MASTER EN SCIENCES INFORMATIQUES

Etude et comparaison de systèmes graphiques

mise en œuvre sous UNIX

Bossard, Etienne

Award date:
1978

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX
NAMUR
Institut d'Informatique

Année Académique 1977 - 1978

ETUDE ET COMPARAISON DE SYSTEMES GRAPHIQUES; MISE EN OEUVRE SOUS UNIX

Etienne BOSSARD

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique.

Promoteur : Monsieur Joël Demarteau

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX
NAMUR
Institut d'Informatique

Année Académique 1977 - 1978

ETUDE ET COMPARAISON DE SYSTEMES GRAPHIQUES; MISE EN OEUVRE SOUS UNIX

Etienne BOSSARD

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique.

Promoteur : Monsieur Joël Demarteau

Monsieur Joël Demarteau m'a aidé par sa présence, ses conseils et son expérience. Je l'en remercie, ainsi que les membres du Département de Chimie qui m'ont permis de travailler sur leur ordinateur.

ETUDE ET COMPARAISON DE SYSTEMES GRAPHIQUES; MISE EN OEUVRE SOUS UNIX

Introduction

UNIX est un sytème d'exploitation à temps partagé, interactif, d'application universelle, développé pour des ordinateurs PDP 11/40, 11/45 et 11/70 de la Digital Equipment Corporation.

Le département de chimie des Facultés possède deux périphériques graphiques, à savoir un écran à rafraîchissement et une table traçante, travaillant à l'aide d'un PDP 11/45. Ces deux périphériques ne sont pas implantés dans le système UNIX. Le projet initial du présent travail a été de compléter le système UNIX en y ajoutant un programme de gestion pour un de ces deux périphériques graphiques. Le choix s'est porté sur l'écran, vu la possibilité du travail interactif qu'il offre.

Pour l'utilisation de graphiques, des choix d'un langage de programmation pour les programmes d'application et d'une bibliothèque de routines graphiques nous ont conduit à effectuer des comparaisons entre des langages de programmation, et entre des bibliothèques graphiques et des extensions graphiques de langages. Après ces différents choix, nous nous sommes appliqués à une phase d'implantation. Cette phase pratique se limite à l'implantation d'une partie d'une bibliothèque de routines graphiques, ainsi que d'un programme de gestion pour l'écran à rafraîchissement réalisant les fonctions graphiques de la partie de la bibliothèque.

I. SUR LES LANGAGES GRAPHIQUES

Un système graphique est un ensemble de matériels, tels que ordinateur et périphérique, géré par des logiciels permettant de travailler sur des appareils à orientation graphique, par exemple: table traçante, écran à rafraîchissement ou écran à mémoire associée. Un tel système est qualifié d'interactif, si son logiciel dispose d'une ou plusieurs fonctions d'introduction d'information et si son matériel offre les outils qui s'y rapportent. Les fonctions d'introduction d'information permettent à l'utilisateur de modifier ou d'orienter le déroulement d'un programme travaillant à l'aide d'un tel système. Des exemples d'outils réalisant des fonctions d'introduction sont le crayon lumineux et le clavier.

Dans la conception d'un sytème graphique interactif, plusieurs langages de programmation mis à la disposition de l'utilisateur permettent à ce dernier de faire un choix qui facilite une bonne utilisation du système graphique. Un des objectifs importants de cette bonne utilisation est la performance des programmes d'application; un autre, la compatibilité d'un programme, appelée aussi portabilité, est la capacité d'un programme à être transplanté d'une installation à une autre, dans le meilleur des cas sans modifications, ou alors avec un nombre restreint de modifications n'altérant pas la structure du programme.

1° Langages de programmation

Des langages particulièrement performants sont les langages orientés machine, appelés assembleurs. Ils ont l'inconvénient d'être peu lisibles et de ne pas respecter l'objectif de compatibilité. Nous les avons d'emblée écartés.

Les langages de programmation de haut niveau, candidats à supporter un sytème graphique interactif (NEW 1), sont des langages à orientation scientifique. Certains d'entr'eux offrent en plus une orientation conversationnelle. Parmi les

langages de haut niveau, tels que Algol, APL, Euler, Fortran, PL1, les langages à orientation scientifique: Algol, Euler, Fortran et PL1 sont rarement implantés en conversationnel. APL est le seul langage réellement orienté en conversationnel, mais se base sur une syntaxe peu usuelle. Parmi les langages à orientation scientifique, le Fortran occupe une place particulière à cause des avantages majeurs, comme son efficacité et sa disponibilité: la plupart des ordinateurs et même des mini-ordinateurs supportent le Fortran. Par contre, Fortran comporte des inconvénients comme une portabilité non absolue, notamment pour des instructions entrée-sortie, et sa lourdeur. La lourdeur d'un programme Fortran, tant pour l'écriture que pour la lecture,est due à l'absence de facilités de structure et à une grande pauvreté en instructions conditionnelles.

Cependant, malgré ces inconvénients, sa grande disponibilité fait que Fortran reste néanmoins le plus portable des langages scientifiques et cet avantage prime sur les avantages des autres langages. Nous conseillons donc l'utilisation du Fortran comme langage de programmation pour les applications.

2° Langages graphiques et biliothèques graphiques.

En 1972, on a pu poser la question de l'existence d'un langage graphique universel et la discuter à Vancouver lors d'une conférence de l'IFIP sur les langages graphiques (IFIP 72). Pour répondre à cette question, on peut comparer des bibliothèques de routines graphiques. De nombreuses bibliothèques graphiques ont comme langage de support le Fortran, par exemple Adage, Calcomp, Dissp1a, Euclid, CGS, Gino-F, Gino-3D, GPGS, Tektronix. Cela est une des raisons qu'invoque Newman pour dire que Fortran est ce langage universel.

Pour examiner cette question, nous croyons nécessaire d'abord de bien distinguer un langage graphique d'une bibliothèque graphique. Un langage graphique est un langage de programmation qui a reçu des extensions graphiques. Des exemples de langages graphiques sont Euler-G (New 2),

Logo (New 3, New 4), Algol 68-G (Hag). L'utilisateur n'a pas le choix du langage de programmation.

Une bibliothèque graphique est un ensemble de routines. Toute opération graphique est un appel de routine. Les langages utilisés pour écrire la bibliothèque et pour écrire le programme utilisateur peuvent différer l'un de l'autre. Les schémas ci-dessous donnent une idée de la différence entre langages graphiques et bibliothèques graphiques.

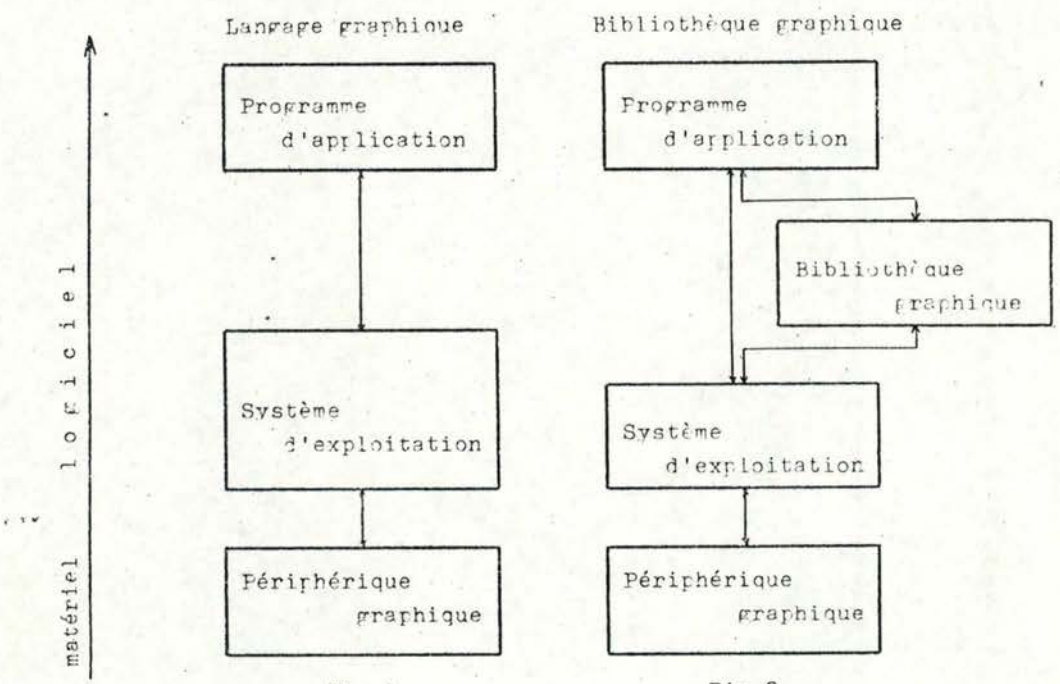


Fig 1. Fig 2.

Dans le présent travail, nous utilisons un système d'axes particulier pour la représentation des figures. L'axe horizontal, souvent omis, représente une échelle de répétition. L'axe vertical indique, soit une graduation qui part du matériel vers le logiciel, soit une frontière qui divise la figure en deux parties, une indépendante et l'autre dépendante du processus graphique. Parfois, les deux significations de l'axe vertical sont utilisées simultanément.

Lorsqu'un centre informatique change de bibliothèque graphique, ses programmes d'application à orientation graphique doivent en général subir des modifications souvent plus importantes que le simple changement d'un ou plusieurs noms de routines. Même dans le cas où ces biblio-

thèques sont écrites dans le même langage de programmation, la structure des programmes en est parfois affectée. Cela est le cas notamment pour les bibliothèques écrites en Fortran. D'autre part, il existe des extensions graphiques du Fortran, mais les mêmes problèmes de modifications de structures se posent lors d'un changement dans le système graphique. Par conséquent, Fortran n'est sûrement pas un langage graphique universel.

Le Fortran est en général conseillé comme langage de programmation d'une bibliothèque graphique. Ce choix est justifié comme auparavant par l'efficacité et la portabilité du Fortran. Toutefois, le choix du Fortran pour l'écriture d'une bibliothèque graphique est moins impérieux que le choix du Fortran pour l'écriture des programmes d'application. En effet, un changement de bibliothèques graphiques est peu fréquent, tandis qu'on échange couramment des programmes d'application entre différents centres informatiques.

3° Comparaisons de bibliothèques graphiques.

Les bibliothèques graphiques et les extensions graphiques de langages sont nombreuses. Les constructeurs de processeurs graphiques proposent des bibliothèques graphiques avec leur matériel, par exemple Adage, Calcomp, Dec, Imlac, Tektronix et Vector General. Des organisations ou centres informatiques en produisent. Voici des exemples de bibliothèques graphiques: Dissplа,Euclid, GCS, Gino-F, GPGS, IG, Omnigraph; et des exemples d'extensions de langages: Euler-G, Fortran, Logo, Algol 68-G.

Une comparaison de huit bibliothèques graphiques a été présentée par les membres du Graphics Standards Planning Committee(GSPC) of ACM/ SIGGRAPH lors de la quatrième conférence annuelle sur les graphiques pour ordinateur et les techniques interactives,appelée SIGGRAPH '77 (SIG 77). La publicationdu rapport de ce Committee (GSPC) a été éditée en automne 1977, après le début du présent travail. Cette comparaison sous forme de tableaux nous a fourni une aide

précieuse et nous a servi de guide pour des comparaisons avec d'autres bibliothèques graphiques. Les principales fonctions comparées sont les fonctions d'introduction d'information aussi bien graphiques qu'alphanumériques et les fonctions de sortie telles que les systèmes de coordonnées, les spécifications de l'image: transformations, segmentation et primitives.

Les bibliothèques comparées sont: Adage, Calcomp, Disspla, GCS, GINO-F, GPGS, IG et Tektronix. Nous avons étudié d'autres bibliothèques dans le but de les comparer entr'elles. Dans cette entreprise, nous nous sommes heurtés à de grandes difficultés, car les documents dont nous disposions n'étaient pas suffisamment complets. Contrairement aux auteurs du rapport du Comité GSPC de SIGGRAPH, nous n'avions pas la possibilité de recours auxproducteurs des bibliothèques. Ainsi, c'est en vain que nous avons sollicité des informations sur la bibliothèque Euclid au Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur à l'Université de Paris XI. De même, il est dommage que la prospection en vue de l'acquisition de terminaux graphiques n'ait pas été entreprise après le choix par les Facultés du DEC 20/50.

Les bibliothèques graphiques que nous avons soumis aux mêmes critères de comparaison que ceux du comité GSPC de SIGGRAPH sont:

- celle fournie par Digital pour le processeur graphique VT 11 sous le système d'exploitation RSX 11M (DEC-F)
- un software de base pour traçeur Benson 441 écrit à l'Université de Bruxelles par le professeur Machgeels (Mach)
- une bibliothèque tournant sous UNIX au Queen Mary College of London, appelée Graf.C (Graf.C)

Les deux premières sont écrites en Fortran; la dernière en C. Les résultats de cette comparaison se trouvent en annexe sous forme de tableaux. Ils sont précédés d'une description des critères de comparaison et des résultats de la comparaison du comité GSPC.

4° Proposition pour des fonctions graphiques standard

a) Nécessité de fonctions graphiques standard

Suite à la comparaison des huit bibliothèques, le comité GSPC propose une méthodologie générale, ainsi que des fonctions graphiques standard . La justification essentielle de la nécessité de standards est la compatibilité des programmes d'application avec des installations différentes. L'utilisation de standards permet d'améliorer la compatibilité des programmes d'application et de limiterles modifications à effectuer pour tourner sur une autre configuration.

Un des problèmes qui s'opposent à cet objectif de compatibilité est la grande diversité dans le matériel graphique et dans les fonctions supportées par ce matériel graphique. Une solution réalisant cet objectif de compatibilité est un système graphique obéissant au schéma de la figure 3

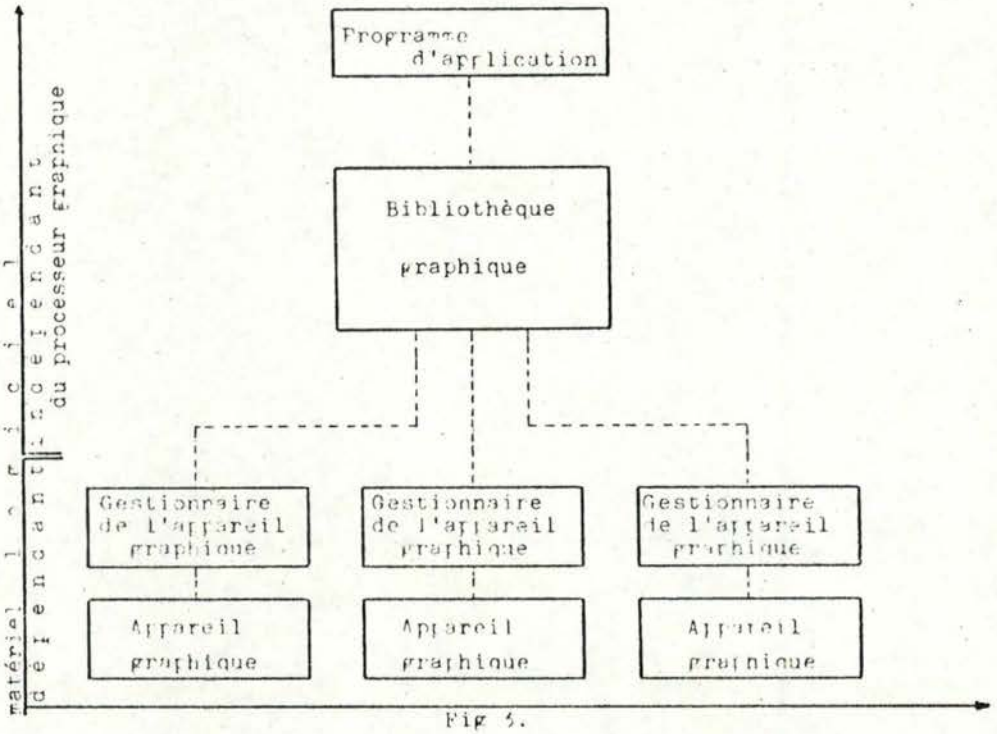


Fig 3.

