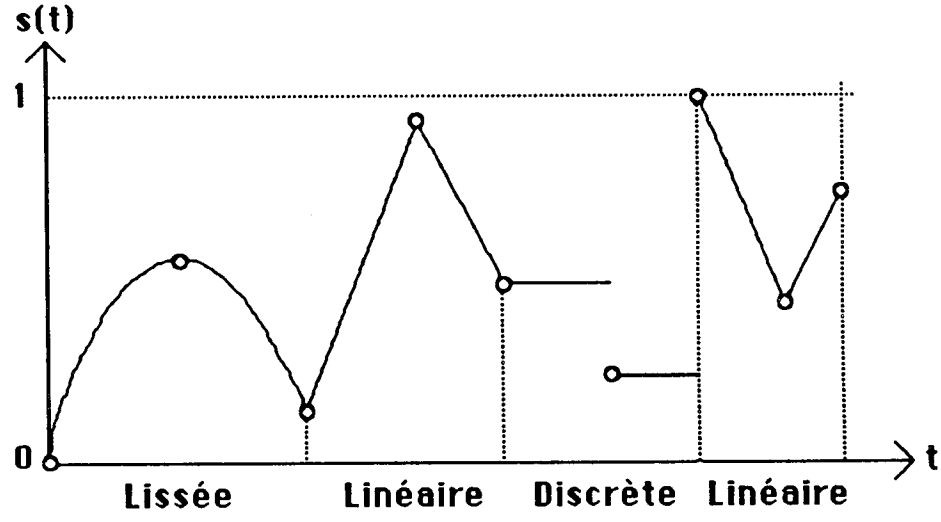
4.3.2 Fonction d'évolution

Les déplacements temporels de l'acteur le long d'une trajectoire sont régis par une fonction d'évolution. A la différence d'une trajectoire, une fonction d'évolution n'est pas forcément continue. En revanche, ce doit être une fonction, au sens mathématique du terme.



Modes de saisie de l'évolution.

L'utilisateur détermine un ensemble de points-clés, de la même manière que pour une trajectoire. Il choisit une fonction de calcul et construit interactivement la fonction. Un point ne peut cependant être validé que si son abscisse (t) est supérieure strictement à l'abscisse du point précédemment saisi. Les points-clés permettent d'établir la correspondance entre des instants précis de l'animation et des lieux précis de la trajectoires. Ils définissent des rendez-vous.



Représentation d'une fonction d'évolution.

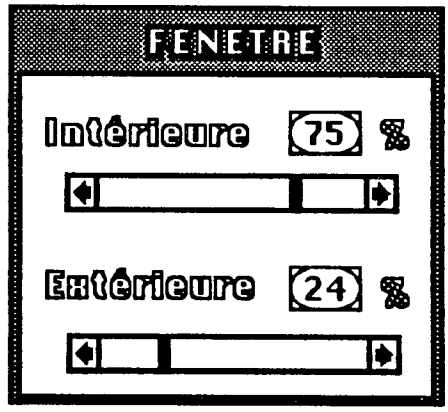
Au cours de la saisie de l'évolution, un symbole peut être déplacé le long de la trajectoire, en fonction de l'abscisse curviligne courante. Inversement, au cours de la modification d'une trajectoire, le graphe d'évolution doit être également modifié interactivement.

4.3.3 Fenêtre et pondérations

L'effet de fenêtrage permet quant à lui de ne visualiser qu'une portion du cell courant d'un acteur. Il permet également de jouer sur les coefficients de transparence associés à la fenêtre et à l'anti-fenêtre de l'acteur.

Une morphologie de type rectangle représente la zone associée à la fenêtre d'un acteur. Le dialogue de saisie et de modification a été présenté au § 3.3.3.1 du chapitre précédent.

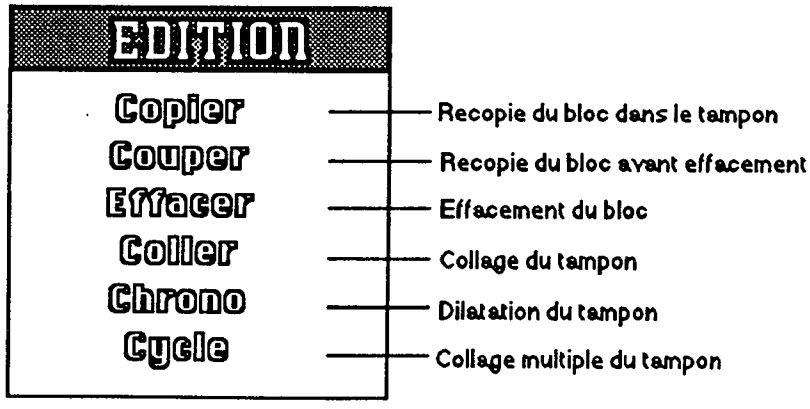
Deux ascenseurs permettent, en outre de saisir les coefficients de transparence extérieure et intérieure à la fenêtre courante. Enfin, deux valeurs jouent le rôle de témoins et permettent la saisie directe de ces coefficients.



Les objets de dialogue associés à la fenêtre.

4.4 Le mode d'édition

Le mode d'édition regroupe toutes les fonctions d'édition permettant à l'animateur de minimiser le création d'une scène ou de faciliter sa mise au point.



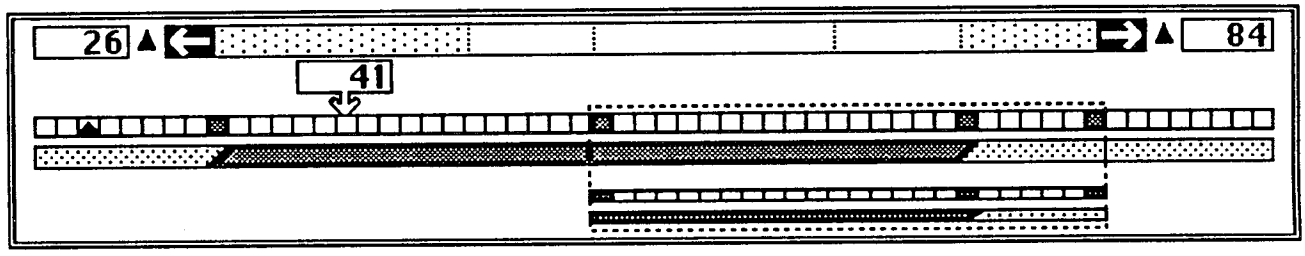
Le choix associé au mode d'édition.

Les trois premières actions proposées concernent toutes le bloc d'édition. En revanche, les trois dernières se rapportent exclusivement au tampon d'édition.

