PROPOSITION

TECHNIQUES D'ANIMATION GRAPHIQUE PAR ORDINATEUR

-- (Phase I) --

préparée pour:

MINISTÈRE DES COMMUNICATIONS DU CANADA
Centre de Recherches en Communications
Highway 17B, Shirley Bay
Ottawa (Ontario)

Mars 1984 Projet CDT P825

CDT
Centre de Développement Technologique

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

PROPOSITION

TECHNIQUES D'ANIMATION GRAPHIQUE PAR ORDINATEUR

-- (Phase I) --

préparée pour:

MINISTÈRE DES COMMUNICATIONS DU CANADA
Centre de Recherches en Communications
Highway 17B, Shirley Bay
Ottawa (Ontario)

Mars 1984 Projet CDT P825

DOC CONTRACTOR REPORT

DOC-CR-TS-84-0042
(section has given this number to report)

DEPARTMENT OF COMMUNICATIONS - OTTAWA - CANADA

INFORMATION TECHNOLOGY AND SYSTEMS
RESEARCH AND DEVELOPMENT

TITLE: TECHNIQUES D'ANIMATION GRAPHIQUE PAR ORDINATEUR (PHASE I)

AUTHOR(S): DR. PAUL J. COHEN, MR. STEPHANE METZ, MR. ALAIN PERRET,
MR. PIERRE ROUSSEAU, DR. HAI HOC HOANG

ISSUED BY CONTRACTOR AS REPORT NO: CDT P825

CONTRACTOR: LE CENTRE DE DEVELOPPEMENT TECHNOLOGIQUE DE L'ECOLE POLYTECHNIQUE
DE MONTREAL
Campus de l'Université de Montréal
Case postale 6079, Succursale A
Montréal (Québec) H3C 3A7

DEPARTMENT OF SUPPLY AND SERVICES CONTRACT NO: OST83-00049

DOC REQUISITION NO: 36100-3-0014

DOC SCIENTIFIC AUTHORITY: MICHELE GUILLET
INFORMATION TECHNOLOGY & SYSTEMS R&D

CLASSIFICATION: UNCLASSIFIED

DATE: MARCH 1984

COMMUNICATIONS RESEARCH CENTRE

RELEASABLE

DOC-CR-TS-84-001

ORIGINAL DOCUMENT REVIEW AND PUBLICATION RECORD

SECTOR	BRANCH	DATE
DGTS	DIP	March 4, 1987

PURPOSE This form is for use during review of the DOC-CR contractor reports. It is designed to: record decisions for classification, record reasons for classification and cautionary mark provide for indexing requirements.

INSTRUCTIONS * 1 copy of the completed form must accompany the contractor report package submitted to the CRC Library.

* Complete the following items as applicable.

1. DOC-CR NO. DOC-CR-TS-84-001	2. DSS CONTRACT NO. 36100-3-0014
3. TITLE: Techniques d'animation graphique par ordinateur : phase I : proposition	4. DATE March 1984
5. CONTRACTOR Ecole Polytechnique de Montréal	
6. SCIENTIFIC AUTHORITY M. Guillet	7. LOCATION DIP/CRC
	8. TEL. NO. 998-2384

9. CONTRACTOR REPORT CLASSIFICATION:

RELEASABLE [XXX] CONDITIONALLY RELEASABLE [ ] NON-RELEASABLE [ ]

* REASONS FOR CLASSIFICATION:

10. NO. OF COPIES SUBMITTED TO LIBRARY:

EXECUTIVE SUMMARY [ ] FINAL REPORT [XXX] one copy

+ Xerox copy

....

Scientific Authority's Signature Date

This form is not official therefore it is not signed.

CDT
Centre de Développement Technologique
École Polytechnique de Montréal

PROJET CDT P825

PROPOSITION

TECHNIQUES D'ANIMATION GRAPHIQUE PAR ORDINATEUR

-- (PHASE I) --

préparée pour:

MINISTÈRE DES COMMUNICATIONS DU CANADA
Centre de Recherches en Communications
Highway 17B, Shirley Bay
Ottawa (Ontario)

par:

Dr Paul J. Cohen
Mr Stephane Metz
Mr Alain Perret
Mr Pierre Rousseau
Dr Hai Hoc Hoang
Département de génie électrique
ÉCOLE POLYTECHNIQUE DE MONTREAL

soumise par:

LE CENTRE DE DÉVELOPPEMENT TECHNOLOGIQUE
DE L'ÉCOLE POLYTECHNIQUE DE MONTREAL
Campus de l'Université de Montréal
Case postale 6079, Succursale A
Montréal (Québec) H3C 3A7

Mars 1984

Dr Hai H. Hoang et Dr Paul Cohen
Co-Responsables du Projet

TABLE DES MATIÈRES

LISTE DES FIGURES

(suite) -

LISTE DES FIGURES

RÉSUMÉ

L'animation constitue une technique permettant de donner l'illusion du mouvement à partir d'une série d'images fixes. De ce fait, l'utilisation de procédés d'animation dans les communications visuelles peut permettre d'améliorer considérablement la qualité et l'efficacité de celles-ci.

Dans leurs formes traditionnelles, les techniques d'animation sont extrêmement laborieuses et coûteuses. L'utilisation d'un ordinateur dans différentes phases du processus d'animation permet d'accélérer considérablement les temps de production et d'en diminuer les coûts.

Malgré la diminution des coûts du matériel informatique, "l'Animation par Ordinateur" implique encore l'utilisation de systèmes d'ordinateurs très coûteux. En effet, les contraintes de performance imposées par l'animation en temps réel, aux niveaux du volume d'archivage, des vitesses de manipulation par le processeur, des vitesses de transfert entre mémoires de masses et tampons de sortie se traduisent par des coûts de systèmes prohibitifs dans la plupart des cas. D'autres part, la généralisation des micro-ordinateurs et leur utilisation accrue dans le domaine du graphisme imposent de mettre au point des techniques d'animation par ordinateur qui, au prix d'une limitation de leurs performances, soient implantables sur des systèmes informatiques limités. Le présent rapport concerne l'étude d'une technique d'animation de ce type, appelée "Technique des Tables de Couleurs".

La caractéristique principale de cette technique réside dans le fait que, durant l'animation, le contenu de la mémoire tampon de sortie reste totalement inchangé. En effet, celle-ci ne contient pas successivement les différentes images de la séquence animée mais contient plutôt une "superposition" de toutes les images de la séquence. L'illusion de mouvement est alors créée en utilisant une séquence adéquate de fonctions de visualisation. Chacune de ces fonctions a pour rôle de sélectionner dans la mémoire tampon les éléments constituant l'image qui lui correspond.

L'avantage essentiel de cette technique d'animation se situe au niveau des vitesses de transfert d'information entre les différentes composantes du système d'animation. En effet, alors qu'avec une technique d'animation traditionnelle, chaque changement de position de l'objet aurait nécessité le rafraîchissement de toute la mémoire tampon, la technique de tables de couleurs n'implique qu'une remise à jour de la fonction de visualisation. Cette diminution considérable de la vitesse et du volume de transfert d'information permet de lever le principal obstacle à l'utilisation d'une certaine forme d'animation graphique sur des systèmes informatiques limités.

INTRODUCTION

Le présent document constitue le rapport final d'une étude exécutée par le Centre de Développement Technologique de l'École Polytechnique de Montréal, pour le Ministère des Communications du Canada, sous l'égide du contrat no. 21ST-36100-3-0014 intitulé "Techniques d'Animation Graphique par ordinateur". Conformément aux objectifs mentionnés dans la proposition de recherches, les objectifs de cette étude consistaient, dans la première phase, à effectuer une revue bibliographique concernant l'animation graphique interactive pour des systèmes informatiques limités et à étudier une technique d'animation graphique particulière appelée "Technique des tables de couleurs".

La durée prévue pour cette première phase était de 12 mois; néanmoins l'autorisation de commencer les travaux ne nous est parvenue qu'après un retard de 4 mois. Malgré ce retard, les objectifs fixés dans la proposition de recherche pour la première phase, ont été atteints et le présent rapport décrit les travaux accomplis et les résultats obtenus au cours de cette phase.

La section I de ce document donne une présentation générale de l'animation graphique par ordinateur et expose la problématique et les objectifs spécifiques de l'étude. La section II décrit la méthodologie de travail adoptée ainsi que la structure générale de la solution envisagée.

La description des logiciels et des données utilisés est effectuée à la section III et les résultats d'expérimentation sont présentés à la section IV. Enfin, la section V présente une conclusion des travaux effectués dans cette première phase et donne une description préliminaire des développements ultérieurs de cette étude.

I. PROBLÉMATIQUE

1.1 Exposé du problème

L'animation constitue une technique permettant de donner l'illusion du mouvement à partir d'une série d'images fixes. De ce fait, l'utilisation de procédés d'animation dans les communications visuelles peut permettre d'améliorer considérablement la qualité et l'efficacité de celles-ci.

Dans leur formes traditionnelles les techniques d'animation sont extrêmement laborieuses et coûteuses. L'utilisation d'un ordinateur dans différentes phases du processus d'animation permet d'accélérer considérablement les temps de production et d'en diminuer les coûts. On distingue en gros trois types différents d'animation:

- l'animation artistique: développée et utilisée principalement dans les studios cinématographiques pour la production de films de science-fiction. Les techniques mises en jeu sont extrêmement variées et complexes; elles font un usage important de l'ordinateur et se caractérisent par des coûts de production très élevés [1];

- l'animation à trois-dimensions: utilisée en particulier dans les domaines de la conception et de la fabrication assistée par ordinateur. Ce type d'animation soulève des problèmes spécifiques tels que le calcul de parties cachées, les questions d'ombrages, de transformation, etc.. Différentes techniques ont été développées pour solutionner ces problèmes [1, 2, 3, 4];

- l'animation en deux-dimensions: utilisée dans les films de type "dessins animés", dans certains documents à caractère pédagogique, dans les jeux vidéo, etc.. Bien que ce type d'animation ne présente pas les mêmes problèmes que l'animation à trois-dimensions, l'utilisation d'un ordinateur peut néanmoins être d'un précieux secours à deux niveaux..[5].

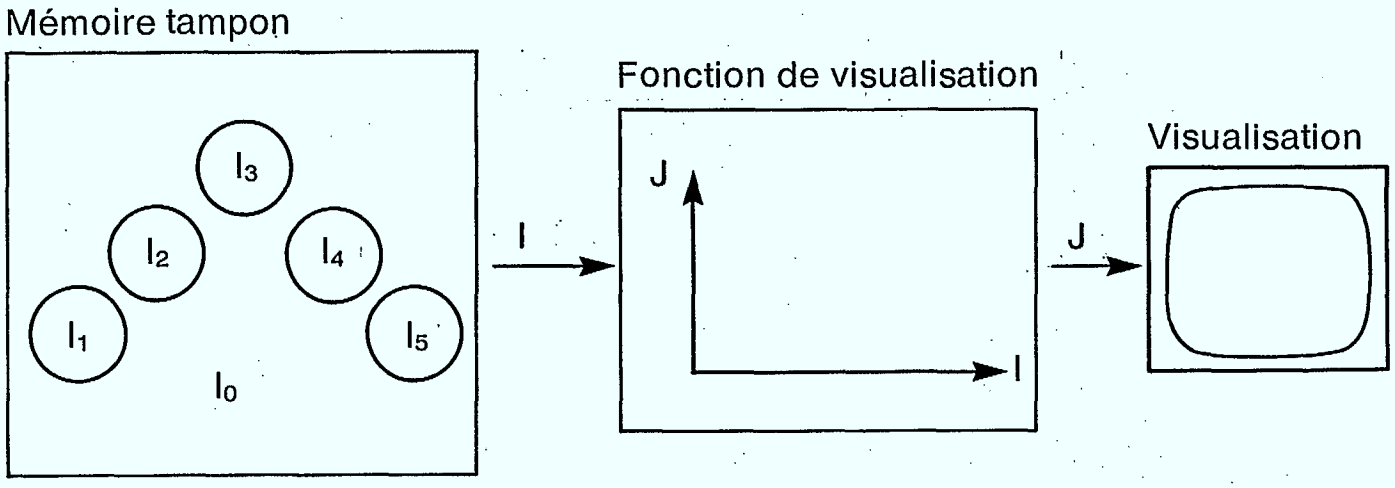
  Le premier niveau concerne la création et l'édition des séquences animées. Il s'agit tout d'abord de créer les images-clefs de la séquence c'est-à-dire les images à partir desquelles on pourra déduire toutes les autres par des opérations d'interpolation et de transformation. Cette phase de création pourra être accélérée grâce à l'utilisation d'un logiciel d'édition graphique performant. En ce qui concerne les processus d'interpolation et de transformation, l'ordinateur peut être également très utile. Cependant, les problèmes impliqués ne sont pas simples et plusieurs approches de solutions peuvent être envisagées [5]. Quoiqu'il en soit, l'introduction de l'ordinateur à ce premier niveau peut être caractérisée par le terme "Animation Assistée par Ordinateur".

  Le deuxième niveau concerne la visualisation au temps réel de la séquence animée, une fois le processus d'édition complété. Dans ce cas, l'ordinateur peut être utilisé comme support d'archivage, de régie et de visualisation de la séquence animée, ce qui donne lieu véritablement à "l'Animation par Ordinateur".

Malgré la diminution des coûts du matériel informatique, "l'Animation par Ordinateur" implique encore l'utilisation de systèmes

d'ordinateurs très coûteux. En effet, les contraintes de performance imposées par l'animation en temps réel, aux niveau du volume d'archivage, des vitesses de manipulation par le processeur, des vitesses de transfert entre mémoires de masses et tampons de sortie se traduisent par des coûts de systèmes prohibitifs dans le plupart des cas. [6], [7]. D'autre part, la généralisation des micro-ordinateurs et leur utilisation accrue dans le domaine du graphisme imposent de mettre au point des techniques d'animation par ordinateur qui, au prix d'une limitation de leurs performances, soient implantables sur des systèmes informatisés limités. Le présent rapport concerne l'étude d'une technique d'animation de ce type, appelée "Technique des Tables de Couleurs". [8, 9, 10]. Le choix de cette technique réside dans le fait que la plupart des systèmes graphiques actuellement disponibles se caractérisent d'une part par une mémoire de rafraîchissement (mémoire tampon) de taille limitée et d'autre part par la présence de tables de couleurs permettant une réassignation des couleurs fondamentales associées à chaque pixel.

La caractéristique principale de cette technique réside dans le fait que, durant l'animation, le contenu de la mémoire tampon de sortie reste totalement inchangé. En effet, celle-ci ne contient pas successivement les différentes images de la séquence animée mais contient plutôt une "superposition" de toutes les images de la séquence. L'illusion de mouvement est alors créée en utilisant une séquence adéquate de fonctions de visualisation. Chacune de ces fonctions a pour rôle de sélectionner dans la mémoire tampon les éléments constituant l'image qui lui correspond. La Figure 1.1.1 illustre un exemple simple d'animation basé sur cette technique. Les différentes positions de l'objet en mouvement sont toutes simultanément présentes dans le mémoire tampon, chacune d'elle étant caractérisée par une valeur de pixel (intensité) particulière. L'effet d'animation est obtenu en appliquant



les I_k désignent les valeurs numériques d'enregistrement des différents pixels dans la mémoire tampon.

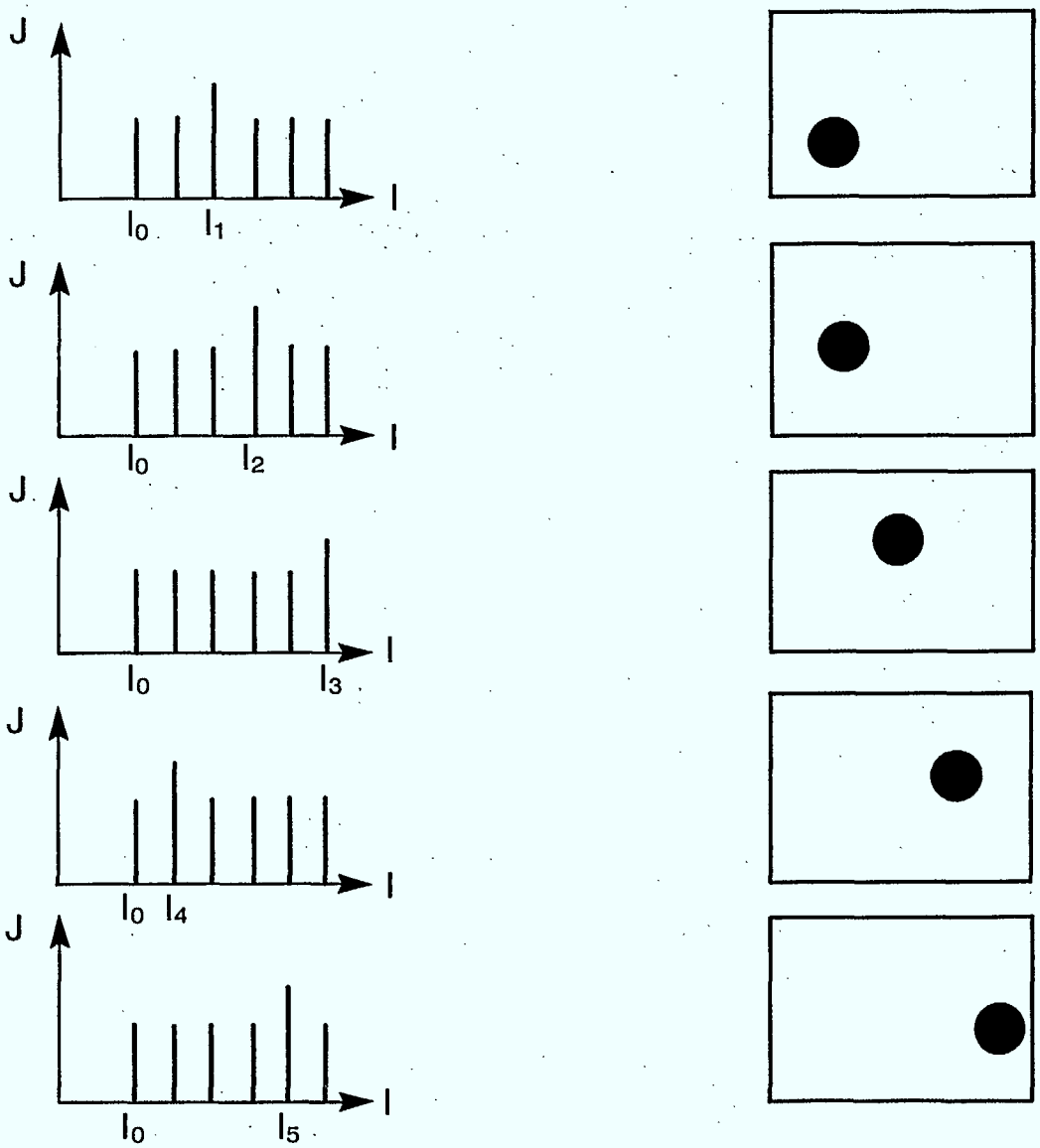


Fig. 1.1.1: Exemple d'animation par table de couleurs.

