

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Réalisation d'un logiciel graphique interactif

Hoang, Hoa Dung

Award date:
1985

Awarding institution:
Universite de Namur

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

REALISATION
D'UN LOGICIEL GRAPHIQUE
INTERACTIF

Mémoire présenté par HOANG HOA DUNG
en vue de l'obtention du titre de
Licencié et Maître en Informatique.

Année académique 1984 - 1985

Je tiens à exprimer mes remerciements à
Monsieur J.P. LECLERC, promoteur du
mémoire, pour ses conseils et ses
encouragements.

TABLE DES MATIERES.INTRODUCTIONI. LES ELEMENTS DE BASE DES SYSTEMES GRAPHIQUES

1. Image graphique et ses caractéristiques
2. Composants matériels d'un système graphique
3. Composants logiciels des systèmes graphiques
 - 3.1. Les instructions graphiques: REGIS
 - 3.2. Le package de routines graphiques
 - 3.3. Le logiciel interactif graphique

II. LA LIBRAIRIE DE PRIMITIVES GRAPHIQUES: LYS

1. Classification des primitives
2. Primitives d'affichage
3. Primitives de contrôle
4. Primitives de mise en page
 - 4.1. Espace d'utilisateur et la fenêtre
 - 4.2. Espace d'affichage et la clôture
 - 4.3. Passage des coordonnées utilisateurs aux coordonnées écrans
 - 4.4. Découpage de la figure dans l'espace de visualisation
 - 4.5. Primitives de mise en page de LYS
5. Primitives de transformations géométriques
 - 5.1. Bases mathématiques du graphisme à deux dimensions
 - 5.1.1. La Translation
 - 5.1.2. La Mise en échelle (scaling)
 - 5.1.3. La Rotation
 - 5.2. Représentation de la transformation: coordonnées homogènes et représentation matérielle
 - 5.3. Les Primitives de transformations dans LYS

III. LE LOGICIEL GRAPHIQUE INTERACTIF: IRIS

1. Spécification de la machine virtuelle: IRIS

1.1. Notion d'état et action

1.2. La Machine IRIS

1.2.1. Etat Init

1.2.2. Etat Monitor

1.2.3. Etat Graphic

1.2.4. Etat Text

1.2.5. Conclusion

2. Structure des données : la représentation de l'image

2.1. La partie graphique

2.1.1. Représentation des sprites

2.1.2. Représentation des opérations

2.1.3. Représentation de la partie graphique

2.2. La partie texte

2.2.1. Représentation d'une chaîne de caractères éditée

2.2.2. Représentation de la partie texte

3. Structure dynamique: architecture et structure du programme

3.1. Architecture

3.2. Structure du programme

4. Manuel d'utilisation

INTRODUCTION

Depuis la préhistoire, l'homme a déjà su se servir des représentations graphiques pour communiquer avec les autres pour transmettre et stocker les connaissances.

L'utilisation des représentations graphiques se justifie par le fait que le contenu, représenté par des dessins, est facilement compréhensible, donc facilement communicable avec autrui.

A côté de sa fonction de communication, l'image joue souvent d'autres rôles, comme celui d'inventaire ou de modélisation du réel perçu. Ce rôle est particulièrement bien illustré dans la cartographie, dans le dessin technique et architectural.

De nos jours, l'utilisation de l'image graphique ne fait que se renforcer avec les technologies nouvelles. La nécessité de communiquer par des moyens graphiques est de plus en plus grande. Cependant la création des dessins est une entreprise quelquefois longue et délicate, nécessitant une certaine dextérité du dessinateur et requérant de nombreux outils: compas, équerre, rapporteur, stylo, règle...

Le besoin d'être assisté par un système graphique interactif se fait rapidement sentir et explique l'arrivée sur le marché d'un très grand nombre de logiciels graphiques pour toute gamme d'ordinateurs.

A côté de ce large éventail de logiciels, florit une littérature assez abondante et variée qui veut guider les concepteurs et réalisateurs de systèmes graphiques, en leur proposant des solutions aux divers problèmes et des éléments de réalisation.

Cependant, aussi variés que ~~ce~~ soient ces ouvrages, on retrouve un certain nombre de points communs et de constantes dans ces différentes propositions de solutions.

Dans le présent travail, nous voulons étudier les principaux problèmes rencontrés dans la conception des systèmes, et essayer de les résoudre. Par application des propositions de solutions, choisies pour leur simplicité et leur facilité de mise en oeuvre, nous essayons de construire, dans la première étape une librairie de primitives graphiques, puis dans une deuxième étape, de réaliser un logiciel graphique interactif, en se basant sur cette librairie.

Chapitre 1: Elément de base d'un logiciel graphique

Ce chapitre contient la présentation des principaux éléments de la communication graphique avec un ordinateur. Les principaux composants matériels et logiciels y sont décrits.

Chapitre 2: Librairie de routines graphiques

Dans ce chapitre nous étudions l'ensemble des primitives graphiques composant la librairie de routines graphiques.

Chapitre 3: Le logiciel graphique interactif

Le chapitre 3 est consacré à la description des principales caractéristiques du logiciel graphique interactif. Nous y décrivons la spécification fonctionnelle de la machine virtuelle: IRIS, ainsi que la structure des données et la structure des traitements du logiciel conçu. Un manuel d'utilisation se trouve en fin du chapitre en guise de mode d'emploi du programme.

I

ELEMENTS DE BASE DES SYSTEMES GRAPHIQUES

1. IMAGE GRAPHIQUE ET SES CARACTERISTIQUES.

Les images traitées par ordinateur appartiennent à différentes catégories:

- + l'image non figurative ou abstraite n'illustre aucun objet du monde physique.
- + l'image figurative qui vise à donner une représentation schématique des éléments reconnaissables. Des dessins techniques et architecturaux sont des images figuratives.
- + les graphiques qui concernent la ^{re}présentation d'information à l'aide d'un système de symboles défini conventionnellement. Un circuit électronique est représenté par un ensemble de symboles représentant diodes, transistors, résistances...

Nous intéressons dans le cadre de notre travail à ce dernier type d'image qui est beaucoup utilisé dans la pratique et qui s'exprime facilement dans une surface de visualisation à deux dimensions.

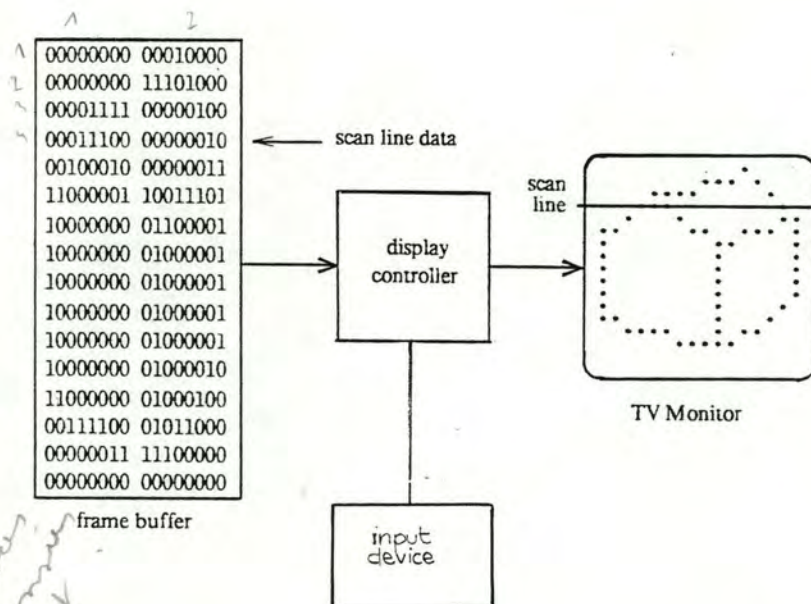
Une image graphique est composée d'éléments graphiques élémentaires que nous nommons "sprite". Chaque élément est caractérisé par

- + sa forme
- + son emplacement sur la surface de visualisation
- + sa taille
- + son orientation par rapport à un système d'axes
- + son intensité du gris (ou sa couleur)

Une image est parfaitement définie si on connaît tous les éléments graphiques élémentaires les composant ainsi que toutes les caractéristiques associées à chaque élément.

2. LES COMPOSANTS MATERIELS D'UN SYSTEME GRAPHIQUE.

Le système graphique est assez simple dans la construction. Il est souvent formé de 4 composants principaux:



? Taille de
le matériel
est comment
les données
du point

+ une mémoire d'entretien appelé "frame buffer" ou encore "bit map buffer" dans laquelle est stocké l'image digitalisée représentée par une matrice de valeurs binaires. Chaque valeur correspond à 1 point de l'écran ou pixels. Suivant la luminosité, ou la couleur du point, cette valeur aura pris une valeur particulière définissant cette caractéristique.

- + un moniteur ou écran de visualisation souvent de très haute résolution
- + un processeur graphique qui est, soit un simple contrôleur d'écran qui a comme tâche de rafraîchissement de l'écran, soit un processeur plus intelligent qui connaît un ensemble d'instructions graphiques et qui peut être programmé pour réaliser des tâches plus sophistiquées
- + des dispositifs d'acquisition de données:

ce sont des dispositifs par lesquels l'utilisateur dialogue avec le système. Ces dispositifs de communication qui doivent favoriser l'acquisition rapide et conviale des données telles que commandes, symboles, coordonnées choisies, sont très variés:

- tablettes graphiques recueillant des signaux codés représentant la position d'un stylet par rapport aux bords de la tablette
- souris, manche à balai, boules roulantes
- le réticule est un dispositif lié à l'écran, constitué de deux filaments qui se croisent et permettent de pointer sur un endroit précis de l'écran. C'est ce dispositif que nous utilisons en parallèle avec le clavier alphanumérique classique dans la réalisation du logiciel graphique interactif.

3. COMPOSANTS LOGICIELS DES SYSTEMES GRAPHIQUES.

Un système graphique peut être défini comme un ensemble de composants matériels et logiciels ayant comme fonction d'offrir aux utilisateurs la possibilité de concevoir facilement des dessins, des compositions graphiques mixés avec des textes tels que légendes, titres...

On peut hiérarchiser les composants logiciels en trois couches logiques

3.1. L'ensemble des instructions graphiques élémentaires: REGIS

Cette couche représente l'interface avec le processeur graphique .
Le terminal graphique VT240 dispose d'un ensemble d'instructions graphiques appelé REGIS (Graphics Instruction Set).

Nous résumons ici les principaux groupes d'instructions existant dans REGIS.

1. Les protocoles d'entrée et de sortie dans le mode graphique:

Pour entrer, ^{ds le mode graphique} le programme d'application doit envoyer au terminal les suites de caractères de contrôle suivantes

DCS 1 p entrer dans REGIS

ou

DCS 3 p entrer dans REGIS en conservant la dernière ligne comme zone de communication

Pour quitter le mode graphique, il faut générer la suite de caractère de contrôle

ST sortir du mode graphique et retourner au mode texte

L'équivalent en code 7-bits des caractères indiqués sont:

DCS : ESC

ST : ESC

2. Les instructions de tracé et de positionnement:

<u>Commande</u>	<u>Nom</u>	<u>Description</u>
P [X,Y]	POSITIONNEMENT	positionner de la nouvelle position X,Y , X et Y peuvent être absolue ou relative

<u>Commande</u>	<u>Nom</u>	<u>Description</u>
V [X,Y]	VECTEUR	dessiner la ligne, à partir de la position actuelle du curseur jusqu'à la nouvelle position spécifiée par X,Y
C [X,Y]	CERCLE	dessiner le cercle dont le centre est la position courante, X,Y est un point de la circonférence
C (A < degrés>)[X,Y]		dessiner un arc de cercle dont le degré est spécifié par degrés

3. Les instructions de texte:

<u>Commande</u>	<u>Nom</u>	<u>Description</u>
T [X,Y]	Text	positionnement des caractères <i>du texte</i>
T(U[<width,height>])		définition de la proportion du caractère unitaire
T(H < 1-25 >)		définir la grandeur du caractère
T 'text'		spécification du texte à afficher

4. Les instructions de contrôle:

W(M < n >) (P < o-q >) (...)	définition des différents modes d'affichage: inverse, négative, multiplication, intensité...
S(A[X ₁ ,Y ₁],[X ₂ ,Y ₂]) (10)	spécification des modes de contrôle de l'écran tel effacement de l'écran

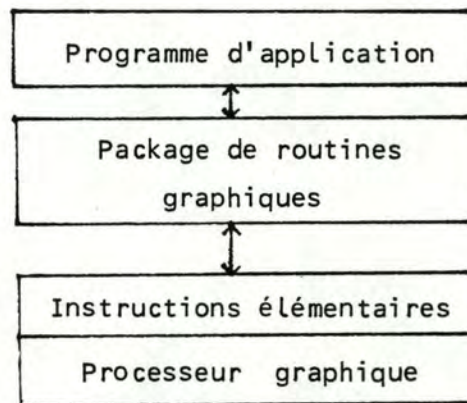
5. Les valeurs par défaut que REGIS donne automatiquement si elles ne sont pas spécifiées explicitement par commande

W(I3)	couleur de fond: noir
T(S1)	grandeur unitaire pour les caractères
T(I0)	pas de mode italique
S(A 0,0 , 799,499)	la totalité de l'écran est adressable

3.2. Le Package de Routines Graphiques: LYS

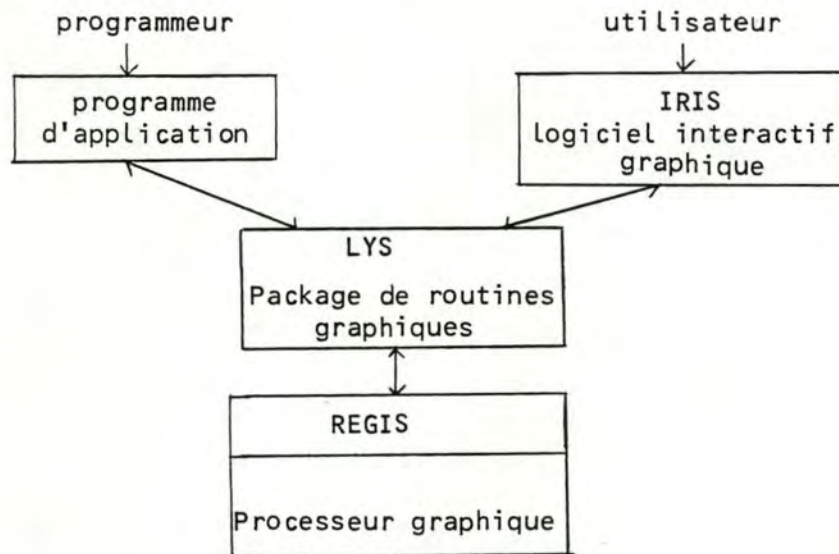
Le package est constitué d'un ensemble de primitives utilisant les instructions graphiques élémentaires et des méthodes de transformations géométriques. Les primitives sont destinées à être appelées par des programmes d'application. Cette couche réalise l'interface entre la première couche avec le programme d'application

de



3.3. Le logiciel interactif graphique: IRIS

C'est un logiciel graphique qui utilise les couches logiques inférieures pour réaliser une machine graphique virtuelle offrant directement ses services à l'utilisateur



CONCLUSION du chapitre 1.

Dans le présent travail, nous essayons

- Premièrement, de construire une librairie de primitives graphiques appelée LYS, sur base des instructions graphiques élémentaires de REGIS (Graphics Instructions Set du VT240). Ces routines peuvent être appelées à partir d'un programme d'application écrit en Fortran
- Deuxièmement, de concevoir puis d'implémenter un logiciel graphique à 2 dimensions que nous donnons comme nom: IRIS.

en fait

II

LIBRAIRIE DE PRIMITIVES GRAPHIQUES : LYS

La librairie de primitives graphiques peut être définie comme un ensemble de routines conçu pour faciliter le travail du programmeur d'application graphique. Le package représente une couche intermédiaire entre le programme d'application et les instructions élémentaires du processeur graphique. Il doit faire abstraction des détails matériels et techniques, tout en offrant aux programmeurs la possibilité d'exploiter pleinement la capacité graphique de la machine.

DIVERSITE DES PACKAGES GRAPHIQUES.

L'étude des différents logiciels distribués sur le marché, tel que GSP, GRAF, NEWMAN, DI3000, nous montre qu'il existe une très grande variété dans le choix des primitives proposées, tant du point de vue syntaxique que sémantique. Les échanges des programmes d'application deviennent rapidement irréalisables.

En vue de réaliser une librairie de primitives graphiques, modeste mais cependant relativement complète, nous avons comparé ces différents systèmes pour y dégager des points communs et constantes.

LIMITATION DE NOTRE PACKAGE GRAPHIQUE.

Sur base de ces points communs et constantes des contraintes matérielles et de temps, nous proposons un ensemble de primitives suffisamment complet, comprenant toutes les fonctions essentielles aptes à satisfaire la plupart des besoins des programmeurs d'application graphique.

Le package graphique LYS ne comporte donc pas des routines de segmentation. Ce type de primitive, souvent appelé, de structuration, donne la possibilité de définir une séquence de primitives graphiques appelé segment. L'invocation de ce segment par son nom générera séquentiellement les primitives contenues dans le corps du segment.

1. CLASSIFICATION DES PRIMITIVES DU PACKAGE

Nous classifions les primitives de notre package en 4 groupes différents.

