Évolution et modularité dans une application interactive d'animation temps réel

Christophe Puget

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL ASSOCIE DE GRENOBLE

MEMOIRE

présenté en vue d'obtenir

LE DIPLOME D'INGENIEUR CNAM

en INFORMATIQUE

par Christophe PUGET

Evolutivité et modularité dans une application interactive d'animation temps réel

Les travaux relatifs au présent mémoire ont été effectués à Getris Images sous la direction de Pascale Chalvin.

REMERCIEMENTS

Je tiens à remercier :

- Mme V. Donzeau-Gouge, Professeur au CNAM de Paris, pour avoir accepté de présider le jury de soutenance;

- Mr C. Kaiser, Professeur au CNAM de Paris, et Mr J. Courtin, Professeur à l'université P. Mendes France, responsable du cycle ingénieur en informatique de Grenoble, pour avoir accepté d'être les membres du jury;

- F. Martinez et P. Chalvin pour la confiance qu'ils m'ont montré en m'intégrant dans leur équipe pour la réalisation d'une nouvelle application;

- Ma femme Laurence pour le soutien moral qu'elle m'a apporté tout au long de ce mémoire;

- Sans oublier tous les membres de l'équipe logicielle R&D de Getris Images pour l'aide qu'ils m'ont apporté et le temps qu'ils m'ont consacré.

TABLE DES MATIERES

INTRODUCTION

L'infographie intervient aussi bien dans les arts graphiques que dans l'audiovisuel. La création d'un journal, la réalisation d'un générique de télévision, l'utilisation d'images de synthèse dans un message publicitaire, les trucages numériques pour le cinéma sont autant d'exemples d'applications.

Non seulement l'ordinateur graphique disposant d'une forte résolution est en mesure d'assister l'animateur dans le domaine des techniques traditionnelles (réalisation d'images intermédiaires et « colorisation » par exemple), mais il permet de créer des images totalement artificielles dégagées de l'empreinte de la nature ou de l'emprise de la main humaine [PER_88].

Les technologies du traitement de l'information permettent de créer et de diffuser des images, qu'elles soient fixes ou animées, qu'elles soient captées dans le monde réel ou issues de l'imagination humaine. L'outil informatique, en tant que système particulièrement performant, va gérer avec compétence les relations entre le créateur et l'idée qu'il se fait de sa création, en l'occurrence, une image concrète ou une séquence d'images.

Lorsque l'utilisateur désigne à l'écran une commande ou une opération à effectuer sur l'image; lorsqu'il fait défiler une séquence d'images pour une animation, on ne sera pas étonné de la simplicité des manipulations. En effet, tout est pris en compte par l'ordinateur de la manière la plus efficace et la moins contraignante possible.

Les différents logiciels d'animation conçus chez Getris Images répondent aux besoins des graphistes en leur proposant une panoplie d'effets d'animation en constante évolution tout en conservant une interface de dialogue structurée et souple d'utilisation.

En animation assistée, lorsque les calculs sont suffisamment rapides pour rafraîchir l'image au rythme de la vidéo, on parle alors de traitement de l'image en temps réel. L'Anim [BRU_89] dans un premier temps a permis l'animation en temps réel d'une série d'acteurs en réalisant des effets simples mais efficaces. Le Sequencer [CHA_92] a complété la gamme de ces effets en proposant des transformations d'acteurs dans un espace 3D comme les rotations, la mise en perspective, au détriment de la rapidité d'exécution des calculs de l'animation. En effet, ces transformations calculées par logiciel ne peuvent pas être réalisées en temps réel.

Il n'y avait donc plus qu'un pas à franchir pour obtenir de telles transformations en temps réel : trouver un mécanisme permettant de traiter le volume des calculs à effectuer. Ce pas a été franchi par la substitution des calculs logiciels par des opérateurs câblés : les nouvelles cartes DVE-Layer proposent outre les transformations d'acteurs 2D dans un espace 3D en temps réel, des calculs au sous-pixel pour une qualité d'image exceptionnelle, des effets d'animation tels les métamorphoses (morphing), les effets de drapeau, de vague, etc.

L'intégration de ces nouveaux modules et des opérations qu'ils effectuent a été réalisée dans une nouvelle application d'animation : le DVE-Composer. Cette application tente de regrouper l'ensemble des commandes de ses deux prédécesseurs. La conception de ce logiciel a été l'occasion de proposer de nouveaux effets vidéo et une nouvelle approche de l'animation en particulier au niveau de la description du scénario.

Le premier chapitre présente les concepts de l'animation qu'elle soit traditionnelle ou assistée par ordinateur. Le logiciel Anim développé par la société Getris Images est un exemple de la « symbiose » entre l'animation et l'ordinateur. La société Getris Images, cadre de cette étude, a

développé en parallèle des systèmes électroniques capables de gérer les volumes de données des images de synthèse vidéo et des systèmes informatiques pour piloter les premiers.

La politique commerciale de Getris Images consiste à sortir un nouveau produit sur le marché régulièrement. Le dernier né de la gamme a conduit à l'élaboration d'un nouveau logiciel d'animation dont le cahier des charges est présenté dans le second chapitre. Parmi les nouveaux aspects de ce logiciel, on trouve en particulier, l'apparition des nouveaux modules DVE-Layer, des effets d'animation sophistiqués et l'animation 2D dans un espace 3D en temps réel.

L'essentiel de l'étude réalisée porte dans le troisième chapitre sur de nouveaux modes de description et de gestion des attributs avec l'apparition entre autres de l'asservissement des attributs.

L'interface homme-machine propose un système interactif basé essentiellement sur un dialogue graphique. Cette interface présentée dans le quatrième chapitre est adaptée aux méthodes de travail des utilisateurs et propose une organisation souple des outils de travail.

Enfin le dernier chapitre propose les choix effectués en terme de réalisation. Il présente les différents modules étudiés et montre leur intégration dans l'application DVE-Composer.

C'est en conclusion que le lecteur pourra apprécier les capacités de cette nouvelle application grâce à une comparaison avec le Sequencer à partir d'un exemple simple.

CHAPITRE 1

L'ANIMATION

1. HISTORIQUE.

Au 19ème siècle, Newton avait expliqué que les images restaient fixées une fraction de seconde dans l'oeil humain avant d'être remplacées par les suivantes. Nous avons alors l'impression d'un mouvement, parce que dans notre esprit, l'image qui disparaît s'ajoute à celle qui lui succède.

Le phénakistiscope, inventé en 1833 était un cercle de carton, fixé sur un pivot et qui portait sur le bord des images en phases successives. En fixant une seule de ces images et en faisant tourner le cercle sur son axe, on avait l'impression que le dessin bougeait.

Des perfectionnements, des applications et des recherches analogues ont donné naissance à des instruments aux noms bizarres : zootrope, kénétoscope, phasmatrope, praxinoscope. Ces gadgets rudimentaires créent l'illusion du mouvement. Edison en Amérique et Lumière en Europe donnèrent naissance au véritable cinéma [BEN_85].

2. DEFINITION DE L'ANIMATION.

L'animation consiste, par la prise de vue image par image de dessins ou de scènes différentes, puis par leur projection à un certain rythme d'images par seconde, à donner l'impression de mouvement.

L'animation n'est pas l'enregistrement d'un mouvement réel donné par échantillonnage de photogrammes instantanés de ce mouvement. C'est exactement l'inverse : création d'illusion d'un mouvement imaginaire qui n'existe pas encore par la création de photogrammes élémentaires.

Le dessin animé a précédé le cinéma et les premières bandes sur papier dont le rythme de projection dépendant essentiellement de l'opérateur devait être de 8 images par seconde environ, ce qui suffisait déjà pour donner l'illusion de mouvement.

Les premiers films cinématographiques de « vue réelle » avaient une fréquence d'échantillonnage et de projection de 18 images par seconde. Pour des raisons diverses, le cinématographe a adopté le standard de 24 images par seconde et le dessin animé, lui aussi, depuis Disney, a repris ce standard.

L'animation est réalisée à l'aide d'un système très encombrant et très coûteux : La Multiplane.

Cet instrument conçu au début du 20ème siècle dans les studios Walt Disney est encore utilisé de nos jours. Les dessins peints sur des feuilles d'acétate sont placés sur des plaques de cristal à des niveaux différents. La caméra est placée au sommet de l'ensemble. Le déplacement vertical des plaques permet de créer un effet de zoom (travelling avant et arrière). Le déplacement horizontal des plaques permet le déroulement du décor et la translation des personnages.

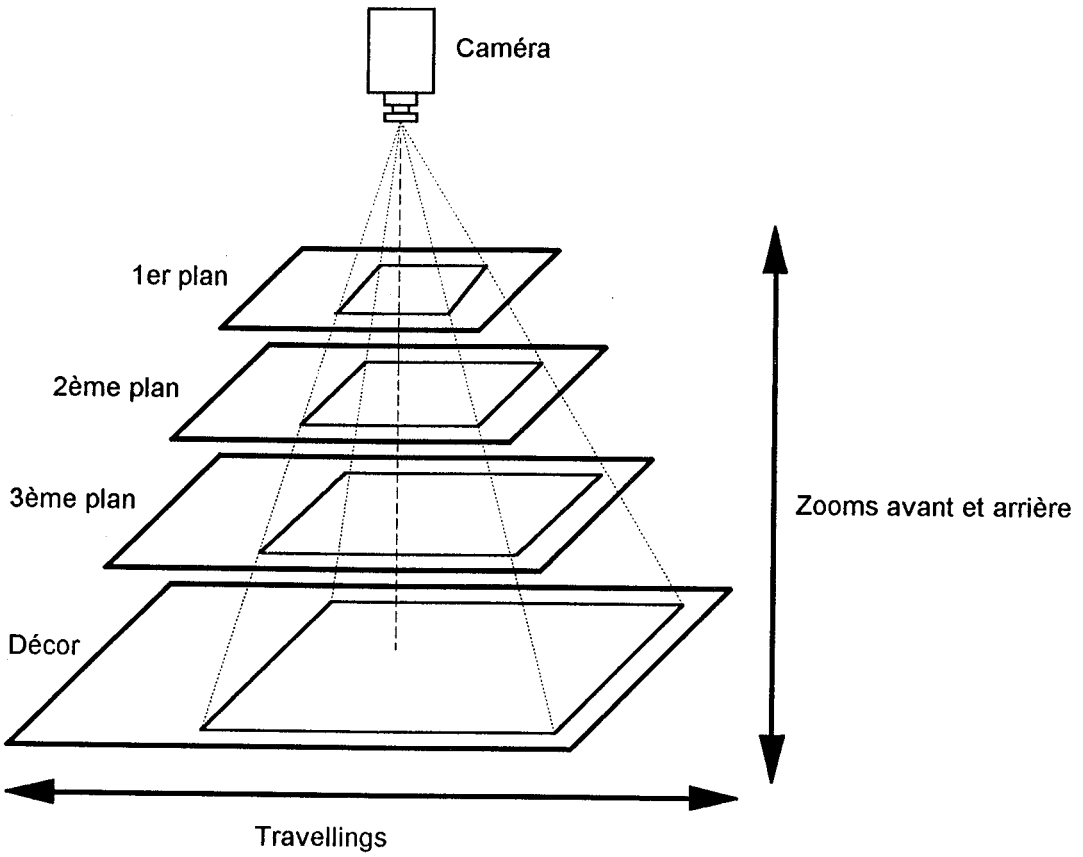


Schéma de la multiplane

Il existe 2 approches essentielles à l'art de l'animation :

- recréer le mouvement naturel pour compléter le réalisme des personnages et des objets par une animation elle aussi réaliste;
- imaginer des mouvements par des accélérations, des déformations et des effets.

Schématiquement, le travail d'animation comporte 4 fonctions soit réalisateur-dessinateur, animateur, intervalliste et opérateur de prise de vue :

Le réalisateur-dessinateur conçoit le scénario, crée les dessins et dirige les opérations de montage. Les personnages animés sont le plus souvent peints sur des feuilles de triacétate transparentes appelées « cellulos ».

L'animateur donne vie aux dessins en déterminant les différentes attitudes ou positions de ceux-ci à chaque image du film. Partant des dessins de mise en place de l'animation, son rôle est de concrétiser le mouvement en fournissant autant de dessins-clé que nécessaire. Il précise le nombre de dessins intermédiaires à générer ainsi que la cadence, c'est à dire leurs positions relatives.

L'intervalliste, partant des dessins clés et en fonction de la cadence définie par l'animateur, fournit les intermédiaires nécessaires pour compléter à 24 dessins par seconde.

L'opérateur de prise de vue fixe l'ensemble de l'animation sur la pellicule. Il utilise pour cela un banc-titre sur lequel est montée une caméra. La caméra est fixée sur une colonne ce qui permet de la rapprocher ou de l'éloigner de la table de travail. Ce mouvement (travelling) peut être horizontal ou vertical [PER_85].

Les mouvements du banc-titre sont calculés :

- par l'animateur, lorsque le déplacement dépend de l'animation;
- par l'opérateur à partir des indications concernant les images de départ et d'arrivée ou par des règles simples d'accélération et décélération linéaires;
- par asservissement informatique; on peut réaliser des mouvements extrêmement complexes, et surtout capables d'être exactement répétés lors de passages successifs.

3. L'ANIMATION ASSISTEE PAR ORDINATEUR.

L'ordinateur disposant d'une forte résolution graphique est en mesure d'assister l'animateur dans le domaine des techniques traditionnelles : réalisation des images intermédiaires et colorisation. L'opérateur définit le décor, les objets ou les sujets devant y évoluer, leur trajectoire, leurs couleurs. Il situe les sources de lumière par rapport au décor et prévoit les déplacements perspectifs. L'ordinateur se charge alors de tous les calculs. Le résultat obtenu est des images dont le réalisme est comparable à la photographie moyennant de longues heures de calculs sur des ordinateurs puissants et coûteux [PIN_91].

L'ordinateur intervient à partir de la phase de création des dessins intermédiaires. Les dessins clés sont récupérés par l'ordinateur à l'aide d'un système d'acquisition vidéo (caméra, scanner, etc.).

La pré-visualisation est un atout essentiel pour l'animateur. L'ordinateur calcule les dessins intermédiaires à l'aide de méthodes mathématiques d'interpolation [BUR_76]. L'animateur peut ainsi juger des mouvements de chacun des personnages et vérifier la synchronisation de son animation.

Le gouachage automatique supprime le problème d'altération des couleurs engendré par la superposition des cellulos. Il permet des effets de dégradés qui sont impossible à réaliser en animation traditionnelle.

L'ordinateur gère les images de l'animation en superposant les dessins et en calculant automatiquement le résultat. Les images sont enregistrées les unes après les autres sur un support vidéo. L'apparition de nouvelles techniques vidéo couplées aux systèmes de synthèse d'images; tels que les vidéodisques, les disques durs numériques; améliore les performances des systèmes d'animation assistée par ordinateur.

4. EXEMPLE DE LOGICIEL D'ANIMATION : L'ANIM.

Le logiciel Anim 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 [BRU_89].

Ce logiciel original se base sur les principes d'animation traditionnelle et offre des possibilités temps réel de description et de visualisation de films d'animation. Anim vise à minimiser les manipulations des animateurs et à optimiser les temps de production des films d'animation. Le pilotage de matériels vidéo permet d'enregistrer plusieurs minutes d'animation en continu et donc de générer un film complet d'animation en réalisant automatiquement la phase de montage.

Le logiciel utilise les avantages des mémoires à haute capacité et des opérations temps réel câblées. Le plan mémoire, support matériel des dessins d'un acteur, est appelé « image ». Le format d'une « image » est de 1024 x 1024 pixels. A chaque pixel est attribuée une couleur choisie parmi 16,7 millions.

Le système complet est composé de plusieurs plans mémoire 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 différents plans mémoire est réalisée en temps réel par le matériel à la fréquence du balayage vidéo.

Anim permet d'animer jusqu'à 10 acteurs simultanément à condition de posséder la configuration matérielle nécessaire. Le système gère chaque image indépendamment. Les images peuvent être superposées comme les films de celluloïde de l'animation traditionnelle.

Trois concepts régissent la réalisation d'un film d'animation :

- l'animation des acteurs et la génération des cells;
- la description de la scène et la génération des séquences;
- la description du scénario et la génération du film.

4.1 ACTEURS ET CELLS.

Un acteur est défini par un ensemble de dessins regroupés sur une image et par des indications de position, de séquence par rapport à l'écran. Les dessins représentent toutes les positions que peut prendre l'acteur au cours de son animation.

Le cell est le support des différents dessins d'un acteur de la même manière que la feuille de celluloïde supporte les dessins en animation traditionnelle. Un cell se matérialise par une boite rectangulaire englobant le dessin de l'acteur. A chaque cell est associé un point guide, placé par défaut au centre du cell. Ce point sert de référence pour tous les dessins d'un même acteur et assure une continuité de la trajectoire lors du passage d'un cell à un autre.

Une animation consiste à afficher le cell courant de chaque acteur à la position désirée sur l'écran avec ses caractéristiques et de changer de cell et de position simultanément et en temps réel.

4.2 SCENES ET SEQUENCES.

La notion de scène est déterminée par trois facteurs :

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

Le cadre de l'action est déterminé par la position des cells par rapport à un cell de référence matérialisant le décor. L'ordre d'affichage des cells traduit la notion de profondeur. La mise en scène représente l'ensemble des informations propres à l'animation comme l'ordre de superposition des cells.

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 est en général enregistrée sur un support vidéo.

L'Anim est prévue pour fonctionner à partir de cells stockés sous la forme de fichiers images sur le disque du système. Le système est totalement interactif et autorise l'enregistrement en continu de séquences d'animation.

4.3 SCENARIO ET FILM.

Le scénario regroupe les différentes informations permettant le raccord des séquences. Le film est le résultat du montage de toutes les séquences décrites dans le scénario.

4.4 LE TEMPS DANS L'ANIM.

Le standard vidéo est de 1/25ème de seconde par image c'est à dire qu'une seconde de film nécessite en animation traditionnelle la prise de 25 photographies de la scène à animer.

En vidéo, la notion de trame désigne une demi-image d'une nature particulière. Une image complète est constituée de deux trames consécutives dont les lignes respectives (paires et impaires) sont imbriquées selon un procédé appelé « entrelacé ». Dans les standards de télévision employés en Europe, une trame est composée de 321.5 lignes et dure 20 ms.

L'unité de temps désignée pour décrire une scène est le 1/50ème de seconde c'est à dire le retour de trame. En effet, l'Anim manipule 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. Une telle contrainte autorise des déplacements visiblement plus lisses que la même animation définie au niveau de l'image.

4.5 DESCRIPTION DE LA TRAJECTOIRE.

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, corrélée avec une fonction d'évolution, qui traduit la progression de l'acteur le long de cette trajectoire au cours du temps.

Une trajectoire est représentée à l'aide d'une suite de morphologies de type ligne brisée et spline. L'utilisateur détermine les points clés aux instants stratégiques et fixe la position de l'acteur. Le placement d'un point clé détermine une étape de la séquence. Lorsqu'il n'existe pas de point clé à un instant t, le système interpole de lui-même les valeurs intermédiaires de tous les paramètres des étapes définies.

4.6 DES FONCTIONS EVOLUEES.

Aux deux fonctions classiques de l'animation, Anim offre de nouveaux effets d'animation :

- Le fenêtrage : une fenêtre peut être associée à chaque acteur. Combinée avec l'effet de transparence, elle permet de rendre l'acteur visible ou invisible à l'intérieur d'une zone rectangulaire.
- La transparence : l'effet de transparence est appliqué distinctement à la zone interne ou externe de la fenêtre. Par défaut un acteur est totalement visible à l'intérieur de sa fenêtre et totalement invisible à l'extérieur. L'effet de transparence peut être appliqué de façon continue entre les deux bornes permettant la réalisation d'un fondu au noir (disparition progressive de l'acteur en le rendant de plus en plus transparent) ou de fondu enchaîné (un deuxième acteur apparaît progressivement pendant que le premier disparaît).
- Déformations : des effets de mosaïque et des déformations cylindriques peuvent être appliqués à un acteur.
- Le système muni d'une carte entrée vidéo permet la combinaison d'acteurs entièrement dessinés avec une séquence de vidéo live. Ce mécanisme est particulièrement utile au montage d'un texte défilant devant une image réelle (générique, messages publicitaires par exemple).

5. LES SYSTEMES GETRIS.

5.1 GETRIS IMAGES.

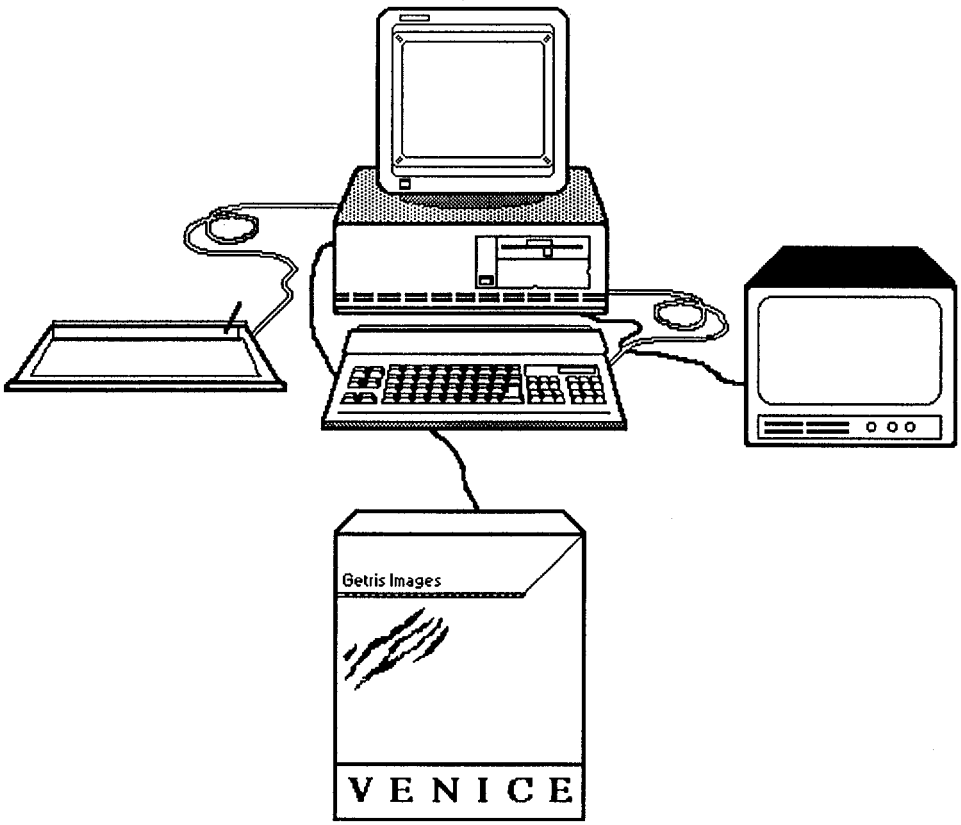
La société Getris Images a été créée en Septembre 1985. Elle est l'aboutissement d'un projet d'industrialisation d'un système de synthèse d'images né de recherches universitaires à Grenoble [MAR_82]. Getris fait partie depuis déjà plusieurs années du cercle restreint des constructeurs de systèmes vidéographiques de post production haut de gamme.

Le premier système commercialisé avait pour origine le système GETRIS pour « GEnérateur Temps Réel d'Images de Synthèse ».

5.2 ARCHITECTURE DES SYSTEMES.

L'architecture d'un système Getris se décompose en modules [GET_94B] :

- la partie de synthèse d'images, spécialisée dans le traitement et la mémorisation des images;
- un ordinateur de type PC servant de calculateur pilote;
- des logiciels : les applications;
- des dispositifs de dialogue entre les applications et l'utilisateur (souris, tablette);
- un dispositif de visualisation (moniteur vidéo);
- des unités de stockage des images (magnétoscopes, disques durs, etc.).



Le PC a été choisi en raison de son rapport qualité/prix. Il est muni d'une carte de communication pour être interfacé avec la machine de synthèse. Le calculateur n'est pas conçu pour gérer la masse d'informations nécessaire en synthèse et traitement d'images.

Une partie matérielle indépendante a donc été développée. Elle se comporte comme un automate câblé, exécutant les traitements au rythme de la vidéo. La machine de synthèse se compose d'une série de cartes électroniques, chacune d'elle ayant un rôle particulier. La puissance et l'originalité des systèmes GETRIS est liée à la présence d'opérateurs câblés directement sur ces cartes et à l'organisation multicouche du système. L'architecture du système VENICE est représentative des systèmes GETRIS.

Le module de synthèse d'images du système peut être décomposé en 2 parties :

- la partie traitement se compose de la carte de contrôle, du plan de menu/stencil (VOP), de la partie traitement des plans mémoire (VPM);
- la partie vidéo des plans mémoire, la carte d'entrée vidéo analogique (VVA), la carte mélangeur, les cartes d'entrée vidéo numérique (VAN), constituent la partie vidéo.

Tous les éléments du système fonctionnent au format vidéo 4:4:4:4 PRVB. Avec ce standard, une couleur est codée sur 32 bits :

- 8 bits pour la pondération P (facteur de transparence);
- 8 bits pour chacune des 3 composantes de couleur Rouge, Vert et Bleu.

Le monde vidéo quant à lui est au format 4:2:2 YUV :

- Y est la luminance de l'image (représentation de l'image en noir et blanc);

- U et V sont les facteurs de chrominance (couleur de l'image).

Le terme 4:2:2 signifie que les signaux de chrominance sont échantillonnés 2 fois moins souvent que celui de luminance.

Il est 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; facilement calculables à partir des relations suivantes; peuvent être directement câblées sur les cartes d'entrée / sortie :

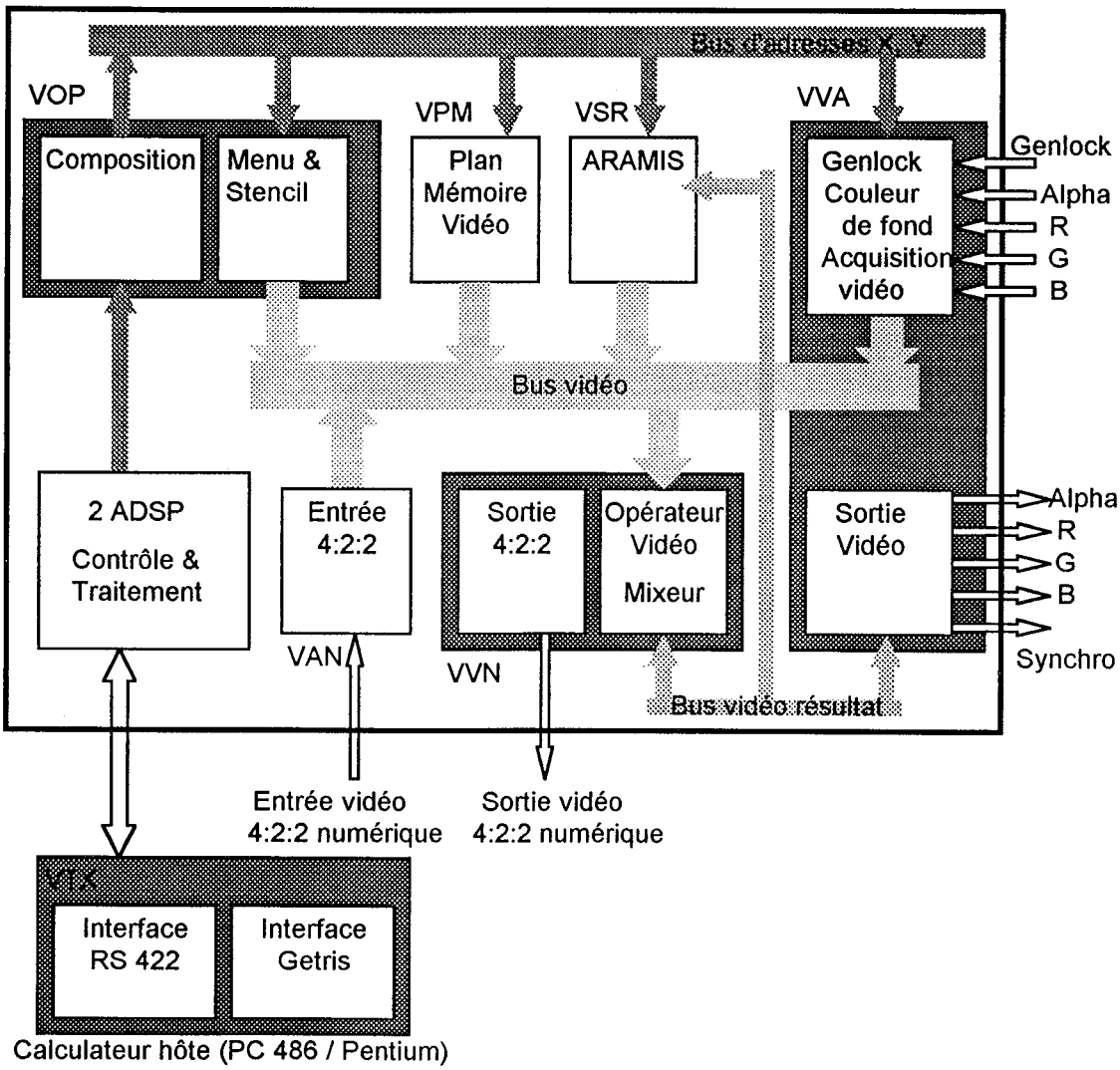
$$Y = 0.299 * R + 0.587 * V + 0.114 * B$$

$$U = 0.713 * (R - Y)$$

$$V = 0.564 * (B - Y)$$

5.3 LES PRINCIPAUX COMPOSANTS DE VENICE

Le système VENICE



5.3.1 Le VPM : Venice Plan Mémoire.

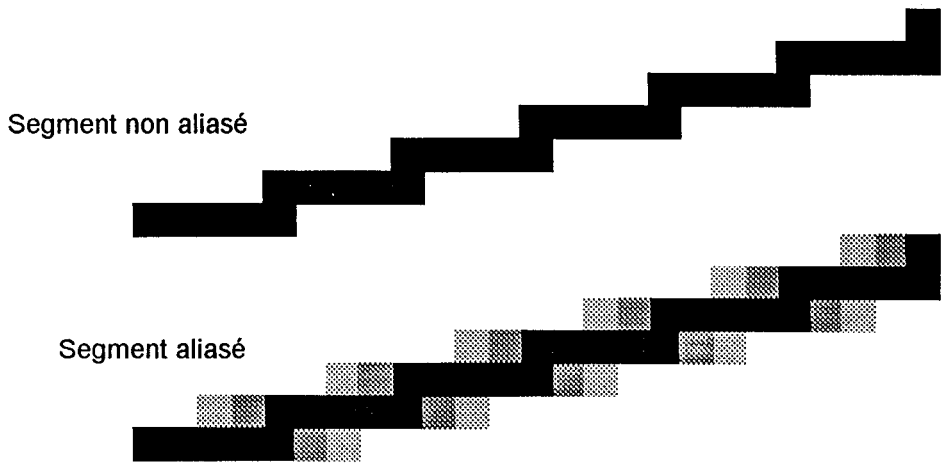
Le VPM est un plan mémoire au format 1024 x 1024 par 32 bits. Il permet de stocker une image en 16 millions de couleurs, plus une information de pondération. Il comporte 2 mémoires de déformation permettant d'effectuer les translations, le fenêtrage, le zoom, les mosaïques.

