Techniques de programmation et d'utilisation en mode conversationnel des terminaux graphiques

Michel Lucas

N° d'ordre

THESE

présentée à

LA FACULTE DES SCIENCES DE L'UNIVERSITE DE GRENOBLE

pour obtenir

LE GRADE DE DOCTEUR DE TROISIEME CYCLE
" MATHEMATIQUES APPLIQUEES "

par

Michel Lucas

Techniques de programmation et d'utilisation en mode conversationnel des terminaux graphiques

Thèse soutenue le 29 Juin 1968, devant la Commission d'examen :

MM. J. KUNTZMANN — Président
N. GASTINEL — Examinateur
L. BOLLIET — Examinateur

FACULTE DES SCIENCES

UNIVERSITE DE GRENOBLE

LISTE DES PROFESSEURS

-:-:-:-:-

DOYEN HONORAIRE :

M. MORET

DOYEN :

M. BONNIER

PROFESSEURS TITULAIRES :

MM.	NEEL Louis	Chaire de Physique Expérimentale
	HEILMANN René	Chaire de Chimie
	KRAVTCHENKO Julien	Chaire de Mécanique Rationnelle
	CHABAUTY Claude	Chaire de Calcul différentiel et intégral
	BENOIT Jean	Chaire de Radioélectricité
	CHENE Marcel	Chaire de Chimie Papetière
	FELICI Noël	Chaire d'Electrostatique
	KUNTZMANN Jean	Chaire de Mathématiques Appliquées
	BARBIER Reynold	Chaire de Géologie Appliquée
	SANTON Lucien	Chaire de Mécanique des Fluides
	OZENDA Paul	Chaire de Botanique
	FALLOT Maurice	Chaire de Physique Industrielle
	KOSZUL Jean-Louis	Chaire de Mathématiques
	GALVANI O.	Mathématiques
	MOUSSA André	Chaire de Chimie Nucléaire
	TRAYNARD Philippe	Chaire de Chimie Générale
	SOUTIF Michel	Chaire de Physique Générale
	CRAYA Antoine	Chaire d'Hydrodynamique
	REULOS R.	Théorie des Champs
	BESSON Jean	Chaire de Chimie
	AVANT Yves	Physique Approfondie

GALLISSOT	Mathématiques
Mlle LUTZ Elisabeth	Mathématiques
BLAMBERT Maurice	chaire de Mathématiques
BOUCHEZ Robert	Physique Nucléaire -
LLIBOUTRY Louis	Géophysique
MICHEL Robert	Chaire de Minéralogie et Pétrographie
BONNIER Etienne	Chaire d'Electrochimie et d'Electrométallurgie
DESSAUX Georges	Chaire de Physiologie Animale
PILLET E.	Chaire de Physique Industrielle et Electrotechnique
YOCCOZ Jean	Chaire de Physique Nucléaire Théorique
DEBELMAS Jacques	Chaire de Géologie Générale
GERBER R.	Mathématiques
PAUTHENET R.	Electrotechnique
VAUQUOIS B.	Chaire de Calcul électronique
BARJON R.	Physique Nucléaire
BARBIER Jean-Claude	Chaire de Physique
SILBER R.	Mécanique des fluides
BUYLE-BODIN Maurice	Chaire d'Electronique
DREFUYS B.	Thermodynamique
KLEIN J.	Mathématiques
VAILLANT F.	Zoologie et Hydrobiologie
ARNAUD Paul	Chaire de Chimie
SENGEL P.	Chaire de Zoologie
BARNOUD F.	Chaire de Biozynthèse de la cellulose
BRISSONNEAU P.	Physique
GAGNAIRE	Chaire de Chimie Physique
Mme KOFLER L.	Botanique
DEGRANGE Charles	Zoologie
PEBAY-PEROULA J.C.	Physique
RASSAT A.	Chaire de Chimie Systématique
DRUCROS P.	Chaire de Cristallographie Physique
DODU Jacques	Chaire de Mécanique Appliquée I.U.T.
ANGLES D'AURIAC P.	Mécanique des Fluides
LACAZE A.	Thermodynamique

BRIERE G.	Physique M.P.C.
DESRE G.	Chimie S.P.C.N.
LAJZROWICZ J.	Physique M.P.C.
VALENTIN P.	Physique M.P.C.
BERTRANDIAS J.P.	Mathématiques Appliquées - TMP
LAURENT P.J.	Mathématiques Appliquées - TMP
CAUBET J.P.	Mathématiques Pures
PAYAN J.J.	Mathématiques
Mme BERTRANDIAS F.	Mathématiques Pures MPC
LONGEGUEUE J.P.	Physique
NIVAT M.	Mathématiques Appliquées
SOHM J.C.	Electrochimie
ZADWORNY F.	Electronique
DURAND F.	Chimie Physique
CARLER G.	Biologie Végétale
AUBERT G.	Physique M.P.C.
DELPUECH J.J.	Chimie Organique
PFISTER J.C.	Physique C.P.E.M.
CHIBON P.	Biologie Animale
IDELMAN S.	Physiologie Animale
BOUVARD Maurice	Hydrologie
RICHARD Lucien	Botanique
PELMONT Jean	Physiologie Animale
BLOCH D.	Electrotechnique I.P.
BOUSSARD J.Claude	Mathématiques Appliquées I.P.
MOREAU René	Hydraulique I.P.
BRUGEL L.	Energétique I.U.T.
SIBILLE R.	Construction Mécanique I.U.T.
ARMAND Yves	Chimie I.U.T.
BOLLIET Louis	Informatique I.U.T.
KUHN Gérard	Energétique I.U.T.
GERMAIN J.Pierre	Construction Mécanique I.U.T.
CONTE René	Thermodynamique
JOLY J.René	Mathématiques Pures
PIERY Yvette	Biologie Animale

PROFESSEURS SANS CHAIRE :

MM.	GIDON P.	Géologie et Minéralogie
	GIRAUD P.	Géologie
	PERRET R.	Servomécanisme
Mme	BARBIER M.J.	Electrochimie
Mme	SOUTIF J.	Physique
	COHEN J.	Electrotechnique
	DEPASSEL R.	Mécanique des Fluides
	GASTINEL N.	Mathématiques Appliquées
	GLENAT R.	Chimie
	BARRA J.	Mathématiques Appliquées
	COUMES A.	Electronique
	PERRIAUX J.	Géologie et Minéralogie
	ROBERT A.	Chimie Papetière
	BIAREZ J.P.	Mécanique Physique
	BONNET G.	Electronique
	CAUQUIS G.	Chimie Générale
	BONNETAIN L.	Chimie Minérale
	DEPOMMIER P.	Etude Nucléaire et Chimie Atomique
	HACQUES Gérard	Calcul Numérique
	POLOUJADOFF M.	Electrotechnique

PROFESSEURS ASSOCIES

MM.	NAPP-ZINN	Botanique
	RODRIGUES Alexandre	Mathématiques Pures
	STANDING Kenneth	Physique Nucléaire

MAITRES DE CONFERENCES

MM.	LANCIA Roland	Physique Atomique
Mme	KAHANE J.	Physique
	DEPORTES C.	Chimie
Mme	BOUCHE L.	Mathématiques
	SARROT-REYNAUD	Géologie Propédeutique
Mme	BONNIER M.J.	Chimie
	KAHANE A.	Physique Générale
	DOLIQUE J.M.	Electronique

MAITRE DE CONFERENCES ASSOCIES :

SAWCZUK A.	Mécanique des Fluides
CHEEKE J.	Thermodynamique
YAMADA O.	Physique du Solide
NATR Lubomir	B.M.P.V.
NAYLOR Arch	Physique industrielle
SILBER Léo	Radioélectricité
NOZAKI Akihiro	Mathématiques Appliquées
RUTLEDGE Joseph	Mathématiques Appliquées
DONOHO Paul	Physique Générale
EGGER Kurt	B.M.P.V.

Je tiens à remercier :

Monsieur Le Professeur Jean KUNTZMANN, Directeur du Service de Mathématiques Appliquées, qui a bien voulu me faire l'honneur de présider le Jury de thèse.

Monsieur le Professeur Noël GASTINEL, Directeur du Laboratoire de Calcul, qui s'est toujours vivement intéressé à mes travaux.

Monsieur Louis BOLLIET, Maître de Conférences à l'I.U.T. d'Informatique, qui m'a orienté vers la recherche en programmation, m'a donné l'idée de cette thèse et m'a toujours encouragé.

Monsieur Olivier LECARME, Maître-Assistant à l'Institut de Programmation, qui a accepté la lourde tâche de me guider tout au long de mes recherches, ce qu'il a fait avec une compétence et une patience dont je lui suis reconnaissant.

Tous mes camarades du Laboratoire, en particulier :
Messieurs BELLOT, CLAUZEL, GUIBOUD-RIBAUD, LE HEIGET, SIRET, qui, par leurs critiques toujours justifiées et leurs nombreuses suggestions, m'ont permis d'améliorer sans cesse la qualité des résultats que j'ai obtenus.

Le personnel des services de dactylographie et de reproduction à qui je dois la réalisation pratique de cet ouvrage.

TABLE DES MATIERES

-.-.-.-.-

I N T R O D U C T I O N

Depuis que l'homme emploie des ordinateurs, un des problèmes majeurs qu'il a dû résoudre est celui de l'introduction des données dans le calculateur et de la récupération des résultats sous une forme immédiatement utilisable. La rapide évolution des techniques d'utilisation des ordinateurs et les performances de plus en plus grandes de ceux-ci ont conduit à rechercher des moyens de communication entre l'homme et la machine qui soient de plus en plus perfectionnés.

Habituellement, l'introduction des données est faite sous forme de cartes ou de bandes perforées, matériaux encombrants et fragiles, de surcroît peu commodes à manier. La sortie des résultats s'effectue sur une imprimante ou éventuellement sur les mêmes supports. Le défaut majeur de ces moyens d'entrées-sorties est leur lenteur, qui n'est compensée qu'insuffisamment par les possibilités de simultanéité. La forme des résultats est difficilement exploitable rapidement, et il n'y a aucune possibilité de dialogue entre l'homme et la machine, ces dispositifs étant tous à sens unique. Enfin, les nombreux intermédiaires entre le programmeur et l'ordinateur sont autant de sources d'erreurs.

Récemment, un grand pas a été fait lors de l'apparition de machines à écrire (que nous appellerons par la suite consoles) pouvant être connectées directement à un ordinateur pour des applications en temps partagé. Leur emploi a permis d'éliminer beaucoup d'opérations d'entrées-sorties, mais surtout on a vu apparaître la notion de conversation entre l'homme et la machine. C'est cette possibilité de donner des renseignements à l'ordinateur et de recevoir en retour une réponse qui a fait donner à ce mode de travail le qualificatif de conversationnel. Ces terminaux étant beaucoup plus lents que les dispositifs précédemment cités (dix à vingt caractères par seconde), le

problème s'est vite posé de trouver un moyen de communication qui soit plus rapide.

La première utilisation d'un terminal graphique remonte à l'année 1950. Depuis, le nombre des consoles de visualisation s'est accru très lentement en raison de leur coût très élevé, du manque de programmes à vocation graphique et de la difficulté de leur emploi en temps partagé. Cependant, il s'agit là d'une véritable révolution, comparable même à l'introduction des premiers ordinateurs dans la vie courante. Pour la première fois, il est fait appel à une des facultés les plus puissantes de l'homme : la possibilité de prendre une décision quasi-immédiate au vu d'une image (courbe, réseau, coupe, perspective etc...). Pour la première fois, un calcul peut être suivi pas à pas, corrigé au fur et à mesure de son déroulement. L'adjonction d'un clavier alphanumérique à une console de visualisation, permettant ainsi de donner des renseignements directement à l'ordinateur, fait de ces consoles de puissants instruments pour la résolution de problèmes très divers, grâce à un dialogue constant entre l'utilisateur et le calculateur.

Nous nous proposons de décrire les méthodes d'emploi des consoles de visualisation en illustrant notre propos de la description détaillée d'un système de dessin que nous avons conçu et réalisé et qui est en usage à l'Institut de Mathématiques Appliquées de GRENOBLE.

CHAPITRE I

D E S C R I P T I O N D U M A T E R I E L

A - GENERALITES SUR LES CONSOLES DE VISUALISATION

1) Classification des consoles de visualisation

On peut diviser les consoles de visualisation en deux groupes. Le premier groupe est constitué d'unités qui ne peuvent afficher que des caractères alphabétiques, numériques ou spéciaux, en des endroits bien déterminés de l'écran. Nous appellerons ces unités terminaux alphanumériques. Le deuxième groupe comprend des unités capables d'afficher des points, des vecteurs et du texte en n'importe quel endroit de l'écran. Nous les appellerons terminaux graphiques.

Les deux groupes de consoles utilisent un tube à rayons cathodiques pour obtenir une image. La plus grosse différence entre les deux types vient de leurs coûts respectifs et de leur souplesse d'emploi.

2) Fonctionnement d'une console de visualisation.

* Le tube à rayons cathodiques.

C'est un tube tout à fait analogue à celui qui est employé dans un poste de télévision ordinaire. Sa face interne est recouverte d'une matière phosphorescente qui s'illumine sous l'impact d'un faisceau d'électrons. La source d'électrons est une cathode placée au fond du tube. Le flot d'électrons est contrôlé par une grille qui permet de modifier à volonté la quantité d'électrons émis en fonction de la différence de potentiel existant entre la cathode et la grille de contrôle. Plus la quantité d'électrons entrant en contact avec l'écran est grande, plus la lumière obtenue est intense.

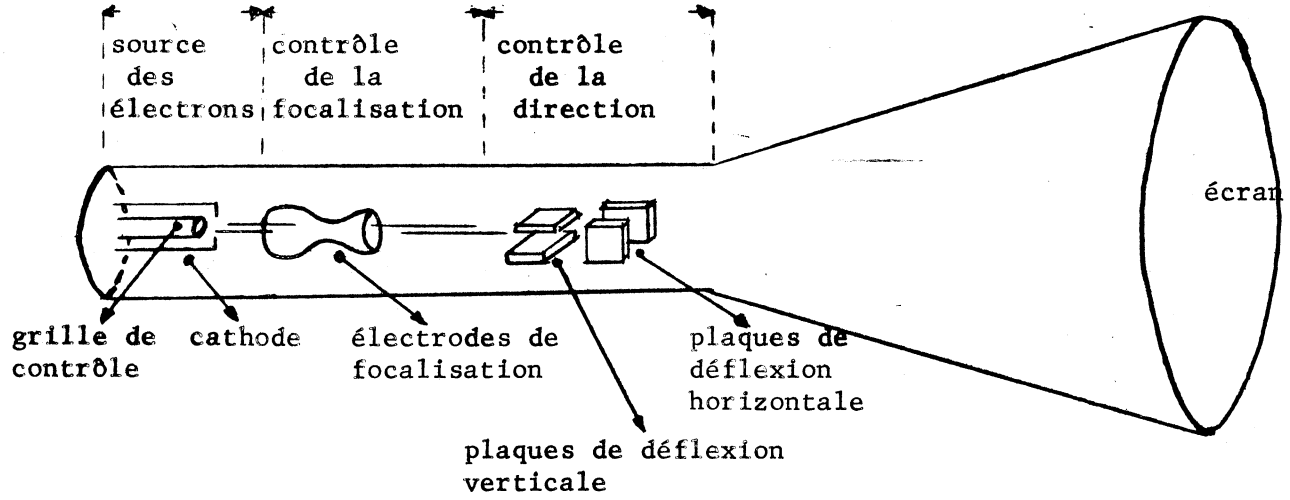
Il faut aussi pouvoir diriger le faisceau d'électrons et le focaliser, afin d'obtenir une tache lumineuse relativement petite à l'endroit voulu. Ceci est obtenu grâce à un dispositif de focalisation qui est placé immédiatement après la grille de contrôle, et un dispositif de deflexion du faisceau qui est placé entre le dispositif de focalisation et la face avant du tube.

Ces deux dispositifs utilisent soit des propriétés électrostatiques, soit des propriétés électromagnétiques pour agir sur le faisceau. Chaque modèle de tube a des avantages et des inconvénients. Leurs caractéristiques principales sont données ci-dessous :

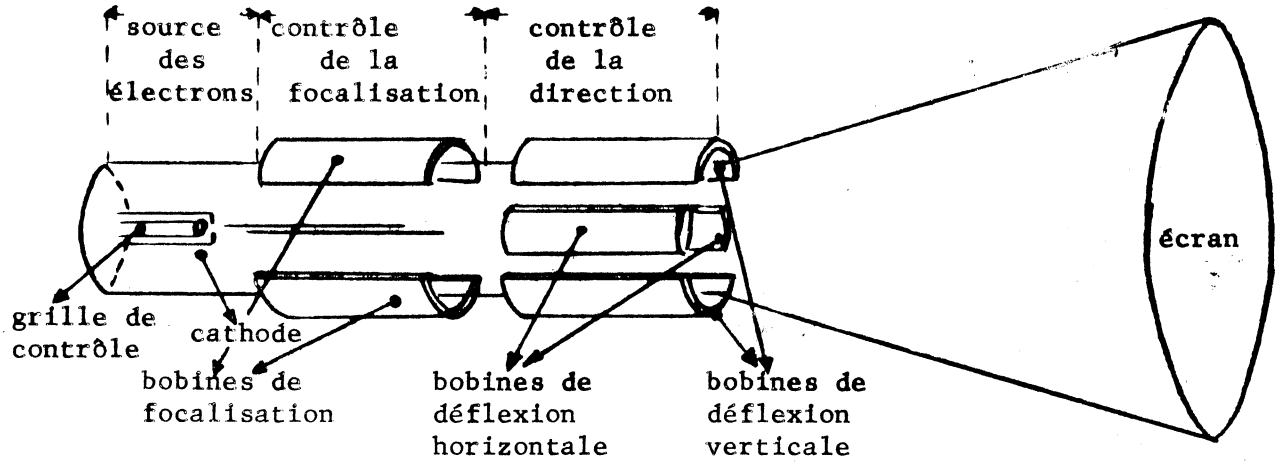
Caractéristique	Dispositif électromagnétique	Dispositif électrostatique
Focalisation du faisceau	Bonne dans tout le tube	Mauvaise aux extrémités
Structure interne	simple	complexe
Longueur du tube	courte	longue
Brillance	forte	moyenne
Sensibilité de déflexion	haute	faible
Complexité des amplificateurs	haute	faible
Vitesse de déflexion	lente	rapide
Poids	lourd	léger

Les qualités propres à chacun de ces types étant hautement désirables, la technique électronique a permis de mettre au point un tube composite qui possède beaucoup des avantages et peu des défauts de chacun des deux groupes précédents. Par exemple, l'apport d'un système secondaire de focalisation électromagnétique a permis d'améliorer la focalisation obtenue par un système principal électrostatique. De la même façon, un système principal de déflexion électromagnétique permet d'obtenir une image lumineuse d'intensité constante tandis que le système secondaire permet d'obtenir une vitesse de déflexion très grande.

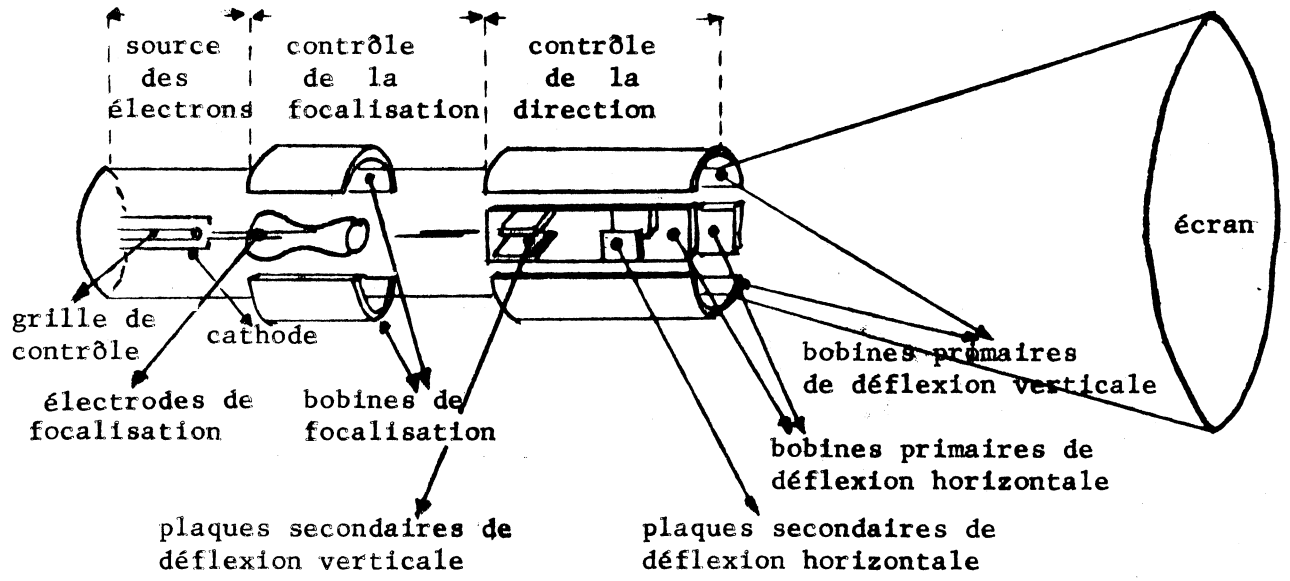
(I)



TUBE CATHODIQUE A CONTROLE ELECTROSTATIQUE



TUBE CATHODIQUE A CONTROLE ELECTROMAGNETIQUE



TUBE CATHODIQUE A CONTROLE HYBRIDE

Généralement, les dispositifs de déflexion électromagnétiques sont employés dans les cas où les déplacements du faisceau d'électrons sont importants, et les dispositifs électrostatiques pour des déplacements plus petits, du genre de ceux qui sont exigés pour tracer des caractères. Le fait d'employer un tube disposant d'un système hybride n'est pas une garantie de bonnes performances. En effet, on obtient plutôt des performances moyennes qui peuvent ne pas être satisfaisantes pour certaines applications particulières, demandant par exemple une haute sensibilité de déflexion.

* l'écran.

La surface utilisable de l'écran forme généralement un carré de trente centimètres de côté. Certains terminaux possèdent un écran plus petit (terminaux alphanumériques), plus grand (terminaux graphiques) ou encore rectangulaire. L'écran est découpé généralement par 1024 lignes et 1024 colonnes. En fait, la précision de ce quadrillage varie non seulement en fonction de la taille de l'écran mais aussi en fonction de la taille de la tache lumineuse faite par le faisceau d'électrons sur l'écran. Le diamètre moyen de la tache lumineuse varie entre 3 et 4,5 dixièmes de millimètre. Ceci veut dire que si on allume deux points situés l'un à côté de l'autre, on ne verra en général qu'une seule tache. Bien que la taille de la tache puisse être réduite, il n'est pratiquement d'aucune utilité de dépasser la finesse d'un quadrillage de 2048 x 2048 points, et pour certaines applications, un quadrillage de 512 x 512 points s'est révélé être suffisant.

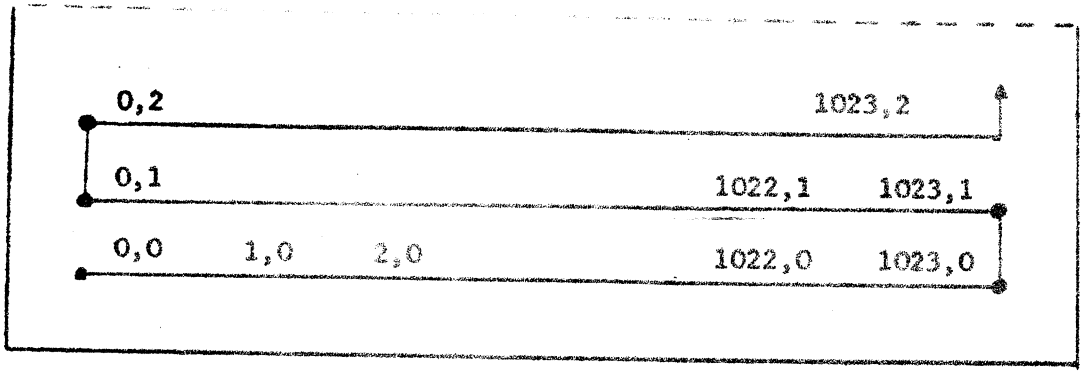
La tache lumineuse produite par le faisceau d'électrons s'affaiblit et disparaît rapidement. La durée de l'illumination est appelée persistance ou rémanence. La persistance peut avoir une durée allant de moins de une microseconde à plusieurs secondes. Sa durée est déterminante dans le choix d'une console de visualisation. En effet, pour obtenir une image stable, il est nécessaire de la régénérer un grand nombre de fois, de manière que l'oeil n'ait pas le temps de voir le dessin

s'éteindre. Plus la persistance est brève, plus le nombre de cycles de régénération sera élevé. En revanche, si la persistance est de trop longue durée, lors d'un changement d'image, la nouvelle viendra momentanément en surimpression sur l'ancienne. On utilise généralement un matériau dont la persistance est telle qu'il suffit de régénérer l'image de trente à soixante fois par seconde. Il existe cependant des tubes dont la persistance permet de ne régénérer l'image que dix à quinze fois par seconde.

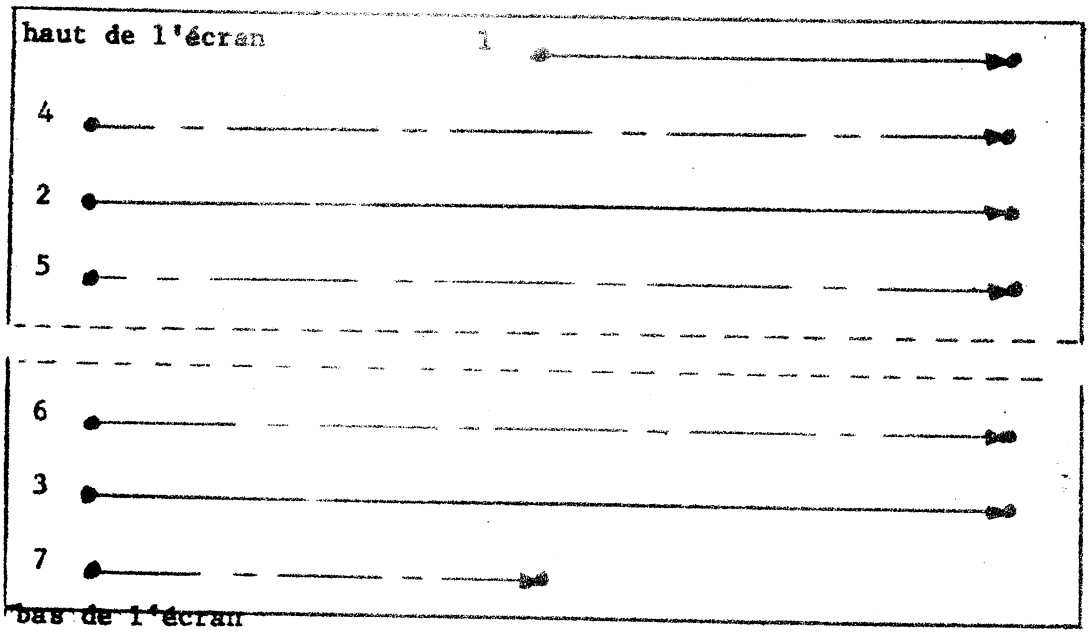
* balayage de l'écran.

La façon de balayer avec le faisceau d'électrons est aussi un facteur important qui intervient surtout au niveau du prix de revient d'un terminal. La nature très particulière de l'emploi des terminaux alphanumériques a conduit à limiter sérieusement leurs possibilités en matière de dessin. Le nombre de caractères par ligne ainsi que le nombre de lignes disponibles sur l'écran sont fixés par le fabricant. Dans le matériel le plus simple, les caractères ne sont mis en place que par rapport au premier caractère de la première ligne, ce qui oblige à utiliser des blancs pour cadrer le texte à afficher. D'autres équipements permettent une plus grande souplesse en autorisant la désignation d'un caractère par sa place dans une ligne donnée.

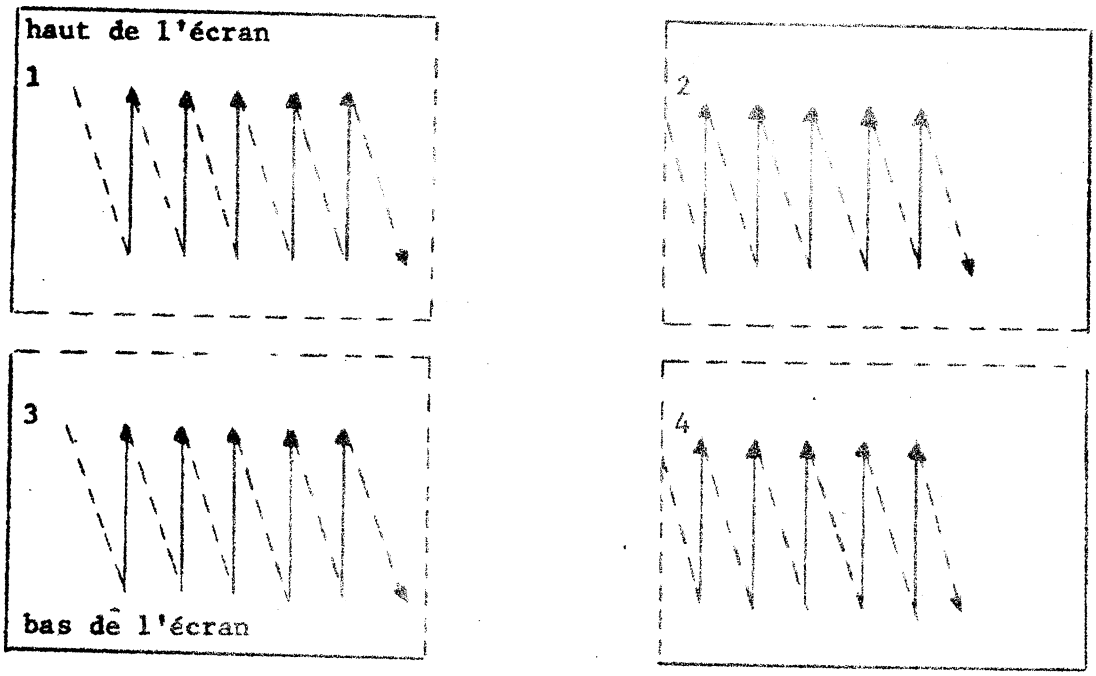
Ces limitations ont permis d'utiliser des techniques visant à réduire le coût d'un terminal alphanumérique. On utilise plusieurs méthodes pour balayer l'écran. La première méthode est celle du balayage horizontal continu. Le faisceau d'électrons part d'un point donné et balaie tous les points de l'écran en ordre séquentiel et à une vitesse donnée. Cette vitesse étant constante, il est facile, à l'aide d'une horloge interne, de connaître les coordonnées du point que le faisceau est en train d'éclairer. En augmentant ou en diminuant le flux d'électrons, on peut éteindre ou allumer certains points du parcours, ce qui permet d'obtenir n'importe quel caractère.



BALAYAGE HORIZONTAL CONTINU



BALAYAGE PAR ENTRELACS



BALAYAGE EN DENTS DE SCIE

En fait, pour des raisons techniques, on emploie des méthodes de balayage plus complexes. La première de ces méthodes est le balayage normal de la télévision, ou balayage par entrelacs. L'écran est balayé en deux passages. Au cours du premier passage, on éclaire la première ligne du haut de l'écran, puis on descend en éclairant une ligne sur deux. Au cours du deuxième passage, on recommence mais cette fois en partant de la deuxième ligne. On voit donc que le faisceau décrit chaque fois un chemin compris entre deux autres précédemment dessinés. Le gros avantage de cette méthode est qu'il suffit d'ajouter un convertisseur digital-analogique à un appareillage de télévision déjà existant, ce qui permet de réduire fortement le prix de revient du terminal.

Une autre méthode employée est celle du balayage en dents de scie. Dans ce cas, le faisceau lumineux décrit un chemin en dents de scie, dont seules les traces verticales sont illuminées. Pour dessiner un caractère, le terminal utilise alors cinq passages verticaux, sept points pouvant être illuminés à chaque passage.

Dans le cas des terminaux graphiques, le chemin suivi est celui indiqué par la succession des points à éclairer. Ce chemin est bien sûr tout à fait arbitraire. On parle alors de balayage cavalier.

La vitesse de balayage de l'écran est très importante. En effet, la lumière émise par l'impact du faisceau d'électrons sur l'écran disparaît rapidement. L'oeil humain conserve un certain temps une impression de luminosité sur sa rétine. Ceci revient à dire que le point lui semble encore allumé alors qu'il est déjà éteint. Si on a déjà rallumé le point entretemps, l'oeil n'aura pas noté la disparition du point. Il est donc important de pouvoir régénérer très souvent une image. Si la fréquence de régénération est suffisante, l'observateur a l'impression que l'image est allumée en permanence. On dit que l'image est stable. Au cas où la régénération n'est pas assez rapide, l'observateur voit l'image s'allumer et s'éteindre sans arrêt. On dit que l'image clignote ou qu'elle est instable. On voit donc que pour un cycle de régénération donné et suffisant pour que l'image soit stable, plus la vitesse de balayage de l'écran

est grande et plus on pourra afficher de points ou de caractères au cours du même cycle.

* représentation des données.

Suivant le degré de perfectionnement de la console de visualisation, les données peuvent être de types très différents :

- dans le cas le plus simple, il s'agira seulement des coordonnées du prochain point à éclairer, avec éventuellement une indication sur la luminosité de celui-ci.

- sinon, il peut s'agir de véritables instructions (tracé de vecteur par exemple) qui devront être décodées au préalable.

Toutes ces données sont fournies par un calculateur digital, quel que soit le type de la console de visualisation. Elles peuvent être rangées dans la mémoire du calculateur : dans ce cas, c'est le calculateur lui-même qui se charge de l'entretien de l'image. Cette solution présente de nombreux inconvénients. En particulier, elle est très coûteuse en temps machine. Les données peuvent aussi être rangées dans une mémoire intermédiaire attachée à la console de visualisation. C'est alors cette mémoire qui se charge du processus de régénération, le calculateur n'intervenant que pour modifier l'image, c'est-à-dire sa représentation dans la mémoire intermédiaire. Cette mémoire est souvent appelée mémoire d'entretien.

Les premières mémoires d'entretien étaient constituées par des tambours ou des disques. On tend de plus en plus à utiliser des mémoires rapides, qui permettent non seulement de conserver des données, mais aussi d'exécuter un certain nombre de programmes élémentaires. Une caractéristique importante de ces matériels est de permettre ou non des ruptures de séquence, et surtout des appels de sous-programmes. Cette dernière possibilité permet de gagner une place considérable en mémoire.

Dans le cas où l'on n'utilise qu'une seule console de visualisation, la régénération automatique de l'image à l'aide d'une mémoire d'entretien peut sembler une solution coûteuse et complexe, et on peut lui préférer l'utilisation directe du calculateur. Mais dans les applications en temps partagé utilisant des consoles de visualisation, il semble que l'utilisation de mémoires d'entretien soit indispensable pour des raisons techniques et économiques.

