Un macro-langage pour la programmation des terminaux graphiques

Edouard Cleemann

N° D'Ordre

T H E S E

présentée à

LA FACULTE DES SCIENCES DE L'UNIVERSITE DE GRENOBLE

pour obtenir

LE TITRE DE DOCTEUR DE TROISIEME CYCLE

"MATHEMATIQUES APPLIQUEES"

par

M. CLEEMANN Edouard

UN MACRO-LANGAGE POUR LA PROGRAMMATION DES TERMINAUX GRAPHIQUES

Thèse soutenue le 3 MARS 1969, devant la Commission d'Examen

MM. KUNTZMANN Président

GASTINEL, BOLLIET } Examinateur

N° D'Ordre

T H E S E

présentée à

LA FACULTE DES SCIENCES DE L'UNIVERSITE DE GRENOBLE

pour obtenir

LE TITRE DE DOCTEUR DE TROISIEME CYCLE

"MATHEMATIQUES APPLIQUEES"

par

M. CLEEMANN Edouard

UN MACRO-LANGAGE POUR LA PROGRAMMATION DES TERMINAUX GRAPHIQUES

Thèse soutenue le 3 MARS 1969, devant la Commission d'Examen

MM. KUNTZMANN Président

GASTINEL, BOLLIET } Examinateur

FACULTE DES SCIENCES — UNIVERSITE DE GRENOBLE

L I S T E des P R O F E S S E U R S

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
	AYANT Yves	Physique Approfondie
	GALLISSOT	Mathématiques
Mlle	LUTZ Elisabeth	Mathématiques
	LAMBERT 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
	DESSEAUX 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
	DREYFUS B.	Thermodynamique

MM.	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
	DUCROS P.	Chaire de Cristallographie Physique
	DODU Jacques	Chaire de Mécanique Appliquée I.U.T.
	ANGLES D'AURIAC P.	Mécanique des Fluides
	LACAZE A.	Thermodynamique

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.R.	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 Génie 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

MAITRES DE CONFERENCES (suite)

Mme	BONNIER M.J.	Chimie
MM.	KAHANE A.	Physique Générale
	DOLIQUE J.M.	Electronique
	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 T.M.P.
	LAURENT P.J.	Mathématiques Appliquées T.M.P.
	CAUBET J.P.	Mathématiques Pures
	PAYAN J.J.	Mathématiques
Mme	BERTRANDIAS F.	Mathématiques Pures M.P.C.
	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.P.	Construction Mécanique I.U.T.
	CONTE René	Thermodynamique
	JOLY Jean René	Mathématiques Pures
Mme	PIERY Yvette	Biologie Animale
	BENZAKEN Claude	Mathématiques Appliquées

MAITRES DE CONFERENCES ASSOCIES

MM.	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 a montré à mon égard son intérêt pour les problèmes de langages graphiques.

Monsieur Louis BOLLIET, Maître de Conférences à l'I.U.T. d'Informatique, à qui je dois d'avoir orienté mes études puis ma recherche dans le domaine de la Programmation.

Monsieur Olivier LECARME, Maître-Assistant à l'Institut de Programmation, qui m'a éclairé dans mes travaux en m'aidant de son expérience et de ses conseils.

Tous les membres du Laboratoire, en particulier :
Messieurs, LUCAS, BELLISSANT, CACHAT, SUTY, ASSABGUI, avec qui j'ai appris à me servir du nouvel ordinateur 360.

Le personnel des services de dactylographie et de reproduction qui a contribué, par ses soins diligents, à la réalisation matérielle de cet ouvrage.

TABLE DES MATIERES

-:-:-:-:-

CHAPITRE V :

CHAPITRE VI :

INTRODUCTION

Si on examine l'évolution des ordinateurs sous l'angle de la programmation et de la communication entre le programmeur et la machine on distingue trois phases :

- au début, quand les calculateurs étaient encore peu puissants (650 IBM, GAMMA-ET) et les langages évolués presque inexistants le programmeur (qui faisait aussi office d'opérateur) assistait à l'exécution de son programme et pouvait intervenir dans le déroulement de celui-ci en arrêtant l'ordinateur et en faisant des modifications au programme ou aux données ; cela requerrait évidemment de la part du programmeur une connaissance parfaite de la technologie de la machine et des dispositifs d'entrée/sortie.

- Quelques années plus tard, les ordinateurs devenant très puissants et les opérations d'entrée et sortie très complexes du fait des techniques de simultanéité et des problèmes de gestion des interruptions, le programmeur est exclu de la salle des machines et doit faire confiance à un système de contrôle moniteur pour tout ce qui concerne les communications avec les multiples organes périphériques ; la correction des erreurs et la mise au point de son programme se font par "passages" successifs, après l'examen laborieux des analyses de la mémoire et des résultats partiels.

- La mise en oeuvre des systèmes en temps partagé et l'installation de machines à écrire connectées aux ordinateurs concourrent à instituer une troisième manière de se servir des ordinateurs (Réf. BOL-1). Dans ce nouveau mode, qui en est à son début, on redonne au programmeur la possibilité de dialoguer avec l'ordinateur. Il faut cependant bien voir la spécificité de cette fonction de dialogue dans l'utilisation d'une telle machine à écrire ; en raison de la lenteur de cet organe (10 à 20 caractères par

seconde en émission), ce n'est qu'exceptionnellement qu'on l'utilisera pour introduire un programme ou un fichier dans la mémoire centrale de l'ordinateur ; en revanche, au moyen d'un langage de commande approprié, on pourra, après s'être présenté au système, chercher un programme dans une bibliothèque et en lancer l'exécution, recueillir des résultats, faire des mises à jour de fichiers, des corrections de programmes, etc...

Par ailleurs depuis quelques années la gamme des unités périphériques de calculateurs s'est enrichie de ce que l'on appelle un terminal graphique. Constitué initialement d'un tube cathodique connecté par un convertisseur digital-analogique à un ordinateur qui lui envoie des ordres pour affichier une succession de points sur l'écran, il s'est adjoint des dispositifs qui en font un organe d'entrée/sortie riche de possibilités de communication et de dessin : pour les communications on peut disposer d'un crayon optique (cellule photo-électrique pour détecter un point éclairé), d'un clavier alphanumérique associé à un générateur de caractères, d'un clavier de fonctions (touches numérotées provoquant des interruptions) ; pour le dessin on a un générateur de vecteurs quelconques et parfois même un générateur d'arcs de cercle.

Nous reviendrons plus en détail sur le fonctionnement d'un terminal graphique dans le premier chapitre, mais nous pouvons déjà esquisser le schéma de notre étude à partir de ce qui précède et de quelques constatations.

Jusqu'à présent les programmes écrits pour utiliser un terminal graphique ont été faits d'une part en langage machine - qui est très peu aisé d'emploi - et d'autre part en vue d'une application bien précise. De ce premier fait l'idée vient naturellement de réaliser un langage de programmation évolué, adapté aux possibilités d'un terminal graphique sans préjudice des applications.

Les possibilités étendues de dialogue offertes alors suggèrent d'utiliser le terminal graphique pour l'écriture et la gestion des programmes graphiques. Autrement dit le projet de système complet serait le suivant :

- Un langage de programmation graphique évolué qui permette toutes les possibilités de l'ordinateur pour les calculs, et du terminal pour l'affichage des dessins et la communication homme-machine (chaque programme, une fois qu'il s'exécute, met en oeuvre son propre langage de communication tel qu'il a été voulu et conçu par le programmeur).

- Un langage de commande permettant :

  * d'écrire un programme
  * de gérer une bibliothèque de programmes, et donc de modifier certains d'entre eux.
  * de lancer l'exécution d'un programme qui prend alors intégralement le contrôle (à l'exception par exemple d'une touche du clavier de fonctions permettant de rendre le contrôle au système).
  * d'interroger le système sur l'état d'un programme que l'on a interrompu, et de modifier la valeur de certaines de ses variables.
  * de reprendre l'exécution après une telle interruption.

La réalisation complète d'un tel projet étant très vaste, nous avons limité notre travail aux points suivants : après un bref rappel technique sur les terminaux graphiques, dont l'étude détaillée se trouve faite par ailleurs (Réf. LUCAS-1), nous exposerons les contours du système complet ; puis suivra l'étude d'un substrat de programmation fourni par IBM (Réf. IBM-3), et son adaptation pour servir à l'écriture du langage de

commande ; nous décrirons ensuite le langage de programmation graphique ; nous terminerons par l'étude de la compilation de ce dernier en insistant sur ses particularités (compilation conversationnelle et incrémentielle, possibilité de retraduire en clair le langage intermédiaire).

CHAPITRE I

RAPPELS GENERAUX SUR LES TERMINAUX GRAPHIQUES.

Les terminaux graphiques appartiennent à la classe des consoles de visualisation dont un autre type est représenté par les terminaux alphanumériques, lesquels, comme leur nom l'indique, ne peuvent recevoir et transmettre que des caractères alphanumériques. Nous nous bornerons volontairement à la description du terminal graphique IBM 2250 Modèle 1 pour lequel sera programmé le système, une étude globale et critique des consoles de visualisation existant déjà par ailleurs (Réf Luc. 1).

COMPOSANTS DU TERMINAL IBM 2250-1

L'écran cathodique

La partie utile de l'écran est un carré de 30 × 30 cm environ qui est subdivisé pour former une grille de 1024 × 1024 points adressables de sorte que l'écart minimal entre deux points est d'environ 0,3 millimètre. Le temps d'affichage d'un point est de 16,8 µs de même que celui nécessaire à l'affichage de points distants d'au plus 113 "écarts minimaux" ; pour des points plus éloignés on a la formule $83{,}2 \times (\frac{N-113}{910}) + 16{,}8$ où N est la déflexion maximale en X ou en Y comptée en "écarts-minimaux" ; par exemple avec N = 1023, le maximum possible, on obtient 100 µs. Cette formule s'applique aussi à l'affichage des vecteurs. Ainsi avec des points moyennement éloignés les uns des autres et une régénération de 40 cycles par seconde on peut afficher 1488 points ou vecteurs (avec 30 cycles par seconde on arrive à 1984 points sans encore craindre de clignotement).

La mémoire d'entretien

Le 2250 est muni d'une mémoire d'entretien de 8192 ortets qui renferme le programme graphique actif et décharge l'ordinateur de la fonction de régénération ; les communications entre cette mémoire et l'unité centrale s'établiront seulement quand on veut modifier une image ou lire une donnée préalablement transmise du clavier alphanumérique à la mémoire d'entretien.

Le clavier alphanumérique et le générateur de caractère associé

Le clavier alphanumérique comporte 63 touches de caractères (alphabétiques, numériques ou spéciaux) et quelques touches particulières. L'existence d'un générateur spécial dispense de programmer le dessin des caractères à l'aide de points et de vecteurs et accélère sensiblement leur vitesse d'affichage ; les caractères existent en deux tailles : les petits, affichés à raison d'un toutes les 14 µs se disposent sur 52 lignes de 74 caractères (16 µs, 35 lignes de 49 caractères pour les grands). Un curseur lumineux ayant l'aspect d'un soulignement est manipulé par le programme ou le clavier (touches saut ("jump"),espacement ("advance") et espace arrière ("back space")) ; il matérialise l'endroit où a-paraîtront les caractères tapés au clavier à la condition que la zone ne soit pas protégée ; cet état "protégé" ou "non-protégé" d'une zone receptrice de caractères est contrôlée par le programme et la touche saut fait avancer le curseur au début de la prochaine zone non-protégée.

Enfin les touches annuler ("cancel") et fin ("end") provoquent des interruptions spécifiques de l'ordinateur.

Le clavier de fonctions

C'est une boîte de 32 touches numérotées de 0 à 31. On peut la recouvrir d'un cache percé de trous et écrire en regard de chaque touche la fonction qu'on y affecte par programme. Lorsqu'on appuie sur une touche, une interruption est provoquée et le numéro de la touche est envoyé à l'ordinateur (cf figure 1). De plus chaque touche comporte un indicateur lumineux qui peut être allumé on éteint à volonté par le programme.

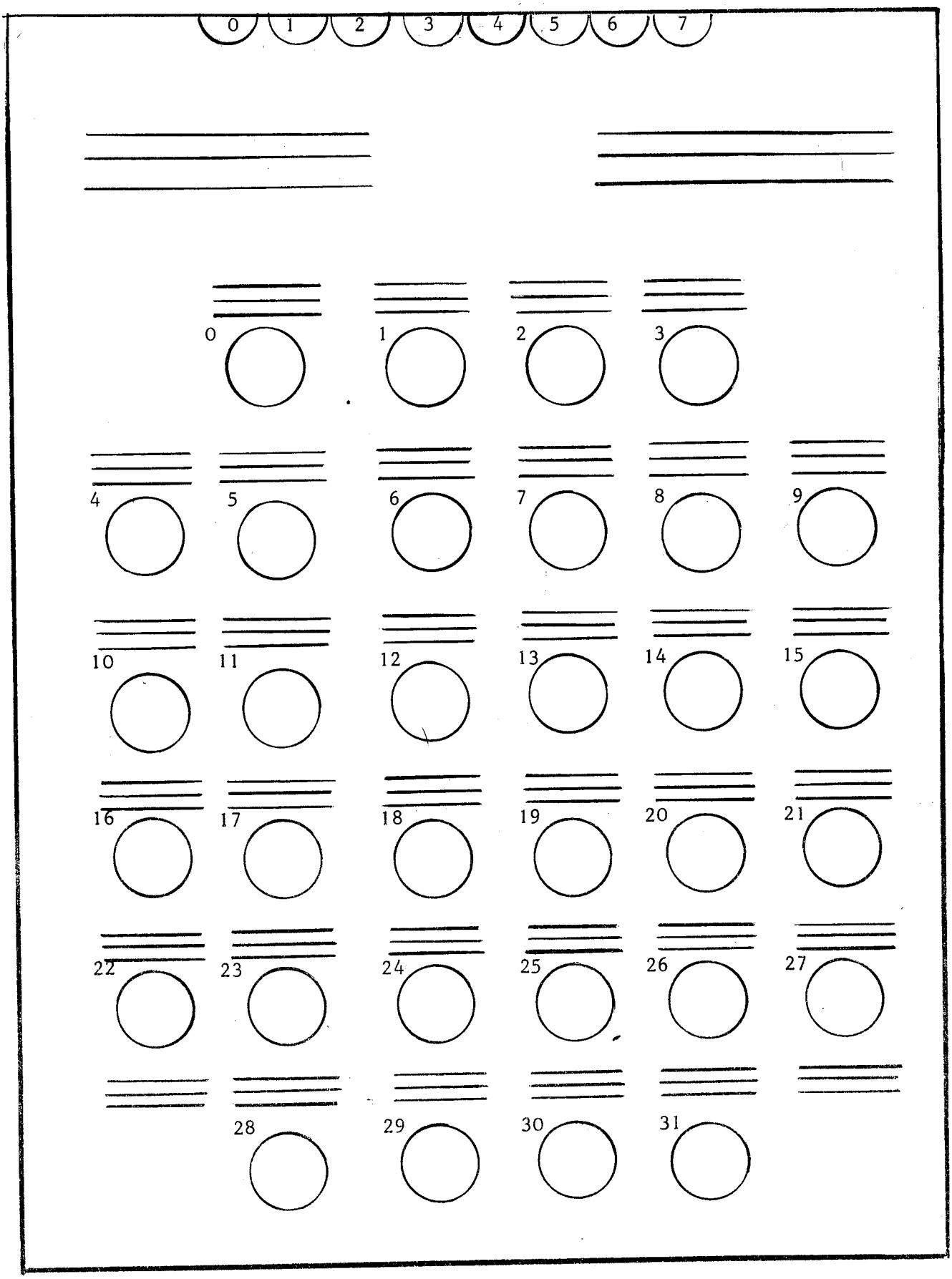


Figure 1 : Cache pour le clavier de fonctions

Le crayon optique

Pour désigner un point du dessin, on place le crayon optique devant ce point et on appuie sur la pédale.

Le terminal graphique signale alors à l'ordinateur cette intervention de l'opérateur, en lui envoyant les informations nécessaires au repérage du point désigné (adresse, dans la mémoire d'entretien, de l'octet de programme qui émet ce point).

LA PROGRAMMATION DE BASE

Le terminal graphique est un dispositif d'entrée/sortie et à ce titre c'est un programme en langage machine 360 dans l'unité centrale qui s'adresse à lui par des "commandes de canal". Par une commande d'écriture on envoie des informations à la mémoire d'entretien, par une commande de lecture on fait l'inverse ; par les commandes de contrôle on peut effectuer diverses actions : allumer ou éteindre des indicateurs lumineux du clavier de fonctions, mettre ou enlever le curseur, spécifier une adresse de la mémoire d'entretien et démarrer ou arrêter le déroulement du programme graphique, actionner un signal sonore pour attirer l'attention de l'opérateur. Enfin une commande de lecture spéciale permet d'analyser les interruptions ou de repérer la position du faisceau lumineux.

Mais nous venons de voir que par une commande d'écriture on envoyait des informations à la mémoire d'entretien et que par une commande de contrôle on spécifiait une adresse et lançait l'exécution d'un "programme graphique" ; cela veut dire que ces informations dans la mémoire d'entretien constituent un véritable programme et que le terminal graphique se comporte comme un petit ordinateur.

En effet ces informations sont organisées en "ordres" et en données associées qui sont décodées et utilisées pour faire mouvoir le faisceau lumineux et tester les détections par le crayon optique. Les ordres se présentent sous la forme d'un octet (cadré sur une adresse paire) contenant la configuration 2 A en hexadécimal, suivi d'un octet indiquant sa fonction suivi éventuellement de données associées dans les octets suivants :

Voici les ordres existant sur le modèle 1 :

GSRT 2A82 initialisation d'un compteur limitant la vitesse de régénération à 40 cycles par seconde

GEPM 2A00 entrée dans le mode graphique pour des points en absolu
GEVM 2A02 ——————————————————— vecteurs ———————

GEPI2 2A04 entrée dans le mode graphique pour des points en relatif
GEVI2 2A05 ——————————————————— vecteurs ———————

GECF 2A40 entrée dans le mode petits caractères pour une zone non-protégée
41 ——————————— grands ———————————————

GECP 2A44 entrée dans le mode petits caractères pour une zone protégée
45 ——————————— grands ———————————————

GESD 2A84 autorisation de détection au crayon optique contrôlée par la pédale

GENSD 2A86 autorisation de détection au crayon optique indépendamment de la pédale

GDPD 2A85 interdiction de détection au crayon optique

GTRU 2AFF rupture de séquence inconditionnelle

GTND 2AFD rupture de séquence si, quoiqu'autorisée, aucune détection au crayon optique n'a eu lieu

GNOP2 2A80 non-opération

GNOP4 2AC0 non-opération (quatre octets)

GEOS 2A81 arrêt programmé (qui envoie une interruption vers l'unité centrale)

On peut regretter de ne pas disposer d'un 2250 Modèle 3 car il comporte un ordre de rupture de séquence avec conservation d'adresse de retour, un ordre de rangement des coordonnées courantes du faisceau lumineux et un ordre de rangement de données, lesquels permettent d'avoir la structure de sous-programme. Cela est très utile pour afficher un même dessin en plusieurs endroits avec une seule séquence de définition en mode relatif. Le modèle 3 comporte aussi quelques ordres pour une utilisation différée des détections au crayon optique.

Malgré l'existence d'un ensemble de macro-opérations (ref IBM 2) pour faciliter la programmation de base, la mise en oeuvre du terminal graphique en langage machine est très lourde : la préparation en mémoire centrale des futurs ordres et surtout la gestion des interruptions est vraiment complexe et il faut des pages et des pages d'écriture pour le moindre dessin comportant quelques actions extérieurs de la part de l'opérateur humain, comme le montre l'exemple suivant :

Le programme écrit plus loin en langage machine se déroulera comme suit à l'éxécution :

1) Sur l'écran apparaît le texte :

"EXEMPLE DE PROGRAMME .... POUR COMMENCER".

2) Ainsi qu'il y est invité, l'opérateur appuie sur la touche de fonction numérotée 1.

3) En haut de l'écran apparaît le texte :

"MONTREZ LE DESSIN .... TERMINER.-" suivi en dessous du zigzag.

4) L'opérateur a le choix entre les actions 5 et 6.

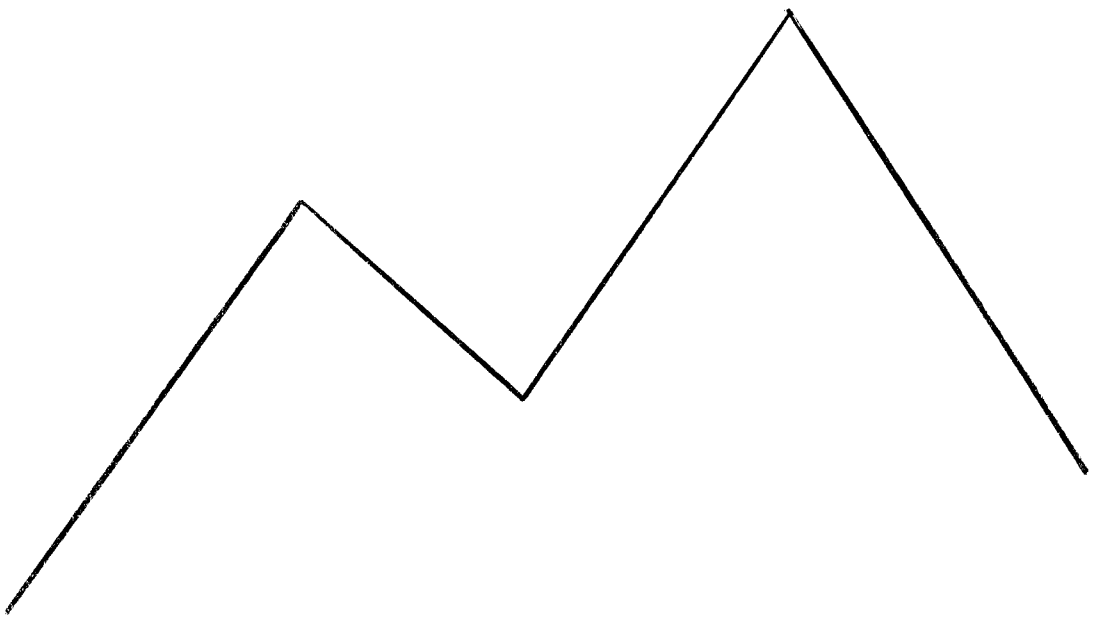
5) L'opérateur tape au clavier alphanumérique la lettre F (qui apparaît sur l'écran à l'emplacement du curseur) et appuie sur la touche "FIN" : le programme est terminé, tout s'éteint.

