Presentation des normes graphiques GKS.Seconde edition

Patrice Bertrand

INRIA

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports Techniques

N° 70

PRÉSENTATION DES NORMES GRAPHIQUES GKS

SECONDE ÉDITION

Patrice BERTRAND

Juin 1986

PRESENTATION DES NORMES GRAPHIQUES GKS

par

BERTRAND Patrice

INRIA

Domaine de Voluceau - Rocquencourt

BP 105

F-78153 Le Chesnay Cédex

Résumé

GKS (Graphical Kernel System) est une norme internationale de programmation dans le domaine du graphisme sur ordinateur, qui a été récemment adoptée par l'organisation ISO. Le but de ce rapport est de mettre à la disposition des programmeurs en graphisme, une documentation en langue française expliquant l'essentiel des normes GKS.

Dans une première partie, nous décrivons les principes généraux de GKS, c'est à dire un ensemble de fonctions réalisant des tâches graphiques de manière indépendante du langage.

Puis, nous expliquons l'emploi des fonctions de base de GKS, en ajoutant une description de ces fonctions (niveau OA) écrites en fortran 77.

Enfin, nous exposons les concepts de segment et d'intéractivité graphique tels qu'ils ont été implantés dans les normes GKS.

Abstract

Recently adopted by the ISO (International Organization for Standardization), GKS (Graphical Kernel System) is an international standard in the Computer Graphics field.

The main object of this report is to provide a documentation on GKS, it is written in the french language and describes the fundamental aspects of GKS.

In a first part, we describe the general principles of GKS, i.e. a set of functions that realize graphical tasks in a language independant way.

We explain the use of basic GKS functions and add a description of these functions (level OA) in the fortran 77 version.

Then, we expose the concepts of segment and graphical interactivity implanted in the GKS standard.

PRESENTATION DES

NORMES GRAPHIQUES GKS

TABLE DES MATIERES

INTRODUCTION

Depuis l'année 1982, il existe pour la première fois, une norme internationale de programmation dans le domaine du graphisme sur ordinateur ; il s'agit de GKS (Graphical Kernel System), adopté en tant que norme ISO, à la suite d'une collaboration internationale de 6 années entre les experts du groupe "Computers Graphics" de l'organisation ISO.

GKS est avant tout une norme de programmation graphique, c'est-à-dire une structure d'organisation logique de programmes permettant de réaliser l'ensemble des tâches graphiques élémentaires. La structure d'organisation proposée par GKS a déjà été écrite dans plusieurs langages de programmation (Fortran 77, langage C,...). D'autre part les normes GKS sont indépendantes des caractéristiques du terminal graphique utilisé ; il est donc nécessaire de créer une interface entre les fonctions GKS (écrites dans un langage donné) et les ordres graphiques que comprend le terminal de l'utilisateur.

Le but de ce texte, est de mettre à la disposition des programmeurs en graphisme, une documentation en langue française expliquant les principes de GKS, et donnant une description précise des fonctions de base de GKS écrites en fortran 77.

Dans une première partie, nous expliquons les principes généraux de GKS, en résumant la partie I de l'ouvrage "Computer Graphics Programming" [1]. Puis, toujours d'après le même ouvrage, nous décrivons l'emploi des fonctions de base de GKS, en ajoutant une description précise des fonctions GKS de niveau OA, écrites en fortran 77. La troisième partie explique les principes d'utilisation des segments et de l'interactivité graphique, à travers GKS. Pour cette dernière partie, nous nous sommes inspirés de plusieurs chapitres du livre "Introduction to the Graphical Kernel System (GKS)" [2].