- + Les primitives d'affichage qui se subdivise en 2 classes,
 - les primitives de tracé
 - les primitives de manipulation de texte

Ces primitives permettent de générer des tracés sur l'écran (cercle, ligne courbe, droite, point) et de les mixer avec des textes

- + Les primitives de contrôle: ce type de primitive permet de contrôler le fonctionnement du système
- + Les primitives de mise en page définissent les espaces de visualisation et d'affichage
- + Les primitives de transformation géométrique permettent la manipulation des figures: rotation, translation, mise en échelle, zooming

Primitives d'affichage	Primitives de contrôle	Primitives de Mise en page	Primitives de Transformation géométrique
---------------------------	------------------------------	--	---

Les principaux groupes de primitives de LYS.

2. LES PRIMITIVES D'AFFICHAGE.

Les primitives de tracé permettent d'obtenir des tracés sur l'écran, par la définition de la suite des points ou caractères formant ce dessin.

On trouve dans cette catégorie de primitive, des primitives générant des lignes droites, lignes courbes, des arcs de cercle. Ces primitives permettent également d'obtenir des graphiques différents, d'intensité et de luminosité différentes.

La caractéristique principale des primitives de tracé est l'utilisation d'un point courant représentant la position actuelle sur l'écran.

La programmation de ces primitives est souvent un habillage du set des instructions élémentaires du processeur graphique.

2.1. Primitives de tracé

<u>Primitives</u>	<u>Description</u>
posit(X,Y)	positionne le curseur aux coordonnées X,Y
vect(X,Y)	affiche la ligne droite définie par la position actuelle et les coordonnées X,Y
dot(X,Y)	affiche le point dont les coordonnées sont X,Y sur l'écran
cercle(Xc,Yc,X,Y, arc,sens trigo)	affiche l'arc de cercle dont le rayon est situé à (Xc,Yc) et dont un des points est situé à (X,Y). Le degré de l'arc est (arc) et le sens du tracé est le sens trigonométrique (sens trigo=vrai) ou sens horlogique (sens trigo=faux)
axe(Xor,Yor,gradX, gradY)	trace les axes d'abscisse et d'ordonnée ayant comme origine (Xor,Yor), l'axe d'abscisse est gradué en (gradX) et l'axe d'ordonnée est gradué en (gradY)

arc du cercle

grille(Xor,Yor,gradX, gradY)	trace la grille ayant comme origine (Xor,Yor), la largeur des mailles est (gradX), la hauteur des mailles est (gradY)
set pat(npatt)	détermine le mode de tracé, en choisissant parmi les 10 modes proposés par (npatt)

2.2. Primitives de texte

Ces primitives nous offrent la possibilité de mélanger les graphiques avec du texte. La manipulation des chaînes de caractères par des primitives permet de les afficher, de les effacer et de définir ou de modifier les caractéristiques des caractères (grandeur, inclinaison, italique)

<u>Primitives</u>	<u>Description</u>
text(string)	affiche la chaîne de caractères (string) à partir de la position actuelle
spacar(dx,dy)	détermine l'espace entre 2 caractères
hsize(ht,size)	définit l'élancement (ht) et la largeur (size) du caractère
italic(deg)	inclinaison le caractère de (deg) degré
tilcar(angle,degré)	rotation de la chaîne de caractère par rapport à la ligne d'horizon de (angle) degré
	rotation du caractère par rapport à sa ligne de base de (deg) degré
textmode (mode)	fixe le mode utilisé dans l'affichage des textes. Le mode peut être normal, inverse, de remplacement ou d'effacement.

*que peut-on faire dans
chaque de ces modes ?*

3. LES PRIMITIVES DE CONTROLE.

Ces primitives permettent de contrôler le fonctionnement du processeur graphique. Ces primitives comportent les protocoles d'entrée et de sortie du mode graphique, le changement des différents modes disponibles du terminal.

Ces primitives ^{est} pour la plupart une simple traduction, sous une forme plus compacte, plus facile à l'utilisation, des différentes commandes et protocoles disponibles de REGIS (Graphics Instructions Set du VT240).

<u>Primitive</u>	<u>Description</u>
INGRAF	fait entrer dans REGIS graphics mode
EXGRAF	fait sortir de REGIS graphics mode, retour au mode standard du terminal
CLS	efface complètement l'écran
INK(valink)	détermine l'intensité du tracé: noir, gris, gris clair, blanc suivant la valeur de (valink)
PAPER(valink)	détermine l'intensité du fond de l'écran suivant la valeur attribuée à (valink)
SCREEN(amin, omin, amax, omax)	détermine la zone adressable de l'écran
TIME(ticks)	fixe le délai de fichage pendant (ticks) portion de temps
REPCURS	rapporte la position du curseur au programme d'application
REPCURSI	active le mode de positionnement par RETICULE et rapporte au programme d'application la position du RETICULE
INGRAFD	fait entrer dans REGIS graphics mode en conservant la dernière ligne de l'écran pour la communication

*affichage
pendant ticks
seulement de
4s*

d'affichage

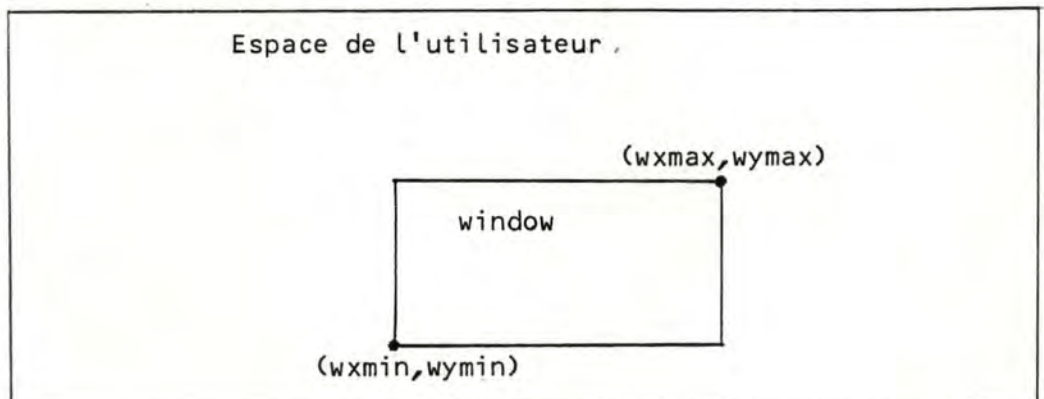
remarque

4. LES PRIMITIVES DE MISE EN PAGE.

4.1. Espace de l'utilisateur et la fenêtre (window)

L'espace de l'utilisateur est un espace à deux dimensions (pour un système graphique à deux dimensions). Théoriquement, il s'étend à l'infini, mais pratiquement il sera limité par l'ordre de grandeur du plus gros nombre réel accepté par l'ordinateur.

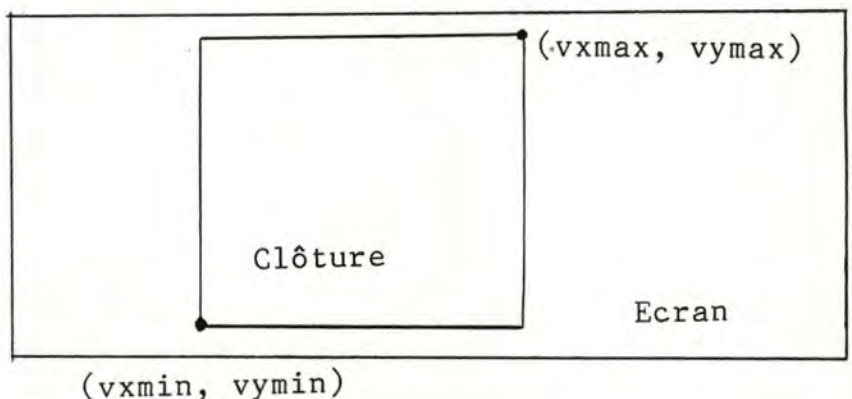
Il est évident que personne ne travaille jamais avec un espace aussi grand. On ne considère généralement qu'une portion rectangulaire de l'espace appelé FENETRE (Window).



La fenêtre spécifie la zone donnée qui doit être vue par l'ordinateur.

4.2. Espace d'affichage et la clôture (viewport)

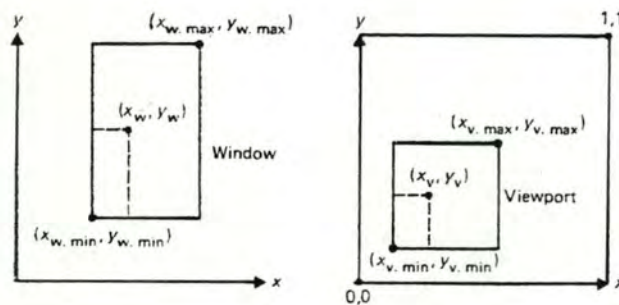
Quand un graphique est dessiné sur un écran, on ne désire pas toujours utiliser l'écran entièrement. Il est donc nécessaire de pouvoir spécifier au processeur graphique la zone rectangulaire dans laquelle on veut voir reproduire le contenu de la fenêtre. Cette zone rectangulaire est appelée CLOTURE (VIEWPORT).



4.3. Passage des coordonnées utilisateurs aux coordonnées écran: correspondance entre Fenêtre et Clôture.

Pour pouvoir projeter convenablement la fenêtre choisie (x_{wmin} , y_{wmin} , x_{wmax} , y_{wmax}) sur la clôture désirée (x_{vmin} , y_{vmin} , x_{vmax} , y_{vmax}) nous devons faire la correspondance entre un point situé dans la fenêtre et un point situé dans la clôture.

Soit les coordonnées de ce point dans la fenêtre et ses coordonnées dans la clôture.



Fenêtre et clôture

Nous avons:

$$\frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}} = \frac{x_v - x_{vmin}}{x_{vmax} - x_{vmin}} \quad (1)$$

et

$$\frac{y_w - y_{wmin}}{y_{wmax} - y_{wmin}} = \frac{y_v - y_{vmin}}{y_{vmax} - y_{vmin}} \quad (2)$$

(1) et (2) peuvent être réduites aux équations

$$x_v = x_{vmin} + \frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}} (x_w - x_{wmin})$$

$$y_v = y_{vmin} + \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}} (y_w - y_{wmin})$$

qui sont de la forme

$$xv = Sx (xw - xmin) + xmin$$

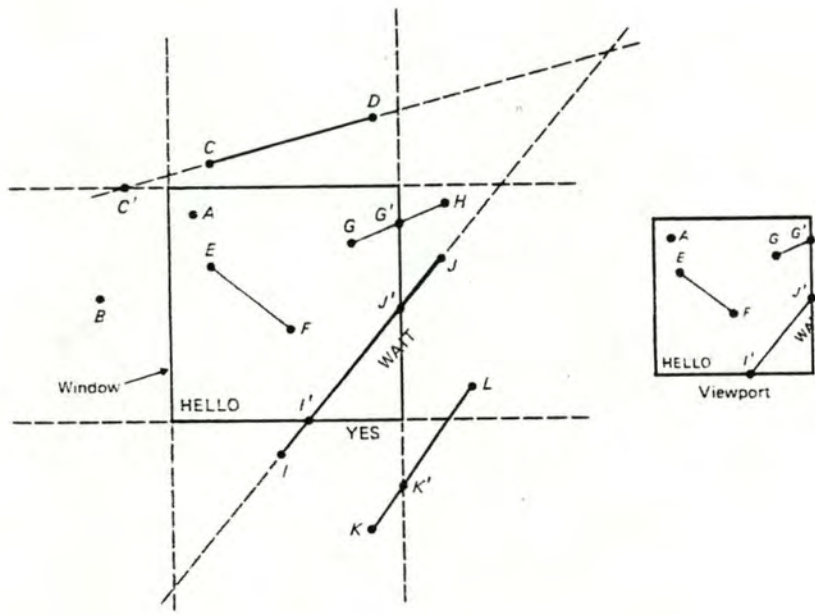
$$yv = Sy (yw - ymin) + ymin$$

où Sx, Sy sont des facteurs de mise en échelle et $xmin, ymin,$ des facteurs de translation.

4.4. Le découpage de la figure dans l'espace de visualisation

On ne doit ~~que~~ ^{que} représenter sur l'écran les points à l'intérieur de la fenêtre. Les points qui sont extérieurs de cette surface sont éliminés.

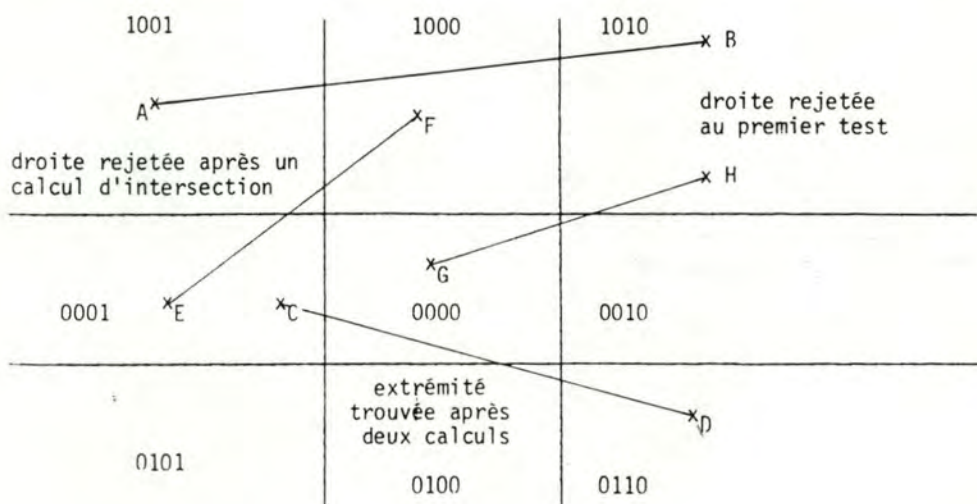
Le travail d'élimination est appelé découpage (CLIPPING)



Découpage de la figure

L'algorithme de découpage de Cohen-Sutherland:

Nous avons choisi cet algorithme pour sa simplicité de prise en oeuvre mais aussi pour son efficacité. La méthode est basée sur la codification de l'espace utilisateur en 9 régions par un code binaire de 4 bits: chaque bit représente la position de la région considérée par rapport à la FENETRE. Ainsi l'intérieur de la fenêtre est codé en 0000.



algorithme de découpage

On distingue 3 sortes de droites:

- celles dont les deux extrémités sont en dehors de la fenêtre
- celles dont aucune des extrémités ^{ni l'une ni l'autre} sont en dehors de la fenêtre
- celles dont l'une des extrémités est en dehors de la fenêtre

Le principe d'élimination est alors simple:

- on fait l'intersection des deux codes binaires associés aux extrémités
- si le résultat de l'intersection est non nul, le segment est hors de la fenêtre et non visible
- si le résultat est nul, on divise le segment en deux et on recommence à appliquer le processus jusqu'à l'élimination des parties cachées

ALGORITHME

Procédure Découpage (X1, Y1, X2, Y2)

Début

C1:= code (X1, Y1)

C2:= code (X2, Y2)

tant que ((C1∧C2)= 0000) et ((C1≠0000) ou (C2≠0000)) faire
début

si C1=0000 alors

échanger X1, X2

échanger Y1, Y2

échanger C1, C2

fin si

si (C1∧000≠0001) alors

Y1:= Y1 + (Y2-Y1) * (xwmin - X1) / (X2-X1)

X1:= Xwmin

sinon si (C1∧0010=0010) alors

Y1:= Y1 + (Y2-Y1) * (Xwmax-X1) / (X2-X1)

X1:= Xwmax

sinon si (C1∧0100=0100) alors

X1:= X1 + (X2-X1) * (Ywmin-Y1) / (Y2-Y1)

Y1:= Ywmin


```

sinon si (C1^1000=1000) alors
    X1:= X1+(X2-X1)*(Ywmax-Y1)/(Y2-Y)
    Y1:= Ywmax
fin si
si (C1^C2)= 0000 alors
    afficher (X1,Y1,X2,Y2)

```

Fin.

4.5. Les Primitives de mise en page de LYS

<u>Primitive</u>	<u>Description</u>
WINDOW(xmin,xmax ymin,ymax)	définit une fenêtre limitée par (xmin,ymin) et (xmax,ymax)
VIEWPORT(amin,amax omin,omax)	définit un viewport limité par (amin,omin) et (amax,omax)
WITTOVI (x,y,abs,ord)	fait la correspondance entre les coordonnées de l'espace utilisateur et les coordonnées de l'écran
OUTCODE (x,y,code)	donne le code binaire d'une coordonnée de l'espace utilisateur.
CHECKCODE (code 1, code 2, ok)	vérifie si le segment de droite est inclus dans la fenêtre.
CLIPPER (X1,Y1,X2,Y2)	découpe le segment de droite dont les extrémités sont (X1,Y1) et (X2,Y2) à travers la fenêtre et affiche la portion visible à l'écran

5. PRIMITIVES DE TRANSFORMATION GEOMETRIQUE.

Dans cette catégorie de primitives, nous groupons toutes les primitives qui effectuent les translations, le changement d'échelle, les rotations des figures dans un espace à deux dimensions

5.1. Bases mathématiques du graphisme à deux dimensions

5.1.1. Les translations (translations)

Un point $P(x,y)$ qui fait une translation de Dx , parallèlement à un axe X , et de Dy , parallèlement à un axe Y vers un nouveau point P' (x',y') peut être représenté par:

$$\begin{aligned}x' &= x + Dx \\y' &= y + Dy\end{aligned}\quad (1)$$

sous forme vectorielle, nous avons

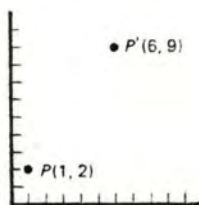
$$\begin{aligned}P &= [x \ y] \\P' &= [x' \ y'] \\T &= [Dx \ Dy]\end{aligned}$$

on peut réécrire (1)

$$[x' \ y'] = [x \ y] + [Dx \ Dy] \quad (2)$$

ou plus simplement

$$P' = P + T \quad (3)$$



Translation of a point.