à la mémoire tampon la séquence de fonctions de visualisation indiquée sur la figure. Chacune de ces fonctions est choisie de façon à laisser apparaître une position de l'objet en mouvement ainsi que le fond, tout en maintenant les autres positions dans l'obscurité. L'avantage essentiel de cette technique d'animation se situe au niveau des vitesses de transfert d'information entre les différentes composantes du système d'animation. En effet, alors qu'avec une technique d'animation traditionnelle, chaque changement de position de l'objet aurait nécessité le rafraîchissement de toute la mémoire tampon, la technique de tables de couleurs n'implique qu'une remise à jour de la fonction de visualisation. Cette diminution considérable de la vitesse et du volume de transfert d'information permet de lever le principal obstacle à l'utilisation d'une certaine forme d'animation graphique sur des systèmes informatiques limités. Cependant le degré de complexité des séquences animées pouvant être ainsi produites est certainement limité et il s'agit d'évaluer cette limite et de déterminer dans quels types d'applications ce genre d'animation constitue une solution acceptable.

1.2 Revue bibliographique

Certains travaux sur l'animation par table de couleurs ont déjà été effectués et publiés [8, 9, 10], l'article de Shoup [9] étant le premier à faire le point sur les possibilités offertes par cette technique. Shoup décrit en particulier trois cas d'animation dans lesquels la technique des tables de couleurs s'avère efficace:

- il s'agit tout d'abord de l'animation par modification cyclique de couleurs, qui consiste à effectuer une rotation périodique des fonctions de coloration. Alors que l'image reste entièrement visible en tout temps, l'impression de mouvement est donnée par le défilement des couleurs à travers l'écran. Comme exemple typiques on peut citer des chûtes d'eau, ou des dessins abstraits;

- le deuxième cas consiste en une animation par alternance de couleurs. Dans ce cas, au moins deux images sont stockées simultanément dans la mémoire tampon en utilisant des ensembles différents d'intensités de pixels pour chaque image. La visualisation de chaque image est alors obtenue en utilisant, dans les fonctions de coloration, le sous-ensemble de valeurs de pixels correspondant (Voir Figure 1.1.1). La principale limitation réside dans le fait que les différentes images ne peuvent pas avoir de région d'intersection (overlap);

- le troisième cas consiste en une amélioration de l'animation par alternance de couleurs afin de permettre l'existence de régions d'intersection entre les différentes images. La technique utilisée consiste à subdiviser la mémoire tampon en plans de bits et à assigner des ensembles disjoints de plans de bits à des images ayant des régions communes [9].

Dans ses travaux, S. Mac Kay [10] propose de combiner la technique des table de couleurs avec une partition spatiale de la mémoire tampon en quatre, huit, seize zones séparées et de répartir les différentes images d'une séquence animée dans les différentes zones ainsi définies, au prix d'une perte de résolution spatiale. Une partition des intensités de pixels en sous-ensembles orthogonaux est également proposée.

Quelles que soient les approches de techniques proposées par Shoup ou Mac Kay, aucune évaluation de ces approches n'est effectuée en ce qui concerne leur viabilité, leur complexité ainsi que leurs performances dans un contexte matériel donné.

Pour qu'une telle évaluation puisse être effectuée, il est nécessaire d'étudier de façon systématique les problèmes posés pour le stockage simultanné en mémoire tampon d'images ayant une intersection non vide, ainsi que les problèmes d'assignation d'intensités de pixels qui en résultent.

1.3 Objectifs du projet

Conformément aux remarques précédentes, l'objectif du présent projet est d'effectuer une étude systématique de la technique d'animation par table de couleurs. Cette étude pour être complète doit comporter les différents aspects suivants:

(a) l'analyse et l'implantation logiciel de la technique d'animation proprement dite. Il s'agit à ce niveau, à partir d'une séquence d'images donnée, de développer une procédure d'archivage de cette séquence dans la mémoire tampon ainsi qu'une procédure d'extraction en temps réel, image par image;

(b) l'analyse et l'implantation d'un éditeur automatique de séquences. Il s'agit à partir d'un squelette de séquence, dans lequel seules certaines images-clefs sont définies, de créer par interpolation et transformation les autres images de la séquence. En dépit des apparences, ce problème n'est pas simple et la méthode à utiliser dépend de la technique d'animation retenue et des caractéristiques des mouvements à décrire (mouvements bi-dimensionnels, mouvement tri-dimensionnels projetés, déformations d'objets, etc.);

(c) la définition d'une interface usager permettant d'une part de créer les images-clefs d'une séquence avec un logiciel

graphique adéquat et d'autre part de communiquer au système, de façon simple et précise, le scénario complet de la séquence animée à produire.

d) l'expérimentation et l'évaluation globale de la technique en ce qui concerne ses performances, ses limitations, la complexité maximale des séquences éditables. Il s'agit aussi de déterminer dans quels contextes de systèmes informatiques et d'applications une telle technique est recommandée.

En ce qui concerne la première phase du projet, qui fait l'objet du présent rapport, les objectifs spécifiques sont l'analyse et l'implantation logicielle de la technique d'animation proprement dite (a), ainsi qu'une expérimentation et une évaluation partielles de cette technique (d). Au niveau de la conception et de l'implantation de la technique, plusieurs problèmes différents sont à résoudre en ce qui concerne:

- l'animation de séquences dans lesquelles les images consécutives possèdent des zones d'intersection;
- la procédure de stockage des images de la séquence dans la mémoire tampon;
- la procédure d'accès en temps réel au contenu de la mémoire tampon image par image.

Au niveau de l'expérimentation, il s'agit de définir quelques séquences animées typiques dont la complexité soit suffisante pour permettre une vérification de la viabilité de la technique et une évaluation partielle de ses performances.

II. MÉTHODOLOGIE

Le présent chapître décrit l'approche d'étude utilisée au cours de la première phase du projet, afin de rencontrer les objectifs mentionnés précédemment. Dans une première section, on décrira la configuration matérielle minimale nécessaire à l'implantation de la technique d'animation par tables de couleurs. La deuxième section définira la méthodologie d'étude adoptée pour la technique d'animation proprement dite ainsi que la terminologie utilisée dans tout la suite du rapport. Une troisième section décrira enfin la structure générale des logiciels développés ainsi que la structure des données.

2.1 Configuration matérielle minimale

Pour être en mesure d'implanter un logiciel d'animation par tables de couleurs, le système matériel utilisé doit avoir la configuration minimale illustrée à la Figure 2.1.1. Il faut tout d'abord une unité de traitement (micro ou mini-ordinateur) équipé d'une mémoire de masse pour le stockage des séquences d'images ainsi que des fonctions de visualisation et de coloration successives. Connectée à cette unité de traitement, il faut une mémoire tampon destinée au stockage de la totalité de la séquence animée à visualiser, sous un format compatible avec la technique des tables de couleurs. Les dimensions de cette mémoire tampon doivent être suffisantes pour contenir au moins une image visualisable, aux résolutions spatiale et lumineuse maximales de l'écran. Néanmoins, le système doit être dôté d'une possibilité de zoom permettant de ne visualiser qu'une section à la fois (un quart ou un seizième etc...) de la mémoire tampon, au prix d'une résolution spatiale moins bonne. La Figure 2.1.2 illustre cette aptitude dans le cas d'un zoom par un facteur linéaire de 2.

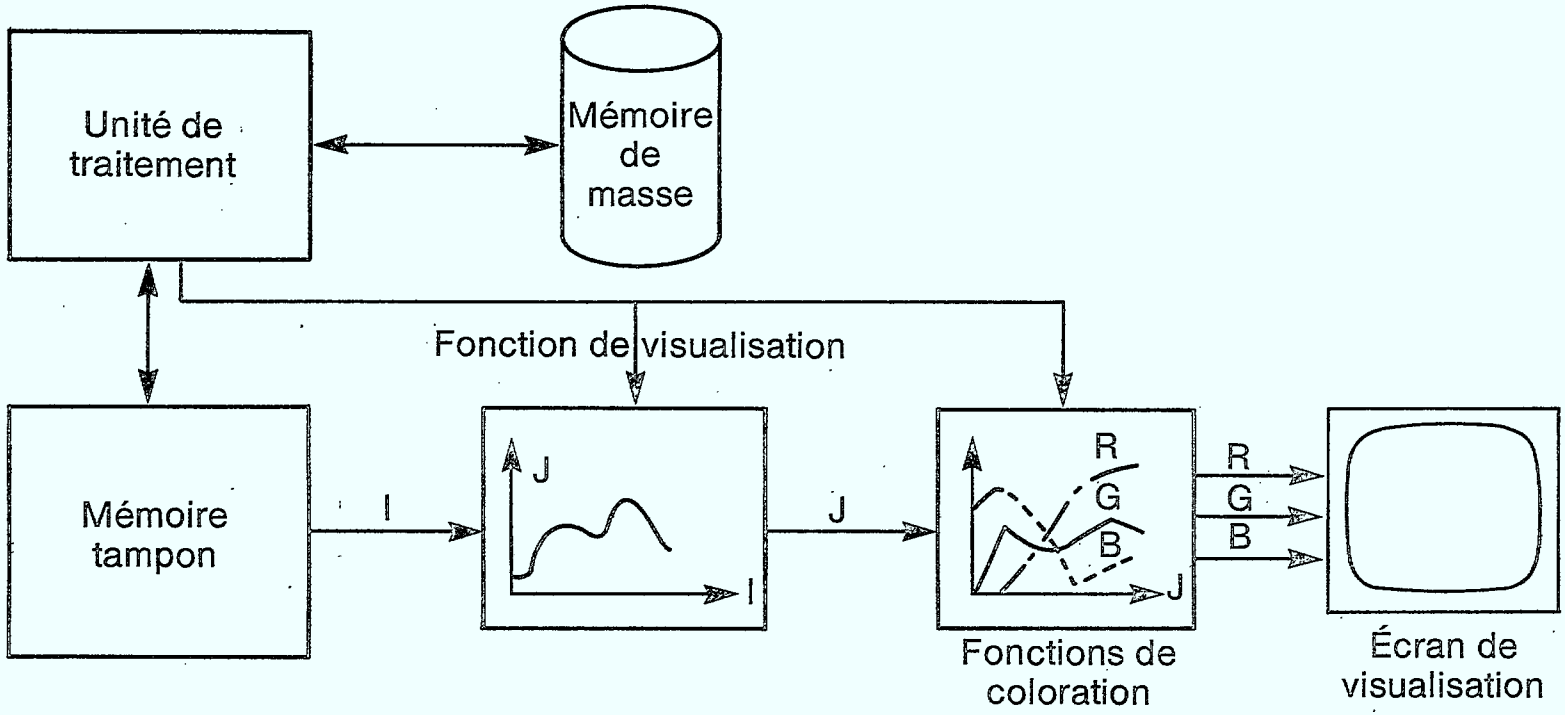


Fig. 2.1.1: Configuration matérielle mininale

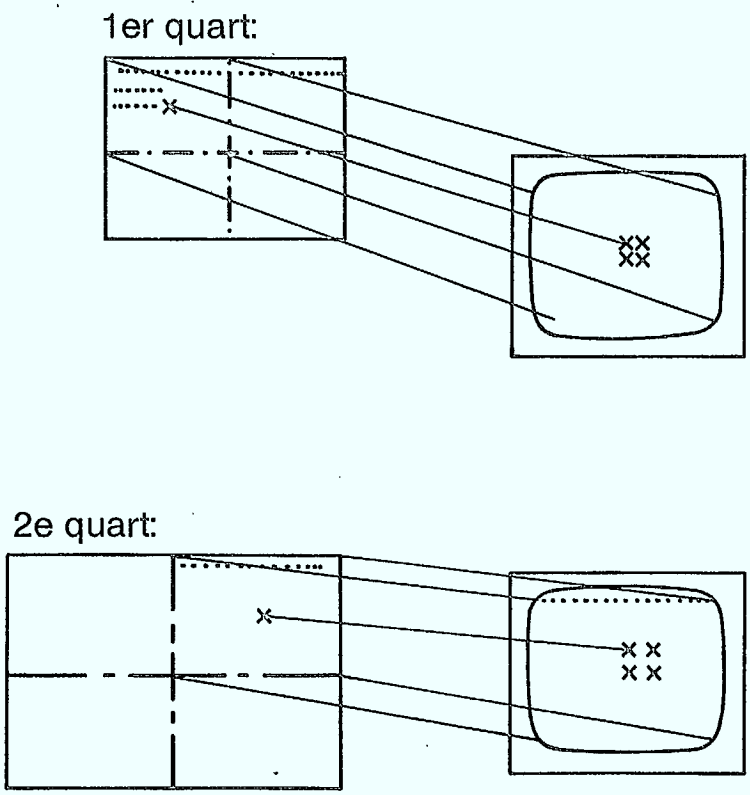


Fig. 2.1.2: Aptitude de zoom du système de visualisation

Avant visualisation sur l'écran, le contenu de la mémoire tampon doit passer à travers deux transformations successives intitulées "Fonction de visualisation" et "Fonctions de coloration". À chaque instant, la fonction de visualisation a pour rôle de sélectionner parmi tous les points de la mémoire tampon ceux qui doivent être visualisés. En conséquence, la valeur de sortie J de cette fonction doit être maintenue à zéro pour toutes les valeurs I de points de la mémoire tampon à maintenir dans l'obscurité. Par contre, J doit être mis à une valeur non-nulle adéquate pour toutes les valeurs I de points à visualiser. Quant aux fonctions de coloration elles ont pour rôle de déterminer les couleurs de tous les points à visualiser. Les caractéristiques des fonctions de visualisation et de coloration doivent pouvoir être remise à jour en temps réel (rythme maximum de 30 fois par seconde) par l'unité de traîtement, et c'est grâce à cette remise à jour que l'illusion du mouvement est créée.

La configuration décrite ci-dessus est une configuration de base, disponible sur la quasi totalité des systèmes de traîtement d'images ou de graphiques actuellement disponibles sur le marché.

Comme indiqué à la Figure 2.1.3, en ce qui concerne le système expérimental effectivement utilisé dans cette étude, l'unité de traîtement est constituée d'un mini-ordinateur DEC PDP 11/60. Quant à la mémoire tampon, aux fonctions de visualisation et de coloration ainsi qu'à l'unité de visualisation, elles font partie d'un processeur d'image COMTAL modèle Vision One/20. La taille de la mémoire tampon utilisée est de 512 lignes par 512 points par 8 bits. En conséquence les fonctions de visualisation et de coloration sont constituées par des tables de 256 octets chacune.