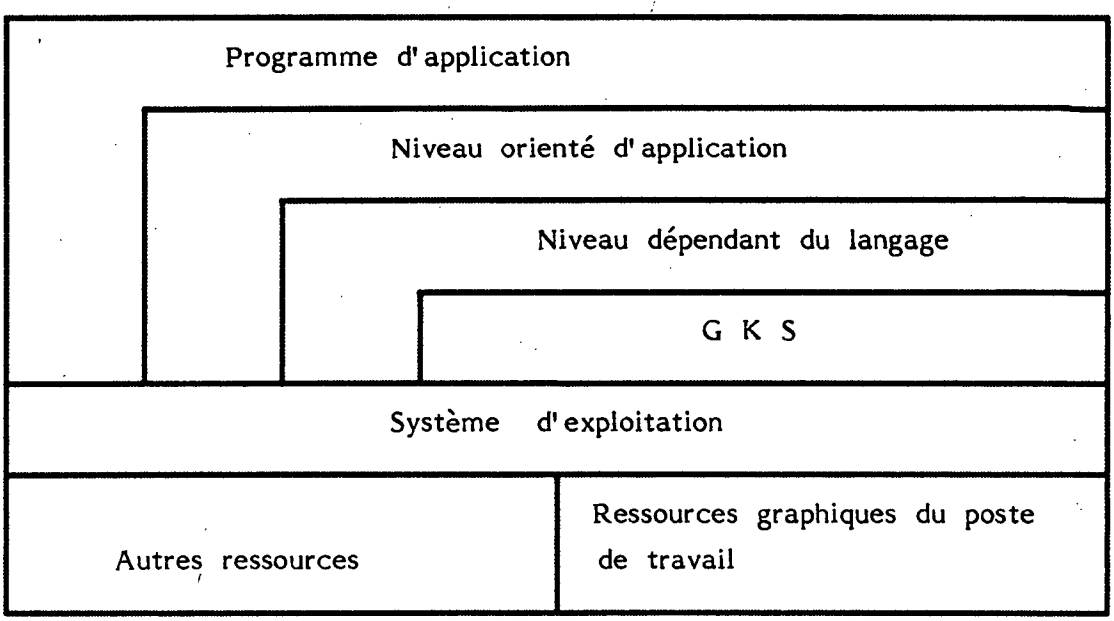
1ère Partie : PRINCIPES GENERAUX DE GKS

SOMMAIRE

1 - INTRODUCTION

Les normes GKS définissent un ensemble de fonctions réalisant des tâches grahiques de manière indépendante du langage. Cependant, dans une implantation du système, ces fonctions doivent être réalisées en tant que sous-programmes (ou procédures) dans un langage donné. Une telle réalisation propre à un langage, dans lequel le noyau du système (indépendant du langage) est "encastré", est appelé niveau de langage. Les fonctions créées par le niveau de langage peuvent être utilisées par le programmeur d'application, en même temps que les fonctions du système utilisé.

Des niveaux dépendant des applications peuvent être construits à partir du niveau dépendant du langage. Le modèle de niveaux, représenté ci-dessous, illustre le rôle de GKS dans un système graphique.



Par la suite, les moyens d'entrée et sortie graphiques sont appelés "terminaux" ou "postes de travail" ou encore "stations de travail". La traduction des fonctions de représentation (indépendante des moyens matériels) en représentations propres au terminal, est assurée par une interface spécifique pour chaque type de terminal ("device driver").

2 - PRINCIPAUX CONCEPTS DE GKS

Ces concepts sont liés aux tâches du système : création, représentation, manipulation et stockage d'images, extraction de parties d'images, contrôle des postes de travail, manipulation des entrées, etc ...

OUTPUT : sortie graphique. Les figures élémentaires dont on a pris l'image, sont les "output primitives". Les sorties graphiques sont contrôlées par un ensemble de paramètres (appelés attributs) qui peuvent être la couleur, la largeur du trait, etc ...

COORDINATE SYSTEMS AND TRANSFORMATIONS : système de coordonnées et de transformations. Les "figures" sont créées dans un ou plusieurs systèmes de coordonnées utilisateurs. Ces figures élémentaires peuvent être portées à la surface écran de plusieurs postes de travail différents.

WORKSTATION : terminal ou poste de travail ou station de travail. En général les moyens matériels de sortie et plusieurs moyens matériels d'entrée sont réunis dans une workstation. Une "workstation" est un traceur ou un écran avec clavier.

INPUT : entrées graphiques. GKS gère aussi des entrées alphanumériques. GKS manipule les entrées dans un mode indépendant du matériel en définissant des moyens d'entrée logique.

SEGMENTATION : la tâche de manipuler des parties d'image conduit au concept de segmentation. Une image est composée de parties appelées SEGMENTS qui peuvent être affichés, transformés, copiés ou détruits indépendemmant les uns des autres. Des segments peuvent être identifiés par un opérateur et leur identification est communiquée au programme d'application.

METAFILE : méta-fichier. Le méta-fichier est un moyen de stocker les images dans un but de création d'archive ou de transfert d'images (en une position différente à l'écran ou dans un autre système).

STATE LISTS : listes d'états. A tout moment, GKS est dans un état opérationnel qui est représenté par les valeurs d'un nombre situé dans une liste d'états. Ces valeurs sont changées par des fonctions GKS appelées par le programme d'application. Un état comprend, par exemple, l'ensemble des postes de travail connectés au système à un moment donné, ou les segments qui existent, etc...

