Conception et réalisation d'un logiciel graphique de base indépendant de son contexte : application au logiciel GRIGRI

André Leduc-Leballeur

THESE

présentée à

Université Scientifique et Médicale de Grenoble

Institut National Polytechnique de Grenoble

pour obtenir le grade de
Docteur Ingénieur

par

André LEDUC-LEBALLEUR

CONCEPTION ET REALISATION D'UN LOGICIEL GRAPHIQUE DE BASE INDEPENDANT DE SON CONTEXTE

APPLICATION AU LOGICIEL GRIGRI

Thèse soutenue le 30 septembre 1977 devant la Commission d'Examen

Président : L. BOLLIET

Examinateurs R.A. GUEDJ
M. LUCAS

Rapporteur : G. VEILLON

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

Monsieur Gabriel CAU : Président
Monsieur Pierre JULLIEN : Vice Président

MEMBRES DU CORPS ENSEIGNant : DE L'U.S.M.G.

PROFESSEURS TITULAIRES

MM	AMBLARD Pierre	Clinique de dermatologie
	ARNAUD Paul	Chimie
	ARVIEU Robert	I.S.N
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique Expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale
	BEAUDOING André	Clinique de Pédiatrie et Puériculture
	BELORIZKY Elie	Physique
	BERNARD Alain	Mathématiques Pures
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques Pures
	BEZEZ Henri	Pathologie chirurgicale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Clinique gastro-entérologique
Mme	BONNIER Marie-Jeanne	Chimie générale
MM.	BOUCHERLE André	Chimie et toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques appliquées
	BOUTET DE MONTVEL Louis	Mathématiques Pures
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie végétale
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Clinique Oto-rhino-laryngologique
	CHATEAU Robert	Clinique de neurologie
	CHIBON Pierre	Biologie animale
	COEUR André	Pharmacie chimique et chimie analytique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie pathologique
Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumophtisiologie

MM.	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée (IUT I)
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	GAGNAIRE Didier	Chimie Physique
	GALVANI Octave	Mathématiques Pures
	GASTINEL Noël	Analyse numérique
	GAVEND Michel	Pharmacologie
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques Pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	KAHANE André	Physique générale
	KLEIN Joseph	Mathématiques Pures
	KOSZUL Jean-Louis	Mathématiques Pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques Appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
Mme	LAJZEROWICZ Janine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie Pharmaceutique
	LAURENT Pierre	Mathématiques Appliquées
	LEDRU Jean	Clinique médicale B
	LE ROY Philippe	Mécanique (IUT I)
	LLIBOUTRY Louis	Géophysique
	LOISEAUX Pierre	Sciences Nucléaires
	LONGEQUEUE Jean-Pierre	Physique Nucléaire
	LOUP Jean	Géographie
Melle	LUTZ Elisabeth	Mathématiques Pures
MM.	MALINAS Yves	Clinique Obstétricale
	MARTIN-NOEL Pierre	Clinique Cardiologique
	MAZARE Yves	Clinique Médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MICOUD Max	Clinique Maladies infectieuses
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie Nucléaire
	NOZIERES Philippe	Spectrometrie Physique
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques Pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET Jean	Semeiologie Médicale (Neurologie)
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REVOL Michel	Urologie
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-Chirurgie
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SIBILLE Robert	Construction mécanique (IUT I)
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale

MM.	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique Nucléaire
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
MM.	VERAIN André	Physique
	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale

PROFESSEURS ASSOCIES

MM.	CRABBE Pierre	CERMO
	DEMBICKI Eugéniuz	Mécanique
	JOHNSON Thomas	Mathématiques appliquées
	PENNEY Thomas	Physique

PROFESSEURS SANS CHAIRE

Mle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Gilbert	Géographie
	BENZAKEN Claude	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BILLET Jean	Géographie
	BOUCHET Yves	Anatomie
	BRUGEL Lucien	Energétique (IUT I)
	BUISSON René	Physique (IUT I)
	BUTEL Jean	Orthopédie
	COHEN ADDAD Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie
	CONTE René	Physique (IUT I)
	DELOBEL Claude	M.I.A.G.
	DEPASSEL Roger	Mécanique des fluides
	FONTAINE Jean-Marc	Mathématiques Pures
	GAUTRON René	Chimie
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	GROULADE Joseph	Biochimie médicale
	HACQUES Gérard	Calcul numérique
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et Médecine préventive
	IDELMAN Simon	Physiologie animale
	JOLY Jean-René	Mathématiques Pures
	JULLIEN Pierre	Mathématiques Appliquées
Mme	KAHANE Josette	Physique
MM.	KRAKOWIACK Sacha	Mathématiques Appliquées
	KUHN Gérard	Physique (IUT I)
	LUU DUC Cuong	Chimie organique
	MAYNARD Roger	Physique du solide
Mme	MINIER Colette	Physique (IUT I)
MM.	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et Minéralogie
	PFISTER Jean-Claude	Physique du solide
Mle	PIERY Yvette	Physiologie animale

MM.	RAYNAUD Hervé	M.I.A.G.
	REBECQ Jacques	Biologie (CUS)
	REYMOND Jean-Charles	Chirurgie générale
	RICHARD Lucien	Biologie végétale
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROBERT André	Chimie papetière
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
MM.	STIEGLITZ Paul	Anesthésiologie
	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques Appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	ARMAND Yves	Chimie (IUT I)
	BACHELOT Yvan	Endocrinologie
	BARGE Michel	Neuro chirurgie
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamie
MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BRODEAU François	Mathématiques (IUT B) (Personne étrangère habilitée à être directeur de thèse
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie papetière
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
	CORDONNIER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	CYROT Michel	Physique du solide
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie végétale
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	FAURE Gilbert	Urologie
	GAUTIER Robert	Chirurgie générale
	GIDON Maurice	Géologie
	GROS Yves	Physique (IUT I)
	GUIGNIER Michel	Thérapeutique
	GUITTON Jacques	Chimie
	HICTER Pierre	Chimie
	JALBERT Pierre	Histologie
	JUNIEN-LAVILLAVROY Claude	O.R.L.
	KOLODIE Lucien	Hématologie
	LE NOC Pierre	Bactériologie-virologie
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et médecine préventive
	MALLION Jean-Michel	Médecine du travail

MM.	MARECHAL Jean	Mécanique (IUT I)
	MARTIN-BOUYER Michel	Chimie (CUS)
	MICHOULIER Jean	Physique (IUT I)
	NEGRE Robert	Mécanique (IUT I)
	NEMOZ Alain	Thermodynamique
	NOUGARET Marcel	Automatique (IUT I)
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B) (Personnalité étrangère habilitée à être directeur de thèse)
	PEFFEN René	Métallurgie (IUT I)
	PERRIER Guy	Géophysique-Glaciologie
	PHELIP Xavier	Rhumatologie
	RACHAIL Michel	Médecine Interne
	RACINET Claude	Gynécologie et Obstétrique
	RAMBAUD André	Hygiène et Hydrologie (Pharmacie)
	RAMBAUD Pierre	Pédiatrie
	RAPHAEL Bernard	Stomatologie
Mme	RENAUDET JAcqueline	Bactériologie (Pharmacie)
MM	ROBERT Jean-Bernard	Chimie Physique
	ROMIER Guy	Mathématiques (IUI B) (Personnalité étrangère habilité à être directeur de thèse)
	SCHAERER René	Cancérologie
	SHOM Jean-Claude	Chimie Générale
	STOEBNER Pierre	Anatomie Pathologie
	VROUSOS Constantin	Radiologie

MAITRESDE CONFERENCESASSOCIES

MM.	DEVINE Roderick	Spectro Physique
	HODGES Christopher	Transition de Phases

Fait à SAINT MARTIN D'HERES, NOVEMBRE 1976

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE — ANNEE 1976-1977

Président : M. Philippe TRAYNARD

Vice-Présidents : M. Pierre-Jean LAURENT
M. René PAUTHENET

PROFESSEURS TITULAIRES

MM.	BENOIT Jean	Radioélectricité
	BESSON Jean	Electrochimie
	BLOCH Daniel	Physique du solide
	BONNETAIN Lucien	Chimie minérale
	BONNIER Etienne	Electrochimie et Electrométallurgie
	BRISSONNEAU Pierre	Physique du solide
	BUYLE-BODIN Maurice	Electronique
	COUMES André	Radioélectricité
	DURAND Francis	Métallurgie
	FELICI Noël	Electrostastique
	FOULARD Claude	Automatique
	LESPINARD Georges	Mécanique
	MOREAU René	Mécanique
	PARIAUD Jean-Charles	Chimie-Physique
	PAUTHENET René	Physique du solide
	PERRET René	Servomécanismes
	POLOUJADOFF Michel	Electrotechnique
	VEILLON Gérard	Informatique fondamentale et appliquée

PROFESSEURS SANS CHAIRE

MM.	BLIMAN Samuel	Electronique
	BOUVARD Maurice	Génie Mécanique
	COHEN Joseph	Electrotechnique
	LACOUME Jean-Louis	Géophysique
*	LANCIA Roland	Electronique
	ROBERT François	Analyse numérique
*	ZADWORNY François	Electronique
*	ROBERT André	Chimie papetière

MAITRES DE CONFERENCES

MM.	ANCEAU François	Mathématiques Appliquées
	CHARTIER Germain	Electronique
	GUYOT Pierre	Chimie Minérale
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du solide
	LESIEUR Marcel	Mécanique
	MORET Roger	Electrotechnique Nucléaire
	PIAU Jean-Michel	Mécanique
	PIERRARD Jean-Marie	Mécanique
	SABONNADIERE Jean-Claude	Informatique Fondamentale et Appliquée
MMe.	SAUCIER Gabrièle	Informatique Fondamentale et Appliquée

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche).

M.	FRUCHART Robert	Directeur de Recherche
MM.	ANSARA Ibrahim	Maître de Recherche
	CARRE René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	LANDAU Ioan Doré	Maître de Recherche
	MATHIEU Jean-Claude	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

A ma femme,

A mes parents.

Je tiens à remercier,

Monsieur le Professeur L. BOLLIET, qui a bien voulu me faire l'honneur de présider le jury de cette thèse,

Monsieur le Professeur G. VEILLON, pour les conseils et les critiques qu'il m'a prodigués tout au long de cette étude,

Monsieur R.A. GUEDJ, pour l'attention qu'il a manifestée en acceptant de siéger à ce jury,

Monsieur M. LUCAS, avec qui j'ai travaillé en étroite collaboration, et qui, par ses judicieux conseils et sa compétence, a déterminé l'orientation de mes recherches et m'a apporté l'aide sans laquelle ce travail n'aurait pu aboutir,

Je tiens également à exprimer ma vive gratitude à tous ceux avec qui j'ai travaillé au sein du Laboratoire, en particulier Messieurs C. LAUGIER et F. MARTINEZ.

Je remercie spécialement Monsieur P. FONTANILLE, qui m'a apporté une aide précieuse pour la réalisation de ce travail.

Je n'oublierai pas d'adresser mes remerciements aux services de dactylographie et de tirage, pour le soin qu'ils ont apporté à la réalisation matérielle de cette thèse.

TABLE DES MATIERES

CHAPITRE I - INTRODUCTION
INDEPENDANCE D'UN LOGICIEL GRAPHIQUE PAR RAPPORT À SON CONTEXTE D'UTILISATION

I.1 Notion de logiciel graphique de base ---------- 1

I.1.1 - Données d'application et données graphiques ---------- 2
I.1.2 - Le logiciel de mise en forme ---------- 3
I.1.3 - Rôle du logiciel graphique de base ---------- 4

I.2 Notion de contexte d'un logiciel graphique ---------- 7

I.3 Indépendance par rapport à l'application ---------- 8

I.3.1 - Acquisition de l'indépendance ---------- 8
I.3.2 - Conséquences sur la définition du logiciel de base --------- 9

I.4 Indépendance par rapport à la console de visualisation ---------- 10

I.4.1 - La mémoire d'entretien ---------- 10
I.4.1.1. - Les différents types de mémorisation ---------- 11
I.4.1.2. - Conséquences sur le logiciel de base ---------- 11
I.4.2 - L'écran ---------- 12
I.4.2.1. - La dimension et la forme de l'écran ---------- 13
I.4.2.2. - Le système de coordonnées ---------- 14
I.4.2.3. - La couleur et la luminosité ---------- 14
I.4.2.4. - La quantité d'information affichable ---------- 15
I.4.2.5. - Le type de balayage ---------- 15
I.4.2.6. - L'effacement sélectif ---------- 15
I.4.3 - Le processeur graphique ---------- 16
I.4.3.1. - Description ---------- 16
I.4.3.2. - Conséquences pour le logiciel de base ---------- 17
I.4.4 - Les dispositifs de communication ---------- 18
I.4.4.1. - Les différents dispositifs de communication ---------- 18
I.4.4.2. - La mise en oeuvre du dialogue en termes de dispositifs 19
I.4.4.3. - La mise en oeuvre du dialogue en termes de "fonctions d'intéraction" ---------- 20
I.4.5 - Choix des primitives ---------- 21

I.5 Indépendance par rapport au langage hôte ---------- 21

I.5.1 - L'insertion ---------- 21
I.5.1.1. - L'intégration syntaxique au langage ---------- 22
I.5.1.2. - L'extension au langage ---------- 22
I.5.2 - Le choix des primitives ---------- 23
I.5.2.1. - Le choix des paramètres ---------- 23
I.5.2.2. - Les chaînes de caractères ---------- 24

I.6 La portabilité : indépendance par rapport au logiciel hôte --------- 25

I.6.1 - Indépendance par rapport au système d'exploitation --------- 25
I.6.1.1. - L'allocation mémoire ---------- 25
I.6.1.2. - La gestion des entrées/sorties ---------- 26
I.6.2 - Le langage de programmation du logiciel ---------- 26

I.7 La portabilité : indépendance par rapport au matériel hôte ------
I.7.1 - Le calculateur ----------
I.7.1.1. - Les mots ----------
I.7.1.2. - La taille de la mémoire ----------
I.7.2 - La connexion de la console ----------

I.8 Conception d'un logiciel de base indépendant ----------
I.8.1 - L'indépendance par rapport à la console ----------
I.8.1.1. - L'affichage ----------
I.8.1.2. - Le dialogue ----------
I.8.2 - La portabilité ----------

I.9 Synthèse ----------

CHAPITRE II

ETUDE DE QUELQUES LOGICIELS GRAPHIQUES EXISTANTS

II.1 Le système "GIPSY" ---------- 3
II.1.1 - Organisation générale ---------- 3
II.1.2 - Les extensions graphiques ---------- 3
II.1.2.1. - Le "terminal général" ---------- 3
II.1.2.2. - Le terminal virtuel ---------- 4
II.1.3 - Le langage de spécification des interactions ---------- 4
II.1.4 - Critique ---------- 4

II.2 GINOF ---------- 4
II.2.1 - Fiche technique ---------- 4
II.2.2 - DINO : protocole réseau ---------- 4
II.2.3 - Critique ---------- 4

II.3 EUCLID ---------- 4
II.3.1 - Fiche technique ---------- 4
II.3.2 - Critique ---------- 4

II.4 Le macro-générateur STAGE2 ---------- 4
II.4.1 - Le logiciel graphique ---------- 4
II.4.2 - La portabilité : utilisation d'un macro-générateur ----- 4
II.4.3 - Critique ---------- 5

II.5 La première proposition du groupe de travail du réseau ARPA ----- 5
II.5.1 - L'affichage ---------- 5
II.5.2 - Le dialogue ---------- 5
II.5.3 - Critique ---------- 5

II.6 La seconde proposition ARPA ---------- 5
II.6.1 - Modèle conceptuel d'un système graphique ---------- 5
II.6.2 - Application à l'indépendance par rapport au terminal --- 5
II.6.3 - Le code intermédiaire ---------- 5
II.6.4 - Critique ---------- 5

II.7 Synthèse ----------

CHAPITRE III

CONCEPTION ET REALISATION D'UN LOGICIEL GRAPHIQUE BASE SUR LE CONCEI DE "CONSOLE VIRTUELLE" - LA PREMIERE GENERATION DE GRIGRI

III.1 Description d'une console virtuelle ---------- 62

III.1.1 - L'écran virtuel ---------- 63
III.1.2 - Le processeur graphique virtuel ---------- 63
III.1.3 - Les dispositifs de dialogue virtuels ---------- 64
III.1.4.1 - La table de description des figures ---------- 65
III.1.4.2 - La table des éléments graphiques ---------- 65
III.1.4.3 - La table des coordonnées ---------- 67
III.1.5 - Stockage des coordonnées ---------- 67
III.1.6 - Le coupage dans l'image virtuelle ---------- 67

III.2 Les primitives d'affichage de la première génération de GRIGRI --- 68

III.2.1 - Terminologie ---------- 69
III.2.2 - Description des primitives d'affichage ---------- 69
III.2.2.1 - Déclaration de figure ---------- 69
III.2.2.2 - Définition de clôture ---------- 70
III.2.2.3 - Génération de points et de segments ---------- 71
III.2.2.4 - Génération de texte ---------- 71
III.2.2.5 - Edition alphanumérique ---------- 72
III.2.2.6 - Effacement ---------- 72
III.2.3 - Gestion de l'image virtuelle ---------- 73
III.2.3.1 - Gestion de la table des figures ---------- 73
III.2.3.2 - Affichage de points et de segments ---------- 73
III.2.3.3 - Affichage de texte ---------- 75
III.2.3.4 - Gestion des zones réservées ---------- 75
III.2.3.5 - Effacement ---------- 77

III.3 Les primitives de dialogue de la première génération de GRIGRI --- 77

III.3.1 - Description des primitives de dialogue ---------- 78
III.3.1.1 - Introduction de valeurs ---------- 78
III.3.1.2 - Identification d'un objet ---------- 78
III.3.1.3 - Collecte de coordonnées ---------- 79
III.3.1.4 - Utilisation d'un menu ---------- 79
III.3.2 - Gestion de la console virtuelle ---------- 80
III.3.2.1 - Introduction de valeurs ---------- 80
III.3.2.2 - Désignation d'un objet ---------- 80
III.3.2.3 - Collecte de coordonnées ---------- 82
III.3.2.4 - Menu ---------- 83

III.4 Organisation d'ensemble du logiciel ---------- 85

III.4.1 - Les classes de modules ---------- 85
III.4.2 - Le code intermédiaire ---------- 86

III.5 Exemple d'interpréteur pour terminal à tube mémoire ----------
III.5.1 - Description du terminal Tektronix 4010 ----------
III.5.2 - Mise en page de l'écran ----------
III.5.3 - Réalisation et mise en oeuvre du dialogue ----------
III.5.3.1 - Entrée de valeurs ----------
III.5.3.2 - Identification ----------
III.5.3.3 - Menu ----------
III.5.3.4 - Collecte de coordonnées ----------
III.5.4 - Gestion de la mise à jour ----------
III.5.5 - Prise en charge des terminaux Tektronix 4014-4015 ------

III.6 Exemple d'interpréteur pour terminal à mémoire d'entretien -------
III.6.1 - Description du terminal IBM 2250 ----------
III.6.2 - Gestion de l'affichage ----------
III.6.3 - Mise en oeuvre du dialogue ----------
III.6.3.1 - Entrée de valeurs ----------
III.6.3.2 - Identification ----------
III.6.3.3 - Menu ----------
III.6.3.4 - Collecte de coordonnées ----------
III.6.4 - Gestion de la mise à jour ----------

III.7 Conclusion ----------
III.7.1 - Le choix des primitives du logiciel ----------
III.7.2 - Indépendance par rapport au matériel ----------

CHAPITRE IV

LA CONCEPTION DU LOGICIEL GRIGRI

IV.1 Les objectifs ----------

IV.2 Terminologie ----------
IV.2.1 - Les espaces ----------
IV.2.2 - Les entités graphiques ----------
IV.2.3 - Autres définitions ----------

IV.3 La définition des primitives graphiques ----------
IV.3.1 - Les concepts de base ----------
IV.3.2 - Les systèmes de coordonnées ----------
IV.3.2.1 - Définition de fenêtre ----------
IV.3.2.2 - Définition de clôture ----------
IV.3.2.3 - Relation entre fenêtre et clôture ----------
IV.3.3 - La déclaration des données ----------
IV.3.3.1 - Déclaration de points ----------
IV.3.3.2 - Déclaration de textes ----------
IV.3.4 - L'interprétation des données ----------
IV.3.4.1 - La primitive ----------
IV.3.4.2 - Syntaxe du descripteur de mode ----------
IV.3.5 - La gestion de l'affichage ----------

IV.4 Les messages à l'opérateur ---------- 118
IV.4.1 - Le concept de base ---------- 118
IV.4.2 - La description d'un message ---------- 118
IV.4.2.1 - Syntaxe du descripteur de message ---------- 119
IV.4.2.2 - La position du message ---------- 120
IV.4.3 - L'édition de données alphanumériques ---------- 120
IV.4.4 - L'effacement d'un message ---------- 121
IV.5 La définition des primitives de dialogue ---------- 121
IV.5.1 - Les concepts de base ---------- 121
IV.5.2 - Introduction de données alphanumériques ---------- 122
IV.5.3 - Collecte de coordonnées ---------- 123
IV.5.4 - Identification ---------- 124
IV.5.5 - Menu ---------- 124
IV.5.5.1 - La primitive ---------- 124
IV.5.5.2 - Syntaxe du descripteur de menu ---------- 124
IV.5.6 - Programmation et opération des primitives de dialogue - 125
IV.5.7 - Autre méthode de gestion du dialogue ---------- 126
IV.6 Conclusion ---------- 128

CHAPITRE V

LA REALISATION DU LOGICIEL GRIGRI

V.1 Le concept de logiciel adaptatif ---------- 129
V.1.1 - Organisation d'un logiciel adaptatif ---------- 129
V.1.2 - Les deux types d'interpréteurs ---------- 130
V.2 Le noyau du logiciel ---------- 134
V.2.1 - La structure de donnée du noyau ---------- 134
V.2.1.1 - La table des figures ---------- 135
V.2.1.2 - La table des fenêtres ---------- 136
V.2.1.3 - La table des clôtures ---------- 136
V.2.1.4 - La table des modes déclarés ---------- 136
V.2.2 - La gestion des modes ---------- 138
V.2.2.1 - Gestion du mode pour une figure ---------- 138
V.2.2.2 - Gestion du mode pour l'ensemble des figures ---------- 139
V.2.3 - Le traitement de l'affichage ---------- 141
V.2.3.1 - La liste des sections à afficher ---------- 141
V.2.3.2 - Traitement des coordonnées ---------- 143
V.2.3.3 - Déroulement de la fonction d'affichage ---------- 143
V.2.4 - Le traitement de l'effacement ---------- 144
V.2.5 - Le traitement des messages ---------- 145
V.2.5.1 - La génération de la chaîne de caractères à afficher 145
V.2.5.2 - La réservation des zones ---------- 145
V.2.5.3 - L'effacement des messages ---------- 146
V.2.5.4 - L'édition alphanumérique ---------- 146

V.2.6 - Les fonctions de dialogue ---------- 14
V.2.6.1 - L'introduction de valeurs ---------- 14
V.2.6.2 - La désignation ---------- 14
V.2.6.3 - La collecte de coordonnées ---------- 14
V.2.6.4 - Le menu ---------- 14
V.2.7 - Le contrôle à l'exécution ---------- 15

V.3 Le code intermédiaire, indépendant du matériel ---------- 15

V.4 L'interpréteur : ses fonctions générales ---------- 15

V.4.1 - Les fonctions de l'interpréteur ---------- 15
V.4.2 - La structure de données ---------- 15
V.4.2.1 - La table des sections affichées ---------- 15
V.4.2.2 - La table des messages affichés ---------- 15
V.4.2.3 - La table des zones réservées ---------- 15
V.4.3 - La mise en page de l'écran ---------- 16
V.4.3.1 - Calcul de la clôture nulle ---------- 16
V.4.3.2 - Comparaison avec la mise en page sur console virtuelle 16
V.4.4 - La gestion des tables ---------- 16
V.4.5 - La gestion de la liste d'affichage ---------- 16
V.4.5.1 - Affichage ---------- 16
V.4.5.2 - Effacement ---------- 16
V.4.5.3 - Récupération de place ---------- 16
V.4.6 - Les sous-programmes de services communs à tous les interpréteurs ---------- 16

V.5 Indépendance du noyau par rapport à son contexte ---------- 17

CHAPITRE VI

ETUDE DES DEUX TYPES D'INTERPRETEURS

VI.1 Interpréteur pour terminal à mémoire d'entretien ---------- 17

VI.1.1 - Description du contexte ---------- 17
VI.1.1.1 - Le calculateur et le logiciel ---------- 17
VI.1.1.2 - Le terminal ---------- 17
VI.1.1.3 - Le logiciel élémentaire ---------- 17
VI.1.2 - La structure de données ---------- 17
VI.1.2.1 - La table des sections ---------- 17
VI.1.2.2 - La table des messages ---------- 17
VI.1.2.3 - La table des zones réservées ---------- 17
VI.1.3 - Les fonctions de gestion du dessin ---------- 17
VI.1.3.1 - Initialisation ---------- 17
VI.1.3.2 - Récupération des attributs d'une section ---------- 17
VI.1.3.3 - Affichage d'une section de points ---------- 17
VI.1.3.4 - Affichage d'une étiquette ---------- 17
VI.1.3.5 - Affichage d'une section de texte ---------- 17
VI.1.3.6 - Effacement d'une section ---------- 17
VI.1.3.7 - Traitement d'un message ---------- 17

VI.1.4 - Les fonctions de dialogue ---------- 180
VI.1.4.1 - Introduction de valeurs ---------- 180
VI.1.4.2 - Identification ---------- 181
VI.1.4.3 - Collecte de coordonnées ---------- 182
VI.1.4.4 - Menu ---------- 182
VI.1.5 - Conclusion ---------- 183

VI.2 Interpréteur pour terminal à tube mémoire ---------- 183

VI.2.1 - Organisation générale de l'interpréteur ---------- 183
VI.2.2 - Description de la liste d'affichage ---------- 184
VI.2.2.1 - Les commandes graphiques ---------- 184
VI.2.2.2 - Choix relatifs à la liste d'affichage ---------- 185
VI.2.3 - Le processeur graphique ---------- 187
VI.2.3.1 - Interface avec le logiciel élémentaire ---------- 187
VI.2.3.2 - Exécution des commandes de la liste d'affichage --- 187
VI.2.4 - La structure de données ---------- 191
VI.2.4.1 - La table des sections affichées ---------- 191
VI.2.4.2 - La table des messages ---------- 191
VI.2.4.3 - La table des zones réservées ---------- 192
VI.2.5 - Gestion de la liste d'affichage ---------- 192
VI.2.5.1 - Initialisation ---------- 192
VI.2.5.2 - Affichage d'une section de points ---------- 193
VI.2.5.3 - Affichage d'une section de texte ---------- 194
VI.2.5.4 - Effacement ---------- 194
VI.2.5.5 - Traitement d'un message ---------- 194
VI.2.5.6 - Edition alphanumérique ---------- 194
VI.2.6 - Gestion de l'écran ---------- 195
VI.2.6.1 - L'affichage ---------- 195
VI.2.6.2 - L'effacement ---------- 195
VI.2.7 - Les fonctions de dialogue ---------- 196
VI.2.7.1 - Introduction de valeurs ---------- 197
VI.2.7.2 - Identification ---------- 197
VI.2.7.3 - Menu ---------- 198
VI.2.7.4 - Collecte de coordonnées ---------- 198
VI.2.8 - Conclusion ---------- 198

VI.3 Copie pour table traçante ---------- 199

VI.3.1 - Le principe ---------- 199
VI.3.2 - Utilisation ---------- 200

VI.4 Proposition d'interpréteur pour terminal à balayage ligne par ligne 201

VI.4.1 - Organisation ---------- 201
VI.4.2 - Le processeur graphique logiciel ---------- 202
VI.4.2.1 - Tri de la liste d'affichage ---------- 202
VI.4.2.2 - Construction de l'image ---------- 203
VI.4.3 - Application à une imprimante ---------- 204
VI.4.4 - Application à une console alphanumérique ---------- 205

VI.5 Accompagnateur pour console alphanumérique ---------- 205

VI.5.1 - Les primitives de déclaration ---------- 206
VI.5.2 - Les primitives d'affichage ---------- 206
VI.5.3 - Les primitives de dialogue ---------- 207

CHAPITRE VII

DIRECTIVES POUR L'INSERTION DU LOGICIEL DANS UN CONTEXTE QUELCONQU

VII.1 Description technique du prototype du noyau ---------- 2(

VII.1.1 - Les modules utilisateurs ---------- 2(
VII.1.2 - La structure de données ---------- 2(
VII.1.3 - Les modules internes ---------- 2:

VII.2 La portabilité du noyau ---------- 21

VII.2.1 - Les sous-programmes de manipulation d'adresses ------- 21
VII.2.1.1 - Stockage de l'adresse d'une variable ---------- 21
VII.2.1.2 - Récupération de valeurs ---------- 21
VII.2.1.3 - Stockage de valeurs ---------- 21
VII.2.2 - Les sous-programmes de manipulation de caractères ---- 22
VII.2.2.1 - Insertion d'un caractère dans une chaîne, avec incrémentation de l'index ---------- 22
VII.2.2.2 - Récupération d'un caractère dans une chaîne avec incrémentation de l'index ---------- 22
VII.2.2.3 - Insertion d'un caractère dans une chaîne sans incrémentation de l'index ---------- 22
VII.2.2.4 - Récupération de sous-chaînes ---------- 22
VII.2.2.5 - Portabilité ---------- 22

VII.3 Description technique des interpréteurs ---------- 22

VII.3.1 - Les modules communs à tous les interpréteurs --------- 22
VII.3.1.1 - Les modules dépendants du terminal ---------- 22
VII.3.1.2 - Les modules indépendants du terminal ---------- 22
VII3.2 - Les modules spécifiques à un interpréteur pour terminal sans image structurée ---------- 22(
VII.3.2.1 - Le processeur graphique ---------- 22(
VII.3.2.2 - Le module de mise à jour ---------- 22(
VII.3.3 - La structure de données commune à tous les interpréteurs 22'
VII.3.4 - La structure de données spécifique ---------- 22'
VII.3.5 - Portabilité d'un interpréteur déjà existant ---------- 22(
VII.3.5.1 - Changement de calculateur ---------- 22(
VII.3.5.2 - Prise en charge d'un nouveau terminal ---------- 22(
VII.3.6 - Portabilité de l'accompagnateur ---------- 22(

VII.4 Optimisation ---------- 22(

VII.4.1 - La dimension des tables ---------- 23(
VII.4.2 - Le type des variables utilisées ---------- 23(
VII.4.3 - Suppression de certaines séquences de traitement ----- 231
VII.4.3 - Ecriture de certains modules en langage machine ------ 231

VII.5 La modularité ---------- 232

VII.5.1 - Le noyau ---------- 232
VII.5.2 - L'interpréteur ---------- 234

VII.6 Utilisation possible du prototype ---------- 236

CONCLUSION

I Etat de la réalisation actuelle ---------- 236

II Les développements possibles ---------- 237

III L'utilisation de GRIGRI dans un réseau ---------- 238

IV Indépendance portabilité ---------- 241

V La standardisation des logiciels graphiques ---------- 242

ANNEXE 1 Les primitives de GRIGRI ---------- 244

ANNEXE 2 Exemple de programme d'application ---------- 246

ANNEXE 3 Exemple de trace fournie par l'accompagnateur ---------- 249

ANNEXE 4 Exemples de possibilités de GRIGRI ---------- 253

BIBLIOGRAPHIE ---------- 244

PREMIÈRE PARTIE :

GÉNÉRALITÉS ET HISTORIQUE

CHAPITRE I - INTRODUCTION

INDÉPENDANCE D'UN LOGICIEL GRAPHIQUE PAR RAPPORT À SON CONTEXTE D'UTILISATION

L'apport des techniques graphiques aux procédés de conception assistée par ordinateur conduit les utilisateurs à exiger des logiciels graphiques permettant de réaliser des programmes d'application utilisant les services des consoles de visualisation de façon simple et efficace. De plus, ces utilisateurs souhaitentpouvoir modifier l'environnement logiciel et matériel de ces programmes sans avoir à les réécrire.

Cette demande conduit les concepteurs de logiciels graphiques à étudier soigneusement les relations de ces logiciels avec leur environnement d'exploitation, afin de proposer des mécanismes assurant une certaine indépendance vis-à-vis de celui-ci. Des critères de choix permettent alors de discerner ce qui peut être mis d'une manière générale à la disposition des programmeurs, tout en réservant un mécanisme d'extension pour les cas particuliers.

Nous nous proposons ici de définir le rôle d'un logiciel graphique général, et d'étudier certains problèmes concernant l'indépendance de celui-ci par rapport à son environnement [I14], [I15].

I-1 - NOTION DE LOGICIEL GRAPHIQUE DE BASE

Le domaine d'application qui nous intéresse essentiellement est celui de la conception assistée en mode dialogué. Un programme de CAO offre au concepteur un ensemble de techniques qui permettent de réaliser peu à peu le projet envisagé.

I.1.1. Données d'application et données graphiques

Le programme d'application traite des données structurées d'une manière spécifique à l'application. Cette structure de données est construite, consultée, mise à jour à partir d'un certain nombre d'opérations d'entrées/sorties. Parmi celles-ci, une partie s'adresse à l'opérateur, lui permettant de contrôler et d'infléchir la marche du programme. Pour des raisons de commodité et d'efficacité, il peut être décidé qu'un sous-ensemble (ou la totalité) des opérations d'entrées/sorties se fera à l'aide d'une console de visualisation. Ceci implique que les informations contenues dans la structure de donnée: subissent une certaine mise en forme, permettant d'en déduire un dessin. Par exemple, un programme de recherche de contraintes travaillera sur des graphes représentés par des matrices. Pour mieux suivre l'évolution des calculs, il peut être souhaitable d'avoir une représentation de ces graphes sur l'écran, sous une forme permettant de saisir d'un coup d'oeil les modifications successives. Un logiciel de mise en forme permettra de passer de la représentation matricielle à une représentation plus adaptée au dessin, comprenant par exemple :

- les coordonnées des sommets du graphe,
- les étiquettes à attacher à ces sommets,
- les indications à porter sur les branches.

Une fois ces éléments calculés, ils seront transmis au logiciel graphique, qui se chargera de les interpréter pour en donner une représentation visuelle.

De la même manière, l'opérateur peut désirer modifier le graphe représenté sur l'écran en ajoutant un sommet. Dans ce cas, il désignera une position sur l'écran ou la tablette. Les coordonnées du point choisi seront relevées par le logiciel graphique et transmises au logiciel de mise en forme. Ce nouveau sommet sera inclus dans la représentation matricielle, non par ses coordonnées, mais de manière cohérente avec la représentation choisie pour le calcul.

Cette approche permet de distinguer les données de l'application qui sont structurées d'une manière spécifique, et les données graphiques qui constituent une représentation particulière contenant les renseignements nécessaires à la production d'un dessin.

I.1.2. Le logiciel de mise en forme

Le logiciel de mise en forme a la charge de préparer les données graphiques, en consultant la structure de données de l'application; et inversement, lors d'une fonction d'entrée, il doit permettre de passer de la donnée purement graphique à la donnée attendue par le programme d'application.

Dans l'exemple précédent, si l'opérateur désire ajouter une liaison, ou supprimer un sommet dans son graphe, il désignera sur l'écran la branche ou le sommet incriminé. Le logiciel graphique transmettra alors des éléments d'identification qui, pris en charge par le logiciel de mise en forme, permettront de mettre à jour la (ou les) matrice concernée.

L'interface entre le programme d'application et la console graphique se compose donc de deux parties :

- un logiciel de mise en forme ("modelling system"), qui permet de passer des informations de la structure de données de l'application aux informations graphiques et vice-versa,
- un logiciel graphique de base ("core") qui permet d'afficher sur l'écran une représentation des informations qui lui sont transmises et de recueillir les données graphiques destinées au programme d'application via le logiciel de mise en forme.

Les relations entre les différents logiciels peuvent être schématisées de lafaçon indiquée dans la figure I.1.

Il est bien évident que le logiciel de mise en forme est fortement dépendant de l'application qu'il sert, puisqu'il a à connaître la nature et l'organisation des informations traitées. Comme, pour une installation donnée, il y a en général plusieurs applications utilisant les services des consoles de visualisation, il doit également y avoir plusieurs logiciels de mise en forme. Dans certains cas, il est possible de définir des logiciels formant un outil général de construction d'objets, adapté à une classe d'applications (construction géométrique par exemple).

Il n'est en tout cas pas possible de définir l'outil général et universel qui puisse servir directement d'interface entre un programme d'application quelconque et une console de visualisation. En effet, comment prendre en charge indifféremment, par exemple, la visualisation d'une surface connue par son équation et la visualisation d'un graphe décrit par sa matrice d'incidence Dans les deux cas, il est nécessaire de disposer d'un logiciel de mise en form qui préparera les données graphiques nécessaires à la construction du dessin. Ce logiciel de mise en forme peut être très complexe (élimination des partie cachées, etc...) il sera en général pris en charge par le concepteur de logiciel graphique.

I.1.3. Rôle du logiciel graphique de base

Les logiciels de mise en forme sont donc aussi nombreux que les appl cations, mais on peut essayer d'imposer une façon de présenter les données au logiciel graphique. Si cet interface est bien défini, il sera possible de ne réaliser qu'un seul logiciel graphique commun à toutes les applications.

De même, il est très courant que plusieurs types de matériel soient employés sur un même site, et les utilisateurs souhaient, dans la mesure du pos sible, ne pas modifier leurs programmes lorsqu'ils passent d'un matériel à un autre. Le logiciel graphique aura donc la responsabilité d'assurer l'interface entre les différentes applications et les différents matériels.

Nous rencontrons ici deux problèmes fondamentaux :
- indépendance du logiciel graphique par rapport au matériel,
- standardisation des logiciels graphiques, afin de définir un interface uniqu avec les divers systèmes de mise en forme. La figure I.2 schématise la place d'un logiciel graphique de base.

Le rôle d'un tel logiciel est, d'une part, de recevoir et de renvoyer des informations purement graphiques en dialogant avec le logiciel de mise en forme et, d'autre part, de gérer le terminal utilisé, en cherchant à être indépendant par rapport à ce dernier.

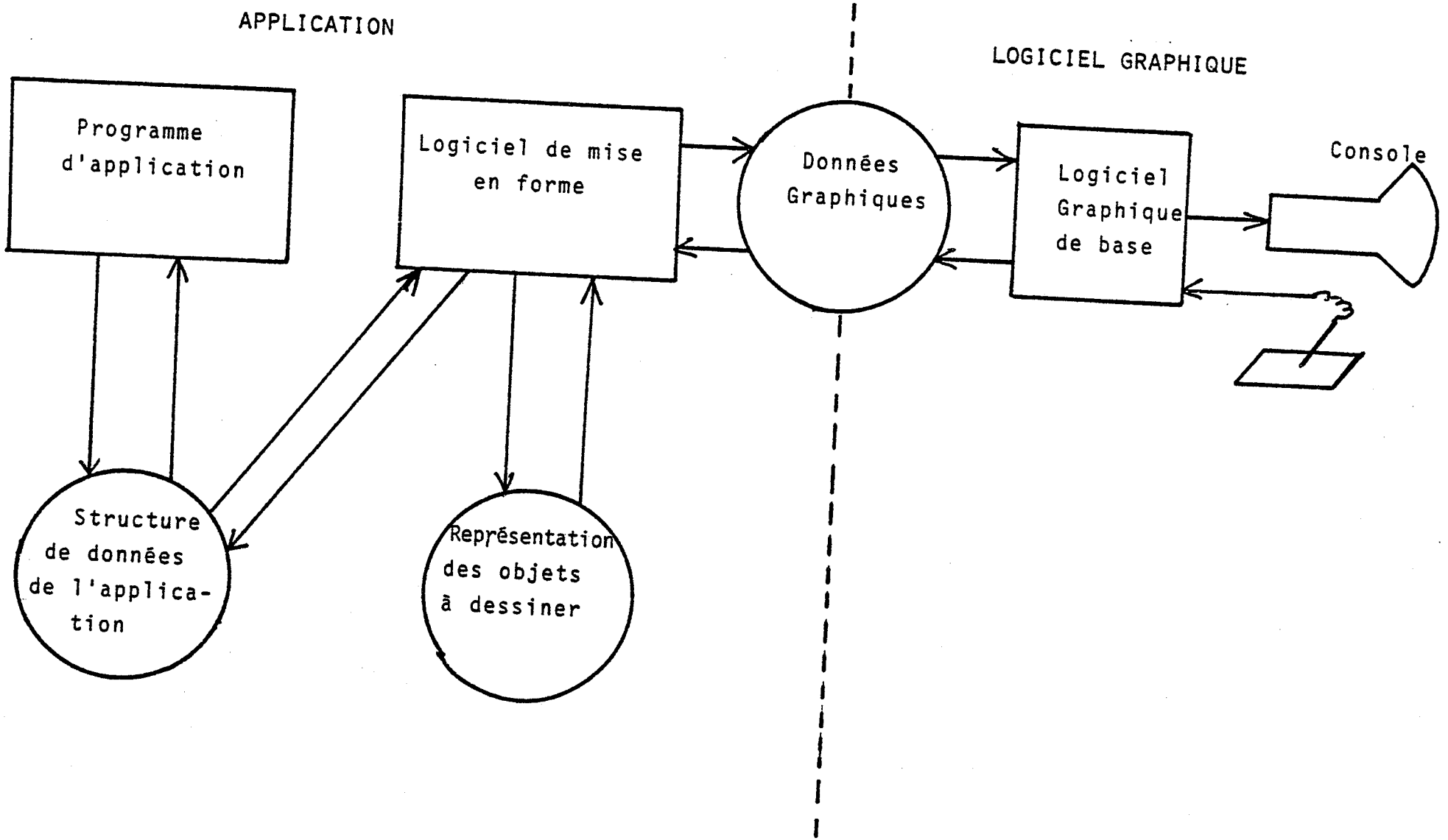


FIGURE I.1. : ORGANISATION D'ENSEMBLE DES LOGICIELS

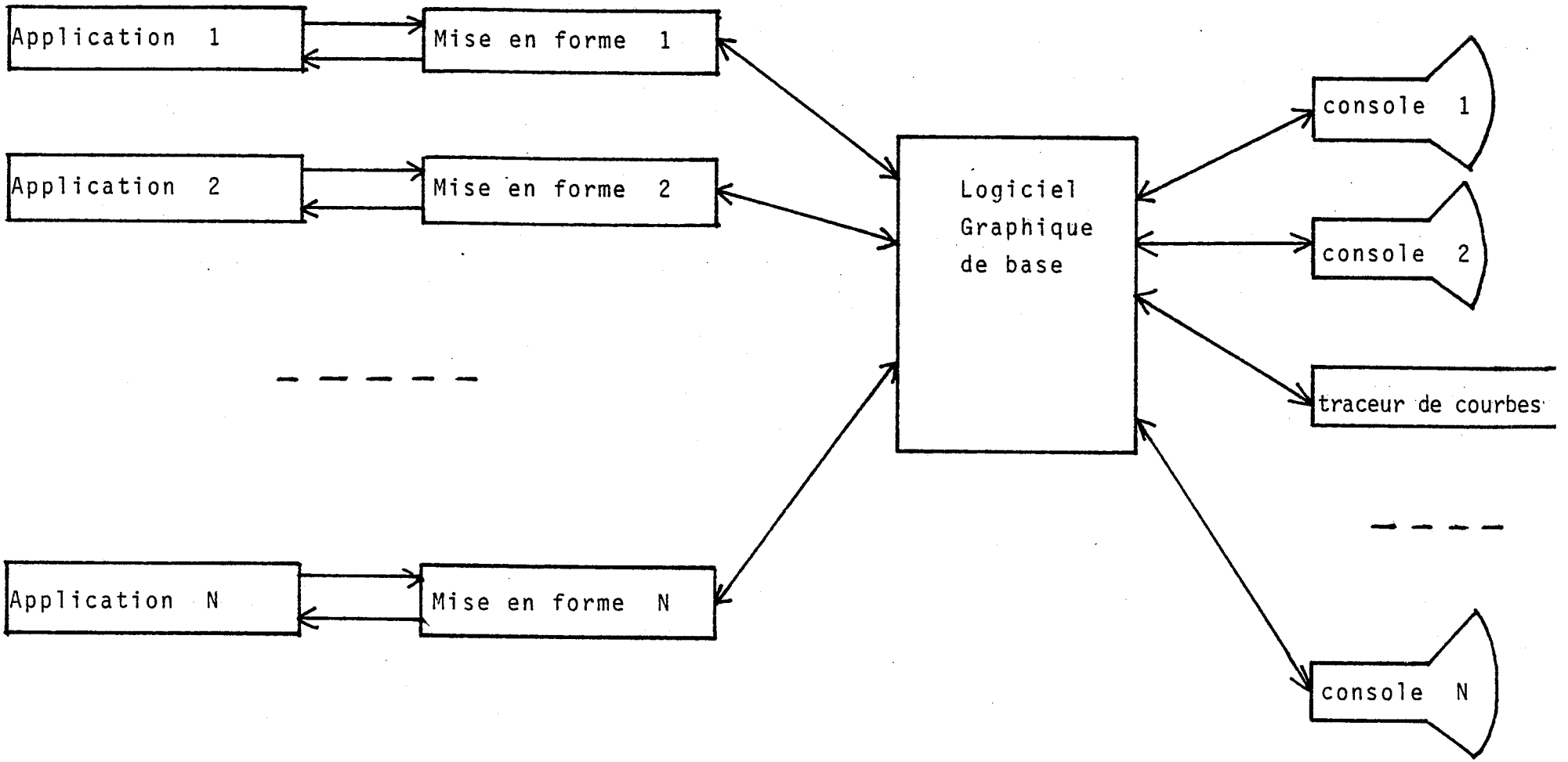


FIGURE I.2. : PLACE DU LOGICIEL GRAPHIQUE DE BASE

En fait, l'application et le terminal ne sont pas les seuls constituants de l'environnement d'un logiciel graphique de base car il faut tenir compte de l'ensemble du logiciel et du matériel qui permettent le déroulement d'une application graphique.

I.2. NOTION DE CONTEXTE D'UN LOGICIEL GRAPHIQUE

Le contexte d'un logiciel graphique est l'ensemble des entités (logicielles ou matérielles) qui jouent un rôle direct, ou indirect par rapport à celui-ci. Nous avons déjà évoqué l'application et le terminal utilisé.

L'insertion du logiciel graphique dans le programme d'application pose un nouveau problème. Le programme d'application est écrit dans un langage de programmation a priori quelconque et il n'est pas question de demander qu'il soit adapté au logiciel graphique. Il faut donc qu'il y ait compatibilité entre les deux types de programmes. Il s'agit clairement d'un problème de compilation qui peut dans certains cas être insoluble (exemple : impossibilité d'utiliser des programmes précompilés).

De plus, l'ensemble des services que nous venons d'évoquer n'est accessible que par le biais d'un système d'exploitation. Les services offerts par celui-ci auront une influence profonde sur l'utilisation d'une console de visualisation. Parmi les principaux facteurs de difficulté, citons :

- le mode d'exploitation (traitement par lots, partage de temps)
- la gestion des entrées/sorties (interruptions)
- le mécanisme d'allocation mémoire (mémoire virtuelle ou non, allocation dynamique),
- les possibilités de chargement dynamique (modification en cours d'exécution de la configuration du programme),
- les conventions de liaison (utilisation de bibliothèques de sous-programmes).

Notons enfin l'importance du calculateur et du type de liaison calculateur-terminal, considérés du point de vue matériel.

Le contexte d'utilisation d'un logiciel graphique de base est donc constitué des éléments suivants :

- l'application concernée (ou l'ensemble des applications concernées),
- le langage hôte (ou les langages hôtes)
- le système d'exploitation et l'ensemble du logiciel
- la console de visualisation,
- le matériel utilisé (ordinateur, liaison)

Cherchant à réaliser un logiciel graphique de base permettant d'exploiter un gramme d'application dans un environnement sujet à des modifications (langage de programmation différents, changement de matériel, transport sur un autre s nous sommes donc conduits à identifier d'indépendance par rapport aux différe constituants du contexte d'utilisation.

I-3- INDÉPENDANCE PAR RAPPORT À L'APPLICATION

Le logiciel graphique, se trouvant au service de diverses applicati doit permettre la mise en oeuvre de celles-ci sans accueillir dans sa définit des caractéristiques trop spécifiques à chaque application particulière.

I.3.1. Acquisition de l'indépendance

Il est évident que la clé de cette indépendance réside dans l'interf ce avec le logiciel de mise en forme. Définir cet interface revient à choisir qui sera rejeté au niveau du programmeur d'application, quitte à fournir des a des à la réalisation d'un logiciel de mise en forme particulier. Le rôle du lo ciel graphique sera donc limité à l'interprétation des données graphiques en v de fournir un dessin dans le plan, et, inversement, à fournir des informations de nature graphique à partir des dispositifs de communication.

Le logiciel graphique n'aura pas la responsabilité de la construction des objets à représenter, ni la responsabilité de la gestion de la structure de données contenant les représentations liées à l'application.

Cependant, le programmeur d'application a besoin de faire un lien entre les informations contenues dans sa structure de données et les informations graphiques qu'il transmet en vue de leur représentation. Ce lien est assuré par la possibilité de donner des noms (identificateurs) aux différents éléments du dessin, noms qui sont à la discrétion du programmeur, et qui lui permettent de les lier aux éléments de la structure de données. Le logiciel graphique a alors la charge de gérer l'ensemble des noms pour pouvoir les restituer à la demande.

En résumé, l'indépendance par rapport à l'application est obtenue en séparant strictement le logiciel de mise en forme et le logiciel graphique. L'interface entre ces deux logiciels est constitué d'informations de type graphique (coordonnées, chaînes de caractères, mode de tracé,...) et de listes de noms qui permettent de lier les informations de la structure de données à leur représentation sur l'écran. Il est clair que cette philosophie semble pénaliser les utilisateurs, puisqu'elle rejette sur eux des tâches qui, jusqu'à présent, étaient souvent réalisées par le logiciel graphique. En fait, cette prise en compte était toujours imparfaite, car habituellement ajustée à une ou deux applications. La mise en oeuvre de nouvelles applications revenait alors à enrichir le noyau du logiciel de base, et donc à pénaliser les applications précédentes (taille grandissante du logiciel, performances dégradées). C'estpourqoi nous avons choisi ce découpage qui permet d'ajuster les services par le biais du logiciel de mise en forme.

I.3.2. Conséquences sur la définition du logiciel de base

Cette approche impose un certain nombre de contraintes sur la définition des primitives du logiciel de base. Le logiciel contiendra trois types de primitives :

- primitives d'association de nom,
- pritimites de gestion du dessin,
- primitives de dialogue.

Le souci de généralité, nous conduit à n'offrir des primitives d'affichage qu'en 2 dimensions (la construction des objets en 3 dimensions est laissée à la charge du logiciel de mise en forme). De plus, des primitives telles que affichage de cercles, d'ellipses, sont trop spécifiques et ne seront pas non plus fournies au niveau du logiciel de base. En fait le logiciel de base permettra uniquement d'interpréter une suite de points, et de la visualiser avec un mode de représentation défini auparavant (ligne brisée, segments disjoints, points, trait continu ou pointillé, précision de la couleur, etc...).

On excluera évidemment du logiciel de base toutes les fonctions de transformation (translation, rotation, etc...).

On constate finalement que cette philosophie conduit à définir un ensemble de primitives de niveau relativement bas. Mais il faut veiller à ne pas tomber dans le piège de la dépendance vis-à-vis du matériel. En effet, plus le niveau est élevé, plus on risque d'être dépendant par rapport à l'application, mais inversement, plus le niveau est bas et plus on risque d'être dépendant du terminal.

I-4- INDÉPENDANCE PAR RAPPORT À LA CONSOLE DE VISUALISATION

En ce qui concerne la console de visualisation, les problèmes sont dus à l'extrême diversité des technologies employées, et à leurs performances parfois incompatibles. Nous passerons en revue les quatre composants majeurs d'une console de visualisation :

- la mémoire d'entretien,
- l'écran,
- le processeur graphique,
- les dispositifs de communication.

I.4.1. La mémoire d'entretien

Le problème de la mémoire d'entretien a une influence profonde sur la conception du logiciel.

Si l'on appelle "mémoire d'entretien", tout dispositif capable d'assurer la permanence de l'image sur l'écran, il est clair que toutes les consoles, de quelque type qu'elles soient, possèdent une mémoire d'entretien.

1.4.1.1. Les différents types de mémorisation

Les types de mémorisation peuvent être classés suivant deux critères :
- mémoire "vive" ou mémoire "morte",
- mémoire synthétique ou mémoire analytique.

Les terminaux à balayage cavalier et à balayage télévision, avec mémoire de rafraîchissement, disposent d'une mémoire "vive" dans laquelle on peut écrire à une adresse donnée, et dont on peut extraire des informations.

On trouve aussi des mémoires analogiques (vidéo-disques) et dans le cas des tubes à mémoire, la mémoire d'entretien est une mémoire "morte" : c'est la surface de l'écran lui-même.

A la limite, on peut considérer que le tracé sur le papier constitue une mémoire "morte" dans le cas des traceurs de courbes. De plus, les renseignements contenus dans ces différents types de mémoire d'entretien sont, soit synthétiques, soit analytiques. Une mémoire synthétique contient un modèle de l'image constitué de primitives graphiques de haut niveau (affichage de points, de segments, de caractères, etc...) et fournit un moyen de retrouver ces divers éléments dans la mémoire. Par contre, une mémoire analytique (mémoire point par point) contient uniquement des informations sur l'état de chaque point de l'écran (allumé ou éteint, intensité, couleur), mais ne fournit aucun renseignement sur la façon dont l'image a été construite.

1.4.1.2. Conséguences pour le logiciel de base

Le point crucial concerne la présence ou l'absence d'une description synthétique de l'image prise en charge par le matériel. En effet, quand on dispose d'une telle description, on peut regrouper les éléments graphiques entre

eux pour constituer des objets et on peut effacer certains éléments par écriture d'un saut dans la mémoire d'une adresse à une autre. Le problème de l'identification d'une partie de dessin est donc très simple dans les cas des mémoires d'entretien contenant une image structurée. De même, l'effacement d'une partie de dessin se réalise très aisément en engendrant un saut par dessus la zone de mémoire correspondant à cette partie de dessin (effacement sélectif

Par contre, si l'on dispose seulement d'une mémoire analytique, on ne peut en tirer aucun renseignement lors d'une désignation (si ce n'est les coordonnées du point désigné) et on ne peut pas effacer une partie de l'image (sauf en effaçant tout et en reconstruisant l'image).

Or les primitives du logiciel graphique de base doivent offrir les deux fonctions suivantes :

- effacement sélectif d'une partie du dessin,
- désignation et identification d'une partie du dessin.

Si l'on veut que le logiciel soit indépendant du matériel, on voit qu'il est nécessaire de simuler, en amont de la mémoire réelle, une mémoire d'entretien contenant une description structurée de l'image. Ainsi, les opérations d'identification et d'effacement sélectif pourront être exécutées au niveau de cette mémoire structurée.

I.4.2. L'Ecran

Les différences essentielles au niveau des écrans portent sur les caractéristiques suivantes :

- taille et forme,
- quantité d'informations affichables,
- système de coordonnées (précision)
- le nombre de luminosités disponibles, (ou le nombre de couleurs),
- le type de balayage employé,
- la possibilité d'effacement sélectif.

La caractéristique principale de ces différents paramètres est qu'il n'est pas possible de les simuler par logiciel. Ceci implique qu'un logiciel général devra permettre de profiter de toutes les qualités de l'écran le plus perfectionné, quitte à filtrer les demandes lorsque le programme d'application utilisera un écran moins évolué.

Nous allons examiner rapidement l'influence des différents paramètres sur la conception d'un logiciel graphique indépendant.

I.4.2.1. La dimension et la forme de l'écran

Lorsqu'il utilise un logiciel graphique indépendant du matériel, le programmeur d'application ne doit pas connaître les dimensions et la forme de l'écran. Il devra travailler dans ses propres repères, à charge ensuite au logiciel graphique d'adapter ces différents espaces à l'espace de l'écran réel. Il aura en fait l'impression de travailler sur un seul écran. A partir de cette idée, le logiciel peut s'organiser de deux façons différentes.

- La première approche consiste à confondre cet écran avec l'espace de l'utilisateur. L'utilisateur travaille uniquement dans son propre repère, le logiciel utilisera au mieux le dispositif physique pour réaliser l'affichage. Cette solution a été adoptée pour la seconde génération de GRIGRI.

- La seconde approche consiste à figer une définition d'un "écran virtuel". L'utilisateur fournira alors des coordonnées relatives à cet écran virtuel. Mais on rencontre ici un problème au sujet du choix de la forme et de la dimension de cet écran. En effet, certains terminaux ont des écrans carrés, d'autres ont des écrans rectangulaires. Si on définit un écran virtuel carré, comment l'adapter à un écran rectangulaire ? Pour ne pas perdre une partie de l'image, il est nécessaire d'afficher uniquement sur le plus grand carré contenu dans l'écran réel; mais on perd alors une partie de l'écran. Enfin, la dimension de l'écran virtuel étant précisée, on ne pourra pas exploiter la dimension "infinie" lors de la prise en charge d'une table traçante.

I.4.2.2. Le système de coordonnées (précision de l'écran)

Les systèmes de coordonnées des divers matériels sont en général fonction de la précision offerte. Celle-ci est très variable : 0,3 mm pour les tubes à balayage cavalier, 0,01 mm pour certaines tables traçantes.

Pour le logiciel graphique, le problème se pose en ces termes : avec quelle précision faudra-t-il transmettre les coordonnées, pour utiliser la précision maximale du matériel, sans envoyer des coordonnées trop précises.

Sur ce point, on retrouve l'alternative précédente : on peut se définir un "système de coordonnées virtuel" relatif à un écran virtuel, ou encore on peut ne rien préciser. Si on se définit a priori un système virtuel, la "précision virtuelle" risque d'être insuffisante dans certains cas et inutile dans d'autres. Sinon, la précision sera celle fournie par le programme d'application.

1.4.2.3. La couleur et la luminosité

Les possibilités offertes par le matériel sont ici très diverses :

- monochrome avec une seule luminosité,
- monochrome avec luminosité variable,
- multichrome sur des traits,
- multichrome sur des surfaces.

Au niveau du logiciel, le problème concerne le choix des options offertes. Là encore, on retrouve les deux philosophies :

- on peut offrir un maximum d'options aux utilisateurs; le logiciel s'adapter au matériel et négligera les options qu'il ne peut prendre en charge,
- on peut définir a priori les possibilités de l'écran virtuel. Mais le choix de ces "possibilités virtuelles" est très délicat : si le niveau est faible on sous exploitera les matériels performants, si le niveau est élevé, on ne pourra pas prendre en charge les terminaux peu évolués.

1.4.2.4. La quantité d'information affichable

La quantité d'information affichable est liée à la nature de l'écran et au type de mémorisation. Les mémoires analytiques permettent une utilisation maximale de l'écran ; par contre les terminaux à mémoire d'entretien sont limités par la taille de celle-ci.

Cette question est insoluble au niveau du logiciel, car il est clair que l'on ne pourra pas transmettre plus d'informations que ne peut en contenir la mémoire d'entretien. Le logiciel pourra seulement vider la mémoire pour continuer à afficher, mais dans ce cas, les opérations de désignation deviennent impossibles.

1.4.2.5. Le type de balayage

Les terminaux peuvent être classés en deux catégories, en fonction de leur type de balayage :

- balayage cavalier (terminaux à mémoire d'entretien, tube mémoire, table traçante),
- balayage ligne par ligne (terminaux à tube télévision).

Ce paramètre joue un rôle important par rapport aux primitives offertes par le logiciel. En effet, si le logiciel permet de construire des dessins composés uniquement de traits, les terminaux à balayage cavalier seront faciles à prendre en charge; par contre, pour les tubes à balayage ligne par ligne, le logiciel devra pouvoir construire une image analytique, à partir d'une image synthétique. Par contre, si le logiciel offre la possibilité de traiter des surfaces, les tubes à balayage ligne par ligne seront bien adaptés; mais il faudra simuler le remplissage sur les terminaux à balayage cavalier (par exemple par tracé de lignes horizontales).

1.4.2.6. L'effacement sélectif

Il est fonction du type de mémorisation. Afin de le rendre possible sur tous les matériels, le logiciel devra simuler une mémoire d'entretien contenant une description structurée de l'image. Mais il y a risque de duplication de l'information pour les terminaux à mémoire d'entretien.

1.4.3. Le processeur graphique

1.4.3.1. Description

Les processeurs graphiques assurent des fonctions extrêmement diverses et variées. Les instructions qu'ils exécutent se divisent en instructions graphiques et instructions non graphiques.

Les instructions graphiques composent un répertoire plus ou moins riche d'ordres de tracé élémentaires tels que :

- points,
- traits,
- caractères,
- coniques,
- etc...

Ces ordres élémentaires peuvent être précisés par des attributs :

- taille,
- couleur,
- texture,
- etc...

La plupart de ces ordres peuvent être simulés par logiciel lorsqu'ils font défaut sur une installation donnée.

Les instructions non graphiques sont en général liées à la présence d'une mémoire d'entretien structurée. On peut les classer en trois groupes, fournis de façon très inégale suivant le matériel, chaque constructeur ayant choisi un sous-ensemble sans qu'une ébauche de standard ne se dessine.

Le premier groupe d'instructions permet de structurer la liste d'affichage, par le biais d'instructions de rupture de séquence et d'appels de sous programmes. Si la première possibilité est très intéressante, la seconde s'est révélée être d'un emploi extrêmement dangereux dans le cadre de logiciels généraux et n'est plus proposée à leur niveau.

Le deuxième groupe d'instructions se trouve dans les consoles à hautes performances et concerne l'ensemble des transformations géométriques. De nouveau, la proposition de séparer le logiciel de mise en forme du logiciel graphique conduit à ne pouvoir employer ces instructions dans le cadre d'un logiciel général.

Par contre, le dernier groupe d'instructions concerne la gestion de la mémoire d'entretien. Dans ce cas, il est bien évident que le logiciel pourra pallier l'absence de telles instructions.

On voit donc qu'au niveau des processeurs graphiques, la définition de logiciels de base généraux conduit à l'abandon des possibilités extrêmement sophistiquées. Cependant, le nombre de consoles présentant ces caractéristiques est relativement faible. Nous noterons quand même que l'utilisation des microprocesseurs et de la microprogrammation pourrait conduire rapidement à une évolution importante dans ce domaine.

1.4.3.2. Conséquences pour le logiciel de base

L'indépendance par rapport au processeur graphique est acquise en rejetant toutes les fonctions de transformations, de représentation en trois dimensions, etc..., au niveau du logiciel de mise en forme. Le logiciel ne contiendra donc pas de telles primitives.

En ce qui concerne l'affichage en deux dimensions, le logiciel devra offrir toutes les possibilités des matériels les plus performants. Les options manquantes (par exemple tracé en pointillé, ou caractères de taille variable) seront simulées par le logiciel sur les terminaux peu évolués. On retrouve ici les deux philosophies précédentes : on peut se fixer une liste des possibilités d'un "processeur graphique virtuel", ou non. Dans le premier cas, on risque de sous exploiter certains terminaux en simulant des fonctions exécutables par le matériel.

Terminons l'étude des problèmes liés au matériel par l'évocation des dispositifs de communication.

1.4.4. Les dispositifs de communication

La diversité d'emploi et de fonctionnement des dispositifs de commu cation est très grande d'un matériel à l'autre.

1.4.4.1. Les différents dispositifs de communication

Les dispositifs de dialogue peuvent être regroupés en quatre classes suivant la technologie utilisée :

-- Les dispositifs à touches (clavier alphanumérique, clavier de fonctions).

Leur fonctionnement, du point de vue du programmeur se résume aux a: pects suivants :

- envoi d'une interruption,
- émission d'un code identifiant le dispositif utilisé et la touche enfoncée.

Les fonctions de base associées au clavier alphanumérique sont au nombre de trois :

- mise en place du curseur,
- retrait du curseur (et desactivation du clavier)
- envoi d'une ligne de texte au calculateur.

Ces instructions correspondent à des actions différentes suivant le type de matériel (tube mémoire ou tube à mémoire d'entretien). Si le clavier alphanu· mérique n'est pas associé à la console, il faudra simuler les fonctions qui p mettent l'affichage du texte sur l'écran.

-- Le photostyle provoque une interruption lorsque sa cellule photoélectriqu perçoit de la lumière. Il renvoie les renseignements suivants :

- coordonnées du faisceau au moment de l'interruption,
- adresse de l'instruction de la liste de visualisation qui était e cours d'exécution,
- si on a montré un caractère : code de celui-ci,
- dans les consoles très évoluées : renseignements concernant l'ent: té graphique à laquelle appartient l'élément montré.

-- Les tablettes permettent de collecter des coordonnées, de façon continue, ou point par point.

Le réticule est un dispositif analogue aux tablettes.

-- Les dispositifs analogiques mesurent des variations de déplacement (direction du déplacement, vitesse).

Citons également les potentiomètres.

Les dispositifs de dialogue sont très divers, mais notons que l'on peut en général simuler un dispositif à l'aide d'un autre.

1.4.4.2. La mise en oeuvre du dialogue en termes de dispositifs

C'est l'approche la plus classique. Les logiciels actuels proposent en général des primitives adaptées à chaque dispositif, primitives fournissant des renseignements de bas niveau qui sont ensuite décodés et exploités par le programme d'application. Il est clair qu'une telle approche ne peut permettre une indépendance par rapport au matériel. La seule solution serait de simuler à l'aide des dispositifs existant réellement, les fonctions des dispositifs manquants; on pourrait par exemple simuler les touches de fonctions sur le clavier alphanumérique. Dans cette approche, c'est à l'utilisateur qu'incombe le choix du dispositif à activer pour exécuter telle ou telle action. Ceci conduira nécessairement à une programmation en fonction du matériel, ou à un mauvais emploi de celui-ci. Ainsi, considérons une console sans touches de fonction, mais disposant d'un clavier alphanumérique et d'un photostyle. Pour exécuter une action du type "menu", le programmeur activera certainement le photostyle (raisonnement lié au matériel); si, par contre, il ne tient pas compte du matériel, il demandera peut être l'utilisation des touches de fonctions, celles-ci seront certainement simulées par le logiciel sur le clavier alphanumérique, ... et le photostyle, particulièrement bien adapté restera utilisé.

Le raisonnement en termes de dispositifs a donc plusieurs inconvénients

- dépendance par rapport au matériel,
- simulation difficile des dispositifs manquants,
- utilisation peu rationnelle du matériel,
- programmation difficile.

1.4.4.3. La mise en oeuvre du dialogue en termes de "fonctions d'interaction"

La méthode employée consiste alors à définir des dispositi logiques, dont le niveau correspond à une fonction de dialogue. L'expérience prouvé que les quatre types de fonctions d'interaction utilisés par les appl cations sont les suivants :

- désignation d'une partie de dessin,
- utilisation d'un menu,
- collecte de coordonnées,
- introduction de valeurs.

Le logiciel graphique se charge de réaliser chaque fonction à l'aide des dis sitifs réels. Ceci peut être fait de deux façons :

- le logiciel graphique peut associer à chaque fonction d'interaction, le di sitif physique le mieux adapté,
- la seconde solution consiste à permettre à l'opérateur d'utiliser tous les dispositifs disponibles et de choisir, au moment du déroulement de l'exécu tion, la façon dont il va réaliser la fonction.

Cette approche a été adoptée dans le logiciel GRIGRI. Les avantages de cette méthode sont nombreux :

- programmation simplifiée,
- indépendance totale par rapport au matériel,
- exploitation maximale du matériel,
- libre choix des dispositifs par l'opérateur.

Le problème d'indépendance par rapport aux dispositifs de dialogue est donc résolu facilement en choisissant de traiter le dialogue en termes d fonctions d'interaction (dispositifs logiques) et non plus en termes de maté

Cependant cette dernière approche est, à l'heure actuelle, encore t controversée, la possibilité d'une réalisation efficace étant encore mise en doute !

1.4.5. Choix des primitives

Pour acquérir l'indépendance par rapport à la console de visualisation, nous sommes donc contraints de faire un choix parmi les diverses possibilités existantes, afin de définir la base du logiciel. Un premier critère consiste à sélectionner, parmi toutes les possibilités existantes, celles qui semblent le plus répandues. Cependant, ce critère ne suffit pas, car il conduit à aligner l'ensemble du matériel sur les consoles les moins évoluées. Un deuxième critère consiste alors à retenir les possibilités qui peuvent être facilement simulées et qui sont d'intérêt suffisamment général pour que leur introduction dans le logiciel de base ne pénalise pas la majorité des utilisateurs.

I-5- INDÉPENDANCE PAR RAPPORT AU LANGAGE HÔTE

Pour qu'un logiciel soit utilisé correctement, il faut qu'il soit bien compris par les programmeurs. Une conséquence est qu'un logiciel graphique, destiné avant tout à compléter les instructions d'un langage de programmation, doit être défini à un niveau équivalent à celui du langage hôte. En particulier, il est souhaitable que son insertion respecte la philosophie du langage employé. Il est clair qu'il y aura donc plusieurs types de logiciels graphiques associés à différentes familles de langages. En effet, l'approche des langages tels que ALGOL, FORTRAN, PL1, est très différente de celle des langages type APL ou BASIC Nous nous sommes intéressés aux langages du premier type, comme étant, à l'heure actuelle, les plus répandus.

I.5.1. L'insertion

L'insertion d'un logiciel dans un langage de programmation peut se faire de deux manières :

- intégration syntaxique à un langage,
- extension à un langage.

Etudions ces deux méthodes dans l'optique de l'indépendance du logiciel par rapport à son contexte.

1.5.1.1. L'intégration syntaxique au langage

C'est en fait l'extension d'un langage de programmation. Dans cette approche, le logiciel graphique permet la définition de "variable graphiques" qui peuvent subir un certain nombre d'opérations.

Pour inclure ces nouveaux éléments au langage, on a le choix entre deux solutions :

- modification du compilateur,
- prétraitement.

La première solution est très délicate à réaliser et conduit à un c pilateur spécifique, ce qui supprime toute indépendance. La seconde solutior consiste à écrire un préprocesseur qui analyse le programme et le traduit er programme assimilable par le compilateur du langage. Cette solution est plus cile à réaliser, mais a l'inconvénient de coûter plus cher en temps de calcu

L'avantage de l'intégration syntaxique est de fournir des programme plus clairs (suppression des longues listes d'appel à des sous-programmes), le coût en est très élevé. De plus, il faut autant de préprocesseurs que de gages utilisés.

1.5.1.2. L'extension au langage

C'est la technique la plus simple. Elle consiste à définir bibliothèque de sous-programmes que l'on rend accessibles par le moyen de dé rations externes.

On se trouve confronté ici à un problème de compilation. En effet, FORTRAN autorise par définition ce type de compilation séparée. Par contre, s'agit d'une extension à des langages de type ALGOL 60, ALGOL W, PASCAL, cet extension peut ne pas être disponible (cas des premiers compilateurs ALGOL W

et donc rendre impossible l'utilisation de cette technique. La forme de la programmation varie évidemment beaucoup d'un langage à l'autre. En Fortran, les sous-programmes seront invoqués par des "CALL". En ALGOL W, les procédures seront appelées directement par leurs noms, mais il faudra les déclarer sous forme de références externes au début du programme.

Une autre difficulté provient du fait que les conventions de liaisons ne sont pas toujours définies, ce qui fait que les codes générés par des compilateurs différents sont incompatibles. La solution est alors, soit de modifier le (les) compilateur(s) en cause, soit de passer par un prétraitement qui engendre le code approprié dans le langage désiré. De toute manière, il s'agit de solutions non satisfaisantes dans l'optique de la portabilité.

En conclusion, l'extension à un langage permet d'utiliser le logiciel à partir de plusieurs langages de programmation à deux conditions :

- autorisation des références externes,
- conventions de liaisons bien définies.

1.5.2. Le choix des primitives

Le choix des primitives implique non seulement le choix des noms (important du point de vue de la compréhension), mais également le choix du type des paramètres et de leur nombre. Les problèmes sont ici non négligeables et conduisent en fait à utiliser les possibilités les plus élémentaires au détriment des caractéristiques évoluées de chacun des langages.

1.5.2.1. Le choix des paramètres

La structure de données la plus évoluée que l'on puisse utiliser est le tableau à une dimension, dont la borne inférieure est égale à 1. En effet, c'est la seule structure qui soit compilée de manière identique dans les différents langages qui nous intéressent.

Pour traiter un tableau, il faut connaître son adresse, mais aussi son arrangement en mémoire. Par exemple, le stockage d'un tableau à deux dimensions se fait en général colonne par colonne en Fortran, et ligne par ligne en Algol. Dans ce cas, l'indépendance est fortement compromise, car il faudrait théoriquement ranger les tableaux en fonction du langage utilisé pour les passer ensuite au logiciel graphique et inversement pour les entrées. Un logiciel indépendant ne doit donc traiter que des tableaux à une dimension.

1.5.2.2. Les chaînes de caractères

La manipulation des chaînes de caractères pose des problèmes dûs aux compilations différentes.