* décodage et exécution des instructions.

Que les commandes d'affichage soient lancées par le calculateur lui-même ou soient placées en mémoire intermédiaire, un dispositif de décodage des instructions et de contrôle de leur exécution est nécessaire. Ce dispositif ne traite qu'une seule instruction à la fois. Il peut être très simple dans le cas des consoles alphanumériques et très complexe dans le cas des terminaux graphiques, surtout si l'on a la possibilité d'utiliser des sous-programmes en mémoire intermédiaire. Dans tous les cas, ce dispositif est tout à fait semblable à l'unité de contrôle d'un calculateur. En particulier, lorsque les données sont rangées en mémoire intermédiaire, le dispositif de décodage est capable d'aller chercher lui-même la prochaine instruction à exécuter en mémoire.

Les données sont fournies généralement sous une forme particulière à chaque unité. Cette forme peut être très simple (coordonnées d'un point) ou très complexe (instruction de tracé de vecteur avec différentes indications de densité, luminosité, etc...). Pour éclairer un point, le dispositif de décodage émet un certain nombre de signaux digitaux. Ces signaux sont convertis en signaux analogiques grâce à un convertisseur. Ce sont ces signaux analogiques qui vont guider le faisceau d'électrons.

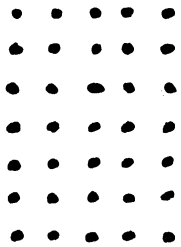
* génération de caractères.

Il existe plusieurs méthodes de génération de caractères. La méthode la plus coûteuse est celle qui consiste à laisser le soin de calculer l'emplacement de chaque point définissant le caractère à l'ordinateur. Il existe heureusement un dispositif appelé générateur de caractères qui se charge d'obtenir automatiquement ce résultat. Un caractère est généralement dessiné au moyen d'une grille de sept points sur cinq. Trois méthodes sont employées : tracé point par point, à l'aide de petits vecteurs ou par contrôle de la forme du faisceau d'électrons.

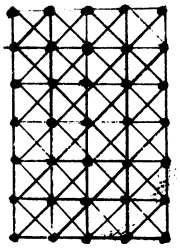
Cette dernière méthode consiste à faire passer le faisceau d'électrons à travers un masque pour l'obliger à prendre la forme du caractère désiré. C'est la méthode la plus rapide, mais il est très malaisé et très coûteux de modifier le jeu de caractères du terminal. Aussi cette méthode est-elle peu employée.

Les deux autres méthodes de tracé de caractères nécessitent un dispositif de décodage qui convertit un code de caractère en une succession d'ordres de tracé de points ou de petits vecteurs. Les caractères obtenus ont habituellement tous la même taille, mais certaines consoles très évoluées possèdent des systèmes permettant d'obtenir des caractères de tailles variées, et même de leur faire effectuer des rotations.

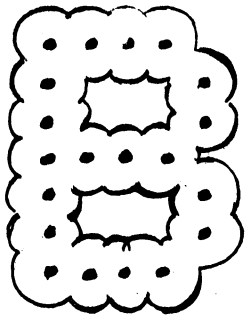
Le schéma III donne différentes méthodes de dessin de caractères. L'idée de base est la même. On dispose d'une grille de définition qui donne les différentes possibilités de tracé de points ou de vecteurs qu'on utilisera pour obtenir un caractère. Plus le nombre de possibilités offertes est grand, et plus le caractère obtenu sera beau, esthétiquement parlant. Le but de ces méthodes est d'accélérer le dessin d'un caractère soit en réduisant le nombre de points qui le forment, soit en employant des vecteurs. En effet, il est plus rapide de tracer un vecteur qu'une succession de points, pour la simple raison que le faisceau d'électrons subit moins d'opérations d'accélération et de décélération pour se mettre en place. Par exemple, si l'on utilise une représentation point par



grille de 7x5 points

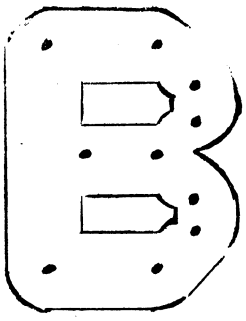


chaque petit vecteur peut être utilisé pour dessiner une portion de caractère.



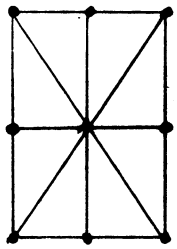
20 points

TRACE POINT PAR POINT

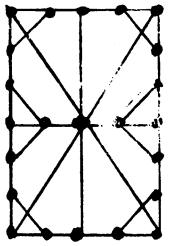


10 vecteurs

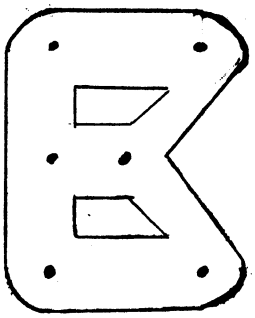
TRACE A L'AIDE DE VECTEURS



grille en étoile

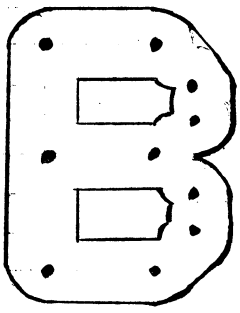


grille en étoile améliorée



7 vecteurs

TRACE A L'AIDE DE VECTEURS



10 vecteurs

TRACE A L'AIDE DE VECTEURS

point, un dessin comprenant 720 caractères ne pourra être régénéré que 75 fois par seconde, alors que dans le cas de la représentation à l'aide de vecteurs, l'image pourra être régénérée 250 fois par seconde. Ceci revient à dire que dans un cycle de régénération suffisant pour que l'image soit stable, on pourra afficher trois fois plus de caractères avec la seconde méthode.

En regardant le schéma III, le lecteur pourra se rendre compte de l'importance de la grille de définition qui est utilisée pour dessiner un caractère. De cette grille va dépendre la vitesse de tracé du caractère et aussi son aspect esthétique. A noter que la grille en étoile est fort peu utilisée pour cette dernière raison. C'est cependant l'utilisation de cette grille qui permet de dessiner le plus rapidement un caractère, car elle utilise de deux à trois fois moins de vecteurs pour définir un caractère que les autres grilles.

* génération de vecteurs.

De la même façon que pour les caractères, on peut calculer chaque point définissant un vecteur à l'aide de l'ordinateur, solution coûteuse et imprécise, ou laisser le calcul à un organe particulier du terminal graphique appelé générateur de vecteurs.

Le gros problème du tracé de vecteurs est celui du choix des points devant définir un vecteur particulier. En effet, à part les droites verticales, horizontales ou diagonales, tous les points du vecteur à dessiner ne se trouvent pas sur un noeud du réseau définissant les coordonnées adressables. La plupart des droites doivent donc être calculées à l'aide d'un algorithme qui permette de placer les points aux meilleurs endroits, de façon à définir le vecteur de manière régulière. C'est un procédé long et imprécis. Les deux caractéristiques d'un générateur de vecteurs doivent donc être une production rapide de lignes droites et un résultat d'apparence convenable.

Un générateur de vecteurs produit des lignes de deux façons : soit point par point, soit à l'aide de petits vecteurs. Dans le premier cas, le générateur de vecteurs calcule les coordonnées de chaque point devant être allumé et les transmet au convertisseur digital-analogique. Certains équipements ne font ce calcul qu'en fonction de la grille habituelle de l'écran, ce qui fait que le problème de l'approximation du tracé n'est pas résolu. D'autres équipements au contraire utilisent pour le tracé des vecteurs une grille plus fine que celle utilisable par le programmeur. La disposition des points est alors telle que l'oeil croit voir une ligne droite parfaite. Le schéma IV illustre ces deux méthodes.

Un générateur de vecteurs opérant point par point ne peut ni abaisser le temps requis pour dessiner une ligne ni augmenter la longueur totale de vecteurs pouvant être dessinée dans une image stable. L'intérêt de tels générateurs réside dans le fait que pour dessiner de longs vecteurs il y a besoin de beaucoup moins de commandes, ce qui fait gagner de la place et du temps. Le calculateur n'a plus à intervenir pour calculer les points définissant un vecteur, ce qui est un gros avantage.

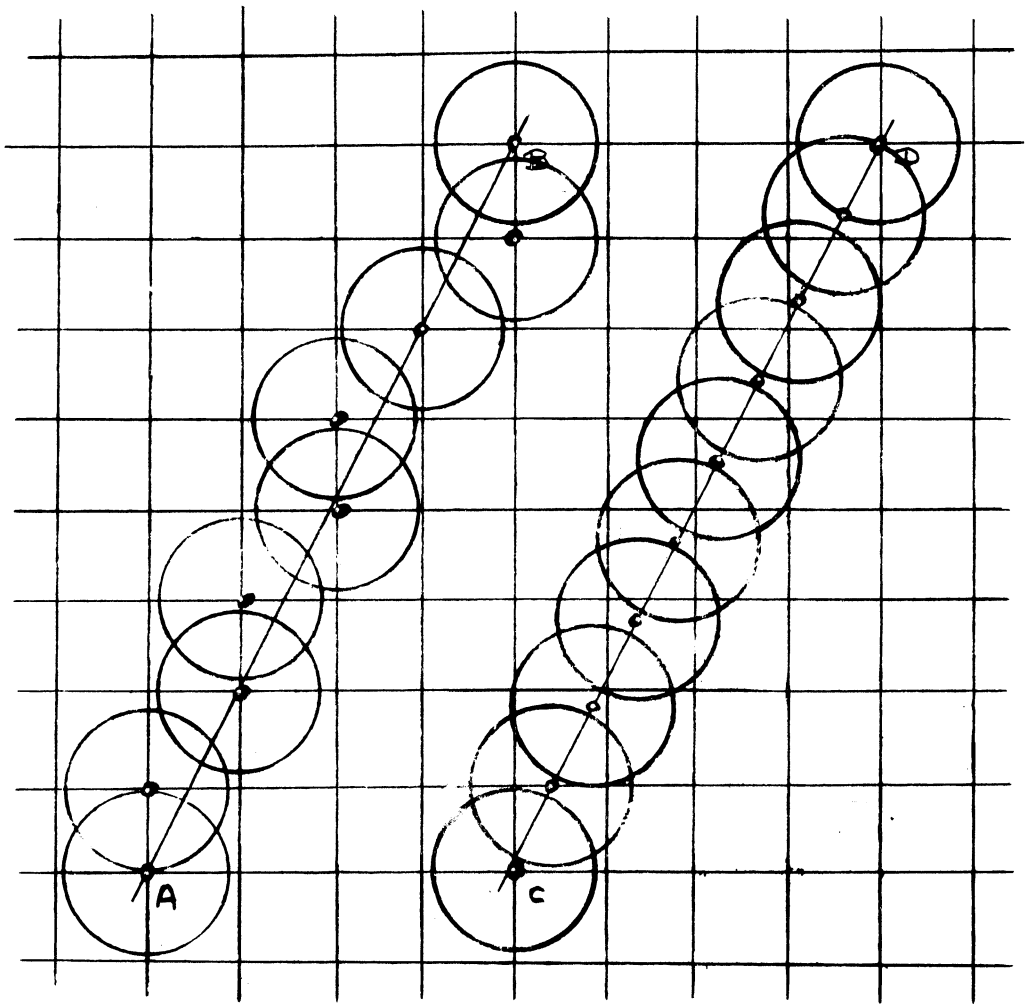
Un générateur de vecteurs utilisant de petits vecteurs élémentaires permettra au contraire de dessiner beaucoup plus de choses, étant donné que chaque ligne sera tracée très rapidement. On a seulement besoin de connaître le point de départ et le point d'arrivée du faisceau lumineux, ce qui évite un certain nombre de calculs. Suivant les équipements, la taille des vecteurs élémentaires varie de quelques centièmes de millimètre à trois dixièmes de millimètre. La longueur de vecteurs dessinés dans une image stable peut aller jusqu'à vingt fois celle obtenue à l'aide de générateurs point par point. C'est pourquoi la plupart des unités graphiques possèdent un générateur de vecteurs à l'aide de vecteurs élémentaires.

* Clavier alphanumérique.

L'introduction des données dans un terminal alphanumérique se fait généralement à l'aide d'un clavier de machine à écrire. Le fait d'appuyer sur une touche place le caractère dans la mémoire et le fait

IV

TRACE DE VECTEURS POINT PAR POINT



Le vecteur AB a été obtenu en faisant une approximation systématique à partir de la grille ordinaire de l'écran.

Le vecteur CD a été obtenu en utilisant une grille plus fine qui n'est pas accessible au programmeur.

apparaître sur l'écran avec les caractères frappés précédemment. La position du caractère est déterminée grâce à un pointeur que l'on peut déplacer de ligne en ligne ou le long d'une ligne. On peut aussi effacer des caractères individuellement ou des lignes entières.

* crayon optique et autres dispositifs de localisation.

Le crayon optique est un dispositif formé d'une cellule photo-électrique reliée à un organe de commande par un fil électrique. Parfois la cellule ne se trouve pas dans le crayon lui-même, mais à l'intérieur de la console de visualisation. Le fil la reliant au crayon optique est alors un fil conducteur de la lumière. On se sert du crayon optique pour désigner un élément éclairé sur l'écran. L'organe de commande du crayon optique prépare alors une interruption qui permet au calculateur, ou au dispositif de décodage de la console de visualisation, de trouver quelle est la portion d'image qui a été vue par le crayon optique. La cellule photo-électrique est équipée en général d'un interrupteur manoeuvré par le doigt, une pédale ou le fait qu'on appuie le crayon sur l'écran. Cet interrupteur permet d'éviter toute fausse reconnaissance dûe à la vue d'une lumière quelconque.

Quand on veut indiquer un point non éclairé de l'écran, le calculateur ou la console de visualisation "suit" le mouvement du crayon optique à l'aide d'un symbole mobile qui se déplace en même temps que la main de l'opérateur. Les techniques de poursuite du crayon optique sont exposées en détail dans le chapitre suivant, à partir de la page 29.

Le crayon capacitif donne directement les coordonnées d'un point lorsqu'il touche une surface conductrice. Cette surface étant rarement transparente, elle se trouve généralement placée sur une tablette en avant de l'écran. Ce crayon capacitif ne peut pas voir de lumière, c'est pourquoi certains terminaux sont équipés de dispositifs qui comparent à tout instant la position du crayon capacitif et celle du faisceau

d'électrons. Une interruption est alors provoquée lorsque le crayon capacitif se trouve à une certaine distance du faisceau d'électrons. Cette facilité permet d'éliminer de gros programmes pour trouver quelle est l'image qu'on est en train de montrer.

Un autre dispositif analogue, la boule roulante, permet d'obtenir les mêmes résultats que le crayon capacitif. Il s'agit d'une boule dont seule une petite portion affleure à la surface d'une tablette. On fait tourner cette boule avec la main, et, de la même façon que dans le cas du crayon capacitif, cette action déplace un symbole sur l'écran, ce qui permet de repérer un point de l'écran. Un dernier dispositif en est dérivé, appelé le manche à balai. Il s'agit de la boule roulante à laquelle on adjoint un manche qui permet de manoeuvrer celle-ci plus facilement. En effet, le déplacement rapide du symbole à l'aide de la boule roulante exige une certaine dextérité. Dans le cas du manche à balai, la vitesse de déplacement du symbole est commandée par l'inclinaison plus ou moins grande du manche.

Actuellement, le crayon optique est l'instrument le plus employé en raison de la simplicité de son emploi. Cependant, les possibilités des autres dispositifs de localisation sont telles, en particulier du point de vue de l'économie de programmation, qu'en dépit de leur prix ils semblent voués à un grand avenir. Mais le crayon optique aura encore la faveur des utilisateurs pendant longtemps.

* Clavier de fonctions.

Il se présente sous la forme d'une boîte surmontée d'un certain nombre de boutons. Chaque bouton est flanqué d'un indicateur lumineux qui peut être allumé ou éteint par programme. Le fait d'appuyer sur un des boutons provoque une interruption. On associe alors à chaque bouton un sous-programme de traitement, ce qui fait que le calculateur sait alors quelle est l'action à entreprendre. Par exemple, si l'utilisateur montre une figure et appuie sur le bouton EFFACER, cette figure disparaîtra de l'écran. On voit donc qu'il s'agit de véritables appels de sous-program-

mes , appels extérieurs au programme lui-même et provoqués par l'utilisateur. Généralement, les boutons de commande associés à un terminal graphique font appel à un programme, alors que les boutons de commande associés à un terminal alphanumérique sont connectés à des dispositifs câblés qui ne peuvent être modifiés par programme.

B - CHOIX D'UNE CONSOLE DE VISUALISATION

1) Les domaines d'utilisation

On peut classer les différents domaines d'application des consoles de visualisation en deux grandes catégories. Le premier domaine est tout ce qui touche à la gestion de fichiers, en ne nécessitant d'avoir à sa disposition que des caractères alphanumériques. Il s'agit par exemple des réservations de place en avion, de certaines formes d'enseignement programmé, de la consultation de fichiers de stock ou encore de fiches de maladie pour un docteur. Le champ est donc immense. Il est tout à fait naturel d'orienter l'utilisateur vers l'emploi de terminaux alphanumériques.

Le deuxième domaine concerne tout ce qui demande un dessin autre que celui de caractères : courbes, réseaux géométrie, dessin industriel, un champ encore plus vaste que le précédent. L'utilisateur sera alors amené à choisir un terminal graphique.

Nous nous proposons d'essayer de montrer quelles sont les options dont on ne peut se passer pour la bonne utilisation d'une console de visualisation et comment on peut se passer des autres si la nécessité s'en fait sentir. Nous verrons les avantages et les inconvénients des différents choix possibles. De nombreux exemples d'utilisation sont donnés dans le chapitre V, à partir de la page 100.

2) Description d'un terminal alphanumérique perfectionné.

Il existe trois grands types de terminaux alphanumériques :

- ceux qui nécessitent l'aide du calculateur pour régénérer l'image.
- ceux qui possèdent une mémoire d'entretien et ne nécessitent l'intervention du calculateur que pour modifier l'image.

- enfin ceux qui possèdent une mémoire d'entretien et qui peuvent non seulement recevoir des informations du calculateur mais aussi lui en envoyer.

Tous les terminaux alphanumériques sont pourvus d'un générateur de caractères et d'un clavier alphanumérique.

Quelles sont les caractéristiques d'un terminal alphanumérique très perfectionné ?

Nous avons déjà vu que celui-ci se devait de posséder un générateur de caractères à l'aide de vecteurs. D'autres dispositifs annexes permettent d'accroître les performances du terminal. Le premier de ces dispositifs est une mémoire d'entretien de grande capacité, pouvant supporter une programmation élémentaire, à savoir des ruptures de séquence et des appels de sous-programmes.

Les autres dispositifs sont essentiellement des dispositifs de communication avec le calculateur et l'utilisateur. Le premier de ces dispositifs est le clavier alphanumérique. C'est certainement le moyen le plus commode d'introduire des données dans la mémoire intermédiaire. Le deuxième dispositif est le clavier de fonctions. Les boutons de ce clavier vont permettre d'effectuer un certain nombre d'opérations de base sur les textes : effacer certaines portions, en décaler d'autres, ajouter ou insérer des phrases à la suite du pointeur, déplacer celui-ci, etc... Ces fonctions sont en général précâblées et ne peuvent être modifiées par programme.

3) Description d'un terminal graphique perfectionné.

La seule chose indispensable sur un terminal graphique est le décodeur d'instructions, qui permet d'obtenir un point sur l'écran. Il s'agit bien d'un minimum vital. A partir de là, on peut ajouter un nombre impressionnant de dispositifs à savoir : un générateur de caractères, un générateur de vecteurs, un générateur de cercles et d'arcs de cercles,

un crayon optique (pour montrer des figures ~~sur~~ l'écran), un crayon capacitif (pour indiquer des positions non éclairées de l'écran) et un certain nombre de dispositifs spéciaux comme une caméra (prise de photos ou de films), un dispositif de digitalisation de photos à partir des négatifs et dans un avenir relativement proche... le téléphone, afin de permettre une entrée directe des données !

4) Conclusion

On voit que le problème du choix entre un terminal graphique ou un terminal alphanumérique est vite résolu. Le vrai problème est en fait de savoir quels sont les dispositifs qu'il faut adjoindre à une console de visualisation. La solution de ce problème dépend tout à fait du genre d'application auquel on destine le terminal à choisir. Il est certain que tous les dispositifs cités peuvent être simulés par programme, et qu'on peut donc s'en passer. Cependant, les résultats obtenus sont moins précis que ceux obtenus à l'aide de générateurs spécialisés, et l'obtention de ces résultats est coûteuse en temps machine. Il est certain aussi que le fait d'employer le calculateur pour régénérer l'image pose un certain nombre de problèmes dans le cas où cette image est le fruit de nombreux calculs. En effet, effectuer de longs calculs, c'est se condamner à ne plus avoir d'image sur l'écran pendant la durée de ceux-ci. Ce n'est pas gênant s'il s'agit par exemple de dessins statiques, mais dans le cas d'images animées, il est évident qu'il serait aberrant de voir l'image disparaître un long moment entre chaque position successive !

En fait le problème du choix des dispositifs est souvent réglé par des considérations financières.

Nous nous proposons dans le chapitre suivant de montrer comment on peut simuler tous les dispositifs précédemment décrits, lorsqu'on dispose d'un matériel élémentaire dont la description est donnée dans les pages qui suivent.

C - LE MATERIEL UTILISE A GRENOBLE

Il s'agit uniquement du matériel employé pour la réalisation du travail qui fait l'objet de cette thèse.

1) Le calculateur.

Nous disposons d'un ordinateur digital PDP-8, machine binaire à mots de douze positions. La capacité de sa mémoire est de 12 K. Elle est découpée en trois champs de 4 K. Chaque champ est lui-même découpé en trente deux pages de cent vingt huit mots. Le cycle de base de la machine est de 1,5 microseconde. Elle possède un accumulateur de douze positions binaires, complété par un lien de une position binaire. Elle ne possède pas d'élément de calcul arithmétique (pas de division, pas de multiplication).

Le PDP-8 a deux modes de fonctionnement :

- mode ION, ou avec interruptions autorisées.

Dans ce mode de travail, le calculateur autorise l'exécution simultanée de deux opérations :

le déroulement normal d'un programme
une opération d'entrées-sorties.

Après avoir commandé une opération d'entrées-sorties, le programme continue de s'exécuter. Quand le transfert de l'information est achevé l'organe d'entrées-sorties provoque une interruption du programme en cours, et le contrôle est donné à un sous-programme dont l'adresse se trouve dans une mémoire privilégiée de la machine. Ce sous-programme exécuté, le contrôle est rendu au programme interrompu. La description détaillée des interruptions est donnée page 49 du présent ouvrage.

- mode IOF, ou sans interruptions autorisées.

Dans ce mode de travail, il n'y a pas de simultanéité possible. Lorsqu'une commande de transfert est lancée, le programme ne peut reprendre son cours tant que la commande n'est pas satisfaite.

Un certain nombre d'organes périphériques sont attachés au calculateur :

- une console connectée directement à l'ordinateur.
- 24 consoles connectées à l'ordinateur par l'intermédiaire d'un multiplexeur.
- un terminal graphique.
- un convertisseur analogique-digital.
- deux lecteurs de bande magnétique.
- un tambour magnétique.
- un interface assurant la liaison avec un ordinateur I.B.M. 7044.

2) Le terminal graphique

Il est constitué essentiellement par le tube cathodique, deux registres non adressables de dix positions binaires contenant les coordonnées du point à allumer, un registre de trois positions binaires contrôlant la luminosité de chaque point et un crayon optique.

Le cycle de base du terminal est de trente cinq microsecondes. Ce cycle n'est pas automatiquement respecté. Chaque point restant allumé environ 0,04 seconde, on voit qu'il est nécessaire d'éclairer chaque point au moins 25 fois par seconde. Une image stable ne pourra contenir qu'un millier de points.

Si l'on désire utiliser le crayon optique, il faut alors respecter un cycle de base de quarante cinq microsecondes.

Le terminal ne possède aucun dispositif de génération de caractères, vecteurs et autres figures. Il ne possède pas non plus de mémoire d'entretien.

3) Les contraintes.

* celles dûes au calculateur.

La plus grosse contrainte est sans conteste le fait de ne pas posséder d'élément de calcul arithmétique. Il faut donc effectuer toute opération de division ou de multiplication par programme. Les mots étant relativement petits, on se heurte à de grosse difficultés, soit pour obtenir un calcul précis, soit pour rassembler un certain nombre de renseignements à l'intérieur d'un mot. Le découpage en pages est aussi un handicap dans la mesure où les programmes doivent être découpés en morceaux de 128 instructions au maximum. D'autre part, l'adressage indirect ne peut se faire qu'à un seul niveau, ce qui rend les recherches en table très malaisées. Enfin, il n'existe aucun dispositif de protection de zônes de la mémoire, ce qui rend la mise au point des programmes assez délicate, toute erreur de programmation pouvant entraîner la destruction de l'ensemble du programme.

* Celles dûes au terminal graphique.

Le plus grave défaut est le manque de mémoire d'entretien. En effet, le PDP-8 est chargé de l'entretien de l'image. Etant donné que son cycle de base est de 1,5 microseconde et que celui du terminal est de 35 à 45 microsecondes, cycle non respecté automatiquement, l'ordinateur va passer une partie importante de son temps à attendre que le terminal graphique soit prêt à éclairer un nouveau point. Le crayon optique est par ailleurs peu précis.

4) Les avantages.

Le plus gros avantage est la rapidité du PDP-8. En effet, cette vitesse est telle que nous avons pu concevoir une simulation de dispositifs tels que générateurs de vecteurs et générateurs de cercle, simulation obtenue à l'aide de programmes interprétés. Les possibilités de simultanéité nous ont permis d'utiliser la console comme un clavier alphanumérique. Un autre avantage non négligeable est le faible coût du matériel.

Le principal avantage du terminal graphique est de posséder un crayon optique, qui permet une localisation commode sur l'écran. D'autrepart, la faculté de pouvoir éclairer des points à des intensités variables est aussi fort intéressante.

Ce matériel élémentaire nous a conduit à étudier certaines techniques de programmation permettant de pallier les insuffisances du terminal graphique. Des considérations générales à propos des techniques de programmation graphique et la description détaillée d'un langage graphique font l'objet du chapitre suivant.

CHAPITRE II

TECHNIQUES DE PROGRAMMATION GRAPHIQUE

A - TECHNIQUES DE PROGRAMMATION GRAPHIQUE

1) Langages graphiques.

Toute image se présente sous la forme d'un ensemble de figures ; ces figures pouvant éventuellement être liées entre elles par un certain nombre de contraintes. La plus grosse difficulté à vaincre est de trouver un moyen commode de représenter en mémoire les figures et les contraintes qui y sont attachées. Il faut aussi que cette représentation permette de manier aisément les objets représentés, de façon à pouvoir leur faire subir facilement des transformations ponctuelles.

Tout terminal graphique, qu'il possède ou non une mémoire intermédiaire, utilise un certain langage qui lui est propre et qui permet de décrire par une suite d'instructions les opérations à effectuer pour obtenir l'image voulue. Ce langage est en général d'un niveau très élémentaire, et il est vite fastidieux d'écrire des programmes de dessin en l'utilisant. D'autre part, il est toujours limité, ses éléments de base étant ceux du terminal graphique, ce qui fait que dans le cas le plus défavorable, on ne pourra employer que des commandes d'affichage de points pour obtenir une figure !

Il paraît donc indispensable de définir un langage de plus haut niveau, composé d'un certain nombre d'instructions du genre "tracé de vecteur" ou "tracé de cercle", chaque instruction pouvant comporter un certain nombre de renseignements portant sur la luminosité, la densité de points, l'utilisation du crayon optique, etc... Ces instructions sont traduites dans le langage propre au terminal graphique pour exécution. Le langage graphique peut être conçu soit comme une insertion dans un langa-

ge de plus haut niveau, soit au contraire comme un tout parfaitement indépendant. Il peut encore être écrit à l'aide d'un langage de très haut niveau, comme LISP, FORTRAN, ALGOL ou PL/1.

L'utilité de ces langages graphiques est grande. Ils permettent de s'affranchir des difficultés de programmation inhérentes à chaque terminal. Ils permettent également de pallier les insuffisances de ces terminaux en donnant la possibilité de simuler un certain nombre de dispositifs qui n'existent pas sur le terminal utilisé. Le fait de simuler des dispositifs permet d'avoir à tout instant un certain nombre de renseignements qui permettent d'éliminer un grand nombre de calculs. Ils permettent enfin, suivant leurs possibilités, de concevoir des systèmes de dessin plus ou moins généraux, utilisant au maximum les facilités qu'ils accordent. On voit donc que le problème est de définir des langages graphiques suffisamment puissants pour permettre de résoudre élégamment les problèmes de représentation de données, de structures des images, de déplacements et de déformations des figures.

Les paragraphes qui suivent ont pour but d'étudier en détail certains aspects de ces langages graphiques. La deuxième partie de ce chapitre sera consacrée à l'étude détaillée d'un langage graphique défini par nous, qui a pour nom LAGROL (LAngage GRaphique pour Ordinateur Limité). Les problèmes de structures et de hiérarchie des images sont abordés au chapitre IV. On trouvera de nombreux renseignements en références 2-1, 2-2, 2-3, 2-4, 2-5 et 2-6.

2) Représentation des données.

* généralités.

La représentation la plus simple consiste à définir chaque figure par une table contenant les coordonnées de tous les points qui la constituent.

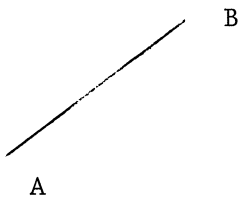
Cette définition point par point est pratiquement la seule exploitable dans le cas de figures ne pouvant être obtenues à partir de fonctions simples (courbes de niveau, par exemple). Cependant, il faut dans la mesure du possible éviter de l'employer, cette représentation étant très difficile à manier. En effet, toute transformation de l'image, aussi minime soit-elle, entraîne une modification de tous les points de la table, opération longue et coûteuse. En fait, dans bien des cas, quelques points suffisent à définir une figure. Toutes les opérations se feront alors sur ces quelques points, d'où un gain de temps de calcul très appréciable. On gagne également une place en mémoire très importante. Nous donnons ci-après un certain nombre de représentations couramment employées.

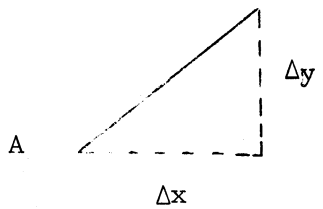
* représentation de points.

Il n'y a ici aucune difficulté, un point étant représenté par ses coordonnées, une suite de points par la suite des coordonnées des points consécutifs. Les coordonnées peuvent être absolues ou relatives (déplacement d'un point par rapport au précédent).

* représentation de vecteurs.

Un vecteur pourra être défini par les coordonnées du point de départ et du point d'arrivée, ou par les coordonnées du point de départ et les composantes scalaires.





* représentation de cercles.

Le choix de la méthode à employer dépendra de la puissance de calcul de l'ordinateur employé. Plus celle-ci sera grande, et moins on aura à donner de renseignement. On donnera par exemple les coordonnées

du centre et un point de la circonférence. On pourra aussi donner trois points de la circonférence. Dans le cas où le calculateur n'est pas assez puissant, il vaut mieux donner le centre et le rayon de la circonférence, ce qui réduit les calculs au strict minimum.

* représentation d'arcs de cercle.

Le choix dépendra encore ici des possibilités de calcul de l'ordinateur. Les représentations demandant le plus grand nombre de calculs utilisent les coordonnées du centre et les coordonnées du point de départ et du point d'arrivée de l'arc, avec la convention que l'on dessine dans le sens trigonométrique ou dans le sens rétrograde. On peut aussi donner le centre, le rayon et les angles polaires des points de départ et d'arrivée de l'arc à dessiner. Une méthode demandant le minimum de calculs consiste à donner les coordonnées du centre, le rayon, le point de départ et le point d'arrivée de l'arc. Dans tous les cas, le tracé de cercles ou d'arcs de cercle sera une opération relativement longue.

* représentation d'ellipses

On utilisera une méthode employant les coordonnées du centre, les coordonnées d'une des extrémités du grand axe et les coordonnées d'une des extrémités du petit axe. On pourra ainsi déterminer facilement les coefficients a et b de l'équation d'une ellipse sous forme normale.

* représentation d'arcs d'ellipse.

On donnera les mêmes renseignements que dans le cas précédent, en ajoutant les coordonnées du point de départ et du point d'arrivée de l'arc à dessiner, une convention étant donnée quant au sens du tracé.

* représentation d'une chaîne alphanumérique.

On donnera les coordonnées du point de départ de la chaîne, suivies de la représentation codée des caractères. Le point de départ de la chaîne pourra être un point du premier caractère (le coin inférieur gauche, plus souvent le centre) ou un point situé à une distance invariable du premier caractère.

A°. CHAINE A REPRESENTER

* représentation matricielle.

Dans le cas où l'on veut représenter des objets dans l'espace à trois dimensions et lorsque l'on veut pouvoir donner de grandes possibilités dans le domaine des transformations ponctuelles, il est intéressant d'utiliser la technique de représentation à l'aide d'un système de coordonnées homogènes. Les courbes seront définies par leur représentation paramétrique. Une figure sera alors représentée par une matrice. L'intérêt de cette méthode est qu'alors toutes les opérations de rotations, similitudes, translations, etc... se réduisent à des opérations de composition de matrices. On a donc une représentation très facile à manier. L'inconvénient de cette représentation est qu'elle coûte beaucoup de place et surtout demande de gros calculs.

3) Définition des figures.

Nous appellerons éléments l'ensemble des images élémentaires qui sont le matériel de base fourni par le terminal graphique. Tout terminal graphique aura l'ensemble des points définis par sa grille élémentaire comme éléments. Un terminal graphique perfectionné mettra à la disposition de l'utilisateur l'ensemble des points, caractères, vecteurs et cercles qu'il peut afficher comme éléments.

Nous appellerons objet un ensemble rigide d'éléments. Cet ensemble aura de plus la propriété suivante : tous les éléments d'un objet sont éclairés avec la même intensité. Ceci revient à dire que si l'on efface un élément d'un objet, l'objet tout entier disparaît. Ce sera par exemple une chaîne de caractères, un triangle, un ensemble de points, plus généralement, tout dessin obtenu au moyen d'une seule instruction. Un objet pourra éventuellement n'être composé que d'un seul élément.

Nous appellerons figure une collection d'objets, à laquelle on donne un nom. Chaque fois qu'il est fait référence à ce nom, c'est l'ensemble de la collection qui est concerné. Les objets constituant une figure sont indépendants. On pourra par exemple éteindre l'un d'entre eux sans éteindre l'ensemble de la figure. Une figure peut être formée d'un seul objet.

Nous appellerons image l'ensemble des figures présentes à un instant donné sur l'écran. Une image peut être formée d'une seule figure.

Nous appellerons fichier graphique l'ensemble de la représentation d'une figure en mémoire et son image sur l'écran.