La pondération agit au niveau de chaque pixel du plan. Elle permet de régler la transparence de l'image. Sa valeur stockée sur 8 bits varie entre 0 et 128. A 128, le pixel est complètement opaque et à 0 il est complètement transparent (invisible). De plus, la transparence englobe la notion d'antialiasing. L'aliasing est le phénomène de marches d'escalier qui se produit lors de la génération de morphologies (cercles, droites, etc.).

Le pourcentage réel de recouvrement d'un pixel par la couleur est appliqué en tant que pourcentage de transparence de la couleur de ce pixel :

Recouvrement 1/2 Transparence 50%

La technique d'antialiasing consiste à lisser les segments pour éviter qu'ils ne présentent les brutales transitions de lumière responsables de l'aliasing [TOU_87] :



5.3.2 La VVA : Venice Vidéo Acquisition.

Elle assure l'interface entre le système de synthèse et le monde vidéo. Elle contient une entrée vidéo analogique permettant l'acquisition et le gel de l'image vidéo. Elle est munie d'un convertisseur 4:2:2 YUV vers 4:4:4 RVB.

La VVA reçoit l'information de visibilité de chaque plan, et en fonction d'une mémoire de priorité, décide de celui qui est le plus devant, et place les informations sur les 4 bus vidéo A, B, C et D.

Lorsque tous les pixels sont transparents, une couleur de fond est affichée. Cette couleur est mémorisée dans les registres de la VVA.

5.3.3 Le VOP : Venice Opérateur Pixel.

Cette carte réalise 2 fonctions :

- elle accélère les calculs à la génération;
- elle réalise de manière câblée les opérations de décalque et de stencil.

Elle possède le plan de menu / stencil. C'est un plan 1024 x 1024 x 16 bits dont 8 sont réservés pour l'affichage des menus, du curseur. Les 8 autres contiennent le stencil (masque de certaines parties des plans mémoire vidéo). Ce plan fonctionne par indirection sur des tables de couleurs.

5.3.4 La VAN : Venice Acquisition Numérique.

C'est une carte d'entrée numérique au format 4:2:2 numérique sur laquelle peut être appliqué une translation et une pondération.

5.3.5 Le DVE-Layer : Digital Vidéo Effects Layer.

Le DVE-Layer est un plan mémoire 1024 x 1024 pixels avec une interface identique à un VPM. Il comporte des mécanismes câblés permettant la réalisation de transformations 3D (rotations, translations) en temps réel.

Les effets sont traités avec une extrême résolution puisque le système génère par interpolation 32 « sous-pixels » pour chaque pixel de l'image. Des zooms de rapport 32 peuvent ainsi être réalisés sans altération de la qualité.

Il est équipé d'un système de filtres avec calcul automatique du flou. L'objectif d'un tel filtre est l'amélioration de la qualité d'image obtenue lors d'une déformation avec un fort rapport de réduction, celle-ci conduisant à un aliasing inacceptable.

5.3.6 Le stockage des informations.

Les systèmes Getris sont capables de gérer des unités de stockage vidéo de tous types :

- disques durs classiques ou optiques;
- magnétoscopes ou VTR (Vidéo Tape Recorder);
- Aramis (RAM recorder).

Les informations stockées sur disque sont compactées : pour un segment horizontal de même couleur, seules sont mémorisées les informations de début de segment et de nombre de pixels.

Certains supports peuvent entraîner des problèmes de dégradation de l'image lors de multi-génération (la même image est lue, travaillée puis sauvegardée plusieurs fois). Ces problèmes de dégradation sont liés à la conversion du format du système de synthèse d'images 4:4:4:4 numérique vers le format analogique des scopes.

Aramis est composé de cartes VSR (Venice Silicon Recorder). Ces cartes contiennent de la mémoire DRAM pouvant mémoriser jusqu'à 10 secondes d'images dans le format du système. L'accès à une image est immédiat et la qualité des images est inaltérable.

5.4 LA GAMME DES SYSTEMES GETRIS.

Eclipse est prévue pour répondre à la plus grande partie des besoins des professionnels : palette graphique, animation et rotoscoping. La configuration de base se compose d'une palette graphique 2 plans avec en option un module d'animation 2D temps réel et 2 plans supplémentaires.

La palette graphique **Paint** permet de travailler sur plusieurs images antialiasées et en 16.7 millions de couleurs. En plus des outils classiques du graphiste tels que pinceaux, brosses, le module permet de réaliser des effets spéciaux, des déformations. Il contrôle les magnétoscopes pour le rotoscoping.

Le module d'animation **Anim** permet le mélange en temps réel de 2 à 4 plans animés avec de la vidéo "live". Les différentes couches sont gérées indépendamment et sont munies de transparence, position et visibilités propres. L'animation peut être modifiée simplement avant son enregistrement sur magnétoscope. Elle est visualisable en temps réel après un temps de calcul réduit.

Pour **Venice**, la palette graphique a été complétée avec de nouvelles fonctionnalités. En version de base, ce système est muni de 4 plans mémoire.

Le module palette graphique est basé sur un concept multicouches : l'utilisateur dispose de plusieurs plans images indépendants et antialiasés. Chaque plan est créé ou modifié indépendamment des autres à l'aide d'outils graphiques tels l'aérographe, les pinceaux, le gaufrage, le zoom. Un système de macros instructions libère le graphiste des tâches répétitives ce qui améliore le temps de création.

Le module Anim permet de créer et contrôler des positions clés pour n'importe quel plan. Il permet entre autres d'ajuster à tout instant la visibilité, la transparence, la position des acteurs dont le nombre peut aller jusqu'à 10. Le module donne la possibilité de produire des effets comme les fondus enchaînés, les déplacements, les déformations géométriques, etc.

Studio Venice est né en septembre 1991. Il est une extension du système Venice auquel a été ajouté le magnétoscope intégré **Aramis**. Muni du logiciel d'animation et de composition : le **Sequencer**, il est adapté aux opérations d'animation image par image et de multi-génération. Il intègre l'animation d'acteurs 2D dans un espace 3D. Le résultat provisoire est stocké sur Aramis évitant l'altération des images. La séquence de film en cours d'édition peut être visualisée sans le temps de pré-enroulage lié aux magnétoscopes classiques.

La présentation de **Hurricane** au NAB (Salon international de la post production vidéo à Las Vegas) en mars 1994, constitue une avancée technologique importante pour les systèmes Getris.

La ligne Hurricane intègre les nouveaux modules DVE-Layer permettant :

- l'animation d'acteurs 2D dans un espace 3D en temps réel;
- des calculs au sous-pixel.

Afin de bénéficier de la totalité des possibilités de ces nouvelles cartes, Hurricane est muni d'un tout nouveau logiciel d'animation : Le **DVE-Composer**.

CHAPITRE 2

LE CAHIER DES CHARGES

1. ETUDE DE LOGICIELS D'ANIMATION.

Force est de constater qu'il existe aujourd'hui une offre de logiciels haut de gamme fonctionnant sur stations de travail, et capables de répondre à toutes les phases de production d'un film. D'un autre côté, les solutions économiques sur PC peuvent satisfaire les budgets moins ambitieux de communication d'entreprise.

Un nombre limité de sociétés commercialisent des systèmes vidéographiques de post-production permettant de réaliser des films d'animation. Ces systèmes haut de gamme permettent d'obtenir une qualité d'image parfaite, utilisable en production cinématographique. Getris Images fait partie de ce cercle restreint dans lequel on trouve également Quantel, Colorgraphics, Symbolics ou Silicon Graphics.

Les logiciels dits de présentation multimédia disponibles sur PC ne nécessitent pas de connaissance du multimédia et permettent de réaliser des petits shows très impressionnants à moindre frais [S&M_109].

1.1 DES SYSTEMES COMPLETS D'ANIMATION

De tels systèmes sont utilisés par des professionnels de la post-production vidéographique pour :

- la réalisation de films et de vidéos d'entreprise;
- la création de génériques et de publicités pour la télévision et le cinéma;
- l'habillage des chaînes de télévision;
- les trucages et les effets spéciaux du cinéma;
- les clips vidéo.

Les systèmes commercialisés par Getris Images sont équipés de logiciels de palette graphique et d'animation : Paint et DynamicPaint pour la palette graphique; Anim, Sequencer et DVE-Composer pour l'animation.

Les concurrents de Getris Images proposent des systèmes munis d'une palette graphique comme la PaintBox de Quantel [QUA_89] ou la DP Painter de Colorgraphics [COL_92] ou encore S-Paint de Symbolics [SYM_90].

Harriet, Harry et Hal de Quantel, Matador de Silicon Graphics, DP Animator de Colorgraphics ou S-Dynamics de Symbolics permettent de réaliser des animations 2D avec transformations

dans un espace 3D. Ces systèmes ne permettent pas l'animation en temps réel car les calculs sont en général effectués par logiciel et nécessitent des calculateurs très puissants comme les stations de travail SiliconGraphics [SGI_93].

Quantel propose un logiciel d'animation proche du Sequencer de Getris Images. L'animation est proposée de façon grossière en prévisualisation (Preview) afin d'être réalisée en temps réel. Par contre le calcul de l'animation sans pour autant être effectué en temps réel est beaucoup plus rapide que dans le Sequencer (utilisation de la puissance de calcul de la station de travail). Le résultat de l'animation est enregistré sur un « Carroussel » (RAM Recorder) permettant la visualisation finale en temps réel. Les effets et transformations proposés sont proche de ceux du Sequencer : effets animés de transformations 2D et 3D, effets vidéos (fondu enchaîné), effets spéciaux (colorisation, contraste).

Ces effets se retrouvent sur d'autres systèmes avec quelques particularités : Symbolics propose l'insertion d'un décor 3D en fond d'une animation 2D ou bien l'effet Scatter & Randomize où l'image est découpée en fragments dispersés dans tous les sens à travers l'écran.

SiliconGraphics produit des stations de travail très performantes malgré un coût très élevé. Elles accueillent des logiciels de post-production vidéographique que leur puissance peut mettre en valeur comme Digital Studio de Softimages [SOF_93]. Ce logiciel inclut l'animation 3D, l'édition de vidéo, de sons et une animation par cells 2D.

1.2 LES LOGICIELS D'ANIMATION SUR PC

Tous ces logiciels manipulent divers objets tels qu'images, graphiques, textes, sons, séquences vidéo digitale. Ils permettent de créer des présentations d'entreprise, des programmes et des jeux éducatifs, des démonstrations de logiciels, etc. [MAC_94].

Ces logiciels se distinguent les uns des autres par le mode utilisé pour la construction et l'animation du scénario. Ainsi, pour Action (Macromédia) [S&M_107], les présentations animées et sonorisées sont divisées en scènes; chaque scène contenant des éléments choisis par l'utilisateur-réalisateur. Les différentes scènes sont liées afin de définir leur ordre d'apparition.

D'autres encore; comme Authorware Professionnal (Macromédia) ou Icon Author (Aimtech); utilisent des icônes placées sur l'écran de travail afin de dessiner un chronogramme de l'application. Chaque icône permet l'accès à un système de dialogue pour paramétrer les fonctions associées.

Multimédia Orchestré (Myclog) et Médiablitz (Asymetrix) utilisent une grille de programmation où une ligne représente une séquence dans laquelle sont placés les objets (sons, images, texte). Multimédia Orchestré inclut des objets de transition entre chaque séquence comme un objet de fond ou une pause.

Certains logiciels haut de gamme sont beaucoup plus complexe à utiliser. Multimédia Book (Asymetrix) inclut un langage de programmation à objets intégrés. Il utilise le concept du livre : une application est constituée de chapitres et de pages. Une page reçoit des objets textes, graphiques et multimédia.

2. UNE NOUVELLE APPLICATION D'ANIMATION.

Le dernier né de la gamme des systèmes Getris, Hurricane, intègre la nouvelle technologie des DVE-Layers. Afin de bénéficier de toutes les capacités de ces cartes, une nouvelle application d'animation temps réel doit être conçue : Le DVE-Composer. Cette nouvelle application est développée par l'ensemble de l'équipe Recherche & Développement Logiciel. Les modules conçus au cours de ce mémoire s'intègrent dans cette application. Ils sont précisés dans la partie traitant des nouvelles notions d'animation de ce cahier des charges.

2.1 LES CONCEPTS DE L'ANIM.

L'Anim est un logiciel d'animation temps réel très performant [GET_92]. Il permet une économie de temps importante pour la description, la visualisation et l'enregistrement des animations. Le mode de description d'une animation est très simple. L'utilisateur peut visualiser en temps réel son animation avant l'enregistrement final sur magnétoscope. Cette technique de « pré-édition » est un atout important de l'animation assistée par ordinateur.

La nouvelle application propose l'ensemble des fonctionnalités de l'Anim c'est à dire la gestion du scénario et du contexte de film, des calculs et une animation en temps réel, la manipulation des acteurs et de leurs attributs.

2.2 DES FONCTIONS DU SEQUENCER.

L'Anim ne permet pas de réaliser des transformations ou des rotations dans un espace 3D. Le Sequencer [GET_93] propose ces transformations mais elles ne sont pas effectuées en temps réel car calculées par logiciel.

Le DVE-Composer inclut l'attribut de géométrie du Sequencer qui permet l'animation d'un acteur 2D dans un espace 3D. Cette animation peut être réalisée en temps réel grâce aux cartes DVE-Layer. Ces dernières offrent des calculs de transformations au sous-pixel, calculs effectués par logiciel dans le Sequencer.

2.3 INTEGRATION DES DVE-LAYERS.

Cette nouvelle application doit être entièrement compatible avec les systèmes qui ne sont pas équipés de DVE-Layers soit Eclipse, Venice et Studio Venice.

Ces modules permettent d'offrir de nombreuses possibilités d'effets :

- effets de géométrie : rotation autour des 3 axes X, Y et Z, mise en perspective;
- effet de Defocus : flou de l'image;
- effets de déformation manuels : morphing, corner pining;
- effets de déformation calculés : drapeau, ondes, cylindre, mosaïque, etc.

2.4 TEMPS REEL ET IMAGE PAR IMAGE.

Un effet temps réel nécessitant un module de type DVE-Layer doit pouvoir être effectué en non temps réel sur un autre module avec si nécessaire l'intervention d'un module supplémentaire pour sa génération.

La séquence de sortie sur laquelle est enregistrée le résultat de l'animation peut être un support quelconque (disque, VTR, Aramis, etc.) ce qui veut dire que l'édition peut être réalisée soit en temps réel, soit en image par image lorsque le support nécessite un temps de réaction.

La visualisation de l'animation en temps réel (Preview) est rendue possible même si certaines opérations ne le sont pas grâce à une estimation du pas de calcul en fonction de l'opération.

2.5 DES MODULES LOGIQUES.

Une perspective d'un objet dont la taille est supérieure ou égale à celle de l'écran, nécessite 2 VPMs : un pour l'objet source et un pour le résultat. Cet effet peut être réalisé en attachant à l'acteur un module logique composé de 2 modules physiques VPMs.

De même, pour réaliser un effet de cylindre propre sur les bords, on a besoin de 3 DVE-Layers. Cet effet sera réalisé en attachant à l'acteur un module logique composé de trois modules physiques DVE-Layers.

La notion de module logique est un concept nouveau qu'il est utile de mettre en place. Ainsi, le nombre des acteurs n'est pas limité au nombre de couches physiques de la machine.

3. DES NOUVELLES NOTIONS D'ANIMATION.

La manipulation des acteurs s'effectue par la description de paramètres spécifiques caractérisant le comportement de chacun d'eux. Ces paramètres constituent à part entière les attributs de l'animation. L'étude qui suit concerne principalement la gestion des attributs et constitue une nouvelle approche de l'animation.

Getris Images veut profiter de la réalisation de cette application pour intégrer de nouvelles notions d'animation, en particulier :

- la composition d'attributs;
- la multiplication des méthodes de description des attributs;
- l'animation d'ensemble, la gestion automatique des profondeurs;
- l'asservissement d'un attribut par un autre.

Ces nouvelles notions peuvent difficilement être introduites dans les applications existantes dans la mesure où elles ont une incidence importante sur les mécanismes de base choisis pour la construction de l'application.

Le calcul de l'animation doit permettre de composer des attributs d'une même classe entre eux (attributs de la classe géométrie par exemple). Cela implique que le même attribut peut être décrit plusieurs fois pour un même acteur et que le résultat doit être la combinaison de plusieurs transformations.

La combinaison d'attributs d'une même classe, peut conduire à la conception de nouveaux attributs intégrant directement tous les attributs composés. Par exemple, l'attribut rotation autour d'un point permet de simplifier la description de l'animation puisque l'utilisateur n'a pas à gérer indépendamment la rotation et la translation.

L'application doit permettre à l'utilisateur de choisir son mode de description d'un attribut parmi plusieurs en fonction de l'effet qu'il veut réaliser.

La trajectoire d'un acteur peut être décrite de façon classique en indiquant sa position en x et y. Cette même trajectoire peut suivre les contours d'une forme géométrique (shape) dessinée à l'aide de la palette graphique. Elle peut également être décrite par un angle de rotation sur un cercle dont l'utilisateur fixe le centre et le rayon.

Ce choix permet à l'utilisateur de décrire une trajectoire complexe ou représentant une forme géométrique en utilisant des formes ou des fonctions d'évolution prédéfinies. Le mode de description des formes est très visuel et facile à comprendre pour un utilisateur non averti et ignorant les lois élémentaires de la cinématique.

Les effets spéciaux se retrouvent au niveau de l'animation d'ensemble : fondu au noir, réduction ou grossissement de toute l'animation, application de géométries (translation, rotation) sur l'ensemble de la scène. L'attribut de profondeur automatique doit libérer l'utilisateur de la gestion de l'ordre de superposition des différents acteurs.

La notion d'asservissement entre attributs peut être mise en oeuvre pour lier l'évolution d'un attribut à celle d'un autre. L'exemple le plus simple est le fondu enchaîné : un acteur apparaît progressivement à la place d'un acteur qui disparaît. Les attributs de pondération qui permettent de réaliser cet effet doivent évoluer au même rythme pour que le résultat soit parfait.

4. DES CONTRAINTES A RESPECTER.

Contrainte technique : le logiciel est écrit en langage C sur micro-ordinateur de type PC 486 ou Pentium avec le système d'exploitation MS-DOS. Cet environnement est imposé par le fait qu'il est celui de toutes les applications des systèmes Getris.

Contrainte commerciale : Un premier prototype doit être conçu et réalisé pour la deuxième moitié du mois de mars 1994, date du NAB (National Association of Broadcasters), salon international déterminant au niveau de la politique commerciale de Getris Images. La première version du produit sera commercialisée à partir du mois de juillet 1994.

CHAPITRE 3

ETUDE ET CONCEPTION

Les nouvelles possibilités temps réel des DVE-Layers à savoir rotations, mise en perspective, effets, etc. doivent être intégrées dans une application d'animation.

La première solution consiste à faire évoluer l'Anim temps réel en ajoutant dans un premier temps un attribut de rotation. Cette solution offre l'avantage de limiter les coûts de développement puisqu'il n'est pas utile de développer une nouvelle application. Par contre, l'ordre d'évaluation des attributs est figé. Autrement dit, l'ordre de composition des commandes de géométrie est imposé : rotation, trajectoire, zoom par exemple; ce qui est rapidement limitatif. D'autre part, La représentation des attributs par des icônes n'est pas très explicite si l'on désire ajouter par la suite des attributs d'effets. Il est parfois nécessaire de se référer à la documentation pour connaître la signification d'une icône.

La seconde solution consiste à développer une nouvelle application d'animation qui permette d'intégrer facilement les DVE-Layers et d'en profiter pour introduire de nouvelles notions d'animation. La clé de la réussite de cette nouvelle application est la mise en place d'un nouveau mécanisme de gestion des attributs qui soit suffisamment souple pour permettre :

- de composer les attributs;
- de ne pas avoir un seul dialogue pour chaque attribut afin de proposer des attributs simples pour l'utilisateur lambda et des attributs complexes pour le professionnel ou le chercheur;
- de ne pas proposer systématiquement tous les attributs mais seulement ceux utilisés par l'utilisateur;
- de représenter les attributs par un texte et non par des icônes.

L'avantage de cette solution est de proposer un système entièrement nouveau, évolutif et extensible mais avec un temps de développement beaucoup plus long.

La direction de Getris Images décide de réaliser une nouvelle application et me confie la mise en place du nouveau mécanisme de base de gestion des attributs.

Ce chapitre présente l'étude accomplie. Il fait apparaître la notion de classe d'attributs permettant la composition des attributs dans une même classe.

En seconde partie du chapitre nous proposerons de nouveaux modes de description de certains attributs. La mise en place de la composition des attributs permet d'obtenir des animations d'ensemble où un attribut s'applique par composition à l'ensemble des acteurs du scénario.

Enfin, il est apparu utile de pouvoir lier l'évolution des attributs entre eux afin de contrôler l'animation à l'aide d'un minimum d'opérations : l'asservissement des attributs tente de résoudre cette question dans la dernière partie du chapitre.

1. DESCRIPTION DES ATTRIBUTS

Il s'agit de mettre en place un mécanisme de gestion des attributs où les acteurs n'ont pas forcément le même nombre d'attributs et où les attributs sont regroupés en classes. Ce système permet une grande souplesse d'utilisation. Seuls les attributs nécessaires à l'animation sont décrits dans le scénario. La gestion par classe permet l'ajout et la suppression des attributs afin de définir l'ordre d'application de ceux-ci.

1.1 LES CLASSES D'ATTRIBUTS

Les attributs peuvent être classés en catégories de part leur nature :

- attribut de cell;
- attributs de géométrie;
- attributs de fenêtre;
- attributs de pondération;
- attributs d'effets spéciaux;
- etc.

La classe des attributs de géométrie est la plus riche avec les rotations, les translations et le changement d'échelle. Certains attributs constituent une classe à eux seuls comme la couleur de fond, l'attribut de focale, le défocus ou la profondeur.

1.2 LES ATTRIBUTS

1.2.1 L'attribut de cell

Un cell est la représentation d'un acteur 2D à un instant donné de l'animation. Une séquence est constituée de l'ensemble des cells de l'acteur qui représentent l'ensemble des positions que peut prendre l'acteur au cours du scénario. L'attribut de cell permet de désigner la séquence utilisée ainsi que le numéro du cell à visualiser. L'attribut contient les références de la séquence (nom du fichier contenant les cells sur disque, numéro du premier et du dernier cell de la séquence). Le numéro de cell courant à afficher est désigné par l'utilisateur.

1.2.2 Les attributs de géométrie

1.2.2.1 Le zoom

Le zoom s'applique selon les deux axes X et Y de manière indépendante à l'aide de 2 facteurs distincts : Zoom en X et Zoom en Y. Ces facteurs en standard valent 1 (pas de zoom) et peuvent varier entre -20 et +20 (valeurs reprises de l'Anim).

1.2.2.2 La trajectoire

L'attribut de trajectoire détermine la position de l'acteur ou plus précisément la position du point guide du cell de l'acteur. Le point guide est le point invariant des différents cells de l'acteur.

Tous les calculs de coordonnées et de transformation sont effectués par rapport à ce point. Par défaut, le point guide est placé au centre de l'écran.

Le mode de description de cet attribut est repris des logiciels Anim et Sequencer. L'utilisateur décrit des points clés qui définissent une trajectoire. Il choisit un mode d'interpolation fixe, linéaire ou lissée pour le calcul des points intermédiaires.

1.2.2.3 La translation en Z

Cet attribut simule le déplacement d'un acteur en avant ou vers l'arrière. La translation varie entre -2000 et 500 (valeurs reprises du Sequencer).

1.2.2.4 Les rotations

Elles sont au nombre de 3; une autour de chacun des axes X, Y et Z. La rotation est obtenue à partir de la valeur de l'angle et du nombre de tours soit :

rotation = nb tours * 360 + angle

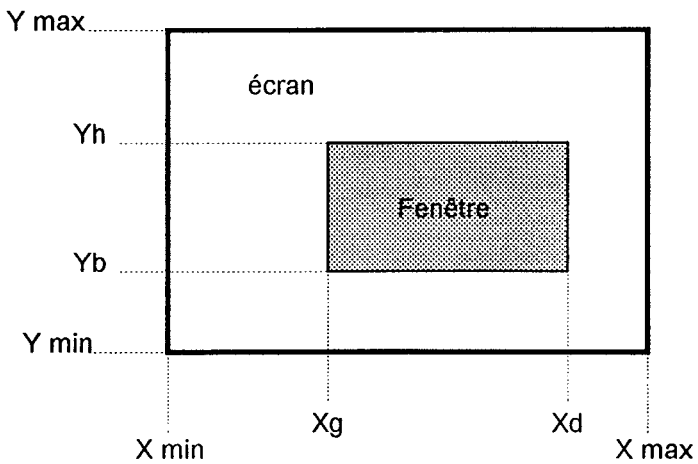
Une fonction de calcul permet de retrouver le nombre de tours et la valeur de l'angle :

angle = rotation modulo 360

nb tours = partie entière de rotation / 360

1.2.3 L'attribut de fenêtre

La fenêtre détermine une zone rectangulaire de l'écran dans laquelle un acteur peut être rendu visible ou invisible. Elle est décrite par les coordonnées du coin bas-gauche et celles du coin haut-droit de la zone soit les 4 valeurs des coordonnées : Xg, Yb et Xd, Yh. Ces valeurs sont comprises entre 0 et les coordonnées maximum de l'écran (largeur et hauteur de l'écran).



1.2.4 L'attribut de pondération

Cet attribut permet de rendre plus ou moins transparent / opaque un acteur. Il permet de réaliser des fondus au noir ou des fondus enchaînés entre acteurs.

La pondération s'applique à 2 niveaux des acteurs :

- la pondération interne est appliquée à l'intérieur de la fenêtre d'un acteur. Par défaut elle prend la valeur maximum (128) rendant l'acteur visible;
- la pondération externe est appliquée à l'extérieur de la fenêtre d'un acteur. Par défaut elle prend la valeur minimum (0) rendant l'acteur invisible en dehors de sa fenêtre;

Ces pondérations sont décrites à l'aide d'une valeur variant entre 0 et 128.

1.2.5 L'attribut de defocus

Cet attribut permet de rendre une image plus ou moins floue.

1.2.6 La couleur de fond

Elle est affichée dans les zones ou aucun acteur n'est visible. Un effet de dégradé vertical peut être réalisé en plaçant des couleurs différentes en haut et en bas de l'écran. Pour cela 8 valeurs sont utilisées : les 4 premières définissent les 4 composantes PRVB de la couleur du haut. Les 4 dernières définissent les 4 composantes de la couleur du bas.

La composante P varie entre 0 et 128 et les 3 composantes de couleur évoluent entre 0 et 255.

1.2.7 L'attribut de focale

La focale détermine la position de l'observateur par rapport à l'écran. Elle simule l'éloignement ou le rapprochement de la caméra par rapport à la scène.

Elle est décrite par une distance sur l'axe des Z. Sa valeur varie entre 500 et 32767 où 32767 représente l'infini.

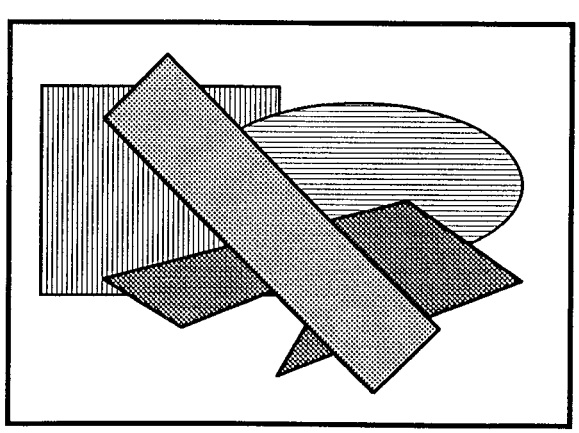
1.2.8 L'attribut de profondeur

Cet attribut permet de positionner la priorité des acteurs les uns par rapport aux autres. Plus un acteur possède une priorité faible et plus il est devant. Un tableau contient les numéros des acteurs dans l'ordre de leur apparition. Les acteurs qui ne sont pas référencés dans ce tableau sont invisibles.

Exemple avec 5 acteurs :

Tableau de priorité	
1	Acteur 2
2	Acteur 4
3	Acteur 1
4	Acteur 3

Acteur 5 invisible



2. LA COMPOSITION DES ATTRIBUTS

2.1 INTRODUCTION

La puissance de la classification des attributs est mise en évidence par la composition des attributs. Afin de produire des animations complexes et de qualité, les différents attributs doivent pouvoir être composés à l'intérieur d'une même classe et notamment dans la classe des attributs de géométrie. Cette classe regroupe les attributs de zoom, de translation (trajectoire et translation en Z) et de rotation. Tous ces attributs agissent sur la position et l'orientation finale de l'acteur à l'écran.

Dans le Sequencer, la composition a été introduite par l'intermédiaire des programmes de géométrie. Le principe de composition reste identique mais le mode d'accès à la composition doit être simplifié : le programme de géométrie du Sequencer n'est pas considéré comme un attribut et il est accessible une fois que le programme de dessin a été créé.

Dans le DVE-Composer, la composition est plus naturelle dans la mesure où les attributs sont regroupés par classes et que seul l'ordre de ces attributs détermine le résultat de la composition.

2.2 LES TRANSFORMATIONS GEOMETRIQUES

La représentation universelle d'une transformation linéaire par une matrice carrée correspond à l'usage des coordonnées homogènes. Un point d'un repère à n dimensions est décrit par un vecteur de n+1 coordonnées. La coordonnée supplémentaire est nommée W [SCH_87].

Dans le cas du plan, un vecteur s'exprime en coordonnées cartésiennes (x, y). En coordonnées homogènes le même vecteur est écrit (X, Y, W) avec :

- X = w * x
- Y = w * y
- W = w où w est le facteur d'échelle en général égal à 1.

Les principales propriétés des coordonnées homogènes sont [MAR_84] :

- la possibilité d'exprimer par une matrice unique toute transformation (rotation, homotétie, translation et même projection);
- la possibilité d'exprimer des points très éloignés ou situés à l'infini;
- la possibilité d'exprimer toute combinaison de transformations quelles qu'elles soient par un simple produit matriciel.