4.4.1 Le bloc d'édition

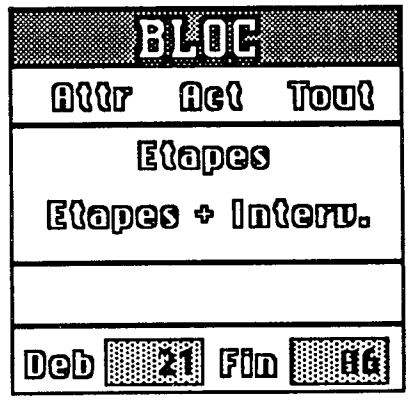
Le bloc d'édition permet la sélection d'une portion du chronogramme. Il est symbolisé par une morphologie de type rectangle, à laquelle a été associée une fonction d'écho bas (cf § 3.2.3.2 du chapitre précédent) réalisant, durant le dialogue, l'affichage de la portion de chronogramme délimitée par les côtés verticaux du rectangle. On affiche nécesairement la portion d'échelle courante. La portion relative aux intervalles de calcul est visualisée en fonction du type du bloc d'édition.

Dans le cas où le bloc dépasse les limites du chronogramme, toutes les informations temporelles déroulent en conséquence (ascenseur, compteurs, échelle, bloc).



Représentation du bloc d'édition.

La fonction d'écho bas réalise également interactivement la mise à jour des indices de début et de fin du bloc dans des valeurs, dont le domaine père est une vue, de titre "BLOC", regroupant toutes les caractéristiques du bloc d'édition.



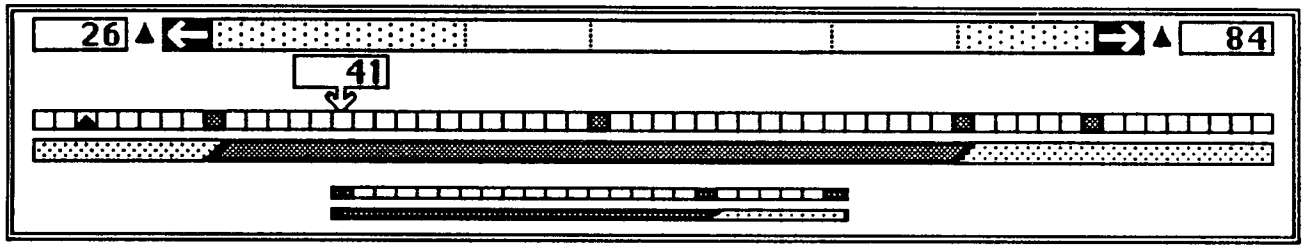
Présentation des informations relatives au bloc d'édition.

Les caractéristiques intrinsèques du bloc sont visualisées par deux choix, concernant d'une part la complexité du bloc d'édition, relatif à l'attribut courant, à

tous les attributs de l'acteur courant ou à tous les attributs de l'animation, et d'autre part le type du bloc d'édition (on ne tient compte que des valeurs-clés, ou bien on considère également les fonctions de calcul associées). Le choix d'une des deux entités de ce dernier menu conditionne l'affichage du bloc d'édition.

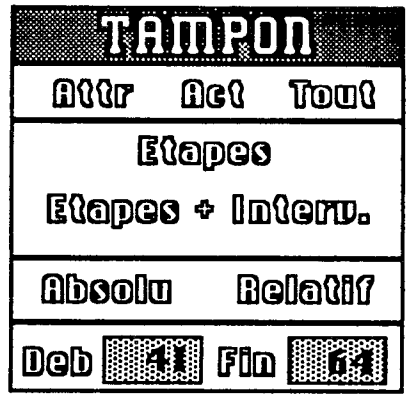
4.4.2 Le tampon d'édition

Le tampon d'édition, quant à lui, matérialise une portion de chronogramme recopiée dans des structures particulières. Il est représenté de la même manière que le bloc d'édition dont il est issu (par recopie). Aucun dialogue n'est autorisé sur le tampon, qui est toujours affiché à partir du temps courant.



Représentation du tampon d'édition.

Les informations relatives au tampon d'édition sont regroupées dans une vue spécifique de titre "TAMPON".



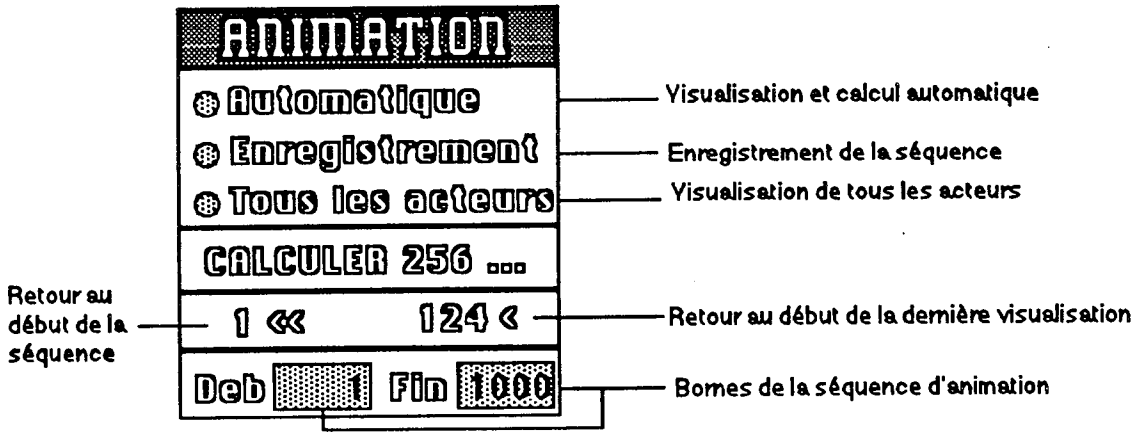
Présentation des informations relatives au tampon d'édition.

4.5 Le mode d'animation

Le mode d'animation propose les outils de dialogue nécessaires à la visualisation de la séquence décrite et mise au point dans les 3 autres modes de travail. Il regroupe notamment une vue spécifique, construite à partir de compteurs et de choix, ainsi que des accessoires pour le pilotage d'appareils vidéo.

4.5.1 La vue d'animation

La vue d'animation, de titre "ANIMATION", est constituée de la manière suivante:



Représentation de la vue d'animation.

La vue d'animation propose à l'animateur plusieurs options de visualisation:

- automatique : dans le cas où la séquence ne peut être précalculée integralement, la validation de cette option permet l'enchainement automatique du calcul et de la visualisation des portions de séquence.