Ce schéma de la bibliothèque graphique GPGS (General Purpose Graphics System) (CARU 77) a été repris par le comité GSPC de SIGGRAPH '77. La configuration représentée dans la figure 3 permet d'obtenir la compatibilité du programme d'application,du moins pour la partie graphique: il suffit que les fonctions constituant l'interface entre le programme

d'application et la bibliothèque graphique soient standard . Le système graphique se présente de la façon suivante: le programme d'application dialogue avec la bibliothèque graphique à travers l'interface. La bibliothèque, qui ignore le type d'appareil, transforme les instructions graphiques du programme d'application en ordres pour le gestionnaire, c'est-à-dire, le programme de gestion de l'appareil graphique (driver). Le gestionnaire génère ensuite les ordres pour l'appareil graphique. L'interface entre la bibliothèque graphique et le gestionnaire est constitué d'un nombre réduit de fonctions primitives. Ces primitives sont , par exemple:"déplacement" (choix de coordonnées)

Une généralisation de la configuration selon la figure 3 a été proposée par Bergeron (Berg 76). Elle tient compte du fait que la taille de l'interface entre la bibliothèque graphique et le gestionnaire est variable. L'interface maximum est constitué de l'ensemble de toutes les primitives. Chaque fonction primitive peut se traduire à l'intérieur de la bibliothèque en une commande pour le gestionnaire de l'appareil graphique. L'interface minimum est constitué d'une partie de ces primitives appelées atomes. Les atomes se distinguent des autres primitives en ce sens que toute primitive qui n'est pas un atome peut être aussi obtenue par une succession d'atomes. L'ensemble des atomes n'est pas nécessairement un ensemble minimum. Par exemple, un ensemble d'atomes peut être:

- une opération de déplacement (choix de coordonnées)
- une opération d'affichage de point
- une opération de tracer une ligne
- une opération d'affichage de texte

Cet ensemble n'est pas minimum. Les deux atomes choix de coordonnées et tracer une ligne forment l'ensemble minimum. En effet, tracer une ligne de longueur nulle est un point et à l'aide de points ou de lignes, on peut écrire des textes. L'ensemble de primitives peut inclure, outre les atomes, des tracés de courbes et de polygones, par exemple.

Lors de l'initialisation du sytème, le gestionnaire de l'appareil graphique positionne un indicateur interne, spécifiant si le processeur graphique réalise ou ne réalise

pas chaque primitive qui n'est pas un atome. Avant de passer une commande au gestionnaire de l'appareil graphique pour une primitive qui n'est pas un atome, la bibliothèque graphique teste l'indicateur de la primitive dans le gestionnaire. Si le processeur graphique ne supporte pas cette primitive, la bibliothèque graphique la transforme en fonction des atomes, ce qui se traduit en une commande au gestionnaire de l'appareil graphique.

La taille de l'interface est fonction des performances du processeur graphique et du choix à l'initialisation de primitives qui ne sont pas des atomes.

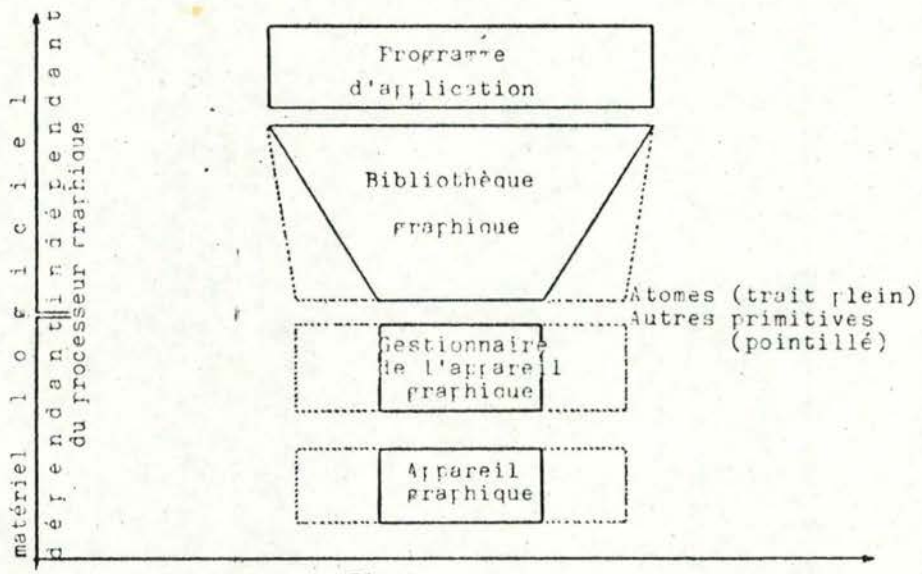


Fig 4.

Une configuration selon la figure 4, ci-dessus, permet une écriture du programme d'application selon des standards, mais permet aussi d'obtenir un rendement optimum de la partie graphique, si le processeur graphique supporte de telles performances. La généralisation selon Bergeron (figure 4) charge le système ou la bibliothèque de tâches supplémentaires, à savoir la génération des primitives qui ne sont pas des atomes en fonction de ces derniers. Cette surcharge risque d'être exagérée, surtout si l'ordinateur de support est un mini-ordinateur.

b) Les fonctions graphiques standard proposées par le comité GSFC

Dans ce paragraphe, nous passons en revue les principales fonctions standard, à savoir les primitives, la segmentation du dessin, la paramétrisation des primitives et des segments, les transformations du champ de vision, les fonctions d'introduction d'information, le contrôle du système graphique, et des interfaces spéciaux. Ces fonctions standard constituent l'interface entre le programme utilisateur et la bibliothèque graphique.

Primitives

Les primitives proposées dans les standards sont au nombre de cinq: déplacement, ligne polygone, texte et classe de repères. Ces cinq primitives peuvent se présenter à deux comme à trois dimensions. Chacune des primitives modifie une position appelée position courante. La position courante représente la dernière position atteint dans le cours de l'exécution d'un programme. Les primitives peuvent modifier cette position courante, soit en coordonnées absolues, soit en coordonnées relatives.

Exemple: Si (x,y,z) est la dernière position courante, une primitive en coordonnées absolues dont les arguments sont $(\bar{x},\bar{y},\bar{z})$ modifie la position courante et lui donne la valeur $(\bar{x},\bar{y},\bar{z})$. Par contre, une primitive en coordonnées relative d'argument (x',y',z') modifie la position courante et lui donne la valeur $(x+x',y+y',z+z')$.

Le jeu de primitives ne comprend pas la primitive "point", mais une primitive classe de repères qui est une généralisation de la notion point. La seule caractéristique géométrique d'un objet décrit par un repère est sa position géométrique. La représentation d'un élément d'une classe de repères se limite donc à un repèrage qui n'est sujet à aucune trasformation. Cette représentation peut être d'un caractère spécial, fourni par le terminal graphique.

Segmentation

Tout dessin est subdivisé en un certain nombre de segments et comprend donc au moins un segment. Il ne peut pas avoir de segments imbriqués les uns dans les autres; on dit que le

niveau de segmentation du dessin est simple. Un segment est une portion du dessin qui en général correspond à l'image d'un objet. L'opération de base en vue d'une segmentation d'un dessin est la création d'un segment. Après cette opération, le segment existe, même s'il n'est pas complètement décrit. Des opérations d'effacement et de changement de nom du segment sont fournies par les standards. Enfin, une opération de verrouillage, appelée fermeture, permet de clôturer la description du segment. Certaines opérations de transformation ne peuvent être effectuées qu'après fermeture des segments.

Paramétrisation

Pour les primitives comme pour les segments, l'utilisateur dispose de paramètres de deux types auxquels il peut attribuer des valeurs. Des paramètres statiques accompagnent les primitives ou les segments de façon immuable jusqu'à la fin du temps de vie de ces entités. Ainsi, la couleur, l'intensité, le style de ligne, la qualité du texte, sont des exemples de paramètres statiques pour les primitives. Ensuite, le type de mémorisation d'un segment est un paramètre statique pour les segments. Le type de mémorisation peut être, par exemple "non retenu" ou "retenu" avec diverses transformations.

Des paramètres dynamiques pour segments sont des paramètres qui peuvent être modifiés pendant le temps de vie des segments. Il existe des paramètres dynamiques qui rendent un segment visible au fur et à mesure de sa description, ou discernable, par exemple par clignotement, ou détectable par une des fonctions d'entrée. D'autres paramètres dynamiques opèrent les transformations de l'image par translation, rotation et mise à l'échelle.

Ces paramètres ne doivent pas nécessairement être spécifiés. A la création de chaque segment, les paramètres statiques reçoivent automatiquement une valeur initiale, appelée valeur par défaut. Il en est de même pour les paramètres dynamiques. La valeur de ces paramètres peut être consultée à tout moment par l'utilisateur.

Transformation du champ de vision

Des opérations transformant le champ de vision sont fournies par les standards. Ces opérations permettent de définir des plans de vision, des projections diverses, des fenêtres et des volumes pour modifier l'image de l'objet. Une telle opération de transformation peut concerner plusieurs segments du dessin et même le dessin tout entier. Elle est obtenue par modification de paramètres statiques et elle exige que tout segment du dessin soit verrouillé.

Fonctions d'introduction d'information

Les appareils qui réalisent des fonctions d'introduction se divisent en deux groupes: les appareils à évènement (event devices) et des appareils échantillonés (sampled devices). Dans la catégorie des appareils à évènement se trouvent le pointeur dont le prototype est le crayon lumineux, servant à identifier un segment et même une primitive à l'intérieur d'un segment, ensuite le clavier pour introduire l'information alphanumérique et le bouton poussoir simulant des clefs logiques. Dans la catégorie des appareils échantillonés se trouvent le localisateur qui renvoie une position d'un curseur et le mesureur qui renvoie une valeur scalaire. A l'aide des fonctions d'introduction, diverses opérations de sélection et de prises de données peuvent être réalisées.

Contrôle

Le système dispose aussi de plusieurs fonctions pour contrôler et interroger le processus de génération du dessin, ainsi que l'état du système.

Interfaces spéciaux

Un premier interface, appelé "crochet" par le comité GSPC est indépendant du processeur graphique. Il permet d'accéder à des informations ou procédures internes à la bibliothèque qui ne sont pas accessibles via l'interface standard. Un crochet permet à un système de modelage de passer outre d'une partie du processus de génération de l'image. En particulier, un crochet peut permettre d'opérer des transformations du champ de vision et de modelage en un seul pas. Les transformations du champ de vision diffèrent des transformations de modelage qui sont utlisées pour la construction des

différentes parties du dessin. Dans un système de modelage, les objets sont décrits dans leur propre système de coordonnées. Une transformation de modelage transforme les coordonnées du système de modelage en des coordonnées propres à l'utilisateur. Selon le comité GSPC, les transformations de modelage et celles du champ de vision utilisent les mêmes opérations matricielles. La constitution et l'utilisation de cet interface "crochet" ne sont pas étayées d'exemples et nous restent obsurs.

Un second interface spécial permet d'accéder à des fonctions non standard d'une manière standard au moyen d'un mécanisme de changement de mode. Il est dépendant du processus graphique. Le mécanisme de changement de mode donne accès à des fonctions spéciales, supportées par le processeur graphique. Un exemple de fonction spéciale est la génération d'une courbe. Le code graphique est généré dans la fonction de changement de mode et non pas à l'intérieur du programme d'application. Afin d'augmenter la compatibilité d'un programme utilisant cette possibilité, il est conseillé d'isoler et de limiter les appels à ce mécanisme et de commenter correctement la fonction appelée.

c) Discussion de la paramétrisation

Un des points qui a retenu notre attention est la paramétrisation des fonctions primitives et des segments. Des choix basés sur des options différentes peuvent se présenter. Rappelons que les paramètres sont le type de ligne, la couleur, l'intensité, le clignotement et les paramètres de transformation: translation, rotation et mise à l'échelle.

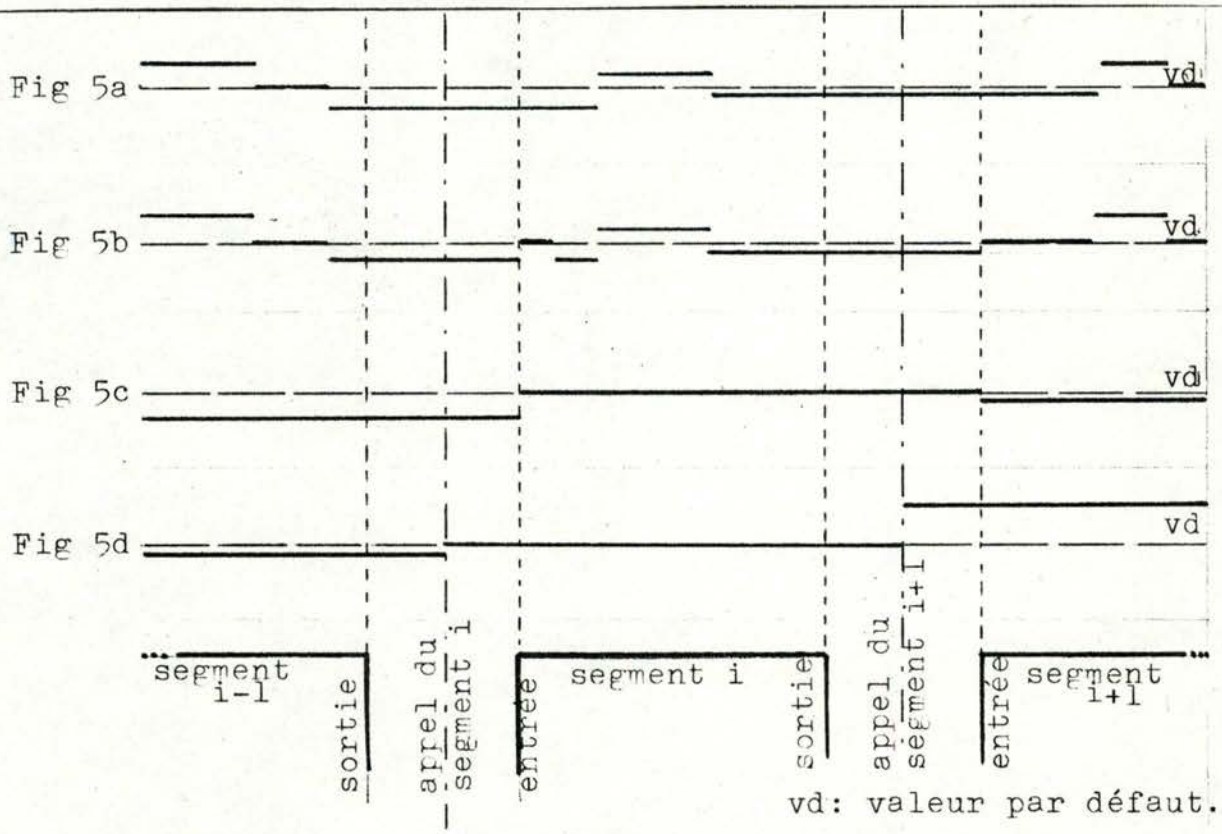
Voici quatre options:

1) Les instructions affectant les valeurs des paramètres sont mélangées avec les instructions de tracer des lignes et d'afficher des caractères de texte à l'intérieur d'un segment, si toutefois une segmentation existe. Les valeurs de ces paramètres passent d'un segment à l'autre sans être altérées. C'est une option orientée processeur graphique.
2) Au lieu de passer d'un segment à l'autre comme dans l'option 1, les paramètres reçoivent une valeur par défaut à l'entrée

d'un segment. C'est l'option que nous désignons par "Paramètres statiques pour primitives".