Dans le plan, toute transformation et toute composition de transformation se représentent par une matrice 3 x 3. Une transformation de l'espace nécessite une matrice 4 x 4. Chaque terme de la matrice joue un rôle particulier. Pour le mettre en évidence, passons en revue les matrices associées aux diverses transformations à 3 dimensions.

2.2.1 La translation

La colonne de droite de la matrice homogène correspond au vecteur de translation :

$$\begin{vmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Tx et Ty sont mis à jour par l'attribut de trajectoire et Tz par l'attribut de translation en Z.

2.2.2 Le changement d'échelle

Les termes diagonaux d'une matrice homogène correspondent aux facteurs d'échelle :

$$\begin{vmatrix} Ex & 0 & 0 & 0 \\ 0 & Ey & 0 & 0 \\ 0 & 0 & Ez & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

L'attribut de zoom positionne les paramètres Ex et Ey. Ez vaut toujours 1.

2.2.3 Les rotations

Les matrices homogènes de rotation d'un angle θ autour des axes de coordonnées Rx, Ry et Rz sont respectivement pour les axes X, Y et Z :

$$Rx = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$Ry = \begin{vmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$Rz = \begin{vmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

θ est fourni par l'attribut de rotation.

2.2.4 Combinaison des transformations

La multiplication des matrices de composition n'est pas commutative. C'est à dire que l'ordre d'enchaînement des transformations géométriques a son importance dans le résultat obtenu. Ce résultat est déterminé par l'ordre des attributs. L'utilisateur doit veiller à respecter un certain ordre pour obtenir le résultat escompté.

La combinaison d'une rotation avec une translation conduit à des résultats différents selon que la rotation est appliquée avant ou après la translation.

Cas 1 : Rotation + Translation : l'acteur tourne sur lui-même (son point guide est le centre de la rotation) tout en se déplaçant sur la trajectoire.

Cas 2 : Translation + Rotation : l'acteur tourne autour d'un point. La distance entre le point guide de l'acteur et ce point est déterminée par la valeur de la translation.

Il est toujours difficile pour un utilisateur de comprendre ces mécanismes et d'imaginer par avance le résultat qu'il va obtenir a moins de procéder par essais successifs. Afin de simplifier les manipulations, de nouveaux attributs peuvent être proposés : ces attributs intègrent directement plusieurs attributs de géométrie composés. Ils sont décrits dans le paragraphe traitant des nouveaux attributs composés dans la suite de ce chapitre.

3. MULTIPLICATION DES MODES DE DESCRIPTION

Avec plusieurs modes de description, l'utilisateur choisit celui qui lui convient en fonction de l'animation qu'il doit réaliser. Les différents modes pour les attributs de trajectoire et de cell sont présentés après une description de l'ensemble des attributs et de leur mode standard de description. Ils sont suivis par la description de nouveaux attributs composés.

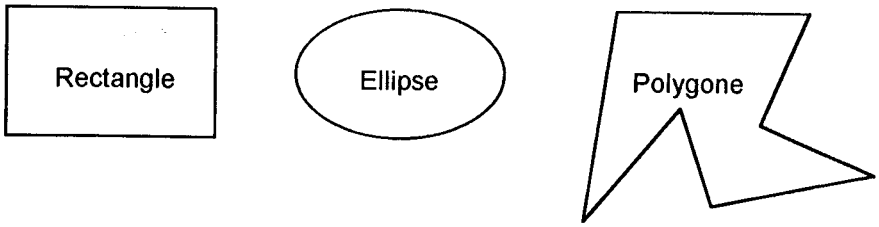
3.1 L'ATTRIBUT TRAJECTOIRE

La description de la trajectoire par points clés limite les possibilités d'animation à des cas peu complexes. Les modes présentés ici permettent de simplifier la description de la trajectoire dans le cas où l'acteur doit suivre une forme particulière.

3.1.1 Description par shape

Une shape est une forme géométrique ou la combinaison de formes géométriques dessinées à l'aide de la palette graphique : rectangle, cercle, ellipse, polygone ou spline, texte vectoriel. Cette forme est sauvegardée dans un fichier. Elle peut être transformée automatiquement par le logiciel en une liste de points définissant la trajectoire. La position de l'acteur sur la trajectoire est déterminée par une abscisse curviligne L(T). La forme est parcourue du premier au dernier point lorsque T varie de 0 à 1.

Exemple de trajectoires en mode shape :



Texte vectoriel

La mise en place d'une telle description nous amène à envisager deux cas :

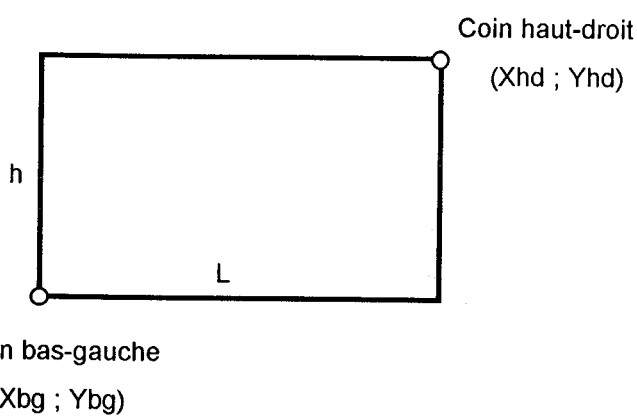
Les caractéristiques de la shape sont conservées au niveau du scénario. Cette solution évite la redondance d'informations mais oblige à recalculer les coordonnées du point courant pour chaque temps lors de l'animation.

La shape est transformée en une liste de points. L'accès à cette liste se fait directement par l'intermédiaire d'un coefficient sur le même principe que pour la description par points clés. Les coordonnées de la shape sont dupliquées dans la liste de points; par contre l'accès à un point ne nécessite pas de calcul et est rapide.

La solution de la liste de points est retenue : les informations concernant la Shape sont stockées dans le domaine d'évolution. Une liste de points est créée. Sa mise à jour est optimisée afin de conserver un nombre de points minimum (suppression des points intermédiaires sur les segments horizontaux ou verticaux). Lors de la sauvegarde du scénario sur disque, seules les informations de la shape sont conservées. La liste de points est reconstruite lors de la restitution du scénario.

La liste de points est générée en fonction de la shape.

3.1.1.1 Cas du rectangle

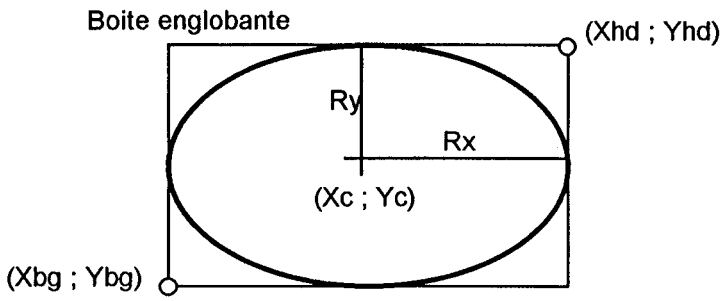


Une shape rectangle est décrite à l'aide de 2 points : l'angle bas-gauche et l'angle haut-droit. La liste de points constitue la forme complète à l'aide de 5 points.

Le coefficient calculé correspond à la distance parcourue sur l'arête du rectangle :

Point	Coefficient
Angle bas-gauche	0
Angle bas-droit	L / (2*h + 2*L)
Angle haut-droit	(L + h) / (2*h + 2*L)
Angle haut-gauche	(2*L + h) / (2*h + 2*L)
Angle bas-gauche	1

3.1.1.2 Cas du cercle ou de l'ellipse



Deux points décrivent la boite englobante de la shape. La première opération consiste à retrouver les coordonnées du centre et celles du rayon :

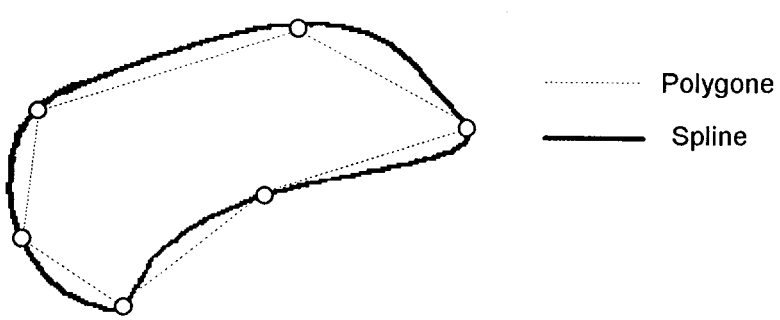
$Xc = (Xhd + Xbg) / 2$ et $Yc = (Yhd + Ybg) / 2$

$Rx = (Xhd - Xbg) / 2$ et $Ry = (Yhd - Ybg) / 2$

La liste de points est créée en prenant un point tous les 10 degrés. Le coefficient est calculé par rapport à un tour complet :

Coefficient = Angle / 360

3.1.1.3 Cas du polygone ou de la spline

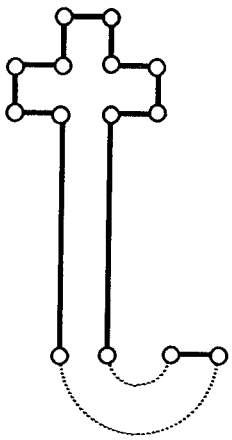


Ces deux morphologies sont décrites à l'aide d'un tableau de points qui constituent les sommets du polygone ou les points de passage de la spline. La liste de points du domaine est créée à l'aide de ces points. Le type d'interpolation linéaire ou lissée permet de distinguer le polygone de la spline.

Dans les 2 cas, le coefficient est calculé en fonction de la distance parcourue depuis le premier point. Cette distance est calculée par cumul des distances linéaires entre deux points de la morphologie :

Coefficient = Distance parcourue / Distance totale

3.1.1.4 Cas du texte vectoriel

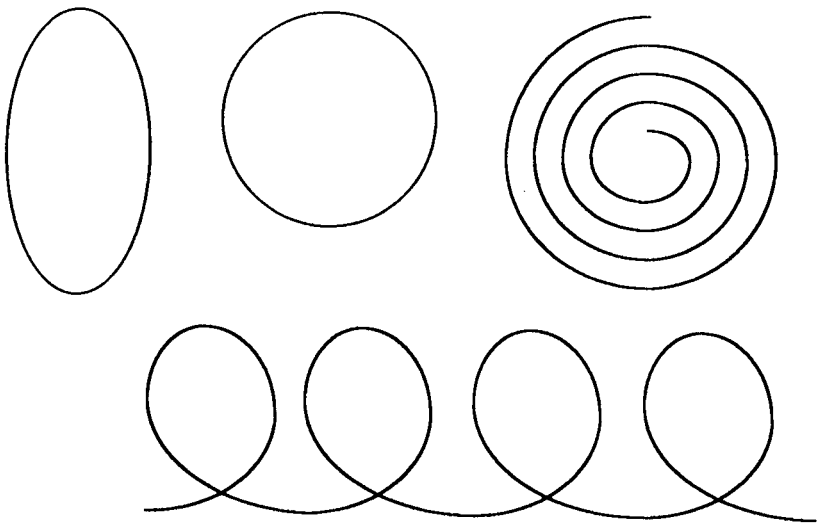


Une shape de type texte est constituée d'un tableau de points. Etant donné la forme complexe d'une telle shape; le nombre de points est en général très important (plus de 50 points par lettre); un algorithme particulier est utilisé. Il réduit le nombre de points notamment dans les segments horizontaux et verticaux. Pour ces segments, seuls les 2 points extrêmes sont conservés dans la liste. La distance point à point est utilisée pour le calcul du coefficient.

3.1.2 Trajectoire circulaire

La base de la trajectoire est un cercle dont l'utilisateur fixe le centre, le rayon et un angle de rotation. En faisant varier cet angle, l'acteur décrit une trajectoire circulaire. Lorsque le rayon varie, la trajectoire suivie est une spirale. Le cercle support de la trajectoire peut également se transformer en ellipse lorsque les rayons en X et Y varient indépendamment. Enfin, le déplacement du cercle peut également être envisagé.

Exemple de trajectoires circulaires :



3.2 L'ATTRIBUT DE CELL

Il est possible d'ajouter un nouveau mode de description de l'attribut de cell afin de gérer la répétition d'une série de cells à l'intérieur d'une séquence : le mode cyclique continu ou discontinu.

Cet effet consiste à répéter une série de cells de longueur fixe. Les cells sont affichés dans l'ordre de leurs numéros. La série de cells est répétée autant de fois que le désire l'animateur. Pour identifier une série, il fixe le numéro du premier cell et celui du dernier.

Le mode continu permet de ne pas afficher le premier cell de la série à partir de la première répétition. Ce mode est utile dans une séquence où le premier et le dernier cell sont identiques. Le mode discontinu permet de repartir au premier cell après le dernier.

Exemple :

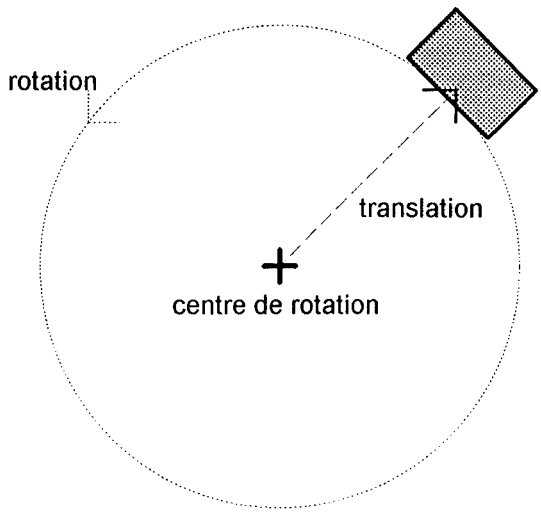
- Premier Cell = 1
- Dernier Cell = 4
- Répétition en mode continu : 1 2 3 4 2 3 4 2 3 4 2 .....
- Répétition en mode discontinu : 1 2 3 4 1 2 3 4 1 2 3 4 1 2 .....

3.3 DE NOUVEAUX ATTRIBUTS COMPOSES

Afin que l'utilisateur ne se pose pas de question lorsqu'il veut combiner des attributs de géométrie; de nouveaux attributs peuvent être proposés tels que « rotation autour d'un point », « rotation de l'acteur sur sa trajectoire », etc.

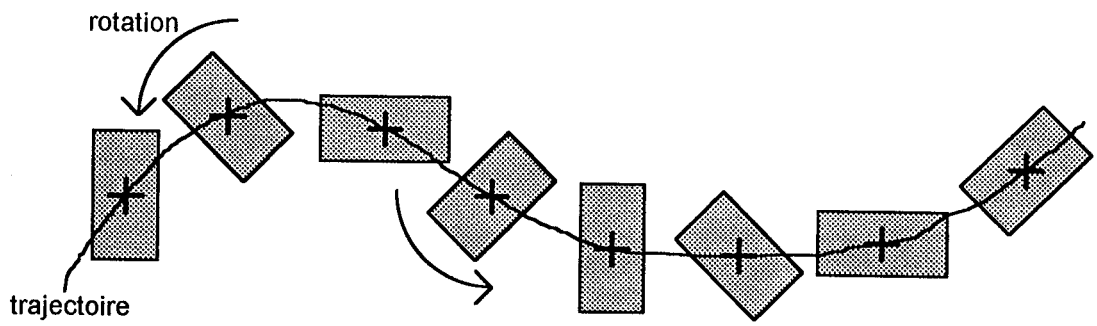
3.3.1 Rotation autour d'un point

Pour cet attribut, les paramètres sont la distance entre le centre de rotation et l'acteur (point fixe) et la valeur de l'angle de rotation.



3.3.2 Rotation sur la trajectoire

La valeur de la translation est déterminée par un couple de coordonnées (X ; Y). La rotation effectuée par l'acteur correspond à l'attribut de rotation classique.



3.3.3 Description des attributs composés

La mise en place de tels attributs peut se faire à l'aide des attributs existants ou bien en concevant de nouveaux attributs.

Solution 1 : utilisation des attributs existants.

Les attributs composés sont créés pour apporter une aide à l'utilisateur entre autres pour garantir l'ordre d'application des différents attributs composants. De tels attributs sont simplement logiques et lorsqu'un de ces attributs doit être ajouté, le système insère automatiquement les attributs réels (trajectoire, rotation) dans l'ordre approprié. Ces derniers sont liés entre eux afin de montrer leur interaction mais la mise à jour des paramètres reste indépendante pour chaque attribut.

Cette solution a l'avantage d'utiliser des attributs déjà existants mais son inconvénient est de proposer une mise à jour indépendante des différents attributs composants.

Solution 2 : de nouveaux attributs composés.

Chaque attribut composé est un attribut à part entière avec ses propres structures de données, panneau de menu et dialogue écran.

Les atouts de cette solution sont que ces attributs sont gérés de la même manière que les autres attributs. Une seule entité est utilisée et les données sont gérées dans un même panneau.

Par contre, elle n'utilise pas les panneaux de menu existants et risque de provoquer une redondance de certains panneaux. De plus, il n'est pas possible de dissocier les transformations ni de combiner plusieurs attributs identiques (trajectoire + rotation + trajectoire par exemple).

4. L'ANIMATION D'ENSEMBLE

Ce nouveau concept est mis en place afin de limiter le nombre des attributs et ainsi diminuer le temps de description de l'animation. Il s'agit de proposer des attributs qui ne sont pas rattachés à un acteur particulier, mais qui s'appliquent à l'ensemble des acteurs du scénario. Par

exemple, si une fenêtre est appliquée de manière identique à tous les acteurs, il est possible d'ajouter un seul attribut de volet pour tout le scénario. Cet attribut s'applique à tous les acteurs.

Exemple :

Cet exemple montre le gain de temps obtenu par rapport à l'Anim où l'utilisateur doit passer tous ses acteurs en revue.

Prenons le cas d'une animation de 3 acteurs, chacun d'eux ayant sa propre trajectoire. L'utilisateur désire ajouter une translation à l'ensemble des acteurs pour que l'animation complète traverse l'écran de gauche à droite.

Dans l'Anim, l'utilisateur définit les positions des 3 acteurs puis il ajoute la translation générale sur chaque étape des 3 trajectoires ce qui implique qu'il doit ajouter les coordonnées de chaque étape des trajectoires avec les coordonnées de la translation générale. La modification d'une étape est très fastidieuse et l'utilisateur n'a pas droit à l'erreur puisqu'il faut corriger les translations des 3 acteurs sans garantie que le résultat final soit correct.

Avec le DVE-Composer, l'utilisateur définit également les 3 trajectoires des acteurs puis il ajoute un attribut de trajectoire à l'acteur général. Cet attribut s'applique par composition avec les attributs de géométrie de tous les acteurs du scénario. Il suffit alors de placer les étapes de la trajectoire générale sans avoir à toucher aux 3 acteurs. Dans ce cas la modification d'une étape est immédiate et son effet est répercuté sur les 3 acteurs en même temps.

4.1 MISE EN PLACE

Deux solutions sont envisageables pour mettre en place une animation d'ensemble.

La première solution consiste à créer une arborescence d'acteurs où chaque noeud représente une scène. Les attributs placés au niveau d'un noeud s'appliquent à l'ensemble des acteurs sous le noeud. Cette solution nécessite la mise en place d'un moyen de dialogue permettant une gestion aisée de l'arborescence. Elle permet de regrouper certains acteurs dans des scènes.

La seconde solution consiste à ajouter un acteur général au scénario. Cet acteur regroupe les attributs généraux. Le moyen de dialogue pour la gestion de l'acteur général est identique à celui d'un autre acteur. La liste des attributs est gérée également de la même façon. L'inconvénient de cette solution est qu'un attribut général s'applique obligatoirement à l'ensemble des acteurs du scénario.

Cette dernière solution est tout de même retenue car la mise en place de la première solution risque de conduire à une interface homme-machine complexe. L'expérience a montré que seuls les dialogues simples sont acceptés par les graphistes. L'acteur général est systématiquement ajouté au scénario. Il ne peut pas être supprimé mais peut ne contenir aucun attribut. A cet acteur peuvent être ajoutés les attributs permettant la gestion du scénario. Ces attributs s'appliquent soit à l'ensemble des acteurs comme la profondeur soit au niveau du scénario sans effet particulier sur un acteur comme la couleur de fond ou la distance focale.

4.2 COMPOSITION DES ATTRIBUTS GENERAUX

La composition des attributs de géométrie généraux avec les attributs de la classe géométrie des acteurs est effectuée au niveau de la matrice de transformation de l'acteur lors du calcul de l'animation.

La première solution est d'obtenir une matrice générale qui sera multipliée avec la matrice de chaque acteur.

Avantages :

L'ordre d'évaluation des différents acteurs n'a pas d'importance à condition que la matrice générale soit combinée à la fin des calculs de toutes les matrices d'acteurs.

Inconvénients :

Certains acteurs ne sont pas capables d'interpréter certaines transformations géométriques (pas de rotation sur un VPM).

Dans la matrice de composition, il n'est pas possible de dissocier la rotation du changement d'échelle car les mêmes coefficients de la matrice sont utilisés pour ces deux transformations.

Une autre solution consiste à évaluer l'acteur général en dernier. Les transformations sont appliquées sur chaque acteur par composition de la transformation dans la matrice de l'acteur si celui-ci sait gérer la transformation. Par exemple, l'attribut de rotation générale invoque tous les acteurs et les compose avec une matrice de rotation si l'acteur sait manipuler les rotations (acteur placé sur un plan DVE-Layer).

Avantages :

Lorsqu'une transformation est appliquée, lorsqu'elle est prise en compte, elle est appliquée correctement (avec la première solution, une rotation sur un VPM a un effet sur le zoom de l'acteur).

L'évolution de l'application est facilitée : dans les développements futurs, il est prévu de mettre en place les transformations dans un espace 3D (rotations notamment) sur des plans VPM. Ces transformations ne seront pas effectuées en temps réel et les acteurs pourront alors les prendre en compte.

Inconvénients :

Certains acteurs ne réagissent pas à un attribut général.

Cette solution impose un ordonnancement des calculs (attribut général évalué en dernier).

Cette seconde solution est retenue : l'acteur général est systématiquement évalué en dernier lieu. Les fonctions de calcul des acteurs sont capables de dissocier les attributs de l'acteur de ceux de l'acteur général.

4.3 LES ATTRIBUTS GENERAUX

Certains attributs ne peuvent être rattachés qu'à l'acteur général. Il s'agit de l'attribut de couleur de fond, de l'attribut de profondeur, de l'attribut de focale. Tous les attributs de la classe géométrie peuvent être affectés à l'acteur général.

L'attribut de pondération est affecté à l'acteur général sous la dénomination d'attribut de transparence. L'attribut de fenêtre générale est appelé volet.

Il est possible de profiter de la mise en place de l'animation d'ensemble pour proposer de nouveaux attributs généraux permettant de réaliser des effets comme les fondus enchaînés ou permettant le calcul automatique des priorités des acteurs (profondeur automatique).

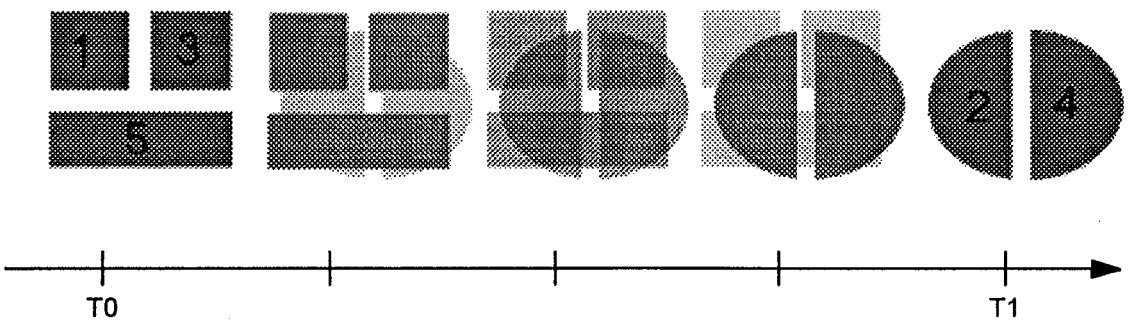
4.3.1 Les fondus

La pondération agit au niveau d'un acteur. Pour générer des effets de fondus au noir ou de fondus enchaînés entre plusieurs acteurs, l'utilisateur doit décrire la pondération de chaque acteur séparément puis les synchroniser.

L'acteur général peut contenir un attribut pour les effets de fondu. Cet attribut détermine à un instant donné parmi une liste d'acteurs du scénario quels sont ceux qui sont visibles et ceux qui sont invisibles.

Exemple : fondu enchaîné entre 5 acteurs

Au temps T0, les acteurs 1, 3 et 5 ont une pondération de 128. Ils sont visibles. Les acteurs 2 et 4 ont une pondération de 0. Ils sont invisibles. Au temps T1, les pondérations sont inversées.



Si la liste des acteurs invisibles à T1 est vide, l'effet réalisé est un fondu au noir. Inversement, l'apparition d'une série d'acteurs est obtenue avec une liste d'acteurs visibles à T0 vide.

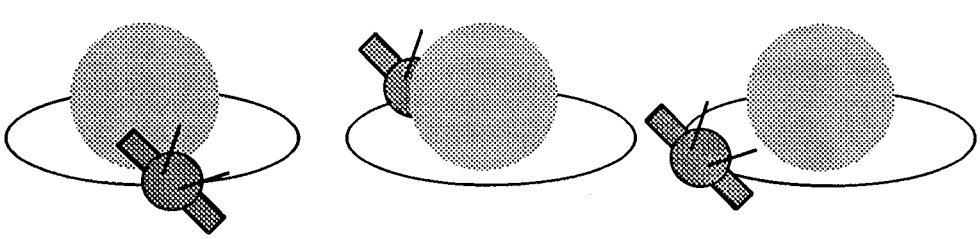
4.3.2 L'attribut de profondeur automatique

L'attribut de profondeur détermine la priorité de l'ordre d'affichage des acteurs. Dans le cas où un acteur tourne autour d'un autre, l'utilisateur doit positionner les priorités en fonction de la rotation de l'acteur mobile, pour qu'il passe derrière celui qui est immobile.

Il est possible de profiter de l'espace 3D pour que le calcul des profondeurs se fasse automatiquement en fonction de la translation en Z.

Cet attribut détermine automatiquement la priorité d'un acteur en fonction de sa position sur l'axe des Z. Le tableau des priorités est généré automatiquement en triant les acteurs par valeur de leur translation en Z décroissante : plus la translation est grande, plus il est à l'avant de la scène et plus sa priorité est faible.

L'exemple proposé présente un acteur immobile et un acteur effectuant une rotation autour de l'axe des Y après avoir été translaté selon l'axe des X.



Pour les angles compris entre 0 et 180°, la translation en Z obtenue par calcul des autres transformations est négative pour l'acteur mobile. Celle de l'autre acteur reste à 0. Ce dernier est devant. Pour les angles supérieurs à 180 et jusqu'à 360°, la translation en Z de l'acteur mobile étant positive, celui-ci passe devant l'autre acteur.

L'attribut de profondeur automatique peut être invoqué au moment du calcul de l'animation lorsque l'attribut de profondeur est supprimé de la liste des attributs de l'acteur général. L'utilisateur n'a donc pas d'opération supplémentaire à effectuer.

Il est également possible de modifier l'attribut de profondeur manuelle permettant de passer certains acteurs en profondeur automatique. Cette option risque de compliquer la gestion et l'utilisation de l'acteur de profondeur manuelle.

5. L'ASSERVISSEMENT DES ATTRIBUTS

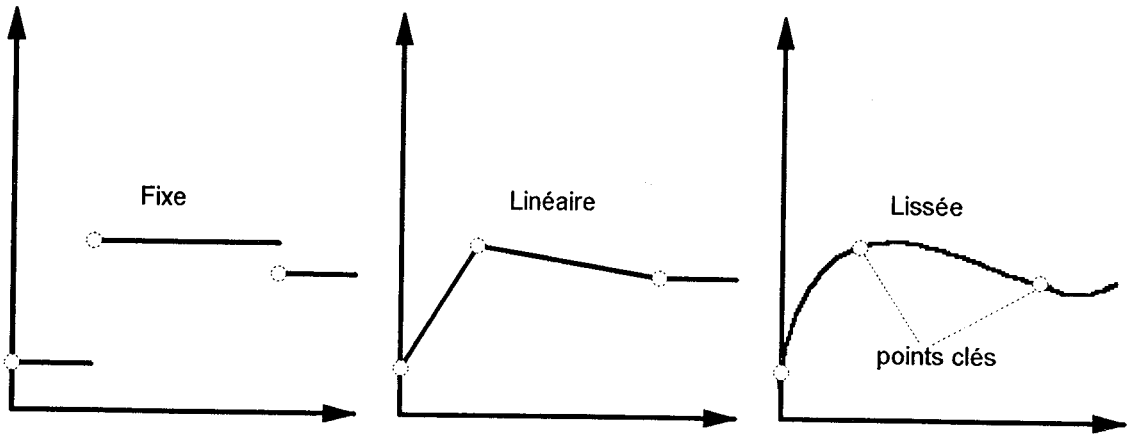
5.1 UNE NOUVELLE APPROCHE DE L'ANIMATION

Il existe deux méthodes de description des attributs. Nous nous proposons d'étudier ces deux méthodes ainsi que leurs variantes afin de choisir celle qui est adaptée à la mise en place de l'asservissement des attributs.

5.1.1 Description par intervalle d'interpolation

L'utilisateur saisit des points clés définissant des étapes à des temps précis. Le système calcule les valeurs intermédiaires par interpolation entre 2 points clés. Il existe plusieurs types de fonctions d'interpolation :

- fixe : la valeur du paramètre ne change que sur un point clé;
- linéaire : la valeur du paramètre évolue linéairement (le long d'une droite) entre 2 points clés;
- lissée : les valeurs calculées appartiennent à une courbe appelée Spline. Cette courbe est définie par les points de passage (points clés).



Cette méthode est utilisée pour la description du scénario de l'Anim et celui du Sequencer. Il possède l'avantage d'être simple à mettre en oeuvre et d'être facile à utiliser. Par contre les possibilités d'animation sont limitées.

La fonction d'évolution d'un attribut particulier a ses propres caractéristiques et il faut prévoir une adaptation pour lier l'évolution de deux attributs différents (asservissement).

La vitesse d'évolution de l'attribut n'est pas très bien maîtrisée. Dans certains cas, une vitesse trop grande à l'approche d'un point clé crée une déformation sur la courbe d'évolution de l'attribut. Ce phénomène a été constaté avec l'attribut trajectoire du Sequencer. La déformation est visible sur la trajectoire : l'acteur effectue une boucle autour du point clé pour compenser le déplacement à grande vitesse dans un laps de temps fixe.