Une certaine ambiguité semble résider dans le fait que lorsque l'on parle d'une image ou d'une figure, on ne sait jamais s'il s'agit de la représentation en mémoire, ou du dessin sur l'écran. En fait, il n'y a aucune différence entre ces deux notions, et il nous a semblé inutile de définir des termes spéciaux pour désigner l'image sur l'écran ou sa représentation en mémoire. Dans les pages qui vont suivre, sauf mention expresse, nous parlerons toujours de la représentation en mémoire.

Le fond du problème est donc de savoir comment définir de façon satisfaisante une figure et une image. Une image peut être simplement une suite de figures, la représentation de chaque figure étant recopiée en

entier dans la liste à chaque occurence. Cette méthode pose le problème de l'utilisation de figures de base. On a souvent besoin de dessiner un grand nombre de fois le même symbole, par exemple une résistance dans un dessin de réseau électrique. L'idée est d'avoir à sa disposition une représentation toute prête de ce symbole et de l'utiliser autant de fois qu'on le voudra. Ce symbole étant destiné à apparaître en des places différentes de l'écran, il faudra donc le définir de façon à pouvoir le déplacer facilement, sans faire de gros calculs. Nous voyons alors apparaître la notion de définition relative d'une figure. On prévoira donc deux définitions de figures :

- figure en position absolue, c'est-à-dire que les coordonnées des points qui la définissent sont données par rapport à un repère fixe. Ce sera par exemple le système de coordonnées propre au terminal. Une telle figure ne pourra donc pas être déplacée, sauf si l'on modifie toutes les coordonnées des points la définissant.

- figure en position relative, c'est-à-dire que les coordonnées des points qui la définissent sont données par rapport à un système de coordonnées mobile. Pour déplacer une telle figure, il suffira alors de changer la base du système de coordonnées, sans modifier la figure elle-même.

Même avec la possibilité de définition relative, le problème n'est pas résolu de façon satisfaisante, aucun gain de place n'ayant été obtenu. L'idée est alors de définir une figure sous forme de sous-programme. L'utilisation de cette figure se résumera à un simple appel de sous-programme précédé du changement de système de coordonnées qui permettra d'afficher la figure en bonne place sur l'écran. On voit que l'on gagne beaucoup de place, au détriment de la vitesse d'exécution.

Chaque figure pouvant être employée en tant qu'ensemble indépendant, il faut qu'elle comporte en elle-même tous les renseignements concernant la luminosité, la densité de points de l'image à éclairer. Elle devra aussi contenir une indication donnant le droit ou non d'utiliser les dispositifs de localisation à son encontre. En effet, une

image peut être formée de figures très diverses, certaines de ces figures étant destinées à rester en permanence sur l'écran, d'autres pouvant au contraire être effacées à l'aide du crayon optique ou tout autre dispositif identique. Il est donc indispensable de protéger la première catégorie de figures, de façon à ne pas les détruire au cours d'une fausse manoeuvre. Le fait de donner tous ces renseignements à l'intérieur d'une figure permet d'automatiser le traitement d'une image, toute figure copiée sur l'écran étant en tout point semblable à l'original.

4) Manipulation de figure.

Nous appellerons manipulation de figures toute modification d'un fichier graphique autre que l'effacement ou l'allumage d'objets. On peut classer ces manipulations en deux grands groupes : les translations et les déformations.

* translation de figures.

Qu'il s'agisse d'amener une figure d'un endroit de l'écran à un autre, ou de lui faire subir des translations en mouvement continu (animation d'images), le problème est facilement résolu dans le cas où l'on s'est donné la possibilité de définir des figures en position relative. Il suffit alors de modifier les systèmes de référence de chaque figure destinée à être déplacée pour obtenir le résultat voulu.

Dans le cas où les images ne sont pas définies de façon relative, il faut recalculer tous les points qui les forment, opération très coûteuse.

* déformation de figures.

Nous incluons sous ce terme toutes les transformations telles que similitudes planes, symétries planes, vues perspectives de figures ayant subi des rotations, c'est-à-dire toutes les modifications que l'on fait

subir à une figure dans sa forme à la suite d'un certain nombre d'opérations. Ce problème ne peut en aucun cas être résolu facilement. La solution de celui-ci dépend à la fois de la structure des images, donc du langage employé pour les définir, de la puissance de calcul de l'ordinateur utilisé et de la généralité du programme de dessin que l'on se propose d'écrire. Il semble qu'un bon moyen, dans le cas où l'on veut un programme de dessin très général, soit d'utiliser une représentation matricielle, les opérations sur des matrices étant relativement faciles à exécuter. Dans tous les cas, il faut se dire qu'il y aura de graves problèmes à résoudre (cas des parties cachées lors d'une rotation en vue perspective), et que le temps de calcul sera toujours très long. En fait, les problèmes de déformation d'images sont plus des problèmes de géométrie analytique que des problèmes proprement graphiques.

5) Utilisation du crayon optique.

Nous nous contenterons de décrire en détail les procédés d'utilisation du crayon optique, à l'exclusion de tous les autres dispositifs de localisation, car c'est certainement le dispositif le plus répandu et le plus populaire. Il est utilisé pour plusieurs tâches :

identifier une figure sur l'écran
indiquer une position de l'écran
dessiner des figures.

Nous allons passer en revue ces différentes utilisations.

* identification de figures.

Ce vocable recouvre des actions très diverses, telles que déplacer une figure, l'effacer de l'écran, ou encore indiquer une action à effectuer. Le facteur commun de ces actions est : le crayon optique ayant vu un point lumineux sur l'écran, identifier quelle est la figure désignée. Ceci peut être plus ou moins facile suivant le procédé qui

a été employé pour définir une figure. Mais on dispose toujours d'un certain nombre de renseignements (index dans une table, par exemple) qui facilitent la recherche.

Une utilisation courante de cette facilité d'identification est de donner un menu, c'est-à-dire une suite d'opérations à effectuer, sur l'écran lui-même. Le menu se présente comme une suite de symboles affichés généralement dans une zone fixe de l'écran. Lorsque l'on montre un de ces symboles, on fait appel à un sous-programme capable d'effectuer une tâche précise ; on peut imaginer par exemple de construire un réseau électrique en donnant dans le menu tous les symboles dont on dispose pour le dessin, et en montrant d'abord le symbole à utiliser puis la place qu'il doit occuper pour l'amener à l'endroit voulu. Ce procédé du menu permet d'éviter de recourir au clavier alphanumérique pour lancer une commande, et il est donc beaucoup plus rapide. Mais il a l'inconvénient de mobiliser en quasi permanence une partie de l'écran qui peut devenir rapidement importante. Ce procédé pose aussi le problème d'éviter les débordements de l'image sur la partie réservée au menu, problème qui n'est pas toujours simple à résoudre.

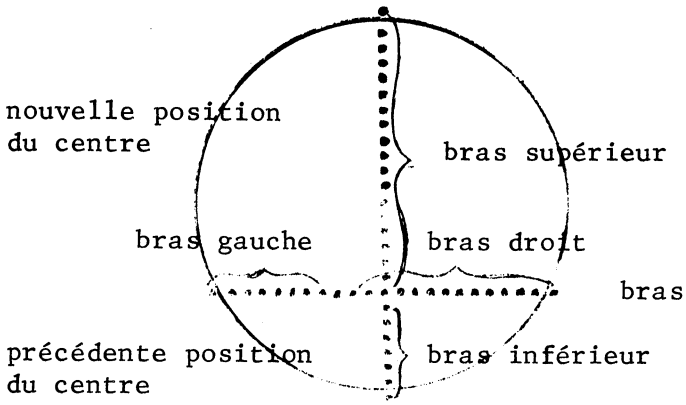
* techniques de poursuite du crayon optique.

Pour pouvoir désigner un point non allumé de l'écran, on a l'habitude de faire apparaître un symbole qui, lorsqu'il est montré avec le crayon optique, peut se déplacer et suivre le mouvement de la main qui tient le crayon. Les techniques mises en oeuvre ont toutes pour but de permettre le déplacement du symbole choisi avec le maximum de rapidité d'une part et le minimum de temps de calcul d'autre part.

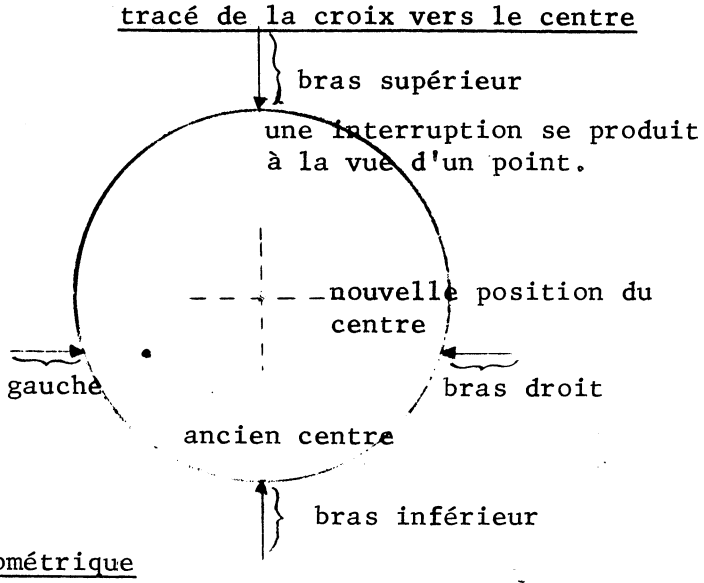
Un symbole employé couramment est une croix, que l'on utilise pour déterminer les coordonnées du centre l'ouverture du crayon optique. Au départ la croix est affichée, centrée généralement sur un point donné de l'écran. Dès qu'un point d'un des bras de la croix a été vu par le

TECHNIQUES DE POURSUITE DU CRAYON OPTIQUE

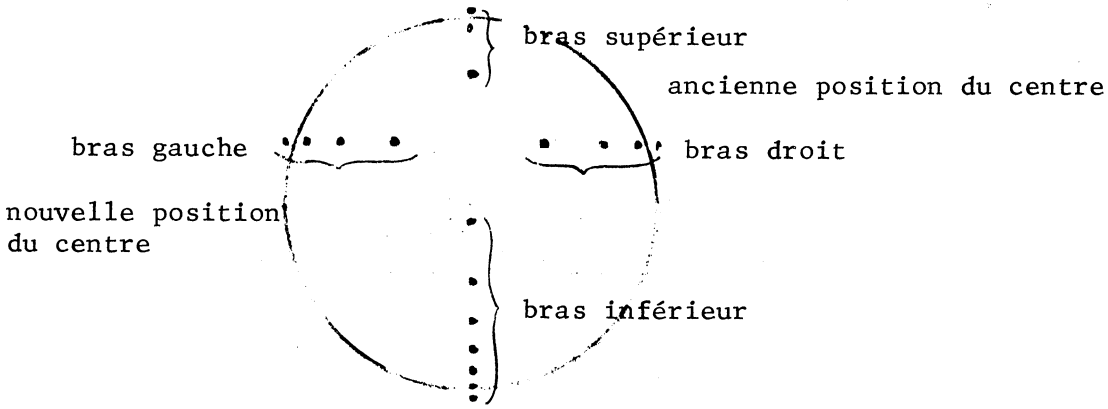
tracé de la croix à partir du centre



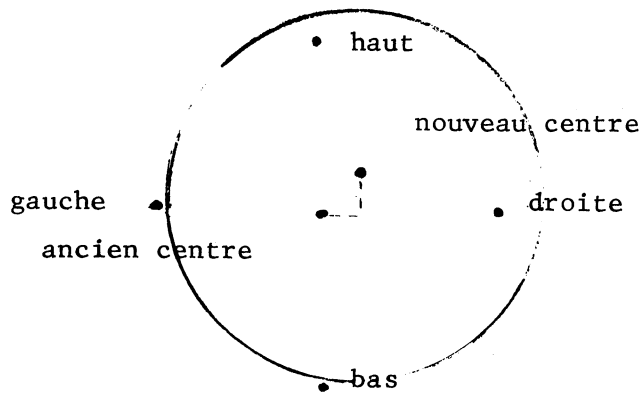
tracé de la croix vers le centre



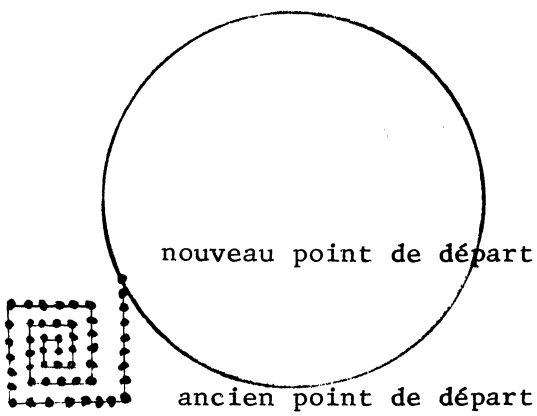
poursuite géométrique



croix simplifiée



poursuite en spirale



crayon optique, on continue à générer ce bras jusqu'à trouver l'autre bord du crayon optique. Ceci est très simple, car dès que le bras de la croix que l'on est en train de générer sort du champ de vision du crayon, il ne se produit plus d'interruption. On calcule alors les coordonnées du milieu du segment ainsi obtenu, et on génère le segment perpendiculaire en ce point, de longueur égale au diamètre de l'ouverture du crayon, toujours en cherchant les deux extrémités qui sortent du champ de vision du crayon. On affiche alors la croix au point milieu de ce segment, ce qui correspond bien au centre de l'ouverture du crayon. Le processus se poursuit tant que l'on montre la croix avec le crayon optique. Dès que le crayon optique ne montre plus la croix, celle-ci s'arrête et reste affichée sur le dernier point "vu", en fait centrée sur le dernier point calculé. La figure située en haut à gauche du schéma V donne un exemple d'utilisation de la croix partant du même principe que celui que nous venons de décrire, mais employant une méthode légèrement différente. On considère en effet que c'est le centre de la croix qui est vu. On génère alors les bras de la croix de la même façon que précédemment pour déterminer le centre de la croix. L'avantage de cette méthode est qu'elle demande l'affichage de moins de points, puisqu'on ne dépasse jamais les limites du champ de vision du crayon optique, alors que la précédente méthode demande l'affichage de points en dehors du champ de vision. Une variante de cette méthode consiste à tracer les bras de la croix non pas en partant du centre mais en allant vers le centre. Cette méthode est employée pour éviter les erreurs qui peuvent se produire lorsqu'une des branches de la croix sort des limites de l'écran.

Les autres méthodes affichent les points dessinant la croix de façon à gagner le plus de temps possible. La première méthode consiste à afficher ces points non plus à intervalles réguliers, mais suivant une progression géométrique. Le premier point de la branche est affiché à environ mi-chemin entre le centre de la croix et un point situé à une distance égale au rayon de l'ouverture du crayon optique. Si ce point est détecté, le point suivant est affiché, mais cette fois à une distance moitié de la précédente, et ainsi de suite jusqu'à ce que le dernier point affiché ne soit pas vu par le crayon optique. La seconde méthode consis-

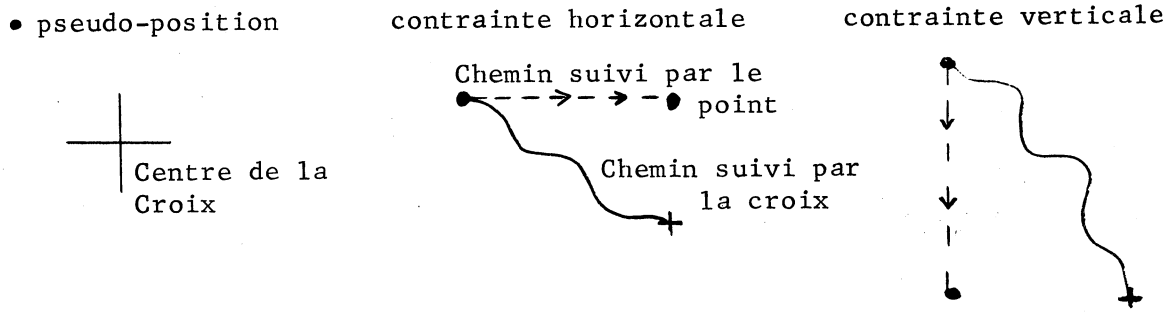
te à n'afficher que le centre et quatre points. Si l'un de ces quatre points se trouve à l'intérieur du champ de vision du crayon optique, le centre est déplacé dans la direction de ce point d'une petite valeur.

Toutes ces méthodes qui utilisent une croix supposent que le crayon optique ne sera pas déplacé assez vite pour que l'un des bras de la croix n'ait pas le temps d'être dessiné, ce qui ferait perdre de vue de crayon optique. Une autre méthode employée est la poursuite en spirale. Si le dernier point vu par le crayon optique n'est pas à nouveau désigné lors de son réaffichage, on affiche une série de points en spirale jusqu'à ce que l'un d'eux soit vu par le crayon optique. Ce point devient alors le nouveau centre. l'erreur commise en employant cette méthode est de l'ordre du rayon de l'ouverture du crayon optique.

* pseudo-position du crayon optique.

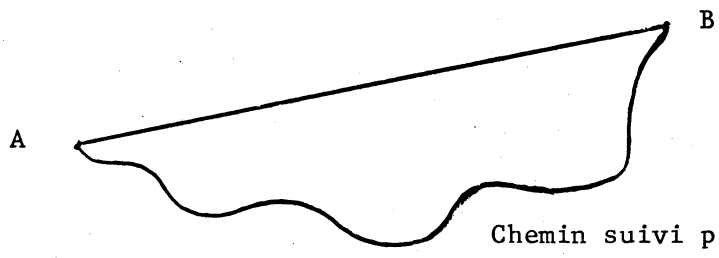
On associe habituellement à la croix un point qui peut être le centre de la croix ou un autre point situé à une distance et dans une direction fixes par rapport au centre de la croix. L'intérêt de la chose est qu'on peut imposer à ce point des contraintes lors du déplacement de la croix. Par exemple, il ne pourra se mouvoir que verticalement, quel que soit le mouvement du crayon optique ; ceci peut être utile pour obtenir des projections verticales ou horizontales, ou suivant une direction donnée à l'avance.

Le schéma ci-dessous donne quelques exemples d'utilisation de la pseudo-position



* dessin à l'aide du crayon optique

On peut avoir l'idée de dessiner à l'aide du crayon optique. Le peu de précision de ce dispositif fait qu'on utilise en général des sous-programmes spéciaux qui utilisent le crayon optique comme moyen d'introduire des données. Par exemple, pour tracer un vecteur avec le crayon optique, on amènera la croix au point de départ du vecteur, on indiquera qu'il s'agit du point de départ à l'aide d'un bouton de commande (par exemple), puis on déplacera la croix en suivant un chemin quelconque jusqu'au point d'arrivée du vecteur. On indiquera de même qu'il s'agit là de l'extrémité du vecteur, et c'est le sous-programme qui se chargera de générer le vecteur, muni de ces renseignements.



Chemin suivi par la croix

Il est tout à fait illusoire de penser pouvoir dessiner à main levée sur un écran sans avoir un programme chargé d'obtenir les courbes désirées, programme qui se servira du crayon optique pour connaître certains paramètres. L'imprécision du crayon optique et la maladresse de l'utilisateur conduisent en effet, dans le cas où le terminal se contente d'afficher les points indiqués par le crayon sans apporter de corrections, à obtenir des dessins sans formes bien définies et aux contours plutôt tremblés.

6) Clavier de fonctions.

Il s'agit d'un auxilliaire très précieux quand on en dispose. Ce dispositif se présente généralement sous la forme d'une boîte avec un certain nombre de boutons. Lorsque l'on appuie sur l'un de ces boutons, il se produit une interruption et on trouve dans un registre spécialisé le numéro du bouton sur lequel on a appuyé. On peut alors se renvoyer par programme à une séquence de traitement de cette interruption. On associera

à chaque bouton une fonction très précise, visant à prévenir le système de dessin de l'action à entreprendre. Cette action peut être EFFACER la figure qui va être désignée, ou faire apparaître la croix, etc... On peut laisser libre cours à son imagination et programmer ainsi de façon commode toutes les fonctions générales utilisées pour dessiner. Un autre intérêt de ce dispositif est sa souplesse d'emploi, puisqu'il suffit de changer les sous-programmes correspondant aux boutons pour obtenir un nouvel emploi du clavier.

7) Fenêtres.

L'écran est souvent découpé par programme en un certain nombre de secteurs, chaque secteur correspondant à un fichier graphique. Un de ces fichiers pourra être par exemple la représentation de l'ensemble des commandes pouvant être utilisées dans un menu. Si le nombre de ces commandes est trop important, on ne pourra pas toutes les afficher sur l'écran. Il faudra donc avoir un moyen de parcourir le fichier graphique pour avoir accès à toutes les commandes, tout en n'affichant qu'un nombre réduit de celles-ci à un instant donné. C'est ce procédé qui est connu sous le nom de fenêtres. Une fenêtre va donc permettre de particulariser une certaine partie d'un fichier graphique. L'action à entreprendre sur cette partie de fichier peut être simple : éclairer tous les points contenus à l'intérieur de la fenêtre avec une intensité très forte, ou très compliquée : représenter les éléments contenus à l'intérieur d'une fenêtre à une échelle supérieure, tout en restant dans les limites de la fenêtre. Cette dernière application s'apparente à l'utilisation d'une loupe qu'on promènerait sur le dessin.

B - UN LANGAGE GRAPHIQUE : LAGROL

Disposant du matériel décrit au chapitre I, nous avons été amenés à définir un langage graphique pour plusieurs raisons. La première, c'est que le terminal dont nous disposons ne peut afficher que des points. Il est alors fastidieux d'écrire des programmes de dessin un tant soit peu compliqués. La deuxième raison est que pour pouvoir utiliser facilement les possibilités de simultanéité du PDP-8, il est intéressant d'avoir à sa disposition un programme de traitement automatique des interruptions. La dernière raison est pédagogique. En effet, des séances de travaux pratiques pour des étudiants de la Maîtrise d'Informatique avaient été prévues. Il était indispensable de donner aux étudiants un moyen commode d'utiliser le terminal graphique, afin qu'ils ne soient pas rebutés par des difficultés de programmation inhérentes au terminal. Nous avons donc été amenés à écrire un programme simulant les dispositifs n'existant pas sur ce terminal, à savoir les générateurs de vecteurs, caractères et cercles. D'autres possibilités ont été données. L'ensemble des instructions permettant de dessiner à l'aide de ces dispositifs simulés, complété par les instructions du langage d'assemblage du PDP-8, forme le langage LAGROL.

1) Généralités sur LAGROL.

Dans la construction du langage, un certain nombre de règles ont été suivies, qu'il sera bon d'avoir à l'esprit lors de l'écriture de tout programme de dessin en LAGROL.

* Lors de l'éxécution, on trouvera à tout instant dans une mémoire appelé GCO l'adresse de l'instruction en cours + 1. GCO est donc le compteur ordinal pour l'exécution d'un programme.

* On trouvera dans GRT le mot du langage qui est en cours de traitement.

* On trouvera dans GABS et GORD les coordonnées du dernier point calculé. Ce n'est pas forcément le dernier point allumé.

* Lorsqu'on parlera d'échelle, ce mot aura une signification légèrement différente du sens ordinaire.

- dans le cas de vecteurs ou de cercles, parler d'échelle 6 signifiera que l'on va allumer un point sur 6 calculés pour définir le vecteur ou le cercle.

- dans le cas de déplacements de points, utiliser l'échelle 6 signifie que l'on désire déplacer le point de 6 unités-écran.

- dans le cas du tracé de caractères, l'échelle aura le sens ordinaire du mot. Un caractère tracé à l'échelle 6 sera 6 fois plus grand qu'un caractère tracé à l'échelle 1.

- dans le cas du tracé de points, le mot échelle n'a pas de sens.
  Le mot échelle recouvre donc deux significations : dans le cas des vecteurs et des cercles, il s'agit d'une pseudo-densité, dans les autres cas, il s'agit d'une échelle au sens ordinaire du mot.

* pour définir un point, on donnera obligatoirement ses deux coordonnées, dans l'ordre : abscisse puis ordonnée.

* lorsque l'on parlera de chaînage, cela voudra dire que derrière une instruction on pourra donner autant de coordonnées que l'on voudra, étant sous-entendu qu'on exécutera toujours la même instruction, en l'appliquant successivement à chacun des couples de coordonnées.

* Lorsque l'on parlera d'option, on indiquera par ce mot la possibilité de définir une certaine qualité de l'objet décrit par une instruction.

* les mots du langage sont divisés en deux catégories :

- les instructions formées de codes d'opération, complétés éventuellement d'options.

- les opérandes formés de codes de définition complétés de données.

* Lorsqu'on parlera d'option standard, cela voudra dire que l'absence de certains codes dans une instruction aura une signification précise, et que ces codes pourront être omis.

* Toute option non précisée sera standard.

* Tout symbole du langage commence par la lettre G, sauf JMP et JMS.

* Toutes les possibilités de l'assembleur PDP sur IBM 7044 (en particulier les pseudo-instructions) peuvent être utilisées.

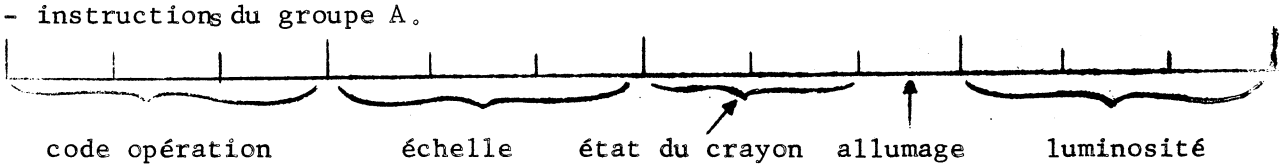
2) Les instructions.

LAGROL est donc une extension du langage d'assemblage PDP-8. Cette extension est réalisée à l'aide de symboles dont la liste suit. Pour une meilleu compréhension, le lecteur aura intérêt à se reporter souvent aux exemples donnés en Annexe A, Annexe B et Annexe C.

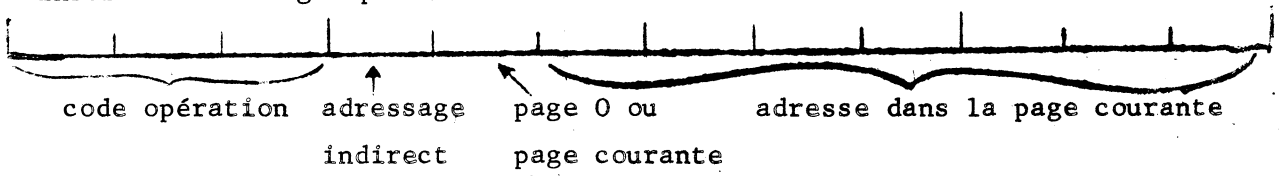
* Structure des instructions en machine.

Il y a trois groupes d'instructions. Elles diffèrent par leur représentation dans la mémoire du PDP-8.

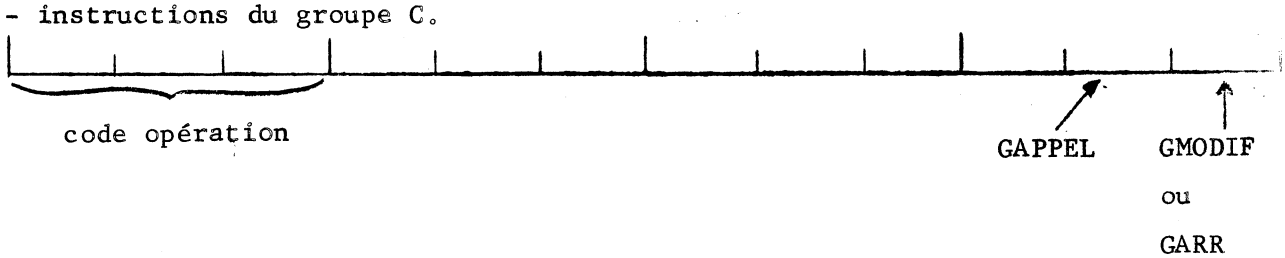
- instructions du groupe A.



- instructions du groupe B.



- instructions du groupe C.



* Code opération 0.

- GPOIN groupe A toutes options; échelle sans signification.

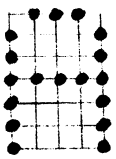
Cette instruction permet d'allumer (suivant l'option) le point dont les coordonnées sont données à la suite. Il y a possibilité de chaînage, c'est-à-dire que l'on pourra allumer n points en n'utilisant qu'une seule instruction. Il suffira de faire suivre cette instruction des n couples de coordonnées. Pour mettre dans les registres GABS et GORD les coordonnées d'un point particulier qu'on ne désire pas voir allumer (par exemple point de départ d'une chaîne alphanumérique), on utilisera l'instruction GPOIN avec l'option 'éteint' (voir page 42)

Structure d'un opérande : code de définition, adresse

* Code opération 1.

- GCAR groupe A toutes options.

Cette instruction permet d'afficher une suite de caractères, l'adresse de départ du dessin étant dans GABS et GORD. Ce point est le coin inférieur gauche du premier caractère. Les autres caractères sont affichés à la suite. Dans le cas de débordement de l'écran, il y a passage automatique à la ligne suivante. La chaîne à produire doit être assemblée à l'aide d'une pseudo-instruction TEXT située à la suite de l'instruction GCAR. Le balayage de la chaîne est arrêté par la rencontre du premier demi-mot nul. Si l'on veut générer cette chaîne par programme, les caractères devront être codés en code ASCII sur 6 positions binaires, à raison de deux caractères par mot. Tous les caractères du code ASCII peuvent être utilisés, sauf @ qui a pour effet d'arrêter l'affichage (code nul), et \ qui fait passer au début de la ligne suivante. Les caractères sont affichés sur l'écran à l'aide d'une grille de 7 × 5 points, grille qui est représentée en mémoire de la façon suivante : chaque caractère est représenté à l'aide de trois mots PDP. Les cinq premières positions binaires du premier mot représentent le haut du caractère, les cinq positions suivantes la ligne en dessous et ainsi de suite. On obtient ainsi une représentation facile à traiter et demandant un minimum de place. L'exemple ci-dessous donne la représentation du caractère A sur l'écran et en mémoire.



0 1 1 1 0 1 0 0 0 1 1 0 premier mot
0 0 1 1 1 1 1 1 1 0 0 0 deuxième mot
1 1 0 0 0 1 1 0 0 0 1 X troisième mot

position inutilisée

* Code opération 2.

- GVEC groupe A toutes options.

Cette instruction permet d'obtenir un vecteur. Les coordonnées du point de départ se trouvent dans GABS et GORD, les coordonnées du point d'arrivée sont données à la suite de l'instruction. Il y a possibilité de chaînage. En effet, dans le cas d'un dessin en trait continu (triangle, polygone, ligne brisée etc...), lorsqu'on a fini de tracer un des composants de la figure, le point d'arrivée devient le point de départ du composant suivant, et ses coordonnées sont déjà dans GABS et GORD. Il suffit donc de mettre derrière l'instruction GVEC les n coordonnées des n points d'arrivée successifs pour tracer n vecteurs consécutifs.

Structure d'un opérande : code de définition, adresse

Le tracé de vecteurs nous a posé des problèmes délicats. La première idée qui vient à l'esprit est d'employer les formules suivantes

$$x_n = x_{n-1} + \Delta_x$$

$$y_n = y_{n-1} + \Delta_y$$

Le problème est de choisir Δ_x et Δ_y de façon à obtenir une ligne qui soit la plus droite possible. Nous avons en fait employé une méthode dérivée de celle-ci Nous employons deux procédés de calcul, suivant que la pente de la droite à trac est supérieure ou inférieure à 1. L'idée de base est de faire progresser en x ou en y de une unité et de calculer le y ou le x correspondant. Pour une pente inférieure à 1, nous faisons progresser les x, pour une pente supérieure à 1 nous faisons progresser les y. Ceci est fait de façon à obtenir toujours un grand nombre de points pour dessiner le vecteur. Nous employons donc les formule

pente supérieure à 1 : $x_n = x_{n-1} + Cte$ $\quad y_n = y_{n-1} + 1$

pente inférieure à 1 : $x_n = x_{n-1} + 1$ $\quad y_n = y_{n-1} + Cte$

On peut alors remarquer que la constante étant toujours inférieure à 1, si on fait les calculs en entier, on n'a plus de division à effectuer. Il suffit de trouver un procédé de correction du tracé qui permette de tenir compte des erreurs d'arrondi qui se font en faisant des additions en flottant. Le procédé employé est très simple : après avoir déterminé le plus grand déplacement en x ou en y, par exemple Δ_y, on ajoute à chaque pas dans un compteur spécial la valeur de Δ_y, on corrige la valeur du y soit en ajoutant 1, soit en retranchant 1, suivant le sens de tracé du vecteur. L'avantage de cette méthode sur la méthode habituelle de correction des arrondis est qu'on tombe toujours sur le point d'arrivée, ce qui n'est pas toujours vrai avec la méthode ordinaire. Par exemple, on désire tracer le vecteur (0-0,10-13).

La progression de 1 en 1 se fera sur l'axe des y.

	méthode utilisée		méthode des arrondis	
y	cumul des Δ_y = 13 n	x retenu	x retenu	x calculé Δ_x = 0,77
0	0	0	0	0
1	10	0	1	0,77
2	7	1	2	1,54
3	4	2	2	2,31
4	1	3	3	3,88
5	11	3	4	3,85
6	8	4	5	4,62
7	5	5	5	5,39
8	2	6	6	6,16
9	12	6	7	6,93
10	9	7	8	7,70
11	6	8	8	8,47
12	3	9	9	9,24
13	0	10	10	10,1

On voit que les résultats obtenus par les deux méthodes sont tout à fait comparables. La première méthode est très avantageuse lorsque le calculateur utilisé ne possède pas d'élément arithmétique, ou ne permet pas de faire des calculs en flottant d'une manière simple. Les résultats obtenus sur l'écran sont tout à fait satisfaisants.

* Code opération 3.

- GDEP Groupe A toutes options.

Cette instruction permet de déplacer le point dont les coordonnées sont dans GABS et GORD dans la (ou les) direction précisée dans les opérandes suivant l'instruction. Il y a possibilité de chaînage. Une description plus complète de la façon d'utiliser cette instruction est donnée page 46.

Structure d'un opérande : code de définition, codes de déplacement.

* code opération 4.

- JMS Groupe B aucune option.

Rupture de séquence vers un sous-programme, avec conservation de GCO dans la première mémoire du sous-programme. Le contrôle est donné à l'instruction qui suit la première mémoire du sous-programme.

```
200     JMS SOUPRO          SOUPRO 0      à l'exécution,
201                           GPOIN       SOUPRO contiendra
                                          la valeur 201
        Pas d'opérande.
```

* Code opération 5.

- JMP Groupe B aucune option.

Rupture de séquence inconditionnelle. Pas d'opérande.

- GNOP Groupe B aucune option.

C'est la non-opération. En fait, c'est l'instruction JMP I GCO (saut inconditionnel à l'adresse contenue dans GCO), ce qui fait qu'elle appartient au groupe B des instructions.