6) L'opérateur "montre" le zigzag avec le crayon optique : le zigzag disparait ; l'opérateur a le choix entre les actions 5 et 7.

7) L'opérateur appuie sur la touche de fonction numérotée 1 : le zigzag réapparaît et l'opérateur est ramené à la phase 4.

EXEMPLE DE PROGRAMME GRAPHIQUE ELEMENTAIRE.
CE PROGRAMME VA DESSINER DES ZIG-ZAG.
APPUYER SUR LA TOUCHE 1 POUR COMMENCER.

MONTRER LE DESSIN AVEC LE CRAYON OPTIQUE, IL DISPARAITRA
APPUYER SUR LA TOUCHE 1 IL REAPPARAITRA.
TAPEZ F PUIS FIN POUR TERMINER._



Etats successifs de l'écran.

EXEMPLE EN LANGAGE MACHINE.

```
         L      1,OLP
         S      1,SLCA              CALCUL DE L'EMPLACEMENT EN MEMOIRE
         S      1,HUIT              D'ENTRETIEN, DU CURSEUR.
         A      1,BLP
         STH    1,CURSADR
         CALL   GSTOR,(CCEP,PAR3)   PREPARATION DU DESSIN EN DESSOUS
         GUSTOR OUTCB,RUPTURE,4
         XC     DECB4(4),DECB4
         GWRITE DECB4,BUF,GDCB,OCBP  ENVOI DE L'ENSEMBLE EN MEMOIRE
         WAIT   ECB=DECB4            D'ENTRETIEN
         CLI    DECB4,X'7F'
         BNE    ER5

ITER     XC     DECB5(4),DECB5
         GCNTRL DECB5,INS,GDCB,CURSADR  INSERTION DU CURSEUR
         WAIT   ECB=DECB5
         CLI    DECB5,X'7F'
         BNE    ER5
         XC     DECB6(4),DECB6
         GCNTRL DECB6,STR,GDCB,BLP+2  COMMANDE POUR DEMARRER
         WAIT   ECB=DECB6             L'AFFICHAGE
         CLI    DECB6,X'7F'
         BNE    ER5

ATTENTE  ATTNING GACB3,MODE=R        ATTENTE D'INTERRUPTIONS
         CLI    REPONSE+2,X'C3'
         BE     CRAYON
         CLI    REPONSE+3,X'02'
         BE     CLEF1
         CLI    REPONSE+3,X'01'
         BNE    ITER
         XC     DECB7(4),DECB7
         GREAD  DECB7,CUR,GDCB,1,CARAC,CURSADR  LECTURE D'UN
         WAIT   ECB=DECB7                       CARACTERE.
         CLI    DECB7,X'7F'
         BE     SUITE
         CLI    DECB7+16,X'43'
         BNE    ER5
SUITE    CLI    CARAC,C'F'           EST-CE UN F
         BNE    ATTENTE
FIN1     L      13,SAUVE3+4
         POST   ARRET                DEBLOCAGE DU PROGRAMME PRINCIPAL
         RETURN (14,12)              FIN DU SOUS-PROGRAMME.

CLEF1    CLI    REPONSE+2,X'01'
         BNE    ATTENTE
         MVI    NOPTRU+1,X'C0'       TRANSFORME GTRU EN GNOP4
         B      RENVOIE
CRAYON   MVI    NOPTRU+1,X'FF'       TRANSFORME GNOP4 EN GTRU
         B      RENVOIE
ER5      WTO    'ERREUR ENTREE/SORTIE SOUS-PROGRAMME'
         B      FIN1
ER3      WTO    'DEPASSEMENT DE CAPACITE MEMOIRE D''ENTRETIEN'
         B      FIN1
```

```
         PRINT NOGEN
*                                        EXEMPLE EN LANGAGE MACHINE.
EXEMPLE  CSECT
DEBUT    SAVE  (14,12)
         BALR  3,0                       ADRESSABILITE
         USING *,3
         ST    13,SAUVE1+4
         LA    13,SAUVE1

         OPEN  (GDCB)                    INITIALISATION DU TERMINAL GRAPHIQUE
         SPAR  GACB1,PRTY=1              INITIALISATION DE 3 SOUS-PROGRAMMES
         CL    15,ZERO                   DE TRAITEMENT D'INTERRUPTIONS
         BNE   ER1                       QUANT A LEUR LIAISON AVEC LE SYSTEME
         SPAR  GACB2,PRTY=2              SUPERVISEUR.
         CL    15,ZERO
         BNE   ER1
         SPAR  GACB3,PRTY=3
         CL    15,ZERO
         BNE   ER1
MEMOIRE  ASGNBFR GDCB,TAILLE             DEMANDE D'ALLOCATION DE MEMOIRE
         CH    15,ZERO                   D'ENTRETIEN (MAXIMUM DEMANDE : 4096
         BE    TABLEMEM                  OCTETS; MINIMUM REQUIS : 512 OCTETS)
         LH    6,TAILLE
         SH    6,K256
         CH    6,K256
         BE    ER2
         STH   6,TAILLE
         B     MEMOIRE

TABLEMEM BUFINQ GDCB,TABLE,16            ORGANISATION DE LA MEMOIRE
         LH    0,TABLE+4                 D'ENTRETIEN, CONSTRUCTION DE LA
         ST    0,BLP                     TABLE DE POINTEURS OUTCB, ET MISE
         AH    0,RUPTURE+2               A JOUR D'ADRESSES DANS DES ORDRES
         STH   0,RUPTURE+2               GRAPHIQUES.
         STH   0,NOPTRU+2
         LH    0,TABLE+6                 TAILLE DE LA PARTIE DE MEMOIRE
         ST    0,LOA                     D'ENTRETIEN UTILISEE.

         GETMAIN R,LV=(0)                DEMANDE D'UNE ZONE DE MEMOIRE
         ST    1,SLOA                    CENTRALE DE MEME DIMENSION POUR Y
         ST    1,CRSA                    CONSTRUIRE LES ORDRES GRAPHIQUES
         ST    1,OLP                     AVANT ENVOI PAR GWRITE.
         LA    1,ER3
         ST    1,ACRP

         CALL  GSTOR,(DCBP,PAR1)         PREPARATION DU PREMIER ENVOI.

         GCNTRL DECB1,IND,GDCB,LUMIERE   ALLUMAGE DE L'INDICATEUR
         WAIT  ECB=DECB1                 LUMINEUX DE LA TOUCHE DE
         CLI   DECB1,X'7F'               FONCTION NUMERO 1.
         BNE   ER4
         GWRITE DECB2,STR,GDCB,CCBP      ECRITURE DU PREMIER MESSAGE
         WAIT  ECB=DECB2                 SUR L'ECRAN.
         CLI   DECB2,X'7F'
         BNE   ER4
```

```
RELIRE   XC    DECB3(4),DECB3
         GREADR DECB3,MIP,GDCB,NUMERO  LECTURE DE LA TOUCHE 1.
         WAIT  ECB=DECB3
         BNE   ER4
         CLI   NUMERO,B'C1000000'  EST-CE UNE TOUCHE DE FONCTION
         BNE   RELIRE
         CLI   NUMERO+1,1          EST-CE CELLE NUMEROTEE 1
         BNE   RELIRE
         WAIT  ECB=ARRET           MISE EN ATTENTE DU PROGRAMME
*                                  PRINCIPAL.

         RLSEEFR GDCB,ALL          REPRISE DU PROGRAMME PRINCIPAL :
FINFIN   DAR   (GACB1,GACB2,GACB3) LIBERATION DE LA MEMOIRE D'ENTRETIEN
         CL    15,ZERO             ET INHIBITION DES SOUS-PROGRAMMES DE
         BNE   ER1                 TRAITEMENT DES INTERRUPTIONS.

SORTIE   CLOSE GDCB                DECONNECTION DU TERMINAL GRAPHIQUE.
         L     0,LGA
         L     1,SLGA
         FREEMAIN R,LV=(0),A=(1)   LIBERATION DE LA MEMOIRE CENTRALE.
         L     13,SAUVE1+4
         RETURN (14,12)            RETOUR AU SYSTEME SUPERVISEUR.

ER1      WTO   'ERREUR SPAR OU DAR'
         B     SORTIE
ER2      WTO   '256 OCTETS DE MEMOIRE D''ENTRETIEN : TROP PEU'
         B     FINFIN
ER4      WTO   'ERREUR ENTREE/SORTIE PROGRAMME PRINCIPAL'
         B     FINFIN

ASYNERR  SAVE  (14,12)             EN CAS D'ERREUR DE FONCTIONNEMENT DU
         DROP  3                   TERMINAL GRAPHIQUE LE SYSTEME
         BALR  4,0                 SUPERVISEUR DONNE LE CONTROLE A CE
         USING *,4                 SOUS-PROGRAMME.
         ST    13,SAUVE2+4         CELUI-CI, DANS L'EXEMPLE, SE BORNE
         LA    13,SAUVE2           A ENVOYER UN MESSAGE A LA CONSOLE DU
         WTO   'ERREUR ASYNCHRONE' 360 ET DE RENDRE LE CONTROLE.
         L     13,SAUVE2+4
         RETURN (14,12)

SOUSPROG SAVE  (14,12)             LE SYSTEME SUPERVISEUR DONNERA LE
         DROP  4                   CONTROLE A CE SOUS-PROGRAMME LORSQUE
         BALR  5,0                 L'ON AURA APPUYE SUR LA TOUCHE DE
         USING *,5                 FONCTION NUMEROTEE 1.
         ST    13,SAUVE3+4
         LA    13,SAUVE3

RENVOIE  MVC   CLP,SLGA            REINITIALISATION POINTEUR

         GUSTOR OUTCB,GDEBUT,2     PREPARATION DU TEXTE EN HAUT
         CALL  GSTOR,(OCBP,PAR2)
```

```
PAR1     DC    2F'0'
PAR2     DC    2F'0'                    MEMOIRES UTILISEES PAR GSTOP
PAR3     DC    2F'0'
NUMERO   DC    F'0'
CURSACR  DC    F'0'                     DIVERSES MEMOIRES DE TRAVAIL
CARAC    DC    F'0'
ZERO     DC    F'0'
HUIT     DC    F'8'
TAILLE   DC    H'4096'
K256     DC    H'256'
SAUVE1   DC    16F'0'
SAUVE2   DC    18F'0'                   ZONES DE SAUVEGARDE
SAUVE3   DC    18F'0'
LUMIERE  DC    B'01000000'
         DC    3X'00'


         GINIT BLIM=4096                PREMIER MESSAGE
GDEBUT   GSRT
         GOPD
         GEPM
         GDV   0,2000,B
         GECF  B
         GTXT  'EXEMPLE DE PROGRAMME GRAPHIQUE ELEMENTAIRE.'
         GCNL  1
         GTXT  'CE PROGRAMME VA DESSINER DES ZIGZAG.'
         GCNL  1
         GTXT  'APPUYEZ SUR LA TOUCHE 1 POUR COMMENCER.'
RUPTURE  GTRU  GDEBUT
         GODEL GDEBUT,PAR1


ANNULER  SAVE  (14,12)                  SI ON APPUIE SUR LA TOUCHE ANNULER
         DROP  9                        LE SYSTEME SUPERVISEUR DONNERA LE
         BALR  6,0                      CONTROLE A CE SOUS-PROGRAMME QUI
         USING *,6                      MET DANS LA MEMOIRE ARRET
         ST    13,SAUVE4+4              L'INDICATEUR QUI FAIT SORTIR LE
         LA    13,SAUVE4                PROGRAMME PRINCIPAL DE SON ETAT
         POST  ARRET                    D'ATTENTE.
         L     13,SAUVE4+4
         RETURN (14,12)


SAUVE4   DC    18F'0'
GACB1    SAEC  EP=ASYNERR,DCB=GDCB,COMAREA=REPONSE,ATTNTYP=(R,AE),      /
               PFKMSK=NULL

GACB2    SAEC  EP=ANNULER,DCB=GDCB,COMAREA=REPONSE,PFKMSK=NULL,         /
               ATTNTYP=(R,CANCEL)

GACB3    SAEC  EP=SOUSPROG,DCB=GDCB,COMAREA=REPONSE,PFKMSK=(R,1),       /
               ATTNTYP=(R,EAC,LP)

GDCB     DCB   DSORG=GS,MACRF=(GRC,WC),DDNAME=CR,CTYPE=BASIC
               *,*** IM0072  EXLST NOT SPECIFIED-RESET TO 0
```

```
                *,***  IHB063  DDNAME SPORT-PADDED TO 8 CHAR
OUTCB    EQU    *
SLOA     DS     F
LCA      DS     F                    POINTEURS ET TABLES
ACRP     DS     F
CRSA     DS     F
CLP      DS     F
BLP      DS     F

TRAVAIL  DS     16F

CCBP     DC     A(OUTCB)
         DC     A(TRAVAIL)

TABLE    DC     4F'0'
REPONSE  DC     4F'0'
ARRET    DC     F'0'


DEUXIEME GDPD
         GEPM
         GDV    0,4000,B
         GECP   B
         GTXT   'MONTREZ LE DESSIN AVEC LE CRAYON OPTIQUE, IL'
         GCNL   1
         GTXT   'DISPARAITRA.'
         GCNL   1
         GTXT   'APPUYEZ SUR LA TOUCHE 1 IL REAPPARAITRA.'
         GCNL   1
         GTXT   'TAPEZ F PUIS FIN POUR TERMINER.'
         GECP   B
         GTXT   '     '
NOPTRL   GNOP4  GDEBUT
         GODEL  DEUXIEME,PAR2


DESSIN   GESD
         GEVM
         GDV    0,0,P
         GDV    1000,2000,U
         GDV    2000,1000,U
         GDV    3000,3000,U
         GDV    4000,500,U
         GODEL  DESSIN,PAR3


         END    DEBUT
```

CHAPITRE II

PRESENTATION DU SYSTEME ET LANGAGE DE COMMANDE

Pour que l'on puisse voir clairement le rôle du langage de commande et celui du langage de programmation graphique, pour comprendre leur spécificité et pour préciser la notion de langage de communication, nous allons décrire la démarche d'un programmeur au cours d'une séance de travail. Ayant par exemple l'idée d'étudier un problème de circulation urbaine il s'installe devant le terminal graphique.

Dans un premier temps le programmeur va écrire son programme graphique, et faire appel pour cela au langage de commande. Celui-ci se sert des touches du clavier de fonction - dont chacune a un rôle unique (cf. figure 2) - et de messages envoyés sur l'écran ; en réponse le programmeur envoie des informations par le clavier alphanumérique ou appuie sur certaines touches du clavier de fonctions pour préciser ses intentions : le programmeur appuie sur la touche Début, se nomme et donne sont mot de passe. Puis il appuie sur la touche Nommer et indique le nom de son nouveau programme. Ensuite il appuie soit sur la touche Déclaration, soit sur la touche Instruction et se met à écrire son programme. Pour ce faire il utilise l'écran comme une feuille de papier et il se déplace à travers les lignes et les pages à l'aide des touches Avancer, Reculer, Descendre, Monter, Corriger, Ajouter et Enlever ; de plus les erreurs éventuelles lui sont signalées, auxquelles il peut remédier sur le champ. Quand il a terminé d'écrire son programme il appuie sur Fin ; à ce moment là il pourrait lancer l'exécution du programme mais il décide de s'accorder une pose de sorte qu'appuyant deux foix sur Fin il quitte le système et son programme est rangé en bibliothèque.

Quand il revient, il redonne son identité (quelqu'un d'autre peut avoir travaillé entre temps), puis après avoir nommé son programme il appuie sur Exécuter, et son programme prend le contrôle ; il n'y a que la touche Arrêter, celle numérotée 0, qui lui permet de renouer avec le système et le langage de commande. (Il peut avoir mis un cache sur le clavier de fonctions pour indiquer l'utilisation qu'il fera des touches).

La façon dont il se sert du crayon optique, du clavier alphanumérique et des touches de fonction pour modifier son réseau de circulation urbaine, introduire des incidents, faire varier le fonctionnement des feux tricolores, etc... constituer un "Langage de Communication" : le programmeur l'a adapté à ses besoins en utilisant au mieux les possibilités du terminal.

Supposons que le programmeur ait utilisé un certain paramètre sans se donner la possibilité de le modifier et qu'il s'aperçoive qu'il aurait avantage à lui donner une autre valeur. Il appuie alors sur Arrêter, l'image disparaît, il appuie sur Changer et définit la nouvelle valeur du paramètre, appuie sur Fin et sur Repartir : le programme graphique reprend son déroulement normal et se termine.

A ce moment le contrôle est revenu au langage de commande et le programmeur, avant de s'en aller, veut faire quelques modifications au programme : les touches Déclaration et Instruction et toutes celles qui leur sont rattachées lui donnent cette possibilité comme dans le premier temps.

Quand il a fini il quitte le système en appuyant sur la touche Fin trois ou quatre fois (cf le tableau hiérarchisé de la figure 3).

Cet exemple ayant précisé le rôle du langage de commande, en voici la description : (Réf. LEC-1).

	0 Arrêter	1 Début	2 Fin	3 Exécuter	
4 Repartir	5	6 Table	7 Nommer	8 Copier	9 Supprimer
10 Changer	11 Reculer	12 Avancer	13 Déclarations	14 Instruc- tions	15 Corriger
16 Interroger	17 Monter	18 Descendre	19 Enlever	20 Ajouter	21
22	23	24	25	26	27
	28	29	30	31	

Figure 2 : Disposition des commandes sur le clavier de fonctions.

Le langage de commande n'autorise à un moment donné que les commande compatibles avec le niveau courant. Pour cela on place les commandes dans une structure hiérarchisée, c'est à dire qu'elles autorisent en même temps un groupe de commandes compatibles ; d'autres font remonter dans la hiérarchie, c'est à dire qu'elles font revenir à l'état où l'on était avant la commande qui a permis de descendre ; d'autres enfin n'ont aucun effet sur la hiérarchie, elles sont dites stationnaires. Un indicateur lumineux placé sous la touche permet de mettre en valeur toutes les commandes autorisées à un moment donné. Si l'on essaie d'envoyer une commande non autorisée, c'est à dire si l'on enfonce une touche dont l'indicateur n'est pas allumé, le système ignore purement et simplement l'interruption correspondante.

Dans le cas où le nom de la commande n'est pas une indication suffisante sur l'action que l'on désire, le système pose des questions sur l'écran ; on y répond au moyen du clavier alphanumérique.

DESCRIPTION DES COMMANDES

Début

Début du travail d'un nouveau programmeur. Le système lui demande son nom et son mot de passe, afin de vérifier s'il est autorisé à travailler et de se référer au fichier de ses programmes. Un "programmeur maître" à seul la possibilité de modifier la liste des programmeurs ayant accès au système.

Commande descendante autorisant Nommer, Table et Fin.

Fin

Commande autorisée dans tous les cas, et ayant toujours pour effet de faire remonter dans la hiérarchie, terminant ainsi le traitement en cours.

Table

Consultation de la table des noms de programmes (ou des noms de programmeurs si c'est le programmeur maître qui travaille).

Cette table apparaît sur l'écran par deux pages de vingt lignes à la fois. Pour chaque programme elle indique son nom, la date à laquelle il a été créé, la date à laquelle il a été modifié pour la première fois, et son encombrement. Pour chaque programmeur, elle donne son nom, son mot de passe et la date à laquelle il a été introduit.

Commande descendante, autorisant Avancer, Reculer et Fin.

Avancer

Avancer d'une page dans le texte affiché. Si l'on affichait sur l'écran les pages n-1 et n d'une table ou d'un programme, cette commande fait afficher les pages n et n+1, si elles existent.

Commande stationnaire.

Reculer

Reculer d'une page dans le texte affiché. Si l'on affichait sur l'écran les pages n et n+1 d'un texte ou d'une table, cette commande fait afficher les pages n-1 et n, si elles existent.

Commande stationnaire.