LEVELS : niveaux fonctionnels. Les fonctions GKS sont regroupées en 9 niveaux différents.

ERROR HANDLING : traitement d'erreur. GKS définit pour chaque fonction, une variable entière dont les valeurs représentent les différentes erreurs possibles.

DIMENSIONS : Pour la majorité des applications graphiques sur ordinateur, un système en 2 dimensions, comme présente GKS, sera suffisant. Une extension de GKS à 3 dimensions est en cours de définition.

3 - CREATION DE SORTIES GRAPHIQUES

a) Graphiques coordonnées et graphiques point par point

COORDINATE GRAPHICS : ce sont les graphiques (conçus sur ordinateur) dans lequel les images affichées sont générées à partir des commandes d'affichage et des données sur les coordonnées. Les éléments de base des images de graphiques en coordonnées sont les vecteurs ou les suites de vecteurs. Le "display file" est un fichier mémorisant l'image au moment de l'affichage.

RASTER GRAPHICS : graphiques point par point. Les graphiques (conçus sur ordinateur) dans lesquels l'image affichée est composée d'un ordre de "pixels" rangés en ligne et en colonne.

PIXEL : le plus petit élément de surface auquel on peut attribuer de façon indépendante une couleur et une intensité. Les éléments de base d'un graphique point par point sont les points isolés d'une surface affichée qui peuvent être adressés indépendamment les uns des autres. Habituellement,

un matériel affichant du graphique point par point créé l'image ligne par ligne. A cause des problèmes de balayage, l'image doit être stockée dans une mémoire appelée "pixel storage" ou "frame buffer". Le texte peut être affiché à la fois par un matériel utilisant les vecteurs, ou un matériel point par point. Le "character generator" engendre des caractères en les décomposant en figures élémentaires qui peuvent être soit des vecteurs soit des points.

b) Output primitives et attributs

Les éléments de base à partir desquels une image est construite sont appelés "output primitives". Ils sont caractérisés par leur géométrie et par la façon dont ils apparaissent sur l'écran d'un terminal. Ces aspects sont contrôlés par un ensemble d'attributs qui appartiennent à une primitive. Certains attributs peuvent différer d'un poste de travail à un autre. Dans GKS, il y a des fonctions pour la création de primitives et la mise en place d'attributs, ceux-ci dépendants du type de terminal utilisé.

OUTPUT PRIMITIVE : primitive graphique ou fonction graphique de base, qui peut être utilisée pour la construction d'une image. Les output primitive de GKS sont POLYLINE, POLYMARKER, TEXT, FILL AREA, CELL ARRAY et GENERALIZED DRAWING PRIMITIVE.

DISPLAY IMAGE : un ensemble d'ordres graphiques générés, à un moment donné sur une même surface.

ATTRIBUTE : une caractéristique particulière qui concerne les "primitives output" ou les segments. Il y a six output primitives : une primitive de lignes, une primitive de points, une primitive de texte, deux raster-primitives et une primitive d'intérêt général pour des stations de travail ayant des possibilités particulières.

Line primitive :
POLYLINE : GKS génère un ensemble de lignes droites joignant une séquence de points.

Point-primitive :
POLYMARKER : GKS génère des symbôles (variés) centrés sur des positions données.

Text-primitive :
TEXT : GKS génère une chaîne de caractères à une position donnée.

Raster-primitives :
FILL AREA : GKS génère un polygone qui peut être rempli d'une couleur, ou hachuré, ou rempli par un motif.

CELL ARRAY : GKS génère un tableau de cellules rectangulaires chacune ayant une couleur spécifique.

General purpose primitive :
GENERALIZED DRAWING PRIMITIVE (GDP) : GKS prend les possibilités particulières de sortie de certains terminaux graphiques comme le tracé d'arcs circulaires ou elliptiques. Les objets sont caractérisés par un identificateur, un ensemble de points, et des données supplémentaires.

Les attributs décrivent les aspects des "output primitives" (cf. Table ci-après)

PICK IDENTIFIER : un numéro assigné à des output primitives à l'intérieur d'un segment et renvoyé par le pointeur. Le même pick identifier peut être assigné à différents output primitives. Les "picks identifiers" n'ont de signification qu'avec les fonctions d'entrées utilisées pour l'identification des segments par un opérateur.

LINETYPE : type de ligne : pleine, en tiret, en pointillé.

LINEWIDTH SCALE FACTOR : épaisseur de ligne.