5.1.1.1 Intégration dans le scénario

La fonction d'évolution est du type F(T) = V où T est le temps compris entre 0 et la durée du scénario exprimé en numéro de trame (50 trames par secondes). V est la valeur prise par l'attribut. Cette valeur est interpolée entre deux étapes.

5.1.1.2 Cas particulier de la trajectoire.

La trajectoire est construite au fur et à mesure de la définition des points clés. Sa modification est aisée par déplacement d'un des points clés. De plus, le résultat est visualisé immédiatement; la trajectoire étant matérialisée par une courbe dessinée à l'écran.

Par contre, il n'est pas possible de revenir en arrière sur la trajectoire. Le sens de parcours est déterminé par la position chronologique des points clés.

Il est difficile de maîtriser le tracé d'une trajectoire selon une forme géométrique particulière (polygone, cercle, ellipse, etc.). Dans le cas simple d'un rectangle, l'utilisateur peut placer un point clé sur chacun des 4 sommets de la forme. Lorsque la forme devient plus complexe, cette méthode est beaucoup plus lourde. On peut imaginer que l'acteur suive le contour d'un texte.

Avec le mode de description par points clés, la forme de la trajectoire ne peut pas évoluer dans le temps. Il n'est pas possible de déterminer une forme à parcourir au début du scénario puis de la faire évoluer par interpolation jusqu'à une autre forme.

La notion de vitesse (nombre de pixels parcourus par trame) n'est pas utilisable pour la gestion des amortis qui sont pratiquement impossible à réaliser.

5.1.2 Description par fonction d'animation

Une fonction d'animation décrit l'évolution d'un attribut, appartenant à une entité quelconque, en fonction du temps [MAR_77]. Elle se compose de :

- un domaine d'évolution;
- une fonction d'évolution à l'intérieur de ce domaine.

Un domaine est défini à l'aide d'un ensemble fini et ordonné de valeurs v_i :

$$D = \{v_0, v_1, ..., v_n\} ; n \in N$$

D définit en fait un ensemble continu de valeurs :

- $v = v_i$ si $i \in N$;
- v = valeur obtenue par interpolation des v_i si $i \in [0, n]$;

- v = valeur obtenue par extrapolation des v_i si i $\notin$ [0, n].

La fonction d'évolution associe un indice d'évolution k à chaque instant t. Elle est indépendante du domaine auquel elle s'applique. De ce fait, elle peut être définie de manière relative en adoptant les conventions suivantes :

- k = 0 désigne la première valeur v_0 du domaine D;
- k = 1 désigne la dernière valeur v_n du domaine D;
- 0 < k < 1 désigne la valeur $v_{k.n}$.

Cette solution permet d'obtenir un coefficient d'évolution au format identique quel que soit le type d'attribut. Par contre elle implique la gestion de deux structures différentes pour le coefficient et le domaine d'évolution. Cet inconvénient peut être limité dans la mesure où les fonctions de gestion du coefficient sont identiques pour tous les attributs.

Exemple : l'attribut de trajectoire

La liste des points clés constitue le domaine d'évolution. Le coefficient associé à chaque point est calculé en fonction de la distance parcourue depuis le premier point de la trajectoire. La variation du coefficient d'évolution au cours du temps permet de déplacer l'acteur sur sa trajectoire. Le principe de base est identique à celui de l'Anim, excepté que ce dernier utilise une liste de points unique : à un temps donné correspond un point de la trajectoire.

La séparation du coefficient dans une liste et des coordonnées dans une autre liste permet de nouveaux effets : la mise à jour de la vitesse d'évolution du coefficient ne remet pas en cause le calcul des coordonnées. Il est possible de faire varier la vitesse du déplacement sur la trajectoire sans que cette dernière soit modifiée pour autant. En effet, dans le Sequencer, lorsque l'acteur arrive trop vite sur un point, la trajectoire interpolée décrit une boucle autour du point pour que l'acteur soit positionné sur le point clé au temps où il a été décrit.

5.1.2.1 Intégration dans le scénario

Le domaine d'évolution décrit les valeurs que peut prendre l'attribut au cours du scénario.

Le coefficient d'évolution est une fonction du type F(T) = C où T est le temps du scénario et C un coefficient calculé relativement aux valeurs du domaine. C est utilisé pour obtenir la valeur de l'attribut dans le domaine.

Il est possible d'envisager deux variantes pour l'interprétation des domaines et coefficients d'évolution.

Variante numéro 1 :

Le domaine est une fonction temporelle associant un temps à un domaine courant D : F(T) = D. Le coefficient est une fonction du type F(T) = C. La valeur est calculée à partir du domaine à l'aide de la fonction F(C, D) = V.

Le domaine d'évolution décrit la plage de variation de l'attribut à un instant donné. D représente un couple de valeurs (min ; max) indiquant la valeur minimum prise par l'attribut ainsi que la valeur maximum.

Le coefficient C est calculé en fonction de ce couple et varie entre 0 et 1 :

- C = 0 correspond à la valeur min;
- C = 1 correspond à la valeur max.

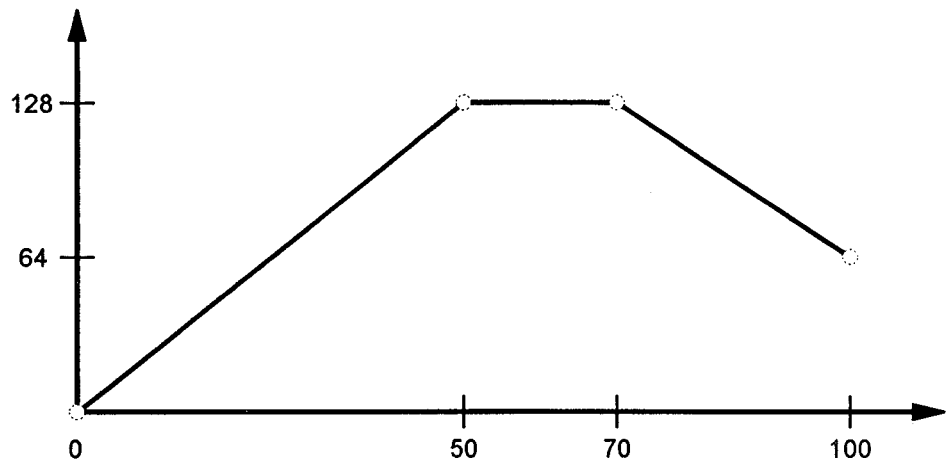
Une fonction de calcul permet de retrouver la valeur de l'attribut en fonction du coefficient et du domaine :

F(C, D) = V avec V = C * (max - min) + min

C et D sont les coefficients et domaines courants c'est à dire extraits au temps courant.

Exemple :

Soit un scénario de 2 secondes (100 trames); soit un attribut de pondération variant selon le schéma suivant :



Le domaine décrit le couple (min ; max) de cet attribut soit (0 ; 128). La variation de l'attribut est définie par l'ensemble des couples (temps ; coefficient d'évolution) soit : { (0 ; 0), (50 ; 1), (70 ; 1), (100 ; 0.5) }. La fonction de calcul permet de retrouver la valeur de l'attribut :

pond = 0 * (128 - 0) + 0 = 0 au temps 0;

pond = 1 * (128 - 0) + 0 = 128 aux temps 50 et 70;

pond = 0.5 * (128 - 0) + 0 = 64 au temps 100.

Variante numéro 2 :

Le domaine décrit l'ensemble des valeurs que peut prendre l'attribut au cours du scénario. Chaque valeur est accessible via un coefficient qui peut être indépendant du temps.

La valeur est lue directement dans le domaine et sa lecture ne nécessite pas de fonction de calcul : F(C) = V.

Le coefficient est utilisé comme une indirection permettant d'obtenir une valeur pour l'attribut à partir du temps : F(T) = C.

Un des moyens de mise en oeuvre le plus simple est de calculer le coefficient en fonction du temps et de la durée du scénario :

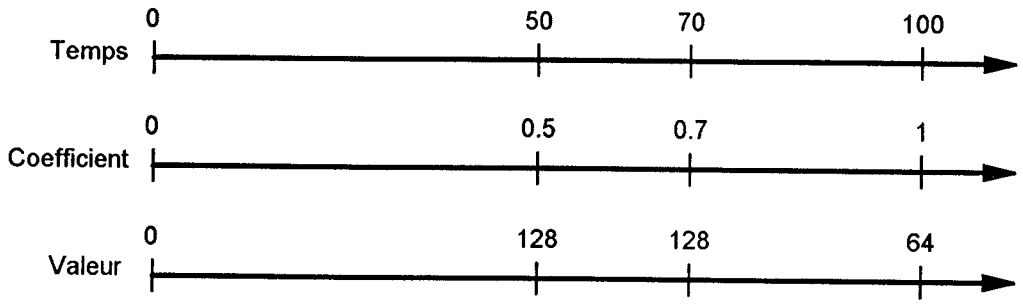
- C = 0 correspond au temps 0
- C = 1 correspond à la durée du scénario
- C = T / Duree_Scenario où T = temps courant

Cette méthode permet la mise à jour indépendante du coefficient et du domaine :

- lors de la création d'une étape, un coefficient est calculé et un couple (T ; C) est stocké. Dans le même temps le couple (C ; V) est également stocké;
- la mise à jour du domaine peut être effectuée indépendamment du temps : un couple (C ; V) peut être constitué pour n'importe quelle valeur de C entre 0 et 1.

Exemple :

Reprenons l'exemple précédent : la correspondance temps-coefficient-valeur est donnée par le schéma suivant :



Le coefficient d'évolution est composé de l'ensemble des couples (temps ; coefficient) soit : { (0 ; 0), (50 ; 0.5), (70 ; 0.7), (100 ; 1) }. Le domaine est composé de l'ensemble des couples (coefficient ; valeur) soit : { (0 ; 0), (0.5; 128), (0.7; 128), (1 ; 64) }.

5.1.3 La solution retenue.

Il est difficile de donner la méthode idéale. Selon l'attribut concerné, une méthode sera plus efficace ou bien moins rapide. Pour les attributs dont la description est complexe, la fonction d'animation sera utilisée en particulier pour rendre indépendants l'évolution de l'attribut et la vitesse d'évolution (cas de la trajectoire : la trajectoire peut être lissée et la vitesse d'évolution peut être linéaire). La description par points clés est utilisée dès qu'elle simplifie les calculs.

En définitive, la solution retenue est de systématiser la fonction d'animation en décrivant un domaine d'évolution et un coefficient d'évolution, ceci afin de garantir une compatibilité entre tous les attributs.

La solution idéale consiste à regrouper les attributs en deux grands groupes selon leur type : les attributs de type intervalle et les autres (que l'on pourra nommer attributs de type temporel). Pour les premiers, la variante numéro 1 est utilisée alors que la variante numéro 2 est retenue pour les seconds. Cette distinction oblige à gérer deux modes de fonctionnement mais permet de limiter le nombre des attributs utilisant la variante numéro 2. En effet, cette variante permet d'obtenir un coefficient variant linéairement entre 0 et 1 et qui n'est pas fonction de la valeur de l'attribut.

Concrètement, la fonction d'évolution est matérialisée par une liste temporelle de valeurs. Le principe est le même que pour la description par points clés. L'utilisateur fixe une valeur de l'attribut à un temps donné. Le système convertit la valeur en fonction du type d'attribut en un coefficient et crée une étape dans la liste. Les valeurs des coefficients sont interpolées entre les étapes. Le domaine d'évolution est mis à jour automatiquement lorsqu'une étape est créée mais l'utilisateur peut s'il le désire modifier le domaine en accédant les valeurs directement via le coefficient. Le type d'interpolation lissée permet à l'utilisateur d'agir s'il le désire sur la vitesse de variation du coefficient; Entre autres pour les effets d'amorti (vitesse nulle sur une étape).

5.2 INTRODUCTION DE L'ASSERVISSEMENT

La notion d'asservissement désigne le fait de lier l'évolution d'un attribut avec celle d'un autre. L'attribut servant de fonction d'asservissement est appelé attribut maître.

L'asservissement d'un attribut par un attribut maître consiste à faire évoluer le coefficient de l'attribut asservi en fonction de celui de l'attribut identifié en tant que fonction d'asservissement.

Exemple :

Un acteur possède les attributs de cell et de trajectoire. La trajectoire est parcourue complètement du premier au dernier point lorsque le coefficient de l'attribut trajectoire varie de 0 à 1. Cet acteur possède 10 cells différents. Les numéros de ces cells évoluent entre 1 et 10. L'asservissement de l'attribut cell par l'attribut trajectoire permet :

- de faire évoluer le numéro de cell entre 1 et 10 lorsque le coefficient de l'attribut de trajectoire évolue entre 0 et 1;
- de faire évoluer le numéro de cell de 10 vers 1 lorsque le coefficient de l'attribut de trajectoire évolue entre 0 et 1.

Cette opération est rendue très simple lorsque les attributs possèdent un coefficient qui peut être décrit de manière relative et prenant des valeurs comprises entre 0 et 1. Lorsque ce n'est pas le cas, il est possible par une transformation de ramener le coefficient dans cet intervalle.

Pour compléter la gamme des effets proposés, une opération supplémentaire est proposée pour modifier l'évolution de l'attribut asservi :

- évolution des coefficients dans le même sens;
- évolution dans le sens inverse;
- réduction de l'intervalle des valeurs que peut prendre l'attribut asservi;
- modulo à l'intérieur d'un intervalle de valeurs.

Exemple de combinaisons qui peuvent être proposées :

Les attributs A et B évoluent tous les 2 entre les valeurs 1 et 10. Le tableau ci-dessous présente l'évolution de B (asservi par A) lorsque A varie entre 1 et 10 :

Evolution de A	1	2	3	4	5	6	7	8	9	10
Evolution de B identique	1	2	3	4	5	6	7	8	9	10
Evolution de B inversée	10	9	8	7	6	5	4	3	2	1
Evolution de B réduite dans un intervalle Borne min = 3 ; Borne max = 7	3	3	4	4	5	5	6	6	7	7
Evolution de B réduite et inversée	7	7	6	6	5	5	4	4	3	3
Evolution de B réduite avec un modulo Borne min = 3 ; Borne max = 5	3	4	5	3	4	5	3	4	5	3

Les applications de l'asservissement sont nombreuses :

- évolution de l'attribut d'un acteur identique à celle du même attribut d'un autre acteur (variation des cells de 2 acteurs identiques, même facteur de zoom appliqué à plusieurs acteurs, effet de fondu enchaîné entre 2 acteurs);
- évolution d'un attribut d'un acteur identique à celle d'un autre attribut du même acteur (numéros de cell en fonction de la position sur la trajectoire : cas de la roue qui tourne quand elle avance);
- évolution de l'attribut d'un acteur liée à celle d'un attribut de type différent d'un autre acteur.

5.3 CALCUL DE L'ASSERVISSEMENT

Le mode de calcul de l'asservissement est largement lié au mode de description de l'attribut. Lorsque l'attribut est décrit par intervalle d'interpolation, la valeur de l'attribut maître doit être transformée au format de l'attribut asservi. Pour un attribut géré par fonction d'animation, l'asservissement consiste à substituer le coefficient d'évolution par un nouveau coefficient. Ce dernier est calculé à partir de l'attribut servant de fonction d'asservissement.

5.3.1 L'asservissement des attributs décrits par intervalle d'interpolation

La première solution consiste à transformer la valeur de la fonction d'asservissement en une valeur comprise entre 0 et 1. N'importe quel attribut peut alors utiliser cette valeur en la transformant à son format.

Exemple :

Soit un attribut de cell avec des numéros de cell compris entre 1 et 12; et un attribut de pondération variant entre 0 et 128. Une règle de trois permet de transformer le coefficient du cell ou celui de la pondération en une valeur entre 0 et 1 : valeur 0 pour le numéro de cell 1 ou pour la pondération 0 et valeur 1 pour le cell 12 ou pour la pondération 128.

La seconde solution propose une fonction de transformation de la valeur pour chaque couple attribut asservi / fonction d'asservissement. Cette fonction est adaptée au type des attributs du couple pour transcoder une valeur en une autre qui soit interprétable par l'attribut asservi.

Exemple :

Lorsque l'attribut de cell est asservi par celui de pondération, une transformation permet de calculer la valeur du cell en fonction de celle de la pondération : cell 1 pour la pondération 0 et cell 12 pour la pondération 128.

Examinons la liste des attributs du Composer. Pour chacun d'eux on donne pour les deux solutions les paramètres permettant de les utiliser comme fonction d'asservissement.

5.3.1.1 Solution numéro 1

La valeur de l'attribut doit être ramenée à une valeur entre 0 et 1. Pour les attributs dont la valeur est comprise entre deux bornes, cette transformation s'effectue à l'aide d'une règle de trois :

Attribut	Minimum	Maximum
Translation Z	-2000	500
Rotations	-90*360	90*360
Pondération	0	128
Defocus (flou)	0	100
Focale	500	32767
Cell	min de la séquence	max de la séquence

Pour d'autres attributs, la transformation n'est pas immédiate et doit être étudiée au cas par cas. Ces attributs sont décrits par des valeurs multiples :

- trajectoire : les 2 valeurs représentent le couple de coordonnées (X ; Y) de chaque point de la trajectoire. X et Y sont décrits entre -2000 et +2000, relativement au centre de l'écran;
- zoom : 2 valeurs décrivent le facteur de zoom : 1 valeur pour le facteur en X et 1 valeur pour le facteur en Y. Ces 2 valeurs évoluent entre -20 et +20;
- volet : 4 valeurs décrivent les coordonnées de la fenêtre : abscisses des coins bas-gauche et haut-droit et ordonnées des coins bas-gauche et haut-droit. Les abscisses varient entre 0 et la largeur de l'écran vidéo. Les ordonnées varient entre 0 et la hauteur de l'écran vidéo;
- couleur de fond : les 8 valeurs représentent les composantes des 2 couleurs : pondération et 3 composantes des couleurs du haut et du bas;

Une solution doit être trouvée pour regrouper ces valeurs en une seule. Ensuite, le principe de conversion des attributs de la première liste peut être appliqué, toutes ces valeurs étant bornées.

Pour les attributs restant, leur description est trop complexe pour que l'attribut soit utilisé en tant que fonction d'asservissement. Le système ne devra donc pas proposer d'identifier ces attributs en tant que fonction d'asservissement (Profondeur, etc.).

5.3.1.2 Solution numéro 2

La solution numéro 2 peut être résumée par un tableau indiquant les couples attribut asservi / fonction d'asservissement :

- cas 1 : l'asservissement d'un attribut par le même attribut d'un autre acteur est immédiat. Les valeurs sont remplacées sans transformation;
- cas 2 : pour tous les attributs de la première liste de la solution numéro 1, la conversion est effectuée à l'aide d'une règle de trois;
- cas 3 : Combinaison non autorisée. Un effet de flou automatique est géré par les cartes DVE-Layer. Le flou est calculé automatiquement par le logiciel de la carte en fonction du facteur de zoom et de l'angle de la rotation;
- cas 4 : Combinaison non autorisée. Attribut général utilisé une seule fois;
- cas 5 : Combinaison non autorisée. L'effet réalisé n'est pas intéressant.

Asservi par :	Cell	Zoom	Traj	Trans Z	Rot	Foc	Def	Pond	Volet	Fond
Cell	1		2	2	2	2	2	2		
Zoom		1		5			3			
Trajectoire			1	2	2	2	2	2		
Translation Z	2	5	2	1	2	2	2	2		
Rotation	2		2	2	1	2	3	2		
Focale	2		2	2	2	4	2	2		
Defocus	2	3	2	2	3	2	1	2		
Pondération	2		2	2	2	2	2	1		
Volet									1	5
Couleur de fond									5	4

5.3.2 L'asservissement des attributs décrits par fonction d'animation

L'asservissement des attributs agit sur le coefficient d'évolution. Les coefficients de tous les attributs sont identiques et la substitution d'un coefficient par un autre est immédiate.

La mise à jour du domaine de l'attribut asservi indépendamment du coefficient permet de préciser des valeurs propres à l'attribut asservi. L'asservissement sert alors dans ce cas à lier l'évolution de deux attributs (fondu enchaîné, points de rencontre sur une trajectoire, etc.).

5.3.3 Choix d'une solution

Le mode de description des attributs est primordial quand au choix d'un mode de calcul. En effet, pour les attributs décrits par fonction d'animation, la réalisation est immédiate et ne nécessite pas de calcul. De plus cette méthode permet l'asservissement de n'importe quel attribut par n'importe quel autre sans restriction si ce n'est le résultat visuel obtenu. Le choix peut être laissé à l'utilisateur.

La solution numéro 1 nécessite une transformation inverse pour calculer le coefficient de l'attribut en fonction de la valeur affectée entre 0 et 1. Pour les attributs de la première liste, l'opération consiste en une autre règle de trois.

Pour les autres attributs des choix doivent être faits :

- attribut ne pouvant pas être asservi;
- affectation de la valeur transformée à toutes les valeurs du coefficient;
- affectation de la valeur transformée à une seule des valeurs du coefficient choisie par l'utilisateur ou arbitrairement.

Cette solution permet d'associer un nombre important d'attributs quel que soit leur type. L'utilisateur a ainsi le choix des associations qu'il peut faire et peut conserver celles qui sont intéressantes au niveau des effets vidéo réalisés.

La solution numéro 2 n'est pas du tout intéressante dans la mesure où elle se rapproche de la première, en permettant éventuellement plus de combinaisons mais en augmentant considérablement le volume des modules à développer.

La solution retenue est de décrire les attributs par fonction d'animation (cf. Nouvelle approche de l'animation) avec la solution correspondante pour l'asservissement.

5.4 MISE EN PLACE DE L'ASSERVISSEMENT

L'utilisateur doit désigner les attributs maîtres qu'il désire utiliser en tant que fonction d'asservissement. Le nom donné à la fonction est unique et valable pour tout le scénario. Plusieurs points seront à régler pour asservir un attribut et utiliser les fonctions d'asservissement :

- Lorsqu'un attribut est asservi, l'est-il pour toute la durée du scénario ou bien seulement sur une portion que l'utilisateur désigne ?
- Un attribut peut-il être asservi par plusieurs fonctions dans un même scénario ?
- Quel est l'impact de la suppression d'une fonction d'asservissement ?
- Il ne doit pas y avoir plusieurs fonctions d'asservissement portant le même nom.

5.4.1 Application de l'asservissement

Lorsque l'attribut est asservi sur toute la durée du scénario, l'utilisateur indique la fonction servant à l'asservissement de l'attribut. La même fonction est utilisée pour toute la durée du scénario.

Si l'attribut est asservi sur une ou plusieurs plages de temps du scénario, il faut un mécanisme comme le chronogramme pour indiquer les limites des plages de temps : un type d'évolution particulier peut être utilisé pour la liste de coefficient; le type asservi. Au niveau du chronogramme, ce type est géré de la même façon que les types fixe, linéaire et lissé. Dans la liste du coefficient, lorsque l'évolution est du type asservi, le module de calcul de l'animation utilise la valeur du coefficient de la fonction d'asservissement à la place de celle lue dans la liste du coefficient asservi.

5.4.2 Une seule fonction d'asservissement

Plusieurs fonctions différentes peuvent être utilisées sur les différentes plages mais afin de ne pas compliquer le mécanisme de description de l'asservissement, il faut limiter le nombre de fonctions utilisées par attribut à une seule. Ainsi l'utilisateur indique la fonction d'asservissement de l'attribut et celle-ci est valable pour toute la durée du scénario même si elle ne s'applique que par plages de temps.

5.4.3 Suppression d'une fonction d'asservissement

Plusieurs opérations peuvent entraîner la disparition d'une fonction d'asservissement :

- suppression de la fonction d'asservissement;
- suppression d'un attribut désigné en tant que fonction d'asservissement;
- suppression d'un acteur contenant un ou plusieurs attributs désignés en tant que fonction d'asservissement.

Il est trop contraignant pour l'utilisateur de mettre en place un mécanisme interdisant la suppression d'un attribut ou d'un acteur aussi est-il préférable d'envisager une autre solution comme la mise en place d'une fonction d'asservissement standard. Cette fonction retourne systématiquement un coefficient à 0 lorsque l'attribut est asservi et que la fonction d'asservissement correspondante n'est pas trouvée dans le scénario.

5.4.4 Gestion des doublons

Il est possible de s'assurer qu'une fonction d'asservissement n'existe pas déjà lors de la saisie d'une nouvelle fonction et d'interdire les doublons. Par contre lorsqu'un acteur est chargé à partir d'un fichier sur disque, un de ses attributs peut être une fonction d'asservissement du même nom que celle d'un des attributs déjà présents dans le scénario.

Quatre solutions sont envisagées :

- Interdiction de charger l'acteur : cette solution est trop contraignante pour l'utilisateur.
- L'attribut contenant la fonction d'asservissement en double n'est pas chargé : l'utilisateur va être obligé de rajouter manuellement l'attribut manquant et de définir ses étapes.
- Mise à jour automatique du nom de la fonction d'asservissement en la post fixant par un numéro. Il faut également modifier le nom de la fonction d'asservissement des attributs qui utilisent la fonction renommée.
- Suppression de la fonction d'asservissement en double : l'attribut utilisant cette fonction va être asservi par la fonction déjà existante dans le scénario.

La solution la plus simple à mettre en oeuvre est la suppression de la fonction d'asservissement en double. Les deux premières solutions sont écartées d'office car elles sont contraignantes pour l'utilisateur. La troisième solution pourra être mise en oeuvre lors d'une évolution future si l'utilisation montre des limites de la solution retenue.

6. INTEGRATION

De nouveaux concepts ont été développés dans ce chapitre. Nous avons déjà pu constater que le mode de description des attributs a une forte influence sur le choix de la méthode utilisée pour l'asservissement des attributs.

En fait, ces nouvelles notions s'intègrent dans le scénario avec la volonté de proposer à l'utilisateur de nouveaux modes de description de son animation :

- la composition des attributs a permis entre autres d'élargir la gamme des attributs de géométrie et surtout d'intégrer les transformations dans un espace 3D;
- la multiplication des méthodes de description des attributs a contribué à l'élargissement de la panoplie des effets obtenus;
- l'animation d'ensemble a été rendue possible grâce à la composition des attributs;
- un nouveau mode de description des attributs, la fonction d'animation, a été proposée pour une mise à jour séparée des listes de coefficient et de domaine;
- l'asservissement des attributs permet le lien entre certains attributs du scénario et bénéficie des avantages de la mise à jour du domaine.

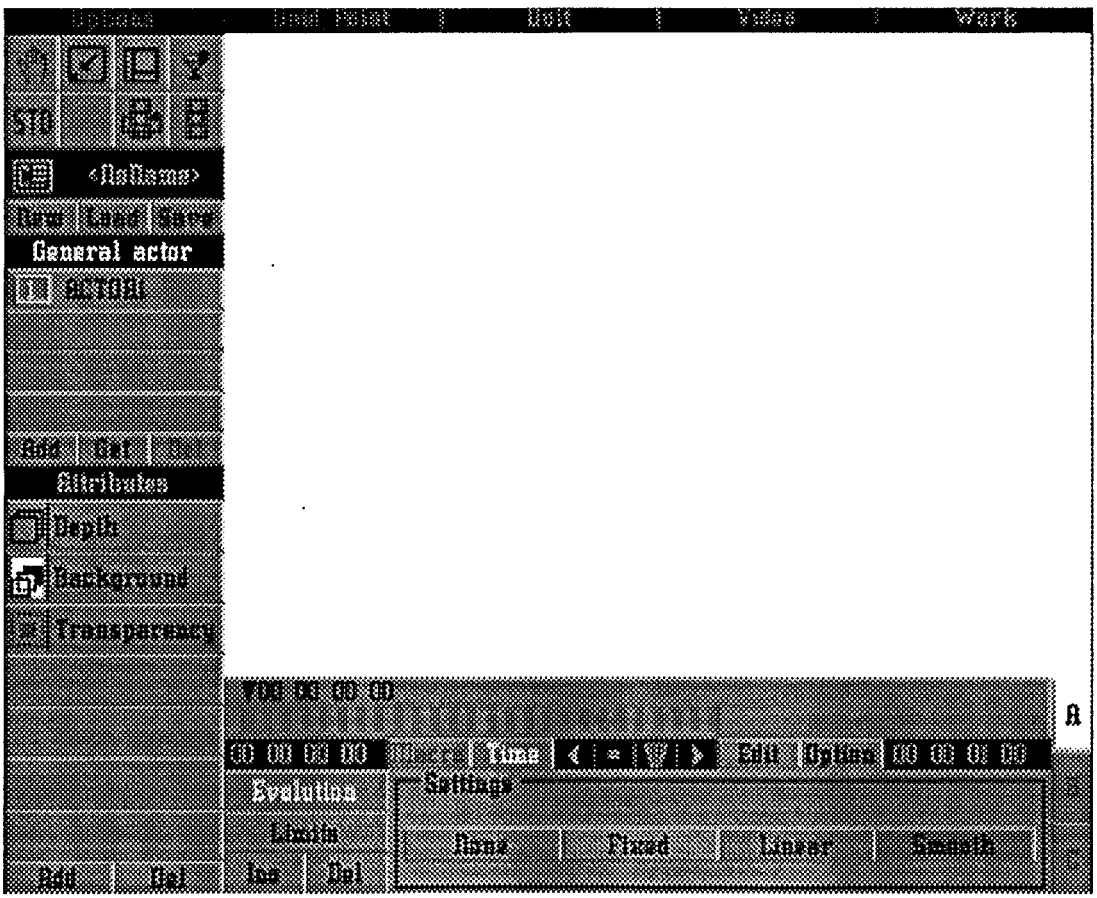
CHAPITRE 4

L'INTERFACE HOMME-MACHINE

Le but de ce chapitre n'est pas d'appréhender l'interface homme-machine complète du Composer mais de détailler certains éléments de l'application entre autres au niveau du scénario. En effet, l'interface graphique de l'application est très proche de celle du Sequencer notamment en ce qui concerne la forme et les éléments (bandes de menu, chronogramme) tels qu'ils sont décrits dans [CHA_92]. Ceci est fait dans un souci d'homogénéité entre les différentes applications des systèmes Getris. De plus, l'interface graphique du Composer est décrite en détail dans sa totalité dans le manuel utilisateur [GET_94A].

On insistera donc plus ici sur les particularités de la gestion des acteurs et de leurs attributs en se rapprochant des concepts traités dans le chapitre précédent (étude et conception).

Le schéma suivant présente l'aspect général de l'interface de l'application avec l'environnement de gestion du scénario :



1. LA DESCRIPTION DU SCENARIO

Le scénario est décrit par :

- une durée;
- une liste d'acteurs dont l'acteur général;
- une liste d'attributs par acteur.

Les attributs de l'acteur général s'appliquent à l'ensemble de la scène. Ils se composent avec les attributs propres à chaque acteur.