5.1.2. La mise en échelle (scaling)

Un point, mis en échelle, par le facteur S_x dans la direction de l'axe X , et par le facteur S_y dans la direction de l'axe Y , donne un nouveau point obtenu par multiplication

$$x' = x \cdot S_x$$

$$y' = y \cdot S_y$$

Définissons S comme

$$\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

nous pouvons écrire dans la forme matricielle

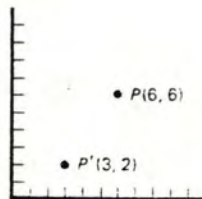
$$[x' \ y'] = [x \ y] * \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

$$\begin{aligned} x' &= x \cdot S_x + y \cdot 0 \\ y' &= x \cdot 0 + y \cdot S_y \end{aligned}$$

ou simplement

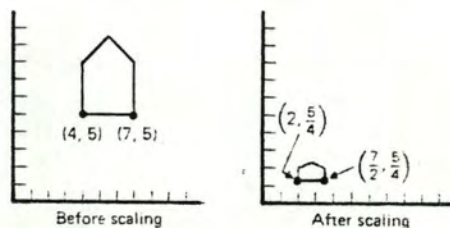
$$P' = P * S$$

Pour faire un agrandissement ou un rapetissement uniforme dans les deux axes, il faut que $S_x = S_y$. Dans le cas où $S_y = S_x$, la mise en échelle est dite différentielle.



$$\begin{aligned} 3 &= 6 * \frac{1}{2} \\ 2 &= 6 * \frac{1}{3} \end{aligned}$$

Fig. 7.3 Scaling of a point.



Scaling of an object.

5.1.3. La rotation

Soit un point $P(x,y)$ subissant une rotation d'un angle θ par rapport à l'origine, et donnant le nouveau point $P'(x',y')$. La distance de P et P' jusqu'à l'origine est r et ne change pas pendant la rotation.

Avant la rotation, nous avons

$$\begin{aligned}x &= r \cos \theta \\y &= r \sin \theta\end{aligned}\quad (1)$$

après la transformation

$$\begin{aligned}x' &= r \cos(\theta + \theta) = r \cos \theta \cos \theta - r \sin \theta \sin \theta \\y' &= r \sin(\theta + \theta) = r \cos \theta \sin \theta + r \sin \theta \cos \theta\end{aligned}\quad (2)$$

substituons (1) dans (2)

$$\begin{aligned}x' &= x \cdot \cos \theta - y \cdot \sin \theta \\y' &= x \cdot \sin \theta + y \cdot \cos \theta\end{aligned}\quad (3)$$

appelons R la matrice

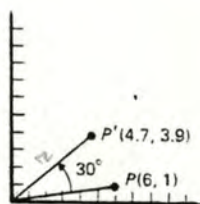
$$\begin{bmatrix} \cos \theta & \sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad d$$

sous forme matricielle (3) devient

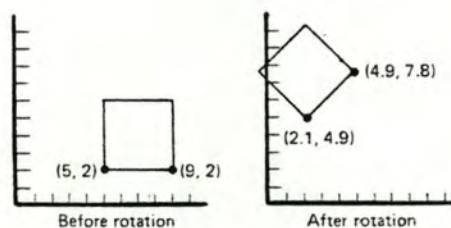
$$[x' \ y'] = [x \ y] * \begin{bmatrix} \cos \theta & \sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad d$$

ou simplement

$$P' = P * R$$



Rotation of a point.



Rotation of a square.

5.2. Représentation de la transformation:coordonnées homogènes et représentation matricielle.

La représentation matricielle de la translation, mise en échelle, et rotation est respectivement

$$P' = P + T$$

$$P' = P * S$$

$$P' = P * R$$

Cependant, la translation est traitée différemment (comme une addition) que la mise en échelle et la rotation (comme une multiplication). Nous voulons traiter toutes les trois transformations de base de façon homogène, pour permettre une combinaison facile des opérations.

Si nous ^{les} exprimons en coordonnées homogènes, toutes les trois transformations peuvent être traitées comme une multiplication.

En coordonnées homogènes, le point $P(x,y)$ est représenté par $P(w.x, w.y, w)$ pour tout facteur de multiplication $w \neq 0$. Nous attribuons à w la valeur 1 pour l'élimination de toute opération de multiplication et de division par w .

Sous forme matricielle et en coordonnées homogènes, l'équation de translation devient:

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx & Dy & 1 \end{bmatrix}$$

Exprimé plus simplement

$$P' = P * T(Dx, Dy)$$

où

$$T(Dx, Dy) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx & Dy & 1 \end{bmatrix}$$

Similairement, l'équation de mise en échelle est réexprimée sous la forme suivante

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} * \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Définissant

$$S(S_x, S_y) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

nous avons

$$P' = P * S(S_x, S_y)$$

Et finalement, l'équation de rotation peut être représentée comme

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

avec

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Nous avons

$$P' = P * R(\theta)$$

Notons que deux translations successives est additive

$$\begin{aligned} P' &= P * T(Dx_1, Dy_1) & P'' &= P * T(Dx_1 + Dx_2, Dy_1 + Dy_2) \\ P'' &= P' * t(Dx_2, Dy_2) \end{aligned}$$

tandis que deux mises en échelle successives est multiplicative

$$\begin{aligned} P' &= P * S(Sx_1, Sy_1) & P'' &= P' * S(Sx_1 * Sx_2, Sy_1 * Sy_2) \\ P'' &= P' * S(Sx_2, Sy_2) \end{aligned}$$

En conclusion, la matrice générale utilisée dans la transformation est de la forme suivante

$$\left[\begin{array}{cc|c} A & B & 1 \\ C & D & 1 \\ \hline M & N & E \end{array} \right]$$

- a) Les termes A,B,C,D produisent les effets de changement d'échelle, de rotation
- b) Les termes M, N produisent les translations
- c) E produit un changement d'échelle uniforme dans les deux sens.

5.3. Les primitives de transformation dans LYS

<u>Primitive</u>	<u>Description</u>
MULMAT(coord 1,mat, coord 2)	multiplie P(coord 1 1 ,coord 1 2) par la matrice de transformation MAT pour donner P(coord 2 1 ,coord 2 2)
SCALE(scax, scay, xi,yi,xt,yt)	mise en échelle de P (xi,yi) par la matrice S(scax,scay) pour donner P'(xt,yt)
ROTAT(rad, xi, yi, xt, yt)	faire la rotation de P (xi, yi) par la matrice R (rad) pour donner P (xt, yt)
TRANSLAT (dx, dy, xi, yi, xt, yt)	faire la translation de P (xi, yi) par la matrice T (dx, dy) pour donner P' (xt, yt)

L'emploi des primitives LYS.

Pour utiliser les primitives graphiques de LYS à partir d'un programme d'application écrit en Fortran, il faut que ce programme ^{est} 'lié' ^{Suit} postérieurement avec le module LYS.

Supposons que le programme d'application s'appelle APPLIC.FOR, qui a une version compilée, APPLIC.OBJ. Le linkage s'effectue avec la commande suivante sous VMS:

```
$ LINK APPLIC.OBJ, LYS.OBJ
```

ou simplement

```
$ LINK APPLIC, LYS
```

Le linkage réalisé, le programme peut être lancé par

```
$ RUN APPLIC
```

Nous présentons ci-dessous un simple programme utilisant des primitives graphiques de LYS, pour mieux illustrer l'utilisation de ces primitives

PROGRAMME DE FORTRAN DESSINANT LE VECTEUR P1 (0,0); P2 (100,200).

```
call ingraf
call cls
call window (0., 799., 0., 479.)
call viewport (0, 799, 0, 479)
call posit (0.,0.)
call vect (100.,200.)
call exgraf
end
```


CONCLUSION du chapitre 2.

Les deux groupes de primitives d'affichage et de contrôle sont relativement faciles à mettre en place. Ce n'est pas de même pour les groupes de primitives de mise en page et de transformation géométrique, qui demandent des outils mathématiques tels que calcul matriciel et géométrie descriptive. En plus, nous devons souvent choisir entre deux critères parfois contradictoires:

- la simplicité de la programmation
- l'efficacité, la rapidité du traitement

Nous avons souvent tranché, en privilégiant le premier critère. Certes les primitives de transformation géométrique pourront être accélérées en court-circuitant certains traitements mais la modularisation et l'homogénéité du module en souffrirait.

Cependant, tenant compte que les primitives pourraient être utilisées dans un logiciel interactif, la vitesse d'exécution est aussi considérée à son juste niveau.

III

LOGICIEL GRAPHIQUE INTERACTIF: IRIS

Le logiciel graphique interactif à réaliser que nous nommons IRIS, doit aider l'utilisateur à construire facilement une image composée d'une partie graphique et d'une partie constituée de texte.

La partie graphique est composée d'objets graphiques élémentaires pré-définis dans le système, que nous appelons 'SPRITES'. Ce sont un ensemble de symboles graphiques très simples, avec lesquels l'utilisateur peut manipuler pour composer la partie graphique. Composer la partie graphique revient à placer des SPRITES à des endroits différents de l'écran, à juxtaposer ceux-ci avec d'autres SPRITES, à déterminer leurs échelles, leurs orientations par des opérations de transformation disponibles.

La partie 'texte' est composée de chaînes de caractères. La composition de la partie texte est effectuée, d'abord, par la formation de la suite des caractères que nous appelons 'LINE', puis par la définition de la grandeur des caractères, de l'oblicité, du degré de l'inclinaison des chaînes par rapport à la ligne d'horizon.

IRIS doit aussi permettre à l'utilisateur de revenir sur ses décisions, c'est à dire de pouvoir effacer un sprite, une ligne quelconque, corriger par modification de la composition graphique et texte.

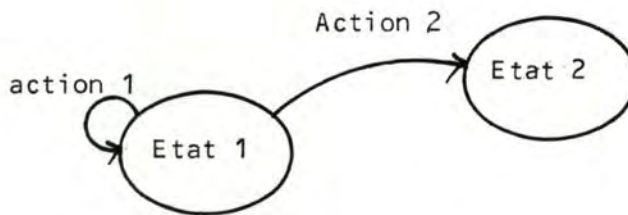
Après la composition, l'utilisateur peut sauver l'image conçue dans la mémoire de stockage, sous forme de fichier, imprimer l'image, si le site matériel possède une imprimante graphique. L'image, une fois stockée en mémoire auxiliaire, peut être rappelée, chargée dans le système pour être réutilisée.

1. SPECIFICATION DE LA MACHINE VIRTUELLE: IRIS.

1.1. Notion d'états et actions

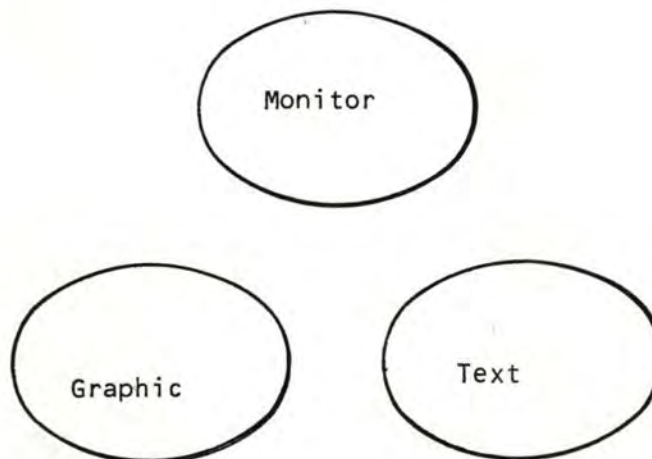
On peut considérer IRIS comme une machine virtuelle, caractérisée par des états et des actions, déclenchés par les commandes. Les états de la machine sont, soit stables, soit instables. Le système est dans un état instable s'il est en train de réaliser une action. Le système tend toujours à revenir à un état stable.

Les commandes demandent à réaliser une action quelconque et force le système de passer dans un autre état stable ou instable. A chaque état, il n'est permis de réaliser qu'un nombre déterminé d'action sur des objets spécifiques à l'état dans lequel on se trouve.



1.2. La machine IRIS

Une fois lancée, à partir d'un terminal VT240 ou VT241 sur VAX/VMS IRIS, après avoir été initialisée, peut se trouver dans trois états stables: Monitor, Graphic et Text.

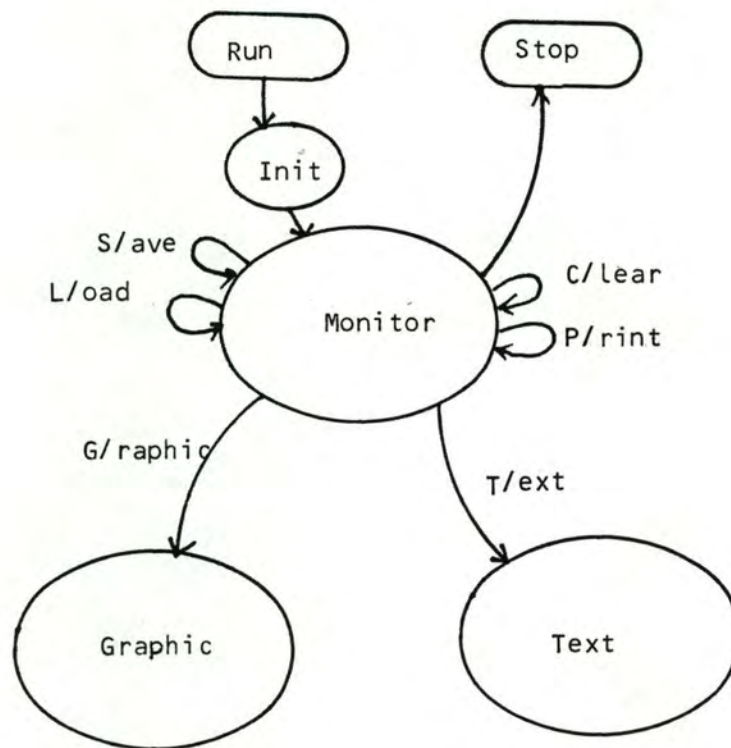


1.2.1. Etat Init

Dans l'état Init, le système graphique démarre en passant au mode graphique, initialise les différents paramètres du système avec des valeurs par défaut et appelle le gestionnaire d'écran pour se présenter. A la fin de l'initialisation, le système passe à l'état Monitor.

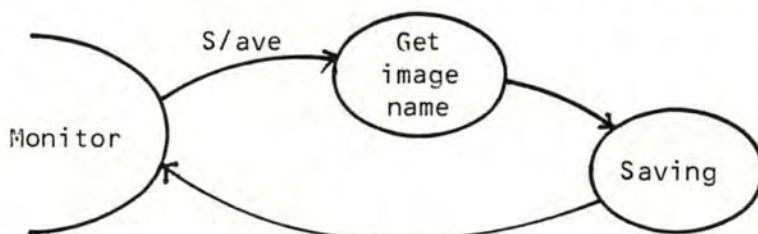
1.2.2. Etat Monitor

Dans l'état Monitor, le type objet manipulé est l'Image. L'utilisateur peut, dans cet état, afficher, charger, sauver, effacer une image; il peut aussi passer dans d'autres états: Graphic, Text et Stop.



Les commandes dans l'état Monitor:

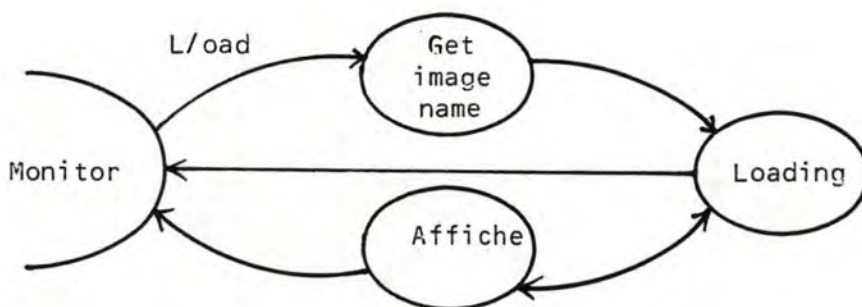
- a. S/ave: demande à IRIS de sauver l'image actuelle se trouvant dans un fichier contenu dans la mémoire du système.



Les états intermédiaires:

- get-image-name: demander et prendre le nom de l'image à sauver. Ce nom sera donné au fichier stockant la description de l'image.
- saving : sauver la description de l'image dans un fichier séquentiel.

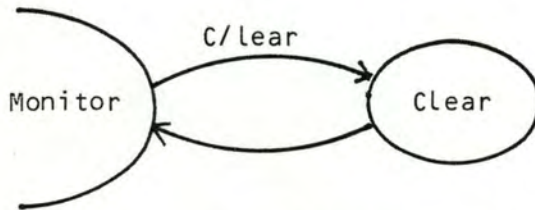
- b. L/oad: demande à IRIS de faire le chargement d'une image stockée en mémoire auxiliaire dans la mémoire du système et l'afficher à l'écran.



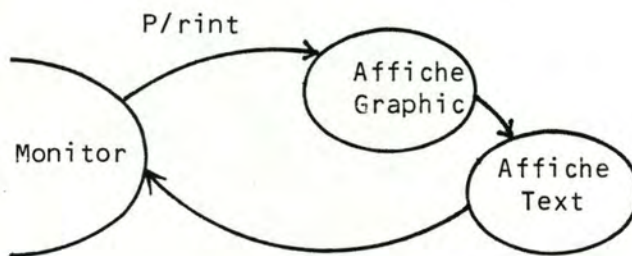
Les états intermédiaires:

- get-image-name:
- loading : charger l'image dont le nom est donné par get-image-name. Si le chargement réussit, passer à 'Affiche', sinon retourner à l'état Monitor en signalant l'échec.
- affiche : afficher l'image chargée à l'écran et retourner à l'état Monitor.