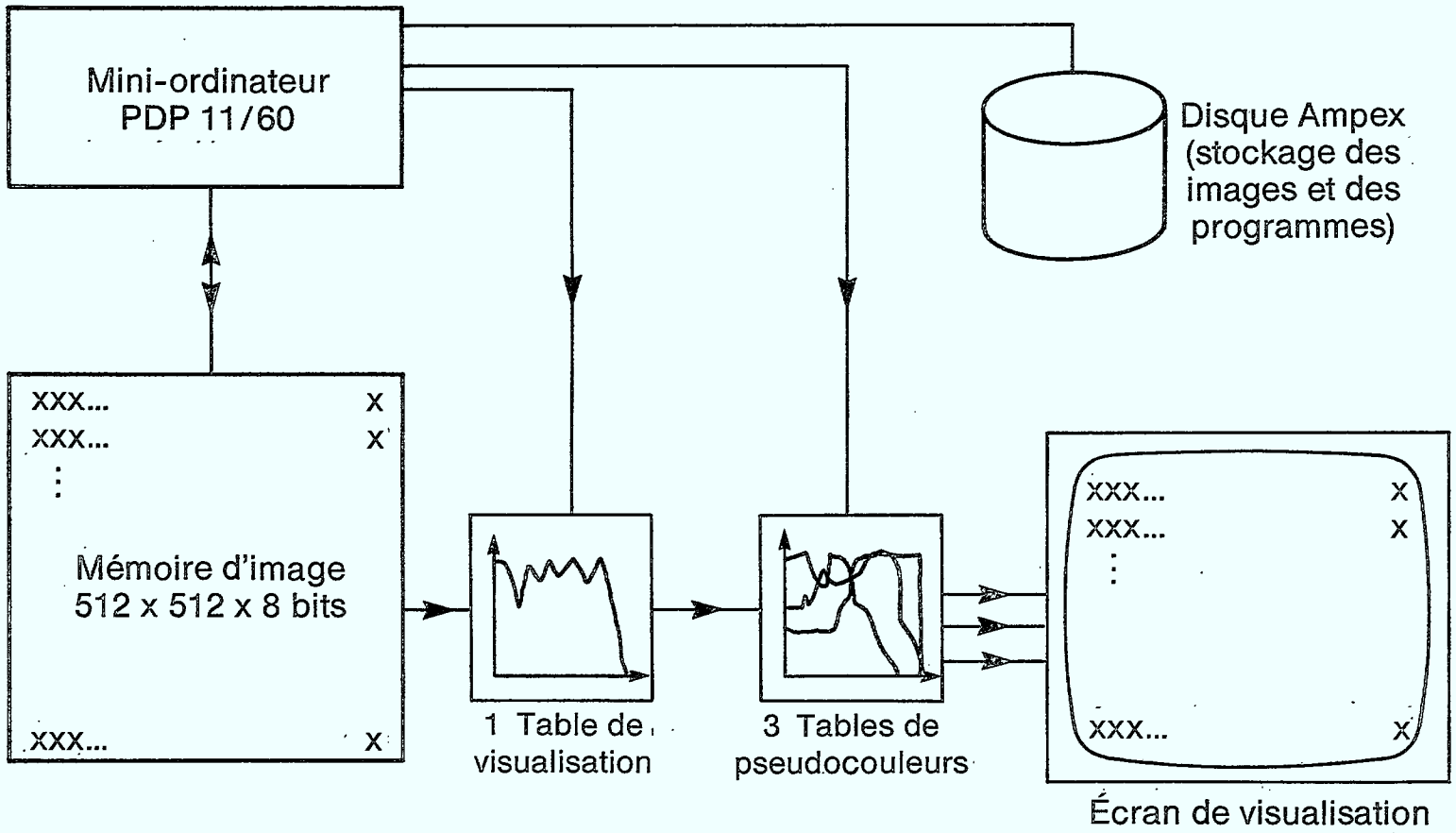


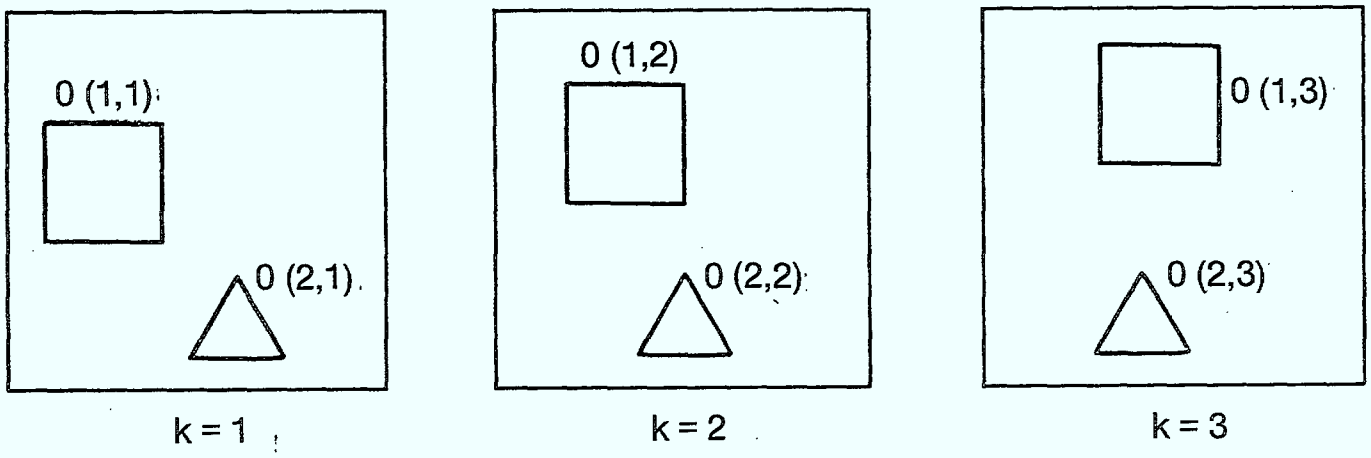
Fig. 2.1.3: Description du matériel employé, montrant la partie utilisée du système de traitement d'image Vision One/20 de Comtal

2.2 Méthodologie et terminologie

Étant donné une séquence d'images décrivant chacune les positions successives d'une scène animée, nous introduisons alors le concept "objet" pour désigner toute entité physique qui figure dans au moins une image à l'intérieur de la scène et qui apparaît toujours en une seule couleur dans une image. Un objet peut être, en fait, formé de plusieurs parties non connectées. En tant qu'ensembles de pixels, deux objets distincts sont toujours disjoints. L'apparition de l'objet i dans l'image k donne naissance à "l'événement graphique" $O(i,k)$, tel qu'illustré sur un exemple à la figure 2.2.1.a. L'"histoire graphique" $O(i)$ de l'objet i est l'énumération des événements $O(i,k)$, $1 \leqslant k \leqslant K$, relatifs à un i fixe. (Voir Figure 2.2.1.b). Pour la visualisation en temps réel, il faut que l'image visualisée réside en "mémoire graphique". Une mémoire graphique est définie comme étant une région (la totalité, un quart, un seizième, etc.) de la mémoire tampon disponible. Dans la technique d'animation graphique par table de couleurs, une mémoire graphique contient nécessairement plusieurs images visualisées, soit le sous-ensemble $H \subseteq \{1, 2, \ldots, K\}$. Dans l'exemple de la Figure 2.2.2.c, toute la séquence est placée dans la même mémoire graphique. Chaque image visualisée est formée des événements graphiques $\{O(i,k), i \varepsilon I\}$ pour tout $k \varepsilon H$. Quoique les événements graphiques $O(i_1,k)$ et $O(i, k_2)$, soient disjoints pour tout les $k \varepsilon H$, tout $i_1, i_2 \varepsilon I$; les événements praphiques $O(i,K_1)$ et $O(i,k_2)$, k_1, k_2 εH, $i \varepsilon I$, ne le sont pas nécessairement.

Dans une mémoire graphique nous définissons une intersection comme l'ensemble de pixels appartenant au même membre de la famille des sous-ensembles de l'ensemble $\{O(i,k), i \varepsilon I, k \varepsilon H\}$ relatif à cette mémoire graphique. (Voir Figure 2.2.1.c). Il faut donc assigner à chaque

(a) Notions d'image et d'évènement graphique



(b) Notion d'histoire graphique

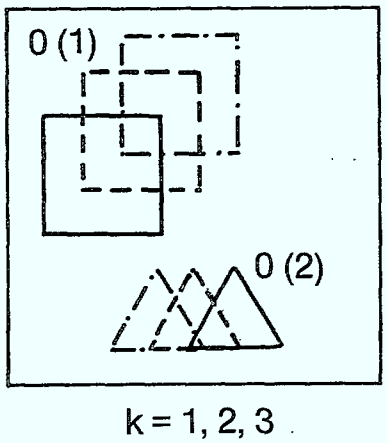
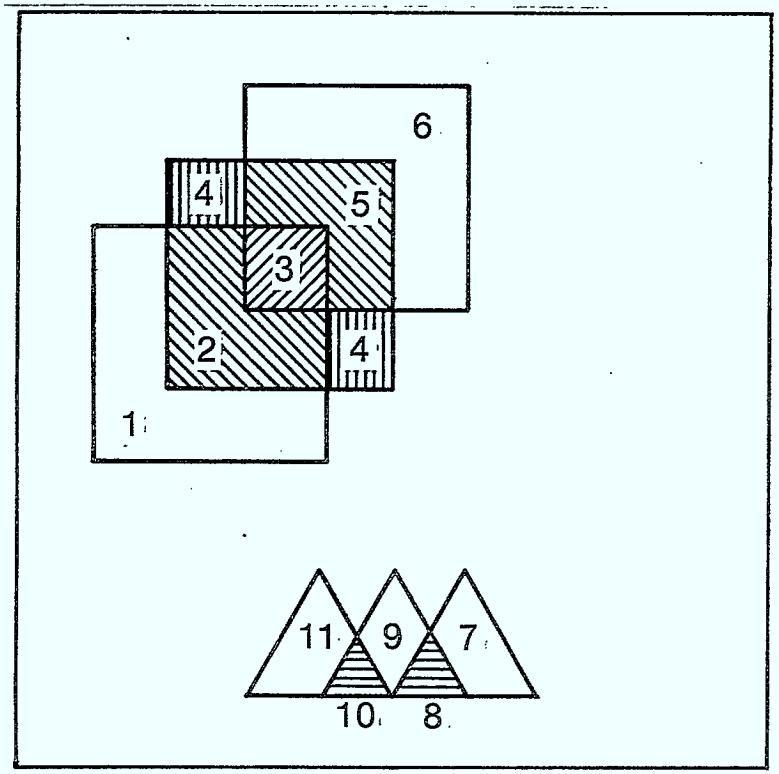


Fig. 2.2.1: Exemple d'animation par table de couleurs

(c) Notions de mémoire graphique et d'intersections



H ={1, 2, 3}

d) Assignation d'intensité f.H

1	I_1
2	I_2
3	I_3
4	I_4
5	I_5
6	I_6
7	I_7
8	I_8
9	I_9
10	I_{10}
11	I_{11}

e) Fonctions de visualisations

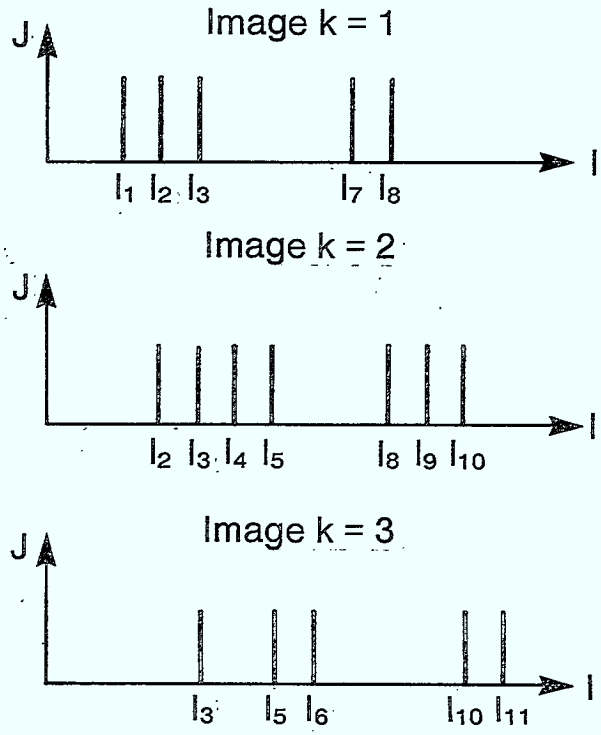


Fig. 2.2.1: Exemple d'animation par table de couleurs

intersection une "intensité" unique. Mathématiquement parlant, l'ensemble des intersections J constitue effectivement une partition de tous les pixels d'une mémoire graphique. Un algorithme d'assignation d'intensité est fonction injective f.H de J dans l'ensemble des intensités disponibles C. (Voir Figure 2.2.1.d). La restriction de f.H dans $\{O(i,k),\ i \varepsilon I\}$ donne la fonction discriminante $f_k.H$ utilisée pour visualiser l'image $k \varepsilon H$. (Voir Figure 2.2.1.e).

Un nombre restreint, disons 4, de mémoires graphiques sont supposées être disponibles. Un algorithme de répartition d'images est utilisé pour distribuer les images visualisées de la séquence animée sur les 4 mémoires graphiques données. Nous représentons cet algorithme par une fonction R définie sur $\{1, 2, \ldots, K\}$ et ayant des valeurs dans l'ensemble $\{H_1, H_2, H_3, H_4\}$. La visualisation en temps réel se fait au moyen de la suite des fonctions $\{f_k.H_\ell\}_{k=1}^{K}$, où $H_\ell = R(k)$. Du point de vue de réalisation pratique sur le système Comtal Vision One/20, les H_ℓ sont en fait des sous-mémoires (quarts de mémoires) que nous présenterons sur l'écran à balayage comme pleine image visualisée grâce aux contrôles de balayage.

2.3 Structure générale des logiciels et des données

Le logiciel d'animation a été développé sous la forme de plusieurs programmes que l'on fait tourner successivement et ceci pour plusieurs raisons. Tout d'abord, cette conception modulaire facilite la mise au point, les modifications et surtout la maintenance du logiciel. De plus, en cas d'erreur dans l'image finale, il est beaucoup plus facile de retracer ou se situe l'erreur de manipulation, et comment obtenir un résultat correct. Cela permet, de plus, de gagner du temps en ne faisant réexécuter que les programmes nécessaires. On a donc un meilleur contrôle sur la création de l'image. Enfin et surtout, les programmes qui constituent ce logiciel sont assez gros, et

certains nécessitent déjà une structure de recouvrement dynamique. Cette structure en arbre permet de ne charger en mémoire que les sous-programmes dont on a besoin et donc d'utiliser le même espace-mémoire pour plusieurs sous-programmes qui ne sont pas appelés en même temps. C'est le seul moyen de faire exécuter des programmes qui sinon dépasseraient les 32 kmots d'espace adressable de la mémoire du PDP 11/60, mais cela ralentit considérablement l'exécution. Séparer le travail à effectuer en plusieurs programmes distincts a donc permis d'accélérer la création de l'image finale.

Le logiciel comprend les étapes suivantes, indiquées sur la figure 2.3.1,

- création de l'image "de base";
- génération des images des positions intermédiaires;
- détermination des intersections entre les événements de ces différentes images;
- attribution des intensités à ces événements;
- animation proprement dite.

Un éditeur graphique interactif [11] est tout d'abord utilisé pour créer ce qu'on appelle "l'image de base", c'est-à-dire une image typique de la séquence contenant tous les éléments visibles de la scène. L'image de base se présente alors sous la forme d'une liste dans laquelle chaque élément visible est décrit sous forme paramétrique.

Cette liste est ensuite copiée, autant de fois qu'on veut d'images dans la séquence et les coordonnées des événements sont progressivement modifiés pour créer le mouvement. Les images sont ensuite recrées et conservées sous la forme de fichiers à accès direct. Sous cette forme, chaque image est représentée comme indiqué figure 2.3.2 par 16 x 8 blocs non formatés, de 32 lignes par 16 pixels.

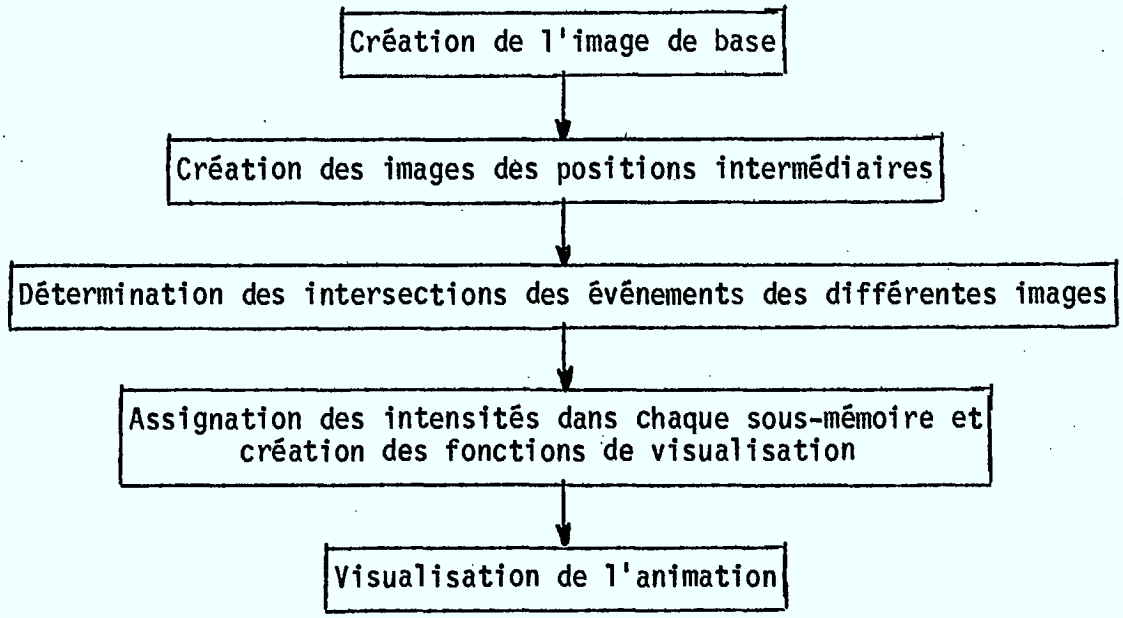


FIGURE 2.3.1: Structure du logiciel

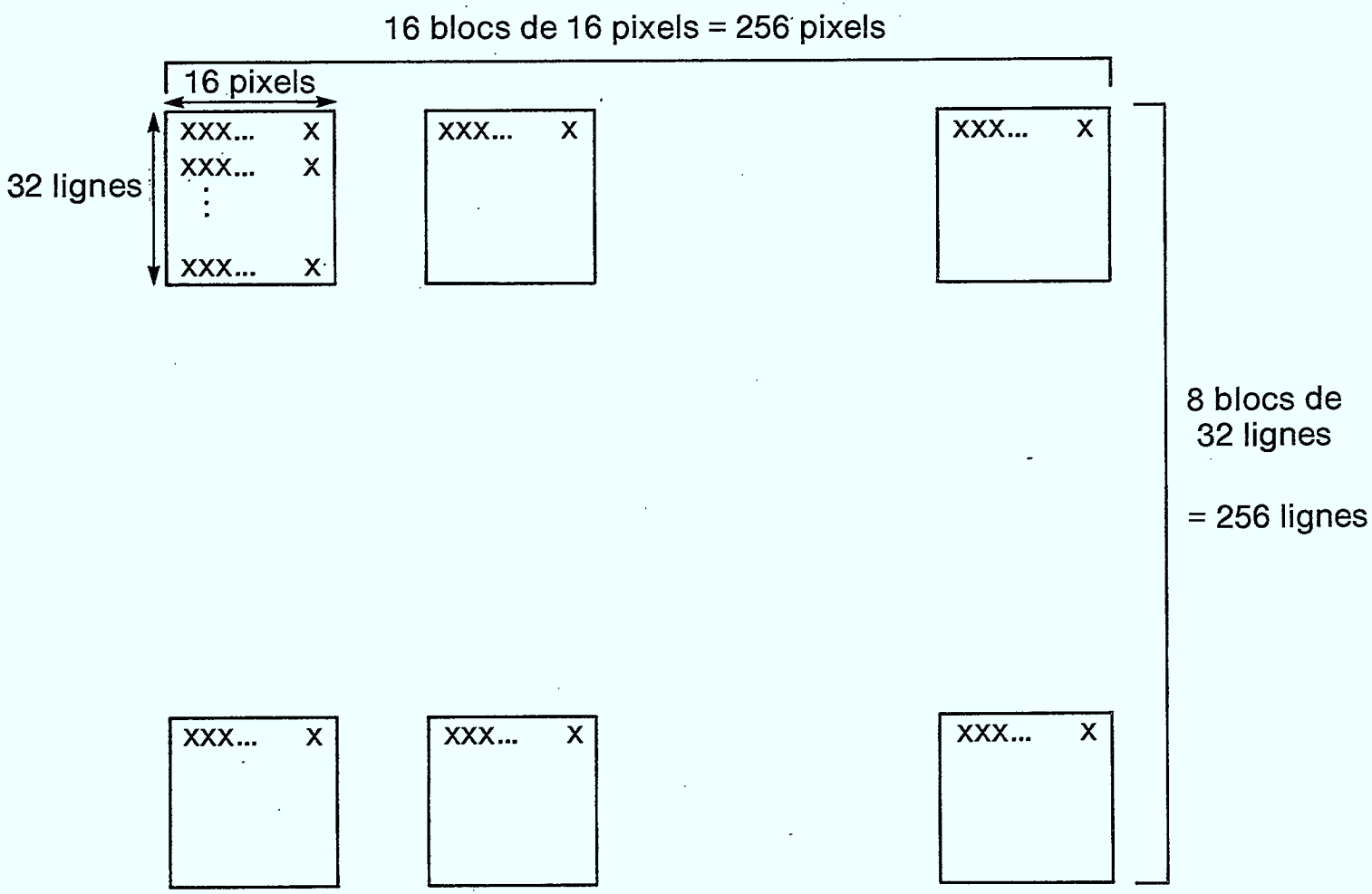


Fig. 2.3.2: Format utilisé pour les images

La structure non formatée à accès direct a été utilisée autant pour un stockage compact et un accès rapide, que pour un traitement sélectif des parties d'images.