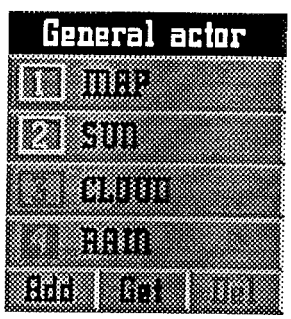
Lorsque l'utilisateur commence une session, un scénario par défaut est proposé. Ce scénario ne contient que l'acteur général et ses attributs par défaut. La durée de ce scénario est de une seconde.

Le panneau de mise à jour du scénario permet à l'utilisateur de :

- redéfinir la durée du scénario (en cliquant sur le nom du scénario);
- commencer un nouveau scénario (bouton New);
- charger un scénario existant sur disque (bouton Load);
- sauvegarder le scénario sur disque (bouton Save).

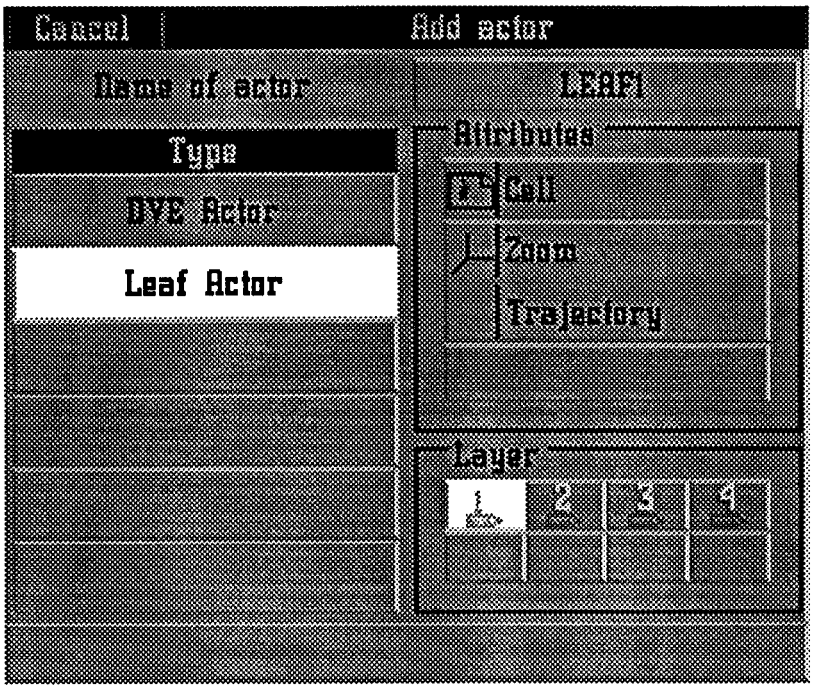
2. LES ACTEURS DU SCENARIO

La liste des acteurs du scénario est présentée avec l'acteur général en tête. Le menu de gestion des acteurs permet plusieurs actions à l'utilisateur :



- l'ajout d'un nouvel acteur (bouton Add);
- le chargement d'un acteur à partir d'un scénario sur disque (bouton Get);
- la suppression d'un acteur de la liste (bouton Del). Cette opération est interdite pour l'acteur général.

Lorsque l'utilisateur ajoute un nouvel acteur, un panneau lui est proposé pour choisir le type d'acteur à ajouter :



L'utilisateur sélectionne un acteur à ajouter dans la liste des types d'acteurs proposés par le panneau. Les attributs par défaut sont indiqués pour cet acteur dans la case « Attributes ». L'acteur « LEAF » est un acteur composé de cells et placé sur un plan mémoire VPM. L'acteur « DVE » est le même acteur placé sur un plan DVE-Layer permettant les transformations 3D en temps réel.

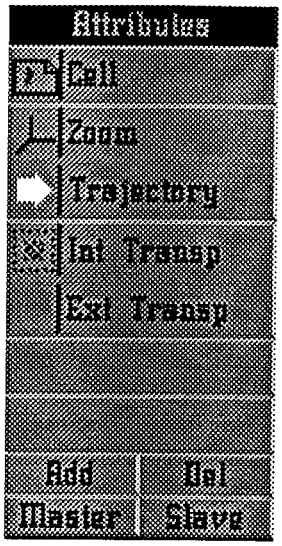
L'acteur courant est sélectionné dans la liste des acteurs du scénario permettant la mise à jour de ses attributs.

3. LES ATTRIBUTS

Lorsqu'un acteur est ajouté au scénario, certains attributs par défaut lui sont automatiquement rattachés. Ces attributs sont ceux qui sont le plus souvent utilisés pour un type d'acteur afin de limiter les manipulations effectuées par l'utilisateur lors de la mise en place de son animation.

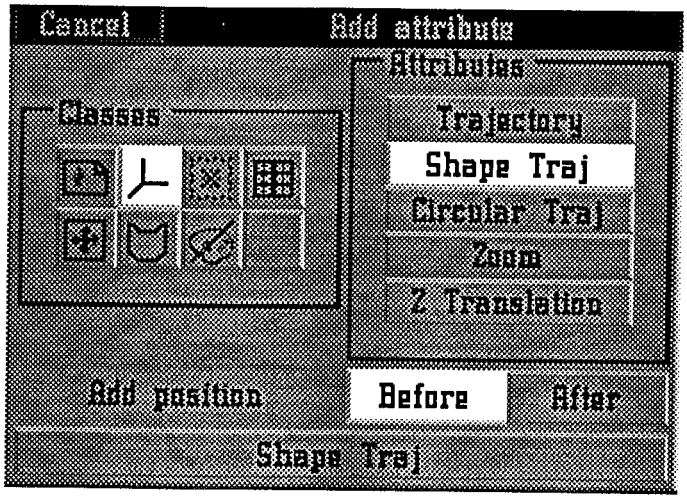
L'utilisateur peut ajouter de nouveaux attributs ou bien en supprimer certains à l'aide du panneau de mise à jour des attributs.

Ce panneau présente la liste des attributs de l'acteur courant. Il contient un menu de gestion des attributs :



- pour l'ajout d'un nouvel attribut (bouton Add);
- pour la suppression d'un attribut de la liste (bouton Del). L'acteur supprimé est celui qui est marqué par la flèche;
- 2 choix pour la gestion de l'asservissement des attributs : mise à jour des fonctions d'asservissement (bouton Master) et asservissement des attributs (bouton Slave).

L'icône de la colonne de gauche représente la classe de l'attribut. Les attributs sont regroupés par classes dans la liste. Dans une même classe, l'ordre des attributs a son importance pour la composition des attributs. L'ajout est effectué avant ou après l'attribut marqué (flèche blanche). Ce choix est effectué dans le panneau d'ajout d'un attribut :



Le panneau de choix des attributs affiche la liste des attributs autorisés pour l'acteur courant. Il est affiché lors de l'ajout d'un nouvel attribut. Ce panneau est constitué de 3 rubriques :

- un menu pour le choix de la classe d'attribut à ajouter;
- un menu de choix de l'attribut parmi ceux de la classe sélectée;
- un menu pour choisir la position d'insertion de l'attribut par rapport à l'attribut marqué dans le panneau affichant la liste des attributs de l'acteur courant.

Lorsqu'un attribut est sélectionné dans la liste, il devient l'attribut courant et le panneau correspondant est alors affiché. Ce panneau permet la mise à jour des paramètres de l'attribut.

3.1 LES PANNEAUX DES ATTRIBUTS

Le panneau de mise à jour des paramètres des attributs se décompose en 3 zones; soit de haut en bas; le menu de sélection des paramètres, le menu de sélection du dialogue écran et le panneau de mise à jour des paramètres.

Le menu de sélection des paramètres permet de choisir le panneau qui est affiché dans la troisième zone. Plusieurs panneaux sont nécessaires pour décrire le nombre important de paramètres qui sont utilisés par rapport à la place réservée aux panneaux de mise à jour des attributs.

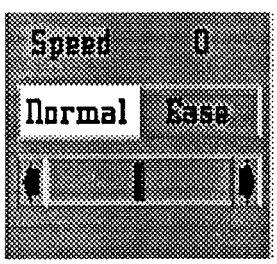
Le menu de sélection du dialogue écran est identique au premier menu. Il permet de choisir le mode de dialogue écran utilisé. Ce menu est absent pour les attributs qui ne possèdent pas de dialogue écran.

La troisième zone permet l'affichage des panneaux de mise à jour des paramètres. Ces panneaux sont propres aux attributs excepté le panneau de mise à jour de la vitesse d'évolution du coefficient qui est identique pour tous les attributs.

3.2 LE DIALOGUE ECRAN

Le dialogue écran est un mécanisme permettant la mise à jour des paramètres de l'attribut de manière graphique à l'aide du mécanisme de dialogue (souris, tablette). La position de la souris est matérialisée par un curseur à l'écran. L'utilisateur peut déplacer ce curseur en déplaçant la souris. Le déplacement combiné avec l'appui sur le bouton de la souris permet la mise à jour des paramètres de l'attribut.

3.3 LE PANNEAU DE VITESSE



Ce panneau est constitué de 3 zones. Il est identique pour tous les attributs.

La valeur courante de la vitesse est affichée. Lorsque cette vitesse est forcée par l'utilisateur, la valeur est affichée avec la couleur des objets sélectés.

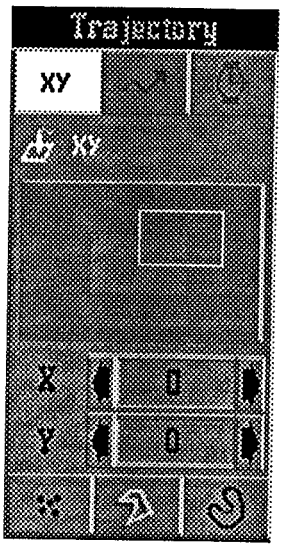
Le menu Normal / Ease (Amorti) : En mode normal, la vitesse est calculée automatiquement par le système en fonction de l'évolution du coefficient de l'attribut. L'utilisateur sélectionne ce mode pour annuler la mise à jour forcée de la vitesse (amorti ou valeur). Le mode amorti permet de forcer la vitesse à une valeur nulle.

L'ascenseur de saisie de la vitesse permet de forcer la vitesse avec une valeur particulière.

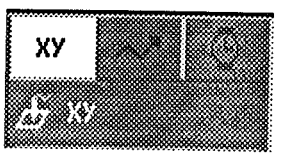
3.4 LES ATTRIBUTS DE TRAJECTOIRE

3.4.1 L'attribut de trajectoire par points clés

Le panneau de cet attribut est composé des 3 zones d'un panneau d'attribut ainsi que d'un menu pour définir l'évolution de la trajectoire :



Le menu de sélection permet le choix entre 3 modes de mise à jour pour les coordonnées du point courant, la direction de la trajectoire et pour la vitesse d'évolution de l'acteur sur la trajectoire.

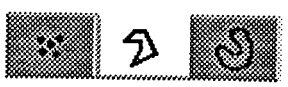


Le menu de sélection du dialogue écran permet le choix parmi les 3 mêmes modes.

Pour la mise à jour des coordonnées, le dialogue écran peut prendre 3 états différents : mise à jour dans le plan (selon les axes X et Y) ou mise à jour selon une direction déterminée, soit en X (la valeur de Y reste inchangée), soit en Y (la valeur de X reste inchangée).

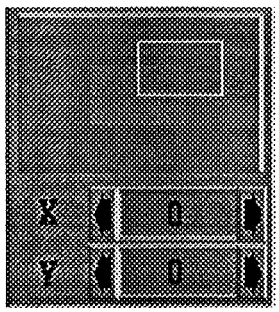
Lorsque le bouton de mise à jour des coordonnées est sélecté, si l'utilisateur clique de nouveau dessus, le dialogue écran change de mode.

Le menu d'évolution de la trajectoire permet de décrire le déplacement de l'acteur entre 2 points clés parmi 3 modes :



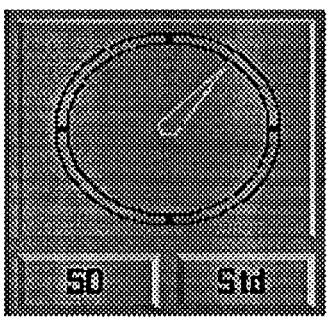
- fixe : l'acteur saute d'un point clé à l'autre;
- linéaire : l'acteur se déplace le long des arêtes d'un polygone défini par les points clés;
- lissée : l'acteur se déplace sur une courbe de type Spline dont les points de passage sont les points clés.

3.4.1.1 Le panneau des coordonnées



Ce panneau se compose de 2 ascenseurs de saisie pour les coordonnées X et Y; et d'une zone graphique (Map). Sur la Map, l'acteur est symbolisé par sa boite englobante (rectangle blanc). La plage de variation de la trajectoire est désignée par la zone en grisé sombre. La partie claire schématise les bornes de l'écran (partie visible par l'utilisateur).

3.4.1.2 Le panneau de direction



Ce panneau permet la mise à jour de la direction prise par la trajectoire (tangente au point de passage de la spline). Ce panneau n'est accessible que lorsque la trajectoire est en mode lissée. Il contient 3 éléments :

- Le compas permet la saisie de la direction de manière graphique;
- la zone de saisie de l'angle sert également à visualiser la valeur de la direction;
- le bouton STD permet de revenir à une courbe où la direction n'est pas forcée.

3.4.1.3 Le dialogue écran

Le dialogue écran de l'attribut de trajectoire est utilisé pour la mise à jour des paramètres de l'attribut dans deux modes :

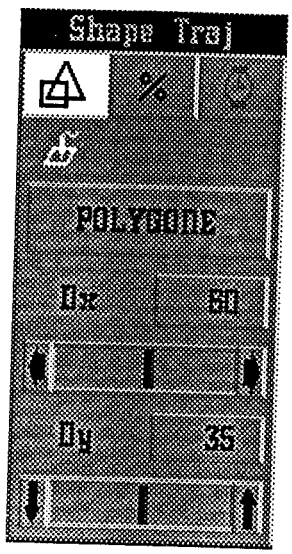
le mode coordonnées permet à l'utilisateur de déplacer l'acteur en déplaçant la souris tout en tenant le bouton de clic enfoncé. Les coordonnées sont modifiées relativement au déplacement.

Le mode direction permet de forcer la tangente de la trajectoire avec un angle précis. L'angle de la direction est recalculé lorsque l'utilisateur clique sur la souris. La direction prise est celle du vecteur constitué du point guide de l'acteur et de la position du curseur sur l'écran.

3.4.2 L'attribut de trajectoire par shape

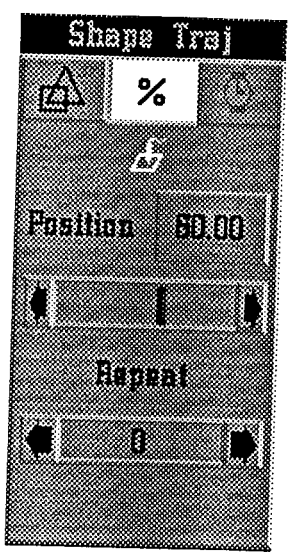
Le domaine d'évolution de cet attribut est calculé en fonction de la shape utilisée. La modification du domaine n'est pas autorisée pour cet attribut et seul le coefficient peut varier afin de positionner l'acteur sur la trajectoire à la position voulue. Le panneau de menu se décompose en 2 sous-panneaux : le premier permet la mise à jour de la shape utilisée et le positionnement de la shape par rapport à la scène. Le second permet le positionnement de l'acteur sur la trajectoire constituée par la shape.

Le premier panneau de menu propose 3 zones de saisie :



- Une zone d'accès au panneau de choix de la shape. Cette zone est renseignée avec le nom de la shape utilisée dans le scénario. Lorsque aucune shape n'est précisée, l'acteur reste positionné au centre de l'écran.
- Deux zones pour le déplacement de la shape sur l'axe des X et des Y par rapport au centre de l'écran.

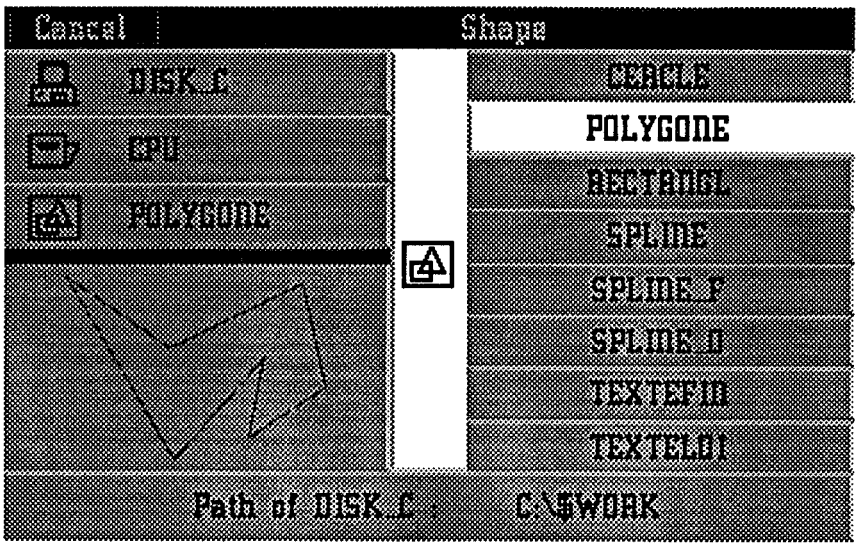
Le second panneau propose la saisie de la position de l'acteur sur la shape :



La position 0 correspond au premier point de la trajectoire et la position 100 correspond au dernier point.

Certaines shapes sont dites fermées : le premier point de la trajectoire est confondu avec le dernier (ex : cercle, rectangle, polygone fermé, etc.). Dans ce cas, il est possible d'effectuer plusieurs tours de la forme à l'aide de l'ascenseur de saisie de la répétition.

Le panneau de choix de la shape permet l'accès aux fichiers disque. Il présente un aperçu de la shape qui est sélectionnée :



3.4.2.1 Le dialogue écran

Le dialogue écran fonctionne selon 2 modes en fonction du choix sélecté dans le menu de dialogue écran : il permet la mise à jour de Dx et Dy si l'icône de dialogue écran est placée sous l'icône du panneau shape. Il permet la mise à jour de la position de l'acteur sur la trajectoire si l'icône de dialogue écran est placée sous l'icône du panneau position. Dans ce cas, le dialogue écran positionne l'acteur sur le point le plus proche sur la trajectoire par rapport à la position du curseur au moment du clic.

3.4.3 L'attribut de trajectoire circulaire

La mise à jour de cet attribut se décompose en 2 phases :

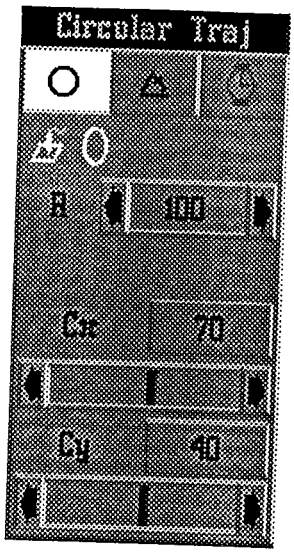
- la description de la rotation est effectuée à l'aide du panneau permettant la mise à jour du centre et du rayon du cercle servant de base à la rotation;
- le second panneau permet de positionner l'angle de rotation de l'acteur sur le cercle précédemment saisi.

3.4.3.1 Le panneau de position

Ce panneau se compose de 3 entités :

- une zone de saisie du rayon;
- un bloc de saisie de la coordonnée en X du centre du cercle constitué d'un ascenseur et d'une zone de visualisation / saisie de la valeur numérique;

- le même bloc pour la coordonnée en Y.

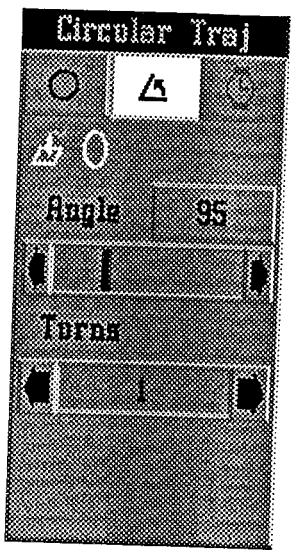


La trajectoire de base peut être soit un cercle, soit une ellipse. Pour le cercle, une zone de saisie unique (ascenseur R) permet la mise à jour des rayons Rx et Ry. Pour une trajectoire en ellipse, cet ascenseur est remplacé par 2 ascenseurs Rx et Ry pour une mise à jour séparée des 2 rayons.

Le premier bouton du menu de choix du panneau permet de basculer d'un mode à l'autre. L'icône de ce bouton représente soit un cercle, soit une ellipse pour indiquer le mode utilisé :

3.4.3.2 Le panneau de rotation

Il contient 3 entités pour la mise à jour de l'angle et du nombre de tours :



- un ascenseur de variation de l'angle;
- une zone de saisie de la valeur numérique de l'angle;

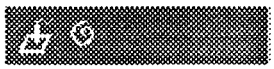
- un ascenseur mixte de variation du nombre de tour permettant également la visualisation et la saisie de la valeur numérique.

3.4.3.3 Le dialogue écran

Le dialogue écran de cet attribut est partagé en deux modes sur le même principe que les panneaux de menu : il permet la mise à jour soit de l'angle de rotation, soit de la position du cercle de rotation.

Dans le mode position, lorsque l'utilisateur clique à l'intérieur du cercle, il peut déplacer le centre de la rotation. Lorsque le curseur est positionné à l'extérieur du cercle, c'est le rayon qui est mis à jour si l'utilisateur clique sur le bouton de la souris.

La trajectoire visualisée représente le cercle ou l'ellipse servant de base à la trajectoire mais il est possible d'obtenir la trajectoire complète afin de vérifier la trajectoire finale de l'acteur. Le choix est effectué à l'aide du premier bouton du menu de choix du dialogue écran :

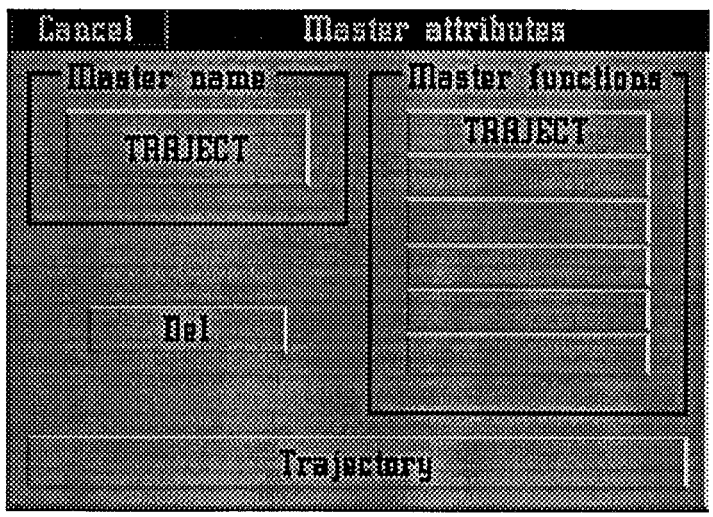


3.5 L'ASSERVISSEMENT DES ATTRIBUTS

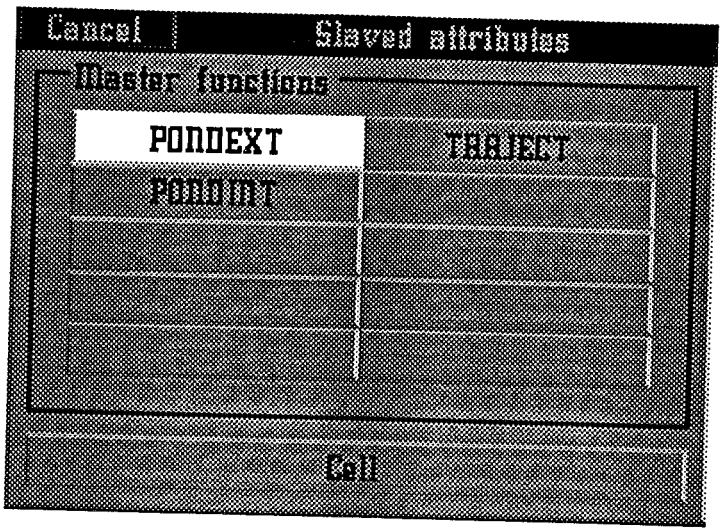
Les deux boutons du bas du menu de gestion des attributs sont dédiés à l'asservissement :

- bouton Master pour l'accès au panneau permettant l'identification de l'attribut en tant que fonction d'asservissement;
- bouton Slave pour accéder au panneau d'asservissement des attributs par l'une des fonctions maître.

Le panneau Master attributes présente dans sa partie de droite, la liste des fonctions d'asservissement identifiées dans le scénario. La zone de gauche (Master name) permet la saisie du nom de la fonction d'asservissement. L'accès à ce panneau n'est pas autorisé lorsque l'attribut est asservi.



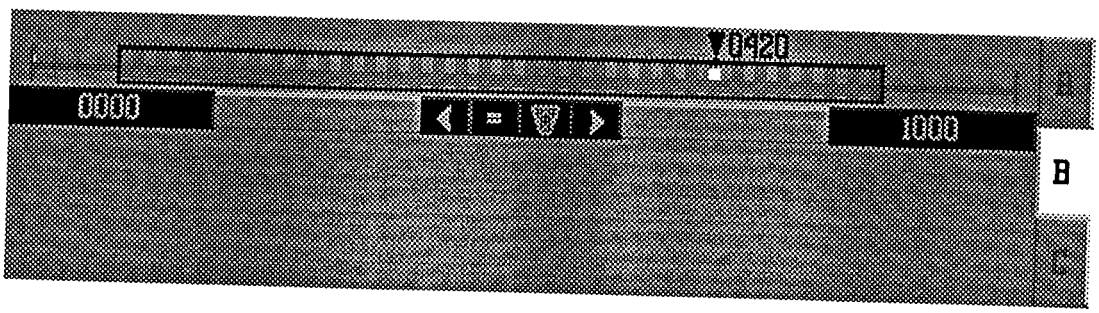
Le panneau Slave attributes affiche la liste des fonctions d'asservissement sous la forme d'un menu. L'utilisateur sélectionne la fonction qui asservit l'attribut courant. Pour désélectionner une fonction, il suffit soit de sélectionner une nouvelle fonction, soit de cliquer sur le choix sélecté. Dans le premier cas, l'attribut sera asservi avec la nouvelle fonction sélectionnée alors que dans le second, il n'est plus asservi. L'accès à ce panneau n'est pas autorisé lorsque l'attribut est une fonction d'asservissement.



Les contraintes d'accès à ces deux panneaux sont utiles pour éviter les conflits liés aux effets de boucle : A est asservi par B qui est asservi par C et C est asservi par A.

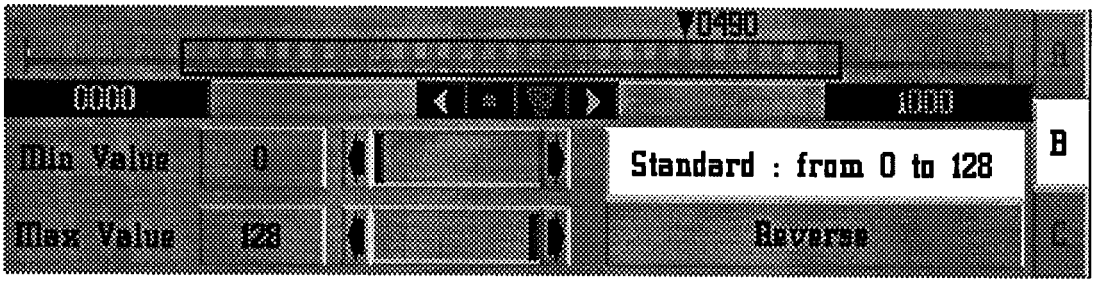
Lorsqu'un attribut est asservi, la mise à jour de son coefficient n'est pas autorisée puisque ce dernier est calculé automatiquement à partir de la fonction d'asservissement. Un mécanisme permettant la mise à jour du domaine d'un attribut a été mis en place. Ce mécanisme est composé d'un chronogramme autorisant le déplacement à l'intérieur du domaine indépendamment du temps courant du scénario.

Pour les attributs de type temporel, le chronogramme est présenté dans le menu B de la bande basse :



La mise à jour du domaine est effectuée par l'intermédiaire des panneaux de menu selon le même principe que lorsque l'attribut n'est pas asservi.

Pour les attributs de type intervalle, les panneaux de menu sont utilisés pour la visualisation des paramètres des attributs et ne permettent pas la mise à jour du domaine. Un panneau spécifique vient compléter le chronogramme. Ce panneau permet la mise à jour des valeurs min et max de l'attribut contenues dans le domaine :



Le menu Standard/Reverse permet le positionnement de min et de max avec des valeurs particulières :

- Standard place les valeurs extrêmes de l'attribut dans min et max; ce choix permet de faire varier 2 attributs dans le même sens;
- Reverse place ces mêmes valeurs en échangeant min et max; ce choix permet de faire varier 2 attributs en sens inverse.

Exemple :

Un attribut de pondération est de type intervalle et ses valeurs extrêmes sont (0 ; 128).

Le bouton Standard permet de forcer min à 0 et max à 128. Dans ce cas, l'attribut varie entre 0 et 128 lorsque le coefficient varie entre 0 et 1.