Certains langages (ALGOL W par exemple) traitent les chaînes de caractères, d'autres non. En conséquence, si le logiciel offre des primitives qui manipulent des chaînes de caractères, il devra assurer une récupération correcte de celles-ci. Ainsi, un logiciel écrit en Fortran devra stocker les chaînes de caractères dans des suites d'entiers et en assurer la gestion.

On constate donc que l'on touche ici un problème clé, qui fait que l'indépendance vis-à-vis du langage est extrêmement difficile à réaliser. En tout état de cause, cette indépendance relative ne permet pas d'assurer un transport immédiat sur une autre installation car les compilations d'un même langage peuvent être différentes sur des machines diverses.

On voit que cette question de la compatibilité des langages de programmation est cruciale pour la réalisation d'un logiciel général, et que les choix qui seront faits auront une influence profonde sur la facilité d'utilisation de celui-ci.

L'indépendance par rapport aux langages fait intervenir la notion de portabilité par le biais de la compatibilité des compilations d'un même langage sur des sites différents.

I-6- LA PORTABILITÉ : INDÉPENDANCE PAR RAPPORT AU LOGICIEL HÔTE

Nous abordons ici un premier aspect de la portabilité : indépendance par rapport à l'ensemble du logiciel présent sur le calculateur hôte.

1.6.1. Indépendance par rapport au système d'exploitation

Il est bien évident que la liberté de manoeuvre est ici presque nulle, puisqu'il n'est pas question de réécrire un système d'exploitation pour le plier aux désirs du concepteur graphique. On ne pourra donc que souhaiter, dans certains cas, de meilleurs services, et attendre un changement du système d'exploitation ...

Deux points nous semblent cependant essentiels : celui de l'allocation de mémoire, et celui de la gestion des entrées/sorties. En ce qui concerne le premier point, il est clair qu'un logiciel graphique un tant soit peu évolué est constitué par un nombre d'instructions non négligeable et qu'il nécessite donc une certaine place en mémoire pour fonctionner correctement.

1.6.1.1. L'allocation mémoire

Les systèmes fournissant des mécanismes de mémoire virtuelle permettront une meilleure exploitation du logiciel, au cas où la mémoire réelle fasse défaut, par rapport à des systèmes ne permettant que d'employer des techniques de recouvrement. Dans ce dernier cas, le découpage du logiciel de base en modules relativement indépendants peut conduire à une multiplication d'appels de sous-programmes, générateurs de difficultés de mise au point et de temps perdu. Notons que les systèmes permettant une reconfiguration dynamique des programmes en cours d'exécution (chargement dynamique) permettraient de mettre au point des logiciels de base dont ne seraient utilisées que les parties nécessaires à l'exécution d'une application donnée. L'absence de telles possibilités conduit alors à construire un logiciel fondé sur un niveau de système d'exploi-

tation minimum. Par exemple, la place nécessaire à la simulation d'une mémoi d'entretien sera réservée a priori, et non demandée au fur et à mesure des besoins.

1.6.1.2. La gestion des interruptions

En ce qui concerne la gestion des entrées/sorties, le logi ciel graphique de base laissera la responsabilité du traitement des interrup tions au système d'exploitation.

Pour des raisons de facilité de portabilité, et de compatibilité av le déroulement des programmes actuels, les dispositifs de communication serc exploités en mode synchrone. Le système d'exploitation aura donc la charge d la récupération des interruptions et du stockage de celles-ci jusqu'à ce que la demande de traitement soit faite par le logiciel.

On voit donc qu'en fait, le logiciel graphique et le système d'expl tation vont être étroitement associés. La seule notion d'indépendance que l' puisse admettre est la non-utilisation de techniques trop spécifiques à un s tème donné, ou trop évoluées.

Le second aspect de la portabilité vis-à-vis du logiciel hôte conce la prise en charge du langage de programmation du logiciel par le système d' ploitation.

1.6.2. Le langage de programmation du logiciel

Pour tenter d'assurer la portabilité du logiciel, il est nécessaire de l'écrire dans un langage de programmation que l'on trouve dans tous les s tèmes d'exploitation.

Si l'on examine la situation, on s'aperçoit que les langages préser sur tous les sites sont très rares. Evidemment, on pense immédiatement à For Mais on peut aussi envisager des langages tels qu'ALGOL, PASCAL. En admettar qu'il existe un langage présent sur tous les sites, le problème n'est pas er résolu, car on se heurte souvent à l'incompatibilité des compilateurs d'un m langage sur des sites différents.

Le problème se pose en particulier en ce qui concerne les types des variables et leur encombrement en mémoire. Prenons le cas de Fortran. Le Fortran IBM autorise la manipulation des "LOGICAL*1" codés sur un octet et des "INTEGER*2" codés sur deux octets. Par contre le Fortran CII ne le permet pas. Pour résoudre ce problème, on adopte au maximum les options standards qui sont présentes dans tous les compilateurs. Ainsi, on n'utilise que des logiques, entiers et réels standards, dont l'encombrement en mémoire est choisi par le compilateur.

On rencontre quelquefois aussi des différences concernant l'initialisation des variables. Le logiciel devra donc systématiquement initialiser ses variables.

On constate finalement que, quelque soit le langage utilisé pour l'écrire, le logiciel sera difficilement portable directement, car il n'existe pas de langage de programmation parfaitement standardisé.

1-7- LA PORTABILITÉ : INDÉPENDANCE PAR RAPPORT AU MATÉRIEL HÔTE

Les trois composants du matériel hôte sont :

- le calculateur,
- la liaison calculateur-terminal,
- la console de visualisation.

Nous avons déjà examiné les problèmes concernant la console de visualisation, étudions maintenant les deux autres constituants.

1.7.1. Le calculateur

En ce qui concerne le calculateur, l'influence directe sur le logiciel graphique est relativement faible, car la plupart des ressources de la machine ne seront accessibles qu'à travers le système d'exploitation qui masquera les réalités. Notons cependant deux problèmes purement technologiques :

- la structure des mots de la mémoire,
- la taille réelle de la mémoire.

1.7.1.1. Les mots

La taille variable des mots pose un problème pour le traitement des chaînes de caractères. En effet, pour manipuler les chaînes de caractères, il est nécessaire de connaître le nombre de caractères contenus dans mot de la mémoire (pour pouvoir extraire ou insérer des caractères dans une chaîne) (ceci veut dire que dans un logiciel indépendant, les routines de manipulations de caractères devraient être paramétrées en fonction du nombre de caractères par mot; mais cette solution risque d'alourdir considérablement le logiciel.

1.7.1.2. La taille de la mémoire

Un logiciel indépendant devrait être supportable par n'importe quel calculateur, quelle qu'en soit la taille. Mais, si le système d'exploitation ne masque pas la taille réelle de la machine, on peut être conduit à mettre en place une version minimale, optimisée, du logiciel, sur certains petits calculateurs.

On se heurte ici à une contradiction : pour être portable, un logiciel doit utiliser au maximum les options standards, or celles-ci ne sont pas du tout optimisées.

1.7.2. La connexion de la console

Le mode de connexion de la console au calculateur est une source de problèmes non négligeable.

Notons tout d'abord que les jeux de caractères utilisés par les calculateurs et les terminaux ne sont pas toujours compatibles. Le logiciel devra donc assurer le transcodage nécessaire. Mais la principale question est celle de la vitesse de transmission des informations, facteur extrêmement influent les performances de la console, et qu'il est impossible de maîtriser par logiciel.

Cependant, le logiciel pourra adapter la gestion de l'écran à la vitesse de transmission (en particulier pour la mise à jour des terminaux à tube à mémoire).

Remarquons enfin que la gestion du dialogue sera différente selon que la console est connectée directement à un gros calculateur travaillant en temps partagé, ou à un mini calculateur satellite qui assurera le traitement du dialogue au niveau local.

I-8- CONCEPTION D'UN LOGICIEL DE BASE INDÉPENDANT

La recherche de l'indépendance du logiciel graphique de base par rapport à son contexte, conduit à imposer un certain nombre de contraintes lors de sa conception et de sa réalisation. Nous évoquerons tout d'abord les problèmes liés à l'obtention de l'indépendance par rapport à la console de visualisation, puis ceux liés à la portabilité.

1.8.1. L'indépendance par rapport à la console

Les appels du programme d'application au logiciel graphique peuvent être considérés comme formant un code (commandes et données) indépendant du matériel utilisé. Le logiciel graphique doit au contraire engendrer un code dépendant du terminal employé. Il apparaît donc que le logiciel peut être décompos en deux modules (foir figure I-3) :

- un module de communication avec le programme d'application,
- un module de gestion du terminal employé.

Nous étudierons tour à tour l'acquisition de l'indépendance pour l'affichage et pour le dialogue.

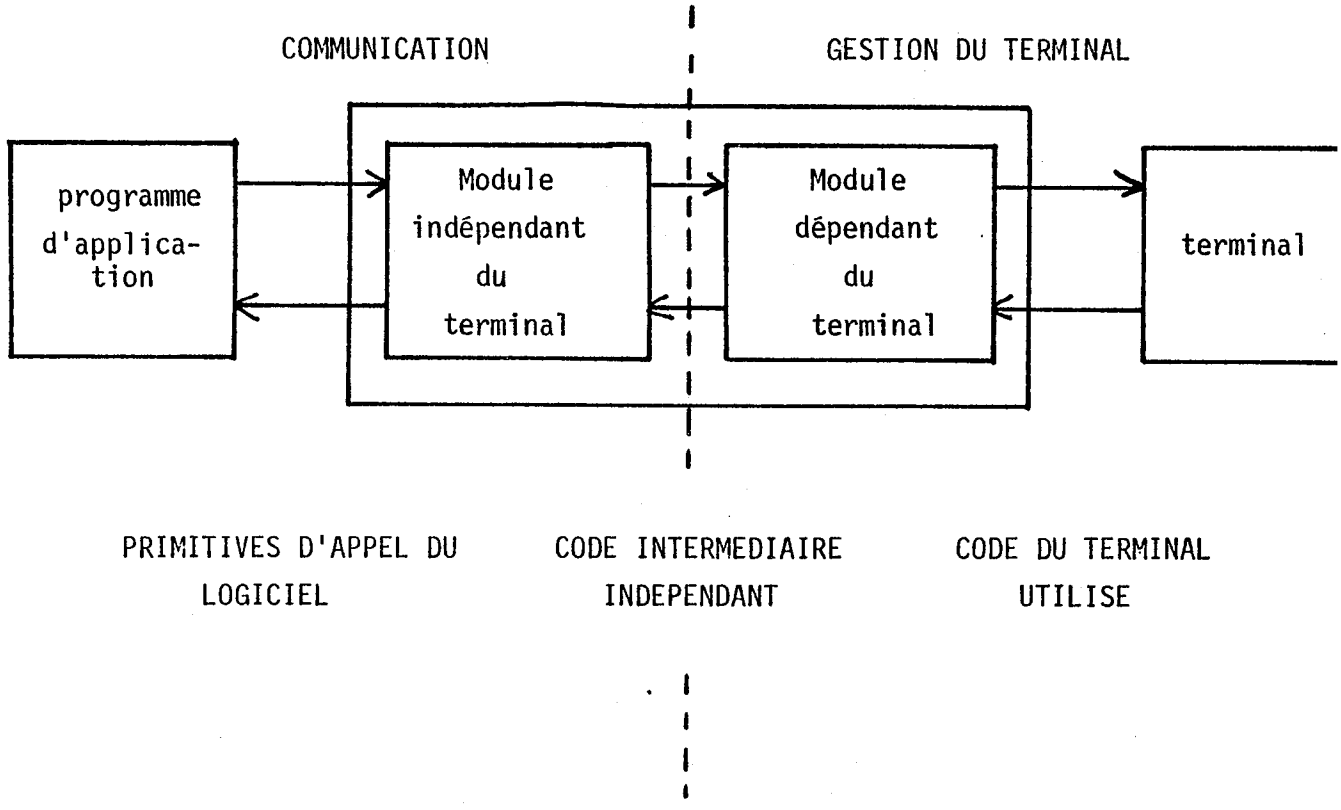


Figure I.4. : Organisation d'ensemble du logiciel

1.8.1.1. L'affichage

La réalisation des fonctions d'effacement sélectif (ainsi les problèmes d'identification dans le cas du dialogue) implique la nécessit de disposer d'une représentation structurée de l'image affichée sur l'écran, quel que soit le terminal utilisé. Dans le cas des matériels ne disposant pa d'une mémoire structurée, le logiciel graphique devra simuler celle-çi, de f çon a pouvoir disposer d'une liste d'affichage synthétique. L'attribution de la gestion de cette liste d'affichage peut se faire à l'un des deux modules logiciel graphique, conduisant à deux solutions qui sont très différentes pa leur philosophie.

-- Nous pouvons attribuer la gestion de la liste d'affichage au mod de communication, indépendant du terminal. La liste d'affichage est alors dé finie à l'aide d'un code intermédiaire indépendant de tout terminal, permett de constituer une image virtuelle qui sera ensuite convertie pour l'écran ré

Il y a donc définition d'une "console virtuelle", matériel idéal, qui, s'il existait, résoudrait tous les problèmes d'adaptabilité. La configuration du logiciel est décrite par la figure I.4. Cette approche rejoint un des choix de l'alternative que nous avons déjà rencontrée en plusieurs circonstances : on se définit a priori un certain nombre d'options (qui sont précisément les possibilités de la console virtuelle) et toutes ces options sont prises en charge au niveau de la console virtuelle. Le module de gestion du terminal n'est qu'un simple traducteur du code virtuel au code réel.

On peut aussi envisager de gérer la liste d'affichage au niveau du module dépendant du terminal (figure 1.5). Cette approche permet d'adapter la liste d'affichage au type du terminal. Ainsi, si l'on utilise un terminal à mémoire d'entretien, la liste d'affichage ne sera pas simulée par le logiciel. Celui-ci conservera uniquement des renseignements (adresses, longueurs) concernant les données rangées dans la mémoire d'entretien. Par contre, si l'on utilise un tube mémoire, la liste d'affichage sera à la charge de l'interpréteur et contiendra des commandes adaptées aux possibilités du matériel. Les données permettant de construire et de gérer l'image sont alors réparties entre les deux modules :

-le module de communication conserve les renseignements concernant l'application (noms, adresses des paramètres, etc...),

- l'interpréteur conserve les renseignements nécessaires à la gestion de l'affichage.

Cette seconde approche conduit à un logiciel de type "adaptatif".

Ces deux méthodes ont été expérimentées et seront décrites en détails ultérieurement.

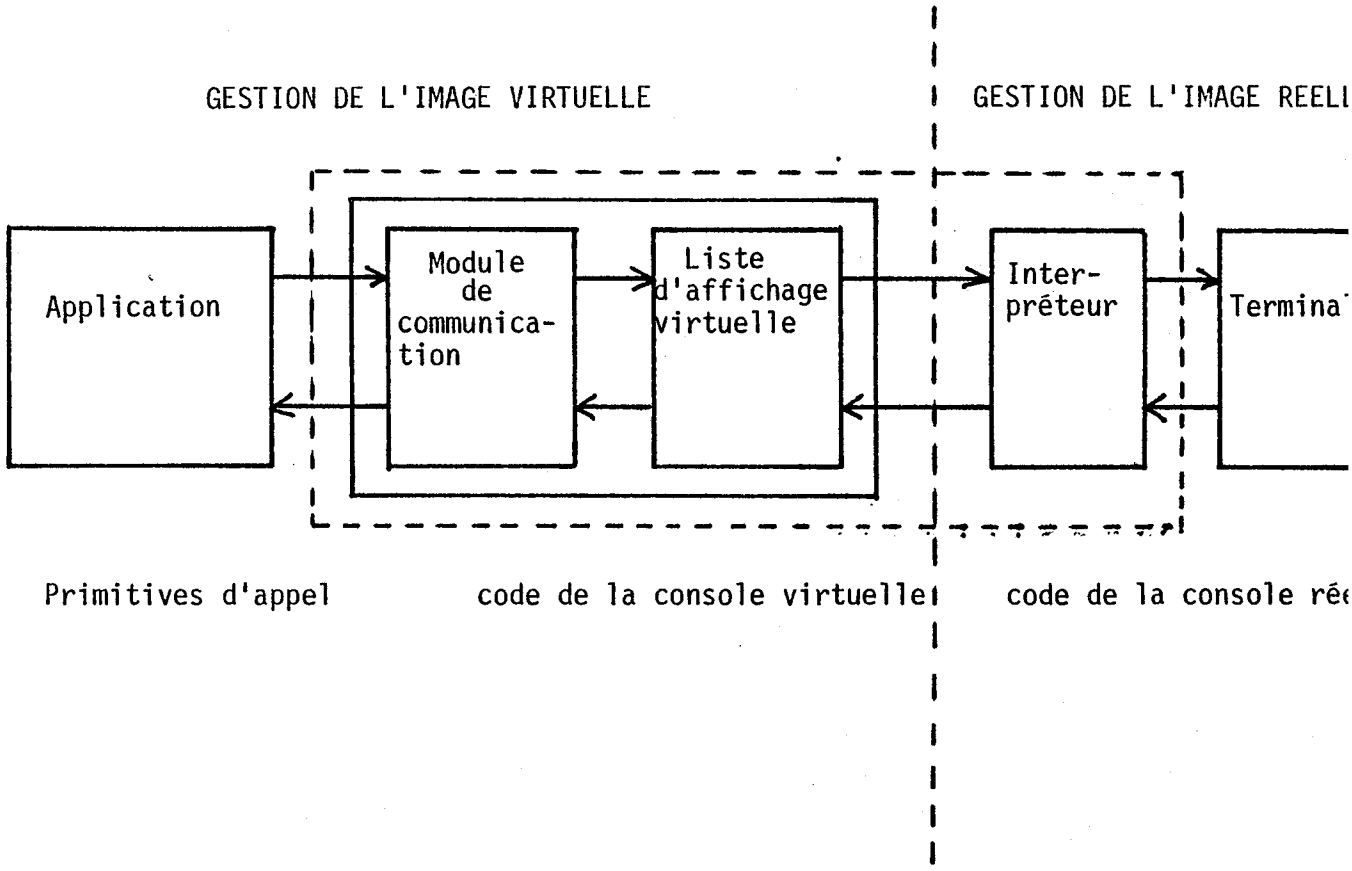


Figure I.5. : Console virtuelle

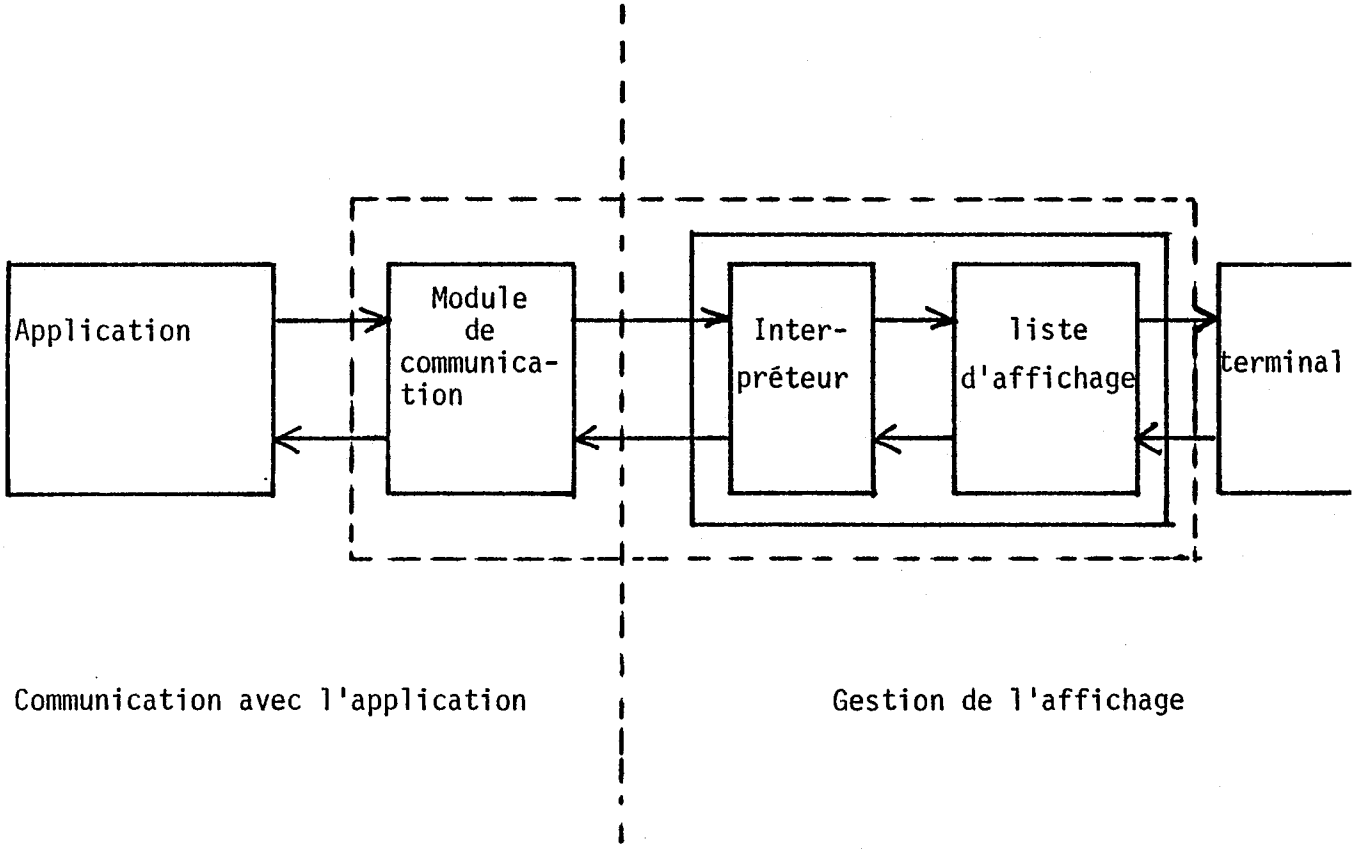


Figure I.6. Logiciel adaptatif

1.8.1.2. Le dialogue

Un raisonnement en termes de dispositifs logiques et la volonté d'offrir aux utilisateurs la possibilité d'utiliser au maximum le matériel a conduit à reporter la prise en charge du dialogue au niveau de l'interpréteur. Le schéma est alors le même que dans le cas de l'affichage. Le logiciel est constitué de deux modules (figure I.6) :

- un module de communication avec l'application, chargé de recueillir les demandes et de transmettre les réponses,
- un module de gestion des dispositifs physiques, chargé de réaliser les demandes

L'interpréteur a pour fonction :

- d'activer les divers dispositifs autorisés,
- de récupérer les résultats renvoyés par les dispositifs réels,
- d'assurer la mise en forme des résultats,
- de renvoyer les réponses en terme de dispositifs logiques.

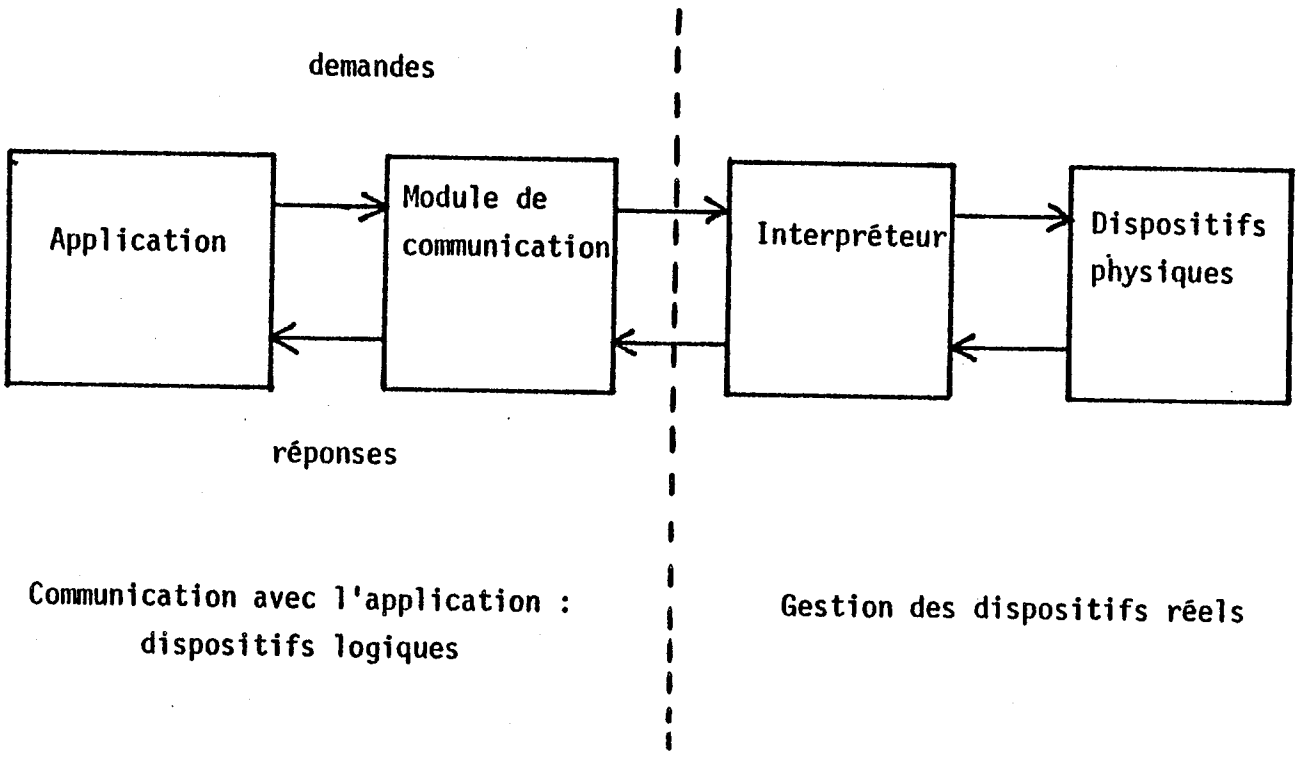


Figure I.7. : Le dialogue

Cette philosophie permet, avec un degré d'efficacité raisonnable :

- d'assurer l'indépendance des fonctions de dialogue, vis à vis du matériel,
- d'exploiter au maximum les possibilités offertes par le matériel,
- de laisser à l'opérateur, lors de l'exécution du programme d'application, l choix du dispositif adéquat.

Notons que les interpréteurs permettent de réaliser des logiciels de niveau plus ou moins élevé, sans que les concepts de base soient remis en cau Par exemple, la collecte de coordonnées peut se faire à travers un véritable teur de dessins, permettant une mise en page à l'aide de fonctions évoluées (recopie, transformations simples, aide au dessin, ...). Le point important e que cette réalisation se fait au niveau le plus bas, et n'affecte en rien les primitives de dialogue. L'évolution se fait au niveau de la réalisation, et n au niveau de la définition, ce qui permet de conserver la structure des progr mes d'application quel que soit le niveau des interpréteurs.

Pour terminer, disons que le seul point délicat est celui de la récu ration des informations en provenance des dispositifs réels. Il est clair que cette fonction est très dépendante du matériel et du système d'exploitation. pendant, elle est suffisamment réduite pour ne pas rendre nulle l'adaptabilit du logiciel graphique.

Cette réalisation du dialogue a été expérimentée dans les diverses v sions du logiciel GRIGRI.

1.8.2. La portabilité

Comme nous l'avons déjà indiqué, la portabilité totale est difficile ment réalisable. Cependant, on peut l'approcher en s'imposant certaines contr tes :

- utilisation d'un langage de programmation largement répandu,
- utilisation maximale des options standards,
- bon choix des paramètres

Ceci ne résoudra pas tous les problèmes, mais permettra de rendre le logiciel graphique facilement "adaptable".

I-9- SYNTHÈSE

Nous avons soulevé ici les principaux problèmes que l'on rencontre lors de la conception d'un logiciel graphique de base indépendant de son contexte.

Des solutions pour résoudre ces problèmes seront proposées dans les chapitres suivants.

Dans un premier temps, nous essaierons de situer quelques logiciels existants, en fonction des solutions adoptées. Ensuite nous décrirons la première génération de GRIGRI, logiciel développé à Grenoble. Cette première expérience nous a conduits à une deuxième génération du logiciel, beaucoup plus souple et permettant une meilleure utilisation des divers matériels. La description de la seconde génération de GRIGRI fera l'objet de la seconde partie de ce travail.

CHAPITRE II - ETUDE DE QUELQUES LOGICIELS GRAPHIQUES EXISTANTS

Nous ne prétendons pas décrire ici tous les logiciels existants. Nous voulons seulement étudier quelques logiciels opérationnels, sous l'angle de l'indépendance vis-à-vis du contexte.

De plus nous décrirons rapidement quelques propositions des groupes de travail qui se penchent actuellement sur les problèmes des graphiques dans les réseaux d'ordinateurs. Nous évoquerons successivement :

- GIPSY (IRIA/LABORIA)
- GINO (CAD Centre)
- STAGE2 (Université de Toulouse)
- EUCLID
- 2 propositions de protocole graphique dans le réseau ARPA.

II-1 - LE SYSTÈME GIPSY

II.1.1. Organisation générale

Le système graphique GIPSY [L2], [L3], [L5], [L6] repose sur la configuration suivante :

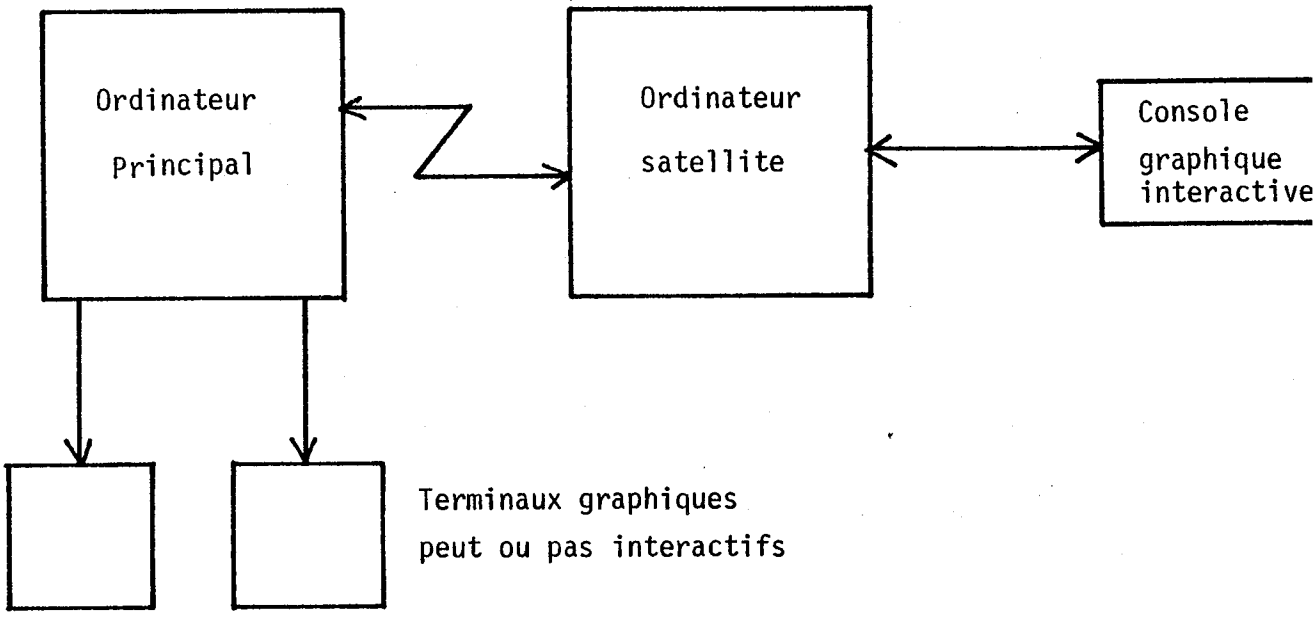


Figure II.1. : Organisation de GIPSY

GIPSY comporte essentiellement :

- pour l'ordinateur principal des extensions graphiques 3D de langages (Algol Fortran, Pascal),
- pour l'ordinateur satellite un langage spécialisé de haut niveau pour la description du dialogue,
- la possibilité depuis l'ordinateur principal, de mettre en oeuvre des fonctions d'interaction sur le satellite,
- inversement, la possibilité depuis le satellite de faire appel à des procédures algorithmiques sur l'ordinateur principal.

II.1.2. Les extensions graphiques

Elles ont pour but la description d'images en deux ou trois dimensions ; on y trouve aussi des concepts de gestion des images et des moyens d'entrée, pour des applications peu interactives. Ces extensions graphiques

sont caractérisées par l'appel graphique de procédure s'inspirant des "display procedures" de W. NEWMAN [G 10].

Ces extensions graphiques sont obtenues à l'aide d'un préprocesseur. Celui-ci interprète les instructions graphiques et leur substitue les appels de sous-programmes de bibliothèque correspondants.

Les sous-programmes de bibliothèque , ainsi que le préprocesseur sont écrits en Fortran.

II.1.2.1. Le "Terminal général"

C'est un ensemble de fonctions logiques qui permettent à l'utilisateur d'ignorer a priori le terminal sur lequel il travaille. Ces fonctions se divisent en 4 classes :

- description d'image,
- gestion des images,
- gestion de l'écran,
- gestion des moyens d'entrée.

- Les images sont décrites à l'aide de constituants élémentaires :

- points
- segments (caractéristique du tracé)
- déplacement faisceau éteint
- étiquettes de marquage
- faces (décrites par leurs pourtours)

Le système de coordonnées est choisi par l'utilisateur dans un espace à deux ou trois dimensions.

On peut regrouper un ensemble de constituants dans une procédure (décrite comme une subroutine Fortran).

Des transformations peuvent être précisées lors d'un appel graphique:

- rotation
- translation
- changement d'échelle
- transformation quelconque définie par une matrice
- perspective
- élimination des parties cachées
- etc...

On peut associer un nom à un élément, pour une désignation ultérieure.
On peut composer les transformations : l'objet "transformation" existe, il es décrit comme les procédures, sous forme d'une subroutine Fortran.

- Les images peuvent être stockées et gérées :

- OPEN début d'une image,
- CLOSE fin d'une image,
- KILL destruction d'une image.

- L'espace de visualisation est défini comme étant rectangulaire, s unité est le centimètre, et son origine le coin bas gauche. Avant de décrire image, l'utilisateur doit spécifier le terminal auquel elle est destinée (ins truction DEVICE). Pour un terminal sans possibilité de stockage, l'image est fichée dès sa description, dans le cas contraire elle est stockée et les prim tives DISPLAY et REMOVE permettent de l'afficher et de l'effacer.

- La gestion des moyens d'entrée se fait par l'intermédiaire de 7 moyens d'entrée logiques :

- booléen,
- entier,
- réel,
- chaîne de caractères,
- identificateur d'image,
- couple de coordonnées + booléen (crayon levé ou baissé),
- temps.

Une étiquette logique permet de différencier les divers accès possi à un moyen d'entrée logique. Le programme s'adresse toujours à un couple (moy d'entrée logique, étiquette logique). On suppose qu'une assignation des coupl utilisés a été faite aux moyens d'entrée réels soit par l'utilisateur, soit p défaut.

Il existe deux méthodes pour gérer les moyens d'entrée logiques :

- on peut en prélever la valeur : instruction assimilable au "Read" (il ne pe y en avoir qu'une à la fois et elle est bloquante)
- on active des couples (M.E.L., E.L.), puis on consulte l'état de ces couple ou on attend qu'une lecture sur l'un d'eux soit terminée - on s'intéresse à fois à l'information et à l'origine de cette information.

II.1.2.2. Le terminal virtuel

Signalons la possibilité de créer une bibliothèque d'images sur un terminal "virtuel". Les images ainsi conservées sont ensuite visualisées sur un terminal quelconque. Le terminal "virtuel" est très élaboré : écran tridimensionnel, coordonnées flottantes, il offre un maximum de possibilités graphiques. Ce terminal reconnaît les facettes et il accepte les ordres de gestion des images.

II.1.3. Le langage de spécification des interactions

Il existe deux modes de programmation :

- Dans le premier le calculateur satellite joue un rôle d'esclave chargé d'assurer la gestion de l'écran et des moyens d'entrée.
- Dans le second, c'est lui qui joue le rôle principal et qui commande éventuellement l'exécution de procédures sur le gros calculateur.

Le programme principal qui s'exécute sur le satellite est en fait l'interpréteur d'un langage de commande de haut niveau : le "Langage de Spécification des Interactions" (L.S.I.).

Le L.S.I. comporte quatre classes d'outils :

- les outils classiques pour la programmation d'algorithmes,
- les outils de création et de manipulation de symboles graphiques et de gestion de l'écran
- les outils de gestion des moyens d'entrées et de description du dialogue
- les outils de liaison avec l'ordinateur principal.

Pour la programmation des algorithmes, le L.S.I. dispose de toutes les notions que l'on trouve en général dans les langages de haut niveau (instructions conditionnelles, de boucle, affectations, etc...), le L.S.I. comporte des variables de type "symbole". Un symbole est une entité graphique construite dans le système de coordonnées de l'écran.

Les moyens d'entrée sont gérés sur la base de la notion de "diagra d'états" introduite par Newman . Chaque état représente un point d'at te du programme. Un choix d'actions "légales" est alors offert à l'utilisate Les moyens d'entrée sur lesquels peut agir l'utilisateur sont divisés en 4 c gories :

- clavier de fonction,
- clavier alphanumérique,
- photostyle,
- système.

La catégorie "photostyle" permet de rendre sensibles ou non des symboles. La tégorie "système" regroupe les fonctions système (fin de tracking, etc...).

On peut regrouper des sous-diagrammes d'état dans des procédures r sant des fonctions d'interaction.

La notion de diagramme d'état implique une gestion synchrone des é ments issus des moyens d'entrée.

Le problème de la liaison avec l'ordinateur principal est un probl délicat dont la solution est très dépendante du type de liaison dont on disp Du point de vue de la programmation le L.S.I. fournit des instructions de dé ration et d'appel de procédures externes.

II.1.4. Critique

Nous intéressant uniquement aux problèmes d'indépendance, nous ne terons pas ici de jugement de valeur sur le logiciel lui-même.

En ce qui concerne le Langage de Spécification des Interactions, l portabilité est impossible (mais ce n'est pas le but recherché). Pour chaque figuration ordinateur satellite-terminal graphique, il faudra écrire un nouv compilateur pour le langage ; ce qui n'est pas très simple.

L'indépendance par rapport aux langages est acquise par l'interméd d'un prétraitement.

Pour ce qui concerne la partie FORTRAN III D (c'est-à-dire les ex sions graphiques), nous pouvons faire les remarques suivantes : FORTRAN III n'est pas un logiciel de base, mais un langage graphique complet (procédures graphiques, objets "transformations"). Il travaille en deux dimensions ou en trois dimensions.

En fait, on trouve les deux niveaux :

- le niveau logiciel graphique de base (affichage de points, segments, caractères en deux dimensions),
- le niveau logiciel de mise en forme (trois dimensions, transformations simples, perspectives, élimination des parties cachées).

Il semble que ces dernières possibilités devraient être offertes dans un logiciel de mise en forme indépendamment du logiciel graphique de base.

De plus FORTRAN III D permet, sur un terminal disposant de possibilités de stockage, de gérer une banque d'images. Celles-ci sont créées et conservées, puis elles peuvent être affichées ou effacées de l'écran à la demande de l'utilisateur.

Cette approche conduit l'utilisateur à gérer une "base de données graphique qui n'a pas de liens directs avec le déroulement de l'application en cours ; c'est-à-dire que l'utilisateur peut afficher des images qui ne sont pas le reflet de son application à un instant donné. Une autre approche consiste à n'afficher que des images issues directement de la base de données de l'application.

Enfin, en ce qui concerne les moyens d'entrée, on peut se demander pourquoi il est nécessaire d'associer des étiquettes logiques (c'est-à-dire en fait des dispositifs physiques) à un moyen d'entrée logique. En effet, l'utilisateur peut ne pas savoir, au moment de l'écriture du programme, quel dispositif il utilisera pour exécuter une fonction logique. Il serait plus agréable de mettre à la disposition de l'opérateur un ensemble de dispositifs physiques, parmi lesquels il choisira, au moment de l'exécution du programme, celui qui l'intéresse.

Pour conclure, disons que FORTRAN III D n'est pas un logiciel de base. C'est un "système" graphique qui offre à la fois les primitives d'un logiciel de base et celles d'un logiciel de mise en forme.

Vis-à-vis du terminal, l'indépendance semble acquise, mais il faut que l'utilisateur indique dans son programme, avant de décrire une image, quel type de terminal il utilise. Or, la description d'une image devrait être totalement indépendante du matériel.

De plus, l'utilisation du centimètre comme unité de la surface de visualisation risque d'inciter l'utilisateur à écrire son programme en fonction de la dimension

de l'écran utilisé, pour avoir des dessins en vraie grandeur. Il y aura perte l'indépendance par rapport au terminal ; et de toute manière, il ne semble pas que le but d'un terminal graphique soit la représentation de plans cotés. Ceci est plutôt du ressort d'une table traçante précise.

II-2- GINO-F

Dans la version originale de GINO, on retrouve un certain nombre de points communs avec FORTRAN III D.

GINO est un ensemble de sous-programmes écrit en Fortran [L 7 , L 8 , L 9]. On peut se demander si le Fortran utilisé est "standard". La communication avec les terminaux est assurée par de petits sous-programmes en Fortran et en langage machine.

II.2.1. Fiche technique

On retrouve les rubriques classiques des autres logiciels, pour les tracés élémentaires :

- tracé d'un segment continu,
- tracé d'un segment avec mode défini,
- déplacement du faisceau éteint,
- affichage d'un point.

La construction et la gestion d'image sont semblables à celles de FORTRAN III GINO utilise la notion de "segment"

- début d'un segment,
- fin d'un segment,
- destruction d'un segment,
- changement de nom d'un segment.

Les images complexes peuvent ainsi être paramétrées en faisant de leur description une "subroutine" Fortran.

GINO fournit les transformations habituelles :
- rotations,
- cadrage,
- etc...

Dans la version originale, les fonctions d'entrée sont très élémentaires :
- lecture alphanumérique (au clavier),
- menu,
- lecture du réticule sur Tektronix 4010.

II.2.2. DINO : protocole réseau

Dans l'optique de l'utilisation de GINO à travers un réseau d'ordinateurs, il lui a été adjoint un protocole de graphique dans les réseaux : DINO. Celui-ci permet de profiter des services graphiques de GINO-F et de plus, le dialogue a été développé. On trouve dans DINO 6 types d'"évènements" :

- Position (récupération d'un couple de coordonnées),
- Poursuite (cas particulier de position),
- Désignation.

Cet évènement contient deux "sous-types" :
- désignation immédiate par photostyle,
- désignation au réticule,
- les évènements "indépendants de l'écran" : points collectés sur une table (dans son propre système), potentiomètre, etc...
- touche de fonctions,
- entrée au clavier alphanumérique.

Chaque type d'évènement peut être autorisé ou interdit par GINO. La liste des évènements autorisés est affichée sur l'écran. L'évènement "courant" est précisé dans cette liste.

II.2.3. Critique

En ce qui concerne GINO lui-même, les remarques sont pratiquement l mêmes que pour FORTRAN 3 D.

GINO offre des transformations qui devraient faire partie d'un logi de mise en forme.

L'indépendance par rapport aux langages est acquise sans préprocess GINO est une bibliothèque de sous-programmes FORTRAN. GINO n'est pas un logic graphique de base.

Il permet à l'utilisateur de gérer les images indépendamment de l'a chage. (La notion de "segment" est semblable à la notion de procédure graphiq Notons cependant que GINO offre beaucoup moins de possibilités que FORTRAN 3 (lignes cachées, etc...) GINO permet pour l'essentiel de simuler un Benson su n'importe quel terminal graphique.

C'est là un point crucial concernant FORTRAN III 3, GINO-F et tous logiciels existant actuellement. La distinction entre la construction de l'ob et sa visualisation proprement dite n'est pas nette. C'est-à-dire que, lorsqu l'utilisateur décrit une image, il pense en même temps à la façon dont elle s tracée : déplacement du faisceau éteint, puis trace d'un segment, etc...
Notons également que la gestion de l'écran ne permet pas l'utilisation complè de celui-ci : une zone dans le bas de l'écran est réservée pour les messages issus du système : liste des "évènements" autorisés, messages d'erreurs, etc.

Le choix du terminal est indiqué très tôt dans le programme.

Enfin, en ce qui concerne les moyens d'entrées, la solution proposé par GINO et DINO est en fait basée sur les dispositifs physiques. Par exemple la désignation au photostyle est distinguée de la désignation simulée avec un réticule, alors que pour l'application, la fonction est la même.

De même, les évènements "indépendants de l'écran" risquent de fourn des résultats liés au matériel (par exemple, coordonnées évaluées dans le rep re de la tablette).

II-3- EUCLID

Nous ne décrirons pas Euclid en détail ici. Signalons seulement qu' Euclid est un logiciel permettant de construire des images [L10].

II.3.1. Fiche technique

Les tracés élémentaires (points, segments, textes en deux dimensions) sont peu développés. La manière normale de procéder est de construire des images.

La construction d'images est assez élaborée : une image est composée d'éléments tels que des lignes, des facettes, etc... EUCLID fournit toutes les transformations usuelles : notions de "vues", représentation perspective, etc...

Ces transformations sont composables, mais il n'existe pas de concept d' "espace des transformations" : On ne peut pas construire et manipuler des objets complexes de type "transformation". Les entrées graphiques sont inexistantes pour l'instant.

La construction de l'image est complètement indépendante de sa visualisation. Une "maquette" est créée, indépendante du terminal, qui peut être utilisée, soit pour visualisation, soit pour rangement sur fichier et traitement ultérieur. Concrètement la maquette est un tableau.

II.3.2. Critique

Nous avons mentionné EUCLID dans ce chapitre pour montrer ce que devrai être un logiciel de mise en forme (voir chapitre I). En effet, EUCLID permet de construire des objets, de leur appliquer des transformations et engendre un code indépendant. De même, le fait que les entrées sont inexistantes n'est pas un problème car la gestion du dialogue n'est pas à la charge d'un logiciel de mise en forme.

EUCLID est un logiciel de mise en forme destiné à une classe d'applications assez large : architecture, (représentation de bâtiments en perspective par exemple), mécanique (représentation de pièces mécaniques avec élimination des parties cachées), etc...

La figure II.2 montre la place d'EUCLID dans l'ensemble du logiciel.

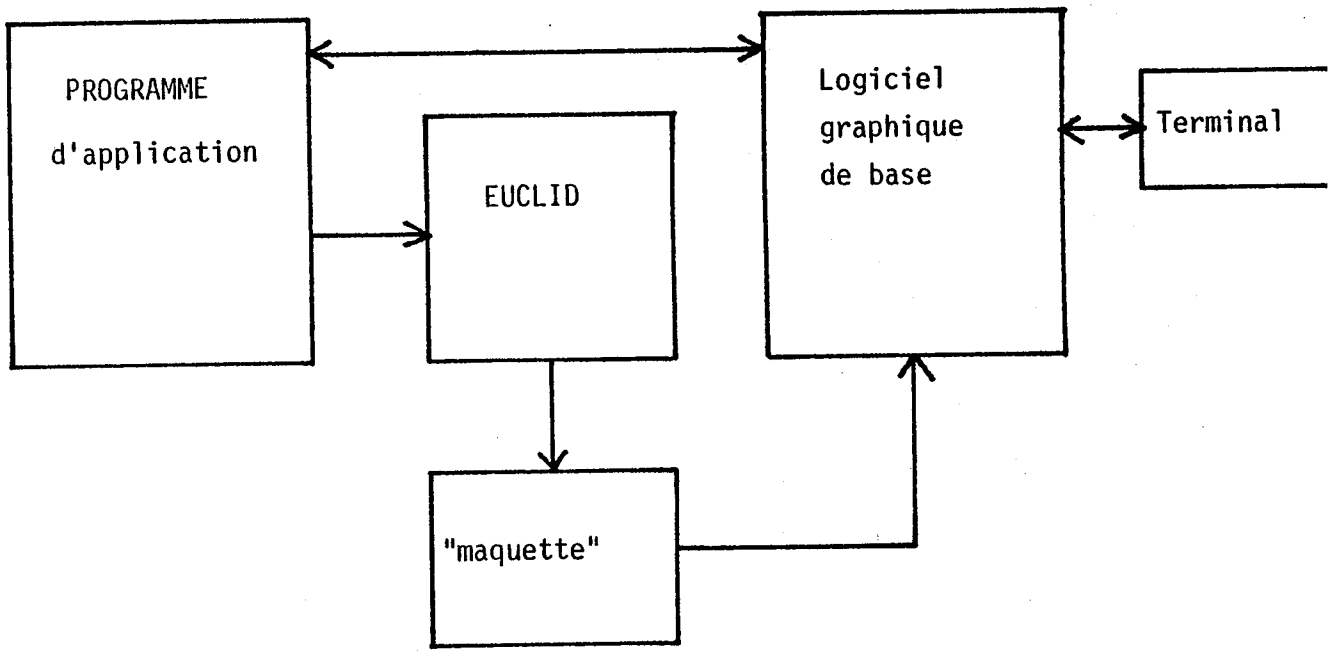


Figure II.2. : EUCLID

La figure II.2 met en évidence le rôle de logiciel de mise en forme joué par EUCLID.

L'expérience a prouvé par exemple qu'un fichier sur bande créé par EUCLID, peut être interprété pour visualisation par un logiciel de base tel que GRIGRI.

Un tel logiciel de mise en forme devrait être portable; pour cela EUCLID est écrit en Fortran, mais il semble que certaines libertés soient prises, qui risquent de limiter la portabilité (sous-programmes à nombre variable d'arguments).

II-4- LE MACRO-GÉNÉRATEUR STAGE 2

Nous faisons ici allusion aux travaux menés à l'Université Paul Sabatier de Toulouse [L1].

Ces travaux ont surtout porté sur la standardisation et la portabilité du logiciel.

La distinction est clairement faite entre :
- l'image : information conceptuelle (phase de construction)
- le dessin : information visuelle (phase de visualisation).

II.4.1. Le logiciel graphique

Il est constitué par une bibliothèque de programmes. On distingue quatre classes de fonctions graphiques :
- contrôle,
- graphique,
- logique,
- interaction.

Les instructions de contrôle sont les suivants :
- initialisation,
- attributs courants (couleur, intensité, clignotement, type de tracé),
- définition de fenêtre sur l'espace utilisateur,
- définition de clôture sur l'espace écran,
- début d'une image,
- fin d'une image,
- interdiction ou autorisation de désignation sur une image.

Les instructions graphiques sont les commandes classiques de tous les logiciel
- mise en place du faisceau éteint en absolu ou en relatif,
- trace d'un segment à partir de la position courante (absolu ou relatif),
- affichage d'un marqueur (caractère spécial) à la position courante,
- appel à une autre image.

Les instructions logiques sont les mêmes que dans FORTRAN III D :
- allumer un dessin,
- effacement d'un dessin,
- desctruction d'un dessin.

On trouve aussi :
- ajouter quelque chose à une image déjà existante,
- modification des attributs,

et les transformations :
- translation,
- rotation,
- changement d'échelle,
- transformation quelconque définie par une matrice 3 x 3.

Les instructions d'interaction sont les suivantes :
- entrée d'un code alphanumérique,
- entrée d'un code clavier de fonctions,
- entrée d'un point au photostyle,
- désignation d'une image au photostyle,
- entrée d'une chaîne alphanumérique.

II.4.2. La portabilité : utilisation d'un macro-générateur

Le macro-générateur STAGE2 opère en deux phases :
- une phase de macro-définition dans laquelle sont lues toutes les macro-opérations pour constituer la bibliothèque du processeur.
- une phase de macro-expansion dans laquelle le processeur compare chaque lig du texte source (constituant alors le programme à porter) aux macro-opératic de la bibliothèque. Chaque ligne lue conduit à l'exécution de la macro-opération qui est :

- soit une génération de code symbolique,
- soit un appel à une autre macro,
- soit un traitement interne au processeur.

La figure II.3 résume le principe du macro-générateur :

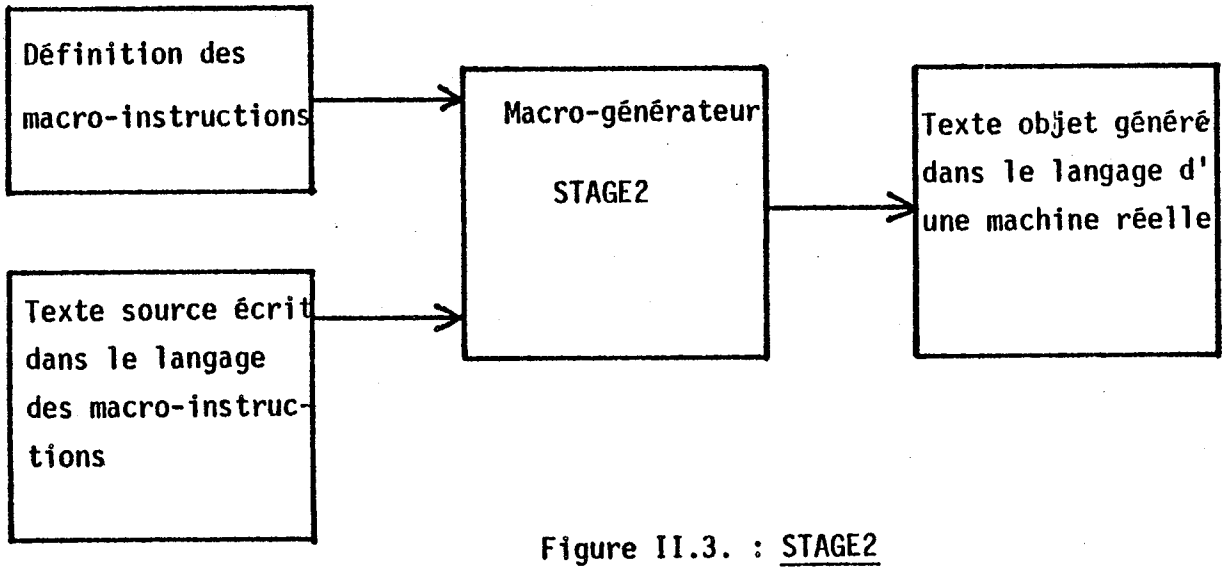


Figure II.3. : STAGE2

Pour utiliser un macro générateur en vue de rendre portable un logiciel, il faut donc :

- définir les macro-opérations une fois pour toutes,
- réécrire les macro-opérations génératrices de code chaque fois que l'on effectue une opération de portabilité.

La définition des macro-opérations constitue le langage d'une "machine-abstraite". Cette machine a intérêt à être très orientée problème, mais la portabilité doit être facile; le compromis réside en la définition d'une hiérarchie de machines abstraites. Le macro-générateur effectue le traitement des machines de niveau supérieur. La machine abstraite effectivement portée est la machine de niveau 1 (figure III.4).

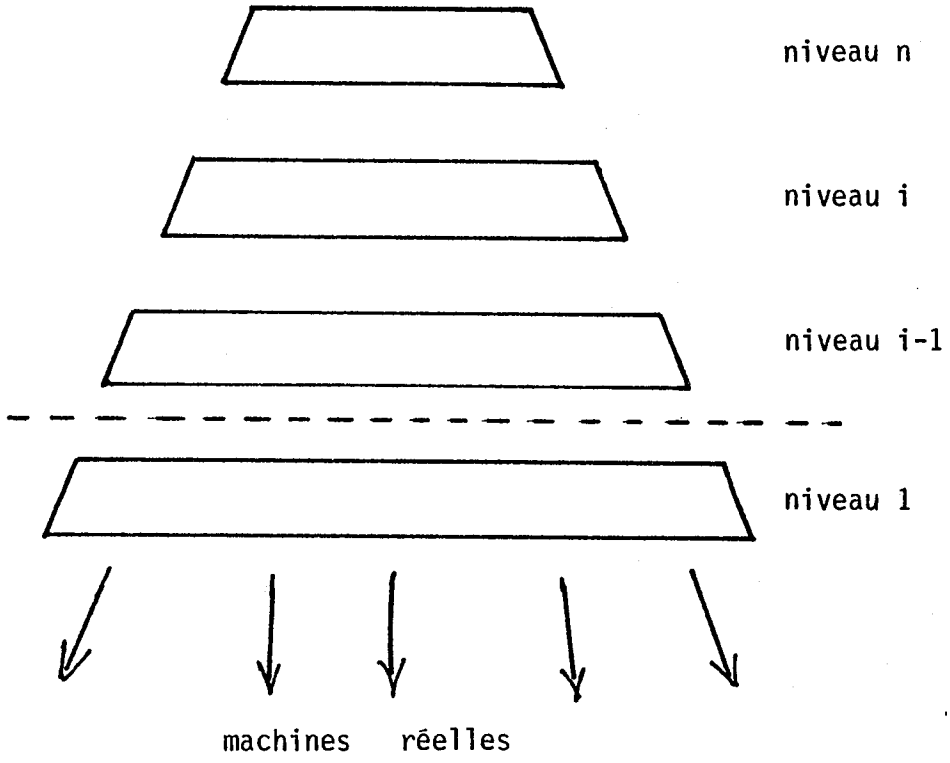


Figure II.4. : Hiérarchie de machines abstraites

La prise en charge du logiciel graphique se déroule donc en plusieurs phases
- définition d'une hiérarchie de machines abstraites et des langages correspo dants,
- écriture du logiciel graphique en terme de ces langages,
- écriture des macro-opérations pour chaque type de matériel.

II.4.3. Critique

Cette approche semble très intéressante,pour ce qui est de l'indépe dance par rapport au calculateur (portabilité). Encore faudrait-il vérifier q la réécriture des macro-opérations n'est pas trop complexe. Le reproche que l'on pourrait faire est le suivant : il faut réaliser une opération de "porta bilité", chaque fois qu'un élément de l'environnement change. Ainsi, si sur u même calculateur, on veut prendre en charge deux consoles, il faudra deux gén rations complètes du logiciel, alors qu'une grande partie de celui-ci est ce tainement commune aux deux.

Il semble donc que cette approche néglige un peu le problème de l'indépendance par rapport au terminal (solution adaptée à la prise en charge de plusieurs terminaux connectés chacun à des calculateurs différents, mais pas à un même ordinateur pilotant un ensemble de terminaux différents).

Cependant, signalons un avantage non négligeable : après transport, le logiciel se trouve écrit en langage machine; d'où un gain de place et une optimisation possible du code (ce qui n'est pas le cas lorsque le logiciel est écrit en Fortran).

En ce qui concerne les primitives, on peut déplorer, là encore la gestion d'une banque d'images indépendante de l'affichage. L'utilisateur construira son image en pensant à l'affichage (position du faisceau, etc...).

En ce qui concerne le dialogue, le raisonnement en terme de dispositifs physiques empêche d'approcher au mieux l'indépendance par rapport au matériel : les fonctions manquantes sont simulées, il serait donc plus intéressant de parler, par exemple, de collecte de coordonnées, sans préciser que cette entrée est mise en oeuvre avec le photostyle.

En conclusion, cette approche de la portabilité est une notion intéressante.

II-5- LA PREMIÈRE PROPOSITION DU GROUPE DE TRAVAIL DU RÉSEAU ARPA

Dans le contexte d'un réseau d'ordinateurs, l'indépendance par rapport au matériel est fondamentale. C'est pourquoi différents groupes travaillent actuellement sur le sujet.

Etudions rapidement la première prosposition du groupe de travail du réseau ARPA [C1], [C2], [C3], [C4].

II.5.1. L'affichage

L'affichage est basé sur la génération d'un code intermédiaire indépendant du terminal utilisé. C'est ce code qui transitera à travers le réseau.

Les coordonnées sont des réels, codés entre -1/2 et 1/2. Ceci revient

en fait à se définir un écran virtuel par rapport auquel seront évaluées toute les coordonnées.

Le découpage de l'image est assuré avant génération du code intermédia ainsi seules les coordonnées utiles seront envoyées sur le réseau.

Les commandes envisagées par ce code intermédiaire ont été réparties en plusieurs niveaus de hiérarchie, au niveau le plus bas on trouve :

- effacement de l'écran,
- déplacement du faisceau en absolu,
- déplacement du faisceau éteint en relatif,
- tracé d'un segment en absolu,
- tracé d'un segment en relatif,
- affichage d'un point en absolu,
- affichage d'un point en relatif,
- affichage d'un texte à la position courante,
- affichage d'un texte à la position courante avec remise en place du faisceau
- fin d'une image,

(cette commande est associée à un effacement précédent).

Les niveaux supérieurs introduisent des commandes telles que :

- définition du type de tracé,
- définition de l'intensité,
- définition de sous-image (début, fin)
- etc...

On y trouve aussi les commandes de transformations (translation, rotation, échle, etc...)

II.5.2. Le dialogue

Au niveau le plus bas les entrées sont décrites par un schéma unique comportant :

- l'origine (dispositif physique)
- le nombre (longueur des données récupérées),
- les données récupérées.

Au départ, seuls deux types "logiques" sont envisagés :
- récupération d'une chaîne d'octets,
- récupération d'un couple de coordonnées.

La chaîne d'octets peut être récupérée après l'activation d'un dispositif quelconque :
- clavier,
- tablette,
- photostyle,
- touche de fonction,
- etc...

De même la commande de collecte de coordonnées peut être activée à partir de n'importe quel dispositif physique. Dans ce cas, les coordonnées renvoyées sont évaluées, comme pour l'affichage entre -1/2 et 1/2.

Le format des deux commandes est le suivant

TEXTIN : DEVICE : COUNT : text-string
POSIT : DEVICE : COUNT : X : Y

II.5.3. Critique

La solution, pour l'affichage, de définition d'un écran virtuel semble assez intéressante. Elle a été reprise à Grenoble, lors d'une première génération du logiciel GRIGRI (voir chapitre III). L'indépendance par rapport au matériel est assurée, mais cette approche conduit en fait, en fonction des commandes intermédiaires autorisées, à une sous-exploitation de certains terminaux et à une duplication de l'information. Ces problèmes sont évoqués en détail dans le chapitre suivant.

En ce qui concerne le dialogue, la solution proposée est peu satisfaisante. En effet, dans la primitive TEXTIN, l'utilisateur devra savoir quel dispositif a été activé, pour pouvoir récupérer correctement les informations.

Notons par contre que lors de l'introduction d'un point, les coordonnées renvoyées sont indépendantes du terminal.

Cette solution, satisfaisante dans l'ensemble, mais trop lourde et trop restrictive a conduit le groupe de travail à envisager une approche prenant en compte le type de matériel.

II-6- LA SECONDE PROPOSITION ARPA

Nous insisterons un peu plus sur cette proposition [I1 , I2] car e s'appuie sur la définition d'un système graphique proposée par NEWMAN et SPRO

II.6.1. Modèle conceptuel d'un système graphique

Cette définition s'inspire de NEWMANN et SPROULL [G7] . La figure montre l'organisation d'ensemble d'un logiciel graphique et du programme d'ap cation.

On trouve respectivement :

- les programmes de traitement des interruptions,
- le programme de construction de l'image structurée,
- la définition structurée de l'image,
- les programmes d'interprétation de l'image structurée,
- le programme de composition des transformations,
- les programmes de transformations et de coupages,
- le générateur de code réel (G2),
- la liste d'affichage,
- le générateur de dessin (G3).

La définition structurée de l'image est ici une définition en terme de primitives très évoluées et de "procédures graphiques" (sous dessins) qui p vent s'appeler entre elles.

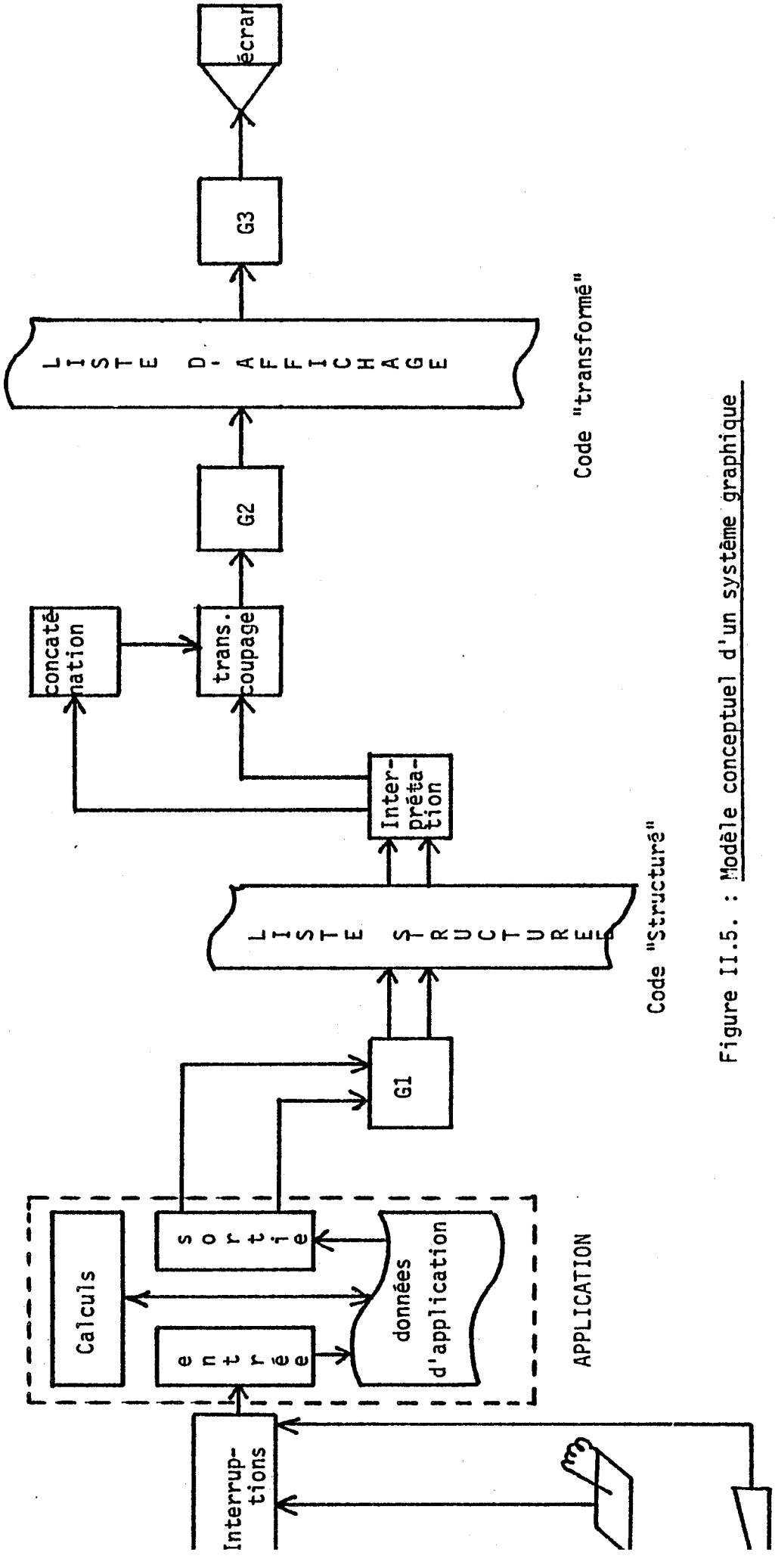


Figure II.5. : Modèle conceptuel d'un système graphique

La liste d'affichage, par contre, est composée d'ordres élémentaires (points, segments, déplacements du faisceau, caractères, etc...) qui constituent le code "transformé".

Les transformations ne sont pas appliquées aux images au niveau de la liste structurée, c'est pourquoi elles sont transmises parallèlement aux images jusqu'au moment de la génération de la liste d'affichage.

II.6.2. Application à l'indépendance par rapport au terminal

Un tel modèle vu sous l'angle de l'indépendance pose deux questions
- quel type de code le logiciel indépendant devra-t-il engendrer ?
- quelle est la limite entre le logiciel indépendant et le logiciel dépendant du terminal ?

De plus, dans le schéma précédent, certaines parties sont logicielles, d'autre matérielles et certaines n'existent pas dans tous les cas (par exemple la list d'affichage dans le cas d'un terminal à tube mémoire).

La solution consistant à engendrer du code structuré indépendant des terminaux a été rejetée a priori car il semble qu'elle poserait d'énormes problèmes aux terminaux peu évolués. En effet, ceux-ci recevant une liste composé d'images et de sous-images auxquelles sont appliquées des transformations auraient à leur charge la transformation de ces informations en code assimilable par leur processeur graphique. Or cette gestion est assez complexe et ne doit pas être à la charge d'un logiciel dépendant du terminal.

La solution retenue consiste à engendrer du code transformé indépendant des terminaux.

Ainsi, le code intermédiaire (c'est-à-dire le protocole réseau) est utilisé pour construire et manipuler des "segments".

Un segment est une liste de commandes graphiques élémentaires. On pe détruire un segment, le remplacer, l'ajouter ou le retirer de la liste d'affichage.

Dans ce contexte, un dessin est donc un ensemble de segments. Le découpage en segments permet de minimiser les échanges entre les deux parties du logiciel : on peut modifier un ou deux segments sans avoir à reconstruire l'image.

Dans cette configuration, le rôle du module dépendant du terminal devrait être réduit au minimum.

III.6.3. Le code intermédiaire

Dans cette proposition, il n'y a pas de terminal "général" vu par le logiciel indépendant. Celui-ci essaie de s'adapter aux possibilités offertes par le matériel.

Ainsi, au début de toute application, les deux parties du logiciel établissent un dialogue : le module qui pilote le terminal utilisé envoie au logiciel indépendant, des renseignements sur le matériel. Ces renseignements sont les suivants :

- la liste des commandes implémentées (code transformé, ou code structuré),
- le système de coordonnées de l'écran,
- des paramètres précisant les options possibles pour la taille des caractères, l'intensité, le type de tracé, la résolution, etc...,
- la liste des dispositifs d'entrée existant et des "évènements" réalisables (un évènement est équivalent à un moyen d'entrée logique : entrée d'un couple de coordonnées, désignation, etc...) à partir de ces renseignements, le logiciel indépendant va élaborer une "stratégie" pour engendrer le code le plus adapté possible au terminal utilisé.

Dans le cas du dialogue, le logiciel central va recenser les dispositifs nécessaires à la mise en oeuvre de ses "évènements". Si les dispositifs réels sont insuffisants, il demande au logiciel dépendant du terminal de simuler les dispositifs manquants, si c'est possible.

Le code intermédiaire est donc composé de trois types de commande :

- renseignement sur le terminal,
- affichage,
- dialogue.

Pour l'affichage, on peut distinguer les instructions de contrôle et les inst
tions graphiques.
Les instructions de contrôle sont relatives aux segments :

- ouverture/fermeture d'un segment,
- affichage/effacement d'un segment,
- destruction d'un segment,
- ajout à un segment,
- fin d'un lot de modifications.

Cette dernière commande est intéressante, car elle permet de regrouper un ens
ble de modifications de la liste d'affichage et de les répercuter en bloc sur
l'écran (minimisation du nombre des mises à jour d'un écran à tube mémoire).
Les primitives graphiques sont là encore, au niveau le plus élémentaires : af
chage d'un point, mise en place du faisceau, tracé d'un segment, affichage de
texte.

Le système de coordonnées est celui de l'écran utilisé. On trouve a
si de nombreuses commandes permettant de préciser les attributs d'un segments
(couleur, intensité, tracé, etc...)

La gestion des moyens d'entrées se fait de deux façons :

- lire l'état d'un dispositif sur demande,
- utiliser un dispositif logique(évènement) tel que clavier alphanumérique, c
  vier de fonctions, collecte d'un point, désignation etc..., à chaque dispos
  tif logique est associé un ensemble de dispositifs physiques qui devront êt
  testés quand l'évènement se produit.

II.6.4. Critique

Une telle approche, si elle est très intéressante sur le plan conce
tuel est pratiquement impossible à réaliser. En effet, par exemple, la mise
en oeuvre d'un dialogue entre les deux parties du logiciel pour mettre au poi
une stratégie pour l'utilisation des moyens d'entrées physiques paraît plutôt
utopique.

Pour l'affichage, l'idée de tenir compte du type du terminal (système de coordonnées, précision de l'écran, etc...) peut être très intéressante. On supprime ainsi le problème rencontré dans la première approche (console virtuelle) : on peut exploiter au maximum les matériels performants. Mais si on pousse un peu trop loin le raisonnement, on constate que le logiciel indépendant fera en fait un traitement spécial pour chaque terminal et on risque de perdre l'indépendance.

En fait, pour utiliser ce principe, il faudra faire un compromis : le code intermédiaire ne sera plus totalement indépendant du matériel, il sera "adapté" au type de terminal.

Dans cette proposition, le traitement du dialogue est décrit de façon assez imprécise.

Pour conlure, disons que l'idée d' "adaptabilité" se substituant à celle d'indépendance est très intéressante. Mais qu'il ne faut pas tomber dans l'excès consistant à trop tenir compte du matériel, car on perd l'indépendance et le logiciel risque de devenir monstrueux. Les auteurs du rapport ARPA eux-mêmes sont revenus sur leurs propositions, car la réalisation complète en était impossible.

Cette approche a beaucoup influencé les travaux actuels, en particulier ceux du groupe de travail sur le problème du graphique à travers le réseau CYCLADES.

II-7- SYNTHÈSE

Nous n'avons évidemment pas décrit ici tous les logiciels existants. Mais on peut relever un certain nombre de caractéristiques communes :

- tous ces logiciels offrent la possibilité de stocker des images dans une banque d'images, pour les afficher ultérieurement à la demande. Le dessin affiché sur l'écran n'est donc pas toujours le reflet de l'évolution des données de l'application.