- enregistrement : pour enregistrer sur support vidéo la séquence d'animation. Ce mode est totalement compatible avec le précédent.

Deux compteurs, affichés en bas de la vue d'animation, déterminent la portion de séquence à visualiser.

Sous les options de visualisation est affichée l'action principale du mode d'animation. Un clic sur la commande **CALCULER X...** permet à l'animateur de déclencher le calcul de la séquence à partir de la trame X. A l'issue du calcul, la commande **ANIM X..Y** est affichée; Y représente la dernière trame de la portion de séquence calculée. Un clic sur cette commande réalise la visualisation de cette portion de séquence. En fin de visualisation, une nouvelle commande de calcul, à partir de la trame Y+1 est proposée, si la borne supérieure de la séquence n'a pu être atteinte. Sinon, la commande ANIM X..Y reste affichée, et permet de visualiser à nouveau la même portion de séquence.

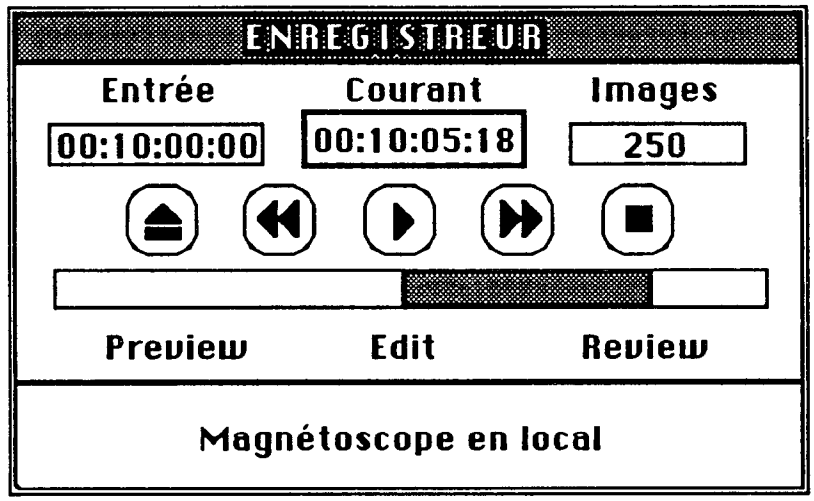
En outre, deux symboles permettent de recalculer la séquence à partir de son origine ou de revenir à la 1ère trame de la dernière visualisation.

Toutes les options et commandes sont étroitement liées. La saisie du moindre paramètre a des incidences sur tous les autres paramètres de la visualisation. Les fonctions de traitement associées à chaque domaine (choix, valeurs) doivent

consulter systématiquement les informations véhiculées par les autres domaines et éventuellement mettre à jour ces informations.

4.5.2 Le contrôle d'appareils vidéo

Le pilotage d'appareil vidéo en enregistrement s'effectue à l'aide d'un accessoire représenté ci-dessous:



Représentation d'un accessoire pour l'enregistrement vidéo.

Cet accessoire regroupe des valeurs et des choix, caractéristiques des fonctionnalités des appareils vidéo enregistreurs. La vue associée à cet accessoire, appelée console, est toujours active, autant du point de vue du dialogue que de l'affichage.

La console joue en effet le rôle de témoin de l'état de l'appareil qui lui est associé. La moindre intervention sur l'appareil provoque aussitôt la visualisation de l'état de l'appareil ou d'un message d'erreur. Cette mise à jour est effectuée même s'il n'y a pas de dialogue direct sur la console. Une fonction spéciale a éte intégrée au niveau de la primitive globale *Collecte()* (cf § 3.4.4.2 du chapitre précédent), chargée de la récupération interactive de l'état et des informations spécifiques à l'appareil vidéo. Ainsi, le temps courant peut-il être en permanence affiché dans la console, et l'appui, par exemple, de la touche de revind au niveau de l'appareil, va-t-il provoquer automatiquement la sélection de la touche correspondante sur la console.

L'animateur peut dialoguer avec la console, à partir du moment où celle-ci est affichée à l'écran. Il peut, entre autres, indiquer le point d'entrée vidéo, représentant la position initiale d'enregistrement, ou encore choisir le nombre d'images à enregistrer.

Il peut piloter à distance l'appareil vidéo en utilisant des touches analogues à celles disponibles sur l'appareil (rewind, play, forward, stop, eject). Un ascenseur fixe en son centre permet même de simuler la fonction de recherche.

Il peut enfin déclencher l'enregistrement de la séquence (Edit), simuler cet enregistrement (Preview) ou encore revoir la séquence enregistrée (Review).

Conclusion

L'organisation de l'environnement d'animation présenté dans ce chapitre a permis de déterminer les fonctionnalités propres à chaque module qui intervient dans sa composition. La répartition choisie a conduit à une programmation structurée et indépendante des différents modules après en avoir défini clairement les interfaces et fonctionnalités.

Le moteur d'animation constitue le point de raccord entre les différentes applications d'animation et autorise une indépendance complète entre ces applications et les fonctions d'animation proposées par les bibliothèques de base.

La spécification du langage d'animation L_ANIM a conduit au développement d'un environnement complet de programmation. Cet environnement gère automatiquement le temps d'animation et réalise un compromis entre les techniques classiques d'animation image par image et l'animation temps réel proposée dans le cadre de cette thèse.

Enfin la réalisation de l'application interactive de description de scènes et de visualisation de séquences en temps réel, G_ANIM, a demandé une grande réflexion au niveau de l'ergonomie. En effet, un dialogue simple et intuitif cache souvent de gros développements et le degré de convivialité est en rapport avec la complexité de programmation des fonctions de dialogue.

CONCLUSION

Le système proposé dans cette étude repose sur la constatation que toute animation, aussi complexe soit-elle, peut se ramener à la superposition d'images indépendantes, de façon analogue aux celluloïdes du dessin animé traditionnel. La réalisation de l'animation consiste alors à visualiser en temps réel toutes les images superposées, encore appelées cells, correspondant aux différentes trames de l'animation. Les avantages d'un tel système sont multiples. La réalisation d'opérateurs pour la visualisation des cells permet de minimiser le nombre de ces cells et diminue du même coup les temps de génération des images, contrairement aux systèmes classiques qui nécessitent la génération image par image de la séquence d'animation L'animation est décrite et mise au point interactivement, l'utilisateur gardant totalement le contrôle de ses actions. Enfin l'enregistrement sur bande vidéo s'effectue en direct et en temps réel, conduisant à une diminution des coûts de production et une économie du matériel vidéo.