Nommer

Début du travail sur un programme précis. Le système demande le nom du programme sur lequel on désire travailler. S'il existait déjà, il l'indique et donne les dates de création et de dernière modification, sinon il indique que le programme est nouveau. Pour le programmeur maître, cette

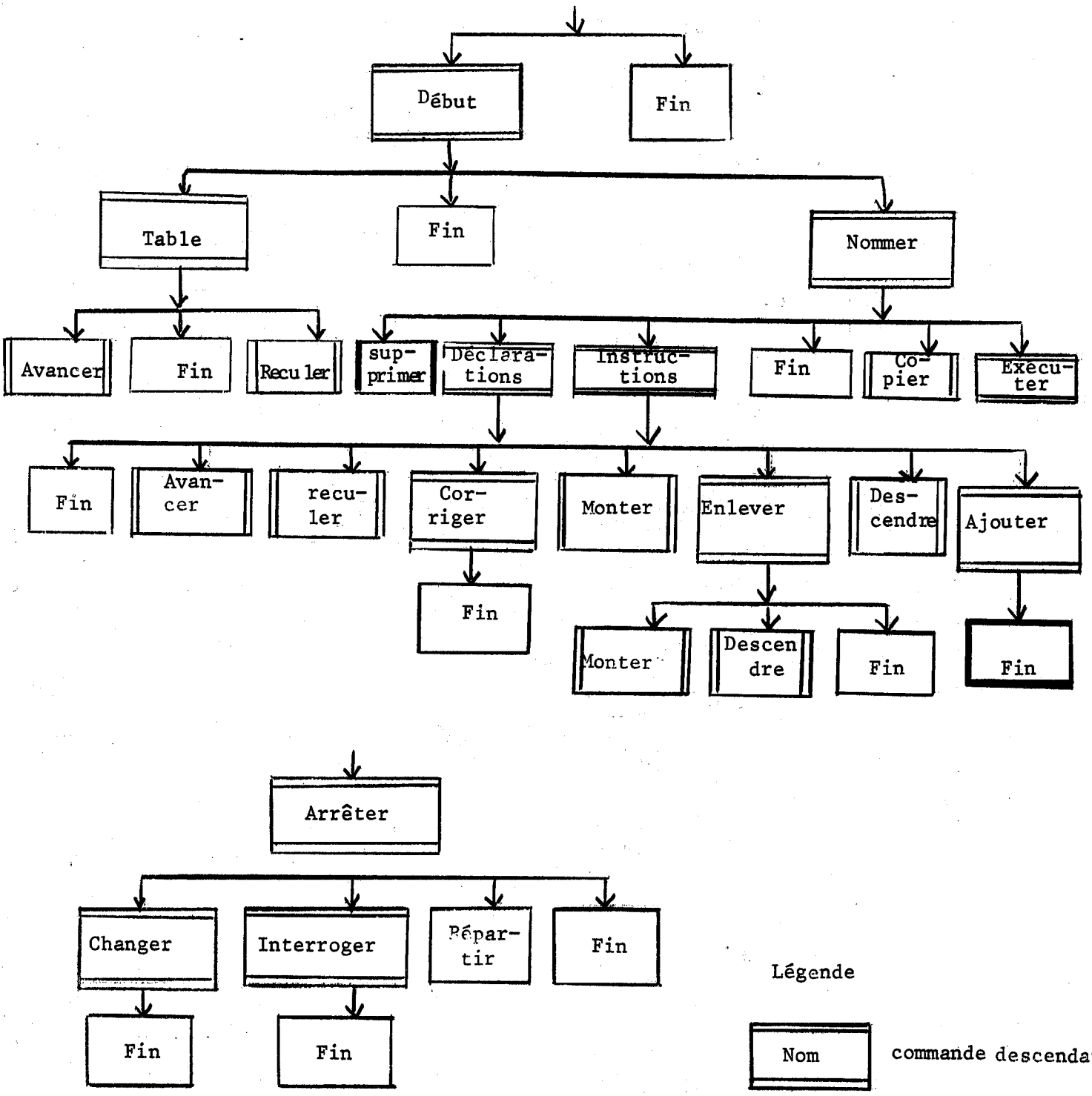


Figure 3 : Hiérarchie des commandes

commande sert à donner le nom d'un programmeur qu'il désire introduire ou retirer.

Commande descendante, autorisant Déclarations, Instructions et Fin, plus Exécuter et Supprimer si le programme est ancien, Copier s'il est nouveau. Pour le programmeur maître, elle autorise Supprimer et Fin.

Déclarations

Début des modifications aux déclarations du programme nommé. Les deux premières pages (si elles existent) apparaissent sur l'écran, avec deux marqueurs de part et d'autre de la ligne supérieure. Les marqueurs indiquent constamment la ligne sur laquelle on travaille, et toutes les modifications nécessitent donc d'abord l'utilisation des commandes qui permettent de les déplacer pour les amener à l'endroit voulu.

Commande descendante, autorisant Avancer, Reculer, Descendre, Monter, Corriger, Ajouter, Enlever et Fin.

Instructions

Début des modifications aux instructions du programme nommé. Les deux premières pages (si elles existent) apparaissent sur l'écran, avec les deux marqueurs de part et d'autre de la ligne supérieure. Contrairement aux déclarations, on travaille par instructions plutôt que par lignes ; en effet, s'il n'y a jamais qu'une instruction par ligne, une instruction peut occuper plusieures lignes, et doit pourtant être considérée comme un tout, puisqu'elle est toujours compilée en entier. Quand dans la suite de la description nous parlerons de lignes, cela pourra donc signifier que l'on parle d'instruction si l'on est en train de les traiter.

Commande descendante autorisant les mêmes commandes que la commande Déclarations.

Supprimer

Effacement du programme nommé. La place occupée dans le fichier est récupérée. Pour le programmeur maître, cette commande permet de retirer un programmeur ; s'il le désire, on lui présente alors un à un tous les programmes de ce programmeur afin qu'il décide de les conserver en les attribuant à un autre programmeur ou de les supprimer.

Commande ascendante, ramenant à l'état où l'on était après la commande Début.

Copier

Recopie d'un programme existant, pour constituer le programme nommé. Le programme nommé doit être nouveau, sinon cette commande n'est pas autorisée Le système demande le nom du programme à recopier. On peut ensuite modifier cette copie sans toucher à l'original.

Commande stationnaire.

Exécuter

Exécution du programme nommé.

Commande descendante, en ce sens que durant l'exécution la commande Arrêter reste toujours utilisable, mais tous les autres dispositifs de communication, en particulier les touches 1 à 31, sont laissés à la libre disposition du programme. Cette commande se termine quand le programme lui-mêm

est terminé (par exécution de l'instruction "FINGRAPHIQUE"), ou quand l'utilisateur envoie les commandes Arrêter puis Fin.

Descendre

Descendre les marqueurs d'une ligne. Le mot "ligne" est pris au sens large indiqué au moment de la description de la commande Instructions. Si l'on est en bas de l'écran ou en fin de texte, cette commande est sans effet. Commande Stationnaire.

Monter

Remonter les marqueurs d'une ligne. Si l'on est en haut d'écran, cette commande est sans effet.

Commande stationnaire.

Corriger

Modification de lignes, sans adjonctions ni suppressions. La première ligne modifiée est celle devant laquelle sont les marqueurs ; le système place un curseur en tête de cette ligne. On effectue les modifications au moyen du clavier alphanumérique, dont les touches permettent de déplacer le curseur sous les caractères puis de remplacer les caractères qu'il indique. Après qu'une ligne ait été correctement modifiée, le système la réécrit pour indiquer comment il l'a comprise, puis les marqueurs descendent devant la suivante, comme si l'on avait utilisé la commande Descendre.

Commande descendante, autorisant uniquement Fin.

Ajouter

Adjonction de lignes. Une ligne blanche est insérée sous la ligne désignée par les marqueurs, qui descendent se placer devant cette nouvelle ligne. Un curseur vient se placer en tête, et l'on compose l'instruction ou la déclaration au moyen du clavier alphanumérique. Les marqueurs ne quittent la ligne sur laquelle on travaille que si elle st syntaxiquement correcte ; dans le cas d'instructions, le système ne commence l'analyse syntaxique et la traduction que quand l'instruction est complète ; si donc elle ne tient pas sur une seule ligne, il ajoute des lignes blanches jusqu'à ce que l'on ait écrit le point-virgule final. Si une ligne contient une erreur, on doit la corriger de la même façon qu'avec la commande Corriger, et les marqueurs restent en place. La commande Ajouter sert en particulier à composer de nouveaux programmes.

Commande descendante autorisant uniquement Fin.

Enlever

Suppression de lignes. La suppression commence à la ligne pointée par les marqueurs, cette ligne comprise. On déplace les marqueurs au moyen des commandes Descendre et Monter ; la suppression n'est faite qu'au moment où l'on envoie la commande Fin, ce qui réduit le risque de suppressions involontaires ; les lignes sont supprimées jusqu'à celle pointée par les marqueurs, cette ligne exclue. Si la nouvelle position des marqueurs est antérieure ou égale à leur ancienne position, la commande reste sans effet.

Commande descendante, autorisant Descendre, Monter et Fin.

Arrêter

Interruption du programme en cours d'exécution. Le contrôle de toutes les possibilités du terminal graphique est rendu au système, pour que l'utilisateur puisse vérifier et modifier l'état de son programme ; en particulier les touches de fonction perdent la signification qu'avait pu leur donner le programme, et l'image qu'il affichait disparaît de l'écran.

Commande descendante, autorisant Changer, Interroger, Repartir et Fin. Dans ce cas précis, la commande Fin signifie que l'on veut terminer l'exécution du programme et rendre le contrôle au système comme après la commande Nommer. En ce sens la commande Arrêter peut être considérée comme au même niveau que la commande Exécuter.

Interroger

Demande d'information sur le programme interrompu. Le système demande le nom de la quantité dont on désire connaître la valeur ; l'utilisateur répond par le nom d'une des variables de son programme, ou de certaines quantités définies par le système ; celui-ci répond en affichant la valeur voulue, dans la représentation appropriée.

Commande descendante, autorisant uniquement Fin.

Changer

Exécution immédiate d'une instruction, sans modification au programme. L'utilisateur écrit une instruction simple et complète, qui est analysée et traduite, puis conservée pour être exécutée en priorité au moment de la reprise du programme. On peut utiliser cette possibilité pour modifier la valeur d'une variable, l'état d'un objet ou d'une figure, etc... Ceci ne modifie

cependant pas le programme lui-même, c'est-à-dire que les instructions ne seront exécutées qu'une seule fois, avant que le programme ne reprenne. Si l'on veut faire une correction réelle, on ne peut pas le faire pendant l'exécution du programme.

Commande descendante, autorisant uniquement Fin.

Repartir

Reprise de l'éxécution du programme interrompu. Le système demande le nom d'une étiquette du programme, où doit reprendre l'exécution ; l'utilisateur peut aussi utiliser les symboles 'COMMENCEMENT' et 'INTERRUPTION', pour demander la reprise respectivement au début du programme et à l'endroit où il a été interrompu.

Commande ascendante, rendant complètement le contrôle au programme, comme après la commande Exécuter.

Avant de décrire le langage graphique, nous allons décrire un ensemble de sous-programmes fournis par IBM sous le noms de GSP ("Graphic Subroutines Package") qui nous servira à deux niveaux dans la réalisation du système : adapté au langage d'assemblage 360 sous forme de macro-instructions il servira à l'écriture du programme de commande ; utilisé pour la partie proprement graphique du langage de programmation il sera utilisé par le compilateur dans sa phase d'interprétation.

CHAPITRE 3

DESCRIPTION DES SOUS-PROGRAMMES "G.S.P."

Les sous-programmes ont été conçus pour utiliser le terminal graphique IBM 2250 modèle 1 ou modèle 3 à l'intérieur d'un programme écrit en FORTRAN ou d'un programme écrit dans le langage d'assemblage de l'ordinateur 360. Ils sont appelés au moyen de l'instruction classique :

CALL NOM (PARAM1, PARAM2, ...)

où NOM est le nom du sous-programme et où PARAM1, PARAM2, etc... sont les noms des mémoires contenant les paramètres.

A chaque sous-programme correspondent un certain nombre de paramètres qui doivent être donnés dans un ordre fixe (paramètres de "position").

Le premier intérêt qu'offre G.S.P. est de décharger le programmeur de la programmation graphique élémentaire ; son emploi permettra donc un gain de temps appréciable dans l'écriture du système. Le deuxième intérêt, dont nous allons parler maintenant, a trait d'une part à la structuration de l'image, et d'autre part à l'organisation du traitement des interruptions provoquées par les actions de l'opérateur.

ORGANISATION DE L'IMAGE

La solution adoptée par G.S.P. consiste à individualiser sous le nom de figure ("graphic data set") un ensemble d'éléments de dessin : on définit une figure en lui donnant un nom et en limitant la portion de mémoire d'entretien dont elle pourra disposer lors de son affichage (INGSP[1]).

On a ensuite à sa disposition pour construire la figure des sous-programmes de génération d'objets - par objets nous entendons les points (PPNT), les vecteurs (PLINE, PSGMT), les chaînes de caractères (PTEXT)-,

(1) Nous mettons entre parenthèses le nom du sous-programme correspondant (cf glossaire à la fin du chapitre).

et des sous-programmes définissant ou modifiant un certain nombre de caractéristiques globales de la figure, caractéristiques qui sont :

* type des coordonnées en x ou en y : mode absolu ou relatif (par rapport à la position actuelle du faisceau lumineux), représentation entière ou réelle (SDATM).
* portion de l'écran allouée à la figure (SGDSL).
* échelle (SDATL).
* type des caractères : petits ou grands, protégés ou non-protégés (SCHAM).
* manière de traiter les parties de dessin qui sortent des limites soit de l'écran, soit de la portion d'écran allouée (SSCIS).

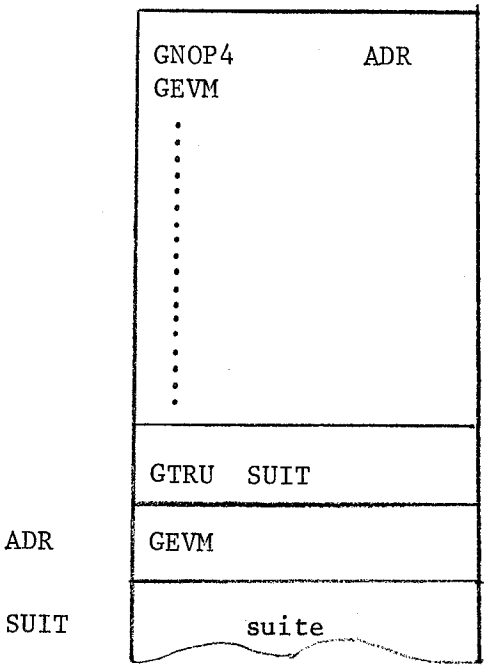
Ceci étant fait, un sous-programme spécial (EXEC) provoque l'affichage de la figure.

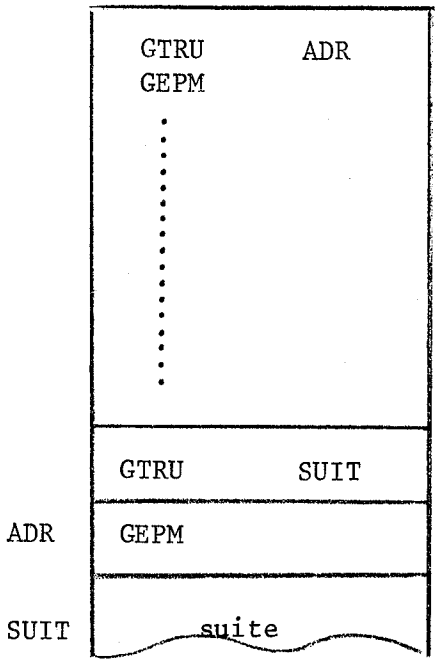
Cependant comme il faut pouvoir manipuler non seulement la figure dans son ensemble mais aussi les objets qui la composent, il y a, dans chaque sous-programme de génération d'objet, la possibilité de mettre un paramètre qui sert à lui donner un nom unique et un autre paramètre qui sert à lui donner un numéro qui, lui n'est pas forcément unique. Ceci permet de se référer à ces objets pour les actions suivantes : extinction (OMIT), rallumage (INCL), suppression (RESET) et modifications.

De plus on peut créer un degré intermédiaire entre la figure entière et l'objet élémentaire : en faisant précéder et suivre une série d'objets générés consécutivement, respectivement par les appels de sous-programmes "début de séquence" (BGSEQ) et "fin de séquence" (ENSEQ), on constitue une séquence qui pourra subir globalement les actions énoncées précédemment.

Du point de vue de la mémoire d'entretien on peut afficher au même moment jusqu'à 23 figures distinctes ; du point de vue du programme en mémoire centrale on peut définir pour chaque figure jusqu'à 49 figures équivalentes, mais on ne peut à un moment donné n'en envoyer qu'une sur l'écran.

Voici quelques schémas montrant l'organisation en mémoire d'entretien des images, des séquences et des objets.





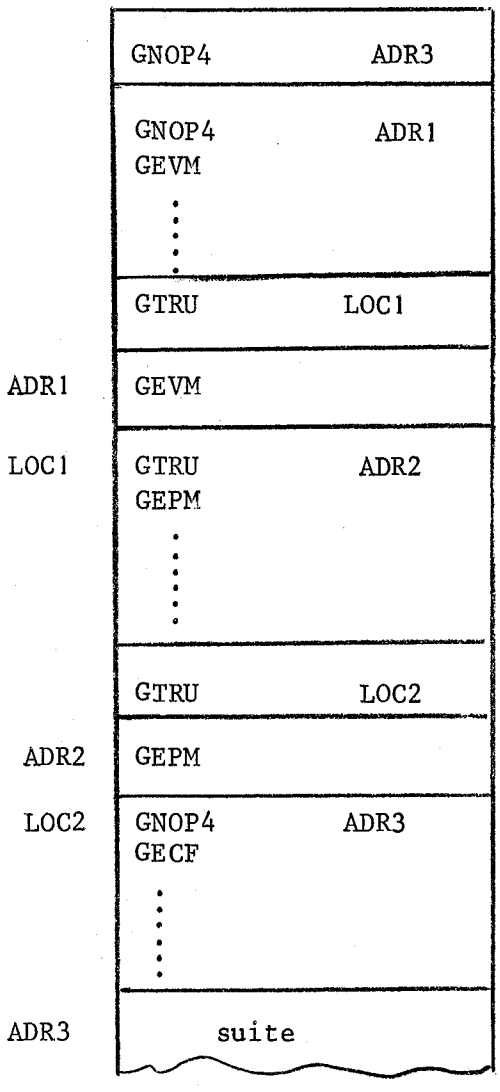
Objet allumé (lignes)

Ici la non-opération GNOP4 fait qu'on exécute l'affichage des lignes, et, par GTRU SUIT on passe directement à la suite.

objet éteint (points)

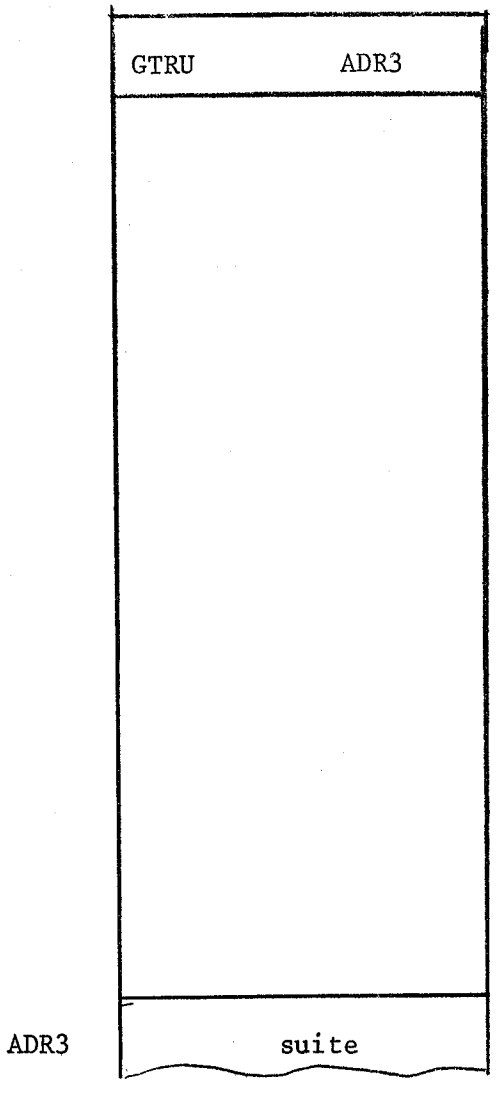
Ici par GTRU ADR on saute l'affichage des points mais l'ordre GEPM figurant en ADR amène le faisceau lumineux là où il devrait être si l'on avait affiché les points.

Donc on voit qu'on allume ou éteint un objet en changeant GNOP4 en GTRU et réciproquement.



Séquence allumée

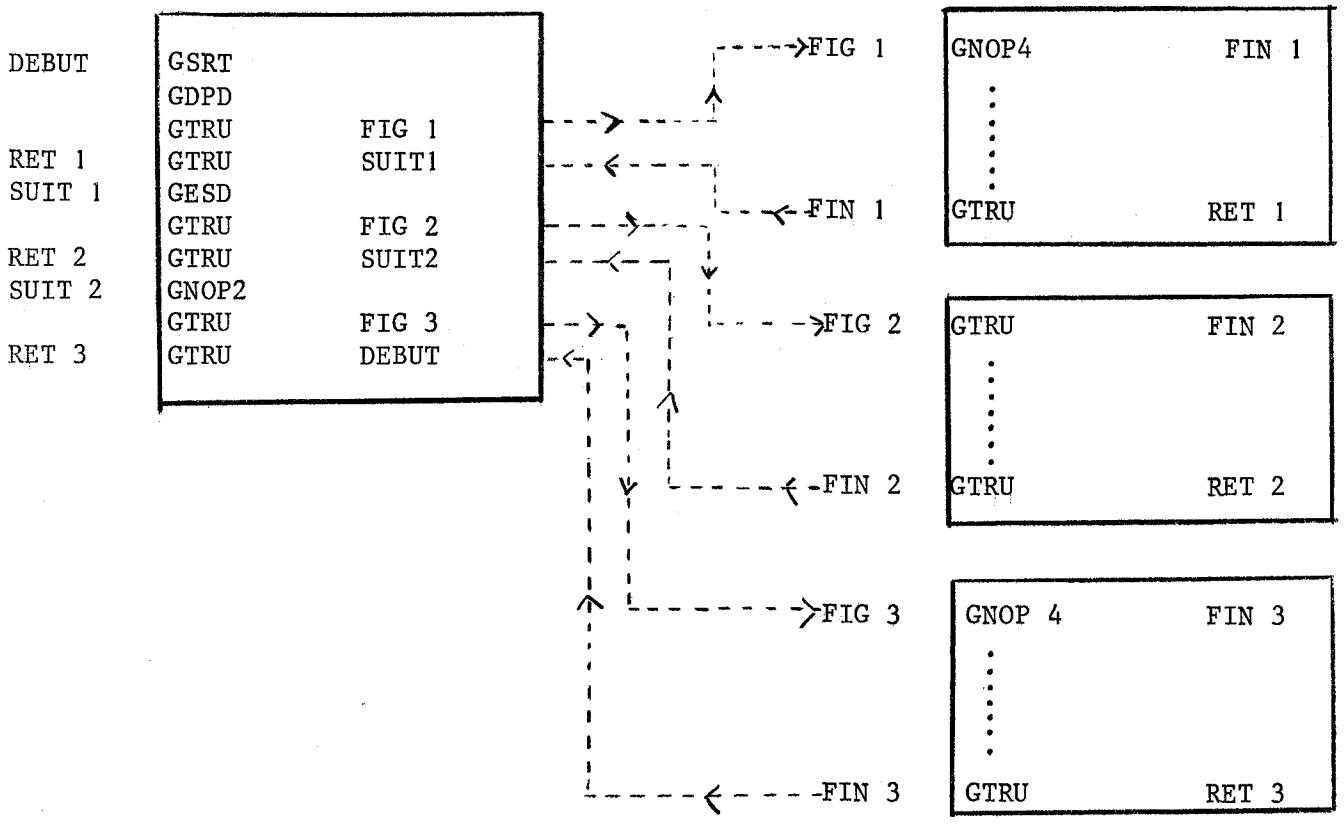
Elle est formée de lignes, de points (éteints) et de caractères.