- les logiciels existants ne distinguent pas les primitives de mise en forme et les primitives graphiques de base, mais offrent les deux au même niveau
- les logiciels existants ne séparent pas la construction d'un objet, de la façon dont il sera visualisé : en effet, ils offrent des primitives qui tracent en terme de déplacement du faisceau allumé ou éteint (ou de plume haute ou plume basse). Le programmeur construit donc ses objets en fonction des déplacements de ce faiseau.

Cette démarche n'est pas naturelle car la construction d'un objet (par exemple le calcul des coordonnées d'une suite de points) est indépendante de la façon dont l'objet sera visualisé. De plus cette approche risque de poser des problèmes en ce qui concerne l'indépendance du logiciel par rapport au matériel : comment interpréter un déplacement du faisceau lorsque l'on travaille sur un terminal à balayage ligne par ligne ?

- une autre caractéristique commune à ces logiciels, est qu'ils sont presque tous écrits en Fortran, par souci de portabilité.
- Enfin, notons que la plupart des logiciels traitent le dialogue en termes de dispositifs physiques (plus ou moins directement).

Les points abordés ici montrent d'ores et déjà que la standardisation des logiciels graphiques sera certainement difficile à réaliser. En ce qui concerne, l'indépendance par rapport au terminal, les logiciels en cours de développement actuellement s'inspirent tous de l'une des deux propositions du réseau ARPA : console virtuelle ou logiciel adaptatif.

Le protocole restreint en cours d'élaboration pour le graphique dans le réseau CYCLADES [I7] utilise le concept de console virtuelle.

CHAPITRE III - CONCEPTION ET REALISATION D'UN LOGICIEL GRAPHIQUE BASE SUR LE CONCEPT DE "CONSOLE VIRTUELLE" - LA PREMIERE GENERATION DE GRIGRI

Tout logiciel graphique peut être considéré comme formé de deux modules (voir I.8) :

- un module de communication avec l'application, indépendant du terminal,
- un module de gestion du terminal.

La communication entre les deux parties du logiciel est réalisée par un code indépendant du terminal. La présence d'une liste d'affichage contenant une description synthétique de l'image est obligatoire, dans tout logiciel, pour réaliser les fonctions d'effacement sélectif et d'identification.

Si nous attribuons la gestion de cette liste d'affichage au module indépendant du terminal, celle-ci devra elle-même contenir un code indépendant. On peut alors considérer que le module gère une "console graphique virtuelle". Le module de gestion du terminal joue alors le rôle d'un simple traducteur de code entre la console virtuelle et le terminal utilisé. La configuration du logiciel est décrite dans la figure III.1.

Ainsi, le simple fait de placer la liste d'affichage dans le module de communication, conduit à assurer l'indépendance vis à vis du terminal en donnant aux utilisateurs la possibilité de travailler avec une console graphique virtuelle quelque soit le matériel utilisé réellement.

III-1- DESCRIPTION D'UNE CONSOLE VIRTUELLE [C5]

Pour constituer une console virtuelle, il est nécessaire de définir tou les composants d'une console graphique :

- l'écran,
- le processeur graphique,
- les dispositifs de dialogue,
- la liste d'affichage.

Dans ce qui suit, nous décrivons la console virtuelle qui a été définie pour supporter la première version du logiciel GRIGRI.

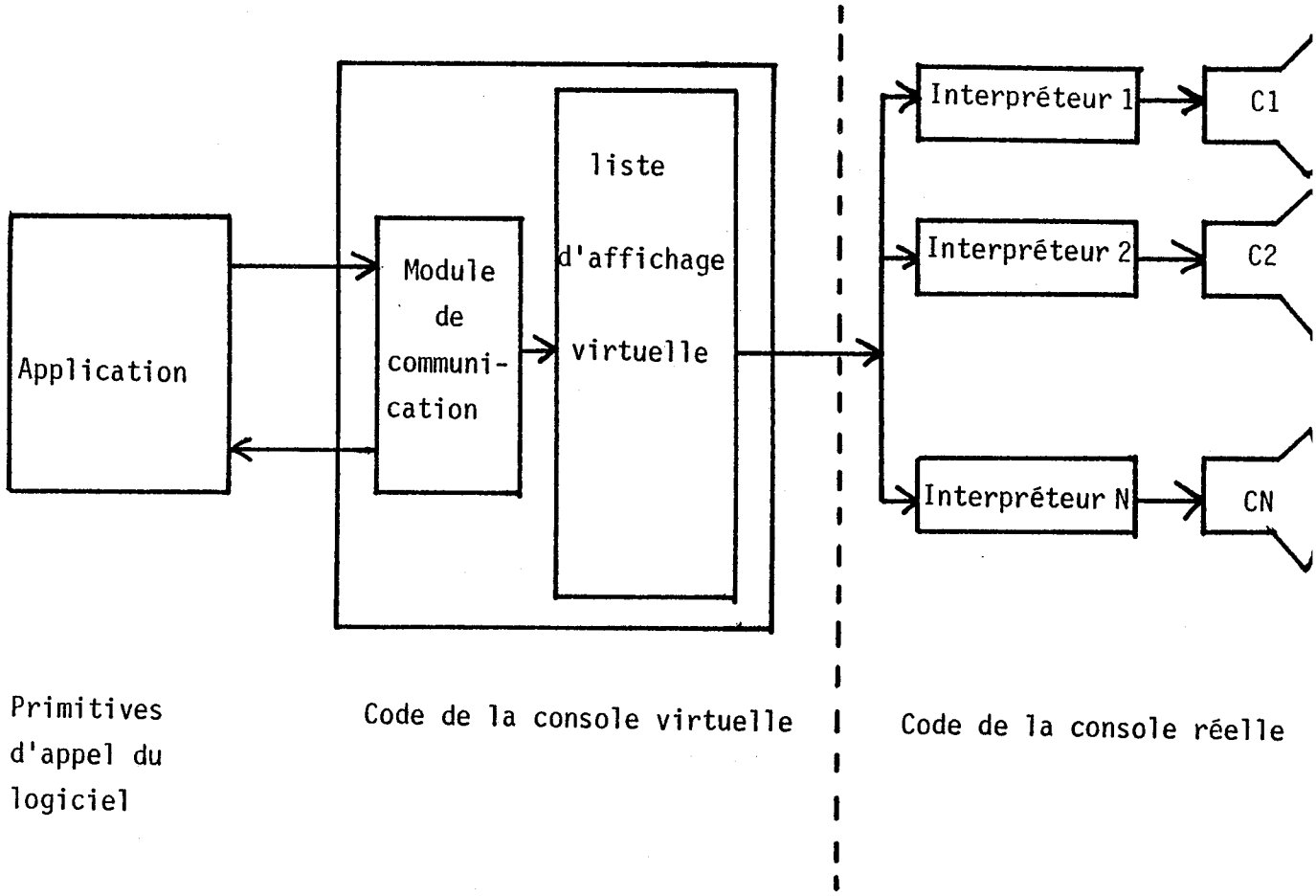


Figure III.1. : Logiciel graphique avec console virtuelle

III.1.1. L'écran virtuel

Quelque soit le terminal sur lequel il travaille, l'utilisateur a l'impression d'afficher ses dessins sur un seul et même écran : l'écran virt Le choix des paramètres de définition de cet écran (taille, forme et précisi est très délicat.

Nous avons choisi les options suivantes :

- forme : carré
- système de coordonnées (précision virtuelle) : coordonnées entières compri entre 0 et 1023.

En ce qui concerne les caractères, nous ne précisons rien sur leur taille au niveau de l'écran virtuel.

III.1.2. Le processeur graphique virtuel

Le choix de la puissance du processeur graphique virtuel est très important, car il permettra de plus ou moins bien employer les diverses conso Pour des raisons de simplicité, les fonctions retenues représentent en généra l'intersection des possibilités des consoles disponibles, ce qui risque de co

duire à une perte de qualité sensible dans le cas des consoles évoluées.

Les commandes graphiques contenues dans la liste d'affichage virtuelle, c'est-à-dire les opérations exécutables par le processeur graphique virtuel sont les suivantes :

- affichage de segments consécutifs en mode absolu,
- affichage de segments consécutifs en mode relatif,
- affichage de segments disjoints en mode absolu,
- affichage de segments disjoints en mode relatif,
- affichage de points en mode absolu,
- affichage de points en mode relatif,
- affichage de texte en mode absolu.

Ces fonctions ont été choisies pour être compatibles avec les primitives du logiciel et pour être facilement prises en charge par les divers interpréteurs.

III.1.3. Les dispositifs de dialogue virtuels

Nous ne définissons pas ici de dispositifs virtuels correspondant à des dispositifs physiques, car cette approche rendrait difficile une prise en charge de terminaux différents. En effet, si la console virtuelle dispose par exemple d'un "photostyle virtuel", que devra faire l'interpréteur pour répondre à une telle demande lorsque le terminal utilisé n'a pas de photostyle ?

Pour résoudre ce problème, les dispositifs de dialogue virtuels sont assimilés à des dispositifs logiques. Pour des raisons de compatibilité avec le logiciel, les dispositifs de dialogue virtuels sont les suivants :

- un dispositif logique d'introduction de coordonnées,
- un dispositif logique d'introduction de valeurs,
- un dispositif logique d'identification,
- un dispositif logique de menu.

La prise en charge de ces fonctions logiques sera faite au niveau des interpréteurs pour utiliser au mieux les possibilités offertes par le matériel.

III.1.4. La liste d'affichage virtuelle

L'image virtuelle simule en fait une mémoire d'entretien. On y trou trois types de renseignements :

- des renseignements sur les figures,
- des renseignements relatifs aux divers éléments graphiques,
- des données : coordonnées, caractères.

III.1.4.1. La table de descriptions des figures

L'utilisateur peut se définir 16 figures au maximum. De plus, le lo ciel se réserve une "figure système" inaccessible à l'utilisateur.

La description des figures est stockée dans une table de 17 élément On y trouve tous les renseignements attachés à une figure :

- le numéro de la figure,
- le type de coordonnées de la figure (absolu ou relatif)
- les bornes de la fenêtre,
- les bornes de la clôture dans le repère écran virtuel,
- le pointeur vers le premier élément de la figure,
- le pointeur vers le dernier élément de la figure.

III.4.1.2. La table des éléments graphiques

Chaque commande graphique élémentaire est constituée des renseignements suivants :

- le code opération,
- le pointeur vers les opérandes (coordonnées dans le repère écran virtuel ou caractères),
- la longueur de la zone contenant les opérandes,
- le pointeur vers l'élément suivant appartenant à la même figure.

Les éléments d'une même figure sont chaînés entre eux.

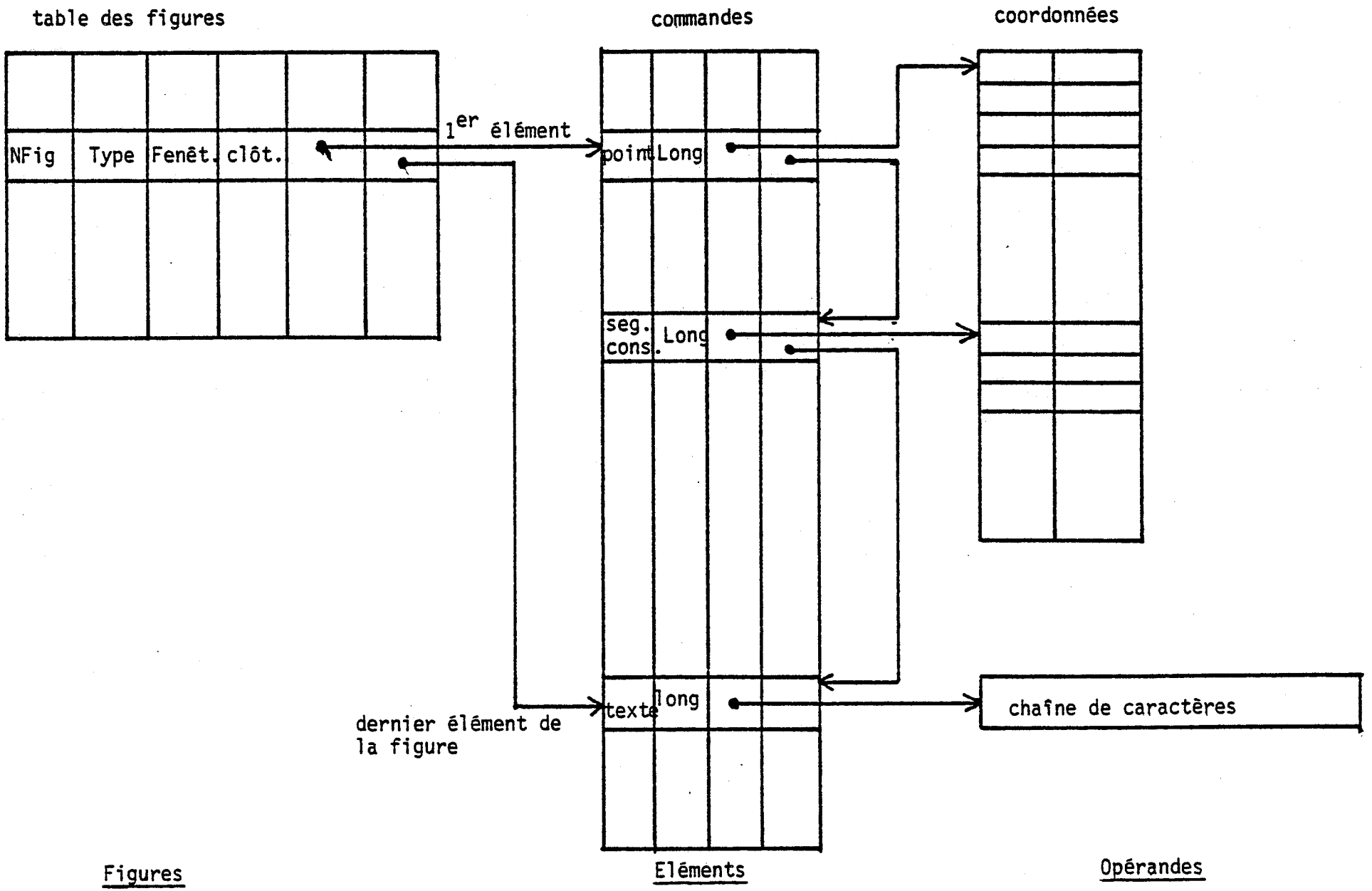


Figure III.2. : Organisation de la console virtuelle

III.1.4.3. La table des coordonnées

C'est un tableau qui contient les coordonnées des points, segments consécutifs ou disjoints, en absolu ou en relatif, dans le repère écran virtuel. Le mode de rangement des coordonnées diffère suivant la commande à laquelle elles sont associées.

La dimension de cette table est fonction du type d'application que l'on désire prendre en charge. En général, une image virtuelle permettant de stocker 4000 ou 5000 coordonnées est suffisante.

L'organisation d'ensemble de l'image virtuelle est schématisée dans la figure III.2.

III.1.5. Stockage des coordonnées

Lors d'un affichage de points ou de segments consécutifs. Les coordonnées sont stockées sans modification, après passage dans le système de l'écran virtuel et coupage.

Le premier point est toujours donné en absolu.

Lors d'un affichage de segments disjoints, on passe systématiquemen les coordonnées du début de chaque segment en absolu. Les coordonnées de l'extrémité sont stockées, soit en absolu, soit en relatif par rapport à l'origin du segment.

On constate que le mode relatif est difficile à mettre en place au niveau de la console virtuelle, d'autant plus que le coupage est exécuté au niveau virtuel afin de ne pas stocker des points inutiles.

III.1.6. Le découpage dans l'image virtuelle

Le découpage étant réalisé au niveau de l'image virtuelle, le nombre de points ou de segments stockés dans la liste virtuelle peut être différent du nombre demandé par l'utilisateur, car on ne stocke que les points visibles Ceci pose un problème dans le cas où la figure est de type relatif.

En effet l'utilisateur envoie des coordonnées relatives dans son propre repère, et la console virtuelle stocke des coordonnées relatives dans le repère écran virtuel. Le déplacement ne sera pas le même dans les deux cas, car les déplacements relatifs dans la console virtuelle doivent être calculés par rapport à la dernière position affichée et non pas par rapport à la dernière position demandée par l'utilisateur.

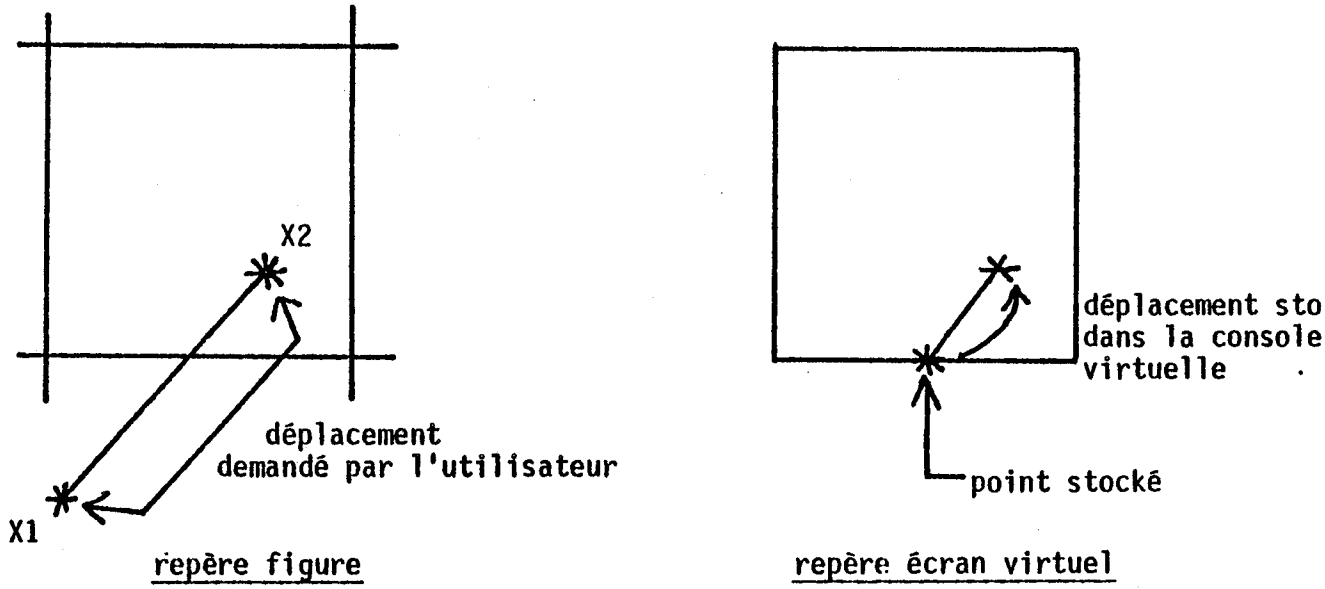


Figure III.3. : Déplacement relatif et coupage

Les coordonnées relatives ne peuvent donc pas être stockées directement, il faut calculer le déplacement par rapport à la dernière position enregistrée dans la console virtuelle.

III-2- LES PRIMITIVES D'AFFICHAGE DE LA PREMIÈRE GÉNÉRATION DE GRIGRI

La première version de GRIGRI [C6] est un logiciel graphique de base, de niveau relativement bas. Nous nous sommes volontairement limités, dans cette première expérimentation à un petit nombre de primitives simples.

III.2.1. Terminologie

Dans ce logiciel, nous utilisons un certain nombre de concepts, dor nous rappelons ici la définition.

- un OBJET est la plus petite entité que l'on puisse manipuler ou repérer. Ce peut être un point ou une suite de points, un segment ou une suite de segme ou une chaîne de caractères. C'est ce qui est créé lors de l'invocation d'u fonction de génération.

- une FIGURE est un ensemble d'objets ayant des caractéristiques communes vis vis de l'application. En particulier, ils portent le même "nom" (numéro de figure) et ils sont définis dans le même espace, par rapport au même référe tiel.

- une CLOTURE est une portion rectangulaire de l'espace écran virtuel, qui se associée à une figure.

- un DESSIN est ce qui "apparaît" sur l'écran virtuel à un instant donné. Il peut être composé de une ou plusieurs figures.

III.2.2. Description des primitives d'affichage

Les primitives d'affichage sont de deux types :

- primitives de définition d'espaces,
- primitives de génération.

III.2.2.1. Déclaration de figure

Chaque figure doit être déclarée et définie dans la console virtuel avant toute utilisation, à l'aide de la primitive :

FIGURE (nfig, coortype, igx, igy, sdx, sdy)

Les paramètres sont respectivement :

- le numéro de la figure,
- le type des coordonnées (absolu ou relatif)
- les bornes, dans le repère figure, du domaine à visualiser.

III.2.2.2. Définition de clôture

On peut allouer à chaque figure déclarée, une portion rectangulaire de l'espace écran virtuel à l'aide de la primitive suivante :

CLOTURE (nfig, igx, igy, sdx, sdy)

Les paramètres sont respectivement :

- le numéro de la figure à laquelle on associe cette clôture,
- les bornes, dans le repère écran virtuel, du domaine alloué à la figure.

Notons que l'utilisateur travaille sur l'écran virtuel, les bornes de la clôture sont donc des valeurs comprises entre 0 et 1023.

La mise en page est ainsi réalisée au niveau de l'écran virtuel, sans se soucier de la mise en page de l'écran réel.

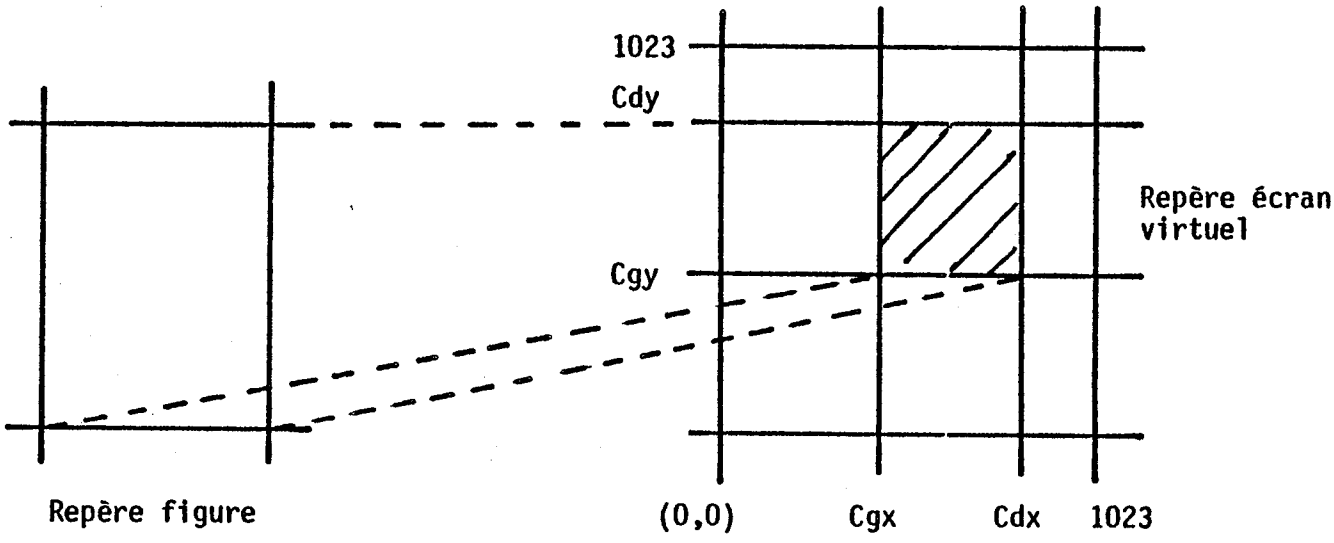


Figure III.4. : CLôture dans l'écran virtuel

III.2.2.3. Génération de points et de segments

L'affichage de points et de segments se fait à l'aide la primitive

AFFICHER (nident, mode, nombre, tabx, taby)

Les paramètres sont respectivement :

- le numéro d'identification de l'objet,
- le mode de génération,
- le nombre de points ou de segments à afficher,
- les tableaux contenant les couples de coordonnées.

Le numéro d'identification est égal au numéro de figure augmenté d'un numéro d'ordre dans cette figure, multiplié par cent. Il permet, à l'utilisateur de distinguer les objets d'une même figure, et au logiciel de savoir à quelle figure appartient l'objet.

Le mode permet de préciser si l'on veut afficher une ligne brisée, des segment disjoints ou des points. Nous avons choisi de regrouper ces trois primitives, car elles correspondent toutes à la visualisation d'une suite de points, seul diffère le mode de représentation.

III.2.2.4. Génération de texte

L'affichage d'une ligne de texte, ainsi que la réservation de "zones blanches" à l'intérieur de cette ligne s'effectue grâce à la primitive :

EDITER (nident, nligne, description, X, Y)

Les paramètres sont respectivement :

- le numéro d'identification,
- le numéro de la ligne dans la figure,
- la description de la ligne,
- les coordonnées (absolues) du début de la ligne.

Le numéro d'identification a la même signification que dans la primitive AFFICHER.

La description de la ligne contient des suites de caractères à afficher sur l'écran et des indications pour réserver des zones pour un emploi ultérieur (édition et entrée de valeurs).

III.2.2.5. Edition alphanumérique

Les zones "blanches" réservées dans une ligne de texte peuvent être garnies à l'aide de données numériques grâce aux 2 primitives suivantes :

EDENTIER (nfig, nligne, nzone, valeur-entière)
EDREEL (nfig, nligne, nzone, valeur-réelle, format)

Les paramètres sont respectivement :
- le numéro de figure à laquelle appartient la ligne,
- le numéro de la ligne à compléter,
- le numéro d'ordre de la zone dans la ligne,
- la valeur à éditer,
- le format d'édition (décimal ou flottant, précision).

Ces fonctions ne sont pas purement graphiques, mais sont une facilité offerte aux utilisateurs, pour les décharger de la conversion numérique-caractères.

III.2.2.6. Effacement

L'effacement se fait au niveau des figures à l'aide la primitive :

EFFACER (nfig)

Le paramètre représente le numéro de la figure à effacer. S'il est égal à zéro, on efface toutes les figures.

III.2.3. Gestion de l'image virtuelle

Nous décrivons succintement, ici, la gestion de la console virtuell lors de l'invocation des différentes primitives d'affichage.

III.2.3.1. Gestion de la table des figures

Lorsqu'une figure est déclarée, on examine la table des figures : s la figure existe déjà, on modifie ses caractéristiques (mais ceci n'a aucun ef fet sur ce qui a déjà été affiché dans la figure), si elle n'existe pas encore on stocke les paramètres de définition d'espace. On initialise les bornes de l clôture associée avec (0,0,1023,0123) qui représente la totalité de l'écran vi tuel, pour le cas où aucune clôture ne serait associée à cette figure (la clôtu re par défaut est la totalité de l'écran virtuel).

Lors de la définition d'une clôture, on cherche, dans la table des figures, celle qui lui est associée et on stocke les bornes de cette clôture.

III.2.3.2. Affichage de points et de segments

Lors de l'invocation de la primitive AFFICHER, on commence par rempl la table des ordres graphiques internes :

- on recherche le numéro de figure,
- on stocke le code opération élémentaire, choisi en fonction du mode et du fai que la figure est en absolu ou en relatif,
- on assure le chaînage avec l'ordre précédent dans la figure,
- on positionne le pointeur vers la zone où seront stockées les coordonnées,
- on initialise la longueur de cette zone à 0,
- on initialise le pointeur de chaînage à -1.

Ainsi, la commande est enregistrée dans la table des éléments. On tr te ensuite les coordonnées.

A chaque figure l'utilisateur associe un système de coordonnées dif férent. Les coordonnées "utilisateur" d'une figure peuvent être évaluées en re latif, ceci oblige à conserver pour chaque figure la dernière position absolue De plus, le découpage étant réalisé dans l'écran virtuel, on doit également co

server pour chaque figure la dernière position absolue visible dans le repère écran. Ainsi, si l'on nomme PABS la dernière position absolue demandée par l'utilisateur, et PAVIRT la dernière position absolue visible, le traitement d'un point ou d'un segment conduira aux situations suivantes :

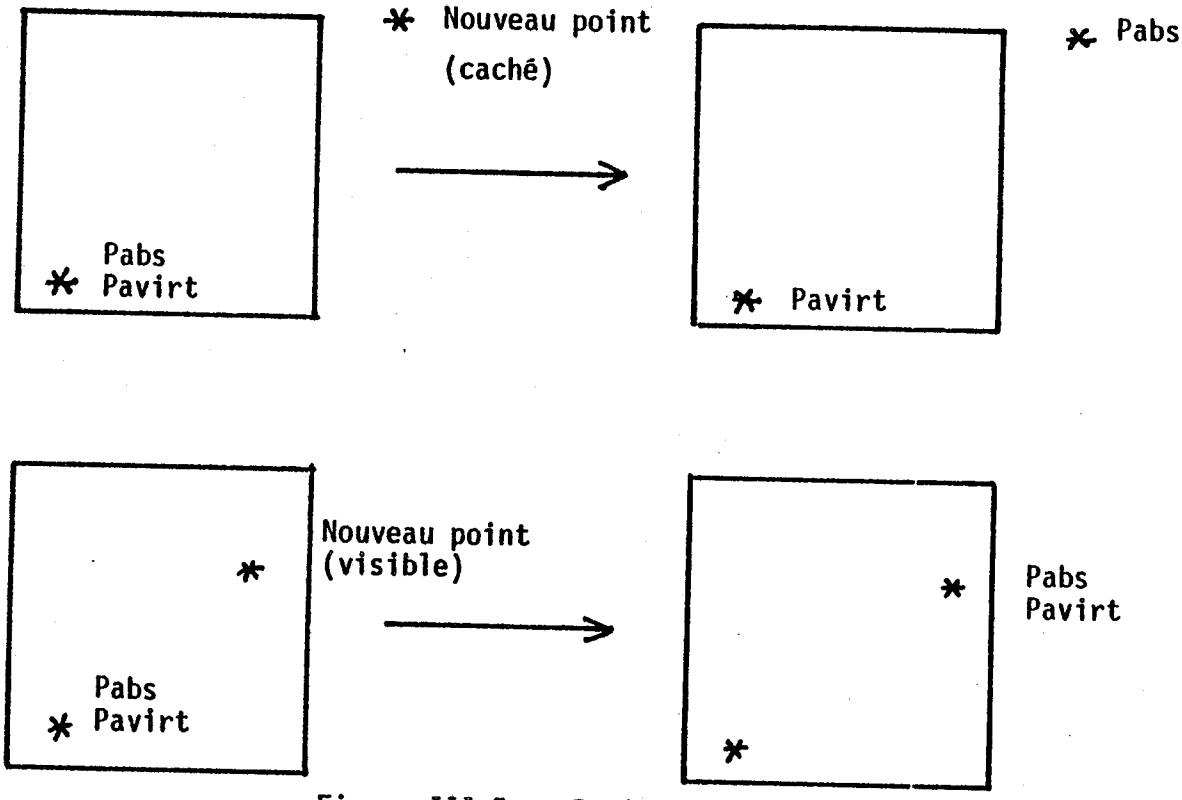


Figure III.5. : Traitement des points

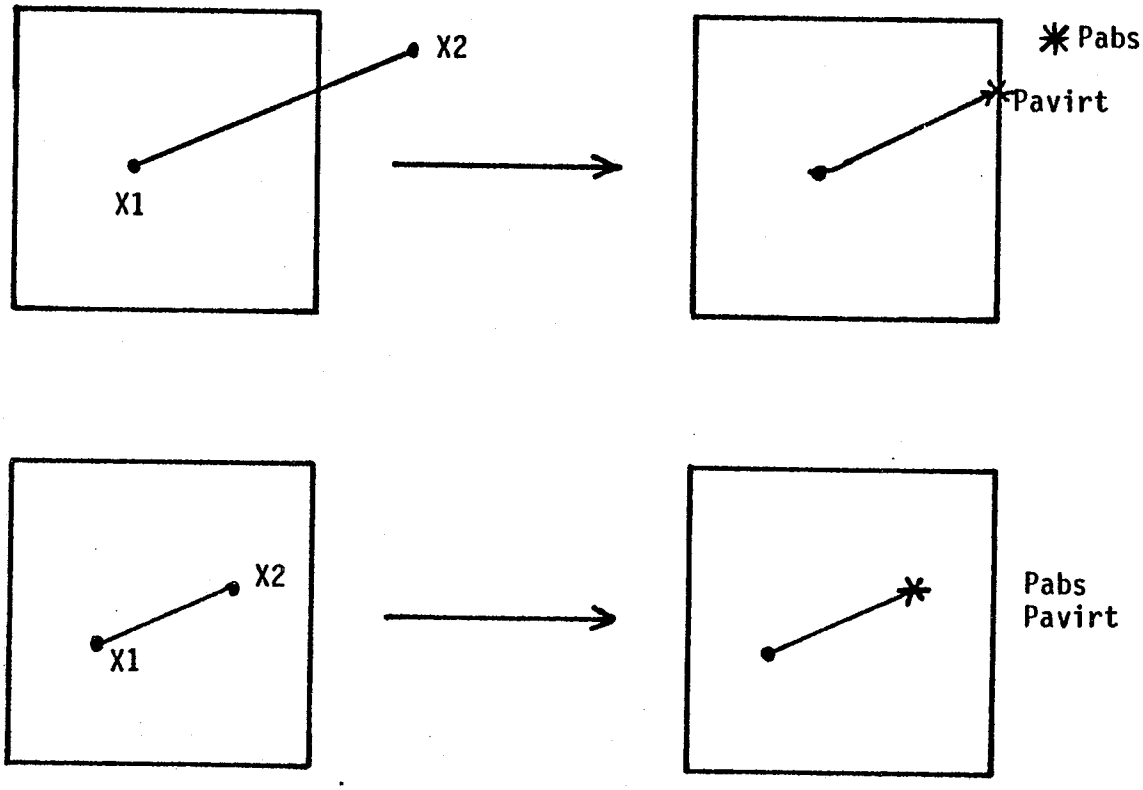


Figure III.6. : Traitement des segments

Notons enfin que lorsqu'une suite de segments "sort" de la fenêtre, on termine la commande en cours; puis, lorsque l'on "rentre" à nouveau dans la fenêtre, on engendre une nouvelle commande élémentaire, avec positionnement en absolu de la première coordonnée. Ainsi, toutes les commandes virtuelles élémentaires commencent par la donnée d'une position absolue dans le repère écran, ce qui évite de tenir à jour une "position courante du faisceau virtuel".

III.2.3.3. Affichage de texte

Lors de l'invocation de la primitive EDITER, l'ordre élémentaire est enregistré de la même manière que pour la primitive AFFICHER et la chaîne de caractères est stockée dans l'image virtuelle.

Dans la console graphique virtuelle, nous n'avons rien dit sur la taille des caractères, ce qui veut dire concrètement que nous n'avons pas muni, par logiciel, la console virtuelle d'un générateur de caractères.

Nous utilisons donc les générateurs fournis par le matériel.

En conséquence, on ne peut pas effectuer de découpage sur les texte car au niveau de l'écran virtuel, on connaît seulement la position du premier caractère. Cette position initiale sera fournie à l'écran réel, et, à partir de celle-ci, le générateur inclus dans le matériel tracera des caractères dont on ne connaît pas les dimensions. La taille fixe des caractères engendrés par matériel nous interdit de leur faire subir une homothétie semblable à celle effectuée sur les points et les segments (voir figure III.7, page 76).

III.2.3.4. Gestion des zones réservées

L'image virtuelle dispose d'une table de zones réservées contenant les renseignements nécessaires au positionnement correct du curseur dans la zone sur l'écran réel, et au stockage ou à la récupération des valeurs contenues dans la zone correspondante de la liste d'affichage virtuelle.

On conserve les renseignements suivants :

- numéro d'identification de la ligne à laquelle appartient la zone,
- numéro interne de la zone,
- coordonnées dans le repère écran virtuel du début de la ligne,
- nombre de caractères contenus dans la ligne, avant la zone réservée,
- nombre de caractères de la zone,
- pointeur vers la zone de texte correspondante dans la liste d'affichage.

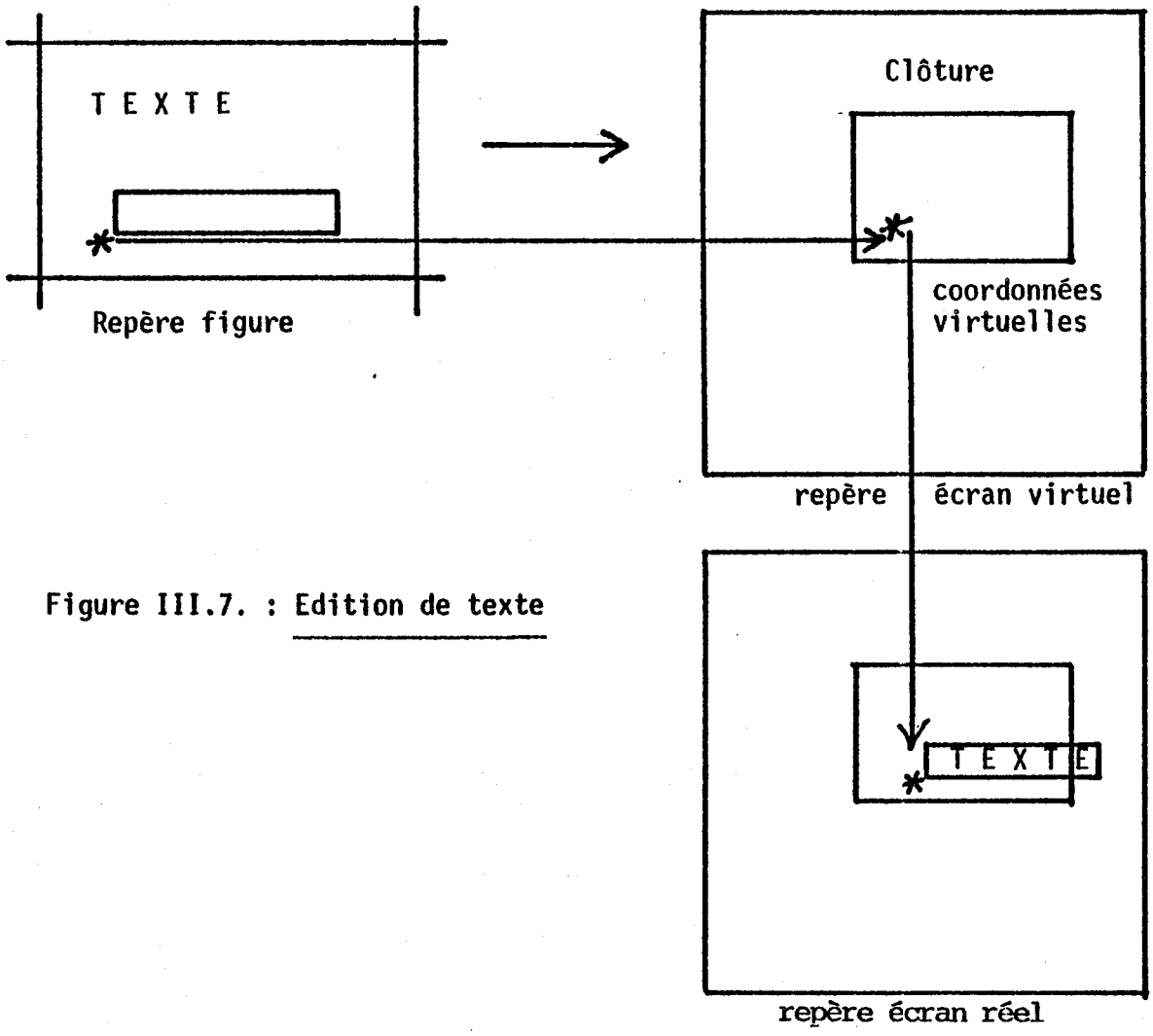


Figure III.7. : Edition de texte

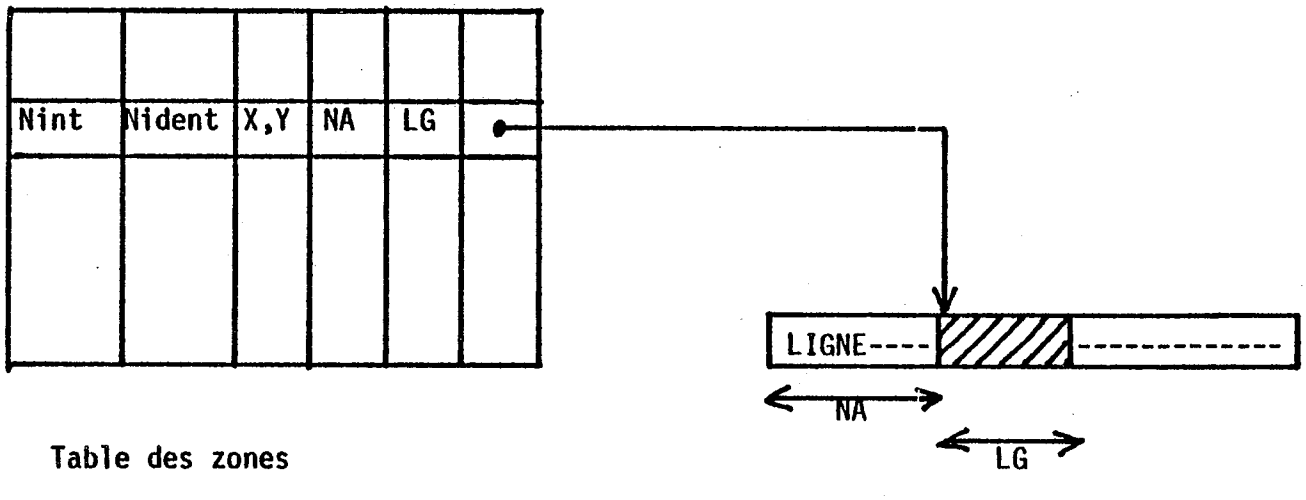


Figure III.8. : Zones réservées

III.2.3.5. Effacement

Lors de l'invocation de la primitive EFFACER, la figure concernée est effacée dans l'image virtuelle par mise à zéro du code opération de tous les éléments la constituant et réinitialisation des pointeurs.

Lorsqu'on utilise un dispositif virtuel, on ne connaît rien sur le terminal réellement utilisé. La réalisation d'un effacement sélectif se fait donc par effacement de l'écran réel et régénération de la nouvelle image à partir de la liste d'affichage virtuelle. Ainsi la conservation et la mise à jour de l'image réelle est assurée quelque soit le matériel utilisé (avec ou sans mémoire d'entretien). Une telle approche conduit à essayer de minimiser les mises à jour de l'écran réel. C'est pourquoi, lors d'une demande d'effacement, celle-ci est exécutée au niveau virtuel, mais n'est pas répercutée systématiquement sur l'écran utilisé. En fait, nous avons estimé que l'utilisateur doit avoir une vision exacte de la situation dans deux cas :

- quant il veut ajouter quelque chose sur l'écran (affichage)
- quand il veut orienter le déroulement de son programme en fonction de la situation actuelle (menu).

Les effacements enregistrés dans la liste virtuelle sont donc répercutés sur l'écran réel lors de la rencontre d'une fonction d'affichage ou de menu.

Le logiciel prend donc en charge une sorte de "mise à jour automatique". Il est dès lors inutile de mettre une procédure de mise à jour à la disposition des utilisateurs. Cette solution a été adoptée pour limiter le nombre de régénérations de l'image (qui sont très lourdes dans le cas des tubes à mémoire surtout lorsque le débit de la liaison calculateur-terminal est faible).

III-3- LES PRIMITIVES DE DIALOGUE DE LA PREMIÈRE GÉNÉRATION DE GRIGRI

Les primitives de dialogue, au nombre de six, correspondent aux quatre types d'actions demandées par les applications :

- introduction de valeurs,
- introduction de coordonnées,
- identification,
- menu.

III.3.1. Description des primitives de dialogue

III.3.1.1. Introduction de valeurs

L'introduction de valeurs est réalisée par l'intermédiaire de trois primitives, qui permettent l'introduction d'une chaîne de caractères à partir du clavier alphanumérique, et assurent le codage sous forme d'une valeur entière ou réelle, si cela est nécessaire.

L'introduction d'un entier, d'un réel, ou d'une chaîne de caractères est activée respectivement à l'aide des primitives :

NVAL (Nfig, Nligne, Nzone)
RVAL (Nfig, Nligne, Nzone)
CHAINE (Nfig, Nligne, Nzone)

Les paramètres sont les suivants :

- le numéro de la figure dans laquelle est engendrée la ligne qui sert de support à l'introduction,
- le numéro de la ligne à compléter,
- le numéro d'ordre, dans la ligne, de la zone.

III.3.1.2. Identification d'un objet

La désignation et l'identification d'une partie d'un dessin peuvent être réalisées physiquement à l'aide de dispositifs divers, mais l'activation se fait par une seule primitive :

IDENTIFIER (Nfig)

Le paramètre Nfig est une variable de type entier qui contient lors de l'appel :

- le numéro de figure dans laquelle est engendré l'objet à identifier. (On peut ainsi rendre certains objets "sensibles" ou non à la désignation),
- 0 si toutes les figures peuvent participer à l'opération.

Au retour, le paramètre contient le numéro de figure à laquelle appartient l'objet qui a été désigné. La fonction prend pour valeur l'entier qui représente le numéro d'identification de cet objet. Il faut noter que plusieurs objets peuvent être engendrés avec le même numéro d'identification et que l'on peut ainsi identifier des classes d'objets.

III.3.1.3. Collecte de coordonnées

L'entrée d'un ensemble de coordonnées peut être réalisée grâce aux divers dispositifs associés au terminal utilisé (photostyle, réticule, clavier alphanumérique, tablette, etc...) et est activée à l'aide de la primitive :

POSITION (Nfig, Nombre, tabx, taby)

Nfig est une variable entière qui contient un numéro de figure. Les coordonnées introduites seront évaluées dans l'espace associé à cette figure.

Nombre est une variable entière contenant lors de l'appel :

- le nombre maximum de coordonnées à récupérer,
- 0, si ce nombre n'est pas connu a priori et que l'on souhaite récupérer toutes les coordonnées entrées jusqu'à un signal de fin.

Au retour "nombre" contient le nombre de points effectivement introduits.

Tabx, Taby sont des identificateurs de tableaux de réels, destinés à recevoir les coordonnées introduites. Ces tableaux doivent être indicés à partir de 1. Quelque soit le type de la figure, les coordonnées retournées sont évaluées en absolu.

III.3.1.4. Utilisation d'un menu

Le choix d'une action parmi une liste d'actions, afin d'influer sur le déroulement du programme est activé à l'aide de la primitive :

MENU (liste-de-fonctions)

La "liste-de-fonctions" est une constante littérale ou une variable de type chaîne, spécifiant la liste des fonctions parmi lesquelles l'utilisateur devra faire son choix.

Chaque fonction est désignée par un numéro et, facultativement, par un nom. Si ce nom est mentionné, il sera affiché sur l'écran et l'opérateur pourra le désigner directement ; sinon la fonction sera accessible uniquement par son numéro. Au retour, la fonction MENU prend pour valeur, soit le numéro entré directement par l'utilisateur, soit le numéro associé au nom de la fonction désignée sur l'écran.

III.3.2. Gestion de la console virtuelle

La console virtuelle intervient relativement peu dans les primitives de dialogue, du fait que l'on n'a pas défini de dispositifs de dialogue virtuels.

La prise en charge du dialogue se fait essentiellement au niveau de l'interpréteur, pour utiliser au mieux le matériel.

III.3.2.1. Introduction de valeurs

Lors d'une introduction de valeurs, il faut passer certains renseignements à l'interpréteur pour qu'il puisse afficher le curseur en bonne place sur l'écran.

Lorsque l'entrée est terminée, l'interpréteur renvoie la chaîne de caractères qui a été frappée au clavier. Cette chaîne est stockée dans l'image virtuelle, en vue des regénérations ultérieures de l'image. La console virtuelle retourne à l'application la valeur entière, ou réelle après codage de la chaîne de caractères (cf. figure III.9.).

III.3.2.2. Désignation d'un objet

Lors de l'invocation de la primitive IDENTIFIER, l'interpréteur active le dispositif (photostyle, réticule, etc...) capable de prendre en charge cette fonction. L'interpréteur renvoie un couple de coordonnées écran ; suivant le type du terminal, on se donne un "rectangle d'erreur" plus ou moins grand autour de cette position ; on passe alors en coordonnées écran virtuel. La recherche de l'élément désigné va se faire dans la liste d'affichage virtuelle (cf. figure III.10.)

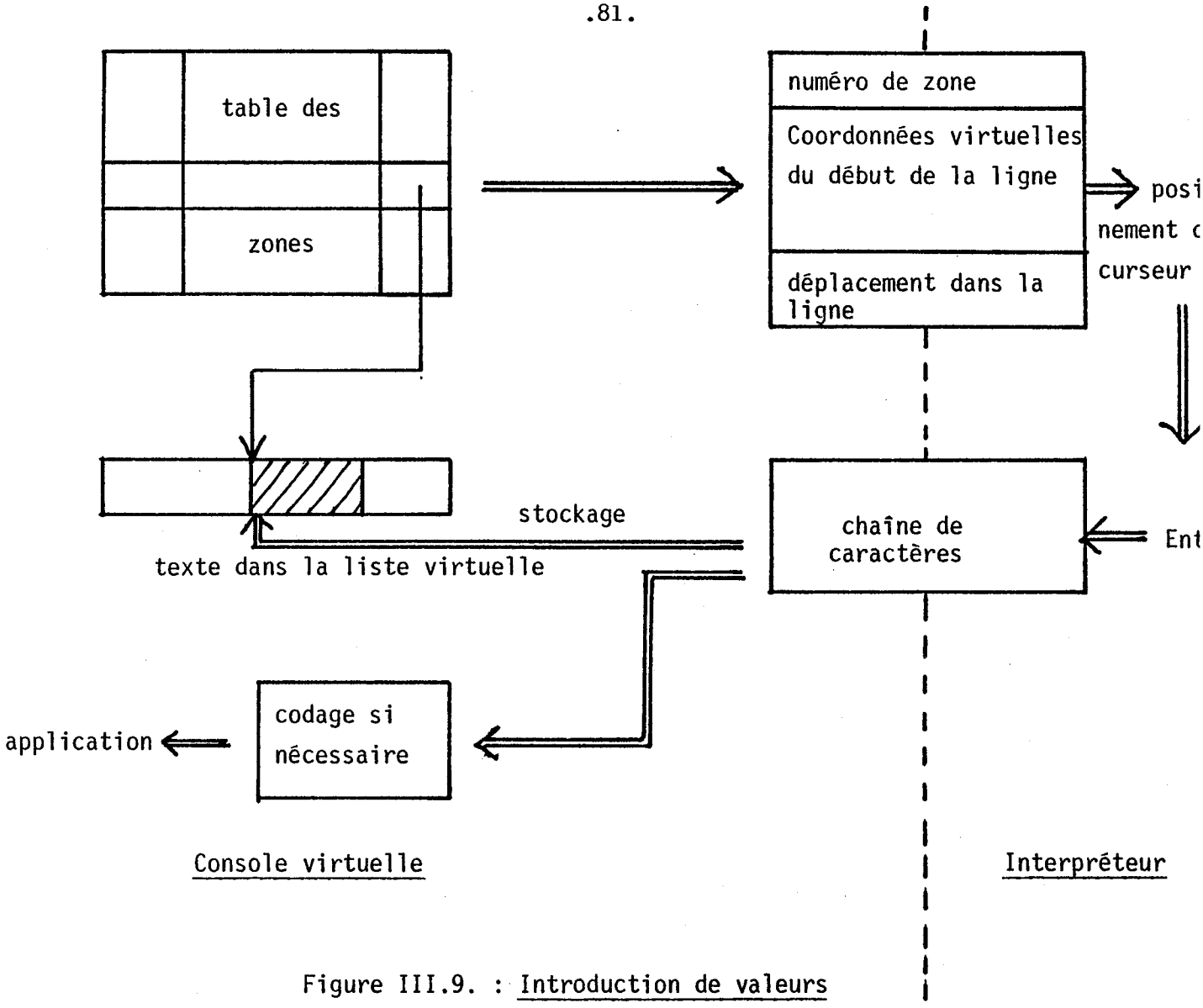


Figure III.9. : Introduction de valeurs

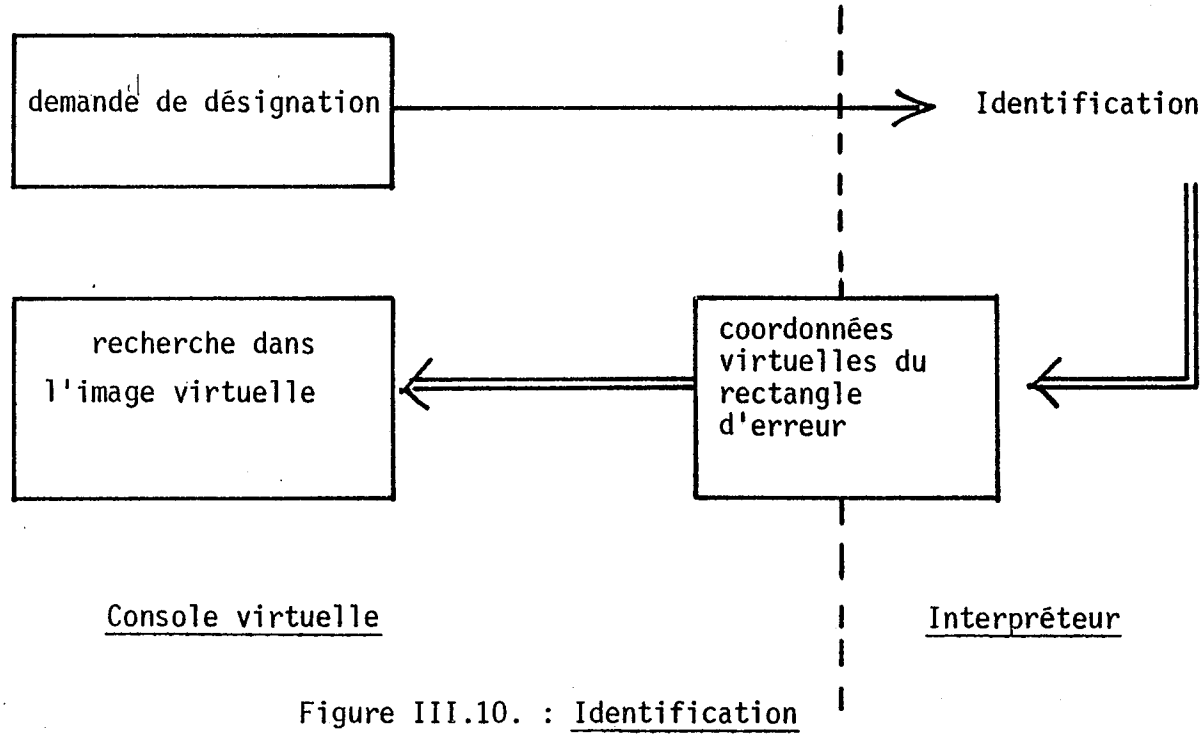


Figure III.10. : Identification

L'identification, au niveau de l'image virtuelle, est réalisée par la recherche d'un élément qui apparaît dans la "fenêtre d'erreur". Cette approche pose un problème en ce qui concerne les textes ; en effet, ceux-ci étant engendrés par le générateur fourni par le matériel, on ne connaît pas leurs dimensions dans la console virtuelle. Pour les caractères, on est donc obligé de revenir au système de coordonnées de l'écran réel et d'étudier la position de la "fenêtre d'erreur" réelle par rapport au rectangle englobant le texte écrit sur l'écran.

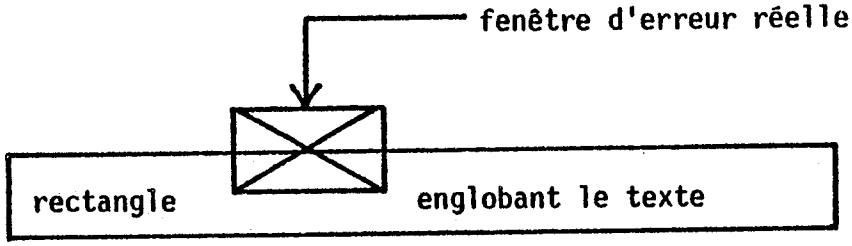


Figure III.11. : Identification d'une ligne de texte

III.3.2.3. Collecte de coordonnées

Lors de l'invocation de la primitive POSITION, l'interpréteur prend en charge l'activation des divers dispositifs disponibles. La console virtuelle assure la réservation de deux zones sur l'écran pour permettre de rentrer des coordonnées à partir du clavier alphanumérique, et l'affichage sur l'écran des points rentrés par l'un quelconque des dispositifs. Ceci est réalisé par l'intermédiaire d'une "figure système" définie dans la console virtuelle, et dans laquelle sont enregistrés les ordres d'affichage des points collectés et de leurs coordonnées. La "figure système" apparaît donc comme suit :

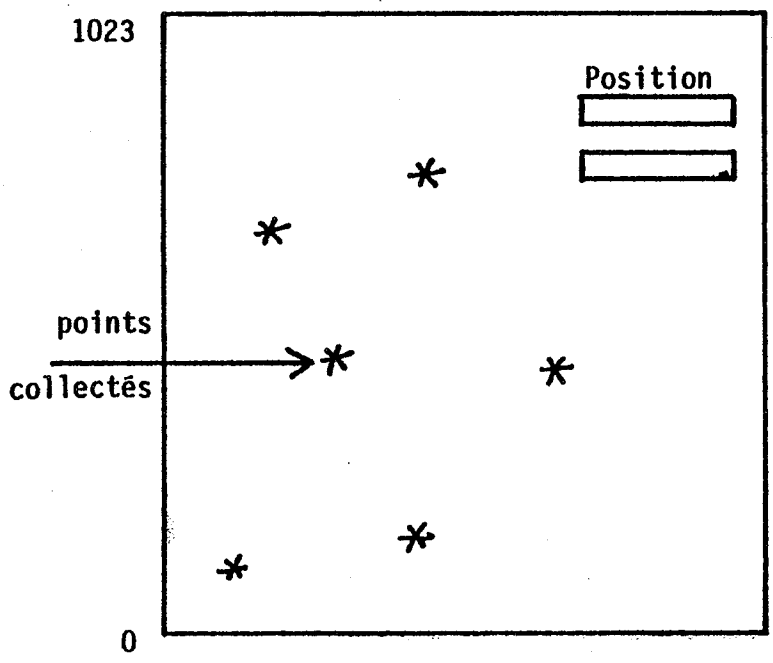


Figure III.12. : Figure système pendant l'exécution de "POSITION"

Lors de la fin de collecte, les coordonnées des points, évaluées dans le repère de la figure système, indépendamment du système de coordonnées de l'écran réel, sont retournées à l'application après passage dans le système de coordonnées de la figure concernée.

La "figure système" est ensuite effacée de l'image virtuelle. Au cours de l'exécution de la primitive POSITION, les actions possibles à chaque instant sont :

- introduction d'un point,
- effacement du dernier point collecté,
- effacement de tous les points,
- arrêt de la collecte et retour à l'application.

Ces actions sont répercutées au niveau de la console virtuelle par effacement des points dans la liste virtuelle et mise à jour des zones dans la liste virtuelle des coordonnées du dernier point collecté.

III.3.2.4. Menu

Dans ce cas, l'indépendance par rapport au matériel est également acquise par l'intermédiaire de la "figure système" dans la liste virtuelle. Les noms des fonctions sont affichés par rapport au système de coordonnées de la figure système, grâce à la fonction EDITER.

L'interpréteur renverra donc soit un numéro de fonction (entier compris entre 1 et 32), soit une ordonnée évaluée dans le repère de la figure système (indépendante du matériel) selon que l'opérateur donnera un numéro on désignera un nom sur l'écran.

La recherche de la fonction désignée se fera en comparant l'ordonnée virtuelle retournée aux positions des différents noms affichés dans l'image virtuelle. (cf. figure III.13.)

A la fin de l'éxécution de la fonction, la "figure système" est effacée de la liste virtuelle.

Notons que l'invocation de la primitive MENU provoque la mise à jour de l'écran réel, par rapport à la liste virtuelle, si les effacements ont été enregistrés précédemment. (cf. figure III.14.)

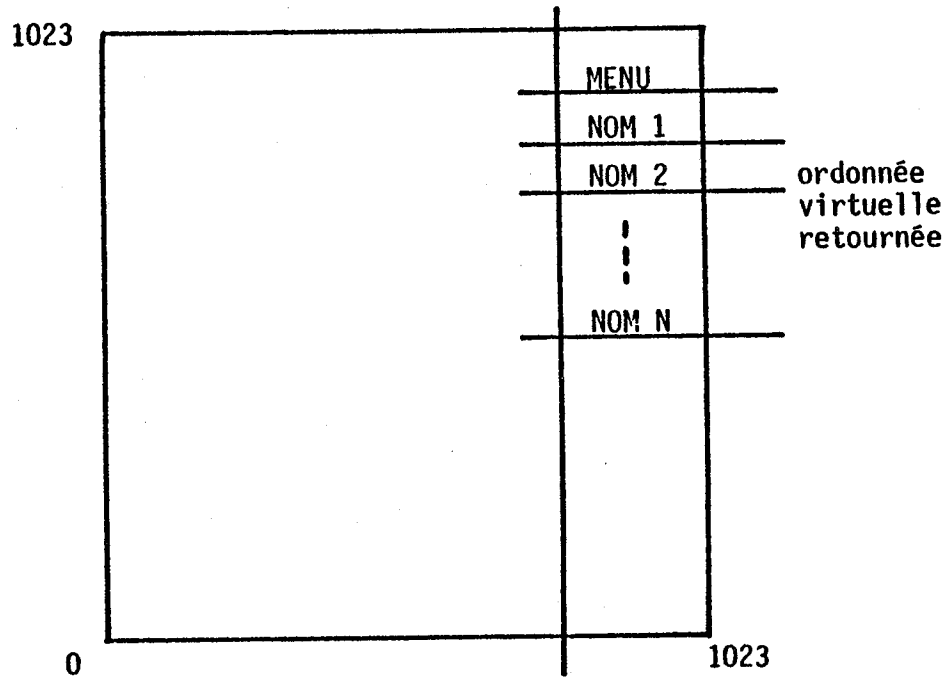


Figure III.13. : Menu affiché dans la figure système de la console virtuelle

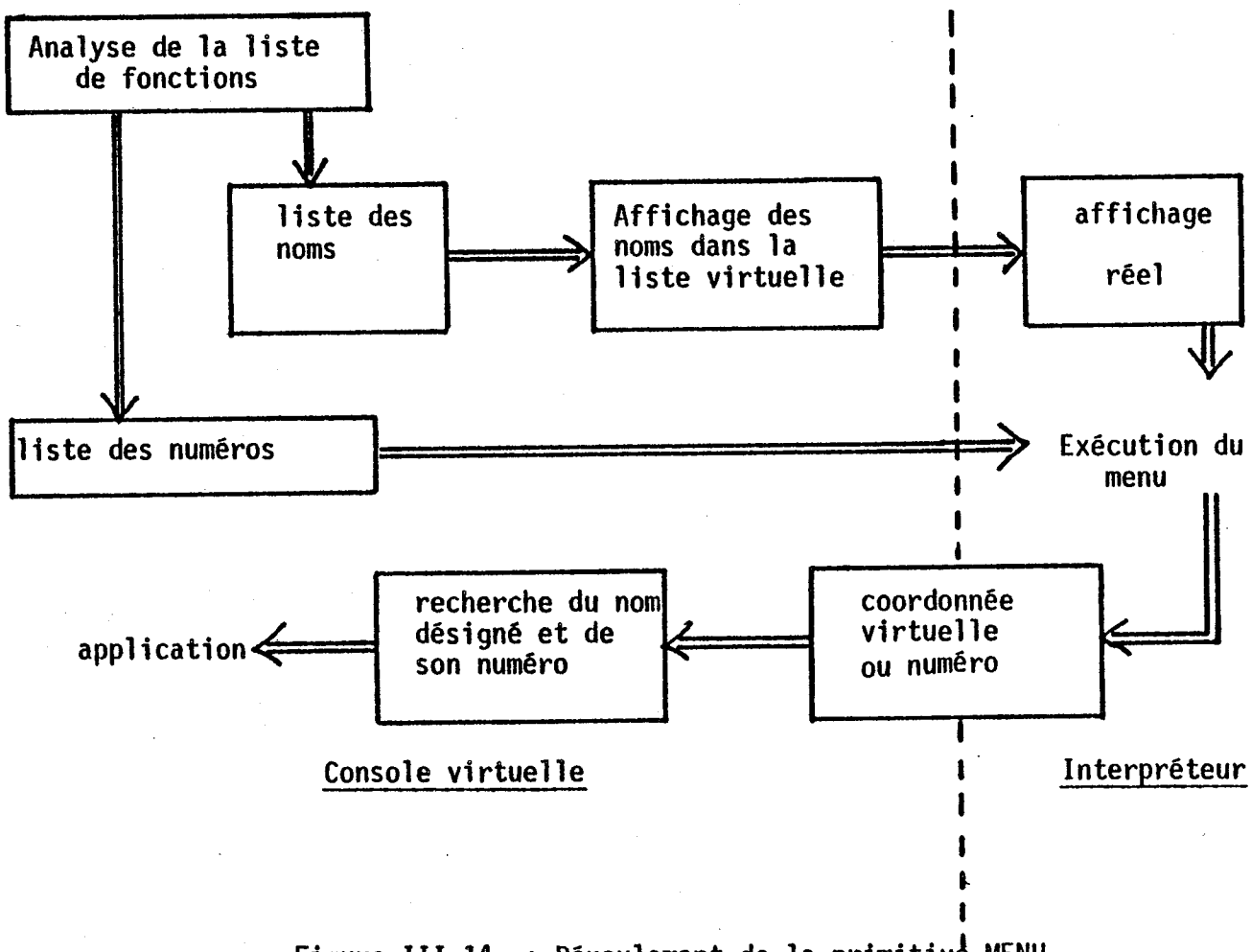


Figure III.14. : Déroulement de la primitive MENU

III-4- ORGANISATION D'ENSEMBLE DU LOGICIEL

Le logiciel constituant la première génération de GRIGRI est écrit en FORTRAN, pour être facilement transportable.

La solution adoptée est d'associer un sous-programme Fortran à chaq primitive du logiciel.

Toutes les données décrivant la console virtuelle sont rassemblées dans un "common" ce qui les rend accessibles à tous les modules du logiciel.

III.4.1. Les classes de modules

Les sous-programmes du logiciel peuvent être regroupés en trois cla ses, suivant leur niveau d'indépendance vis-à-vis du matériel :

Niveau 1	Modules "primitives"
Niveau 2	Modules "Gestion console virtuelle"
Niveau 3	Modules "Interpréteurs"

Les modules "primitives" correspondent en fait aux sous-programmes qui sont act vés directement par les primitives d'appel du logiciel contenues dans le progra me d'application. Ces modules "primitives" font appel aux modules de Gestion de la console virtuelle et aux modules interpréteurs.

Les modules de gestion de la console virtuelle assure la gestion des différentes tables et de la liste d'affichage virtuelle.

Les modules interpréteurs, sur réception d'un code indépendant du ma tériel, assurent la gestion du terminal réel et, renvoient des réponses indéper dantes du matériel lors de l'invocation de fonctions de dialogue.

L'organisation d'ensemble du logiciel est donc la suivante :

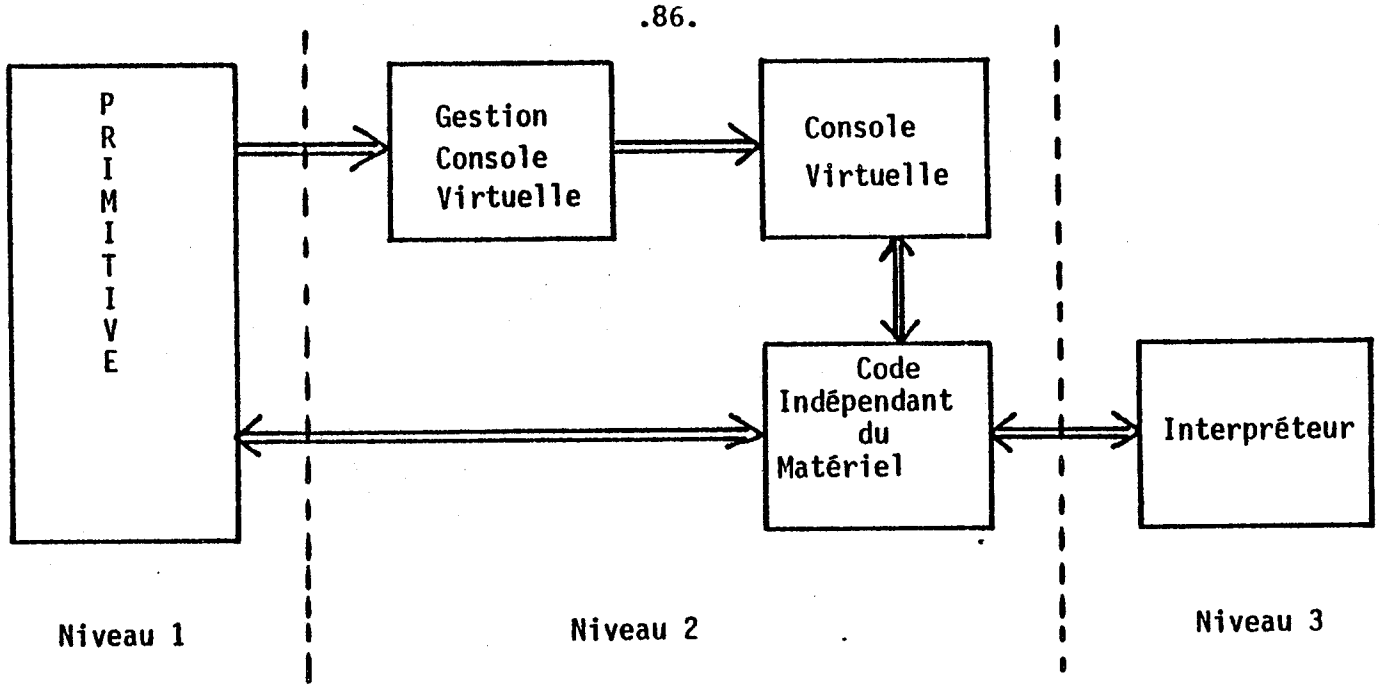


Figure III.15. : Les classes de modules

La distinction du niveau "console virtuelle" et du niveau "interpréteur" correspond au principe même de base de ce logiciel. Notons que les traitements au niveau virtuel et au niveau réel sont complètement déconnectés. Ce qui veut dire que l'écran réel n'est pas toujours l'image exacte de la liste d'affichage virtuelle. En particulier on minimise le nombre de mise à jour de l'écran, ce qui est indispensable lors de l'utilisation d'un tube à mémoire.

Reste à définir le code intermédiaire, indépendant du terminal, des commandes passées à l'interpréteur.

III.4.2. Le code intermédiaire

Ce code intermédiaire est en fait le code de commande du processeur graphique de la console virtuelle. En effet, la console virtuelle complète peut être schématisée par la figure III.16.

Les commandes retenues sont les suivantes :

- initialisation,
- affichage,
- effacement de l'écran,
- demande d'entrée de valeur,
- demande de menu,
- demande d'identification,
- demande de collecte de coordonnées,
- édition alphanumérique dans une zone réservée.

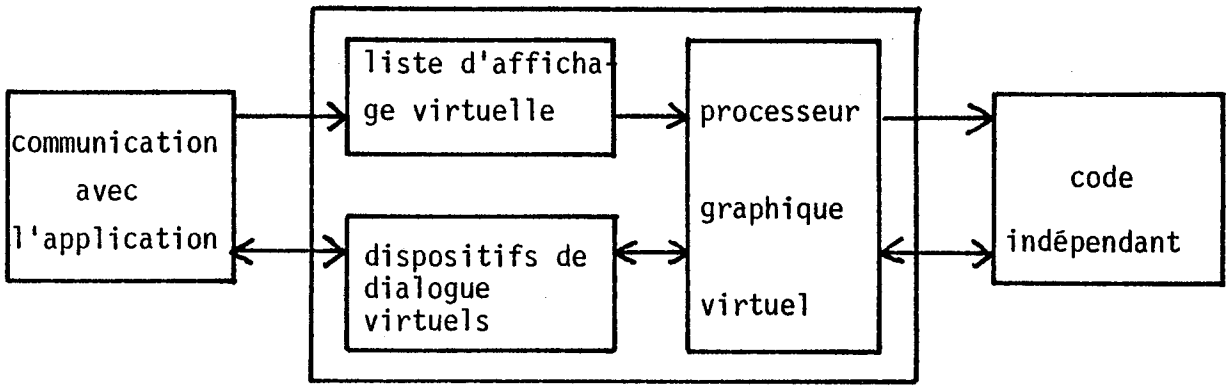


Figure III.16. : Console virtuelle

Ces commandes sont indépendantes du matériel et permettent la réalisation de toutes les primitives du logiciel. Lorsqu'on effectuera une mise à jour, on enverra tout d'abord une commande d'effacement de l'écran, puis on activera l'interpréteur d'affichage pour tous les éléments graphiques contenus dans la liste d'affichage virtuelle. Les paramètres de ces commandes doivent également être indépendants du matériel. La liste des opérandes est donnée dans la figure III.17.