Pas d'opérande.

* Code opération 6.

- GARR groupe C aucune option.

Le code complet de cette instruction est 6000. Pour comprendre son fonctionnement, il faut savoir que le sous-programme interprète est conçu comme un sous-programme, qui peut être appelé plusieurs fois. A chaque fois, l'adresse de l'appel est conservée dans la première mémoire de l'interprète. L'instruction GARR est l'instruction de sortie du sous-programme interprète. Le contrôle est donné à l'instruction qui suit le dernier point d'appel de l'interprète.

Pas d'opérande.

- GMODIF groupe C aucune option.

Le code complet de cette instruction est 6001. Cette instruction permet de changer le système de référence d'une figure. Les coordonnées définies de manière relative sont modifiées par addition du contenu de la mémoire GRELX pour les abscisses, GRELY pour les ordonnées. L'in truction GMODIF permet de modifier le contenu de ces deux mémoires de la façon suivante:

si l'on écrit GMODIF
GA,200 le contenu de GRELX devient 200
GAF,100 le contenu de GRELY devient -100

Nous parlerons de modification absolue.

si l'on écrit GMODIF
GR,100 le contenu de GRELX est augmenté de 100
GRF,400 le contenu de GRELY est diminué de 400

Nous parlerons de modification relative.

Les codes de définition ont donc une signification précise, permettant de modifier à volonté la base de référence. C'est grâce à cette instruction que l'on peut envisager l'emploi d'une même figure plusieurs fois dans une image. Il suffit de faire précéder l'appel de la figure par une instruction GMODIF.

- GAPPEL groupe C aucune option.

Le code complet de cette instruction est 6002. Cette instruction possède au moins un opérande, qui doit être une adresse. Le contrôle est alors donné à la séquence de programme débutant à cette adresse. Il s'agit d'un simple saut,

pas d'un appel de sous-programme. Cette séquence de programme doit être écrite en suivant un certain nombre de règles qui sont exposées en détail au chapitre V, page 94 . Un exemple d'utilisation très complet est donné dans le même paragraphe.

* Code opération 7.

- GCERCL groupe A toutes options.

Cette instruction permet de dessiner des arcs de cercle. Les règles à suivre sont :

- les coordonnées du centre sont dans GABS et GORD.
- le premier opérande est le rayon du cercle, en unités-écran.
- les deux opérandes suivants sont les abcisses des points de départ et d'arrivée. Pas d'ordonnées.
- on dessine toujours dans le sens trigonométrique.

Pour définir sans ambiguité l'arc à dessiner, une convention est donnée page 44.

Il n'y a pas de chaînage possible.

Structure des opérandes : code de définition, adresse

Le tracé des cercles est fait à l'aide des formules suivantes :

$$x_n = x_{n-1} + 1/R\,(y_{n-1} - y_c)$$

$$y_n = y_{n-1} - 1/R\,(x_{n-1} - x_c)$$

où x_c et y_c sont les coordonnées du centre. Le résultat obtenu est très acceptabl

* Les options.

Il y en a quatre sortes.

- luminosité.

Elle peut aller de 0(pratiquement invisible) à 7 (très forte). Il suffit de mettre un chiffre compris entre 0 et 7 dans la liste des options.

Option standard : luminosité 0.

- allumage des points.

Il y a deux options, de façon à permettre de calculer les points formant une figure sans éclairer celle-ci.

GETEIN on exécute l'instruction sans allumer les points calculés.

GALUM on allume les points calculés, en tenant compte éventuellement de l'échelle.

Option standard : GALUM.

- crayon optique.

Lorsqu'on a une image sur l'écran, on peut avoir intérêt à interdire l'emploi du crayon optique lors du tracé de certaines figures. On utilisera pour ce faire les options suivantes :

GCLEN crayon optique en fonction

GCLOR crayon optique hors fonction

Option standard : l'état du crayon est laissé tel qu'il était à l'instruction précédente.

- échelle

Il y a 8 codes possibles, allant de

GECH1 on allume tous les points calculés, l'échelle des caractères est minimale, les déplacements se font sur une unité-écran.

jusqu'à

GECH8 on allume un point sur huit pour les vecteurs et les cercles, l'échelle des caractères est maximale, les déplacements sont de huit unités-écran.

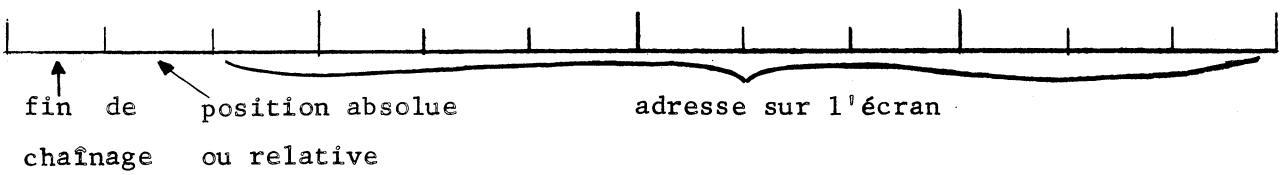
Cette option est ignorée dans le cas des points, puisqu'elle n'a alors pas de signification.

Option standard : GECH1.

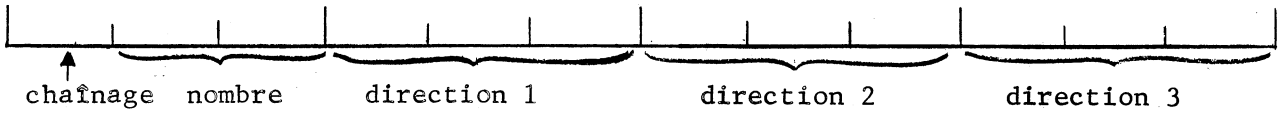
3) Les opérandes.

* structure des opérandes.

- opérande de coordonnées



- Opérande de déplacement.



* Codes de définition.

Il y en a deux groupes : les codes de coordonnées
les codes de déplacement

Ces codes de définition indiquent la nature de l'opérande. Ils servent aussi à indiquer si l'on est en cours ou en fin de chaînage. Les codes de définition sont groupés par couples, ne différant l'un de l'autre que par la lettre F en fin de code. Cette lettre signifie que l'opérande en question est le dernier de l'instruction, sauf dans le cas de GMODIF et de GCERCL.

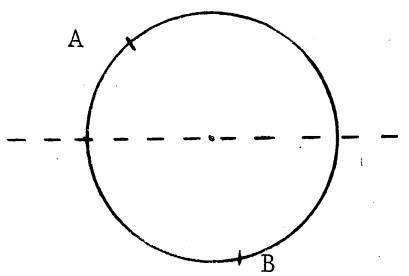
- codes de coordonnées.

Il y en a quatre, soient deux d'adressage absolu
deux d'adressage relatif.

Dans le cas de l'adressage relatif, lors du calcul de la valeur exacte de l'adresse contenue dans l'opérande, on ajoute à celle-ci la valeur de GRELX si c'est une abscisse, de GRELY si c'est une ordonnée. Tous les problèmes de déplacement et animation d'une figure sont donc très facilement résolus : Il suffit de modifier le contenu de ces deux mémoires pour obtenir le résultat désiré.

Les codes sont :

GA	adresse absolue
GAF	adresse absolue, fin de chaînage
GR	adresse relative
GRF	adresse relative, fin de chaînage



Dans le cas de GCERCL, la lettre F a une autre signification : c'est elle qui indique que le point désigné par son abscisse se trouve sur l'arc inférieur du cercle.

A sera défini par GA (ou GR), abscisse de A
B sera défini par GAF (ou GRF), abscisse de B

Dans le cas de GMODIF, la lettre F veut dire que la quantité qui suit est négative.

GA,100	signifie +100
GAF,100	signifie -100

- Les codes de déplacement.

Il y en a quatre couples.

GO	0 déplacement
GOF	0 déplacement , fin de chaînage
G1	1 déplacement
G1F	1 déplacement, fin de chaînage
G2	2 déplacements
G2F	2 déplacements, fin de chaînage
G3	3 déplacements
G3F	3 déplacements, fin de chaînage

L'intérêt des codes GO et GOF apparaît dans le cas où l'on veut supprimer un ou plusieurs déplacements et que ceux-ci se trouvent dans un chaînage.

Il suffit alors de les remplacer par GO ou GOF pour que le chaînage se déroule correctement.

Pour les directions de déplacement, voir dans les données.

* Codes de données.

Il y a deux groupes de données les adresses
les directions de déplacement.

La nature d'un code de donnée est spécifiée par le code de définition.

- les adresses

Elles doivent être comprises entre 0 et 1777(octal), ou 0 et 1023 (décimal). Aucun contrôle n'est effectué quant aux bornes. On devra donc faire attention à ne pas sortir des limites de l'écran, sous peine de voir l'image apparaître de l'autre côté de celui-ci. Les adresses peuvent être symboliques. Elles ne peuvent suivre qu'un code de définition de coordonnées.

- les directions de déplacement.

Il y en a huit groupes de trois :

GE1	GE2	GE3	déplacement en direction de l'Est
GNE1	GNE2	GNE3	déplacement en direction du Nord-Est
GN1	GN2	GN3	déplacement en direction du Nord
GNO1	GNO2	GNO3	déplacement en direction du Nord-Ouest
GO1	GO2	GO3	déplacement en direction de l'Ouest
GSO1	GSO2	GSO3	déplacement en direction du Sud-Ouest
GS1	GS2	GS3	déplacement en direction du Sud
GSE1	GSE2	GSE3	déplacement en direction du Sud-Est

Nous avons été obligés d'utiliser trois codes différents pour désigner la même direction de déplacement à cause du processus d'assemblage utilisé sur IBM 7044. L'explication complète de ce point est donnée page 48.

Cette contrainte nous oblige à écrire :

un déplacement	G1,GN1	ou G1F,GN1	par exemple
deux déplacements	G2,GN1,GS2	ou G2F,GN1,GS2	
trois déplacements	G3,GN1,GS2,GNE3	ou G3F,GN1,GS2,GNE3	

On voit que l'on peut donner différentes directions de déplacement au sein du même opérande. Ces déplacements sont effectués en décodant l'instruction de gauche à droite.

Ces codes de directions de déplacement ne peuvent suivre que les codes de définition de déplacement. Ici encore, aucun contrôle n'est possible. Le fait de mélanger des codes de définition de coordonnées avec des directions de déplacement peut donner des résultats assez remarquables et à coup sûr inattendus.

La liste des instructions et opérandes est donnée en annexe.

4) Mise en oeuvre et exécution de programmes écrits en LAGROL.

Dans ce paragraphe, nous parlerons d'abord de l'écriture et de l'assemblage des programmes de dessin. Nous parlerons ensuite de leur exécution,

soit en mode IOF, soit en mode ION. Nous terminerons en donnant des exemples de programme.

* Ecriture des programmes.

Les programmes d'affichage de dessins sont composés en général de parties écrites en langage symbolique LAGROL et de parties écrites en langage machine. Ces parties sont d'ailleurs vite prépondérantes dès qu'il s'agit d'écrire un programme de dessin relativement complexe.

En ce qui concerne les parties de programme écrites en langage machine, on respectera les conventions exposées dans le manuel en référence 4-1 .

En ce qui concerne les parties écrites en LAGROL, les seules conventions sont les suivantes :

- écriture des instructions et des opérandes à partir de la colonne 8.
- les codes formant un mot doivent être séparés par des virgules, sauf dans le cas de JMP ou de JMS. Dans ce dernier cas, l'adresse doit être séparée du code opération par un blanc. Dans le cas où les codes sont séparés par dès virgules, leur ordre d'apparition est sans signification.

L'appel de l'interprète se fait de la façon suivante :

```
TAD (IMAGE      IMAGE est le nom de la première instruction
                du programme de dessin
LAGROL          cette instruction appelle l'interprète
```

En raison desfacilités accordées pour le travail en mode ION (voir page 50), on ne pourra pas utiliser la page 0 pour l'écriture de programmes. On dispose des pages 1 à 20 incluses pour ce faire. L'interprète peut être facilement modifié pour permettre l'utilisation du champ I ou du champ II de la machine et laisser ainsi plus de place.

Toutes les facilités de l'assembleur peuvent être utilisées, toutes les contraintes doivent être respectées...

* Assemblage des programmes.

Pour des raisons de commodité et de rapidité, l'assemblage des programmes se fait sur IBM 7044. Pour celà, on fait appel à l'assembleur cité dans le paragraphe précédent, qui génère le code binaire correspondant à la codification du PDP-8. Lorsque l'assemblage s'est bien terminé, l'assembleur demande une communication avec le PDP-8. Le programme est alors envoyé à sa place dans la mémoire du PDP-8, et est chargé, généralement, sur bande magnétique.

Pour obtenir un assemblage correct, il faudra ajouter au programme proprement dit :

- les cartes de contrôle habituelles
- un paquet de cartes donnant les équivalences en octal de tous les symboles utilisés dans l'écriture du programme en LAGROL.

Il ne faut pas oublier la carte END à la fin du paquet. Elle doit porter l'adresse de lancement du programme de dessin.

REMARQUE : Lorsque l'assembleur rencontre une série de codes séparés par des virgules, il fait l'union logique de ces codes afin de générer le mot correspondant. Ceci explique pourquoi les codes sont séparés par des virgules, et pourquoi il faut disposer de trois codes pour chaque direction de déplacement, chaque code correspondant à une place en mémoire.

Exemple : GPOIN,GETEIN,7 sera assemblé sous la forme 0017 (octal)
0000,0010,0007

G3F,GNE1,G02,GS3 sera assemblé sous la forme 7146
7000,0100,0040,0006

Ce mode de travail explique aussi pourquoi l'ordre des options séparées par des virgules n'a aucune importance. L'ordre employé dans les exemples donnés en annexe n'est qu'une commodité de lecture.

* Exécution des programmes en mode IOF.

Lorsque nous parlons de programmes écrits en LAGROL, nous entendons l'ensemble d'un programme de dessin comportant des parties écrites en langage machine et des parties écrites en LAGROL. Les parties écrites en LAGROL servent à générer l'image, celles en langage machine servent à la modifier (animation, déplacements, déformations) grâce à des commandes lancées à l'aide de la console ou du crayon optique.

Si l'on a choisi de travailler en mode IOF, il n'y a pas de simultanéité possible. Le lancement de commandes à partir de la console ou du crayon optique n'est alors plus possible, puisque chaque fois que l'on chercherait à exécuter une instruction d'entrées-sorties, le dessin s'éteindrait jusqu'à ce que la commande soit satisfaite.

Ce mode de travail n'est donc intéressant que dans la mesure où l'on veut afficher un dessin statique, ou dont l'animation est entièrement prévue à l'avance. Ce mode de travail sera employé rarement.

* Exécution des programmes en mode ION.

Le fait de disposer de la simultanéité est un avantage considérable. En effet, le programme n'est interrompu que lorsque une demande émanant d'un organe périphérique est prête à être traitée. Le dessin ne s'éteint donc plus.

- description détaillée du principe.

Pour travailler en mode ION, il est nécessaire de faire exécuter l'instruction ION. A partir de ce moment, on peut travailler en simultanéité avec des opérations de transfert de données. A chaque registre de transfert est attaché un indicateur, qui a habituellement la valeur 0. Tous les indicateurs sont reliés en série à une seule sortie. L'indicateur attaché au registre considéré prend la valeur 1, lorsqu'une demande est prête à être satisfaite. La sortie commune prend alors la valeur 1. Lors du prochain test de cette sortie, il y a déroutement. Cette sortie est testée entre chaque exécution d'une instruction.

Lorsqu'il y a déroutement

le programme en cours est interrompu

le compteur ordinal est rangé dans la mémoire 0 du champ 0
le contrôle est donné à l'instruction contenue en mémoire 1 du champ 0
le PDP-8 passe alors en mode IOF

En général, l'instruction contenue en mémoire 1 renvoie à une séquence de traitement de l'interruption. Cette séquence doit

préserver l'accumulateur et le lien

trouver quel est l'organe qui a provoqué le déroutement

traiter ou ignorer cette demande

remettre à 0 l'indicateur concerné
restaurer l'accumulateur et le lien
redonner le contrôle au programme principal par la séquence suivante, (utilisée généralement)

```
ION              retour au mode ION
JMP  I  0        saut à l'adresse contenue en mémoire 0
```

On voit qu'il y a un grand nombre de précautions à prendre, ce qui fait la difficulté de ce mode de travail. Mais le gain de temps et la souplesse de travail obtenus compensent largement cet inconvénient.

- application à LAGROL.

Afin de faciliter le travail en mode ION, l'interprète LAGROL contient une séquence de traitement des interruptions. Les seules interruptions qui sont prises en considération sont celles qui viennent du crayon optique ou de la console. Le lien et l'accumulateur sont préservés, on cherche quel est l'indicateur allumé, et, suivant le résultat du test, on va exécuter le sous-programme dont l'adresse se trouve dans l'un des trois mots suivants :

```
GSPCRA    pour le traitement du crayon optique
GSPCLA    pour le traitement du clavier de la console (lecture de caractères)
GSPTEL    pour le traitement de l'imprimante de la console.
```

Les autres interruptions sont ignorées. A la sortie du sous-programme de traitement, les indicateurs concernés sont remis à 0, l'accumulateur et le lien sont restaurés, et la séquence de retour employée est

```
ION
JMP  I  0
```

L'utilisateur n'a donc qu'à écrire les sous-programmes de traitement des interruptions et donner le contrôle à ceux-ci en donnant leur adresse dans l'un des trois registres concernés. A l'entrée de ces sous-programmes de traitement, l'accumulateur contiendra

```
le contenu de GCO      s'il s'agit d'une interruption dûe au crayon optique
le caractère lu        s'il s'agit d'une interruption dûe au clavier de la
                       console
la valeur 0            s'il s'agit de l'imprimante
```

REMARQUES : Dans le cas de la lecture d'un caractère au clavier de la console le programmeur devra prendre soin de commander l'écriture en écho sur la console s'il veut voir écrit le caractère frappé. D'autre part, il est bon de prévoir un sous-programme permettant d'ignorer les interruptions venant de la console ou du crayon optique, dans le cas où l'on veut interdire l'usage de l'un ou l'autre de ces dispositifs. Ce sous-programme peut être

```
IGNORE   0
         CLA              remise à 0 de l'accumulateur
         JMP I IGNORE     Retour au programme de traitement
```

Il suffira alors de mettre l'adresse de ce sous-programme dans le ou les registres attachés aux dispositifs dont on veut interdire l'usage.

* Exemples de programmes.

Nous donnons deux exemples de programme, l'un en mode IOF, l'autre en mode ION.

- le premier programme fait apparaître une enveloppe vue de dos, puis vue de face, et ainsi de suite. L'enveloppe vue de dos est dessinée d'un seul trait (chaînage). L'enveloppe vue de face comporte une adresse (usage de la pseudo-instruction TEXT), un timbre, un cachet dessiné à l'aide de déplacements et trois lignes ondulées, dessinées à l'aide de trois appels d'un sous-programme, qui dessine une ligne ondulée à l'aide de points.

Les points intéressants de ce programme sont l'emploi des possibilités de chaînage, l'utilisation de l'instruction GMODIF et le changement d'image grâce à GARR.

- le deuxième programme commence par afficher le menu :

vecteur

texte

auto

Pour faire son choix il suffit de pointer son crayon optique vers l'un de ces trois mots. On est alors renvoyé à un sous-programme de dessin correspondant à la demande.

Si l'on montre "vecteur", on voit apparaître une croix et un point. En déplaçant la croix et en montrant deux fois le point, on obtient un vecteur entre les deux points définis par la croix.

Si l'on montre "texte", le texte frappé à la console est automatiquement affiché sur l'écran, en commençant par le coin supérieur gauche.

Si l'on montre "auto", une petite auto apparaît. Pour mettre celle-ci en mouvement, il suffit de frapper sur la console la lettre A si on désire la faire avancer, la lettre R si on désire la faire reculer. Si on tape plusieurs fois la même lettre, la voiture accélère dans la direction indiquée. Par des combinaisons des deux commandes, on obtient donc des mouvements et des vitesses variées.

L'intérêt de ce deuxième programme est de montrer en détail comment utiliser les registres GSPCRA, GSPCLA et GSPTEL pour travailler en mode ION.

L'étude détaillée et la compréhension de ces deux programmes permettant alors d'écrire tous les programmes de dessin qu'il est possible d'imaginer, dans la limite des possibilités du programmeur... et de la machine.

5) Limitations et possibilités de LAGROL.

* Limitations.

Elles sont essentiellement technologiques.

- le nombre de points que l'on peut éclairer au sein d'une image stable est faible.
- le fait de devoir régénérer l'image avec le calculateur est un handicap sérieux.

Si l'on désire avoir toujours une image sur l'écran, il faudra proscrire tout programme demandant de gros calculs. A noter que les délais d'éclairage des points sont automatiquement respectés par l'interprète.

-le manque d'élément arithmétique empêche de concevoir des programmes de dessin extrêmement précis.
- on peut considérer en fait que toutes les instructions de LAGROL ne sont que des simulations de dispositifs existant sur d'autres terminaux plus évolués. Mais pour obtenir des résultats comparables, il faut plus de temps avec LAGROL, et la précision du dessin est plus mauvaise.
- il ne faut pas oublier qu'aucun contrôle n'est exercé sur les adresses fournies en paramètres ou calculées. On aura donc toujours présent à l'esprit le fait que toute image sortant des limites de l'écran réapparaît de l'autre côté de celui-ci.
- enfin, le fait qu'il n'y ait aucune protection de mémoire rend la mise au point des programmes délicate. En particulier, lorsqu'un programme n'a pas marché, il est bon de vérifier que cette erreur n'a pas détruit le programme interprète.
* Possibilités.
- pour aider à la mise au point des dessins, nous avons pensé qu'il serait utile de pouvoir disposer commodément de l'échelle et de la luminosité. Pour cela, lors de l'exécution d'un programme, on peut utiliser à tout moment les clés qui sont sur le tableau de commande de PDP-8. Il y en a 12, numérotées de 0 à 11 à partir de la gauche.

clé 0 levée — l'intensité de tout le dessin est indiquée par les clés 9 à 11
clé 1 levée — l'échelle de tout le dessin est indiquée par les clés 6 à 8

Les deux options peuvent être utilisées en même temps ou indépendamment.
- la simplicité des conventions de LAGROL en fait un langage simple à assimiler et le rend propre à un usage pédagogique.
- le temps d'exécution d'un programme écrit en LAGROL est à peine supérieur à celui d'un programme écrit en langage machine. Par contre, l'écriture d'un programme de dessin est beaucoup plus rapide et agréable.
- le fait de disposer de sous-programmes et de possibilités de représentation des données permet d'obtenir un important gain de place en mémoire.
- du fait que les figures composant un dessin peuvent être définies comme des ensembles indépendants, on peut très bien imaginer des dessins en relief, ou plutôt en trompe-l'oeil, en utilisant des échelles et des luminosités différentes suivant les parties du dessin.

- enfin, l'étude de LAGROL permet de comprendre certains problèmes qui sont cachés lors de l'emploi de terminaux graphiques plus évolués. La bonne connaissance de ces problèmes est un gage d'efficacité dans l'emploi de ces terminaux évolués.

* Conclusion.

On voit que l'emploi de langages graphiques tels que LAGROL est très vite limité, dans la mesure où pour définir un dessin, il faut commencer par écrire un gros programme, processus fastidieux, d'autant plus que dans le cas précis de LAGROL l'emploi de lecteurs de cartes que nous avions dénoncé dans notre introduction n'est pas évité. L'idée est alors d'utiliser LAGROL comme un sous-programme de traitement d'un système plus complet permettant à l'utilisateur de dessiner à partir de la console et du crayon optique. C'est le but du système de dessin que nous allons étudier en détail dans les chapitres suivants et qui a pour nom "GENérateur d'Images Adaptées à Lagrol", ou GENIAL.

CHAPITRE III

U T I L I S A T I O N D'U N T E R M I N A L G R A P H I Q U E
E N M O D E C O N V E R S A T I O N N E L

A. GENERALITES SUR LE MODE DE TRAVAIL CONVERSATIONNEL.

1) Intérêt du travail en mode conversationnel.

Utiliser une console de visualisation uniquement comme un dispositif de sortie de résultats, c'est-à-dire comme un dispositif ne fonctionnant que dans le sens ordinateur-utilisateur, semble être une mauvaise conception de l'utilité des terminaux alphanumériques ou graphiques. En effet, l'intérêt de ces consoles est de nous permettre de suivre pas à pas le déroulement d'une action, et au vu des renseignements affichés sur l'écran, de pouvoir prendre une décision. Cette décision est souvent une modification du processus de calcul : changement de paramètres, correction du programme en cours d'exécution, par exemple. Si le seul moyen dont nous disposons est d'aller perforer un certain nombre de cartes pour corriger l'erreur et remettre tout le programme en route, alors nous retombons dans les difficultés des systèmes de passage en moniteur, ce qui, dans le cas de l'utilisation des consoles de visualisation, est parfaitement aberrant.

Le besoin se fait donc sentir d'avoir à sa disposition un moyen d'entrer les données de façon à pouvoir agir facilement sur le déroulement d'un programme en vue de le mettre au point rapidement ou d'améliorer ses résultats. L'idée est de disposer d'un système conversationnel, c'est-à-dire d'un programme qui soit capable d'analyser une demande en provenance de l'utilisateur, de vérifier que cette demande est conforme, si oui d'en lancer l'exécution, si non de la refuser en donnant un commentaire d'erreur et en attendant la prochaine demande. On voit qu'il s'agit de permettre à l'utilisateur de dialoguer avec le

calculateur. L'utilisateur va demander à l'ordinateur d'effectuer un certain nombre de tâches (calculs, vérifications) et l'ordinateur lui donnera en retour une réponse. Le fait de travailler directement avec l'ordinateur implique l'emploi de moyens d'entrée directs, excluant dont les lecteurs de cartes ou bandes perforées. Il reste les consoles,qui sont bruyantes et relativement lentes, et surtout les consoles de visualisation équipées de dispositifs dont nous avons parlé : boutons de commande, clavier alphanumérique, dispositifs de localisation.

Les difficultés de conception et de mise au point de tels systèmes sont nombreuses et assez importantes. Elles sont d'ailleurs différentes suivant qu'on se place du point de vue de l'utilisateur ou du point de vue de la conception. Il faut pouvoir fournir à l'utilisateur un choix de commandes qui couvre tous ses besoins. Mais l'efficacité de ces commandes va dépendre en grande partie de la représentation qui a été choisie pour les données, ce qui se fera au niveau de la conception du système. Le problème est donc double : définir d'abord un langage graphique permettant de représenter les données, puis créer un système superviseur qui permette de créer des figures et de les manipuler, et qui donnera le contrôle à l'interprète du langage graphique pour fournir à l'utilisateur une réponse sous forme d'images.

2) Quelques règles pour la conception de systèmes conversationnels.

Lors de la définition des spécifications auxquelles devra répondre le système, il faudra avoir à l'esprit deux règles primordiales :

- les facilités accordées à l'utilisateur doivent être maximales.
- le temps de réponse à une demande doit être minimal.

Ces deux remarques, apparemment évidentes, posent de graves problèmes, car elles sont contradictoires.

Accorder de grandes facilités à l'utilisateur peut vouloir dire que dans le corps d'une commande il pourra donner les paramètres dans un ordre quelconque, en oublier quelques-uns, ou encore que le champ de cette commande peut

être variable. Malheureusement, ces facilités ne sont obtenues qu'au prix d'un traitement relativement long, qui pourra dans certains cas devenir démesuré par rapport à la facilité accordée. Il s'agit d'arriver à un compromis entre les besoins de l'utilisateur et les facilités accordées pour lui permettre de réaliser facilement ses projets.

Les problèmes à résoudre ne sont donc pas simples, et dépendent de beaucoup de facteurs. Le premier est de savoir ce que l'on veut faire. Suivant qu'il s'agira de suivre la marche d'un satellite ou de tracer des réseaux de courbes, on n'aura pas les mêmes spécifications pour le système. Le deuxième facteur est le matériel employé. Que nous ayons une console de visualisation modeste ou très évoluée, que nous ayons un calculateur de faible puissance ou un calculateur très puissant, chaque fois notre ambition sera limitée par les performances du matériel : il nous faudra donc tenir compte, lors de la conception du système, des limitations apportées par le matériel. On peut cependant donner un certain nombres de règles générales qui seront employées avec profit.

Les commandes pourront être lancées à partir d'un clavier alphanumérique, auquel cas elles devront apparaître au fur et à mesure dans une zone spécialisée de l'écran. On pourra utiliser les boutons de commande. C'est le moyen qui demande le moins de traitement en machine pour vérifier quelle est la commande. Les boites de commande sont hélas limitées à un certain nombre de boutons (32 en général), et on a vite fait d'épuiser les possibilités de celles-ci. On pourra alors créer ces commandes à l'aide d'un crayon optique. Pour ce faire, on disposera d'une liste plus ou moins longue de mots-clés. On aura en permanence sur l'écran une portion de cette liste. Si le mot désiré n'est pas sur l'écran, on pourra le faire apparaître à l'aide d'une fenêtre, selon le procédé décrit plus haut. On montre alors ce mot avec le crayon optique et on l'amène dans une zone réservée à la fabrication des commandes. Lorsque la commande est achevée, l'ordinateur se charge de la décoder. Ce procédé est celui qui demande un temps de traitement le plus long, mais il est assez commode pour l'utilisateur.

La syntaxe des commandes doit être la plus simple possible, sans risque d'ambiguité. Quand c'est possible, il est recommandé d'aider l'utilisateur à former une commande correcte en l'obligeant à donner certains renseignements à un moment précis : on peut, par exemple, attirer son attention en faisant apparaître un code mnémotechnique. Le nombre de commandes doit être le moins élevé possible. Il faut éviter les commandes spécialisées chargées de traiter un cas particulier. Il vaut mieux employer des commandes générales, ce qui évite ainsi une prolifération inutile des mots de commande.

Le choix de l'unité d'information est important. On pourra décoder une commande caractère par caractère, mot par mot, ou encore ligne par ligne. Chaque méthode à ses avantages et ses inconvénients. Le traitement caractère par caractère impose un certain nombre de vérifications lors de l'entrée de chaque caractère, mais permet d'avertir l'utilisateur au premier caractère incorrect. Dans le cas de la vérification ligne par ligne, on n'entreprend toutes ces vérifications qu'à la fin de la ligne, ce qui fait qu'un message d'erreur éventuel ne peut être fourni qu'assez tard. Certains dispositifs obligent au traitement ligne par ligne.

Le problème des messages d'erreur est aussi important. Il faut d'abord pouvoir éliminer tout risque d'erreur, ce qui demande en général une bonne partie du temps de traitement. D'autre part, à la détection d'une erreur, il faut que le message fourni soit bref mais explicite. Un message particulier ne pourra être employé pour deux erreurs différentes qu'à la condition que ces erreurs ne diffèrent pas sur le fond. Il est totalement inutile d'utiliser le message "erreur" chaque fois que l'utilisateur commet une faute. Au vu d'un message, l'utilisateur doit pouvoir dire quel genre de faute il a commis (syntaxe, sémantique, etc...), avant même de savoir précisément quelle est la faute. Si on le peut, il est préférable de lancer des messages en clair plutôt que des avertissements codés qui nécessitent un manuel de décryptage pour en tirer parti.

Un dernier point très important est de prévoir la possibilité d'ajouter de nouvelles commandes ou de donner de nouvelles facilités au système.

Ceci permet de l'enrichir constamment, à la demande de l'utilisateur, sans avoir besoin de reprendre la conception du système à la base. Moyennant le respect de ces quelques règles simples, l'écriture d'un système conversationnel devient relativement plus aisée, sans être facile cependant. La suite du chapître donne la description détaillée des commandes mises à la disposition de l'utilisateur du système de dessin GENIAL.

B. LE SYSTEME DE DESSIN GENIAL, VU DU COTE DE L'UTILISATEUR.

1) Principes généraux.

* Généralités sur les commandes.

Ne disposant pas de clavier alphanumérique, nous nous servons de la console du PDP-8 pour lancer les commandes. Les réponses sont également données sur la console.

- toutes les commandes même celles qui utilisent exclusivement le crayon optique, sont lancées à partir de la console.
- pour lancer une commande, il suffit de taper les deux premières lettres de celle-ci, sauf dans le cas de DEBUT, DEPLACER et DETRUIRE, où il faut frapper les 3 premières lettres. Le nom de la commande est complété automatiquement et le cadrage assuré.
- toute commande doit être entièrement terminée avant le lancement de la commande suivante.
- lorsqu'une commande est terminée, le chariot de la machine à écrire vient se placer en tête de la ligne suivante. Tant que ce mouvement n'a pas été effectué, la commande n'est pas entièrement traitée.
- on peut frapper à tout instant les caractères suivants :

retour du chariot	touche RETURN
avance ligne	touche LINE FEED
tabulation	touche ALT MODE

La tabulation fait avancer le chariot de dix positions. Elle sert à obtenir une présentation élégante du texte des commandes.

- lors de la lecture d'un nombre à la console, on le calcule au fur et à mesure de la frappe des chiffres. La fin du nombre est indiquée par une virgule ou un point-virgule. Tout caractère autre qu'un chiffre est refusé (voir page 72 "message d'erreur").

- un certain nombre de précautions sont prises lors de la lecture d'un nombre. S'il s'agit d'une adresse, seuls les 4 derniers chiffres octaux de 0 à 1777 sont gardés à la fin du calcul. S'il s'agit d'une échelle ou d'une densité, seul le dernier chiffre est conservé. Ceci peut être une aide ou un inconvénient.

- tant que l'on n'a pas spécifié autre chose, les nombres sont lus en octal.

- les commandes comportant des données(coordonnées, échelle, densité) doivent se terminer par un point-virgule.

- un point-virgule frappé en cours de commande est traité comme une virgule.

- on donnera les coordonnées d'un point dans cet ordre : abscisse puis ordonnée.

- lorsqu'un renseignement particulier doit être donné (échelle, luminosité, nom de fonction), une lettre est tapée par la machine

  E pour échelle
  L pour luminosité
  F pour fonction

  il suffit alors de donner le nombre correspondant à l'option désirée.

- Lorsque l'on veut définir un objet, il faut indiquer si l'on désire employer la console ou le crayon optique pour donner certains renseignements. On utilisera le caractère ,, pour désigner la console
  et le caractère ! pour désigner le crayon.
  Cette information doit être donnée immédiatement après le nom de la commande après TRIANGLE, RECTANGLE, DEPLACER, EFFACER, CERCLE ; dans le cas de VECTEUR ou POINT, un traitement spécial est prévu, qui est décrit page 64 .

- si l'on frappe autre chose que l'un de ces deux caractères, la commande est refusée.

- en cas d'erreur, toute la commande est refusée, sauf s'il s'agit d'une erreur dans la lecture d'un nombre. Toutes les précisions sont données au paragraphe "messages d'erreurs", page 72 .

- il existe une commande qui permet de donner des commentaires entre chaque commande. Il est recommandé de l'utiliser pour indiquer en particulier la nature de l'image qui vient d'être générée. On pourra ainsi garder une trace du travail effectué, ce qui pourra être utilisé plus tard.