Le principe de manipulation des images en temps réel constitue une approche originale dans le domaine de l'animation et répond aux exigences de nombreuses applications du monde de la vidéo. Le système est aussi bien adapté pour les sociétés de production de dessins animés que pour les sociétés de prestations graphiques (publicités, shows...) ou bien encore pour les télévisions pour lesquelles la qualité des images et les possibilités interactives sont des critères essentiels (résultats de la bourse, journal météo...).

Un des principaux atouts du système réside dans ses possibilités d'extension à de nouveaux produits spécifiques, à moindre frais. Ce système d'animation constitue en effet un noyau d'animation 2D en temps réel, autour duquel peuvent être menés de nombreux développements.

En particulier, l'application de montage de films d'animation, G_FILM, dont les concepts ont été énoncés dans cette étude, devra permettre de manipuler interactivement plusieurs séquences d'animation, décrites dans l'application G_ANIM, et de visualiser en temps réel le film résultant de leur montage. Ces séquences pourront être affectées à différents niveaux correspondant à la notion de plans pour le cinéma et à chacun de ces niveaux seront associés des attributs géométriques temps réel tels que position de la séquence à l'écran ou encore facteur de zoom, ainsi que des attributs d'aspect permettant de générer facilement des effets de régie tels que fondu enchainé, fondu à une couleur... De plus le réalisateur aura la possibilité de jouer sur les ralentis ou accélérés pour la visualisation de ces séquences. Cette application intègrera bien sûr le pilotage d'appareils vidéo tels que disques durs numériques, notamment pour les génération multiples de film. C'est également dans cette application que le montage sonore du film sera réalisé. L'interface avec un système audio est en effet en cours d'étude.

De plus, la réalisation d'interfaces avec des logiciels d'animation disposant de leur propre méthode de description, et plus particulièrement dans le domaine du 3D, devrait permettre de transposer automatiquement une animation complexe au niveau de l'animation de cells. Une telle interface aurait la charge d'interpréter toutes les commandes issues de la description de l'animation, afin de générer un ensemble minimal de cells et de leur appliquer les attributs 2D temps réel spécifiques au système présenté dans cette étude. La mise au point de l'animation pourrait dans certains cas s'effectuer en temps réel directement au niveau des cells. Ce style d'interface peut également avoir un grand intérêt pour la visualisation interactive de phénomènes physiques. L'expression de contraintes physiques ou de lois mathématiques pourrait ainsi être transcrite en commandes interprétables par le système d'animation et donc visualisée en temps réel. Le système présenté constituerait donc le noyau d'animation de toute une gamme d'applications d'animation.

Des développements peuvent être également menés dans le domaine de l'animation traditionnelle assistée par ordinateur afin d'optimiser toutes les étapes de production de dessins animés, depuis la création des dessins-clés jusqu'au montage final, en passant par la génération des dessins intermédiaires et le gouachage automatique. L'expérience montre, en effet, que dans ce domaine particulier, l'ordinateur doit se contenter d'informatiser les méthodes de travail traditionnelles, plutôt que de fournir ses propres méthodes.

Enfin, certaines fonctionnalités matérielles du système sont encore mal exploitées pour la réalisation d'animations. Il s'agit notamment des tables de couleurs. Le principal développement à mener concernerait la génération des images à animer, pour la création de cycles de couleurs ou encore la mémorisation des positions-clés d'un mouvement présentant ou non des intersections. La puissance de traitement de la machine proposée devrait également permettre d'exploiter les possibilités de transferts d'images pour la génération d'animation mettant en jeu par exemple un grand nombre de petits objets. Chaque image de la séquence d'animation devrait pouvoir être constituée en temps réel.

Ces futurs développements devront, eux aussi, intégrer une interface homme/machine conviviale, de telle sorte que leur utilisation ne nécessite aucune connaissance informatique. L'outil doit s'effacer devant la créativité, condition essentielle à son utilisation par l'artiste, qui peut alors considérer l'ordinateur comme un nouveau mode d'expression .

L'étude menée dans le cadre de cette thèse a conduit à la réalisation d'un système d'animation professionnel. Cet environnement d'animation a atteint le stade de commercialisation et constitue le produit pilote de la société Gétris Images. De nombreux systèmes ont été vendus au cours des deux dernières années (environ une centaine sur l'Europe). Les retours des utilisateurs sont très favorables, tant du point de vue conceptuel que du point de vue ergonomique.

ANNEXES

ANNEXE I : Génération de SPLINES

Une courbe 2D est habituellement représentée par une fonction:

$$Q(\bar{u}) = (X(\bar{u}), Y(\bar{u})),$$

où $X(\bar{u})$ et $Y(\bar{u})$ sont des fonctions du paramètre $\bar{u}$.

La méthode exposée consiste à déterminer, pour chacune de ces 2 fonctions X et Y, un polygône de degré 3 vérifiant des conditions particulières.

A chaque point de contrôle de la courbe d'indice i, correspond une valeur de $\bar{u}$, notée $\bar{u}i$, telle que $\bar{u}i <= \bar{u}i+1$.

Entre 2 points de contôle, on peut se ramener à une fonction locale de u, telle que

$$u = (\bar{u} - \bar{u}i) / (\bar{u}i+1 - \bar{u}i) \qquad 0 <= u <= 1$$

La courbe $Y(\bar{u})$ se ramène donc à déterminer pour chaque intervalle les paramètres de la fonction locale Yi(u) associée:

$$\mathbf{Yi(u) = ai + bi\ u + ci\ u^2 + di\ u^3}$$

Expression des coordonnées limites:

$$Yi(0) = yi = ai;$$
$$Yi(1) = yi+1 = ai + bi + ci + di$$

Expression de la continuité des dérivées aux limites:

$$Yi^{(1)}(0) = Di = bi$$
$$Yi^{(1)}(1) = Di+1 = bi + 2\ ci + 3\ di$$

On obtient le système suivant:

$$\mathbf{ai = yi}$$
$$\mathbf{bi = Di}$$
$$\mathbf{ci = 3\ (yi+1 - yi) - 2\ Di - Di+1}$$
$$\mathbf{di = 2\ (yi - yi+1) + Di + Di+1}$$

Conditions aux limites de la courbe:

$Y0\ (0) = y0$
$Ym\text{-}1\ (1) = ym$

Si l'on exprime la continuité de la dérivée seconde en chaque point de contrôle:

$Yi\text{-}1^{(2)}\ (1) = Yi^{(2)}(0)$

nous obtenons

$$\mathbf{2\ ci\text{-}1 + 6\ di\text{-}1 = 2\ ci}$$

En substituant dans le système précédent, nous obtenons

$$2\ [\ 3\ (yi - yi\text{-}1) - 2\ Di\text{-}1 - Di] + 6\ [\ 2\ (yi\text{-}1 - yi) + Di\text{-}1 + Di] = 2\ [\ 3\ (yi+1 - yi) - 2\ Di - Di\text{-}1].$$

soit

$$\mathbf{Di\text{-}1 + 4\ Di + Di+1 = 3\ (yi+1 - yi\text{-}1)}$$

Les conditions aux limites nous donnent:

$2\ c0 = 0$
$2\ [3\ (y1 - y0) - 2\ D0 - D1] = 0$ soit $2\ D0 + D1 = 3\ (y1 - y0)$

et $Dm\text{-}1 + 2\ Dm = 3\ (ym - ym\text{-}1)$.