3) Les paramètres sont associés à un segment entier ou à un appel de segment. L'entrée dans un segment donne aux paramètres les valeurs par défaut ou les valeurs associée l'appel. C'est l'option que nous désignons par "paramètre statique et dynamique pour segments".

4) Les paramètres sont associés à l'appel d'un segment. L'entrée dans un segment donne aux paramètres les valeurs associées à l'appel ou les valeurs par défaut, si ces paramètres ne sont pas mentionnés à l'appel du segment. C'est l'option que nous appelons "paramètre dynamique de segment".



Les figures 5a, 5b, 5c, 5d représentent des diagrammes de modification de paramètres en fonction de l'entrée et de la sortie de segments. A la fig. 5a, la valeur d'un paramètre passe au dessus des limites d'un segment, sans être altérée, mais peut changer à l'intérieur d'un segment, ce qui correspond à l'option 1. A la figure 5b, la valeur d'un paramètre à l'entrée d'un segment passe par la valeur par défaut, mais peut être modifiée à l'intérieur du segment; ceci correspond à l'option 2. A la figure 5c, la valeur du paramètre passe par

la valeur par défaut ou reçoit la valeur associée au segment tout entier, ceci correspond au paramètre statique de l'option 3. A la figure 5d, les valeurs des paramètres reçoivent la valeur par défaut, sauf si ces paramètres sont spécifiés à l'appel. Ceci correspond aux paramètres dynamiques de l'option 3 et à l'option 4. Dans les Figures 5c et 5d, les paramètres ne sont pas modifiés à l'intérieur d'un segment.

La première option est très orientée processeur graphique. Elee représente la manière dont les paramètres du dessin sont modifiés dans une bonne partie des processeurs graphiques. L'inconvénient de cette première option est de trouver les primitives graphiques proprement dites groupées avec des paramètres dans une seule instruction, par exemple: line(x,y,a,b,c) a,b,c sont des paramètres. Ce type d'instruction est orienté vers un processeur graphique et les paramètres y sont parfois optionnels.

Une instruction amalgamant les valeurs d'une fonction avec d'autres données qui n'ont aucun rapport avec la fonction décrite par cette instruction met en péril la compatibilité d'un programme d'application. Ainsi, si un des paramètres a, b, c est le clignotant, il n'a aucun rapport avec la notion de tracer une ligne. Une telle instruction est souvent illisible, a fortiori si les paramètres sont optionnels. Ceci, les standards ne peuvent pas admettre.

Comme les valeurs passent d'un segment à l'autre sans être altérées cette option est aussi à rejeter dans le cadre d d'un travail interactif. Notamment dans le cas où la suppression d'un segment donne lieu à des modifications autres que la simple disparition de ce segment. Des exemples de cette option sont DEC, GINO-3D, GINO-F, Omnigraph.

La deuxième option "paramétrisation statique pour primitives" permet de décrire un dessin indépendemment de l'ordre dans lequel les segments sont appelés. Cela permet un travail interactif, sans distorsion du dessin.

La troisième option "paramètres statiques et dynamiques pour segments" présente, outre les avantages de la deuxième option, l'économie d'une répétition de la description d'un segment lors d'une modification d'un paramètre dynamique. Toutefois, la segmentation du dessin doit être pousséeplus loin que dans la deuxième option.

La quatrième option où tous les paramètres sont dynamiques est prônée par E. L. Thomas (Thom 76). Elle présente, outre les avantages des deuxième et troisième options, la facilité de pouvoir utiliser le même segment avec des paramètres totalement différents, ainsi qu'un traitement généralisé pour tous les paramètres. Par contre, cette option compte comme inconvénients une surcharge due à la multiplication des zones de paramètres, une subdivision exagérée de l'image en segments et sous-segments et enfin nécessite une gestion des paramètres à l'entrée et à la sortie d'un segment.

Si un choix était à effectuer parmi les quatre options pour la conception d'un système graphique, la première option orientée processeur graphique est à éliminer pour ses mauvaises performances interactives. La dernière option présente de nombreux avantages, mais le prix à payer pour ceux-ci risque d'être non négligeable: tout système ne supporte pas la surcharge en zones de mémoire pour les paramètres et leur gestion à l'entrée et à la sortie d'un segment.

Il est intéressant que certains paramètres aient un caractère dynamique. C'est le cas des paramètres de transformation des coordonnées (translation, rotation, mise à l'échelle)

Dans les standards proposés pour la paramétrisation des primitives et des segments, les paramètres se divisent en deux classes: statiques et dynamiques. L'affectation de certains paramètres a lieu dans le segment; ce sont les paramètres statiques pour les primitives qui correspondent à la deuxième option. Par contre, les paramètres statiques et dynamiques pour segments sont gérés selon la troisième option.

Une solution mitigée entre les deuxième et troisième options présente comme avantage une limitation du nombre de segments et par le même fait une diminution des zones pour le paramètres. L'inconvénient est que tous les paramètres ne soient pas traités de la même façon.

d) Discussion de la segmentation.

Suivant les standards proposés, la segmentation du dessin n'a lieu qu'à un seul niveau. Ce choix peut s'opposer à une segmentation du dessin à plusieurs niveaux (Foley 76)

L'avantage de la segmentation à plusieurs niveaux se manifeste dans la dénomination et la modification des segments.

Il est possible d'admettre plusieurs niveaux de dénomination de segment; ceci permet de décrire plus facilement un dessin qui contient de nombreux éléments identiques. Ainsi, des filtres électriques, souvent nombreux dans un circuit électrique sont constitués chacun d'une résistance et d'un condensateur. Une dénomination à un seul niveau oblige à répéter l'ensemble résistance et condensateur à chaque apparition d'un filtre.

Il est également possible de distinguer plusieurs niveaux de modification. Il est évident que le nombre des niveaux de modification ne peut pas être supérieur au nombre des niveaux de dénomination. L'avantage de plusieurs niveaux par rapport à un seul est le même tant pour la modification que pour la dénomination. Pour un seul niveau de modification, des modifications sur un filtre, par exemple, sont effectuées en deux étapes: une pour la résistance, ensuite une pour le condensateur au lieu d'une seule pour le filtre directement.

La structure d'une segmentation à plusieurs niveaux ressemble aux structures des données en Cobol et aux structures de bloc en Algol. L'inconvénient majeur de ces types de structures est la surcharge due à la gestion des paramètres à l'entrée et à la sortie d'un segment. Si les paramètres ne passent pas avec leur valeur au-dessus des entrées et des sorties d'un segment, c'est-à-dire s'ils suivent une des options deux, trois ou quatre, décrites plus haut, une gestion dynamique de ces paramètres doit être envisagée. Il faut alors retenir les anciens paramètres à l'entrée du segment et les restaurer à la sortie. Ceci est la raison du fait que la plupart des langages n'ont qu'un niveau de modification. Des tableaux de comparaison concernant la dénomination et la modification des segments, tirés en partie de la littérature (Foley 76) et en partie dressés par nous se trouvent dans l'annexe III.

II. LE CHOIX D'UNE BIBLIOTHEQUE GRAPHIQUE OU D'UN LANGAGE GRAPHIQUE.

Dans une première partie, nous allons tenter de limiter l'ensemble des bibliothèques et langages graphiques à un ensemble réduit, contenant les bibliothèques et langages les plus intéressants et performants. Dans une seconde partie, nous décrirons le choix effectué pour la configuration du département de chimie des Facultés.

1) Critique des bibliothèques et langages graphiques.

Un langage graphique oblige l'utilisateur à adopter certaines formes standard pour l'écriture d'un programme et à respecter la syntaxe du langage. Au fait, la partie graphique d'un langage suit nécessairement la syntaxe du langage, c'est un grand avantage. Les instructions graphiques d'un langage graphique sont des opérations d'entrée-sortie s'adressant à un périphérique particulier. Par conséquent, les erreurs de syntaxe, y compris celles de la partie graphique, sont détectées à la compilation; c'est un autre avantage. Toutefois, l'extension de la partie graphique d'un langage n'est pas chose aisée. En effet, il faut d'une part respecter la syntaxe du langage et d'autre part, le compilateur doit être étendu de façon à être capable de traiter cette extension. Par contre, la compatibilité ou portabilité d'un programme d'application écrit à l'aide d'un langage graphique est réduite; c'est le cas notamment pour les langages Euler-G,et Logo qui d'ailleurs ne sont pas disponibles partout. Même en Fortran, les opérations d'entrée et de sortie sont loin d'être standardisées.

Une bibliothèque graphique dispose d'avantages majeurs, notamment une bibliothèque graphique peut être d'une conception indépendante du sytème informatique de support et présenter une possibilité d'extension aisée, ce qui n'est pas le cas pour un langage. De plus, au moins dans certains cas, elle offre une portabilité non négligeable.

Enfin, une bibliothèque graphique est plus simple à écrire qu'un nouveau langage. Par contre, la syntaxe d'une bibliothèque graphique est en général peu cohérente. Nous considérons qu'une bibliothèque graphique est préférable à un langage graphique. De ce fait, nous limitons la suite de l'exposé aux bibliothèques graphiques.

Portabilité

La portabilité d'une bibliothèque graphique est sa capacité à être transplantée d'une installation à une autre, ce qui implique que cette bibliothèque est compatible (1) avec l'ordinateur et (2) avec le processeur graphique. La portabilité d'une bibliothèque graphique peut être obtenue, dans le cas idéal, par l'indépendance vis-à-vis du ou des processeurs graphiques que cette bibliothèque supporte. La portabilité est un des critères déterminants pour les divers choix à opérer, décrits ci-dessus.

Une bibliothèque graphique de qualité doit être capable de supporter plusieurs appareils graphiques sur lesquels on peut travailler en parallèle. La bibliothèque peut être reliée à un écran à rafraîchissement, une table traçante, un tube à mémoire associée, construits par des fabricants différents.

Comparaison et choix de bibliothèques graphiques

Les performances des bibliothèques graphiques sont généralement très valables et les possibilités de bibliothèques différentes sont sensiblement les mêmes. C'est pourqui nous allons concentrer nos comparaisons sur un ou deux avantages et un ou deux inconvénients de chaque bibliothèque que nous considérons comme principaux. Cela nous permettra ensuite de procéder à des éliminations. Le tableau ci-dessous reprend les bibliothèques suivantes, chacune avec ses avantages (+) et inconvénients (-) principaux: Adage, Calcomp, Dissp1a, GCS, Gino-F, GPGS, IG, Tektronix, DEC, Software pour traceur digital, Graf.C

Tableau de comparaison.

Adage	(+)	un jeu de fonctions d'introduction très complet
	(-)	peu portable, très orienté vers un type d'ordinateur et vers un type de matériel graphique.
Calcomp	(+)	une très grande portabilité pour les ordinateurs.
	(+)	le prix d'achat: 500 à 1500 $
	(-)	pas de fonctions d'introduction d'information.
	(-)	pas de possibilité de segmentation de l'image et limitation à deux dimensions.
Disspla	(+)	grande portabilité avec des ordinateurs et avec des appareils graphiques
	(-)	pas de fonctions d'introduction d'information
	(-)	prix d'achat 22000 $
GCS	(+)	portabilité grande pour ordinateurs et appareils gr.
	(+)	prix d'achat "free"
	(-)	pas de paramètres de segments, tels que détectabilité
	(-)	ne travaille que sur gros ordinateurs
Gino-F	(+)	grande portabilité pour ordinateurs et appareils gr .
	(-)	texte à deux dimensions seulement
	(-)	prix d'achat 15000 $
GPGS	(+)	ses fonctions d'introduction d'information
	(+)	prix d'achat 800 $
	(-)	portabilité réduite pour des appareils graphiques
IG	(+)	portabilité avec nombreux appareils graphiques
	(+)	prix "free"
	(+)	généralisation des traitements des paramètres, pas de paramètres de primitives
	(-)	génération de points à deux dimensions seulement.
	(-)	référenciel gaucher
Tektronix	(+)	portabilité pour les ordinateurs
	(+)	prix d'achat 775 $
	(-)	pas de segments, pas de paramètres, pas de message d'erreur.
DEC	(+)	possibilité de segmentation à plusieurs niveaux
	(-)	très orienté processeur; lisibilité réduite:
Softw.	(-)	limité à table traçante, pas interactif
Graf. C	(+)	travaille sur UNIX
	(-)	limité à deux dimensions

Dans la suite, nous nous limitons aux bibliothèques présentant une portabilité appréciable et nous éliminons les bibliothèques destinées à un processeur déterminé, souvent fournies par des constructeurs. Seules nous intéressent des bibliothèques qui permettent de changer de support graphique et de travailler avec des terminaux graphiques de performances différentes, par exemple un écran à rafraîchissement et une table traçante, même de constructeurs différents. Les constructeurs ne fournissent pas toujours une table traçante et un écran à rafraîchissement. Il est donc raisonnable de ne pas prendre en considération les bibliothèques qui ne sont adaptés qu'à un matériel. Par conséquent, les bibliothèques suivantes sont éliminés de notre étude: Adage, Calcomp, Tektronix, DEC, Software digital pour traceurs et Graf.C

Dissplа manque de fonctions d'introduction d'information. C'est un handicap pour le travail interactif. Aucun travail interactif ne peut être effectué à l'aide de cette bibliothèque sur aucun appareil graphique.