Séquence éteinte

On remarque qu'il n'y a pas d'ajustement du faisceau lumineux pour une séquence éteinte (ni d'ailleurs pour des caractères).

Pour les figures il y a une chaîne principale en début de mémoire d'entretien avec débranchement vers chaque figure affichée ; en tête de la chaîne se trouve l'ordre d'initialisation du compteur de temps qui règle l'espacement des régénérations ; avant le transfert à chaque figure on a soit un GNOP2 (non opération), soit un ordre gouvernant les détections au crayon optique : autorisation (GESD) ou interdiction (GDPD).



Situation où l'on a 3 figures dont la deuxième éteinte

ORGANISATION DU TRAITEMENT DES INTERRUPTIONS

Les causes d'interruption sont ainsi numérotées :

1 à 31 : les touches du clavier de fonction

32 : la touche FIN du clavier alphanumérique

33 : la touche ANNULER du clavier alphanumérique

34 : le crayon optique

35 : l'arrêt programmé.

On a accès aux renseignements afférents à une interruption par un sous-programme d'interrogation (RQATN) ; on y précise d'une part où recueillir ces renseignements, et d'autre part la, ou les interruptions qui nous intéressent, grâce à leur numéro.

De plus on mentionne dans ce sous-programme le nom d'une entité spéciale à laquelle se rattachent ces interruptions, le "niveau".

En effet, pour pouvoir différer dans le temps l'exploitation des interruptions, et pouvoir donc les mémoriser et les relier à une tranche de temps (ou de programme), on crée au fur et à mesure des besoins ces niveaux (CRATL). Le dernier créé est dit actif et c'est à lui que se rapporteront les interruptions qui surviennent.

Les niveaux forment ainsi une hiérarchie ; un sous-programme permet de modifier cette hiérarchie (MPATL), et un autre de supprimer des niveaux, rendant actif le dernier non supprimé (ENATL).

On dispose des deux sous-programmes nécessaires à l'autorisation (ENATN) ou à l'interdiction (DSATN) des interruptions, que l'on utilisera à volonté pour chaque niveau.

Par ailleurs, pour ce qui est des interruptions dues aux détections par le crayon optique, il y a, indépendamment de ce que l'on vient

de voir, un sous-programme pour autoriser ou interdire les détections sur telle ou telle figure (SLPAT).

Un programme graphique simple peut se contenter d'un seul niveau on a encore la souplesse de manoeuvre due à la faculté de modifier les interruptions permises.

Un exemple d'utilisation de plusieurs niveaux est donné par un programme principal qui appelle un sous-programme : le programme principal a son propre niveau ; chaque fois qu'on se transfère un sous-programme, on crée un nouveau niveau qui devient actif : les interruptions autorisées qui surviennent lui sont rattachées et on les exploite ; au moment de retour au programme principal on supprime ce niveau, ce qui a pour effet de rendre nouveau actif le niveau du programme principal. Ainsi, s'agissant du clavier de fonctions, une même touche pourra avoir une signification dans le programme principal, et une autre dans le sous-programme ; en particulier si on a appuyé sur une telle touche pendant le déroulement du sous-programme et que l'on n'a pas exploité ce fait, le programme principal n'en sera pas perturbé ensuite.

```
        PRINT NOGEN
VOICI LA PROGRAMMATION DU MEME EXEMPLE QU'AU CHAPITRE 2
AVEC LES SOUS-PROGRAMMES G.S.P.
EMPLE   CSECT
BUT     SAVE  (14,12)
        BALR  3,0
        USING *,3
        ST    13,SAUVE+4
        LA    13,SAUVE

        CALL  INGSP,(GEXEMPLE,NUL)
        CALL  INDEV,(GEXEMPLE,UNITE,TERMINAL,K4096)
        CALL  INGDS,(TERMINAL,PREMIER)
        CALL  INGDS,(TERMINAL,DEUXIEME)
        CALL  PTEXT,(PREMIER,PHRASE,LG,NUL,NUL,NUL,X1,Y1)
        CALL  PTEXT,(DEUXIEME,LIGNES,LG2,NUL,NUL,NUL,X2,Y2)
        CALL  SCHAM,(DEUXIEME,LIBRE)
        CALL  PTEXT,(DEUXIEME,VIDE,K4,NUL,LABAS)

        CALL  BGSEQ,(DEUXIEME)
        CALL  STPOS,(DEUXIEME,XC,YC)
        CALL  PLINE,(DEUXIEME,X,Y,NUL,NUL,NUL,K4,NUL,K1,XINCR)
        CALL  ENSEQ,(DEUXIEME,ZIGZAG)

        CALL  CRATL,(TERMINAL,NIVEAU1)
        CALL  ENATN,(NIVEAU1,K1)
        CALL  MLITS,(TERMINAL,K4,K1)
        CALL  EXEC,(PREMIER)
        CALL  RQATN,(NIVEAU1,REPONSE,ATTENTE,NUL,K1)
        CALL  CRATL,(TERMINAL,NIVEAU2)
        CALL  TMGDS,(PREMIER)
        CALL  EXEC,(DEUXIEME)

ITER1   CALL  ENATN,(NIVEAU2,K1,FIN,CRAYON)
        CALL  SLPAT,(DEUXIEME,OUI)

ITER2   CALL  ICURS,(DEUXIEME,NUL,LABAS,K1)
        CALL  RQATN,(NIVEAU2,REPONSE,ATTENTE,NUL,K1,FIN,CRAYON)
        CLC   REPONSE,K1
        BE    RALLUME
        CLC   REPONSE,FIN
        BE    TESTFIN
        CLC   REPONSE,CRAYON
        BNE   ITER1
        CALL  OMIT,(DEUXIEME,NUL,ZIGZAG)
        CALL  SLPAT,(DEUXIEME,NON)
        CALL  DSATN,(NIVEAU2,CRAYON)
        B     ITER2

RALLUME CALL  INCL,(DEUXIEME,NUL,ZIGZAG)
        B     ITER1

TESTFIN CALL  GSPRD,(DEUXIEME,CARAC,K1,TEXTE,NUL,NUL,LABAS)
        CLI   CARAC,C'F'
        BNE   ITER2
```

```
         CALL  ENATL,(NIVEAU1)
         CALL  TMDEV,(TERMINAL)
         CALL  TMGSP,(GEXEMPLE)
         L     13,SAUVE+4
         RETURN (14,12)


         DC    F'-5'
EMPLE    DS    F
TE       DC    F'10'
MINAL    DS    F
96       DC    F'4096'
MIER     DS    F
XIEME    DS    F
RE       DC    F'3'
         DC    F'4'
AS       DS    F
         DC    F'1'
ZAG      DS    F
EAU1     DS    F
ONSE     DS    F
ENTE     DC    F'2'
EAU2     DS    F
         DC    F'32'
YCN      DC    F'34'
         DC    F'1'
         DC    F'2'
AC       DS    F
TE       DC    F'1'
VE       DS    18F

ASE      DC    CL74'EXEMPLE DE PROGRAMME GRAPHIQUE ELEMENTAIRE.'
         DC    CL74'CE PROGRAMME VA DESSINER DES ZIGZAG.'
         DC    CL74'APPUYEZ SUR LA TOUCHE 1 POUR COMMENCER.'

         DC    F'222'
         DC    E'0'
         DC    E'2000'

NES      DC    CL74'MONTREZ LE DESSIN AVEC LE CRAYON OPTIQUE,IL'
         DC    CL74'DISPARAITRA.'
         DC    CL74'APPUYEZ SUR LA TOUCHE 1 IL REAPPARAITRA.'
         DC    CL74'TAPEZ F PUIS FIN POUR TERMINER'

         DC    F'252'
         DC    E'0'
         DC    E'4000'
E        DC    C'    '
         DC    E'0'
         DC    E'0'
         DC    E'1000'
CR       DC    E'1000'
         DC    E'2000,1000,3000,500'

         END   DEBUT
```

GLOSSAIRE DES PRINCIPAUX SOUS-PROGRAMMES DE G.S.P.

Nous donnons entre parenthèse la phrase anglaise dont le nom est l'abréviation, et nous indiquons succintement l'usage des sous-programmes dont il n'a pas été question encore au cours du chapitre.

INGSP (initialize the graphic Subroutine Package)
INDEV (initialize a graphic device)
INGDS (initialize a graphic data set)
TMGDS (terminate the use of a graphic data set)
TMDEV (terminate the use of a graphic device)
TMGSP (terminate the use of the graphic Subroutine Package)
SDATM (set data mode)
SCHAM (set character mode)
SGDSL (set graphic data set limits)
SDATL (set data limits)
SSCIS (set scissoring option)
MVPOS (move beam to a position)
STPOS (set beam at an absolute position)
} placent ou déplacement le faisceau lumineux sans produire d'image
PLINE (plot lines)
PPNT (plot points)
PSGMT (plot line segments)
PTEXT (plot text)
STEOS (set an End-order-Sequence order) insère un arrêt-programmé
BGSEQ (begin a sequence of elements)
ENSEQ (end a sequence of elements)
RESET (reset a graphic data set)
EXEC (execute)
INCL (place in inclu de status)
OMIT (place in omit status)

Nota : Les modifications on mises à jour d'un objet se font en rappelant le sous-programme de génération qui l'a initialement crée ; pour que cela se fasse très vite et sans réorganisation de la mémoire d'entretien, seule la partie données est remplacée ; ceci implique deux restrictions :

1°/ On ne peut pas changer de catégorie d'objet (des points sont remplacés par d'autres points, des lignes par d'autres lignes, etc...).

2°/ la quantité de mémoire d'entretien prise par les données ne doit pas dépasser celle initialement utilisée.

ORGDS (order graphic data set) modifie l'ordre dans lequel sont affichées successivement les figures. Ceci a son importance vis à vis des détections au crayon optique : quand deux ou plusieures images se chevauchent sur l'écran celle affichée en premier est privilégiée par rapport aux autres.

ICURS (insert cursor)
RCURS (remove cursor) } gouvernent le curseur

GSPRD (read data) permet de transférer vers la mémoire centrale des morceaux de mémoire d'entretien (sert surtout à l'analyse des caractères entrés au clavier alphanumérique).

CRATL (create an attention level)

ENATL (end an attention level)

ENATN (enable attention sources)

DSATN (disable attention sources)

SLPAT (set light pen attention)

RQATN (request attention information)

MLITS (modify status of the programmed function indicator lights) régit l'état des indicateurs lumineux des touches de fonction en liaison avec les niveaux.

MPATL (modify position of an attention level)

SALRM (sound andible alarm) provoque un signal sonore

LOCPN (locate the position of the light-pen) ce sous-programme permet de localiser un point quelconque de l'écran grâce à un balayage à grande vitesse de tout l'écran avec des caractères.

ITRC (test return code) permet de savoir si un sous-programme s'est bien déroulé ; cette possibilité de contrôler soi-même les erreurs est très intéressante pour un travail conversationnel.

ITBP (test integer beam position) } donne l'abscisse ou l'ordonné du
RTBP (test real beam position) } faisceau lumineux pour une figure dor

ITST (test status) renseigne sur le type des coordonnées ou des caractères d'une figure : son emploi permet d'écrire des sous-programmes généraux applicables en toutes circonstances.

INCONVENIENTS DE G.S.P. UTILISE AU NIVEAU DU LANGAGE D'ASSEMBLAGE

L'écriture de l'appel des sous programmes de G.S.P. est rendue pénible par les faits suivants :

- Il y a beaucoup de paramètres à faire figurer
- L'emplacement des paramètres est impératif (paramètres de positi
- Les différents types de paramètres (coordonnée, nom ou numéro d'objet, nom d'entier spécifiant une option) ne ressortent pas assez.
- La technique d'omission de paramètres est lourde : on doit faire figurer un paramètre spécial à son emplacement (cf NUL de l'exemple).
- Enfin on ne peut mettre directement en clair une valeur comme paramètre : on doit mettre le nom de la mémoire contenant cette valeur et définir cette mémoire ailleurs dans le programme.

Ces considérations nous ont amené à utiliser la souplesse du macro-assembleur du 360 pour construire un ensemble de "macro-instructions' dont la syntaxe soit plus simple et l'emploi plus aisé.

Cette réalisation aura eu pour effet de nous permettre d'une par de pénétrer tous les détails de G.S.P., et d'autre part de réfléchir à la forme du langage de programmation graphique à créer, but principal de notre étude.

CHAPITRE IV

DESCRIPTION DE L'ENSEMBLE DE MACRO-INSTRUCTIONS

A) BUTS A ATTEINDRE

Nous avons voulu, en mettant au point ces quelques trente-cinq définitions de macro-instructions, poursuivre les buts suivants :

- nous affranchir des contraintes du formalisme des appels de sous-programmes de G.S.P.
- fournir un outil de programmation graphique utilisable commodément pour la rédaction du programme de commande.
- Commencer à dégager les principales caractéristiques qui présideront à la définition du langage de programmation graphique.

B) MOYENS UTILISES

Nous nous sommes servis de toutes les possibilités qu'offre le macro-Assembleur du système d'exploitation de l'ordinateur IBM-360 (Réf. IBM 4).

Une macro-instruction se présente sous la forme suivante :
référence facultative nom paramètres de position , paramètres à mot-clé

Exemples :

```
ITER    PLACER   FIGURE2,  CØØR = (500,1000)
        ØPTIØNS  FIGURE2, LETTRES = 'GRANDES'
```

```
GENERER   MESSAGE, '<==' ,  TEXTE = 'DESSIN  DØNT  LE  NØM  SERA
AFFICHER  FIGURE2
METTRE    CURSEUR = MESSAGE, ENDRØIT = 25
LIRE 'CARACTERES', DE = MESSAGE, DANS = ZØNE LEC, JUSQUA = 'CURSEUR'
```

Comme la plus grande liberté est permise dans l'expression des paramètres, nous avons fait les choix suivants :

les mots indiquant des options seront mis entre apostrophes ;

les mots sans apostrophes seront des identificateurs (ils se réfèreront à des mémoires créées par le programmeur ou générées automatiquement dans l'expansion de la macro-instruction) ;

un nombre provoque automatiquement la création d'une mémoire qui le contient.

Il faut remarquer de plus que les paramètres introduits par des mots clés peuvent apparaître dans un ordre quelconque. Un paramètre peut être écrit comme une liste de "sous-paramètres" délimitée par une paire de parenthèses (cf CØØR de l'exemple).

Les paramètres à mot-clé sont privilégiés d'une part pour rendre plus clair l'usage des paramètres et d'autre part pour que leur omission éventuelle n'oblige pas à faire figurer des virgules supplémentaires.

On se souvient que les sous-programmes de G.S.P. utilisent un très grand nombre de paramètres , en particulier, même quand un "objet" à un nom, il faut mentionner aussi le nom de la figure à laquelle celui-ci est rattaché. Nous avons introduit des macro-instructions jouant le rôle de déclaration pour rattacher les objets doués de noms aux figures. Ensuite, grâce au mécanisme des "variables d'assemblage", on retrouvera automatiquement à quelle figure se rapporte un objet. En effet le macro-assembleur dispose d'une classe de variables, douées éventuellement d'une nature de tableau à une dimension, dont le but est de conserver des valeurs numériques, des valeurs booléennes ou des identificateurs, et de pouvoir les

transmettre d'une macro-expansion à une autre.

Par exemple si l'on a les deux macro-définitions :

```
          MACRØ
          DECLARE  & PARAM1, &VAL1, &PARAM2, &N
          GBLC     &GRØUPE1(100),&GRØUPE2(100)
&GRØUPE1(&N) SETC '&PARAM1'
&GRØUPE2(&N) SETC '&PARAM2'
&PARAM1  DC     F '&VAL1'
&PARAM2  DC     F '0'
          MEND
           et
          MACRØ
&ETIQ     UTILISE &P, &AVEC =
          GBLC &GROUPE1(100), &GRØUPE2(100)
&ETIQ     L    R4, &AVEC
          ST   R4, &GRØUPE2(&P)
          L    R4, &GRØUPE1(&P)
          ST   R4, &AVEC
          MEND
```

les trois lignes suivantes provoqueront ce qui est écrit plus loin :

```
          DECLARE HUIT, 8, STØCK, 3
          DECLARE K206, 206, GARAGE, 4
DEBUT     UTILISE 3, AVEC = MEMØIRE
HUIT      DC   F'8'
STØCK     DC   F'0'
K 206     DC   F'206'
GARAGE    DC   F'0'
DEBUT     L    R4,MEMØIRE
          ST   R4, STØCK
          L    R4, HUIT
          ST   R4, MEMØIRE
```

Ainsi le seul pointeur 3 nous a fait retrouver les noms STØCK et HUIT qui ont été rangés par l'assembleur dans les variables d'assemblage &GRØUP 1(3) et &GRØUPE2(3).

Les variables d'assemblage n'ont évidemment d'existence qu'au moment de l'assemblage. Elles nous permettent aussi de faire faire des contrôles statiques par l'assembleur sur les paramètre : les noms de toutes les entités graphiques sont associés à des variables d'assemblage qui gardent des renseignements sur leur nature (point, segement ou texte par exemple), sur leurs caractéristiques (coordonnées entières ou réelles pour une figure : ainsi une coordonnée indiquée par un nombre donnera automatiquement lieu à la création d'une constante entière ou réelle).

Signalons enfin que certains paramètres sont facultatifs. Dans le cas d'une omission il y a une option par défaut choisie comme étant la plus courante ;cela allège l'écriture.

Avant de décrire toutes les macro-instructions, voyons ce que devient l'exemple déjà développé plus haut.

```
        PRINT NOGEN
```

C) EXEMPLE

```
XEMPLE  CSECT
EBLT    SAVE  (14,12)
        BALR  3,0
        USING *,3                    ADRESSABILITE DU PROGRAMME
        LA    4,=A(@#$CONS)          ET DES CONSTANTES.
        USING @#$CONS,4
        ST    13,SAUVE+4


        LA    13,SAUVE
        GRAPHIC NIVEAU=(NIVEAU1,NIVEAU2),ENTIER=REP,                    /
              ECONS=(CRAYON,34,FIN,32),CHAINE=(CARAC,' ')

        FIGURE PREMIER
        FIGURE DEUXIEME
        TEXTE (LIGNES,LABAS),FIGURE=DEUXIEME
        SEQUENCE ZIGZAG
        TEXTE PHRASE,FIGURE=PREMIER


        COMMENCE FIGURE=(PREMIER,DEUXIEME)

        GENERER PHRASE,'<==',TEXTE='EXEMPLE DE PROGRAMME GRAPHIQUE ELE/
              MENTAIRE. CE PROGRAMME VA DESSINER DES ZIGZAG. APPUYEZ S/
              UR LA TOUCHE 1 POUR COMMENCER.',COOR=(0,2000)

        GENERER LIGNES,'<==',TEXTE='MONTREZ LE DESSIN AVEC LE CRAYON O/
              PTIQUE, IL DISPARAITRA. TAPEZ F PUIS FIN POUR TERMINER',/
              COOR=(0,4000)

        OPTIONS DEUXIEME,LETTRES='PETITES'
        GENERER LABAS,'<==',TEXTE='    '

        COMMENCE SEQUENC=ZIGZAG
        REPRISE DEUXIEME,COOR=(0,0)
        GENERER DEUXIEME,'<==',OBJET='LIGNE',COOR=(1000,Y),             /
              XINCR=1000,NB=4
        TERMINER SEQUENC=ZIGZAG

        COMMENCE NIVEAU=NIVEAU1
        TRAITER NIVEAU1,NUMEROS=1
        TOUCHES NUMEROS=1


        AFFICHER PREMIER

        ETAT  NIVEAU1,REPONSE=REP,MODE='ATTENTE',NUMEROS=1

        COMMENCE NIVEAU=NIVEAU2

        TERMINER FIGURE=PREMIER
```

```
         AFFICHER DEUXIEME

ER1      TRAITER NIVEAU2,NUMEROS=(1,32,34)
         CRAYON DEUXIEME,MODE='TRAITE'

ER2      METTRE CURSEUR=LABAS

         ETAT  NIVEAU2,REPONSE=REP,MODE='ATTENTE',NUMEROS=(1,32,34)

         CLC   REP,UN
         BE    RALLUME
         CLC   REP,FIN
         BE    TESTFIN
         CLC   REP,CRAYON
         BNE   ITER1

         ETEINDRE ZIGZAG
         CRAYON DEUXIEME,MODE='ECRIRE'
         B     ITER2

LLUME    ALLUMER ZIGZAG
         B     ITER1

STFIN    LIRE  'CARACTERES',DE=LABAS,DANS=CARAC,NOMBRE=1

         CLI   CARAC,C'F'
         BNE   ITER2


         TERMINER NIVEAU=NIVEAU1
         FINGRAPH

         L     13,SAUVE+4
         RETURN (14,12)


UVE      DS    18F
         DC    F'1'
         DC    F'2000,1000,3000,500'

         END   DEBUT
               =A(@#$CONS)
```

C) REMARQUES

Les macro-instructions dont nous avons mis au point les définitions ne concernent que les aspects graphiques de la programmation ; elles serviront en effet à l'intérieur de programmes en langage d'assemblage (le programme de commande par exemple).