On obtient donc le système diagonal suivant:

$$\begin{vmatrix} 2 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & 4 & 1 & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 2 \end{vmatrix} \begin{vmatrix} D0 \\ D1 \\ D2 \\ \\ Dm\text{-}1 \\ Dm \end{vmatrix} = 3 \begin{vmatrix} y1 - y0 \\ y2 - y0 \\ y3 - y1 \\ \\ ym - ym\text{-}2 \\ ym - ym\text{-}1 \end{vmatrix}$$

On peut se ramener à une matrice triangulaire par les changements de variables suivants:

```
ß0 <- 1/2;
for i <- 1 step 1 until m-1 do
        ßi <- 1/ (4 - ßi-1);
endfor;
ßm <- 1/(2 - ßm-1).

∂0 <- 3 (y1 - y0) ß0;
for i <- 1 step 1 until m-1 do
        ∂i <- (3 (yi+1 - yi-1) - ∂i-1) ßi
endfor
∂m <- (3 (ym - ym-1) - ∂m-1) ßm.
```

On se ramène ainsi au système suivant:

$$\begin{vmatrix} 1 & ß0 & & & \\ & 1 & ß1 & & \\ & & & & \\ & & & 1 & ßm-1 \\ & & & & 1 \end{vmatrix} \begin{vmatrix} D0 \\ D1 \\ \\ Dm-1 \\ Dm \end{vmatrix} = \begin{vmatrix} ∂0 \\ ∂1 \\ \\ ∂m-1 \\ ∂m \end{vmatrix}$$

qui se résoud très facilement de la manière suivante:

```
Dm  <- ∂m
for i <- m-1 step -1 until 1 do
        Di <- ∂i - ßi Di+1
endfor
```

En remplaçant dans le système initial, on obtient les coefficients de tous les polygônes caractéristiques Yi(u) de degré 3 reliant 2 à 2 tous les points de contrôle.

Les mêmes calculs doivent être menés pour déterminer les coefficients des polygônes caratéristiques Xi(u).

Le calcul sur chaque intervalle d'un ensemble de valeurs des polygônes Xi(u) et Yi(u), pour des valeurs de u comprises entre 0 et 1, permet ainsi de discrétiser chaque intervalle en un ensemble de points donnés et donc d'approximer plus ou moins finement la spline par une suite de points 2D.

ANNEXE II : Génération de droites

Génération d'une droite reliant les points (x0, y0) et (xn, yn) dans le 1er octant de l'espace ((xn - x0) >= (yn - y0) >= 0).

L'équation de la droite s'écrit : $y = \partial y / \partial x * x$ ou encore $y \partial x - x \partial y = 0$.
Cette expression,qui doit être nulle lorsqu'un point est sur la droite, représente la fonction d'erreur à minimiser (en valeur absolue) lors de la génération. Soit donc:

$$E = y \partial x - x \partial y$$

Deux mouvements élémentaires suffisent à la génération de la droite :

Mouvement a

x <- x + 1, qui a pour effet de diminuer l'erreur de ∂y
E <- E - ∂y

Mouvement b

x <- x + 1
y <- y + 1, qui a pour effet d'augmenter E de $\partial x - \partial y$
E <- E + ∂x - ∂y

On peut proposer un algorithme itératif à partir de ces 2 mouvements

```
E <- 0;
y <- y0;            x <- x0;
∂x <- xn - x0;      ∂y <- yn - y0;
∂a <- -∂y;          ∂b <- ∂x - ∂y;
Afficher_point (x0, y0);
Faire tant que x < xn
        x <- x + 1;
        si E < 0 alors              (mouvement b)
                y <- y + 1;
                E <- E + ∂b;
        sinon                       (mouvement a)
                E <- E + ∂a;
        finsi
        Afficher_point(x, y);
FinFaire;
```

Génération de droite à l'ordre n

Dans ce cas, on ne cherche pas à générer une droite continue, mais on veut obtenir pour tout x une valeur y appartenant à la droite passant par les 2 points quelconques (x0, y0) et (xn, yn). On suppose seulement que xn > x0, sinon on renverse la droite.

L'ordre correspond à la pente de la droite n = [∂y / ∂x] (valeur entière)

L'erreur est toujours représentée par l'équation E = y ∂x - x ∂y

On a 2 mouvements possibles :

Mouvement a

x <- x + 1
y <- y + n, qui a pour effet d'augmenter l'erreur de ∂a = n ∂x - ∂y

Mouvement b

x <- x + 1
y <- y + n + 1, qui a pour effet d'augmenter l'erreur de ∂b = (n + 1) ∂x - ∂y

On peut proposer un algorithme itératif pour le calcul de toutes les ordonnées de la droite.

```
E <- 0;
y <- y0;            x <- x0;
∂x <- xn - x0;      ∂y <- yn - y0;
∂a <- n ∂x -∂y;           ∂b <- ∂a + ∂x;
Afficher_point (x0, y0);
Faire tant que x < xn
        x <- x + 1;
        si E < 0 alors              (mouvement b)
                y <- y + n + 1;
                E <- E + ∂b;
        sinon                       (mouvement a)
                y <- y + n;
                E <- E + ∂a;
        finsi
        Afficher_point(x, y);
FinFaire;
```

ANNEXE III : Syntaxe BNF du langage L_ANIM.

Les symboles du méta-langage figurent en ombré.