- c. C/lear: demande à IRIS d'effacer l'image actuelle sur l'écran puis ^{de} retourner à l'état Monitor.



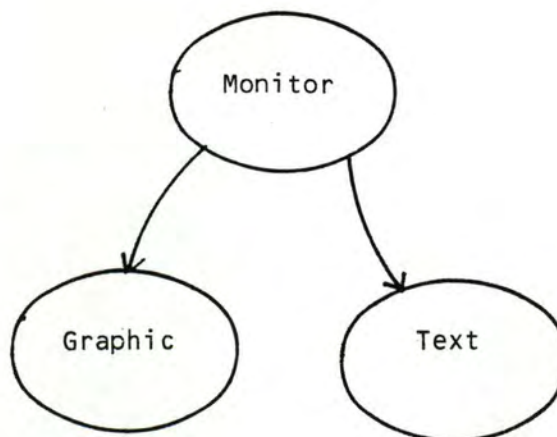
- d. P/rint: demande à IRIS d'afficher l'image actuelle à l'écran.



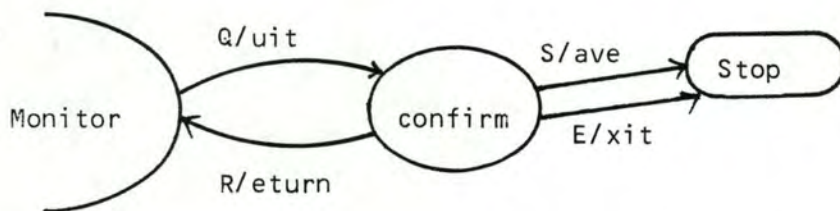
Les états intermédiaires:

- Aff-graphic: afficher la partie graphique de l'image
- Aff-text : afficher la partie texte de l'image

- e. G/raphic et T/ext: demande respectivement le passage vers les états Graphic et Text.



f. Q/uit: demande à IRIS d'arrêter la machine IRIS



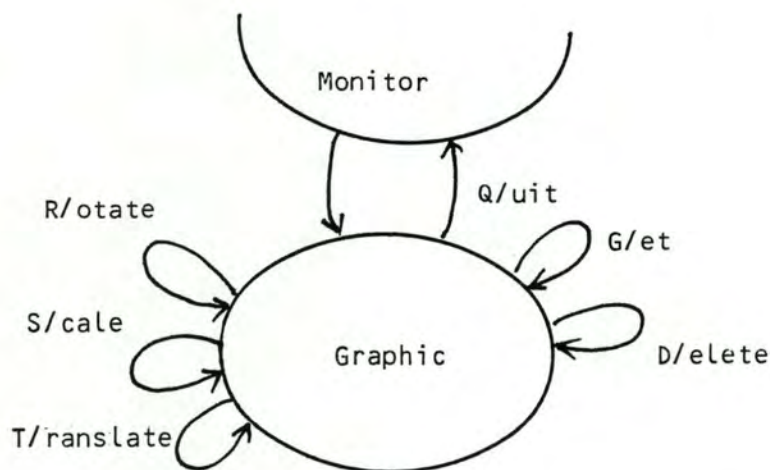
Les états intermédiaires:

- confirm: vérifier si l'image composée est sauvée. Si un sau-
-vetage est nécessaire, demande la confirmation de
l'utilisateur, soit pour sauver l'image et arrêter
soit arrêter simplement la système, soit retourner
à l'état Monitor.

1.2.3. Etat Graphic

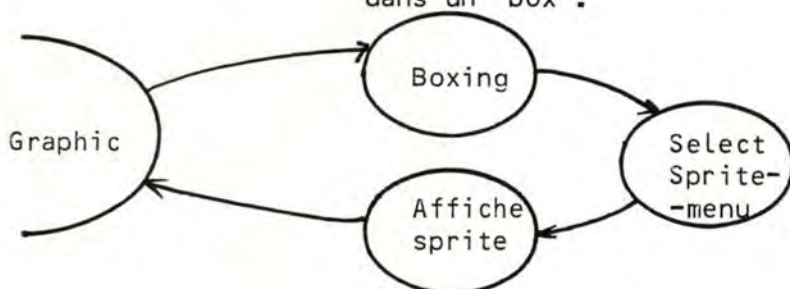
Dans l'état Graphic, le type d'objet manipulé est le 'sprite' .
L'utilisateur peut, dans cet état, appeler un sprite, le détrui-
-re, faire une translation, une rotation, une mise en échelle
sur un sprite, ou retourner dans l'état Monitor.

*↳ = carré, rectangle, cercle - -
figures élémentaires*



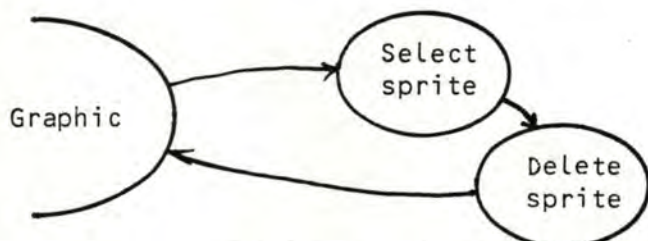
Les commandes dans l'état Graphic:

- a. G/et: demande à IRIS d'aider dans la composition d'un sprite choisi par le positionnement et par la mise en échelle dans un 'box'.



Les états intermédiaires:

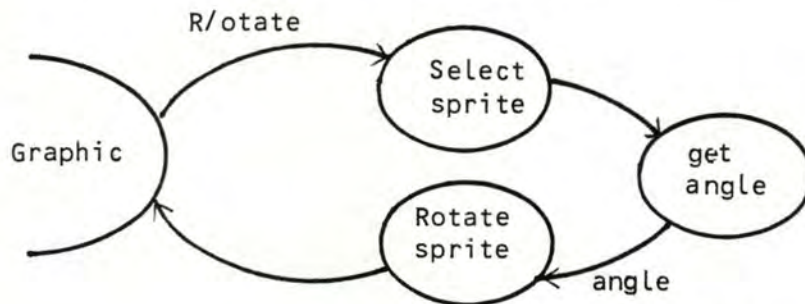
- Boxing : demande à l'utilisateur de déterminer la longueur, la largeur et la position du sprite.
 - Select-sprite-menu: noter le sprite choisi propose par la taille des sprites
 - Aff-sprite : affiche le sprite avec les caractéristiques demandées, ajout du sprite et ses caractéristiques dans la description de l'image
- b. D/elete: demande à IRIS d'éliminer un sprite de l'image actuelle. Toutes descriptions se rapportant à ce sprite sont détruites.



Les états intermédiaires

- Select-sprite: demande à l'utilisateur de désigner quel sprite dans l'image il faut effacer, et noter la décision.
- Delete-sprite: efface le sprite désigné, élimination de la spécification de ce sprite dans la description de l'image et retourner à l'état graphic

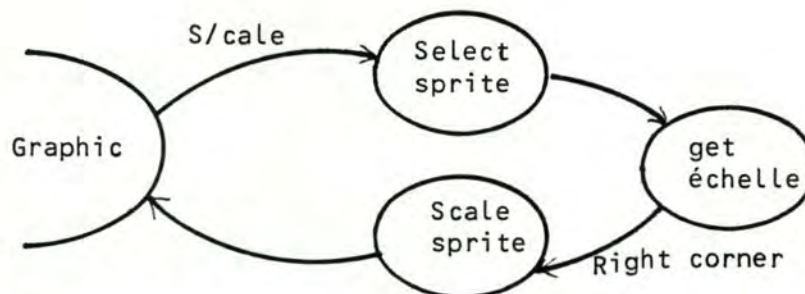
c. R/otate: demande à IRIS de faire la rotation d'un sprite de l'image actuelle



Les états intermédiaires:

- Select-sprite: cfr commande précédente
- Get-angle : demande à l'utilisateur de spécifier l'angle de rotation, d'enregistrer sa décision
- Rotate-sprite: effectue la rotation du sprite. Mise à jour de la description de l'image et retourner à l'état graphic

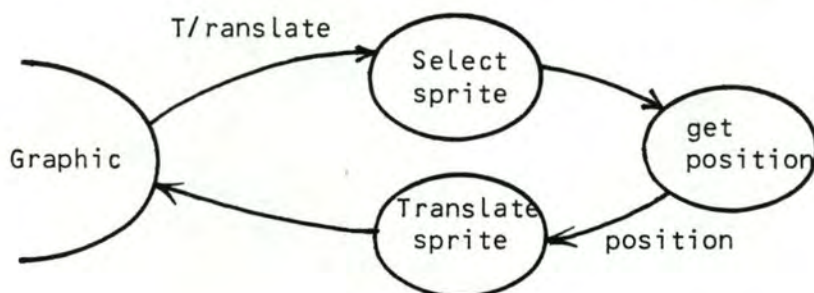
d. S/cale: demande à IRIS de faire la mise en échelle d'un sprite de l'image actuelle.



Les états intermédiaires:

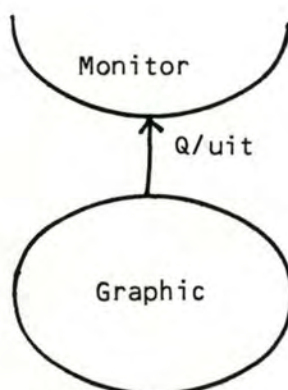
- Select-sprite: cfr commande précédente
- get-échelle : demande à l'utilisateur de spécifier les échelles, d'enregistrer ses décisions
- scale-sprite : mise en échelle du sprite, mise à jour de la description de l'image, retourner à l'état graphic

- e. T/ranslate: demande à IRIS de faire la translation d'un s-
-prite vers une position à préciser



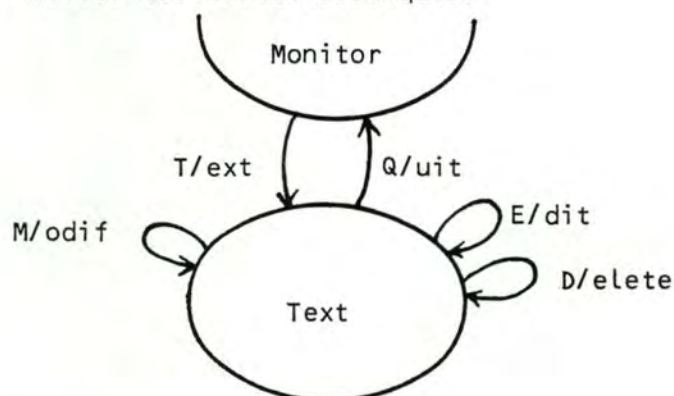
Les états intermédiaires:

- select-sprite: cfr commande précédente
 - get-position : demande à l'utilisateur de préciser la posi -
-tion vers laquelle il faut faire la transla -
-tion du sprite désigné , d'enregistrer cette
position
 - translate-sprite: effectue la translation du sprite désigné
vers la position indiquée; mise à jour de
la description de l'image, et retourne à
l'état-graphic.
- f. Q/uit: demande à IRIS de quitter l'état-graphic et de re -
-tourner à l'état-Monitor.



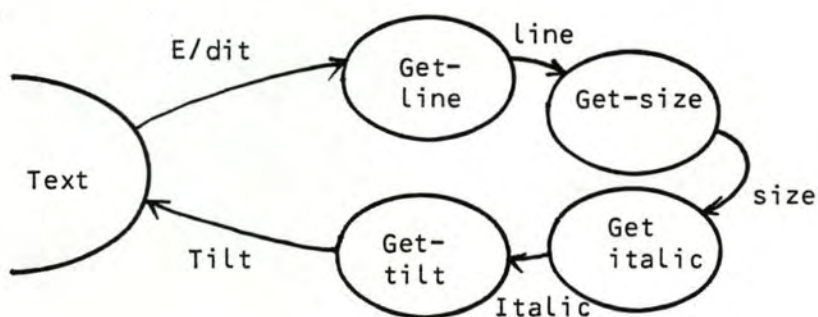
1.2.4. Etat Text

Dans l'état-Text, le type objet manipulé est la chaîne de caractères. L'utilisateur peut, dans cet état, composer la chaîne de caractères à partir du clavier, spécifier la largeur, l'élancement, l'italicité et l'oblicité des caractères. L'utilisateur peut aussi effacer une chaîne déjà insérée dans l'image ou modifier ces caractéristiques.



Les commandes dans l'état-Text:

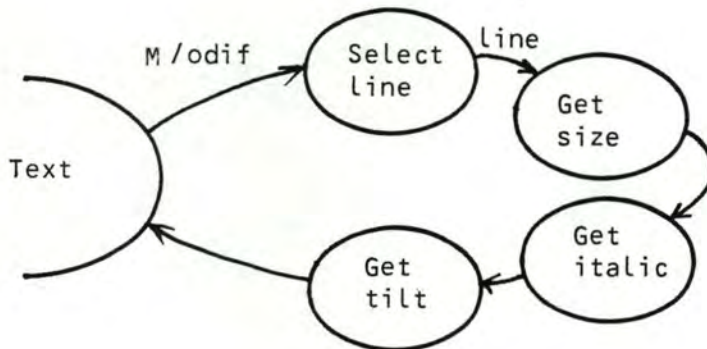
- a. E/dit: demande à IRIS d'aider dans la composition d'une chaîne de caractères et ensuite spécifier ses caractéristiques: élancement, grandeur...



Les états intermédiaires:

- Get-line : prendre la chaîne de caractères tapée par l'utilisateur
- Get-size : demande et prend la grandeur des caractères
- Get-italic : demande s'il faut mettre les chaînes de caractères en italique
- Get-tilt : demande et prend le degré de l'obliquité de la chaîne et retourner à l'état-Text

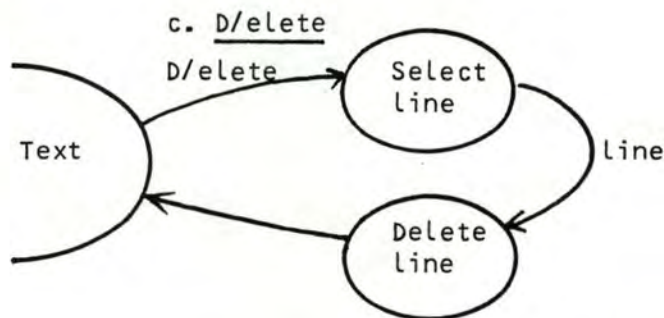
b. M/odif: demande la modification d'une chaîne de caractères de la partie text en précisant les changements des caractéristiques



Les états intermédiaires:

- Select-line : demande à l'utilisateur de désigner quelle chaîne de caractères il faut modifier et noter sa décision,
- Get-size : cfr commande précédente
- Get-italic : "
- Get-tilt : "

comment est-ce désigné?



Les états intermédiaires:

- Select-line: cfr commande précédente
- Delete-line: efface la chaîne désignée, élimine la spécification de cette chaîne dans la description de l'image et retourner à l'état-Text.

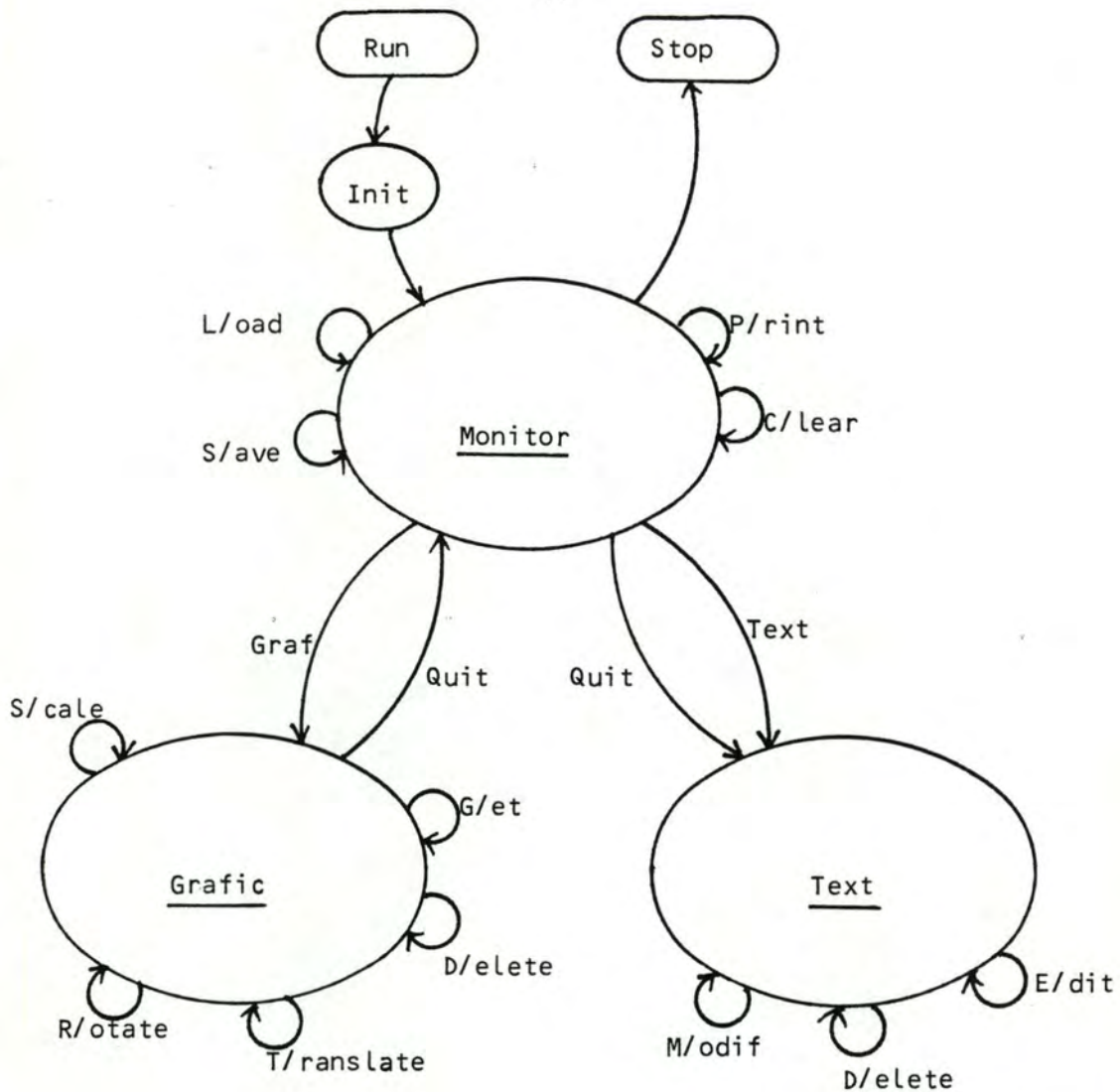


Diagramme d'états d'IRIS

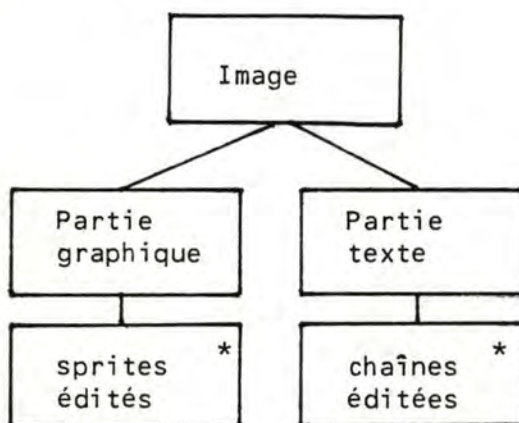
CONCLUSION

Nous pouvons représenter synthétiquement IRIS par le diagramme des états ci-dessus.