Les performances graphiques des bibliothèques GCS, GINO-F, GPGS sont sensiblement les mêmes. Les différences se situent dans les attributions des paramètres aux segments ou aux primitives, ainsi que dans les extensions pour des applications.

Gino-F est un peu cher à l'achat pour des performances qui sont atteintes à moindres frais par d'autres bibliothèques, notamment GCS et GPGS. L'utilisation interactive de Gino-F peut aussi poser des problèmes à cause du passage des valeurs des paramètres au dessus des limites des segments.

GCS n'est pas supporté par des miniordinateurs

Enfin, la bibliothèque GPGS est limitée à des opérations graphiques générales, sans extension en vue d'applications telles que les représentations graphiques à échelles linéaires, logarithmiques...etc., et le lissage de courbes.

IG, malgré ses avantages, comporte quelques options peu cohérentes. Le référentiel est gaucher. Pour un utilisateur courant, une telle option nécessite un travail d'adaptation supplémentaire. IG propose des générations de lignes dans les trois dimensions et de points seulement dans deux dimensions.

Ces deux inconvénients: référenciel gaucher et points à deux dimensions seulement pourraient être contournés par une transformation d'un référenciel droitier en un référenciel gaucher et l'addition d'une génération de points à trois dimensions à l'aide d'une ligne en coordonnées relatives de longueur nulle, mais cela représenterait un travail supplémentaire pour l'utilisateur.

A cause de leurs inconvénients, nous éliminons de notre étude aussi les bibliothèques IG et Disspla. Il nous reste donc à choisir parmi les bibliothèques graphiques GCS, Gino-F et GPGS.

A ce stade, notre choix va faire intervenir les appareils graphiques et les ordinateurs de support avec lesquels ces bibliothèques devront travailler. En effet, l'indépendance de la bibliothèque vis-à-vis du support matériel est limitée.

Nous pensons toutefois que vu son prix d'achat élevé et les complications qu'il entraîne lors d'un travail interactif, Gino-F ne doit être choisi que dans le cas où la configuration ne pourrait supporter GCS ou GPGS qui sont beaucoup moins chers. Cependant, Gino-F offre incontestablement la portabilité la plus grande, tant du côté des ordinateurs que du côté des appareils graphiques.

Dans le cas où l'ordinateur de support est un mini-ordinateur, GCS ne peut pas convenir, comme déjà mentionné ci-dessus.

GPGS est la bibliothèque de portabilité la plus faible des trois bibliothèques restantes, du moins vis-à-vis des appareils graphiques. Il s'ensuit qu'un choix concret doit nécessairement tenir compte de l'éventail du matériel qui est appelé à être utilisé avec cette bibliothèque.

2) Choix d'une bibliothèque graphique pour le département de chimie des Facultés.

Le département de chimie dispose d'un PDP 11/45 pouvant travailler sous les systèmes d'exploitation RSX-11M ou Unix (Rich 1). Pour le travail graphique sous RSX-11M, une bibliothèque du constructeur permet de travailler avec un

terminal graphique qui est un écran de rafraîchissement du type GT42. Cette bibliothèque ne supporte pas la table traçante Benson 220.

Aucun des terminaux graphiques n'est implanté dans le système UNIX. GCS n'est pas pris en considération. Il n'est pas supporté par un mini-ordinateur du type PDP 11. Un choix valable est celui de Gino-F. Gino-F supporte les terminaux graphiques du type GT 42 de Dec et les traceurs Benson, mais c'est une solution onéreuse.

Un autre choix serait de prendre GPGS et de se charger de la phase d'implantation. GPGS a déjà été implanté sous sytème UNIX pour divers matériels graphiques, mais pas pour le terminal graphique GT 42 de Dec. Une implantation de GPGS pour le Dec GT 42 pourrait éventuellement être moins onéreuse que le choix de Gino-F. L'examen d'une telle question serait possible dans une extension du présent mémoire.

Une troisième solution serait de prendre les standards proposés par le comité GSPC et d'en tenter l'implantation. Comme l'objectif de compatibilité est primordial, en cas de transfert d'un programme d'application dans une autre installation, les modifications de ce programme doivent être limitées dans la mesure du possible aux appels des sous-routines graphiques. L'implantation des standards devrait permettre cette limitation.

Suiteà ces considérations, nous avons effectué l'implantation d'une bibliothèque sous UNIX. Nous avons choisi les options proposés par les standards du comité GSPC et adopté une terminologie semblable.

III. MISE EN OEUVRE SOUS UNIX

1) Démarches

L'écriture d'une bibliothèque générant un code interprétable par le processeur graphique VT 11 peut être effectuée de deux façons:
-une prmière démarche est de partir des instructions du langage machine du processeur VT 11 et d'écrire, en s'inspirant des possibilités de ces instructions, un ensemble de sous-routines. Le résultat est une bibliothèque orientée processeur, très peu portable.
-une seconde démarche est de partir d'une bibliothèque graphique et d'aller progressivement vers les instructions du processeur graphique. Sur cette démarche peut se greffer l'objectif d'indépendance vis-à-vis de l'appareil. Cet objectif conduit à une double codification: une première, arbitraire au niveau de la bibliothèque et une seconde pour le processeur graphique à l'intérieur du gestionnaire d'un appareil graphique. La second démarche complique sensiblement le gestionnaire. Par contre, la première donne lieu à un gestionnaire assez simple.

Nous avons suivi, une après l'autre, les deux démarches. La première nous a permis d'étudier les possibilités du processeur graphique et de préparer la seconde qui répondait mieux à l'objectif de compatibilité. La description qui va suivre concerne la deuxième démarche. Cette description se compose de celle d'une bibliothèque appelée "libgt" et de celle du gestionnaire pour le terminal graphique GT 42 qui comprend un processeur graphique du type VT 11. Une explication du fonctionnement et des possibilités du processeur graphique VT 11 est donnée en annexe.

2) La bibliothèque libgt

Vu le nombre impressionant d'opérations proposées par les standards, des limites ont été choisies avant d'écrire libgt, une version d'une bibliothèque indépendante du processeur graphique. Les primitives de libgt sont les suivantes: déplacement, ligne, polygone, texte et classe de repères. Chacune de ces primitives est à deux dimensions et peut s'écrire en coordonnées absolues ou relatives. Une segmentation à un seul niveau est obtenue à l'aide des opérations: création, fermeture et effacement. A la fin de l'exécution d'un programme d'application, tous les segments sont effacés.

Les paramètres des primitives sont: type de ligne, type de caractère et intensité. Ces paramètres sont statiques. Les paramètres des segments sont: clignotement, transformation de l'image. Ces paramètres sont dynamiques. Dans les transformations de l'image, seule la translation à été implantée. La rotation et la mise à l'échelle nécessitent une mémorisation du segment que nous n'avons pas réalisée.

Enfin, un localisateur a été implanté et c'est la seule fonction d'introduction d'information réalisée.

Nous avons implanté deux transformations du champ de vision: une fenêtre rectangulaire dont les côtés sont parallèles aux axes de coordonnées et une limitation du champ de vision sur l'écran.

Le dessin est découpé en un certain nombre de segments au moins un. Ces segments constituent le seul niveau de découpage logique du dessin. Les niveaux de dénomination et de modification des segments sont uniques. Les opérations sur les segments sont la création, la fermeture et l'effacement. Un segment peut être effacé à tout moment.

Un segment contient un certain nombre d'appels à des primitives et à des paramètres statiques pour primitives. Ces appels doivent se situer entre la création et la fermeture du segment, c'est-à-dire lorsque le segment est ouvert. Un segment devient visible au fur et à mesure de la génération de ses primitives.

Le dessin imaginé par l'utilisateur est défini dans un système d'axes orthogonaux dont les échelles des coordonnées ne sont soumises à aucune limitation.

L'utilisateur peut, pour obtenir une vision de son dessin, définir une fenêtre rectangulaire. Il choisit les positions des côtés de la fenêtre dans les coordonnées utilisées pour son dessin. A défaut de profiter de cette possibilité, l'utilisateur est mis en présence d'une fenêtre carrée qui est le carré unitaire dans les corrdonnés de l'utilisateur.

Il est possible de ne pas utiliser tout l'écran comme surface de dessin et d'en prendre seulement une partie. Cette partie, appelée champ de vision a une forme rectangulaire et doit être spécifiée en coordonnées normalisées pour l'écran, donc entre zéro et un, la largeur de l'écran étant l'unité. En l'absence d'une spécification du champ de vision, ce dernier est le carré unitaire, ce qui représente tout l'écran.

Le traitement des coordnonnées qui sont les arguments des primitives suit le schéma décrit dans la figure 6.

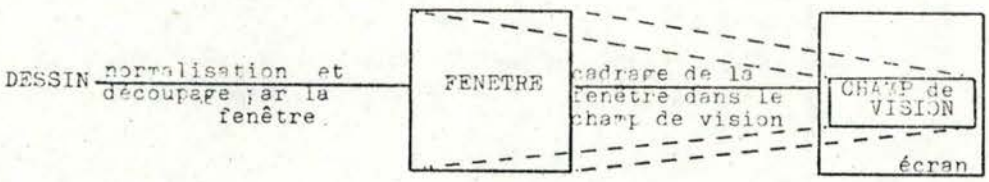


Fig 6.

Les arguments sont d'abord normalisés par rapport à la dimension de la fenêtre. Par cette opération, la fenêtre devient le carré unitaire. La fenêtre découpe une partie visible du dessin. Après ce découpage, cette partie est définie en coordonnées normalisées dans un sytème d'axes dont l'origine est le coin inférieur gauche de la fenêtre. La fenêtre est ensuite projetée dans un champ de vision qui est une partie de l'écran au moyen d'une transformation linéaire de chacun des axes. Cette transformation est homothetique, mais le rapport d'homothétie peut différer d'un axe à l'autre.

Si l'utilisateur veut se servir de ces possibilités, il doit les mentionner, avant de décrire le dessin auquel il veut faire subir ces transformations, donc avant l'ouverture d'un segment.

La bibliothèque assure une protection contre différentes erreurs:

- une invocation d'une primitive doit s'effectuer avec un segment ouvert.
- une création de segment ne peut avoir lieu, si un segment est déjà ouvert.
- une fermeture de segment ne peut avoir lieu, si aucun segment n'est ouvert.
- des paramètres invalides.
- des limites de fenêtres et de champs de vision erronées.

Le programme d'application est averti de chaque erreur et, selon la gravité de l'erreur, est laissé intact, est modifié ou même simplement arrêté.

La bibliothèque codifie chaque appel pour le gestionnaire et lui envoie, en plus, l'information nécessaire, lui permettant de générer le code graphique pour le processeur VT 11.

3) Le gestionnaire pour le périphérique GT 42.

Un gestionnaire pour périphérique est une partie d'un système d'exploitation spécifique à un périphérique déterminé. Sous UNIX, un périphérique est traité comme un fichier et par conséquent, il est permis d'utiliser les opérations ouverture, lecture, écriture, fermeture, comme sur les fichiers normaux (Rich). Une ouverture d'un périphérique est exécuté comme un appel d'entrée-sortie particulier. Le système d'exploitation aiguille l'exécution de cet appel vers une routine du gestionnaire. Ainsi, à l'ouverture du GT 42, la routine "gtopen" va être exécutée.

Le processeur graphique VT 11 fait partie du terminal graphique GT 42. Il dispose de quatre registres et de trois vecteurs d'interruption qui renvoient à des routines d'interruption du gestionnaire.

Le gestionnaire GT 42 est composé de sept routines: quatre correspondant aux opérations sur les fichiers et trois aux interruptions. Les routines d'ouverture et de fermeture servent uniquement à l'initialisation et à la clôture d'un dessin. La routine de lecture sert, conjointement avec la routine d'interruption pour le crayon lumineux, à envoyer à l'utilisateur la position pointée par le crayon lumineux. La gestion du dessin s'effectue à l'aide de la routine d'écriture, de la routine d'interruption du processeur central par le processeur graphique, et de la routine d'interruption sur caractères illégaux.

4) Gestion de la mémoire.

Une des premières questions auxquelles nous avons été confrontées était de savoir où placer l'ensemble des instructions graphiques que nous appelons fichier graphique. Le processeur graphique VT est connecté sur l'Unibus et est un appareil à accès direct sur la mémoire. Il travaille à l'aide d'un compteur programme qui lui est propre et qui contient l'adresse de l'instruction graphique à exécuter. Cette adresse est une adresse physique.

Une possibilité est de laisser le fichier graphique dans la zone utilisateur (figure 7a); une autre est de le placer dans le système d'exploitation (figure 7b); une troisième est de gérer dynamiquement la zone de mémoire adjacente au système d'exploitation (figure 7c)

Fichier graphique en zone utilisateur

Cette solution pose des problèmes d'adressage. L'utilisateur travaille en adresses relatives par rapport au début de zone de mémoire qui lui est allouée, tandis que le processeur graphique utilise des adresses absolues ou physiques. Le fichier graphique risque de se trouver dans la partie de la la mémoire que le processeur graphique ne peut plus atteindre. Pour échapper à cet inconvénient, on pourrait s'arranger pour localiser la zone de mémoire d'un utilisateur travaillant avec le processeur graphique dans la zone accessible par ce

dernier, c'est-à-dire dans les 64 kBytes du début de la mémoire. Un tel arrangement nécessiterait une condition très sévère: l'utilisateur ne pourrait pas être éjecté de la mémoire avant la fin de l'exécution d'un programme.

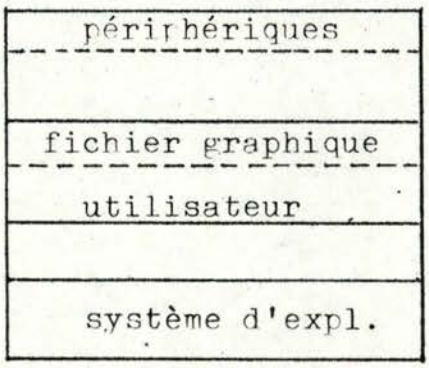


Fig. 7a

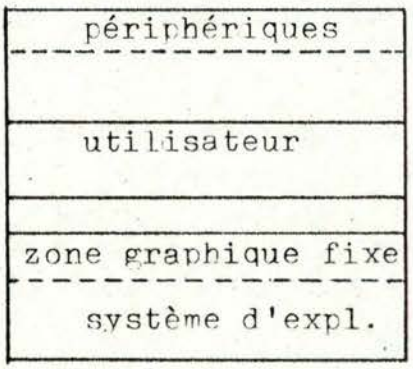


Fig.7b

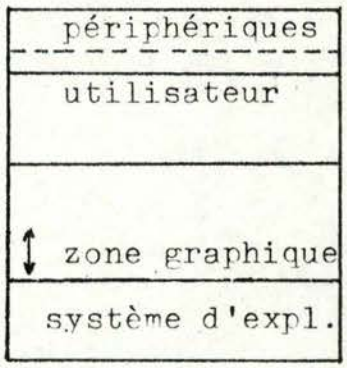


Fig. 7c

Fichier graphique interne au sytème d'exploitation

Cette deuxième solution n'est pas optimale. Un utilisateur moyen qui travaille en graphique ne se sert du système graphique qu'une partie de son temps de connexion. Cette partie du temps de connexion où le système travaille en graphique représente un faible pourcentage. Bloquer une zone de mémoire de taille non négligeable est un handicap et un gaspillage de ressources dont il faut bien être conscient.