* Définition des modes de travail.

Le système GENIAL donne à l'utilisateur le choix entre deux modes de travail, modes complémentaires comme on va le voir. Le premier mode, appelé mode de génération, permet de créer en mémoire un certain nombre de figures à partir d'objets que fournit le système. Ces images portent un nom, (en fait un numéro), qui va permettre de leur faire référence. Toutes les commandes lancées en mode de génération auront donc pour but de modifier une liste, dite liste de génération, soit en ajoutant des éléments à celle liste (nouvelles figures), soit en recopiant des éléments de cette liste dans d'autres figures . A cette liste de génération est associée une liste des figures qui contient l'adresse en mémoire de chaque figure et qui permet ainsi de retrouver toute figure connue.

Le deuxième mode, appelé mode d'utilisation, permet d'employer des figures déjà nommées, c'est-à-dire déjà générées, pour faire des images. Ce mode de travail ne permet pas de créer de nouvelles figures. On se contente de générer des appels de sous-programmes dans une liste dite liste d'affichage qui constitue la description exacte des composants de l'image qui est sur l'écran, dans ce mode de travail. On voit donc que généralement, on commencera par travailler en mode de génération, pour se créer un certain nombre de figures de base, puis on travaillera en mode d'utilisation pour obtenir l'image désirée.

Lorsque l'on est en mode de génération, la seule chose qui apparaît sur l'écran est la figure que l'on est en train de construire. Lorsque l'on est en mode d'utilisation, l'image obtenue est formée de toutes les figures contenues dans la liste d'affichage.

Il faudra faire attention à tout moment au mode de travail en cours,

car certaines commandes ne sont acceptées qu'en mode de génération, et d'autres commandes, acceptées sous les deux modes, ont une signification différente suivant qu'elles s'adresseront à la liste de génération ou à la liste d'affichage.

* Utilisation du crayon optique.

L'utilisation du crayon optique peut se faire de deux façons : soit il s'agit d'obtenir les coordonnées d'un point, soit il s'agit de désigner un objet figurant sur l'écran. Dans le premier cas, en plus du dessin déjà sur l'écran, apparaissent une croix et un point situé dans le coin supérieur droit de l'écran. En montrant la croix avec le crayon optique, on la déplace jusqu'à l'endroit voulu, sachant que le point de repère est le centre de la croix. Si l'on désire alors que les coordonnées de ce point soient prises en compte, on montre le point de commande. Celui-ci disparaît alors, pour éviter que le crayon ne voie le point plus d'une fois. Pour dessiner deux vecteurs consécutifs, on amène la croix au point de départ du premier vecteur, on montre le point d'arrêt, puis on amène la croix sur le deuxième point définissant le premier vecteur, on montre le point d'arrêt qui est revenu dès que la croix s'est déplacée, on amène enfin la croix au point d'arrivée et on montre le point d'arrêt une dernière fois. La croix disparait alors de l'écran et le chariot de la machine à écrire passe à la ligne. Tant que le chariot de la machine à écrire n'est pas passé à la ligne, cela veut dire qu'on n'a pas donné assez de points et que l'on ne peut pas lancer de nouvelle commande. A noter que le dessin apparaît au fur et à mesure que les points successifs sont donnés.

Dans le deuxième cas, il suffit de montrer avec le crayon optique un objet de la figure que l'on veut déplacer ou mettre en mouvement ou l'objet même qu'on désire effacer. Attention aux intersections de figures différentes ! Il ne faut pas oublier que même si l'image est stable, c'est une illusion d'optique et le point éclairé à l'intersection appartient consécutivement à chacune des figures qui se coupent en ce point. En désignant ce point avec le crayon optique, on obtiendra l'une des figures, celle qui était allumée à ce moment précis, mais pas forcément celle que l'on voulait.

Il ne faut pas oublier d'appuyer sur le bouton commandant l'ouverture de la cellule photoélectrique si l'on désire montrer un point de l'écran. En revanche il vaut mieux éviter de montrer toute autre source de lumière, le crayon optique étant incapable de faire la différence entre une lumière vue sur l'écran ou produite par une ampoule électrique, ou encore... par le soleil. On s'expose à obtenir des résultats pour le moins curieux.

2) Description détaillée des commandes.

* Commandes utilisables en mode de génération.

- GENERER

Cette commande place l'utilisateur en mode de génération. Les commandes qui suivent devront pouvoir être acceptées dans ce mode de travail. Ce sont des commandes de génération de figures ou de modification de la représentation en mémoire. La commande GENERER est obligatoire lorsqu'on définit une nouvelle figure. Elle ne peut être frappée qu'en mode utilisation. Ce qui veut dire que l'on ne peut pas commencer à générer une figure sans avoir au préalable conservé ou détruit la précédente.

- VECTEUR ou VECTEURS

Cette commande permet d'obtenir le tracé d'un ou plusieurs vecteurs consécutifs. Dans le cas où l'on désire tracer un seul vecteur, on complètera le mot VECTEUR en frappant un espace. La machine attendra alors l'indication du dispositif employé (console ou crayon optique), puis l'échelle et la luminosité. Si l'on a demandé un tracé à l'aide du crayon optique, il faudra alors terminer par un point virgule. Si non, on donne les coordonnées du point de départ et du point d'arrivée, en prenant soin de terminer par un point virgule. Dans le cas où l'on désire tracer plusieurs vecteurs, on complètera en frappant la lettre S, et on donnera alors le nombre de vecteurs que l'on désire dessiner, suivi des mêmes renseignements que précédemment.

Exemple: VECTEUR #,E3,L5,0,0,1777,1777;↵
VECTEURS 12,!,E3,L6;↵

Le signe ↵ signifie que le chariot passe à la ligne.

Nous avons souligné ce qui est frappé par l'utilisateur.

- POINT ou POINTS.

Cette commande permet d'obtenir le tracé d'un ou plusieurs points. On suivra les mêmes règles que dans le cas des vecteurs. L'échelle n'ayant aucun sens n'est pas demandée.

- TRIANGLE

Cette commande permet de dessiner un triangle dont on donne les sommets.

Exemple: TRIANGLE ,E2,L4,0,0,500,500,1200,100;

TRIANGLE !,E2,L4;

- RECTANGLE

Cette commande permet d'obtenir un quadrilatère dont on donne les quatre sommets. Elle s'appelle RECTANGLE, parce que ce nom...est plus court que quadrilatère. Elle se lance de la même façon que TRIANGLE.

- CHAINE

Cette commande permet d'afficher une chaîne de caractères en un point quelconque de l'écran. Il faut donner l'échelle, la luminosité, le point de départ, et enfin le texte de la chaîne, qui se termine par le caractère @ . Ne pas oublier que le caractère \ fait passer à la ligne sur l'écran.

Exemple: CHAINE E3,L5,0,1000,TOUS LES CARACTERES SONT PERMIS@

On ne peut générer de chaîne avec le crayon optique.

- CERCLE

Cette commande permet d'obtenir des cercles ou des arcs de cercle. Les renseignements à fournir sont exactement ceux que demande l'interprète LAGROL. On donnera donc l'échelle, la luminosité, les coordonnées du centre,

le rayon, et les abscisses des points de départ et d'arrivée de l'arc précédées d'un signe + ou d'un signe - (voir les conventions de LAGROL, page 42). Pour obtenir un cercle complet, on donnera les abscisses de deux points consécutifs.

Exemple : CERCLE #, E3, L4, 1000, 1000, 400, +1000, -1400;

CERCLE #, E4, L5, 1000, 1000, 400, +1000, +1001;

Ne pas oublier que le cercle est dessiné dans le sens trigonométrique. Aucun contrôle n'est effectué quant aux points fournis en paramètre. Il faut donner des points se trouvant bien sur le cercle.

- CONSERVER

Cette commande est utilisable dans les deux modes de travail. Ici, elle veut dire que la figure que l'on a commencé à définir à partir de la dernière commande GENERER est terminée. On reçoit en retour un numéro, qui caractérisera la figure dans la suite du travail. On se trouve alors en mode d'utilisation, ce qui veut dire que l'on peut soit créer des images, soit redonner la commande GENERER pour définir une nouvelle figure.

- AJOUTER

Cette commande permet de recopier au sein de la figure en cours de génération une figure déjà générée, c'est-à-dire ayant un numéro. L'intérêt de cette commande est qu'on pourra changer les caractéristiques de la copie sans modifier la figure de base, ce qui est souvent utile. On donne simplement le numéro de la figure à copier, et celle-ci apparaît dans l'état exact où elle se trouve définie au moment du lancement de la commande.

Exemple : AJOUTER 3;

- DETRUIRE

Cette commande est valable dans les deux modes de travail. En mode de génération, elle signifie que la figure que l'on était en train de créer ne nous intéresse plus. On l'efface entièrement de la liste de génération, et on

passe alors en mode d'utilisation. On voit donc que pour sortir du mode de génération on devra frapper soit CONSERVER, soit DETRUIRE.

Il faut noter que les commandes d'animation qui sont décrites un peu plus loin ne peuvent être employées qu'en mode de génération. D'autres commandes peuvent également être employées dans ce mode, mais elles sont classées dans des catégories étudiées plus loin.

* Commandes employées en mode d'utilisation.

- AFFICHER

C'est la commande de base du mode d'utilisation. Elle permet d'afficher la figure dont le numéro est donné en premier, à partir du point dont les coordonnées suivent. A noter que c'est le point de départ de la figure qui est pris en compte. Il y a génération d'un appel de sous-programme précédé d'une correction à l'aide d'un ordre GMODIF. Cette figure est ajoutée à la liste d'affichage. Pour former une image complexe, on répètera autant de fois qu'il le faudra la commande AFFICHER.

Exemple: AFFICHER 3,212,546;↵
AFFICHER 1,1000,1777;↵

- DETRUIRE

Dans ce mode de travail, DETRUIRE permet d'effacer la liste d'affichage, afin de pouvoir construire une nouvelle image. On reste dans le mode d'utilisation.

- CONSERVER

Dans ce mode de travail, on peut conserver une image de la liste d'affichage. Celle-ci est recopiée dans la liste de génération, et la nouvelle figure créée peut être utilisée au même titre que les figures générées, avec le numéro qui lui a été attribué. On reste dans le mode d'utilisation. La liste d'affichage n'est pas modifiée.

* Commandes de manipulation d'images.

Nous désignons sous ce vocable un ensemble de commandes permettant de modifier certains paramètres dans la liste de génération de façon à déplacer ou modifier certaines portions d'images. Ces commandes peuvent être lancées indifféremment en mode de génération ou en mode d'utilisation. En fait, leur signification est essentiellement liée au fait que l'on emploie la console ou le crayon optique pour donner les renseignements voulus. En règle générale, l'emploi de la console voudra dire que l'on désire modifier une figure de base en entier (effacer la totalité de la figure), l'emploi du crayon optique voudra dire que l'on désire modifier une portion d'image (effacer un objet dans une figure).

- EFFACER

Cette commande permet d'effacer des figures ou des objets définissant une figure. Si l'on écrit EFFACER #,4; l'image numéro 4 n'apparaîtra plus sur l'écran même en utilisant AFFICHER. Si l'on écrit EFFACER !; seul l'objet désigné par le crayon optique (ou plutôt toutes ses occurences) disparaîtra. On reçoit en retour le numéro de la figure qui a été désignée.

- DEPLACER

Cette commande permet de modifier l'implantation d'une figure. Si l'on utilise la console, on modifie l'implantation de la figure de base, ce qui veut dire que toutes les occurences de la figure dans une image se déplaceront. On écrit simplement DEPLACER #,4,+5,-5; ce qui veut dire que l'on déplace l'image numéro 4 de 5' unités vers la droite, et de 5' vers le bas.

Si l'on utilise le crayon optique, deux cas se présentent : soit l'on se trouve en mode de génération, et c'est l'implantation de l'image en cours de génération qui est modifiée, soit l'on se trouve en mode d'utilisation et seule la figure montrée par le crayon optique est déplacée. On voit donc que lorsque l'on utilise le crayon optique, seule une figure est déplacée.

- RALLUMER

Cette commande permet de rallumer une figure éteinte avec la commande EFFACER. Il suffit de donner le numéro de la figure à rallumer. Attention! Si

une figure a été éteinte en partie à l'aide du crayon optique, et si l'on a enfin éteint complètement la figure à l'aide de la console, il sera nécessaire de donner deux ordres de rallumage consécutifs si l'on désire rallumer toute la figure.

* Commandes d'animation de figures.

Ces commandes ne peuvent être employées que dans le mode de génération. Elles servent à définir des figures contenant des objets dotés de mouvement. Ces mouvements sont obtenus en faisant précéder la liste des objets à animer d'un appel à une fonction spécialisée qui se charge de faire exécuter des translations continues ou alternatives, ou tout mouvement à l'idée du programmeur.

- APPLIQUER

Cette commande se place en tête de la liste des objets à animer. Tous les objets qui suivent seront animés du même mouvement. On doit donner ensuite le nom de la fonction que l'on désire employer, suivi d'un certain nombre de paramètres qui dépendent de la fonction choisie. On aura par exemple la fonction F0, qui fait exécuter un mouvement de translation rectiligne continu, et qui a 2 paramètres, le pas en X et le pas en Y. On pourra utiliser aussi la fonction F1 qui effectue le même mouvement, mais alternativement dans un sens puis dans l'autre , sur une longueur donnée en paramètre. Un exemple d'utilisation de ces deux fonctions est donné un peu plus loin. Pendant la génération, l'image est fixe.

- ANIMER

Cette commande permet de mettre en mouvement l'objet désigné par le crayon optique. En réponse, on obtient le numéro de la figure à laquelle appartient l'objet (ou la liste d'objets) que l'on vient d'animer. On vérifie que l'objet désigné était bien sous contrôle d'une fonction d'animation.

- ARRETER

Cette commande permet d'arrêter un objet mis en mouvement par ANIMER, en le désignant avec le crayon optique. On obtient les mêmes réponses que pour la commande ANIMER.

- FIN D'APPLICATION

Cette commande indique que le dernier objet généré est le dernier de la liste qui doit être animée par la fonction désignée dans la précédente commande APPLIQUER. On ne pourra conserver une figure tant que l'on n'aura pas défini complètement le champ d'application d'une commande APPLIQUER.

Exemple:

```
GENERER
CHAINE     E2,L3,0,0,ANIMATION D'IMAGE@
APPLIQUER F0,+10,+10;
VECTEUR    !,E4,L5;
FIN D'APPLICATION
APPLIQUER F1,+10,-50,300,0;
TRIANGLE   !,E2,L5;
APPLIQUER F2,-5,+30,500,0;
RECTANGLE  !,E7,L4;
FIN D'APPLICATION
APPLIQUER F0,-25,+30;
VECTEURS   14,!,E4,L2;
FIN D'APPLICATION
FIN D'APPLICATION
CHAINE     E2,L4,0,1500,DERNIER OBJET DE LA FIGURE@
CONSERVER FIGURE  0001
```

* Commandes diverses.

- DEBUT

Cette commande sert à initialiser correctement le système au cas où l'on désire entreprendre un nouveau travail. Les listes des figures, de génération et d'affichage sont nettoyées, le mode de travail est l'utilisation, la lecture des nombres se fait en octal. Cette commande n'est pas obligatoire. En particulier lorsque l'on continue un travail sauvegardé par RESERVER, il ne faut pas la taper.

-LOCALISER

Cette commande sert à demander les coordonnées d'un point de l'écran en utilisant le crayon optique. On obtient ces coordonnées à la suite de la commande.

- NOMMER

Cette commande permet de demander le numéro d'une figure qui apparaît sur l'écran. Ce numéro est donné en réponse derrière la commande lorsque 1 on montre un objet quelconque appartenant à la figure qu'on désire connaître. Cette commande sert à reconnaître à quelle figure appartient l'occurence qui nous intéresse.

- PRESERVER.

Cette commande permet de sauvegarder sur bande magnétique le fruit d'un travail que l'on désire conserver. Il s'agira par exemple de la définition d'un ensemble de symboles de base permettant de dessiner ensuite sans avoir à redéfinir ces symboles (réseaux électriques, par exemple). Lorsque cette commande est lancée, le contrôle est alors donné à un programme qui se charge de transférer sur bande magnétique le contenu de la zone donnée en paramètre (ici 0,7577). Le point de départ est la mémoire 4. On préserve les listes des figures d'affichage et de génération ainsi que le contenu d'un certain nombre de compteurs. Pour réutiliser ce travail, il ne faudra pas taper DEBUT lors de la prochaine utilisation. Après une préservation, il faut rappeler le générateur en mémoire (voir page 76 la description complète des commandes à lancer) .

- *OCTAL

Cette commande veut dire que la lecture de tous les nombres qui seront donnés après elle se fera en octal. Il faut noter que c'est l'option standard. Tous les nombres écrits par la machine sont en octal.

- *DECIMAL

Cette commande permet d'utiliser des nombres écrits en décimal.

- *ADRESSE

Cette commande permet de savoir à tout instant la place qui reste à

utiliser dans la page courante de la liste de génération. Cette commande est très importante, puisque le découpage de la mémoire du PDP-8 interdit de générer des figures de plus d'une page. Lorsque l'on lance cette commande, on obtient en retour l'adresse du prochain mot qui sera utilisé dans la page courante. Cette adresse est comprise entre 0 et 177.

- *PAGE

Cette commande permet de changer de page lorsqu'on a épuisé la place utilisable dans la page courante. Cette instruction ne peut être tapée qu'en mode d'utilisation. Ce dernier point interdit de générer une figure à cheval sur deux pages.

- **

Cette commande permet d'introduire des commentaires entre deux commandes successives. Il est recommandé d'en faire usage, surtout pour créer des bandes avec un certain nombre de symboles de base, en indiquant à côté de la figure créée ce qu'elle représente. On peut utiliser tous les caractères, sauf l'étoile, qui termine la commande. Une deuxième étoile est alors imprimée automatiquement... pour des questions d'esthétique.

3) Messages d'erreur.

Chaque fois qu'une erreur est détectée, la commande en cours est annulée (sauf dans la lecture des nombres) et l'utilisateur est averti par l'impression d'un message, précédé d'un coup de sonnette.

- INTERDIT

Ce commentaire indique que la commande que l'on vient de lancer ne peut être utilisée dans ce mode de travail. Il s'agit le plus souvent d'une commande qui ne peut être utilisée qu'en mode de génération. La commande est ignorée.

- ??

Les deux points d'interrogation sont suivis d'un retour à la ligne. Ils signifient que la commande que l'on vient de frapper (ou qui était en cours

de frappe) est incorrecte. Ce message apparaît dans quatre cas :

la commande n'est pas terminée par un point-virgule

on n'a pas indiqué le moyen d'entrée (# ou !)

on a frappé autre chose que S ou un espace après VECTEUR ou POINT

on a frappé une commande inconnue

Toute la commande est ignorée, sauf dans le cas de DEPLACER.

- ?

Si un seul point d'interrogation apparaît, suivi d'un espace, c'est que l'on a commis une erreur dans l'écriture d'un nombre. Il y a deux types d'erreur : soit on a frappé un caractère non numérique, soit on a frappé un chiffre supérieur à 7 en mode de lecture octal. L'ensemble des chiffres frappés à partir de la dernière virgule est ignoré et il suffit de taper le bon nombre pour corriger l'erreur. Ceci peut servir au cas où l'on s'aperçoit qu'on a commis une erreur en donnant un nombre. Si l'on n'a pas encore frappé la virgule terminant le nombre, il suffit de frapper n'importe quel caractère non numérique pour annuler le nombre qu'on vient de frapper. Si l'on désire annuler toute la commande, il suffit d'appuyer sur la touche d'effacement (RUB OUT).

- PAGE DEPASSEE

Ce commentaire indique que l'objet que l'on est en train de construire est trop grand et ne tient pas dans le reste de la page courante. On donne en plus du commentaire PAGE DEPASSEE l'adresse de la place disponible après le dernier objet entièrement créé. Cette indication est précieuse car elle va permettre de gérer de façon rationnelle la mémoire. En effet, lorsque l'on obtient ce commentaire, plusieurs décisions peuvent être prises. Premièrement, la figure que l'on était en train de créer ne nous convient absolument pas. On peut alors DETRUIRE celle-ci et recommencer à définir une nouvelle figure, sans changer de page. Ou alors le début de la figure en cours de génération nous convient, et alors, suivant la place restant dans la page, on peut décider de la CONSERVER, ou de générer de taille moindre que le précédent avant de la CONSERVER. On changera alors de page en frappant *PAGE.

Il faut se rappeler que l'on ne pourra changer de page que lorsque l'on aura lancé une commande CONSERVER ou DETRUIRE. Ceci interdit de créer des figures dont la tête se trouverait dans une page et la fin dans une autre.

- FIGURE INCONNUE

On a donné dans une commande le numéro d'une figure qui n'existe pas. Il faut faire attention au fait que la numérotation des figures commence à 1. Il n'y a pas de figure 0. Attention aussi au fait que les numéros des figures sont donnés en octal. Si on travaille en décimal, ne pas oublier la conversion.

- FONCTION INCONNUE

On a donné comme paramètre d'APPLIQUER une fonction qui n'existe pas dans le catalogue. La commande est ignorée.

- FIN D APPLICATION??

Ce commentaire apparaît dans un cas : on a lancé une commande CONSERVER alors que le champ d'application d'une commande APPLIQUER n'a pas été complètement défini. Il suffit de frapper FIN D'APPLICATION pour pouvoir poursuivre le travail, si l'on désire conserver l'image en cours de génération.

- SANS ANIMATION

Ce commentaire est donné lorsque l'objet que l'on a montré avec le crayon optique après une commande ANIMER ou ARRETER ne fait pas partie d'une liste précédée d'une fonction d'animation. La commande est ignorée.

- MEMOIRE SATUREE

L'espace libre réservé à la liste de génération est épuisé. On peut encore se servir de la liste d'affichage pour créer des images, mais on ne peut plus rien conserver.

- TROP DE FIGURES.

Ce commentaire apparaît lorsque la liste des figures est saturée. On ne peut plus conserver de dessin, mais on peut encore utiliser la liste d'affichage. On peut créer environ une centaine de figures différentes.

- DESSIN COMPLET

Il y a dépassement de capacité de la liste d'affichage. On peut alors effacer celle-ci, ou au contraire la conserver, effacer la liste d'affichage puis afficher la dernière figure créée. On obtiendra la même image que précédemment, mais la liste d'affichage pourra contenir de nouvelles figures. On peut afficher jusqu'à 63 figures simultanément.

On voit donc que la liste des messages d'erreur est relativement brève. Toutes les possibilités d'erreur sont cependant couvertes, sauf celles qui viennent de l'inattention de l'utilisateur, donnant par exemple pour un triangle deux sommets confondus, ou des points situés en dehors du cercle à créer.

4) Mise en route du système.

La mise en route du système se fait de façon très simple. Elle est fondée sur l'utilisation d'un petit programme du PDP-8 qui doit se trouver théoriquement toujours en mémoire et qui, après avoir lu le nom d'un fichier sur la console, va chercher ce fichier sur une bande magnétique, le charge en mémoire et donne le contrôle à l'adresse de départ qui lui a été fournie lors de la mise sur bande du fichier. Pour lancer ce programme, il faut

afficher aux clés du PDP-8	7600
appuyer sur la touche	LOAD ADDRESS
appuyer sur la touche	START

La bande magnétique en question se met à effectuer quelques mouvements, s'arrête et le chariot de la machine à écrire revient à la ligne. C'est à ce moment là seulement que l'on pourra commencer à frapper le nom du fichier. Lors de la mise en route du système, on doit chercher deux ou trois fichiers sur bande. Il faudra donc attendre chaque fois le retour du chariot à la ligne pour pouvoir donner le nom du fichier désiré. Il y a deux cas possibles : soit l'on désire commencer un nouveau travail, auquel cas on charge un premier fichier INISYS qui contient les tables de messages et les fonctions d'animation, puis un deuxième fichier GENIAL contenant le superviseur et l'interprète LAGROL; soit il s'agit de la suite d'un travail préservé sur bande, il faudra encore appeler le fichier contenant ce travail. D'où les deux schémas possibles :

INISYS↵	INISYS↵
N OU S?↵	N OU S?↵
N↵	S↵
APPELEZ LE GENERATEUR↵	NOM DE VOTRE IMAGE↵
GENIAL↵	XXXXXX↵
	APPELEZ LE GENERATEUR↵
	GENIAL↵

Le chariot de la machine à écrire se place alors en début de ligne, et un coup de sonnette prévient l'utilisateur qu'il peut commencer à travailler. Dans le premier cas, il frappera DEBUT, dans le deuxième il peut commencer à AFFICHER ou lancer toute autre commande.

Pour préserver sur bande un travail, on utilise la séquence des opérations la suivante :

PRESERVER↵	commande lancée par l'utilisateur
APPELEZ 'UPDATE'↵	réponse du générateur
UPDATE↵	nom du programme de service
PROGRAM NAME : XXXXXX↵	nom du fichier à conserver
SA (OCTAL) : 4↵	cette adresse est impérative
LOCATION : <0,7677> ;↵	ces limites sont obligatoires
GENIAL↵	on rappelle le générateur en mémoire.

A partir de ce moment, on peut soit continuer le travail entrepris, soit tout recommencer en frappant DEBUT.

5) Quelques conseils pour bien dessiner à l'aide de GENIAL.

* Utilisation du crayon optique.

- si les dessins obtenus à l'aide du crayon optique sont obtenus plus rapidement qu'à l'aide de la console, il faut bien voir qu'ils sont forcément moins précis, à moins de posséder des points de repère. Ceux-ci peuvent être facilement obtenus en créant une image ne contenant qu'un seul point, et que l'on affichera en plusieurs endroits en des points de coordonnées ainsi bien connues; il suffira alors d'amener la croix sur ces points pour obtenir une image précise. Il est bien évident que ceci ne peut servir que pour tracer des dessins de vecteurs, ou encore pour déplacer une image en un point précis. Ce procédé n'est d'aucune utilité pour dessiner une suite de points. Pour des dessins très précis obtenus à l'aide de points, il faudra se résoudre à donner les coordonnées à l'aide de la console. Ne pas oublier alors de terminer la commande par un point virgule... sinon elle sera perdue en entier.

- la croix étant dessinée à partir du centre, on peut utiliser cette possibilité pour la déplacer plus rapidement d'une extrêmité à l'autre de l'écran. Au lieu de la déplacer sur toute la surface de l'écran, il suffit de faire dépasser légèrement un des bras de la croix au bord de l'écran. Un petit trait, le morceau manquant, apparaît de l'autre côté de l'écran ; il suffit de montrer avec le crayon optique pour voir apparaître la croix à cet endroit.

- pour viser un élément d'un menu, il est recommandé de donner aux lettres de ce menu l'échelle maximum en employant les clés du PDP-8.

- lorsque l'image affichée sur l'écran est complexe, la croix ne peut être déplacée que très lentement car elle n'est dessinée qu'après tout le reste de l'image lors de chaque balayage. Pour accélérer le mouvement, il est intéressant d'EFFACER à l'aide de la console les figures ne servant à rien, quitte à les rallumer ensuite. On a intérêt à réduire l'intensité de l'image en utilisant les clés, de façon à bien voir ce que l'on fait avec le crayon optique. L'intensité de la croix est invariable.

- pour amener la croix avec précision sur un point, il est intéressant d'utiliser le bouton de commande de la sensibiblité du crayon optique, situé en bas et à gauche de l'écran. En diminuant la sensibilité du crayon optique, on parvient à ralentir les mouvements de la croix et ainsi à la placer au point désiré.

* Remarques diverses

- lors de la génération des figures, il y a intérêt à employer les clés du PDP-8 pour trouver quelles sont les meilleures valeurs d'intensité et d'échelle à employer pour définir l'objet. Il ne faut alors pas hésiter à détruire la figure en cours de génération pour la recommencer.

- si l'on emploie une intensité très forte, il faut employer une grande échelle. Si l'on emploie une intensité faible, il vaut mieux au contraire employer une échelle faible. Ceci permet d'obtenir une image dont l'éclairement est à peu près uniforme pour l'oeil.

- si l'on veut définir une image très compliquée, il y a intérêt à utiliser des intensités élevées, sinon l'image paraît très vite instable. On a intérêt aussi à employer des échelles très grandes, pour abaisser le nombre de points utilisés.

- lorsque l'on a créé une figure à l'aide du crayon optique, il est recommandé d'utiliser immédiatement après la commande LOCALISER pour connaître les coordonnées du point de départ de la figure ainsi obtenue. Ceci permet d'éviter des surprises lors de l'affichage de cette figure.

- le fait que l'image est peu stable peut être utilisé avec profit pour créer des images possédant une certaine animation du style des enseignes lumineuses ou journal lumineux.

- si l'on désire définir une figure dont le point de départ ne se trouve pas sur le tracé (par exemple intersection des diagonales pour un rectangle), il suffit de définir d'abord un point à l'endroit désiré, de définir le reste de la figure et d'effacer ce point à l'aide du crayon optique.

- Il ne faut pas hésiter à créer des listes très précises avec de nombreux commentaires, surtout si le travail qu'on entreprend est destiné à être utilisé par d'autres (symboles en particulier).

Connaissant maintenant le système GENIAL du point de vue de l'utilisateur, il est intéressant de chercher à voir les problèmes qui se sont posés au moment de la conception, et quelles solutions ont été adoptées. Ceci fait l'objet du chapitre suivant, qui traitera de différentes structures d'images pour étudier en détail celle qu'utilise GENIAL.

CHAPITRE IV

N O T I O N S S U R L E S S T R U C T U R E S D' I M A G E S

A. GENERALITES A PROPOS DES STRUCTURES D'IMAGES.

1) Qu'est-ce qu'une structure d'image ?

Les problèmes à résoudre pour la représentation des données ne consistent pas seulement à pouvoir représenter d'une façon commode un certain nombre de figures, mais aussi à pouvoir représenter certaines relations qui les unissent, relations appelées contraintes. Par exemple, une droite devra toujours passer par tel point, ou être perpendiculaire à une autre, etc...Dans le cas le plus simple, une image est représentée par la suite des instructions qui permettent de l'obtenir. La structure ainsi définie est dite à un seul niveau. Si maintenant on a la possibilité d'utiliser des sous-programmes, on crée aussitôt un niveau supplémentaire. En fait, on a la possibilité de définir un certain nombre de niveaux, la difficulté étant alors de savoir à quel niveau dans la hiérarchie se place une certaine figure. Il faut savoir exactement ce qu'entraîne l'action d'effacement d'une figure, par exemple : doit-on effacer la figure en question ou toutes celles qui lui sont rattachées ?. Si on déplace un objet, il faut pouvoir trouver les modifications que ce déplacement impose aux autres objets. On est alors amené à définir une représentation qui permette d'exprimer convenablement les liens et les contraintes entre images. Cette représentation liée à celle des figures est appelée structure d'image.

2) Quelques structures d'images.

* Structures simples.

Cette structure ne possède qu'un seul niveau. Une image est formée

de la suite des instructions qui la définissent. Pour afficher cette image, on exécute en séquence les ordres de la liste jusqu'au dernier et on recommence au début. Pour modifier l'image, il faut refaire toute la liste. On se trouve donc en présence d'une structure linéaire. Cette structure est surtout employée pour les terminaux alphanumériques, et parfois pour obtenir des courbes sur les terminaux graphiques.

* Structure en tables.

Pour gagner de la place, surtout lorsque l'on a la possibilité de construire des figures sous forme de sous-programmes, on peut imaginer que l'image sera composée d'appels de sous-programmes, effectués à l'aide de tables contenant les adresses de chacune des figures de base. On gagne ainsi une place importante, mais ces structures ne sont pas d'une grande souplesse d'emploi. Un exemple détaillé d'une telle structure est donné dans la deuxième partie de ce chapitre.

* Structures de liste.

Il peut être intéressant d'utiliser une structure de liste du genre de celles utilisées par LISP ou IPL-V (références 4-5 et 4-6). On définit ainsi très facilement un certain nombre de niveaux, relativement aux noeuds et aux branches d'un arbre. Chaque noeud porte le rang du niveau auquel il se place, ce qui permet de faciliter les recherches. Ces structures sont particulièrement commodes lorsqu'on a de gros problèmes de suppression ou d'adjonction d'éléments. Elles ont en revanche un grave défaut, d'être à sens unique : on ne peut pas remonter des feuilles aux niveaux supérieurs. Ceci est très malcommode pour résoudre certains problèmes de contraintes.

On peut alors utiliser la structure plus complète, employée par SLIP (référence 4-8). Chaque élément comporte non seulement l'adresse de l'élément suivant de la liste, mais aussi de l'élément qui le précède . On parle

alors de structure symétrique. On peut donc à partir de tout élément retrouver tous les éléments le précédant et tous ceux qui le suivent. Les recherches sont cependant fort longues, et la place nécessitée par ce mode de représentation est assez grande.

* Structures en anneau.

Une amélioration a été apportée par le système de représentation à l'aide d'anneaux. Un anneau est une chaine de données qui se referme sur elle-même. Il y a deux éléments principaux dans une structure en anneau : l'élément de jonction, qui permet de passer l'anneau à un autre de rang inférieur, et l'élément d'anneau qui est composé d'un certain nombre de renseignements(pointeurs, données).

L'élément de jonction contient l'adresse du premier anneau de niveau supérieur et l'adresse du premier anneau de niveau inférieur. L'élément d'anneau contient l'adresse du prochain élément de l'anneau et l'adresse de l'élément précédent. Le dernier élément de l'anneau contient l'adresse du premier élément ce qui ferme l'anneau. L'élément d'anneau peut contenir une figure.

Cette structure est suffisamment souple pour permettre d'appliquer de nombreuses contraintes aux figures. Il est par contre difficile de regrouper les figures en tenant compte des contraintes appliquées. Ceci peut être utile dans le cas où l'on veut appliquer une certaine contrainte à plusieurs images indépendantes et qu'on se propose d'étudier les effets de la variation de certains paramètres de la contrainte sur l'ensemble des figures.

* Tables de corrélation.