Dans la fonction d'identification, l'interpréteur joue deux rôles :
- il renvoie le rectangle d'erreur, résultat de la désignation,
- il étudie les lignes de texte pour savoir si elles ont été désignées ou non.

En effet, pour rester indépendant du matériel, on doit rejeter l'étude des textes au niveau de l'interpréteur, car celle-ci doit être effectuée dans le système de coordonnées de l'écran réel, pour connaître la taille exacte des caractères. Ainsi, lors de la recherche de l'élément désigné, on étudie les points et segments dans l'image virtuelle, mais pour les textes, on fait appel à l'interpréteur qui les étudiera dans l'image réelle.

Lors d'un affichage et lors d'une demande de menu, on met à jour l'image si des effacements ont été enregistrés. Si oui on envoie successivement les commandes :
- effacement écran,
- affichage (pour tous les éléments de la liste virtuelle),
- affichage ou menu.

Commande	Paramètres d'appel	Paramètres de retour
Intitialisation	x	x
Affichage	adresse de l'élément graphique à interpréter, dans la console virtuelle	x
Effacement écran	x	x
Edition alphanumérique	- numéro de la zone à remplir - chaîne de caractères à éditer	x
Entrée de valeur	- numéro de la zone concernée	- chaîne de caractères entrée
Menu	- tableau des numéros autorisés	- soit un numéro compris entre 1 et 32 - soit une coordonnée virtuelle comprise entre 100 et 1023
Identification	0	- coordonnées virtuelles comprises entre 0 et 1023 du rectangle d'erreur
	adresse dans la console virtuelle de l'ordre texte à étudier	- résultat : désigné / non désigné
Collecte de coordonnées	- nombre maximum de points à collecter	- nombre de points collectés - coordonnées virtuelles comprises entre 0 et 1023 des points collectés

Figure III.17 : Code intermédiaire

Les interpréteurs reçoivent ce code intermédiaire et prennent en charge les commandes en essayant d'utiliser au mieux le matériel dont il dispose. Mais : est évident que le niveau relativement bas de ce code pénalisera certainement les matériels évolués.

Nous allons étudier maintenant la réalisation de deux types d'interpréteurs :

- un interpréteur pour tube à mémoire,
- un interpréteur pour console à mémoire d'entretien.

III-5- EXEMPLE D'INTERPRÉTEUR POUR TERMINAL À TUBE MÉMOIRE

L'interpréteur décrit ici était prévu pour piloter un terminal Tektronix 4010. En fait, il a permis la prise en charge de plusieurs écrans à tube mémoire : Tektronix 4010, 4012, 4014, 1015 [T2] , [T3].

III.5.1. Description de terminal Tektronix

Le Tektronix 4010 [T3] est un terminal à tube mémoire, simple, donc il ne dispose pas de possibilité d'effacement sélectif. L'écran est rectangulaire, et ses dimensions sont :

- longueur : 191 mm
- hauteur : 143 mm

Le nombre de points adressables est 1024 × 1024. Mais en fait, l'écran étant rectangulaire, les points affichables ont des coordonnées telles que :

$$0 \leq X \leq 1023$$
$$0 \leq Y \leq 780$$

Ces coordonnées sont entières.
En ce qui concerne les caractères, il existe une seule taille qui permet d'écrire 74 caractères par ligne et 35 lignes sur l'écran.
Les dispositifs de dialogue sont :

- un clavier alphanumérique,
- un réticule.

Le réticule retourne les renseignements suivants :
- abscisse du point repéré,
- ordonnée du point repéré,
- caractère associé.

Le caractère associé est le caractère frappé au clavier alphanumérique pour désactiver le réticule.

Le terminal dispose de trois modes de fonctionnement :
- mode alphanumérique (lecture et écriture de caractères),
- mode graphique (affichage de points et segments)
- mode d'entrée graphique (collecte d'un couple de coordonnées).

Les fonctions élémentaires, permettant de gérer les interruptions et les entrées entrées/sorties, sont les suivantes :

- positionnement du faisceau éteint,
- déplacement du faisceau allumé,
- passage en mode alphanumérique,
- activation du signal sonore,
- affichage de caractères,
- effacement de l'écran,
- entrée de caractères au clavier,
- activation du réticule.

Les fonctions de positionnement et de déplacement assurent le passage du terminal en mode graphique, l'activation du réticule provoque le passage du terminal en mode d'entrée graphique.

III.5.2. Mise en page de l'écran

L'interprétation pour affichage, d'un élément de la liste de visualisation virtuelle est simple ; l'interpréteur assure uniquement le passage du code virtuel au code réel. Le seul problème d'affichage concerne la mise en page de l'écran réel. En effet, l'écran virtuel est carré (0,0,1023,1023). Or l'écran

réel est rectangulaire. On a décidé de faire correspondre à l'écran virtuel l partie (0,0,780,780) de l'écran réel. La mise en page est donc la suivante :

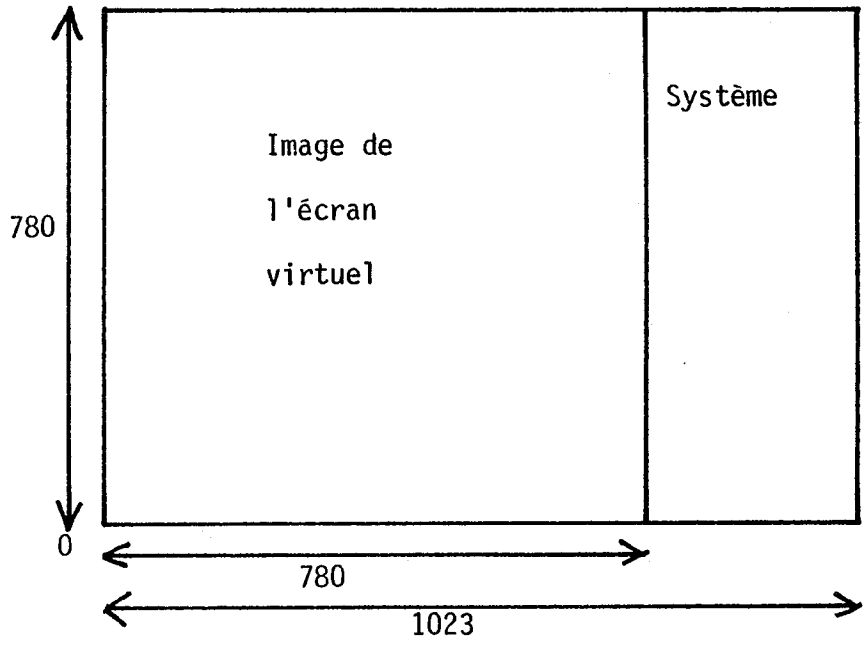


Figure III.18. : Mise en page d'un écran rectangulaire

Ainsi, le contenu de l'image virtuelle n'est pas déformé. Mais cette solution diminue la taille de l'image et sous-exploite l'écran.

On utilise la zone inaccessible à l'utilisateur pour afficher les messages systèmes (message d'erreur, menu, etc...).

III.5.3. Réalisation et mise en oeuvre du dialogue

La mise en oeuvre du dialogue est assez lourde sur un tel terminal, car on dispose de peu d'outils de dialogue, en particulier, il n'y a pas de c vier de fonctions.

III.5.3.1. Les entrées de valeurs

Pour des raisons de facilité de mise en oeuvre, la table des zones décrite précédemment est stockée au niveau de l'interpréteur. Connaissant le numéro de zone, on connaît :

- les coordonnées virtuelles du début de la ligne,
- le nombre de caractères qui précèdent la zone dans la ligne (Navant).

Ces renseignements permettent de placer le curseur dans l'écran réel.

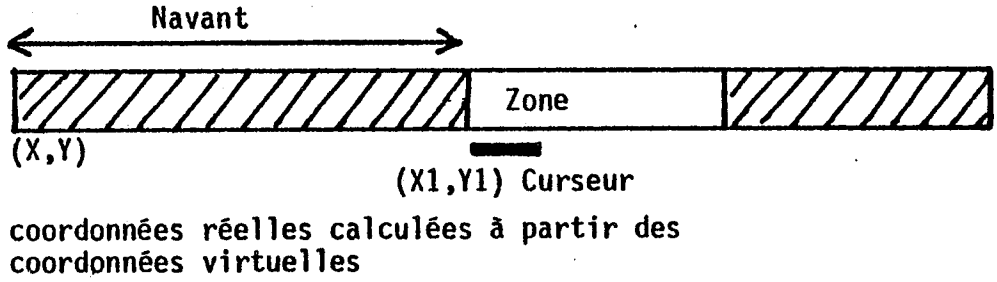


Figure III.19. : Position du curseur sur l'écran

Les coordonnées du curseur sont données par les formules suivantes :

$$X1 = X + \frac{1023 \times Navant}{74}$$

$$Y1 = Y$$

quand le faisceau est mis en place, on fait apparaître le curseur en passant en mode alphanumérique et on attend la lecture.

La fin de chaîne est indiquée par un retour chariot.

III.5.3.2. Identification

L'identification est réalisée à l'aide du réticule.

L'opérateur place le réticule sur l'objet qu'il désire identifier et il frappe un caractère au clavier alphanumérique pour désactiver le réticule

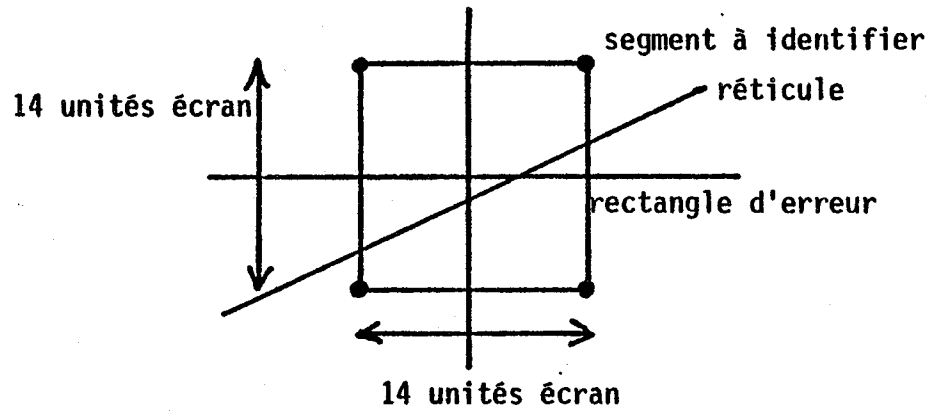


Figure III.20. : Identification d'un segment

On retourne les coordonnées, dans le repère écran virtuel, du rectangle d'erreur.
Sur l'écran réel, la fenêtre d'erreur est un carré de 14 unités écrans centré sur le point désigné.

III.5.3.3. Menu

On active le réticule pour le cas où l'opérateur désirerait montrer un nom affiché sur l'écran.

Si c'est le cas, l'opérateur place le réticule sur le nom désiré et frappe un caractère non numérique sur le clavier alphanumérique.

Par contre, si l'opérateur veut envoyer un numéro, il devra taper le premier chiffre du numéro ; ceci fait disparaître le réticule, et le chiffre frappé s'affiche sur l'écran ; puis le logiciel attend le second chiffre que l'opérateur frappera sur le clavier.

On retourne, soit le numéro frappé au clavier, soit l'ordonnée de la position du réticule.

III.5.3.4. Collecte de coordonnées

On utilise le clavier alphanumérique et le réticule pour introduire des couples de coordonnées.

De plus, le caractère frappé pour désactiver le réticule est analysé et permet de choisir à chaque instant parmi les différentes actions possibles.

La collecte de coordonnées est donc mise en oeuvre de la façon suivante :

- à chaque instant, on active le réticule,
- on analyse le caractère frappé :
  - . si c'est le caractère "E", on efface le dernier point,
  - . si c'est le caractère "F", on efface tout le tracé,
  - . si c'est le caractère "S", on arrête la collecte,
  - . si c'est un chiffre, on demande une entrée au clavier.

Dans tous ces cas, la position courante du réticule n'est pas prise en comp

- Si le caractère frappé n'est, ni un chiffre, ni les lettres "E", ou "F", ou "S", la position actuelle du réticule est prise en compte et le point est stocké.
- Dans le cas où le caractère est un chiffre, ce chiffre est affiché sur l'écran et considéré comme premier chiffre de l'abscisse d'un point. Le logiciel attend alors les chiffres suivants de l'abscisse, puis ceux de l'ordonnée qui seront frappés sur le clavier alphanumérique.

III.5.4. Gestion de la mise à jour

L'écran est mis à jour par rapport à la liste virtuelle, lorsque l'on trouve une primitive d'affichage ou de menu. Ceci permet de minimiser les regénérations de l'image qui coûtent cher et sont désagréables.

En ce qui concerne les zones réservées remplies par édition alphanumérique ou entrée de valeur, il y a évidemment superposition des caractères car on a jugé qu'il était impossible de faire la mise à jour à chaque fois. Quand l'image est reconstruite, les zones sont réinitialisées avec les valeurs qu'elles contenaient lors de la dernière opération et qui sont stockées dans la liste d'affichage virtuelle.

De même, les messages systèmes ne disparaissent de l'écran que lors de la mise à jour suivante.

III.5.5. Prise en charge des terminaux Tektronix 4014-4015

Cet interpréteur permet de piloter les terminaux Tektronix 4010, 4012, mais aussi 4014, 4015.

En effet, ces derniers acceptent des coordonnées comprises entre 0 et 1023, donc l'affichage se fait bien sur la totalité du plus grand carré affichable.

Par contre, ces terminaux disposent de possibilités supplémentaires :
- quatre tailles de caractères,
- tracé de segments en pointillé, tireté court et long et trait mixte.

Ces options ne sont pas prévues au niveau du logiciel, ni au niveau de la console virtuelle, il n'est donc pas possible de les utiliser.

Nous trouvons là un exemple de sous-emploi du matériel, dû à une définition trop restrictive des possibilités de la console graphique virtuelle Sur le terminal Tektronix 4015 nous n'utilisons pas non plus l'option d'affichage non permanent car il faudrait entretenir l'image et pendant ce temps, le calculateur ne pourrait rien faire d'autre.

III-6- EXEMPLE D'INTERPRÉTEUR POUR TERMINAL À MÉMOIRE D'ENTRETIEN

L'interpréteur décrit ici a permis la prise en charge d'un terminal IBM 2250 [T1].

III.6.1. Description du terminal IBM 2250

Le terminal IBM 2250 est un terminal disposant d'une mémoire d'entretien de 8K octets.
L'écran est un carré de 30 cm × 30 cm
Le nombre de points affichables est 1024 × 1024
Les possibilités du processeur sont les suivantes :

- affichage de points, ou de segments,
  en absolu ou en incrémentiel
- affichage de caractères de deux tailles
  en mode protégé ou non protégé.

Les dispositifs de dialogue sont au nombre de quatre :

- clavier alphanumérique,
- clavier de 32 touches de fonctions,
- photostyle,
- tablette.

La liaison calculateur-terminal est assurée par un canal.

Les fonctions élémentaires utiles pour gérer les interruptions et les programmes canaux sont les suivantes :

- initialisation du terminal et de la mémoire,
- positionnement du faisceau éteint,
- choix d'un mode de fonctionnement pour le processeur graphique,
- écriture d'une suite de coordonnées dans la mémoire d'entretien,
- écriture d'une suite de caractères dans la mémoire d'entretien,
- activation du signal sonore,
- insertion du curseur,
- lecture d'une zone dans la mémoire d'entretien,
- attente d'une interruption parmi une liste d'interruptions autorisées,
- autorisation d'une liste de touches de fonctions,
- balayage de l'écran par un rideau de lettres pour repérer un point éclairé sur l'écran.

Lorsqu'une interruption se produit, on récupère le type de celle-ci (clavier, touche, photostyle ou tablette) et les renseignements qui lui sont attachés.

III.6.2. Gestion de l'affichage

L'écran est utilisé dans sa totalité, car il y a correspondance exacte entre l'écran virtuel et l'écran réel. La mémoire d'entretien est gérée de la façon la plus simple : pour afficher, on range séquentiellement les commandes graphiques et leurs opérandes à partir des adresses basses ; pour effacer, on n'utilise pas la possibilité d'effacement sélectif, car au niveau de l'interpréteur, on ne dispose pas de table de figures ou d'objets.

Dans ce cas, la mémoire d'entretien est utilisée uniquement pour stocker une copie de l'image virtuelle. Lorsqu'il y a effacement, la nouvelle liste virtuelle est entièrement transmise à la mémoire d'entretien (il n'y a donc pas de "trou" dans la mémoire d'entretien).

Notons que l'on utilise une seule taille de caractères.

III.6.3. Mise en oeuvre du dialogue

La mise en oeuvre du dialogue est ici beaucoup plus facile, car le terminal fournit suffisamment de dispositifs de dialogue.

III.6.3.1. Entrée de valeurs

On dispose au niveau de l'interpréteur d'une table des zones réservées qui contient pour chacune d'elle l'adresse dans la mémoire de la zone correspondante. Le positionnement du curseur ne se fait pas par l'intermédiaire d'un couple de coordonnées, mais par insertion du curseur à une adresse donnée de la mémoire d'entretien.

Lorsque l'entrée est terminée, on lit dans la mémoire le contenu de la zone associée.

III.6.3.2. Identification

L'identification d'un objet est réalisée à l'aide du photostyle. Celui-ci fournit l'adresse de l'instruction interrompue dans la mémoire et les coordonnées du point désigné. Le premier renseignement n'est pas utilisable car on ne dispose pas d'une description synthétique de l'image au niveau de l'interpréteur. On retourne donc les coordonnées du point et la recherche se fait dans l'image virtuelle de la même façon que pour les tubes à mémoire.

III.6.3.3. Menu

Pour la mise en oeuvre du menu, on utilise 2 dispositifs :

- le photostyle,
- le clavier de fonctions.

Le photostyle permet la désignation d'un nom sur l'écran. Le clavier de fonctions permet de choisir directement un numéro.

On renvoie donc, soit l'ordonnée du point désigné, soit le numéro de la touche enfoncée.

III.6.3.4. Collecte de coordonnées

Pour la mise en oeuvre de la collecte de coordonnées, on utilise tous les dispositifs :

- le clavier alphanumérique permet d'entrer un couple de coordonnées,
- le photostyle permet de montrer un point quelconque de l'écran balayé par un rideau de lettres,
- la tablette permet de rentrer des suites de points,
- les touches de fonctions sont utilisées pour indiquer le type d'action désiré :
    - utilisation du photostyle
    - utilisation de la tablette
    - fin d'utilisation de la tablette
    - annulation du dernier point,
    - annulation de tout le tracé,
    - arrêt de la collecte.

A chaque instant, les zones réservées aux coordonnées affichent la position du dernier point.

III.6.4. Gestion de la mise à jour

Le logiciel indépendant, c'est-à-dire la console virtuelle, ne connaît pas la mémoire d'entretien du terminal. La gestion de l'écran est donc la même que pour un terminal à tube mémoire. L'image est mise à jour par rapport à la liste virtuelle. Cependant, les possibilités d'effacement sélectif sont utilisés pour tout ce qui est à la charge de l'interpréteur, en mémoire; les messages "système" servant de support à l'interaction sont effacés dès la fin du dialogue. La figure système est également gérée par l'interpréteur, c'est-à-dire que son contenu (messages, liste des noms du menu, points collectés, etc...) est effacé dès que la fonction est terminée. Ceci permet de laisser l'écran "propre" après le dialogue et de ne pas encombrer la console virtuelle avec des ordres inutiles.

III-7- CONCLUSION

Les conclusions tirées ici sont le fruit d'une expérimentation d'une année sur les deux types de terminaux et sur des applications diverses.

III.7.1. Le choix des primitives du logiciel

Le but d'un tel logiciel est, non pas de répondre à une application spécifique, mais de permettre à un maximum de personnes d'utiliser facilement un terminal graphique. Cet objectif a été largement atteint.

En ce qui concerne les primitives d'affichage, on peut cependant faire deux remarques :

- La primitive FIGURE remplit deux fonctions qui n'ont aucun lien logique entre elles. En effet, elle permet, d'une part de définir un espace de visualisation dans un repère de l'utilisateur (fenêtre), d'autre part, elle spécifie les objets à visualiser qui constituent la figure. Ces deux fonctions devraient être disjointes.
- On peut également déplorer la pauvreté des options concernant l'affichage :
  - pas de tracé en tireté ou en pointillé,
  - pas de taille ou d'orientation variable pour les caractères.

III.7.2. L'indépendance par rapport au matériel

Grâce à la définition d'un terminal graphique virtuel, l'indépendance du logiciel par rapport au matériel est parfaitement atteinte, mais les contraintes imposées par une telle approche ne sont pas négligeables.

Notons tout d'abord qu'en ce qui concerne les calculs des coordonnées le logiciel est obligé de faire deux changements de repère :

- passage du repère de l'utilisateur ou repère de l'écran virtuel,
- passage du repère de l'écran virtuel à celui de l'écran réel.

Ces opérations coûtent cher et sont répétées pour chaque point. Dans un tel logiciel, il y a risque de duplication de l'information ; en effet lors de la prise en charge d'un terminal disposant d'une mémoire d'entretien, la sit tion est la suivante :

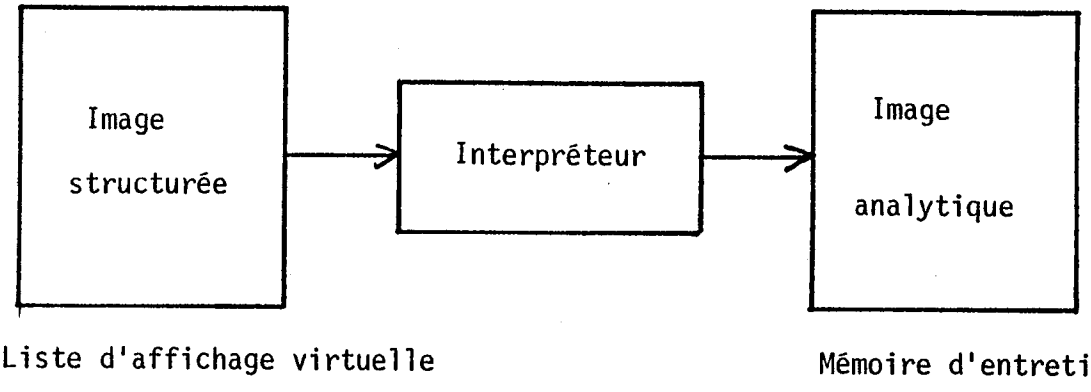


Figure III.21. : Prise en charge d'un terminal avec mémoire d'entreti

Il y a duplication de l'information car les renseignements contenus dans la liste d'affichage virtuelle sont les mêmes que ceux contenus dans la mémoire d'entretien du terminal.

Le principe de base d'un tel logiciel est de faire exécuter toutes les fonctions par la console virtuelle; ce qui veut dire que tous les renseignements (en particulier la table des figures et des objets) seront stockés niveau de l'image virtuelle et concerneront la liste d'affichage virtuelle. conséquence immédiate et que l'on ne pourra pas utiliser les possibilités d'effacement sélectif existant sur les terminaux à mémoire d'entretien.

De plus, ne disposant pas d'une table d'éléments dans l'interpréteur, la seule possibilité de désignation d'un objet se fait par récupératio d'une position et balayage systématique de la liste virtuelle.

Ces deux points, assurant une indépendance parfaite, conduisent en fait à une sous-exploitation des terminaux à mémoire d'entretien. Ceci provi du fait que le code indépendant du matériel reçu par l'interpréteur est une description analytique de l'image qui ne contient aucun renseignement autres que ceux nécessaires à l'affichage pur et simple sur l'écran.

Notons que la désignation des textes pose un gros problème car, ceux-ci étant engendrés par le matériel, on ne connaît pas leur taille au niveau virtuel et la recherche doit se faire dans le système de coordonnées de l'écran réel.

Enfin, le choix du niveau du dispositif virtuel est très délicat ; en effet, on pourrait donner au processeur graphique virtuel des possibilités telles que :

- tracé en tireté ou pointillé,
- génération de caractères de taille et orientation quelconques
- couleur
- etc...

Ceci éviterait de sous employer des terminaux tels que les Tektronix 4015 ; mais en contre partie, lorsque ces options n'existent pas dans le matériel, l'interpréteur devrait les prendre en charge et les simuler lorsque cela est possible.

La valeur d'un tel logiciel réside donc dans le fait qu'il permet d'acquérir une indépendance pratiquement totale vis-à-vis du matériel. Il est particulièrement bien adapté à la prise en charge des terminaux à tube mémoire peu performants. Mais il conduit à une sous-exploitation des terminaux évolués car on est obligé de définir le niveau du dispositif virtuel en fonction des matériels les moins performants (pas de mémoire d'entretien, pas de structuration de l'image au niveau de l'interpréteur, options minimales pour le tracé, etc...).

Ces limitations et ces contraintes nous ont conduits à envisager un autre type de logiciel qui, tout en restant indépendant du matériel, s'adapte à celui-ci et utilise au mieux les possibilités offertes. La deuxième génération de GRIGRI résoud ce problème.

DEUXIÈME PARTIE : LE LOGICIEL GRIGRI

Nous nous intéressons ici à la conception et la réalisation du logiciel graphique GRIGRI. Ce logiciel a été développé pour expérimentation sur les installations du Centre Interuniversitaire de Calcul de Grenoble.

Nous étudierons successivement :

- les primitives du logiciel,

- la réalisation du logiciel,

- la prise en charge des différents types de terminaux.

Enfin, nous donnerons quelques indications pour la prise en charge d'un nouveau terminal.

CHAPITRE IV : LA CONCEPTION DU LOGICIEL GRIGRI

IV-1- LES OBJECTIFS

L'expérimentation d'une première génération du logiciel (cf. Chapitre III) nous a conduits à ajuster et préciser certains objectifs, quant à la définition des primitives offertes aux utilisateurs. Les points principaux sont décrits ci-dessous.

Nous nous plaçons dans la situation où le logiciel graphique est composé de deux parties (cf. Chapitre I) :

- un logiciel de mise en forme,
- un logiciel graphique de base.

C'est à ce dernier que nous nous intéressons ; il ne manipulera pas la structure de données de l'application, mais seulement des données graphiques qui lui seront fournies par les logiciels de mise en forme. Par ce biais, le logiciel de base devra pouvoir supporter un maximum d'applications. La solution réside dans un bon choix des primitives, c'est-à-dire de l'interface entre le logiciel de mise en forme et le logiciel de base.

Les applications pouvant utiliser des dispositifs divers, les primitives devront être indépendantes du matériel ; il faudra donc offrir à l'utilisateur des primitives de définition d'espaces qui le permettent.

Nous abandonnons, dans ce logiciel, la définition d'une console graphique virtuelle, pour nous orienter vers un système capable de s'adapter à son environnement. Ceci a deux conséquences pour la définition des primitives :

- l'utilisateur ne travaille plus sur un écran virtuel,
- les options envisageables ne sont plus limitées par les possibilités de la console virtuelle.

Nous offrirons donc des primitives plus riches pour l'affichage.

Les primitives de dialogue devront être plus puissantes que celles offertes dans la première génération du logiciel (ch. Chapitre III), car celles-ci sont insuffisantes dans certains cas.

Enfin, nous avons constaté que les programmeurs d'application affichent deux types d'informations sur l'écran :

- des informations graphiques (points,segments, ... et textes) constituant le dessin proprement dit,
- des informations à caractère non graphique, qui sont en fait des renseignements fournis à l'opérateur pour le guider, et qui servent de support au dialogue.

Il nous a paru souhaitable d'offrir ces deux fonctions, de façon distincte, aux utilisateurs.
Compte-tenu de l'expérience précécente et de ces objectifs supplémentaires, nous avons défini la génération actuelle du logiciel GRIGRI. Celle-ci est beaucoup plus riche, elle satisfait donc plus de programmeurs d'application, et permet une meilleure utilisation des divers terminaux graphiques.

IV-2- TERMINOLOGIE

Nous voulons préciser ici le sens de certains termes utilisés par le logiciel [R1].

IV.2.1. Les espaces

— Une fenêtre est une portion rectangulaire de l'espace de l'utilisateur. Pour offrir à l'utilisateur les mêmes conditions de travail sur l'écran (limité), que dans son propre espace (illimité), la seule solution consiste à n'afficher qu'une partie du dessin initial sur l'écran. Tout se passe comme si l'utilisateur disposait d'une "fenêtre" sur son espace de travail, et que l'on affiche sur l'écran ce qui apparaît dans la fenêtre. En approchant, ou en recu-

lant, la fenêtre, on provoque des effets de grossissement. En déplaçant la fenêtre, on visualise différentes parties de l'espace de travail de l'utilisateur.

Le problème du découpage d'un dessin consiste à découvrir quelles sont les parties visibles de ce dessin, à l'intérieur d'une fenêtre.

-- Une clôture est une portion rectangulaire de l'écran. Les bords d'une clôture sont parallèles aux bords de l'écran et définis par rapport au système de coordonnées de celui-ci.

On constate que pour afficher ou collecter des points, il est nécessaire de connaître la taille et la position de la fenêtre dans l'espace de l'utilisateur, ainsi que celles de la clôture par rapport à l'écran.

IV.2.2. Les entités graphiques

-- Les éléments de base, constituant les données à visualiser sont de deux types :
- les points (couple de coordonnées),
- les caractères alphanumériques.

-- Une section est un ensemble d'éléments de base.
Pour définir une section, il faut préciser :
- le nombre d'éléments qu'elle contient,
- la liste des abscisses,
- la liste des ordonnées.

Ces coordonnées sont définies dans le repère de l'utilisateur.
Une section de texte sera définie par :
- le nombre de caractères qu'elle contient,
- la chaîne de caractères la composant.

-- Une figure est un ensemble de sections (de points et de texte). Les sections appartenant à une même figure sont définies par rapport au même système de coordonnées chez l'utilisateur.

-- Un dessin est ce qui apparaît sur l'écran à un instant donné. Il peut être composé d'une ou plusieurs figures.

IV.2.3. Autres définitions

-- un mode est un ensemble de caractéristiques graphiques que l'on associe à une section.

Celui-ci permet d'interpréter graphiquement les éléments de la section : précision de la texture, de la couleur, de la taille et de l'orientation des caractères, etc...

-- un marqueur est un symbole alphanumérique que l'on désire attacher à un point repéré par ses coordonnées. Ce peut être :

- un caractère alphanumérique simple,
- un caractère alphanumérique indicé,
- une suite de caractères.

-- un message est une ligne de texte. Il contient des indications données à l'opérateur et des zones réservées à l'édition ou à l'acquisition de données alphanumériques. Les messages servent de support au dialogue.

Un message est essentiellement une notion non graphique.

-- le numéro de corrélation (ou numéro d'identification) est une caractéristique de la section. C'est un nombre entier. La corrélation peut être utilisée pour regrouper plusieurs sections (en leur associant la même corrélation). On peut ainsi définir un niveau intermédiaire entre les "figures" et les "sections".

IV.2.4. Le dialogue

-- Une fonction d'interaction est un élément de base du dialogue. Elle permet de recueillir des informations d'un type déterminé, à partir des dispositifs de dialogue.

Certains termes utilisés ici sont relativement nouveaux, mais ils ont déjà été proposés dans divers groupes de travail sur la standardisation des logiciels graphiques [N1], [N2], [N3], [N4].

IV-3- LA DÉFINITION DES PRIMITIVES GRAPHIQUES

Nous étudierons ici les outils qui permettent d'obtenir et de gérer une image sur l'écran [R1], [I14].

IV.3.1. Les concepts de base

L'idée fondamentale qui nous a guidés est que le logiciel graphique a pour tâche d'interpréter un certain nombre de données afin de créer une représentation de celles-ci sous la forme d'un dessin. Les éléments de définition sont de trois ordres :

- les systèmes de coordonnées employés,
- la nature des données,
- le mode d'interprétation.

En application de ce que nous avons dit en IV.1., le logiciel développé ne s'intéresse qu'au dessin dans le plan. Pour obtenir un dessin, il faut repérer un certain nombre de positions à l'aide de coordonnées, ces positions servant de base au tracé. Il est donc nécessaire de disposer d'au moins un système de coordonnées. L'expérience montre qu'il est souhaitable d'en donner deux, un qui concerne l'utilisateur et qui est à son libre choix, un autre qui concerne l'écran et qui dépend du matériel. Le logiciel doit donc donner la possibilité de définir l'espace de départ (utilisateur) et l'espace d'arrivée (écran), la transformation faisant passer d'un espace à l'autre étant réalisée automatiquement

Les données à traiter sont de nature profondément différentes, et nous proposons de les différencier nettement, nous éloignant en ceci des logiciels classiques. Nous distinguons par exemple, le traitement d'ensembles de coordonnées du traitement de textes, même si les outils peuvent être utilisés concurremment. L'interface entre le logiciel de mise en forme et le logiciel graphique consiste donc en une structure de données délivrant des valeurs qui, associées à un mode de tracé, permettent de construire les dessin. Par exemple, un ensemble de coordonnées conduira à dessiner un graphe, un nuage de points, une courbe, suivant

l'interprétation désirée. Cette notion est valable également pour les textes, puisqu'un ensemble de caractères peut être tracé avec des tailles, des orientations diverses. Le logiciel graphique offrira donc des primitives permettant de définir le type de donnée et la structure de rangement.

Le tracé lui-même sera défini grâce à la donnée d'un mode d'interprétation, qui fixera les caractéristiques du dessin.

Une fois l'image affichée, l'opérateur prend un certain nombre de décisions qui sont transmises au programme d'application. Ces décisions conduisent en général à des modifications de la structure de données, modifications qui vont se refléter sur l'écran, par exemple sous la forme d'effacement de parties d'images. Ceci implique qu'une relation puisse être établie entre les données graphiques et les données de l'application. Pour ce faire, nous proposons d'utiliser des listes de noms, permettant une structuration du dessin à deux niveaux :

- un niveau global (nom de figure), qui permet de regrouper un ensemble de données
- un niveau secondaire (corrélation), qui permet, à l'intérieur d'une figure, de distinguer des éléments.

Le logiciel devra donc prendre en charge la gestion des listes de noms, afin de pouvoir, soit les restituer sur une demande d'identification, soit retrouver une partie d'image à partir d'un identificateur.

Nous étudions ci-après les primitives correspondant à ces concepts.

IV.3.2. Les systèmes de coordonnées

IV.3.2.1. Définition de fenêtre

FENETRE(nfen, igx, igy, sdx, sdy)

Cette primitive permet de définir un espace de départ chez l'utilisateur. Le numéro *(nfen)* sert d'identificateur et permet d'utiliser la même fenêtre pour des ensembles de données différents. Les coordonnées des coins de la fenêtre *(igx, igy, sdx, sdy)* sont exprimées dans le système choisi par l'utilisateur.

IV.3.2.2. Définition de clôture

CLOTURE(nclot, igx, igy, sdx, sdy)

Cette primitive permet de définir un espace d'arrivée, sur l'écran. Les coordonnées des coins de la clôture sont exprimées dans un système de coordonnées arbitraire, permettant de définir la portion d'écran choisie pour l'affichage, indépendamment des systèmes de coordonnées réels.

La solution adoptée est de préciser les bornes de la clôture en centièmes d'écran.

Notons qu'il existe une clôture prédéclarée (de numéro 0) correspondant à la totalité de l'écran.

IV.3.2.3. Relation entre fenêtre et clôture

Le logiciel graphique assure automatiquement la transformation qui affiche le contenu de la fenêtre de la meilleure manière possible dans la clôture :

- utilisation maximale de la clôture,
- centrage de la fenêtre par rapport à la clôture,
- pas de déformation du contenu de la fenêtre.

Notons que la forme d'une clôture dépend de la forme de l'écran. Nous donnons deux exemples de correspondance entre fenêtre et clôture dans les figures IV.1 et IV.2.

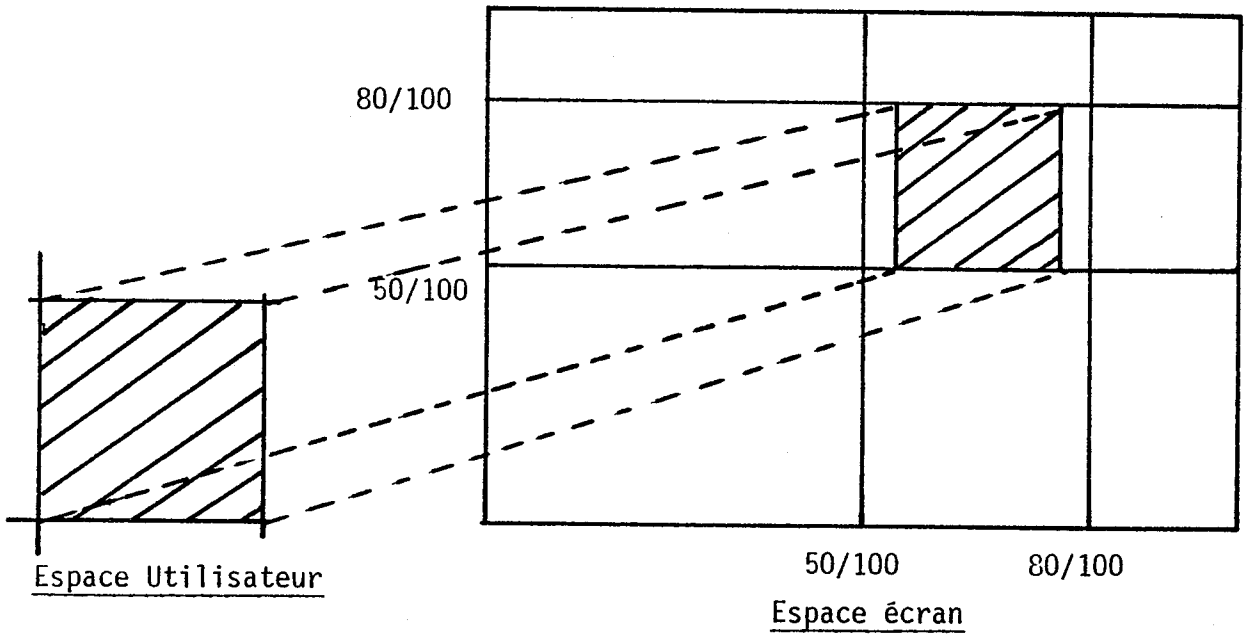


Figure IV.1. : Ecran rectangulaire

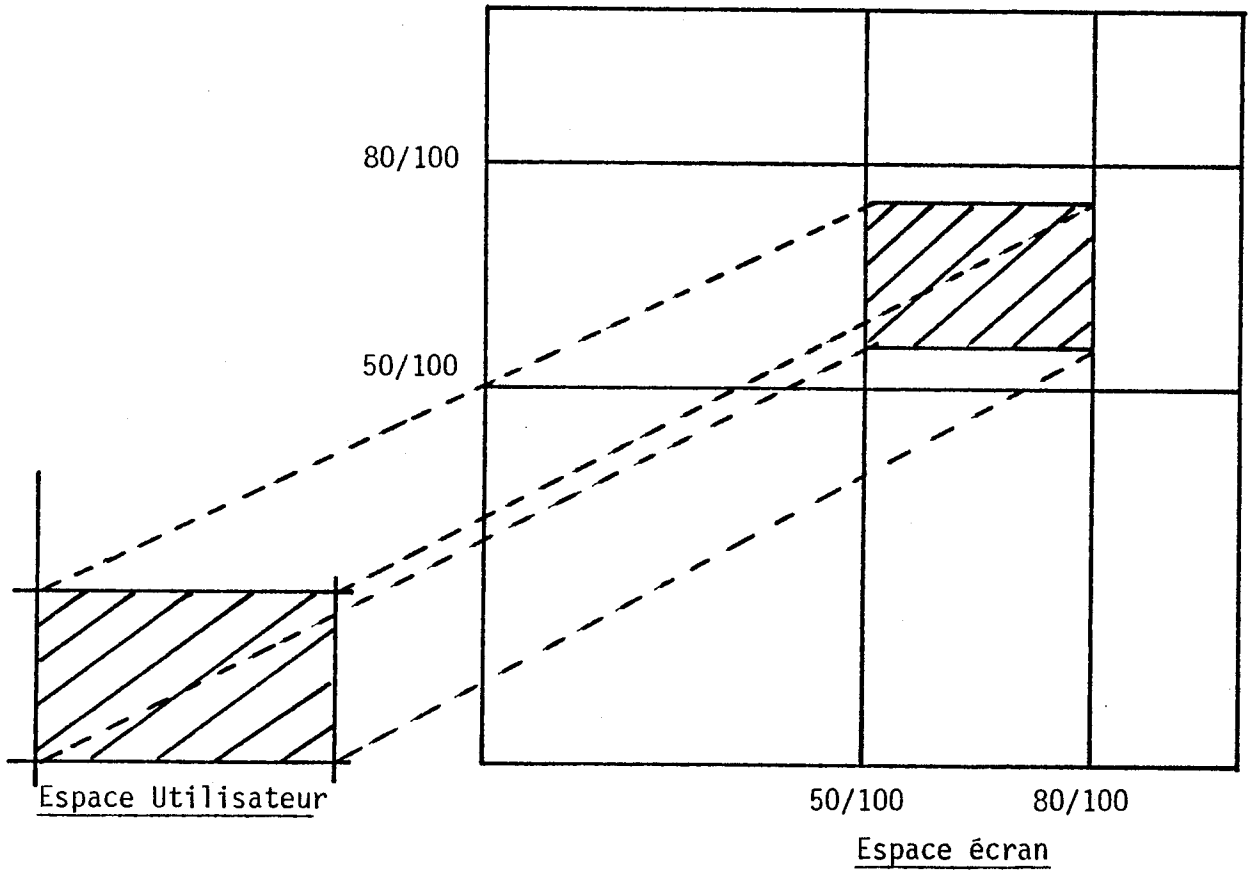


Figure IV.2. : Ecran carré

IV.3.3. La déclaration des données

POINTS(nfig, tablong, tabx, taby, tabmarqueurs)
TEXTES(nfig, tablong, tabtextes, tabx, taby)

Ces deux primitives permettent de prévenir le logiciel graphique du type des données qu'il aura à manipuler, et de définir la structure employée.

IV.3.3.1. Déclaration de points

La primitive *POINTS* permet de déclarer une suite de coordonnées, suite contenue dans les tableaux à une dimension *tabx, taby*. La longueur de chaque section est définie grâce au tableau *tablong*. Pour préciser le nombre de sections de la figure et indiquer la dernière longueur significative dans le tableau *tablong*, on terminera celui-ci par la valeur "0". Le tableau *tabmarqueurs* correspond à la notion de marquage d'une section. Chaque élément de *tabmarqueurs* contient un groupe de caractères (par exemple 4), qui est destiné à marquer la première position de la section correspondante. Ce tableau permet ainsi d'étiqueter (si nécessaire) les différentes sections d'une figure.

Les sections sont repérées au niveau du système par leur numéro d'ordre dans le tableau des données (à partir de 1). Par contre, le programmeur d'application pourra attribuer à chaque section un numéro d'identification (corrélation), permettant de la différencier des autres. Les deux niveaux de dénomination correspondent donc :

- globalement, aux données repérées par le même numéro de figure,
- au niveau en dessous, aux différentes sections d'une même figure.

IV.3.3.2. Déclaration de textes

Il est intéressant de constater la parfaite symétrie de la primitive *TEXTES*. Le texte, rangé dans le tableau *tabtextes*, est découpé en sections, dont la longueur est spécifiée dans *tablong*. A chaque section est associé un couple de coordonnées contenues dans *tabx, taby*, permettant de spécifier le

point de départ du texte. L'omission de ce couple permet de lier un texte au texte de la section précédente. On doit obligatoirement associer une position à la première section.

Notons que les primitives *POINTS* et *TEXTES* sont l'équivalent de déclarations pour le logiciel, mais ce ne sont pas des affectations. C'est-à-dire que les différents tableaux passés en paramètre ne sont pas nécessairement remplis au moment de la déclaration.

IV.3.4. L'interprétation des données

IV.3.4.1. La primitive

MODE(nfig, ncorrel, nsection, descripteur)

Cette primitive permet d'associer à une section *(nsection)*, d'une figure *(nfig)*, un certain nombre de qualifications qui permettront de l'interpréter et d'obtenir le tracé la représentant. Nous considérons que le numéro de corrélation *(ncorrel)* est une caractéristique comme une autre, de la section, ce qui explique qu'il soit attribué à travers la primitive *MODE*.

Le mode s'adresse aussi bien aux sections de points qu'aux sections de textes, la distinction est réalisée par le fait que le descripteur est différent suivant le cas.

IV.3.4.2. Syntaxe du descripteur de mode

Il est évident que la liste des paramètres contenus dans le descripteur, ainsi que celle des valeurs possibles de ceux-ci ne sont pas figées une fois pour toutes. La prise en charge de nouvelles options à ce niveau est relativement simple, ce qui permettra d'élargir les possibilités du logiciel, si nécessaire. Les paramètres retenus dans la version actuelle du logiciel sont les suivants :

- pour les sections de points :
  - couleur du trait,
  - texture du trait,
  - épaisseur du trait,
  - intensité,

- type de représentation,
- système de coordonnées,
- type de marquage.

- pour les sections de textes :
  - taille des caractères,
  - couleur,
  - intensité,
  - épaisseur,
  - position initiale.

Dans le descripteur, chaque nom de paramètre est représenté par une lettre, et la valeur choisie pour ce paramètre est désignée par un nombre.
Le marquage peut être réalisé de trois façons :

- par affichage en chaque position d'un même symbole alphanumérique,
- par affichage en chaque position d'un symbole alphanumérique, suivi d'un indice, mis à jour pour chaque point,
- par affichage d'une étiquette (4 caractères) sur la première position de la section.

Dans le cas où les coordonnées sont relatives, celles-ci sont évaluées par rapport à la position précédente dans la section.

Les premières coordonnées de la section sont évaluées par rapport à (0,0). Ceci veut dire que les coordonnées relatives sont évaluées à l'intérieur de la section, et non pas à l'intérieur de la figure.

Le paramètre "position initiale" pour les textes permet d'indiquer si la position initiale est précisée ou si on doit écrire le texte à la suite du précédent.

Les tableaux des figures IV.3 et IV.4 précisent les différents paramètres, leurs options, ainsi que leur codage.

La syntaxe du descripteur est donnée ci-dessous.

Syntaxe du descripteur

<descripteur> ::= <format> | <format> <descripteur>
<format> ::= <param> <option>
<param> ::= |C|T|E|I|R|M|S|G|O|P|N|
<option> ::= <nombre> | <marqueur> | - |
<marqueur> ::= (symbole quelconque | symbole quelconque <nombre> |0|
<nombre> ::= <chiffre> | <chiffre> <nombre>
<chiffre> ::= 1|2|3|4|5|6|7|8|9|0|

l'ensemble des "symboles quelconques" et l'ensemble de tous les symboles y compris les lettres et les chiffres.

Des contrôles seront effectués sur le descripteur, car celui-ci peut être syntaxiquement correct, mais sémantiquement incohérent. Si le programmeur d'application invoque la fonction *MODE* pour associer une corrélation à une section, sans se soucier du mode de preprésentation de celle-ci, il dispose d'un mode neutre qui choisira les options standards pour les différents paramètres.

Paramètres	Codage	Valeur du paramètre	Codage des valeurs
Mode neutre	N		
Couleur du trait	C	Vert	1
		Jaune	2
		Rouge	3
		Bleu	4
Texture du trait	T	Continu	1
		Tireté court	2
		Tireté long	3
		Pointillé	4
		Mixte	5
Epaisseur du trait	E	9 valeurs possibles	1 à 9
Intensité du trait	I	Clignotement	-
		99 valeurs possibles	1 à 99
Représentation des coordonnées	R	Ligne brisée	1
		Segments disjoints	2
		Points	3
Marquage	M	Symbole simple	symbole
		Symbole indicé	symbole suivi de l'indice
		Etiquette	0
Système de coordonnées	S	Absolu	1
		Relatif	2

Figure IV.3. : Tableau des modes pour les points

Paramètres	Codage	Valeur du paramètre	Codage des valeurs
Taille des caractères	G	4 tailles possibles	1 à 4
Orientation du texte	O	Angle quelconque avec l'horizontale	angle
Couleur du texte	C	Vert	1
		Jaune	2
		Rouge	3
		Bleu	4
Epaisseur	E	9 valeurs possibles	1 à 9
Intensité	I	Clignotement	-
		99 valeurs possibles	1 à 99
Position initiale	P	position initiale précisée	1
		position initiale non précisée	2

Figure IV.4. : Modes pour les textes

Si un paramètre n'apparaît pas dans le descripteur, le logiciel lui affectera par défaut une valeur standard.

Le tableau de la figure IV.5 précise ces valeurs par défaut.

Paramètre	Valeur par défaut	Codage des valeurs
Texture	Continu	T1
Représentation	Ligne brisée	R1
Système de coord.	Absolu	S1
Marquage	Aucun	
Orientation du texte	Horizontale	00
Position initiale du texte	Précisée	P1

Figure IV.5. : Les options standards du mode

Les options par défaut des autres paramètres (couleur, etc...) sont fonctions du type de matériel utilisé.

IV.3.5. La gestion de l'affichage

AFFICHER (nfig, ncorrel, nfen, nclot)
EFFACER (nfig, ncorrel)

La primitive *AFFICHER* permet d'obtenir sur l'écran la représentation des ensembles de données qui ont été déclarés lors de l'attribution du numéro de figure.

Les systèmes de coordonnées sont définis par référence à une fenêtre et une clôture. Si l'on ne désire pas afficher toute la figure, on peut se servir du numéro de corrélation passé en paramètre. Pour simplifier certaines écritures, nous proposons la convention suivante :

	nfig	*ncorrel*	*nsection*
nul	toutes les figures	toutes les sections	toutes les sections
positif	la figure *nfig*	les sections corrélées *ncorrel*	la section *nsection*
négatif	toutes les figures sauf *nfig*	toutes les sections sauf celles corrélées *ncorrel*	toutes les sections sauf *nsection*

Cette convention permet d'alléger la tâche d'écriture.

De plus, toutes les combinaisons possibles sont autorisées.

L'affichage se fait dès l'invocation de la primitive *AFFICHER*. La notion d'affichage différé (en général proposée pour des notions d'efficacité) n'existe pas. Par contre, on peut noter que l'on peut construire toute une figure avant de l'afficher, quelque soit le mode de tracé de chaque section. Ce fait est extrêmement intéressant du point de vue du programmeur d'application, car il permet de conserver la logique de construction des objets à représenter, sans se soucier des problèmes de tracé proprement dit.

En ce qui concerne la primitive *EFFACER*, il y a peu de choses à dire, sinon que la combinaison des numéros de figure et de corrélation permet d'obtenir des effacements sélectifs au niveau de chaque section (ou d'un ensemble de sections de même corrélation).

Notons que le mode d'une section peut être modifié à tout moment. Ceci n'a aucun effet sur ce qui est déjà affiché, mais si l'utilisateur demande un nouvel affichage de la section, le nouveau mode sera pris en considération.

En conclusion, nous dirons que si le nombre de primitives d'affichage est réduit, la richesse d'expression réside dans les possibilités offertes par le mode. Celles-ci peuvent évidemment être variées à l'infini : il suffit d'inclure au niveau du logiciel les sous-programmes de traitement nécessaires, et d'ajouter de nouveaux codes dans la description du mode.

IV-4- LES MESSAGES À L'OPÉRATEUR

IV.4.1. Le concept de base

Nous rappelons que nous nous plaçons dans le cadre d'applications graphiques interactives. Lors du déroulement d'un programme, un certain nombre d'indications sont fournies à l'opérateur, qui lui permettent de suivre et de contrôler l'exécution. Ces indications, qui ne font pas partie du dessin proprement dit, prennent en général la forme de textes courts, accompagnés d'indications numériques. Nous nommons ces informations messages. Nous mettons à la disposition du programmeur des primitives permettant de composer, puis d'afficher ces messages.

Les chaînes de caractères ainsi constituées seront affichées exclusivement grâce au générateur de caractères de la console, l'utilisateur n'ayant aucune influence sur l'apparence de ce texte (il n'y a pas de mode attaché à un message, car celui-ci ne fait pas partie du dessin).

IV.4.2. La description d'un message

MESSAGE (nmess, descripteur, nclot, x, y)

Un message est constitué d'une chaîne de caractères portant des zones de texte et des zones réservées. L'emplacement du message est défini par la donnée d'une

clôture *(nclot)* et de la position du début du texte dans cette clôture *(x,y)*. Chaque message est repéré par un numéro *(nmess)*.

Lors de l'invocation de la primitive *MESSAGE*, le texte spécifié est affiché. Les zones réservées sont vides lors de la première invocation. Elles contiennent ensuite des valeurs qui ont été soit éditées (voir IV.4.3.), soit introduites lors d'une phase de dialogue.

IV.4.2.1. Syntaxe du descripteur de message

Le descripteur passé en paramètres contient du texte à afficher et des indications sur les zones à réserver (nombre, longueur).

La syntaxe est la suivante :

```
<descripteur>      ::= <format> | <format><descripteur>
<format>           ::= &<liste-de-zones>& |<chaîne-de-caractères>
<liste-de-zones>   ::= <sous-liste> | <sous-liste>,<liste-de-zones>
<sous-liste>>      ::= <entier*entier> | <entier>
<entier>           ::= <chiffre> | <chiffre><entier>
<chiffre>          ::= 0|1|2|3|4|5|6|7|8|9
<chaîne-de-caractères> ::= <caractère> | <caractère><chaîne-de-caractères>
```

Le format <chaîne-de-caractères> est utilisé pour faire apparaître sur l'écran une chaîne de caractères (le caractère "&" est interdit).

Le format "&n&" est utilisé pour réserver une zone de n espaces pour un emploi ultérieur.

Exemple :

'PARAMETRE1&5&AUTRES PARAMETRES&2*3,5&'

La chaîne affichée sera la suivante :

Figure IV.6. : Message

IV.4.2.2.2. La postion d'un message

La position d'un message doit être connue indépendamment du terminal utilisé ; pour ce faire, on définit une clôture (en centières d'écran) et la position de départ du message est elle-même précisée en centièmes, par rapport à cette clôture.

Exemple :

CLOTURE (1,50,50,100,100)
MESSAGE (1,'CECI EST UNE CHAINE',1, 10, 90)

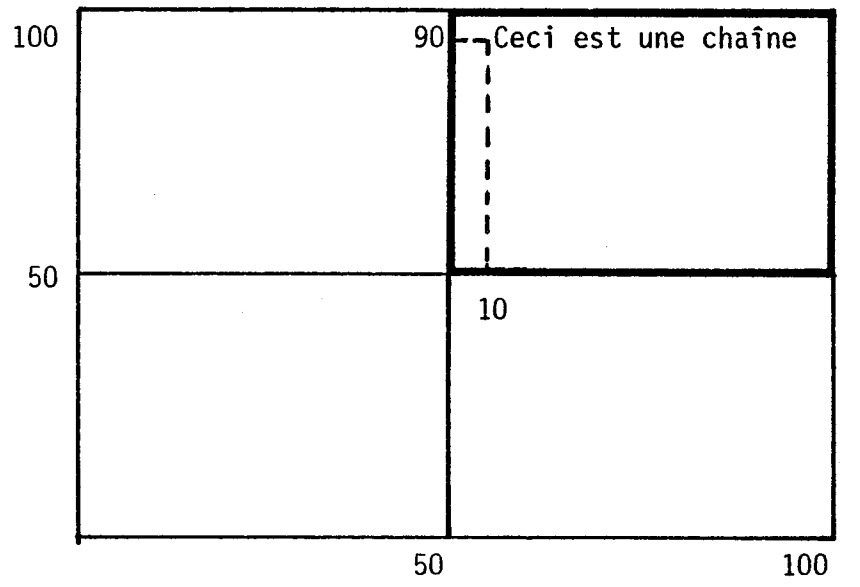


Figure IV.7. : Position du message

IV.4.3. L'édition de données alphanumériques

EDENTIER (nmess, nzone, entier)
EDREEL (nmess, nzone, reel)
EDCHAINE (nmess, nzone, chaîne)

Les zones réservées à l'intérieur d'un message peuvent servir à l'édition de valeurs alphanumériques de type entier, réel, ou chaîne de caractères. L'édition d'une variable se fait en sélectant une zone *(nzone)* d'un message donné *(nmess)*. Le contenu de la variable est alors affiché dans l'emplacement choisi. Cet emplacement ne sera plus modifié jusqu'à rencontrer un ordre d'effacement du message, ou une nouvelle édition avec les mêmes paramètres.

On voit donc que ces primitives, associées à l'emploi de *MESSAGE* permettent de suivre les variations d'une valeur et donc de contrôler la marche d'un programme.

IV.4.4. L'effacement d'un message

L'effacement d'un message est réalisé par l'invocation de la primitive :

EFFMESSAGE (nmess)

Si *nmess* est égal à zéro, on efface tous les messages.

IV-5- LA DÉFINITION DES PRIMITIVES DE DIALOGUE

IV.5.1. Les concepts de base

Nous écartons délibérément la définition de primitives faisant appel explicitement à tel ou tel dispositif de communication, nous utilisons la notion de dispositif logique, qui permet de définir des outils de dialogue en termes fonctionnels, sans se préoccuper de leur réalisation effective.

Nous proposons quatre fonctions d'interaction de base :

- introduction d'une valeur alphanumérique,
- introduction d'un couple de coordonnées,
- identification d'un élément du dessin,
- sélection d'une action parmi n proposées (menu).

On remarquera que nous offrons moins d'actions de base que ce qui est proposé

dans la littérature ([G6],[G7]), mais toutes les actions supplémentaires peuvent être réalisées au niveau du programme d'application. Par contre, nous ne proposons pas d'actions impossible à réaliser sur certain type de matériel, telle que l'action d'entraînement *(dragging)*.

La première réalisation de GRIGRI (voir chapitre III), nous a permis de constater l'intérêt des primitives proposées. Cependant, l'expérimentation a rapidement montré qu'elles étaient toujours utilisées par composition. Si l'on prend l'exemple de la collecte de coordonnées, on voit qu'en général l'opérateur désire introduire un nombre variable de couples de coordonnées, nombre parfois fixé à l'avance (un couple, n couples), ou inconnu a priori (relevé de courbes). Cette constatation conduit donc à adjoindre à la primitive de collecte de coordonnées un paramètre représentant le nombre de répétitions de l'action élémentaire "relever un couple de coordonnées". Le même raisonnement peut être appliqué aux trois autres primitives. Nous proposons donc d'associer un nombre de répétitions à chacune des actions élémentaires évoquées ci-dessus.

IV.5.2. Introduction de données alphanumériques

RVAL (nb, nmess, Tréel)
NVAL (nb, nmess, Tentier)
CVAL (nb, nmess, Tchaîne)

L'introduction de données alphanumériques se fait par l'intermédiaire des messages. A chaque invocation d'une fonction d'introduction de données, une zone du message spécifié *(nmess)* est sélectée. L'opérateur tape la donnée au clavier alphanumérique et celle-ci est transmise en fin d'action au programme d'application par l'intermédiaire d'un tableau *(treel, tentier ou tchaîne)*.

Le nombre de répétition *(nb)* est indiqué de la manière suivante :

- *nb = 0* nombre inconnu a priori. Une indication de fin d'action sera donnée par l'opérateur. Le nombre de données relevé sera alors transmis en retour.
- *nb > 0* nombre maximum d'entrées. L'arrêt se fait soit sur réception d'une indication de fin de donnée par l'opérateur, soit lorsque le maximum est atteint. Le nombre de données relevé est également transmis en retour.

Un nombre négatif n'a aucune signification.
Cette convention est utilisée pour toutes les primitives de dialogue.

Lors d'une demande multiple de valeurs, le curseur progresse de zone en zone à l'intérieur du message. Si on arrive à la dernière zone réservée, on boucle sur le message et la prochaine entrée se fait sur la première zone. Quand on active une nouvelle fois la fonction d'entrée de valeurs pour le même message, on commence par remplir la zone suivant la dernière zone utilisée lors de l'appel précédent (les messages sont donc comparables aux formats Fortran).

IV.5.3. Collecte de coordonnées

POSITION (nb, nfig, nfen, nclot)

Le nombre de coordonnées spécifié *(nb)* est relevé par tout dispositif de communication utilisable et rangé dans les tableaux *tabx, taby* indiqués lors de la déclaration de données de type "points" correspondant à *nfig*. Le tableau des longueurs de section est également mis à jour. Les coordonnées sont rangées à la suite de celles existant déjà dans la figure. Ainsi, si une figure comporte déjà 2 sections, les points collectés feront partie de la troisième section et des suivantes.

La donnée de la clôture *(nclot)* et de la fenêtre *(nfen)* permet de relever les points dans l'espace écran souhaité, mais de restituer des coordonnées exprimées dans le système de l'utilisateur.

L'opérateur dispose d'une indication de "fin de section" lui permettant de collecter des coordonnées dans plusieurs sections successives de la figure.

Nous avons cherché à offrir une primitive d'entrée correspondant à la primitive *POINTS*. Il nous parait en effet indispensable d'avoir une certaine symétrie entre les primitives d'affichage et les primitives d'entrée pour que la programmation puisse se faire au même niveau logique dans les deux cas.

IV.5.4. Identification

IDENTIFIER (nb, nfig, tfig, tcorrel)

Cette primitive permet d'identifier une suite d'éléments d'un dessin. L'identification (numéro de figure, numéro de corrélation) est rangée dans les tableaux *tfig, tcorrel*.

La convention donnée en IV.3.5. permet de spécifier quelle figure, ou quel ensemble de figures peut participer à l'opération. On peut ainsi rendre certaines figures "sensibles" ou non à la désignation.

Cette primitive permet également de désigner des messages. Dans ce cas, le numéro de figure en retour est nul (par convention) et le numéro du message remplace le numéro de corrélation.

IV.5.5. Menu

IV.5.5.1. La primitive

MENU (nb, descripteur, tretour)

Le descripteur du menu est formé d'une liste de nombres entiers, chacun d'entre eux pouvant être étiqueté. Le résultat de l'action est une suite de *nb* nombres, suite rangée dans le tableau tretour. Les valeurs sont introduites par l'un quelconque des moyens de communication disponibles. Tous les nombres portant une étiquette pourront être obtenus par le biais d'un menu affiché sur l'écran et constitué de la liste des étiquettes. Les étiquettes apparaissent sur l'écran, les unes au-dessous des autres, à partir du coin supérieur droit.

IV.5.5.2. Syntaxe du descripteur de menu

La syntaxe du descripteur de menu est la suivante :

<descripteur> ::= <fonction> | <fonction>,<descripteur>
<fonction> ::= <nombre> | <nombre><nom-de-fonction>
<nombre> ::= entier compris entre 1 et 28
<nom-de-fonction> ::= chaîne de 8 caractères au maximum, commençant obligatoirement par une lettre, associée au numéro qui la précède et séparée de celui-ci par un espace au moins.

Le <nom-de-fonction> peut-être omis, dans ce cas, la fonction n'est représentée que par un numéro et ne pourra être désignée par l'opérateur qu'à l'aide de ce numéro.

Exemple :

'1,2,3 GAUCHE, 4 DROITE, 5 HAUT, 6 BAS, 28'.

les fonctions sont représentées par les numéros :

1, 2, 3, 4, 5, 6, 28

les étiquettes affichées sur l'écran sont :

GAUCHE
DROITE
HAUT
BAS

les fonctions 1, 2, 28 ne peuvent être désignées que par leur numéro.

IV.5.6. Programmation et opération des primitives de dialogue

Il est clair que le faible nombre des primitives de dialogue, ainsi que leur définition fonctionnelle permettent une programmation simple. La mise en oeuvre de l'interaction ne pose plus de problème, puisqu'il suffit d'adapter les primitives graphiques aux actions de base prévues lors de la conception du programme d'application. Nous avons pu constater par exemple l'abandon de la lecture des valeurs alphanumériques sur un terminal par le biais d'ordre "read", la primitive d'obtention de telles données par le biais de messages étant d'un emploi très facile.

En ce qui concerne le point de vue de l'opérateur, les facilités offertes par le logiciel graphique permettent un emploi souple et efficace des différents dispositifs existant sur la console qu'il utilise à un moment donné. Il peut donc adapter son comportement au problème qu'il traite, utilisant le dispositif qui lui paraît le plus approprié à la phase en cours d'exécution. Par exemple, lors de l'introduction d'une suite de coordonnées, l'opérateur peut utiliser indifféremment une tablette graphique pour relever un certain nombre de points sur une courbe, le photostyle (ou son équivalent) pour indiquer des point directement sur l'écran, ou encore le clavier alphanumérique pour donner des coordonnées précises obtenues par ailleurs. Le logiciel se contente d'analyser

les entrées en provenance des différents dispositifs et de les interpréter en fonction de la demande en cours d'exécution.

Il est bien évident que la prise en charge de ces entrées peut être plus ou moins complexe. Par exemple, nous offrons au niveau des quatre primitives un minimum de gestion :

- annulation de la dernière entrée,
  (commande que l'on peut répéter)
- annulation de toutes les entrées que l'on vient de faire,
- validation de l'ensemble
  (arrêt de la collecte et retour au programme d'application).

Ce principe, appliqué à la collecte de coordonnées, permet ainsi d'entrer une courbe avec un minimum d'erreurs. De plus, dans le cas de l'utilisation d'une tablette graphique le logiciel effectue un filtrage automatique des points recueillis de manière à assurer un espacement correct des relevés. Une adaptation à une application particulière peut donc se faire à ce niveau, en rendant plus complexe la partie chargée de gérer les entrées.

Nous pensons que cette approche est extrêmement riche, en particulier du fait de la symétrie avec les concepts proposés pour l'affichage. De plus, le mécanisme d'extension est très simple, consistant en adjonctions au niveau du logiciel graphique, ce qui permet de l'adapter à certaines classes d'applications fréquemment mise en oeuvre.

IV.5.7. Autre méthode de gestion du dialogue

En fait, la composition des actions de base du dialogue peut se faire de deux manières :

- par composition d'une même action avec un facteur de répétition (actions multiples)
- par composition de plusieurs actions de type différent (actions composées).

La première solution est celle décrite plus haut.

Une deuxième manière de composer les actions de dialogue est, en effet, de créer, à la demande du programmeur, une combinaison conduisant le logiciel à proposer, à un moment donné, non pas un type, mais un ensemble de types d'actions

Ceci est fait en regroupant par deux, ou trois, ou quatre, les actions élémentaires. Le mécanisme est relativement simple, conduisant à donner une primitive "déclaration d'action" qui précise les différents types valides à un moment donné et les paramètres.

La déclaration d'action spécifie les types des actions qui peuvent être exécutées à un instant donné. Ces types sont au nombre de 4 :

P : collecte de coordonnées

M : menu

V : introduction de valeurs

I : identification.

Le dialogue se réduit dans ce cas, à une seule primitive, on peut imaginer quelque chose comme :

ACTION (liste de types autorisés, liste de paramètres, liste des actions exécutées).

En retour, les résultats des diverses actions seront rangés dans les paramètres et la fonction fournira la liste des actions qui ont été effectivement exécutées par l'opérateur. Ainsi, le programme pourra aller chercher les résultats correspondants.

Pour des raisons de facilité de réalisation, nous n'avons pas, à l'heure actuelle, implémenté ce second type de généralisation. Notons que cette approche par "fonctions composées" conduit à un certain nombre de cas d'ambiguité pour l'utilisation des divers dispositifs de dialogue, surtout pour les terminaux peu évolués. Ainsi, par exemple, si le programme demande les quatre types d'actions sur terminal Tektronix ne disposant que de deux dispositifs d'entrée (réticule, clavier alphanumérique), il y aura ambiguité pour les deux dispositifs ; le réticule peut en effet répondre à 3 types d'actions :

- collecte de coordonnées (P),
- identification (I),
- menu (M).

La clavier alphanumérique permet d'exécuter 3 actions également :

- collecte de coordonnées (P),
- entrée de valeurs alphanumériques (V),
- menu (M).

Pour lever ces ambiguités, on est obligé d'imposer certaines contraintes à l'opérateur, contraintes qui alourdissent beaucoup l'utilisation des terminaux simples. C'est donc pour des raisons de facilité de réalisation et de mise en oeuvre que nous nous sommes intéressés à la première méthode : actions à caractère répétitif.

IV-6- CONCLUSION

Les primitives choisies n'entâchent en rien la notion d'indépendance du logiciel par rapport à son contexte.

- Leurs fonctions assurent une bonne indépendance par rapport aux applications,
- la philosophie des primitives d'affichage autorise l'indépendance par rapport au terminal. En effet, le logiciel n'utilise jamais la notion de "faisceau" ou de "plume" sur laquelle s'appuie la plupart des logiciels existants.

D'autre part, l'approche fonctionnelle du dialogue assure l'indépendance du logiciel par rapport aux divers outils de dialogue.

La liste récapitulative des primitives est donnée en Annexe 1, ainsi qu'un exemple de programme d'application (Annexe 2).

CHAPITRE V - LA RÉALISATION DU LOGICIEL GRIGRI

Nous décrivons ici le prototype réalisé sur les installations du Centre Interuniversitaire de Calcul de Grenoble.

V-1- LE CONCEPT DE LOGICIEL "ADAPTATIF"

En I-8, nous avons constaté qu'un logiciel de base général pouvait toujours être considéré comme formé de deux modules :

- un module indépendant du terminal utilisé,
- un module dépendant du terminal utilisé.

La première réalisation de GRIGRI, basée sur la définition d'une console graphique virtuelle, nous a montré les limitations d'une approche qui attribue un rôle trop important au module indépendant du terminal (voir III.7). En effet, cette philosophie fige les possibilités d'utilisation optimale du matériel en fonction du choix du niveau et des options du dispositif virtuel. En particulier, la gestion de la liste d'affichage structurée ainsi que la gestion de la mise en page de l'écran sont entièrement assurées au niveau de la console virtuelle.

Il est clair que la présence d'une liste d'affichage structurée est indispensable, mais nous avons été conduits à envisager une répartition différente des tâches, donnant plus de responsabilité aux modules d'interprétation.

V.1.1. Organisation d'un logiciel adaptatif

Un logiciel adaptatif est composé de deux modules :

- un module de communication avec l'application (indépendant du terminal utilisé),
- un module "interpréteur", de gestion du terminal.

Dans la seconde réalisation de GRIGRI, nous avons reporté le stockage et la gestion de la liste d'affichage au niveau du logiciel dépendant du terminal.

La figure V.1. présente l'organisation d'ensemble d'un "logiciel adaptatif".

Cette approche permet d'adapter la liste d'affichage au type du terminal :

- Si on utilise un terminal à mémoire d'entretien, la liste d'affichage ne sera pas simulée par le logiciel.
- Si on utilise un tube à mémoire, la liste d'affichage sera à la charge de l'interpréteur et elle pourra contenir des commandes adaptées aux possibilités du matériel.

Les données sont, elles aussi, réparties en deux groupes :

- les données de communication avec l'application,
- les données nécessaires à la gestion de l'affichage.

V.1.2. Les deux types d'interpréteurs

La mise en oeuvre de ce principe conduit à deux structures de découpage, suivant que l'on dispose, ou non d'une mémoire structurée fournie par le matériel.

L'expérience prouve (voir I.4.1.) que les consoles de visualisation disposent toutes d'une "mémoire d'entretien", et que l'on distingue deux classes de matériel suivant le type de mémorisation :

- terminal disposant d'une mémoire contenant une description structurée de l'image,
- terminal disposant seulement d'une description analytique.

Ces deux classes de matériel nous conduisent tout naturellement à deux types d'interpréteurs pour un logiciel adaptatif.

Ces deux classes se distinguent suivant les rôles respectifs joués par le logiciel et par le matériel.