2. STRUCTURE DES DONNEES: LA REPRESENTATION DE L'IMAGE.

L'image conçue par l'utilisateur est composée d'une partie graphique et d'une partie texte. La partie graphique est formée de sprites placés et arrangés par l'utilisateur, que nous appellons sprites édités. La partie ^ktexte est formée de chaînes de caractères tapées, positionnées et formatées par l'utilisateur, que nous appellons chaînes de caractères édités.

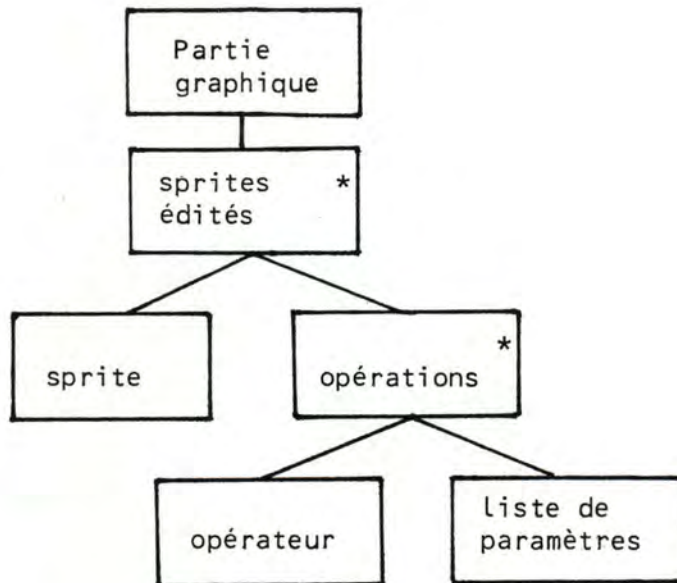
L'image est complètement définie si toutes les caractéristiques de la partie graphique et de la partie texte sont précisées.



2.1. La partie graphique

La partie graphique, comme nous avons déjà mentionné, est composée de sprites édités sur écran, c'est à dire transformés par des opérations géométriques.

Donc, pour construire la partie graphique, l'utilisateur doit disposer d'une bibliothèque de sprites et d'une liste d'opérations de transformations applicables sur ces entités graphiques. A chaque sprite sélectionné, l'utilisateur va en associer des opérateurs de transformation paramétrés, pour le positionner, lui faire tourner pour l'agrandir....



2.1.1. Représentations des sprites

- a. Définition: Les sprites sont les plus petites entités graphiques manipulables dans la système. Il existe 15 sprites prédéfinis dans le système et stockés dans la table de sprite. *Pourquoi on ne peut pas dire que ça ressemble à un sprite*
 La table des sprites doit contenir toutes les informations nécessaires à la construction d'un sprite au format standard: positionné à l'origine

sprite(1) descrip- -tion	sprite(2) descrip- -tion			sprite(15) descrip- -tion
--------------------------------	--------------------------------	--	--	---------------------------------

et dimensionné unitairement.

b. Structure interne des sprites

Pour construire un sprite sur écran, IRIS a besoin des informations sur l'identité du sprite à dessiner, les coordonnées de certains points importants (sommets, centre, point de départ) la longueur et largeur maximum du sprite (nécessaire pour le calcul d'échelle), le nombre de point à dessiner. Ces informations sont fournies par la 'sprite-description' définie comme suit

+ Sprite description: (polygones)

```

identifiant du sprite : numsprite
nombre de points      : numpoint
suite de coordonnées : X(1), Y(1)
                     : X(2), Y(2)
                     :
                     :
                     : X(nbpoint), Y(nbpoint)

longueur du sprite   : xmax
hauteur du sprite    : ymax

```

Remarque: on peut reconnaître les arcs de cercle qui peuvent être considérés comme un sprite dont le nombre de point est ∞ ou 0.

(cercle):

```

identifiant du sprite : numsprite
nombre de point      : 0
suite de coordonnées : XC, YC (coordonnées du centre)
                     : XR, YR (coordonnées d'un point
                               sur la circonférence)

longueur du sprite   : diamètre
hauteur du sprite    : diamètre

```

numsprite
nbpoint
X(1)
Y(1)
X(2)
Y(2)
X(nbpoint)
Y(nbpoint)
xmax
ymax

sprite description

2.1.2. Représentations des opérations

- a. Définition: les opérations de transformations déterminent le traitement que doivent subir les sprites manipulés. Une opération a deux composants:
- l'opérateur: rotation, translation, mise en échelle
 - les paramètres associés à chaque opérateur. A chaque sprite peuvent être associées plusieurs opérations

b. Les opérateurs

Les opérateurs sont codifiés de façon suivante

- translation = 00
- mise en échelle = 01
- rotation = 10

La combinaison des opérations est représentée par la somme des codes des opérateurs

Par exemple:

- translation + mise en échelle + rotation =
 $00 + 01 + 10 = 11$ en binaire
- translation + rotation =
 $00 + 10 = 10$

toujours

Cette méthode suppose que la translation est présente. Ce qui est généralement vrai dans la pratique. D'ailleurs un positionnement à l'origine peut toujours se traduire par une translation dont le déplacement par rapport à l'origine est égale à zéro ($Dx=0$, $Dy=0$)

c. Les paramètres

opération	paramètres
- translation	déplacement horizontal : Dx déplacement vertical : Dy
- rotation	angle de rotation : ang
- mise en échelle (scaling)	échelle horizontale : Sx échelle verticale : Sy

d. Structure interne de la combinaison des opérations

combinaison des opérations : codop
 angle de rotation : ang
 déplacement horizontal : dx
 déplacement vertical : dy
 échelle horizontale : sx
 échelle verticale : sy

codop	ang	dx	dy	sx	sy
-------	-----	----	----	----	----

2.1.3. Représentation de la partie graphique

Pour pouvoir reconstituer la partie graphique d'une image, nous devons savoir le nombre de sprites édités, leur identifiant et la combinaison des opérations à exécuter lors de l'affichage, ainsi que leurs paramètres

nombre total de sprites à éditer : nbsprite
 liste des identifiants des sprites : lsprites
 liste des combinaisons des opérations : lcodop
 liste des déplacements dx : ldx
 liste des déplacements dy : ldy
 liste des angles de rotation : lang
 liste d'échelle sx : lsx
 liste d'échelle sy : lsy

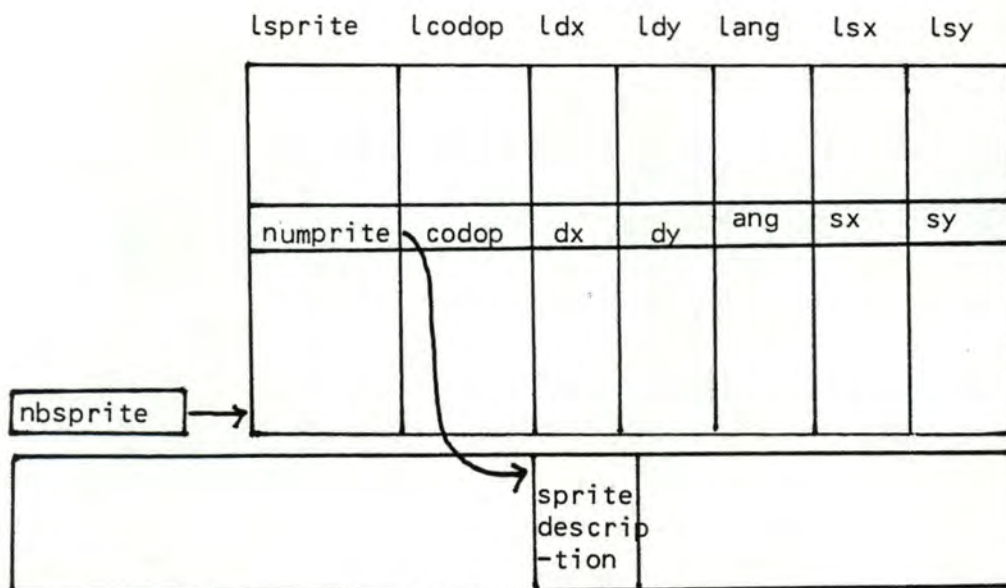
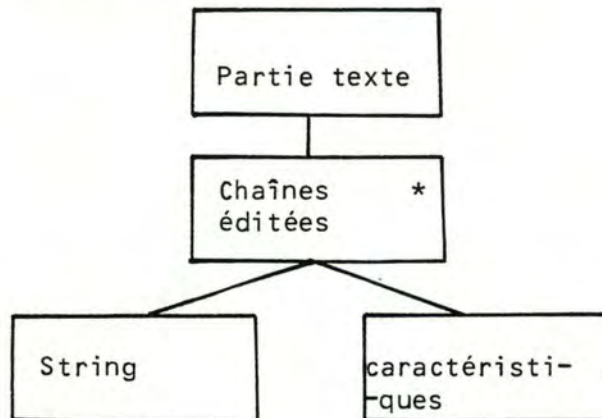


Table des sprites

2.2. La partie texte

La partie texte est formée de chaînes de caractères éditées sur l'écran avec les caractéristiques qui sont, soit spécifiées explicitement par l'utilisateur, soit spécifiées par défaut à des valeurs standard



2.2.1. Représentation d'une chaîne de caractères éditée

La chaîne de caractères à éditer tapée par l'utilisateur sera enregistrée à côté des caractéristiques qui lui sont attribuées.

```

chaîne à éditer      : line
grandeur de la chaîne : size
italique             : ital
inclinaison de la chaîne : tilt
position horizontale  : tx
position verticale    : ty
  
```

La chaîne 'schéma' est écrite avec la grandeur 2, est non italique, horizontale et à la position 10,20

line	size(*)	ital(*)	tilt(*)	tx(*)	ty(*)
'SCHEMA	2	0	0	10	20

2.2.2. Représentation de la partie texte

Pour reconstituer la partie texte d'une image, nous devons savoir la description de chaque chaîne de caractères éditée, ainsi que le nombre de chaînes de caractères à éditer.

```

nombre de chaîne à éditer      : nbline
liste des chaînes à éditer     : lline
liste des grandeurs            : size
liste des italiques            : ital
liste des inclinaisons        : tilt
liste des positions horizontales : tx
liste des positions verticales  : ty

```

	lline	size	ital	tilt	tx	ty
	'schema'	2	0	0	10	20
	'année 1985'	4	1	0	25	50
nbline						

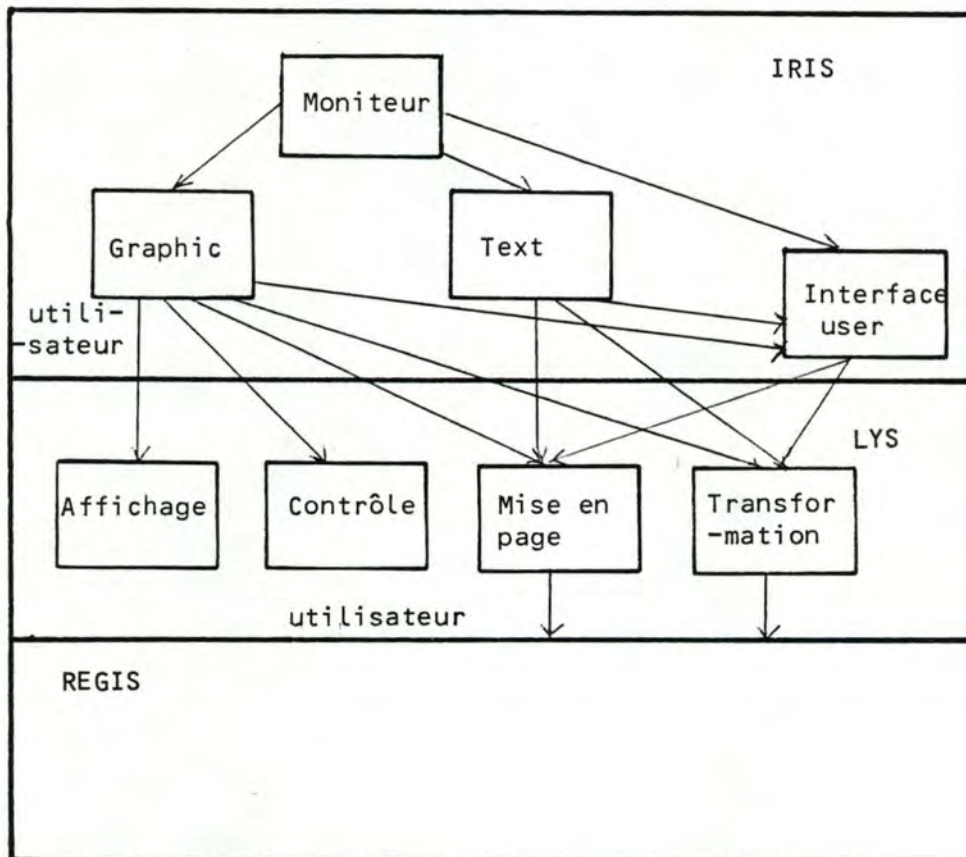
3. STRUCTURE DYNAMIQUE: ARCHITECTURE ET STRUCTURE DU LOGICIEL.

3.1. Architecture du logiciel

IRIS est construit en s'appuyant sur la librairie de routine graphique LYS qui est réalisée en utilisant les instructions de REGIS.

- + IRIS possède 4 modules: - le moniteur
 - le module graphique
 - le module texte
 - le module interface avec utilisateur: gestion d'écran, acquisition des données

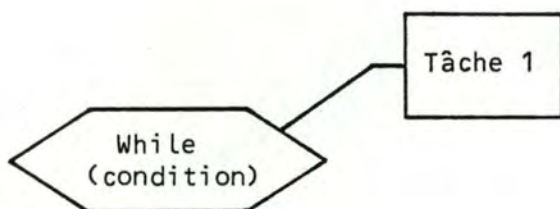
- + LYS se subdivise en 4 groupes de primitives:
 - affichage
 - contrôle
 - mise en page
 - transformation



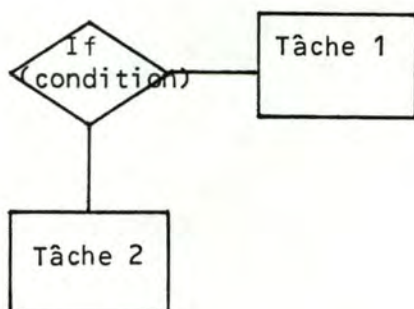
3.2. Structure du programme

Le logiciel est construit, suivant la démarche top-down. Nous suivons la même démarche pour présenter la structure du programme, à travers d'organigrammes structurés. Nous avons parfois simplifié l'algorithme présent dans la souci d'augmenter la lisibilité et d'aider le lecteur à saisir rapidement les tâches principales et leur enchaînement dynamique.