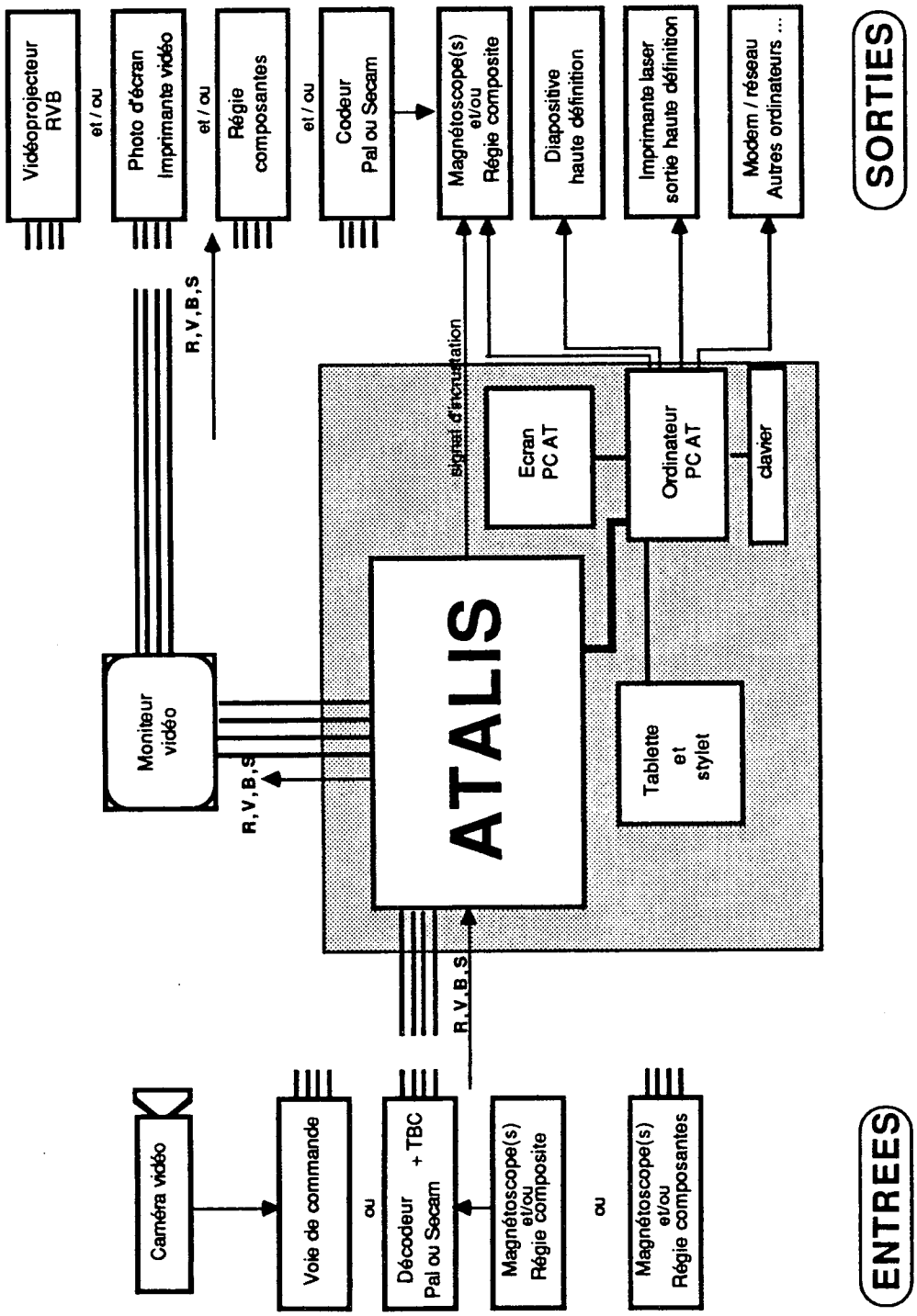
<program> ::= { { <decl_conf> } { <decl_zone> } { <decl_file> }
{ <decl_var> } { <decl_prog> } } <corps_prog>
<decl_conf> ::= CONFIG <idf_image><const_decl>
<decl_zone> ::= ZONE<idf><idf_image><entier+><entier+><entier+><entier+>
<decl_file> ::= FICHIER <entier+> <idf>
<decl_var> ::= VARIABLE <idf> <bloc_var>
<bloc_var> ::={<entier>||<entier>..<entier>||<fonc_def><entier>{<entier>} }
<decl_proc> ::= PROCEDURE <idf_proc> (<liste_param_zones>) <bloc>
<corps_prog> ::= PROGRAMME <bloc>
<bloc> ::= { <intervalle> { <intervalle> } }
<intervalle> ::= <prédicat> | <instruction>; { <instruction> ; }
<prédicat> ::= <entier+> .. <entier+>
<instruction> ::= <appel_fonc> || <expression> || <appel_proc>
<appel_fonc> ::= <idf_fonction> [[[<liste_param_entiers>]]]
[[(<liste_param_zones>)]]
<expression> ::= <idf_zone> [[[<liste_param_entier>]]] <= <opérateur>
[[[<liste_param_entiers>]]] [[(<liste_param_zones>)]]
<appel_proc> ::= <idf_proc> [[<temps>]]
[[[<liste_param_entiers>]]] [[(<liste_param_zones>)]]

<liste_param_zones> ::= <idf_zone> { , <idf_zone> }
<liste_param_entiers> ::= <entier> { , <entier> }
<param_entier> ::= <entier> || <entier> .. <entier> || <idf_define>

<fonc_def> ::= CONST || LINEAIRE || BRESHM || UNIFORME
<idf_image> ::= IMAGE0 || IMAGE1 || IMAGE2 || IMAGE3 || IMAGE4
<const_decl> ::= COUL12 || COUL24 || VIDEO
<fonction> ::= EXTERN||SEQUENCE||PROFOND||FENETRE||ORIGINE||EFFACE
||STOREH|| STOREV||HACHUH||HACHUV||PAUSE
<opérateur> ::=IMA||TGA||DECALQUE||TRANSFERT||ZOOMA||ZOOME||FILTRE