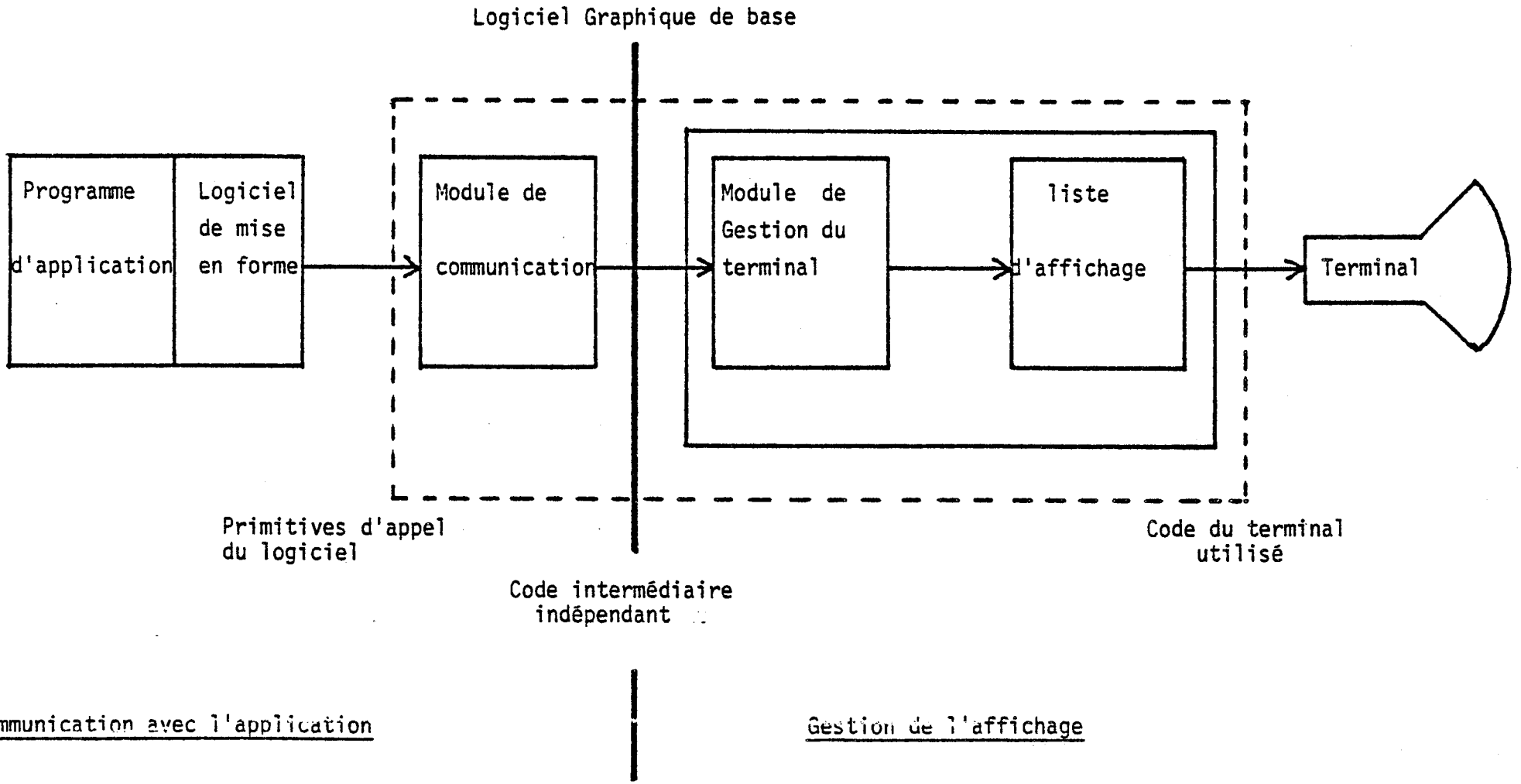


Figure V.1. : Logiciel adaptatif

Notons que tous les types de terminaux rentrent dans l'une ou l'autre classe, c'est-à-dire que nous aurons seulement deux types d'interpréteurs associés à un logiciel adaptatif.

Le premier type ne contient pas de simulation d'image structurée, il est destiné à prendre en charge les terminaux à balayage cavalier disposant d'une mémoire d'entretien. La figure V.2. décrit la configuration de l'ensemble logiciel-matériel pour ce type d'interpréteur.

Le second type est destiné à prendre en charge les terminaux qui disposent seulement d'une description analytique de l'image. Dans ce cas, le logiciel simule une mémoire d'entretien contenant une image structurée. Ce type d'interpréteur pourra piloter :

- les terminaux à tubes à mémoire,
- les terminaux à balayage ligne par ligne disposant d'une mémoire de points,
- les tables traçantes.

La figure V.3. décrit ce type d'interpréteur.

Il est clair que pour chaque terminal, il faudra un interpréteur déterminé pour exploiter au mieux le matériel, mais dans tous les cas, cet interpréteur rentrera dans l'une des catégories citées plus haut. A l'intérieur de l'une des deux classes, les interpréteurs se distinguent seulement par un codage adapté au terminal, mais l'organisation d'ensemble reste la même pour un logiciel donné.

La liste d'affichage ne contient pas forcément le code réel destiné à la console, même s'il est adapté au terminal final. Ceci explique que l'interpréteur ait à engendrer, dans certains cas, du code réel à partir de celui contenu dans la liste virtuelle.

L'expérimentation que nous avons conduite à partir de cette structure de logiciel nous a montré que le problème de l'indépendance par rapport à la console était alors résolu de façon plus souple, permettant de préserver la plupart des qualités de chaque terminal considéré.

Signalons une question de vocabulaire :
le module de communication avec l'application, indépendant du terminal sera désormais appelé "noyau" du logiciel. Le logiciel est donc composé :

- d'un noyau,
- d'une série d'interpréteurs.

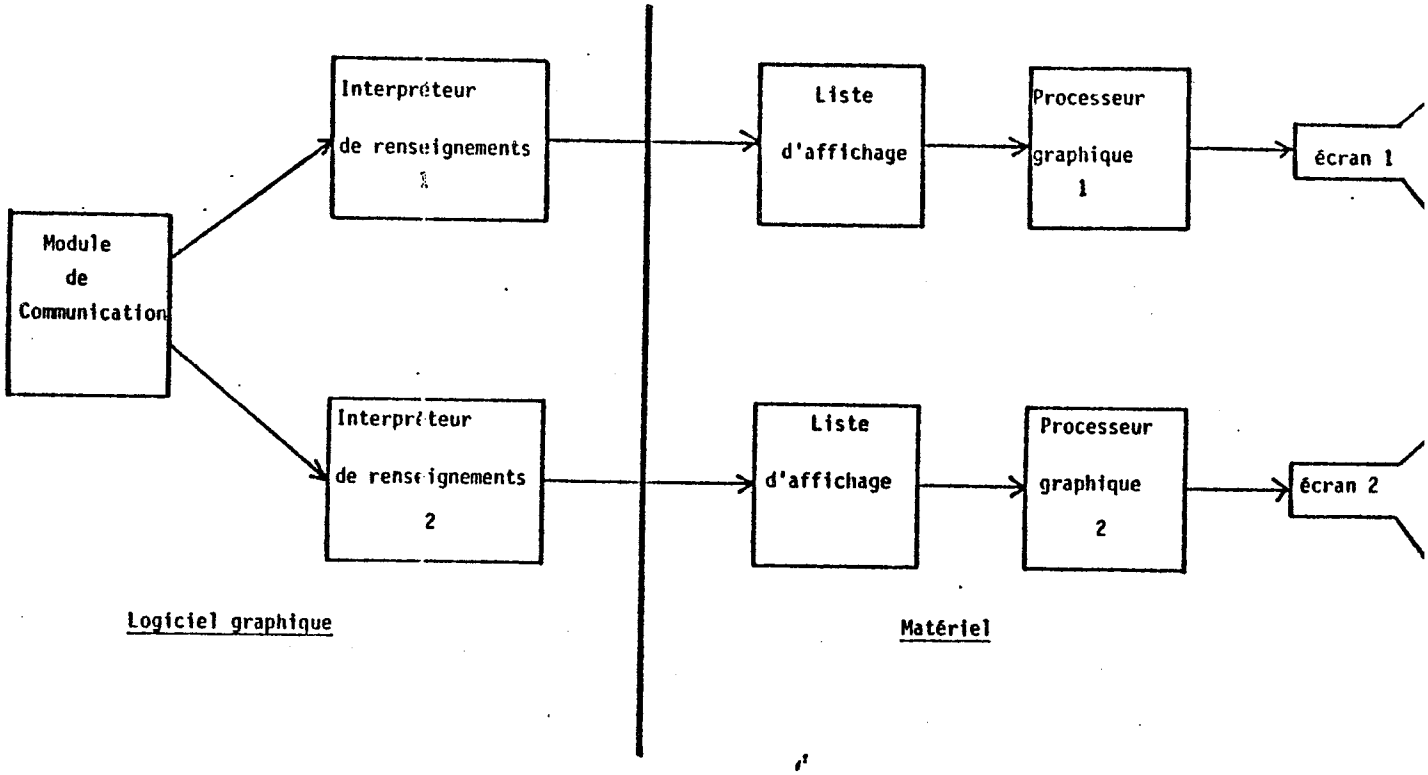


Figure V.2. : Prise en charge de terminaux à mémoire structurée

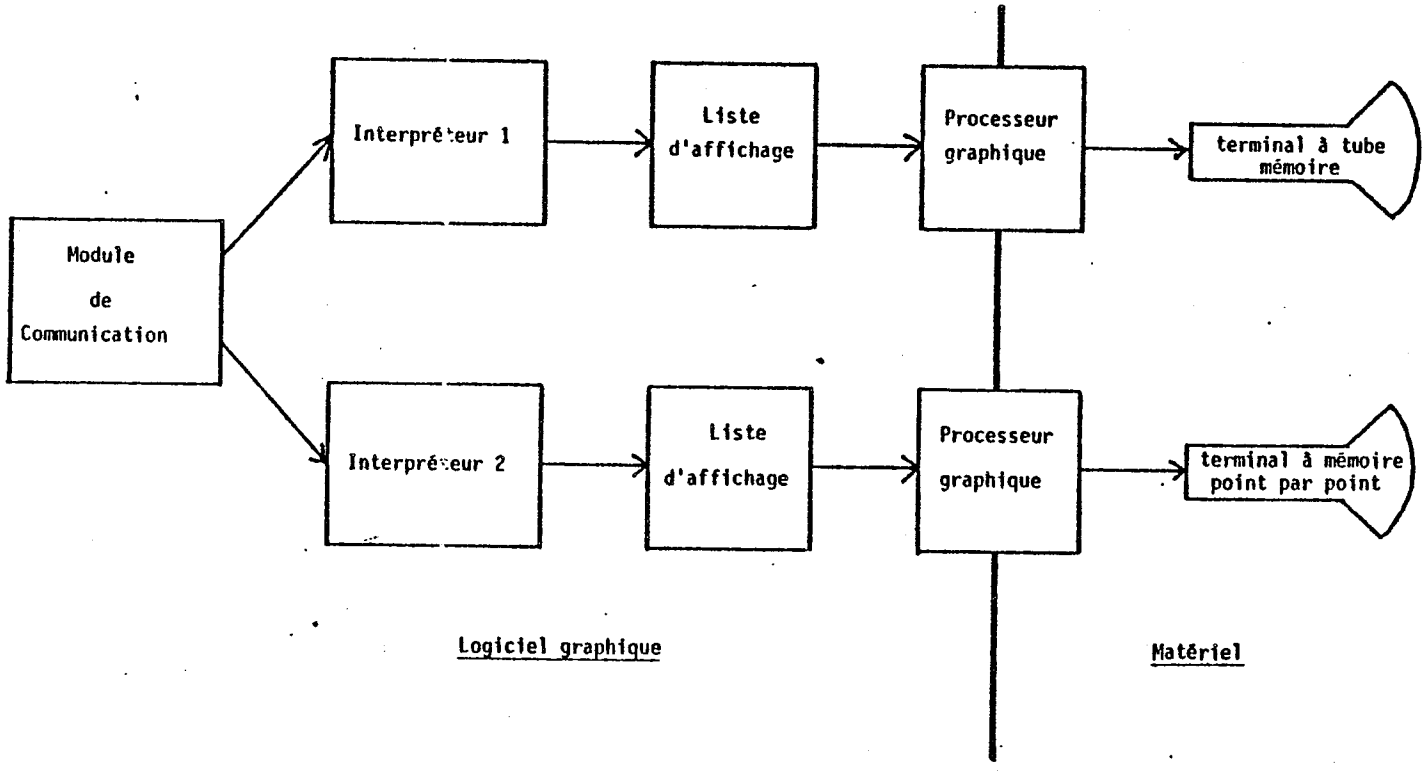


Figure V.3. : Prise en charge de terminaux à mémoire analytique

V-2- LE NOYAU DU LOGICIEL

Le rôle est essentiellement un rôle de communication avec l'application.

Les fonctions du noyau sont les suivantes :

- communication avec l'application,
- stockage et gestion des listes de "noms" et de certains renseignements envoyés par l'application,
- contrôle de la validité des paramètres reçus,
  (contrôle syntaxique et sémantique si possible).
- génération d'un code intermédiaire, indépendant des terminaux, destiné aux interpréteurs.

Nous examinerons successivement la réalisation de ces différentes fonctions.

Notons que, par cohérence avec le schéma adopté en V.1, le noyau ne stocke aucune donnée graphique (coordonnée, etc...).

V.2.1. La structure de données du noyau

Les données stockées au niveau du noyau concernant les renseignements relatifs à l'application, c'est-à-dire des renseignements fournis par les primitives de déclaration et de définition. Ces primitives sont les suivantes :

- POINTS, TEXTES — Déclaration
- MODE, FENETRE, CLOTURE — Définition

A chacune de ces fonctions correspond dans le noyau une structure de données qui est mise à jour lors de l'invocation de la primitive correspondante.

Notons que ces primitives de déclaration et de définition ne font pas intervenir l'interpréteur.

Au niveau du noyau on trouve donc les structures de données suivantes :

V.2.1.1. Table des figures

Cette table permet de stocker les adresses des différents paramètres des fonctions POINTS ET TEXTES.

Pour des raisons de simplicité, on stocke indépendamment les renseignements concernant les suites de points et ceux concernant les suites de caractères.

La table des figures contenant des sections de points conserve les renseignements suivants :

- nom de la figure,
- adresse du tableau des longueurs des sections,
- adresse de la suite des abscisses,
- adresse de la suite des ordonnées,
- adresse du tableau des marqueurs.

La table des figures contenant des sections de texte conserve le même type de renseignements :

- nom de la figure,
- adresse du tableau des longueurs des sections,
- adresse de la suite de caractères,
- adresse de la liste des abscisses des positions initiales,
- adresse de la liste des ordonnées des positions initiales.

Une même figure peut apparaître dans les deux tables, si elle contient à la fois des sections de points et des sections de texte.

Si une nouvelle suite de points (ou de textes) est définie pour une figure existant déjà, les nouveaux paramètres seront stockés à la place des anciens.

Notons que le logiciel ne fournit pas de primitive de "destruction" de figure, ce qui veut dire que l'on ne détruit jamais d'éléments dans les tables décrites ci-dessus ; on peut seulement associer de nouveaux éléments à une figure existant déjà.

V.2.1.2. La table des fenêtres

Celle-ci est destinée à recevoir les renseignements concernant les fenêtres définies par le programmeur grâce à la primitive FENETRE. Elle contient les renseignements suivants :

- nom de la fenêtre,
- abscisse du coin inférieur gauche de la fenêtre,
- ordonnée du coin inférieur gauche de la fenêtre,
- abscisse du coin supérieur droit de la fenêtre,
- ordonnée du coin supérieur droit de la fenêtre.

Ces coordonnées sont évaluées dans le système de l'utilisateur. Si de nouveaux paramètres sont définis pour une fenêtre existant déjà, ceux-ci viennent remplacer les anciennes valeurs.

V.2.1.3. La table des clôtures

Elle a la même structure que la table des fenêtres.

Elle contient les renseignements fournis lors de l'invocation de la primitive CLOTURE :

- nom de la clôture,
- abscisse du coin inférieur gauche de la clôture,
- ordonnée du coin inférieur gauche de la clôture,
- abscisse du coin supérieur droit de la clôture,
- ordonnée du coin supérieur droit de la clôture.

Ces coordonnées sont des valeurs entières comprises entre 0 et 100. Notons que le logiciel n'offre pas de primitive de "destruction" de fenêtre ou de clôture ; dans les cas, on ne détruit jamais d'éléments dans les tables, on peut seulement redéfinir les paramètres pour une clôture (ou fenêtre) existant déjà.

V.2.1.4. La table des modes déclarés

Elle est destinée à recevoir les renseignements fournis lors de l'invocation de la primitive MODE.

Pour simplifier la gestion, le stockage des modes relatifs aux sections de points est distinct de celui des modes associés aux sections de textes.

La table des modes associés aux sections de points contient les renseignements suivants :
- numéro de la figure à laquelle appartient la section concernée,
- numéro de la section concernée dans la figure,
- numéro de corrélation associé à la section,
- couleur du tracé,
- texture,
- épaisseur,
- intensité,
- type de représentation,
- type de marquage,
- système de coordonnées.

Notons que le type de marquage est codé spécialement et permet de retrouver le symbole utilisé et l'indice initial, si nécessaire. Les autres paramètres de description du mode sont des valeurs entières.

La table des modes associés aux sections de texte contient le même type de renseignements :
- numéro de la figure à laquelle appartient la section,
- numéro de la section concernée,
- numéro de corrélation associé à la section,
- taille des caractères,
- orientation du texte,
- couleur,
- épaisseur,
- intensité.

L'utilisateur peut ne pas définir de mode pour certaines sections (auquel cas le logiciel choisira les options standards) ; nous stockons donc dans ces tables, non pas les modes de toutes les sections, mais seulement ceux définis dans le programme d'application. Lors de l'invocation de la primitive MODE, le programmeur peut utiliser la convention définie en IV.3.5., concernant les numéros de figure et de section, ceci rend la gestion de la table des modes assez délicate.

V.2.2. La gestion des modes

Le problème est complexe pour plusieurs raisons :

- utilisation de la convention décrite en IV.3.5.,
- possibilité de définir des modes pour des sections non encore définies,
- possibilité de définir un nouveau mode pour une section (ou un ensemble de sections) en ayant déjà un.

Nous allons décrire ici le traitement de la primitive :

MODE (Nfig, Ncorrel, Nsection, descripteur)

Selon la convention citée en IV.3.5., Nfig peut prendre 3 valeurs qui définissent la portée du mode :

Nfig = F : on traite la figure F seule,
Nfig = 0 : on traite toutes les figures,
Nfig = -F : on traite toutes les figures,sauf la figure F.

Le traitement de la primitive mode se fait donc à deux niveaux :

- on établit la liste des figures concernées,
- on traite chaque figure indépendamment (niveau Figure),
- pour chaque figure on traite les sections concernées (Niveau section) (car le numéro de section répond à la même convention).

V.2.2.1. Gestion du mode pour une figure

Pour une figure F donnée, on peut trouver, dans la table des modes des définitions de modes particuliers à une section et un mode "général", c'est-à-dire relatif à toutes les sections, ou à toutes les sections sauf une.

La gestion des modes est basée sur le fait qu'un mode associé à une section particulière définie par le couple (F,S) l'emporte sur le mode général défini pour (F,0) ou (F, -S') s'il existe.

Pour simplifier l'écriture, nous identifierons désormais le mode au couple (figure, section) auquel il est associé.

Quand l'utilisateur définit un mode général (F,0) pour la figure, il faut éliminer tous les modes précédemment définis pour celle-ci. Ceci est réalisé en mettant à 0 le numéro de corrélation correspondant dans la table. (un vrai numéro de corrélation est strictement positif).

Quand l'utilisateur définit un mode particulier (F,S), on le stocke (s'il existe déjà, on remplace les anciennes valeurs par les nouvelles).

Quand l'utilisateur définit un mode général (F,-S), il faut éliminer tous les modes particuliers associés à la figure sauf (F,S) s'il existe. D'autre part, il faut éliminer le mode général existant (s'il y en a un) afin de toujours conserver un seul mode général par figure. Ceci doit être fait avec précautions car l'ancien mode général n'a pas nécessairement la même portée que le nouveau.

Le tableau de la figure V.4. décrit le traitement du mode associé à un couple (F, nsection) suivant la valeur de nsection et l'état de la table des modes. Nous constatons que le traitement le plus délicat est celui d'un mode (F,-S).

V.2.2.2. Gestion du mode pour l'ensemble des figures

Le traitement du mode pour une figure est pris en charge par un sous-programme.

- Lorsque Nfig est positif, on active ce sous-programme pour la figure concernée.
- Lorsque Nfig = -F, on active le sous-programme successivement pour toutes les figures sauf la figure F.

Mais ceci permet de définir un mode uniquement pour les figures existant déjà à cet instant dans le programme. Pour conserver le mode associé à (-F, nsection) le logiciel définit une "figure" de numéro 0, qui représentera toutes les figures qui ne sont pas encore définies et qui apparaîtront ultérieurement, on stocke le mode pour (0, nsection).

Reste le problème de la figure F elle-même, si un mode a déjà été défini pour cette figure, on le conserve. Si elle n'apparaît pas dans la table des modes, il faut indiquer que le mode défini pour (0, nsection) ne la concerne pas. Pour cela, on stocke (F, nsection) avec une corrélation égale à -1. Par convention, cette corrélation indique que le mode de la figure F n'est pas défini, même s'il existe un mode (0, nsection).

- Enfin, lorsque Nfig = 0, on active le sous-programme de mise à jour de la table, successivement pour toutes les figures existantes. Puis on active le sous-programme pour la "figure" 0.

nsection	état de la table	modes à éliminer	mode à stocker
0	quelconque	(F,S) ∀S	(F,0) nouveau mode
S > 0	le couple (F,S) existe	(F,S)	(F,S) nouveau mode
	le couple (F,S) n'existe pas		(F,S) nouveau mode
-S	le couple (F,S) existe	(F,S1) ∀ S1 ≠ S	(F,-S) nouveau mode on garde (F,S)
	le couple (F,S) n'existe pas, il existe un couple (F,0)	(F,0) (F,S1) ∀ S1 > 0	(F,S) avec l'ancien mode de (F,0) (F,-S) nouveau mode
	le couple (F,S) n'existe pas, le couple (F,-S) existe	(F,-S) (F,S1) ∀ S1 > 0	(F,-S) nouveau mode
	le couple (F,S) n'existe pas, il existe un couple (F,-S1) avec S1 ≠ S	(F,-S1) (F,S2) ∀ S2 ≠ S	(F,S) avec l'ancien mode de (F,-S1) (F,-S) nouveau mode
	il existe uniquement des couples (F,S1) avec S1 > 0 et S1 ≠ S	(F,S1) ∀ S1	(F,-S) nouveau mode
	aucune référence à la figure F		(F,-S) nouveau mode

FIGURE V.4. : GESTION DE LA TABLE DES MODES POUR UNE FIGURE

Finalement, en cours de déroulement d'un programme, on trouve dans la liste des numéros de figure :
- des figures particulières,
- la "figure" 0 qui regroupe toutes les figures dont le mode n'a pas encore été défini spécialement.

Notons que cette "figure" 0 est gérée de la même façon qu'une figure quelconque.

Nous avons renoncé à gérer des définitions de mode dont la portée soit supérieure à une figure, c'est pourquoi, lorsque le programmeur invoque MODE avec Nfig = 0 ou Nfig = -F, nous découpons le traitement en une suite de traitements plus simple au niveau de chaque figure.

Les deux tables de mode sont gérées de la même manière.

V.2.3. Le traitement de l'affichage

L'ensemble des sections à afficher est indiqué par :
- un numéro de figure,
- un numéro de corrélation.

Ces deux numéros peuvent obéir à la convention citée en IV.3.5. L'unité de base pour l'affichage est la section.

V.2.3.1. La liste des sections à afficher

Le tableau des modes nous fournira deux types de renseignements :
- la corrélation de chaque section (pour savoir si l'on doit afficher celle-ci ou non),
- le mode de la section.

Lorsque la primitive AFFICHER est invoquée, on commence par établir la liste des figures concernées. On traite les différentes figures successivement.

Traitement d'une figure F

On récupère le tableau des longueurs des sections.

On s'intéresse alors à chaque couple (F,S) représentant une section de la figure, pour savoir si on doit l'afficher et comment. Pour cela, on recherche dans le tableau des modes un couple (figure, section) minimum qui recouvre (F,S)

Cette recherche se fait de la façon suivante :

1 - recherche du couple (F,S),
2 - si (F,S) n'existe pas, recherche d'un couple (F,O) ou (F,-S1) avec S1 ≠ S,
3 - si on n'a rien trouvé, recherche du couple (O,S),
4 - si on n'a rien trouvé, recherche d'un couple (O,C) ou (O,-S1) avec S1 ≠ S
A la fin de cette recherche, on a trouvé un couple (figure,section) minimum qui recouvre (F,S), ou non.

paramètre Ncorrel	on n'a pas trouvé de couple qui recouvre (F,S)	on a trouvé un couple qui recouvre (F,S)
	La section S n'a pas de mode ni de corrélation	La section S à un mode M et une corrélation C
0	On affiche la section avec le mode standard	On affiche la section avec le mode M
< 0	On affiche la section avec le mode standard	- on affiche la section avec le mode M si ABS(NCORREL) ≠ C - on n'affiche pas la section si ABS(NCORREL) = C
> 0	On n'affiche pas la section	- on affiche la section avec le mode M si NCORREL = C - on n'affiche pas la section si NCORREL ≠ C

FIGURE V.5. : AFFICHAGE D'UNE SECTION

On fait ce traitement uniquement s'il y a des modes définis.

V.2.3.2. Traitement des coordonnées

En fonction du mode attaché à la section étudiée, on effectue deux traitements sur les coordonnées au niveau du noyau :

- si les coordonnées sont relatives, on calcule leurs valeurs en absolu (le premier point est connu en absolu)
- on effectue le découpage afin de ne pas transmettre des points inutiles à l'interpréteur.

V.2.3.3. Déroulement de la fonction d'affichage

- En fonction du paramètre Nfig, on commence par établir la liste des figures concernées par l'affichage.

- Pour chaque figure, on récupère chez l'utilisateur la liste des longueurs des sections.

- Pour chaque section, on examine le tableau des modes et corrélations pour savoir si elle doit être affichée et avec quel mode.

Si on ne l'affiche pas, on passe à la section suivante.

Si on l'affiche, on envoie à l'interpréteur les renseignements nécessaires à l'affichage :

- mode,
- bornes de la fenêtre,
- bornes de la clôture.

Ces renseignements sont conservés par l'interpréteur.

Ensuite, on traite les coordonnées :

- récupération chez l'utilisateur,
- passage en absolu,
- découpage

et on envoie celles-ci à l'interpréteur en précisant leur nombre.

Enfin, pour chaque section, on transmet l'étiquette et la position initiale si nécessaire.

On fait ce traitement pour toutes les sections de la figure et pour toutes les figures concernées.

On fait évidemment le même raisonnement pour les sections de textes. Dans ce cas, on a deux phases de dialogue avec l'interpréteur :

- envoi des renseignements :
  - mode,
  - bornes de la fenêtre,
  - bornes de la clôture.
- envoi de la section de texte, avec le nombre de caractères qu'elle contient.

Remarque :

Lorsque le type de représentation d'une suite de points est la ligne brisée, on passe également une ligne brisée à l'interpréteur. Pour cela, lorsque la ligne brisée "sort" de la fenêtre, on envoie à l'interpréteur la liste de coordonnées en cours ; puis quand la ligne brisée "rentre" à nouveau dans la fenêtre, on engendre une nouvelle commande pour l'interpréteur. Ainsi, celui-ci tracera uniquement des lignes brisées entièrement contenues dans la fenêtre.

Notons enfin que les coordonnées transmises sont évaluées dans le système de l'utilisateur et le passage au repère écran et la mise en page de l'écran utilisé sont à la charge de l'interpréteur. Le passage d'un numéro de section à l'interpréteur est réalisé par une numérotation interne qui identifie chaque section.
- pour une section de points le codage est le suivant :

$$1000 * Nfig + Nsection$$

- pour une section de texte, c'est le même codage en négatif :

$$-1000 * Nfig - Nsection.$$

Ceci permet de distinguer les sections de points et celles de texte portant les mêmes numéros dans les mêmes figures.

V.2.4. Le traitement de l'effacement

Lors de l'invocation de la primitive EFFACER, le noyau analyse les paramètres Nfig et Ncorrel. En fonction de leurs valeurs, en appliquant le même algorithme de recherche dans le tableau des modes que pour l'affichage, on peut établir la liste des sections à effacer.

Cette liste des noms de sections à effacer est transmise à l'interpréteur qui se chargera de l'effacement effectif.

V.2.5. Le traitement des messages

En ce qui concerne les messages, le traitement à la charge du noyau consiste à préparer à partir de la chaîne passée en paramètre, le message qui devra être transmis à l'interpréteur pour affichage. De plus, il doit repérer les zones réservées. Le traitement se fait en deux phases.

V.2.5.1. Génération de la chaîne de caractères à afficher

Le noyau analyse la chaîne passée en paramètre :
- il conserve les zones de texte à afficher,
- il analyse les demandes de réservation de zones,
- il remplit ces zones avec des espaces.

La chaîne ainsi constituée est envoyée à l'interpréteur avec son numéro de message et sa longueur.

Comme pour les sections, l'affichage proprement dit du message est à la charge de l'interpréteur.

V.2.5.2. La réservation des zones

Les indications concernant les zones réservées sont stockées au niveau de l'interpréteur pour deux raisons :
- c'est l'interpréteur qui prendra en charge la réalisation de la fonction d'entrée de valeurs,
- les renseignements nécessaires au repérage d'une zone ne sont pas les mêmes suivants les terminaux utilisés.

Lors de l'analyse du message, le noyau repère l'emplacement des zones dans le message et leur longueur.

Dans une seconde phase de dialogue avec l'interpréteur, le noyau lui fournit les renseignements suivants :
- nombre de zones réservées dans le message,
- pour chaque zone :
  - . emplacement dans le message,
  - . longueur.

Ces données sont récupérées par l'interpréteur et utilisées pour calculer les renseignements qu'il aura besoin de conserver.

V.2.5.3. L'effacement des messages

Lors de l'invocation de la primitive EFFMESSAGE, le rôle du noyau est réduit au minimum : il transmet seulement à l'interpréteur le numéro du message à effacer.

V.2.5.4. L'édition alphanumérique

Lors de l'invocation des primitives EDENTIER, EDREEL et EDCHAINE, le noyau se charge de la mise sous forme de chaîne de caractères de la variable à éditer.

Le noyau dispose donc d'un sous-programme assurant le passage d'une valeur entière ou réelle, à sa représentation sous forme de chaîne de caractères, suivant un format désiré.

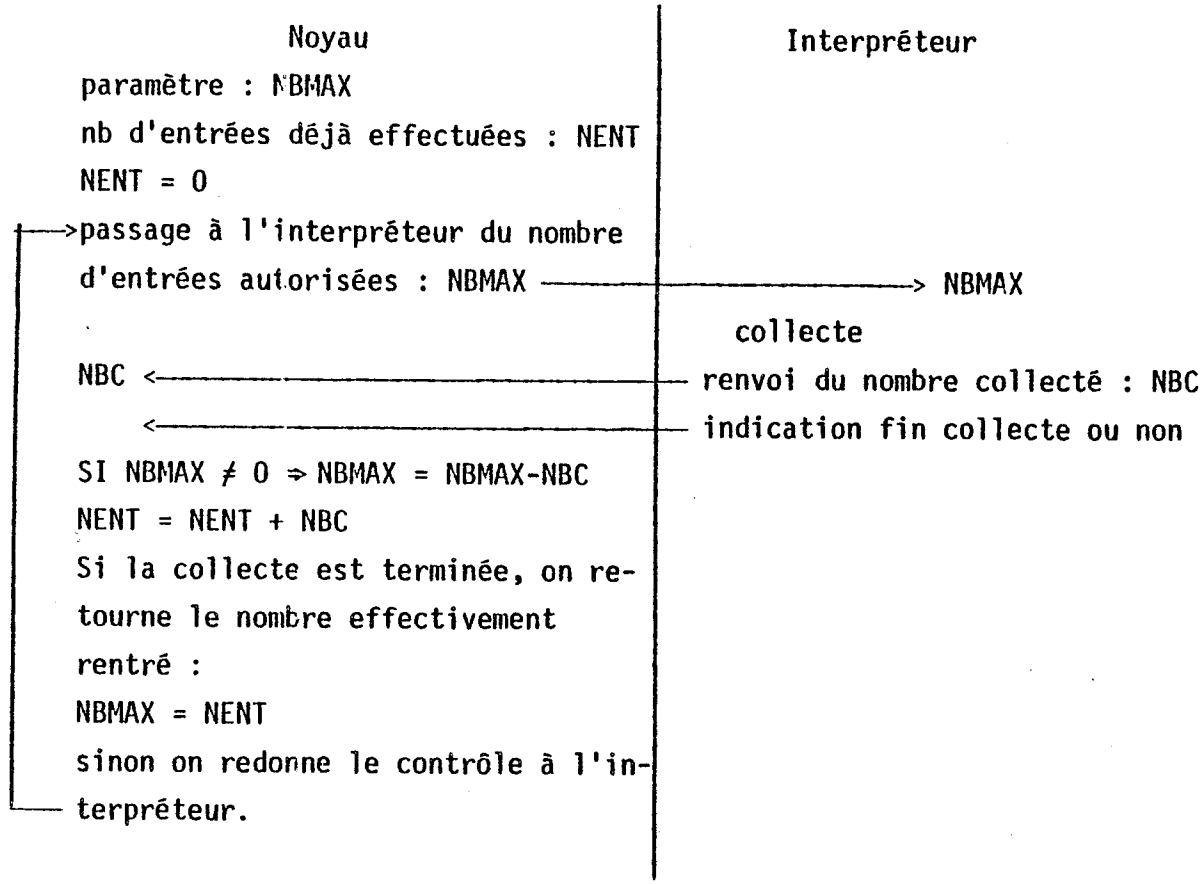
Dans les trois cas, le noyau fournit à l'interpréteur :
- le numéro du message auquel appartient la zone,
- le numéro de la zone dans le message,
- la longueur de la chaîne à éditer,
- la chaîne elle-même.

V.2.6. Les fonctions de dialogue

La réalisation des fonctions est prise en charge par l'interpréteur, le noyau joue seulement un rôle de contrôle et de communication. Dans tous les cas, le programmeur fournit un nombre maximum d'entrées : NBMAX.

La collecte peut se faire par plusieurs appels successifs à l'interpréteur car la zone de communication a une taille limitée. L'interpréteur retournera donc les résultats avec une indication précisant si la collecte est terminée ou non. Le noyau comptabilise les entrées déjà effectuées grâce à une variable intermédiaire NENT et donner à nouveau le contrôle à l'interpréteur si nécessaire.

Le schéma général des quatre fonctions de dialogue est donc le suivant :



V.2.6.1. L'introduction de valeurs

En plus du nombre d'entrées autorisées, le noyau fournit à l'interpréteur le numéro du message sur lequel se fera l'introduction, ainsi que le type de valeur attendu : réel, entier ou chaîne.

En retour, l'interpréteur renvoie des entiers, des réels, ou des chaînes de caractères. Le passage de la chaîne alphanumérique à la valeur entière ou réelle est assuré par l'interpréteur, car on pourrait imaginer d'utiliser des dispositifs tels que des potentiomètres pour introduire ces valeurs. Ainsi pour rester indépendant du matériel, il est nécessaire de collecter des valeurs et non des chaînes de caractères introduites au clavier. Dans le cas de l'introduction de chaînes, l'interpréteur retourne avec chaque chaîne, la longueur de celle-ci.

V.2.6.2. La désignation

Le noyau fournit à l'interpréteur le paramètre précisant quelles figures peuvent participer à l'opération.

En retour, l'interpréteur renvoie pour chaque désignation :

- numéro de figure,
- numéro de section,

et dans le cas de désignation d'un message :

- 0,
- numéro de message.

V.2.6.3. La collecte de coordonnées

Le noyau fournit à l'interpréteur les renseignements utiles pour que celui-ci retourne des coordonnées évaluées dans le système de l'utilisateur :

- bornes de la fenêtre utilisée,
- bornes de la clôture utilisée.

Lorsque la collecte est terminée, ou lorsque l'opérateur signale une fin de section, l'interpréteur rend le contrôle au noyau (l'élément de base, comme pour l'affichage, est ici la section).

Le noyau doit alors stocker les résultats dans les tableaux de l'utilisateur, derrière les points existant déjà. Pour cela, il faut d'abord récupérer le tableau des longueurs des sections de la figure, pour pouvoir calculer le déplacement dans les tableaux d'abscisses et d'ordonnées. Il faut également mettre à jour le tableau des longueurs. La figure V.6. décrit l'évolution des tableaux d'une figure, après une collecte de coordonnées dans cette figure.

V.2.6.4. Le Menu

Le rôle du noyau est ici plus important.

Il analyse la chaîne de description du menu et en tire deux types de renseignements :

- la liste des numéros autorisés,
- la liste des noms à afficher.

Ces deux types de données sont traités séparément.

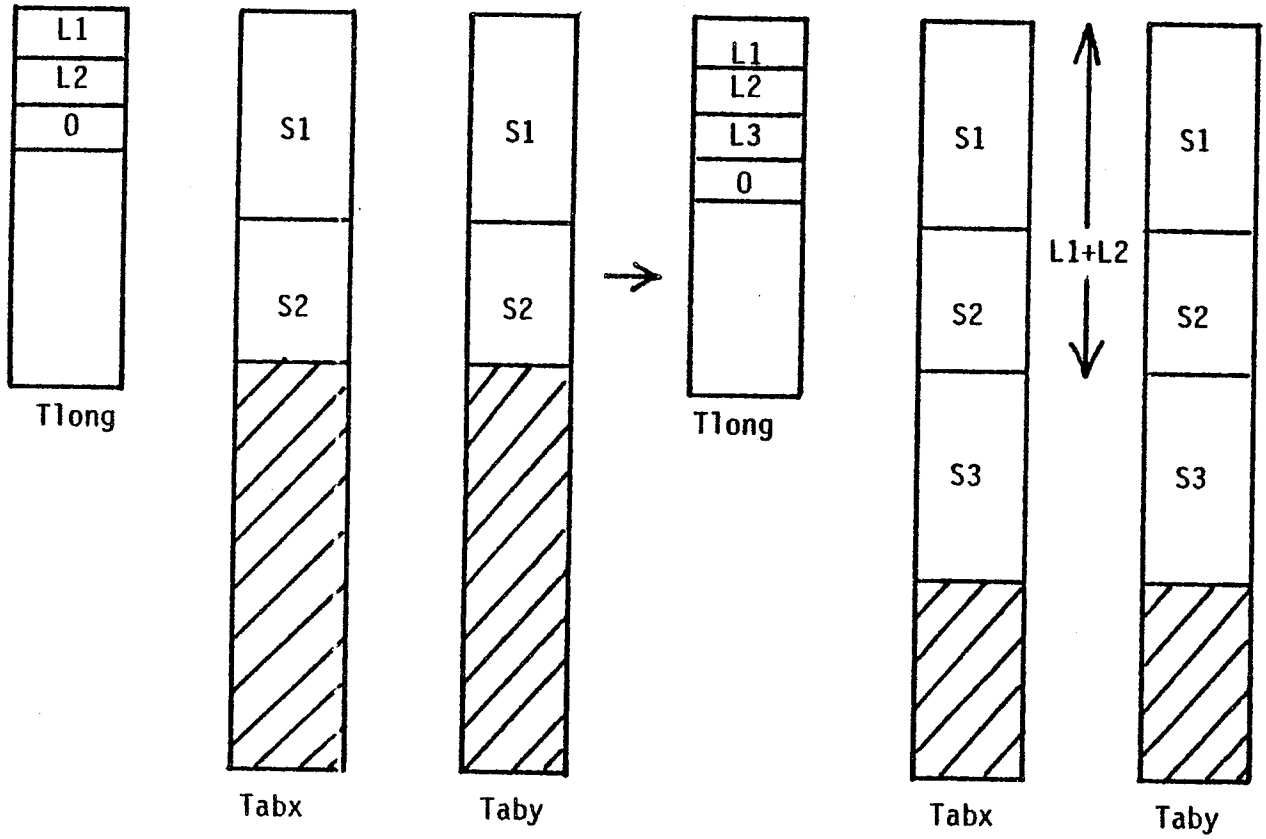


Figure V.6. : Collecte de coordonnées

Dans un premier temps, pour afficher les noms sur l'écran, en restant indépendant du matériel, on utilise la primitive MESSAGE. Ces messages sont engendrés sous un numéro système qui ne peut pas être confondu avec un numéro du programme d'application, dans la clôture 0 (totalité de l'écran). Dans un second temps le noyau fournit à l'interpréteur la liste des numéros autorisés.

En retour, l'interpréteur renvoie soit un numéro (compris entre 1 et 28) soit l'ordonnée d'un nom désigné sur l'écran. Cette ordonnée est un entier compris entre 9 et 100 (car elle est évaluée par rapport à la clôture 0). Si c'est une coordonnée, le noyau se charge de retrouver le nom affiché à cet endroit et son numéro.

Ainsi, par l'intermédiaire de l'utilisation de la primitive MESSAGE, dans la clôture 0, on assure l'indépendance du noyau par rapport au matériel.

A la fin de l'exécution d'un menu, les messages sont effacés.

V.2.7. Le contrôle à l'exécution

Un certain nombre de contrôles sont effectués par le logiciel sur les paramètres des différentes primitives.

Quelques uns sont effectués par l'interpréteur, mais la plupart sont effectués au niveau du noyau.

Lorsque le logiciel détecte une erreur, il interrompt l'exécution de la primitive en cours, afin d'empêcher le programme d'application de perdre le contrôle.

Un message est émis, qui a toujours le format suivant :

<nom-de-fonction><message><paramètre-erroné>

Le tableau de la figure V.7 donne la liste des principaux messages d'erreur, les primitives concernées, ainsi qu'une indication précisant si l'erreur est détectée par le noyau (N) ou par l'interpréteur (I).

Messages		Fenêtre	Clôture	Points	Textes	Mode	Afficher	Effacer	Message	Effmessage	Edentier	Edréel	Edchaîne	Rval	Nval	Cval	Identification	Position	Menu
Numéro incorrect	N	X	X																
Coordonnées incorrectes	N		X						X										
Figure non déclarée	N					X	X	X									X	X	
Chaîne incorrecte	N					X			X										X
Corrélation inexistante	N						X	X											
Fenêtre non déclarée	N						X											X	
Clôture non déclarée	N						X		X									X	
Zone (xx,xx) inexistante	I										X	X	X	X	X	X			
Message inexistant	I									X	X	X	X	X	X	X			
Edition impossible	I										X	X	X						
Réponse incorrecte	I													X	X	X	X	X	X
Trop de déclarations	N I	X	X	X	X				X										
Trop de zones réservées	I								X										

Figure V.7 : Message d'erreur

V.3. LE CODE INTERMÉDIAIRE INDÉPENDANT DU MATÉRIEL

Nous avons examiné les différentes tâches du noyau du logiciel. Mais nous constatons en fait que le rôle principal du noyau est d'engendrer un code intermédiaire, indépendant du terminal utilisé. Ce code indépendant est ensuite interprété pour le terminal réel.

Dans la réalisation du prototype décrit ici, la zone de communication entre le noyau et les interpréteurs comporte trois zones :

- une zone "commande" (BUFCDE) de un entier destinée à recevoir le code opération de la commande intermédiaire,
- une zone opérande de 36 entiers (BUFN),
- une zone opérande de 256 réels (BUFR).

C'est par cette zone de communication que se font tous les échanges entre noyau e interpréteurs (et inversement). Dans le cas présent, cette zone est représentée p un "common", car le logiciel est écrit en Fortran.

L'utilisation d'une telle zone commune permet d'assurer une séparati complète entre les deux modules ; ce qui est fondamental si l'on envisage un jou de répartir le logiciel sur deux calculateurs.

Le tableau suivant décrit les commandes intermédiaires, ainsi que le opérandes, commandes qui sont engendrées lors de l'invocation des différentes pr mitives par le programme d'application.

Pour donner une description complète du code intermédiaire, nous re- groupons ici les commandes correspondant aux primitives de déclaration. Comme no l'avons déjà dit, il n'y a pas, pour celles-ci, activation de l'interpréteur, pu que les tables sont contenues dans le noyau. Cependant, l'interpréteur sera acti dans ce cas, quand on travaille, non pas sur un terminal graphique, mais sur un terminal alphanumérique, pour obtenir une trace du programme. Nous reviendrons s cet interpréteur ultérieurement. Le code engendré est décrit dans la Figure V.9.

PRIMITIVE	CODE	OPERANDES
AFFICHER (le traitement se fait section par section)	ATT attributs d'une section de points	- nom de la section = 1000* nfig+nsection - corrélation de la section - bornes de la clôture - bornes de la fenêtre - paramètres de définition du mode
	COOR envoi d'une suite de points	- nombre de points - suite de couples (X,Y) (X,Y exprimés dans le repère de l'utilisateur)
	LEG envoi d'une étiquette	- étiquette (sur 4 caractères) - position initiale (X,Y)
	ATT attributs d'une section de texte	- nom de la section = -1000*Nfig-Nsection - corrélation de la section - bornes de la fenêtre - bornes de la clôture - paramètres de définition du mode
	CAR envoi d'une suite de caractères	- nb de caractères du texte - la chaîne de caractères (4 caractères par mot) - position initiale (X,Y)

Figure V.8.a : Code intermédiaire

PRIMITIVE	CDE	OPERANDES
EFFACER	EFFS effacement liste de section	- nombre de sections à effacer - noms des sections à effacer : { 1000*Nfig + Nsection (points) { 1000*Nfig - Nsection (textes) Si le nombre de sections à effacer = 0 On efface toutes les sections de toutes les figures
MESSAGE	MESS envoi du message à afficher	- numéro du message - longueur du message à afficher - coordonnées de la position initiale (en 1/10000 d'écran) - message à afficher
	MESS liste des zones réservées	- numéro du message - nombre de zone (en négatif pour ne pas le confondre avec une longueur) - emplacement dans le message - longueur } pour chaque zone
EFFMESSAGE	EFFM effacement d'un message	- nom du message ou 0, auquel cas on effacera tous les messages
EDENTIER EDREEL EDCHAINE	EDEN EDRE EDCH édition alpha-numérique	- numéro du message - numéro de la zone - type d'édition (1:réel, 2:entier, 3:chaîne) - longueur de la chaîne à éditer - chaîne à éditer

Figure V.8. b

PRIMITIVE	CDE	OPERANDES
RVAL	RVAL NVAL	- nombre d'entrées maximum (ou 0) - numéro du message - type (entier ou réel)
	←	- nombre d'entrées effectuées - liste des valeurs introduites
CVAL	CVAL	- nombre d'entrées maximum (ou 0) - numéro du message
	←	- nombre d'entrées effectuées - pour chaque chaîne : - longueur - chaîne introduite
MENU	MENU envoi de la liste des numéros autorisés	- nombre d'entrées maximum (ou 0) - suite de 28 entiers valant 0 ou 1 { 0 : numéro interdit { 1 : numéro autorisé
	←	- nombre d'entrées collectées - liste des entrées : - entier ⩽ 28 ⇒ numéro - entier > 100 ⇒ coordonnée par rapport à la clôture 0, augmentée de 100 (désignation d'un nom sur l'écran)

Figure V.8.c

PRIMITIVE	CDE	OPERANDE
POSITION traitement section par section	POSIT	- nombre d'entrées maximum (ou 0) - bornes de la clôture - bornes de la fenêtre
	←	- nombre de points collectés - indication de fin de section ou non - indication de fin de collecte ou non - liste des couples (X,Y) coordonnées évaluées dans le repère de l'utilisateur
IDENTIFICATION	IDENT	- nombre de désignations maximum (ou 0) - numéro de la figure participant à l'opératio (positif, négatif, ou nul)
	←	- nombre de désignations effectuées - pour chaque désignation : - numéro de figure - numéro de section et pour la désignation du message : - 0 - numéro de message
activé lors de l'invocation de la première primitive	INIT INIALISATION	/

Figure V.8.d

PRIMITIVE	CDE	OPERANDES
FENETRE	FEN	- numéro de la fenêtre - bornes de la fenêtre (dans le repère de l'utilisateur
CLOTURE	CLO	- numéro de la clôture - bornes de la clôture (en 1/100 d'écran)
POINTS	POI	- numéro de la figure
TEXTES	TEX	- numéro de la figure
MODE	MODE	- paramètre Nfig de la primitive MODE - paramètre Ncorrel - paramètre Nsection - type { points / textes - valeurs des différents paramètres définissants le mode

Figure V.9 : Code intermédiaire pour les primitives de déclaration

V.4. L'INTERPRÉTEUR : SES FONCTIONS GÉNÉRALES

V.4.1. Les fonctions de l'interpréteur

Son rôle est de passer du code indépendant du matériel, au code du terminal et inversement.

En ce qui concerne l'affichage, l'interpréteur devra assurer la gestion de l'image réelle. Pour cela, il devra gérer une liste d'affichage structurée si le terminal ne dispose pas d'une mémoire d'entretien.

L'interpréteur exécutera les commandes du code intermédiaire en essayant d'utiliser au maximum les possibilités offertes par le matériel. Lorsque ce n'est pas possible, il devra assurer la simulation des options manquantes, dans la mesure du possible.

Pour le dialogue, l'interpréteur est chargé d'activer, pour chaque primitive, les dispositifs autorisés, il devra assurer la gestion de ces dispositifs et collecter les résultats. De plus, l'interpréteur doit gérer la structure de données contenant les renseignements sur les divers éléments affichés.

V.4.2. La structure de données

Nous décrivons ici les renseignements que l'on doit conserver quel que soit le type de matériel utilisé. Ces données doivent permettre d'identifier et d'effacer sélectivement.

On trouve dans tout interpréteur trois tables :
- la table des messages,
- la table des zones réservées,
- la table des sections.

V.4.2.1. La table des sections affichées

Elle contient, pour chaque section affichée les indications suivantes :
- nom de la section (numéro interne)
- corrélation de la section,
- pointeur vers le début de la zone associée à la section, dans la liste d'affichage,
- longueur de la zone associée à la section dans la liste d'affichage.

Lorsqu'une section est effacée, sa définition dans la table est détruite par mise à zéro de son nom. Ainsi, la place pourra être récupérée dans la table.

V.4.2.2. La table des messages affichés

Elle contient, pour chaque message affiché les mêmes renseignements que pour une section :

- numéro du message,
- pointeur vers le début de la zone associée au message, dans la liste d'affichage,
- longueur de la zone associée au message, dans la liste d'affichage.

De plus, les messages étant gérés comme des formats Fortran pour les entrées alphanumériques (voir IV.5.2.). On doit conserver deux renseignements permettant de savoir dans quelle zone réservée doit avoir lieu la prochaine introduction de valeur. Pour cela, on conserve :
- le numéro de la zone dans laquelle se fera la prochaine entrée (zone courante),
- le nombre total de zones réservées du message.

(pour pouvoir boucler quand on est arrivé sur la dernière zone et pour pouvoir reculer lors d'une demande d'annulation).

Lorsqu'un message est effacé, sa définition dans la table est détruite par mise à zéro de son numéro. Ainsi la place pourra être récupérée dans cette table également.

V.4.2.3. La table des zones réservées

Elle contient pour chaque zone réservée les informations suivantes :
- numéro interne de la zone (100*Nmessage +Nzone)
- pointeur vers le début de la zone dans la liste d'affichage
- longueur de la zone.

On trouve au moins ces trois renseignements dans tous les interpréteurs. Pour certains terminaux, l'interpréteur est obligé de conserver des informations supplémentaires, spécifiques au matériel.

Les trois structures de données décrites ci-dessus sont présentes dans tous les interpréteurs. Il est clair que la notion de pointeur n'a pas la même signification dans tous les cas :
- ce peut être une adresse de la mémoire d'entretien quand celle-ci existe,
- mais si le terminal ne dispose pas d'une mémoire d'entretien, le pointeur se réfère alors à la liste d'affichage simulée par le logiciel.

V.4.3. La mise en page de l'écran

Une fonction importante de l'interpréteur est la mise en page de l'écran du terminal utilisé. Il est clair qu'à ce niveau les dimensions et la forme de l'écran interviennent, mais le raisonnement et le calcul sont toujours les mêmes c'est pourquoi nous les mentionnons ici.

Le problème est le suivant : connaissant une fenêtre et une clôture en 1/100 d'écran, l'interpréteur doit calculer les bornes de la clôture réelle, dans le système de coordonnées de l'écran. La clôture réelle est, non pas la clôture demandée dans le programme d'application, mais la plus grande clôture homothétique de la fenêtre, inscriptible dans la clôture d'origine et centrée par rapport à celle-ci.

V.4.3.1. Calcul de la clôture réelle

Nous indiquons ici le raisonnement à suivre : il faut tout d'abord calculer les bornes de la clôture demandée par l'application, dans le repère écran réel, c'est-à-dire qu'il faut transformer les coordonnées comprises entre 0 et 100 en coordonnées écran. Le résultat dépend de la forme du terminal : une clôture "carrée" en centièmes d'écran donnera effectivement une clôture carrée sur un écran carré, mais le résultat sera une clôture rectangulaire sur un écran rectangulaire.

A l'intérieur de cette clôture d'origine, déformée en fonction de la forme de l'écran réel, il faut chercher la plus grande clôture homothétique de la fenêtre. Ce travail est commun à tous les interpréteurs, c'est pourquoi nous le détaillons ici (voir figure V.10.).

Soient LF et HF la longueur et la hauteur de la fenêtre,

Soient LC et HC la longueur et la hauteur de la clôture.

On calcule les rapports $\frac{LF}{LC}$ et $\frac{HF}{HC}$ pour savoir quelle dimension de la clôture pourra être gardée sans modification. Si $\frac{LF}{LC} > \frac{HF}{HC}$ on gardera la longueur LC de la clôture et on modifiera sa hauteur, pour cela on calcule la hauteur moyenne de la clôture d'origine :

$$Ycentre = \frac{1}{2} (sdy1 + igy1)$$

les nouvelles bornes de la clôture en y seront donc :

$$\begin{cases} igy2 = Ycentre - \frac{1}{2} \frac{LC}{LF} \times HF \\ sdy2 = Ycentre + \frac{1}{2} \frac{LC}{LF} \times HF \end{cases}$$

Inversement, si $\frac{LF}{LC} < \frac{HF}{HC}$, on garde la hauteur de la clôture et on modifie sa longueur en calculant :

$$Xcentre = \frac{1}{2} (igx1 + sdx1)$$

$$\begin{cases} igx2 = Xcentre - \frac{1}{2} \frac{HC}{HF} \times LF \\ sdx2 = Xcentre + \frac{1}{2} \frac{HC}{HF} \times LF \end{cases}$$

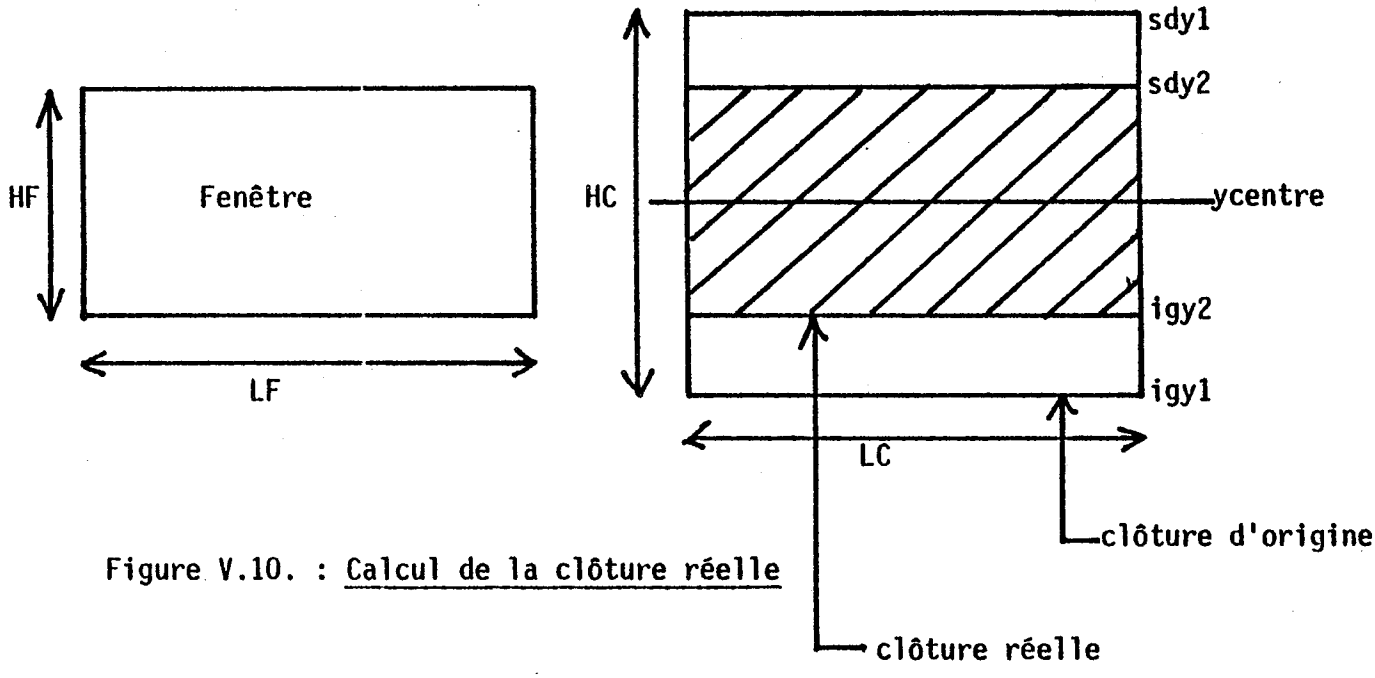


Figure V.10. : Calcul de la clôture réelle

Ce calcul est tout à fait général; quand on dispose des coordonnées de la clôture d'origine dans le repère écran, la suite des calculs est entièrement indépendante de la forme de l'écran,de celle delaclôture d'origine et de celle de la fenêtre.

V.4.3.2. Comparaison avec la mise en page sur console virtuelle

Nous rappelons que dans la première réalisation de GRIGRI (voir chapitre III), la mise en page était assurée au niveau d'un écran virtuel de forme carrée. Cet écran était ensuite placé dans le plus grand carré visualisable sur l'écran réel.

Prenons le cas où la fenêtre est homothétique de l'écran réel (rectangulaire), la figure V.11 montre les deux mises en page, et se passe de commentaires

Cette méthode permet une utilisation de l'écran entier quelque soit sa forme.

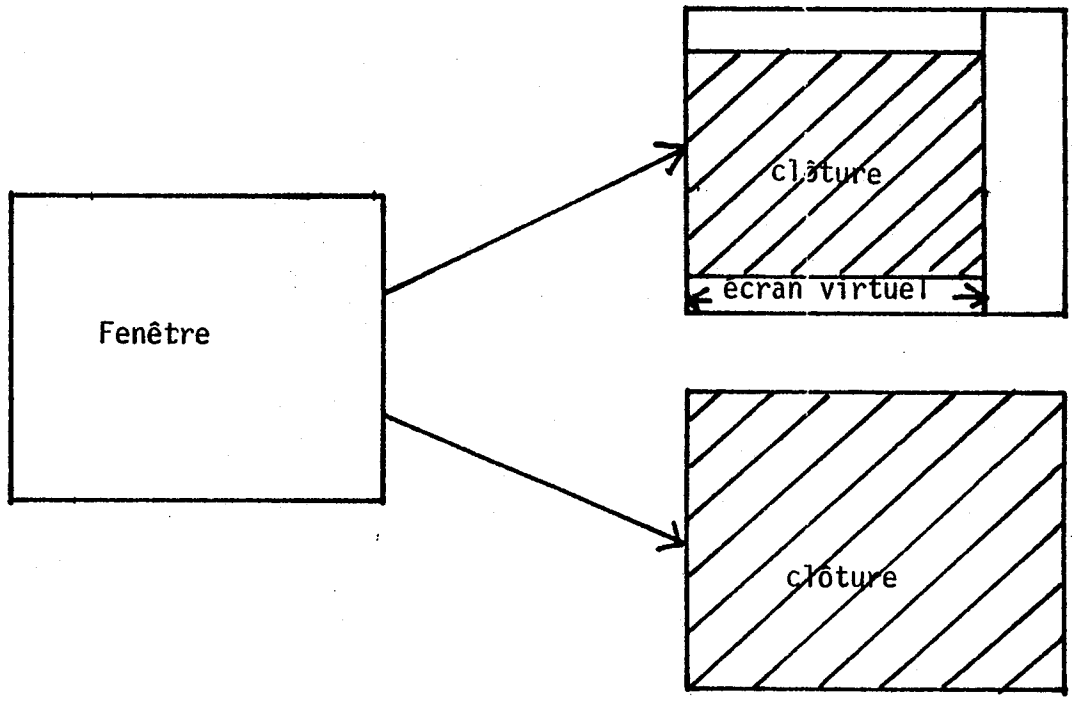


Figure V.11. : Comparaison des mises en page

V.4.4. La gestion des tables

Quelque soit le type d'interpréteur, la gestion des tables de la structure de données est la même.

L'élimination d'un élément dans une table se fait par mise à zéro du numéro correspondant (tous les numéros sont différents de zéro).

Lorsqu'on efface une section, elle doit être éliminée de la table des sections ; de même pour un message ; de plus, dans ce dernier cas, les zones associées au message doivent être éliminées de la table des zones réservées.

Lorsqu'on affiche une nouvelle section, celle-ci est rangée dans la table à la suite des sections existant déjà.

Ainsi, par affichages et effacements successifs, il peut arriver que la table des sections soit pleine ; on active alors un algorithme de retassement de la table. Cet algorithme travaille avec deux pointeurs et intervertit les cases libres et les cases occupées, selon le schéma de figure V.12.

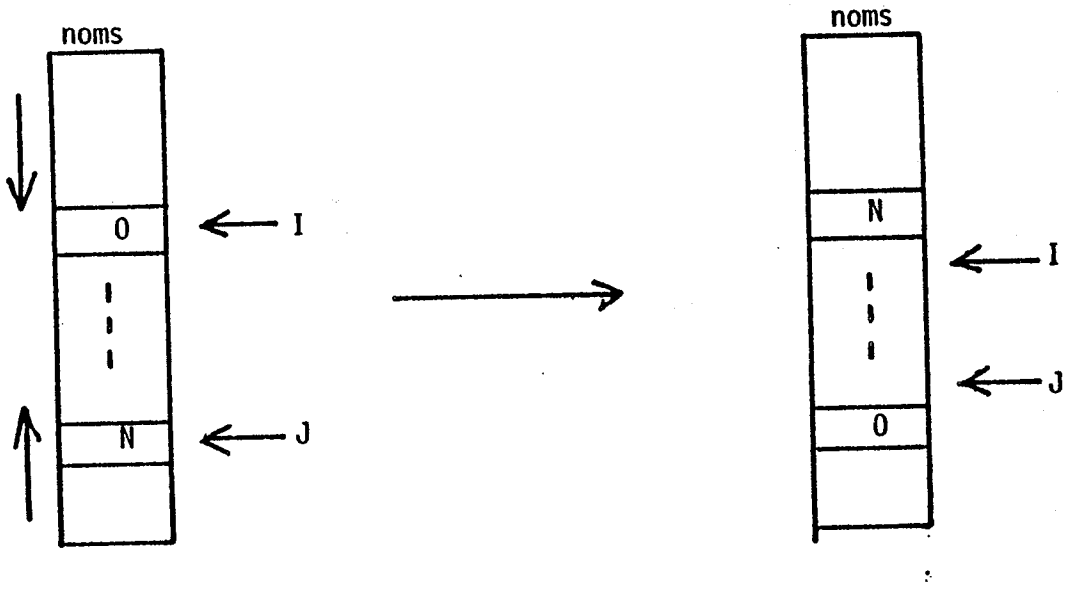


Figure V.12. : Retassement des tables

La même méthode est utilisée pour réorganiser la table des messages et la table des zones réservées.

Notons que dans le noyau, la table des modes déclarés est traitée de la même façon en cas de débordement.

V.4.5. La gestion de la liste d'affichage

La liste d'affichage peut être contenue dans la mémoire d'entretien d terminal si celui-ci en possède une, ou peut être simulée par le logiciel. Dans l deux cas, elle contient une description synthétique de l'image. Les commandes de liste de visualisation sont regroupées pour former des entités graphiques de deu types :

- les sections,
- les messages.

Une "section" de la liste d'affichage regroupera toutes les commandes graphiques élémentaires nécessaires à l'affichage d'une section de points ou de t te ; de même pour les messages. Nous mentionnons ici la gestion de la liste d'aff cgage, car celle-ci est la même quelle que soit la nature de cette liste.

V.4.5.1. Affichage

L'expérience nous a prouvé que les sections et les messages étaie traités différemment par le programmeur d'application (Effacement des sections en conservant les messages pour le dialogue, ou inversement effacement des messages pour afficher des sections sur la totalité de l'écran). C'est pourquoi nous avons décidé de les traiter différemment au niveau de la liste d'affichage, pour avoir gestion plus simple et éviter de perdre de la place. Le principe est le suivant :

- les sections sont stockées dans la partie haute de la liste d'affichage, en des cendant,
- les messages sont stockés dans la partie basse de la liste d'affichage, en remo tant.

Cette approche nous oblige à disposer d'une fonction de saut dans la liste d'affichage : celle-ci existe dans les mémoires d'entretien, mais il faudra également la réaliser lorsque la liste est simulée par logiciel.

Les figures V.13 et V.14 montrent le stockage d'une section et celui d'un message.

La gestion des sections est celle des messages sont ainsi complètement indépendantes.

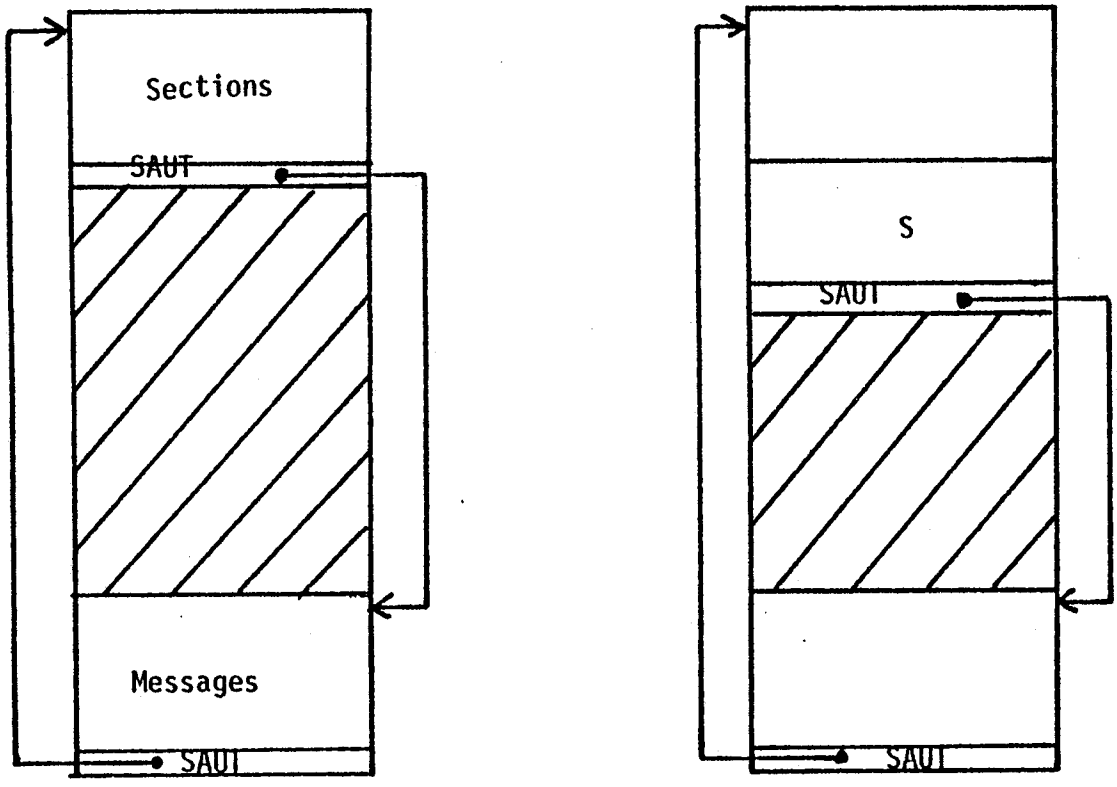


Figure V.13. : Affichage d'une section

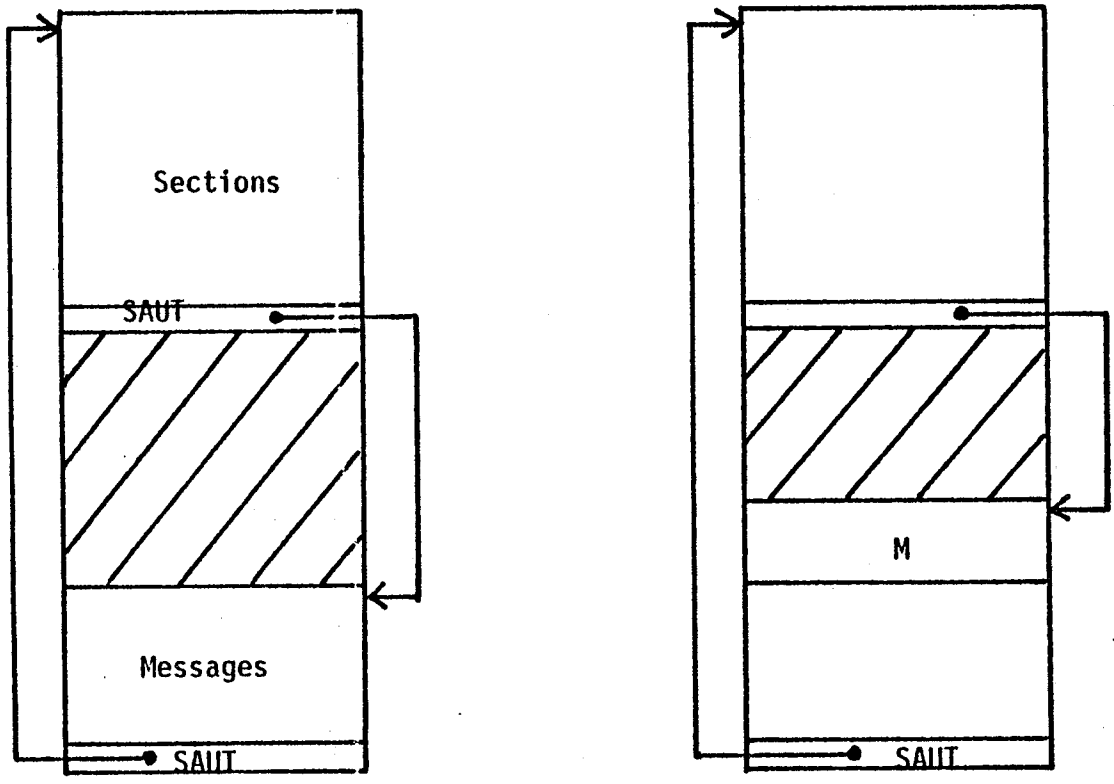


Figure V.14. : Affichage d'un message

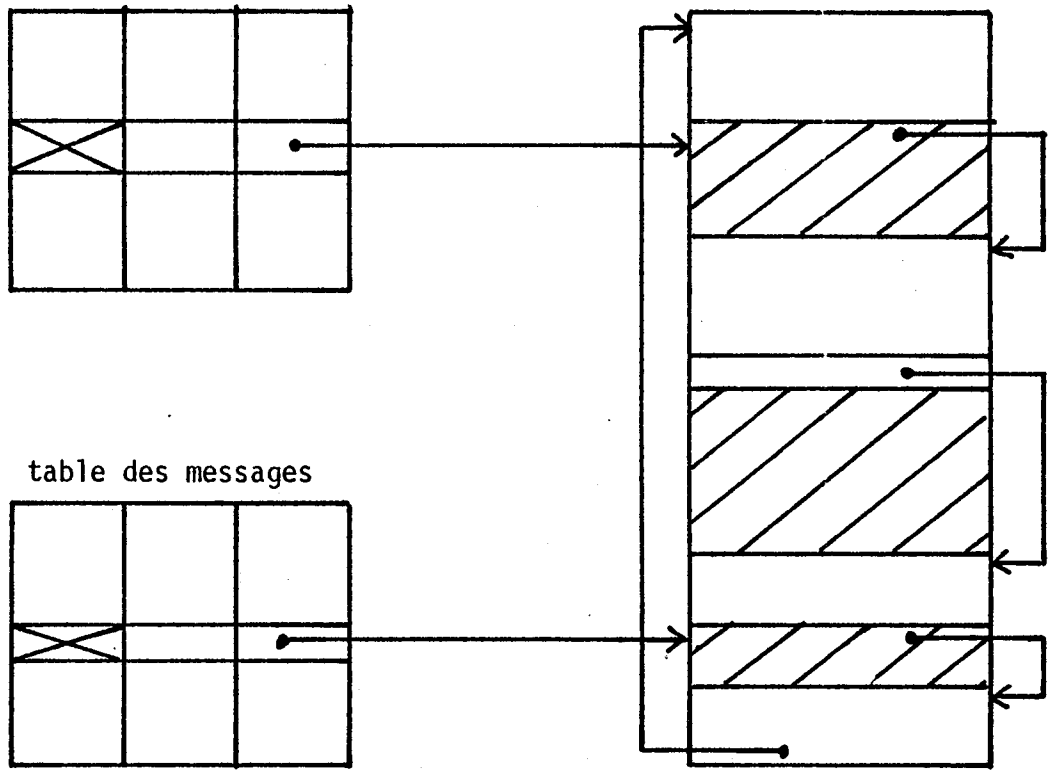


Figure V.15. : Effacement d'une section et d'un message quelconques

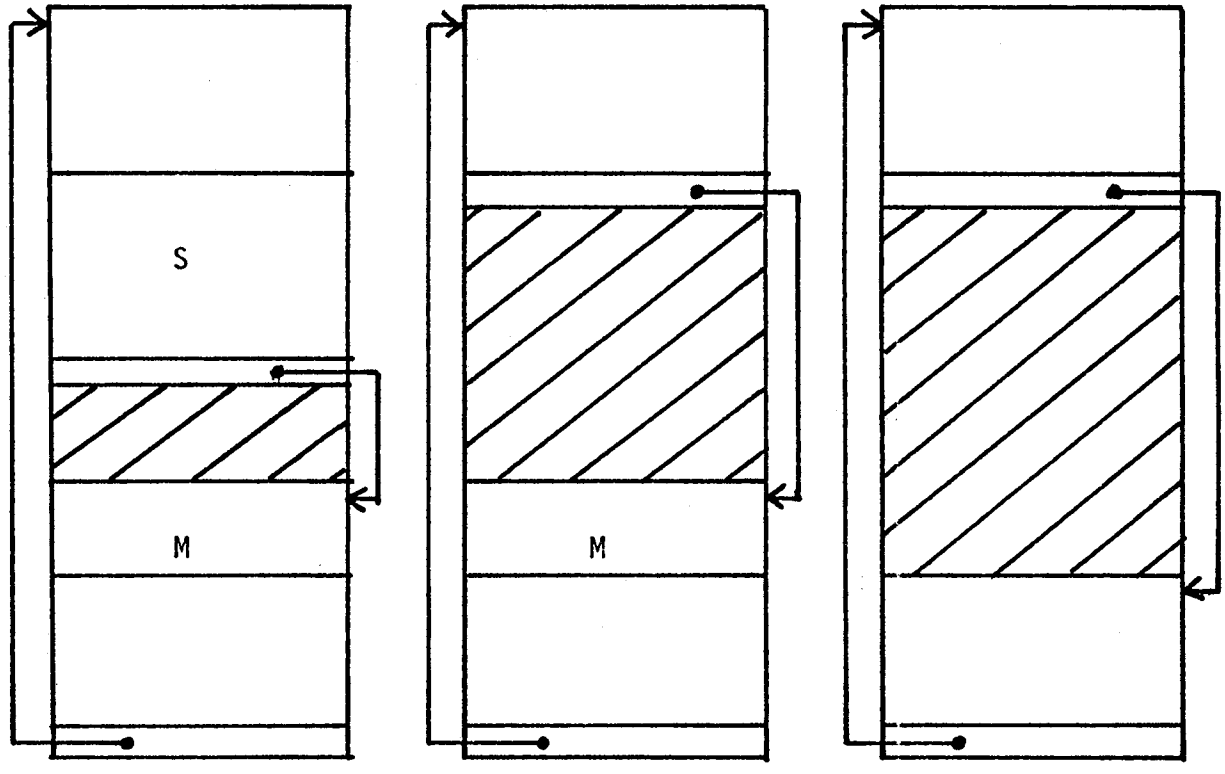


Figure V.16. : Effacement du dernier message et de la dernière section

V.4.5.2. Effacement

Lorsque l'on efface une section ou un message quelconques, il suffit d'engendrer dans la liste d'affichage un saut par dessus la zone correspondante.