Ces images sont ensuite légèrement modifiées pour tenir compte des insuffisances de l'éditeur interactif. Le traitement proprement dit peut alors commencer.

Les images sont d'abord placées les unes après les autres dans leur sous-mémoire respective pour déterminer les intersections. Le choix de l'une ou l'autre des quatre sous-mémoires pour chacune des sous-images est fait suivant certains critères visant à réduire le nombre total d'intersections. Les intensités sont ensuite assignées à chacune des intersections en tenant compte des contraintes entre les sous-mémoires. L'image finale est enfin créée en concaténant les quatre sous-mémoires.

Une fois l'image finale créée, il ne reste plus qu'à lancer le programme de visualisation de l'animation. Si celle-ci a une forme cyclique et prédéterminée comme dans l'exemple du moteur (cf. paragraphe 4.2), il suffit de spécifier les mouvements au préalable dans un fichier de données, et si le mouvement est variable d'une fois à l'autre, comme dans l'exemple des tours de Hanoï (cf. paragraphe 4.3), il faut une procédure qui décide des mouvements à effectuer.

Donc la manière de procéder avec le logiciel est de créer d'abord une image, ensuite de générer toutes les images intermédiaires, puis de les réunir dans les sous-mémoires. Après avoir mis ces sous-mémoires bout à bout, on peut visualiser l'animation.

III. DESCRIPTION DES LOGICIELS ET DES DONNÉES

3.1 Création des images de la séquence

Une partie importante de l'animation est le calcul de chacun des états et positions de toutes les pièces (pistons, valves, bielles, etc. dans l'exemple du moteur; voir paragraphe 4.2) à l'intérieur des K images constituant la séquence animée. Ce calcul, guidé par le scénario de la séquence établi au préalable, montre la position de chaque objet pour les K images de la scène.

L'objectif de cette première phase d'étude n'étant pas de développer un logiciel d'édition de mouvements, les différentes images d'une séquence ont été, dans un premier temps, créées une à une à partir d'un scénario préétabli.

Aucun des objets ne disparraissant à l'intérieur de la séquence désirée, seuls les déplacements d'objets sont à consédérer. Les déplacements d'objets se font par modification des coordonnées de chacune des primitives constituant cet objet.

Le format de la base de données générée par l'éditeur interactif utilisé associe une fenêtre comprenant des primitives à l'intérieur, à chacun des objets. La position de ces primitives (généralement des segments de droite ou des courbes) étant relative à cette fenêtre, il nous est donc possible de déplacer toutes les primitives constituantes d'un objet en déplaçant la fenêtre associée à ce dernier.

Afin de mieux comprendre les techniques utilisées lors de la création des fichiers, nous examinerons ici le format de la base de données. Chaque fichier est composé de deux parties: une décrivant la fenêtre et l'autre les primitives de l'objet comme indiqué sur la figure 3.1.1.a.

a) Base de données

FENÊTRE

xmin	xmax	ymin	ymax	intensité	priorité	ligne pointeur									couleur	objet plein
205	258	284	319	174	36	153	2	0	0	0	1	0	0	0	5	1

⋮

PRIMITIVES

0	53	0	35	0	0	0	0	
1	2	1	52	1	2	1	0	
1	52	1	52	26	2	1	0	← ligne 153
1	52	26	30	34	2	1	0	
1	30	34	21	34	2	1	0	
1	21	34	2	26	2	1	0	
1	2	1	2	26	2	1	0	

b) Reconstruction de l'objet à partir de la base de données

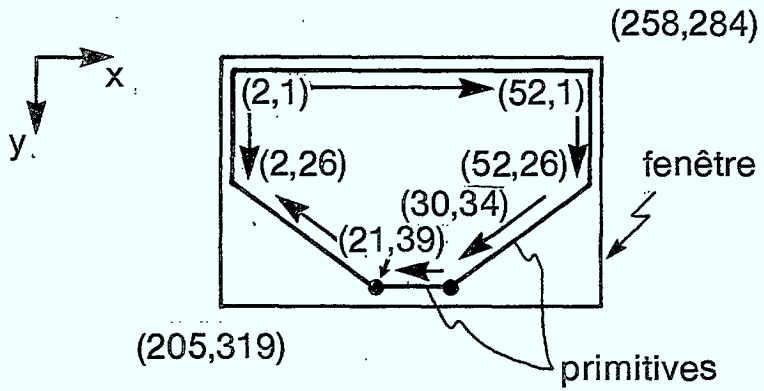


Fig. 3.1.1: Exemple d'un objet et de sa représentation à l'aide de l'éditeur V20.

L'objet graphique représenté par ces données sera tel que sur la figure 3.1.1.b.

Il se peut que certains événements (rotation d'une came par exemple) rendent nécessaire la modification de la forme de l'objet. L'objet graphique résultant de la projection sur le plan d'affichage d'un objet réel non-cylindrique par rapport à son axe de rotation dans l'espace virtuel verra sa forme se modifier au fur et à mesure que les événements s'enchaînent.

Dans le contexte de notre projet, la projection des objets, animés d'un mouvement de rotation par rapport à un axe quelconque, était toujours de forme rectangulaire ce qui simplifie beaucoup notre tâche. La modification de la forme découlant d'un événement graphique quelconque impliquant des rotations s'effectue après le déplacement de l'objet à sa nouvelle position.

Pour la modification de la forme d'un objet, il sera nécessaire de modifier les primitives qui le composent, après avoir calculé la nouvelle position de sa fenêtre. La méthode utilisée est celle des plans de coupe ("clipping planes") et comporte les étapes suivantes:

(1) Lecture de l'ancienne fenêtre (avant l'événement).
(2) Lecture de la nouvelle fenêtre (après l'événement).
(3) Calcul du changement de la forme.
(4) Remplacement de l'ancienne fenêtre par la nouvelle dans le fichier.
(5) Modification des primitives.

Ceci résume les opérations nécessaires à la création des fichiers d'images de la séquence à animer. Une fois ces fichiers créés, il s'agit de les traiter à l'aide de logiciels adéquats afin de générer le contenu de la mémoire tampon. Ce contenu représente toute l'information de la séquence animée sous un format compatible avec la technique des tables de couleurs.

3.2 Détermination des intersections et assignation des intensités

Conformément aux observations faites dans la méthodologie, si l'on désire que la technique des tables de couleurs permette de produire de l'animation dans des cas non triviaux, il est nécessaire de rendre cette technique applicable à des situations de séquences dans lesquelles les images consécutives présentent des zones d'intersections. Conformément à l'exemple de la figure 2.2.1, il est alors nécessaire de calculer toutes les zones d'intersections et de leur assigner des intensités de pixels particulières avant de les archiver dans la mémoire tampon. Pour ce faire, il faut définir au préalable la notion d'histoire d'un pixel ou d'une intersection.

3.2.1 Concept d'histoire

soit: P: l'ensemble des pixels p d'une image
T: l'ensemble du temps de visualisation $\{t/t \in N,\ 0 \leq t \leq t_{fin}\}$

à chacun des éléments de T est associé une image

soit: J: l'ensemble des intensités lumineuses j

On définit la fonction d'animation f comme étant la fonction donnant l'intensité associée à chaque pixel en fonction du temps.

$$f: P \times T \rightarrow J$$

Cette fonction est suffissante pour faire toute l'animation, c'est-à-dire pour voir toutes les images les unes après les autres.

Pour utiliser la méthode d'animation par tables de couleurs, il faut trouver deux fonctions, g et h, qui produiront la même animation que f, c'est-à-dire:

$$g: P \rightarrow I$$

et

$$h: I \times T \rightarrow J$$

où I représente l'ensemble des intersections i des images de la séquence.

La fonction g réalise, à partir des images successives de la séquence, le calcul de toutes les zones d'intersections de ces images. Les pixels constituant chaque zone d'intersection sont alors identifiés, en mémoire tampon, par une valeur particulière d'enregistrement i. (On désigne par i aussi bien la zone d'intersection que la valeur d'enregistrement en mémoire de ses pixels). Les zones d'intersections, constituant l'ensemble I, sont toutes des régions disjointes dont l'union constitue l'ensemble P de tous les pixels. La fonction h réalise, à chaque instant de visualisation, l'assemblage des zones d'intersections i de I qui sont nécessaires pour constituer l'image à visualiser. Elle permet également d'assigner une intensité lumineuxe j à chaque point à visualiser. La fonction h est donc identique à la fonction de visualisation (voir figure 2.1.1).

En conséquence, la fonction g sera évaluée une seule fois alors qu'une fonction h doit être évaluée pour chaque instant de visualisation.

La quantité d'information devant être transmise pour chaque image est proportionnelle au nombre de couples (p,j) lorsque l'on utilise f et au nombre de couples (i,j) pour h. On tire donc avantage de la méthode lorsque la cardinalité de I est inférieure à celle de P.

Nous introduisons maintenant le concept d'histoire, ce qui nous aidera à trouver un ensemble I le plus petit possible.

-- Définition: Une histoire est un ensemble de couples (t,j) associés à un pixel, une intersection ou un objet; en termes plus simples, l'histoire est l'évolution de l'intensité associée à un objet dans le temps.

On notera l'histoire d'un pixel par

$$H_p = \{(t,j)/t\varepsilon T,\ j\varepsilon J,\ j = f(p,t)\}$$

et l'histoire d'une intersection par

$$K_i = \{(t,j)/t\varepsilon T,\ j\varepsilon J,\ j = h(i,t)\}$$

Désignons l'ensemble des histoires des pixels par

$$M = \{H_p/p\varepsilon P\}$$

et l'ensemble des histoires des intersections par

$$N = \{K_i/i\varepsilon I\}$$

Pour tout $i\varepsilon I$, si $p\varepsilon P$ est tel que: $i = g(p)$ alors $K_i = K_{g(p)} = H_p$

ce qui implique que: $N \subseteq M$

Pour tout $p \varepsilon P$, si $i \varepsilon I$ est tel que: $i = g(p)$ alors $H_p = K_{g(p)} = H_i$

c'est-à-dire: $M \subseteq N$

En conséquence: $M = N$

De ce fait, si l'on désire minimiser le nombre de zones d'intersections, il est nécessaire de regrouper dans une même zone tous les pixels ayant des histoires identiques, même si la zone alors formée est non connexe. Par exemple, dans le cas de la figure 2.2.1(a), une minimisation a été effectuée en définissant comme une seule intersection les deux régions identifiées par "4".

Il est donc évident que pour avoir un minimum d'intersections, il faut:

$$\text{Card}(I) = \text{Card}(N) = \text{Card}(M)$$

En d'autres mots, à chaque intersection doit être associée une histoire différente.

3.2.2 Détermination des histoires

Avant de se lancer dans la conception d'un algorithme de détermination des histoires, il faut connaître les contraintes qui seront imposées lors de la réalisation de cette algorithme.

Les images que l'on veux voir lors de l'animation sont disponibles dans des fichiers. Il y a une image par fichier. Chaque pixel de cette image est codé sous forme d'un octet qui représente l'intensité associée à ce pixel (256 intensités possibles).

Pour connaître l'histoire d'un pixel, il faut accéder à un grand nombre de fichiers (autant que d'images dans la séquence). À cause des ressources que cela demande (mémoire pour le bloc de description de fichier et pour les tampons), tous ces fichiers ne peuvent être ouverts en même temps. L'analyse des images pixel par pixel n'est donc pas avantageuse car pour connaître l'histoire d'un seul pixel, il faudrait ouvrir et fermer un grand nombre de fichiers. De plus, lors de la lecture d'un octet sur disque, au minimum un secteur doit être transféré à la mémoire centrale. Un grand nombre d'octets serait alors transféré sans être utilisé, ce qui est loin d'être efficace.

Un algorithme utilisant chacun des fichiers une seule fois a beaucoup plus de chance d'être efficace car, il minimiserait les ouvertures et fermetures de fichier et utiliserait tous les octets transférés lors de la lecture d'un secteur.

Pour nous aider à développer un tel algorithme, nous allons introduire une fonction donnant l'histoire d'un pixel de l'instant 0 jusqu'à l'instant t.

Notons l'histoire du pixel p jusqu'au temps t par

$$Q_{pt} = \{(t_1,j)/t_1 \varepsilon T,\ t_1 \leqslant t,\ j = f(p,t)\}$$

et l'ensemble des histoires jusqu'au temps t par

$$R_t = \{Q_{pt}/p \varepsilon P,\ t \varepsilon T\}$$

L'algorithme dont nous donnons une description un peu plus loin, ajoute les images une par une à l'image statique. Après chaque ajout, à chacune des intersections de l'image statique correspondra une

histoire dans R_t. À la fin, lorsque $t = t_{fin}$, on aura $R_{t_{fin}} = M$ et l'image statique contiendra les intersections correspondants aux histoires de M.

Après l'ajout d'une image, les nouvelles histoires seront déterminées par:

$$Q_{pt} = Q_{p(t-1)} + \{(t, f(p,t)\}$$

Algorithme de détermination des histoires

- initialisation de l'image statique
- initialisation de la table des anciennes histoires
- pour chaque image:
  - -- initialisation de la table des nouvelles histoires
  - -- NBINT ← 0 (nombre d'intensités assignées)
  - -- pour chaque pixel:
    - si $Q_{p(t-1)} + \{(t,f(p,t))\}$ n'est pas dans la table des nouvelles histoires
    - mettre la nouvelle histoire dans la table et lui assigner le numéro suivant possible d'intersection (NBINT)
    - NBINT ← NBINT + 1
    - utiliser l'intensité associée (dans la table) à l'histoire $Q_{p(t-1)} + \{(t,f(p,t))\}$ pour représenter le pixel dans l'image statique
  - -- table des anciennes histoires ← table des nouvelles histoires

3.2.3 Utilisation de plusieurs sous-mémoires

Le système étant doté d'une possibilité de zoom permettant de ne visualiser qu'une section à la fois de la mémoire tampon, il est

possible, pour obtenir des séquences plus longues ou plus complexes, de conserver plusieurs images au lieu d'une seule dans la mémoire tampon.

La mémoire tampon du système est donc divisée en quatre sous-mémoires (ou mémoires graphiques) contenant chacune une image différente pouvant être visualisée séparément grâce à un zoom de facteur 2.

Lors d'une séquence d'animation, il peut être nécessaire d'effectuer simultanément un changement de sous-mémoires et un changement de fonctions de visualisation. Or, cela est impossible avec le système utilisé et ces deux changements doivent donc être faits l'un après l'autre. Il en résulte donc que tout changement de sous-mémoires se fait tout en conservant la même fonction de visualisation, alors que tout changement de fonctions de visualisation se fait à l'intérieur d'une même sous-mémoire. En conséquence, comme lors du changement de sous-mémoire, la fonction de visualisation doit rester la même, il est nécessaire, pour ne pas avoir de modification visible dans l'image visualisée, qu'il y ait une certaine dépendance entre les intersections des différentes sous-mémoires. Ceci augmente alors le nombre réel d'intensités utilisées dans chaque sous-mémoire.

Si lors d'une séquence il est requis d'avoir un changement d'une sous-mémoire 1 vers une sous-mémoire 2, alors ces deux dernières doivent être telles que:

- aux numéros d'intersections utilisés dans l'une <u>et</u> dans l'autre des deux sous-mémoires correspondent des intensités lumineuses identiques lors du changement de sous-mémoire. Soit donc, qui dans leur ensemble forment les mêmes événements mais sans nécessairement que ces intersections soient identiques. La figure 3.2.3.1.a montre que dans les sous-mémoires

1 et 2, l'événement à visualiser est formé des intersections 2, 3 et 4 bien que ces intersections ne soient pas identiques dans chacune des deux mémoires;

et/ou:

- si aux mêmes événements à visualiser lors du changement de sous-mémoire, il ne correspond pas les mêmes intersections dans leur ensemble, alors les numéros d'intersections utilisés dans l'une des sous-mémoires ne doivent pas l'être dans l'autre (cf. figure 3.2.3.1.b).

Donc, pour rendre compatible les intensités associées à chaque intersection dans les différentes sous-mémoires, il est nécessaire d'effectuer une réassignation des intensités. Cette réassignation se fait à l'aide des histoires des intersections: à deux intersections ayant des histoires compatibles est associée une intensité finale identique.