Un autre problème surgit, lorsqu'un utilisateur du système graphique crée un fichier graphique très important. Dans ce cas, la zone de mémoire doit être étendue.

Gestion dynamique de la mémoire

Cette troisième solution où l'on gère dynamiquement la zone graphique de la mémoire présente le plus d'intérêt. Si l'utilisateur ne travaille pas en graphique, aucune place ou très peu de la mémoire reste bloquée et vide. Si ensuite le fichier graphique apparaît trop grand, la zone peut être étendue.

La troisième solution est la solution conseillée. Nous souhaitions y arriver, mais pour des questions de temps, nous avons dû y renoncer.

Le gestionnaire actuel a été écrit pour la seconde solution avec une zone de mémoire de taille fixe à l'intérieur du système d'exploitation. Une amélioration serait de passer à la troisième solution. Ce pourrait être une extension de ce travail.

5) Génération du code graphique.

Le gestionnaire est capable de générer le code graphique à l'aide des informations envoyées par la bibliothèque libgt. Les codes graphiques envoyés par la bibliothèque libgt sont transformés en réels codes graphiques. Les coordonnées sont transformées en coordonnées écran, c'est-à-dire de 0 à 1023 (10 bits) par simple multiplication.

La génération du code graphique est réalisée suivant le schéma de la figure 8 qui complète la figure 6 ci-dessus.

Fig. 8

La génération du code graphique s'effectue à l'aide d'une suite d'opérations élémentaires (somme logique) entre des codes opératoires et des paramètres.

Une description détaillée des programmes, ainsi que du processeur graphique et des instructions est donnée en annexe. Comme C est le langage dans lequel UNIX est écrit (Rich 2), tous les programmes sont écrits en C.

6) Application graphique: programme CATGT

L'équipement du département de chimie ne dispose pas de terminaux alphanumériques majuscules et minuscules. Or, l'écriture d'un programme en C utilise les majuscules et les minuscules. Nous avons écrit un programme, appelé CATGT, dans le but d'utiliser le GT 42 comme terminal d'affichage.

Le programme CATGT lit un argument qui est un nom de fichier. Il subdivise le fichier en pages de 32 lignes et passe séquentiellement d'une page à la suivante. Il peut être abandonné après l'affichage de chaque page. Une utilisation de ce programme a permis de lire majuscules et minuscules et notamment de visualiser des erreurs d'écriture comme ADdR pour ADDR que les terminaux alphanumériques du département de chimie ne montrent pas.

Certaines transformations pourraient conduire à donner au programme CATGT un statut de commande, analogue à la command CAT. Faute de temps, cette transformation n'a pas pu être envisagée.

Les descriptions du programme CATGT, de la bibliothèque libgt, et du gestionnaire du GT 42 se trouvent en annexe, aussi à l'intérieur des programmes sources, dont les noms sont respectivement CATGT.C, libgt.C, GT 42.C.

ANNEXES

Annexe 1 - Tableaux de comparaison de bibliothèques graphiques

Les tableaux de comparaison ci-dessous sont constitués d'une reproduction des tableaux de comparaison de bibliothèques graphiques du comité GSPC et de quelques tableaux complémentaires dressés au cours du présent travail.

Dans ce qui suit, nous commentons brièvement l'ordonnance des tableaux et leur contenu, en mettant entre parenthèses les mots de langue anglaise différant sensiblement des termes français utilisés dans le présent travail.

Le tableau I-1 concerne les fonctions d'introduction d'information (input functions), dans lesquels on distingue les fonctions d'introduction à caractère graphique (graphical), le texte et les clefs logiques (button). Le titre: sélection (picking) désigne des objets définis par l'utilisateur, par exemple un segment ou une primitive dans un segment. Dans la colonne "menu" est mentionné si l'information alphanumérique est renvoyée par la sélection, et dans la colonne "item" le niveau d'identification que la sélection est capable de reconnaître, ainsi que l'information s'y rapportant, comme le nom de l'image, un numéro d'élément graphique, une adresse d'un élément pointé. Le localisateur (locator) sert à renseigner une position et/ou une orientation dans l'espace de l'écran; dans cette colonne sont mentionnés les dimensions supportées et les types de coordonnées. Un évaluateur (valuator) renvoie à l'utilisateur une valeur scalaire; dans cette colonne sont mentionnés les types de coordonnées et les dimensions de cette fonction d'introduction d'information. La colonne "appareils d'entrée"(input devices) contient la liste des appareils d'introduction d'information supportés par la bibliothèque graphique. Dans la colonne"texte" figurent les types de caractères qui sont admis comme fonctions d'introduction. Dans la colonne "button" est mentionné si la bibliothèque supporte des clefs logiques. Voir aussi Wallace (Wall 76)

Les taleaux I-2, I-3, I-4a et I-4b se rapportent aux fonctions de sortie: les transformations, les systèmes de coordonnées,la segmentation de l'image et les fonctions primitives.

Une première partie du tableau I-2 concerne les transformations de l'objet(object transforms). Ici un objet est considéré comme une entité graphique et peut être une primitive ou une collection de primitives. Les trois premières colonnes spécifient les dimensions dans lesquelles s'effectuent les transformations de "rotation","mise à l'échelle" (scaling), et "translation". La quatrième colonne mentionne la possibilité d'une "autre" transformation, telle que le cisaillement et les dimensions dans lesquelles s'effectue cette transformation.

Une deuxième partie du tableau I-2 concerne les transformations de vision. La colonne "3D projection" fournit les types de projection supportés. La colonne "découpe de fenêtre"(windowing) mentionne les dimensions dans lesquelles une fenêtre peut être découpée. Les colonnes "champ de vision" (viewporting) et "champ de vision opaque" (shielding) indiquent, si ces transformations sont possibles. La dernière colonne du tableau I-2 "manipulation des transformations" mentionne les possibilités de mémoriser des transformations, de les retrouver et de leur faire subir une concaténation.

Une première partie du tableau I-3 concerne les systèmes de coordonnées. On mentionne, dans les colonnes "coordonnées utilisateur"(world) et"coordonnées de l'appareil"(device), les limites des coordonnées et les unités utilisées. Dans la colonne "orientation" figure l'orientation droitière ou gauchère des réferenciels.

La deuxième partie du tableau I-3 concerne la segmentation de l'image (picture segments). Rappelons qu'un segment est une portion du dessin qui en général correspond à l'image d'un objet et qui possède un nom. Dans la colonne "opérations", la liste des opérations sur le segment est donnée. Dans la colonne "paramètre"(attribute), les paramètres pour segments sont donnés.

Les deux tableaux I-4a et I-4b concernent les fonctions primitives. Dans la colonne "paramètres" (attribute) figure une liste de paramètres pour primitives. Les colonnes suivantes donnent les possibilités de "positionnement et génération de points", de"tracer des lignes"(straight lines)

de"tracer des courbes"(curved lines), de "représentation de surfaces", de"repérage"(marker), d'affichage de "texte et les formats pour "l'affichage de nombres"(numeric output).

Le tableau I-5 des fonctions de commande et de contrôle (control) décrit dans la colonne "initialisation" comment la bibliothèque graphique est initialisée, par exemple l'initialisation d'un appareil, la réservation de places de mémoire. Dans la colonne des "caractéristiques de l'appareil"(device caracteristics) figurent des opérations de sélection et de choix pour les appareils d'affichage et les outils d'introduction d'information. Dans la colonne de "détection d'erreur" (error reporting) est mentionnée le type d'erreur détecté. La colonne "routine de clôture"(termination routine) indique s'il existe une phase de clôture.

Le tableau I-6 concerne l'environnement d'implantation et les extensions en vue d'applications. Dans la colonne "implementation environment" est mentionné le langage de codification de la bibliothèque. Dans trois colonnes relatives à des opérations de "représentation de graphes"(graphing operations), de"lissage de courbes"(curve fitting) et d'affichage "d'informations alphanumériques en sorties spéciales" (positioned text) sont données les possibilités des bibliothèques dans chacune des rubriques.

Un dernier tableau I-7, "divers"(miscellaneous information) contient des informations sur le "nombre de routines", les "appareils graphiques supportés"(graphics hardware supported), les"ordinateurs avec lesquels ces bibliothèques sont compatibles" (host computers),le nombre"d'installations" existantes et une estimation du"coût"(approximate cost)

Remarques

(1) Les "graphic software packages", repris dans les tableaux I-1 à I-7 ont été considérés comme bibliothèques graphiques et ce dans tout le mémoire. Toutefois, ADAGE pourrait être un langage graphique. En effet, il est fait mention dans le tableau I-6 "implementation environment" du compilateur Fortran à extensions graphiques, ce qui indique qu'il pourrait s'agir d'un langage.

(2) Le tableau I-3 concernant les paramètres pour segments ne mentionne pas de tels paramètres pour la bibliothèque graphique GCS. Nous signalons que Foley (Foley 76) donne la visibilité d'un segment comme paramètre pour segments en GCS.

	GRAPHICAL					TEXT	BUTTON
	PICKING		LOCATOR	VALUATOR	INPUT DEVICES		
	MENU	ITEM					
ADAGE	Returns menu index	One level, address returned	2D, device units	1D, device units	Light pen, tablet, joystick/mouse, ball, control dials	FORTRAN extensions	X
CALCOMP							
DISSPLA							
GCS	Returns menu index		2D, device or world units		Cursor, joystick, light pen, tablet, mouse, trackball	String of ASCII characters	
GINO-F		One level segment no. returned	2D, device units		Installation dep.	String of ASCII characters	X
GPGS		n levels, segment no. returned	2D or 3D, device units	1D, device units	Clock, light pen, 1, 2, and 3D scalar devices	String of characters in A1 format	X
IG		n levels, item no. returned	2D, world units		Installation dep.		Inst. dep.
TEKTRONIX			2D, device or world units		Thumbwheels or joystick	String of ASCII characters	

TABLE I-1
INPUT FUNCTIONS

	OBJECT TRANSFORMS				VIEWING TRANSFORMS				TRANSFORM
	ROTATION	SCALING	TRANSLATION	OTHER	3D PROJECTION	WINDOWING	VIEWPORTING	SHIELDING	MANIPULATION
ADAGE	2D 3D	2D 3D	2D 3D			2D 3D	Limited		Save, restore concatenate, nesting
CALCOMP		2D				2D (optional)			
DISSPLA	2D	2D	2D	User supplied	Axon.	2D 3D	X	X	
GCS	2D 3D	2D 3D	2D 3D		Perspective Ortho.	2D 3D	X		Save and restore
GINO-F	2D 3D	2D 3D	2D 3D	2D & 3D shear	Perspective Axon.	2D 3D (limited)	Limited		Save, restore, concatenate, copy
GPGS	2D 3D	2D 3D	2D 3D	2D & 3D shear and user supl	Perspective Ortho. Axon.	2D 3D	X		Save, restore, concatenate
IG	2D 3D	2D 3D	2D 3D		Perspective Ortho.	2D	X		Concatenate
TEKTRONIX	2D	2D				2D	X		

TABLE I-2
TRANSFORMATIONS

	COORDINATE SYSTEMS			PICTURE SEGMENTS	
	WORLD	DEVICE	ORIENTATION	OPERATIONS	ATTRIBUTES
ADAGE	Fractional numbers from -1 to +1	14-bit fractional numbers	Right	Open, invoke, transform, modify, close	Specified for primitives only
CALCOMP	Inches or centimeters extrema given by plotter size	Inches or centimeters extrema given by plotter size	Right		
DISSPLA	Based on 8 1/2" x 11" page size, which may be reset	User does not deal with device units	Right		Specified for primitives only
GCS	Range of floating point numbers	Raster units, inches, centimeters, percent, or character units	Right or left	Open, invoke, transform, delete, close	Specified for primitives only
GINO-F	Range of floating point numbers	Millimeters	Right	Open, close, invoke, extend, delete, copy, rename	Detectability, visibility, blinking, intensity, color
GPGS	Range of floating point numbers or integers	Unit square	Right	Open, invoke, copy, close	Visibility, intensity, blinking, color, depth cueing, detectability
IG	Floating point numbers from -1 to +1 may be reset	Unit square	Left	Open, extend, invoke, close	Intensity, hue, text size and font
TEKTRONIX	Range of floating point numbers	1024 x 1024 raster units	Right		

TABLE I-3

COORDINATE SYSTEMS AND PICTURE SEGMENTS

	PRIMITIVES				
	ATTRIBUTES	POSITION AND POINT GENERATION	STRAIGHT LINES	CURVED LINES	SURFACES
ADAGE	Solid, dashed lines and text orientation	2D and 3D, absolute	2D and 3D, absolute	2D, circles and arcs	
CALCOMP		2D, absolute	2D, absolute and polyline	2D, curves and arcs, (Functional Software)	Contour and surface packages (Application Software)
DISSPLA	Solid, four dashed, twelve arrowhead, and user-defined lines	2D and 3D, absolute and multipoint plots	2D and 3D absolute and polyline	2D and 3D, Several curve routines	Three methods for displaying surfaces
GCS	Color, intensity, blink dashed lines, line terminators, text attributes	2D and 3D, absolute and relative	2D and 3D, absolute and relative	2D, circle, arc, and conic	
GINO-F	Solid, six dashed lines of specified color, width and type	2D and 3D, absolute and relative	2D and 3D, absolute and relative	2D and 3D, absolute and relative arcs	
GPGS	Six line types	2D and 3D, absolute and relative	2D and 3D, absolute and relative	2D and 3D absolute and relative arcs	
IG		2D positioning only, absolute and relative	2D and 3D, absolute and relative and polyline		
TEKTRONIX		2D, absolute and relative	2D, absolute and relative		

TABLE I-4a
PRIMITIVES

	PRIMITIVES			
	MARKER	TEXT	NUMERIC OUTPUT	OTHER
ADAGE		2D transformable and 3D, one font, horizontal and vertical	FORTRAN extensions	Raster output mode
CALCOMP	15 centered markers	2D transformable, one font	F, I formats	
DISSPLA	15 centered markers and user defined	2D and 3D transformable, thirteen alphabets and fonts	E, F, I formats	
GCS	Several markers and user defined	2D and 3D hardware and software transformable, two fonts	E, F, I formats	
GINO-F	8 markers	2D hardware and software transformable, two fonts	E, F, I formats	
GPGS	256 transformable markers	2D and 3D hardware and software transformable, two fonts	E, F, I formats	
IG	Several markers	2D hardware or software transformable, one font	E, F, I formats	
TEKTRONIX	7 markers (AGII)	2D hardware, one font, four sizes	E, F, I formats (AGII)	

TABLE I-4b
PRIMITIVES (cont.)