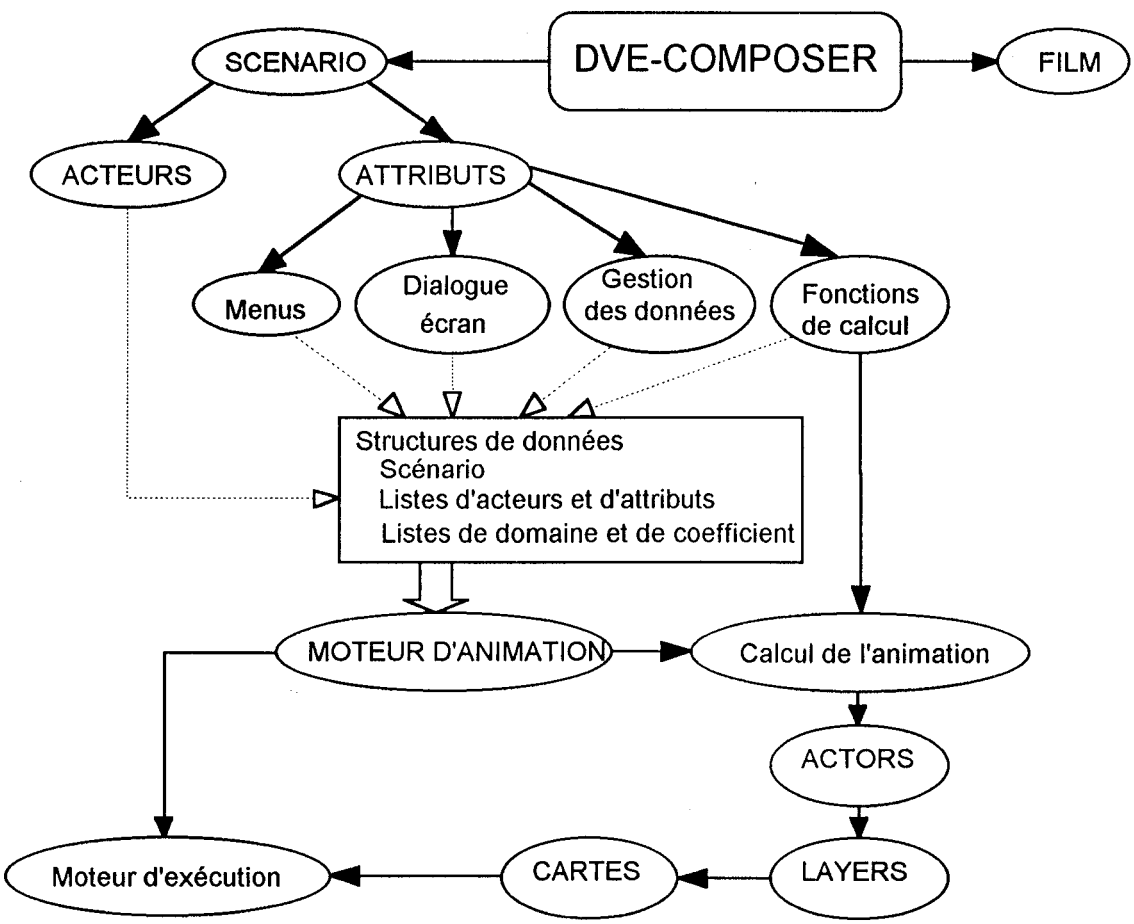
Le bouton Reverse permet de forcer min à 128 et max à 0. Dans ce cas, l'attribut varie entre 128 et 0 lorsque le coefficient varie entre 0 et 1.

CHAPITRE 5

LA MISE EN OEUVRE

Ce chapitre présente les choix faits au niveau des structures de données et des algorithmes de traitement après avoir présenté l'architecture globale du Composer afin de situer les modules développés et décrits dans le reste du chapitre.

1. LE DVE-COMPOSER



L'application DVE-Composer est découpée en deux contextes du point de vue de l'utilisateur. Le contexte de film permet la définition d'une séquence de sortie pour l'enregistrement de l'animation finale sur un scope ou sur disque. Le contexte de scénario est chargé de la gestion du scénario, de ses acteurs et de ses attributs. Il permet la description de l'animation. Ce

découpage se retrouve au niveau des modules du Composer. Les deux contextes utilisent un troisième bloc modulaire pour la gestion et le calcul de l'animation : le moteur d'animation.

Dans le cadre de l'étude effectuée, on s'intéressera plus particulièrement aux modules de gestion du scénario pour tout ce qui concerne les acteurs et les attributs. L'introduction des différents modules de niveau le plus bas (moteur, actors, layers, cartes) permet au lecteur de situer les éléments de l'application.

1.1 LE MOTEUR D'ANIMATION

Le calcul de l'animation est effectué dans deux contextes pour les deux modes de fonctionnement direct et pré-calculé. Le mode direct permet l'exécution immédiate d'une commande. La commande est transmise directement au module carte sans utiliser les buffers et le moteur d'animation. Le mode pré-calculé effectue l'animation en deux phases via le moteur d'animation décomposé en deux parties : une partie calcul et une partie exécution.

La première effectue la mise à jour des structures des Actors à partir du scénario. Pour chaque trame de l'animation, le moteur parcourt la liste des acteurs et pour chaque acteur, il examine la liste des attributs. Il effectue les calculs à partir du coefficient et du domaine d'évolution de chaque attribut. Les structures des Actors sont mises à jour par combinaison des résultats de calcul. Le moteur utilise pour cela les fonctions de calcul des modules d'attributs décrits dans ce chapitre.

Le moteur d'exécution intervient au niveau des buffers. Cette partie du moteur parcourt le contenu des buffers et provoque l'exécution des commandes élémentaires au niveau des cartes.

1.2 LES ACTORS

La couche Actor sert d'interface entre les layers et l'application Composer (à ne pas confondre avec les acteurs du scénario). Le module Actor reçoit les informations provenant du calcul de l'animation, analyse, filtre et transmet les commandes aux layers.

1.3 LES LAYERS

Un layer est un module logique réalisant l'interface entre le niveau matériel (les cartes et leurs modules) et le niveau logiciel (les applications). La mise en place de ces modules garantit l'indépendance entre le niveau matériel des systèmes de synthèse d'images et le niveau logiciel qui le pilote. Le logiciel utilise les fonctions du layer sans connaître l'implémentation au niveau le plus bas. De la même façon, les modules de base effectuent des commandes élémentaires ordonnées par les layers et informent ces derniers du résultat obtenu (commande exécutée en temps réel ou non).

1.4 LES CARTES

Le pilotage des différents éléments physiques du système de synthèse d'images est réalisé par des modules logiciels interfacés avec le niveau matériel. Ces modules sont capables de travailler indépendamment de l'application qui leur envoie les commandes à effectuer. Ces commandes sont interprétées et transformées en commandes élémentaires transmises aux cartes électroniques.

Les commandes élémentaires sont stockées dans des buffers, indicés par le numéro de trame. L'exécution d'une animation est réalisée en parcourant les buffers trame par trame et en envoyant les commandes aux cartes. La préparation des buffers pour visualisation du scénario complet est effectuée dans une phase de calcul spécifique. Le parcours des buffers peut alors être effectué de manière répétitive pour visualiser l'animation plusieurs fois.

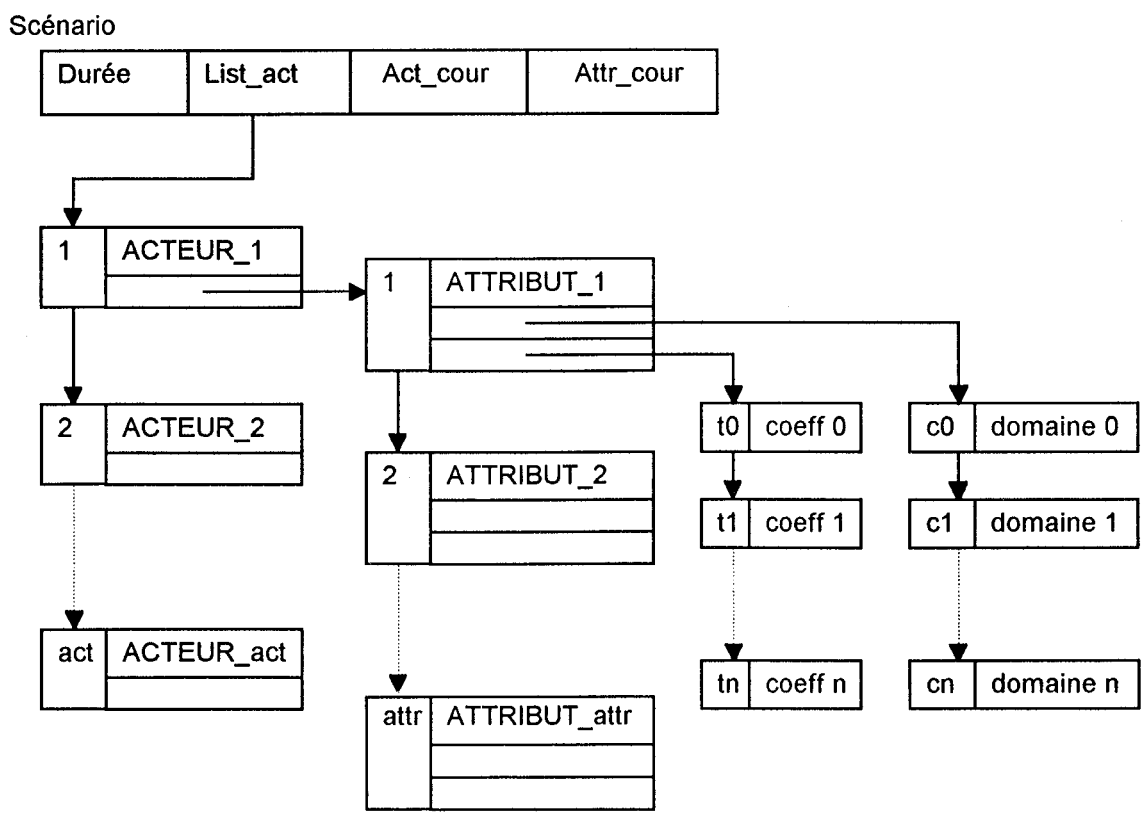
2. LE SCENARIO

La liste de données est un des éléments essentiels de gestion des données. Elle permet la manipulation des données de façon très souple : ajout, suppression et modification de n'importe quel élément accédé par un indice. Le module de gestion des listes utilisé par les applications, permet le calcul automatique des valeurs interpolées entre deux étapes lors de la lecture d'un élément.

Plusieurs éléments sont organisés en liste dans le scénario :

- liste des acteurs du scénario;
- liste des attributs d'un acteur;
- liste des étapes du domaine d'évolution de l'attribut;
- liste des étapes du coefficient d'évolution de l'attribut.

La description du scénario est un enchaînement de ces listes :



Les informations propres au scénario sont regroupées dans une structure de données :

```
typedef struct dsce
{
        int duree;              // durée du scénario en nombre de trames
        int tps_cour;           // temps courant (numéro de trame courante)
        int nb_act;             // nombre d'acteurs du scénario
        int num_act;            // numéro de l'acteur courant
        liste *list_act;        // liste des acteurs du scénario
        dact *act_cour;         // acteur courant
        dattr *attr_cour;       // attribut courant
} dsce;
```

Deux structures sont prévues pour recevoir les données de l'acteur courant et de l'attribut courant lors de l'accès aux listes. Ces structures sont utilisées soit pour la mise à jour d'un des éléments de la liste, soit pour la consultation des données.

2.1 LES ACTEURS

```
typedef struct dact
{
        char name[10];          // nom de l'acteur
        long attr_autorise;     // masque des attributs autorisés
        int num_attr;           // numéro de l'attribut courant
        liste *list_attr;       // liste des attributs
        refer ref_actor;        // référence de l'actor attaché à l'acteur
} dact;
```

Il existe plusieurs types d'acteurs. Chaque acteur possède en fonction de ce type une liste d'attributs autorisés ainsi qu'une liste d'attributs par défaut. Ces derniers sont automatiquement insérés dans la liste d'attributs de l'acteur lorsqu'il est ajouté au scénario.

2.2 LES ATTRIBUTS

```
typedef struct dattr
{
        int num_typ;            // numéro et type d'attribut
        char name[10];          // nom de la fonction d'asservissement
        char asservi[10];       // fonction d'asservissement de l'attribut
        void *dom_cour;         // domaine courant
        void *coef_cour;        // coefficient courant
        liste *list_dom;        // liste du domaine d'évolution
        liste *list_coef;       // liste du coefficient d'évolution
} dattr;
```

L'attribut est identifié par son numéro et son type : chaque attribut possède un numéro unique. Le type permet de regrouper les attributs en classes. Les chaînes de caractères « name » et « asservi » sont utilisées pour l'asservissement des attributs : « name » identifie l'attribut en tant que fonction d'asservissement et « asservi » est le nom de la fonction avec laquelle

l'attribut est asservi. Les structures dom_cour et coef_cour sont prévues pour recevoir respectivement les données du domaine et du coefficient lors de l'accès aux listes list_dom et list_coef.

3. LA GESTION DES ATTRIBUTS

La gestion des attributs est organisée en 3 niveaux :

- le niveau interface homme-machine comprend les panneaux de menu et le dialogue écran pour la saisie des paramètres du scénario;
- le niveau données est consacré à la mise à jour des informations dans les différentes structures de stockage : listes pour le domaine d'évolution et son coefficient;
- le niveau calcul permet la mise à jour des composants du système (cartes) via le moteur d'animation.

A chaque attribut ATTR correspond un module distinct pour la gestion du panneau de menu (MATTR.C), la gestion du dialogue écran (EATTR.C), les fonctions de mise à jour des données et de calcul (ATTR.C).

Exemple :

Pour l'attribut de translation décrite par points clés, on trouve :

- TXYPTS.C : module de gestion de l'attribut (calcul, mise à jour);
- MTXYPTS.C : module de gestion du panneau de menu;
- ETXYPTS.C : module de gestion du dialogue écran.

Cette organisation permet l'indépendance entre les différents attributs et assure l'extensibilité de l'application. L'ajout d'un nouvel attribut nécessite le développement des 3 modules et l'ajout de la structure d'adresses des fonctions au niveau de l'application. Il est possible de revoir l'habillage des menus sans impact sur le module de gestion des données ou bien de changer le mécanisme de calculs sans que les menus changent d'aspect.

3.1 MISE A JOUR ET CALCUL

Ce module se compose de fonctions de traitement des données, de fonctions de mise à jour des informations et de fonctions de calcul. Ces fonctions se retrouvent pour tous les attributs. L'accès à une des fonctions est effectué par un tableau d'indirection indicé par le numéro d'attribut et contenant les adresses des fonctions du module :

```
typedef struct dfctattr
{
    int (*trtlist)();          // fonction de traitement de la liste d'attribut
    int (*trtlistcoef)();      // fonction de traitement de la liste de coefficient
    int (*trtlistdom)();       // fonction de traitement de la liste de domaine
    int (*init)();             // initialisation des listes de domaine et coefficient
    int (*standard)();         // mise à jour des listes de domaine et coefficient
                               // avec les valeurs par défaut
    int (*setactor)();         // calcul de la valeur de l'attribut et mise à jour de l'Actor
```

```
        int (*version)();          // numéro de version de l'attribut
        int (*loadress)();         // lecture des informations sur disque
        int (*saveress)();         // sauvegarde des informations sur disque
        .....
} dfctattr;
```

Chaque attribut initialise sa propre structure d'adresses.

3.1.1 Les fonctions de traitement des listes

Le gestionnaire de listes est prévu pour appeler automatiquement une fonction lors de la manipulation d'un des éléments de la liste (ajout, suppression, consultation, etc.). Cette fonction est signalée lors de la création de la liste.

Le module de gestion de l'attribut décrit 3 fonctions de traitement pour les listes d'attribut, de domaine et de coefficient. Ces fonctions effectuent des opérations particulières propres à l'attribut lors de la manipulation des listes correspondantes.

Fonction de traitement de la liste d'attributs :

```
int trtlist(int operation, int indice, dattr *attr, void *infos)
{
    Selon opération :
        Cas ADD :
            // créer la liste de domaine attr->list_dom
            // créer la liste de coefficient attr->list_coef
            // initialiser le domaine courant attr->dom_cour
            // initialiser le coefficient courant attr->coef_cour
        Cas DEL :
            // vider puis détruire la liste de domaine attr->list_dom
            // vider puis détruire la liste de coefficient attr->list_coef
    FinSelon
}
```

Fonction de traitement de la liste de domaine :

```
int trtlistdom(int operation, int temps, int evol, void *infos)
{
    Selon opération :
        Cas ADD :
        Cas DEL :
        Cas MAJ :
            // recalculer les coefficients de la spline utilisés pour l'évolution
            // en mode lissé.
    FinSelon
}
```

Fonction de traitement de la liste de coefficient :

```
int trtlistcoef(int operation, int temps, int evol, void *infos)
{
    Selon opération :
```

```
            Cas ADD :
            Cas DEL :
            Cas MAJ :
                // recalculer les coefficients de la spline utilisés pour l'évolution
                // en mode lissé.
        FinSelon
}
```

3.1.2 Les fonctions de mise à jour des listes

La fonction d'initialisation est utilisée par la fonction trtlist() pour initialiser les listes de domaine et de coefficient lors de leur création. Elle ajoute un élément dans chacune des 2 listes au temps 0 avec la valeur prise par défaut par l'attribut.

```
int init(dattr *attr)
{
        // attr->coef_cour = valeur initiale du coefficient
        // insérer un nouveau coefficient au temps 0 dans attr->list_coef
        // attr->dom_cour = valeur initiale du domaine
        // insérer un nouveau domaine au temps 0 dans attr->list_dom
}
```

La fonction standard est appelée par un bouton du menu de gestion du scénario. Elle permet à l'utilisateur de placer une étape au temps courant avec les valeurs par défaut de l'attribut.

```
int standard(dattr *attr)
{
        // attr->coef_cour = valeur standard du coefficient
        // mettre à jour la liste de coefficient au temps courant
        // attr->dom_cour = valeur standard du domaine
        // mettre à jour la liste de domaine au temps courant
}
```

La fonction version() indique le numéro de version de l'attribut. Ce numéro est utilisé pour assurer la compatibilité lors d'une évolution du mode de description de l'attribut en particulier lorsque le format des données change. Il permet d'assurer la compatibilité des différentes versions de l'application avec celles qui sont déjà livrées à des clients. Cette fonction est utilisée par la fonction loadress() de lecture des informations de l'attribut à partir d'un fichier.

```
int version()
{
        // retourne le numéro de version de l'attribut
}
```

La fonction saveress() est invoquée par la procédure de sauvegarde du scénario sur disque. Elle est utilisée pour écrire les informations contenues dans les listes de domaine et de coefficient de l'attribut dans le fichier de sauvegarde.

```
int saveress(file *fichier, dattr *attr, long duree)
{
        Pour chaque élément de la liste attr->list_dom
            Si indice_element <= duree
```

```
                    // écrire le contenu de attr->dom_cour dans fichier
                FinSi
            FinPour
            Pour chaque élément de la liste attr->list_coef
                Si indice_element <= duree
                    // écrire le contenu de attr->coef_cour dans fichier
                FinSi
            FinPour
    }
```

La fonction loadress() est utilisée pour la restitution des informations écrites dans le fichier par la fonction saveress().

```
    int loadress(file *fichier, dattr *attr)
    {
            Si version() = 1
                loadress01(fichier, attribut);
            Sinon
                Si version() = 2
                    loadress02(fichier, attribut);
                Sinon
                    .....
                FinSi
            FinSi
    }

    int loadress01(file *fichier, dattr *attr)
    {
            Pour les informations de domaine du fichier
                // Lecture des informations de domaine dans fichier
                // mettre à jour attr->dom_cour avec les informations lues
                // Insérer un élément dans attr->list_dom
            FinPour
            Pour les informations de coefficient du fichier
                // Lecture des informations de coefficient dans fichier
                // mettre à jour attr->coef_cour avec les informations lues
                // Insérer un élément dans attr->list_coef
            FinPour
    }
```

3.1.3 Les fonctions de calcul

La fonction setactor() assure la mise à jour du niveau Actor avec les informations du scénario. Elle provoque la mise à jour des cartes au niveau le plus bas de l'application pour visualiser l'animation.

Par exemple dans le cas des attributs de géométrie; cette fonction prépare la matrice 4 x 4 de transformation puis compose cette matrice avec celle de l'Actor. Ce dernier envoie les ordres au niveau des cartes pour effectuer les opérations de base réalisant la transformation demandée.

```
setactor(refer ref_actor, void *coef_cour, void *dom_cour)
{
        // calculer la valeur de l'attribut à partir de coef_cour et dom_cour
        // transformer cette valeur dans le format de l'Actor
        // envoyer cette valeur à l'Actor ref_actor
}
```

3.2 LES PANNEAUX DE MENU

Chaque attribut ATTR possède son propre panneau de menu développé dans le fichier MATTR.C. Le point d'entrée du module est la fonction permettant de définir le panneau. Celui-ci est créé en tant qu'objet dans la banque des objets de l'application. Un mécanisme unique de gestion des environnements permet le parcours des objets de la banque et appelle les fonctions appropriées. Ces fonctions sont liées au panneau au moment de sa définition et sont décrites dans la suite de ce paragraphe.

Le point d'entrée du panneau est nommé par convention DefMAttr(). La valeur retournée par la fonction est la référence du panneau en tant qu'objet dans la banque.

```
refer DefMAttr(refer ref_mere, refer ref_cadre, int coor)
{
        // création de l'objet panneau :
        refpanel = DefPanel(ref_mere, ref_cadre, coor);
        .....

        // définition des environnements fils du panneau
        env_fils = .....
        DefFctVue(env_fils, F_ExtDial, extdial_fils);
        DefFctVue(env_fils, F_ExtSet, extset_fils);

        // fonction de mise à jour du panneau
        DefFctVue(refpanel, F_Set, set_panel);

        return(refpanel);
}
```

La fonction DefFctVue() permet d'associer une fonction à un environnement :

- une fonction de dialogue associée à un environnement fils du panneau (cas F_ExtDial);
- une fonction de set externe pour la mise à jour des panneaux fils (cas F_ExtSet);
- une fonction de set interne pour la mise à jour du panneau (cas F_Set).

3.2.1 La fonction de dialogue

Une fonction de dialogue (également appelée fonction de séance) est associée à chaque environnement fils du panneau. Elle permet la mise à jour des données du panneau ou de l'application en fonction des événements survenus dans l'environnement fils. Cette fonction est automatiquement invoquée par le système de gestion des environnements lorsqu'une mise à jour est intervenue dans le fils : modification de la valeur d'une zone de saisie, d'un ascenseur, choix dans un menu.

```
int extdial_fils(int message, refer ref, .....)
{
        Si message = FFF_MAJ
                // mise à jour des structures de données de l'application en fonction des
                // informations passées en paramètre
        FinSi
        // Envoi d'un message au panneau père pour signaler la mise à jour
}
```

3.2.2 La fonction de Set externe

Une fonction de set externe est associée à chaque panneau ou environnement. Cette fonction est utilisée pour mettre à jour les données du panneau et de ses fils en fonction d'événements survenus soit dans un des panneaux fils, soit dans le panneau, soit dans l'application. Elle invoque les fonctions de Set des environnements fils pour leur communiquer les nouvelles informations et provoquer leur mise à jour.

```
int extset_panel(refer ref)
{
        message = flag de mise à jour positionné pour le panneau ref.

        // mise à jour des éléments du panneau ref à partir des informations de l'application
        Si message = MAJ_ENV1
                // mettre à jour tous les fils sauf ENV1
                SetEnv(ENV2, MAJ_DATA, .....);
                SetEnv(ENV3, MAJ_DATA, .....);
                .....
        FinSi
        Si message = MAJ_ENV2
                // mettre à jour tous les fils sauf ENV2
                SetEnv(ENV1, MAJ_DATA, .....);
                SetEnv(ENV3, MAJ_DATA, .....);
                .....
        FinSi
        .....
}
```

3.2.3 La fonction de Set interne

Cette fonction permet la mise à jour d'un panneau à partir d'informations communiquées par le panneau père.

```
int set_panel(refer ref, int message, ...)
{
        Si message = MAJ_DATA
                // mettre à jour le panneau ref à partir des informations passées en paramètres
        FinSi
}
```

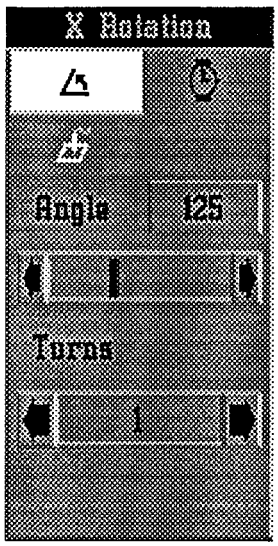
Exemple : cas de la rotation en X (module MROTX.C)

Dans cet exemple, les éléments fils du panneau sont des objets de base gérés dans la bibliothèque graphique (ascenseurs, zones de saisie). La fonction de set externe du panneau reçoit les messages des fonctions de dialogue de ces éléments et provoque leur mise à jour à l'aide de leur fonction de set : SetSlider pour un ascenseur et SetEntry pour une zone de saisie.

La fonction de set externe reçoit également des messages provenant d'autres panneaux de l'application comme le message « nouveau temps courant » positionné par le chronogramme.

Le panneau de cet attribut est composé des éléments suivants :

- une zone de saisie de la valeur numérique de l'angle;
- un ascenseur de réglage de l'angle;
- un ascenseur de réglage du nombre de tours.



La fonction de définition du panneau est la suivante :

```
refer DefMRotx(refer ref_mere, refer ref_cadre, int coor)
{
        refpanel = DefPanel(ref_mere, ref_cadre, coor);
        // définition de la fonction de Set du panneau :
        DefFctVue(refpanel, F_ExtSet, extset_panel);

        // définition de la zone de saisie de l'angle :
        ref_ea = DefEntry(refpanel, coordonnées de la zone);
        DefFctVue(ref_ea, F_ExtDial, extdial_ea);

        // définition de l'ascenseur de l'angle :
        ref_sa = DefSlider(refpanel, coordonnées de la zone, 0, 360);
        DefFctVue(ref_sa, F_ExtDial, extdial_sa);

        // définition de l'ascenseur du nombre de tours :
        ref_st = DefSlider(refpanel, coordonnées de la zone, -90, 90);
        DefFctVue(ref_st, F_ExtDial, extdial_st);
```

```
        return(refpanel);
    }
```

La fonction de set externe du panneau :

```
    int extset_panel(refer ref)
    {
        // prise en compte des messages externes au panneau
        message = flag de mise à jour positionné pour le panneau ref.

        // mise à jour des fils du panneau à partir des informations du domaine et du
        // coefficient courant de l'attribut
        Si message = nouveau temps courant
            SetEntry(ref_ea, angle);
            SetSlider(ref_sa, angle);
            SetSlider(ref_st, nb_tour);
        FinSi

        Si message = MAJ_SLIDER_ANGLE
            SetEntry(ref_ea, angle); // mise à niveau de la zone de saisie
        FinSi

        Si message = MAJ_ENTRY_ANGLE
            SetSlider(ref_sa, angle); // mise à niveau de l'ascenseur
        FinSi
    }
```

Les fonctions de dialogue externe des fils du panneau :

```
    int extdial_ea(int message, refer ref, int valeur)
    {
        Si valeur <> angle courant
            // mise à jour de la liste de coefficient
            // mise à jour de la liste de domaine
            // envoi du message MAJ_ENTRY_ANGLE au panneau père pour indiquer
            // la nouvelle valeur
        FinSi
    }

    int extdial_sa(int message, refer ref, int valeur)
    {
        Si valeur <> angle courant
            // mise à jour de la liste de coefficient
            // mise à jour de la liste de domaine
            // envoi du message MAJ_SLIDER_ANGLE au panneau père pour indiquer
            // la nouvelle valeur
        FinSi
    }

    int extdial_st(int message, refer ref, int valeur)
```

```
{
        Si valeur <> nb_tour courant
                // mise à jour de la liste de coefficient
                // mise à jour de la liste de domaine
                // envoi du message MAJ_SLIDER_TURN au panneau père pour indiquer
                // la nouvelle valeur
        FinSi
}
```

3.3 LE DIALOGUE ECRAN DES ATTRIBUTS

Bien que faisant partie de l'interface homme-machine au même titre que le panneau de menu, les fonctions permettant le fonctionnement du dialogue écran sont développées dans un module séparé.

Le dialogue écran permet la mise à jour des paramètres de l'attribut en utilisant les interfaces de saisie telles que la souris ou la tablette sans intervenir dans le panneau de menu.

Dans le cas de la rotation en X, l'utilisateur peut, avec la souris, faire tourner l'objet : il déplace la souris dans un sens ou l'autre en tenant le bouton appuyé pour incrémenter ou décrémenter l'angle.

Le module de dialogue écran se compose de 6 fonctions :

```
int def()
{
        // définition des objets nécessaires au dialogue écran
        // exemple : la boite englobante du cell, le point guide, ...
}

int fin()
{
        // suppression des objets définis dans la fonction def
}

int dial()
{
        // mise à jour des structures de données au cours du dialogue
}

int echob()
{
        // mise à jour des structures de données de l'attribut lorsque l'utilisateur clique
        // sur le bouton de la souris
}

int echoh()
{
        // idem echob; lorsque le bouton est relâché
}

int aff()
```

```
{
        // affichage, mise à jour des éléments du dialogue écran
}
```

Exemple : cas de la rotation en X (module EROTX.C)

```
int def()
{
        DefTriedre(0);          // définition du repère de la rotation matérialisé
                                // par les 3 axes
        DefPtGuide(1);          // définition du point guide de l'acteur
        DefBox(2);              // définition de la boite englobante du cell de l'acteur
}

int fin()
{
        EffScreen(Tout);        // efface les 3 objets du dialogue
        FreeScreen(Tout);       // détruit les objets Trièdre, PtGuide et Box
}

int dial(ref, x, y)
{
        // transformation des coordonnées du curseur (x, y) en un angle de rotation
        // mise à jour de la liste de coefficient
        // mise à jour de la liste de domaine
}

int aff(ref, mode)
{
        Si mode = ENV_AFF
                AffScreen(Tout);        // affichage des 3 objets du dialogue
        FinSi

        Si mode = ENV_UPDATE
                UpdateScreen(Tout);  // mise à jour des 3 objets
        FinSi
}
```

Les fonctions echob() et echoh() permettent de calculer les coordonnées du curseur dans le plan en fonction des transformations effectuées dans un espace 3D sur l'acteur.

4. LES ATTRIBUTS DE TRAJECTOIRE

L'apparition de plusieurs modes de description pour le même attribut constitue une nouveauté dans l'application Composer. Dans ce paragraphe, nous étudions le cas de l'attribut trajectoire. Comme nous avons pu le constater lors de la définition de nouveaux attributs, celui de trajectoire peut être décrit de 3 façons différentes pour une plus grande variété d'effets proposés à l'utilisateur.

L'attribut de translation décrite par points clés est l'attribut classique de trajectoire. Chaque point de la trajectoire est décrit par un couple de coordonnées.

L'attribut de trajectoire par shape est décrit par une liste de points. Cette liste est constituée automatiquement à partir de la description de la shape.

L'attribut de trajectoire circulaire est basé sur la rotation de l'acteur sur un cercle. Il est décrit par le centre et le rayon du cercle ainsi que par un angle de rotation sur ce cercle.

4.1 L'ATTRIBUT DE TRAJECTOIRE CLASSIQUE

Avec ce type de trajectoire, l'utilisateur maîtrise plusieurs éléments de la trajectoire :

- position d'une étape (coordonnées entre -2000 et +2000), 0 étant le centre de l'écran;
- évolution entre deux étapes fixe, linéaire ou lissée;
- direction de la trajectoire (tangente sur un point);
- vitesse d'évolution de l'acteur sur la trajectoire.

Ces éléments peuvent être mis à jour indépendamment.