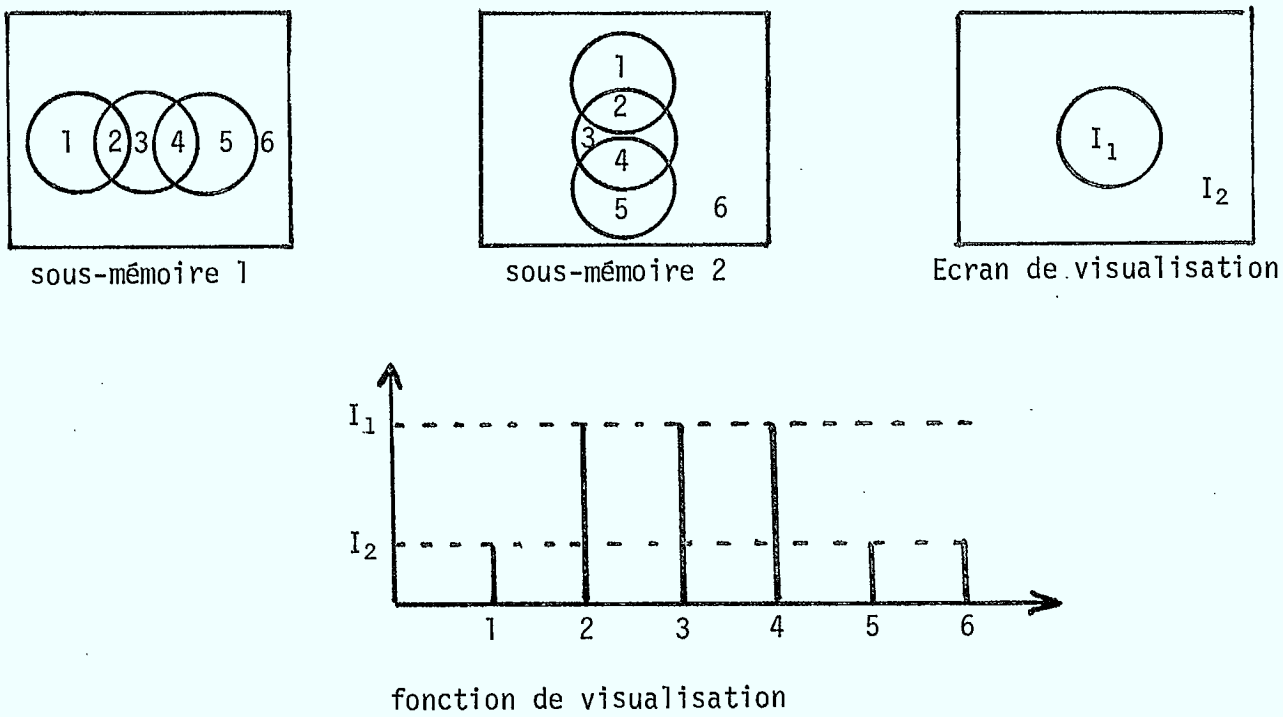
Soient: A: l'ensemble des numéros de fonctions de visualisation et
S: l'ensemble des numéros de sous-mémoires.

On définit

$$y: T \rightarrow A$$
$$y(t) \rightarrow a$$

Quand on utilise une seule sous-mémoire, cette fonction y est une bijection puisque à tout instant correspond une et une seule fonction de visualisation et toute fonction de visualisation est associée à un temps unique. Alors que quand plusieurs sous-mémoires sont utilisées, cette fonction y est une surjection puisque on a toujours une fonction unique associée à chaque instant t mais par contre une même

(a) mêmes numéros d'intersections utilisés



(b) numéros d'intersections utilisés différents

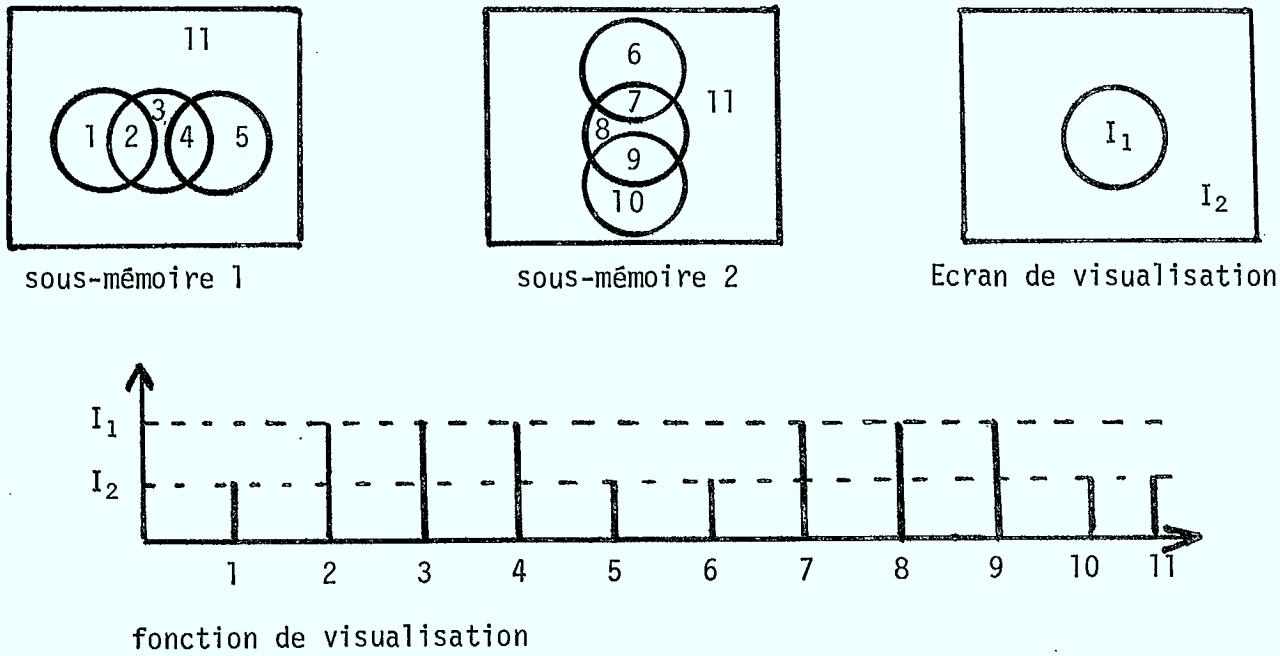


FIGURE 3.2.3.1: Exemple de changement de sous-mémoire en conservant la même fonction de visualisation

fonction de visualisation est associée à deux instants t et t+1 correspondant aux instants précédant et suivant le changement de sous-mémoire.

On définit maintenant $B_{i,s}$ comme étant l'histoire de l'intersection i dans la sous-mémoire s.

Pour tout $(a_1,j_1) \varepsilon B_{i_1,s_1}$ et $(a_2,j_2) \varepsilon B_{i_2,s_2}$, si $a_1 = a_2$

implique que: $j_1 = j_2$

alors on peut assigner le même numéro d'intersection final aux deux intersections i_1 dans s_1 et i_2 dans s_2.

La figure 3.2.3.2 nous montre un exemple de réassignation des numéros d'intersections.

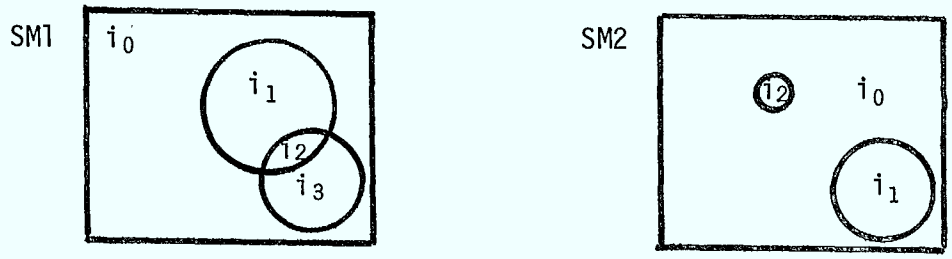
Le nombre d'intensités étant limité à 256, il est nécessaire d'en diminuer le nombre utilisé, c'est-à-dire de réduire le nombre d'intersections pour permettre la réalisation de séquences plus complexes ou plus longues. Pour cela, il faut:

- réduire le nombre de fonctions de visualisation communes à plusieurs sous-mémoires pour réduire le nombre de points $a \varepsilon A$ où il y a incomptabilité;

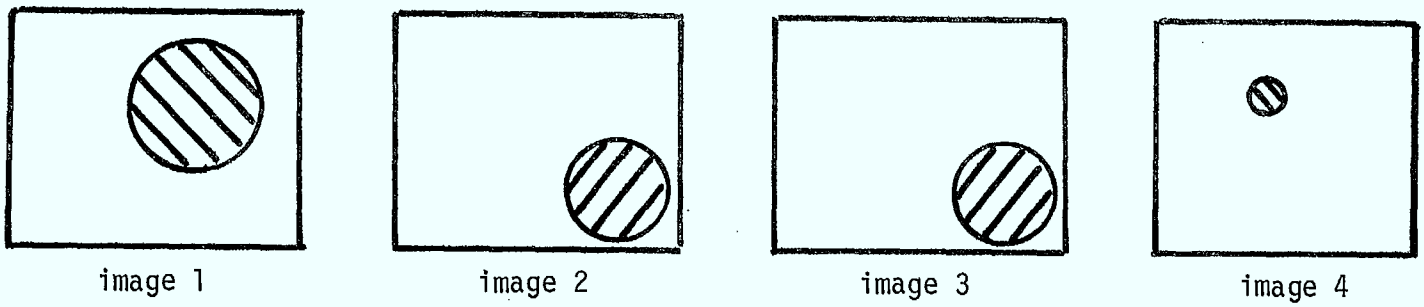
- réduire le nombre d'intensités sur les images, ce qui diminuera les risques d'incomptabilité;

- réduire le nombre de sous-mémoires utilisées, ce qui réduit le nombre de points $a \varepsilon A$ où il y a possibilité d'incomptabilité.

(a) sous-mémoires utilisées



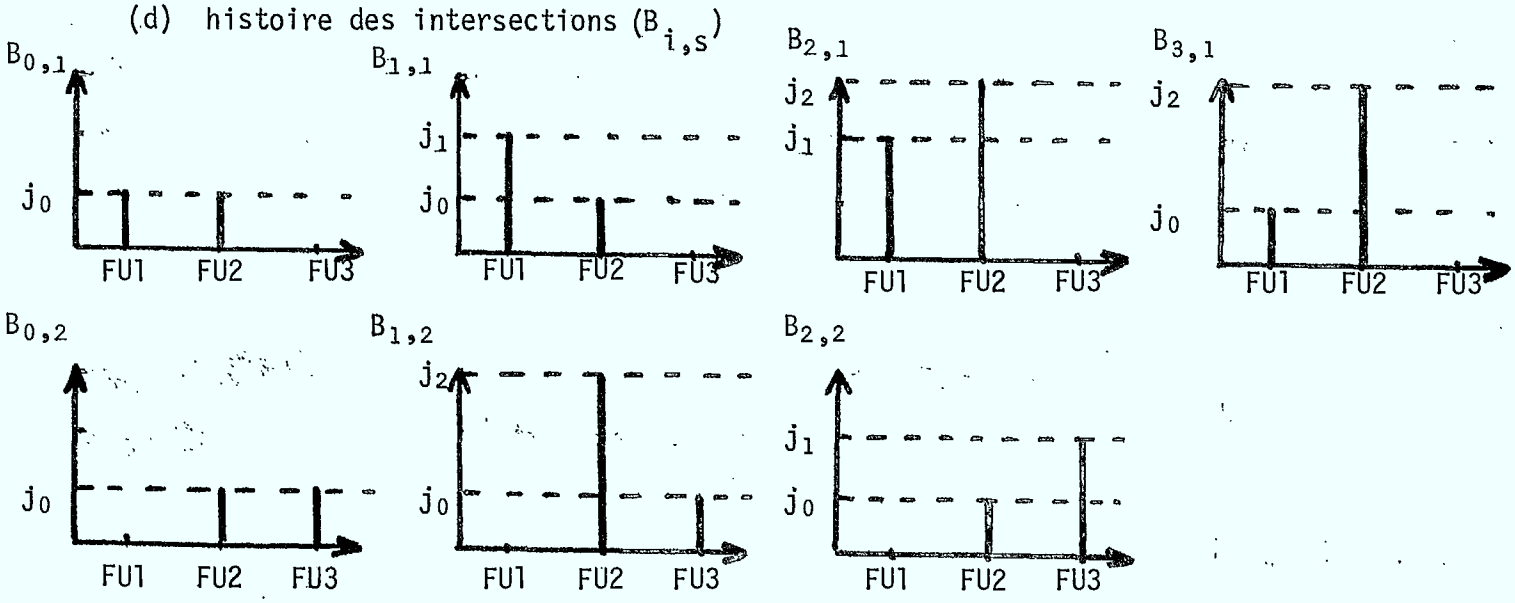
(b) séquence désirée



(c) assignation des fonctions de visualisation

image 1 ⟷ FU1
image 2
image 3 ⟷ FU2
image 4 ⟷ FU3

(d) histoire des intersections ($B_{i,s}$)



(e) assignation des numéros finaux d'intersection

dans SM1

initial		final
i_0	→	i_0
i_1	→	i_1
i_2	→	i_2
i_3	→	i_3

dans SM2

initial		final
i_0	→	i_0
i_1	→	i_3
i_2	→	i_1

(f) sous-mémoire après réassignation des numéros d'intersection

(g) fonctions finales de visualisation

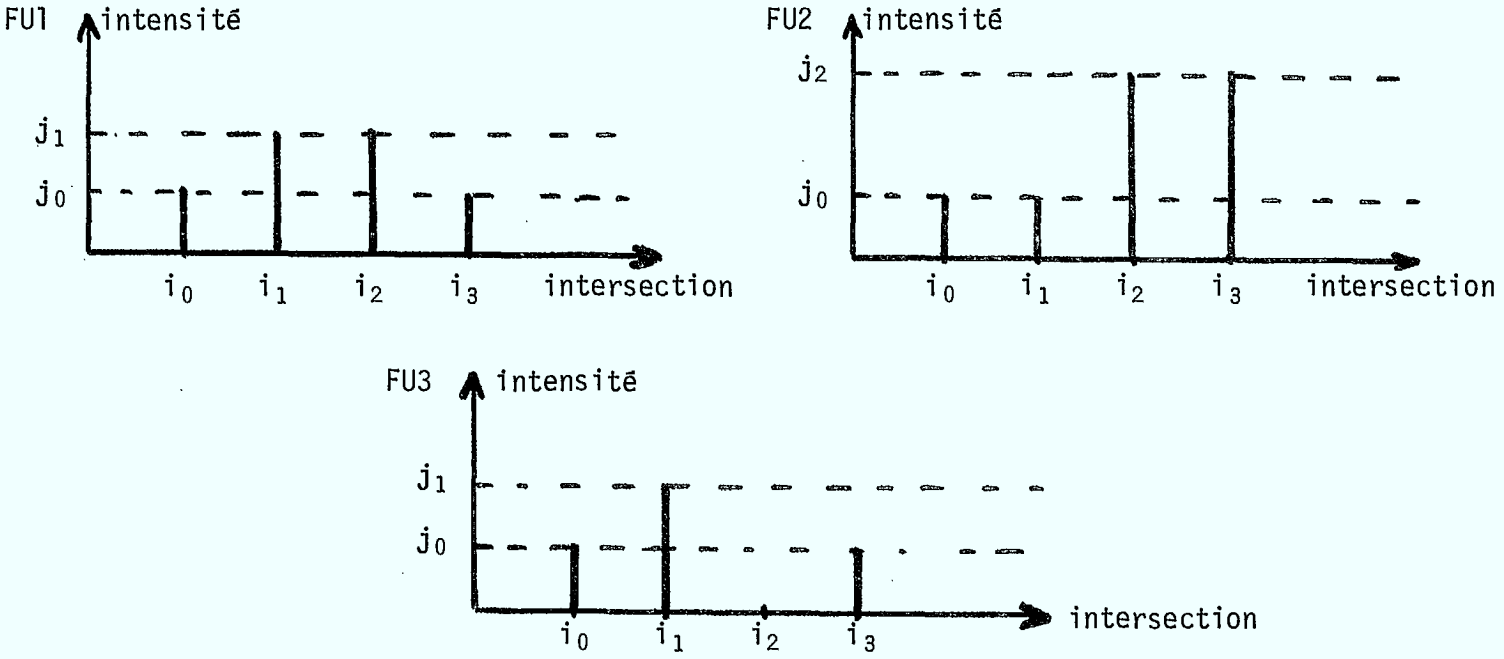


FIGURE 3.2.3.2: Exemple de réassignation de numéros d'intersection en utilisant les histoires des intersections

3.2.4 Programme de détermination des histoires

Ce programme est essentiellement la réalisation de l'algorithme de détermination des histoires en Pascal. La seule différence entre le programme et l'algorithme est qu'il n'y a pas deux tables pour garder en mémoire les histoires, mais plutôt une structure qui s'apparente un peu à celle d'un arbre. Soit n_t le nombre d'histoires au temps t:

$$n_t \leqslant n_{t+1}$$

le nombre de nouvelles histoires est supérieur ou égal au nombre d'anciennes et les nouvelles histoires sont composées en grande partie (pour 0 t $\leqslant$ t-1) des anciennes. Copier les anciennes histoires dans chacune des nouvelles nous force donc à garder plusieurs fois la même information. Pour éviter cela, les histoires sont représentés en mémoire par une structure apparentée un peu à celle de l'arbre. Il y a un vecteur de pointeurs dont chacun des éléments pointe un des noeuds de la structure. À chaque intersection $i_{(t-1)}$ correspond un pointeur dans ce vecteur, qui indique le numéro de l'histoire associé à cete intersection. Chacun des noeuds de la structure est formé d'un couple (t,j) et d'un pointeur indiquant où se trouve la suite de l'histoire.