Ce travail nous a occupé un certain temps car il a fallu vérifier le bon fonctionnement de l'assembleur dans tous les cas possibles. Cela était d'autant plus important que l'assembleur dont nous nous sommes servi n'a pas de description syntaxique rigoureuse.

Cependant à la suite de cette étude et de cette mise au point, nous avons pu tirer des enseignements en vue de la définition du langage de programmation graphique:

- un certain nombre de paramètre peuvent être supprimés grâce à quelques déclarations liminaires ;

- d'autres paramètres, indiquant des options dans le fonctionnement d'une instruction graphique, peuvent être spécifiés sous forme de mots-clefs dont la signification est "parlante" ;

- la technique d'omission de paramètres, avec valeurs par défaut, vient encore alléger l'écriture ;

- le rôle auxiliaire joué par les "variables d'assemblage" de l'Assembleur pourra être le fait du compilateur avec autant et même plus de souplesse.

D) DESCRIPTION FORMELLE DES MACRO-INSTRUCTIONS

Nous utilisons le formalisme suivant, qui a l'avantage d'être le plus lisible possible :

- les éléments entre crochets sont facultatifs
- lex accolades indiquent un choix à faire
- les options soulignées sont celles qui dont prises par défaut
- les points de suspension indiquent une répétition.

Nous n'avons pas fait figurer d'explications sémantiques parce qu'elles feraient double emploi avec celles qui accompagnent la description du langage de programmation graphique du chapitre 5.

ABREVIATIONS

id	identificateur
nb	nombre
n	numéro de registre (0 à 15)
nomf	nom de figure.

MACRO-INSTRUCTIONS DE DECLARATION

```
GRAPHIQ   [NIVEAU=(id,...)][,ENTIER=(id,...)][,REEL=(id,...)]
          [,CHAINE=(id,chaîne,...,...)]
          [,ECØNS=(id,nb,...,...)][,RCØNS=(id,nb,...,...)]
          [,ETAB=(id,nb,...,...)][,RTAB=(id,nb,...,...)]

FIGURE nomf [,ENTREED=( {'REELABSØLU'  } [,{'REELABSØLU'  }])]
                       {'REELRELATIF'  }      {'REELRELATIF'  }
                       {'ENTIERABSØLU' }      {'ENTIERABSØLU' }
                       {'ENTIERRELATIF'}      {'ENTIERRELATIF'}

           [,LETTRES=( {'PETITES'} [,'PRØTEGEES'])]
                      {'GRANDES'}

           [,ZØNE =({nb},{nb},{nb},{nb})[,ECRAN=({nb},{nb},{nb},{nb})]]
                    {id}{id}{id}{id}      {id}{id}{id}{id}

           [,DØNNEES=({nb},{nb},{nb},{nb})]
                      {id}{id}{id}{id}

           [,TAILLE={nb}]                [,CØUPAGE= {'EC'}] [,SØRTIED= {'ØPTIMISE'   }]
                    {id}                           {'EA'}             {'ABSØLU'     }
           [,RENØME=(id,...)]              {'FC'}             {'INCREMENTIEL'}
                                           {'FA'}
                                           {'L' }

POINT  id, ... [,FIGURE = nomf]
LIGNE  id, ... [,FIGURE = nomf]

          SEGMENT  id, ... [,FIGURE=nomf]
          TEXTE    id, ... [,FIGURE=nomf]
          PØSITION id, ... [,FIGURE=nomf]
          ARRET    id, ... [,FIGURE=nomf]
          SEQUENCE id, ... [,FIGURE=nomf]
```

MACRO-INSTRUCTIONS D'INITIALISATION

CØMMENCE FIGURE = (nomf,...)

CØMMENCE SEQUENC= nomséquence

CØMMENCE NIVEAU = nomniveau [,PILE = {'LIBERER' / 'GARDER'}]

TERMINER FIGURE = (nomf,...)

TERMINER SEQUENC= nomséquence

TERMINER NIVEAU = nomniveau [,JUSQUA = {'LUI' / 'ENDESSØUS'}]

ØPTIONS nomf [,ENTREED = ({'REELABSØLU' / 'REELRELATIF' / 'ENTIERABSØLU' / 'ENTIERRELATIF'} [, {'REELABSØLU' / 'REELRELATIF' / 'ENTIERABSØLU' / 'ENTIERRELATIF'}]

[,LETTRES = {'PETITES' / 'GRANDES'} [,PRØTEGEES])]

[,ZONE=({nb / id / (n)},{nb / id / (n)},{nb / id / (n)},{nb / id / (n)}) [,ECRAN =({nb / id / (n)},{nb / id / (n)},{nb / id / (n)},{nb / id / (n)})]]

[,DØNNEES = ({nb / id / (n)},{nb / id / (n)},{nb / id / (n)},{nb / id / (n)})]

[,CØUPAGE= {'EC' / 'EA' / 'FC' / 'FA' / 'L'}]

[,SØRTIED= {'OPTIMISE' / 'ABSØLU' / 'INCREMENTIEL'}]

CRAYØN nomf [,MODE={'TRAITE' / 'IGNORE'}]

MACRO-INSTRUCTIONS DE TRAITEMENT DES INTERRUPTIONS

TRAITER nomniveau, NUMERØS = (nb,...)

IGNØRER nomniveau, NUMERØS = (nb,...)

MØDIFIER nomniveau, DE = {nb | id | (n)} [,VERS = {'LE HAUT' | 'LE BAS'}] [,DEPUIS = nomniveau]

ETAT nomniveau, REPØNSE = id, NUMERØS = (nb,...), MØDE = {'ATTENTE' | 'RETOUR'}

[,TABLEAU = id]

TØUCHES nomniveau, { ACTION = {'DEFAUT' | 'ETEINT' | 'NØRMAL'} | NUMERØS = (nb,...) }

TOUCHES { NUMERØS = (nb,...) | ACTIØN = {'ETEINT' | 'NØRMAL'} }

MACRO-INSTRUCTIONS DE GENERATION ET DE MISE A JOUR

1°) Génération

GENERER { mon texte,'<{=|#}{=|#}' [,OBJET='TEXTE'] | nomf ,'<{=|#}{=|#}', OBJET='TEXTE' } [,NUMERØ={nb | id | (n)}]

,TEXTE = {chaîne | id} [,CØØR = ({nb | id | (n)}, {nb | id | (n)})] [,ETAT = {'ALLUME' | 'ETEINT'}]

GENERER { nom position,'<{=}{=}'[,ØBJET='PØSITION'] / nomf ,'<{=}{=}',ØBJET='PØSITIØN' } [,NUMERØ={nb / id / (n)}]

,CØØR=({nb / id / (n)},{nb / id / (n)})[,ETAT={'ALLUME' / 'ETEINT'}]

GENERER { {nom ligne / non point},'<{=}{=} '[,ØBJET={'LIGNE' / 'PØINT'}] / nomf ,'<{=}{=} ' ,ØBJET={'LIGNE' / 'PØINT'} } [,NUMERØ={nb / id / (n)}]

,CØØR =({nb / id / (n)},{nb / id / (n)}) [,ETAT = {'ALLUME' / 'ETEINT'}]

[,NB={nb / id / (n)}],XPAS={nb / id / (n)}][,YPAS={nb / id / (n)}][,XINCR={nb / id / (n)}][,YINCR={nb / id / (n)}]]

GENERER { non segment,'<{=}{=}' [,ØBJET='SEGMENT'] / nomf ,'<{=}{=} ',ØBJET='SEGMENT' } [,NUMERØ={nb / id / (n)}]

,CØØR =({nb / id / (n)},{nb / id / (n)},{nb / id / (n)},{nb / id / (n)}) [,ETAT = {'ALLUMÉ' / 'ETEINT'}]

[,NB={nb / id / (n)}[,XPAS =({nb / id / (n)},{nb / id / (n)})][,YPAS =({nb / id / (n)},{nb / id / (n)})]

[,XINCR=({nb / id / (n)},{nb / id / (n)})][,YINCR=({nb / id / (n)},{nb / id / (n)})]]

2°) Mise à jour

GENERER { nom texte,'<{=/#}{=/#}>' [,ØBJET='TEXTE'] [,NUMERØ={nb/id/(n)}] }
{ nomf ,'<{=/#}{=/#}>',ØBJET='TEXTE',NUMERØ={nb/id/(n)} }

,TEXTE={chaîne/id} [,CØØR=({nb/id/(n)},{nb/id/(n)})]

GENERER { nom position,'<{=/#}{=/#}>' [,ØBJET='PØSITION'] [,NUMERØ={nb/id/(n)}] }
{ nomf ,'<{=/#}{=/#}>',ØBJET="PØSITIØN',NUMERØ ={nb/id/(n)} }

,CØØR=({nb/id/(n)},{nb/id/(n)})

GENERER { {nom ligne/non point},'<{=/#}{=/#}>' [,ØBJET={'LIGNE'/'PØINT'}] [,NUMERØ={nb/id/(n)}] }
{ nomf ,'<{=/#}{=/#}>',ØBJET={'LIGNE'/'PØINT'},NUMERØ={nb/id/(n)} }

,CØØR=({nb/id/(n)},{nb/id/(n)})

[,NB={nb/id/(n)}] [,XPAS={nb/id/(n)}] [,YPAS={nb/id/(n)}] [,XINCR={nb/id/(n)}] [,YINCR={nb/id/(n)}]]

GENERER { nom segment,'<{=/#}{=/#}>'[,ØBJET='SEGMENT'][,NUMERØ={nb/id/(n)}] | nomf ,'<{=/#}{=/#}>',ØBJET='SEGMENT',NUMERØ={nb/id/(n)} }

,CØØR=({nb/id/(n)},{nb/id/(n)},{nb/id/(n)},{nb/id/(n)})

[,NB={nb/id/(n)}[,XPAS=({nb/id/(n)},{nb/id/(n)})][,YPAS=({nb/id/(n)},{nb/id/(n)})]
[,XINCR={nb/id/(n)},{nb/id/(n)})][,YINCR=({nb/id/(n)},{nb/id/(n)})]]

MACRO-INSTRUCTIONS DE TRAITEMENT DES FIGURES

PLACER nomf, CØØR = ({nb/id/(n)},{nb/id/(n)})

REPRISE nomf, CØØR = ({nb/id/(n)},{nb/id/(n)})

ANNULER { nomf[,NUMERØ={nb/id/(n)}] | nom objet | nom séquence | nom arrêt }

METTRE { ARRET = nom arrêt | ARRET=nomf[,NUMERØ={nb/id/(n)}] } [,ETAT= {'EFFECTIF'/'IGNØRE'}]

```
              nomf [,NUMERØ={nb | id | (n)}]
ALLUMER     { nom objet
              nom séquence
              nom arrêt }

              nom [,NUMERØ={nb | id | (n)}]
ETEINDRE    { nom objet
              nom séquence
              nom arrêt }

AFFICHER   nomf,...

ØRDONNER   nomf,...
```

MACRO-INSTRUCTIONS DIVERSES

```
METTRE  { CURSEUR=nomf,NUMERØ={nb | id | (n)}
          CURSEUR= {nom texte | nom séquence} }  [,ENDROIT={nb | id | (n)}]

ENLEVER CURSEUR=nomf

LIRE [{'CARACTERES' | 'DØNNEES'} ,]DANS=id [,JUSQUA={'CURSEUR' | 'FIN'}],NØMBRE={nb | id}

     [,REPONSE=id][,FIGURE=nomf] [,DE={nom objet | nom arrêt | nb | id | (n)}] [,A={nom objet | nom arrêt | nb | id | (n)}]
```

LICRAYØN nomf, CØØR=(id,id)

SØNNETTE

FINGRAPH

ERREUR INTERØ = $\left\{\begin{matrix}1\\2\\3\\4\\5\\9\end{matrix}\right\}$

LIEUDEXY nomf $\left[\text{,XØUY=}\left\{\begin{matrix}\text{'X'}\\\text{'Y'}\end{matrix}\right\}\right]\left[\text{,TYPE=}\left\{\begin{matrix}\text{'REEL'}\\\text{'ENTIER'}\end{matrix}\right\}\right]\left[\text{,LIEU=}\left\{\begin{matrix}\text{'SUPPOSE'}\\\underline{\text{'EFFECTIF'}}\end{matrix}\right\}\right]$

TYPE nomf $\left[\begin{matrix}\\\text{,DØNNEES =}\left\{\begin{matrix}\underline{\text{'ENTREEX'}}\\\text{'ENTREEY'}\\\text{'SØRTIE'}\\\text{'LETTRES'}\end{matrix}\right\}\\\text{-}\end{matrix}\right]$

NOTA : A l'exécution ces trois dernières macro-instructions procurent des renseignements qui sont rangés dans le registre 0.

CHAPITRE 5

LE LANGAGE DE PROGRAMMATION GRAPHIQUE

A) GENERALITES

Le programme doit permettre de contruire des images et de les manipuler, c'est à dire de les faire apparaître, les déplacer, les modifier, les éteindre, les rallumer, les effacer définitivement, etc. Pour cela, on dispose de différents niveaux d'association des composants d'images, et de possibilités corrélatives d'identification par nom ou par numéro.

Voici quelques définitions préliminaires :

a) Pour les dessins sur l'écran.

Image : ensemble des points éclairés que se trouvent à un instant donné sur l'écran.

Figure: partie d'image qui forme un tout relativement aux faits suivants : la figure a un nom qui lui est propre ; il lui est attribué, explicitement ou implicitement, un certain nombre de caractéristiques : portion de l'écran affectée, partie de mémoire d'entretien réservée, système de coordonnées (relatives, absolues, avec ou sans échelle), dimension des caractères de textes, manière de traiter les parties de figure qui sortent des limites imposées ; la détection d'un point de la figure par le crayon optique est reconnue ou ignorée pour toute la figure.

Objet : partie de figure construite à l'aide d'une seule instruction graphique, identifiable par un nom unique, ou par un numéro et le nom de la figure dont il fait partie ; exemples d'objets : un point,

une suite de points, un vecteur, une suite de vecteurs consécutifs, un segment, une suite de segments, un texte, une position de départ du faisceau lumineux.

Séquence: union d'objets, appartenant à la même figure, construits consécutivement et identifiables comme un tout par un nom global.

Elément: ce qui est manipulé de façon élémentaire au point de vue technologique (point, vecteur, caractère) ; le programmeur ne peut y accéder que par l'intermédiaire d'un objet.

b) Pour la gestion des interruptions.

Niveau : (d'interruptions) : ensemble d'interruptions repérable par un nom ; ces interruptions sont prises dans l'ensemble de celles que peuvent provoquer les 31 touches de fonction, le crayon optique, la touche "fin" du clavier alphanumérique et l'arrêt programmé. Un niveau est actif ou inactif ; s'il est actif, les informations afférentes aux interruptions qui se produisent parmi celles qu'il contient sont rangées dans l'ordre d'arrivée, et récupérables par le programme de façon sélective. Il peut exister soit un seul niveau, soit plusieurs ; dans ce dernier cas, les niveaux sont hiérarchisés, un seul, le plus haut dans la pile, étant actif à un instant donné.

Le langage de programmation graphique se compose de déclarations et d'instructions.

Les déclarations comprennent les déclarations de variables, de tableaux et de chaînes, avec éventuellement des valeurs initiales, les déclarations de figures qui comprennent l'énoncé des caractéristiques de la figure, modifiables au cours du programme par l'instruction options,et les déclarations d'objets et de séquences.

Les instructions comprennent les instructions spécialement adaptées au maniement du terminal graphique, et que nous appellerons les instructions graphiques, et les autres. Ces dernières sont l'instruction d'affectation simple, l'instruction allera, l'instruction si et l'instruction pour.

Les instructions graphiques ont une syntaxe particulière décrite complètement dans le paragraphe B). Cette syntaxe, de forme fixe, ponctuée de connecteurs graphiques et usant de mots-clés est en relation avec les paramètres nécessaires aux sous-programmes de G.S.P. qui accomplissent les actions désirées, dont voici les principales :

- Initialiser ou terminer une figure ou un niveau.
- Construire des objets.
- Délimiter une séquence.
- Afficher une figure.
- Allumer, éteindre ou rallumer une figure, une séquence un objet.
- Modifier (mettre à jour) un objet.

- Modifier les caractéristiques d'une figure.

- Autoriser ou interdire des interruptions pour un niveau, ou la détection par le crayon optique sur une figure donnée.

- Rendre actif ou inactif un niveau.

- Modifier la hiérarchie des niveaux.

- Allumer ou éteindre les indicateurs lumineux de certaines touches du clavier de fonction.

- Interroger un niveau sur les interruptions qui ont pu se produire.

- Manipuler le curseur et lire des caractères tapés sur le clavier alphanumérique.

- Localiser un point sur une figure.

- Localiser un point quelconque de l'écran.

- Actionner la "sonnette"(signal auditif).

- Lire les coordonnées actuelles du faisceau lumineux.

- Vérifier l'état actuel de certaines caractéristiques d'une figure.

Dans les instructions interviennent des expressions arithmétiques simples, de syntaxe semblable à celles d'Algol, des fonctions standard (sinus, cosinus, racine carrée, etc...) et des expressions de texte mettant en jeu la sélection de sous-chaînes et des expressions de texte mettant en jeu la sélection de sous-chaînes et la concaténation. Il n'est pas prévu pour le moment de véritables booléennes, mais seulement des relations entre expressions arithmétiques ou entre expressions de texte.

B) DESCRIPTION SYNTAXIQUE ET SEMANTIQUE DU LANGAGE DE PROGRAMMATION GRAPHIQUE.

SYMBOLISME UTILISE POUR LA DESCRIPTION:

Les unités syntaxiques sont soit des constantes, c'est à dire des symboles de base, soit des variables décrites par ailleurs ; leur désignation est donnée par des mots groupés par des tirets (exemple : entier-sans-signe).

Les parenthèses servent à mettre en facteur un groupe d'unités syntaxiques pour les considérer comme une seule unité dans l'application d'une ou plusieurs des opérations suivantes :

- Choix possible entre deux unités ; les unités entre lesquelles on peut choisir sont séparées par le signe |.
- Omission possible d'une unité, indiquée par le signe ? suivant immédiatement l'unité qui peut être omise.
- Répétition possible d'une unité, indiquée par le signe & suivant immédiatement l'unité qui peut être répétée.

Le symbole ::= sépare le nom d'une unité de sa description. Quand une unité du langage utilise un des symboles du langage de description, ce symbole est souligné pour lever l'ambiguité.

SYMBOLES DE BASE

lettre ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

Chiffre ::= 0|1|2|3|4|5|6|7|8|9

opérateur-arithmétique ::= +|-|*|* *|/

opérateur-de-relation ::= <|<=|=|>=|>|¬=

opérateur-de-concaténation ::= ||

crochet ::= (|)|(*|*)|"

séparateur ::= ,|.|!|:|:=|;|pas|à

opérateur-de-sélection-de-texte ::=|sélection|depuis|longueur

opérateur-séquentiel ::= allerà|si|sinon|finsi|pour|finpour

déclarateur ::= réel|entier|tableau|chaîne|caractères

symboles-graphiques ::= à|abscisse|absolu|afficher|allumé|allumer|annuler|arrêt|attente|caractères|commencer|coupage|crayon|curseur|de|défaut|depuis|données|ea|ec|écran|effectif|en|enbas|enhaut|enlever|entier|entierabsolu|entierrelatif|entrée|état|éteindre|éteint|erreur|fa|fc|figure|fin|fingraphique|fois|gardé|grandes|incrémentiel|ignoré|ignorer|jusqua|l|lettres|libéré|ligne|lire|lirecrayon|mettre|modifier|niveau|numéro|optimisé|options|ordonnée|ordonner|petites|placer|point|position|protégées|réel|réelabsolu|réelrelatif|renomme|reprise|retour|résultat|segment|séquence|sonnette|supposé|terminer|texte|touches|traité|traiter|type|x|y|zone|<|=|>|#

DECLARATIONS DE VARIABLES, DE TABLEAUX ET DE CHAINES.

identificateur ::= lettre (lettre|chiffre)?&

entier-sans-signe ::= chiffre&

signe ::= +|-

entier ::= signe ? entier-sans-signe

facteur-de-cadrage ::= ! entier

partie-décimale ::= . entier-sans-signe

nombre-décimal ::= entier-sans-signe|(entier-sans-signe ? partie-décima

nombre-sans-signe ::= nombre décimal|(nombre-décimal ? facteur-de-cadrag

nombre ::= signe ? nombre-sans-signe

chaîne ::= "suite-de-caractères-ce-comportant-des-guillements-que-par-paires-contigües"

déclaration-de-variables-réelles ::= réel identificateur (:=nombre) ? (,identificateur (:=nombre) ?) ? § ;

déclaration-de-variables-entières ::= entier identificateur (:=entier) ? (,identificateur (:=entier) ?) ? & ;

élément-de-déclaration-de-tableaux ::= (réel|entier) identificateur (,identificateur) ? & (* entier-sans-signe *)

déclaration-de-tableaux ::= tableau élément-de-déclaration-de-tableaux (,élément-de-déclaration-de-tableaux) ? & ;

élément-de-déclaration-de-chaînes ::= entier-sans-signe caractères identificateur (:=chaîne) ? (,identificateur (=chaîne) ?) ? &

déclaration-de-chaînes ::= chaîne élément-de-déclaration-de-chaînes (élément-de-déclaration-de-chaînes) ? & ;

Exemples de déclarations :

```
entier    X , I := 1, Y ;
réel      AA := 12.5, BB, CC := -3, PI := 314.15926535 ! - 2 ;
tableau réel X1, Y1 (*24*), entier T1, T2, T3 (*10*), entier U (*100*) ;
```

chaîne 10 caractères A := "BLABLA", B, C := "0123456789",
26 caractères D,E, ALPHABET := "ABCDEFGHIJKLMNOPQRSTUVWXYZ" ;

EXPRESSIONS

expression-arithmétique ::= cf. l'expression arithmétique simple d'Algol 60.

identificateur-de-chaîne ::= identificateur

élément-de-texte ::= chaîne|identificateur-de-chaîne

sélection-de-texte ::= sélection élément-de-texte depuis expression-arithmétique (longueur expression-arithmétique) ?

texte ::= élément-de-texte|sélection-de-texte

expression-de-texte ::= texte (||expression-de-texte) ?

expression-booléenne ::= (expression-arithmétique opérateur-de-relation expression-arithmétique)|(expression-de-texte (=|¬=) expression-de-texte)

Exemples d'expressions de texte :

ALPHABET||C a pour valeur "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
sélection C depuis 2 longueur 9 a pour valeur "123456789"
sélection ALPHABET depuis 24||"T" a pour valeur "XYZT"

INSTRUCTIONS NON GRAPHIQUES

étiquette ::= identificateur

instruction-d'affectation-arithmétique ::= variable := expression-arithmétique ;

instruction-d'affectation-de-texte ::= (identificateur-de-chaîne|sélection-de-texte) := expression-de-texte ;

instruction-d'affectation ::= instruction-d'affectation-arithmétique | instruction-d'affectation-de-texte

instruction-aller-à ::= allera étiquette ;

instruction-pour ::= pour variable := expression-arithmétique pas expression-arithmétique à expression-arithmétique ; instruction & finpour ;

instruction-si ::= si expression-booléenne ; instruction & (sinon ; instruction &) ? finsi ;

instruction-nom-étiquetée ::= instruction-d'affectation | instruction-aller-à | instruction-pour | instruction-si | instruction-graphique

instruction ::= (étiquette :) ? instruction-non-étiquetée

NOTA : Le délimiteur finpour (ou finsi) se rapporte toujours au dernier pour (ou si) qui le précède ; le délimiteur sinon se rapporte au dernier si qui le précède.