Un point de coordonnées est décrit par une structure de type dhpoint :

```
typedef struct dhpoint
{
        dhermite x;
        dhermite y;
} dhpoint;
```

La structure dhermite permet le calcul de l'interpolation : a, b, c, d sont les coefficients de la spline.

```
typedef struct dhermite
{
        int flag;
        float a, b, c, d;
} dhermite;
```

a est la valeur du paramètre, b est la dérivée première de a, c est la dérivée seconde de a et d est la dérivée troisième de a. b détermine la vitesse d'évolution du paramètre. Si a est une coordonnée, b donne la direction, la tangente de la trajectoire au point a. Si a est un coefficient d'évolution, b donne la vitesse de variation de a et c donne l'accélération.

Une fonction calcule automatiquement les coefficients b, c et d de tous les points d'une liste excepté pour les points ayant un flag égal à Forcé. Dans ce cas, la valeur de b est gérée par l'utilisateur et seuls c et d sont recalculés.

La mise à jour des coordonnées détermine une valeur du coefficient pour x et y. La mise à jour de b est provoquée par la direction de la trajectoire via le panneau de menu. Le coefficient c vaut 0 lorsque l'évolution est linéaire (vitesse constante). Il est utilisé dans le calcul de l'interpolation de a lorsque l'évolution est lissée.

Positionnement d'une étape :

Une étape est créée dans la liste de coefficient et la mise à jour des cordonnées est effectuée dans la liste de domaine.

Evolution entre deux étapes :

L'interpolation saisie dans le chronogramme permet la mise à jour de l'interpolation du coefficient d'évolution. Le type lissé permet la mise à jour de la vitesse d'évolution du coefficient interdite avec le type fixe ou linéaire.

Direction de la trajectoire :

La direction de la trajectoire est mise à jour en positionnant le coefficient b des coordonnées du domaine. Cette modification est autorisée uniquement si l'interpolation de la trajectoire est lissée.

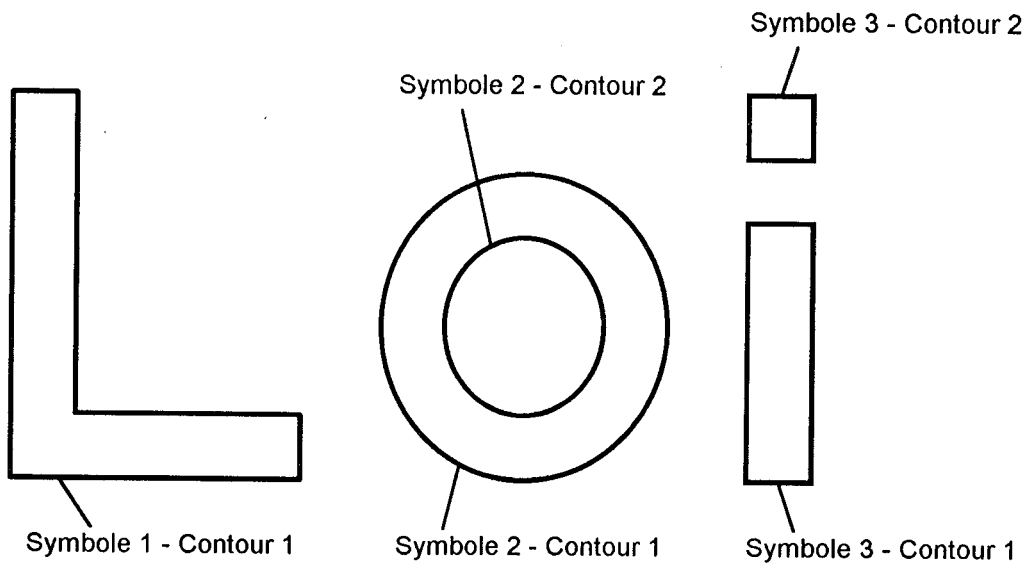
Vitesse d'évolution :

La mise à jour de la vitesse se traduit par la mise à jour du paramètre b dans la liste de coefficient d'évolution.

4.2 L'ATTRIBUT DE TRAJECTOIRE SHAPE

Une shape est construite à l'aide du logiciel de palette graphique. Elle est constituée d'un ensemble de symboles (rectangle, cercle, polygone, etc.). Une shape peut être la combinaison de plusieurs symboles. Dans le cas d'un texte vectoriel, chaque lettre est un symbole différent et les symboles sont eux même divisés en contours. Les contours peuvent être considérés comme des polygones avec un nombre important de points rapprochés les uns des autres.

Exemple : le mot Loi



La génération de la liste de points est effectuée en 3 étapes :

- analyse de la shape et génération d'un tableau intermédiaire des coordonnées à partir des points de chaque symbole de la shape;
- calcul de la longueur de la shape;
- construction de la liste à partir du tableau.

La liste de points est regénérée chaque fois que l'utilisateur précise le nom d'une shape à utiliser :

```
int Traiter_Shape(dshape *Shape, dattr *attr)
{
        // Initialiser TabPt
        Nb_Points = 0;
        Evolution = 0;
        AnalyseShape(Shape, Nb_Points, Evolution);
        CalculeLongueur(Nb_Points);
        GenereListe(Nb_Points, attr, Evolution);
}
```

4.2.1 Analyse de la shape

Les coordonnées de la shape sont décrites à l'aide de la structure dcoor :

```
typedef struct dcoor
{
        int x;
        int y;
} dcoor;
```

Le tableau des points contient les coordonnées et la longueur de la trajectoire pour le calcul du coefficient :

```
typedef struct dtabpt
{
        int x;
        int y;
        int lg;
} dtabpt;

dtabpt TabPT[1000];

int AnalyseShape(dshape *Shape, int *Nb_Points, int *Evolution)
{
        Pour chaque symbole de Shape
                Selon Type_Symbole
                        cas SYMB_RECT :
                                Evolution = LINEAIRE;
                                // PointBG = premier point du contour
                                // PointHD = second point du contour
                                Nb_Points = MajShapeRectangle(PointBG, PointHD);
                        cas SYMB_CIRCLE :
                        cas SYMB_ELL :
                                Evolution = LISSEE;
                                // Point BG = premier point du contour
                                // Point HD = second point du contour
                                Nb_Points = MajShapeCercle(PointBG, PointHD);
                        cas SYMB_POLYG :
                                Evolution = LINEAIRE;
```

```
                    Pour chaque point du contour
                        MajShapePolyg(Point, Nb_Points);
                    FinPour
                cas SYMB_CURVE :
                    Evolution = LISSEE;
                    Pour chaque point du contour
                        MajShapePolyg(Point, Nb_Points);
                    FinPour
                cas SYMB_TEXTE :
                    Evolution = LINEAIRE;
                    Pour chaque Contour du symbole
                        Pour chaque Point du Contour
                            MajShapePolyg(Point, Nb_Points);
                        FinPour
                    FinPour
            FinSelon
        FinPour
}

int MajShapeRectangle(dcoor PointBG, dcoor PointHD)
{
        // Le point bas-gauche est placé au centre de l'écran (0 ; 0).
        TabPt[0].x = 0;
        TabPt[0].y = 0;

        TabPt[1].x = PointHD.x - PointBG.x;
        TabPt[1].y = 0;

        TabPt[2].x = PointHD.x - PointBG.x;
        TabPt[2].y = PointHD.y - PointBG.y;

        TabPt[3].x = 0;
        TabPt[3].y = PointHD.y - PointBG.y;

        TabPt[4].x = 0;
        TabPt[4].y = 0;

        return(5);
]

int MajShapeCercle(dcoor PointBG, dcoor PointHD)
{
        // Le centre du cercle est placé au centre de l'écran
        Rayon_X = (PointHD.x - PointBG.x) / 2;
        Rayon_Y = (PointHD.y - PointBG.y) / 2;

        Pour I allant de 0 à 360
                TabPt[i] .x = Rayon_X * cos(i);
                TabPt[i].y = Rayon_Y * sin(i);
        FinPour
```

```
        return(361);
}

int MajShapePolyg(dcoor Point, int *Nb);
{
        TabPt[Nb].x = Point.x;
        TabPt[Nb].y = Point.y;
        Nb = Nb + 1;
}
```

4.2.2 Calcul de la longueur de la trajectoire

```
int CalculeLongueur(int Nb)
{
        // calcul de la distance parcourue à chaque point
        TabPt[0] = 0;
        Pour I variant de 1 à Nb
                // distance au point I = distance parcourue jusqu'au point I-1 + distance entre
                // les points I-1 et I
                TabPt[I].lg = TabPt[I-1].lg + DISTANCE(TabPt[I-1], TabPt[I]);
        FinPour
}
```

4.2.3 Construction de la liste de points

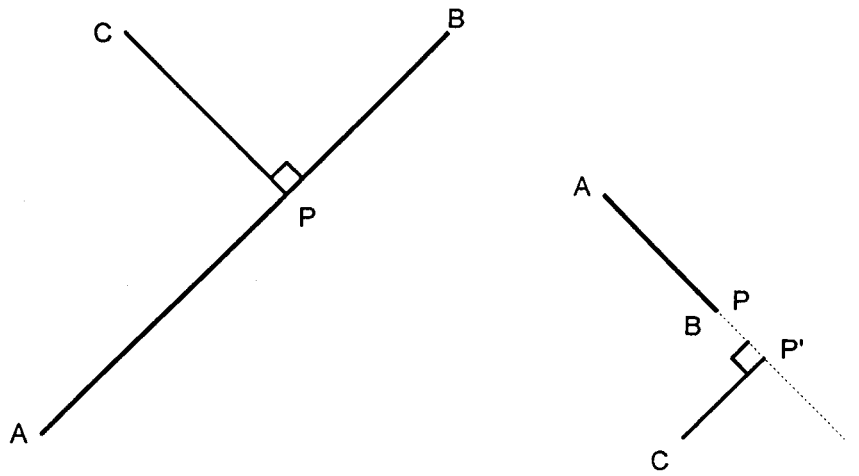
```
int GenereListe(int Nb, dattr *attr, int Evolution)
{
        // Evolution détermine le type d'interpolation de la trajectoire
        // LINEAIRE pour un polygone, un rectangle
        // LISSE pour une courbe ou un cercle
        // inserer le premier point au coefficient 0
        point_cour = TabPt[0];
        InsElement(liste_points, 0, Evolution);
        Pour I variant de 1 à Nb
                // coefficient calculé entre 0 et 1000 (SCELISTE_MAX_DUREE)
                coef = TabPt[I].lg / TabPt[Nb].lg * SCELISTE_MAX_DUREE;
                // inserer le point courant au coefficient calculé
                point_cour = TabPt[I];
                InsElement(liste_points, coef, Absent);
        FinPour
}
```

4.2.4 Le dialogue écran

La fonction de dialogue de cet attribut permet le calcul du coefficient en fonction de la position du curseur à l'écran. Cette fonction calcule le point de la trajectoire le plus proche du curseur.

Elle analyse les segments de la trajectoire :

- AB est un segment de la trajectoire;
- C est le point du curseur ;
- P est la projection orthogonale de C sur la droite passant par AB.



Lorsque P n'appartient pas au segment [AB] (cf. point P'); P est l'extrémité du segment la plus proche de C. L'algorithme recherche le point P tel que la distance (P;C) soit minimale en examinant tous les segments de la trajectoire.

4.3 L'ATTRIBUT DE TRAJECTOIRE CIRCULAIRE

La liste de domaine contient 5 informations :

- l'angle de rotation;
- les coordonnées en X et en Y du centre de la rotation;
- les rayons en X et en Y de la rotation.

La fonction de calcul prépare la translation appliquée à l'Actor :

$$Tx = Cx + Rx * \cos(rot) \qquad Ty = Cy + Ry * \sin(rot)$$

L'angle de rotation rot est exprimé en fonction d'un angle et d'un nombre de tours :

$$rot = angle + nb\_tours * 360$$

La transformation inverse permet de retrouver les valeurs de l'angle et du nombre de tours.

5. LE CALCUL DE L'ANIMATION

Le calcul de l'animation effectue l'analyse des attributs du scénario et prépare la mise à jour des modules d'Actor pour la visualisation de l'animation.

Lors de la description de l'animation, la fonction de calcul est invoquée dès qu'un paramètre est modifié (paramètre de l'attribut courant ou temps courant). L'animation est recalculée pour la trame courante afin de visualiser l'animation (mode Attitude).

La fonction de calcul est utilisée par le moteur d'animation qui prépare le calcul dans une phase spécifique avant la visualisation complète de l'animation en temps réel. Dans ce cas, le calcul est effectué pour chaque trame du scénario (mode Anim).

Ces 2 modes de fonctionnement sont transparents pour la fonction CalcTimeStory() qui calcule l'animation au numéro de trame passé en paramètre.

```
Int CalcTimeStory(dsce *sce, int tcalc)
{
    Pour chaque acteur du scénario sce
        // Initialiser les structures des Actors avec les informations par défaut.
    FinPour

    Pour chaque acteur du scénario sauf l'acteur général
        Pour chaque attribut de l'acteur
            // Lecture du coefficient courant dans la liste de coefficient d'évolution
            // Lecture du domaine courant dans la liste de domaine d'évolution
            // Appel de la fonction setactor de l'attribut
        FinPour
    FinPour

    // l'acteur général doit être traité en dernier
    Pour chaque attribut de l'acteur général
        // Lecture du coefficient courant dans la liste de coefficient d'évolution
        // Lecture du domaine courant dans la liste de domaine d'évolution
        // Appel de la fonction setactor de l'attribut
    FinPour

    Si mode_calcul = PROF_AUTO
        Pour chaque acteur du scénario sce
            // Lire la valeur de la translation en Z dans la matrice de l'acteur
        FinPour

        // Trier le tableau des profondeurs en fonction de la translation

        // Mettre à jour la profondeur des acteurs
    FinSi
}
```

5.1 COMPOSITION DES ATTRIBUTS

La fonction setactor() de chaque attribut calcule la valeur courante de l'attribut à partir des coefficients et domaines courants puis met à jour les structures de données de la couche Actor par l'intermédiaire de la fonction générique SetActor(). Cette dernière invoque une fonction de set du module Actor qui traite les informations en fonction du type d'Actor concerné.

Exemple : la fonction setactor de l'attribut Rotation en X :

```
int setactor(refer refactor, dhermite *coef, void *dom)
{
        // angle = valeur de l'angle de rotation calculé à partir de coef et dom
        SetActor(refactor, ACTOR_ROTATIONX | ACTOR_COMPO, angle);
}
```

La constante ACTOR_COMPO est utilisée pour signaler à la fonction de set que la valeur fournie en paramètre doit être composée avec la valeur déjà contenue dans la matrice de transformation.

Toutes les fonctions setactor() des attributs utilisent cette constante afin que les différents attributs se composent. C'est pour cette raison que la première étape du calcul de l'animation consiste en une remise à zéro des structures d'Actor.

Les attributs de l'acteur général se composent avec les attributs de tous les acteurs du scénario pour réaliser l'animation d'ensemble. Dans ce cas, la fonction setactor doit être capable de reconnaître l'acteur général et d'appliquer l'attribut à l'ensemble des acteurs par composition.

```
int setactor(refer refactor, dhermite *coef, void *dom)
{
        // calcul de la valeur courante de l'attribut à partir de coef et dom

        Si refactor = référence de l'acteur général
                Pour tous les acteurs du scénario sauf l'acteur général
                        refact = référence de l'Actor attaché à l'acteur
                        SetActor(refact, ACTOR_????? | ACTOR_COMPO, valeur);
                FinPour
        Sinon
                SetActor(refactor, ACTOR_????? | ACTOR_COMPO, valeur);
        FinSi
}
```

La constante ACTOR_????? doit être remplacée par la constante appropriée en fonction du type d'attribut comme :

- ACTOR_PONDINT pour la pondération interne;
- ACTOR_TRANSLATIONXY pour les attributs de trajectoire;
- ACTOR_ROTATIONX pour la rotation en X;
- ACTOR_CELL pour le numéro de cell.

Au niveau Actor, chaque module est capable de gérer les informations qu'il reçoit. Un module de type VPM n'est pas capable d'effectuer des transformations dans un espace 3D (rotation par exemple). Lorsqu'il reçoit une demande de mise à jour provenant d'un tel attribut, il ne traite pas les informations et ne réagit pas à la transformation. Ce cas peut se produire dans une animation d'ensemble, si l'acteur général possède un attribut de rotation.

5.2 PROFONDEUR AUTOMATIQUE

Le calcul automatique de la profondeur est déclenché au moment du calcul de l'animation lorsque l'attribut de profondeur n'existe pas dans le scénario.

Afin de détecter la présence ou l'absence de l'attribut de profondeur, une variable est positionnée avant le calcul de l'animation :

mode_calcul = PROF_AUTO

Cette variable indique que par défaut le calcul de la profondeur est effectué automatiquement. La fonction setactor() de l'attribut de profondeur modifie la valeur de cette variable :

mode_calcul = PROF_MANU

La fonction setactor() d'un attribut étant appelée par le calcul de l'animation uniquement si l'attribut est présent dans le scénario, ce mécanisme permet de détecter le mode de calcul de l'attribut de profondeur.

Le calcul automatique est décomposé en 3 phases (cf. algorithme de CalcTimeAnim). La première étape consiste à charger un tableau de travail contenant pour chaque acteur du scénario, la référence de l'Actor et la translation en Z de l'acteur.

```
typedef struct dinfosprof
{
        int tprof[12];              // numéros d'Actor
        double transz[12];          // translations en Z
        int nb;                     // nombre d'Actor
} dinfosprof;


dinfosprof infos;

infos.nb = 0;
Pour chaque acteur du scénario
        infos[infos.nb].tprof = refactor de l'acteur
        infos[infos.nb].transz = valeur de la translation en Z lue dans la matrice de
        transformation de l'acteur
        infos.nb = infos.nb + 1;
FinPour
```

La seconde étape consiste à trier le tableau de travail selon les translations en Z décroissantes.

```
Pour i variant de 0 à infos.nb
        Pour j variant de i+1 à infos.nb
                Si infos[j].transz > infos[i].transz
                        // permuter le contenu de infos[i].tprof avec infos.[j].tprof
                        // permuter le contenu de infos[i].transz avec infos.[j].transz
                FinSi
        FinPour
FinPour
```

La dernière étape met à jour les profondeurs à l'aide du tableau trié des références d'Actor.

```
SetActor(ACTORCONFIG, ACTOR_PROF, infos.tprof);
```

6. LES ATTRIBUTS ASSERVIS

Le nouveau format des attributs a largement contribué à la mise en place de l'asservissement. La substitution d'un coefficient par un autre est immédiate lorsque tous les coefficients ont le même format ce qui est le cas grâce à la description des attributs par fonction d'animation.

Avant d'étudier les impacts liés à la mise en place de l'asservissement, nous rappellerons les choix effectués pour la gestion des fonctions d'asservissement.

La structure de donnée d'attribut contient une chaîne de caractères (asservi). Cette zone permet de décrire une seule fonction d'asservissement par attribut dans un scénario. Il n'est pas possible d'asservir un attribut par plusieurs attributs différents. De plus, lorsqu'un attribut est asservi, il l'est pour toute la durée du scénario.

Une fonction d'asservissement standard est mise en place. Elle est utilisée lorsqu'un attribut est asservi par une fonction qui n'existe plus dans le scénario par suite de la suppression d'un attribut ou d'un acteur.

Lors du chargement d'un acteur depuis un scénario sur disque, une recherche des fonctions d'asservissement de cet acteur permet d'éviter les doublons en supprimant une des fonctions d'asservissement redondante.

La gestion des panneaux de description de l'asservissement (panneaux Master et Slave) permet d'éviter qu'un attribut asservi puisse être fonction d'asservissement et qu'un attribut déclaré en tant que fonction d'asservissement puisse être asservi.

6.1 IMPACTS DE L'ASSERVISSEMENT

6.1.1 Impacts sur la gestion des attributs

Les attributs ont été regroupés en deux catégories : les attributs de type intervalle et les attributs de type temporel.

Le domaine d'évolution des premiers permet de décrire l'intervalle d'évolution de l'attribut :

```
typedef struct dinterv
{
        float min;          // valeur minimale de l'attribut
        float max;          // valeur maximale de l'attribut
        float val;          // valeur courante
} dinterv;
```

val est la valeur courante calculée à partir des bornes et du coefficient. Cette variable est uniquement utilisée pour le stockage provisoire de la valeur.

Le coefficient d'évolution est décrit à l'aide d'une structure dhermite permettant son interpolation.

La description du domaine des attributs temporels varie en fonction du type d'attribut et du format des données propres à l'attribut : par exemple, une structure dhpoint est utilisée pour les attributs de trajectoire et de zoom.

Le format du coefficient est identique pour les deux catégories d'attributs. Pour des raisons pratiques, le coefficient varie entre 0 et 1000 (calculé entre 0 et 1 puis multiplié par 1000).

Un mode de dialogue particulier a été mis en place pour la gestion des attributs asservis et notamment pour la mise à jour du domaine d'évolution indépendamment du temps. La clé d'accès au domaine étant le coefficient, l'utilisateur doit pouvoir placer des étapes pour décrire les valeurs du domaine. Pour cela, l'utilisateur manipule un chronogramme réservé à la mise à jour du domaine. L'échelle des temps de ce scénario varie entre 0 et 1000 et correspond à la plage d'évolution du coefficient.

Exemple :

Un acteur doit se déplacer sur une droite entre les points (0 ; 0) et (500 ; 200).

Lorsque l'attribut de trajectoire n'est pas asservi, l'utilisateur crée une étape au temps 0 avec le premier point et une étape au temps Duree_Scenario avec le second point.

Lorsque cet attribut est asservi, la première étape est insérée dans la liste de domaine au coefficient 0 et la seconde au coefficient 1000. Quelque soit le coefficient utilisé; celui de l'attribut ou celui d'une fonction d'asservissement; l'utilisateur a la garantie que l'acteur restera sur cette droite.

6.1.2 Impacts sur le calcul de l'animation

6.1.2.1 Impacts sur le temps courant

Lorsque l'attribut courant n'est pas asservi, le chronogramme utilisé permet le déplacement du temps sur une échelle variant entre 0 et la durée du scénario. Le calcul de l'animation est effectué pour tous les attributs au temps courant en attitude et entre 0 et la durée du scénario en mode Anim. Pour un attribut asservi, le coefficient utilisé correspond à celui lu au temps courant dans la liste de coefficient de la fonction d'asservissement.

Lorsque l'attribut courant est asservi, le chronogramme permet l'accès au domaine et son échelle du temps varie entre 0 et 1000. Dans ce cas, le calcul de l'animation peut être effectué entre 0 et la durée du scénario pour tous les attributs en mode Anim.

En attitude, par contre, le calcul de l'animation doit lire le domaine de l'attribut courant avec un coefficient entre 0 et 1000 correspondant au coefficient courant du chronogramme. Pour tous les autres attributs, le calcul doit lire un coefficient dans la liste de coefficient et la clé d'accès à cette liste est un temps entre 0 et la durée du scénario. Dans ce cas, le coefficient utilisé doit être converti en un temps :

- au coefficient 0 correspond le temps 0;
- au coefficient 1000 correspond le temps Duree_Scenario.

La conversion du temps est effectuée au moment du calcul de l'animation.

6.1.2.2 Evaluation des fonctions d'asservissement

Le calcul de l'animation doit être précédé par une évaluation des fonctions d'asservissement. La fonction CalculAsservissement() utilise une structure de données permettant le stockage des fonctions d'asservissement :

typedef struct dasservi

{

```
        int nb;                     // nombre de fonctions d'asservissement du scénario
        char name[64][10];          // fonctions d'asservissement
        dhermite coef[64];          // coefficient de la fonction d'asservissement
} dasservi;


struct dasservi _asservi;


int CalculAsservissement(dsce *sce, int tcalc)
{
        _asservi.nb = 0;
        Pour chaque acteur du scénario
                Pour chaque attribut de l'acteur
                        Si l'attribut est une fonction d'asservissement
                                _asservi.name[_asservi.nb] = nom de la fonction d'asservissement
                                // lire la valeur du coefficient dans la liste de coefficient au temps
                                // tcalc
                                _asservi.coef[_asservi.nb] = coefficient courant;
                                _asservi.nb = _asservi.nb + 1;
                        FinSi
                FinPour
        FinPour
}
```

6.1.2.3 Evaluation des attributs asservis

Le calcul de la valeur courante de l'attribut doit tenir compte du fait que l'attribut est asservi :

- pour un attribut de type intervalle, le coefficient utilisé est celui de la fonction d'asservissement;
- pour un attribut de type temporel, le coefficient utilisé est le temps courant s'il est l'attribut courant ou le coefficient de la fonction d'asservissement sinon.

La fonction CalculCoefAsservi() permet de calculer le coefficient d'un attribut asservi à partir de la fonction d'asservissement. Cette fonction utilise la structure de données _asservi initialisée par la fonction CalculAsservissement() :

```
int CalculCoefAsservi(dattr *attr)
{
        Pour i variant de 0 à _asservi.nb
                Si _asservi.name[i] est la fonction d'asservissement de attr
                        coefficient courant de attr = _asservi.coef[i];
                        return(i);
                FinSi
        FinPour

        // La fonction d'asservissement de attr n'a pas été trouvée dans _asservi,
        // on utilise la fonction standard d'asservissement
        coefficient courant de attr = coefficient standard;
        return(Absent);
```

```
}
```

6.1.2.4 Calcul de l'animation

Les fonctions décrites précédemment sont utilisées par la fonction CalcTimeStory() dont l'algorithme doit être revu :

```
Int CalcTimeStory(dsce *sce, int tcalc)
{
        int TPS;

        Pour chaque acteur du scénario sce
                // Initialiser les structures des Actors avec les informations par défaut.
        FinPour

        Si l'attribut courant du scénario est asservi et Si le calcul est effectué en ATTITUDE
                TPS = tcalc / 1000 * Duree_Scenario;
        Sinon
                TPS = tcalc;
        FinSi

        // évaluation des fonctions d'asservissement du scénario
        CalculAsservissement(sce, TPS);

        Pour chaque acteur du scénario sauf l'acteur général
                Pour chaque attribut de l'acteur
                        CalculAttribut(attribut, tcalc);
                FinPour
        FinPour

        // l'acteur général doit être traité en dernier
        Pour chaque attribut de l'acteur général
                CalculAttribut(attribut, tcalc);
        FinPour

        Si mode_calcul = PROF_AUTO
                .....
}

Int CalculAttribut(dattr *attr, int tcalc)
{
        int TPS;

        Si l'attribut courant du scénario est asservi Et Si le calcul est effectué en ATTITUDE
                TPS = tcalc / 1000 * Duree_Scenario;
        Sinon
                TPS = tcalc;
        FinSi

        Si attr est l'attribut courant Et Si attr est de type temporel Et Si attr est asservi
```

```
            // Lecture du domaine courant dans la liste de domaine au temps tcalc
        Sinon
            Si attr est asservi
                CalculCoefAsservi(attr);
            Sinon
                // Lecture du coefficient courant dans la liste de coefficient au temps TPS
            FinSi
            // Lecture du domaine courant dans la liste de domaine au temps TPS
            // Appel de la fonction setactor de l'attribut
        FinSi
}
```

6.1.3 Impacts sur la gestion de l'application

La cohabitation de deux chronogrammes placés dans la bande basse du menu de l'application implique une gestion de l'affichage de ces deux chronogrammes. A un instant donné, un seul est visible. Lorsque l'attribut courant est changé par l'utilisateur, l'application doit afficher l'un ou l'autre des chronogrammes selon que l'attribut est asservi ou non.

Pour les attributs de type intervalle asservis, le dialogue dans les panneaux de menu et le dialogue écran doivent être invalidés. La mise à jour du domaine est effectuée par l'intermédiaire du panneau spécifique placé avec le chronogramme dans la bande basse du menu.

Le changement de chronogramme, en fonction de l'attribut courant implique un changement d'échelle du temps : entre 0 et Duree_Scenario pour les attributs non asservis et entre 0 et 1000 pour les attributs asservis. Le passage d'une échelle à l'autre est effectué par l'application lors du changement d'attribut. Le temps courant doit également est converti :

- Passage d'un attribut asservi à un attribut non asservi :

  tcour = tcour / 1000 * Duree_Scenario

- Passage d'un attribut non asservi à un attribut asservi :

  tcour = tcour / Duree_Scenario * 1000

6.2 LA GESTION DES LISTES

La mise à jour de la liste du coefficient et de la liste du domaine d'évolution est effectuée lorsque l'utilisateur place, enlève une étape sur le chronogramme ou lorsque les paramètres d'une étape sont modifiés. Le fait de disposer de 2 listes séparées offre l'avantage de permettre la mise à jour séparée du domaine et du coefficient.

Pour un attribut de type intervalle, le coefficient courant permet le calcul de la valeur courante de l'attribut. Si l'attribut est asservi, le coefficient utilisé est celui de la fonction d'asservissement.

L'accès au domaine d'évolution est effectué en parallèle avec l'accès au coefficient. Les deux listes sont lues avec la même clé d'accès c'est à dire le temps courant. Le calcul de la valeur est effectué ensuite.

Pour un attribut de type temporel, le coefficient lu dans la liste de coefficient au temps courant est utilisé comme clé d'accès à la liste de domaine pour lire le domaine courant d'où est

extraite la valeur courante de l'attribut. Si l'attribut est asservi, le coefficient n'est pas lu dans la liste de coefficient mais est fourni par la fonction d'asservissement.

Afin de gérer tous ces cas de façon quasi transparente pour l'application, une encapsulation du module de gestion des listes peut être écrite.

6.2.1 Le module de gestion des listes

Les principales fonctions décrites dans le module de listes sont :

```
void *InsElement(liste *l, int t, int evol)
{
        // insertion d'une étape dans la liste au temps t avec le type d'évolution evol
}

int DelElement(liste *l, int t)
{
        // suppression de l'étape de la liste au temps t
}

void *MajElement(liste *l, int t, int evol)
{
        // mise à jour de l'étape de la liste au temps t : mise à jour des informations de l'étape
        // et / ou du type d'évolution
}

int LoadElement(liste *l, int t, int *evol)
{
        // Lecture des informations de l'étape dans la liste au temps t
}
```

6.2.2 L'encapsulation du module de listes

Ce module doit permettre l'accès aux listes de coefficient et de domaine soit individuellement soit ensembles. Pour cela, le module SCELISTE prévoit 4 modes de fonctionnement :

Le mode domaine (cas SCELISTE_DOM) permet la mise à jour de la liste de domaine indépendamment du coefficient. L'indice d'accès de l'élément dans la liste est un coefficient fourni en paramètre.

Le mode coefficient (cas SCELISTE_COEF) permet la mise à jour de la liste de coefficient sans impact pour la liste de domaine. Ce mode est utilisé pour les attributs dont le domaine est invariant (cas de la trajectoire par shape : la liste de points du domaine n'est pas modifiée par la mise à jour d'un coefficient).

Avec le mode mixte (cas SCELISTE_TOUT) la mise à jour ou la consultation de la liste de coefficient est répercutée en parallèle dans la liste de domaine. La fonction calcule le coefficient de l'étape et utilise ce coefficient pour la mise à jour ou la consultation de la liste de domaine.