L'exemple suivant illustre plus clairement comment sont organisés les histoires en mémoire.

soit: au temps t = 1

i_1 associée à $\{(t_1, j_3)\}$
i_2 associée à $\{(t_1, j_1)\}$

au temps t = 2

i_1 associée à $\{(t_1, j_3), (t_2, j_8)\}$
i_1 associée à $\{(t_1, j_7), (t_2, j_7)\}$
i_3 associée à $\{(t_1, j_3), (t_2, j_4)\}$
i_4 associée à $\{(t_1, j_3), (t_2, j_9)\}$
i_5 associée à $\{(t_1, j_7), (t_2, j_4)\}$

on aura au temps t = 2 la structure telle qu'indiquée à la figure 3.2.4.1

Pour savoir si $Q_{p(t-1)} + \{(t,f(p,t)\}$ existe, il faut déterminer si une histoire $Q_{p(t-1)}$ suivie d'une intensité f(p,t) a déjà été rencontrée. Afin de déterminer le plus rapidement possible si le couple $(i_{(t-1)},j)$ existe, il a été décidé d'utiliser un arbre binaire. Pour chacune des intersections $i_{(t-1)}$, ces arbres contiennent les "sous-intersections" i_t ainsi que les intensités associées à ces dernières.

Avec cette technique, il n'est pas nécessaire de déterminer tout de suite les nouvelles histoires. Après l'examen de tous les pixels d'une image, on peut ajouter les noeuds manquants pour faire les nouvelles histoires en examinant les arbres des intersections (pour chaque $i_{(t-1)}$)et en utilisant le vecteur pointant sur les anciennes histoires.

Dans l'exemple précédent, les arbres d'intersections auraient pu être tels qu'indique la figure 3.2.4.2.

Lorsque toutes les images ont été ajoutées, l'image statique est conservée sur disque et les histoires associées à chacune des intersections de cette image sont mises dans un fichier.

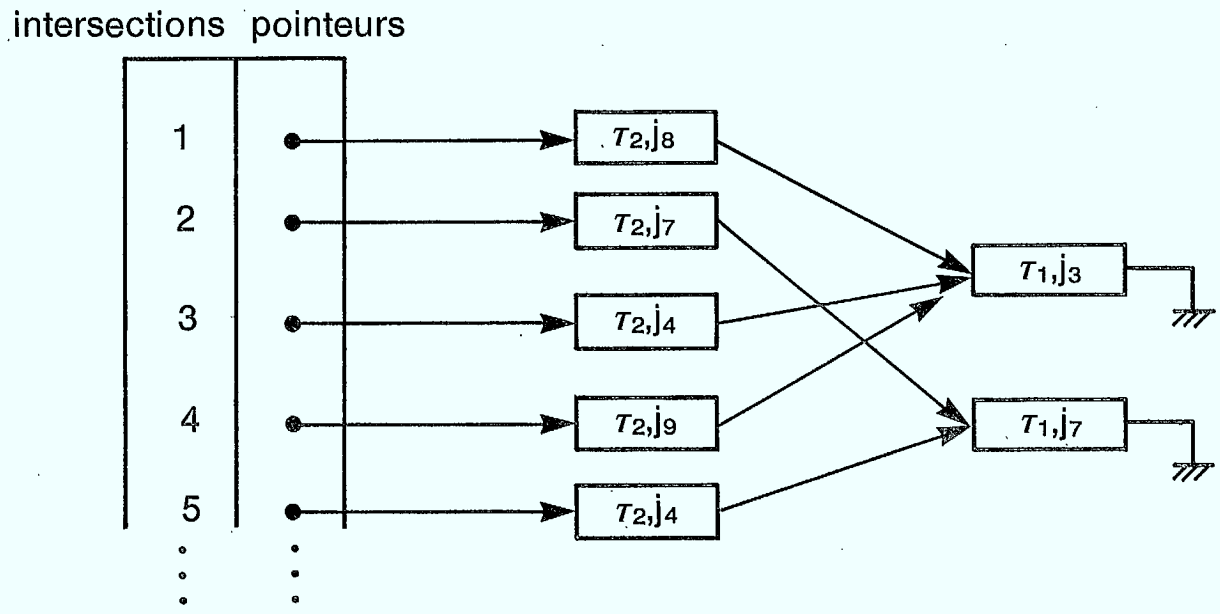


Fig. 3.2.4.1: Structure en arbre de l'histoire des intersections.

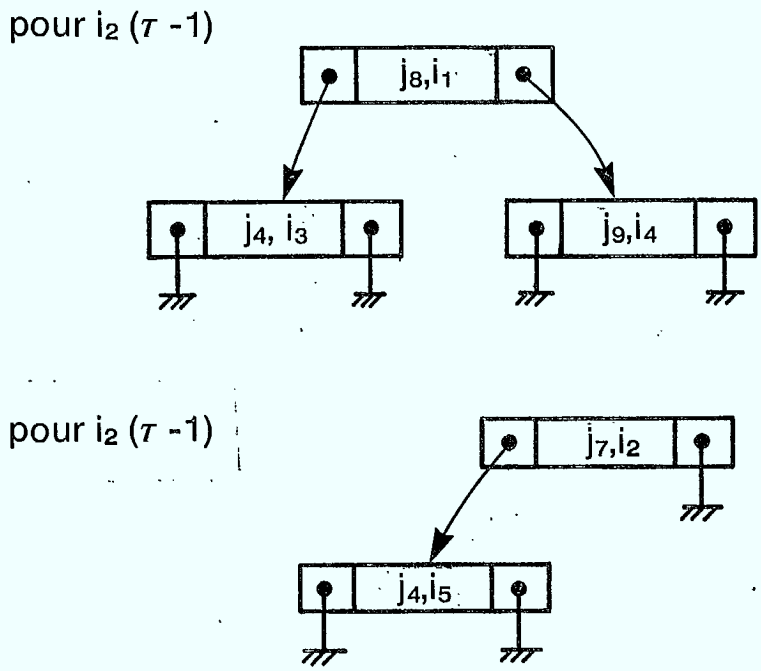


Fig. 3.2.4.2: Arbres d'intersections.

3.3 Programme de génération des fonctions de visualisation

Ce programme produit les tables de visualisation (tables de couleurs) pour une ou plusieurs sous-mémoire (s) (image (s) statique (s)).

Deux fonctions sont remplies par ce programme:

- il lit les fichiers histoires pour chacune des sous-mémoires et assigne un numéro d'intersection final à chacune des histoires différentes. Ensuite, il change les numéros d'intersection des sous-mémoires par les numéros finaux;

- à partir des histoires des intersections finales, il produit les tables de visualisation pour le groupe de sous-mémoire demandé. Ces tables ne sont rien d'autre qu'un vecteur de 256 bytes dont les éléments représentent l'intensité de l'intersection à laquelle il est associé.

3.4 Programme d'animation de la séquence

Après avoir crée l'image finale, contenant toutes les images, il faut maintenant pouvoir la visualiser. C'est ici qu'apparaît vraiment la différence entre une animation préétablie, où les numéros d'apparition des images sont invariants, et une animation interactive, où l'ordre de défilement dépend des demandes de l'usager.

Deux programmes ont donc été développés, un pour chaque cas. Dans le premier cas, les ordres sont lus dans un fichier de données, tandis que dans le second, une procédure devra bien sûr être écrite par l'usager, et être incluse dans le programme d'animation, dont elle constituera la partie la plus importante. Le programme présenté pour

l'animation "interactive" a donc surtout valeur d'exemple, et il est probable que l'usager désirant créer une séquence interactive réécrira une grande partie du programme en fonction de ses besoins propres.

Les deux programmes ont néanmoins une structure générale assez similaire, comme le montre la figure 3.4.1; ils sont séparés en deux parties bien distinctes, la préparation et l'exécution de l'animation.

La préparation consiste à lire dans un fichier de données les noms de l'image, des fonctions de visualisations et des fonctions de pseudocouleur, à configurer le système de visualisation pour qu'on puisse voir un quart de l'image zoomée et à transmettre l'image vers la mémoire tampon. Il faut ensuite lire toutes les fonctions de visualisation. Si on peut, c'est-à-dire s'il n'y en a pas trop, on les garde en mémoire sinon, il va faloir scinder l'animation en sous-séquences, et placer les fonctions de visualisation dans un fichier à accès direct, d'où on les retirera chaque fois qu'on en a besoin.

Dans le cas d'une animation préétablie, il faut aussi lire tous les ordres de défilement. Pour cela, un macro-langage a été développé, où chaque ordre autorisé est représenté par un numéro. Ces numéros sont lus et placés dans un vecteur, dont chaque valeur sera interprétée au moment de l'animation.

La détermination des ordres autorisés a été faite en fonction de 2 critères; en effet, on veut le plus d'ordres autorisés possibles, pour ne pas être limité, mais pas trop quand même, pour que leur interprétation soit très rapide.

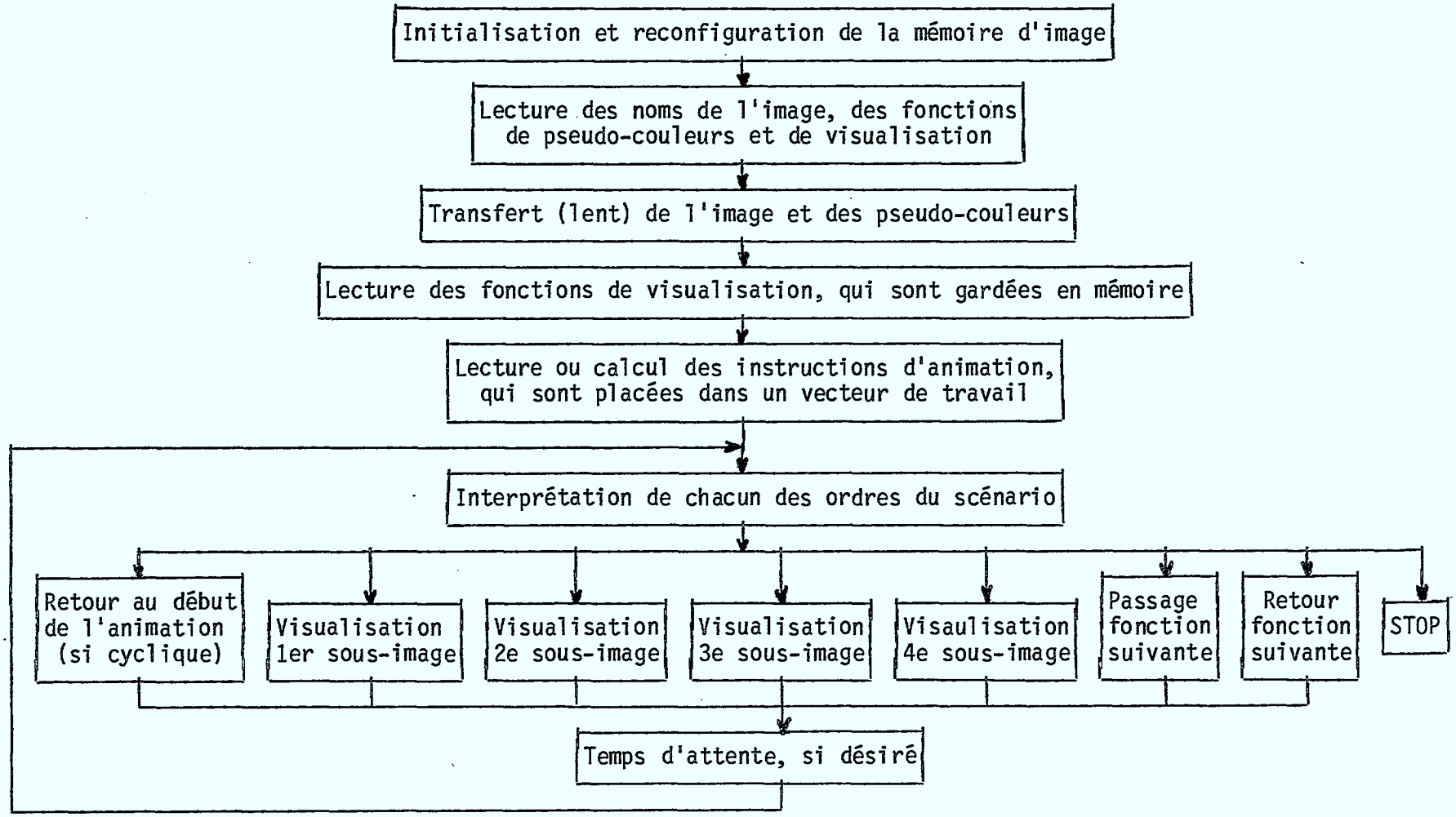


FIGURE 3.4.1: Organigramme du programme de visualisation de la séquence animée

Finalement, les seuls ordres retenus ont été:

- quatre ordres de sélection d'une sous-fenêtre;
- un ordre de passage à la fonction de visualisation suivante;
- un ordre de retour à la fonction de visualisation précédente;
- un ordre de retour au début de la séquence permettant une animation cyclique.

Tout autre ordre provoque l'arrêt du programme.

La vitesse de défilement des images est choisie par l'usager au début du programme et un temps d'attente peut être introduit entre l'envoi de deux ordres, permettant de ralentir l'animation sur l'écran. Ceci peut être important, par exemple, dans l'enseignement, pour pouvoir décomposer une action en plusieurs phases, et l'expliquer plus facilement. Toutefois, un temps d'attente trop long nuira à la qualité de l'animation, en montrant un mouvement saccadé.

Les deux programmes de visualisation de l'animation ont été écrits dans des langages différents; celui correspondant à une séquence animée prédéterminée a été écrit en FORTRAN 77 pour faciliter les communications avec tous les sous-programmes FORTRAN qu'il appelle, tandis que celui de la tour de Hanoï a été écrit en PASCAL, langage plus approprié pour traiter un problème récursif.

IV. LES SÉQUENCES ÉDITÉES

4.1 Choix des séquences

Le choix des exemples à réaliser a d'abord été fait en fonction de l'utilisation possible future de la méthode d'animation par table de couleurs. Deux domaines où cette méthode a de fortes chances de s'implanter sont:

- l'enseignement assisté par ordinateur, où un peu d'animation suffit à rendre un cours beaucoup plus clair;

- les jeux vidéos dont la faible qualité graphique pourrait être notablement améliorée par cette méthode.

Nous avons donc décidé de faire une démonstration dans chacun de ces domaines:

- pour l'enseignement, le fonctionnement d'un moteur à quatre temps, exemple typique d'une animation cyclique de durée infinie;

- pour le jeu vidéo, la résolution du problème des tours de Hanoï, exemple non cyclique et non prédéterminé.

Ces deux séquences très différentes donnent un bon aperçu de la complexité du mouvement, de la qualité des images, et de la longueur de l'animation que l'on peut espérer réaliser avec la méthode des tables de couleurs. Pour chaque séquence, il a fallu créer une image de base, des fonctions du temps pour le déplacement de chaque objet et un scénario [12], [13] et [14], pour relier l'image aux déplacements donnés.

4.2 Fonctionnement d'un moteur à 4 temps

4.2.1 Description de l'image de base

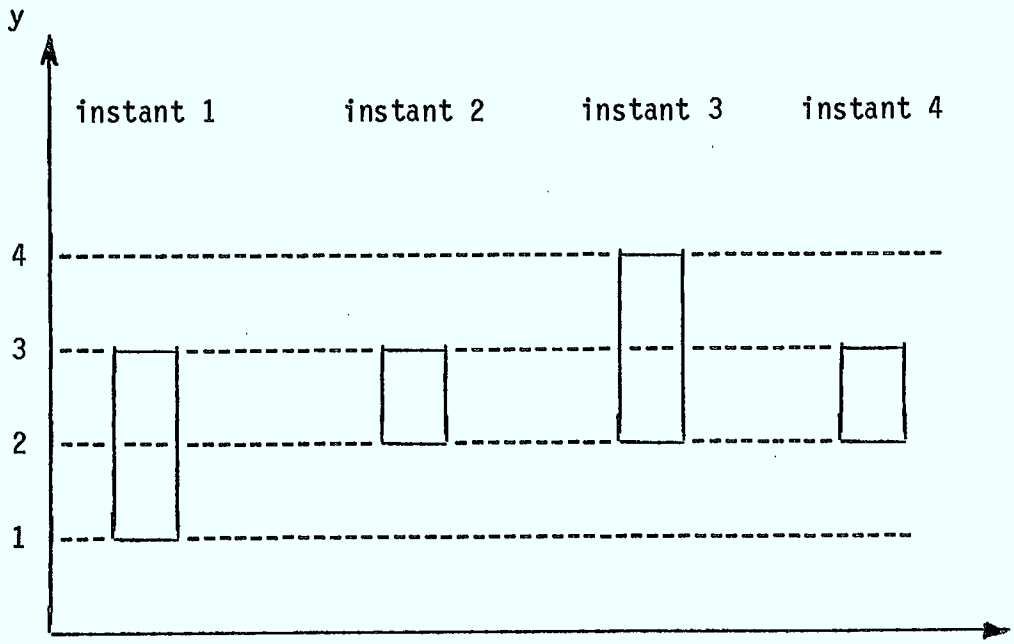
Afin de rendre le fonctionnement d'un moteur clair, le modèle utilisé a été très stylisé. C'est un moteur à quatre cylindres, vu en coupe, montrant les quatres pistons, avec un arbre à cames en tête,

pour que le mouvement soit mieux visible. Certaines parties ont été volontairement omises, comme l'huile, afin de rendre l'image plus facile à comprendre. Néanmoins, un grand nombre d'objets mobiles a été conservé, afin de bien montrer comment le mouvement d'une pièce fait bouger les autres; le piston actionne la bielle qui fait tourner le vilebrequin, celui-ci entraîne une chaîne reliée à l'arbre à cames. Les cames poussent les soupapes d'admission et d'échappement, permettant ainsi l'arrivée du mélange détonnant qui fait avancer le piston.