	CONTROL			
	INITIALIZATION	DEVICE CHARACTERISTICS	ERROR REPORTING	TERMINATION ROUTINE
ADAGE	None required	Set at system generation	Reports only display buffer overflow	
CALCOMP	Single call to reserve buffer	Select a pen	Must be provided by user	X
DISSPLA	Call to initialize device Call to initialize common area	Select a "virtual" device	Complete error checking and reporting	X
GCS	Selects device and sets default	Set terminal mode, erase, flush buffer, manipulate status area, query status area	Complete error reporting to user	X
GINO-F	Select device	Sets device, plotter size, heading, speed, pen, erase, and query device	Error reporting to user	X
GPGS	Initialize device, initialize buffer	Select device, clear device, release device	Complete error checking and reporting, four severity levels	X
IG	Creates picture buffer, device selected by JCL	Select device	Error reporting to user	
TEKTRONIX	Select terminal type and communication speed	Set terminal mode, erase, hardcopy, buffer type, flush buffer	None	X

TABLE I-5
CONTROL

	IMPLEMENTATION ENVIRONMENT	APPLICATION EXTENSIONS		
		GRAPHING OPERATIONS	CURVE FITTING	POSITIONED TEXT
ADAGE	FORTRAN compiler with graphical extensions			Allows special menu generation text
CALCOMP	Coded in FORTRAN for larger machines, some assembler	Continuous, discrete, and 3D graphs supported by higher level	Spline and polynomial	
DISSPLA	Coded in FORTRAN	Continuous, discrete, and 3D graphs, shading provided	Spline, polynomial, linear interpolation	Allows many text display options
GCS	Coded in FORTRAN	Continuous and discrete graphs supported, 3D line plots	Spline, polynomial linear interpolation	Allows text windows
GINO-F	Coded in FORTRAN	Continuous and discrete graphs supported		
GPGS	ANSI FORTRAN, 360 assembler, PDP11 macro			
IG	Assembly language			
TEKTRONIX	Coded in FORTRAN	Continuous and discrete graphs supported by AGII		Allows text windows

TABLE I-6
IMPLEMENTATION ENVIRONMENT AND APPLICATION EXTENSIONS

	NUMBER OF ROUTINES	GRAPHICS HARDWARE SUPPORTED	HOST COMPUTERS	NUMBER OF INSTALLATIONS	APPROXIMATE COST
ADAGE	IMAGE - 80 extensions GRAFX- 145 ops.	AGT/10, AGT/30, AGT/50, AGT/300 series	Adage DPR2 or DPR4 processors	50+	Supplied with hard-ware
CALCOMP	Basic - 10 Functional - 43 Applications - 9 packages	All CalComp plotters and COM units	All major mainframes and several mini-computers	Several thousand	Basic soft-ware $500-$1,500
DISSPLA	206 routines	Calcomp, CIL, Xynetics, EAI, Tek-tronix, HP, Zeta, Broomall, & Ger-ber plotters; Tektronix, Computek, HI, VG, Adage, IBM CRTs; Calcomp, III, and SC microfilm	Burroughs B6700, CDC 6000, 7000; IBM 360/370; DEC 10,20 UNIVAC 1100; XDS SIGMA9, Honeywell 6000	100	$22,000
GCS	97 routines	CalComp, Gould, and TSP plot-ters; Computek and Tektronix storage tubes; IMLAC PDS1D and PDS4; line printer terminals	Burroughs 5700/6700/7700; CDC 6000; DEC 10; Honeywell 6000; IBM 360/370; UNIVAC 1100	118	Free
GINO-F	231 routines	Benson, Coradi, CalComp, ICL, Ger-ber, Kongsberg, Versatec, Zeta, CIL, Laserscan plotters; Tektronix 4000 series; IMLAC, VG, DEC re-fresh; CADC raster device	Burroughs B6700, CDC 6600, DG Nova, DEC 10,11, GEC, Honeywell 6000, IBM 370, ICL, Phillips 1400, Prime 300, UNIVAC 1100, Varian 620L, Xerox Sigma	120	$15,000
GPGS	116 routines	Tektronix 4000 series VG CalComp plotters	360/370 series PDP 11 UNIVAC 1108, PDP15, Harris mini	40+	$800
IG	35 routines	CalComp, HP, inkjet plotters; Tek-tronix, Computek, PEP storage tubes; Adage, DEC, IBM refresh de-vices; FR80, Plato IV, and line printers	IBM 360/370 AMDAHL 470	8	Free
TEKTRONIX	TCS - 84 AGII - 86	Tektronix 4000 series	All major mainframes and several mini-computers	2000+	TCS - $775 AGII - $775

TABLE I-7
MISCELLANEOUS INFORMATION

	GRPHICAL					TEXT	BUTTON
	PICKING		LOCATOR	VALUATOR	INPUT DEVICES		
	MENU	ITEM					
DEC		Renvoie le n° de segment	2D unités de l'appareil		Crayon lumineux		
Software pour traceur							
Graf.c			2D unités de l'appareil		Crayon lumineux		

	OBJECT TRANSFORMS			VIEUWING TRANSFORMS	
	ROTATION	SCALING	TRANSLATION	WINDOWING	VIEUWPORTING
DEC		2D	2D	En rapport avec la mise à l'échelle	
Software pour traceur		2D		2D	2D
Graf.c			2D		

Note: les colonnes non reprises sont vides.

	COORDINATE SYSTEMS			PICTURE SEGMENTS	
	WORLD	DEVICE	ORIEN-TATION	OPERATIONS	ATTRIBUTES
DEC	Intervalle de nombres réels	1024 x 1024	Droitière	Ouverture Fermeture Appel Effacement	Visibilité
Software pour traceur	Intervalle de nombres réels	Au maximum 500 x 73	Droitière		
Graf.c	1024 x 1024	1024 x 1024	Droitière	Ouverture Fermeture Effacement Remplacement	Visibilité Clignotement

	PRIMITIVES				
	ATTRIBUTES	POSITION AND POINT GENERATION	STRAIGHT LINES	CURVED LINES	SURFACES
DEC	intensité, clignotement, type de ligne, action du crayon lumineux.	2D absolu	2D relatif		
Software pour traceur	type de ligne, type de caractere.		2D absolu, relatif	2D arc, cercle	
Graf.c		2D absolu	2D absolu, relatif.		

	PRIMITIVES			
	MARKER	TEXT	NUMERIC OUTPUT	OTHER
DEC		2D horizontal	format Fortran	
Software pour traceur		2D orientable	format Fortran	tracés speciaux
Graf.c		2D horizontal		

	CONTROL			
	INITIALISATION	DEVICE CHARACTERISTCS	ERROR REPORTING	TERMINATION
DEC	Initialisation du tampon			
Software pour traceur	Initialisation de variables		Envoi de messages en cas d'erreur	X
Graf.c	Initialisation pour positionner le mode du terminal		Seulement pour une position mal reçue du crayon lumineux	X

	IMPLEMENTATION ENVIRONMENT	APPLICATION EXTENSIONS		
		GRAPHING OPERATIONS	CURVE FITTING	POSITIONNED TEXT
DEC	Codé en Fortran	Tracé de courbes à 2D		Generation spéciale de menus
Software pour traceur	Codé en Fortran		Interpolation linéaire	Génération de titres
Graf.c	Codé en C			

Annexe 3.

Tableaux de comparaison de dénomination et modifications d'un segment

Ces tableaux sont partiellement extraits de l'article de Foley et partiellement dressés par nous. Ils reprennent, pour les bibliothèques comparées précédemment, le nombre de niveaux de dénomination et le nombre de niveaux de modification.

Remarques

(1) Pour certaines bibliothèques, nous n'avons pas l'information nécessaire pour remplir avec certitude les tableaux. Un point d'interrogation est ajouté aux cadres concernés.

(2) Un seul niveau de segmentation ne signifie pas un seul niveau d'inclusion. Il est possible de n'avoir qu'un seul niveau de segmentation et de faire référence de l'intérieur d'un segment à un autre segment, par exemple par une opération de copie.

	Niveaux de dénomination	Niveaux de modification
Adage	? au moins un: segment	? au moins un: segment
Calcomp	zéro	zéro
Dissp1a	zéro	zéro
GCS	un: cadre (frame)	un: cadre
Gino-F	un: segment	un: segment
GPGS	plusieurs: bloc segment d'image primitive copie d'un segment d'image	un: segment d'image
IG	? au moins un: segment	? au moins un: segment
Tektronix	zéro	zéro
DEC	un: segment (subpicture)	un: segment
Graf.c	un: cadre	un: cadre
Software digital	zéro	zéro

Annexe 3.

LE PROCESSEUR GRAPHIQUE VT11

Fonctionnement

Le processeur graphique VT11 est un processeur qui peut se connecter comme un périphérique quelconque sur l'Unibus d'un PDP 11. Le processeur VT11 est un appareil à acces direct sur la mémoire et capable de dérouler son programme indépendamment du processeur du PDP 11, si toutefois l'acces à l'Unibus lui est accordé par ce dernier.(VT11)

Le processeur VT11 émet des demandes d'accès NPR (non processor request) à l'Unibus. Lorsque une demande est agréée, le VT11 extrait un mot de programme de la mémoire, le décode et l'exécute. Il émet ensuite une autre demande d'accès NPR à l'Unibus pour le mot suivant.

Le processeur VT11 peut aussi envoyer des demandes d'interruption au processeur central lorsqu'il rencontre des caractères illégaux ou lorsque la mémoire reste sans réponse. Un programme graphique peut autoriser le VT11 à envoyer une demande d'interruption au processeur central lorsque il rencontre une instruction stop ou s'il détecte une action du crayon lumineux.

Le processeur VT11 démarre lorsque son compteur programme est garni. Le compteur programme du VT11 est noté DPC (Display Program Counter). Le PDP 11 place l'adresse du DPC sur les lignes des adresses le l'Unibus et la valeur à donner au DPC sur les lignes des données de l'Unibus. Le décodage de l'adresse du DPC met automatiquement le processeur graphique VT 11 en action. Sa première action est de lancer une demande d'accès NPR à l'Unibus.

Instructions et registres du VT11

Le processeur VT11 travaille à l'aide d'un jeu de cinq instructions qui opèrent sur six types différents de données, selon les instructions.
Ces instructions sont les suivantes:

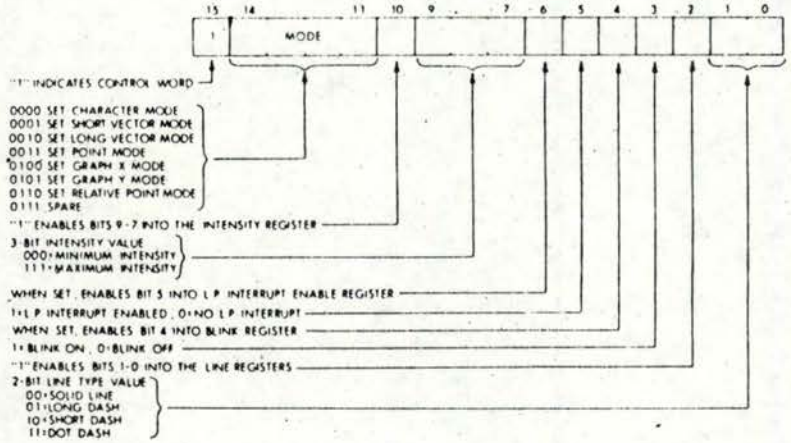
- positionner le mode graphique
- saut à une adresse
- pas d'opération
- positionner des paramètres
- positionner l'incrément pour le tracer de courbes.

Le jeu d'instructions du VT11 offre à l'utilisateur d'intéressantes possibilités graphiques. Toutefois, ces possibilités sont loin d'être complètes. Il manque au VT11 une instruction de saut à une sous-routine. Cette instruction est pourtant très performante et pour parvenir à générer un saut à une sous routine un artifice utilisant une interruption du PDP par le processeur graphique peut être employé. D'autres intructions sont aussi absentes du jeu d'instructions du VT11. Par exemple, des transformations comme la translation, la rotation et la mise à l'échelle. Ces manquements sont moins importants que l'absence d'un saut à une routine.

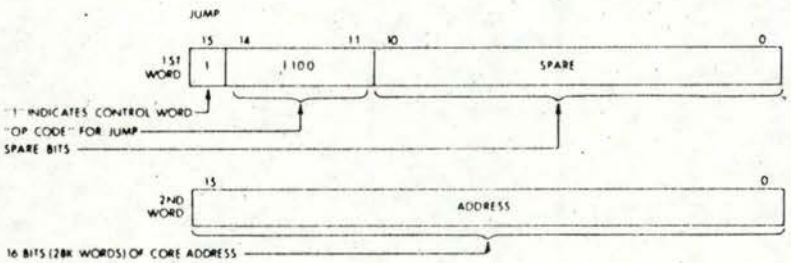
Le processeur VT11 dispose de quatre registres de seize bits. Le premier contient le compteur programme graphique (DPC); c'est le seul accessible en lecture et en écriture. Les trois autres registres, accessibles uniquement en lecture, contiennent l'état d'un certain nombre de bascules et de signaux.

Les pages suivantes contiennent les définitions et les formats des instructions et des registres ainsi qu'une liste des possibilités du processeur VT11.