Le problème que nous venons d'évoquer est parfois résolu à l'aide de tables de corrélation. Ceci veut dire que lorsque l'on veut appliquer une

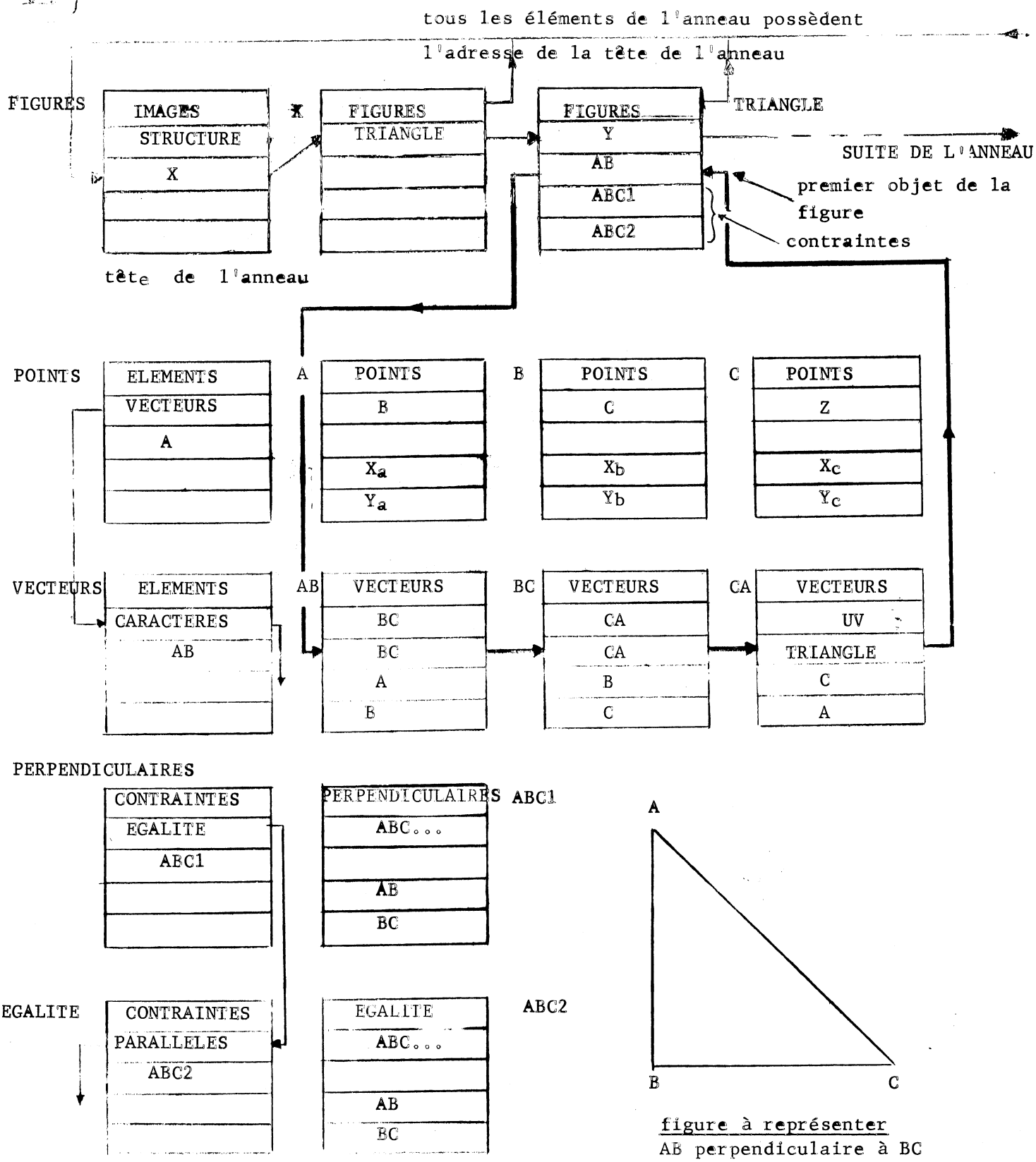


figure à représenter

AB perpendiculaire à BC

AB de même longueur que BC.

seul l'anneau définissant la figure est représenté.

EXEMPLE DE STRUCTURE EN ANNEAU

contrainte à une image, on met l'adresse de cette image dans une table attachée à la fonction qui définit la contrainte. On peut ainsi retrouver très facilement toutes les images auxquelles on a appliqué une certaine contrainte, et on peut effectuer toutes les modifications voulues en cas de changement de certains paramètres de la contrainte. L'inconvénient de cette méthode est l'inverse de la structure en anneau. Il est difficile de remonter des éléments aux contraintes.

B. LE SYSTEME DE DESSIN GENIAL, VU DU COTE DE LA CONCEPTION

1) Principes généraux.

* Spécifications demandées.

Il s'agissait de créer un système qui permette d'employer de façon simple le matériel décrit au chapitre I. Le fait que ce matériel soit assez élémentaire nous a conduit à étudier une certaine structure d'image permettant un nombre restreint d'opérations. Les limitations technologiques et le fait que l'on désirait avoir en permanence une image sur l'écran nous ont conduit à éliminer d'office toutes les opérations nécessitant de gros calculs, comme les vues perspectives et les déformations.

Il fallait aussi que ce système permit d'utiliser au mieux le crayon optique et la console.

* Caractéristiques du système.

Le système de dessin est composé de trois parties : le superviseur, l'interprète et le programme de traitement des interruptions. L'interprète est celui du langage LAGROL, légèrement modifié de manière à pouvoir utiliser le champ I de la mémoire. Le superviseur décode les commandes, les exécute et donne le contrôle à l'interprète pour afficher l'image désirée. Il est écrit à l'aide d'un grand nombre de sous-programmes très spécialisés, ce qui a permis d'écrire les séquences de traitement des commandes de façon systématique et très agréable. Toutes les commandes se font en mode ION, ce qui permet d'utiliser un temps maximum pour le programme interprète. Le superviseur utilise la console et le crayon optique pour les commandes. Toutes les autres interruptions sont ignorées. Il est très facilement extensible, ce qui est un avantage certain.

* Organisation de la mémoire.

N'ayant aucune limitation de place à respecter, nous avons utilisé toute la mémoire du PDP-8, en séparant bien les différents éléments.

CHAMP 0

traitement des interruptions
GENIAL
LAGROL

- ce programme reconnait et traite les interruptions.
- cette zône contient les sous-programmes de traitement du superviseur, les tables de décodage des commandes, la liste des fonctions d'animation et le sous-programme de mise en place des appels de fonctions d'animation.
- cette zône contient le programme interprète de LAGROL. Ce programme trouve ses données en champ I. Les tables de caractères sont en champ II.
- petit programme permettant d'appeler le système en mémoire.

CHAMP I

liste des figures
liste d'affichage
liste de génération

- dessin d'une figure en cours de génération.
- séquence permettant d'utiliser un dessin sauvegardé.
- cette liste contient les adresses de toutes les figures. déjà générées et celle de la figure en cours de génération.
- zône de sauvegarde des pointeurs dans le cas d'une préservation.
- cette zône contient les appels de sous-programmes formant une image.
- cette zône contient la représentation de toutes les figures.

CHAMP II

initialisation
caractères
table des messages
fonctions d'animation

- cette séquence met en place les tables de caractères, les tables des messages et les fonctions d'animation.
- table permettant de dessiner les caractères sur le terminal.
- utilisée par le superviseur pour lancer tous les messages.

Le champ 0 contient essentiellement un programme de traitement des interruptions, le superviseur et l'interprète LAGROL. Le programme de traitement des interruptions est celui de l'interprète LAGROL. Il est chargé de reconnaître les interruptions venant de la console ou du crayon optique et de donner le contrôle aux sous-programmes dont les adresses sont dans GSPCRA, GSPCLA ou GSPTEL. Ces trois registres sont chargés par le superviseur.

L'interprète LAGROL est celui qui a été défini au chapitre II. Il a été légèrement modifié pour des questions de gain de place. Il trouve ses données en champ I et ses tables de caractères en champ II. D'autre part, l'utilisateur n'a plus la possibilité d'utiliser les options relatives au crayon optique, le soin de les choisir étant confié au superviseur.

Le superviseur est en fait formé d'un grand nombre de sous-programmes. La structure du PDP-8 se prête bien à cette utilisation. Le fait d'avoir défini des sous-programmes très spécialisés a permis alors une écriture rapide des séquences de traitement des différentes commandes. Leur définition de façon très systématique rend ainsi la lecture du programme relativement aisée. Le superviseur inclut des tables de décodage des commandes et la liste des fonctions d'animation.

Le champ I contient essentiellement les données destinées à l'interprète LAGROL. La liste des figures va contenir toutes les adresses des figures générées, ainsi que celle de la figure en cours de génération. Cette liste permet de construire la liste d'affichage qui contient des appels de sous-programmes qui font référence indirectement à la figure concernée à l'aide de la liste des figures. La liste de génération contient la représentation en LAGROL de toutes les figures générées ainsi que la représentation de la figure en cours de génération.

Le champ II contient un programme qui permet de mettre le système en route, la table des caractères utilisée par LAGROL, la table des textes de commande et des messages d'erreur et les fonctions d'animation. Elles ont été mises en champ II de façon à avoir assez de place pour que l'utilisateur en définisse de nouvelles à son gré.

* Fonctionnement du superviseur.

En mode de génération, le superviseur fait exécuter à l'interprète la séquence située au début du champ I, qui dessine uniquement l'image en cours de génération. A tout instant, on trouve dans différents registres les renseignements suivants :

CONTE1 adresse du prochain mot de la liste de génération à utiliser

CONTE2 adresse du dernier objet entièrement généré.

CONTE3 adresse du début de la figure.

CONTE4 adresse de la prochaine mémmoire à utiliser dans la liste d'affichage.

ADRESS contient l'ordre correspondant au retour de sous-programme. Ce registre sert à générer le retour de sous-programme afférent à la figure en cours de génération.

NBIMAG nombre de figures générées et position dans la table des figures.

PSW cette mémoire sert à lancer une demande de renseignements. Lorsque pour achever une commande on a besoin d'un paramètre, on met l'adresse de la suite du traitement dans cette mémoire et on va afficher l'image. Lorsque la demande est satisfaite, le contrôle est redonné à l'adresse contenue dans ce mot.

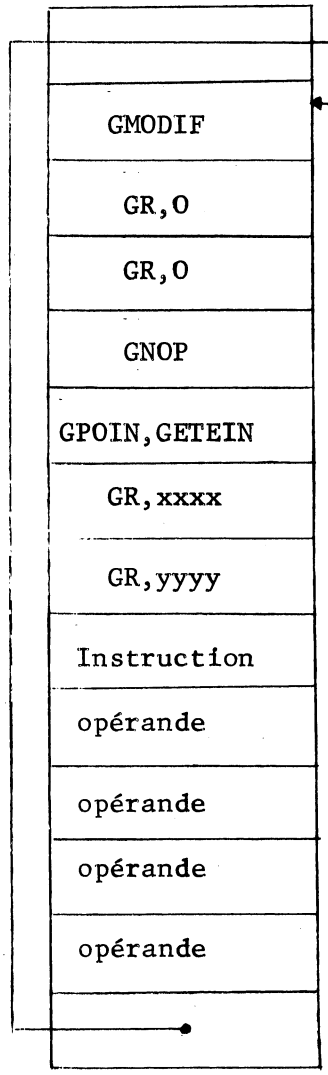
En mode d'affichage, le superviseur fait exécuter le programme contenu dans la liste d'affichage. On voit donc que la structure utilisée est une structure de table. Elle a été choisie pour des raisons technologiques. Les mots du PDP-8 étant trop petits, il eût été aberrant de songer à une structure de liste qui aurait pris trop de place.

2) Organisation générale des images.

Nous allons étudier en détail la structure des listes de génération et d'affichage ainsi que la façon de traiter les demandes d'utilisation du crayon optique. Un certain nombre de conventions graphiques sont utilisées. Il est demandé au lecteur de bien vouloir regarder en bas des schémas les indications éventuelles.

STRUCTURE DE LA LISTE DE GENERATION.

Figure simple (un objet)



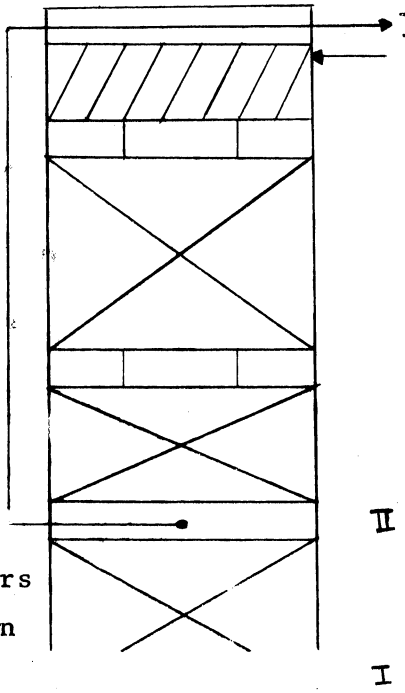
cette mémoire contient l'adresse de retour du sous-programme.

cet ensemble de trois mémoires va permettre de modifier le système de référence de la figure.

coordonnées du point de départ.

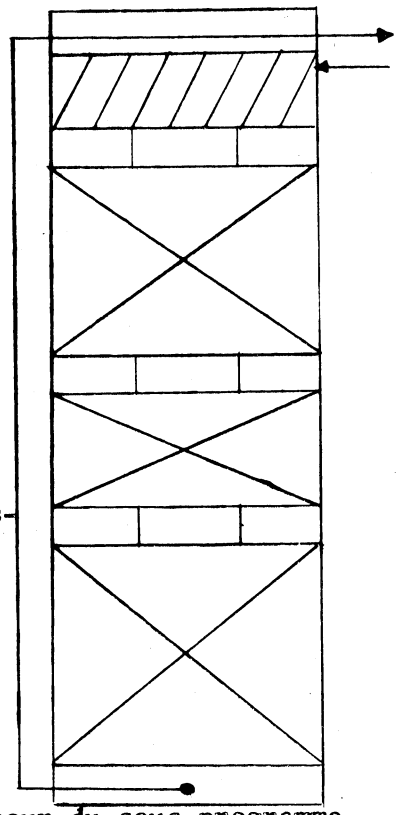
retour du sous-programme.

Figure composée en cours de génération



objet en cours de génération

Figure composée entièrement générée

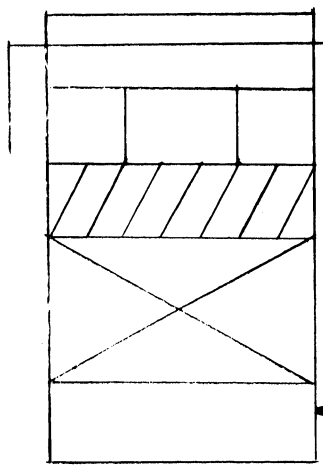


le retour de sous-programme a été remplacé par un GNOP.

retour du sous-programme

I, II, III

Symbolisme utilisé



saut à l'adresse contenue dans cette mémoire

GNOP

GMODIF et ses deux opérandes

objet

rupture de séquence vers la mémoire désignée.

I , II , III , IV contenu de CONTE1, CONTE2, CONTE3, CONTE4

* Structure de la liste de génération.

Elle contient des figures définies sous forme de sous-programmes. Chaque figure est contenue dans une page de PDP-8. On distingue entre figures simples, formées d'un seul objet, et figures composées, formées de plusieurs objets.

- structure d'une figure simple.

La première mémoire contiendra l'adresse d'appel du sous-programme. Le retour de sous-programme s'effectue par rupture de séquence indirecte à travers cette mémoire. Les trois mémoires suivantes sont la représentation d'une instruction GMODIF et de ses deux opérandes. Les deux opérandes sont définis comme relatifs. Le but de ces trois mémoires est de faciliter les opérations de déplacement. On trouve ensuite une mémoire contenant un GNOP. Le GNOP est utilisé comme repère et comme séparateur lors de la définition des figures. Derrière ce GNOP commence la définition de l'objet. Un GPOIN, GETEIN suivi de deux coordonnées donne le point de départ du tracé. On trouve ensuite l'instruction de tracé, suivie par un certain nombre d'opérandes, variable. La dernière mémoire contient l'instruction de retour de sous-programme.

- Structure d'une figure composée.

Une figure composée est formée de plusieurs objets séparés entre eux par des GNOP. La tête de la figure est la même : une modification des paramètres de l'instruction GMODIP de tête permet de déplacer en bloc tous les objets de la figure. Lorsqu'une figure est en cours de génération, l'instruction qui suit le dernier objet entièrement généré est le retour de sous-programme. Ceci permet d'avoir en permanence sous les yeux la portion de figure déjà créée. Lorsque l'objet en cours de création est terminé, on génère un retour de sous-programme à la suite et le précédent retour de sous-programme est remplacé par un GNOP, qui assure ainsi le lien entre le début de la figure et le dernier objet créé. Le point de départ de la figure est le point de départ du premier objet de la figure. On voit donc que l'on ne peut déplacer une figure qu'en entier, et non pas un des objets qui la composent.

* Structure de la liste d'affichage.

Pour obtenir une image, l'interprète LAGROL parcourt cette liste jusqu'au dernier élément et retourne au premier jusqu'à ce qu'il soit arrêté par le superviseur. Au départ, la liste contient quatre GNOP. La tête de la liste sera toujours formée ainsi, afin d'éviter tout danger d'erreur de la part de l'utilisateur, lors de l'effacement d'une image. Il y en a quatre parce que l'unité de base de la liste d'affichage est formée de quatre mémoires, qui contiennent une instruction GMODIF, ses deux opérandes définis de façon absolue et un appel de sous-programme vers la figure à dessiner. Cet appel de sous-programme se fait par référence à la liste des figures. L'instruction GMODIF se fait avec des opérandes absolus car il s'agit là de changer le système référentiel de la figure à dessiner, mais il faut que la figure qui suit soit dessinée indépendemment de la première.

La liste d'affichage formant une image est donc formée de groupes de quatre mémoires, chaque groupe correspondant à l'affichage d'une figure sur l'écran.

* Traitement du crayon optique.

Le traitement du crayon optique est relativement simple. Suivant le mode de travail, on ajoute à la fin de la figure en cours de génération ou à la fin de la liste d'affichage une instruction GARR, qui renvoie alors à l'instruction qui suit l'appel de l'interprète. Cette instruction renvoie à un sous-programme qui dessine la croix, et permet les interruptions dûes au crayon optique. Lorsque la croix est dessinée, on retourne au dessin primitif. On voit donc qu'on s'est contenté de rajouter un objet à la liste de tous ceux qui sont affichés sur l'écran. Ceci explique pourquoi dans le cas de dessins complexes la croix est lente à mouvoir. Le sous-programme vérifie avant de rendre le contrôle au programme de dessin qu'on n'a pas montré le point d'arrêt. Si tel est le cas, il donne alors le contrôle à l'instruction dont l'adresse se trouve dans le registre PSW. On trouve dans deux mémoires LATI et LONGI les coordonnées du centre de la croix.

Lorsque le travail est terminé, l'instruction GARR est remplacée soit par un retour de sous-programme, soit par une rupture de séquence vers le début de la liste d'affichage.

Dans le cas où l'on n'a pas besoin de la croix, on se contente de mettre dans le registre GSPCRA l'adresse du sous-programme d'identification de la figure repérée. On obtient ainsi une facilité très grande pour traiter tous les cas d'utilisation du crayon optique.

3) Traitement des commandes.

Nous ne traiterons pas ici toutes les commandes, mais seulement celles qui ont le plus d'importance et montrent comment peut être manipulée la structure définie.

* Commandes de génération.

- GENERER

Cette commande provoque la mise en place de la tête d'une nouvelle figure, à savoir :

une première mémoire nulle pour les adresses de retour
un ordre GMODIF
deux opérandes relatifs nuls GR,0
un GNOP

L'adresse de la première mémoire de la figure (CONTE3) est placée à la suite des autres dans la liste des figures. Le registre ADRESS est chargé de l'instruction de retour de sous-programme correspondant à cette figure.

- CONSERVER

La seule chose à faire est d'augmenter le nombre de figures (NBIMAG) de une unité, ce qui fait progresser automatiquement dans la liste des figures.

- DETRUIRE

On ramène la valeur des registres CONTE1 et CONTE2 à la valeur de CONTE3. La place occupée précédemment par la figure en cours de génération est ainsi totalement récupérée.

- AJOUTER

On recopie la liste des objets définissant la figure à ajouter à la suite du dernier objet généré de la figure en cours de génération. On génère ensuite un retour de sous-programme et on remplace le dernier par un GNOP.

* Commandes d'utilisation.

- AFFICHER

Connaissant le numéro de la figure à afficher, on va chercher les coordonnées du point de départ. On calcule alors la correction nécessaire pour amener la figure à la position demandée. On génère les quatre instructions qui permettent d'ajouter la figure à l'image de la liste d'affichage.

- DETRUIRE

On ramène CONTE4 au début de la liste d'affichage. L'interprète tourne alors sur la séquence des quatre GNOP consécutifs.

- CONSERVER

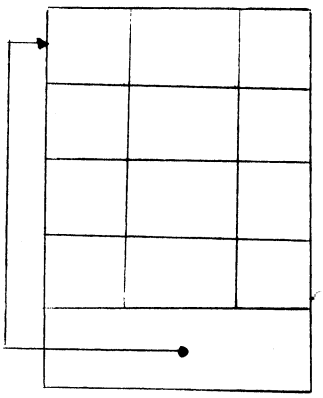
On crée une nouvelle figure dans la liste de génération, en commençant par en fabriquer la tête. Puis on recopie la liste d'affichage à la suite de ce chapeau en changeant tous les opérandes absolus en opérandes relatifs. Si cette précaution n'était pas prise, on ne pourrait pas utiliser à nouveau cette figure en plusieurs emplacements sur l'écran.

* Commandes diverses.

- LOCALISER

On met le crayon optique en fonction, ainsi que la croix. Lorsque l'utilisateur montre le point d'arrêt, on se contente de faire écrire le contenu de LATI et LONGI.

STRUCTURE DE LA LISTE D'AFFICHAGE

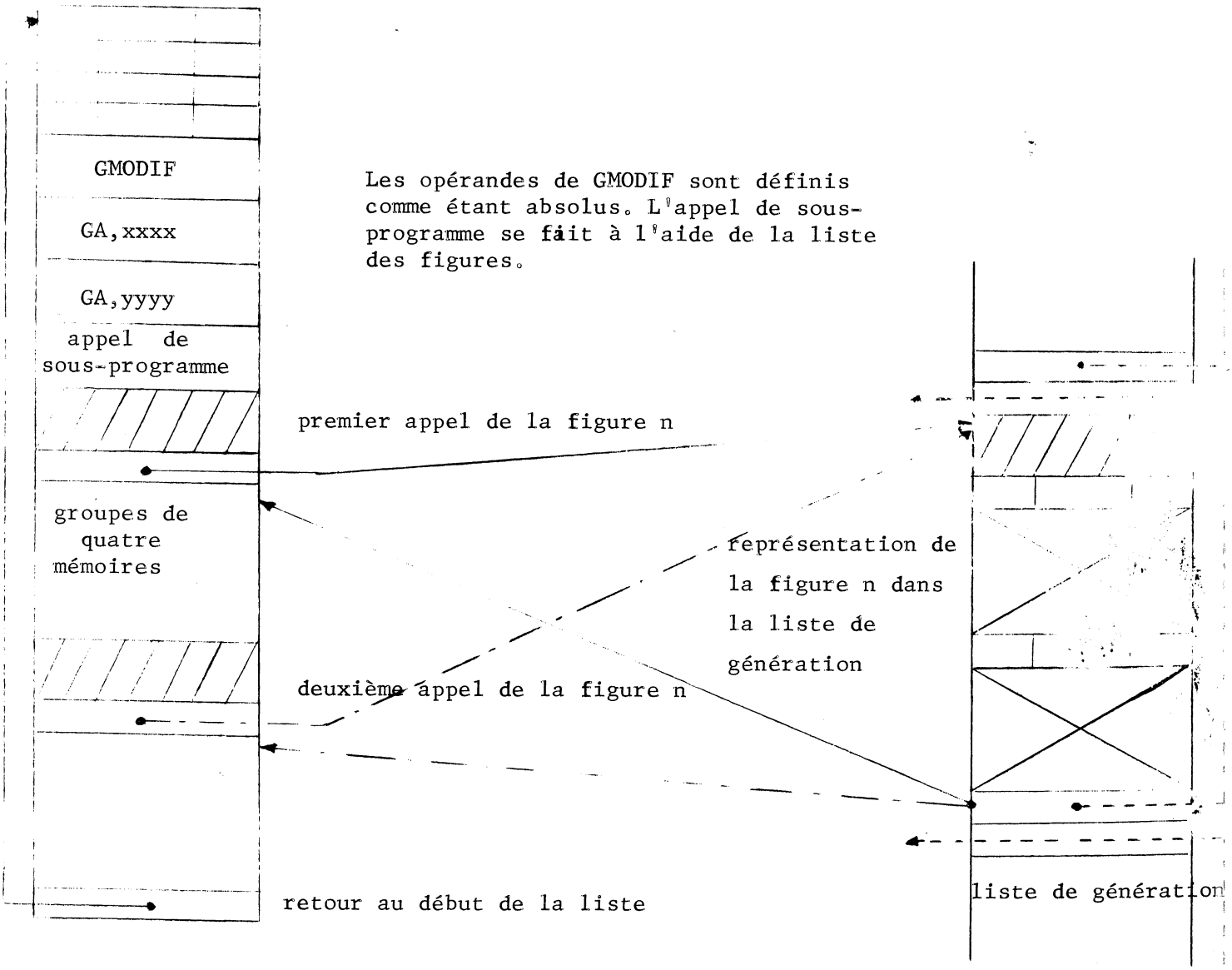


liste d'affichage sans image

La liste est composée de quatre GNOP pour éviter les erreurs de manipulation.

retour au début de la liste

liste d'affichage avec une image



Les opérandes de GMODIF sont définis comme étant absolus. L'appel de sous-programme se fait à l'aide de la liste des figures.

-PRESERVER

On donne le contrôle à un programme situé en champ II, qui se charge de vider le champ I en champ 0 après avoir sauvegardé les registres essentiels en champ I. On fait alors appel au chargeur pour mettre le champ 0 sur bande.

- NOMMER

On se contente de mettre le crayon optique en fonction. Lors de l'interruption, on obtient l'adresse du mot qui était en cours de traitement ce qui permet de trouver de quelle figure il s'agit en consultant la table des figures.

- DEBUT

Les registres CONTE1, CONTE2, CONTE3 sont chargés avec l'adresse de début de la liste de génération. Le registre CONTE4 est chargé de l'adresse de début de la liste d'affichage (après les quatre GNOP). On se met en mode d'utilisation et en lecture de nombres en octal. Le registre PSW est chargé de l'adresse du sous-programme de décodage des commandes.

* Manipulations d'images.

- DEPLACER

Dans le cas de DEPLACER avec la console, le numéro de la figure permet de trouver les opérandes de l'instruction GMODIF de tête. On se contente alors de changer les valeurs de ces opérandes en fonction des modifications données par l'utilisateur. On modifie donc la figure de base, ce qui fait que toutes les occurences de cette figure sont déplacées en même temps.

Si l'on utilise le crayon optique, deux cas se présentent : soit l'on est en mode de génération, soit l'on est en mode d'utilisation. Dans le cas de mode de génération, on calcule les coordonnées du point de départ en tenant

compte des corrections éventuelles du GMODIF de tête. On fait apparaître la croix jusqu'à ce que l'utilisateur indique le point d'arrêt. On calcule alors les corrections nécessaires pour déplacer la figure et on les donne dans les opérandes du GMODIF de tête.

Dans le mode d'utilisation, on calcule de même les coordonnées du point de départ, mais on va chercher quelle est l'occurence de la figure que l'on désire déplacer. Ceci est facile puisque l'on a l'adresse d'appel dans la tête de la figure. Lorsque l'utilisateur indique le point d'arrêt, on change alors les opérandes de l'appel de sous-programme . La figure de base n'est pas modifiée.

- EFFACER

Si l'utilisateur demande d'effacer une figure à l'aide de la console, le système génère un retour de sous-programme à la place du GMODIF de tête. On voit donc qu'il n'y a pratiquement pas de temps perdu si l'on revient afficher cette figure.

Si l'on a utilisé le crayon optique, on va générer l'option GETEIN dans l'instruction qui décrit l'objet à éteindre. Les autres objets sont toujours allumés. A noter que le temps pris pour décrire l'image en entier n'est pas diminué.

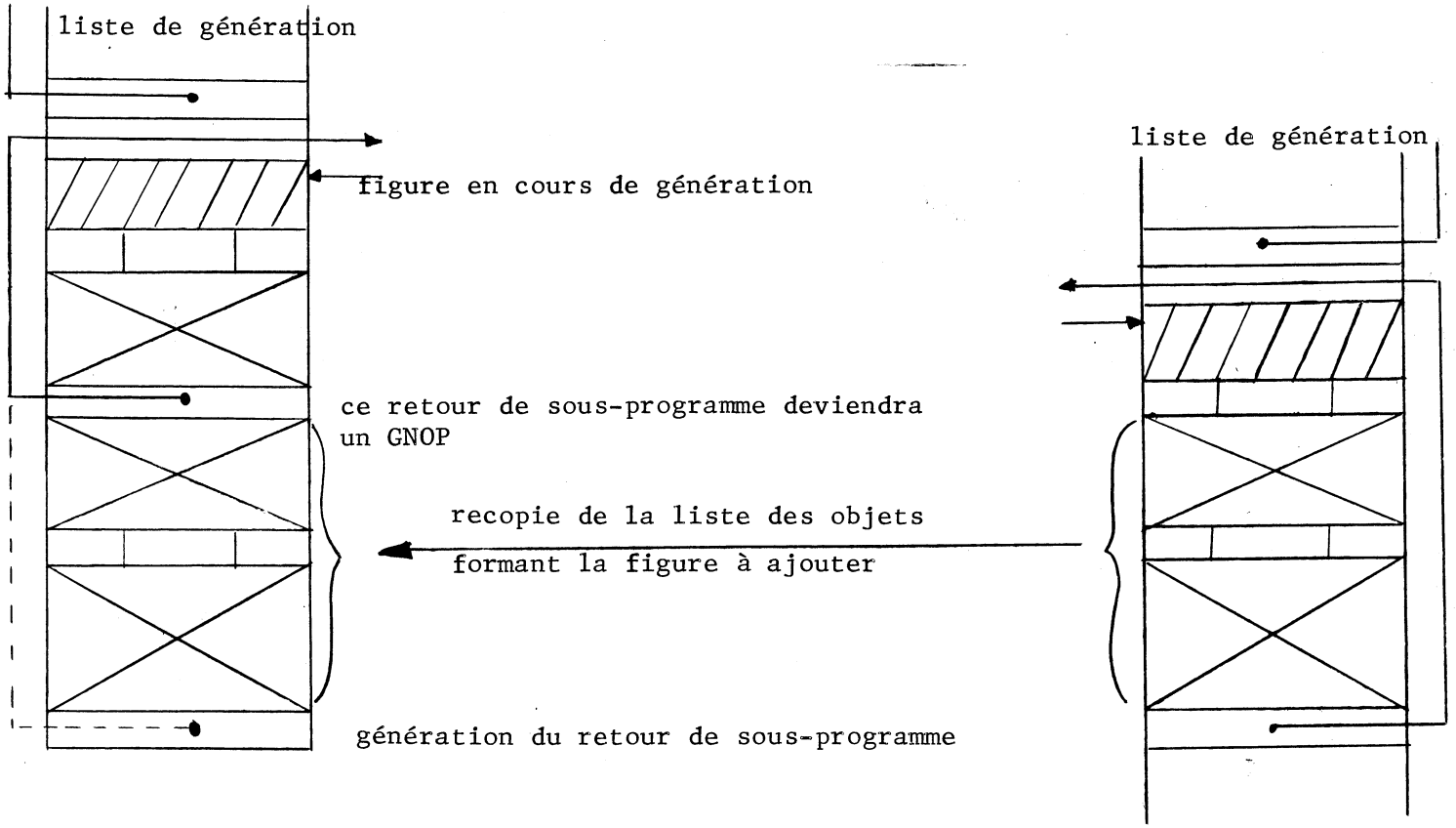
- RALLUMER

On vérifie que la première instruction est un GMODIF. Si non, on remet un GMODIF et le travail se termine. Si c'était un GMODIF, on parcourt toute la figure à la recherche des instructions et on remet des options GALUM.

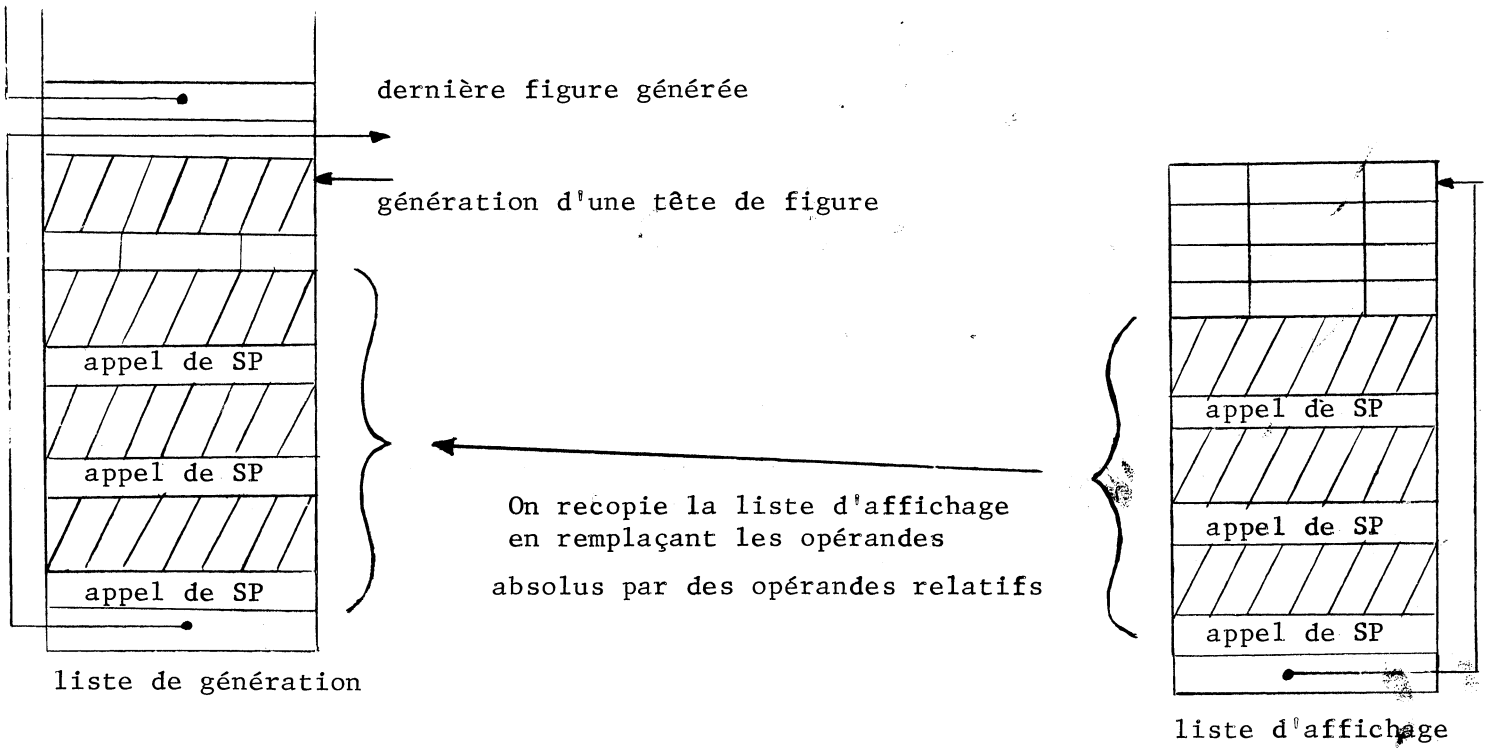
* Traitement des erreurs.