La figure V.15 montre l'état de liste d'affichage après effacement d'une section et d'un message.

En fait, avant d'effacer, le logiciel teste la liste d'affichage pour savoir si la section (le message) n'est pas le dernier élément de la liste. Si on efface la dernière section ou le dernier message, le logiciel récupère la place libérée. Cette opération, qui est peu coûteuse, permet de récupérer facilement de la place dans la liste d'affichage. La figure V.16 montre l'état de la liste d'affichage après effacement de la dernière section, puis du dernier message.

Notons à ce sujet que dans la liste d'affichage, une même section peut apparaître plusieurs fois (dans le cas où le programme d'application invoque plusieurs fois la primitive AFFICHER avec les mêmes paramètres). Cette situation est d'ailleurs relativement courante. Quand l'utilisateur désire effacer cette section, il faut évidemment effacer toutes les occurrences rencontrées dans la table des sections. Pour cela, à chaque effacement, on balaie toute la table des sections, et pour avoir plus de chance de récupérer de la place en mémoire, on balaie la table en partant du bas. Ainsi, on effacera d'abord la dernière occurrence, qui est en bas de la liste, et on récupèrera de la place. Par contre, si l'on commençait par la première occurrence, la liste d'affichage serait remplie avec des sauts, mais on ne récupèrerait pas de place.

Les figure V.17 et V.18 montrent les deux résultats.

Il est clair que cette "astuce" ne coûte rien et assure la récupération maximale de place quand c'est possible.

Le même raisonnement peut être appliqué pour les messages.

V.4.5.3. Récupération de place

Malgré tout, il peut arriver que la liste d'affichage soit pleine c'est-à-dire que la zone des sections rejoigne celle des messages. Dans ce cas, lorsqu'une demande d'affichage survient, l'algorithme se déroule en deux temps :

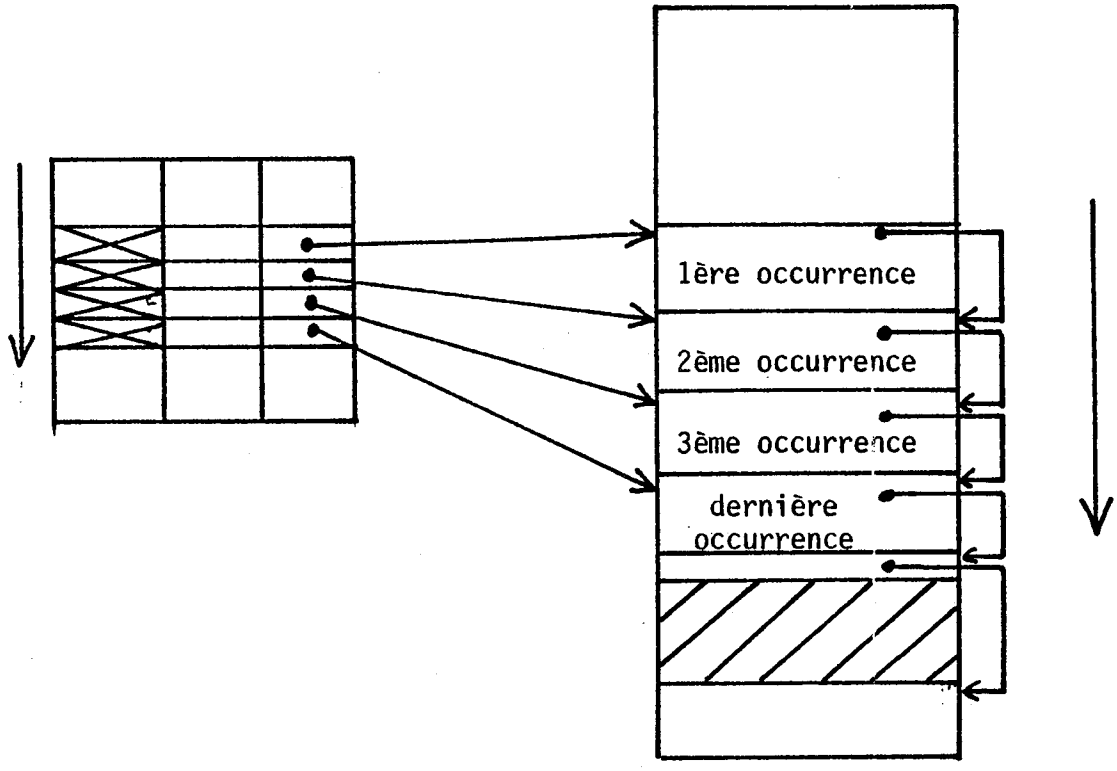
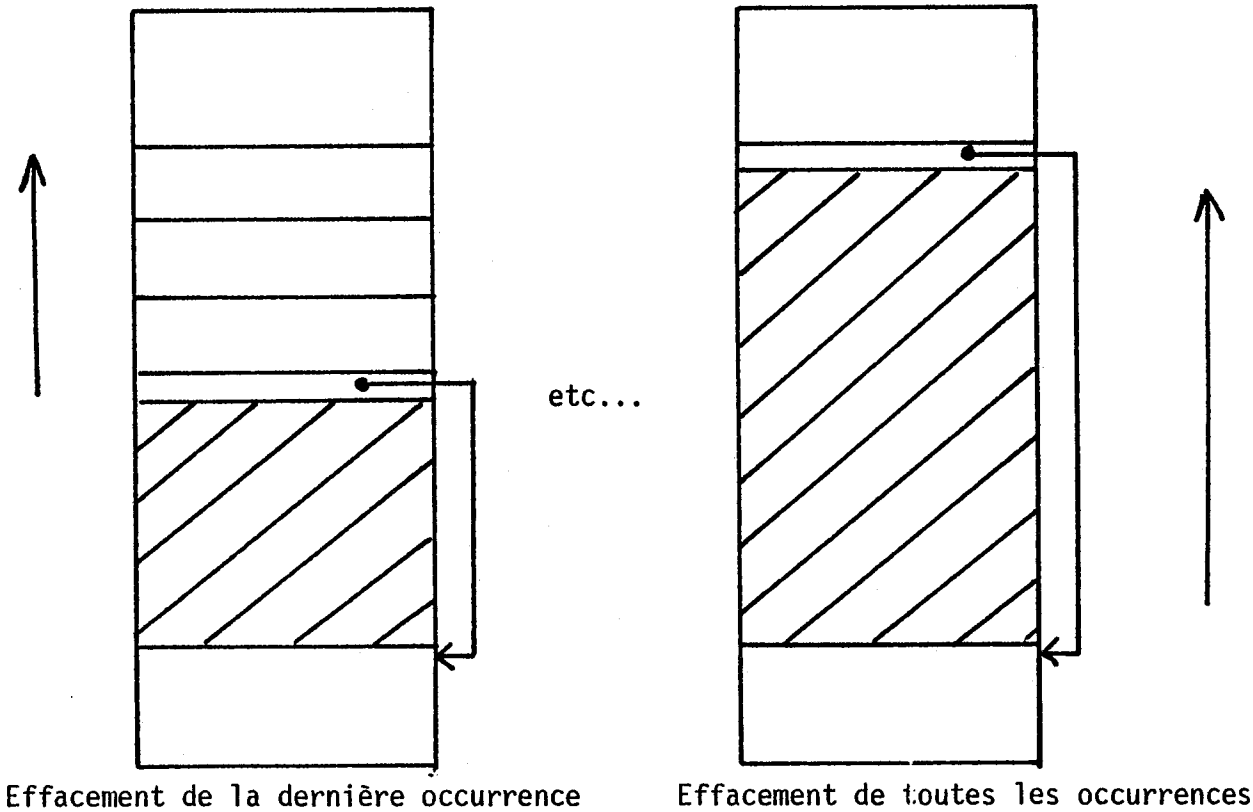


Figure V.17. : Analyse de la table des sections en partant du haut



Effacement de la dernière occurrence

Effacement de toutes les occurrences

Figure V.18. : Analyse de la table des sections en partant du bas

- recherche d'un "trou" suffisamment grand pour pouvoir insérer le nouvel élément (figure V.19).
- s'il n'existe pas de "trou" assez grand, on active alors un algorithme de réorganisation de la liste d'affichage.

Cet algorithme retasse les sections vers le haut et les messages vers le bas en éliminant tous les "trous" (figure V.20). Cette réorganisation de la liste est invoquée en dernier ressort, car elle coûte cher en temps et en place. Après réorganisation, on essaie à nouveau de stocker l'élément.

Notons que, lors de la recherche d'un trou, on ne mélange pas les deux types de données : c'est-à-dire que l'on cherche à insérer un message dans la zone des messages et une section dans la zone des sections.

Si après réorganisation complète de la liste d'affichage, la place récupérée n'est pas suffisante, on efface tout dans la liste, après avoir prévenu l'opérateur et lui avoir offert la possibilité de conserver sur traceur l'image de l'écran.

Mais dans ce cas, les opérations d'identification sur les données effacées sont évidemment interdites.

V.4.6. Les sous-programmes de service communs à tous les interpréteurs

Signalons enfin quelques programmes de service nécessaires dans tous les interpréteurs :

- un sous-programme de codage d'une valeur entière à partir d'une chaîne de caractères,
- un sous-programme de codage d'une valeur réelle à partir d'une chaîne de caractères,
- un sous-programme de traitement des marqueurs. Celui-ci à partir du symbole de marquage et de l'indice numérique engendre la chaîne de caractères à afficher en précisant le nombre de caractères qu'elle contient.

Nous constatons déjà que les traitements communs à tous les interpréteurs sont relativement nombreux, ce qui facilitera d'autant la prise en charge d'un nouveau terminal. Dans le chapitre suivant, nous préciserons les caractéristiques spécifiques aux deux types d'interpréteurs mentionnés en V.1.2.

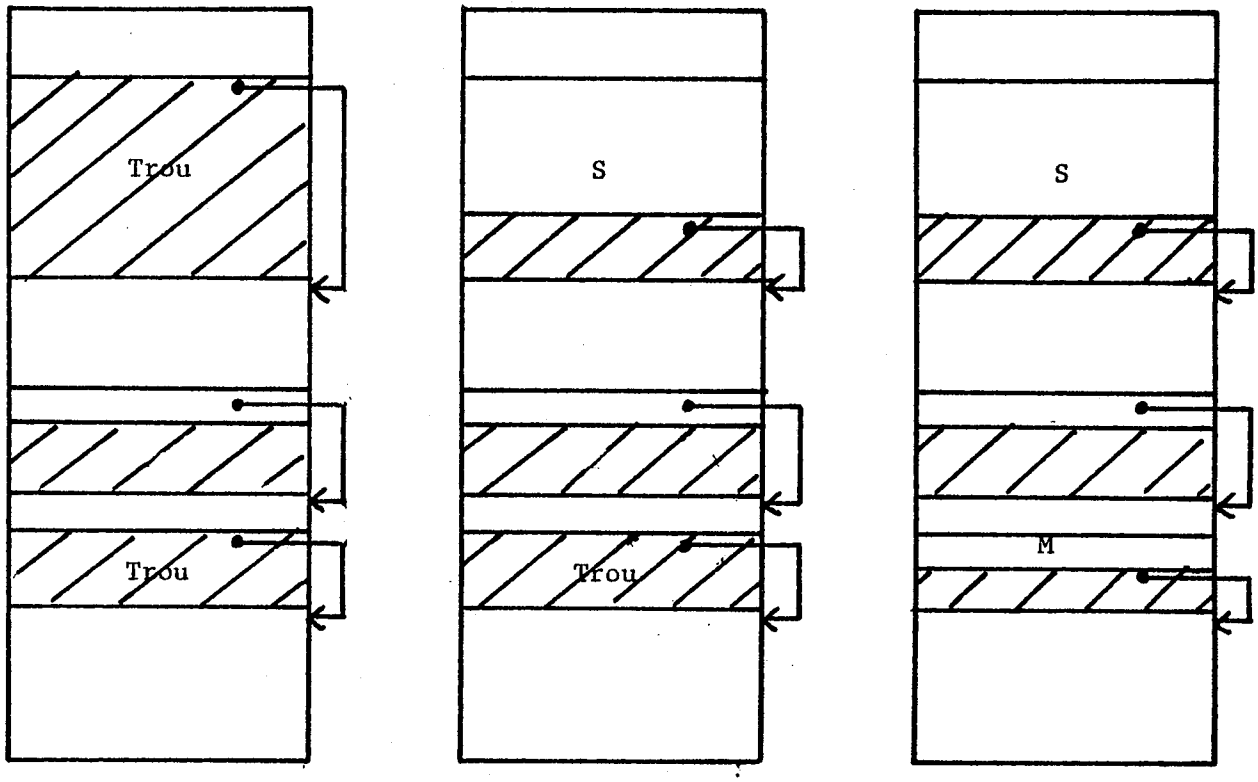


Figure V-19 : Insertion d'un message et d'une section

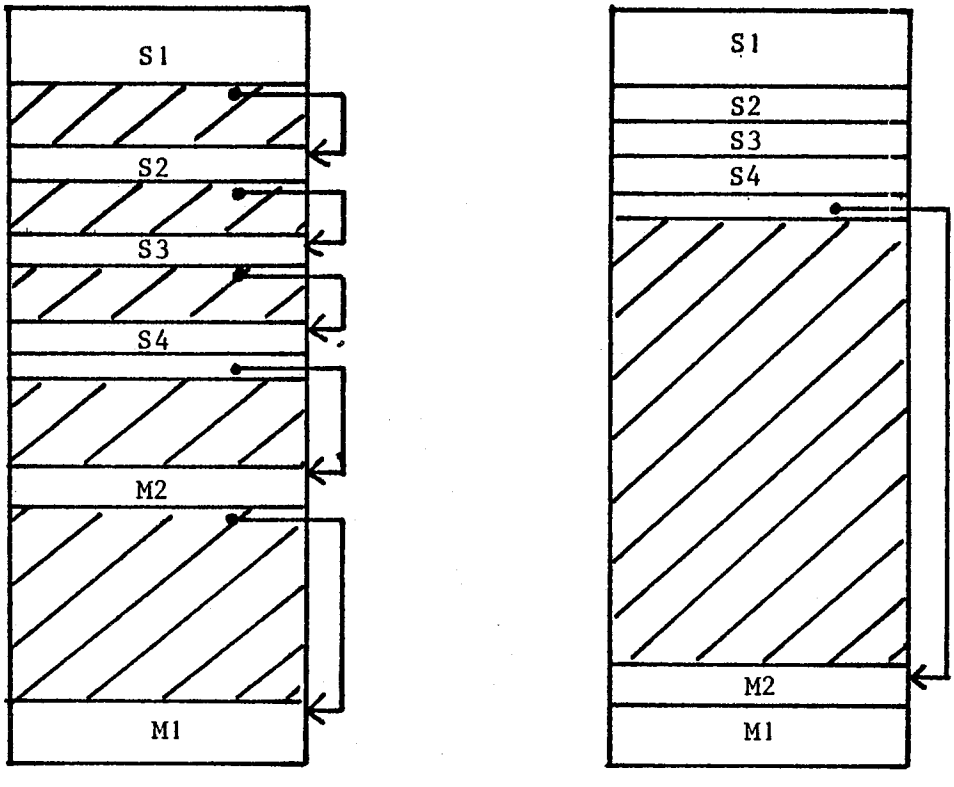


Figure V-20 : Réorganisation de la liste d'affichage

V.5. INDÉPENDANCE DU NOYAU PAR RAPPORT À SON CONTEXTE

L'indépendance d'un logiciel graphique par rapport à son contexte est évidemment fonction de l'indépendance du noyau.

- l'indépendance du noyau par rapport à l'application est fonction des primitives constituant le logiciel. Dans le cas présent, on peut admettre qu'elle est acquise
- l'indépendance du noyau par rapport à la console de visualisation est atteinte parfaitement : le code intermédiaire engendré par le noyau est indépendant du terminal ainsi que le code reçu en réponse aux primitives de dialogue.
- le noyau a été écrit en Fortran, en utilisant uniquement les options standards et les fonctions les plus répandues, on peut donc espérer que le logiciel du noyau est portable et indépendant du calculateur et du système d'exploitation.
- le noyau est constitué par une bibliothèque de sous-programmes, accessible à partir de tous les langages qui respectent les mêmes conventions de liaisons: Fortran, Algol, PL/1, etc.

On peut donc conclure que l'indépendance du noyau par rapport à son contexte est approchée au mieux.

Cependant, signalons deux classes de sous-programmes de service pour lesquels la portabilité n'est pas assurée :

- le noyau utilise trois sous-programmes qui traitent des adresses, ceux-ci sont évidemment écrits dans le langage de la machine,
- le noyau utilise des sous-programmes qui traitent les chaînes de caractères. Ceux ci sont écrits en Fortran, mais utilisent des variables logiques de longueur 1. Il n'est pas certain que ce type de variable existe dans tous les compilateurs Fortran.

Enfin, signalons que tout le traitement des chaînes de caractères est basé sur le fait qu'un mot de la machine contient quatre caractères.

Nous reviendrons en détails dans le chapitre VII sur tous ces points délicats.

CHAPITRE VI - ETUDE DES DEUX TYPES D'INTERPRETEURS

Comme nous l'avons indiqué en V.1., les interpréteurs se répartissent en deux classes selon que le matériel utilisé dispose ou non d'une mémoire d'entretien contenant une description synthétique de l'image.

Nous étudierons ici trois interpréteurs destinés à prendre en charge respectivement :

- un terminal à mémoire d'entretien,
- un terminal à tube mémoire,
- un terminal à mémoire de points.

De plus, nous aborderons le problème de la copie d'une image par un traceur de courbe

Enfin, nous évoquerons un interpréteur particulier, destiné à permettre le déroulement d'un programme sur un terminal alphanumérique.

VI.1 - INTERPRÉTEUR POUR TERMINAL À MÉMOIRE D'ENTRETIEN

Le prototype réalisé a permis d'utiliser le logiciel GRIGRI sur un terminal IBM 2250 (modèle 1) connecté à l'ordinateur IBM 360/67 du C.I.C.G.

VI.1.1. Description du contexte

VI.1.1.1. - Le calculateur et le logiciel

Le calculateur dispose de 1024 K de mémoire réelle.

Le système conversationnel CP/CMS est générateur de machines virtuelles de taille quelconque. Il n'y a donc dans ce cas aucun problème de place en mémoire.

Le calculateur dispose de mots de 32 bits (4 caractères de 8 bits chacun

La liaison calculateur-terminal est assurée par un canal.

Les langages utilisables sont nombreux : Fortran, Algol W, PL1, etc... Ils respectent tous les conventions de liaisons. Le logiciel graphique est donc constitué d'une bibliothèque de sous-programmes utilisable à partir de tous ces langages

VI.1.1.2. - Le terminal

Le terminal graphique IBM 2250 [T1] est un terminal à balayage cavalier. Il dispose d'une mémoire d'entretien de 8K-octets.

L'écran est carré, de 30 × 30 cm, et adressable sur 1024 points.

Le processeur graphique fournit les options suivantes :

- affichage de points, de segments, en absolu ou en relatif,
- affichage de caractères en mode protégé ou non protégé, avec deux tailles possibl

Notons que le processeur graphique ne dispose pas de certaines options

- texture du trait unique (contenu),
- taille et orientation fixe du texte (horizontale),
- couleur unique,
- intensité et épaisseur uniques.

Les dispositifs de dialogue sont au nombre de quatre :

- clavier alphanumérique,
- clavier de 32 touches de fonctions,
- photostyle,
- tablette.

VI.1.1.3. Le logiciel élémentaire

Il est évident qu'une partie de l'interpréteur est entièrement dé pendante du terminal et du calculateur. Ce logiciel élémentaire est chargé de gérer les entrées/sorties et les interruptions. Il est obligatoirement écrit en langage machine (dans le cas présent il est constitué de programmes canaux).

Les fonctions du logiciel élémentaire sont les mêmes que celles définie en III.6.1., pour l'utilisation de ce terminal par un logiciel graphique basé sur l concept de console virtuelle.

Cependant, pour utiliser la possibilité d'effacement sélectif, nous avc ajouté une fonction élémentaire : génération d'un saut d'une adresse à une autre da la mémoire d'entretien qui permet de réaliser des ruptures de séquences.

VI.1.2. La structure de donnée

Les différentes tables décrites V.4., sont adaptées au matériel et se réfèrent dans ce cas à la mémoire d'entretien.

VI.1.2.1. La table des sections

On y trouve les informations suivantes :

- nom interne de la section,
- numéro de corrélation associé à la section,
- longueur en octets de la zone correspondant à la section dans la mémoire d'entretien,
- adresse du début de la section dans la mémoire d'entretien.

VI.1.2.2. La table des messages

On y trouve le même type de renseignements :

- numéro du message,
- longueur en octets de la zone correspondant au message dans la mémoire d'entretien,
- adresse du début du message dans la mémoire d'entretien,
- numéro de la prochaine zone à remplir,
- nombre total de zones du message.

VI.1.2.3. La table des zones réservées

Elle contient les indications suivantes :

- numéro interne de la zone,
- longueur de la zone,
- adresse de la zone correspondante et mémoire d'entretien.

La structure de donnée de l'interpréteur est donc relativement simple. Ajoutons que l'on conserve également les adresses de début et de fin de la zone libre en mémoire, entre la zone des sections (mémoire haute) et celle des messages (mémoire basse).

VI.1.3. Les fonctions de gestion du dessin

A chaque commande du code intermédiaire engendré par le noyau, corres pond un sous-programme de traitement dans l'interpréteur. Nous allons examiner pidement ces différents traitements.

VI.1.3.1. Initialisation

En dehors de l'initialisation des tables, qui est faite dans les interpréteurs, nous devons initialiser le terminal lui-même et la mémoire d tretien, pour prévoir la gestion ultérieure. Nous exécutons donc la séquence su vante :

- remise à zéro de la mémoire,
- stockage en haut de la mémoire de l'ordre de synchronisation de la régénérati
- écriture d'un saut du haut vers le bas de la mémoire,
- écriture d'un saut du bas vers le haut de la mémoire.

La figure VI.1. montre l'état de la mémoire après initialisation.

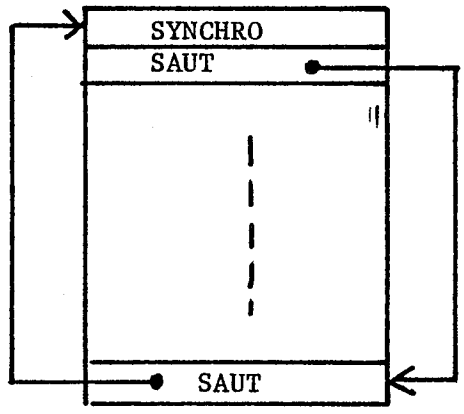


Figure VI-1 : Initialisation de la mémoire

VI.1.3.2. Récupération des attributs d'une section

Ce traitement est en fait commun à tous les interpréteurs : lorsque l'interpréteur reçoit les caractéristiques d'une section, il les stocke dans des variables qui lui sont propres, pour pouvoir les récupérer au moment du traitement de l'affichage de la section. Les bornes de la clôture réelle sont également calculées et stockées (voir V.4.3.1.).

Si la table des sections est pleine, elle est retassée si c'est possible (voir V.4.4.). L'interpréteur stocke dans cette table le nom de la section et l'adresse à laquelle elle sera rangée dans la mémoire d'entretien. La longueur est initialisée à zéro.

VI.1.3.3. Affichage d'une section de points

On commence par calculer les coordonnées des points dans le repère de l'écran, (ici valeurs entières comprises entre 0 et 1023). Ensuite, on s'intéresse au mode de la section. Les paramètres couleur, épaisseur, intensité ne sont pas pris en considération.

Pour afficher des points, deux commandes élémentaires sont nécessaires :
- passage en mode points,
- envoie des coordonnées.

Pour afficher une ligne brisée en continu, on envoie trois commandes :
- mise en place du faisceau éteint,
- passage en mode segments,
- envoi des coordonnées.

Pour afficher des segments disjoints, on affiche chaque segment séparément :
- mise en place du faisceau éteint à l'origine du segment,
- passage en mode segments,
- envoi des coordonnées de l'extrémité du segment.

Si la texture de la ligne brisée ou des segments disjoints est tiretée, pointillée, ou mixte, on utilise un générateur de tireté logiciel, qui engendre, pour chaque segment, la suite des segments élémentaires constituant le tracé. Toutes les coordonnées des points intermédiaires sont envoyées dans la mémoire d'entretien. On constate que l'utilisation des tiretés ou pointillés sur un tel terminal risque très vite de saturer la mémoire d'entretien (il faut 12 octets pour afficher un segment, aussi petit soit-il !).

Enfin, si un marquage est demandé, la chaîne de caractères à affiche: (symbole et indice) est engendrée d'une manière commune pour tous les interpré teurs, puis elle est affichée par envoi à la mémoire d'entretien des commandes suivantes :

- mise en place du faisceau éteint,
- passage en mode petits caractères,
- envoi des caractères.

Là encore, la mémoire risque d'être saturée rapidement. (10 octets pour affiche un symbole simple). La longueur de la section est mise à jour dans la table.

VI.1.3.4. Affichage d'une étiquette

L'étiquette, associée (facultativement) à une section, est t tée comme un marqueur particulier :

- calcul des coordonnées écran du premier point de la section,
- mise en place du faisceau éteint,
- passage en mode caractères,
- envoi des caractères

(pour une étiquette de 4 caractères, on utilise 12 octets en mémoire).

VI.1.3.5. Affichage d'une section de texte

Comme pour le tireté ou le pointillé, l'affichage de texte (taille et d'orientation quelconque est pris en charge par un générateur logic: Ce générateur engendre pour chaque caractère une suite de segments disjoints (consécutifs.

Le risque de saturer la mémoire d'entretien est très grand (il faut moyenne 30 octets pour afficher un caractère !).

VI.1.3.6. Effacement d'une section

La gestion de la liste d'affichage, commune à tous les inte teurs est décrite en V.4.5.

L'effacement effectif est réalisé ici par écriture, dans la mémoire d'entretien d'un saut par dessus la section à effacer (ceci est facile à réal puisque l'on connaît son adresse en mémoire et sa longueur).

VI.1.3.7. Traitement d'un message

Le texte constituant le message est affiché en mode protégé (non modifiable par le clavier alphanumérique), les zones réservées, par contre, sont affichées en mode non protégé.

Les deux commandes intermédiaires :

- récupération du message à afficher,
- liste des zones réservées,

sont ici indissociables, car il faut connaître les zones pour afficher le message.

Lors de la réception de la première commande, les caractéristiques du message sont stockées dans la table, après retassement si nécessaire, mais le message n'est pas affiché.

Lors de la réception de la seconde commande, les zones sont stockées dans la table et le message est affiché parallèlement. Au départ, on met en place le faisceau éteint sur la position initiale du message. Ensuite, on exécute la séquence suivante jusqu'à épuisement des zones réservées :

- passage en mode caractères protégés,
- envoi des caractères qui précèdent la zone,
- passage en mode non protégé,
- envoi du nombre d'espaces correspondant à la zone,
- stockage dans la table des zones de l'adresse en mémoire.

A la fin, la longueur de la zone associée au message est mise à jour dans la table L'effacement d'un message est réalisé, comme pour les sections, par écriture d'un saut par dessus la zone qui lui est associée dans la mémoire d'entretien.

VI.1.3.8. Edition alphanumérique

L'interpréteur contrôle l'existence du message et de la zone concernés.

L'édition elle-même est très simple : la chaîne de caractères est stockée dans la mémoire d'entretien, dans la zone correspondante.

Nous avons quelque peu détaillé les diverses fonctions, pour pouvoir comparer avec les fonctions équivalentes sur d'autres terminaux.

VI.1.4. Les fonctions de dialogue

Les dispositifs disponibles nous permettent une mise en oeuvre aisée du dialogue.

Notons tout d'abord une caractéristique commune à toutes les fonctions de dialogue : le contrôle du déroulement de la fonction est fait par l'intermédiaire du clavier de fonction.

En effet 4 touches sont inaccessibles aux utilisateurs. Ces touches sont utilisées respectivement pour :

- annuler la dernière opération effectuée,
- annuler toutes les opérations effectuées,
- valider l'ensemble et indiquer la fin de fonction.

VI.1.4.1. Introduction de valeurs

Elle est réalisée par l'intermédiaire du clavier alphanumérique. L'interpréteur contrôle la validité du numéro de message. Il fait ensuite apparaître le curseur sur l'écran au début de la zone (courante dans le message (cf. V.4.2.2.); ceci se fait par insertion du curseur à l'adresse associée à la zone dans la mémoire d'entretien. La récupération de la chaîne entrée se fait par lecture, dans la mémoire d'entretien, de la zone correspondante.

La figure VI.2. montre le schéma d'ensembledes opérations.

Après chaque entrée, l'interpréteur assure le codage de la valeur entière ou réelle à partir de la chaîne de caractères dont la syntaxe a été préalablement contrôlée.

Lors d'une entrée de chaîne alphanumérique, l'interpréteur associe, à chaque chaîne introduite, la longueur de la zone pour que le noyau puisse connaître le nombre de caractères utiles.

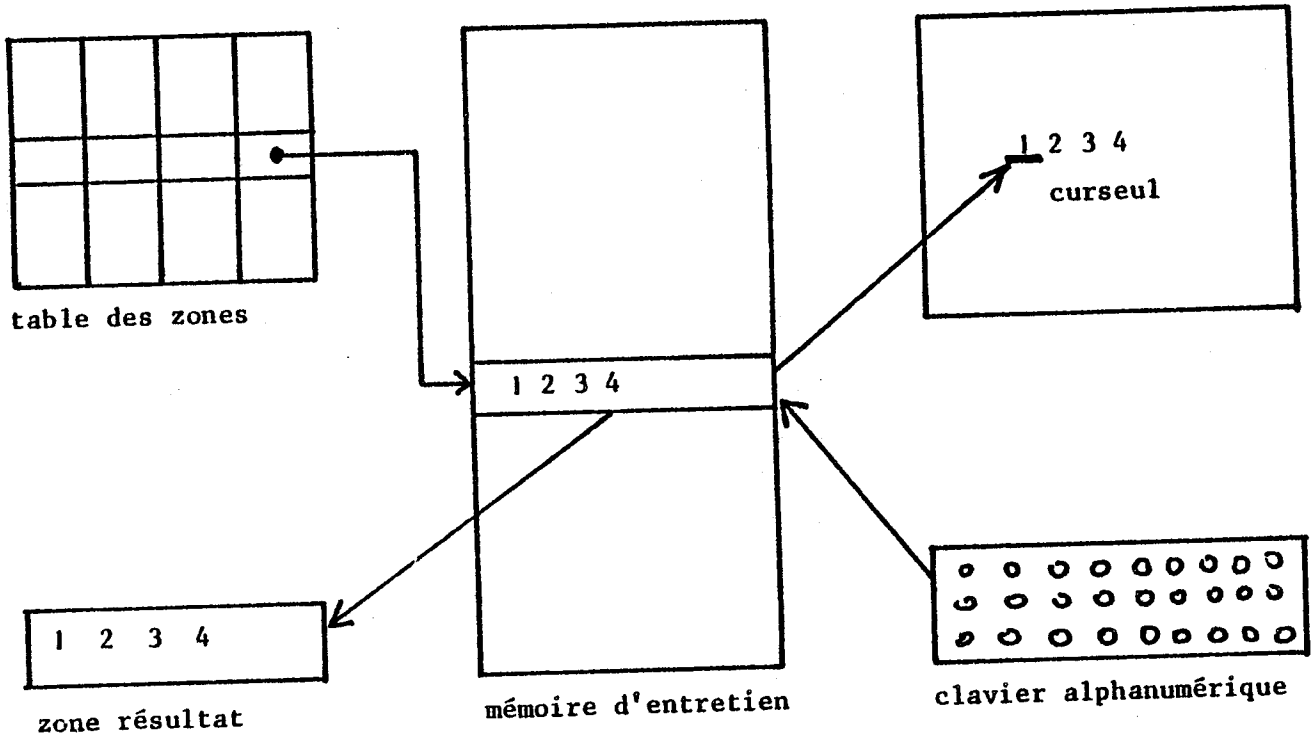


Figure VI-2 : Récupération d'entrées clavier sur une console à mémoire d'entretien

VI.1.4.2. Identification

Elle est réalisée grâce au photostyle.

Lorsqu'il détecte de la lumière, celui-ci engendre une interruption qui fournit entre autres renseignements l'adresse de l'instruction interrompue dans la mémoire d'entretien. Or cette instruction est précisément celle qui est exécutée pour afficher l'élément "vu" par le photostyle. Dès lors, l'identification est très simple : connaissant l'adresse en mémoire d'entretien, il suffit de balayer la table des sections, dont on connaît l'adresse de début et la longueur, pour savoir à quelle section appartient l'élément désigné. La table des sections nous founit alo la corrélation associée. Nous constatons ici une meilleure utilisation du matériel que celle décrite dans le chapitre III pour la même fonction.

VI.1.4.3. Collecte de coordonnées

La collecte de coordonnées utilise 3 dispositifs :

- le clavier alphanumérique,
- le photostyle,
- la tablette.

Le clavier de fonctions est utilisé pour les trois fonctions décrites plus haut, et également pour indiquer la fin d'une section et le choix du dispositif désiré

A chaque instant, l'opérateur peut utiliser le clavier pour rentrer un couple de coordonnées, ou appuyer sur une touche pour choisir un dispositif.

Lorsqu'il demande le photostyle, l'écran est balayé par un rideau de lettres, pour pouvoir montrer n'importe quel point sur l'écran. Ceci se fait san intervention du calculateur : c'est la mémoire d'entretien qui "boucle" tant que l'interruption du photostyle n'est pas récupérée.

Lorsqu'il utilise la tablette, l'opérateur dispose d'une touche pour i diquer la fin d'utilisation.

Les points introduits sont affichés sur l'écran, ainsi que les coordon nées du dernier point rentré.

Lorsque l'opérateur annule tout ou partie de son relevé, les points co respondants sont effacés de l'écran et les coordonnées courantes sont mises à jo

Les bornes de la fenêtre sont visualisées sur l'écran, afin de permett à l'opérateur d'introduire des points corrects. A la fin de la fonction, tout le support système (fenêtre, points collectés, coordonnées) est effacé de l'écran.

VI.1.4.4. Menu

Le menu utilise deux dispositifs.

A chaque numéro d'action, on associe une touche du clavier de fonctions. De plus, les noms sont affichés sur l'écran et peuvent être désignés au moyen du photosty En retour, l'interpréteur renvoie soit le numéro de la touche enfoncée, soit l'o donnée du nom désigné (évaluée entre 0 et 100).

VI.1.5. Conclusion

L'interpréteur décrit ici contient des traitements communs à tous les interpréteurs et des traitements spécifiques au type de matériel.

Il est clair que cet interpréteur peut prendre en charge tous les terminaux à mémoire d'entretien qui ont les mêmes caractéristiques. Il suffit pour cela de disposer d'un logiciel élémentaire remplissant les mêmes fonctions. Notons que, sur ce type de matériel, le dialogue est facilement mis en oeuvre. Par contre, la génération par logiciel de tireté, pointillé, ou de caractères risque de très vite saturer la mémoire d'entretien.

VI.2. INTERPRÉTEUR POUR TERMINAL À TUBE MÉMOIRE

Ce type de matériel ne dispose pas d'une mémoire d'entretien. Il faudra donc que l'interpréteur simule et gère une liste d'affichage contenant une description structurée de l'image. C'est le second type d'interpréteur (voir V.1).

VI.2.1. Organisation générale de l'interpréteur

Pour ce type de matériel, l'interpréteur se rapproche un peu d'un logiciel basé sur le concept de console virtuelle, mais ici, la liste de visualisation peut être adaptée au matériel.

Toutes les commandes graphiques sont exécutées en deux temps :

- gestion de la liste d'affichage,
- appel à un "processeur graphique" logiciel pour exécuter la commande précédemment enregistrée dans la liste, et la répercuter sur l'écran.

C'est la principale différence avec l'interpréteur pour terminal à mémoire d'entretien dans lequel il suffit de stocker les données dans la liste d'affichage, pour que l'image apparaisse sur l'écran. La figure VI.3 montre l'organisation de l'interpréteur.

Il est clair qu'une même liste d'affichage pourra être utilisée par tous les terminaux à tube mémoire.

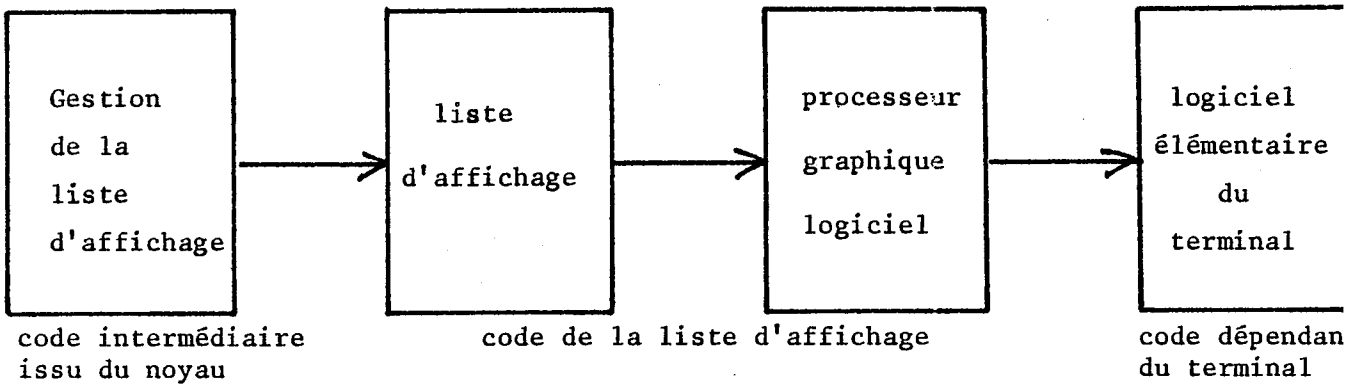


Figure VI-3 : Interpréteur pour terminal à tube mémoire

Ainsi, le module de communication avec le noyau et de gestion de la liste d'affi chage, ainsi que la liste elle-même peuvent être définis une fois pour toutes. C est le processeur graphique logiciel qui traduira les commandes de la liste d'af fichage en code dépendant du terminal.

C'est pourquoi nous allons décrire très précisément le contenu et l'or ganisation de la liste d'affichage.

VI.2.2. Description de la liste d'affichage

Les commandes internes de la liste d'affichage ont été choisies en fon tion des primitives du logiciel.

VI.2.2.1. Les commandes graphiques

Les commandes retenues pour constituer la liste d'affichage sc les suivantes :

- mise en place du faisceau éteint,
- affichage de points,
- affichage d'une ligne brisée,
- affichage de segments disjoints,
- marquage d'une suite de positions,
- affichage d'une suite de caractères en mode standard,
- affichage d'une suite de caractères en mode quelconque,
- saut.

La tableau de la figure VI.4 donne le format de chacune des commandes et de ses opérandes dans la liste d'affichage.

VI.2.2.2. Choix relatifs à la liste d'affichage

Cet interpréteur est destiné à prendre en charge des terminaux Tektronix dont les écrans ont tous la même forme :

Les points affichables ont des coordonnées telles que:

$$\begin{cases} 0 \leq X \leq 1023 \\ 0 \leq Y \leq 780 \end{cases}$$

Les coordonnées stockées dans la liste d'affichage sont donc des entiers qui respectent ces conventions. Ainsi, le processeur graphique n'aura pas à recalculer les coordonnées pour les passer à l'écran utilisé.
Dans le prototype réalisé, la liste d'affichage est rangée dans un tableau de 4096 entiers de longueur 2. C'est pourquoi nous stockons deux caractères par entier.
Le codage des différents opérandes est le suivant :

- le symbole de marquage est cadré à droite dans l'entier.
- l'indice initial est un entier s'il existe, sinon la case correspondante est remplie avec une valeur "bidon".

- la texture peut prendre 5 valeurs :

  1 - continu,
  2 - tireté court,
  3 - tireté long,
  4 - pointillé,
  5 - mixte.

- la taille des caractères peut prendre 4 valeurs (de 1 à 4).

- l'orientation du texte est spécifiée par la valeur (en entier) de l'angle avec l'horizontale.

Commande	Codop	Opérandes					
Mise en place du faisceau éteint	- 1	X	Y				
Affichage de points	- 2	nombre de points	X	Y	---	---	---
Marquage de positions	- 3	symbole de marquage	Indice initial	nombre de positions	X	Y	---
Affichage d'une ligne brisée	- 4	texture	nombre de segments	X	Y	---	---
Affichage de segments disjoints	- 5	texture	nombre de segments	X1	Y1	X2	Y2
Affichage de caractères en mode standard	- 6	nombre de caractères	C1 | C2	---	---	---	---
Affichage de caractères en mode quelconque	- 7	taille des caractères	orientation du texte	Nombre de caractère	abscisse de l'origine	ordonnée de l'origine	C1 | C2
Saut	- 8	Adresse					

Figure VI.4 : Les commandes graphiques

Notons qu'ici nous n'avons pas stocké les paramètres couleur, épaisseur, intensité. S'ils sont utilisables sur le matériel, ceux-ci peuvent être ajoutés au même titre que la texture en tête des commandes graphiques (ou l'on peut envisager, pour économiser de la place, un codage sur une seule variable de 4 paramètres : texture, couleur, épaisseur, intensité. Ce qui aurait pour avantage de ne pas désorganiser la liste telle qu'elle est décrite ici).

Les codes opérations se distinguent par le fait que ce sont les seules valeurs négatives.

- tous les opérandes sont des entiers positifs ou nuls.

VI.2.3 - Le processeur graphique

C'est un module qui permet d'exécuter les commandes stockées dans la liste d'affichage. En sortie, il engendre des appels au logiciel élémentaire qui, lui, envoie le code réel au terminal, mais dépend du calculateur.
Le processeur graphique, écrit en Fortran, ne dépend pas du calculateur. Il pourra être utilisé pour piloter n'importe quel terminal à tube mémoire, dans n'importe quel contexte, à condition que l'interface avec le logiciel élémentaire soit respec

VI.2.3.1 - Interface avec le logiciel élémentaire

Les commandes du logiciel élémentaire invoquées par le processeur graphiq sont les suivantes :

- passage en mode alphanumérique
- déplacement du faisceau éteint (qui assure le passage en mode graphique
- déplacement du faisceau allumé,
- écriture d'une suite de caractères (par le générateur matériel).

Cet interface est très simple et ces fonctions existent auprès de tout terminal à tube mémoire.

VI.2.3.2 - Exécution des commandes de la liste d'affichage

Pour afficher une suite de points, le processeur graphique engendre pour chaque point les deux commandes :

- déplacement du faisceau éteint,
- déplacement du faisceau allumé avec les mêmes coordonnées.

- Pour afficher des marqueurs, la séquence est la suivante :
    - génération de la chaîne de caractères à afficher,
    - déplacement du faisceau éteint,
    - passage en mode alphanumérique,
    - affichage de la chaîne de caractères.

- Pour afficher une ligne brisée, le processeur graphique invoque la fonction de déplacement du faisceau allumé vers les différents points.

- Pour afficher des segments disjoints, la séquence est la suivante pour chaque segment :
    - déplacement du faisceau éteint (origine),
    - déplacement du faisceau allumé (extrémité).

- Pour afficher des segments avec une texture différente du trait continu, le processeur graphique fait appel à un générateur de tiretés qui affiche les segments avec la texture désirée

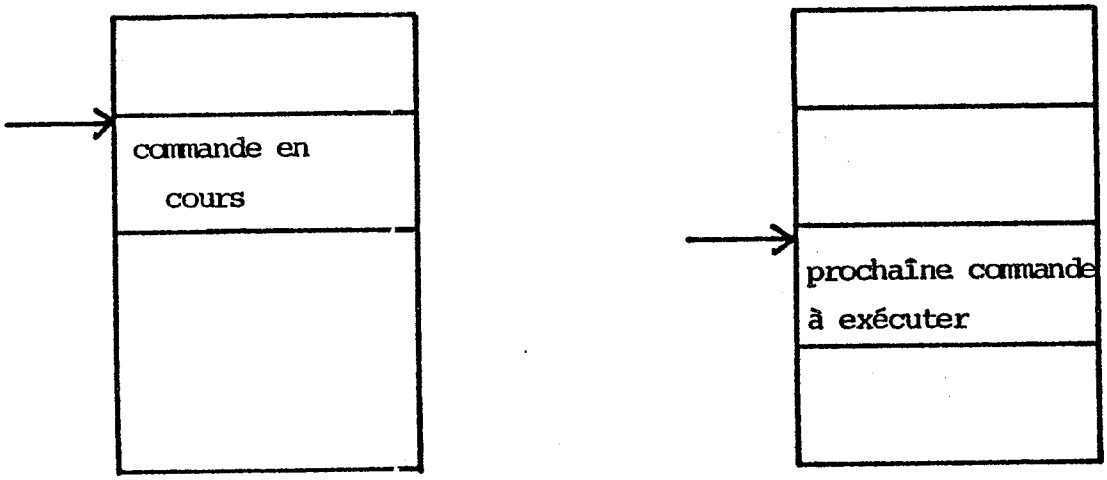
- L'affichage de texte en mode standard se fait par invocation directe du générateur de caractères fournis par le matériel.

- Par contre, pour afficher du texte de taille et orientation quelconques, le processeur graphique fait appel à un générateur de caractères logiciel [R2] qui affiche les caractères sous forme d'une suite de segments.

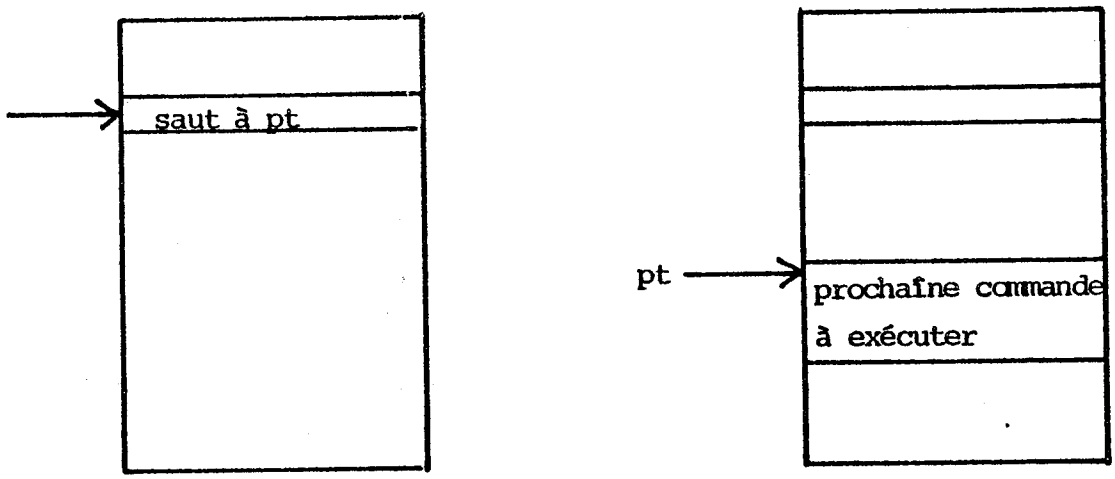
- Enfin, l'exécution d'un saut est réalisée par le processeur graphique en mettant à jour l'adresse de la prochaine commande à exécuter. En fait, le processeur graphique est invoqué avec deux pointeurs : un pointeur sur la commande à exécuter et un pointeur qui sera mis à jour par le processeur graphique lui-même et qui indiquera la prochaine commande à exécuter.

Ainsi, si la commande en cours est une commande graphique, le processeur renverra comme prochaine commande à exécuter, la commande suivante dans la liste d'affichage ; par contre si la commande en cours est un saut, le processeur graphique indiquera comme prochaine commande celle qui se trouve à l'adresse précisée par le saut.

La figure VI.5 montre les deux résultats.



(a) exécution d'une commande graphique



(b) exécution d'un saut

Figure VI-5 : Progression du pointeur courant

Notons que, contrairement à ce qui se passe dans le cas d'un terminal à mémoire d'entretien, les générations de marqueurs, de segments en tireté ou pointillé, et de caractères de taille et orientation variables, ne demandent aucune place supplémentaire dans la liste d'affichage. Ceci provient du fait que, la liste d'affichage étant dans le logiciel, il est plus intéressant de placer les générateurs spéciaux au délà de cette liste (figure VI.6).

Par contre, dans le cas des terminaux à mémoire d'entretien, on est obligé de placer les générateurs en amont de la mémoire et de stocker ce qui est engendré (figure VI.7).

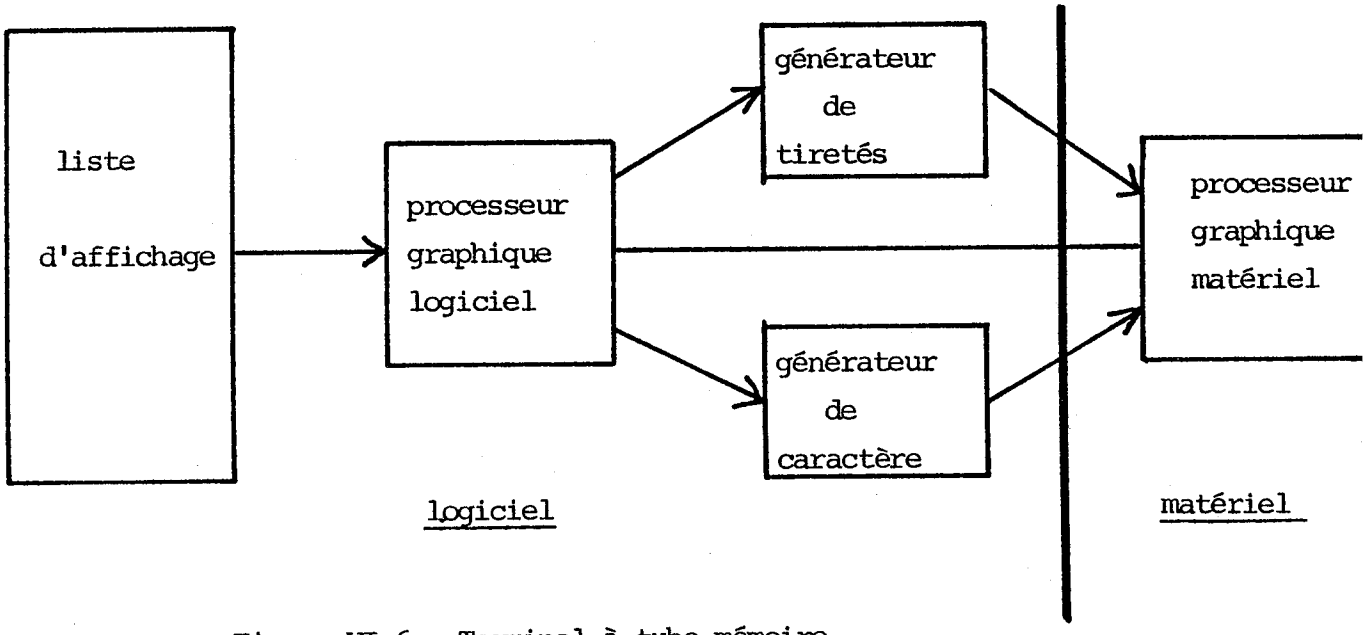
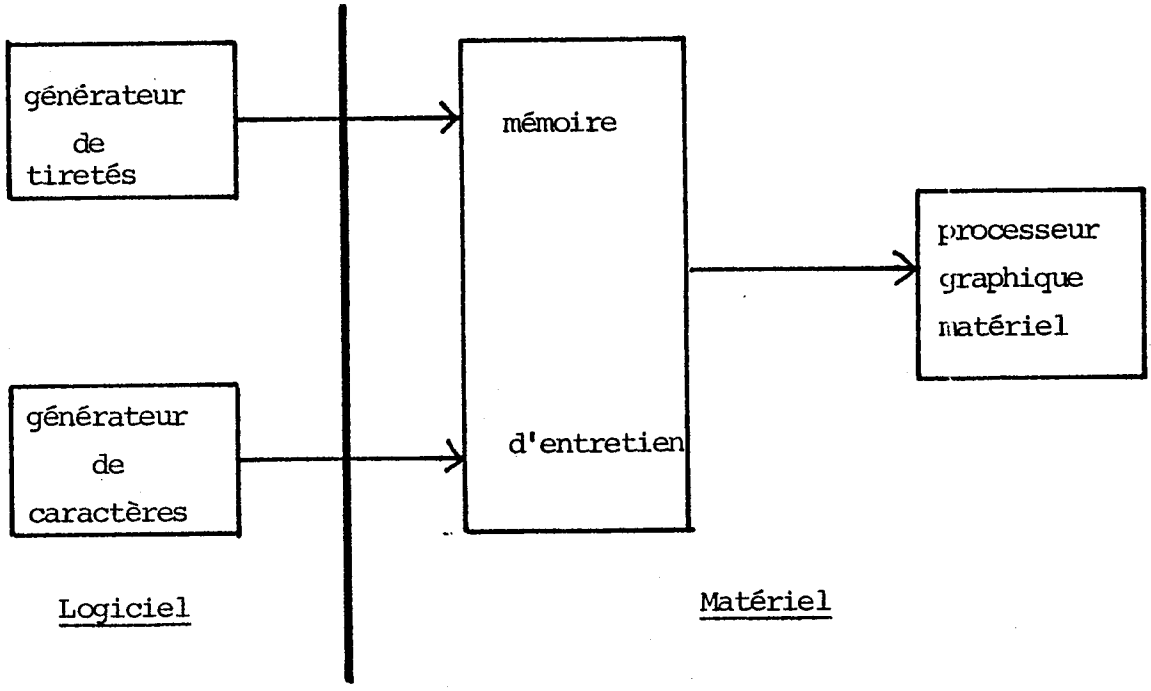


Figure VI-6 : Terminal à tube mémoire



Cette organisation permet d'utiliser la caractéristique principale des terminaux à tube mémoire, qui est de pouvoir afficher beaucoup d'informations sur l'écran.

VI.2.4. La structure de donnée

En dehors de la liste d'affichage elle-même, l'interpréteur renferme des données spécifiques aux terminaux à tube mémoire.

VI.2.4.1. La table des sections affichées

Elle contient les mêmes renseignements que pour un interpréteur pour terminal à mémoire d'entretien, maix ceux-ci font référence à la liste d'affichage simulée par la logiciel. On trouve respectivement :
- nom interne de la section,
- le numéro de corrélation associé à la section,
- index du début de la zone associée à la section dans la liste d'affichage,
- la longueur de la section dans la liste d'affichage.

VI.2.4.2. La table des messages

Elle contient les mêmes renseignements que pour les sections, pour la gestion de la liste d'affichage :
- numéro du message,
- index du début de la zone associée au message dans la liste d'affichage,
- longueur du message dans la liste d'affichage.

On y trouve également les paramètres nécessaires à la gestion des introductions de valeurs sur le message :
- numéro de la zone courante dans le message,
- nombre total de zones dans le message.

Enfin, on trouve deux paramètres supplémentaires nécessaires à la gestic de l'écran. En effet, pour placer correctement le curseur dans une zone réservée sur l'écran, il faut connaître la position initiale du message. On ajoute donc à l

table des messages les informations suivantes :

- abscisse du début du message, dans le repère écran,
- ordonnée du début du message, dans le repère écran.

VI.2.4.3. La table des zones réservées

On y trouve les mêmes renseignements que pour un terminal à mémoire d'entretien :

- numéro interne de la zone,
- longueur de la zone.

De plus, pour placer le curseur correctement sur l'écran, à partir de la positi initiale du message, on stocke pour chaque zone le nombre de caractères précéda la zone dans le message.

Notons que les données supplémentaires stockées sont utilisées pour g rer l'affichage proprement, affichage qui n'est pas lié à la gestion de la list comme dans le cas d'une mémoire d'entretien.

VI.2.5. Gestion de la liste d'affichage

La gestion de la liste d'affichage, ainsi que celle des tables qui lu sont associées est en tous points semblable à celle décrite pour un terminal à moire d'entretien. Seules diffèrent les commandes qui sont stockées dans cette liste.

Notons cependant une différence importante : on n'engendre pas de sau du bas vers le haut de la liste, car on n'a pas besoin de boucler sur la liste affichage pour assurer l'entretien de l'image sur l'écran.

Nous allons maintenant examiner les différentes commandes qui sont st kées dans la liste d'affichage pour chaque commande du code intermédiaire reçu par l'interpréteur.

VI.2.5.1. Initialisation

L'initialisation de la liste d'affichage est très simple : on engendre un saut du haut de la mémoire vers le bas.

La figure VI.8 montre la liste d'affichage initialisée.

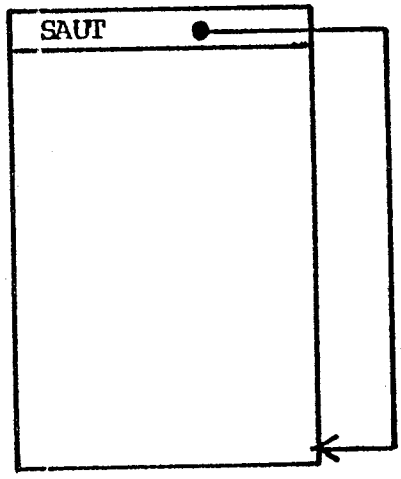


Figure VI-8 : Initialisation

VI.2.5.2. Affichage d'une section de points

La récupération des attributs d'une section se fait comme pour un terminal à mémoire d'entretien (voir VI.1.3.2.).

Les commandes stockées dans la liste d'affichage sont différentes suivant le mode de représentation de la section.

Tout d'abord, les coordonnées reçues (absolues) sont transformées en coordonnées dans le repère de l'écran. Ensuite, on engendre les commandes nécessaires dans la liste d'affichage.

- Si la section est représentée par des points, on stocke une seule commande dans la liste : affichage de points, en rangeant correctement leur nombre et leurs coordonnées.
- Si la section est représentée par une ligne brisée, on stocke deux commandes dans la liste :
  - mise en place du faisceau éteint sur le premier point,
  - affichage d'une ligne brisée.
- Si la section est représentée par une suite de segments disjoints on stocke la commande d'affichage de segments disjoints dans la liste.

Dans ces deux cas, la texture correspondante est enregistrée dans la liste.

- Enfin, si un marquage est demandé, on stocke dans la liste la commande de marquage, avec le symbole utilisé et l'indice initial. Si une étiquette est attachée à la section, celle-ci est enregistrée dans la liste par deux commandes :
  - mise en place du faisceau éteint,
  - affichage de texte.

VI.2.5.3. Affichage d'une section de texte

Celui-ci se fait par stockage dans la liste de la commande d'affichage de texte en mode quelconque. Les paramètres taille, orientation, position initiale sont stockés dans la liste.

VI.2.5.4. Effacement d'une section

L'effacement d'une section dans la liste d'affichage est réalisé par écriture dans celle-ci d'un saut par dessus la zone correspondante. Notons que contrairement au cas d'une mémoire d'entretien, ceci n'a aucune conséquence sur l'image affichée sur l'écran.

VI.2.5.5. Traitement d'un message

- Les deux commandes intermédiaires sont ici traitées indépendamment.

Dans un premier temps, le message à afficher est stocké dans la liste d'affichage, par deux commandes :
  - mise en place du faisceau éteint (sur la position initiale),
  - affichage de texte (les messages sont affichés par le générateur de caractères fourni par le matériel).

Dans un second temps, les caractéristiques des zones réservées sont stockées dans la table. L'adresse du début du message dans la liste est conservée pour chaque zone.

- L'effacement d'un message dans la liste est réalisé par écriture d'un saut, comme pour les sections.

VI.2.5.6. Edition alphanumérique

L'édition d'une donnée alphanumérique est répercutée au niveau de la liste d'affichage par stockage dans la zone correspondante de la chaîne de caractères à afficher. On trouve cette zone grâce à :

- l'adresse de début du message,
- le nombre de caractères qui précèdent la zone dans le message.

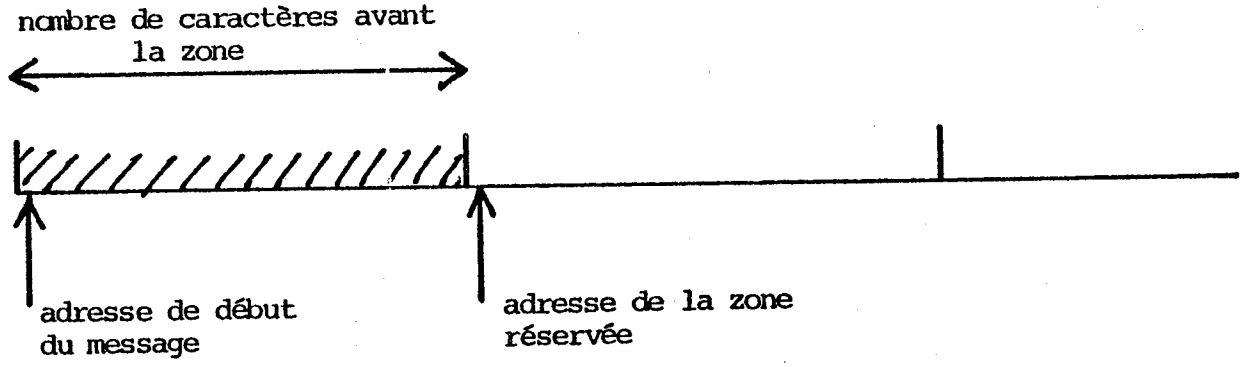


Figure VI-9 : Recherche d'une zone réservée dans la liste d'affichage

Notons que le stockage de la donnée dans la liste d'affichage se fait sans engendrer de nouvelle commande, car la zone est partie intégrante du texte du message.

VI.2.6. Gestion de l'écran

VI.2.6.1. L'affichage

Pour répercuter sur l'écran les opérations effectuées, chaque commande enregistrée dans la liste d'affichage doit être exécutée par le processeur graphique logiciel.

Lorsque l'on prend en charge une demande d'affichage, on engendre les commandes nécessaires dans la liste de visualisation, puis on invoque le processeur graphique en lui précisant l'adresse de la commande à exécuter ; ainsi, l'affichage stocké dans la liste est effectivement réalisé sur l'écran.

VI.2.6.2. L'effacement

Les effacements enregistrés dans la liste d'affichage n'ont aucun effet immédiat sur l'écran. L'effacement effectif se fait par effacement total de l'écran et exécution par le processeur graphique de toutes les commandes contenues dans la liste d'affichage (en exécutant les sauts, le processeur graphique n'exécutera pas les commandes des éléments effacés). Mais cette regénération

de l'image coûte cher, car il faut recalculer toutes les coordonnées intermédiaires pour les tiretés, pointillés, pour les marqueurs et pour les caractères tracés par logiciel. De plus, si la vitesse de transmission des informations est faible, cette reconstruction risque de durer longtemps si l'image est complexe. C'est pourquoi nous ne mettons pas l'écran à jour systématiquement après chaque effacement. Nous avons décidé de reconstruire l'image lorsque l'application demande un nouvel affichage, ou une fonction de dialogue.

Pour cela, nous disposons de trois indicateurs :

- le premier indique si un effacement a été enregistré,
- le second indique si l'utilisateur a demandé un effacement de toutes les sections (correspondant à EFFACER(0,0)),
- le troisième indique si l'utilisateur a demandé un effacement de tous les messages (EFFMESSAGE(0)).

Lorsqu'un effacement est demandé, après l'avoir répercuté sur la liste d'affichage, l'interpréteur teste si les deux dernières conditions sont remplies (effacement complet), auquel cas, l'écran est effectivement nettoyé; sinon l'effacement effectif est différé.

Avant chaque affichage et chaque demande de dialogue, le logiciel teste si un effacement est enregistré, si oui, l'image est mise à jour sur l'cran. Le seul cas où l'effacement est exécuté immédiatement est donc celui o l'utilisateur a demandé successivement l'effacement de toutes les sections et l'effacement de tous les messages.

VI.2.7. Les fonctions de dialogue

Sur ce type de terminaux, nous dispons de deux dispositifs :

- le clavier alphanumérique,
- le réticule ou son équivalent.

Le contrôle du déroulement de la fonction est fait par l'intermédia re du clavier alphanumérique : nous avons privilégié trois lettres (E,F,S) qu lorsqu'elles sont frappées seules, sont interpréter comme des requêtes de l'o pérateur pour respectivement :

- annuler la dernière opération (E),
- annuler toutes les opérations effectuées (F),
- valider l'ensemble et indiquer la fin de fonction (S).

VI.2.7.1. Introduction de valeurs

Elle est réalisée par l'intermédiaire du clavier alphanumérique. La mise en place du curseur au début de la zone sur l'écran se fait en utilisant les informations suivantes (voir figure VI.10) :

- coordonnées écran du début du message,
- taille des caractères engendrés par le matériel,
- nombre de caractères qui précèdent la zone dans le message.

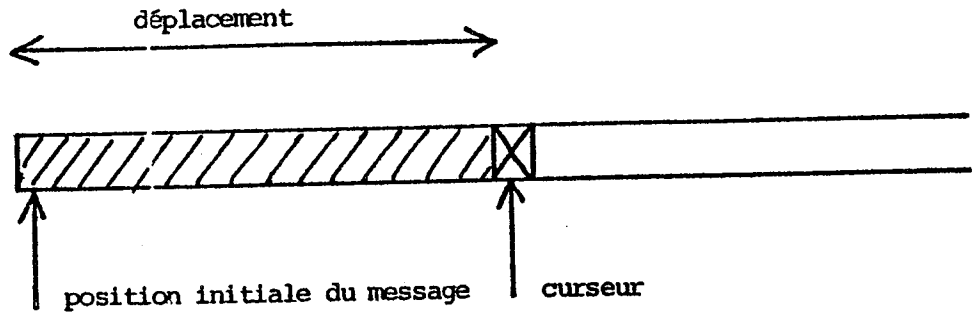


Figure VI-10 : Mise en place du curseur

Les informations introduites viendront évidemment se superposer à celles affichées déjà dans la zone.

La chaîne de caractères récupérée sera rangée dans la zone correspondante de la liste d'affichage. Ainsi, les informations à afficher seront mises à jour lors de la prochaine régénération de l'image.

VI.2.7.2. Identification

Elle est réalisée grâce au réticule. Le caractère frappé pour désactiver le réticule est utilisé par l'opérateur pour indiquer s'il désire exécuter une des trois fonctions associées aux lettres E, F ou S. Si ce n'est pas une de ces trois lettres, l'identification en cours est prise en compte.

Le réticule nous fournit la position du point désigné, dans le système de coordonnées de l'écran. Nous définissons autour de ce point une fenêtre d'erreur (en coordonnées écran). La recherche de l'élément désigné se fait en étudiant la position de tous les éléments de la liste de visualisation par rapport à cette fenêtre. Mais dans ce cas, le problème est simplifié par rapport à celui de l'identification dans une console virtuelle (voir III) car toutes les données à visualiser sont connues dans le système de coordonnées de l'écran. En particulier, à ce niveau, nous connaissons la taille des caractères engendrés par le matériel (ce qui n'est pas le cas dans un écran virtuel). Pour éviter de réactiver le générateur de caractères logiciel, on teste seulement si la fenêtre d'erreur recouvre une partie du rectangle englobant le texte. Ainsi, la fonction d'identification, si elle est aussi lourde à réaliser, ne pose plus de problème crucial de la taille des textes rencontré dans la console virtuelle.

Lorsque l'élément désigné est localisé dans la liste d'affichage, nous retombons sur le même problème que pour la mémoire d'entretien :

connaissant l'adresse de l'élément dans la liste, il suffit de balayer la table des sections pour savoir laquelle contient la commande repérée.

VI.2.7.3. Menu

Le menu utilise les deux dispositifs.

On active le réticule et le caractère renvoyé est analysé :

- si c'est une lettre de contrôle, on exécute la requête de l'opérateur,
- si c'est un chiffre, on considère que c'est le premier chiffre d'un numéro dans le menu. L'opérateur peut alors frapper le second chiffre au clavier.
- Si c'est une lettre, la position du réticule est relevée et l'interpréteur retourne l'ordonnée du nom de fonction montré sur l'écran.

VI.2.7.4. Collecte de coordonnées

On active le réticule. Le caractère frappé est analysé; ce peut être :

- la lettre "R", dans ce cas, la position du réticule est relevée ;
- la lettre "S", dans ce cas, l'opérateur indique une fin de section ;
- un chiffre, dans ce cas l'opérateur désire rentrer un couple de coordonnées au clavier. Le chiffre récupéré est considéré comme premier chiffre de l'abcisse, il est affiché dans la zone des coordonnées et l'utilisateur peut alors finir de rentrer ces valeurs au clavier.

VI.2.8. Conclusion

La majeure partie de l'interpréteur décrit ici est valable quel que soit le terminal à tube à mémoire utilisé.

En effet, le module de gestion de la liste d'affichage, ainsi que la liste d'affichage elle-même sont indépendants du matériel. Seul peut différer le système de coordonnées de l'écran, auquel cas, il suffit de calculer différemment les coordonnées à stocker, mais ceci n'intervient pas sur l'organisation d'ensemble.

On peut noter que cet interpréteur est valable pour tous les terminaux Tektronix (en ce qui concerne l'affichage).

Les modules dépendant du terminal sont :

- le processeur graphique logiciel (pour utiliser au mieux toutes les possibilités offertes par le matériel)
- les modules de gestion du dialogue (pour utiliser tous les dispositifs existants).

Le prototype décrit ici a permis de piloter les terminaux suivants :

- Tektronix 4010, 4012
- Tektronix 4014, 4015
- Tektronix 4051.

Le processeur graphique est le même dans tous les cas, sauf pour le Tektronix 4015 qui dispose d'un générateur de traits de texture quelconque fourni par le matériel. Dans ce cas, le processeur graphique logiciel n'inclut pas de générateur de tiretés.

Le prototype a été écrit en Fortran, ce qui permet d'assurer une certaine portabilité (il pilote actuellement les terminaux décrits plus haut à partir d'un IBM 360/67 et à partir d'un IRIS 80).

VI.3 - COPIE POUR TABLE TRAÇANTE

Le but n'est pas ici de fournir un interpréteur particulier pour une table traçante. Ceci est réalisable, mais l'intérêt est faible car il n'y a aucune possibilité d'interaction.

Le service que l'on veut offrir à l'utilisateur est de pouvoir conserver sur le papier les images affichées sur l'écran, lorsqu'il le désire.

Dans le cas où la table traçante ne travaille pas "on line", les dessins successifs sont stockés dans un fichier qui sera ensuite mis sur bande et cette bande pilotera la table traçante.

VI.3.1. Le principe

Le principe est simple : la liste d'affichage est analysée et chaque commande est interprétée pour engendrer le code correspondant pour la table traçante.

La figure VI.11 montre l'enchaînement des opérations.