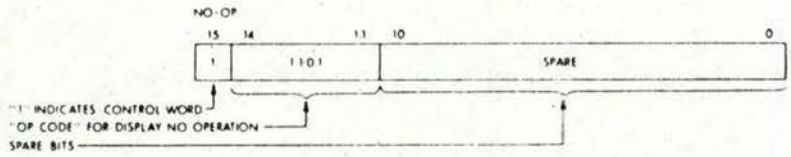
SET GRAPHIC MODE



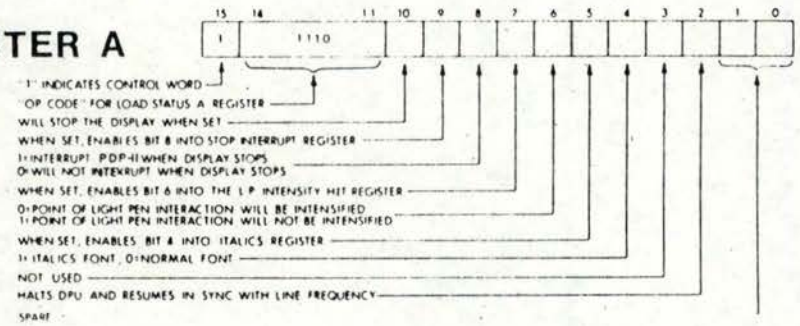
JUMP



NO-OP



LOAD STATUS REGISTER A



LOAD STATUS REGISTER B

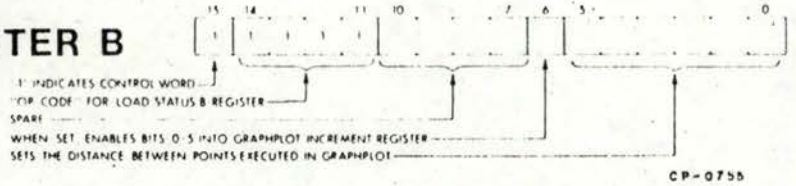
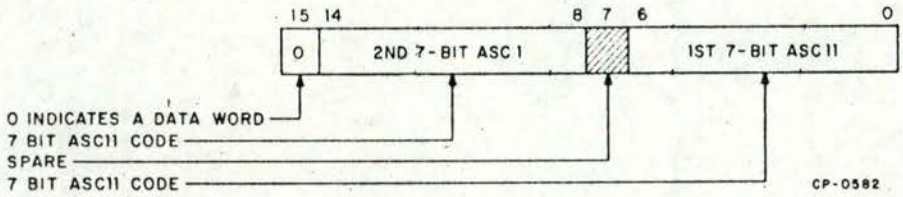
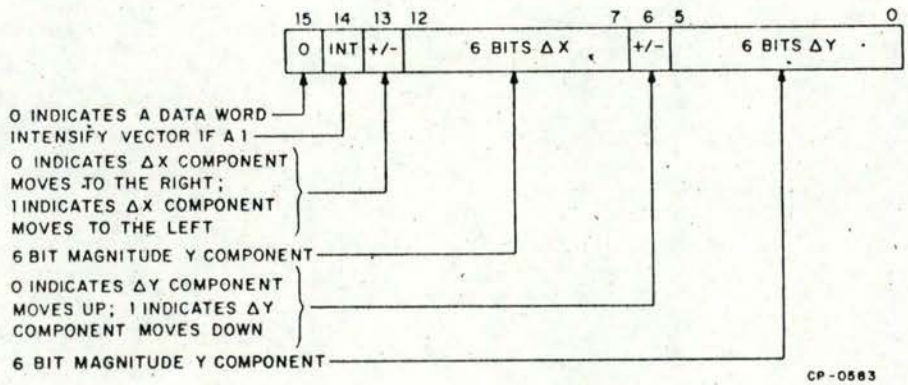


Figure 2-8 Instruction Word Functions

CHARACTER DATA FORMAT- Mode 0000



SHORT VECTOR MODE- Mode 0001



LONG VECTOR DATA FORMAT- 0010

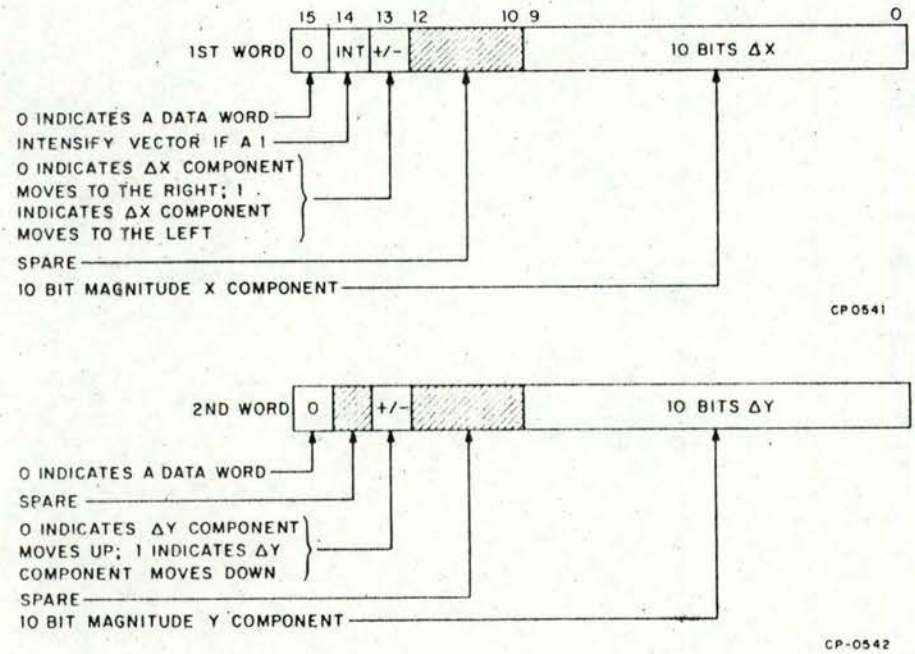
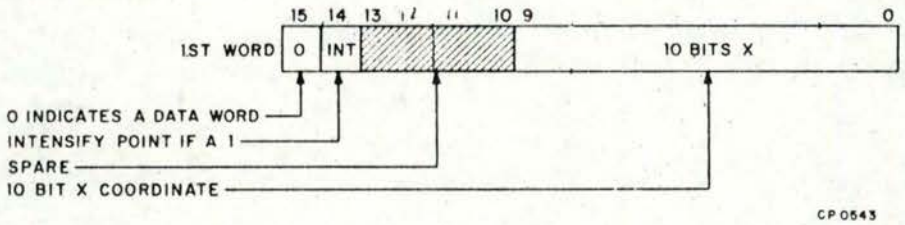
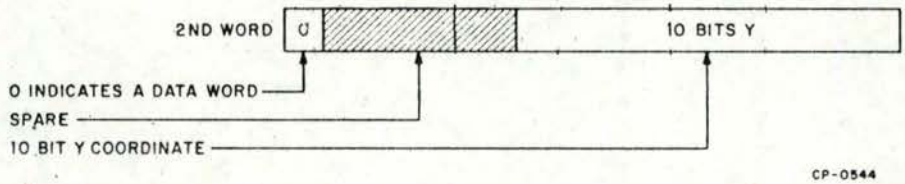


Figure 2-18 Data Word Formats (Sheet 1 of 2)

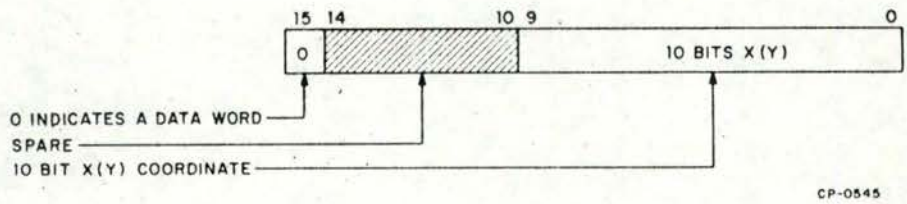
POINT DATA MODE-
Mode 0011





GRAPHPLOT X-
Mode 0100

GRAPHPLOT Y-
Mode 0101



RELATIVE POINT MODE-
Mode 0110

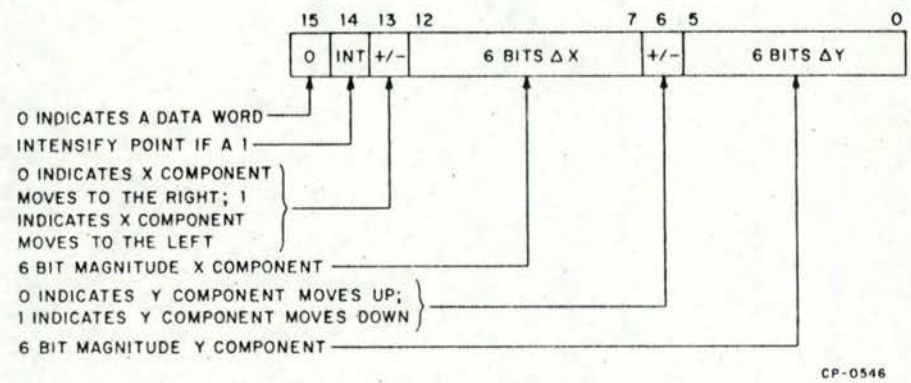


Figure 2-18 Data Word Formats (Sheet 2 of 2)

Instruction Word Length
16 bits

Raster Definition
10 bits

Viewable Area
x = 1024 (1777_8) raster units
y = 768 (1377_8) or 1024 (1777_8) raster units

Paper Size
12 bits

Hardware Blink
Programmable

Hardware Intensity Levels
8

Line Frequency Synchronization
Hardware programmable

Character Font
6 X 8 dot matrix

Characters/Line
73 (85 possible)

Number of Lines
31 on VR14 (29 possible), 42 on VR17 (39 possible)

Character Set
96 ASCII – upper and lower case plus 31 specials (Greek letters, math symbols, etc.) (Refer to Appendix A)

Control Characters
Carriage return
Line feed
Backspace

Italics
Hardware programmable

Line Type
Solid
Long dash
Short dash
Dot-dash

Data Formats
Character (2 char/word)
Short Vector (1 word)
Long Vector (2 words)
Point (2 words)
Relative Point (1 word)
Graphplot x/y (1 word/pt)

DPU Instructions
Set Graphic Modes
Jump
No operation (NOP)
Load Status Register A
Load Status Register B

VT11 Hardware Registers

Register	Unibus Address*	CPU Operation	Contents	Bit
Program Counter	772000	Read/Write	Address of next word in display file	(15:0)
Status Register	772002	Read Only	Stop Flag	(15)
			Mode	(14:11)
			Intensity	(10:8)
			Light Pen Flag	(7)
			Shift Out	(6)
			Edge Indicator	(5)
			Italics	(4)
			Blink	(3)
			Spare (Not Used)	(2)
			Line	(1:0)
X Position & Graphplot Increment	772004	Read Only	X Position	(9:0)
			Graphplot Increment	(15:10)
Y Position & Character	772006	Read Only	Y Position	(9:0)
			Character Register	(15:10)

*The two high order bits are forced to a 1 to assert an MSD = 7.

Annexe 4.

Programme libgt.c

Les programmes qui vont être présentés ci-dessous sont décrits en trois points: la fonction,
les tests effectués,
les sous programmes appelés.

Les primitives

Les primitives de dessin proprement dites ont un comportement général commun. Les routines qui les décrivent
positionnent le code instruction correspondant,
transforment les coordonnées de l'utilisateur en des coordonnées normalisées par rapport à la fenêtre, routines reduc,
mettent à jour la nouvelle position courante newp,
garnissent les variables argx et argy des déplacements relatifs en coordonnées normalisées,
font appel à la routine de découpage appelée clip.
Ces routines renvoient un message et sont abandonnées lorsque elles sont appelées tant qu'aucun segment n'est ouvert.

Pour la description fonctionnelle des routines, nous supposons que la position courante est (xo,yo).

Moveabs(x,y)

La position courante est mise à la valeur (x,y). Cette routine n'a aucun effet visible sur le dessin.

Moverel(dx,dy)

La position courante est mise àla valeur (xo+dx,yo+dy). Cette routine n'a aucun effet visible sur le dessin.

Lineabs(x,y)

Une ligne est tracée de la position courante jusqu'à la position (x,y) qui devient la nouvelle position courante.

Linerel(dx,dy)

Une ligne est tracée de la position courante jusqu'à la position (xo+dx,yo+dy) qui devient la nouvelle position courante.

Polyabs(xarray,yarray,n)

Une sucession de lignes est générée à l'aide des deux tableaux xarray et yarray. La suite des lignes part de la

position courante vers (xarray(1),yarray(1)) pour arriver à la position (xarray(n),yarray(n)) qui devient la nouvelle position courante.

Polyrel(dxarray,dyarray,n)

Une sucession de lignes est générée à l'aide des deux tableaux dxarray et dyarray. La suite des lignes part de la position courante vers (xo+dxarray(1),yo+dyarray(1)) pour arriver à la position (xo+dxarray(1)+..+dxarray(n), yo+dyarray(1)+..+dyarray(n)) qui devient la nouvelle position courante.

Markabs(x,y,n)

Un repère de la classe n est placé au point de coordonnées (x,y) qui devient la nouvelle position courante. Si n ne correspond à aucune classe, la classe du repère est la classe numéro 1.

Markrel(dx,dy,n)

Un repère de la classe n est placé au point de coordonnées (xo+dx,yo+dy) qui devient la nouvelle position courante. Si n ne correspond à aucune classe, la classe du repère est la classe numéro 1.

Une seule primitive de texte permet d'afficher des caractères.

Text(straddr,n)

Cette routine va afficher n caractères,pris à partir de l'adresse straddr, en commençant à la position courante. Si le nombre de caractères dépasse 510, seuls les 510 premiers sont envoyés.Cette routine fait appel à la routine transfer pour le passage des arguments au gestionnaire du gt42.

Plusieurs routines qui ne sont pas accessibles à l'utilisateur, ont été écrites pour généraliser les traite ments. Ces routines ont pour nom: bornes,clip,posit,reducabs, reducrel,vieuwmap.

Bornes()

Le tracé d'une ligne peut conduire la nouvelle position courante hors des limites de la fenêtre choisie. La routine bornes calcule les coordonnées de l'intersection d'une ligne

avec le cadre découpé par la fenêtre. Le calcul est effectué en coordonnées normalisées et les valeurs des coordonnées de l'intersection sont renvoyées dans les variables varx, vary. Cette routine fait appel à la routine posit.

Clip()

La routine clip découpe le dessin de l'utilisateur au travers de la fenêtre. Pour effetuer ce découpage clip, garde mémoire de la position courante. Cette position est réelle ou imaginaire selon que cette position est ou n'est pas dans le cadre de la fenêtre. Elle détermine d'abord la caractéristique réelle ou imaginaire de la nouvelle position courante. Ensuite, en fonction de la caractéristique de la position courante et de celle de la nouvelle position courante, elle effectue un calcul d'intersection avec le cadre de la fenêtre. Enfin, si un champ de vision a été defini, elle fait appel à la routine de cadrage de la fenêtre dans le champ de vision vieuwmap et appelle la routine transfer pour le passage des arguments au gestionnaire gt42.c.

Posit(x,y)

La routine posit teste si les arguments x et y sont intérieurs à la fenêtre. Si oui, elle renvoie une valeur signifiant 'réel' et sinon une valeur 'imaginaire.

Reducabs(x,y)

Reducrel(dx,dy)

Ces routines normalisent x et y (dx et dy) par rapport à la fenêtre. Les valeurs normalisées de x et y (dx et dy) sont placées respectivement dans argx et argy. La normalisation a lieu en coordonnées absolues (reducabs) ou relatives (reducrel).

Vieuwmap(x,y)

Cette routine cadre le fenêtre dans un champ de vision préalablement défini. Les coordonnées "cadrées" sont placées dans argx et argy.

Segmentation

Trois routines en vue de segmenter un dessin sont décrites ci-dessous. Il s'agit des routines de création, de fermeture et d'effacement d'un segment.

Createseg(name)

Cette routine crée un segment dont le nom commence à l'adresse name. 8 caractères sont retenus pour le nom du segment créé. La routine teste d'abord qu'un nombre maximum * de segments n'est pas dépassé, ensuite qu'aucun autre segment n'est ouvert et enfin qu'aucun autre segment ne porte le même nom. La routine donne au segment créé le numéro d'une entrée dans une table qui contient les noms de segments et envoie un ordre de création de segment au gestionnaire gt42 c

Closeseg()

Cette routine cloture le dernier segment ouvert et envoie au gestionnaire du gt42 un ordre de cloture d'un segment.