Il est très simple. Si l'on est en mode de génération, on ramène CONTE1 à la valeur de CONTE2, c'est-à-dire derrière le dernier objet correctement généré. Dans le cas du mode d'utilisation on ramène CONTE4 à une valeur multiple de 4.

TRAITEMENT DE AJOUTER

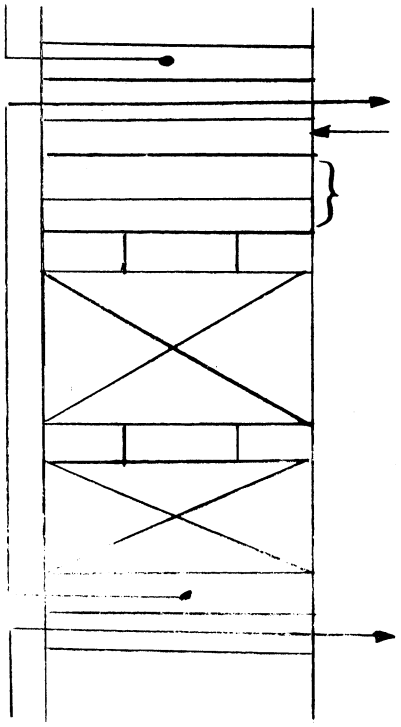


TRAITEMENT DE CONSERVER EN MODE D'UTILISATION



TRAITEMENT DE DEPLACER

liste de génération



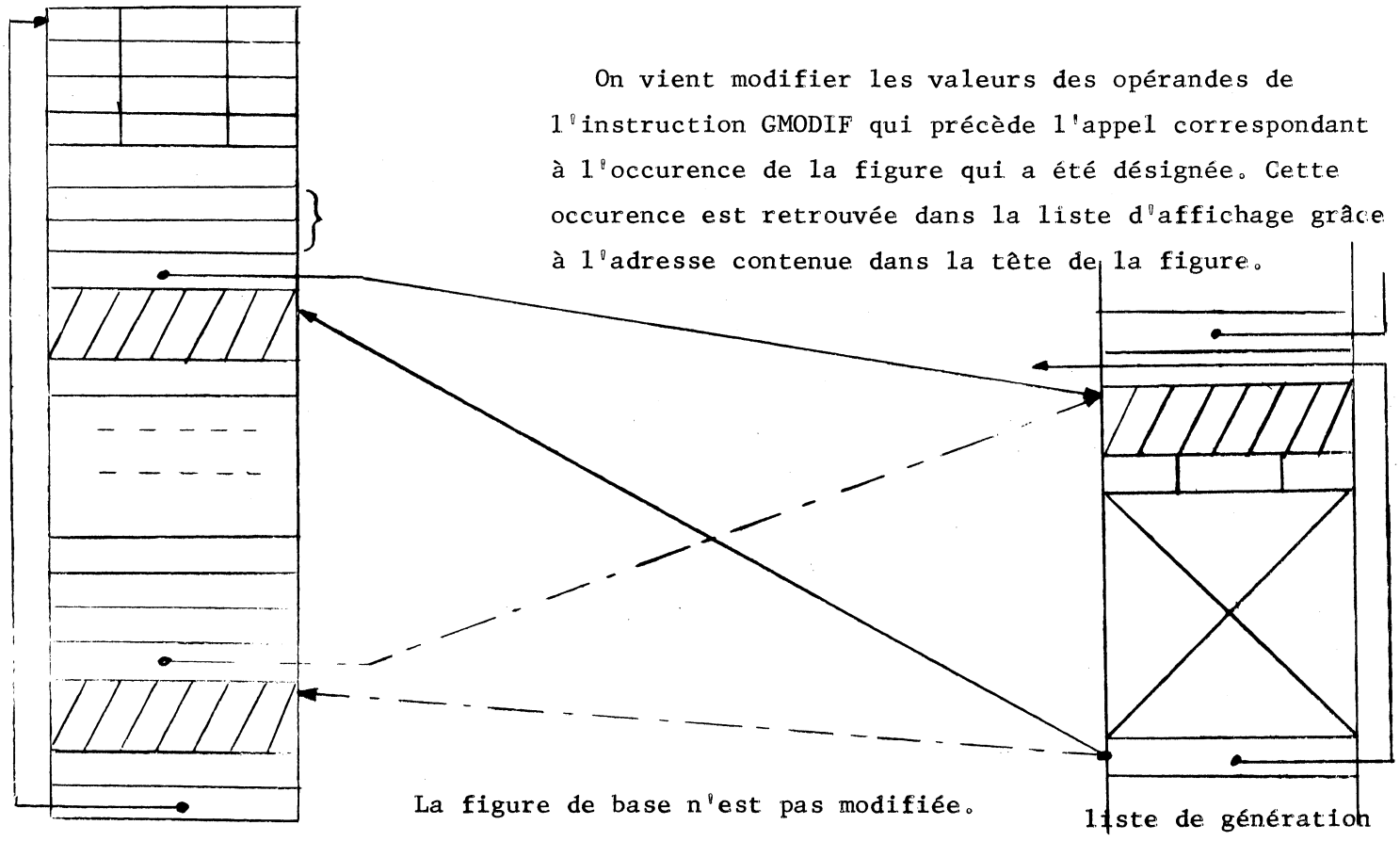
utilisation de la console

On vient modifier les valeurs des opérandes de l'instruction GMODIF qui se trouve en tête de la figure.

La figure de base est donc modifiée.

utilisation du crayon optique

liste d'affichage



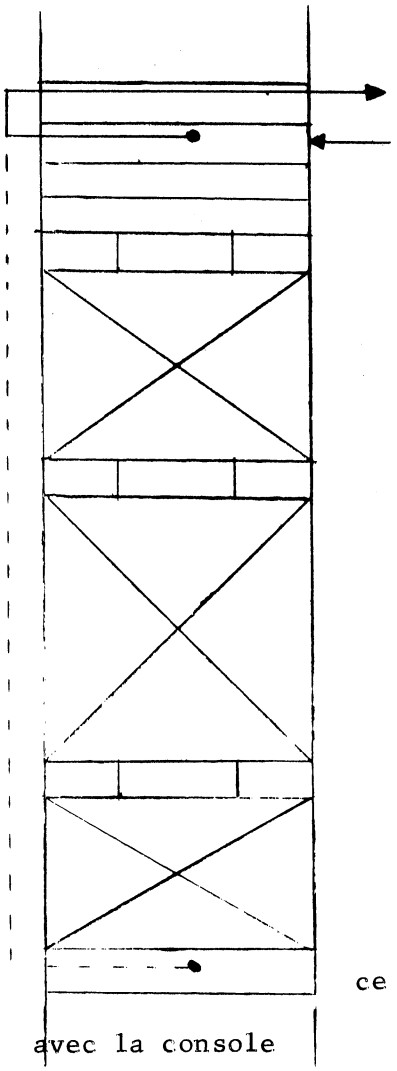
On vient modifier les valeurs des opérandes de l'instruction GMODIF qui précède l'appel correspondant à l'occurence de la figure qui a été désignée. Cette occurence est retrouvée dans la liste d'affichage grâce à l'adresse contenue dans la tête de la figure.

La figure de base n'est pas modifiée.

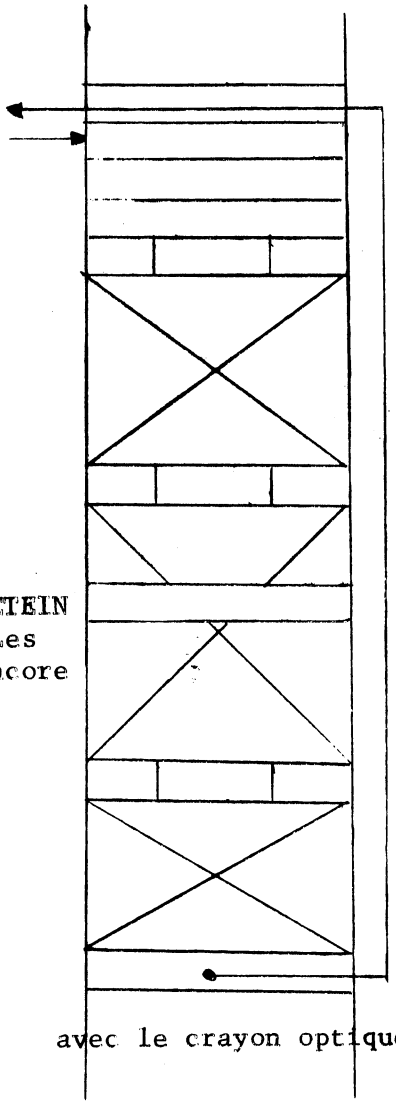
liste de génération

XII

TRAITEMENT DE EFFACER



on génère un retour immédiat.
La figure n'apparaît plus.



On génère l'option GETEIN dans l'instruction. Les autres objets sont encore allumés.

ce retour de sous-programme ne sert plus

avec la console

avec le crayon optique

TRAITEMENT DE RALLUMER

On vérifie que la première mémoire de la figure contient GMODIF. Si non, on remet l'instruction GMODIF et on s'en va. Si oui, on parcourt toute l'image en cherchant les instructions. On remet l'option GALUM dans toutes les instructions.

* Adjonction de nouvelles commandes.

Il y a quelques règles à observer :

- Il faut compléter la liste des commandes, la liste de décodage des commandes et la liste des textes de commandes.
- Il faut augmenter le nombre de commandes d'une unité.
- Il faut mettre la séquence de traitement de cette nouvelle commande en champ 0.
- Toute commande doit se terminer par un appel au sous-programme qui permet de préparer la prochaine commande.

Il nous reste à étudier la structure des images animées, structure légèrement différente mais utilisant les mêmes règles.

4) Organisation des images animées.

Le fait de pouvoir disposer d'images animées ne modifie pas la structure générale des images. Il est simplement fait appel à une des possibilités de LAGROL, à savoir la faculté d'inclure dans la définition d'un objet, un appel à une fonction. Ceci est réalisé grâce à l'instruction GAPPEL, décrite en détail au chapitre II. Le problème de l'animation des images est relativement simple. Il suffit en effet de modifier les contenus de GRELX et de GRELY avant l'affichage de l'objet et en suivant une certaine loi pour obtenir le mouvement désiré. La petite difficulté vient du fait que ces manipulations doivent se faire indépendamment des autres figures. Il faut donc préserver puis restaurer GRELX et GRELY avant et après chaque affichage d'une figure animée. Ces quelques réflexions préalables permettent de comprendre la représentation choisie.

* Structure d'une figure animée.

L'idée est donc d'inclure avant la définition de l'objet à animer un appel à une certaine fonction d'animation. Cette fonction d'animation a besoin de plusieurs renseignements, qui lui sont fournis à la suite de l'instruction GAPPEL. Ces renseignements sont composés d'une partie fixe, composée de six mémoires, et d'une partie variable pouvant être inexistante. La partie fixe

est ainsi composée : la première mémoire contient la valeur du pas de déplacement en X. La deuxième contient soit 0, soit la valeur précédente. La troisième contient le déplacement total déjà effectué en X. Les trois mémoires suivantes contiennent les mêmes renseignements, appliqués cette fois au déplacement en Y. Les deux mémoires qui contiennent soit 0 soit la valeur du pas ont un rôle particulier. C'est leur contenu qui est ajouté au déplacement total en X(contenu de GRELX) ou au déplacement total en Y (contenu de GRELY). On voit donc que tant que ces mémoires contiennent 0, la figure est immobile. Dès que ces mémoires contiennent une valeur différente de 0, il y a mouvement. La partie variable contient un certain nombre de paramètres, suivant la fonction d'animation. Ce peut être par exemple le chemin à parcourir dans un sens le long de l'axe des X dans un mouvement alternatif.

A la suite de l'appel de la fonction d'animation se trouve la représentation de l'objet à animer . Il peut s'agir en fait de plusieurs objets, qui sont ainsi astreints au même mouvement. Le seul problème est de remettre dans GRELX et GRELY les valeurs initiales, c'est-à-dire celles qui ont été modifiées par la fonction d'animation. Ceci est fait par un appel à une nouvelle fonction qui se contente de prendre le contenu de deux mémoires chargées des valeurs de GRELX et de GRELY par la fonction d'animation avant d'effectuer le mouvement et de remettre ainsi les bonnes valeurs. On appelle champ d'application d'une fonction d'animation la liste d'objets se trouvant entre un appel de fonction d'animation et l'appel de la fonction de la remise en ordre. Il faut noter qu'il s'agit là très exactement d'une structure parenthésée. Si dans une figure il y a des objets devant avoir des mouvements différents, ces objets seront compris entre des couples d'appels de fonction, un appel de fonction d'animation et un appel de fonction de remise en ordre, ces appels pouvant être imbriqués.

* Structure d'une fonction d'animation.

Les fonctions d'animation devront être écrites en suivant un certain nombre de règles, afin de permettre le bon fonctionnement de l'interprète.

XIII

STRUCTURE DES FIGURES ANIMEES

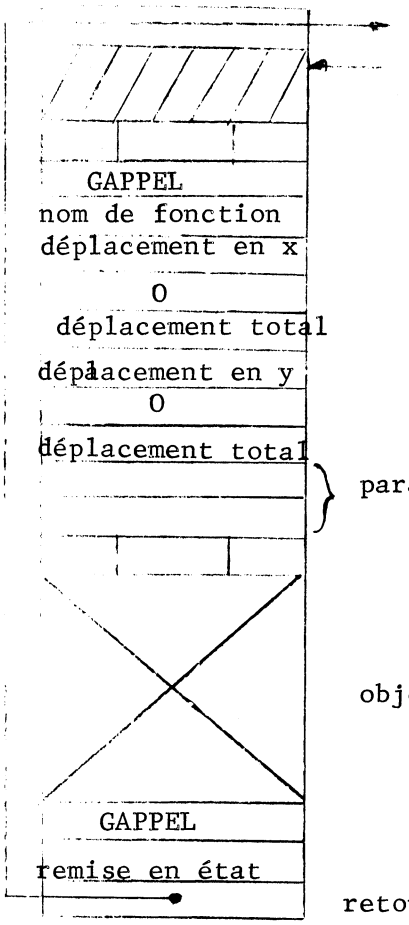


figure simple animée

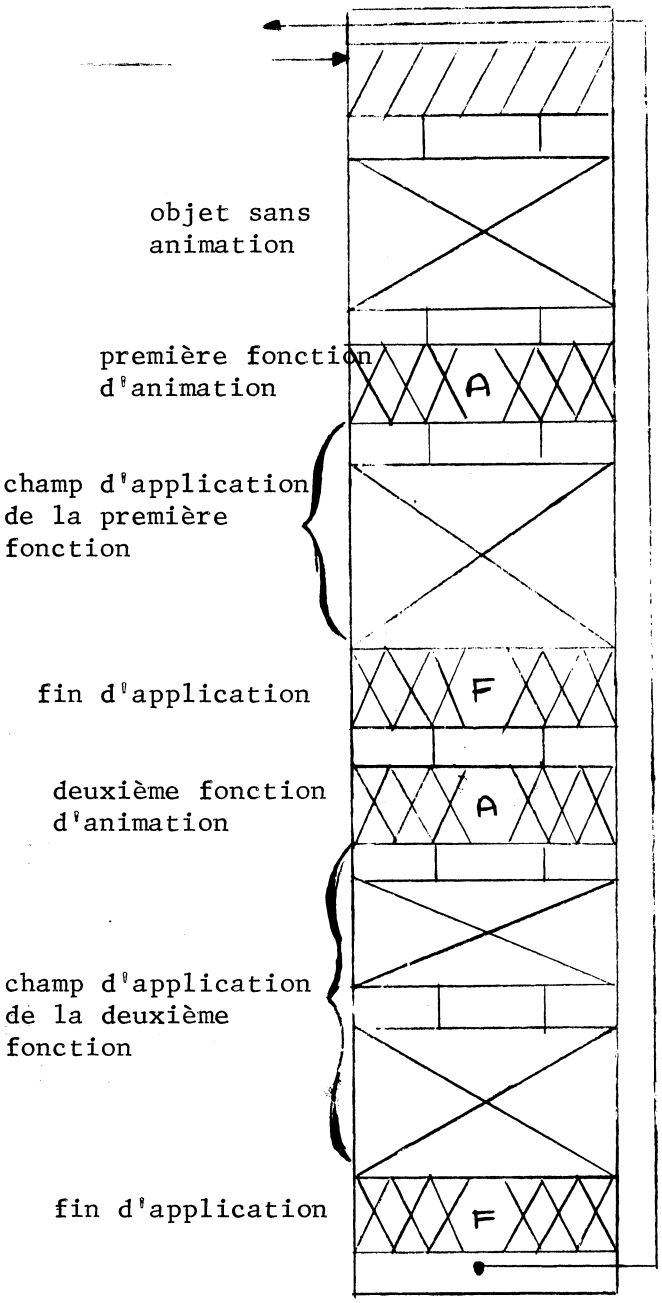
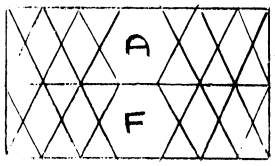


figure composée animée

symbolisme employé



appel d'une fonction d'animation

appel de la fonction remettant les valeurs initiales de GRELX et GRELY (fin d'animation).

La première chose à faire est de garer le contenu des deux mémoires GRELX et GRELY. Ceci étant fait, le calcul du prochain point de déplacement peut se faire. Ce qui est très important, c'est qu'il ne faut pas oublier de faire progresser en même temps le compteur ordinal de l'interprète, puisque les paramètres des fonctions d'animation sont des mots d'un programme écrit en LAGROL. D'autre part, la dernière instruction d'une fonction d'animation doit être un retour à l'interprète, afin que celui-ci puisse continuer l'affichage de l'image. Pour des raisons de place en mémoire, les fonctions d'animation sont placées en champ II, ce qui pose de petits problèmes de commutation de champ. L'écriture de fonctions d'animation est donc une chose relativement délicate. Un certain nombre de sous-programmes spécialisés sont fournis, pour alléger la tâche du programmeur. La seule règle à suivre impérativement quant aux paramètres des fonctions est qu'il faut donner un pas de déplacement pour les commandes d'animation. Autrement le programmeur n'est limité que par la place en mémoire. En effet, définir une image animée est une opération très coûteuse à cause du grand nombre de mémoires de travail nécessaire. Mais on peut imaginer tous les mouvements possibles, et disposer ainsi d'une grande facilité de dessin, à condition de bien vouloir enrichir constamment la bibliothèque des fonctions d'animation.

* Adjonction de nouvelles fonctions d'animation.

Il suffit de faire un petit nombre de choses :

- Ajouter le nom de la fonction dans la liste des fonctions
- Augmenter le nombre des fonctions d'une unité
- Ecrire le programme qui permet de lire les paramètres de la fonction lors de la génération. Ce programme doit être en champ 0. Un sous-programme de lecture de paramètres est fourni, auquel il suffit de donner la longueur de la partie variable.
- Ecrire la fonction d'animation en suivant les règles énoncées plus haut. On voit qu'il est relativement facile d'enrichir la bibliothèque, la seule difficulté résidant dans l'écriture de la fonction d'animation elle-même.

* Traitement des commandes d'animation.

- APPLIQUER

Cette commande permet de générer l'appel de la fonction d'animation. Le superviseur donne le contrôle à une séquence d'instructions qui génère l'ordre GAPPEL, le nom de la fonction et la représentation des différents paramètres.

- FIN D'APPLICATION

Cette commande permet d'indiquer la fin du champ d'application d'une fonction d'animation. Il y a génération de l'appel à la fonction de remise en ordre.

- ANIMER

Lorsque l'utilisateur a montré l'objet à animmer, on vérifie que l'objet se trouve bien dans une liste d'objets à animer. Si oui, on charge les deux déplacements en X et en Y dans les deux mémoires de mouvement.

- ARRETER

On effectue l'opération inverse, c'est-à-dire que l'on va mettre 0 dans les mémoires de mouvement.

5) Conclusion

La structure décrite est en fait assez rudimentaire. Les problèmes de hiérarchie des figures sont imparfaitement résolus, et ne permettent pas de résoudre d'une façon satisfaisante tous les problèmes de déplacement d'images. Cette structure permet cependant d'obtenir des résultats intéressants, si l'on considère que le matériel employé est lui-même très élémentaire. Cette structure a permis l'écriture et la mise au point rapide d'un système de dessin qui nous a conduit à étudier d'une manière très complète les problèmes de programmation graphique. Le bilan de cette expérience est fait au chapitre suivant.

CHAPITRE V

EXEMPLES D'UTILISATION DES CONSOLES DE VISUALISATION.

A. BILAN DE L'EXPERIENCE TENTEE A GRENOBLE.

1) Utilisation de LAGROL.

La définition du langage graphique a permis de mettre en évidence un certain nombre de problèmes de base. Leur résolution nous a conduit à une connaissance profonde des problèmes de dessin. Le premier problème a été d'entretenir l'image sur l'écran. Nous avons vu que la conclusion tirée est que pour un travail demandant de gros calculs, il est indispensable de disposer d'un terminal possédant une mémoire intermédiaire. Le deuxième problème a été celui du tracé de figures géométriques : tracé de vecteurs, de cercles aussi. Les difficultés à vaincre étaient surtout : mode de calcul et récupération des erreurs. Le troisième problème à résoudre a été l'emploi de sous-programmes, ce qui nous a conduit à donner la possibilité de définir des figures en position relative ou absolue. Le dernier problème a été de donner la possibilité d'inclure des séquences en langage machine dans un programme de dessin.

La résolution de ces différents problèmes nous a permis de disposer d'un certain nombre de sous-programmes de tracé immédiatement utilisables. Il s'agit en particulier des sous-programmes de tracé de caractères et de tracé de vecteurs. Ces sous-programmes ont pu être mis à la disposition des programmeurs, leur évitant ainsi de les réécrire. Deux exemples peuvent être donnés : le sous-programme de tracé de caractères a été utilisé dans le cadre d'un projet de troisième année de l'Institut Polytechnique (programme de correction), et le sous-programme de tracé de vecteurs a été utilisé pour améliorer le tracé de courbes de niveau, résultat d'un programme travaillant sur l'ordinateur IBM 7044.

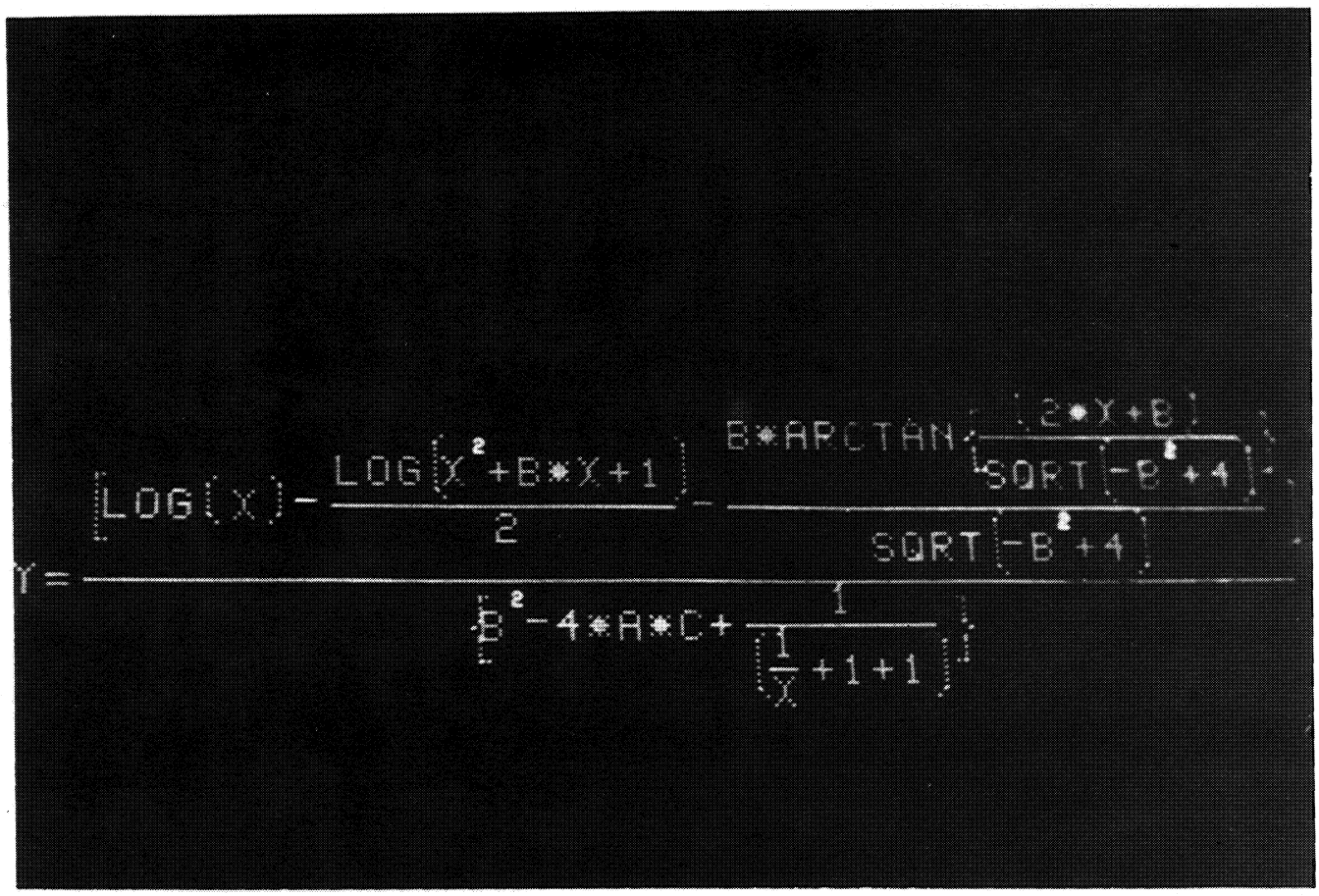
Une autre application de LAGROL a permis d'obtenir sur l'écran la recomposition dans l'espace à deux dimensions de formules écrites sous la forme d'ALGOL. Ceci est fait à l'aide d'un programme en LISP, qui génère le programme en LAGROL correspondant. Ceci a permis de mettre au point la première partie d'un programme destiné à faire du calcul formel à l'aide d'un terminal graphique.

Enfin, la facilité d'assimilation de LAGROL a permis de faire une série de travaux pratiques pour les étudiants en Maîtrise. Nous parlons un peu plus loin de cette partie de l'expérience.

2) Utilisation de GENIAL.

Le premier résultat a été de nous faire connaître les problèmes de structure d'images. Nous avons déjà dit que la structure employée par GENIAL ne permettrait pas de résoudre tous les problèmes de hiérarchie de figures. Cependant, elle permet d'utiliser d'une façon raisonnable un terminal graphique élémentaire. Le fait d'utiliser un système conversationnel nous a permis de découvrir peu à peu un certain nombre de commandes indispensables pour un tel système. Nous disposons ainsi d'un système de dessin permettant de comprendre ce que doit être un dialogue entre l'homme et la machine, c'est-à-dire comment mieux utiliser les possibilités des ordinateurs.

A côté des résultats théoriques, nous avons obtenu un certain nombre de résultats pratiques. Le premier est la possibilité de définir des bandes-symboles, c'est-à-dire de générer des bandes contenant un certain nombre de symboles de base pour dessiner des schémas. Nous possédons actuellement une bande 'symboles électroniques' et une bande 'symboles d'organigrammes', et on peut imaginer toute autre sorte d'application. L'intérêt de ces bandes est qu'on a toujours sous la main un ensemble permettant d'obtenir rapidement des schémas précis et bien dessinés. La possibilité de prendre ensuite des photos de l'écran permet de fabriquer rapidement des documents qui ordinairement sont soit mal dessinés, soit réalisés par un dessinateur industriel et à ce moment là longs à obtenir et coûteux.



EXEMPLE DE FORMULE OBTENUE
A PARTIR D'UNE FORMULE ECRITE EN ALGOL

Le fait même de disposer de la possibilité d'animer les images est aussi intéressant. Le champ d'application de cette possibilité n'est pas encore reconnu. En effet, il dépend beaucoup des fonctions qui seront mises dans la bibliothèque. Il n'est pas interdit de penser que l'on pourra étudier ainsi des mouvements assez compliqués, comme par exemple le mouvement d'une bielle, d'un distributeur, d'un régulateur, etc...

Il est donc certain que l'étude de LAGROL et de GENIAL a permis d'avoir une bonne connaissance des problèmes de base de l'utilisation des consoles de visualisation. Cette connaissance permet d'aborder l'étude de l'utilisation de terminaux plus puissants avec une base solide qui est garante d'un emploi rationnel de ceux-ci. Le fait d'avoir été obligés de résoudre des problèmes d'un niveau aussi bas que le tracé des vecteurs nous permet de mieux comprendre le fonctionnement d'un générateur de vecteurs (par exemple) et par là même nous conduira à mieux utiliser les dispositifs de génération, donc les terminaux graphiques évolués.

3) Utilisation pédagogique.

Le terminal graphique a été reçu au mois de juin 1967. La réalisation rapide de l'interprète LAGROL et du système GENIAL a permis d'envisager de monter une série de travaux pratiques destinés à des étudiants de la Maîtrise d'Informatique, qui ont pu commencer au mois de Mars 1968. Ces étudiants ont suivi un cours complet sur les langages graphiques, qui leur a permis de se familiariser avec les consoles de visualisation. En illustration de ce cours, ils ont eu à étudier le langage LAGROL. On leur a alors demandé d'écrire des programmes de dessin en LAGROL, puis ils ont utilisé le système de dessin GENIAL pour obtenir des images animées au gré de leur imagination. Le but de ces travaux pratiques était de les familiariser avec l'emploi des terminaux graphiques afin de leur montrer l'intérêt de ces matériels et les difficultés d'utilisation. Les étudiants se sont montrés dans l'ensemble très intéressés par ce qu'ils ont vu et fait. Trente-cinq étudiants ont été ainsi formés aux diverses techniques d'emploi des consoles de visualisation, et ceci dans un délai extrêmement court.

B. EXEMPLES D'UTILISATION DES CONSOLES DE VISUALISATION.

1) Utilisation des terminaux alphanumériques.

Le champ d'utilisation est très vaste. En effet, on peut utiliser les terminaux alphanumériques pour toute application qui ne demande que le traitement de pages de caractères. Le problème en fait est de savoir s'il vaut mieux utiliser un terminal alphanumérique ou une console. Trois points peuvent être déterminants dans ce choix

- la vitesse à laquelle une réponse doit être donnée par le terminal
- le besoin ou l'inutilité de posséder une 'copie' sur bande ou carte perforée
- le coût de l'équipement et la charge des lignes téléphoniques reliant les consoles.

Il semble que l'idée fondamentale soit d'utiliser un terminal graphique pour des consultations de fichiers, avec éventuellement la possibilité de modifier de tels fichiers.

* Gestion de stocks.

Un terminal alphanumérique peut être utilisé pour obtenir une gestion de stocks efficace, en évitant en particulier le remplissage de formulaires et les recherches longues dans des fichiers volumineux. On peut très bien imaginer par exemple un terminal alpahnumérique à la disposition du magasinier. Celui-ci peut à tout instant savoir combien de pièces sont en réserve, modifier ce nombre suivant les entrées et les sorties de pièces, connaître les prix, éventuellement même disposer de statistiques sans avoir à consulter de livres ou ensemble de fiches. Pour obtenir une information, l'utilisateur n'a qu'à taper un message sur le clavier du terminal pour obtenir l'information désirée. Il y a un gain certain, surtout dans le domaine des frais de gestion.

* Réservation des places en avion.

Ce problème est bien connu. Il s'agit de connaître à tout instant le nombre de places restant libres dans un avion, compte tenu d'un certain nombre de facteurs, le poids des bagages par exemple. L'agence de réservation peut savoir à tout moment le nombre de places restant inoccupées pour tous les vols devant avoir lieu dans les trente jours à venir. On peut envoyer toute l'information voulue à l'aide d'un clavier, par exemple la réservation ou l'annulation d'une place. En général, en face de chaque place réservée est donnée l'indication du nom du passager et de son adresse. L'intérêt des consoles de visualisation est qu'elles sont beaucoup plus rapides que les consoles ordinaires et que l'on évite également l'utilisation d'une grande quantité de formulaires.

* Contrôle de livraisons.

On peut imaginer de pouvoir demander par exemple le chargement par camion, ainsi que la route que le camion doit suivre. Ceci permet de préparer le chargement optimal des camions, de pouvoir répondre à toute demande d'un client quant à la date de livraison d'un produit, et d'effectuer un contrôle constant sur la bonne utilisation des camions.

* Gestion hospitalière.

De nombreux problèmes demandant des réponses rapides se posent à tout instant dans un hopital. Il faut savoir le nombre de lits inoccupés, les entrées et les sorties de malades, les noms des malades présents et ce pour quoi ils sont soignés, les quantités de médicaments disponibles, les jours de service des personnes employées et bien d'autres renseignements encore. On voit donc qu'il s'agit de résoudre rapidement des problèmes complexes. L'emploi des consoles de visualisation permet d'éliminer l'usage intensif de téléphones et de registres d'un emploi fastidieux.

* Aide aux médecins.

Ici encore, il s'agit de pouvoir consulter un fichier concernant les clients d'un ou plusieurs médecins pour savoir rapidement si la personne qui se présente a déjà été soignée, et si oui pourquoi. On peut aussi imaginer un système d'aide du diagnostic par consultation de fichiers qui renferment les symptômes observés à propos de chaque maladie rencontrée. Ces dossiers sont vite très volumineux, et dans l'état actuel des choses il n'est possible d'imaginer ces systèmes que sur ordinateurs. Les terminaux alphanumériques permettent d'assurer le temps de réponse très rapide nécessaire au médecin qui ne peut demander à son patient de bien vouloir attendre une journée pour qu'il puisse savoir si son cas s'est déjà présenté.

* Enseignement programmé.

Le fait de pouvoir poser des questions et obtenir des réponses dans les deux sens fait des terminaux alphanumériques des instruments privilégiés pour l'enseignement programmé. On peut imaginer par exemple une question avec un certain nombre de réponses parmi lesquelles il faut choisir, ou encore la fabrication d'un texte analysé par l'ordinateur et en retour des commentaires sur la validité de ce texte. Nous voyons trois avantages par rapport à une console ordinaire : une plus grande rapidité lors de l'émission de messages (longs énoncés de problèmes), une plus grande propreté (il se pose un gros problème avec les consoles, celui de l'évacuation de tous les papiers par les étudiants!), et surtout un travail en silence. Il est tout à fait impossible de s'entendre dans une salle où travaillent vingt consoles. Il y a en revanche un grave défaut : le professeur n'a aucune trace du travail de l'étudiant tout au long de la séance. Il semble cependant que ce soit la voie future en particulier pour l'enseignement de la programmation sur une grande échelle.

* Edition et correction de programmes.