4.2.2 Calcul du mouvement des pièces et détermination des positions intermédiaires

Le plus "gros" mouvement, au sens de la longueur du déplacement, est celui du piston. C'est donc le déplacement du piston qui détermine le nombre d'images intermédiaires nécessaires à l'obtention d'une animation de qualité. Afin d'obtenir un déplacement maximum entre deux images consécutives de moins de 10 pixels sur 512, nous avons décidé de mettre 12 positions pour la descente du piston. Comme il faut deux aller et retour, du piston pour faire un tour de l'arbre à cames, cela donne un nombre total d'images pour la séquence de 48, nombre assez important, mais qui assure un mouvement sans à coups.

Connaissant le nombre total d'images désiré, nous avons alors pu calculer les positions intermédiaires de tous les objets et construire le scénario, qui indique à chaque instant les positions de tous les événements. Si la dimension apparente des événements varie au cours du temps, comme c'est le cas pour les cames, le scénario indique les positions du haut et du bas de la pièce. Tous les déplacements étant verticaux, aucune indication horizontale n'est nécessaire. La figure 4.2.2.1 donne un exemple simple de scénario représentant le mouvement d'une came.



Scénario:

Came haut :	3	3	4	3
bas :	1	2	2	2

FIGURE 4.2.2.1: Explication d'un scénario sur un exemple simple.

Le scénario du moteur a été construit petit à petit, en calculant toutes les positions de toutes les pièces mobiles, pour finalement devenir un tableau de 48 lignes (correspondant aux 48 positions intermédiaires) de 67 colonnes (correspondant aux 67 parties mobiles).

Les positions des différents éléments ont été obtenues de manières variées:

- à partir des équations de leur mouvement pour le piston, la bielle et le vilebrequin;

- d'après les valeurs données par la littérature, pour les cames et les soupapes. Les déplacements ont néanmoins été notablement augmentés afin d'être visibles sur l'écran;

- à la suite de tests judicieux pour les maillons de la chaîne.

4.2.3 Organisation en mémoire

Après avoir créé les 48 images correspondant à un tour complet de l'arbre à cames, il restait à les répartir dans les quatres sous-mémoires. Pour cela, seules de grandes lignes de raisonnement existent, et la répartition se fait de manière assez heuristique à partir de ces idées directrices. Dans le cas du moteur, plusieurs essais de répartition ont été réalisés jusqu'à ce que l'une des configurations en mémoire parvienne à contenir toutes les images sans dépasser le seuil fatidique des 256 intensités disponibles.

La répartition qui a donné les meilleurs résultats n'est pas du tout celle que nous attendions; en effet, c'est celle qui consiste à mettre dans la même sous-mémoire les 48 images! Cette organisation en

mémoire pour le moins surprenante a néanmoins une expliquation. On a vu au paragraphe 3.2 que deux facteurs antogonistes interviennent lors de la répartition. D'une part, il faut répartir les images dans les sous-mémoires pour économiser des intersections, mais d'autre part, le passage d'une sous-mémoire à une autre interdit la réutilisation de certaines intensités présentes dans la première sous-mémoire dans la deuxième sous-mémoire, à cause de l'obligation de garder la même fonction de visualisation. La structure assez particulière de l'image du moteur fait que beaucoup d'événements d'images différentes se recouvrent sans toutefois avoir les mêmes histoires, si on les met dans une même sous-mémoire, ce qui permet ainsi d'économiser des intensités. Tandis que s'ils étaient placés dans des sous-mémoires différentes, il faudrait qu'ils aient des histoires compatibles pour qu'ils puissent réutiliser les mêmes intensités.

Cette répartition en sous-mémoire est particulière au moteur, et ne saurait en aucun cas, être généralisée à toutes les animations possibles. La répartition en mémoire dépend énormément du contenu sémantique des images, ainsi que du mouvement des objets. Si toutes les images ont été placées dans la même sous-mémoire pour le moteur, ce n'est pas le cas pour l'exemple suivant.

4.3 Jeu de la tour de Hanoï

4.3.1 Description générale du jeu

Le matériel utilisé pour ce jeu consiste en un support avec trois tiges verticales, et un certain nombre de disques de diamètres différents, posés sur une tige, ainsi que le montre la figure 4.3.1.1.

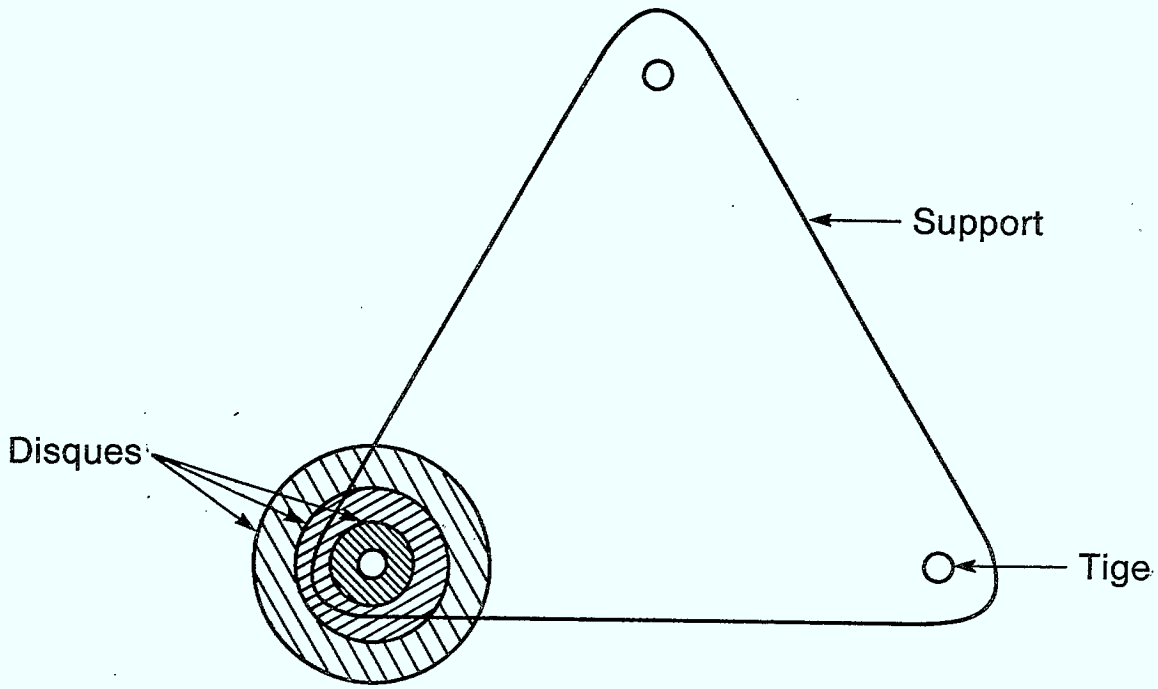


Fig. 4.3.1.1: Vue de dessus du jeu de la Tour de Hanoi

Le jeu consiste à déplacer la "tour", formée de l'empilement de tous les disques, ordonnés par diamètre décroissant vers le haut, d'une tige à une autre. Pour cela, on ne peut déplacer qu'un seul disque à la fois, il faut que ce soit le disque le plus haut sur la tige de départ, et on ne peut jamais déposer un disque sur un disque plus petit.

La figure 4.3.1.2 explique la manière de procéder pour déplacer une tour de trois disques.

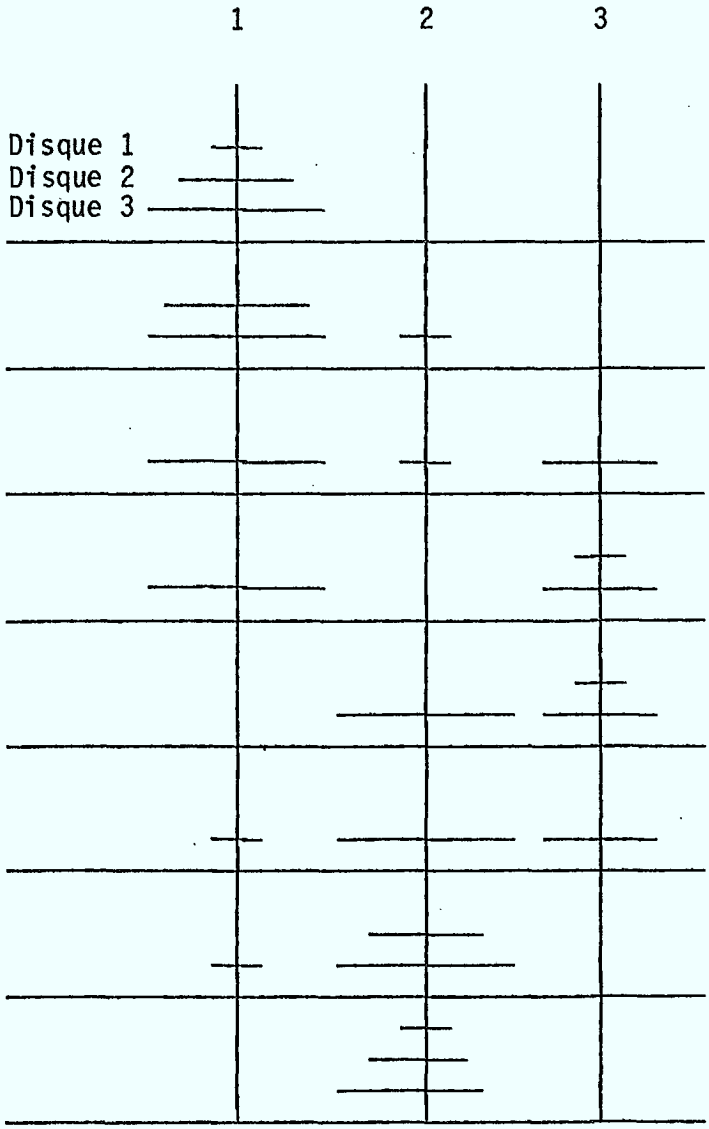
4.3.2 Adaptation à la méthode des tables de couleurs

Comme nous voulions réaliser un jeu, et non pas une démonstration magistrale de la résolution du problème de la tour de Hanoï, il fallait introduire une participation interactive de l'usager.

Pour cela, l'usager devait avoir plus de possibilités de choix que juste ceux des tiges de départ et d'arrivée, afin de ne pas se lasser trop vite. Néanmoins, nous avons préféré faire jouer l'ordinateur, c'est-à-dire que les mouvements sont calculés par l'ordinateur et non pas décidés par l'usager.

Toutefois, seule une modification minime dans le programme de visualisation est nécessaire pour permettre à l'usager de décider du mouvement.

Pour avoir une participation minimale de l'usager, nous avons décidé de lui laisser choisir les positions de départ de chaque disque. Ceci modifie un peu le jeu, en le généralisant. De plus, le nombre minimal de coups pour arriver à la solution, est en général moins grand si les positions de départ sont quelconques, que si tous les disques sont sur la même tige.



Position de départ: Les disques 1, 2 et 3 sont sur la tige 1.

1ère étape: on déplace le disque 1 de la tige 1 à la tige 2.

2e étape: on déplace le disque 2 de la tige 1 à la tige 3.

3e étape: on déplace le disque 1 de la tige 2 à la tige 3.

4e étape: on déplace le disque 3 de la tige 1 à la tige 2.

5e étape: on déplace le disque 1 de la tige 3 à la tige 1.

6e étape: on déplace le disque 2 de la tige 3 à la tige 2.

7e étape: on déplace le disque 1 de la tige 1 à la tige 2. Position d'arrivée: tous les disques sont empilés dans l'ordre sur la tige 2.

FIGURE 4.3.1.2: Déplacement d'une tour de trois disques de la tige 1 à la tige 2.

En effet, le nombre de mouvements croît de manière exponentielle avec le nombre de disques, puisque si on a N disques, il faut (2^N-1) mouvements pour déplacer une tour d'une tige à une autre [15], [16], [17]. Pour que l'usager puisse prévoir facilement les mouvements et les interpréter sans trop de difficultés, il ne faut pas qu'il y en ait trop. Mais pour que la résolution ne lui semble pas trop évidente, il en faut quand même un certain nombre. Le compromis finalement retenu consiste à animer 4 disques, ceci pouvant créer jusqu'à $(2^4-1 = 15)$ déplacements de disques.

Chaque déplacements ayant une durée de plusieurs secondes, l'animation résultante aura une durée d'environ une minute.

L'adaptation du jeu des tours de Hanoi à notre méthode d'animation nous a donc conduit à modifier légèrement les règles du jeu pour avoir une position de départ quelconque, et à ne considérer qu'une tour de quatre disques.

4.3.3 Algorithme de résolution du problème

Le problème de la tour de Hanoi est souvent utilisé dans les cours d'initiation à l'informatique pour démontrer la puissance de la récursivité. En effet, l'algorithme récursif est extrêmement simple, élégant et concret. Il est montré sur la figure 4.3.3.1 où l'on voit bien qu'il s'appelle lui-même deux fois.

Toutefois, malgré son élégance apparente, cet algorithme n'est pas toujours pratique, et en particulier n'est pas directement utilisable dans le cas de positions de départ quelconques. Si l'on a beaucoup de disques et donc un nombre important de mouvements à générer, cet algorithme n'est pas non plus le plus rapide à exécuter.

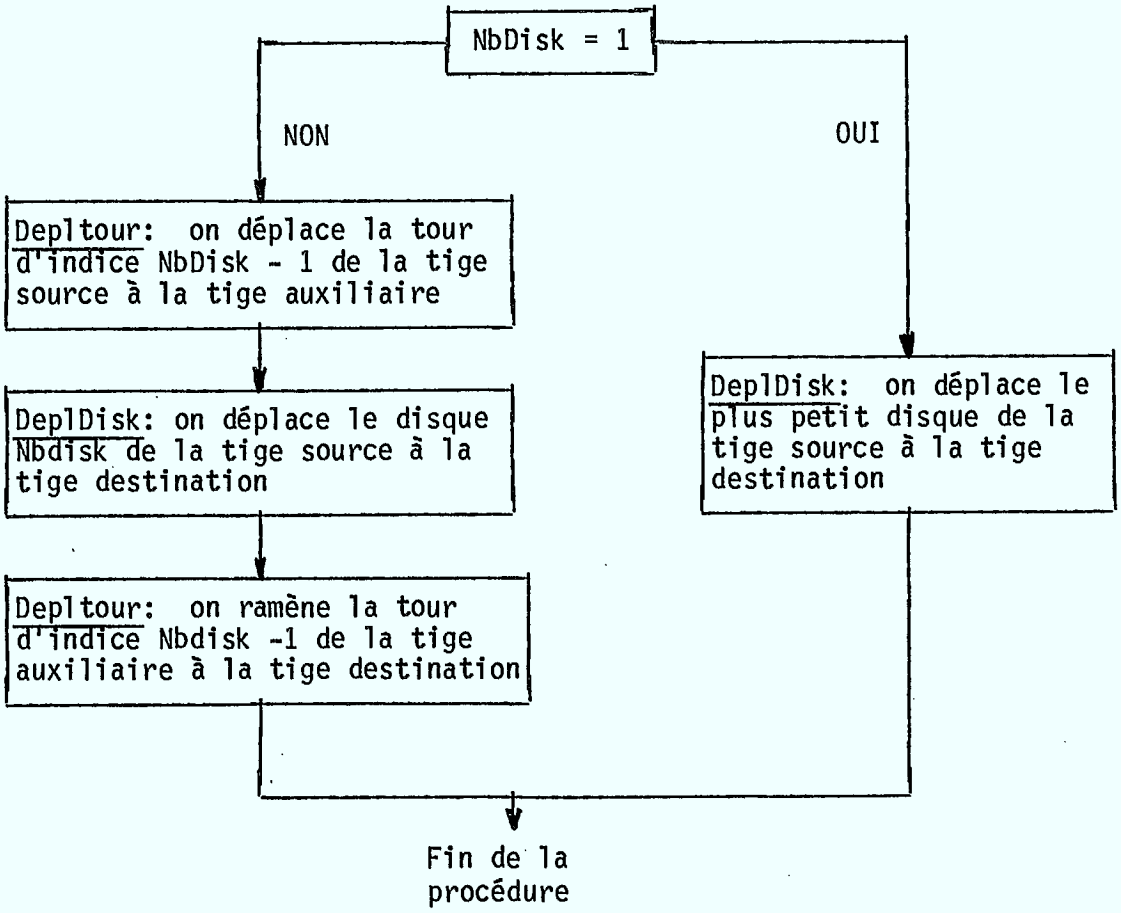


FIGURE 4.3.3.1: Algorithme de la procédure Depltour.

Plusieurs auteurs ont alors essayé de comprendre la solution, plutôt que le problème, pour trouver d'autres manières d'y arriver. P. Hayes [15] a remarqué qu'il y avait toujours alternance entre un mouvement du petit disque, et un mouvement d'un autre disque. M.C. Er [16] a ensuite analysé la solution de façon beaucoup plus complète et a construit un algorithme pour l'obtenir, dans lequel la séquence de mouvements à effectuer est obtenue de façon originale sous forme de chaîne de bits. Cette chaîne, qui a des propriétés de symmétrie par rapport au mouvement central (correspondant à DeplDisk dans l'algorithme récursif) représente chaque mouvement par un bit, qui dit dans quel sens (trigonométrique ou non), il va se faire. Cet algorithme ne s'applique bien sûr pas au problème général, qui à notre connaissance n'avait encore été abordé par personne, mais qui présentait une idée intéressante: quand un disque est déplacé par DeplDisk, les autres déplacements sont facilement obtenus par symétries successives. Ceci nous a fait définir une entité instinctive appelée un "gros mouvement" et qui est le mouvement qui permet de se ramener à un problème de dimension inférieure. Un "gros mouvement" déplace "le plus gros disque mal placé de sa tige source", vers "sa tige destination".

Après cette approche intuitive, revenons à une démarche rigoureuse. Pour chaque mouvement, appelons:

- Source: la tige où se trouve le disque ou la tour à déplacer.
- Destination: la tige où on veut le (respectivement la) déplacer.
- Auxiliaire: la troisième tige.
- K: le numéro du disque, K variant de 1 (plus petit disque) à Nbdisk (plus gros disque).
- Dim: la dimension d'une tour de Hanoi, c'est-à-dire le nombre de disques qui la constituent. C'est aussi le numéro du plus gros disque.