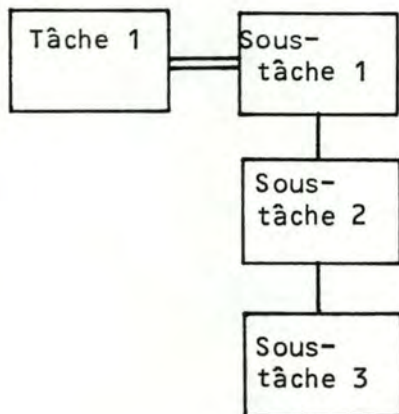
Notations employées dans la représentation:



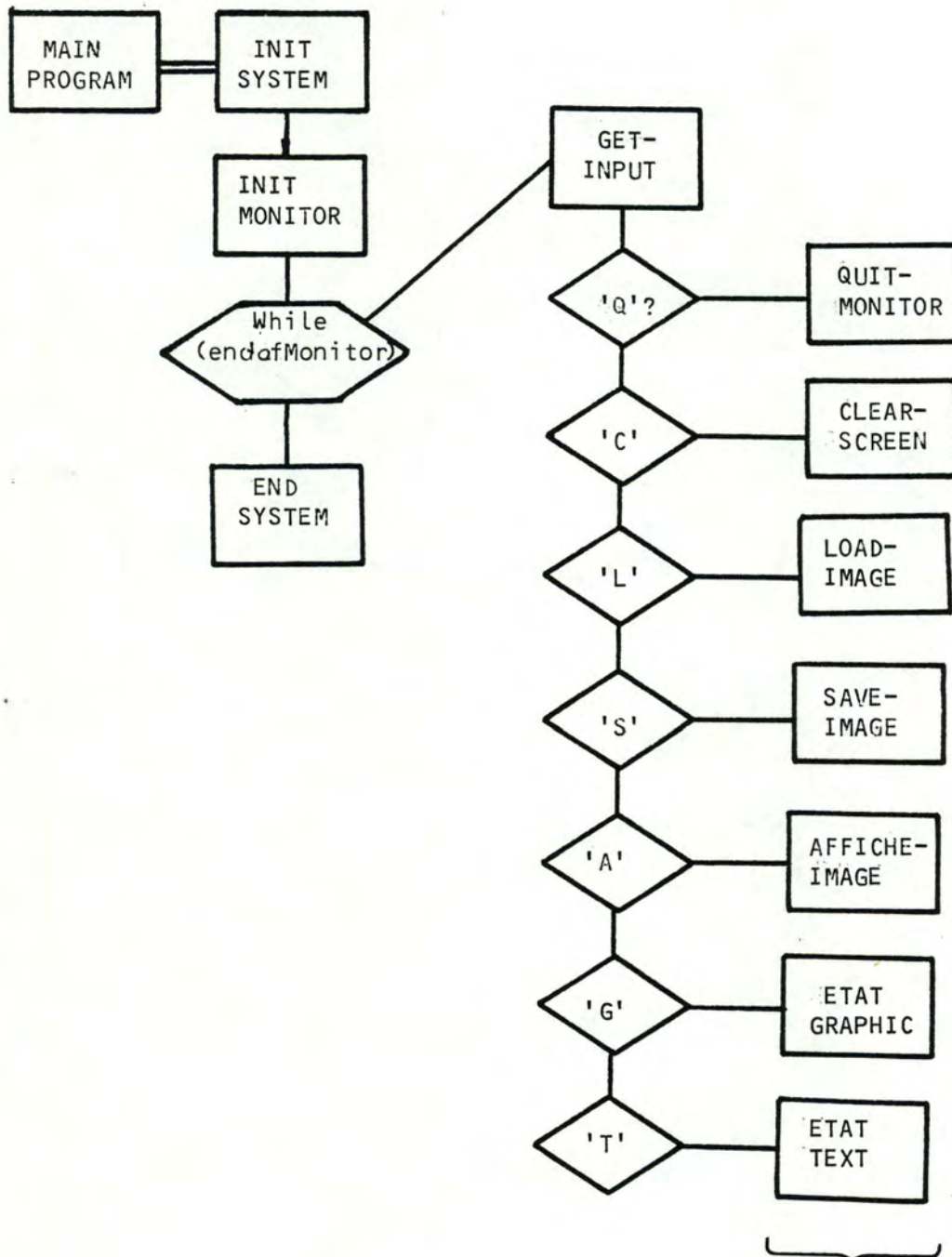
While (condition)
do tâche 1



If (condition) then tâche 1
else tâche 2



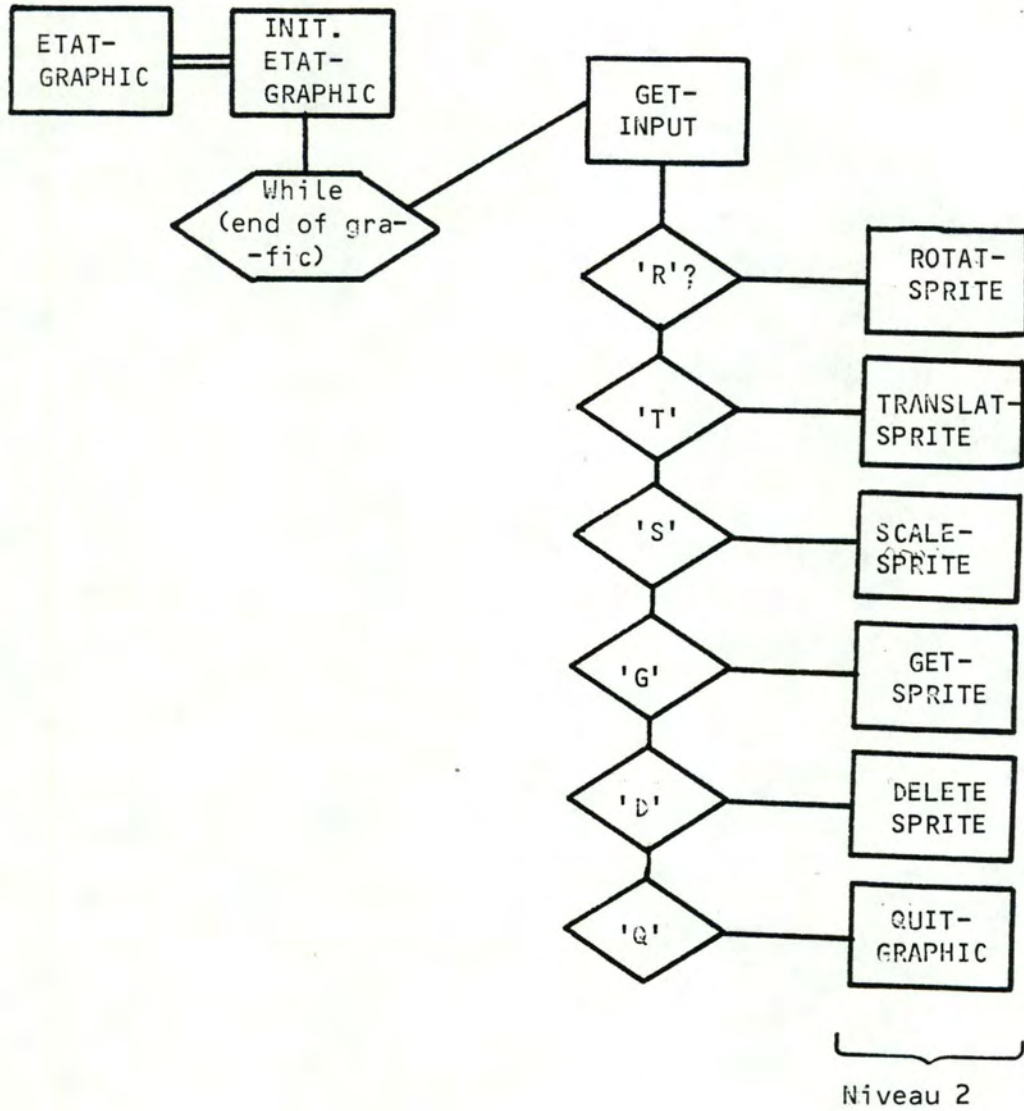
Tâche 1:
sous-tâche 1
sous-tâche 2
sous-tâche 3

Niveau 0: PROGRAMME PRINCIPAL : LE MONITEUR.

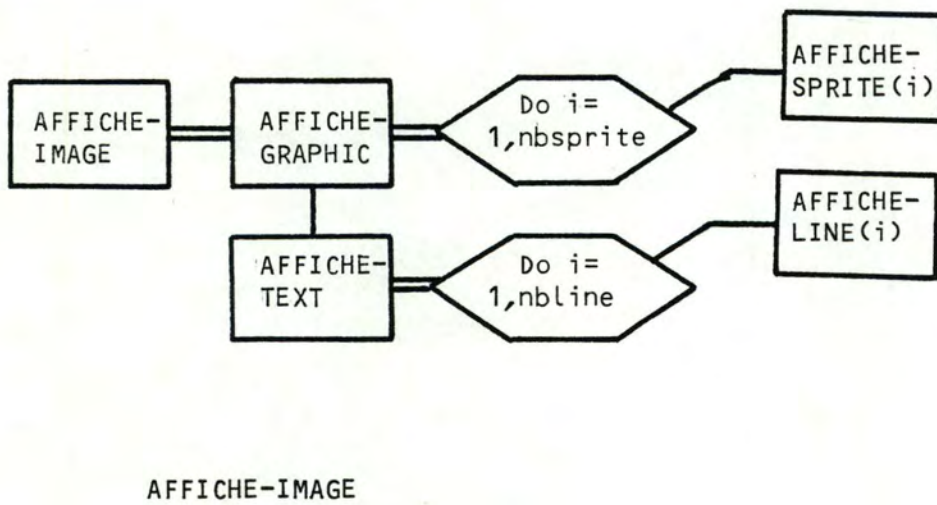
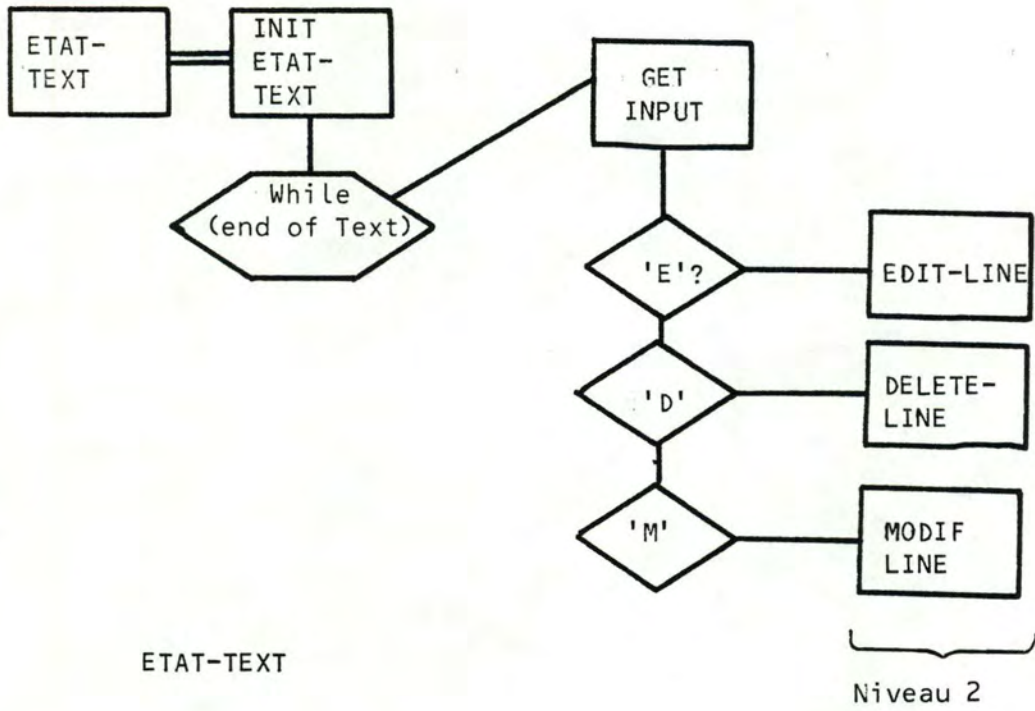
Niveau 1

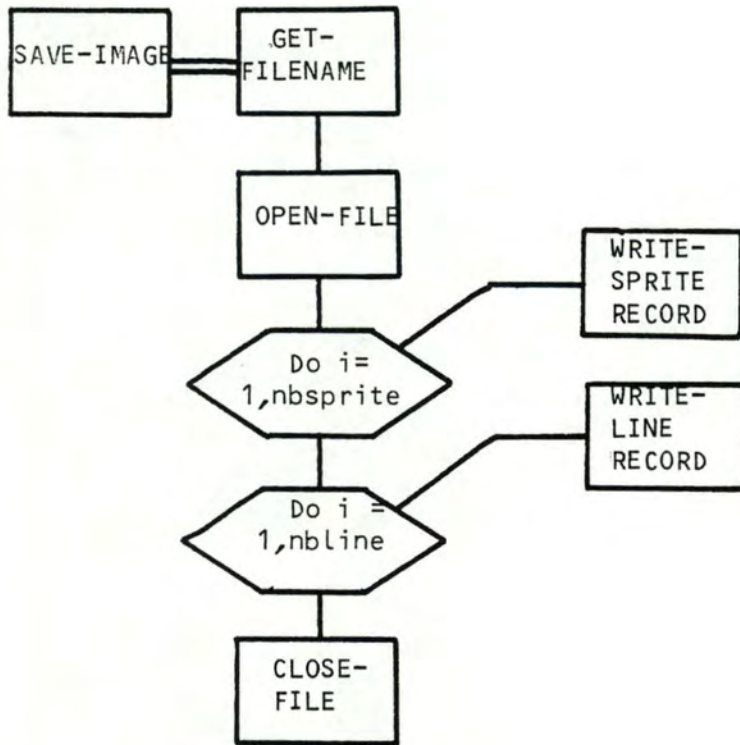
ETAT-MONITOR

Niveau 1:

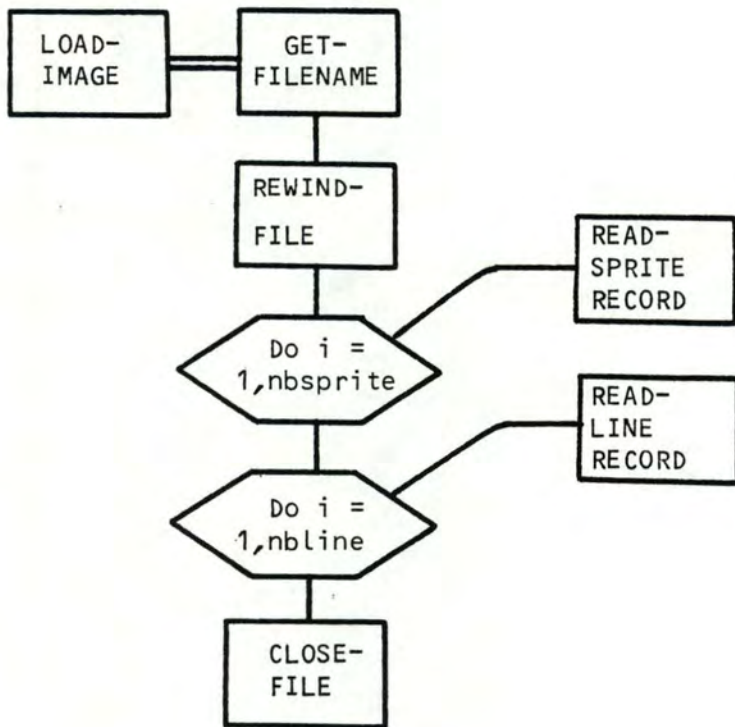


ETAT-GRAPHIC

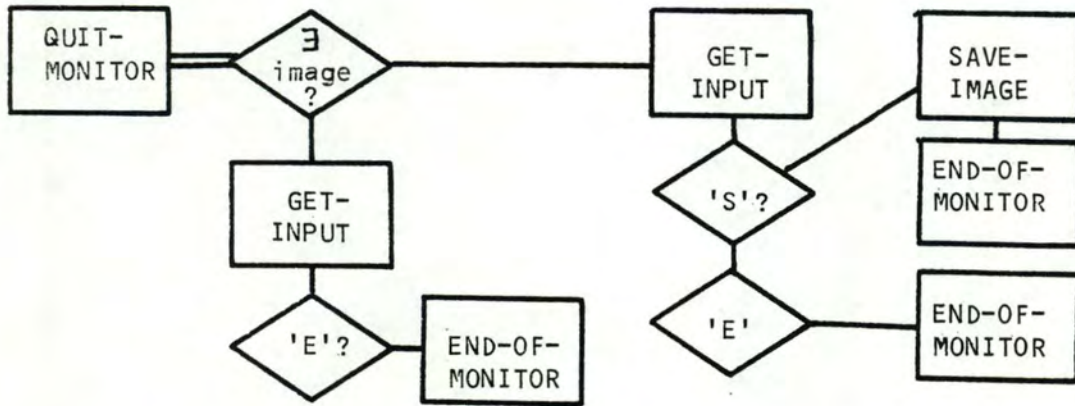




SAVE-IMAGE

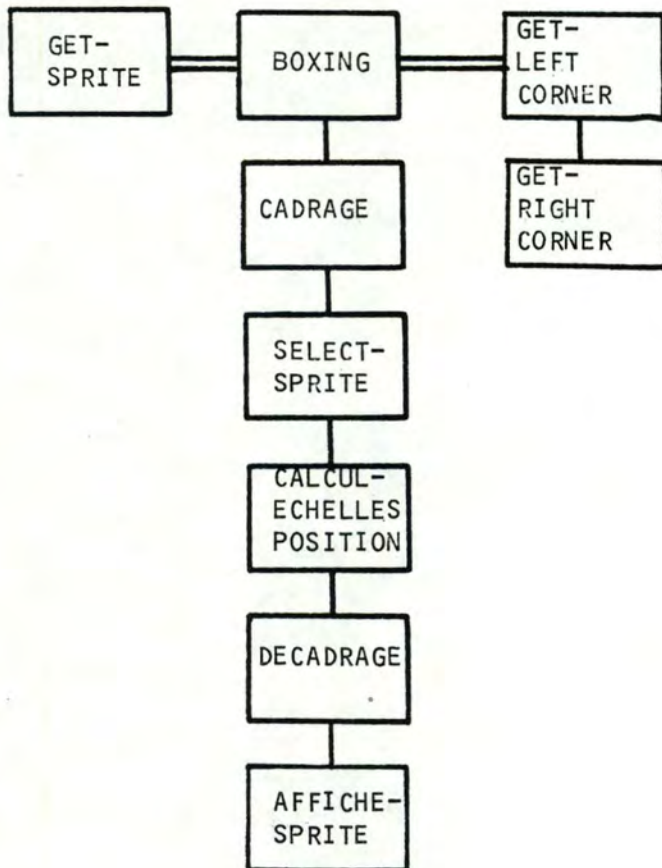


LOAD-IMAGE

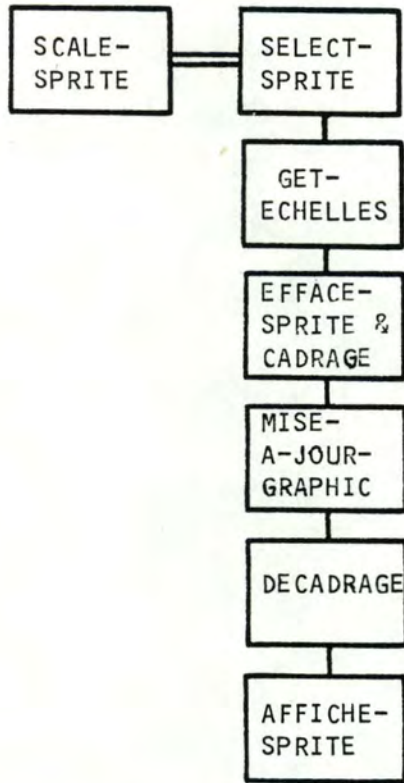


QUIT-MONITOR

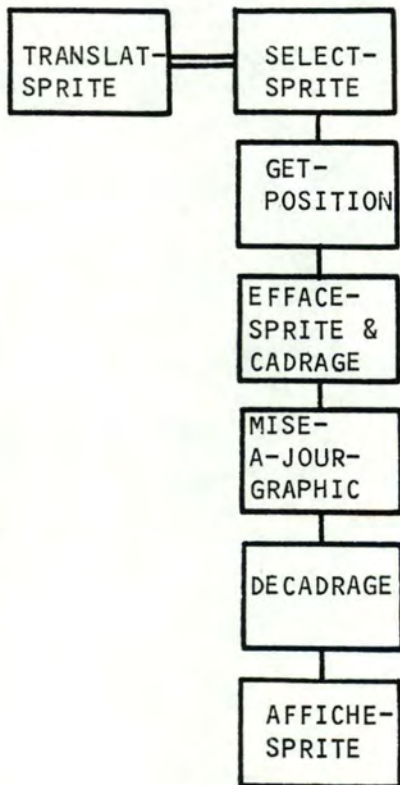
Niveau 2:



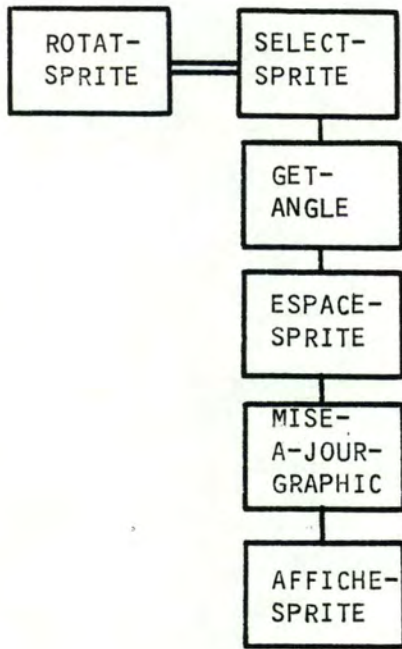
GET-SPRITE



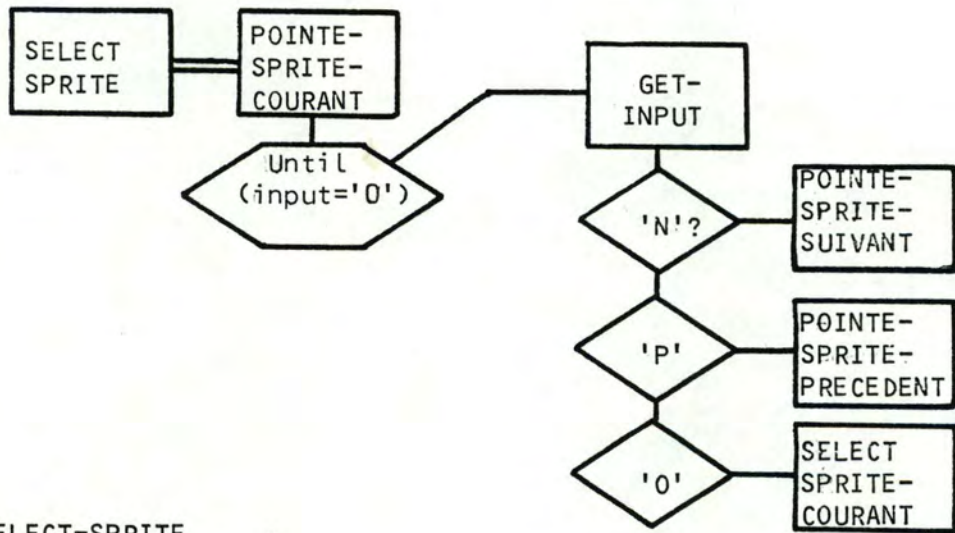
SCALE-SPRITE



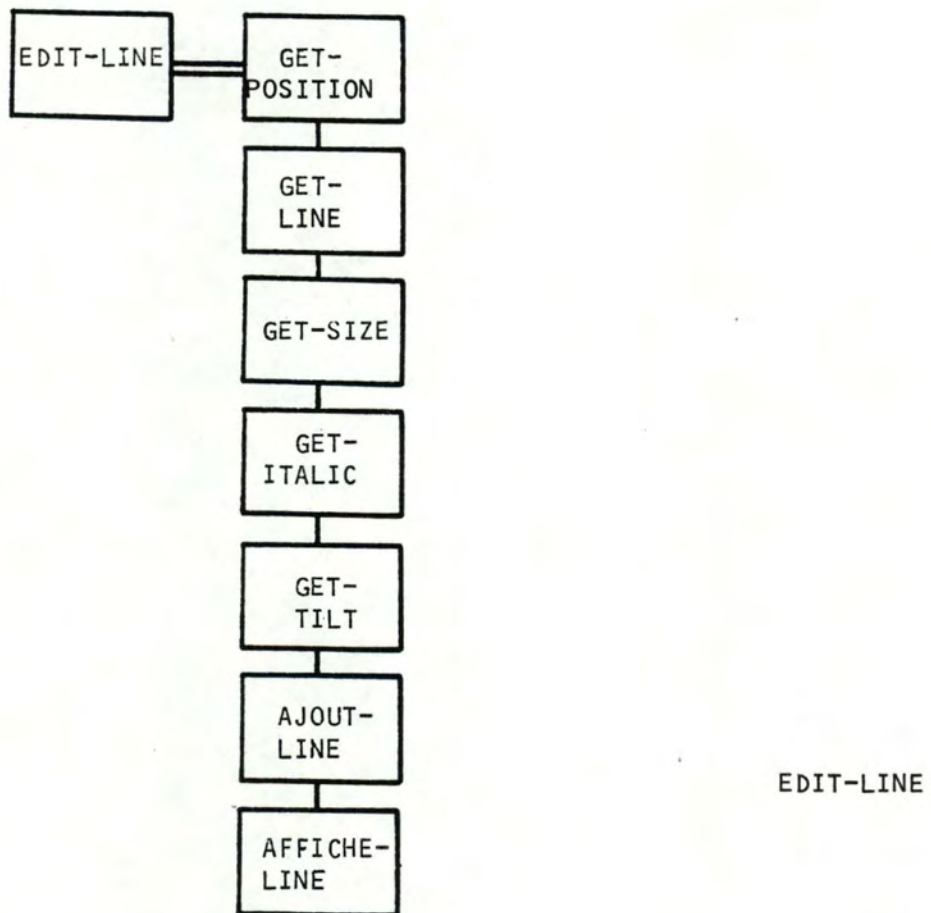
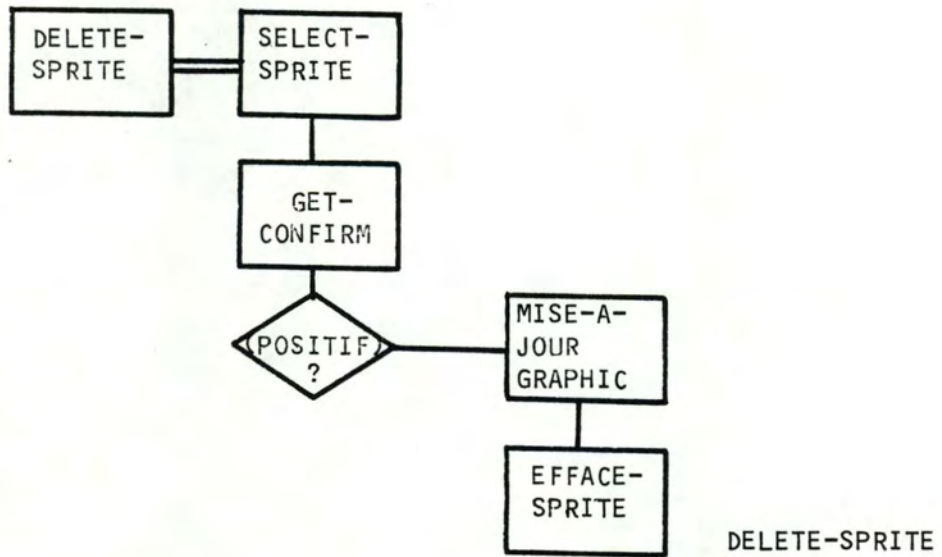
TRANSLAT-SPRITE

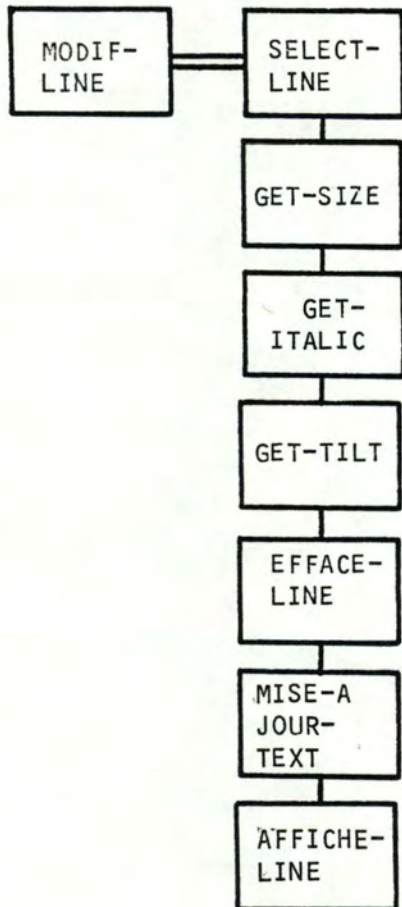
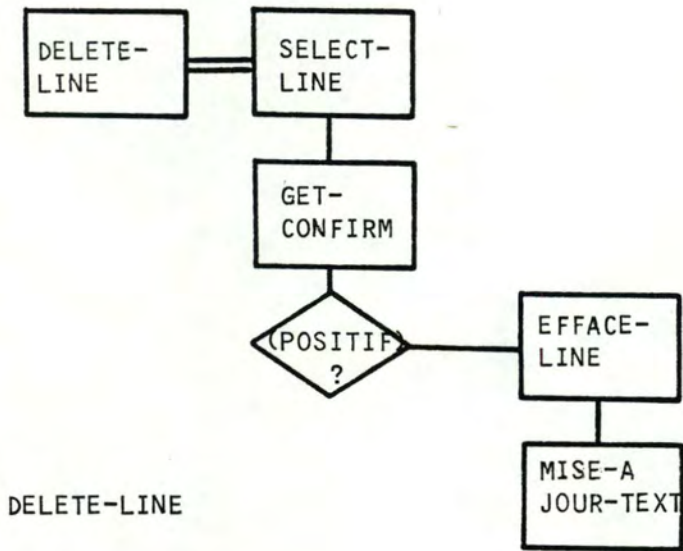


ROTAT-SPRITE

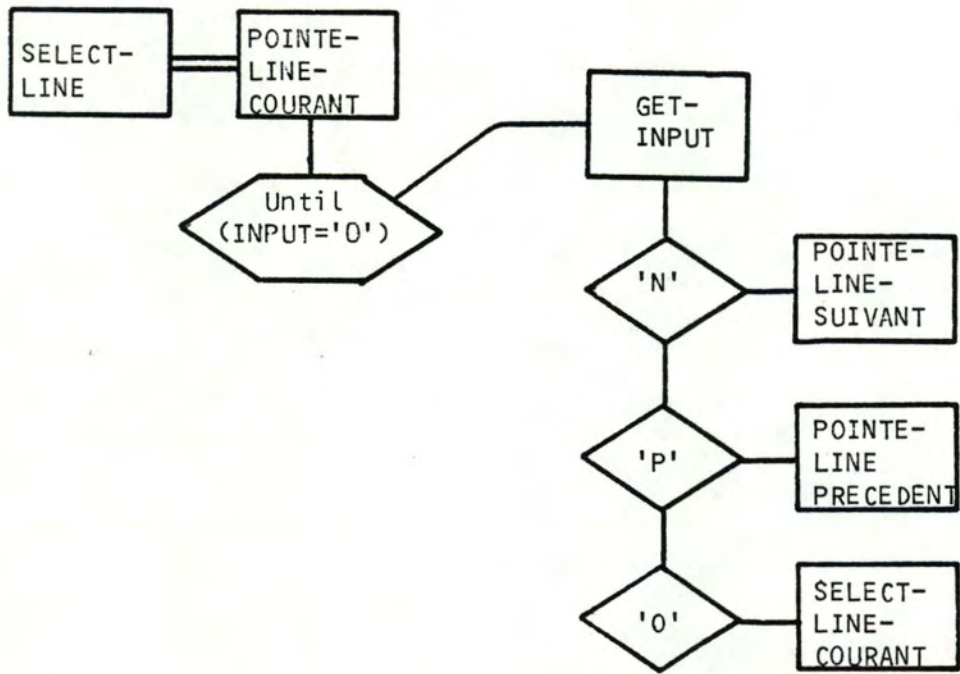


SELECT-SPRITE





MODIF-LINE



4. Manuel d'utilisateur

Ce manuel se propose de vous aider à apprendre à utiliser le programme IRIS de manière facile et rapide.

A. Lancement du programme IRIS

Les éléments suivants sont nécessaires pour l'utilisation du programme

- une copie du fichier IRIS . EXE
- un terminal graphique VT240 ou VT 241
- un système tournant sur VAX/VMS

Supposant que tous ces éléments sont à votre disposition, pour lancer le programme IRIS, tapez

```
$ RUN IRIS
```

Le programme commence à initialiser. En haut de l'écran, IRIS se présente:

```
'IRIS: INTERACTIVE GRAPHIC SYSTEM'
```

B. L'écran

Lorsque IRIS a terminé son initialisation, l'écran du moniteur est divisé en 4 zones:

- la zone de travail (ZT) représente la feuille de dessin où l'utilisateur peut dessiner, écrire dessus pour composer son image
- la zone des Méthodes (ZM) communique à l'utilisateur les commandes et les actions, appelées communément Méthodes, qu'il peut entreprendre. C'est par cette zone que le système communique avec l'utilisateur pour le guider

- La zone d'état (ZE) indique l'état dans lequel le programme se trouve. Cette zone rappelle à l'utilisateur dans quel sous système il est en train de travailler.
- La table des sprites (TS) présente le 'menu' des sprites prédéfinis dans IRIS. L'utilisateur peut choisir des éléments pour former son image parmi ces sprites

ZM	ZE
ZT	TS

Les sous-systèmes dans IRIS.

IRIS est composé de 3 sous-systèmes caractérisés par 3 états:

- Le Moniteur (Monitor): ce sous-système vous aide à manipuler les images, les afficher, les charger, les sauver les effacer. A partir du moniteur on peut aussi passer à d'autres sous-systèmes par exemple graphic ou text.
- Le Graphiste (Graphic): ce sous-système vous aide à manipuler les symboles graphiques, les placer, les transformer, les composer pour former la partie graphique de l'image

- L'Editeur de texte (Text): ce sous-système vous aide à manipuler les chaînes de caractères, les éditer, les effacer, les composer pour former la partie texte de l'image

C. Les commandes dans IRIS

Nous utilisons certains symboles et abréviations pour exprimer succinctement, plus précisément les fonctions des commandes. Ces notations évitent aussi des répétitions qui peuvent devenir lassant à la longue.