Deleteseg(name)

Cette routine efface le segment dont le nom commence à l'adresse name. Elle effectue une recherche (routine recherch) dans la table des noms de segments pour retrouver le numéro de segment et envoie au gestionnaire du gt42 un ordre d'effacement accompagné du numéro de segment à effacer.

Les erreurs détectées par ces routines conduisent à l'envoi d'un message et àl'abandon de ces routines.

Recherch(name)

Cette routine recherche le numéro du segment dont le nom commence à l'adresse name dans la table des noms de segments. Si aucun segment ne porte le nom demandé, la valeur -1 est renvoyée.

* fixé à 128 dans notre système.

Paramètres

a) paramètres pour primitives.(Il faut un segment ouvert)

Settyp(val)

Cette routine positionne le type de ligne. 4 types de ligne sont admis. En cas de valeur erronée, l'ancienne valeur est maintenue.

Setchar(val)

Cette routine positionne le type de caratère de texte : normal (0) ou italique (1). En cas de valeur erronée, l'ancienne valeur est maintenue.

Setint(val)

Cette routine positionne l'intensité. L'intensité peut s'echelonner de 0 à 7. En cas de valeur erronée, l'ancienne valeur est maintenue.

A l'entrée d'un segment, les valeurs de ces paramètres sont les suivantes: type de ligne 0 (ligne continue),
type de caractère normal,
intensité moyenne 4.

b) paramètres pour segments.

Setblink(name,val)

Cette routine fait clignoter (val = 1) ou arrêter le clignotement(val = 0) d'un segment dont le nom commence à l'adresse name. En cas d'erreur sur le nom de segment ou sur la valeur val, aucune modification n'est effectuée.

Translate(name,tx,ty)

Cette routine translate le segment dont le nom commence à l'adresse name de tx selon l'axe des x et de ty selon l'axe des y. En cas d'erreur sur le nom de segment, aucune modification n'est effectuée. Les valeurs tx et ty suivent la phase de découpage par la routine clip.

Fonction d'introduction

Readlocator(x,y)

Cette routine reçoit les valeurs des coordonnées du point visé par le localisateur dans les arguments x et y.Elle se charge de régler le problème d'autorisation de pointer. La valeur est renvoyée en coordonnées utilisateur.

Transformation de la vision

Window(xm,xmm,ym,ymm)

Cette routine positionne les limites d'une fenêtre sur le dessin. Les arguments sont donnés dans le système de coordonnées de l'utilisateur. Si ces arguments sont invalides, la fenêtre garde la forme qu'elle avait avant l'appel de cette routine. Par défaut, la fenêtre correspond à l'écran et ses limites sont 0 et 1 sur chacun des axes. Les arguments sont invalides si la coordonnée inférieure xm ou ym n'est pas inférieure à la coordonnée supérieure xmm ou respectivement ymm.

Vieuwport(xm,xmm,ym,ymm)

Cette routine positionne les limites d'un champ de vision sur l'écran. Ces limites sont données en coordonnées normalisées. Si les arguments sont invalides, le champ de vision garde la forme et la position qu'il occupait avant l'appel de cette routine. Par défaut, le champ de vision correspond à l'écran entier et ses limites sont 0 et 1 sur chacun des axes axes. Les arguments sont invalides si leurs valeurs ne sont pas comprises entre 0 et 1 ou si la coordonnée inférieure xm ou ym n'est pas inférieure à la coordonnée supérieure xmm ou respectivement ymm.

Commandes: initialisation et clôture

Opengt()

Cette routine se charge de l'ouverture du périphérique gt42 en vue d'initialiser le gestionnaire du gt42. Elle positionne des valeurs par défaut pour différents paramètres, initialise la numérotation des segments à zéro et considère tout segment fermé.

Closegt()

Cette routine ferme le périphérique gt42 et arrête l'affichage.

Routine de transfert

Transfer(adr)

Cette routine transfère au gestionnaire du gt42 un argument situé à l'adresse adr.

Annexe 5.

Programme gt42.c

Le programme gt42.c est le programme de gestion du périphérique graphique gt42. Il génère le code graphique à partir des ordres reçus de la bibliothèque libgt.c au travers de la routine transfer de libgt.c et de la routine gtwrite. Une zone de mémoire dont la taille actuelle est de 4 kBytes est réservée pour recevoir le programme graphique interprétable par le processeur graphique VT11.

Gtopen()

Cette routine initialise le gestionnaire. Elle positionne des variables de travail ainsi que le pointeur vers le début de la zone pour le fichier graphique.

Gtread()

Cette routine de lecture sert à renvoyer la position pointée par le crayon lumineux. Elle commence par s'endormir. A son réveil, elle envoie les valeurs des coordonnées du point visé par le crayon lumineux. Le réveil de cette routine est réalisé par la routine penint.

Gtwrite()

Cette routine génère le code graphique. Elle reçoit un argument en provenance de libgt.c et est capable de distinguer une instruction d'une donnée.

Dans le cas d'une instruction, l'argument reçu est entier. Les 8bits de droite contiennent un paramètre et les 8 bits de gauche sont significatifs de l'instruction. En fonction de l'intruction, les traitements diffèrent.Pour les primitives, le véritable code graphique est positionné et un compteur d'arguments de données est initialisé. Pour les segments, les options par défaut sont générées lors de la création et repositionnés à la cloture. Pour ces traitements, un tableau à double entrée permet,pour un segment de numéro connu,de mémoriser et de retrouver les adresses de début et de fin du segment dans le fichier graphique. Un autre tableau appelé master retient pour chaque segment le segment qui le précède dans le fichier graphique. Ces tableaux servent à gérer

les opérations de clôture, d'effacement et de positionnement de paramètres des segments.

Dans le cas d'une donnée, un argument de texte est envoyé dans le fichier graphique. Un autre argument, qui correspond à une coordonnée, est d'abord transformé en coordonnée physique de l'écran puis envoyé dans le fichier graphique.

Une gestion particulière des arguments a été rendue nécessaire par la forme du programme graphique. Le programme graphique doit boucler sur lui-même pour rester visible. C'est ce bouclage qui réalise le rafraîchissement de l'écran. Le bouclage est formé d'une instruction de saut sur le début du programme graphique et occupe deux mots mémoire. Après chaque instruction, c'est-à-dire lorsque tous les arguments sont parvenus au gestionnaire, le saut est généré et le saut précédent est remplacé par les deux premiers mots de l'instruction qui vient d'être générée. Cette gestion utilise les variables mot1, mot2, lastjmp.

Gtclose()

Cette routine termine le programme graphique en arrêtant le processeur graphique.

Gtint(): routine d'interruption

Cette routine d'interruption sert avec la routine gtwrite à la gestion des sauts de bouclage sur le début du programme graphique. Lorsque gtwrite a reçu tous les arguments d'une instruction, elle ajoute une instruction d'arrêt du processeur graphique avec interruption du processeur central. La routine gtint est exécutée après l'arrêt du processeur graphique. Elle remplace le précédent saut de bouclage par les valeurs contenues dans les variables mot1 et mot2 et met à jour lastjmp. Enfin, elle relance le processeur graphique (pour la première instruction le lancement du processeur est effectué dans le routine gtwrite).

Penint(): routine d'interruption

Cette routine est exécutée lorsque une action du crayon lumineux a été détectée. Elle mémorise les coordonnées de la position visée par le crayon lumineux, relance le processeur graphique et réveille la routine de lecture gtread.

Charint(): routine d'interruption

Cette routine est exécutée lorsque le processeur graphique rencontre un caractère illégal. Elle remplace par des instructions nulles tous les mots du fichier graphique entre celui qui contient le caractère illégal et l'instruction graphique suivante. Ensuite, elle relance le processeur graphique.

Annexe 6.

Programme catgt.c

Le programme catgt.c permet de paginer un fichier en pages affichables sur le gt42.
Il lit d'abord le fichier à paginer et tente d'ouvrir ce fichier en lecture. Si ce fichier n'existe pas, un message d'erreur est envoyé à l'utilisateur et le programme est terminé. Si le fichier existe, le programme lit un caractère à la fois.
Si le caractère lu est un saut de ligne, il ajoute le caractère "retour du chariot".
Si ce caractère est un catactère de tabulation, il ajoute un nombre de "blanc" pour compenser ce caractère
Si le nombre de ligne a atteint la dimension de la page, il affiche la page sur le gt42 et se met en attente. Pour obtenir la page suivante, l'utilisateur doit envoyer un caractère; si ce caractère est "fin de fichier" , le programme se termine.
Lorsque la fin de fichier est atteinte, le programme affiche la partie de la page qui est remplie. Un caractère quelconque envoyé par l'utlilisateur termine le programme.

Les programmes édités sous le système Unix sont disponibles au secrétariat.

Annexe 7.

BIBLIOGRAPHIE

Berg 76 Bergeron "Picture primitives in device independent graphic systems" - ACM Sigplan Notices, vol II, n°6 June 1976, pp. 57-61; - Computer Graphics vol 10, n°1, Spring 1976.

Caru 77 L. C. Caruthers, J. van den Bos, A. van Dam, "GPGS, a device independent general purpose graphic system for stand-alone and satellite graphics." Computer Graphics, vol 11 n°2 Summer 1976, pp. 112-119

Dec-F RSX-11M/Fortran, Graphic extensions. Users guide - Digital Equipment Corporation

Foley-76 J. D. Foley, "Picture naming and modification; an overview." - ACM Sigplan Notices, vol II, n°6 June 1976 pp. 49-53 - Computer Graphics, vol 10, n° 1 Spring 1976.

Graf.C Graf.C Listing d'un programme,"The new graphics library package for Unix", non publié, aimablement communiqué par Mr. Rowson of Queen Mary's College, London, daté 21 mars 1977.

Hag. P. J. W. ten Hagen, "ALGOL68G: graphic extension of ALGOL 68" , Mathematical Centre, Amsterdam non publié, aimablement communiqué par Monsieur Paul Klint.

IFIP 72 Discussion: "Are we near a universal graphic language?" -Proceedings of the IFIP Working Conference on Graphic Languages, North Holland 1972 pp.354-368

Mach C. Machgeels, "Software de base et software fonctionnel pour le traçeur digital Benson 441, relié au CDC 6500"- Laboratoire d'Informatique Théorique ULB, novembre 1974: Manuel pour utilisateurs.

New 1 W.N. Newmann & R.F.Sproull, "Principles of interactive computer graphics" Chap.16, New-York McGraw Hill 1973

New 2 W.N. Newmann,"Display procedures" - C.ACM, vol 14 n°1 , October 1971 p.651

New 3 W.N. Newmann,"A prototype graphics system" - Proceedings of the IFIP Working Conference on Graphic Languages, North Holland 1972 pp. 291-301

BIBLIOGRAPHIE (Suite)

New 4 W.N. Newmann,"Interactive graphics on a small micro-programmable computer."- Members Report n° 14, Queen Mary College, London Sept 1973

Rich 1 D.M. Ritchie and K. Thompson, "The Unix time-sharing system." - Bell Telephone Laboratories, rapport interne -CACM vol.17 n° 7 July 1974 pp. 365-375

Rich 2 D.M. Ritchie,"C-Reference Manual", -Bell Telephone Laboratories, rapport interne.

Sig 77 Status report of the graphic standards planning committee of ACM/Siggraph - Computer Graphics vol.11 n° 3 Fall 1977

Thom 76 E.L. Thomas,"Methods for specifying diplay parameters in graphics programming languages." ACM Sigplan Notices, vol. II n°6 June 1976 pp.54-56 Computer Graphics, vol 10 n° 1 Spring 1976

VT 11 VT11 Graphic Display Processor Digital Equipment Corporation

Wall 76 V. L. Wallace, "The semantics of graphic input devices." - ACM Sigplan Notice , vol II, n°6 June 1976 pp. 61-65 -Computer Graphics vol 10, n° 1 Spring 1976

Ci-dessous la bibliographie des bibliothèques comparées dans le rapport du comité GPSC.

BIBLIOGRAPHY OF PACKAGES REVIEWED

Adage

"FORTRAN IV Language and Programming System," Revision A, Adage Inc., Boston, MA, February 1972.

"GRAFX - AMOS/2 Graphics Image Processor," Revision C, Adage Inc., Boston, MA, August 1975.

"RGS - Adage/300 Refresh Graphics System," Revision C, Adage Inc., Boston, MA, March 1974.

CalComp

"CalComp Software Reference Manual," California Computer Products Inc., Anaheim, CA, October 1976.

"Programming CalComp Electromechanical Plotters," California Computer Products Inc., Anaheim, CA, January 1976.

User's guides to all CalComp Functional and Application Packages.

DISSPLA

"DISSPLA Advanced Manual," Integrated Software Systems Corp., San Diego, CA, 1970.

"DISSPLA Beginners/Intermediate Manual," Integrated Software Systems Corp., San Diego, CA, 1970.

"DISSPLA Pocket Manual," Integrated Software Systems Corp., San Diego, CA, 1975.

"DISSPOP - DISSPLA Post Processor Option," Integrated Software Systems Corp., San Diego, CA, 1976.

GCS

"Graphics Compatibility System Programmer's Reference Manual," U.S. Army Corps of Engineers, Waterways Experiment Station, Vicksburg, MS.

"Primer on Computer Graphics Programming," U.S. Army Corps of Engineers, Waterways Experiment Station, Vicksburg, MS.

"GCS - The Graphics Compatibility System," Computing Center, Purdue University, Lafayette, IN, June 1976.

GINO-F

"GINO-F Technical Information Booklet," Issue 4, Computer Aided Design Centre, Cambridge, England, 1977.

"GINO-F User Manual," Issue 2, Computer Aided Design Centre, Cambridge, England, December 1976.

GPGS

Groot, D., Hermans, E., Caruthers, L., and J. Pathberg, "General Purpose Graphic System Reference Manual," University of Nijmegen, Nijmegen, The Netherlands, November 1975.

Caruthers, L.C., and A. van Dam, "General Purpose Graphic System User's Tutorial," First Edition, University of Nijmegen, Nijmegen, The Netherlands, October 1975.

"GPGS-F Users Guide," RUNIT Computer Center, University of Trondheim, Trondheim, Norway, September 1975.

IG

Goodrich, A.C., "Integrated Graphics Implementation Guide," Computing Center, University of Michigan, Ann Arbor, MI, March 1977.

Goodrich, A.C., "Integrated Graphics System User's Guide," Computing Center Memo 229, University of Michigan, Ann Arbor, MI, January 1976.

Goodrich, A.C., "Recent Changes and Additions to the Integrated Graphics System," Computing Center, University of Michigan, Ann Arbor, MI, February 1977.

Tektronix

"Advanced Graphing II User's Manual," Tektronix, Inc., Beaverton, OR, 1973.

"Terminal Control System User's Manual," Tektronix, Inc., Beaverton, OR, 1976.

TABLE DES MATIERES