-- Propriété:

Pour pouvoir déplacer le disque k il faut que les (k-1) disques plus petits soient sur la tige auxiliaire.

-- Corollaire:

Pour pouvoir déplacer le disque k, il faut que les (k-1) disques plus petits forment une tour de dimension (k-1) sur la tige auxiliaire.

-- Démonstration de la propriété:

Raisonnement par l'absurde. Si un disque plus petit était sur la tige source, il devrait être sur le disque d'indice k, d'après la règle du jeu. Dans ce cas, on ne pourrait pas soulever le disque k pour le déplacer. De même, si un disque plus petit que k était sur la tige destination, toujours d'après la règle du jeu, on ne pourrait pas poser le disque k dessus: le déplacement ne pourrait donc pas se faire. Aucun disque plus petit que le disque k ne peut donc se trouver ailleurs que sur la tige auxiliaire.

Le corrolaire découle de la propriété. Si tous les disques d'indices inférieurs à k sont sur la tige auxiliaire, il faut qu'ils soient ordonnés par taille décroissante pour satisfaire à la règle du jeu et ils forment donc une tour de Hanoi. Le plus gros disque de cette tour est le plus gros disque d'indice inférieur à k, donc le disque (k-1), et la tour a une dimension (k-1).

Après ces préliminaires, passons à la résolution du problème proprement dit, qui va être faite en deux étapes. D'abord, on teste si le plus gros disque est "bien" placé, c'est-à-dire si la tige où il se trouve est sa tige "destination". S'il y est, on effectue ce test au

disque de dimension immédiatement inférieure. S'il n'y est pas, il faudrait le déplacer de la tige où il se trouve ("source") vers la tige où l'on veut voir arriver la tour à la fin ("destination"). Pour cela, il faut que tous les autres disques soient sur la tige auxiliaire, ce qui nous ramène à un problème de dimension NbDisk-1. La tige auxiliaire du problème d'indice NbDisk devient la tige destination du problème de dimension NbDisk-1 puisque c'est sur elle que l'on veut placer les autres disques. Ce test est effectué pour tous les disques. La figure 4.3.3.2 résume l'algorithme de détermination des "gros mouvements".

À l'opposé de la première étape, où l'étude se faisait du plus gros au plus petit disque, la deuxième étape suit une démarche du plus petit au plus grand, assez similaire à l'algorithme récursif. C'est dans cette partie que la tour est reconstruite étape par étape, en utilisant la procédure DeplDisk, qui effectue le "gros mouvement" calculé précédemment. La plus grosse différence est que le premier appel à la procédure Delptour ne déplace pas forcément une tour d'indice immédiatement inférieur, car il se peut que certains disques soient déjà placés sur la tige auxiliaire.

4.3.4 Réalisation

Le gros problème pour réaliser une animation du jeu des tours de Hanoi est celui du nombre d'images. En effet, chaque disque doit pouvoir se déplacer de n'importe quelle tige vers chacune des autres, sans que ce déplacement ne soit trop brusque. On voit tout de suite que le nombre total d'images à générer est très grand. Afin de maximiser le nombre d'images et donc la qualité de l'animation, nous avons cherché la répartition en mémoire la plus astucieuse pour avoir le moins d'intersections possibles, pour un nombre d'images fixé. Cette

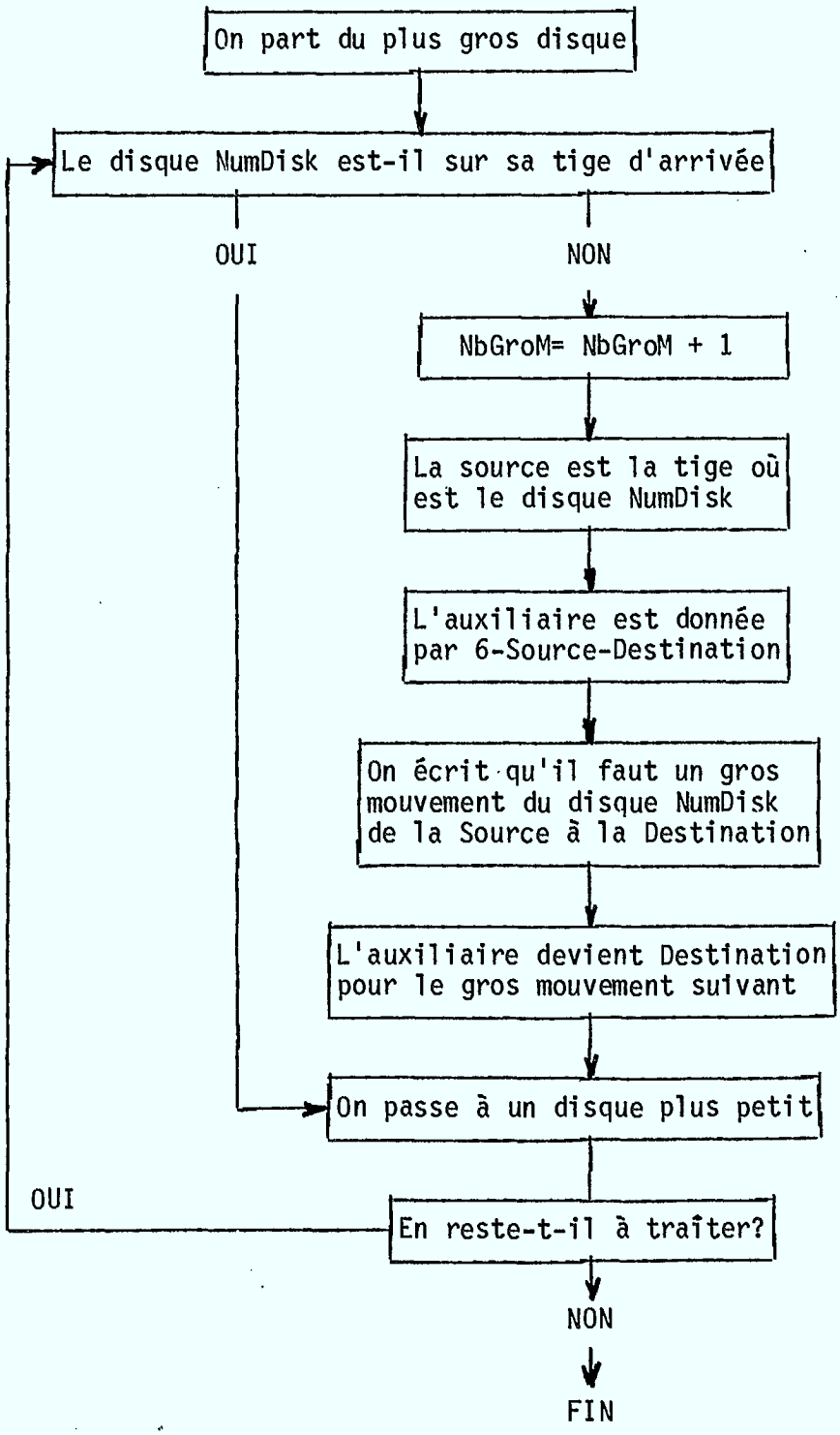


FIGURE 4.3.3.2: Organigramme de la détermination des gros mouvements

répartition optimale est celle qui alloue une sous-mémoire à chacun des disques, c'est-à-dire que la taille du disque est le seul critère à employer lors du choix de la sous-mémoire à utiliser. Ceci est dû au fait qu'il faut minimiser le nombre de changements de sous-mémoire autant que possible, et donc éviter de changer de sous-mémoire lors du déplacement d'un disque. La figure 4.3.4.1 montre la façon dont les mouvements des disques ont été répartis entre les sous-mémoires.

Le nombre d'images possible par sous-mémoire a ensuite pu être facilement déterminé, vu la forme très simple des objets. Il varie d'une sous-mémoire à une autre, bien sûr, car les petits disques engendrent moins d'intersections que les gros, ce qui permet donc d'en placer plus par sous-image ce qui est important. En effet, à déplacement égal, le mouvement apparaît beaucoup plus facilement saccadé avec un petit objet qu'avec un plus gros. Le nombre total d'images ajoutées a été de 45 dans la première sous-mémoire, 24 dans la deuxième, 24 dans la troisième et 18 dans la quatrième, soit un total de 111 images. Ce nombre très élevé s'explique par la simplicité des positions intermédiaires d'un disque sans aucune mesure avec la complexité des images du moteur.

Le programme de visualisation de cette animation a été écrit en Pascal essentiellement, avec une petite partie en Fortran, qui est en fait le "Driver" du système de traitement d'images. Cette partie, dépendant très fortement du périphérique de sortie, est donc peu transportable et appelle un certain nombre d'autres programmes Fortran. Le corps du programme, de cette manière est totalement indépendant du périphérique de sortie utilisé. Cette structure où les tâches sont bien séparées est indispensable pour assurer une maintenance facile et laisse présager une transportabilité aisée. La figure 4.3.4.2 donne l'organigramme général du programme de visualisation de l'animation de la tour de Hanoi.

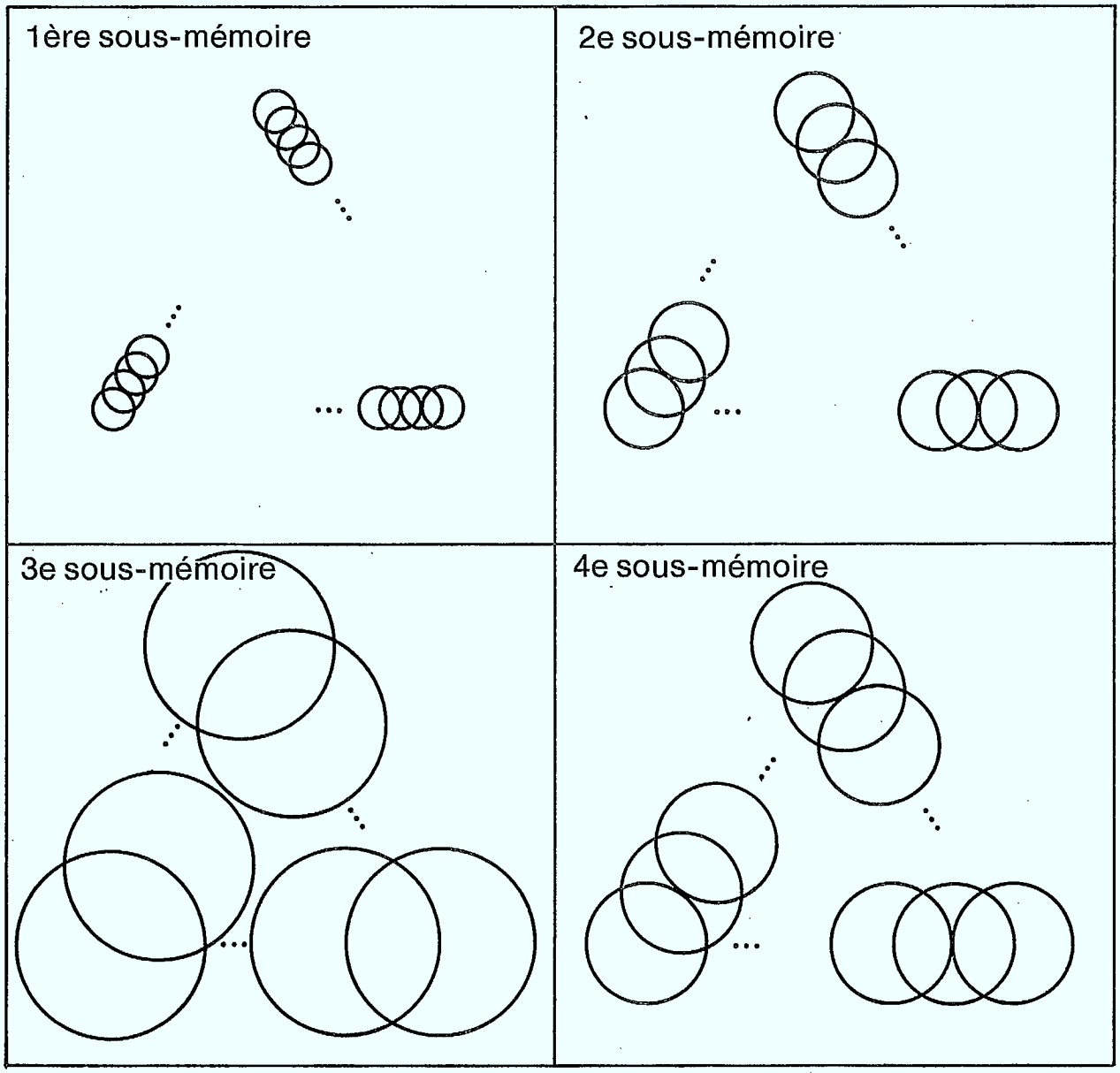


Fig. 4.3.4.1: Répartition en mémoire des images des disques en mouvements.

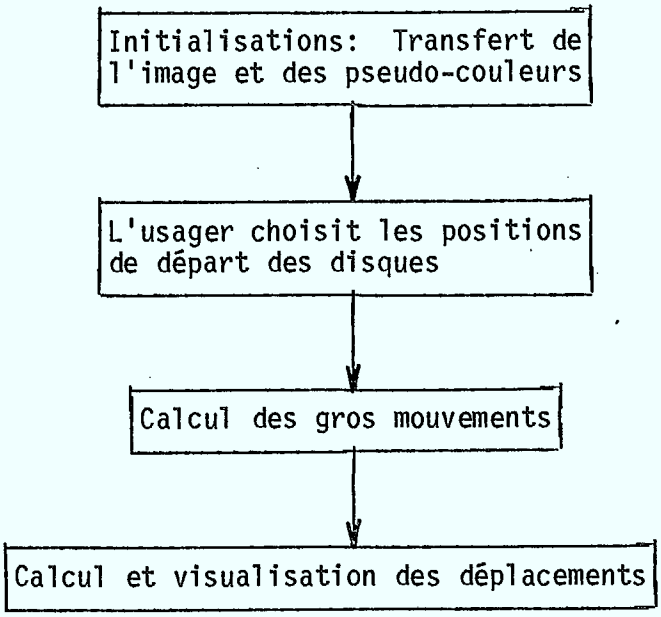


FIGURE 4.3.4.2: Organigramme général du programme des tours de Hanoi

La visualisation du déplacement d'un disque s'effectue en deux temps:

- Clignotement de celui-ci pendant quelques instants indiquant ainsi quel disque va se déplacer.

- Déplacement rapide de celui-ci de la tige source à la tige destination.

V. CONCLUSIONS

La première phase d'étude de la technique d'animation par table de couleurs, résumée dans le présent document, a permis de montrer la viabilité de cette technique pour la production de séquences animées de type cyclique et non cyclique. La simplicité de mise en oeuvre de cette technique et le peu de moyens matériels requis en font une solution particulièrement attrayante pour des systèmes informatiques limités (micro-ordinateurs par exemple).

Une étude complète doit comporter nécessairement les aspects suivants:

(1) mise au point de la technique d'animation en temps réel;
(2) édition automatique des séquences animées (définition du scénario, choix des positions clefs, interpolation automatique);
(3) interface usager/système (embryon de langage de commande);
(4) expérimentation globale de la méthode et évaluation de ses performances.

En conséquence, il s'agira dans une seconde phase de se consacrer aux aspects (2), (3) et (4) énumérés ci-dessus.

En ce qui concerne l'édition automatique des séquences, il s'agira de choisir parmi les techniques disponibles celles qui sont particulièrement adaptables à la méthode des tables de couleurs ainsi qu'à des systèmes informatiques limités.

L'interface usager/système sera en grande mesure conditionnée par le choix de la technique d'édition retenue.

VI. BIBLIOGRAPHIE

[1] NEWMAN, W.M. and SPROULL, R.F., "Principles of Interactive Computer Graphics", McGraw Hill, 1979.

[2] BRAID, I.C., "The Synthesis of Solids Boundes by Many Faces", CACM 18 (4), April 1975.

[3] CROW, F.C., "Shadow Algorithm for Computer Graphics", Computer Graphics 11 (2), Summer 1977.

[4] WHITTED, T., "A Scan-Line Algorithm for Computer Display of Curved surfaces", Computer Graphics 12 (3), August 78.

[5] CATMULL, E., "The problems of Computer Assisted Animation", Computer Graphics 12 (3), August 78.

[6] WEINGERG, R., "Computer Graphics in Support of Space Shuttle Simulation", Computer Graphics 12 (3), August 78.

[7] LEROY, M., "A Color Animation System Based on the Multiplane Technique", Computer Graphics 11 (2), Summer 1977.

[8] SLOAN, K.R. and BROWN, C.M., "Color Map Techniques", Computer Graphics and Image Processing, Vol. 10, 1979.

[9] SHOUP, R.G., "Color Table Animation", Compute Graphics and Image Processing, Vol. 10, 1979.

[10] MACKAY, S.A., "Techniques for Frame Buffer Animation", Master Thesis, University of Waterloo, Ont., 1982.

[11] McKENZIE, E.L., "Interactive Computer Graphics in Science Teaching".

[12] GUEDJ, T., "Methodology in Computer Graphics", IFIP Workshop on Methodology in Computer Graphics, Seillac, France 1976.

[13] YARBROUGH, "*Café*: a non procedural language for Computer Animation", extrait de Pertinent Concepts in Computer Graphics.

[14] COURTIEUX, F. and POLIMER, B., "An interpreter for the interactive generation and animation of two-dimensional pictures", extrait de "Graphic Languages" de Rosenfeld.

[15] HAYES, P.J., "A note on the towers of Hanoi Problem", Comp. Journal, Vol. 20, No. 3, p. 282-285, 1977.

[16] ER, M.C., "A representaion Approach to the tower of Hanoi Problem", Comp. Journal, Vol. 25, No. 4, p. 442-447, 1982.

[17] RATSIMBAZAFY, C., DESANTIS, R.M. and HURTEAU, R., "Implantation des algorithmes optimaux de résolution du problème de la tour de Hanoi sur Micro-ordinateur", Publication interne, École Polytechnique de Montréal, 1984.