DECLARATIONS GRAPHIQUES

nom-de-niveau ::= identificateur
nom-de-figure ::= identificateur
nom-de-ligne ::= identificateur
nom-de-point ::= identificateur
non-de-segment ::= identificateur
nom-de-texte ::= identificateur
nom-de-position ::= identificateur
nom-de-séquence ::= identificateur
nom-d'arrêt ::= identificateur

nom-d'objet ::= nom-de-ligne | nom-de-point | nom-de-segment | nom-de-texte | nom-de-position

désignation-par-numéro ::= nom-de-figure numéro expression-arithmétique

coordonnée ::= (expression-arithmétique pas expression-arithmétique) | (identificateur-de-tableau (*expression-arithmétique pas expression-arithmétique*))

borne ::= variable|nombre

déclaration-de-niveau ::= niveau nom-de-niveau(,nom-de-niveau) ? & ;

mode-absolu ::= =

mode-relatif ::= #

mode ::= mode-absolu|mode-relatif

mode-des-abscisses ::= mode

mode-des-ordonnées ::= mode

symbole-de-génération-graphique ::=<mode-des-abscisses mode-des-ordonnées

symbole-de-mise-à-jour ::= symbole-de-génération-graphique>

déclaration-de-figure ::= figure nom-de-figure
(entrée(réel absolu §|réel relatif|entier absolu|entier relatif)
(,(réelabsolu|réelrelatif|entierabsolu|entierrelatif)) ?) ?
(coupage(ec §|ea|fc|fa|1)) ?
(sortie(optimisé §|absolu|incrémentiel))?
(lettres(petites §|grandes)protégées ?) ?
(zone borne, borne borne (écran borne, borne, borne, borne) ?) ?
(données borne, borne, borne, borne) ?
(taille entier-sans-signe) ?
(renomme nom-de-figure (,nom-de-figure) ? &) ?
;

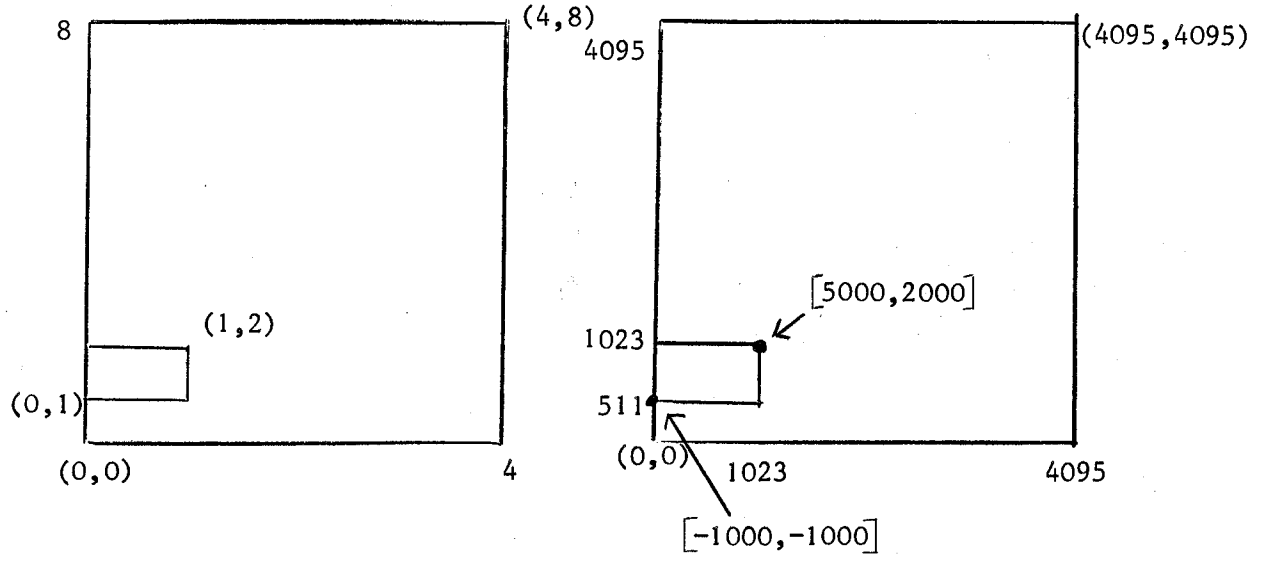
SEMANTIQUE :

Toutes les caractéristiques, sauf taille et renomme, peuvent être modifiées par l'instruction options ; le symbole § désigne l'option supposée implicitement si on ne l'indique pas.

- entrée : la première spécification est relative aux abscisses, la deuxième aux ordonnées si l'on désire un mode différent des abscisses. Dans le cas des coordonnées relatives, les coordonnées des objets sont calculées comme des variations par rapport à la position actuelle du

faisceau lumineux ; celle-ci doit être initialisée au moyen de l'instruction reprise, qui travaille toujours en coordonnées absolues. L'instruction placer peut être utilisée quand on connaît la position du faisceau lumineux et qu'on veut la communiquer au programme comme origine des coordonnées relatives.

- zone et écran : il s'agit de définir la zone de l'écran réservée à la figure ; si tout est omis, il s'agit de l'écran entier ; si écran est omis, les limites de l'écran ont pour coordonnées 0,0,4095 et 4095. Par exemple les spécifications suivantes sont équivalentes :
  zone 0,1,1,2 écran 0,0,4,8
  et zone 0,511, 1023, 1023



- données : il s'agit ici de permettre la définition d'une échelle ; on pourrait ajouter aux spécifications précédentes la suivante :
  données - 1000, - 1000, 5000, 2000.

- Coupage : sert à indiquer de quelle façon on doit traiter les parties de la figure qui sortent soit de la zone de l'écran affectée, soit de l'écran lui-même ; e signifie écran, f signifie figure ; c signifie que l'on coupe les parties de la figure qui tombent en dehors de la zone, a signifie que l'on arrête la génération de la figure à partir du moment où une partie sort d'une zone ; 1 (pour libre) signifie que les parties tombant, en tout ou non, en dehors de l'écran, ne sont pas générées.

- sortie : ceci indique de quelle façon seront codés les ordres destinés au terminal et produits par les sous-programmes de génération.

- protégées : des caractères protégés ne peuvent pas être modifiés au moyen du clavier alphanumérique.

- renomme : on peut associer à une figure jusqu'à 49 figures dites équivalentes, qui partagent la même zone de mémoire du terminal ; ces figures sont mutuellement incompatibles, et l'affichage de l'une d'elles implique l'effacement automatique de celle des figures équivalentes qui est actuellement affichée (s'il y a lieu).

déclaration-d'objets-de-séquence-ou-d'arrêts ::= (ligne|point segment|texte|position|séquence|arrêt) identificateur (,identificateur)? & (figure nom-de-figure) ? ;

NOTA : si l'on omet le nom de figure, les objets, séquences ou arrêts appartiennent à la dernière figure citée dans une déclaration d'objets, de séquences ou d'arrêts.

INSTRUCTIONS GRAPHIQUES

a) Initialisations

instruction-commencer-ou-terminer ::= (commencer|terminer)
(figure nom-de-figure(,nom-de-figure) ? &)|
(séquence nom-de-séquence)|
(niveau nom-de-niveau(libéré§|gardé) ?)
;

Sémantique :

La séquence est constituée des objets faisant partie de la même figure que cette séquence et dont les générations sont faites entre l'instruction commencer et l'instruction terminer qui portent sur cette séquence ; deux séquences ne peuvent être imbriquées.

L'instruction commencer niveau établit un niveau comme niveau actif jusqu'à ce que l'on en commence un nouveau, ou que l'on termine celui-là ; libéré signifie que l'on doit fournir l'information concernant une interruption dès qu'elle est demandée par la fonction état, et la retirer de la file d'attente ; gardé signifie que l'on ne doit fournir cette information que si le niveau est actif, sinon la garder dans la file d'attente.

Dans l'instruction terminer niveau, libéré signifie que l'on veut terminer le niveau indiqué et tous ceux qui lui sont inférieurs dans la hiérarchie ; gardé signifie que l'on doit terminer les niveaux inférieurs et rendre actif le niveau indiqué.

instruction-crayon ::= crayon nom-de-figure (traité|ignoré) ;

instruction-options ::= options nom-de-figure
(entrée|réelabsolu|réelrelatif|entierabsolu|entierrelatif).
(,(réelabsolu|réelrelatif|entierabsolu|entierrelatif) ?) ?
(coupage(ec|ea|fc|fa|1)) ?
(sortie(optimisé|absolu|incrémentiel)) ?
(lettres(petites|grandes)protégées ?) ?
(zone borne, borne, borne, borne (écran borne, borne, borne, borne) ?) ?
(données borne, borne, borne, borne) ?
;

instruction-traiter-ou-ignorer ::= (traiter|ignorer) nom-de-niveau (* expression-arithmétique (,expression-arithmétique) ? & *) ;

Sémantique :

Les instructions permettent d'autoriser - ou d'interdire - pour le niveau indiqué, les interruptions spécifiées entre crochets, suivant la codification suivante : les valeurs 1 à 31 désignent les touches de fonctions de numéro correspondant, 32 la touche "Fin" du clavier alphanumérique, 33 touche "Annuler", 34 le crayon optique et 35 l'arrêt programmé ; la touche 0 est réservée à l'arrêt du programme par intervention extérieure de l'utilisateur ; on peut raccourcir la liste en désignant un éventail de valeurs au moyen d'une valeur négative : 1,-5 signifie 1,2,3,4,5 ; les instructions sont valides que le niveau concerné soit actif ou non. Si une certain interruption est autorisée et se produit sans être exploitée, puis qu'on l'interdit, une nouvelle interruption de ce type sera ignorée, mais l'information concernant la précédente restera dans la file d'attente et pourra être exploitée.

instruction-touches ::= touches nom-de-niveau ? (défaut|éteint normal|((*expression-arithmétique (,expression-arithmétique) ? & *)) ;

Sémantique :

Cette instruction gouverne l'état des indicateurs lumineux associés aux trente et une touches numérotées du clavier de fonctions. Il existe toujours un état global, non lié au niveaux, dont la disposition initiale est le non-éclairage des touches ; on altère cet état global quand on utilise l'instruction touches sans indiquer de nom de niveau : normal allume les touches autorisées pour chaque niveau actif, éteint supprime tout éclairage, une liste de valeurs entre crochets indique quelles touches on veut allumer.

On peut associer un état précis à un niveau en indiquant le nom de celui-ci dans l'instruction touches, cet état étant valable aux moments où le niveau est actif ; l'option défaut associe à un niveau l'état global, initial ou altéré.

instruction-modifier ::= modifier nom-de-niveau de expression-arithmétique (enhaut §|en bas) ?

Sémantique :

Cette instruction sert à déplacer un niveau dans la hiérarchie, du nombre de places indiqué par l'expression arithmétique, vers le haut ou vers le bas, à partir de sa position actuelle, ou à partir du niveau de référence spécifié après depuis.

fonction-état ::= état nom-de-niveau (* expression-arithmétique (,expression-arithmétique) ? & *) (retour attente) (réponse identificateur-de-tableau) ?

Sémantique :

La fonction état est une variable graphique qui peut prendre l'une des valeurs suivantes :

0 - aucune interruption n'a eu lieu
1 à 31 - interruption par une des touches de fonctions
32 - touche "fin" du clavier alphanumérique.
33 - touche "annuler" du clavier alphanumérique
34 - interruption par le crayon optique
35 - arrêt programmé.

La liste d'expressions arithmétiques indique quelles causes d'interruption on veut examiner parmi celles qui sont autorisées pour le niveau indiqué ; attente signifie que le programme doit s'arrêter à cet endroit jusqu'à ce que l'une des interruptions spécifiées se soit produite ; la variable état n'est alors jamais nulle, et le niveau indiqué doit être actif.

Dans le cas 34 et 35, si l'on a mis après réponse le nom d'un tableau de dix mots, on y trouvera les renseignements suivants :

mot 1 nom de la figure détectée ;

mot 2 nom de l'objet ou de l'arrêt programmé, s'il y en a un, sinon 0 ;

mot 3
- si le crayon optique a détecté un caractère qui fait partie d'un texte ayant un nom, numéro d'apparition de ce caractère dans le texte ;
- si le crayon optique a détecté un élément d'un objet ayant un nom, numéro d'ordre de l'octet de programme de la mémoire d'entretien qui a produit l'élément détecté, à partir du début de l'objet ;
- si l'objet détecté n'a pas de nom, numéro d'ordre de l'octet générateur à partir du début de la figure ;

mot 4 si l'objet avait un numéro, celui-ci, sinon 0 ;

mot 5 si l'objet faisait partie d'une séquence, nom de celle-ci, sinon 0 ;

mot 6 numéro d'ordre de l'octet générateur à partir du début de la séquence s'il y en a une, sinon 0 ;

mot 7 0 ;

mots 8 et 9 coordonnées absolues du point détecté, dans le système de coordonnées de la figure concernée ;

mot 10 code EBCDIC, cadré à gauche, du caractère détecté s'il y a lieu, sinon 0.

C) GENERATION ET MISE A JOUR

E.A. ::= expression-arithmétique

génération-de-points-lignes-ou-positions ::= (nom-de-point | nom-de-ligne | nom-de-position | ((point | ligne | position) nom-de-figure)) symbole-de-génération-graphique E.A,E.A(numéro E.A.) ? (allumé § | éteint) ? ;

génération-de-segments ::= (nom-de-segement | (segment nom-de-figure)) symbole-de-génération-graphique E.A,E.A.,E.A.,E.A.(numéro E.A.)? (allumé § | éteint)? ;

génération-de-texte ::= (nom-de-texte | (texte nom-de-figure)) symbole-de-génération-graphique expression-de-texte (E.A.,E.A.)? (numéro E.A.)? (allumé § | éteint)? ;

génération-multiple-de-points-ou-de-ligne ::= (nom-de-point | nom-de-ligne | ((point | ligne) nom-de-figure)) symbole-de-génération-graphique E.A. fois coordonnée, coordonnée (numéro E.A.)? (allumé § | éteint) ? ;

génération-multiple-de-segements ::= (nom-de-segement | (segment nom-de-figure)) symbole-de-génération-graphique E.A. fois coordonnée, coordonnée, coordonnée,coordonnée (numéro E.A.) ? (allumé § | éteint) ? ;

Sémantique :

Un objet peut être désigné par un nom ou un numéro ; le nom est unique, le numéro ne l'est pas forcément ; quand on agit ultérieurement sur un objet repéré par un numéro, c'est le premier objet qui porte ce numéro dans la figure considérée que l'on atteint. On peut ajouter un numéro à un objet désigné par un nom.

mise-a-jour-de-points-lignes-ou-positions ::= (nom-de-point|nom-de-ligne|nom-de-position|((point|ligne|position) nom-de-figure)) symbole-de-mise-à-jour E.A.,E.A. (numéro E.A.) ? ;

mise-à-jour-de-segments ::= (nom-de-segment|(segment nom-de-figure)) symbole-de-mise-à-jour E.A.,E.A.,E.A.,E.A.(numéro E.A.)? ;

mise-à-jour-de-texte ::= (nom-de-texte|(texte nom-de-figure)) symbole-de-mise-à-jour expression-de-texte (E.A.,E.A.à ? (numéro E.A.)? ;

mise-à-jour-multiple-de-points-ou-lignes ::= (nom-de-point|nom-de-ligne|((point ligne)nom-de-figure)) symbole-de-mise-à-jour E.A. fois coordonnée, coordonnée (numéro E.A.)? ;

mise-à-jour-multiple-de-segments ::= (nom-de-segment|(segment nom-de-figure)) symbole-de-mise-à-jour E.A. fois coordonnée, coordonnée, coordonnée, coordonnée (numéro E.A.) ? ;

Sémantique :

Dans le cas où l'on utilise la forme "désignateur nom-de-figure", il est nécessaire d'indiquer "numéro expression-arithmétique", qui est le seule moyen de se référer à l'objet à mettre à jour. On ne peut en effet mettre à jour qu'un objet qui a déjà été créé, et qui normalement est même déjà affiché ; on ne peut pas changer la nature d'un objet en le mettant à jour. D'autre part, comme il s'agit d'un remplacement, dans la mémoire

d'entretien du terminal, des seules données qui concernent l'objet en question, on peut mettre moins de données qu'avant, autant, mais pas plus ; de même, les caractéristiques suivantes ne peuvent pas être modifiées : caractéristique "sortie" indiquée pour la figure, état allumé ou éteint, taille et protection des caractères.

Avant d'effectuer une mise à jour, il faut informer le système du nouveau point de départ absolu du faisceau lumineux, au moyen de l'instruction reprise, ou confirmer que la position de départ est la position actuelle,connue par un moyen quelconque, au moyen de l'instruction placer. Ceci est également nécessaire si l'on veut générer de nouveaux objets après avoir effectué une mise à jour dans la même figure. Si à un objet désigné par son nom on associe un numéro, celui-ci remplace celui qu'il avait précédemment, ou le supprime si l'on donne un numéro nul.

instruction-placer ::= placer nom-de-figure en E.A., E.A. ;
instruction-reprise ::= reprise nom-de-figure en E.A., E.A. ;

D) TRAITEMENT DES FIGURES

instruction-mettre-arrêt ::= mettre arrêt (nom-d'arrêt|désignation-par-numéro)(effectif § ignoré) ? ;

Sémantique :

Quand le programme passe sur un arrêt effectif, il se produit une interruption (de numéro 35), et l'image disparaît par arrêt de la régénération automatique ; celle-ci reprend dès que l'on a exploité cette interruption au moyen de la fonction état.

instruction-allumer-ou-éteindre ::= (allumer|éteindre) (nom-de-figure|nom-d'objet|nom-de-séquence|nom-d'arrêt|désignation-par-numéro) ;

Sémantique :

On peut allumer ou éteindre une figure entière, une séquence, un objet seul ou un arrêt (c'est-à-dire le rendre effectif ou ignoré) ; dans le cas des textes, des positions et des séquences éteints, le faisceau lumineux ne se déplace pas, alors que pour les autres objets il vient prendre la place qu'il aurait prise si l'objet avait été allumé.

instruction-afficher ::= afficher nom-de-figure (,nom-de-figure)?8
instruction-ordonner ::= ordonner nom-de-figure (,nom-de-figure)?8

Sémantique :

L'instruction ordonner modifie l'ordre de succession dans le cycle de régénération ; en effet, si l'on pointe le crayon optique à l'intersection de deux objets appartenant à deux figures différentes, seule la détection de la figure qui apparaît la première dans le cycle de régénération est reconnue, même si la détection par le crayon optique n'est autorisée que pour la seconde. Les figures non mentionnées dans cette instruction sont placées à la suite de celle qui sont citées, dans l'ordre où elles ont été commencées.

instruction-annuler ::= annuler (nom-de-figure|nom-d'objet|nom-de-séquence|nom-d'arrêt|désignation-par-numéro) ;

Sémantique :

L'objet, la séquence ou l'arrêt et tout ce qui suit dans la figure, c'est-à-dire tout ce qui a été généré après, est supprimé de la figure. Si l'on a indiqué un nom de figure, toute celle-ci est supprimée, mais son utilisation n'est pas terminée, c'est-à-dire que l'on peut générer à nouveau des objets, en n'oubliant pas de commencer par une instruction placer ou reprise.

E) INSTRUCTIONS DIVERSES

instruction-mettre-curseur ::= mettre curseur (nom-de-texte | désignation-par-numéro | nom-de-séquence) (en E.A.)? ;

Sémantique :

Le curseur est un petit symbole placé en-dessous de l'emplacement d'un caractère, et unique pour un terminal graphique ; il marque l'endroit de l'écran où pourra être écrit le prochain caractère au moyen du clavier alphanumérique ; il peut aussi servir de délimiteur pour une lecture de la mémoire d'entretien. Si l'expression arithmétique après en n'est pas précisée, le curseur est placé au début du texte indiqué ou du premier texte de la séquence indiquée ; si l'on met un curseur dans une figure équivalente non affichée, il n'apparaîtra que quand on affichera cette dernière figure. Enfin, même s'il apparaît un curseur sur l'écran, on ne peut effectivement écrire à l'aide du clavier alphanumérique que si le texte au début duquel est placé le curseur, est en mode non protégé.

instruction-enlever-curseur ::= enlever curseur nom-de-figure ;

instruction-lire ::= lire E.A. (données | caractères) en identificateur-de-tableau (jusqua fin | curseur) ?
(de (nom-d'objet | nom-d'arret | E.A.))?
(à (nom-d'objet | nom-d'arrêt | E.A.))?
(figure nom-de-figure)? (réponse variable)?
;

Sémantique :

Cette instruction lit en mémoire centrale une partie du contenu de la mémoire d'entretien ; si l'on omet l'option jusqu'a, l'option par défaut est fin s'il s'agit de données et curseur s'il s'agit de caractères. La quantité exprimée après lire est un nombre d'octets.