<u>Notations</u>	<u>Interprétations</u>
L/oad	commande Load
L/oad Moniteur	commande Load appartenant au sous-système Moniteur
\Rightarrow	a pour effet
ZM	Zone des Méthodes
ZT	Zone de Travail
ZE	Zone des Etats
MS	Menu des sprites
<a>	tapez touche 'a' suivi de 'carriage return'
<(val)>	suite de chiffres représentant une valeur numérique
<'file name'>	rentrez la chaîne de caractères associée au nom du fichier suivi de 'carriage return'
$\begin{matrix} \Delta \\ \triangleleft \quad \triangleright \\ \nabla \end{matrix}$	touches fléchées déplaçant le réticule dans le sens indiqué par la flèche
:::	affiche à l'écran
*	ce qui suit est un commentaire ou 1 exemple
(1)	alternative 1
(2)	alternative 2
< CR >	touche 'carriage return'

C.1. S/ave (Monitor)

Fonction: sauvegarde de l'image qui réside actuellement en mémoire pour être stockée dans un fichier en mémoire auxiliaire

<u>Utilisateur</u>	<u>IRIS</u>
< S >	⇒ ZM:: 'INPUT FILE NAME FOR SAVING'
< 'filename' >	⇒ sauvegarde de l'image dans un fichier stocké en mémoire auxiliaire, sous le nom de 'filename.DAT'
	ZM:: 'SAVED'
	fin de la sauvegarde

C.2. L/oad (Monitor)

Fonction: chargement de l'image stockée dans un fichier en mémoire auxiliaire dans la mémoire centrale

<u>Utilisateur</u>	<u>IRIS</u>
< L >	⇒ ZM:: 'INPUT FILE NAME FOR LOADING'
< 'filename' >	⇒ (1) chargement de l'image stockée dans le fichier 'filename.DAT' dans la mémoire centrale si le fichier 'filename.DAT' existe
	ZM:: 'LOADED'
	fin du chargement
	⇒ (2) ZM:: 'FILE NOT FOUND' si le fichier 'filename.DAT' n'existe pas

C.3. C/lear (Monitor)Fonction: effacement de l'écran

<u>Utilisateur</u>		<u>IRIS</u>
<C>	==>	L'image est complètement effacée ainsi que les zones ZM,ZE,MS. Puis réutilisation de l'écran avec re-construction des zones ZM, ZE, MS. *L'image est effacée, rendue invisible mais n'est pas détruite, est toujours résidente dans la mémoire centrale.

C.4. P/rint (Monitor)Fonction: affichage d'une image résidente en mémoire à l'écran

<u>Utilisateur</u>		<u>IRIS</u>
<P>	==>	(1) S'il existe une image résidente en mémoire, l'image est affichée (2) sinon ne fait rien

C.5. G/rafic (Monitor)Fonction: passage au sous-système Graphic où les commandes graphiques sont accessibles à l'utilisateur

<u>Utilisateur</u>		<u>IRIS</u>
<G>	==>	ZE:: 'GRAPHIC' passage au sous-système 'GRAPHIC' ZM:: 'G/et R/otate T/ranslate S/cale D/elete Q/uit' présentation des commandes activables à cet état

C.6. T/ext (Monitor)

Fonction: passage au sous-système Text où les commandes textes sont accessibles à l'utilisateur

<u>Utilisateur</u>		<u>IRIS</u>
<T>	==>	ZE:: 'TEXT' passage au sous-système TEXT
		ZM:: 'E/dit M/odif D/elete' présentation des commandes activables à cet état



C.7. Q/uit (Monitor)

Fonction: arrêt du programme

<u>Utilisateur</u>		<u>IRIS</u>
<Q>	==>	(1) S'il existe une image résidente en mémoire: ZM:: 'S/ave E/xit R/eturn' demande confirmation avant l'exécution
(1) <S>		S/ave (Monitor) puis arrêt du programme
(2) <R>		ZE:: 'MONITOR' retour à l'état Monitor ZM:: 'G/et R/otate T/ranslate S/cale D/elete Q/uit' présentation des commandes activables à cet état
(3) <E>	==>	ZM:: 'SAYONARA...' clôture du programme sans sauvegarde de l'image
		(2) S'il n'existe aucune image en mémoire ZM:: 'SAYONARA...' clôture du programme

C.8. G/et (Graphic)

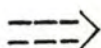
Fonction: placement de sprite sur l'écran à l'aide du réticule et du menu de sprite

<u>Utilisateur</u>		<u>IRIS</u>
<G>	==>	ZE:: 'GET' ZM:: 'SELECT LEFTCORNER' ZT:: réticule invitation à positionner à l'aide du réticule le point inférieur gauche du Box qui circonscrit le sprite par la suite
	==>	déplacement du réticule suivant la direction indiquée par la flèche pour placer LEFTCORNER
<CR>	==>	assignation de la position actuelle du réticule au point inférieur LEFTCORNER affichage d'un point à cet endroit ZM:: 'SELECT RIGHTCORNER'
	==>	déplacement du réticule suivant la direction indiquée par la flèche pour placer RIGHTCORNER
<CR>	==>	assignation de la position actuelle du réticule au point supérieur droit RIGHTCORNER affichage du Box, limité par LEFTCORNER et RIGHTCORNER qui contiendra le sprite ZM:: 'SELECT SPRITE' ZT:: réticule invitation à choisir le sprite dans MS à l'aide du réticule

UtilisateurIRIS

déplacement du réticule suivant la direction indiquée par la flèche, pour pointer le sprite choisi dans le MS

<CR>



Le sprite sélectionné est le sprite pointé actuellement par le réticule
affichage de ce sprite dans le box qui disparaît à l'instant

R/otate S/cale T/ranslate G/et D/elete Q/uit	Graf
	◊
	★
	◊
	◻
	↓
	↘
	◊
	◻
	↖
	△
	∪
	∪
	∪

Schema: l'écran pendant commande get

Exemple:

Supposons qu'on a déjà le sprite 1 et le sprite 2 à l'écran. Par 'get' on est en train de placer le troisième sprite à l'écran, qui est une étoile placée entre deux sprites. Le box est d'abord inséré entre les deux sprites, puis l'étoile est choisie par pointage du réticule pour être affichée en lieu et place du Box.

C.9. D/elete (Graphic)

Fonction: suppression d'un sprite affiché de l'écran

<u>Utilisateur</u>		<u>IRIS</u>
<D>	==>	ZE:: 'DELETE' ZM:: 'SELECT SPRITE: O/r N/ext P/revious' invitation à indiquer le sprite affi- ché à l'écran, à supprimer Le sprite courant est affiché <u>en hau- te luminance</u> pour distinguer <u>avec les autres sprites</u>

* Sélection du sprite

(1) <N>	==>	le sprite <u>suitant</u> (dans l'ordre de construction) devient le sprite cou- rant et est affiché en haute lumi- -nance
(2) <P>	==>	le sprite <u>précédent</u> devient le spri- -te courant et est affiché en haute luminance
(3) <O>	==>	sélection du sprite courant ZM:: 'DELETE SPRITE: O/k N/o' demande la confirmation de la sup- -pression

* Confirmation

(1) '<O>	==>	le sprite sélectionné est disparu de l'écran et est supprimé en mémoire
(2) '<N>	==>	annulation de la commande



Suppression de l'étoile

C.10. R/otate (Graphic)

Fonction: Rotation d'un sprite affiché à l'écran

<u>Utilisateur</u>		<u>IRIS</u>
<R>	==>	ZE:: 'ROTATE' ZM:: 'SELECT SPRITE: 0/k N/ext Previous' invitation à <u>indiquer le sprite af-</u> <u>-fiché</u> dont sa rotation est attendue Le sprite courant est affiché en haute luminosité

* Sélection du sprite

(1) <N >	passage au sprite <u>suisant</u> (cfr D/e -lete)
(2) <P >	passage au sprite précédent(cfr D/e -lete)
(3) <0 >	<u>Sélection du sprite courant.</u> Ce spr -te est sélectionné pour subir une rotation ZM:: 'INPUT ANGLE' invitation de préciser l'angle de ro -tation en degré

<(angle)> ==>

Le sprite courant disparaît

Affichage d'un nouveau sprite qui fait une rotation de (angle) degré par rapport à l'ancien sprite déjà disparu

Rotation du triangle
de 90°

C.11. T/ranslate (Graphic)

Fonction: translation d'un sprite affiché à l'écran

Utilisateur

<T> ==>

IRIS

ZE:: 'TRANSLATE'

ZM:: 'SELECT SPRITE: 0/k N/ext
Previous'

invitation à indiquer le sprite af-
fiché dont sa translation est at-
tendue

Le sprite courant est affiché en
haute luminosité

* Sélection du sprite

(1) <N>

passage au sprite suivant

(2) <P>

passage au sprite précédent

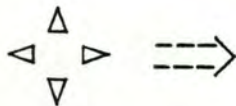
(3) <0>

Sélection du sprite courant. Ce spr-
ite est sélectionné pour subir une
translation

ZM:: 'SELECT POSITION'

ZT:: réticule

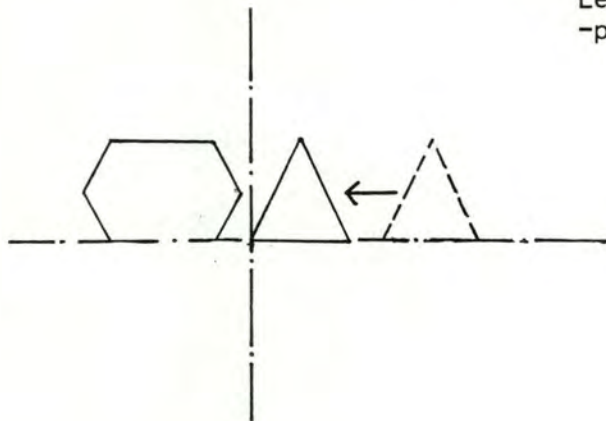
invitation à indiquer à l'aide du
réticule la position finale du spr-
ite après translation



<CR> ==>

déplacement du réticule suivant la
direction indiquée par la flèche ;
pour pointer la position

assignation de la position actuelle
du réticule à la position finale
Le sprite courant disparaît et réap-
paraît à la nouvelle position.



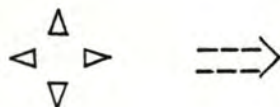
C.12. S/cale (Graphic)

Fonction: Mise en échelle d'un sprite: agrandissement et rapetissement du sprite dans les deux dimensions

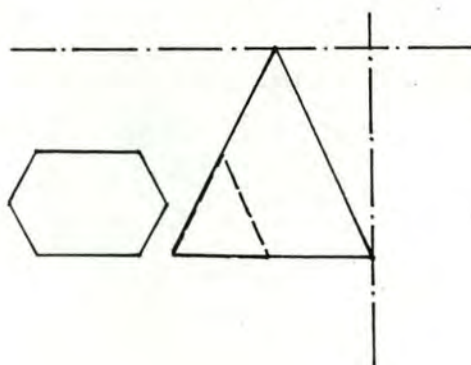
<u>Utilisateur</u>		<u>IRIS</u>
<S>	⇒⇒⇒	ZE:: 'SCALE' ZM:: 'SELECT SPRITE: 0/k N/ext Previous' invitation à indiquer le sprite affiché dont la mise en échelle est attendue. Le sprite courant est affiché en haute luminosité

* Sélection du sprite

(1) <N>		passage au sprite suivant
(2) <P>		passage au sprite précédent
(3) <0>		Sélection du sprite courant; ce sprite est sélectionné pour subir une mise en échelle ZM:: 'SELECT RIGHTCORNER' invitation à indiquer, à l'aide du réticule, le coin supérieur droit du Box (RIGHTCORNER), en supposant que le coin inférieur gauche est resté toujours le même



<CR> ⇒⇒⇒



déplacement du réticule suivant la direction indiquée par la flèche pour pointer le RIGHTCORNER

assignation de la position actuelle du réticule au point supérieur droit
affichage du Box à l'écran; affichage du sprite en échelle dans le Box et disparition du Box

C.13. Q/uit (Graphic)

Fonction: Abandon de l'état Graphic
Retour à l'état Monitor

<u>Utilisateur</u>		<u>IRIS</u>
<Q>	==>	ZE:: 'MONITOR' passage de l'état Graphic à l'état Monitor ZM:: 'G/et R/otate T/ranslate S/cale D/elete Q/uit' présentation des commandes activa- bles à cet état

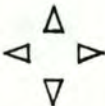
C.14. Q/uit (Text)

Fonction: Abandon de l'état Text
Retour à l'état Monitor

<u>Utilisateur</u>		<u>IRIS</u>
<Q>	==>	ZE:: 'MONITOR' passage de l'état Text à l'état Monitor ZM:: 'G/et R/otate T/ranslate S/cale D/elete Q/uit' présentation des commandes activa- bles à cet état

C.15. E/dit (Text)

Fonction: Edition des chaînes de caractères à l'écran selon les caractéristiques spécifiées

<u>Utilisateur</u>		<u>IRIS</u>
<E>	==>	ZE:: 'EDIT' ZM:: ' '
		invitation à composer la ligne de texte à éditer sur l'écran; chaque ligne de texte peut avoir au maximum 16 caractères.
<'ligne de texte'>		ZM:: 'GET?'
		demande s'il faut enregistrer cette ligne
<G>		enregistrement de la ligne ZM:: 'SIZE: ITALIC: TILT: '
		invitation à préciser les caractéristiques de la ligne à éditer:
		grandeur 1...25 italique 0 90 inclinaison 0 259
<(grandeur)>		
<(italique)>		
<(inclinaison)>	==>	ZT:: réticule ZM:: 'SELECT POSITION'
		invitation à indiquer la position du premier caractère de la ligne, à l'aide du réticule
	==>	déplacement du réticule pour pointer la première position de la ligne
<CR>	==>	affichage de la ligne de texte à la position indiquée et aux caractéristiques spécifiées

TEXT

C.16. D/elele (Text)

Fonction: Suppression d'une ligne de texte

<u>Utilisateur</u>		<u>IRIS</u>
<D>	==>	ZE:: 'DELETE' zm:: 'SELECT LINE: 0/k N/ext Previous'
		invitation à indiquer la ligne de texte éditée à l'écran, à supprimer
		La ligne courante est affichée en haute luminosité pour distinguer avec les autres lignes

* Sélection de la ligne

(1) <N>		passage à ligne suivante
(2) <P>	==>	passage à la ligne précédente
(3) <0>		sélection de la ligne courante
		ZM:: 'DELETE LINE: 0/k N/o'

* Confirmation

(1) <0>	==>	la ligne sélectionnée est effacée de l'écran et est supprimée en mé- moire
(2) <N>		annulation de la commande

C.17. M/odif (Text)

Fonction: Modification des caractéristiques associées à une ligne

<u>Utilisateur</u>		<u>IRIS</u>
<M>	==>	ZE:: 'MODIF' ZM:: 'SELECT LINE: 0/k N/ext Previous' invitation à indiquer la ligne de texte éditée à l'écran, à modifier La ligne courante est affichée en haute luminance
* Sélection de la ligne		
(1) <N>		passage à la ligne suivante
(2) <P>	==>	passage à la ligne précédente
(3) <0>		sélection de la ligne courante
		ZM:: 'SIZE: ITALIC: TILT: ' invitation à indiquer les modifica- -tions des caractéristiques
* Modifications des caractéristiques		
<(grandeur)> <(italique)> <(inclinaison)>	==>	ZT:: réticule ZM:: 'SELECT POSITION' invitation à indiquer la nouvelle position de la ligne déplacement du réticule pour poin- -ter la nouvelle position
<CR>	==>	affichage de la ligne suivante des nouvelles caractéristiques et à la nouvelle position

CONCLUSION du chapitre 3.

Au début du présent travail, nos objectifs concernant le logiciel graphique interactif IRIS sont plus ambitieux que ceux spécifiés dans le 3^e chapitre et réellement réalisés par le programme IRIS.

Nos objectifs dans la période de préparation intégraient les fonctions de manipulation de groupe de sprites composés, donc qui subissent déjà des opérations de transformation. Ces fonctions offriraient à l'utilisateur la possibilité d'effectuer des opérations de rotation, de translation et de zooming sur une entité graphique composée de sprites plus élémentaires. L'introduction de ce niveau éviterait à l'utilisateur la reconstruction répétitive des parties d'image quasi-identiques. Ces facilités sont souvent absentes dans les logiciels graphiques interactifs.

A la mesure de l'avancement du projet, tenant compte des limites de temps, et des difficultés techniques, nous avons limité nos objectifs à ceux spécifiés dans le présent chapitre.

Ces objectifs sont atteints par le programme IRIS qui permet à l'utilisateur de composer facilement une image graphique avec possibilité de manipuler des lignes de textes.

La composition de l'image est plus hiérarchisée et structurée puisque bénéficiant en plus des niveaux existants, à savoir l'image et les sprites, un niveau intermédiaire, celui des scènes composées de sprites.

Comme pour tout logiciel interactif graphique, la convivialité est une caractéristique importante. Elle détermine essentiellement la qualité du programme. Nous avons toujours essayé de favoriser cette propriété en simulant le dispositif d'entrée par pointage (système à réticule), pour donner à l'utilisateur l'impression de travailler avec une souris. Cependant la rapidité d'un tel système est assez limitée. La performance du logiciel serait considérablement accrue s'il disposait des dispositifs matériels d'entrée: tablette graphique ou souris.

BIBLIOGRAPHIE

- J. ENCARNACAO
Computer Graphics : Programmierung und Anwendung
von Graphischen Systemen
R. Oldenburg Verlag, Munich, 1975.

- W.K. GILOI
Interactive Computer Graphics : Data Structures,
Algorithms, Languages
Prentice Hall, 1978.

- P. MORVAN , M. LUCAS
Images et Ordinateur - Introduction à l'infographie
interactive
Larousse, Série Informatique, Paris, Septembre 1976.

- W. NEWMAN , R.F. SPROULL
Principles of Interactive Computer Graphics
Mc Graw Hill, New York, Second Edition, 1979.

- M. LUCAS
Contribution à l'étude des techniques de communication
graphique avec un ordinateur. Eléments de base des
logiciels graphiques
Thèse d'Etat, IMAG, Grenoble, Décembre 1977.

- FOLEY, VAN DAM
Fundamentals of Interactive Computer Graphics
Addison Wesley

* * *