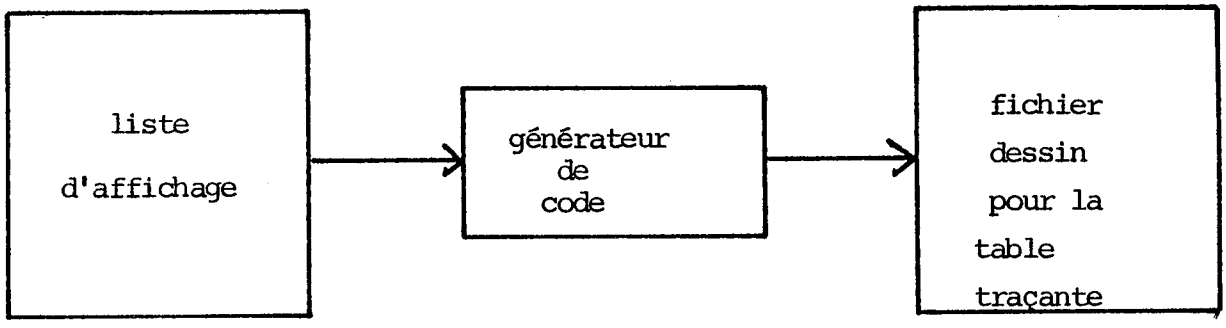


Figure VI-11 : copie pour traceur

Une remarque s'impose immédiatement : l'interprétation de la liste d'affich est liée au type de terminal utilisé. Le générateur de code pour le traceur donc dépendant du type de traceur, mais aussi du type du terminal.

Dans le cas d'un terminal à mémoire d'entretien, il faut lire le c tenu de la mémoire pour l'interpréter. Par contre, dans le cas d'un termina tube mémoire, c'est la liste d'affichage contenue dans le logiciel qui est terprétée (Le principe est le même que pour regénérer l'image, mais au lieu utiliser le processeur graphique logiciel associé au terminal, on utilise l générateur de code pour le traceur.

Notons que le sous-programme de copie pour traceur permet de défin deux paramètres :

- une échelle (en 1/10 d'écran) permettant d'avoir des copies plus petites plus grandes que la taille réelle de l'écran,
- un paramètre indiquant si l'utilisateur désire qu'un cadre soit tracé aut de son dessin pour matérialiser les bords de l'écran ; ce paramètre perme également de spécifier si le dessin en cours doit être superposé, ou non, dessin précédent.

VI.3.2. Utilisation

Ce sous-programme peut être utilisé à chaque instant par l'opérate Ceci est en général réalisé par activation d'une fonction de menu après con

truction d'une image. Suivantla réponse de l'opérateur au menu, une copie sera ou non, faite. Mais ce sous-programme est également utilisé par le logiciel. En effet, lorsque la mémoire d'entretien (ou sa simulation) est pleine et qu'u dessin est en cours, un message est envoyé à l'opérateur et la mémoire est vidée ; mais avant de la vider, le logiciel permet à l'opérateur, s'il le désire de faire une copie pour traceur de l'image de l'écran. Ainsi, par superpositio des dessins successifs, le dessin fait par le traceur représentera l'ensemble de l'image qui n'a pas pu être affichée en une seule fois sur l'écran. Ceci permet de conserver sur papier la trace d'images très complexes.

VI.4 - PROPOSITION D'INTERPRÉTEUR POUR UN TERMINAL À BALAYAGE LIGNE PAR LIGNE

Comme nous l'avons indiqué en V.1., il n'y a pas de différences fondamentales entre un interpréteur pour terminal à tube mémoire et un interpréteur pour terminal à balayage ligne par ligne. La mémoire d'entretien du terminal est une mémoire de points, contenant une information pour chaque point (couleur, intensité). [T5], [T6], [T7].

VI.4.1. Organisation

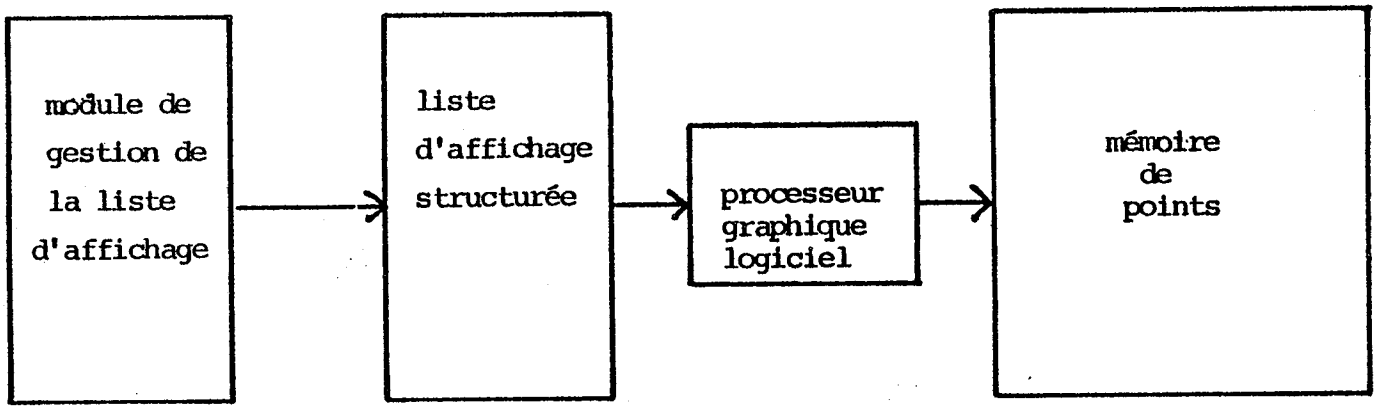


Figure VI-12 : mémoire de points

L'information analytique contenue dans la mémoire de points n'est pas suf sante pour répondre aux primitives d'effacement sélectif et de désignation

Comme dans le cas du tube à mémoire, l'interpréteur simule et gèr une liste d'affichage structurée.

La différence réside dans le rôle du processeur graphique logicie Dans le cas présent, celui-ci doit être capable de passer d'une informatio synthétique du type "droite" à une information analytique du type "point".

VI.4.2. Le processeur graphique logiciel

Pour allumer un point dans la mémoire, il est nécessaire de recon truire toute la ligne contenant ce point.

Lorsque l'on veut afficher un segment, il est donc obligatoire de construire toutes les lignes que traversent ce segment.

Pour minimiser les échanges avec la mémoire d'entretien, on trait les lignes par "paquets".

La tâche du "processeur graphique logiciel" est donc la suivante à partir d'un ensemble de segments, il doit reconstruire les lignes les su portant.

Lorsque l'on reçoit une commande d'affichage ou d'effacement, la lution la plus simple est sans doute de reconstruire toute l'image. Cela s fait en deux temps :

- tri de la liste d'affichage,
- traitement de chaque groupe de lignes.

VI.4.2.1. Tri de la liste d'affichage [R3]

Chaque segment de la liste d'affichage est classé en fonc du numéro du groupe de lignes dans lequel il apparaît pour la première foi

Pour chaque groupe de lignes, on constitue une liste de tous les segments qui apparaissent pour la première fois. La figure VI.13 montre l' ganisation de la liste.

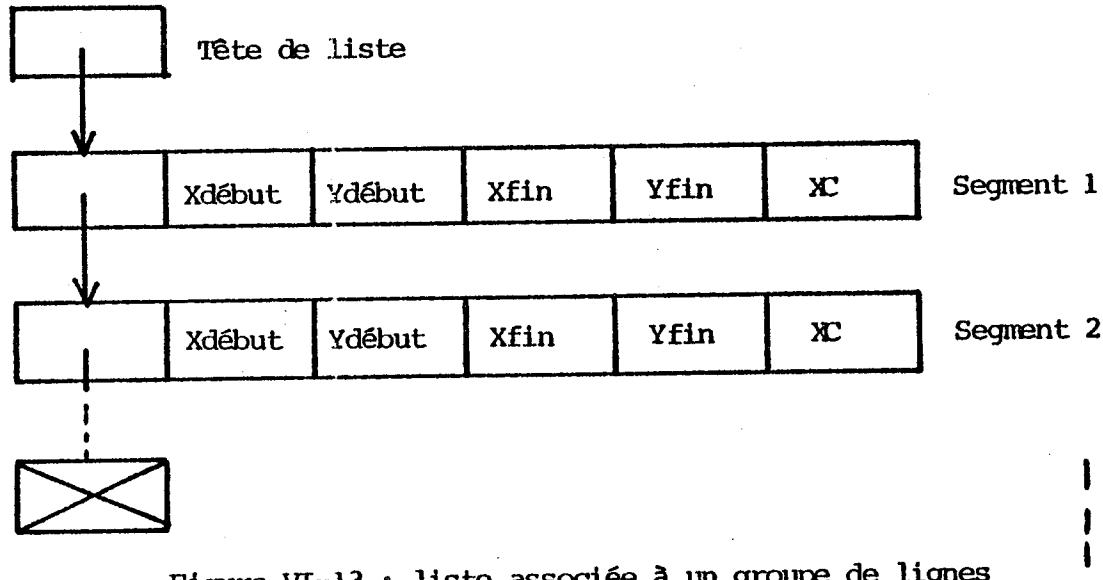


Figure VI-13 : liste associée à un groupe de lignes

On stocke les coordonnées des extrémités de chaque segment et on réserve une variable XC destinée à recevoir la position courante d'un point du segment lors du traitement du groupe de lignes.

VI.4.2.2. Construction de l'image

Le balayage se faisant du haut vers le bas, on commence par étudier le groupe de lignes du haut de l'écran. On engendre les points représentant chaque segment, jusqu'à ce que celui-ci quitte le groupe de lignes. Quand on a terminé, on envoie le groupe de lignesdans la mémoire d'entretien et la zone tampon est récupérée pour traiter le groupe suivant.

Quand on passe d'un groupe à l'autre, on transfère au nouveau groupe les segments du groupe précécent qui ne sont pas terminés (voir figure VI.14)

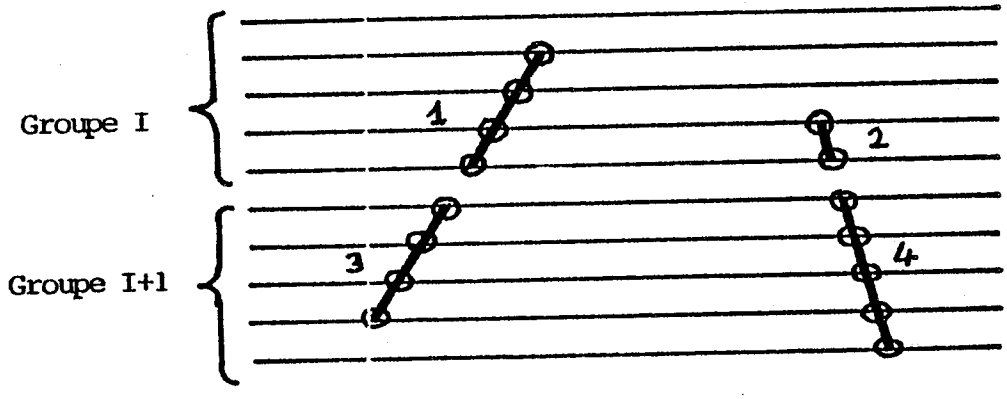


Figure VI-14 : découpage des segments

Pour afficher les segments AB et CD, on traite les segments 1 et 2 dans le groupe I et on passe 3 et 4 au groupe I+1. Enfin, pour chaque morceau de seg ment, il faut activer un générateur de points qui engendre les points les pl "proches" du segment réel [R4].

L'avantage de cet algorithme est de réutiliser la même zone tampon, ce qui évite un gaspillage de place.

VI.4.3. Application à une imprimante

L'interpréteur décrit dans ce qui précède peut permettre de sortir des dessins simples sur une imprimante.

En effet, une imprimante se comporte comme un tube à balayage ligne par ligne.

Lorsque l'on rencontre un ordre d'affichage ou d'effacement, le pro cesseur graphique logiciel construit l'image par groupes de lignes et envoie ceux-ci sur l'imprimante.

On pourra ainsi utiliser une imprimante comme un écran de 132 x 13: points par exemple.

Des symboles différents peuvent être utilisés pour simuler l'intens té lumineuse ou le type de texture.

Ce type d'interpréteur n'est pas à négliger, car dans certains cas, il peut donner des résultats très satisfaisants, de façon beaucoup plus rapi et très économique (pas d'utilisation d'écran, pas d'utilisation de table tr çante).

Les types d'applications envisageables sont les suivants :

- schémas géométriques simples,
- graphes,
- courbes simples,
- histogrammes,
- etc.

Notons que dans ce cas, le dialogue peut être pris en charge au ni veau de la console alphanumérique si le système d'exploitation est conversa tionnel.

Ce type d'interpréteur pourrait être considéré comme un outil per- mettant une première mise au point des programmes et fournissant des images simplifiées.

VI.4.4. Application à une console alphanumérique

Au même titre que l'imprimante, on peut envisager un interpréteur pour console alphanumérique. Les consoles alphanumériques simples se comportent très exactement comme une imprimante.

Par contre, certaines consoles (type Anderson-Jacobson) permettent la mise en place "graphique" du chariot et le déroulement du papier en avant et en arrière [74]. Pour ce type de matériel, deux solutions sont envisageables pour le "processeur graphique logiciel" :

- on peut négliger le déroulement du papier en arrière ; dans ce cas, on traite la console comme un dispositif à balayage ligne par ligne (on optimise le mouvement du papier).
- on utilise le déroulement du papier en arrière ; dans ce cas, la console est considérée comme un dispositif à balayage cavalier (on optimise les déplacements du chariot).

On peut penser que la seconde approche est plus intéressante car elle ne demande pas une génération point par point de l'image et elle utilise l'originalité du matériel. De plus, l'utilisation du "balayage cavalier" n'oblige pas à redessiner toute l'image lorsque l'on ajoute quelque chose au dessin, alors que le balayage ligne par ligne impose cette contrainte.

Là encore, l'utilisation de tels terminaux n'est pas à négliger car elle permet de réaliser, sans écran, sous les yeux de l'opérateur des dessins simples. Les types d'applications envisageables sont les mêmes que pour une imprimante. Le terminal peut servir de support à un dialogue simple (menu, entrée de valeurs).

Remarquons que si l'on utilise la possibilité de "balayage cavalier" l'interpréteur est très exactement le même que celui décrit pour le terminal à tube mémoire en VI.2.

VI.5. ACCOMPAGNATEUR POUR CONSOLE ALPHANUMÉRIQUE

L'"accompagnateur" est un outil mis à la disposition des utilisateur pour mettre au point les programmes en l'absence d'un écran graphique.

En effet, on constate facilement que la mise au point d'un programme graphique se fait en deux phases :

- mise au point algorithmique (algorithme de l'application et déroulement du dialogue),
- test de la validité des images affichées.

Or, la première étape ne nécessite pas l'utilisation d'un terminal graphique c'est pourquoi nous mettons à la disposition des programmeurs d'application la possibilité de suivre le déroulement de leurs programmes sur une console alphanumérique.

L'accompagnateur reçoit les mêmes informations du noyau qu'un interpréteur pour terminal graphique.

La trace des différentes fonctions du logiciel est écrite sous forme du nom de la fonction et de la valeur de certaines variables.

VI.5.1. Les primitives de déclarations

Dans le cas d'un terminal graphique, ces primitives ne sont pas répercutées au niveau de l'interpréteur, car les données correspondantes sont stockées dans le noyau. Cependant, le noyau engendre des commandes intermédiaires pour ces primitives ; celles-ci sont ignorées dans tous les cas, sauf par l'accompagnateur qui trace ainsi l'exécution des primitives de définition FENETRE, CLOTURE et MODE, et des primitives de déclarations TEXTES et POINTS.

VI.5.2. Les primitives d'affichage

L'affichage d'une section est tracé en deux temps :

- description des attributs (mode, fenêtre, clôture) de la section
- description des coordonnées.

Ces deux fonctions sont réalisées après consultation de l'opérateur : s'il ne désire pas la description des attributs, celle-ci n'est pas faite. S'il le désire, il peut faire lister les coordonnées des points par groupe de 10.

Les sections de texte sont décrites de la même façon :

- description des attributs,
- écriture du texte.

Les messages sont écrits en clair.

Pour l'effacement, l'accompagnateur fournit la liste des sections à effacer

VI.5.3. Les primitives de dialogue

Celles-ci sont mises en oeuvre par l'intermédiaire du clavier alphanumérique.

Les entrées de valeurs ne posent aucun problème. Lorsqu'une demande est reçue, l'accompagnateur indique le nombre maximum d'entrées et le numéro du message puis l'utilisateur peut taper ses valeurs au clavier alphanumérique Lors de l'invocation d'un menu les noms et les numéros des fonctions autorisées sont listés puis l'opérateur peut répondre au menu en tapant des numéros au clavier. La collecte de coordonnées est réalisée en tapant les coordonnées des points successifs au clavier.

Enfin, la fonction d'identification n'est pas simulable. Cependant, pour pouvoir contrôler le déroulement du programme, l'opérateur peut taper au clavier des couples (numéro de figure, numéro de corrélation), ceux-ci seront renvoyés au noyau comme les résultats de désignations effectives.

Les possibilités d'annuler tout ou partie des entrées, ou de valider l'ensemble sont réalisées, comme sur les terminaux Tektronix par des lettres de contrôles.

Un exemple de trace de programme fournie par l'accompagnateur est donné en annexe.

L'expérience a prouvé l'utilité d'un tel outil, pour que les écrans soient utilisés au maximum.

Il est clair que l'accompagnateur ne rentre dans aucune des deux catégories d'interpréteurs, car les données graphiques ne sont pas conservées dans une liste d'affichage.

CHAPITRE VII - DIRECTIVES POUR L'INSERTION DU LOGICIEL DANS UN CONTEXTE QUELCONQ

Dans ce chapitre, nous allons signaler les points à examiner lorsque l'on désire implanter le logiciel dans un nouveau contexte.

Nous décrivons tout d'abord les différents modules constituant le noya et ceux constituant un interpréteur quelconque.

VII-1- DESCRIPTION TECHNIQUE DU PROTOTYPE DU NOYAU

Nous listons ici les sous-programmes constituant le noyau.
Rappelons que celui-ci est écrit en Fortran "standard".
Nous pouvons distinguer deux niveaux de sous-programmes :

- les sous-programmes utilisateur (correspondant aux différentes primitives du logiciel),
- les sous-programmes de traitement interne (inaccessibles au programmeur d'appl cation).

VII.1.1. Les modules utilisateur

On retrouve ici toutes les primitives sous forme de sous-programmes Fortran :

FENETR (nfig, igx, igy, sdx, sdy)

les bornes de la fenêtre sont des réels.

CLOTUR (nclot, igx, igy, sdx, sdy)

les bornes de la clôture sont des entiers compris entre 0 et 100

POINTS (nfig, Tlong, Tx, Ty, Tlegende)

Tlong et Tlegende sont des tableaux d'entiers à une dimension.
Tx, Ty sont des tableaux de réels à une dimension.

TEXTES (nfig, Tlong, Texte, Tx, Ty)

Tlong et Texte sont des tableaux d'entiers à une dimension.
Tx, Ty sont des tableaux de réels à une dimension.

MODE (nfig, ncorrel, nsection, descripteur)

descripteur est un tableau d'entiers à une dimension.

AFFICH (nfig, ncorrel, nfen, nclot)
EFFACE (nfig, ncorrel)
MESSAG (nmess, message, nclot, x, y)
message est un tableau d'entiers à une dimension.
x, y sont des entiers compris entre 0 et 100.

EFFMES (nmess)
EDENTI (nmess, nzone, entier)
EDREEL (nmess, nzone, format, réel)
EDCHAI (nmess, nzone, chaine)
chaine est un tableau à une dimension de 19 entiers.
(76 caractères maximum).

NVAL (nbmax, nmess, tentier)
tentier est un tableau d'entiers à une dimension.
RVAL (nbmax, nmess, tréel)
tréel est un tableau de réels à une dimension.
CVAL (nbmax, nmess, tchaine)
tchaine est un tableau d'entiers à deux dimensions.
La première dimension est égale à 19 (76 caractères).
Ainsi une chaine est récupérée sur une colonne du tableau Fortran.

MENU (nbmax, descripteur, tretour)
descripteur et tretour sont des tableaux d'entiers à une dimension.

POSIT (nbmax, nfig, nfen, nclot)
IDENT (nbmax, nfig, tfig, tcorrel)
tfig et tcorrel sont des tableaux d'entiers à une dimension.

D'une façon générale, tous les numéros sont des entiers. Notons que NBMAX est obligatoirement une variable car il sert de paramètre d'entrée et de paramètre de retour (risque d'effet de bords si on met une constante).

VII.1.2. La structure de donnée

Les diverses tables du noyau décrit dans le chapitre V sont rassembl dans un common (ŞSCŞ).

Pour chaque table, on trouve une série de tableaux avec leur dimensio maximale et l'indice de la première zone libre. Pour les fenêtres on trouve :

NFEN	numéro des fenêtres	(tableau d'entiers)
FENIGX, FENIGY, FENSDX, FENSDY	bornes des fenêtres	(tableaux de réels)
MAXFEN	dimension de la table des fenêtres	
IFEN	indice de la première zone libre.	

Pour les clôtures on trouve les mêmes rubriques :

NCLO	numéro des clôtures	(tableau d'entiers)
CLOIGX, CLOIGY, CLOSDX, CLOSDY	bornes des clôtures	(tableaux d'entiers)
MAXCLO	dimension de la table des clôtures	
ICLO	indice de la première zone libre.	

La structure de données du noyau contient également deux tables répertoriant d'une part les figures qui contiennent des sections de points, d'autre part les figures qui contiennent des sections de texte.

table des figures contenant des sections de points :

NFIG	numéro de la figure	(tableau d'entiers)
FIGTLO	adresse du tableau des longueurs	(tableaux d'entiers)
FIGTX	adresse du tableau des abscisses	
FIGTY	adresse du tableau des ordonnées	
FIGLEG	adresse du tableau des légendes	
MAXFIG	dimension de la table	
IFIG	indice de la première zone libre	

table des figures contenant des sections de texte :

NTEX	numéro de la figure	(tableaux d'entiers)
TEXTLO	adresse du tableau des longueurs	
TEXCH	adresse de la chaîne de texte	
TEXTX	adresse du tableau des abscisses	
TEXTY	adresse du tableau des ordonnées	
MAXTEX	dimension de la table	
ITEX	indice de la première zone libre	

Ces tables sont remplies lors de l'invocation des primitives POINTS TEXTES. Enfin la structure de donnée du noyau contient deux tables de descript des modes déclarés.

table des modes pour les sections de points :

SECFIG	numéro de la figure à laquelle appartient la section
NSEC	numéro de la section dans la figure
SECCOR	numéro de corrélation associé à la section
SECMOD	description du mode
MAXSEC	dimension de la table
ISEC	indice de la première zone libre.

Cette table est mise à jour lors de l'invocation de la primitive MOI (voir la mise à jour dans le chapitre V).

SECMOD est un tableau à deux dimensions, il est composé en fait de sept sous-tableaux qui contiennent respectivement :

- la couleur
- la texture
- l'épaisseur
- l'intensité
- le type de représentation
- le marquage
- le système de coordonnées

La valeur interne par défaut, au niveau du noyau est : 1000.

table des modes pour les sections de texte

SECFT	numéro de la figure
NSECT	numéro de la section
SECCT	corrélation de la section
SECMT	description du mode
MAXST	dimension de la table
ISECT	indice de la première zone libre.

SECMT est un tableau à deux dimensions, il est composé de cinq sous-tableaux qui contiennent respectivement :

- la taille des caractères
- l'orientation du texte
- la couleur
- l'épaisseur
- l'intensité

La valeur par défaut est 1000.

Enfin le "common" du noyau (ßßCß) contient une variable (INITIA) qui vaut zéro au début de l'exécution et passe à 1 dès que l'initialisation a été faite.

Le sous-programme d'initialisation est activé par la première primitiv du logiciel rencontrée dans l'application.

Le noyau contient également une autre structure de donnée : la zone de communication avec les interpréteurs. C'est un "common" (ßBUFß) qui contient

BUFCDE	: variable entière destinée à recevoir la commande intermédiai
BUFN	: tableau d'entiers
MAXBN	: longueur de la zone des entiers
BUFR	: tableau de réels
MAXBR	: longueur de la zone des réels.

VII.1.3. Les modules internes

Ils sont relativement nombreux, nous allons ici donner leurs noms, leurs paramètres et leurs fonctions.

- $INITI : sous-programme d'initialisation.
  appelé par n'importe quelle primitive du logiciel.
  il affecte les dimensions des différentes tables.
  il met à 1 les indices des zones libres dans les tables.
  il envoie la commande d'initialisation à l'interpréteur.
  il met à 1 la variable d'initialisation.

- $MODG (nf, ncorrel, nsection, description, secfig, nsec, seccor, secmod, nfi
  maxsec, isec, id2, *, *)
  Ce sous-programme est invoqué par la primitive MODE
  nf, ncorrel, nsection, description sont les paramètres d'appel de MODE.
Le sous-programme $MODG permet d'indiquer quelle table de mode devra être mise à jour, en fonction du type du mode (points ou texte) : les paramètres secfig, nsec, seccor, secmod représentent l'une des deux tables de mode; nfig représent une des deux tables de figures; maxsec, isec représentent les indices de la tab des modes concernée; id2 permet d'indiquer la dimension du tableau de descripti du mode (5 ou 7); enfin, les deux sorties en erreur, permettent de sortir du sc programme :
- sur détection d'une erreur de syntaxe dans le descripteur,
- lorsque la table des modes est pleine.

Le sous-programme $MODG gère la table des modes pour l'ensemble des figures déclarées (voir chapitre V).

- $MODF (fig, ncorrel, nsection, mode, ...)
Les paramètres non mentionnés sont les mêmes que ceux du sous-programme SMODG. $MODF est invoqué par $MODG pour gérer la table concernée en fonction de la déc ration du mode pour chacune des figures existantes (voir gestion des modes, au chapitre V).

- $FS (fig, sec, cor, secfig, nsec, seccor, isec, aff, iamod)
Ce sous-programme est invoqué par la primitive AFFICHER, pour chaque section de chaque figure concernée par l'affichage.
fig et cor sont les paramètres de AFFICHER
sec est le numéro de la section étudiée.
secfig, nsec, seccor, isec représentent l'une ou l'autre table des modes.
le sous-programme $FS a pour but de déterminer si la section doit être affichée ou non (réponse dans le paramètre aff) et si oui avec quel mode (l'indice du mo dans la table est renvoyé dans iamod).

- $LIST (nfig, Lf, nf, Tfig, Tmax)
Ce sous-programme recense toutes les figures du tableau Tfig (de dimension Tmax) qui sont concernées par le paramètre nfig. En effet celui-ci peut être négatif, nul, ou positif. En fonction de sa valeur, les indices dans Tfig des figures co cernées sont rangés dans le tableau Lf et nf reçoit le nombre total de figure à étudier.
Concrètement, ce sous-programme est invoqué par les primitives AFFICHER et EFFA pour établir la liste des figures à étudier.

- $ABS (Xabs, Yabs, x, y)
Xabs et Yabs sont les coordonnées absolues d'un point courant dans une section, X et Y sont les coordonnées relatives d'un point (le suivant).
Ce sous-programme est invoqué par la primitive AFFICHER quand les coordonnées de la section sont relatives.
Au retour x et y contiennent les coordonnées absolues du point et le point cour (Xabs, Yabs) a progressé jusqu'à ce nouveau point.

- $MODC (mode, modgr, type, *)
Ce sous-programme est invoqué par $MODG(donc indirectement par MODE).
Il analyse le descripteur du mode contenu dans le tableau mode. Type indique s' s'agit d'un mode pour points (1) ou pour texte (2).

Au retour les valeurs des différents paramètres du mode sont rangés dans modgr (de longueur 5 ou 7 suivant la valeur de type).

Il y a sortie en erreur sur détection d'une erreur de syntaxe dans la description du mode.

Le traitement du retassement des tables, commun au noyau et aux inte préteurs utilise trois sous-programmes

- $TASS (index, T1)

Ce sous-programme analyse le tableau T1, dont la dimension est égale à (index- Si T1(I) est nul, la case est libre, sinon, elle est occupée. En fonction de c critère, $TASS remplit un tableau TL de logique avec "vrai" si la case est lib "faux" sinon. Ce tableau est commun à $TASS et aux deux sous-programmes suivan

- $TASS2 (T, imax, index)

Ce sous-programme retasse le tableau T (de dimension imax) en fonction des val contenues dans le tableau TL. L'algorithme de retassement est décrit au chapit Au retour index contient l'index de la première case libre.

- $TASS3 (T, D2, imax, index)

Ce sous-programme exécute la même fonction que $TASS2, mais le tableau T a deu dimensions : D2 et imax.

Ces trois sous-programmes sont utilisés pour retasser les tables de mode dans le noyau, et les tables contenues dans les interpréteurs.

- $CLIP (X1, Y1, X2, Y2, XR, XL, YB, YT)

Ce sous-programme est invoqué par AFFICHER, pour réaliser le coupage sur le se ment d'origine (X1, Y1) et d'extrémité (X3, Y2) les derniers paramètres sont l bornes de la fenêtre.

Au retour (X1, Y1) et (X2, Y2) contiennent les extrémités du segment visible. $CLIP partage avec AFFICHER un "common" ($COUP) qui contient trois renseigneme qui indiquent :

- si le segment sort de la fenêtre,
- si le segment rentre dans la fenêtre,
- s'il a quelque chose à afficher ou non.

Le sous-programme de coupage utilise trois sous-programmes de servic

- $SWAP (X, Y)

interversion de deux réels

- $SWAPL (L1, L2)

interversion de deux logiques

- $CODE (X, Y, M, C, D, U, XR, XL, YB, YT)

ce sous-programme donne la position d'un point (X, Y) par rapport au quatre droites qui limitent la fenêtre (XR, XL, YB, YT) le résultat est stocké dans les logiques M, C, D, U qui prennent la valeur "vrai" si le point est du même côté que la fenêtre par rapport à la droite considérée, "faux" sinon.

Enfin, le noyau contient un certain nombre de sous-programmes de servi

- $INDEX (val, T, imax)

Ce sous-programme recherche l'élément val dans le tableau T jusqu'à l'index (imax-1).

Cette fonction entière prend pour valeur l'indice de l'élément dans le tableau s'il existe, sinon la valeur retournée est égale à imax.

Cette fonction entière est utilisée dans beaucoup de primitives, pour savoir si un élément existe déjà et si oui quelle est sa place dans la table. S'il existe déjà, on écrasera les anciens paramètres, sinon, on stocke les paramètres dans la première case libre (imax).

- $CODCH (val, ich, ilg, ifor)

Ce sous-programme engendre dans ich, la chaîne de caractères représentant l'entier ou le réel val. ifor précise le format de l'édition. Au retour ilg contient la longueur de la chaîne engendrée.

- $TRI

- $DECID

Ces deux sous-programmes sont utilisés par les différentes primitives analysant des chaînes de caractères (mode, menu, message).

- $ERR (nfonction, nerreur, num, chaîne)

Ce sous-programme est appelé par toutes les primitives en cas d'erreurs.
nfonction précise dans quelle fonction l'erreur est détectée.
nerreur précise le type de l'erreur.
num et chaîne contiennent les paramètres erronés si nécessaire.

VII-2- LA PORTABILITE DU NOYAU

Le noyau est écrit en Fortran standard.

Toutes les variables simples et les tableaux sont de longueur standard (ici 4 octets).

La portabilité des modules décrits plus haut semble assurée. Le prototype actuel a été écrit sur IBM 360/67 ; il a été porté sans aucun problème sur IRIS 80. Cependant, le noyau utilise quelques sous-programmes de service dont nous n'avons pas encore parlé. Ces sous-programmes se répartissent en deux classes :

- manipulation d'adresses
- manipulation de chaînes de caractères.

VII-2-1- Les sous-programmes de manipulation d'adresses

Ces sous-programmes sont liés au calculateur, au système d'exploitation et au compilateur FORTRAN utilisés.

Les manipulations nécessaires sont les suivantes :

- stockage de l'adresse d'une variable,
- récupération d'une valeur contenue dans une variable dont on connaît l'adresse
- stockage d'une valeur dans une variable dont on connait l'adresse.

Nous décrivons très précisemment ces sous-programmes car leur réécriture est nécessaire quand on veut porter le noyau sur une nouvelle machine.

VII-2-1-1- Stockage de l'adresse d'une variable

$ADR (variable, adresse)

- variable est l'identificateur de la variable dont on veut stocker l'adresse.
- adresse est l'identificateur de la variable dans laquelle on stocke l'adresse.

Le résultat est décrit par la figure VII.1:

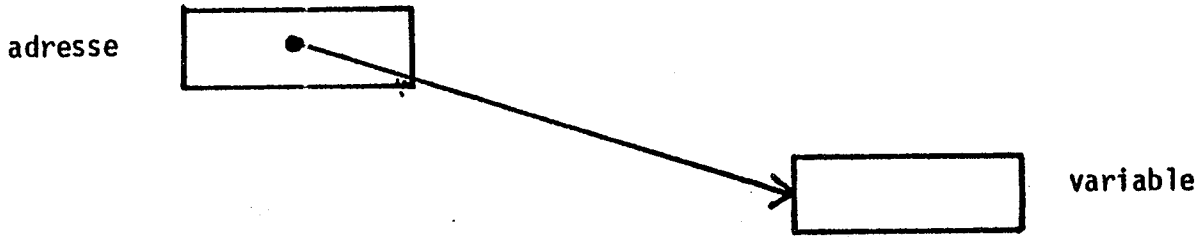


Figure VII.1 - Stockage d'une adresse

VII-2-1-2 - Récupération de valeurs

$VAL (adresse, valeurs, longueur)

- adresse est l'identificateur de la variable contenant l'adresse.
- valeurs est l'identificateur de la variable qui recevra les valeurs récupérées
- longueur est la longueur de la zone dont on veut récupérer le contenu (cette longueur est évaluée en nombre d'entiers, ou de réels, de longueur standard).

La figure VII.2 décrit l'exécution de ce sous-programme.

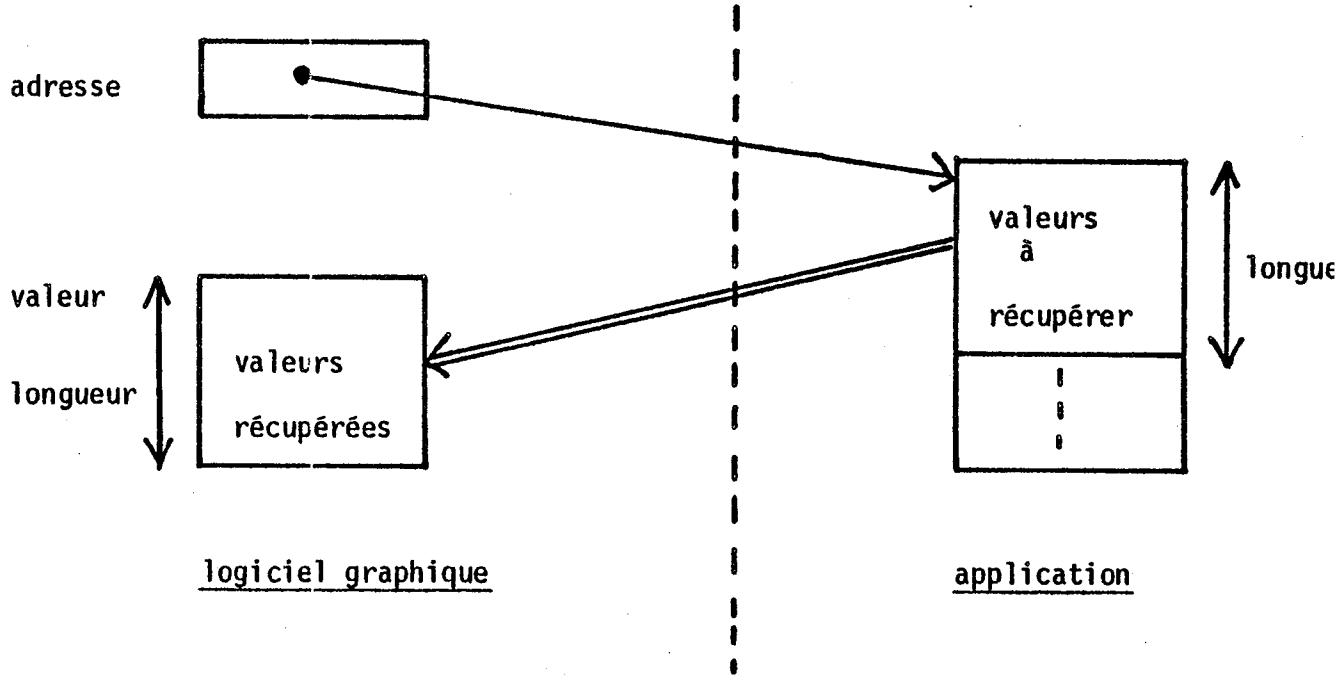


Figure VII.2 : Récupération de valeurs

Ce sous-programme est utilisé par la primitive AFFICHER, pour récupérer chez l'utilisateur le contenu des différents tableaux associés à une figure.

VII-2-1-3 - Stockage de valeurs

%INV (adresse, valeurs, longueur)

Ce sous-programme est le symétrique du précédent.

- adresse est l'identificateur de la variable contenant l'adresse, chez l'utilisateur, où l'on veut stocker les valeurs.
- valeurs est l'identificateur de la zone contenant les valeurs à stocker.
- longueur est la longueur de cette zone (évaluée en nombre d'entiers, ou de réels de longueur standard.

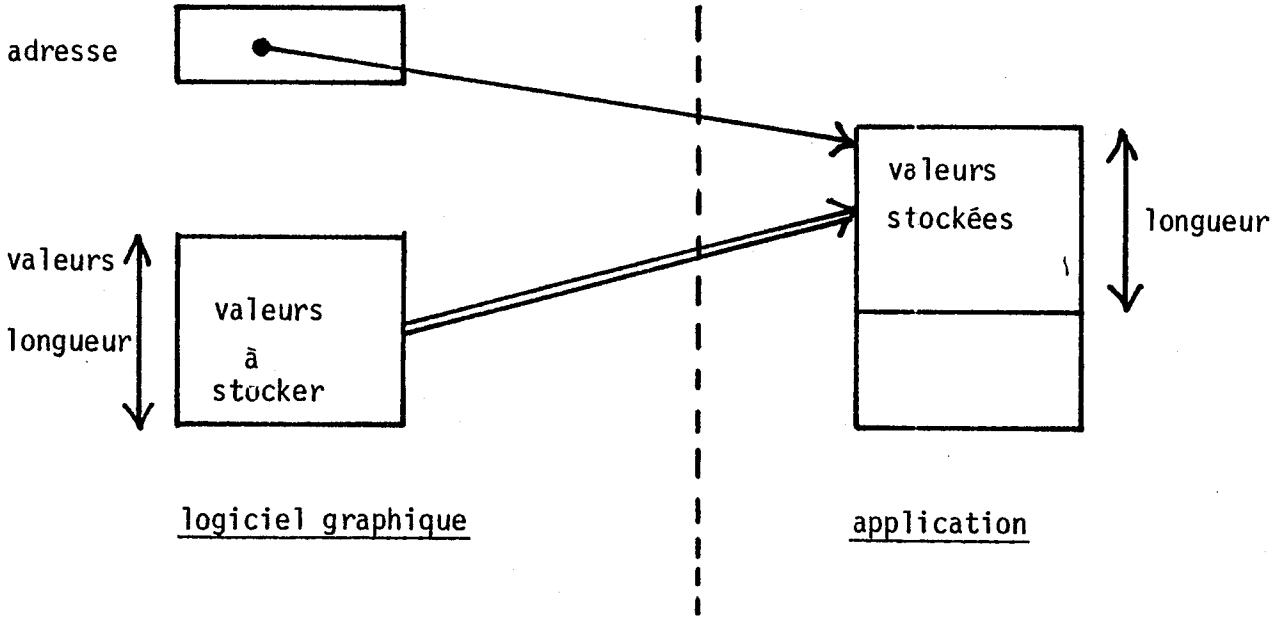


Figure VII.3 : stockage de valeurs

Ce sous-programme est utilisé par la primitive POSITION pour stocker dans les tableaux de l'utilisateur, les points collectés.
Ces trois sous-programmes travaillent à partir d'adresses réelles en mémoire ; ils sont donc nécessairement écrits dans le langage de la machine.

Notons qu'ils sont actuellement disponibles en Assembleur IBM et en méta symbole IRIS 80 ; et qu'en tout état de cause, leur réécriture est très simple.

VII-2-2 - Les sous-programmes de manipulation de caractères

Ces sous-programmes sont utilisés à la fois dans le noyau et dans les interpréteurs. Ils sont au nombre de quatre.

VII-2-2-1 - Insertion d'un caractère dans une chaîne avec incrémentation de l'index

$ECRI (chaîne, index, indexmax, car, *)

Ce sous-programme insère le caractère car, dans la chaîne à l'indice index+1. On effectue un contrôle sur l'index courant, qui ne doit pas dépasser indexmax (sinon, sortie en erreur).
Le caractère est placé dans l'octet de droite d'un mot.

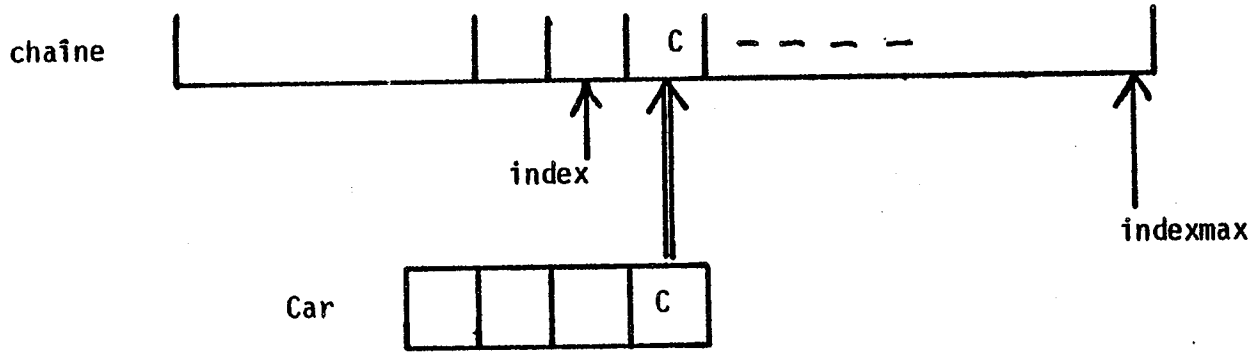


Figure VII-4 : Insertion d'un caractère ($ECRI)

Au retour l'index courant a été incrémenté de 1.

VII-2-2-2 - Récupération d'un caractère d'une chaîne avec incrémentation de l'index

Ce sous programme est le symétrique du précédent.

$NEXTC (chaîne, index, indexmax, car, *)

On récupère le caractère à l'indice index+1 de la chaîne, dans l'octet de droite du mot car.

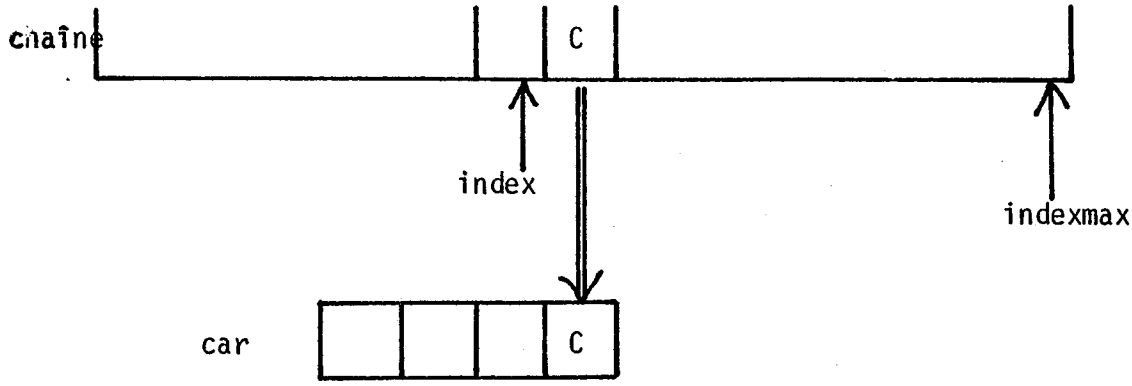


Figure VII-5 : récupération d'un caractère

Au retour, l'index courant a été incrémenté de 1.

VII-2-2-3 - Insertion d'un caractère dans une chaîne sans incrémentatic de l'index

$REPC (chaîne, index, car)

Ce sous-programme insère le caractère contenu dans car, à l'indice index dans la chaîne.
Il n'y a pas de contrôle sur le dépassement de la chaîne, ni d'incrémentation de l'index.

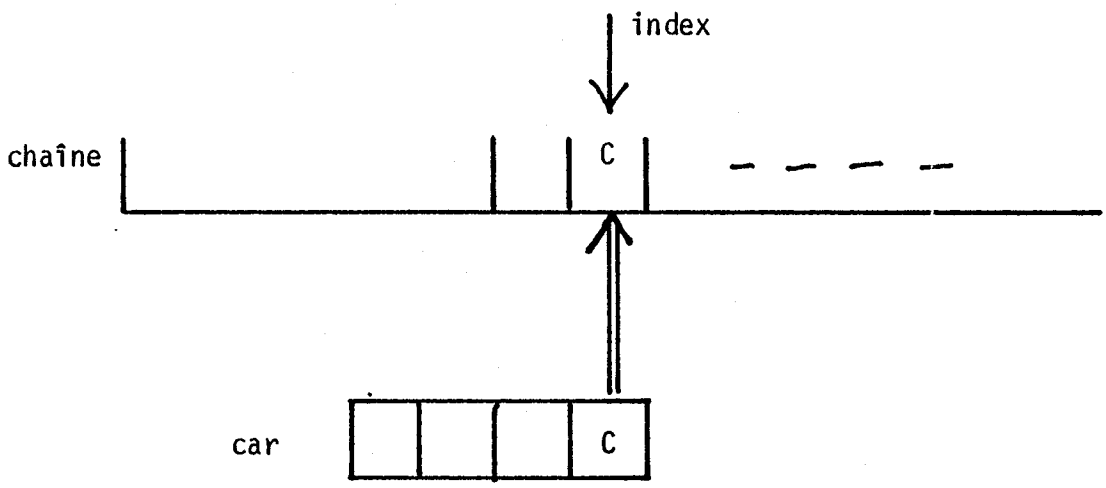


Figure VII-6 : Insertion d'un caractère ($REPC)

VII-2-2-4 - Récupération d'une sous-chaîne

$GETCH (chaîne, index, longueur, sous-chaîne)

Ce sous-programme charge dans la sous-chaîne, les caractères de la chaîne, à

partir de l'indice index, sur la longueur indiquée.

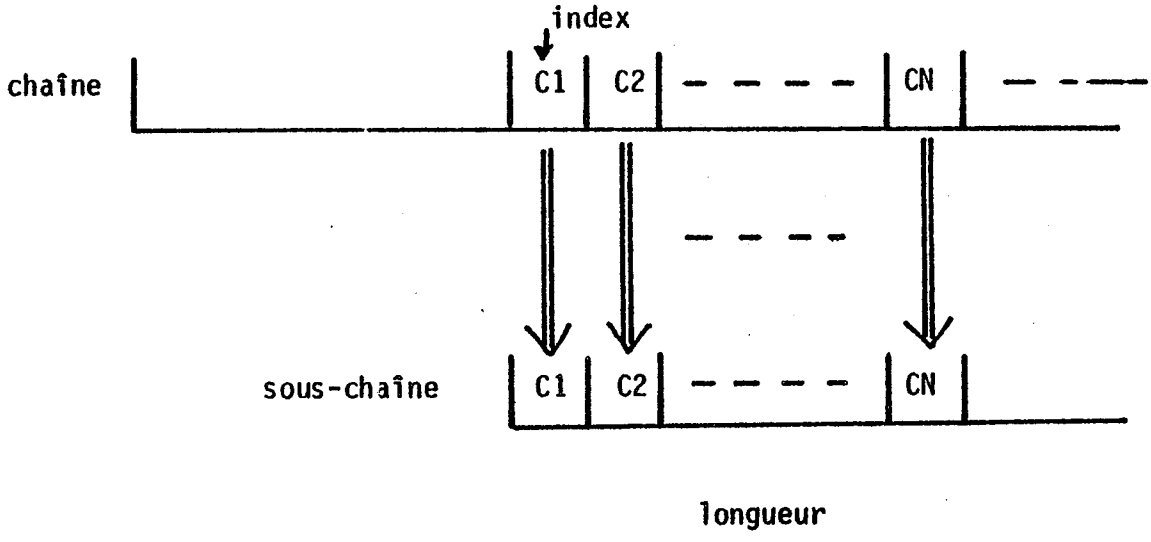


Figure VII.7 : Récupération d'une sous-chaîne

La longueur est exprimée en nombre de caractères.

VII-2-2-5- Portabilité

Ces quatre sous-programmes ont d'abord été écrits en Fortran standard : utilisat d'entiers standards uniquement. Nous disposons donc d'une version portable de ce sous-programmes (à condition que les mots de la machine contiennent quatre carac tères) ; mais celle-ci est très lourde (multiplications et divisions par 256 pou faire les décalages). Dans un second temps, nous avons réalisé une version plus élégante en Fortran : elle utilise des logiques de longueur 1. Cette version, plus rapide à l'exécution est portable à condition que le compilateur Fortran accepte les logiques de longueur 1.
Quoi qu'il en soit, si l'on voulait réaliser ces primitives indépendamment de la machine, il faudrait les paramétrer en fonction du nombre de caractère par mot de la machine utilisée. Mais ceci risquerait de les rendre très lourdes.

VII-3 - DESCRIPTION TECHNIQUE DES INTERPRETEURS

Nous mentionnerons ici les sous-programmes communs à tous les interpréteurs (dépendants ou indépendants du type de matériel) et les modules spécifiques aux interpréteurs pour terminaux ne disposant pas d'une image synthétique.

VII-3-1 - Les modules communs à tous les interpréteurs

On regroupe ici les modules existant dans tous les interpréteurs, mais dépendant du terminal et les modules de service, indépendants du matériel.

VII-3-1-1 - Les modules dépendant du terminal

Ce sont tous les sous-programmes de traitement de chacune des commandes du code engendré par le noyau.
Ces sous-programmes n'ont pas de paramètres, car ils utilisent, en entrée ou en sortie, la zone de communication avec le noyau.

- $INT
C'est le sous-programme d'accès à l'interpréteur.
C'est lui qui est invoqué par les différents programmes du noyau. C'est le seul lien entre le noyau et l'interpréteur.
Il analyse seulement le code de la commande reçue dans la zone de communication et fait appel au sous-programme de traitement correspondant. Il y a donc autant de sous-programmes de traitement que de commandes intermédiaires (voir chapitre

- I : Initialisation
- S : récupération des attributs d'une section à afficher
- C : récupération et affichage d'une section de points
- L : récupération et affichage d'une étiquette
- T : affichage d'une section de textes
- F : effacement de section(s)
- A : affichage d'un message et stockage des caractéristiques de ces zones réser
- Q : effacement de message(s)
- E : édition alphanumérique dans une zone
- V : introduction de valeurs entières ou réelles
- W : introduction de chaînes de caractères
- D : désignation
- P : collecte de coordonnées
- M : menu

Les lettres qui précèdent sont des symboles permettant d'identifier chaque sous-programmes.

Enfin, signalons que les interpréteurs disposent tous d'une possibilité d'édition en "local" ; c'est-à-dire d'une édition sur l'écran qui n'est pas liée à une donnée graphique de l'application. Ce sous-programme est utilisé pour écri des messages systèmes qui donnent des indications sur le dialogue et seront effa dès la fin de celui-ci. La forme générale est la suivante :

$EDIT (message, longueur, Xecran, Yecran)

Le message sera affiché à partir des coordonnées écrans (Xecran, Yecran).
Le traitement interne à tous ces sous-programmes est évidemment très lié au type de terminal, mais tout interpréteur doit contenir l'ensemble de ces modules.

VII-3-1-2 - Les modules indépendants du terminal

Ce sont d'une part les modules de services et d'autre part les générateurs particuliers.
Parmi les modules de service on trouve :

- $VALUR . (chaîne, lgmax, valeur, *)

Ce sous-programme convertit la chaîne de caractères en un réel qui sera renvoyé dans valeur (le format d'entrée est quelconque)
Si la chaîne n'est pas correcte, il y a sortie en erreur.

- $VALUI (chaîne, lgmax, valeur, *)

Ce sous-programme convertit la chaîne de caractère en une valeur entière.

- $CLOT

Ce sous-programme calcule les bornes dans le repère écran de la clôture réelle (cf. chapitre V). Il n'a pas de paramètres, car il travaille sur un "common" ($ATTR) qui contient les bornes de la fenêtre et de la clôture d'origine dans le repère-écran.

- $MARQ (marqueur, indice, chaîne, nbcar)

Ce sous-programme engendre la chaîne de caractères représentant le marqueur. Le symbole de marquage est contenu dans marqueur, cadré à droite. L'indice courant est contenu dans indice. Au retour chaîne contient la suite de caractères constituant le marqueur et nbcar contient leur nombre.

Ces modules apparaissent dans tous les interpréteurs et sont indépenda du matériel.
Les générateurs sont au nombre de deux :

- générateur de caractères
- générateur de segments de texture quelconque.

Ceux-ci ne sont pas complètement indépendants du matériel. En effet, le traitem d'ensemble est commun, mais ils font appel à deux fonctions du logiciel élément du terminal :

- mise en place du faisceau éteint,
- tracé d'un segment à partir de la position courante.

En dehors de ces appels, le traitement proprement dit est commun à tous les int préteurs.

- ⁄GECAR (X,Y, base, hauteur, angle, chaîne, nbcar)

Ce sous-programme engendre des caractères quelconque. X,Y est la position initi le ; base et hauteur sont la longueur et la hauteur du rectangle englobant le texte ; angle est l'angle orienté avec l'horizontale.

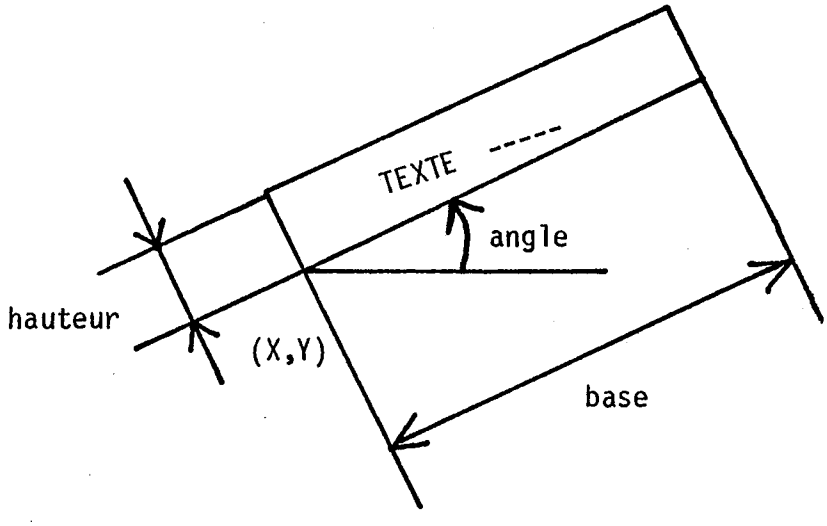


Figure VII-8 : Génération de caractères

Les variables X, Y, base, hauteur sont des réels évalués entre 0. et 1.
Le générateur de tireté est invoqué par :

- ØTIRET (IXD, IYD, IXF, IYF, mode)

(IXD, IYD) sont les coordonnées (dans le repère écran) du début du segment.
(IXF, IYF) sont les coordonnées de la fin du segment.
mode précise la texture :

1 - tireté court,
2 - tireté long,
3 - pointillé,
4 - mixte

VII-3-2 - Les modules spécifiques à un interpréteur pour terminal sans image structurée

Ils sont au nombre de deux :

- le processeur graphique logiciel
- le module de mise à jour de l'écran.

VII-3-2-1 - Le processeur graphique

-ØPG (pt1, pt2)

Il exécute les commandes qui sont contenues dans la liste d'affichage simulée.
Lors de l'appel, on lui fournit l'adresse de l'instruction à exécuter dans la liste (pt1).
Au retour, après exécution, le processeur graphique retourne dans pt2 l'adresse de la prochaine instruction à exécuter (en séquence, ou non).

VII-3-2-2 - Le module de mise à jour

- ØMAJ

Ce module assure la mise à jour de l'écran par rapport à la liste d'affichage.
Le principe est simple :

- effacement de l'écran,
- mise en place de pt1 en haut de la liste d'affichage,
- appel de ØPG (pt1, pt2),
- pt1 = pt2
- nouvel appel au processeur graphique, et ainsi de suite jusqu'à ce que pt1 soi arrivé en bas de la liste.

VII-3-3 - La structure de données commune à tous les interpréteurs

On y trouve essentiellement trois tables (voir chapitre V).

La table des sections affichées

SN : nom interne de la section (± 1000 * nfig ± nsection)
SCO : corrélation de la section
SLG : longueur de la section dans la liste d'affichage
SPT : adresse de la section dans la liste d'affichage
MAXSAF: dimension de la table
IS : indice de la première zone libre dans la table

La table des messages

MN : numéro du message
MLG : longueur du message dans la liste d'affichage
MPT : adresse du message dans la liste d'affichage
MCMAX: nombre de zones réservées dans le message
MCURS: numéro de la zone courante
MAXMES: dimension de la table
IM : indice de la première zone libre dans la table.

La table des zones réservées

ZN : numéro interne de la zone (100 * nmess + nzone)
ZLG : longueur de la zone
ZPT : soit l'adresse de la zone en mémoire
soit le nombre de caractères qui précèdent la zone dans le message
MAXZON: dimension de la table
IZ : indice de la première zone libre dans la table

Les dimensions des tables sont précisées au moment de l'initialisation. Toutes ces tables sont constituées de tableaux d'entiers de longueur standard.

VII-3-4 - Structure de données spécifique

Elle concerne la table des messages et la liste d'affichage. La table des messages contient deux tableaux supplémentaires :

MX : abscisse du début du message, dans le repère écran
MY : ordonnée du début du message

La liste d'affichage contenue dans le prototype est un tableau d'entiers de longueur 2 :

MEMOIR (de 4096 éléments)

Enfin, on dispose de trois indicateurs :

MAJ : indique si un effacement a été enregistré
EFFTS: indique si un effacement de toutes les sections a été enregistré
EFFIM: indique si un effacement de tous les messages a été enregistré.

Signalons également trois variables relatives à la liste d'affichage (qu'elle soit dans une mémoire d'entretien ou non) :

MAXMEN : précise la dimension de la mémoire (ou de sa simulation) c'est donc la longueur maximale de la liste d'affichage.

PTBUFS : indique l'adresse en mémoire de la zone libre derrière la zone contenant les sections.

PTBUFM : indique l'adresse du début de la zone des messages dans la mémoire.

Ces deux index sont soit des adresses en mémoire d'entretien, soit des indices dans le tableau simulant la liste d'affichage.

VII-3-5 - Portabilité d'un interpréteur déjà existant

Comme nous l'avons déjà indiqué, l'interpréteur pour terminal à tube mémoire est composé de deux parties

- un module de gestion de la liste d'affichage
- un processeur graphique logiciel.

VII-3-5-1 - Changement de calculateur

Si l'on veut transporter un interpréteur, pour un même terminal, sur un site différent, deux points sont à étudier :

- l'interpréteur utilise les mêmes sous-programmes de manipulation de caractères que le noyau. Il faut donc vérifier que ces sous-programmes sont présents et compatibles.
- il est nécessaire de réécrire le logiciel élémentaire du terminal graphique en respectant les noms et les paramètres des sous-programmes élémentaires.

En dehors de ces deux problèmes, la portabilité du prototype d'interpréteur pou terminal à tube mémoire semble possible.

VII-3-5-2 - Prise en charge d'un nouveau terminal à tube mémoire

Si l'on veut prendre en charge un nouveau terminal sans mémoire d'ent alors qu'il existe déjà un interpréteur de ce type, sur le site, le module de gestion de la liste d'affichage peut être commun aux deux interpréteurs.
La différence portera sur le processeur graphique logiciel, qui est écrit en Fortran, mais dépend du type de matériel, et sur les modules de traitement du dialogue.

VII-3-6 - Portabilité de l'accompagnateur

L'accompagnateur pour console alphanumérique est écrit en Fortran. Il utilise la commande standard "PRINT" pour l'écriture de tous les messages (ceci évite de spécifier un numéro pour l'unité de sortie, numéro qui varie d'un système à l'autre).
L'accompagnateur est portable au même titre que le noyau : il faut lui fournir les sous-programmes de traitement des chaînes de caractères.
En dehors de cette question, la portabilité de l'accompagnateur ne semble pas poser de problèmes.

VII-4- OPTIMISATION

Le problème actuel, par soucis de portabilité et d'indépendance est écrit en Fortran "standard".
De plus, tous les cas d'erreurs sont prévus : les diverses tables sont contrôlé pour empêcher le débordement.
Enfin, le logiciel est écrit en utilisant uniquement des instructions "simples" de Fortran. Il ne contient aucune "astuce" de programmation susceptible de gagn du temps ou de la place, mais qui ferait échec à la portabilité.

VII-4-1 - La dimension des tables

Dans le prototype, les tables sont relativement importantes :

dimension de la table des fenêtres = 16
dimension de la table des clôtures = 16
dimension de la table des figures contenant des points = 32
dimension de la table des figures contenant du texte = 32
dimension de la table des modes déclarés pour les points = 256
dimension de la table des modes déclarés pour les textes = 64

De même, dans l'interpréteur on trouve :

dimension de la table des sections affichées = 256
dimension de la table des messages = 64
dimension de la table des zones réservées = 64

Toutes ces valeurs sont indiquées lors de l'initialisation, elles peuvent donc être modifiées facilement. Les tests de débordement inclus dans le logiciel seror encore valables puisque les dimensions ne sont pas des valeurs numériques, mais des paramètres.

Dans l'interpréteur pour terminal à tube mémoire, la dimension de la "mémoire" simulée est également précisée à l'initialisation. Dans le prototype, nous avons choisi 4096 entiers de longueur 2 (8 K octets) cette valeur peut également être modifiée.

VII-4-2 - Le type des variables utilisées

Dans le prototype, toutes les variables simples et les tableaux sont des variables de longueur standard (ici 4 octets). Il est évident que l'on pourrait stocker les diverses tables dans des variables beaucoup plus petites (1 ou 2 octets). Par exemple, chaque entrée de la table des modes occupe 40 octets (10 entiers de longueur 4) ; sa dimension est de 256, elle occupe donc 10 K octets.
Sans perdre beaucoup en ce qui concerne la portabilité, on pourrait la limiter à 64 entrées et coder chaque valeur sur un entier de longueur 2. La taille serai alors réduite à 1,25 K octets (divisée par 8).

Il en est de même pour les fenêtres, les clôtures, les messages, les figures etc. Sans perdre beaucoup de généralité , la taille des structures de données pourrait être divisée par 8.

VII-4-3 - Suppression de certaines séquences de traitement

Pour diminuer la taille du logiciel, on pourrait envisager de supprimer certains traitements. Notamment, tous les contrôles de débordement de tableaux pourraient être supprimés (aux risques et périls des utilisateurs).

On pourrait admettre également que les tables ne sont pas retassées ce qui supprimerait les trois sous-programmes de retassement. Certains aménagements sont possibles : par exemple, si l'écran est toujours utilisé dans sa totalité, la table des clôtures est inutile. De même si les sections de texte ne sont pas utilisées, la table des figures contenant du texte pourrait être réduite à une seule entrée (on ne peut pas la supprimer complètement, car sa présence est nécessaire), la table des modes pour les textes pourrait être supprimée et dans l'interpréteur, le générateur de caractères (qui est relativement lourd) deviendrait inutile. Nous signalons cette éventualité, car beaucoup d'utilisateurs ne se servent pas de la possibilité d'écrire du texte de taille et d'orientation quelconques. Enfin, notons que la gestion des modes est relativement lourde à cause de la convention adoptée pour la portée des paramètres (voir chapitre IV). Notamment, le cas où le paramètre est négatif (traitement de toutes les sections, ou toutes les figures, ou toutes les corrélations, sauf celle qui est mentionnée en négati pose beaucoup de problèmes. On pourrait envisager une convention plus simple :

- paramètre > 0 : traitement de l'élément concerné,
- paramètre = 0 : traitement de tous les éléments.

Cette convention, relativement souple, serait plus simple à traiter et permettra d'alléger beaucoup la gestion des modes.

VII-4-4 - Ecriture de certains modules en langage machine

Il est bien évident que l'écriture de certains sous-programmes serait beaucoup plus intéressante en langage machine. Ceci permettrait d'optimiser la place en mémoire et le temps d'exécution.

En particulier, les sous-programmes de traitement des chaînes ne sont portables qu'à condition que la machine possède des mots de 4 octets. La réécriture étant nécessaire dans certains cas, il semble qu'il soit alors plus intéressant de les écrire directement dans le langage de la machine.

VII-5 - LA MODULARITE

Les différents sous-programmes du logiciel sont relativement indépenda les uns des autres.
En fait, si l'on analyse le logiciel, pour prévoir par exemple une possibilité d'overlay, on constate que l'on trouve une "branche" par primitive dans le noyau

VII-5-1 - Le noyau

Dans le noyau, les différentes branches sont les suivantes :

- FENETR, $INITI, $INDEX, $ERR
- CLOTUR, $INITI,$INDEX, $ERR
- POINTS, $INITI, $INDEX, $ADR, $ERR
- TEXTES, $INITI, $INDEX, $ADR, $ERR
- MODE, $INITI, $NEXTC, $MODG, $MODF, $MODC, $TASS, $TASS2, $TASS3, $TRI, $ERR, $INDEX
- AFFICH, $INITI, $INDEX, $LIST, $VAL, $FS, $ATT, $PT, $CLIP, $CODE, $SWAP, $SWAPI, $GETCH, $ERR, $ABS
- EFFACE, $INITI, $INDEX, $FS, $LIST, $VAL, $ERR
- MESSAG, $INITI, $INDEX, $DECID, $NEXTC, $ECRI, $ERR
- EFFMES, $INITI
- EDENTI, $INITI, $CODCH, $GETCH, $ECRI
- EDRELL, $INITI, $CODCH, $GETCH, $ECRI
- EDCHAI, $INITI, $CODCH, $GETCH, $WEXTC, $ECRI
- NVAL, $INITI
- RVAL, $INITI
- CVAL, $INITI, $GETCH
- MENU, $INITI, $NEXTC, $DECID, $REPC, MESSAG, $ERR
- POSIT, $INITI, $INDEX, $VAL, $FS, $INV, $ERR
- IDENT, $INITI, $INDEX, $ERR

On constate que l'on peut diviser le noyau en autant de modules indépendants que de primitives, à condition d'isoler un module, qui pourrait être résident en mémoire, qui contient tous les sous-programmes de services.

Module de service :

$INITI	initialisation
$ERR	traitement des erreurs
$INDEX	traitement des chaînes
$NEXTC	
$ECRI	
$GETCH	

VII - 5-2 - L'interpréteur

Il est parfaitement modulaire. Chaque sous-programme de traitement d'une commande intermédiaire est indépendant.
Il suffit, là encore de constituer un module de service, qui comprendra notamment le processeur graphique logiciel et le module de mise à jour.

VII-6 - UTILISATION POSSIBLE DU PROTOTYPE

Le noyau est portable, à condition d'examiner deux points :

- traitement des manipulations d'adresses
- traitement des manipulations de caractères

De plus, on peut diminuer la taille des tables et supprimer certains traitements
En ce qui concerne l'interpréteur, nous distinguons deux parties :

Le module de gestion de la liste d'affichage est portable, au même titre que le noyau, de plus il est indépendant du terminal à mémoire utilisé.

Le processeur graphique et les modèles de gestion du dialogue sont portables, ma ils dépendent du type du terminal.

Enfin le logiciel élémentaire dépend à la fois du terminal et du calculateur.

La figure VII-9 résume la situation.

Signalons que l'ensemble noyau+accompagnateur console est facilement portable, avec les mêmes remarques que celles concernant le noyau seul.

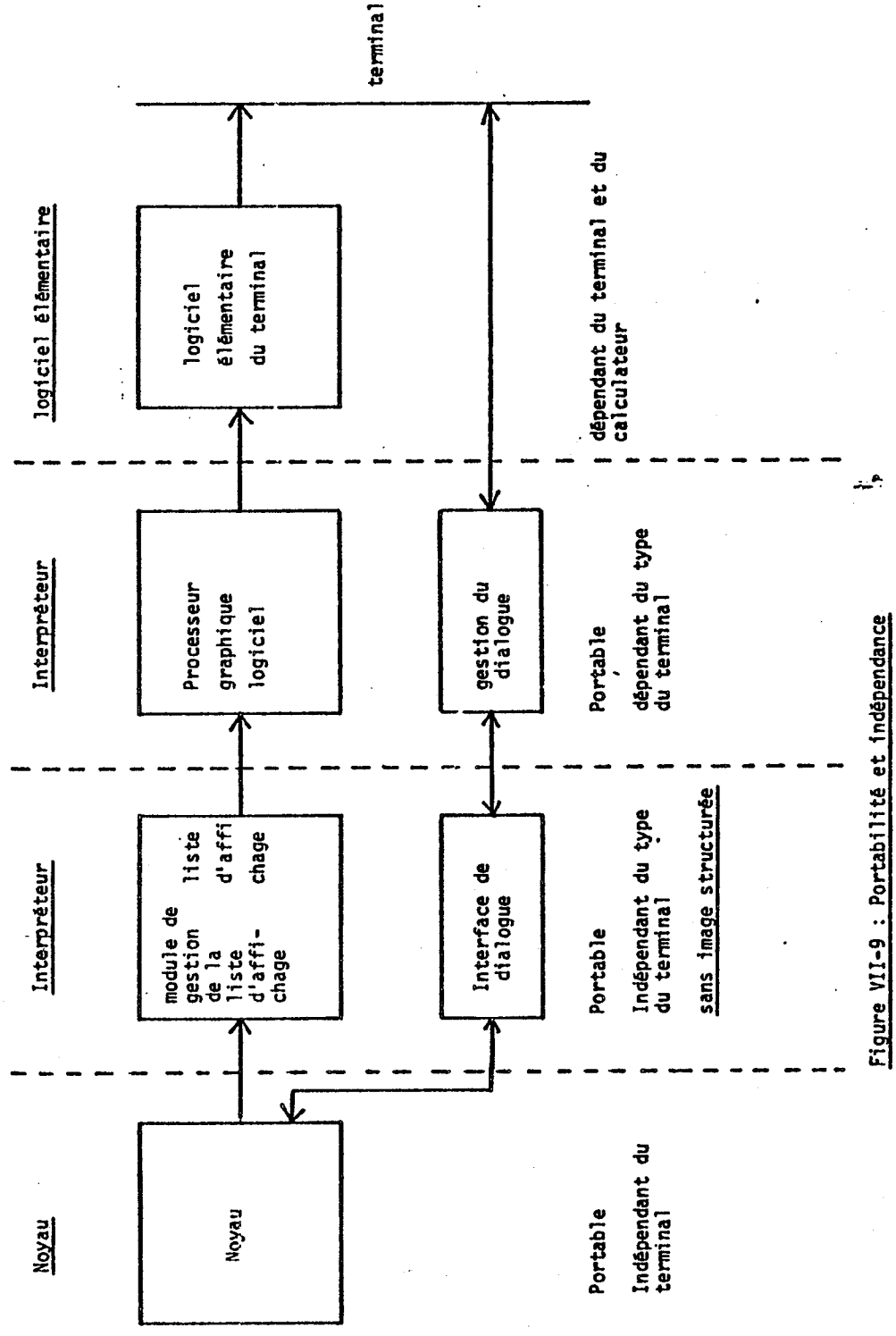


Figure VII-9 : Portabilité et indépendance

CONCLUSION

I - ETAT DE LA REALISATION ACTUELLE

Le prototype réalisé à l'heure actuelle prend en charge les différents matériels disponibles au Centre Interuniversitaire de Calcul de Grenoble. Les différents contextes d'utilisation sont schématisés ci-dessous.

Calculateur	*Système d'exploitation*	*Langages*	*Interpréteurs*	*Dispositifs de dialogu*
IBM 360/67	CP/CMS *(partage de temps)* *(mémoire virtuelle)*	ALGOL W FORTRAN assembleur PL/1	*(mémoire structurée)*	Clavier alphanumérique tablette SYLVANIA photostyle clavier de fonctions
IRIS 80	SIRIS 8 *(partage de temps)*	FORTRAN	*(tube mémoire)* TEKTRONIX 4010 TEKTRONIX 4012 TEKTRONIX 4015 TEKTRONIX 4051 ACCOMPAGNATEUR *(IBM 2741, TTY)* copie traceur de courbe ANDERSON/JACOBSON	 clavier alphanumérique réticule clavier ordinaire clavier ordinaire

Nous souhaitons mettre en valeur le rôle de l'accompagnateur. Cet interpréteur particulier est destiné à faciliter la mise au point des programmes d'application en fournissant une trace des différents appels aux fonctions graphiques et de dialogue. Toutes les sorties se font sur un terminal ordinaire (IBM 2741, TELETYPE, SYNELEC etc...) ce qui permet de ne pas monopoliser une console de

visualisation simplement pour vérifier que les différentes phrases de dialogue s'enchaînent bien, ou que l'interface est bien initialisé. De plus, lors de l'im plantation de GRIGRI sur de nouvelles machines, cet accompagnateur permet de vérifier que le noyau du logiciel est mis en place correctement, permettant ains de séparer la mise au point du noyau du système et la mise au point des différents interpréteurs. Cet accompagnateur s'est révélé un instrument très précieux facilitant grandement la recherche d'erreurs.
Signalons que la mise en place sur un SOLAR, avec les terminaux TEKTRONIX 4015 et 4051 est en cours de réalisation.