Une application très intéressante peut être faite quant à l'obtention

de programmes corrects. On peut imaginer de pouvoir corriger certaines parties de programmes apparaissant sur l'écran, de demander l'analyse syntaxique de ce programme, l'exécution en pas à pas avec apparition sur l'écran d'un certain nombre de renseignements relatifs au déroulement du programme. Ceci permet de mettre au point rapidement de gros programmes, par exemple des systèmes superviseurs dont la mise au point est toujours délicate et longue. Une sous-application de ce système permet de faire de la mise en page et de la correction de textes, afin d'obtenir des textes présentés d'une façon parfaite.

* Etude dynamique de programmes.

Il est intéressant de savoir à tout moment quels sont les sous-programmes d'un système de programmation qui sont le plus utilisés afin de voir si une amélioration ne peut être apportée à ceux qui sont les plus utiles ou encore si certains sous-programmes ne peuvent pas être entièrement supprimés pour gagner de la place.

On voit que la variété des applications est très grande, et nous n'avons donné là qu'un aperçu des possibilités des terminaux alphanumériques. Les terminaux graphiques offrent un éventail de possibilités tout aussi vaste, sinon plus.

2) Utilisation des terminaux graphiques.

Le champ des applications est différent de celui des terminaux alphanumériques puisque l'on peut afficher autre chose que des caractères. Il faut noter que l'on peut utiliser un terminal graphique comme un terminal alphanumérique, mais c'est là un gaspillage certain. En fait, un terminal graphique sera utilisé surtout pour dessiner des courbes.

* Dessin industriel.

Une des principales applications est sans conteste le dessin industriel.

Le gros problème dans ce cas est qu'il s'agit d'un procédé long et couteux en matériel et en hommes. D'autre part, le dessin d'une pièce est rarement définitif, et les modifications apportées à une pièce doivent être portées sur le dessin, d'où une nouvelle perte de temps. Les consoles de visualisation ont permis de réduire les délais dûs en grande partie au dessin des plans. En effet, grâce à un système de dessin approprié, on peut non seulement obtenir les mêmes vues et coupes que présente un dessinateur industriel, mais on peut encore demander de changer l'angle de vision, modifier certains détails, en ajouter ou en supprimer. Ces travaux qui demandent de longs mois par les méthodes classiques peuvent s'effectuer quasi-instantanément grâce aux consoles de visualisation. Le problème de la maintenace des plans est alors facilement résolu. Les plans peuvent être conservés sur microfilms , ce qui permet de gagner beaucoup de place.

Un tel programme de dessin industriel a en général les possibilités suivantes : affichage de points, de vecteurs, d'arcs de cercle, d'arcs d'ellipses, et d'arcs de paraboles. On peut encore faire du dessin à main levée et disposer d'un programme de lissage de courbes. Il faut d'autre part avoir la possibilité de manipuler ces images. Ces manipulations peuvent être des déplacements, des rotations, des symétries, des changements d'échelle ou des vues d'objets en perspective ou en coupe. Il semble indispensable de travailler dans un espace à trois dimensions. On voit donc que les spécifications sont nombreuses, mais la difficulté de conception et de mise en oeuvre de tels systèmes est largement compensée par le gain de temps obtenu.

* Contrôle des machines - outils.

Il s'agit là en fait d'un prolongement de l'application précédente. En effet, lorsqu'un plan a été réalisé, il comporte (ou peut comporter) toutes les indications nécessaires à la réalisation d'une pièce. L'idée est alors de coupler le système de dessin avec un générateur du langage APT par exemple qui transformera la représentation du système de dessin en une suite de codes destinés à guider des machines-outils en vue de réaliser la pièce représentée. Il y a encore un gain très important, puisque les charges de la machine peuvent être calculées sans intermédiaire.

* Dessin d'architecture.

Cette application permet de gagner beaucoup de temps en confiant au calculateur le soin de dessiner les plans en tenant compte des contraintes particulières. On pourra faire par exemple des essais de rupture en charge d'une poutre en modifiant les paramètres à l'aide de la console. On peut obtenir ainsi rapidement les meilleurs profils ou les meilleures répartitions d'efforts. Le gros problème est ici de pouvoir appliquer un certain nombre de contraintes à une pièce donnée.

* Schémas électroniques.

Il ne s'agit pas seulement de faire de beaux schémas, mais aussi de pouvoir trouver la meilleure implantation des éléments sur une plaquette, ou encore de pouvoir calculer les caractéristiques d'un circuit que l'on aura tout d'abord dessiné sur l'écran. En faisant varier les paramètres du circuit, on pourra obtenir rapidement les valeurs critiques ou les temps de réponse recherchés. On peut ainsi étudier très rapidement les effets de variation des paramètres, sans avoir à construire chaque fois les circuits correspondants.

* Lissage de courbes.

A la suite d'expériences, on peut très bien imaginer avoir obtenu comme résultat une série de points. On désire savoir la forme de la courbe reliant ces points, en évitant éventuellement certains points qui manifestement ne sont pas sur celle-ci. Ceci peut être obtenu rapidement grâce à un système d'analyse des données, qui utilise un terminal graphique grâce auquel l'utilisateur pourra indiquer quels sont les points à ignorer, ou encore en ajouter d'autres.

* Dessins animés.

En laissant au calculateur le soin de modifier lui-même les images sur l'écran par calcul les différentes positions des éléments de cette image,

on obtient très rapidement des films animés. L'intérêt de cette application réside dans le fait que l'on peut obtenir des dessins de heute précision qui permettent d'étudier par exemple le mouvement de satellites autour de la terre.

* Dépouillement de clichés.

Il s'agit surtout du dépouillement des clichés d'une chambre à bulles. En utilisant le crayon optique, on peut obtenir une représentation exacte du cliché en mémoire (digitalisation). On peut alors laisser le calculateur faire les analyses nécessaires, ce qui permet de gagner le temps de transformation du cliché en renseignements acceptables par l'ordinateur (cartes ou bandes perforées).

De la même façon que dans le cas des terminaux alphanumériques, la conclusion à tirer est que les limites des terminaux graphiques ne sont pas encore connues. Les services qu'ils peuvent rendre sont immenses. Le tout est de trouver les points vraiment intéressants... et de réaliser les systèmes correspondants. Actuellement, seuls sont réalisés des systèmes très spécialisés. Il semble que ce soit une perte de temps et d'argent à long terme, et qu'il faille s'orienter vers la conception de systèmes généraux.

C O N C L U S I O N

Il est indéniable que les consoles de visualisation présentent un immense intérêt. Leur souplesse d'emploi en fait des instruments adaptés à la résolution des problèmes les plus divers. Le fait de pouvoir les employer en mode conversationnel permet de penser que leur emploi va se généraliser rapidement. Les difficultés d'utilisation seront peu à peu surmontées au fur et à mesure que se développeront les bibliothèques de programmes de dessin et les études théoriques sur la structure des images. Les recherches actuelles sont orientées dans plusieurs voies. La première, et non des moindres, est une recherche technologique. Il s'agit de trouver comment réaliser des matériels de plus en plus performants, mais surtout d'un prix plus accessible. Il semble que tant que ce point ne sera pas résolu de façon satisfaisante, la progression du nombre des consoles de visualisation sera très lente. Le deuxième point est celui de la recherche théorique. Il s'agit de trouver la philosophie profonde des structures d'images, non seulement au sens où nous l'avons entendu jusqu'à présent, mais aussi dans le sens de la reconnaissance d'une image à partir de ses composants. C'est là un grand problème sur lequel de nombreuses personnes se sont penchées et travaillent encore. Le jour où le problème de la reconnaissance des formes sera résolu, le champ d'application des consoles de visualisation n'aura alors pratiquement plus de limites.

Les autres directions de recherche concernent plus particulièrement un aspect pratique : mettre à la disposition de l'utilisateur un moyen d'écrire des programmes de dessin comme il écrit un programme de gestion ou de calcul scientifique. Deux voies sont actuellement explorées : soit l'inclusion d'un langage graphique dans un langage de haut niveau (LISP, FORTRAN, PLV), soit définition d'un nouveau langage permettant d'écrire entièrement toutes les phases d'un programme de dessin, y compris le dialogue entre l'homme et la machine. Il permettrait alors l'écriture de programmes aussi complexes que possibles et surtout aussi variés que différents les uns des autres, ce que l'on ne peut

faire actuellement. En effet, tous les systèmes de dessin existants sont conçus en fonction d'une application précise, et ne permettent pas d'obtenir satisfaction en dehors de cette application. Le langage dont nous parlons aurait une "vocation universelle".

Il reste donc encore beaucoup de travail à faire. Nous espérons simplement que le présent ouvrage suscitera un nombre de vocations suffisant pour permettre de réaliser des projets de plus en plus ambitieux, qui permettront d'utiliser au mieux les terminaux graphiques.

B I B L I O G R A P H I E

*****

Le lecteur interessé trouvera un grand nombre de renseignements sur tout ce qui touche aux consoles de visualisation dans

THE COMPUTER DISPLAY REVIEW Adams Associates Incorporated

Cette publication comporte un grand nombre de rubriques et elle est mise à jour périodiquement. Nous en avons fait un grand usage. En particulier, la rédaction de certains passages de cet ouvrage en est fortement inspirée. Il s'agit essentiellement :

au CHAPITRE I, de la partie A, GENERALITES SUR LES CONSOLES DE VISUALISATION.

au CHAPITRE II, partie A, du paragraphe 5, Utilisation du crayon optique.

au CHAPITRE IV, de la partie A, GENERALITES SUR LES STRUCTURES D'IMAGES.

au CHAPITRE V, de la partie B, EXEMPLES D'UTILISATION DES CONSOLES DE VISUALISATION.

Les schémas I, II, III, IV, et V ont également été tirés de cette publication.

Le lecteur pourra encore consulter avec profit :

1) OUVRAGES GENERAUX.

1-1 A.G. ARKADEV et E.M. BRAXEMAN, Teaching Computers to Recognize Patterns, Academic Press, 1967

1-2 E.A. FEIGENBAUM and J.FELDMAN, Computers and Thought, Mc Graw Hill, 1963

1-3 HARRY H.POOLE, Fundamentals of Display Systems, Mac Millan and Company, London, 1966

1-4 R.A. SIDERS and OTHERS, Computer Graphics : A Revolution in Design, American Management Association, New-York, 1966

1-5 F.D. SKINNER, Computer Graphics : Where are we ?, Datamation, Mai 1966

1-6 I.E. SUTHERLAND, Computer Graphics ; Ten Unsolved Problems, Datamation, Mai 1966

2) LANGAGES GRAPHIQUES.

2-1 E.CLEEMANN, Langage de programmation graphique pour le terminal IBM 2250; Thèse de Doctorat de Troisième Cycle, Grenoble, Octobre 1968

2-2 E.CLEEMANN, O.LECARME et M.LUCAS, Langages de Programmation graphique, article à paraître dans la Revue Française de l'Informatique.

2-3 IBM SYSTEM/360 OPERATING SYSTEM : Graphic Programming Services for Fortran IV, Form C27-6932.

2-4 H.E.KULSRUD, A general Purpose Graphic Language, Communications of the ACM, Avril 1968

2-5 O.LECARME et M.LUCAS, LAGROL : Un langage pour l'utilisation d'un terminal graphique, note technique de l'I.M.A.G., janvier 1968

2-6 W.R.SUTHERLAND, The CORAL Language and Data Structure, M.I.T., Lincoln Lab.

3) SYSTEMES DE DESSIN GRAPHIQUE.

3-1 T.E. JOHNSON, Sketchpad III : A computer Program For Drawing in Three Dimensions. AFIPS Conf.Proc.(1963, Spring Joint Computer Conference).

3-2 O.LECARME, Etude et réalisation d'un système général conversationnel de programmation graphique, Thèse de Doctorat d'Etat es-Sciences Appliquées, 1970.

3-3 O.LECARME et E.CLEEMANN, Système d'utilisation rationnelle d'un terminal graphique évolué, Séminaire de programmation de l'I.M.A.G., Juin 1968

3-4 M.LUCAS, Expériences sur un terminal graphique élémentaire, Séminaire de programmation de l'IM.A.G., Janvier 1968.

3-5 WILLIAM M.NEWMAN, An Experimental Program For Architectural Design, Imperial College of Science and Technology, Novembre 1966.

3-6 L.G. ROBERTS, Machine Perception of Three Dimensional Solids, Ph. D. Thesis, M.I.T., Electrical Engineering Dept Cambridge, Mass, Février 1963.

3-7 R. STOTZ, Man-Machine Console Facilities for Computer-Aided Design, AFIPS Conf. Proc. (1963, Spring Joint Computer Conference)

3-8 I.E.SUTHERLAND, SKETCHPAD; A man-Machine Graphical Communication System, AFIPS Conf. Proc. (1963, Spring Joint Computer Conference)

3-9 R.A.WEISS, Be Vision-A Package of IBM Fortran Programs to Draw Orthographic Views of Combinations of Plane and Quadric Surfaces, Journal ACM, Avril 1966.

4) OUVRAGES DIVERS

4-1 D.CLAUZEL, Caractéristiques et langage de commande de l'assembleur PDP-8 sur IBM 7044, note technique I.M.A.G.

4-2 The DIGITAL Small Computer Handbook, DEC, Maynard, Mass, 1966-67

4-3 IBM SYSTEM/360 Component Description : IBM 2250 Display Unit Model 1, Form A27-2701

4-4 IBM SYSTEM/360 OPERATING SYSTEM : Graphic Programming Services for IBM 2250 Display Unit, Form C27-6909

4-5 Mac CARTHY, LISP 1.5, M.I.T., 1962

4-6 NEWELL, IPL Programmer's reference manual, Rand Corporation, 1960

4-7 L.G. ROBERTS, Homogeneous Matrix Représentation and Manipulations of N Dimensional Constructs, M.I.T. Lincoln Lab.

4-8 WEIZENBAUM, Symetric List Processor, Communications of the A.C.M. Septembre 1963.

```
/       DEFINITION DE TOUS LES SYMBOLES UTILISES PAR LAGROL EN PLUS
/       DU LANGAGE D'ASSEMBLAGE DU PDP-8.

/       INSTRUCTIONS
GPOIN =0000             POINT
GCAR  =1000             CARACTERE
GVEC  =2000             VECTEUR
GDEP  =3000             DEPLACEMENT
/ GJMP=JMP              RUPTURE DE SEQUENCE INCONDITIONNELLE
/ GJMS=JMS              APPEL DE SOUS-PROGRAMME
GARR  =6000             SORTIE DU PROGRAMME INTERPRETE
GMODIF=6001             MODIFICATION DU SYSTEME DE REFERENCE
GAPPEL=6002             APPEL DE FONCTION EN LANGAGE MACHINE
GCERCL=7000             CERCLE
GNOP  =JMP I GCO        NON OPERATION

/       OPTIONS
GETEIN=0010             ETEINDRE
GALUM =0000             ALLUMER
GCLEN =0060             CRAYON OPTIQUE EN FONCTION
GCLOR =0040             CRAYON OPTIQUE HORS FONCTION
GECH1 =0000             ECHELLES
GECH2 =0100
GECH3 =0200
GECH4 =0300
GECH5 =0400
GECH6 =0500
GECH7 =0600
GECH8 =0700
/ CHIFFRE DE 0 A 7      INTENSITE

/       CODES DE DEFINITION
GA      =0000           POINT ABSOLU
GAF   =4000             POINT ABSOLU,FIN DE CHAINAGE
GR    =2000             POINT RELATIF
GRF   =6000             POINT RELATIF,FIN DE CHAINAGE
G0    =0000             0 DEPLACEMENT
G0F   =4000             0 DEPLACEMENT,FIN DE CHAINAGE
G1    =1000             1 DEPLACEMENT
G1F   =5000             1 DEPLACEMENT,FIN DE CHAINAGE
G2    =2000             2 DEPLACEMENTS
G2F   =6000             2 DEPLACEMENTS,FIN DE CHAINAGE
G3    =3000             3 DEPLACEMENTS
G3F   =7000             3 DEPLACEMENTS,FIN DE CHAINAGE

/       CODES DE DONNEES
/ ADRESSES SYMBOLIQUES OU ABSOLUES
GE1   ==0000            DIRECTION DE DEPLACEMENT
GE2   =0000             EST
GE3   =0000
GNE1  =0100
GNE2  =0010             NORD-EST
GNE3  =0001
```

```
/       EXEMPLE DE PROGRAMME EN MODE IOF

        PAGE 1
DEBUT   HLT CLA
        TAD (-62
        DCA AVARE       INITIALISATION DU COMPTEUR DE TOURS
        TAD (IMAG1
        LAGROL          DESSIN DE LA PREMIERE IMAGE
        ISZ AVARE       A LA FIN DU DESSIN DE LA PREMIERE IMAGE, ON
        JMP .-3         REVIENT DERRIERE L'APPEL DE L'INTERPRETE
        TAD (-62        ON A DESSINE N FOIS LA PREMIERE IMAGE
        DCA AVARE
        TAD (IMAG2      ON RECOMMENCE POUR LA DEUXIEME
        LAGROL          DESSIN DE LA DEUXIEME IMAGE
        ISZ AVARE       PEUT ON RETOURNER DESSINER LA PREMIERE IMAGE
        JMP .-3         NON, DESSINER ENCORE LA DEUXIEME
        JMP DEBUT+1     OUI, ALLONS-Y

ONDES   0               SOUS-PROGRAMME DE DESSIN D'UNE ONDULATION
        GPOIN,4         ON DESSINE PLUSIEURS POINTS EN CHAINE
        GA,600
        GR,740          ADRESSE RELATIVE EN Y SEULEMENT
        GA,612
        GR,750
        GA,624
        GR,755
        GA,636
        GR,750
        GA,650
        GR,740
        GA,662
        GR,730
        GA,674
        GR,723
        GA,706
        GR,730
        GA,720
        GR,740
        GA,732
        GR,750
        GA,744
        GR,755
        GA,756
        GR,750
        GA,770
        GRF,740
        JMP I ONDES     RETOUR AU PROGRAMME DE DESSIN
AVARE   0
```

```
        PAGE
IMAG1   GPOIN,GETEIN    DESSIN DE L'ENVELOPPE A L'ENVERS
        GA,404
        GAF,404
        GVEC,GECH3,5    INSTRUCTION DE TRACE DE VECTEURS
        GA,404
        GA,1140
        GA,1414
        GA,1140         ON DESSINE L(ENVELOPPE D'UN SEUL
        GA,1010         TRAIT GRACE AU CHAINAGE.
        GA,1416
        GA,404
        GA,1140
        GA,1414
        GA,404
        GA,404
        GA,404
        GA,1414
        GA,1140
        GA,1414
        GAF,404         FIN DU CHAINAGE
        GARR            SORTIE DE L'INTERPRETE

IMAG2   GPOIN,GETEIN    DESSIN DE LA DEUXIEME IMAGE
        GA,404
        GAF,1140
        GVEC,GECH5,5    DESSIN DU TOUR DE L'ENVELOPPE
        GA,404
        GA,404
        GA,1414
        GA,404
        GA,1414
        GA,1140
        GA,404
        GA,1140
        GA,1010
        GA,1416
        GA,1414
        GAF,1140
        GPOIN,GETEIN    DESSIN DU TIMBRE
        GA,1370
        GAF,1120
        GVEC,GECH3,4
        GA,1370
        GA,750
        GA,1240
        GA,750
        GA,1240
        GA,1120
        GA,1370
        GAF,1120
        GPOIN,GETEIN    DESSIN DU CACHET
        GA,1200         A L'AIDE DES DEPLACEMENTS
        GAF,1306
```

```
GDEP,GECH8,4
G3,GE1,GE2,GE3
G3,GSE1,GSE2,3SE3
G3,GS1,GS2,GS3
G3,GSO1,GSO2,GSO3
G3,GO1,GO2,GO3
G3,GNO1,GNO2,GNO3
G3,GN1,GN2,GN3
G3F,GNE1,GNE2,GNE3
GMODIF          DESSIN DES ONDULATIONS
GA,0
GA,10
JMS ONDES       ON UTILISE LE SOUS-PROGRAMME ONDES EN
GMODIF          MODIFIANT L'EMPLACEMENT DE CHAQUE
GA,0            ONDULATION GRACE A GMODIF
GA,20
JMS ONDES
GMODIF
GA,0
GA,30
JMS ONDES
GPOIN,GETEIN    DESSIN DE L'ADRESSE
GA,600
GAF,600
GCAR,GECH2,5
TEXT 14,LUCAS MICHEL
GPOIN,GETEIN
GA,600
GAF,550
GCAR,GECH2,5
TEXT 10,      IMAG
GARR            SORTIE DE L'INTERPRETE
END DEBUT
```

```
/       EXEMPLE DE PROGRAMME EN MODE ION

        PAGE 1.
DEBUT   HLT CLA
DEPART  TAD (ARRET          ON IGNORE LES INTERRUPTIONS VENANT
        DCA GSPCLA          DU CLAVIER DE LA CONSOLE
        TAD (TEXTEL         TRAITEMENT DE L'IMPRIMANTE
        DCA GSPTEL
        TAD (CRAYON         TRAITEMENT DU CRAYON OPTIQUE
        DCA GSPCRA
        ION                 ON PERMET LES INTERRUPTIONS
        TAD (MENU           ON VA AFFICHER LE MENU
        LAGROL              APPEL DU PROGRAMME INTERPRETE

CRAYON  0
        CIA                 L'ACCUMULATEUR CONTIENT GCO
        TAD (OUESCE-3       SUIVANT LA LIGNE DU MENU QUI A ETE MONTREE
        SPA
        JMP AUTO            ON DESSINE LA VOITURE
        TAD (-3
        SPA CLA
        JMP TEXTE           OU ON ECRIT UN TEXTE
VECTEU  JMS GCROIX          OU ON FAIT APPARAITRE LA CROIX
        TAD LATI            PREMIER POINT,COORDONNEES DANS LATI ET LONGI
        DCA X1
        TAD LONGI
        TAD (4000
        DCA X1+1
        JMS GCROIX          AFFICHAGE DE LA CROIX
        TAD LATI            DEUXIEME POINT DU VECTEUR
        DCA X1+3
        TAD LONGI
        TAD (4000
        DCA X1+4
        DCF
        ION
        TAD (X1-1
        LAGROL              DESSIN DU VECTEUR

SPECLA  0                   SEQUENCE PERMETTONT DE RECONNAITRE LE
        TAD (-301           CARATERE FRAPPE ET DE DEDUIRE LE TRAITEMENT
        SNA
        JMP AVANT           C'EST UN A
        TAD (-21
        SZA CLA
        JMP I SPECLA        NI UN A NI UN R, ON IGNORE
        TAD (-1             C'ETAIT UN R
        TAD AV
        DCA AV
        JMP I SPECLA
AVANT   ISZ AV              ON AUGMENTE DE 1
        CLA
        JMP I SPECLA
```

```
AUTO    CLA             DESSIN DE L'AUTO
        TAD (ARRET
        DCA GSPCRA
        DCA AV
        DCA GRELX
        TAD (SPECLA
        DCA GSPCLA
        DCF
        ION
        TAD (BAUTO
CAUTO   LAGROL
        TAD AV
        TAD GRELX
        DCA GRELX       NOUVELLE VALEUR DE GRELX.
        TAD GRELX       CETTE VALEUR FAIT AVANCER OU RECULER L'AUTO.
        SPA
        JMP TROGAU      SORTIE DES LIMITES DE L'ECRAN
        TAD (-1700
        SPA CLA
        JMP CAUTO-1
        JMP AUTO+4
TROGAU  CLA             RECTIFICATION
        TAD (1700
        DCA GRELX
        JMP CAUTO-1
AV      0


TEXTE   TAD (TEXCLA     ECRITURE DU COMMENTAIRE
        DCA GSPCLA      'JE FAIS DEUX CHOSES A LA FOIS'
        TAD (TEXTEL
        DCA GSPTEL
        TAD (ARRET
        DCA GSPCRA
        DCF
        ION
        TAD (BTEXTE
        LAGROL

TEXCLA  0
        NOP             CETTE SEQUENCE PERMET D'ECRIRE UN
        TLS             CARACTERE SUR LA CONSOLE
        CLA
        TAD (JMP TEXCLA+1
        DCA TEXCLA+1
        JMP I TEXCLA

TEXTEL  0
        TAD (NOP
        DCA TEXCLA+1
        TCF
        JMP I TEXTEL
```

```
        PAGE
        GPOIN,GETEIN     DESSIN DU VECTEUR
X1      GA,0
        GAF,0
        GVEC,GECH3,4
        GA,0
        GAF,0
        GPOIN,GETEIN
        GA,0
        GAF,0
        GCAR,3
        TEXT 30,TAPEZ AU TELETYPE POUR ARRETER
        JMP X1-1
BAUTO   GPOIN,GETEIN     DESSIN DE L'AUTO
        GR,0             RELATIF
        GAF,500
        GVEC,GECH3,5
        GR,10
        GA,523
        GR,13
        GA,526
        GR,40
        GA,526
        GR,43
        GA,523
        GR,45
        GA,513
        GR,60
        GA,513
        GR,63
        GA,510
        GR,65
        GA,500
        GR,0
        GAF,500
        GPOIN,GETEIN
        GR,16
        GAF,500
        JMS ROUE
        GPOIN,GETEIN
        GR,63
        GAF,500
        JMS ROUE
        GPOIN,GETEIN
        GA,0
        GAF,30
        GCAR,3,*CLEN
        TEXT 33,A=ACCELERER R=RALENTIR CRAYON=FIN
        GARR

ARRET   0
        CLA              CETTE SEQUENCE REDONNE LE CONTROLE AU
        JMP DEBUT+1      DEBUT DU PROGRAMME APRES UNE INTERRUPTION
```

```
MENU    GPOIN,GETEIN     AFFICHAGE DU MENU
        GA,0
        GAF,1000
        GCAR,GCLEN,GECH3,5
        TEXT 19,VECTEUR TEXTE AUTO
OUESCE  JMP MENU

ROUE    0                SOUS PROGRAMME DE DESSIN D'UNE ROUE
        GDEP,5
        G3,GS1,GS2,GSO3
        G3,GSO1,GO2,GO3
        G3,GNO1,GNO2,GN3
        G1F,GN1
        JMP I ROUE

TEXTE   GPOIN,GETEIN
        GA,50
        GAF,500
        GCAR,GECH8,6
        TEXT 30,JE FAIS DEUX CHOSES A LA FOIS.
        JMP BTEXTE

        END DEBUT
```

NOUS NOUS PROPOSONS SEULEMENT DE MONTRER QUELQUES POSSIBILITES DU SYSTEME 'GENIAL', AFIN DE DONNER AU LECTEUR UNE BONNE IDEE DE CE QUE PEUT ETRE UN DIALOGUE ENTRE L'HOMME ET LA MACHINE.

```
INISYS
N OU S?
N
APPELEZ LE GENERATEUR
GENIAL

          ** NOUS COMMENCONS UN NOUVEAU TRAVAIL **
DEBUT

GENERER
          CHAINE     E3,L4,0,0,PETITE DEMONSTRATION @
CONSERVER FIGURE 0001
          ** EXEMPLE DE FIGURE SIMPLE **

GENERER
          VECTEURS   10,#,E3,L5,404,404,404,1140,1414,1140,1010,
                     1416,404,1140,1414,404,404,404,1414,1140,
                     1414,404;
CONSERVER FIGURE 0002
** CETTE FIGURE REPRESENTE L'ENVELOPPE DECRITE DANS LE PROGRAMME
          DONNE EN ANNEXE B **

*ADRESSE  0060
** CE RENSEIGNEMENT PERMET DE CONNAITRE LA PLACE RESTANT DANS LA
          PAGE COURANTE **

GENERER
          VECTEUR    #,E3,L5,1000,0,1000,200;
          TRIANGLE   #,E3,L5,1000,200,400,0,1600,0;
          CHAINE     E2,L4,1010,100,HAUTEUR @
          CHAINE     E2,L4,1000,204,A @
          CHAINE     E2,L4,360,0,B @
          CHAINE     E2,L4,1610,0,C @
CONSERVER FIGURE 0003
** EXEMPLE DE FIGURE COMPOSEE: IL S'AGIT D'UN TRIANGLE ABC ET DE
          LA HAUTEUR PARTANT DU SOMMET A **

*ADRESSE  0146

GA??
          ** ON A FAIT UNE ERREUR EN FRAPPANT LA COMMANDE **
```

```
AFFICHER  3,200,1000;
AFFICHER  3,1000,400;
AFFICHER  1,0,0;
          ** EXEMPLE D'IMAGE A L'AIDE DES FIGURES **

DETRUIRE
          ** L'IMAGE QUI VIENT D'ETRE CONSTRUITE EST EFFACEE **

GENERER
          CERCLE    #,E1,L5,1000,1000,400,+777,-1200;
          CHAINE    E2,L3,200,300,ARC DE CERCLE @
PAGE DEPASSEE 0162
          ** LE PROGRAMMEUR DECIDE DE NE PAS GARDER LA FIGURE **
DETRUIRE

*ADRESSE  0146
GENERER
          POINTS    30,9??
          ** LE PROGRAMMEUR AURAIT DU INDIQUER LE MOYEN D'ENTREE **
          POINTS    10,!,L3;
          ** LA CROIX APPARAIT SUR L'ECRAN ET EST UTILISEE POUR
          INDIQUER QUELS SONT LES POINTS A RETENIR **
CONSERVER FIGURE 0004
*ADRESSE  0154

*PAGE
          ** LE PROGRAMMEUR ESTIME QU'IL NE LUI RESTE PAS ASSEZ
          DE PLACE DANS LA PAGE POUR CONTINUER **

GENERER
          CHAINE    E4,L5,0,8? 9? 200,ESSAI NUMERO 1 @
** LES CHIFFRES 8 ET 9 SONT SUIVIS D'UN POINT D'INTERROGATION CAR
LE PROGRAMMEUR A DONNE DES CHIFFRES NON OCTAUX **

          CHAINE    E4,L5,0,1325??
** LE PROGRAMMEUR A ANNULE LA COMMANDE PAR LA TOUCHE 'RUB OUT' **

*DECIMAL
          CHAINE    E4,L5,0,800,ESSAI NUMERO 2 @
          ** LE CHIFFRE 8 EST ACCEPTE MAINTENANT **

*OCTAL
CONSERVER FIGURE 0005

AFFICHER  1,0,0;
AFFICHER  2,100,500;
AFFICHER  3,200,1560;
AFFICHER  4,400,1000;
AFFICHER  5,1300,420;
AFFICHER  6, FIGURE INCONNUE
```

```
DETRUIRE

AFFICHER   1,0,0;
AFFICHER   5,1000,200;
AFFICHER   5,1200,1200;
DEPLACER   #,5,+100,-200;
           ** TOUTES LES OCCURENCES DE LA FIGURE 5 SE DEPLACENT **
AFFICHER   3,1000,1500;

EFFACER    #,1;
           ** L'IMAGE NE CONTIENT PLUS LA FIGURE 1 **

EFFACER    #,3;
           ** LA TOTALITE DE LA FIGURE 3 DISPARAIT **

RALLUMER   3;

EFFACER    !, FIGURE 0003
** SEULE UNE PARTIE DE L'IMAGE DISPARAIT, CELLE CONSTITUEE PAR
L'OBJET DE LA FIGURE 3 QUI A ETE MONTRE AVEC LE CRAYON OPTIQUE.
LE RESTE DE LA FIGURE 3 APPARAIT TOUJOURS **

VECTEUR    INTERDIT
           ** CETTE COMMANDE NE PEUT ETRE UTILISEE DANS CE MODE **

GENERER
APPLIQUER  F0,+10,-20;
           VECTEURS  20,1,E4,L5;
           POINT     #,L3,234,420;
FIN D'APPLICATION
CONSERVER  FIGURE 0006
           ** LA FIGURE DESSINEE AVEC LE CRAYON OPTIQUE SERA ANIMEE

           D'UN MOUVEMENT DE TRANSLATION CONTINUE **

DETRUIRE
AFFICHER   6,1000,1000;
           ** LA FIGURE EST IMMOBILE POUR L'INSTANT **
ANIMER     FIGURE 0006
           ** LA FIGURE EST EN MOUVEMENT **
AFFICHER   4,1000,1000;
ANIMER     SANS ANIMATION
           ** LA FIGURE DESIGNEE PAR LE CRAYON OPTIQUE N'EST PAS ANIMEE *
ARRETER    FIGURE 0006
           ** LA FIGURE EST DE NOUVEAU IMMOBILE **

GENERER
           AJOUTER   5;
           ** RECOPIE DE L'IMAGE 5 **
AFFICHER     INTERDIT
           ** MODE DE TRAVAIL NON RESPECTE **
CONSERVER  FIGURE 0007
```

```
AFFICHER  1,200,200;
AFFICHER  4,1000,1000;
AFFICHER  4,300,300;
AFFICHER  5,200,1500;
AFFICHER  5,500,500;
AFFICHER  6,0,1230;
AFFICHER  7,1600,1600;

** LE PROGRAMMEUR DESIRE DEPLACER UNE FIGURE ET NE SAIT PLUS SON
          NUMERO **
NOMMER    FIGURE 0004

** IL DECIDE ALORS DE DEPLACER UNE SEULE OCCURENCE DE CETTE FIGURE
A UN ENDROIT QU'IL VA CHOISIR A L'AIDE DU CRAYON OPTIQUE **

DEPLACER  !, FIGURE 0004

          ** IL VEUT CONNAITRE LES COORDONNEES D'UN POINT DE L'ECRAN **

LOCALISER 0544 0356
AFFICHER  4,550,350;

** LE PROGRAMMEUR EST ALORS SATISFAIT ET DECIDE DE CONSERVER SON
TRAVAIL SUR BANDE **

PRESERVER
APPELEZ .UPDATE.
UPDATE

PROGRAM NAME    :IMAGE
SA (OCTAL)      :4
PAGE LOCATIONS :<0,2577>;



** LE PROGRAMMEUR DECIDE DE REUTILISER SON PROGRAMME **

INISYS
N OU S?
S
NOM DE VOTRE IMAGE
IMAGE

APPELEZ LE GENERATEUR
GENIAL

AFFICHER  4,1000,1000;
** LE TRAVAIL PEUT COMMENCER TOUT DE SUITE, SANS QUE LE PROGRAMMEUR
FRAPPE LA COMMANDE DEBUT **
```

```
GN1    =0200
GN2    =0020                NORD
GN3    =0002
GNO1   =0300
GNO2   =0030                NORD-OUEST
GNO3   =0003
GO1    =0400
GO2    =0040                OUEST
GO3    =0004
GSO1   =0500
GSO2   =0050                SUD-OUEST
GSO3   =0005
GS1    =0600
GS2    =0060                SUD
GS3    =0006
GSE1   =0700
GSE2   F0070                SUD-EST
GSE3   =0007

/REGISTRES GENERAUX
LAGROL=JMS I 20
GSPCRA=2
GSPCLA=3
GSPTEL=4
GABS   =5
GORD   =6
GECH   =7
GCO    =21
GRT    =22
GRELX  =23
GRELY  =24
```

VU

Grenoble, le

Le Président de la Thèse

VU

Grenoble, le

Le Doyen de la Faculté des Sciences

Vu, et permis d'imprimer,

Le Recteur de l'Académie de GRENOBLE