ANNEXE IV : Synoptiques ATALIS

Environnement du système ATALIS



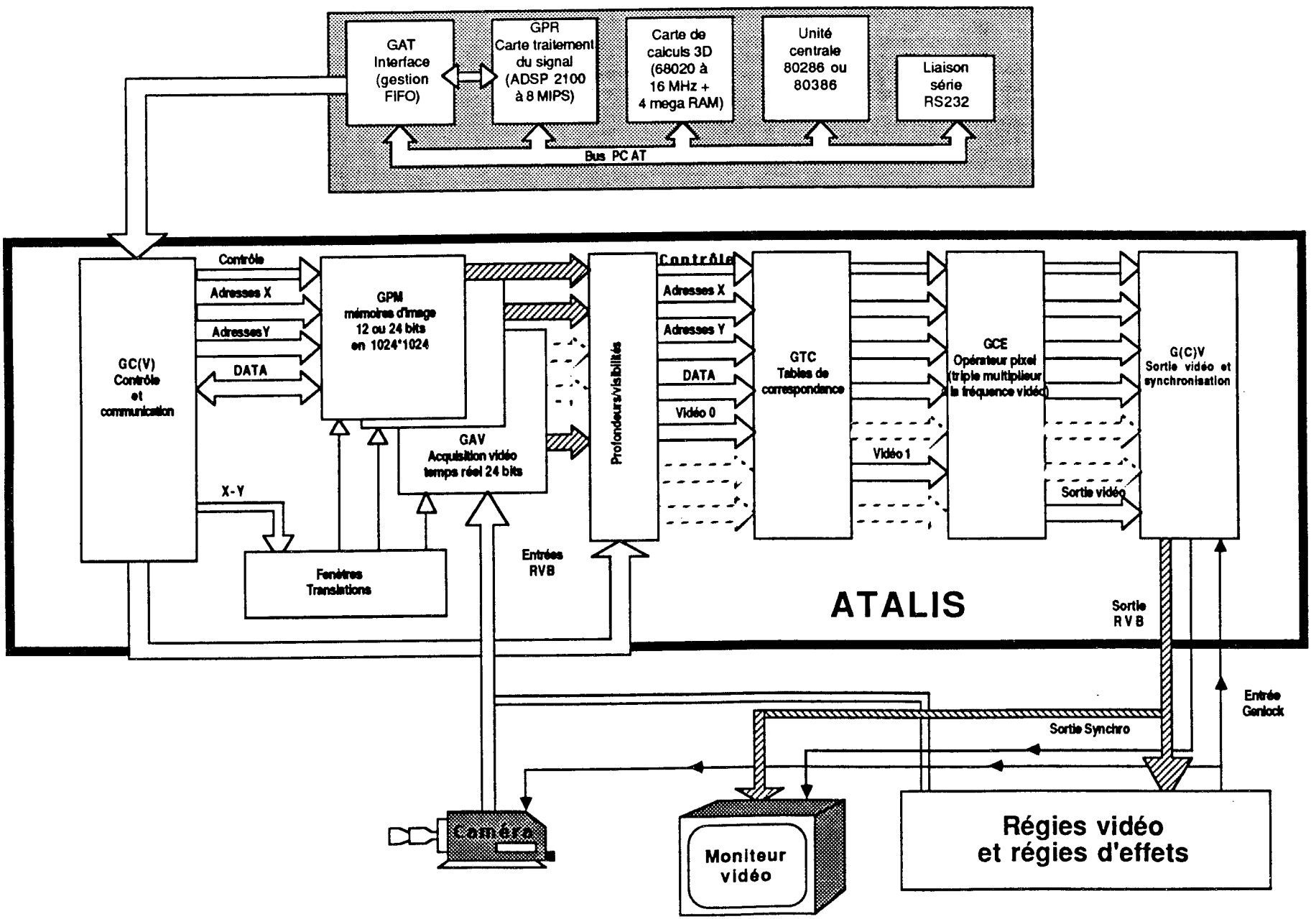


Schéma technique du système ATALIS

BIBLIOGRAPHIE

[Ama 87a] Amanatides J.
A fast Vixel Algorithm for Ray Tracing.
Proc. Computer Graphics, Londres, Vol.10, octobre 87.

[Ama 87b] Amanatides J.
Realism in Computer Graphics.
Proc. Computer Graphics, Londres, Vol.10, octobre 87.

[Bar 84] Barr A.H.
Global and Local Deformations of Solid Primitives.
Proc. Siggraph'84, Computer Graphics, Vol.18, N°3, juillet 84.

[BBe 83] Barsky B.A., Beatty J.C.
Local Control of Bias and Tension in Beta-Splines.
Proc. Siggraph'83, Computer Graphics, Vol.18, N°3, juillet 83.

[Bou 80] Boulle P.
Etude et réalisation d'algorithmes pour la visualisation de scènes composées de facettes planes.
Thèse de Docteur-Ingénieur, INP Grenoble, septembre 80.

[Bur 76] Burtnik N., Wein M.
Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation.
Proc. ACM Vol. 19, N°10, octobre 1976.

[Bre 65] Bresenham J.E.
Algorithm for Computer Control of a Digital Plotter
IBM System Journal, Vol.4, N°1, 1965.

[Bre 82] Bresenham J.E.
Incremental Line Compaction.
The Computer Journal, Vol.25, N°1, 1982.

[BWM 87] Bowyer A., Wallis A., Milne P.
Symbolic Ray Tracing
Proc. Computer Graphics, Londres, Vol.10, octobre 87.

[Cur 87] Curington I.J.
Radiosity methods for Light Diffusion.
Proc. Computer Graphics, Londres, Vol.10, octobre 87.

[Den 82] Denber M.J
A Differential Compiler for Computer Animation.
Proc. Siggraph'86, Computer Graphics, Vol.20, N°4, août 86.

[Edw 87] Edwards G.J
SCRIPT:an interactive animation environment.
Computer Graphics, Londres, Vol.10, octobre 87.

[FLT 86] Fortin D., Lamy J.F., Thalmann D.
A Multiple Track Animator System for Motion Synchronisation.
Proc. Siggraph/Sigart, Workshop on Motion, 86.

[Fou 86] Fournier A.
A Simple Model of Ocean Waves
Proc. Siggraph'86, Computer Graphics, Vol.20, N°4, août 86.

[Gen 89] Genoud P.
Contribution à la définition d'un logiciel graphique pour la visualisation et le dialogue structuré.
Thèse 3ème cycle, Université Joseph Fourier, Grenoble 1, janvier 89.

[Gla 87] Glazzard N.
Particle Systems : Methods, Implementation techniques and Experiences.
Proc. Computer Graphics, Londres, Vol.10, octobre 87.

[GMa 85] Girard M., Maciejewski A.A.
Computational Modeling for the Computer Animation of Legged Figures.
Proc. Siggraph'85, Computer Graphics, Vol.19, N°3, juillet 85.

[HCF 83] Hiroaki Chiyokura, Fumihiko Kimura,
Design of Solids with Free-Form Surfaces.
Proc. Siggraph'83, Computer Graphics, Vol.17, N°3, juillet 83.

[KBa 83] Korein J., Badler N.
Temporal Anti-Aliasing in Computer Generated Animation
Proc. Siggraph'83, Computer Graphics, Vol.17, N°3,juillet 83.

[Kno 64] Knowlton K.C
A computer technique for producing animated movies.
Proc. SJCC AFIPS Conference, Vol.25.