II - LES DÉVELOPPEMENTS POSSIBLES

L'exploitation du prototype peut se faire dans plusieurs directions :

- Il serait souhaitable de prendre en charge de nouveaux terminaux de type diver afin de confirmer l'indépendance du logiciel par rapport au matériel. Notammen il faudrait exploiter les idées mentionnées au chapitre VI, pour pouvoir prend en charge un terminal à balayage ligne par ligne et à mémoire de points.

- Une optimisation du prototype est envisageable, mais il faut prendre garde à n pas perdre la notion de portabilité.

- En ce qui concerne les primitives offertes dans le logiciel, il est clair que la fonction MODE est la porte ouverte à toutes les options possibles : il suff d'ajouter des paramètres dans le descripteur du mode et de prévoir les program mes de traitement correspondant.

- Cependant, il semble qu'il faille envisager un type de donnée supplémentaire. Le logiciel traite actuellement deux types de sections :

  - les sections de points,
  - les sections de textes.

  Il serait utile qu'il traite également les sections de "taches". Le schéma gén reste le même, il faut prévoir une primitive "tache" :

  TACHES(nfig, tlong, description)

le paramètre description contient la description des différentes taches (liste des sommets, par exemple).
Le tableau tlong permet de regrouper les taches dans différentes sections.
L'affichage est toujours réalisé par la primitive AFFICHER.
La primitive MODE doit prévoir des paramètres de description du mode "tache", tels que :
- couleur,
- intensité.

Ce type de donnée supplémentaire sera indispensable pour bien exploiter les terminaux à mémoire de points.

- une généralisation des primitives de dialogue serait souhaitable en étudiant le moyens de réaliser des combinaisons simples entre les actions élémentaires, et surtout leur enchaînement.

- Il serait souhaitable également de réaliser une meilleure symétrie entre les primitives d'affichage et les primitives de dialogue. Par exemple, actuellement GRIGRI ne fournit pas de primitive d'introduction de textes équivalente à la primitive de collecte de coordonnées.

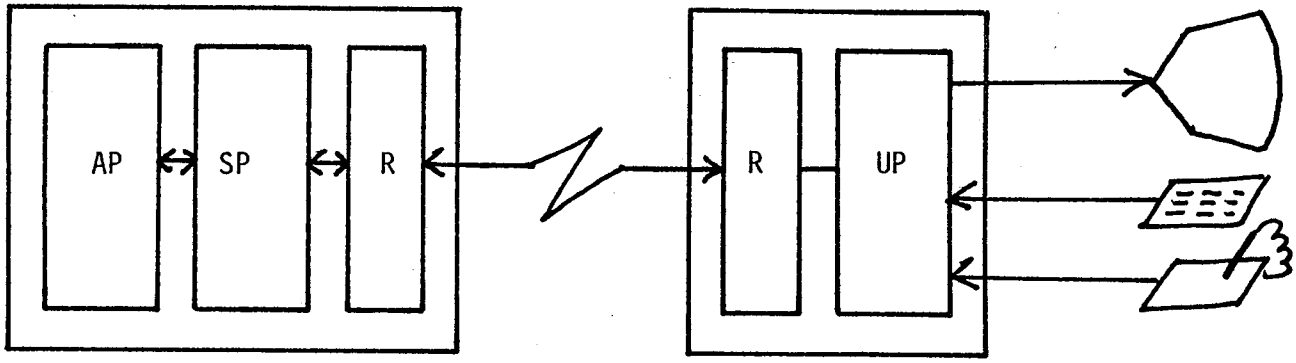
III - L'UTILISATION DE GRIGRI DANS UN RESEAU

Actuellement, un groupe de travail étudie les possibilités et les probl de l'utilisation des graphiques à travers le réseau CYCLADES.
Au cours de cette étude, nous avons abordé divers sujets, notamment :

- indépendance du logiciel par rapport au terminal,
- indépendance du logiciel par rapport au calculateur,

et nous avons souligné l'utilisation intéressante d'un logiciel réparti.

En effet, dans un réseau la situation est la suivante : un gros calculateur contient un logiciel graphique indépendant des terminaux ; un terminal graphique accède aux services du logiciel par l'intermédiaire du réseau et d'un mini-calculateur (Figure C.1)



Calculateur hôte du logiciel réseau calculateur pilotant le terminal

Figure C1 : Configuration réseau

AP ; programme d'application
SP : logiciel serveur
R : protocoles réseau
UP : logiciel utilisateur

Notons que cette configuration est également envisageable sans réseau, car il est prouvé que l'exploitation maximale d'une console fortement interactive est obtenue, non pas par connexion directe à un gros calculateur (partage de temps, etc...) mais par connexion à un calculateur satellite.

Dans cette configuration, le logiciel graphique est coupé en deux (SP et UP) et le logiciel serveur (SP) doit être indépendant du terminal, ainsi que le code intermédiaire engendré.

Nous signalerons deux possibilités concernant GRIGRI.
Tout d'abord, en dehors du contexte réseau, si on doit prendre en charge un terminal connecté à un calculateur satellite, la structure de GRIGRI est très b adaptée :

- le noyau sera placé dans le gros calculateur,
- l'interpréteur sera placé dans le satellite,

ainsi, en particulier, la gestion de l'image pourra être assurée au niveau loca (si le calculateur est trop petit, on peut très bien stocker la liste d'afficha sur disque) et le dialogue se déroulera entièrement au niveau local ; c'est

seulement à la fin de la fonction que le satellite transmettra les données collectées (ceci est fondamental pour assurer un dialogue efficace et rapide).

La seconde remarque concerne l'utilisation de GRIGRI dans le réseau CYCLADES. Quand le protocole graphique sera définitivement figé, il serait étonnar que le code indépendant transitant dans le réseau soit le même que le code intermédiaire de GRIGRI. Donc, pour que GRIGRI puisse être utilisé à distance, il devra engendrer le code du réseau. Cela ne pose pas de problèmes particuliers : le réseau sera considéré comme un terminal graphique particulier ; il suffira d'écrir l'interpréteur correspondant. Cet interpréteur sera en fait un traducteur permettant de passer du code intermédiaire de GRIGRI au code du réseau (Figure C.2).

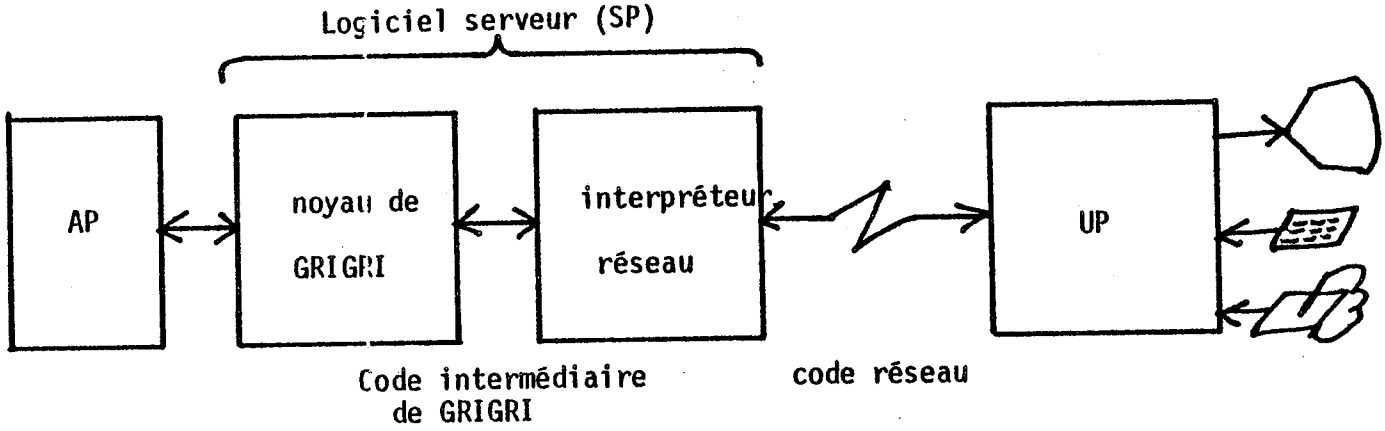


Figure C2 : Utilisation de GRIGRI à distance

Inversement, GRIGRI pourrait servir pour piloter un terminal utilisant un logicie graphique distant. Dans ce cas, il faut prévoir un traducteur permettant de passe du code réseau au code intermédiaire de GRIGRI, qui sera récupéré par l'interpréteur (Figure C.3)

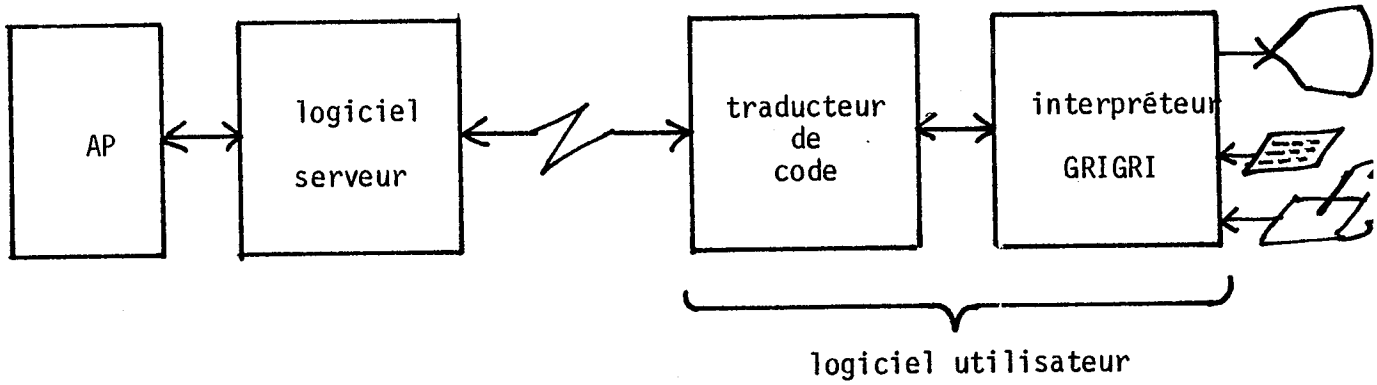


Figure C3 : Utilisation d'un logiciel distant par un interpréteur GRIGRI

On peut penser que, au moins en ce qui concerne l'affichage, les codes intermédiaires de GRIGRI et du réseau seront proches et que les traducteurs seront relativement faciles à réaliser.

IV - INDÉPENDANCE, PORTABILITE

La réalisation de GRIGRI a permis de prouver que l'indépendance par rapport au terminal était réalisable.
De même en ce qui concerne l'indépendance par rapport au reste du contexte. Mai il est certain que l'approche de la portabilité coûte cher : par exemple, pour traiter les chaînes de caractères de façon complètement indépendant du calculateur, il faut paramétrer tous les programmes de traitement en fonction du nombr de caractères par mot et du nombre de bits par caractères. Il est certain que ceci est réalisable (R2)mais alourdit particulièrement le logiciel. La solutio la plus sage est de fournir un logiciel "facilement portable" en mentionnant soigneusement les points délicats.

De plus, en dehors des problèmes liés au calculateur, la portabilité ne pourra jamais être réalisée sans la standardisation d'au moins un langage de programmation. Ce qui est loin d'être fait [G.12] !

De même, pour résoudre les problèmes d'indépendance par rapport au terminal, l'idée importante est qu'il n'existe que deux types d'interpréteurs, dont l'organisation d'ensemble est invariable.

Notons que l'indépendance par rapport à l'application semble réalisée. Des applications de type très divers ont déjà utilisé GRIGRI à Grenoble :

- analyse numérique
- architecture (perspective-élimination des parties cachées)
- etc...

Pour chaque type d'application, l'équipe graphique de Grenoble fournit un logiciel de mise en forme adapté :

- visualisation de matrices
- visualisation de graphes
- algorithmes d'élimination des parties cachées
- projections -perspectives
- etc...

Par ce biais, l'indépendance par rapport à l'application est approchée au mieux.

Finalement on constate que l'indépendance complète par rapport à l'ensemble du contexte est une utopie. Il est par contre fondamental de fournir un logiciel "facilement adaptable et facilement portable".

V - LA STANDARDISATION DES LOGICIELS GRAPHIQUES

La leçon que l'on peut tirer, même dans l'état actuel des choses, est que des logiciels du type de GRIGRI se répandront, du fait même des qualités qu'ils offrent aux programmeurs : simplicité, cohérence, indépendance vis à vis du contexte [N1], [N2], [N3], [N4]. Cependant, les principes de base de la conception de GRIGRI étant fondamentalement différents de ceux qui ont inspiré la conception des logiciels déjà distribués, une difficulté majeure est apparue lors des travaux qui ont pu être faits, soit dans le cadre de la normalisation de logiciels graphiques, soit dans le cadre de l'utilisation de consoles de visualisation dans un réseau : il y a incompatibilité entre ces différents logiciels. Par exemple, dans les logiciels existant, les objets sont construits en raisonnant en terme de déplacements du faisceau allumé ou éteint ; par contre, dans GRIGRI, la construction est totalement disjointe du mode de visualisation.

De même en ce qui concerne les primitives de dialogue : s'il est toujours possi d'affecter arbitrairement un dispositif de communication à la réalisation d'une demande fonctionnelle, l'inverse est impossible : lors de l'invocation du photostyle, il est impossible de savoir s'il doit servir à une identification, une collecte de coordonnées, ou un menu. Ceci signifie que l'on verra probablem se développer des familles de logiciels fondées sur des principes contradictoir de la même manière que l'on a développé des familles de langages.

La conception et la réalisation de GRIGRI ont contribué à la définition d'une telle famille.

ANNEXES

ANNEXE 1 Les primitives de GRIGRI

ANNEXE 2 Exemple de programme d'application

ANNEXE 3 Exemple de trace fournie par l'accompagnateur

ANNEXE 4 Exemple des possibilités de GRIGRI

ANNEXE 1

LES PRIMITIVES DE GRIGRI

I - PRIMITIVES DE DESSIN

1. Définition d'espaces de dessin

FENETRE (nfen, igx, igy, sdx, sdy)
CLOTURE (nclot, igx, igy, sdx, sdy)

2. Déclaration de donnée

POINTS (nfig, tablong, tabx, taby, tablegende)
TEXTES (nfig, tablong, tabtextes, tabx, taby)

3. Mode de représentation

MODE (nfig, ncorrel, nsection, descripteur)

4. Affichage

AFFICHER (nfig, ncorrel, nfen, nclot)
EFFACER (nfig, ncorrel)

II - GESTION DES MESSAGES

1. Description d'un message

MESSAGE (nmess, descripteur, nclot, x,y)

2. Edition de données alphanumériques

EDENTIER (nmess, nzone, entier)
EDREEL (nmess, nzone, réel)
EDCHAINE (nmess, nzone, chaîne)

3. Effacement d'un message

EFFMESSAGE (nmess)

III - TRAITEMENT DU DIALOGUE

1. Introduction de données alphanumériques

RVAL (nb, nmess, tréel)
NVAL (nb, nmess, tentier)
CVAL (nb, nmess, tchaîne)

2. Collecte de coordonnées

POSITION (nb, nfig, nfen, nclot)

3. Identification

IDENTIFIER (nb, nfig, tfig, tcorrel)

4. Utilisation d'un menu

MENU (nb, liste, tretour)

Les valeurs des identificateurs soulignés ont la signification suivante :

	nfig toutes figures	*ncorrel* toutes sections	*nsection* toutes sections	*nmess* tous message
positif	la figure *nfig*	toutes sections corrélées *ncorrel*	la section *nsection*	le message *nmess*
négatif	toutes figures sauf *nfig*	toutes sections sauf celles corrélées *ncorrel*	toutes sections sauf *section*	pas de sens

ANNEXE 2

EXEMPLE DE PROGRAMME D'APPLICATION

```
      REAL A,B,Z,TABX(100), TABY(100)
      REAL D, COSA,COSB,COSC,D1,XPLAN(100),YPLAN(100)
      INTEGER N
      INTEGER IMCDE (19,2),IMODE1(19,1)

C DECLARATION DE CLOTURE
      CALL CLOTUR (2,0,0,100,85)

C DECLARATION DE FENETRE
      CALL FENETR (1,-3.,-3.,3.,3.)

C AFFICHAGE DE 5 MESSAGES
      CALL MESSAG (3,'NOMBRE D ELLIPSES &4&&&',0,0,100)
      CALL MESSAG (1,'NB DE POINTS PAR ELLIPSE &4&&&',0,0,95)
      CALL MESSAG (2,'XOEIL &8& YOEIL & &8ZOEIL &8&&&',0,0,90)
      CALL MESSAG (10,'MODE DES ELLIPSES &11&&&',0,0,85)
      CALL MESSAG (11,'MODE DU TEXTE &15,16&&&',0,0,80)

C ENTREE D'UN MODE PAR CVAL
1     NB = 1
      CALL CVAL (NB,10,IMODE1)
      CALL MODE (0,10,0,IMODE1)

C DECLARATION DE TEXTES
      INTEGER BLABLA(5)
      REAL XXX(5),YYY(5)
      INTEGER TTT(5)
      CALL TEXTES (2,TTT,BLABLA,XXX,YYY)
C DEUX SECTIONS
      TIT(1) = 6
      TIT(2) = 10
      TIT(3) = 0
      DATA BLABLA/''TRACE D''ELIPSES '/
      XXX(1) = 0.
      YYY(1) = 0.
      XXX(2) = 1.
      YYY(2) = 2.
      NB = 2
C ENTREE DU MODE DES TEXTES PAR CVAL
      NB = 1
      CALL CVAL (NB,11,IMODE)
      CALL MODE (1,1111,1,IMODE(1,1))
      CALL MODE (2,2222,2,IMODE(1,2))
C AFFICHAGE DES TEXTES
      CALL AFFICH (2,0,2,2)
```

```
      NELL = 10
      N = 10
      XOEIL = 150.
      YOEIL = 100.
      ZOEIL = 30.

C ENTREE D'UN ENTIER
      NB = 1
      CALL NVAL (NB,3,NELL)
      NELL = NELL +1

C ENTREE D'UN ENTIER
      NB = 1
      CALL NVAL (NB,1,N)
      NN = N+1

C ENTREE DE 3 REELS
      REAL OEIL(3)
      NB = 3
      CALL RVAL (NB,2,OEIL)
      XOEIL = OEIL(1)
      YOEIL = OEIL(2)
      ZOEIL = OEIL(3)

C EDITION ALPHANUMERIQUE
      CALL EDENTI(1,1,NEG)
      CALL EDENTI (3,1,NELL)
      CALL EDENTI (1,1,N)
      CALL EDREEL (2,1,3,XOEIL)
      CALL EDREEL (2,2,-3,YOEIL)
      CALL EDRELL (2,3,4,ZOEIL)
      C1 = SQRT (XOEIL**2+YOEIL**2+ZOEIL**2)
      COSA = XOEIL/D1
      COSB = YOEIL/D1
      COSC = ZOEIL/D1
      D = 1.3 * D1

C DECLARATION DE SUITE DE POINTS
      INTEGER TL(5),TLEG(15)
      CALL POINTS (1,TL,XPLAN,YPLAN,TLEG)
      DATA TLEG/'AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJ'/

C CALCUL
      KK = 0
      DO 8 I=1,NELL
      JJJ = 10 + I
      CALL MODE (1,JJJ,I,IMODE1)
      Z = -2.+3.*(I-1)/(NELL-1)
         IF ((Z.EQ.-1.).OR.(Z.EQ.0.)) GOTO 2
              A=ABS(Z)
              B=ABS(1+Z)
              CALL ELLIPS (A,B,TABX,TABY,N)
              GOTO 6
```

```
2          IF (Z.EQ.-1.) GOTO 4
                DO 3 K=1,NN
                TABX(K)=O.
3               TABY(K)=-2.+4.*(K-1)/N
                GOTO 6
4          DO 5 K=1,NN
           TABX(K)=-1.+4.*(K-1)/N
5          TABY(K)=O.
6          NN=N+1
           DO 7 J=1,NN
7          CALL PROJEC. (TABX(J),TABY(J),Z,XOEIL,YOEIL,ZOEIL,
       *    D,COSA,COSB,COSC,XPLAN(J+KK),YPALN(J+KK)
           KK = KK + NN
           TL(I) = NN
           TL(I+1) = O

C AFFICHAGE DE LA FIGURE 1 QUI CONTIENT UNE SEULE SECTION
8      CALL AFFICH (1,JJJ,2,2)

C IDENTIFICATION
       NB = 5
       INTEGER TFIG(5),TCOR(5)
       CALL IDENT (NB,O,TFIG,TCOR)


C EFFACEMENT DE L'ELEMENT DESIGNE
       CALL EFFACE (TFIG(1), TCOR(1))

C COLLECTE DE COORDONNEES
       NB = 1O
       CALL POSIT (NB,1,2,2)

C DEFINITION D'UN MODE
       CALL MODE (1,4444,4,'T3M* &&')

C AFFICHAGE DE LA SECTION ENTREE
       CALL AFFICH (1,4444,2,2)

C MENU
       INTEGER NMENU(19),TCI(5)
       DATA NMENU/'1 BOUCLE,27 BENSON,28 STOP &&'/
9      NB = 2
       CALL MENU (NB,NMENU,TCI)
C      IF (TCI(2).EQ.27) CALL DJNN(15,O)
       IF (TCI(1).NE.1) GOTO 1O

C EFFACEMENT DE TOUTES LES SECTIONS
       CALL EFFACE (O,O)
       GOTO 1

C EFF. DE TOUTES LES SECTIONS ET DE TOUS LES MESSAGES
1O     CALL EFFACE (O,O)
       CALL EFFMES (O)
       STOP
       END
```

ANNEXE 3

EXEMPLE DE TRACE FOURNIE PAR L'ACCOMPAGNATEUR

Nous présentons ici la trace du programme donné en Annexe 2

```
EXECUTION BEGINS...

*** ACCOMPAGNATEUR CONSOLE A VOTRE SERVICE ***

   EQUIPE GRAPHIQUE, TEL. 269,220 BUREAUX B113,B115

TRACES AVEC OPTIONS : TAPER 1
TRACES ELEMENTAIRES : TAPER 2
1

INTERPREATION BEGINS ...
*** DECLARATION DE LA CLOTURE  2  BORNES (IG,SD):  0      0      100      85
*** DECLARATION DE LA FENETRE  2  BORNES (IG,SD): -3.00    -3.00     3. 3.
*** MESSAGE      3 LONGUEUR= 23

NOMBRE D'ELLIPSES
*** MESSAGE      1 LONGUEUR= 30
NB DE POINTS PAR ELLIPSE
*** MESSAGE      2 LONGUEUR= 49
XOEIL             YOEIL            ZOEIL
*** MESSAGE     10 LONGUEUR= 29
MODE DES ELLIPSES
*** MESSAGE     11 LONGUEUR= 45
MODE DU TEXTE
*** CVAL SUR MESSAGE  10 NB D'ENTREES  1
<-- CHAINE&&
t2r3&&
*** MODE DE POINTS ***
  FIGURE     0 CORRELATION    10 SECTION 0
DESCRIPTEUR: C* T2 E* I** R3 M     1000
*** CVAL SUR MESSAGE 11 NB D'ENTREES   2
<-- CHAINE&&
g2o45&&
g4o90&&
```

```
*** MODE DE POINTS ***
  FIGURE     2 CORRELATION 1111 SECTION      1

DESCRIPTEUR: C* T2 E* I** R3 M    ****

*** MODE DE POINTS ***
  FIGURE     2 CORRELATION 2222 SECTION      2

DESCRIPTEUR: C* T2 E* I** R3 M    ****

*** AFFICHER (ATTRIBUTS SUITE DE CARACTERES)
  FIGURE    2 SECTION    1 CORRELATION  1111
CLOTURE + FENETRE + MODE ?
oui

  CLOTURE     0     0  100     85
  DESCRIPTEUR :  G2 0 45 C*  E* I**
  FENETRE    -3.00  -3.00     3.00    3.00

*** AFFICHE     6 CARACTERES DE LA SECTION    1

  CHAINE=TRACE

*** AFFICHER (ATTRIBUTS SUITE DE CARACTERES)
  FIGURE    2 SECTION    2 CORRELATION  2222
CLOTURE + FENETRE + MODE ?
non

*** AFFICHE     10 CARACTERES DE LA SECTION    2

  CHAINE=D'ELLIPSES

*** NVAL SUR MESSAGE     3 NB D'ENTREES 1
 <-- UN ENTIER PAR LIGNE

*** NVAL SUR MESSAGE     1 NB D'ENTREES 1
 <-- UN ENTIER PAR LIGNE
10

*** RVAL SUR MESSAGE     2 NB D'ENTREES 3
 <-- UN REEL PAR LIGNE
8
** RVAL **REPONSE INCORRECTE
5.0
15.0
50.0

*** EDENTIER: MESSAGE    3 ZONE    1 LONGUEUR    1 ENTIER=2

*** EDENTIER: MESSAGE    1 ZONE    1 LONGUEUR    2 ENTIER=10

*** EDREEL  : MESSAGE    2 ZONE    1 LONGUEUR    5 REEL=4.999

*** EDRELL  : MESSAGE    2 ZONE    2 LONGUEUR    8 REEL=0.149E02

*** EDRELL  :MESSAGE     2 ZONE    3 LONGUEUR    7 REEL=49.9999

***DECLARATION DE LA FIGURE  2

*** AFFICHER  (ATTRIBUTS DE POINTS)
  FIGURE      1 SECTION       1 CORRELATION  10
CLOTURE + FENETRE + MODE ?
oui
```

```
CLOTURE       0      0    100    85
   DESCRIPTEUR : C* T2 E* I** R3 M  ****
   FENETRE  -3.00 -3.00   3.00   3.00

*** AFFICHE   11 POINTS POUR LA FIGURE  1 SECTION   1
LISTE DES COORDONNEES ?
oui

   2.37   0.00   2.12   -0.71   1.03   -0.84   -0.46   -0.34   -1.78   0.59
LISTE DES COORDONNEES ?
non

*** AFFICHER (ATTRIBUTS DE POINTS)
  FIGURE     1  SECTION      1 CORRELATION  10
CLOTURE + FENETRE + MODE ?
non

*** AFFICHE   10 POINTS POUR LA FIGURE 1  SECTION  1
LISTE DES COORDONNEES ?
oui

   1.25   -0.80   0.72   -2.28   -0.86   -2.73   -1.30   -1.52   -1.23   0.14
LISTE DES COORDONNEES ?
non

*** IDENTIFIER   5 ENTREES SUR TOUTES LES FIGURES

COLLECTE= SECTION + CORRELATION OU 0 + MESSAGE:
SECTION=
1
CORRELATION=
2222
SECTION=
s

*** COLLECTE DE COORDONNEES SUR  10 POINTS
TRACE FENETRE + CLOTURE + MODE ?
non

<-- COORDONNEES

 X=
4.5
 Y=
9.3
 X=
s

*** MODE DE POINTS ***
  FIGURE     1 CORRELATION  4444 SECTION   4

DESCRIPTEUR: C* T3 E* I** R* M 6049872

*** MENU  NB D'ENTREES MAX=  3
  NUMEROS VALIDES PAR L'ANALYSE=
  1
 27
 28.
```

```
    <-- UN NUMERO PAR LIGNE
3
 ?
28
s
*** EFFACEMENT DE TOUTES LES SECTIONS DE POINTS
*** EFFACEMENT DE TOUS LES MESSAGES
```

ANNEXE 4

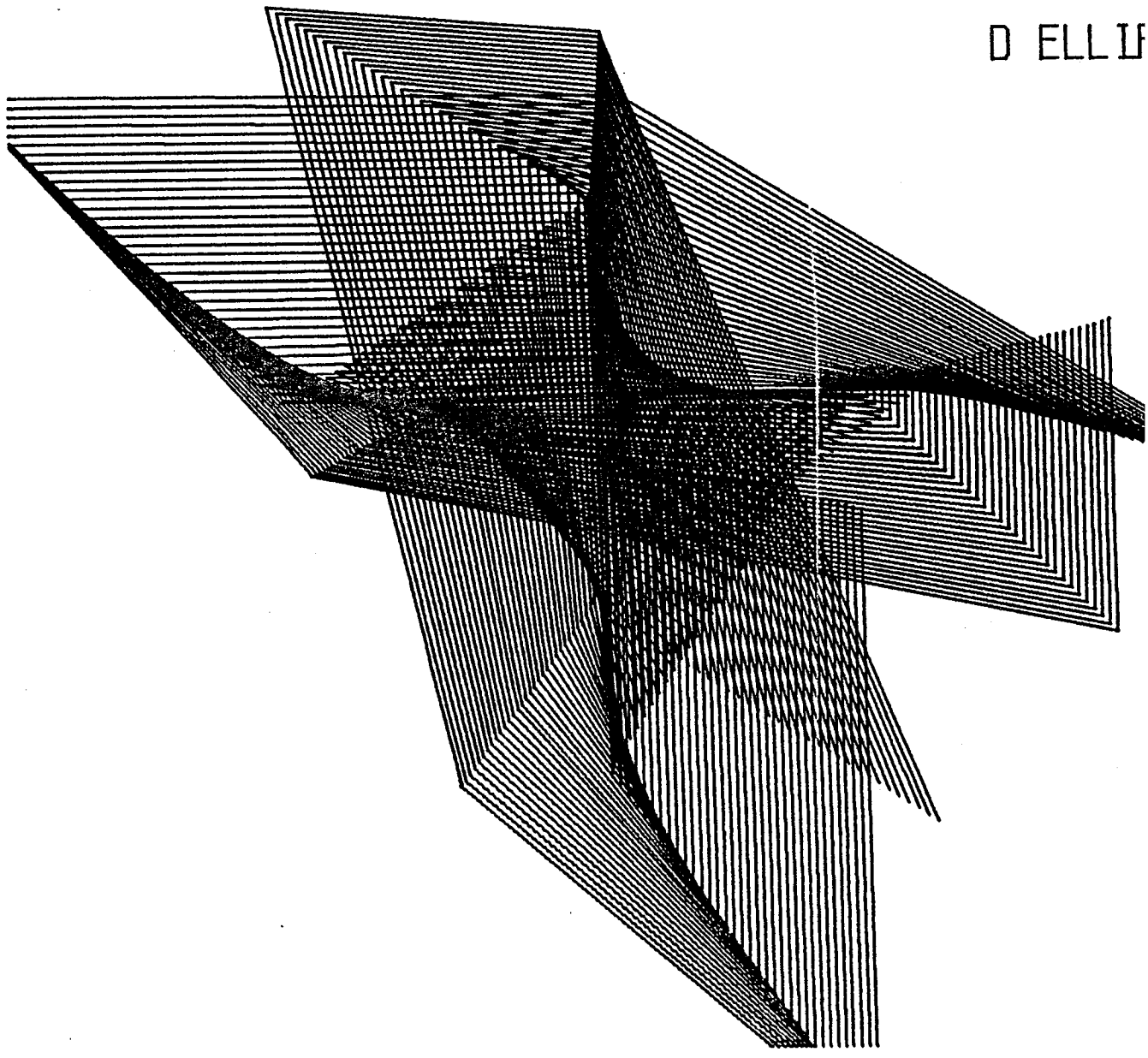
EXEMPLE DES POSSIBILITES DE GRIGRI

Les dessins suivants sont les résultats du programme contenu dans l'annexe 2.

Le mode de tracé est acquis par l'intermédiaire d'une entrée de chaîne. Ainsi par simple modification du mode on obtient les représentations suivantes :

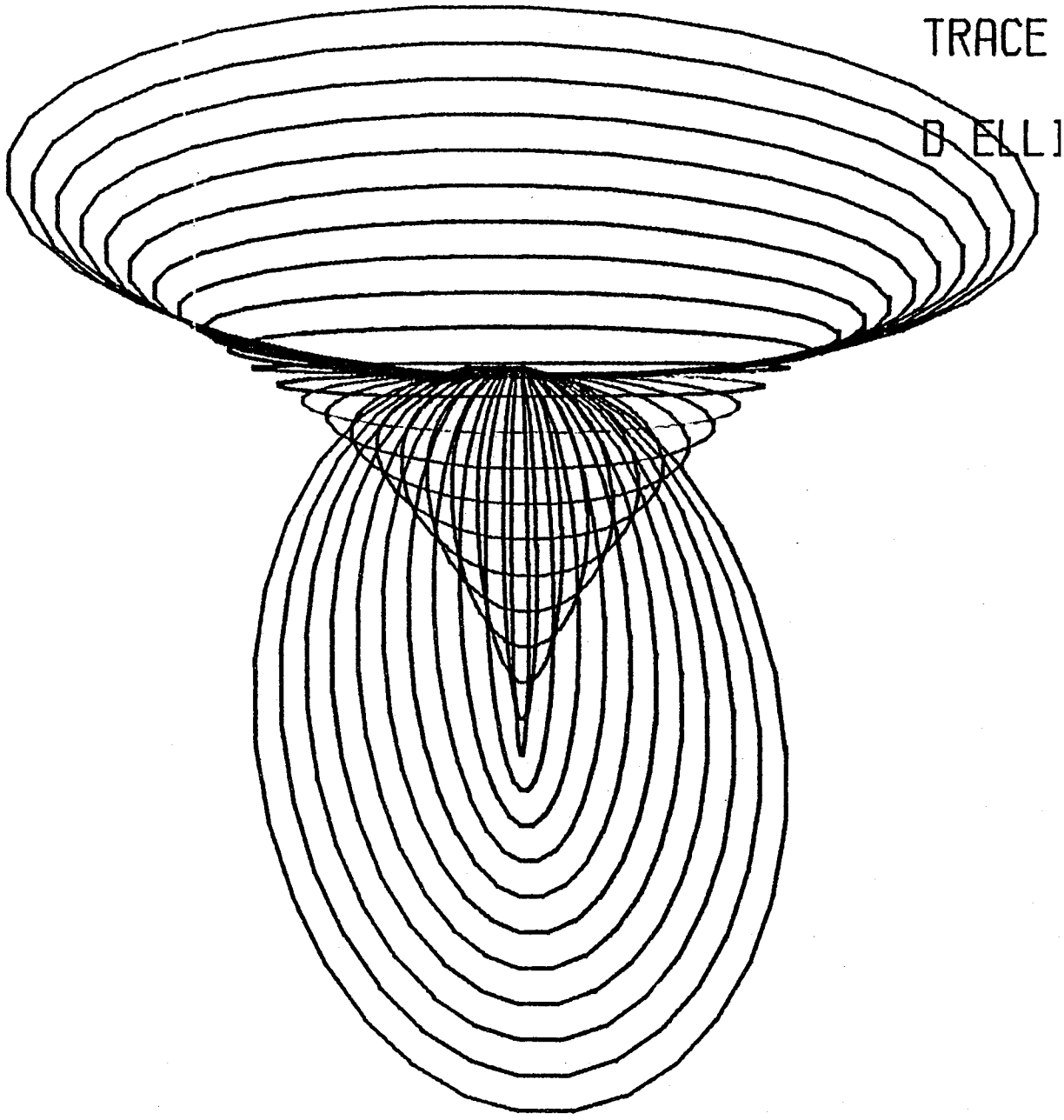
- ligne brisée, trait continu,
- " tireté court,
- " tireté long,
- " trait pointillé,
- " trait mixte,
- " avec marqueur indicé,
- tireté long et marqueur,
- nuage de points.

NOMBRE D ELLIPSES 101
NB DE POINTS PAR ELLIPSE 5
XOEIL -1.000 YOEIL 0.500E02 ZOEIL 200.0000
MODE DES ELLIPSES T1 &&
MODE DU TEXTE G4 && G4 &&



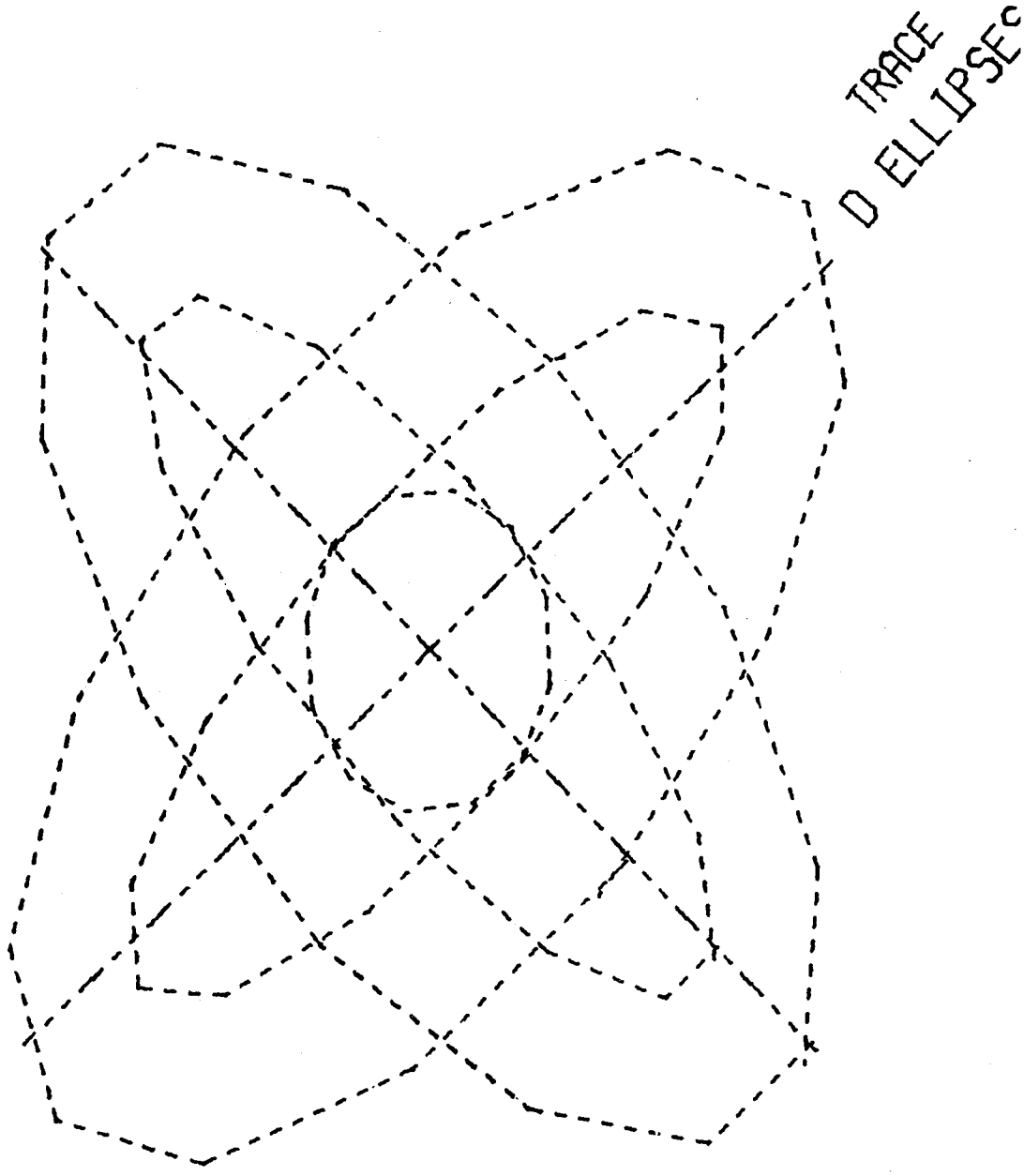
LEDUC LE 6494010011/17:31:51 DESSIN NO 001 ECHELLE: 07

NOMBRE D ELLIPSES 32
NB DE POINTS PAR ELLIPSE 30
XOEIL 0. YOEIL 0.500E02 ZOEIL 50.0000
MODE DES ELLIPSES T1 &&
MODE DU TEXTE G4 && G4 &&



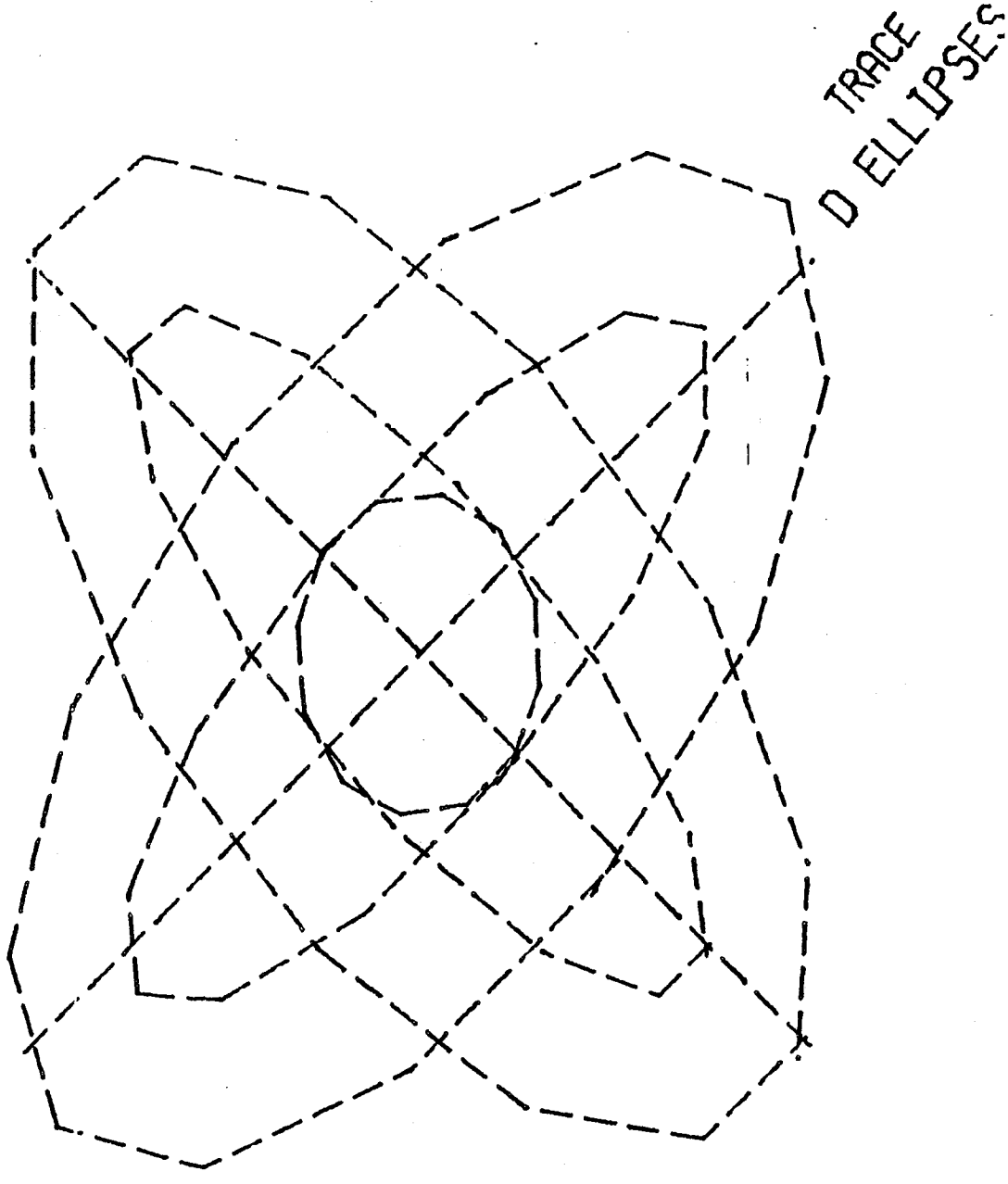
LEDUC LE 6494010012/15:37:06 DESSIN NO 011 ECHELLE: 0?

NOMBRE D ELLIPSES 7
NB DE POINTS PAR ELLIPSE 12
XOEIL 1.000 YOEIL 0.100E01 ZOEIL 100.0000
MODE DES ELLIPSES T2R1&&&
MODE DU TEXTE G3050&& G4050&&



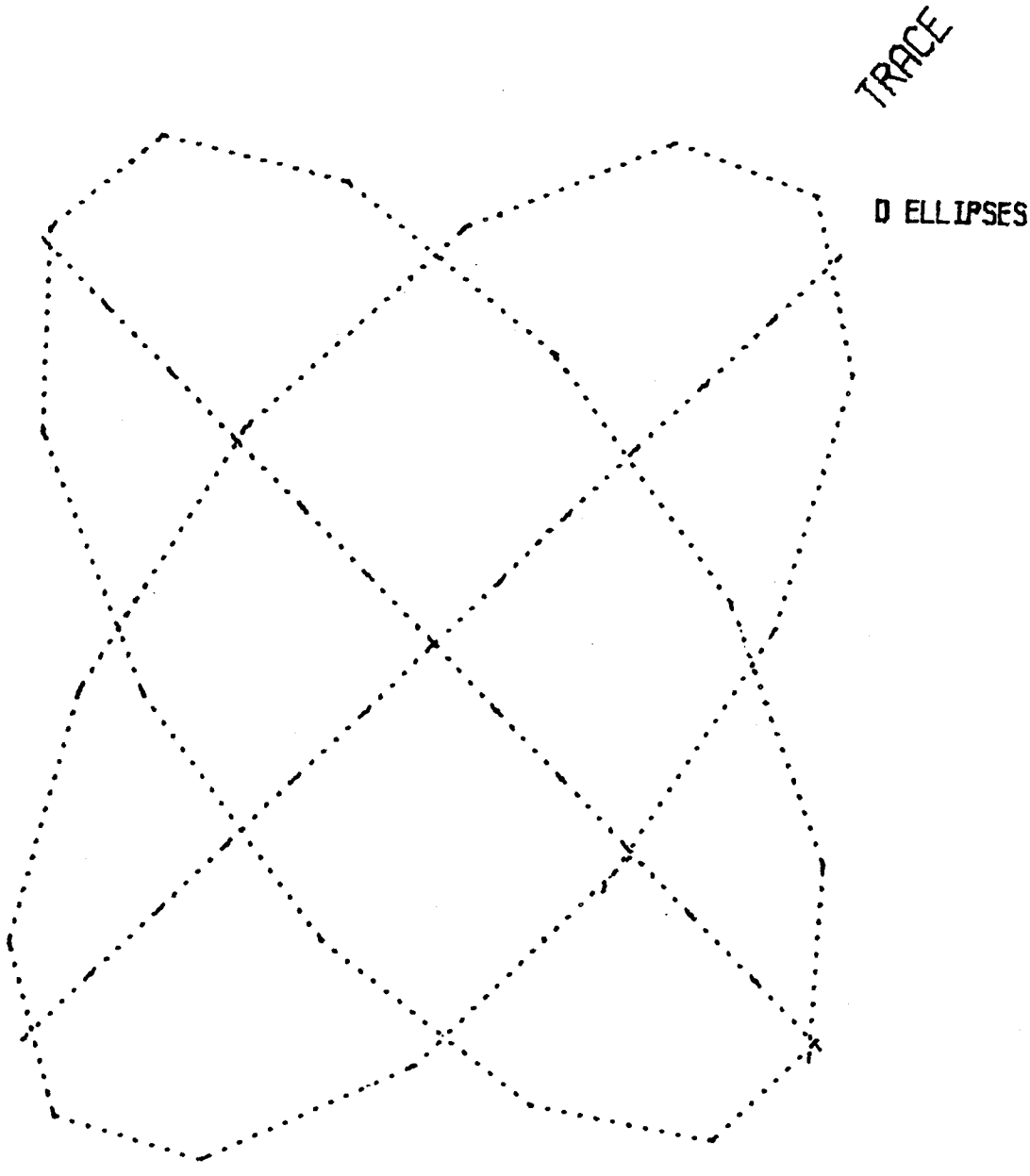
LEDUC LE 6494010012/16:23:32 DESSIN NO 014 ECHELLE: 07/

NOMBRE D ELLIPSES 7
NB DE POINTS PAR ELLIPSE 12
XOEIL 1.000 YOEIL 0.100E01 ZOEIL 100.0000
MODE DES ELLIPSES T3R1&&
MODE DU TEXTE G3C50&& G4C50&&



LEDUC LE 6494010012/16:24:22 DESSIN NO 015 ECHELLE: 07

NOMBRE D ELLIPSES 4
NB DE POINTS PAR ELLIPSE 12
XOEIL 1.000 YOEIL 0.100E01 ZOEIL 100.0000
MODE DES ELLIPSES T4R1&&
MODE DU TEXTE G3050&& G1000&&

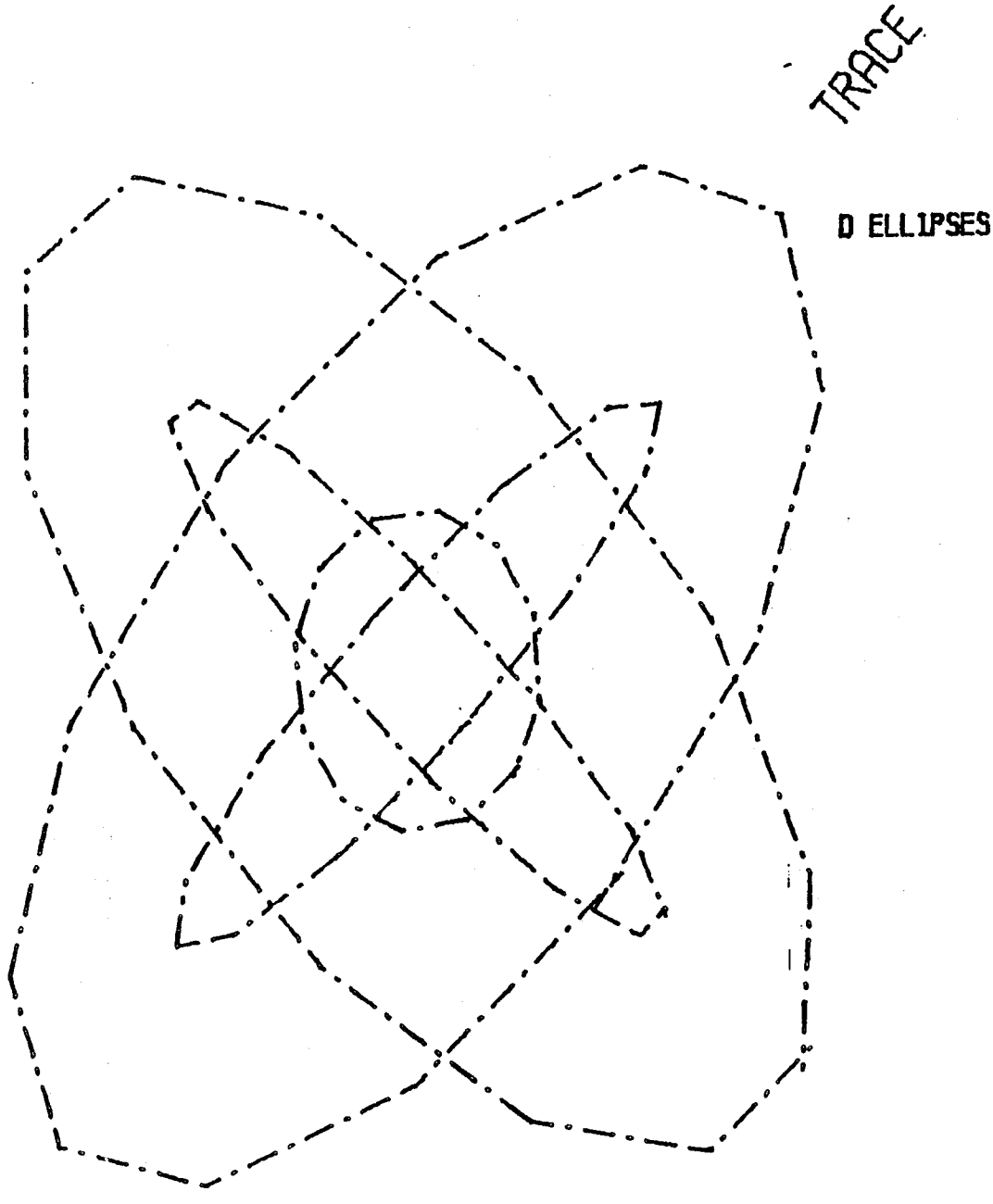


LEDUC LE 6494010012/16:25:45 DESSIN NO 016 ECHELLE: 07/1

E'MBRE D ELLIPSES 5

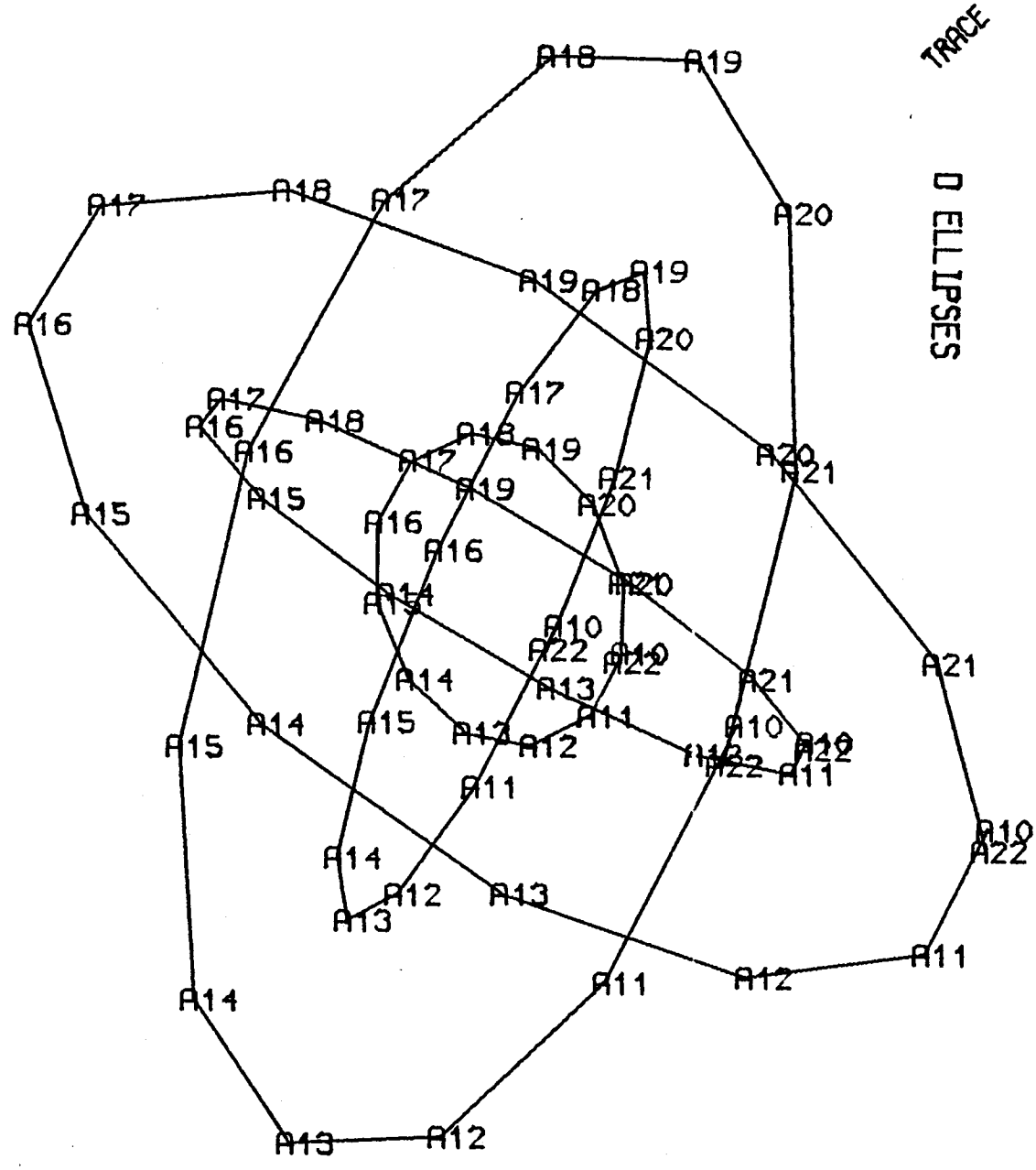
DE DES ELLIPSES T5R1&&

DE DU TEXTE G4050&& G10270&



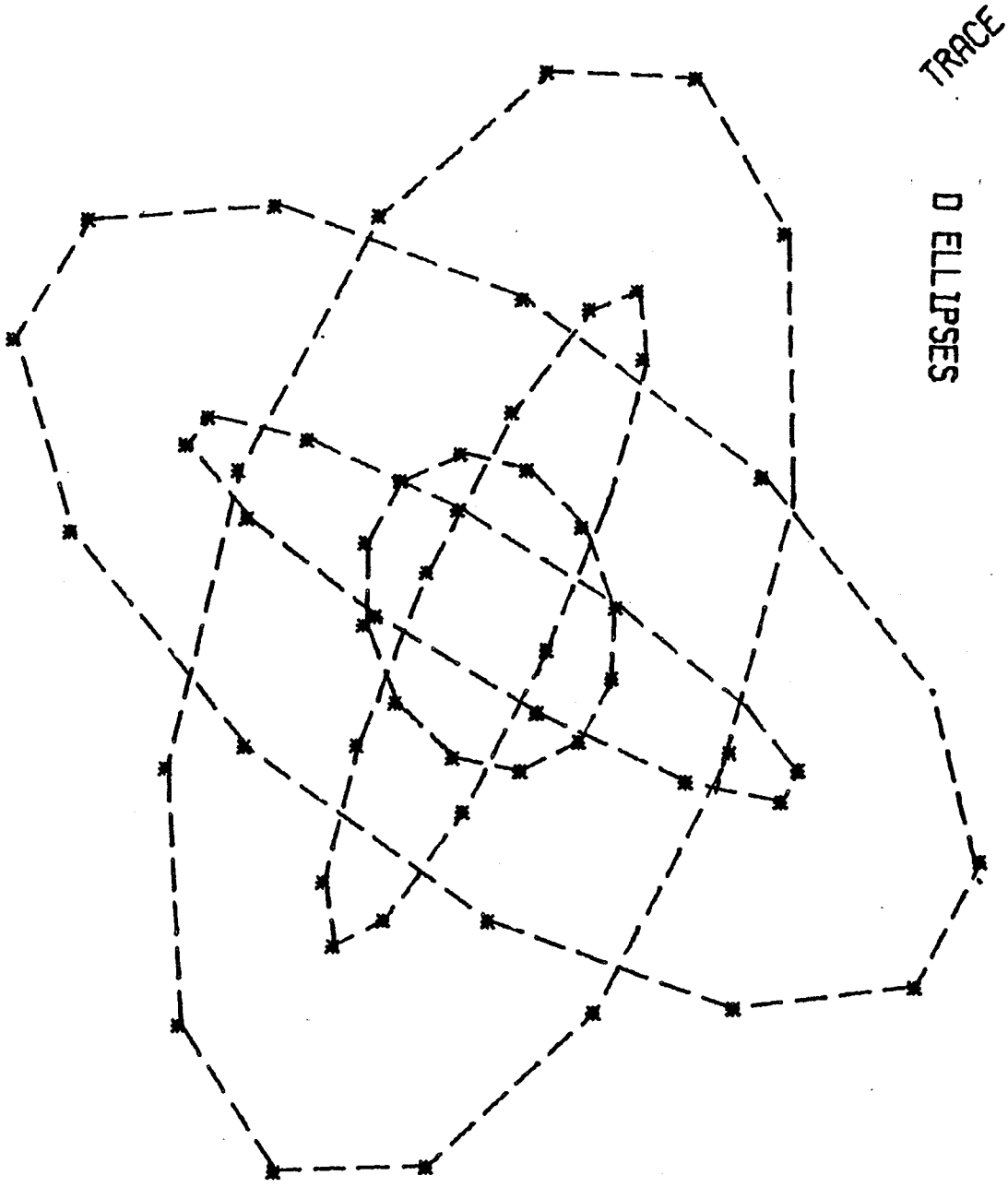
LEDUC LE 6494010012/16:27:20 DESSIN NO 017 ECHELLE: 07/1

NOMBRE D ELLIPSES 5
NB DE POINTS PAR ELLIPSE 12
XOEIL 1.000 YOEIL 0.200E01 ZOEIL 110.0000
MODE DES ELLIPSES T1MA10&&
MODE DU TEXTE G1045&& G20270&&



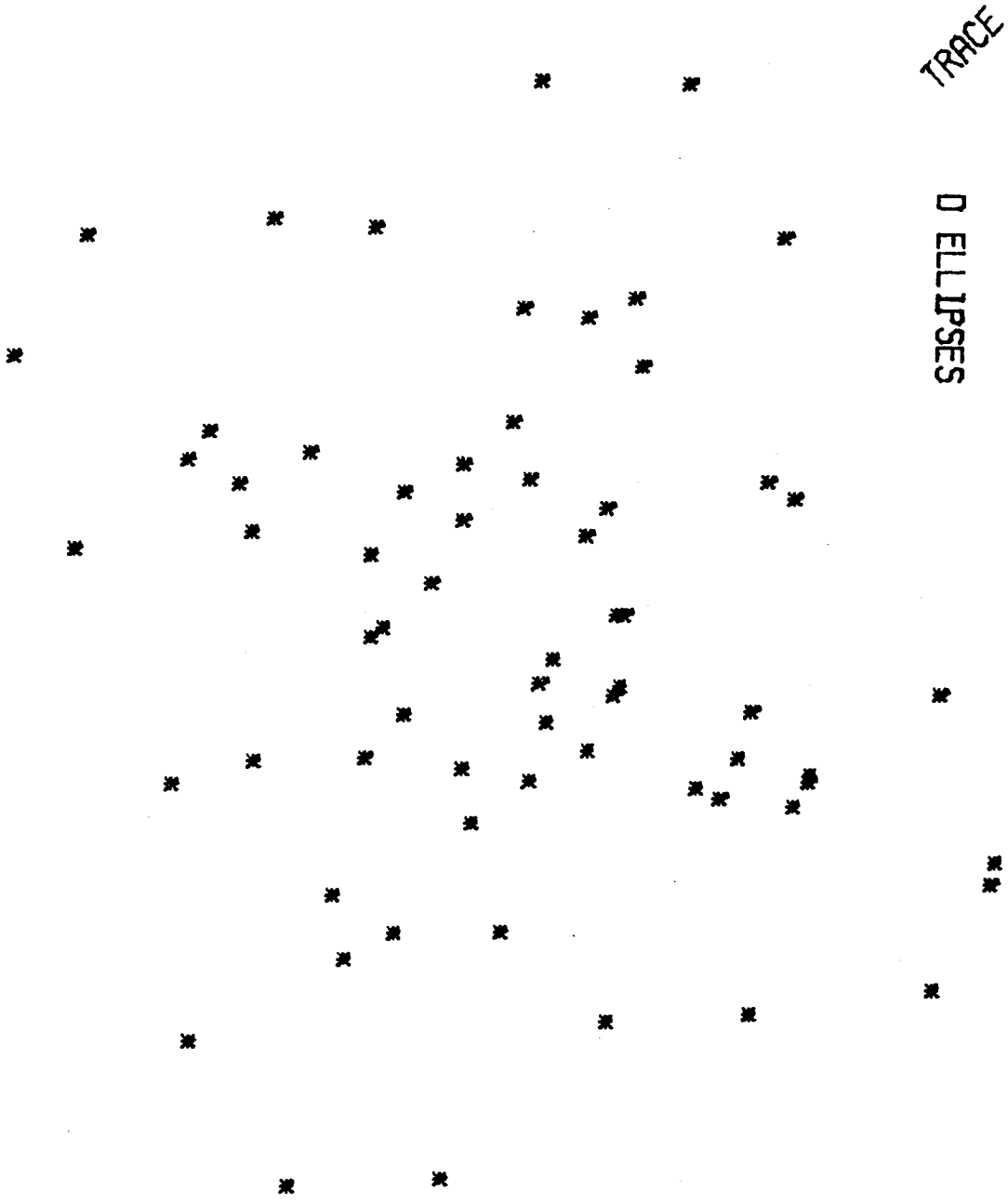
LEDUC LE 6494010012/16:31:15 DESSIN NO 018 ECHELLE: 07.

NOMBRE D ELLIPSES 5
NB DE POINTS PAR ELLIPSE 12
XOEIL 1.000 YOEIL 0.200E01 ZOEIL 110.0000
MODE DES ELLIPSES T3M* &&
MODE DU TEXTE G2045&& G20270&&



LEDUC LE 6494010012/16:32:54 DESSIN NO 019 ECHELLE: 07/

IOMBRE D ELLIPSES 5
IB DE POINTS PAR ELLIPSE 12
(OEIL 1.000 YOEIL 0.200E01 ZOEIL 110.0000
10DE DES ELLIPSES R3M* &&
10DE DU TEXTE G2045&& G20270&&



LEDUC LE 6494010012/16:33:49 DESSIN NO 020 ECHELLE: 07/

BIBLIOGRAPHIE

BIBLIOGRAPHIE

OUVRAGES GENERAUX

G1 P. MORVAN, M. LUCAS
Images et ordinateur. Introduction à l'infographie interactive.
Larousse, Collection Sciences Humaines, série Informatique Septembre 1976

G2 W.M. NEWMAN, R.F. SPROULL
Principles of Interactive Computer Graphics
Mc Graw Hill, Computer Science Series 1973

G3 M. LUCAS
Evolution des matériels graphiques interactifs.
Journée évaluation des matériels graphiques interactifs. AFCET, Février 1

G4 M. LUCAS
Les systèmes de visualisation graphique. Etat des recherches actuelles.
Séminaire de programmation IMAG, mars 1975

G5 W.M. NEWMAN
A system for interactive graphical programming.
Proc. AFIPS 1968 SJCC, vol. 32, p. 47-54

G6 J.D. FOLEY, W.L. WALLACE
The Art of natural graphic man-machine conversation
Proceedings of the IEEE, vol 62, n° 4, Avril 1974

G7 W.M. NEWMAN, R.F. SPROULL
An approach to graphics system design
Proccedings of the IEEE, vol 62, n° 4, Avril 1974

G8 J.L. POSDAMER
Some criteria for the evaluation of graphics implementation languages
Computers and Graphics, vol 2, p. 91-95, 1977

G9 H.E. KULSRUD
A general purpose Graphic Language
ACM Vol 11, n° 4, Avril 1968

G10 W.M. NEWMAN
Display Procedures
CACM, vol 14, n° 10, Octobre 1971

G11 O. LECARME

Niveau des langages et puissance des outils en programmation graphique
AFCET, journée "Visualisation des informations", octobre 1969

G12 O. LECARME

Fiabilité et transportabilité
L'influence du langage de programmation
IMAN, décembre 1975

NORMALISATION ET STANDARDISATION DES LOGICIELS GRAPHIQUES

N1 R.A. GUEDJ

Methodology in Computer Graphics
Compte-rendu du séminaire de SEILLAC (France), mai 1976

N2 M. LUCAS

Méthodologie et standardisation des techniques graphiques
Compte rendu du séminaire de SEILLAC (France), mai 1976
IFIP W.G. 5.2

N3 SIGGRAPH GSPC INTERFACE SUBGROUP

Interface Guidelines for Graphics Standards Development
Report, Septembre 1976

N4 SIGGRAPH GSPC CORE DEFINITION SUBGROUP

Core Subgroup Working Paper
Report, Octobre 1976

LOGICIELS GRAPHIQUES EXISTANTS

L1 M. LEMOINE

Une expérience probante de standardisation d'un logiciel graphique.
Congrès AFCET, Novembre 1976

L2 A. DUCROT, A. LEMAIRE

METAVISU, un système graphique intéractif.
Rapport de recherche n° 36 IRIA/LABORIA, novembre 1973

L3 EQUIPE GRAPHIQUE IRIA

Mise en oeuvre de METAVISU IV, Manuel
Liste IRIS 80 décrivant Fortran III D.

L4 E. SALTEL

Utilisation de la génération dynamique d'images et des procédures graphiq
dans le software METAVISU III.
Thèse de 3ème Cycle, Paris, Mai 1974

L5 A. LEMAIRE

Description comparative de GINO et Fortran III D
Groupe Graphique AFCET, 1974

L6 GROUPE GRAPHIQUE DE L'IRIA

"GIPSY", un système général pour la programmation graphique intéractive.
Congrès AFCET, novembre 1976

L7 CAD CENTRE

Cetec 'DINO' specification for IMLAC
EIN/CADC/76/001

L8 CAD CENTRE

Comparison of the Sproull and Thomas and Protocol A
Network Graphics Protocols
EIN/CADC/76/002

L9 GINOF

Code Generator Interface Specification, issue 2
Avril 1976

L10 LIMSI

Manuel d'EUCLID
Orsay 1975

LOGICIELS GRAPHIQUES BASES SUR LE CONCEPT DE "CONSOLE VIRTUELLE"

C1 I.W. COTTON
Network Graphic Attention Handling
ONLINE 1972, Conf. Proc., pp. 465-490

C2 C. IRBY, K. VICTOR
A proposed Network Text/Graphics Protocol
R.F.C. ≠ 553, juillet 1973

C3 J. MICHENER, I. COTTON, K. KEKKEY, D. LIDDLE, E. MEYER
Graphics Protocol
R.F.C. ≠ 493, avril 1974

C4 I. COTTON
Level O Graphics Input Protocol
R.F.C. ✦ 336, Mai 1972

C5 A. LEDUC-LEBALLEUR
Conception et réalisation d'un logiciel graphique basé sur le concept de console virtuelle.
Rapport interne, Grenoble, Janvier 1976

C6 L. LAUGIER, A. LEDUC-LEBALLEUR, M. LUCAS, F. MARTINEZ
GRIGRI : logiciel de base pour l'utilisation des consoles de visualisation du CICG
Note technique T 29, novembre 1975

LOGICIELS INDEPENDANTS PAR RAPPORT AUX TERMINAUX

LOGICIELS "ADAPTATIFS" DANS LE CADRE DES RESEAUX D'ORDINATEURS

I1 R.F. SPROULL, E. THOMAS
A network Graphics Protocol
Siggraph-ACM Computer Graphics, 8, 3, Fall 1974

I2 D. COHEN, E. TAFT
An interactive Network Graphics System
Computer and Graphics, vol 1, pp. 27-31, 1975

I3 J.C. MICHENER
The design of the ARPA Network Graphics Protocol

I4 W.M. NEWMAN
Device Independant Graphics Systems
Journées Graphiques AFCET-IRIA, décembre 1973

I5 R.F. SPROULL
Network Graphics Isn't Networking

I6 H. HOLMES
A simple Implementation Strategy for the ARPA Network Graphics Protocol
Juin 1976

I7 GROUPE DE TRAVAIL "GRAPHIQUES À TRAVERS LE RESEAU CYCLADES"
A restricted Protocol
Proposition de protocole, novembre 1976

I8 J. ENCARNACAO
A recommendation on Methodology in Computer Graphics
Janvier 1976

I9 J.E. GEORGE
Algorithms to Reveal Graphic Terminal Characteristics
SIGGRAPH-ACM, vol 9, n° 1, printemps 1975

I10 J.R. DAVIS
Device Independant Graphics Software is it possible ?
SIGGRAPH-ACM, vol 9, n° 1, printemps 1975

I11 J. MICHEL, A. VAN DAM
Experience with Distributed Processing on a Host/Satellite Graphics Syste
SIGGRAPH-ACM, vol 10, n° 2, Eté 1976

I12 G. HAMLIN
Configurable Applications for Satellite Graphics
SIGGRAPH-ACM, vol 10, n° 2, Eté 1976

I13 S.D. MOULTON, P.J. CORMAN
Remote Programmability of Graphic Interactions in a Host/Satellite Configuration
SIGGRAPH-ACM, vol 10, n° 2, Eté 1976

I14 A. LEDUC-LEBALLEUR, M. LUCAS, F. MARTINEZ
Conception et Réalisation d'un logiciel graphique interactif, independant du contexte d'utilisation : le logiciel de base GRIGRI
Séminaire d'Informatique SI n° 10, IMAG, juin 1977

I15 A. LEDUC-LEBALLEUR, M. LUCAS
Quelques problèmes d'indépendance des logiciels graphiques interactifs par rapport à leur contexte.
Rapport de recherche, à paraître.

REALISATION DE GRIGRI

R1 A. LEDUC-LEBALLEUR, M. LUCAS, F. MARTINEZ

GRIGRI : logiciel de base pour l'utilisation des consoles de visualisati
du CICG.
Note technique T42, décembre 1976

R2 T. WRIGHT

SIGCHR, a portable character generator
Computer and graphics, vol 10, n° 4, Hiver 1976

R3 B.W. JORDAN, R.C. BARRETT

A Scan Conversion Algorithm with reduced storage requirements
CACM, vol 16, n° 11, novembre 1973

R4 J.E. BRESENHAM

Algorithm for computer control of a digital plotter
IBM Syst, J.4, 1 (1965) pp. 25-30

DOCUMENTS TECHNIQUES

T1 IBM SYSTEM/360 COMPONENT DESCRIPTION
IBM 2250 Display unit model 1

T2 TEKTRONIX
4014, 1014-1, 4015 and 4015-1 Computer Display Terminal Service Instruction manual

T3 TEKTRONIX
4010 and 4010-1 Maintenance Manual

T4 ANDERSON-JACOBSON
AJ 830
Keyboard Printer Terminal
Operator's Manual

T5 CIT ALCATEL
Image processing and display terminals TRIM
NC 410/6B ed. C, 20/04/76

T6 CIT ALCATEL
MULTITRIM CIT ALCATEL
Notice de programmation NC 426

T7 CIT ALCATEL
MONOTRIM, logiciel de base pour consoles TRIM
NEIC 098