Si l'on précise curseur, la lecture s'arrête avant épuisement du comptage si l'on rencontre un curseur. La variable après réponse contiendra zéro si l'on a lu la quantité demandée, sinon soit une quantité positive représentant ce qui a été lu jusqu'à la fin de l'objet désigné par à, ou jusqu'à la fin de la figure si à est omis, soit une quantité négative si l'on a été arrêté par le curseur.

Si après de ou à on indique un nom, il n'est pas nécessaire d'indiquer de nom de figure ; de toutes manières, il faut qu'il s'agisse de la même figure, et que l'une au moins des trois mentions de, à et figure soit présente.

instruction-lirecrayon ::= lirecrayon nom-de-figure en variable, variable ;

Sémantique :

Un balayage de tout l'écran au moyen d'un caractère rend possible la détection par le crayon optique d'un point quelconque, dont les coordonnées sont rangées dans les deux variables indiquées, dans le système et l'échelle de la figure précisée, qui n'est là que pour cela.

instruction-sonnette ::= sonnette ;

fonction-abscisse-ou-ordonnée ::= (abscisse|ordonnée) nom-de-figure(entier|réel)(supposé|effectif §)?

Sémantique :

Les deux variables graphiques, de type entier ou réel, repèrent la dernière position du faisceau lumineux dans la figure indiquée ; par l'option supposée on demande la position où serait le faisceau s'il n'y avait pas de coupage (si la figure ne dépasse pas les limites imposées, supposé et effectif donnent le même résultat). Si le type indiqué est incorrect, la variable prend la valeur négative maximum.

fonction-type ::= type nom-de-figure (x|y|sortie|texte)

Sémantique :

Cette variable graphique entière permet de savoir quelles sont les caractéristiques de la figure, d'après le tableau suivant :

option	valeur	signification
x ou y	1	réelabsolu
"	2	réelrelatif
"	3	entierabsolu
"	4	entierrelatif
sortie	1	optimisé
"	2	absolu
"	3	incrémentiel
texte	1	petit protégé
"	2	grand protégé
"	3	petit non protégé
"	4	grand non protégé

fonction-erreur ::= erreur entier-sans-signe

Sémantique :

Cette variable graphique entière donne des renseignements sur les erreurs qui ont pu éventuellement se produire dans le déroulement de l'instruction graphique précédente. Les éventualités que l'on examine sont les suivantes :

1 : Y a-t-il eu coupage ?
2 : Y a-t-il eu erreur d'échelle ?
3 : Y a-t-il eu dépassement de capacité de la mémoire d'entretien
4 : Y a-t-il eu des paramètres incorrects ?
5 : Y a-t-il eu une erreur d'entrée ou sortie ?

La variable erreur prend soit la valeur zéro, pour indiquer qu'il n'y a pas eu d'erreur, soit la valeur la plus forte supérieure ou égale à celle qui est examinée. Si l'on obtient une réponse de 1 à 4, on peut demander un supplément d'information au moyen de erreur 9 qui prend, dans chacun des quatre cas, une valeur donnant l'un des renseignements suivants :

cas 1 : en cas de génération multiple, le numéro de l'élément pour lequel s'est produit la coupure, ou 1 pour une position ou une reprise.

cas 2 : le même renseignement, mais relatif à l'erreur d'échelle.

cas 3 : 1 : pas assez de place pour une mise à jour.
2 : les 128 octets d'une figure de cette taille ont été dépassés.
3 : plus assez de place en mémoire centrale.

cas 4 : 0 : erreur dûe à plusieurs paramètres.
1 à n : numéro d'ordre du paramètre fautif (suivant l'ordre interne du sous-programme G.S.P.).

instruction-fingraphique ::= fingraphique ;

Sémantique :

Cette instruction marque la fin logique du programme, et doit être la dernière exécutée.

C) EXEMPLE

Premier exemple :

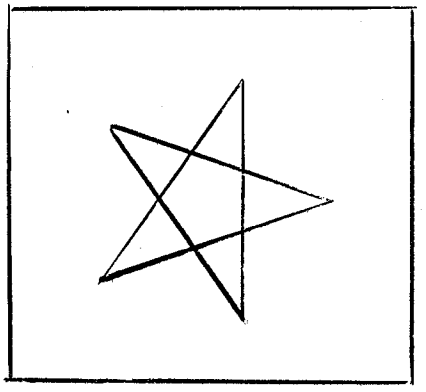
Le programme graphique rédigé plus loin effectue le travail suivant :

1) Il affiche 5 points et les relie entre eux pour construire un pentagone régulier étoilé.

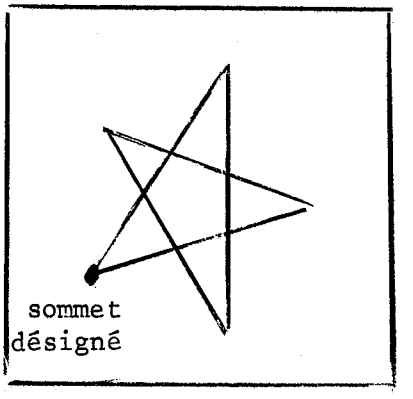
2) Il autorise et attend une détection de l'un des cinq points à l'aide du crayon optique. Il interdit alors les détections sur les points, et rend possible, grâce à l'instruction lirecrayon, la détection d'un point quelconque de l'écran. Il reconstruit alors la figure en utilisant,à la place du point repéré plus haut, le point déterminé à l'instant.

3) Les touches 1,2 et 3 sont rendues actives et leurs indicateurs allumés ; si l'opérateur appuie sur la touche 1, on recommence à la phase 1), si il appuie sur la touche 2 on recommence à la phase 2) ; la touche 3 permet de terminer le travail.

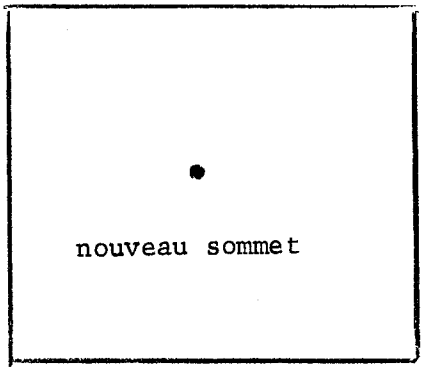
1



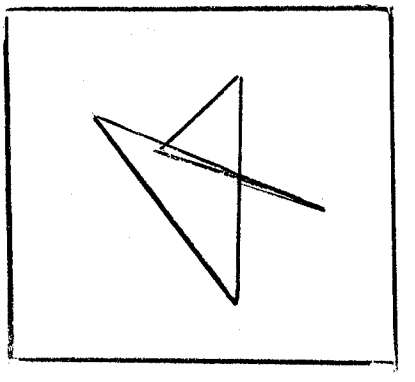
2



2



2



```
tableau réel PX, PY, XIN, YIN (* 5 *), entier REPØNSE (* 10 *) ;
réel C, R = 1 200, THETA = 72, RADIAN ;
niveau INTERØ ;
figure PØINTS ; figure LIGNES ;
position DEPART figure LIGNES ; ligne PØLYGØNE ;
commencer figure POINTS, LIGNES ;
commencer niveau INTERØ ;
C := 3.14159/180 ;
pour I := 0 pas 1 à 4 ;
RADIAN := THETA * I * C ;
XIN (* I + 1 *) := 2047 + R * CØS (RADIAN) ;
YIN (* I + 1 *) := 2047 + R * SIN (RADIAN) ;
finpour ; (calcul des coordonnées des sommets du pentagone)
E1 : PX(*1*) := XIN(*1*) ; PY(*1*) := YIN(*1*) ;
PX(*2*) := XIN(*3*) ; PY(*2*) := YIN(*3*) ;
PX(*3*) := XIN(*5*) ; PY(*3*) := YIN(*5*) ;
PX(*4*) := XIN(*2*) ; PY(*4*) := YIN(*2*) ;
PX(*5*) := XIN(*4*) ; PY(*5*) := YIN(*4*) ;

(réordonner les sommets pour composer un pentagone étoilé)

pour I := 1 pas 1 à 5 ;
point PØINTS <== PX(*I*), PY(*I*) numéro I ; finpour ;
  (génération des sommets avec numéros d'identification)
DEPART <== PX(*5*), PY(*5*) ;
POLYGØNE <== 5 fois PX(*1 pas 1*), PY(*1 pas 1 *) ;
     (génération des côtés en une seule instruction)
afficher PØINTS, LIGNES ;
E2 : traiter INTERØ(*34*) ; (permettre l'utilisation du crayon optique)
crayon PØINTS traité ;
I := état INTERØ(*34*) attente résultat REPØNSE ; (attente d'une détection
     rangement des caractéristiques de l'élément détecté dans le tableau
     REPONSE)
```

```
ignorer INTERØ(*34*) ;
I := REPØNSE(*4*) ; (numéro de l'objet détecté)
crayon PØINTS ignoré ;
lirecrayon LIGNES en PX(*I*),PY(*I*);
point PØINTS <==> PX(*I*),PY(*I*)numéro I ;
    (mise à jour du point)
reprise LIGNES en PX(*5*),PY(*5*);
PØLYGØNE <==> 5 fois PX(*1 pas 1*),PY(*1 pas 1*) ;
      (mise à jour des côtés)
traiter INTERØ(*1,2,3*);(permettre l'utilisation des touches 1,2 et 3)
touches INTERØ normal ; (allumer les indicateurs lumineux correspondants)
I := état INTERØ(*1,2,3*) attente ; (attente d'action sur une touche)
touches INTERO éteint ;
si I = 1 ; annuler POINTS ; annuler LIGNES ; allera E1 ; finsi ;
si I = 2 ; allera E2 ; finsi ;
fingraphique ;
```

Deuxième exemple :

Il s'agit de l'exemple que nous avons déja traité au cours des chapitres précédents.

```
entier I, UN = 1, FIN = 32, CRAYON = 34 ;
tableau réel Y (*4*) ; chaîne 1 caractères CARAC = " " ;
niveau NIVEAU 1, NIVEAU 2 ;
figure PREMIER ; figure DEUXIEME ;
texte LABAS, LIGNES figure DEUXIEME ; séquence ZIGZAG ;
texte PHRASE figure PREMIER ;
Y(*1*) := 2000 ; Y(*2*) := 1000 ; Y(*3*) := 3000 ; Y(*4*) := 500 ;
commencer figure PREMIER, DEUXIEME ;
```

```
PHRASE <== "EXEMPLE DE PRØGRAMME GRAPHIQUE ELEMENTAIRE. N
CE PRØGRAMME VA DESSINER DES ZIGZAG. N APPUYEZ SUR LA TOUCHE 1
PØUR CØMMENCER. "0,2000 ;
LIGNES <== "MØNTREZ LE DESSIN AVEC LE CRAYØN OPTIQUE, IL N
DISPARAITRA. N APPUYER SUR LA TØUCHE 1, IL REAPPARAITRA. N
TAPEZ F PUIS FIN PØUR TERMINER. "0,4000 ;
option DEUXIEME lettres petites ;
LABAS <=="    ";
commencer séquence ZIGZAG ; reprise DEUXIEME en 0, 0 ;
ligne DEUXIEME <== 4 fois 1000 pas 1000, Y(*1 pas 1*) ;
terminer séquence ZIGZAG ;
commencer niveau NIVEAU1  ;
traiter NIVEAU1  (*1*) ; touches (*1*) ;
afficher PREMIER ; I := état NIVEAU1  (*1*) attente ;
commencer niveau NIVEAU2  ;
terminer figure PREMIER ; afficher DEUXIEME ;
ITER1 : traiter NIVEAU2 (*UN,FIN,CRAYON*) ;
crayon DEUXIEME traité ;
ITER2 ; mettre curseur LABAS ; I := état NIVEAU2  (*UN,FIN,CRAYON*)
attente ;
si I=1 ; allera RALLUME ; finsi ;
si I=FIN ; allera TESTFIN ; finsi ;
si I[1] = CRAYON ; allera ITER1 ; finsi ;
éteindre ZIGZAG ; crayon DEUXIEME ignoré ; allera ITER2 ;
RALLUME : allumer ZIGZAG ; allera ITER1 ;
TESTFIN : lire UN caractères en CARAC de LABAS ;
si CARAC[1] = "F" ; allera ITER2 ; finsi ;
fingraphique ;
```

D) EXTENSIONS ENVISAGEES.

Nous pensons qu'il sera possible d'inclure aisément l'instruction d'affectation multiple, l'instruction pour avec énumération de valeurs, l'usage de tableaux à plusieurs dimensions et de tableaux dont les éléments soient des chaînes de caractères.

Il faudra aussi introduire les procédures. A ce sujet, nous nous demandons s'il ne serait pas intéressant d'envisager un formalisme analogue à celui des instructions graphiques pour ce qui est des appels de procédures ; la déclaration de procédure préciserait la nature des paramètre et définirait éventuellement de nouveaux mots-clés, et même des options par défaut.

Ces idées sont à mettre en liaison avec l'étude de processus simplifiés de compilation des instructions graphiques (traduction en langage intermédiaire décompilable et interprétation) dont nous parlerons pour finir dans le chapitre suivant.

CHAPITRE VI

APERÇU SUR LA COMPILATION DU LANGAGE DE PROGRAMMATION GRAPHIQUE

I - GENERALITES

La compilation se fait en deux phases : traduction du langage graphique en un langage intermédiaire rangé sous forme de listes ; interprétation de ce langage intermédiaire à l'exécution.

De plus la phase de traduction est incrémentielle et conversationnelle :

- incrémentielle : chaque instruction ou déclaration est analysée et traduite de point-virgule en point-virgule, et va accroître la liste du programme, les listes des identificateurs, des variables et des constantes, et éventuellement la liste des étiquettes ;

- conversationnelle : chaque instruction ou déclaration, tapée au clavier par le programmeur, est immédiatement analysée ; si elle est incorrecte, un message d'erreur apparaît sur l'écran avec un pointeur sur la partie fautive ; si elle est correcte, elle est effacée puis réécrite à partir du langage intermédiaire, pour que le programmeur vérifie ce que le compilateur a compris.

Nous voyons ainsi que l'analyse syntaxique est faite complètement, au fur et à mesure, de telle sorte qu'à l'exécution il ne doit plus y avoir d'erreur de ce type.

Enfin le langage intermédiaire doit être "décompilable", c'est-à-dire tel qu'il puisse être retraduit en langage de programmation graphique. En effet, en plus de la vérification indiquée ci-dessus, nous avons vu qu'un programme, précédemment écrit et traduit en langage intermédiaire, peut être repris et modifié (cf Chapitre 2), donc qu'il doit pouvoir apparaître à nouveau en clair sur l'écran.

Pour ce qui est de la phase d'interprétation - que nous ne traiterons pas -, nous rappelons cependant qu'elle fera appel, pour les instructions et les fonctions graphiques, aux sous-programmes de G.S.P.

II - CODIFICATION ET REPRESENTATION

Les informations que manipule le compilateur dans son travail de traduction et de retraduction sont :

1°) La chaîne de caractères correspondant à une instruction ou à une déclaration graphique ;

2°) Des éléments de la liste du programme en langage intermédiaire ;

3°) Des éléments de la liste des identificateurs graphiques, de la liste des variables et des constantes, et de la liste des étiquettes.

Les listes vont constituer des fichiers sur disques pour être conservées et gérées ultérieurement au moyen du langage de commande.

LISTES DES IDENTIFICATEURS, DES VARIABLES ET DES ETIQUETTES

La liste des identificateurs regroupe les noms des entités graphiques déclarées, avec des renseignements sur leur nature (nom de figure, nom de point, etc...); pour les objets, les séquences et les arrêts, on a un pointeur sur la figure à laquelle ils appartiennent ; pour les figures, on a un pointeur sur la description de leurs caractéristiques (provenant de la déclaration de figure).

Dans la liste des variables et des constantes, nous trouvons, pour chacune d'entre elles, un nom, un type et un pointeur vers d'autres renseignements, utilisés à l'interprétation, comme la dimension d'un tableau, la taille d'une chaîne, une valeur initiale.

Pour une étiquette, nous avons besoin de son nom et du pointeur vers l'élément qu'elle désigne dans la liste du programme en langage intermédiaire.

Toutes ces listes sont composées d'éléments, qui occupent chacun 32 octets, soit huit mots-machine. Les éléments sont groupés en blocs de huit éléments. Une piste de disque "2311" peut recevoir 11 blocs, et un cylindre (10 pistes) 110 blocs ; une pile de disques "2311" peut contenir jusqu'à 22 000 blocs (une unité "2314", 6 fois plus).

La partie de mémoire centrale où sont traités les éléments, est organisée en pages - une page correspondant à un bloc - et gérée de la façon suivante (Réf Cohen-1) :

Supposons qu'il y ait place pour 6 pages dans la mémoire centrale ; au fur et à mesure des besoins on fait venir des blocs dans ces six pages. Par ailleurs on tient à jour une table où sont indiqué, pour chaque page,

a) le dernier moment où cette page a été utilisée

b) l'indication que la page a été modifiée ou non.

Lorsque l'on a besoin de faire venir en mémoire centrale une nouvelle page et que tous les emplacements sont occupés, on procède ainsi : on introduit la nouvelle page à l'emplacement de celle dont on s'est servi le moins récemment ; au préalable on aura renvoyé sur disque la page qui se trouvait à cet emplacement, dans le seul cas où elle avait été modifiée.

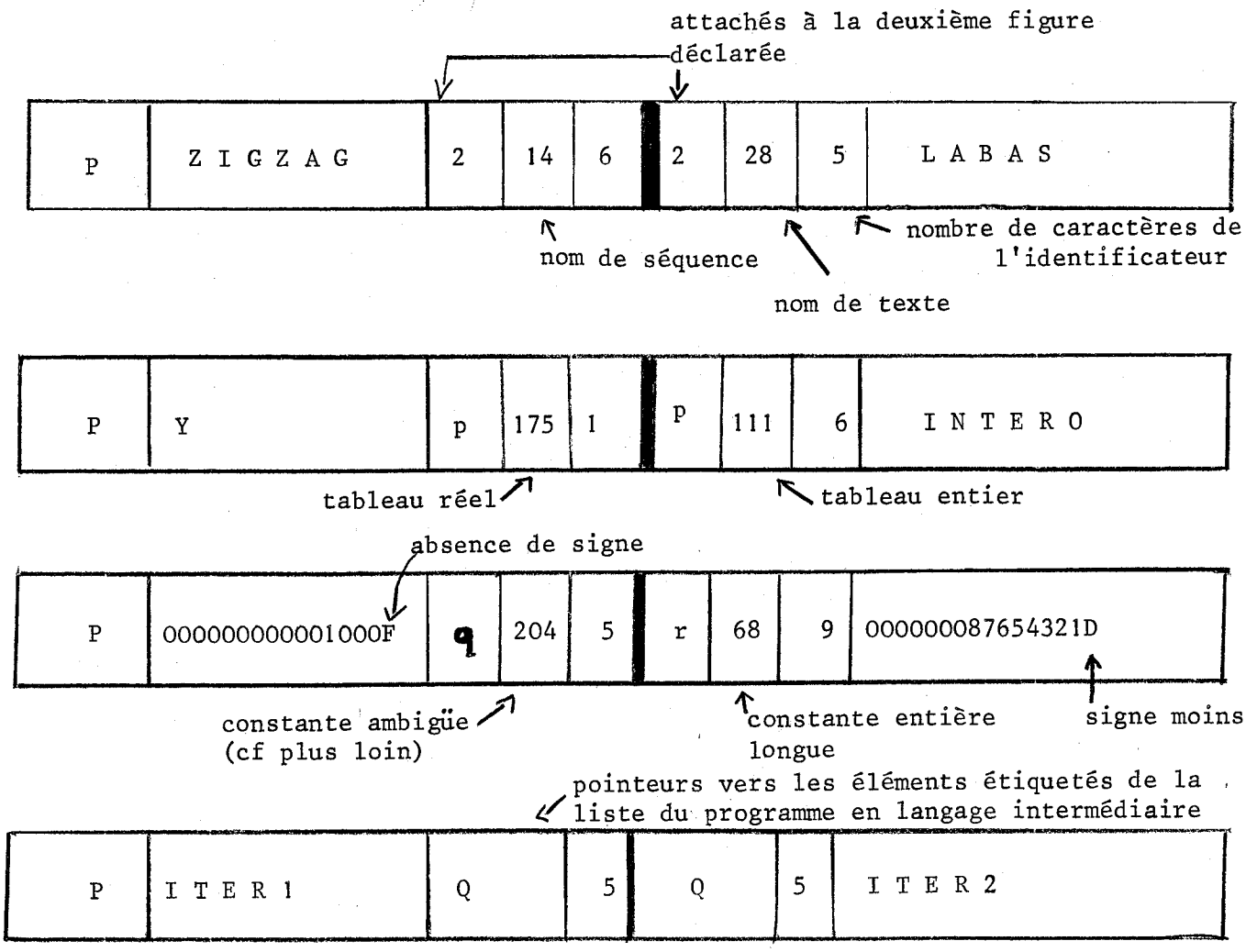
Cette technique a l'avantage de minimiser les opérations de lecture et écriture sur la mémoire de masse que constituent les disques.

En choisissant une codification pour représenter les informations telles que le type d'une variable, la nature d'une entité graphique, etc..., nous avons eu les deux objectifs suivants :

- adapter la configuration des codes et leur emplacement aux instructions du langage machine du 360 qui devront les manipuler ;

- ne pas utiliser toutes les combinaisons, de manière à pouvoir faire aisément des extensions.