[Kor 83] Korein J. , Badler N.
Temporal AntiAliasing in Computer Generated Animation.
Proc. Siggraph'83, Computer Graphics, Vol.17, N°3.

[Kun 89] Kuntz G.
Vers une Intégration des Systèmes Graphiques et Audiovisuels.
Thèse 3ème cycle, Université Joseph Fourier, Grenoble I, février 89.

[Mal 87] Mallinder H.
Modelling Ocean Surfaces
Proc. Computer Graphics, Londres, Vol.10, octobre 87.

[Mar 77] Martinez F.
Etude des problèmes de conception et de réalisation d'animation : le système SAFRAN.
Thèse de Docteur 3ème cycle, INP Grenoble, Mai 77.

[Mar 82] Martinez F.
Vers une approche systématique de la synthèse d'image. Aspects logiciel et matériel.
Thèse de Docteur d'Etat, INP Grenoble, Novembre 82.

[Moi 84] Moissinac J.C.
Aides informatiques à la réalisation de dessins animés.
Thèse de Docteur-Ingénieur, ENSM Saint-Etienne, décembre 1984

[MTh 83] Magnenat Thalmann N., Thalmann D.
The Use of High-Level 3D Graphical Types in the MIRA Animation System.
Proc. IEEE, Computer Graphics and Applications, Vol.3, N°19, Décembre 83.

[MTh 85a] Magnenat Thalmann N., Thalmann D.
Computer Animation, Theory and Practice.
Springer-Verlag Tokyo, Editeur Tosiyasu L. Kunii, 85.

[MTh 85b] Magnenat Thalmann N., Thalmann D.
Miranim : An Extensible Director-Oriented System for the Animation of Realistic Images.
Proc. IEEE, Computer Graphics and Applications, Mars 85.

[MTh 88] Magnenat Thalmann N., Thalmann D.
Construction and Animation of a Synthetic Actress.
Congrès PIXIM, Paris, septembre 88.

[Pea 86] Peachey D.
Modeling Waves and Surf.
Proc. Siggraph'86, Computer Graphics, Vol. 20, N°4, août 86.

[Pla 81] Platt S., Badler N.
Animating Facial Expressions.
Proc. ACM, Computer Graphics, Vol.15, N°3, août 81.

[Per 88] Peroche B., Argence J., Ghazanfarpour D., Michelucci D.
La synthèse d'images.
Traité des nouvelles technologies. Assistance par ordinateur. Editions Hermès.

[Ree 81] Reeves W.T
Inbetweenig for Computer Animation Utilizing Moving Point Constraints.
Proc. Siggraph' 81, Computer Graphics, Vol. 15, N°3.

[Req 80] Requicha A.
Representations for Rigid Solids : Theory, Methods and Systems.
Computing Surveys, Vol. 12, N°4, décembre 80.

[Rey 82] Reynolds C.W.
Computer Animation with Scripts and Actors.
Proc. Siggraph'82, Computer Graphics, Vol.16, N°3, juillet 82.

[Rey 87] Reynolds C.W.
Time
Proc. IEEE, Computer Graphics, Courses notes, Vol 7, 1987.

[Rey 87] Reynolds C.W.
Flocks, Herds ans Schools : a Distributed Behavioural Model.
Proc. Siggraph'87, Computer Graphics, Vol.21, N°4, juillet 87.

[Sho 79] Shoup R.
Color Table Animation.
Proc. Siggraph'79, Computer Graphics, Vol.13, N°3.

[Ste 79] Stern G.
SoftCel : An application of Raster Scan Graphics to Conventional Cel Animation.
Proc. Siggraph'79, Computer Graphics, Vol.13, N°3.

[Wat 87] Waters K.
A Muscle Model for Animating Three-Dimensionnal Facial Expression.
Proc. Siggraph'87, Computer Graphics, Vol. 21, N° 4, juillet 87.

[Zar 85] Zarate Silva V.H
Etude et réalisation d'un synthétiseur d'images basé sur une architecture banalisée.
Thèse de docteur ingénieur, INPG, Novembre 85.

[Zel 82] Zelter D.
Motor control techniques for figure animation.
Proc IEEE, Computer Graphics and Application, Vol.2, N°9, 82.

[Zel 86] Zelter D.
Knowledge-based Animation.
Proc. Siggraph/Sigart, Workshop on Motion, 86.

GLOSSAIRE

antialiasing Méthode logicielle qui consiste à augmenter la résolution de l'écran afin de faire disparaître les effets de marche d'escalier.

canal alpha Information propre à la transparence d'un pixel d'une image.

cell Provient du mot celluloïde, feuille d'acétate sur laquelle est peint un décor ou un personnage d'une animation.

driver Module de gestion d'un matériel graphique ou vidéo.

flip-book Méthode qui consiste à empiler tous les dessins d'une animation et à les faire défiler manuellement afin de percevoir le rythme global de cette animation.

forward Déroulement d'une bande vidéo.

line-test Méthode qui consiste à visualiser une animation en dessins au trait.

play Défilement en lecture d'une source vidéo.

play-back Méthode d'animation différée, qui consiste à précalculer toutes les images d'une animation, puis à visualiser successivement toutes ces images.

point de montage Point d'arrêt puis de reprise d'un enregistrement.

preroll Manipulation qui consiste à laisser défiler un magnétoscope en lecture pendant une durée déterminée avant de commuter en enregistrement, afin de le synchroniser sur les signaux vidéo.

rewind Enroulement d'une bande vidéo.

spline Méthode mathématique d'interpolation à partir de polygônes caractéristiques.

storyboard Description préparatoire du film d'animation. Il indique les éléments nécessaires à une vue d'ensemble du film.

temps codé Temps inscrit sur une bande vidéo pour chaque image..

AUTORISATION DE SOUTENANCE

DOCTORAT 3ème CYCLE, DOCTORAT INGENIEUR,
DOCTORAT DE L'UNIVERSITE JOSEPH FOURIER - GRENOBLE 1

Vu les dispositions de l'Arrêté du 16 avril 1974,

Vu les dispositions de l'Arrêté du 5 juillet 1984,

Vu les rapports de Monsieur ... MERIAUX Michel .....

Monsieur ... MARTINEZ Francis .....

Monsieur ... BRUNEAUX Sylvère ..... est autorisé(e)
à présenter une thèse en vue de l'obtention du titre de .....
..... Docteur - Ingénieur ..... "Informatique" .....

Grenoble, le 18 OCT. 1989

Le Président de l'Université
Joseph Fourier - Grenoble 1

UNIVERSITE JOSEPH FOURIER GRENOBLE

A. NEMOZ