Le mode automatique (cas Absent) réagit en fonction du contexte au moment de l'accès aux listes :

- si l'attribut n'est pas asservi, le mode mixte est utilisé;
- si l'attribut est asservi et qu'il est de type temporel, le mode domaine est utilisé;
- si l'attribut est asservi et qu'il est de type intervalle, le mode utilisé est fonction de l'opération à effectuer dans la liste :
  - si l'opération est une insertion, une suppression ou une mise à jour de la liste alors aucune commande ne doit être effectué sur la liste;
  - si l'opération est une lecture de la liste alors le mode utilisé est le mode mixte.

La constante SCELISTE_MAX_DUREE détermine la valeur maximale du coefficient (égale à 1000). Les fonctions ajoutées dans le module SCELISTE sont au nombre de 4.

Insertion d'un élément dans les listes :

```
void *SceInsElem(int cas, dattr *attr, int t, int evol)
{
        int mode;

        mode = cas;
        Si cas = Absent Et Si attr est asservi
                Si attr est de type intervalle
                        return;
                FinSi
                Si attr est de type temporel
                        mode = SCELISTE_DOM;
                FinSi
        FinSi

        Selon mode
                Cas SCELISTE_DOM :
                        InsElement(attr->list_dom, t, evol);

                Cas SCELISTE_COEF :
                        Si attr est de type intervalle
                                // calculer le coefficient en fonction du domaine courant
                                // et de la valeur courante de l'attribut
                        FinSi
                        InsElement(attr->list_coef, t, evol);

                Cas SCELISTE_TOUT:
                        Si attr est de type intervalle
                                // calculer le coefficient en fonction du domaine courant
                                // et de la valeur courante de l'attribut
                        Sinon
                                attr->coef_cour = t / Duree_Scenario * SCELISTE_MAX_DUREE;
                                InsElement(attr->list_dom, attr->coef_cour, Absent);
                        FinSi
                        InsElement(attr->list_coef, t, evol);
```

```
        FinSelon;
}
```

Suppression d'un élément des listes :

```
int SceDelElem(int cas, dattr *attr, int t)
{
        int mode;

        mode = cas;
        Si cas = Absent Et Si attr est asservi
                Si attr est de type intervalle
                        return;
                FinSi
                Si attr est de type temporel
                        mode = SCELISTE_DOM;
                FinSi
        FinSi

        Selon mode
                Cas SCELISTE_DOM :
                        DelElement(attr->list_dom, t);

                Cas SCELISTE_COEF :
                        DelElement(attr->list_coef, t);

                Cas SCELISTE_TOUT:
                        Si attr est de type temporel
                                attr->coef_cour = t / Duree_Scenario * SCELISTE_MAX_DUREE;
                                DelElement(attr->list_dom, attr->coef_cour);
                        FinSi
                        DelElement(attr->list_coef, t);
        FinSelon;
}
```

Mise à jour d'un élément des listes :

```
void *SceMajElem(int cas, dattr *attr, int t, int evol)
{
        int mode;

        mode = cas;
        Si cas = Absent Et Si attr est asservi
                Si attr est de type intervalle
                        return;
                FinSi
                Si attr est de type temporel
                        mode = SCELISTE_DOM;
                FinSi
        FinSi
```

```
        Selon mode
            Cas SCELISTE_DOM :
                MajElement(attr->list_dom, t, evol);

            Cas SCELISTE_COEF :
                Si attr est de type intervalle
                    // calculer le coefficient en fonction du domaine courant
                    // et de la valeur courante de l'attribut
                FinSi
                MajElement(attr->list_coef, t, evol);

            Cas SCELISTE_TOUT:
                Si attr est de type intervalle
                    // calculer le coefficient en fonction du domaine courant
                    // et de la valeur courante de l'attribut
                Sinon
                    attr->coef_cour = t / Duree_Scenario * SCELISTE_MAX_DUREE;
                    MajElement(attr->list_dom, attr->coef_cour, Absent);
                FinSi
                MajElement(attr->list_coef, t, evol);
        FinSelon;
}
```

Lecture d'un élément des listes :

```
int SceLoadElem(int cas, dattr *attr, int t, int *evol)
{
        int mode;

        mode = cas;
        Si cas = Absent
            mode = SCELISTE_TOUT;
            Si attr est asservi Et Si attr est de type temporel
                mode = SCELISTE_DOM;
            FinSi
        FinSi

        Selon mode
            Cas SCELISTE_DOM :
                LoadElement(attr->list_dom, t, evol);

            Cas SCELISTE_COEF :
                LoadElement(attr->list_coef, t, evol);

            Cas SCELISTE_TOUT:
                LoadElement(attr->list_coef, t, evol);
                LoadElement(attr->list_dom, attr->coef_cour, evol);
        FinSelon;
}
```

CONCLUSION

Avant de conclure, prenons un exemple simple qui illustre bien l'avancée technologique réalisée grâce à l'apparition du tout nouveau système de synthèse d'images Hurricane équipé des modules DVE-Layer et du logiciel DVE-Composer.

Un utilisateur possède un système Studio Venice avec le logiciel Sequencer. Il désire réaliser une petite animation d'une seconde soit 50 trames montrant une des roues d'un engrenage dessinée dans le plan de l'écran et effectuant un tour complet.

Plusieurs étapes sont nécessaires à la réalisation de son animation :

- chargement du dessin de l'engrenage sur un plan mémoire;
- création d'un programme de géométrie contenant une rotation autour de l'axe des Z;
- mise à jour des paramètres de la rotation pour effectuer un tour;
- le scénario est au point, il génère la séquence comprenant 50 cells résultants de la transformation du dessin;
- pour que la qualité du résultat soit bonne, il demande des calculs au sous-pixel ($1/16^{ème}$) et de l'antialiasing;
- la génération de la séquence nécessite un temps de calcul de plusieurs secondes par trame;
- il peut maintenant visualiser sa séquence en temps réel, séquence qui a été enregistrée sur Aramis.

Notre utilisateur vient d'équiper son système d'un module DVE-Layer. Il a acquis par la même occasion le nouveau logiciel d'animation DVE-Composer.

Il renouvelle son animation avec l'aide de ses nouveaux outils :

- chargement du dessin de l'engrenage sur le DVE-Layer;
- mise à jour de l'attribut de rotation en Z pour effectuer un tour;
- visualisation de son animation en temps réel et après moins d'une seconde de calcul.

En 1987, Anim, le premier logiciel d'animation équipant les systèmes Getris permet l'animation d'acteurs 2D en temps réel ce qui constitue l'originalité de ce système par rapport aux logiciels concurrents de l'époque qui construisent l'animation suivant la méthode dite de « l'image par image ».

Par la suite, la réalisation des transformations d'acteurs 2D dans un espace 3D a mobilisé l'énergie des ingénieurs de Getris Images. Ce n'a pas été en vain puisque de ces recherches est né en 1992, le Sequencer, un logiciel d'animation dans un espace 3D image par image; mais ce n'était pas suffisant. Il fallait maintenant effectuer ces opérations en temps réel.

Le DVE-Composer a parfaitement combiné les techniques employées par ses prédécesseurs et l'évolution technologique des cartes électroniques équipant les systèmes Getris. En effet, étant donnée la puissance actuelle des calculateurs, seuls des opérateurs câblés sont capables de

réaliser les nombreux calculs nécessaires à la réalisation d'effets d'animation complexes en temps réel et en 3D.

J'ai étudié une nouvelle approche de l'animation qui a abouti à la réalisation d'un nouveau mécanisme de gestion des attributs pour l'application d'animation DVE-Composer. Cette étude m'a été confiée dans le cadre de mon mémoire d'ingénieur CNAM. Cette expérience a été pour moi l'occasion de réaliser un souhait : changer complètement de domaine d'application au moins pendant une année. C'est ainsi que j'ai plongé dans le monde de la vidéographie et des images de synthèse.

Le DVE-Composer est issu de l'expérience acquise dans deux applications précédentes et est un compromis entre la simplicité de l'Anim et la complexité du Sequencer. Il profite du nouveau système d'exploitation DOS-Extender qui ôte toute limitation dans la taille du code développé.

Les attributs sont regroupés en classes en fonction de leur type. A l'intérieur d'une classe, les attributs se composent selon un ordre bien précis. La composition a été rendue nécessaire par la présence d'attributs divers comme les rotations, les translations ou le zoom. L'ordre des attributs est défini par la position d'insertion de l'attribut dans la liste des attributs d'un acteur. Ce mécanisme est plus facile à comprendre que celui qui est proposé par le Sequencer. Il offre la possibilité d'effectuer des animations simples très rapidement. L'augmentation du nombre des attributs permet d'obtenir des effets beaucoup plus spectaculaires. Il est surtout à la base de l'animation d'ensemble. Il n'est plus utile maintenant de décrire un attribut identique pour tous les acteurs lorsque celui-ci doit s'appliquer à l'ensemble de la scène.

Jusqu'à présent aucun mécanisme n'était prévu pour lier l'évolution de certains attributs. L'utilisateur avait la charge de gérer l'évolution de ses attributs en parallèle pour leur donner la même évolution. La mise à jour d'un des attributs provoque aussitôt une désynchronisation de l'attribut par rapport aux autres. Avec l'asservissement, l'utilisateur définit l'évolution d'un attribut du premier acteur. Il peut ensuite indiquer au système que le même attribut du second acteur évolue de manière identique, puis que ce même attribut sur le troisième acteur évolue en sens inverse. Le nombre des manipulations est grandement diminué lorsqu'il s'agit de mettre à jour les valeurs de l'attribut servant de base. Le système reporte de lui-même toute nouvelle valeur sur les attributs asservis.

Le nouveau mécanisme de gestion des attributs a permis d'obtenir une application très extensible dans la mesure où des moules de programmation ont été mis en place. Les trois modules permettant la gestion d'un attribut sont structurés et l'ajout d'un nouvel attribut peut aisément s'effectuer par duplication d'un attribut existant.

Les différents modes de description d'un même attribut permettent d'une part une utilisation simplifiée pour l'utilisateur amateur et d'autre part donne libre court à la création pour un professionnel. L'effet de vague, en cours de développement et qui sera introduit dans la prochaine version de l'application permet l'ajout de deux attributs :

- L'attribut de drapeau, où l'utilisateur fixe des paramètres concrets et facile à cerner comme la force et la direction du vent. Pour cet attribut, le système détermine tous les paramètres de l'effet.
- L'attribut de vague, est identique au précédent mais l'utilisateur fixe tous les paramètres soit l'amplitude, la période, la longueur d'onde, le point de départ de l'onde et le rayon d'action.

La description des attributs à l'aide d'une fonction d'animation et le stockage des coefficients d'évolution et du domaine dans deux listes indépendantes ont supprimé le problème lié à la mise à jour de la vitesse de variation du coefficient. Pour l'attribut de translation, un amorti ou la variation de la vitesse de déplacement de l'acteur n'entraîne plus de déformation de la trajectoire.

La liste des attributs a été allongée grâce à de multiples modes de description du même attribut. Ainsi l'attribut de trajectoire par point clé a été complété par deux nouveaux attributs : la trajectoire par shape et la trajectoire circulaire. Le premier permet d'obtenir des trajectoires bien plus complexes sans difficulté (acteur se déplaçant sur un texte en suivant les contours des lettres). Des trajectoires en forme de spirale sont réalisables facilement grâce au second. Avec ces deux attributs, l'acteur peut se déplacer dans un sens ou dans l'autre sur la trajectoire, revenir en arrière, chose qui n'est pas possible avec l'attribut de trajectoire classique.

Le DVE-Composer est un logiciel d'animation très complet. Le nombre des attributs d'animation est très important et en constante évolution. Les algorithmes utilisés doivent sans cesse être optimisés pour accélérer les temps de calcul. Après quelques mois d'utilisation, les retours des graphistes ayant utilisé cette application sont favorables mais quelques limites du système apparaissent.

Le mécanisme de base de gestion des attributs est extensible et général mais beaucoup plus lent que l'Anim. La manipulation des trajectoires reste lente bien que les algorithmes utilisés soient sans cesse optimisés pour accélérer les temps de calcul. La lenteur du système est également liée au nombre important de multiplications effectuées sur des valeurs flottantes car la composition des attributs utilise des matrices. Une étude est en cours pour utiliser le système DOS-Extender 32 bits afin de réduire les temps de calcul (optimisation du code, opérations d'accès à la mémoire en un seul cycle, etc.).

L'attribut de profondeur automatique libère le graphiste de la manipulation des priorités des acteurs. L'inconvénient de cet attribut est que tous les acteurs sont logés à la même enseigne. Il n'est pas possible actuellement de définir la liste des acteurs qui ont une profondeur calculée automatiquement, les autres étant gérés manuellement. Pour cela, il est possible d'ajouter une colonne supplémentaire à l'attribut de profondeur manuelle permettant de placer les attributs dont la profondeur doit être calculée automatiquement.

De nombreuses évolutions comme celle-ci sont possibles dans cette application. La mise en place d'un nouveau mécanisme est en général limité par l'interface graphique de dialogue qui doit rester claire, facile à appréhender. L'échec d'un produit ou d'une évolution est souvent lié au rejet de l'interface homme-machine par les graphistes qui ne souhaitent pas se poser des questions au cours de leurs manipulations.

Lors de la description d'une animation d'ensemble, les graphistes veulent manipuler des groupes d'acteurs afin de limiter l'action des attributs à quelques acteurs. La solution permettant d'obtenir une arborescence du scénario organisé en scènes regroupant des acteurs a été écartée lors de la conception de l'application. C'est une bonne chose que les graphistes se soient aperçus d'eux même qu'il faut des groupes d'acteurs. Ils sont prêts à accepter une interface un peu plus complexe et permettant de résoudre ce problème. Si ce mécanisme avait été proposé dès le début, il risquait d'être mal accepté.

L'asservissement des attributs reste un concept difficile à appréhender. Pour cette raison, il n'a pas été intégré dans l'application et reste à l'état d'étude. Un prototype est mis au point afin d'améliorer l'interface homme-machine qui n'est pas suffisamment simple et naturelle et qui ne peut pas être proposé aux graphistes en l'état.

J'ai réalisé cette étude dans le cadre d'un projet majeur dont l'aboutissement est la production d'un nouveau logiciel d'animation. Cette expérience a été très valorisante pour plusieurs raisons :

- elle m'a permis de découvrir un nouveau domaine d'application;
- elle m'a permis d'exprimer mes idées, d'intervenir dans la conception d'une application et de montrer mes capacités à réaliser un projet;
- le système est utilisé par les graphistes de Getris Images et les démonstrations que j'ai pu voir sont époustouflantes;

- l'application DVE-Composer remplace désormais le module Anim et le système Hurricane équipé avec cette application est commercialisé dans le monde entier; une machine a déjà été vendue en Russie et en Asie du Sud-Est.

ANNEXES

SONOVISION HEBDO

LUNDI 9 MAI 1994

L'ART DE COUPER LES PIXELS EN 32

Getris Images a présenté sa nouvelle station infographique haut de gamme "Hurricane" à sa clientèle parisienne au début de la semaine dernière après une première mondiale à la NAB 94 qui lui a déjà permis de faire trois ventes aux USA. "Cette nouvelle machine va encore plus loin dans le concept du multicouche cher à Getris, a indiqué Antoine Pane, directeur marketing de la firme grenobloise, puisque chaque plan fonctionne comme un générateur d'effets indépendant, en temps réel et avec une extrême précision, chaque pixel etant subdivisé en 32 éléments." Système dédié, Hurricane est, selon Getris, plus performant que les solutions purement logicielles tout en restant ouvert grâce à son interface de supervision PC et aux fichiers à la norme TGA qui peuvent être échangés avec d'autres plates-formes. Actuellement la station est capable de réaliser des perspectives, des zooms ou des trajectoires, d'autres effets comme le drapeau ou le morphing étant attendus pour bientôt. Une option "DVE multicouche" est aussi disponible pour s'intégrer dans des stations Eclipse ou Venice. Selon Antoine Pane, Getris enregistre des succès significatifs à l'export en particulier au Moyen-Orient, en Amérique du Sud (Brésil, Argentine) en Asie, à travers sa filiale basée à Singapour, ainsi qu'en Europe de l'Est où dix unités ont été vendues. Le marché français, essentiellement représenté par les équipements infographiques pour les réseaux câblés, commence à se réveiller, notamment après la commande de TF1 pour "La Chaine Info".

ECRAN TOTAL

1 - 7 JUIN 1994

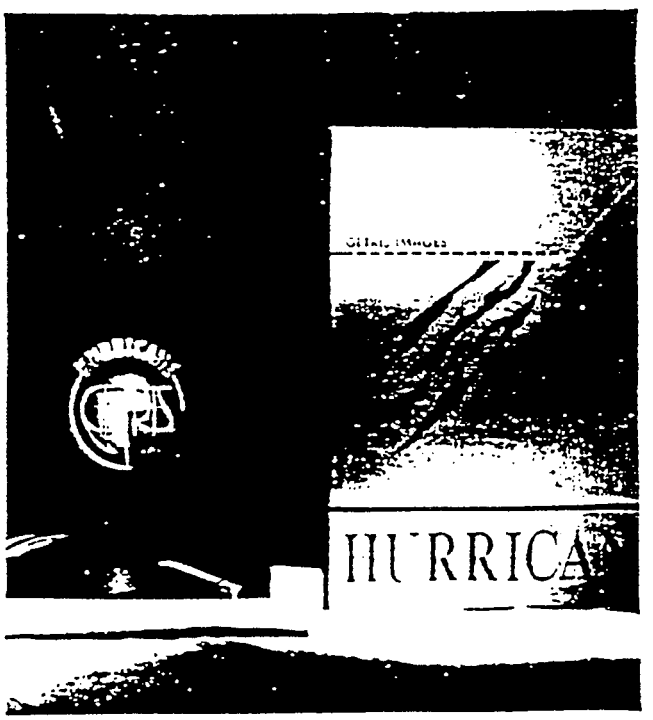
La station Hurricane de Getris : quel souffle !

La société Getris vient de présenter sa nouvelle station qui ajoute des possibilités d'effets spéciaux numériques (DVE Digital Video Effects) temps réel aux spécifications (Palette, Animation, Rotoscoping) en mode de travail multicouche, déjà proposées par Venice et Eclipse, les précédentes machines de la société. Il s'agit d'un système dédié (chaque couche est traitée par un processeur spécialisé) plus efficace et plus rapide qu'une solution purement logicielle, pouvant manipuler de quatre à dix plans, chaque plan disposant d'un générateur d'effets indépendant.

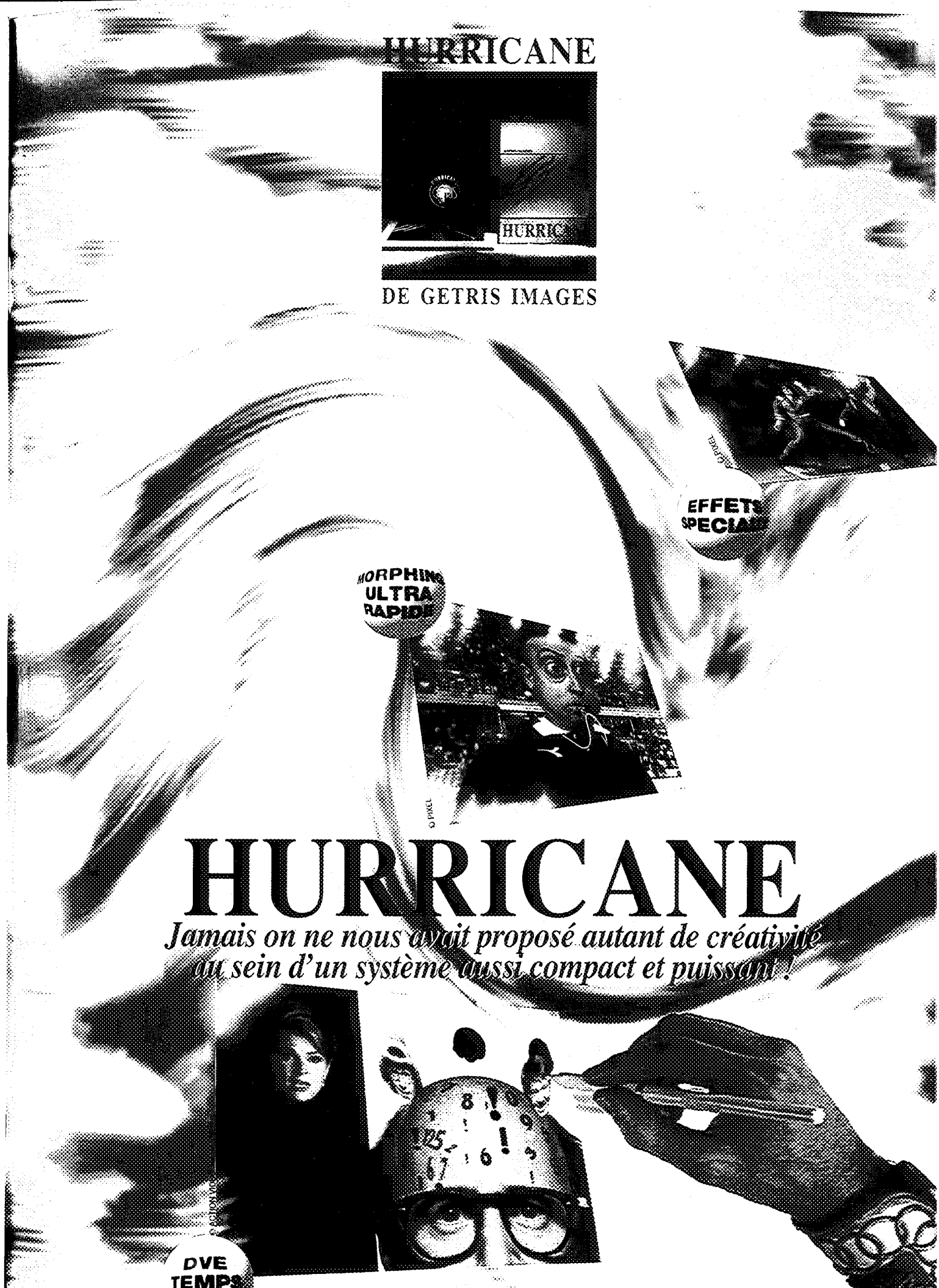
Les effets sont traités avec une extrême résolution puisque le système génère par interpolation 32 "sous-pixel" pour chaque pixel de l'image ; des zooms de rapport 32 peuvent ainsi être réalisés sans altération de la qualité. Les effets spéciaux actuellement implantés comprennent perspective, zoom et trajectoires, mais le logiciel sera rapidement abondé pour des effets de vagues, de drapeau, de morph... Ce sont 3 000 objets qui peuvent être manipulés indépendamment.

La démonstration qu'il nous a été donné de suivre est tout simplement étonnante de virtuosité, de rapidité et, apparemment, de simplicité. Présenté en avant-première au NAB, Hurricane a immédiatement été commandé en trois exemplaires aux Etats-Unis. Lors de cette présentation, Antoine Patte, directeur marketing de Getris, a indiqué que sa société se portait bien. Celle-ci continue son développement à l'export en ouvrant une filiale à Singapour et enregistre des résultats intéressants aussi bien au Moyen-Orient (6 machines vendues) qu'en Amérique du Sud ou dans les pays de l'Est (10 machines vendues). ■

François Luxereau



La station Hurricane traite les effets spéciaux avec une extrême résolution.


HURRICANE
DE GETRIS IMAGES
EFFETS SPECIAUX
MORPHING ULTRA RAPIDE
HURRICANE
Jamais on ne nous avait proposé autant de créativité au sein d'un système aussi compact et puissant !
DVE TEMPS

[MAR_82] Martinez F.
Vers une approche systématique de la synthèse d'images. Aspects logiciels et matériels. Thèse de doctorat d'état. Institut National Polytechnique de Grenoble. Nov 1982.

[MAR_84] Martinez F.
La synthèse d'images : concepts, matériels et logiciels. Edi Tests. 1984.

[PER_85] Perisic Z.
La prise de vue en animation. Editions Dujarric. 1985.

[PER_88] Peroche B., Argence J., Ghazanfarpour D., Michelucci D.
La synthèse d'images. Traité des nouvelles technologies. Assistance par ordinateur. Editions Hermès. 1988.

[PIN_91] Pinel V.
Techniques du cinéma. Que sais-je ?. Presses universitaires de France. 1991.

[QUA_89] Quantel.
Documentation commerciale : PaintBox. 1989.

[S&M_107] Soft & Micros.
Essais : Action 2.5.1. N°107, p 146, mai 1994.

[S&M_109] Soft & Micros.
Les outils du multimédia. N°109, p 52-60, juillet-août 1994.

[SCH_87] Schweizer P.
Infographie. Volumes 1 et 2. Presses polytechniques romandes. 1987.

[SGI_93] SiliconGraphics.
Documentation commerciale : La famille INDIGO. 1993.

[SOF_93] SoftImages.
Documentation commerciale : Animation Studio. 1993.

[SYM_90] Symbolics.
Documentation commerciale : S-PAINT, S-GEOMETRY, S-DYNAMICS, S-RENDER. 1990.

[TOU_87] Touchard J.B.
Images numériques. Edition Cedic / Nathan. 1987.

HURRICANE
de Getris Images

HURRICANE, c'est :

- des plans DVE multiples - temps réel, sur base matérielle dédiée... jusqu'à 10 plans DVE
- une précision au sous-pixel - pour une qualité d'effets optimale
- des effets 3D instantanés - des effets immédiats qui ne nécessitent pas de rendu. Plus de temps perdu à attendre que les images se calculent
- un morphing ultra-rapide - ainsi que du cross-morphing et des métamorphoses de formes et de polices de caractères
- des filtres - effets givre, BD, glace, aquarelle... et autres
- un chroma Key.

HURRICANE révolutionne les effets spéciaux en les exécutant en temps réel. En sortant de la logique image par image, HURRICANE permet de maximiser ces effets et, grâce aux capacités multi-couches du système, d'en programmer simultanément jusqu'à 10 en temps réel.

Toute la puissance d'HURRICANE procède d'un nouveau concept : celui de cartes électroniques individualisées qui permettent d'animer des effets digitaux.
Chaque carte (ou plan) au sein d'HURRICANE fonctionne comme un DVE... aussi bien que comme un plan graphique ou plan d'animation. Chaque plan peut être au choix un plan graphique, un plan vidéo live, être découpé en cellos pour une animation, un plan vidéo retouché, etc... et mélangé avec tous les autres plans du système.

Le résultat en est un nouveau concept, dynamique et puissant : le DVE multi-couches.

HURRICANE est un système graphique haut-de-gamme réunissant Palette, Animation, Rotoscoping, Compositing et EFFETS SPECIAUX.

A ce jour, le plus puissant des produits Getris Images, HURRICANE complète parfaitement les gammes ECLIPSE et VENICE en augmentant leurs fonctionnalités et leur puissance grâce :
- au multi-couches temps réel
- à l'introduction de plans DVE
- aux effets spéciaux.

Les Modules HURRICANE

Développé pour répondre aux demandes les plus en pointe des marchés de la production et de la post-production, HURRICANE est l'outil le plus adapté à la création d'effets spéciaux pour la publicité, l'habillage, les génériques... et la liste est loin d'être exhaustive !
Les caractéristiques d'HURRICANE se déclinent dans les modules logiciels suivants :

PAINT & EFFECTS

Ce module intègre tous les outils palette les plus sophistiqués et les plus malins pour créer et retoucher des images : brosse, aérographe, texture, morphing, effet givre, déformation, effet drapeau, chroma key... qui, associés à des macro-commandes interactives, permettent d'automatiser les tâches répétitives et donc d'accélérer les processus créatifs.

DVE ANIM

Ce module permet de manipuler en temps réel et simultanément jusqu'à 10 acteurs indépendants avec interpolation automatique entre des points clés, et ce à la trame ou à l'image. Le module d'animation DVE permet en même temps de générer des effets instantanés sur chaque plan/acteur comme des trajectoires courbes au sous-pixel, des changements de profondeur, des zooms, des flous, des filtres, des rotations, des mises en perspective, des fondus, des volets, des grilles, des déroulants, des mosaïques, des changements de transparence, etc...

DVE COMPOSER

Ce module ajoute aux possibilités du précédent module le contrôle de l'option matérielle ARAMIS et permet d'éditer des séquences, d'enregistrer et de lire vidéo/ou image et key en temps réel, de mélanger plusieurs séquences live-sur-live... et donc d'obtenir un compositing et des effets superbes.

L'incroyable puissance d'HURRICANE provient essentiellement du concept matériel totalement innovant des cartes électroniques, chaque plan DVE étant considéré comme un véritable DVE indépendant, capable de créer en temps réel au sous-pixel n'importe quel type de déformation au sein des tout mouveaux modules d'animation DVE et de compositing.

HURRICANE apporte donc une nouvelle dimension à la création graphique avec une formidable qualité d'image, en proposant toutes ces caractéristiques pour la première fois véritablement en temps réel, ce qui permet une visualisation et des modifications immédiates, sans temps de rendu.

GETRIS IMAGES
Siège social : 23, chemin des Prés - BP 172 - 38244 Meylan Cedex - Tél. 76 90 87 77

BIBLIOGRAPHIE

[BEN_85] Bendazzi G.
Le Film d'animation. La Pensée Sauvage/JICA. 1985.

[BRU_89] Bruneaux S.
Etude des concepts logiciels et matériels pour un système interactif d'animation en temps réel. Thèse de Doctorat en informatique. 1989.

[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.

[CHA_92] Chalvin P.
Etude et réalisation d'une application d'animation de dessin et d'animation géométrique image par image. Mémoire d'ingénieur CNAM. 1992.

[COL_92] Colorgraphics.
Documentation commerciale. 1992.

[GET_89] Getris Images.
G_Dial. Bibliothèque graphique de dialogue. Manuel de programmation V1.2, 1989.

[GET_90] Getris Images.
G_LIB. Bibliothèque graphique de base. Manuel de programmation V2.1, 1990.

[GET_92] Getris Images.
Anim. Real Time Animation Software. Reference Manual. 1992.

[GET_93] Getris Images.
Sequencer. Frame by Frame Animation and Drawing. User Guide. 1993.

[GET_94A] Getris Images.
DVE-Composer. Multi-DVE Layer Animation. Reference Manual. 1994.

[GET_94B] Getris Images.
Videographic Workstations. User Guide. 1994.

[MAC_94] Macromédia.
Documentation commerciale : Director 4.0 for Macintosh. 1994.

[MAR_77] Martinez F.
Etude des problèmes de conception et de réalisation d'animation : le système SAFRAN. Thèse de Docteur de 3ème cycle, Institut National Polytechnique de Grenoble. Mai 1977.