EXEMPLES D'ELEMENTS DE LISTES



P : pointeur de liste ; p : pointeur vers la taille du tableau

q : pointeur vers la représentation réelle ; r : pointeur vers la représentation binaire.

LISTE DU LANGAGE INTERMEDIAIRE

Le langage intermédiaire doit être adapté d'une part à la future interprétation, d'autre part à la retraduction.

En ce qui concerne les instructions graphiques, nous savons que l'interprétation consiste à appeler des sous-programmes dont les paramètres seront utilisés suivant un ordre fixe. Cet ordre n'est pas forcément celui de l'apparition des paramètres dans la syntaxe du langage de programmation graphique. Nous gardons cependant cet ordre pour accélérer au maximum le processus d'interprétation.

Une instruction du langage intermédiaire est placée dans un ou plusieurs éléments de liste, chaque élément occupant huit mots de quatre octets chacun .

Les premiers mots du premier élément renferment des renseignements particuliers : pointeur vers la suite, pointeur vers l'instruction précédente, pointeur vers l'instruction suivante, code de l'instruction et nombre total de mots utilisés par la partie fixe de l'instruction.

Les mots suivants contiennent les paramètres dans l'ordre où ils seront utilisés à l'interprétation pour servir à l'appel des sous-programmes de G.S.P.

Ces paramètres sont codifiés de la façon suivante : l'octet de gauche indique sa nature, les trois autres octets servent

- soit à mettre un pointeur vers un élément d'une des listes définies plus haut ;
- soit à mettre sa valeur ;

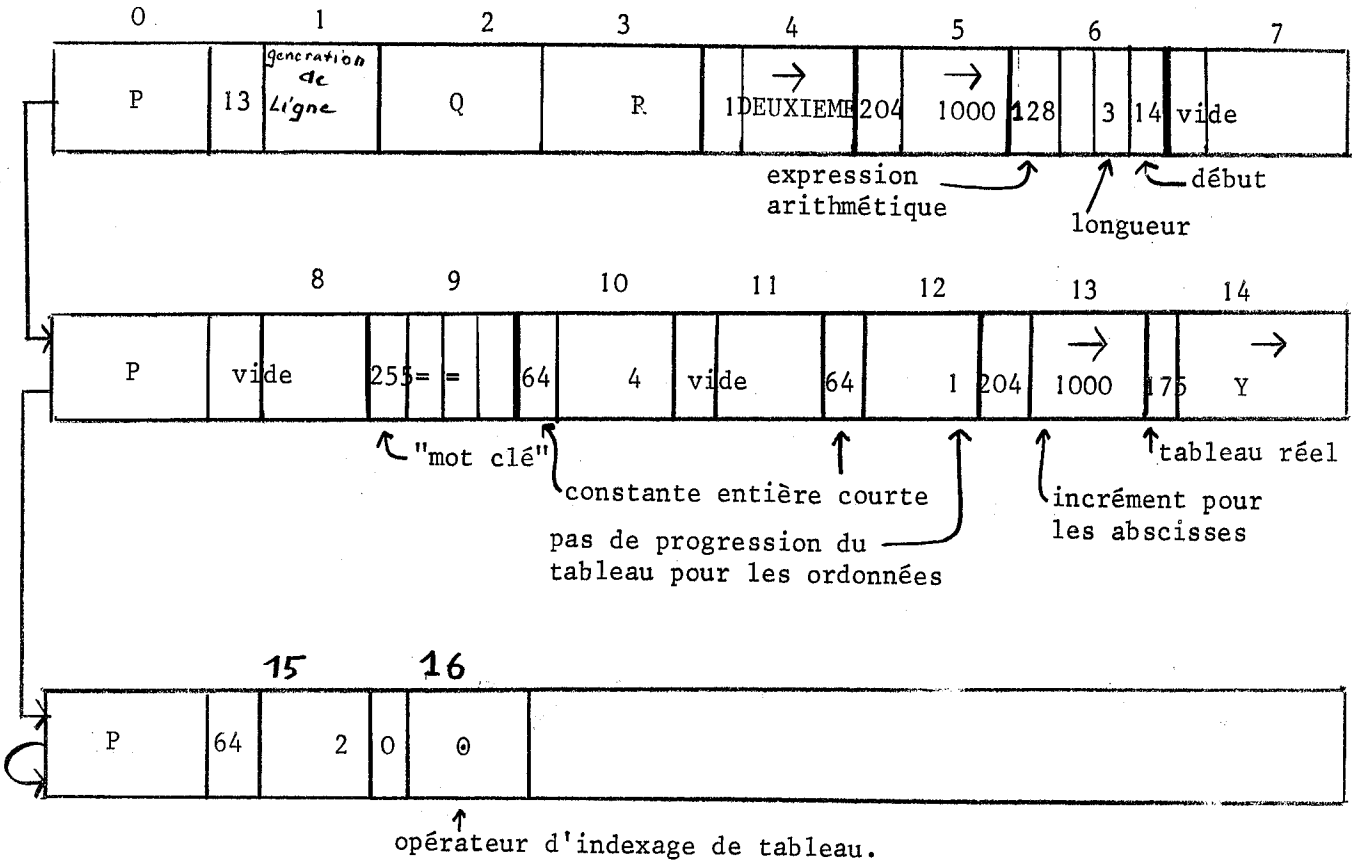
- soit à renvoyer vers la description post-fixée d'une expression (de telles expressions sont rangées après la partie fixe correspondant aux paramètres).

Exemples

L'instruction

ligne DEUXIEME <== 4 fois 1000 pas 1000 , Y (* 2 pas 1 *)

sera ainsi représentée en langage intermédiaire :



P : pointeur sur l'élément suivant

Q : pointeur sur l'instruction précédente

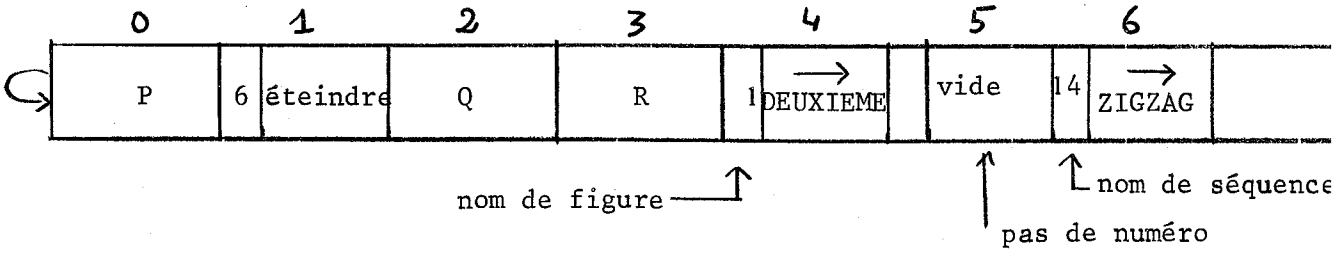
R : pointeur sur l'instruction suivante

mémoire 5 : première abscisse

mémoire 6 : première ordonnée

→ : "pointeur vers"

de même pour l'instruction éteindre Z I G Z A G ; on aura :



Voyons plus précisément ce que l'on peut trouver dans ces mots :

1°) Dans la partie fixe.

a) Pour un identificateur graphique : octet de gauche, codification de sa nature (figure, point, ...) ; partie droite, pointeur vers la liste des identificateurs graphiques.

b) Pour un mot-clé : octet de gauche, code de valeur binaire maximum ; partie droite, valeur numérique entière qui est celle utilisée par G.S.P et qui est en relation biunivoque avec le mot-clé (traité : 1, ignoré : 2) .

c) Pour une variable simple : octet de gauche, son type ; partie droite, pointeur vers la liste des variables et des constantes.

d) Pour une variable indicée : octet de gauche, code d'une expression arithmétique ; partie droite, renvoi vers la partie variable où est mise en notation post-fixée sa désignation complète.

e) Pour une constante de type entier ; on a deux cas :

- si la constante est comprise entre -99 et +99, octet de gauche, code de "constante entière courte" ; partie droite sa valeur.

- Dans le cas contraire, octet de gauche, code de "constante entière" ; partie droite, pointeur vers la liste des variables et des constantes. (Nota : une telle constante sera traitée comme une variable dont le nom est sa représentation décimale codée condensée ; sur huit octets on peut mettre un entier de quinze chiffres avec signe).

f) Pour une constante de type réel : octet de gauche, code de "constante réelle" ; partie droite, pointeur vers la liste des variables et des constantes (en retraduction elle apparaîtra sous une forme normalisée).

g) Pour une constante de type ambigu : il s'agit d'une constante ne comportant ni point décimal ni exposant mais qui est un élément de coordonnée sur l'écran. Etant donné qu'on ne connaîtra son type qu'au moment de l'exécution, on garde son aspect "entier" comme en e), mais on prépare également une représentation "réelle".

Octet de gauche, code de "constante ambiguë" ; partie droite, pointeur vers la liste des variables et des constantes.

h) Pour une constante de type chaîne de caractères : octet de gauche, code de "chaîne" ; partie droite, pointeur vers la liste des variables et des constantes.

i) Pour une expression : octet de gauche, code d'"expression arithmétique entière", d'"expression arithmétique", ou d'"expression de texte" ; partie droite, pointeur vers les mots de la partie variable où se trouve la représentation post-fixée de l'expression (ce pointeur est

constitué du numéro du mot, où commence l'expression, et du nombre de mots utilisés.

2°) Dans la partie variable

On retrouve les cas cités en c), e), f), g) et h) ; de plus nous avons besoin de représenter les opérateurs et les séparateurs : + - * / * * ||sélection depuis longueur @ (opérateur d'indexage de tableau) et ↓ (opérateur d'appel de fonction). Pour ces derniers l'octet de gauche aura la valeur zéro et la partie droite une codification adéquate.

Enfin les fonctions standard seront indiquées suivant le même procédé : octet de gauche, "fonction standard" ; partie droite, nature de celle-ci (sinus, racine carrée, etc...).

III - MODULARITE DU TRAITEMENT PAR MACRO-INSTRUCTIONS

Les actions à mener pour traiter une instruction graphique, tant en traduction en langage intermédiaire qu'en retraduction en langage de programmation graphique, sont en petit nombre, et sont simples. Ce sont par exemple :

- traitement d'un identificateur graphique ;
- traitement d'un mot-clé ;
- traitement d'une expression ;
- traitement d'une coordonnée.

En traduction, il y a évidemment des contrôles de validité, lesque ne figurent plus en retraduction.

Ces constatations nous conduisent à organiser les phases de traduc tion et de retraduction du compilateur - en ce qui concerne les instructions graphiques - de la manière suivante : chaque action simple est définie sous la forme d'une Macro-Instruction ;

la séquence de traitement d'une instruction graphique est formée d'une succession de telles macro-instructions.

Exemples en traduction :

Pour l'instruction crayon, dont le modèle est

crayon nom-de-figure { traité / ignoré } ;

nous aurons la séquence de traitement

```
IG          4,1
CODE        5,('TRAITE','IGNORE'),ERMOTCLE
                  (1)
```

qui fabriquerait l'élément de langage intermédiaire qui suit, pour l'instruction crayon DEUXIEME traité ;

0	1	2	3	4		5
P	5 crayon	Q	R	1	→ DEUXIEME 255	1

I G recherche un nom de figure et garnit la mémoire 4.
CODE recherche un des deux mots-clés et met 1 pour traité, 2 pour ignoré, ou détecte une erreur.

---

(1) : Il est probable que les mots réservés du langage graphique, écrits en minuscules et soulignés dans notre étude, seront en fait écrits en majuscules, entre apostrophes ou non.

Pour l'instruction mettre arrêt, de modèle

mettre arrêt { nom-d'arrêt / nom-de-figure numéro expression-arithmétique } [[{ effectif / ignoré }]]

nous aurons :

```
            IG      6,19,E1
            B       E2
E1          IG      4,1
            LIRE    'NUMERO'
            SPEAE   5,8
E2          CODE    7,('EFFECTIF','IGNORE')
```

dont le résultat pour mettre arrêt PREMIER numéro 15 ignoré ; serait :

0		1	2	3		4		5	6		7
P	7	mettre arrêt	Q	R	1	→ premier	64	15		255	

Ici 19 est le code d'un nom d'arrêt ; si le nom examiné n'est pas un nom d'arrêt, il y a branchement en E1 pour analyser s'il s'agit bien alors d'un nom de figure.

La macro-instruction SPEAE appelle un sous-programme de traitement des expressions arithmétiques entières, et on indique ($2^{ème}$ paramètre : 8) à partir d'où mettre, le cas échéant, l'écriture post-fixée.

Enfin le mot-clé final est facultatif.

Avantages de cette méthode

L'écriture du compilateur, dans ses phases de traduction et de retraduction, est rendue plus aisée et plus rapide au moyen de ces macro-instructions. De même la mise au point et l'essai du compilateur sont simplifiés une fois que chaque macro-instruction a été duement testée.

Nous pensons qu'une technique analogue pourra être utilisée pour l'écriture de la phase d'interprétation du compilateur, spécialement pour l'appel des sous-programmes de G.S.P.

Un autre intérêt de cette méthode de macro-instructions est de se prêter à une adaptation sur un autre ordinateur : les fonctions restant les mêmes, il n'y a qu'à changer les macro-définitions. On peut envisager par exemple d'adapter le système complet à l'ordinateur IBM 1130 auquel peut être relié un terminal graphique IBM 2250 modèle 4.

IV - LA COMPILATION DE LA PARTIE NON GRAPHIQUE DU LANGAGE.

Nous donnerons quelques précisions sur ce sujet sans développer tous les aspects car les problèmes soulevés ont été déjà pour la plupart étudiés et résolus.

Nous avons vu que les propositions pour et si ont une portée signalée par les délimiteurs finpour et finsi, de sorte que l'analyse en sera simplifiée, et qu'une technique de pile résoudra les situations d'imbrication, à la traduction comme à l'interprétation.

Pour les expressions nous avons déjà dit qu'elles sont conservées sous forme post-fixée ; en effet d'une part cette forme est adaptée à une exécution interprétative, d'autre part on peut décompiler une telle expression, c'est à dire la reconstituer telle que le programmeur l'avait primi-

tivement écrite. Voici, décrit en Algol, l'algorithme qui recompose l'expression minimale - c'est à dire sans parenthèses superflues - à partir d'une expression post-fixée:

```
début entier I,J,K,L,PARGAUCHE,PARDROITE,OMEGA ;
      entier tableau POSTFIXE[1:30],PILE[1:10];
      entier procédure PRIORITE(X);début commentaire DONNE LA PRIORITE
            DE L'OPERATEUR X (0 POUR UNE PARENTHESE,1 POUR + OU -,
            2 POUR * OU /, 3 POUR ↑);... fin;
      entier procédure NOP(X);début commentaire DONNE LE NOMBRE D'OPERANDES
            ASSOCIES A L'OPERATEUR X, OU 0 SI X EST UNE PARENTHESE ; ... fin
      booléen procédure OPERATEUR(X);...;
      booléen procédure OPERANDE (X);...;
      I := 31 ; J := 50 ; L := 0 ; K := 1 ;
      pour I := I-1 tant que POSTFIXE[I]≠ OMEGA faire
      début si OPERATEUR(POSTFIXE[I])alors
            début si PRIORITE(POSTFIXE[I])<L alors
                  début PILE[K] := PARGAUCHE ;
                        PILE[K+1] := POSTFIXE[I] ; K := K+2 ;
                        RECOMPOSE[J] := PARDROITE ; J := J-1
                  fin sinon début PILE[K] := POSTFIXE[I] ; K := K+1 fin ;
            L := PRIORITE(POSTFIXE[I]) fin ;
            si OPERANDE(POSTFIXE[I]) alors
               début RECOMPOSE[J]:= POSTFIXE[I] ; J := J-1 ;
            E : K := K-1 ; RECOMPOSE[J] := PILE [K] ; J := J-1
                L := PRIORITE(PILE[K]) ;
                si NOP(PILE[K]) ≠ 2 et K ≠ 1 alors allerà E
               fin
      fin
fin ;
```

Exemple

L'expression

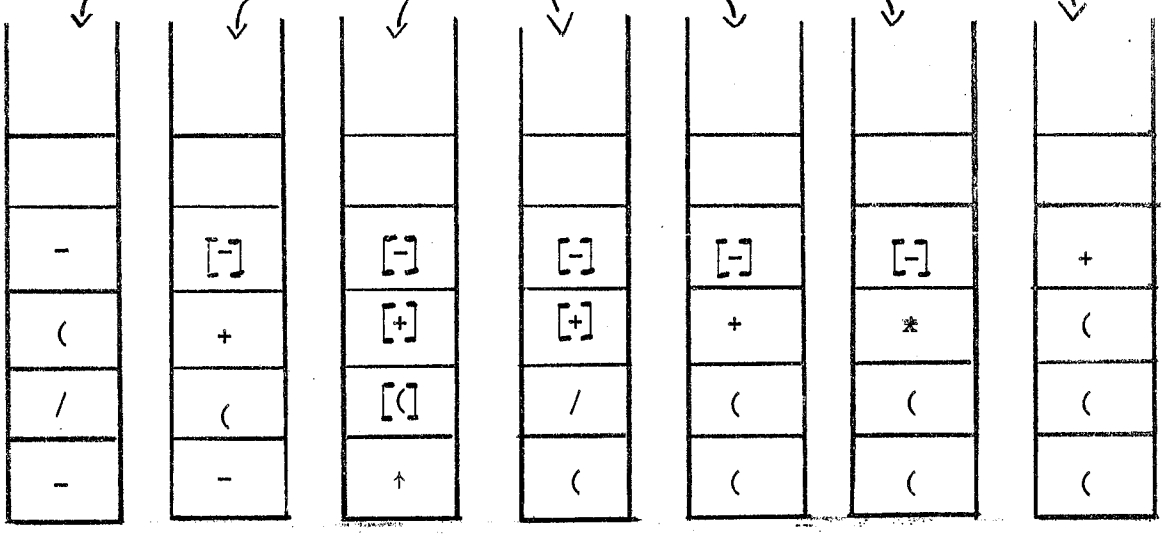
11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Ω	A	B	+	C	*	D	+	E	/	F	↑	G	H	+	I	J	-	/	-	

est récomposée en

22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
(	(	(	A	+	B	)	*	C	+	D	)	/	E	)	↑	F	-	(	G	+	H	)	/	(	H	-

les valeurs successives de K et de L, et les dispositions de la pile étant :

L	0	1	2	1	1	0	2	1	1	0	1	3	3	2	2	1	1	2	2	1	1	0	0	0		
K	1	2	3	5	4	3	2	4	3	2	1	2	1	3	2	4	3	4	3	5	4	3	2	1		



Conclusion

Nous espérons avoir apporté une modeste contribution aux efforts qui se déploient actuellement dans le domaine de la programmation et de l'utilisation des terminaux graphiques.

Nous avons principalement centré notre travail sur la définition d'un langage de programmation graphique ; étant donné que la mise en oeuvre de ce dernier s'insère dans un travail d'équipe en cours de réalisation, nous n'avons pas à proprement parler de "résultats" quantitatifs à présenter sur la souplesse d'utilisation, la rapidité d'exécution ou la variétés des champs d'applications.

Nous devons rappeler aussi que, dans le but d'obtenir rapidement un système opérationnel, nous nous sommes servi d'un soubassement de programmation déjà existant, ce qui diminue la généralité de l'étude ; mais la porte reste ouverte à une recherche ultérieure plus fine sur les problèmes liés d'une part à la structure et à la manipulations de données destinées à l'affichage de dessins, d'autre part à l'usage optimal des dispositifs technologiques d'un terminal graphique évolué.

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

-:-:-:-:-:-

Bol-1 — *L. BOLLIET* : Utilisation des ordinateurs à distance en temps réel et en temps partagé. (DUNOD).

Bol-2 — *L. BOLLIET* : Notation et processus de traduction des langages symboliques, Thèse, Université de Grenoble, Juin 67.

CLL — *E. CLEEMANN, O. LECARME, M. LUCAS* : Langages de programmation graphique, Revue Française d'informatique et de recherche opérationnelle, N° 12, 1968.

COHEN-1 — *J. COHEN* : Langage pour l'écriture des compilateurs, Thèse, Université de GRENOBLE, Juin 67.

LEC-1 — *O. LECARME, E. CLEEMANN* : Système d'utilisation rationnelle d'un terminal évolué, Séminaire de Programmation de l'I.M.A.G., Juin 68.

LEC-2 — *O. LECARME* : Système de programmation graphique conversationnelle, Colloque sur les systèmes conversationnels, GRENOBLE Novembre 68

LUCAS-1 — *M. LUCAS*, Techniques de programmation et d'utilisation en mode conversationnel des terminaux graphiques, Thèse, Université de GRENOBLE, Juin 68.

IBM-1 — *IBM System/360* component description : IBM 2250 display unit mode form A27-2701.

IBM-2 — *IBM System/360* operating system : graphic programming services for IBM 2250 display unit, form C27-6909.

IBM-3 *IBM System/360* operating system : graphic programming services for Fortran IV, form C27-6032.

IBM-4 *IBM System/360* operating system : graphic programming services for Fortran IV (Program Logic), form Y27-7152.

IBM-5 *IBM System/360* operating system : Assembler language, form C28 - 6514.

S.A. COONS : An outline of the requinements for a computer-aided design system.Proceedings-Spring Joint computer conférence 1963)

T.E. JOHNSON : SKETCHPAD III, A computer program for drawing in three dimensions (Proceedings-SJCC 1963)

D.T. ROSS et J.E. RODRIGUEZ : Théoretical foundations for the computer-aided design system.(proceedings-SJCC 1963)

R. STOTZ : Man machine console facilities for computer-aided design (Proceedings - SJCC 1963)

I.E. STUHERLAND : SKETCHPAD, A man-machine graphical communication system (Proceedings - SJCC 1963).

I.E. SUTHERLAND : Computer graphics, ten unsolved problems. (Datamation may 1966).

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 GRENOBL