COLOUR : la couleur est précisée suivant 3 intensités dans les 3 couleurs rouge, vert et bleu. L'intensité varie de 0 à 1 ; (0,0,0) représente la couleur noire, (1,1,1) la couleur blanche.

MARKER TYPE : il s'agit d'un numéro précisant le symbole utilisé pour l'identification des positions du polymarker.

MARKER SIZE SCALE FACTOR : il s'agit d'un facteur d'échelle utilisé pour la représentation du polymarker.

TEXT FONT : c'est un numéro d'identification de la police des caractères utilisés (roman, gothique, vieil anglais, ...).

TEXT PRECISION : un attribut (string, character et stroke) décrivant la qualité de précision concernant la position, la taille, l'orientation et le style des caractères obtenus en sortie.

CHARACTER HEIGHT : hauteur des caractères.

CHARACTER EXPANSION FACTOR : la déviation du rapport largeur/hauteur par rapport à celui stocké avec la police de caractère.

TEXT PATH : sens de lecture du texte (i.e. : right (de la gauche vers la droite), left (de la droite vers la gauche), up (de bas en haut), down (de haut en bas)).

CHARACTER SPACING : espace entre deux charactères (précisé de façon implicite par l'attribut "text font").

CHARACTER ALIGNMENT : attribut permettant d'afficher une chaîne de caractères à un point de référence (e.g., left aligned, centred).

INTERIOR STYLE : détermination du mode de remplissage d'une aire : il y a 4 modes : contour vide, aire remplie, aire remplie avec un motif répété et aire hachurée.

PATTERN SIZE : taille du rectangle du motif de base, servant à remplir une aire.

PATTERN REFERENCE POINT : origine du rectangle du motif de base. Le coin gauche inférieur de ce rectangle est placé au point de référence. Puis le motif de base est répété jusqu'à ce que toute l'aire soit remplie.

PATTERN ARRAY : un motif est défini par un tableau de cellules rectangulaires, chaque cellule ayant une couleur assignée. Des valeurs sont utilisées pour attribuer des couleurs au motif rectangulaire de base.

HATCH STYLE : les hachures sont précisées par un numéro de sélection du type de hachures (verticales, obliques, etc...).

Table : attributs des primitives graphiques

Primitive	Attributs	
POLYLINE	PICK IDENTIFIER LINEWIDTH SCALE FACTOR	LINETYPE COLOUR
POLYMARKER	PICK IDENTIFIER MARKER SIZE SCALE FACTOR	MARKER TYPE COLOUR
TEXT	PICK IDENTIFIER CHARACTER HEIGHT CHARACTER UP VECTOR CHARACTER EXPANSION FACTOR TEXT PATH CHARACTER SPACING	TEXT FONT TEXT PRECISION COLOUR TEXT ALIGNMENT
FILL AREA	PICK IDENTIFIER PATTERN SIZE PATTERN REFERENCE POINT PATTERN ARRAY	INTERIOR STYLE HATCH STYLE COLOUR
CELL ARRAY	PICK IDENTIFIER	
GENERALIZED DRAWING PRIMITIVE	PICK IDENTIFIER autres attributs dépendant de la définition de GDP	

c) Indices, Bundles et Tables

Certains attributs sont toujours précisés de manière indépendante du poste de travail. Pour la plupart, le programme d'application peut préciser s'il choisit de rendre les attributs dépendants ou non de la station de travail.

Les attributs indépendants de la station de travail sont établis par des fonctions GKS et ce jusqu'à ce qu'il y ait un changement déclaré de la même façon ; par exemple :

SET CHARACTER HEIGHT (hvalue) : stocke la valeur "hvalue" et tout texte sera généré avec la hauteur de caractère "hvalue".

Les attributs dépendants de la station de travail sont précisés par un indice associé à une primitive (à l'exception des couleurs et des motifs). Cet indice pointe dans une table existant pour chaque poste de travail et prévue pour tous les types de primitives. L'ensemble des attributs dépendant d'une station de travail pour un type de primitive est appelé "bundle", la table qui contient les attributs est appelée "bundle table". Par exemple, pour les polylines, l'indice du POLYLINE pointe dans une bundle table qui contient les indices linewidth, linetype, colour.

POLYLINE INDEX, POLYMARKER INDEX, TEXT INDEX et FILL AREA INDEX peuvent être établis de façon indépendante.

Un indicateur (du type 0 ou 1) est présent pour chacun de ces attributs qui peut ainsi devenir dépendant ou non de la station de travail. Ces indicateurs sont appelés "aspect source flags" et peuvent prendre l'une des deux valeurs "BUNDLED" (dépendant de la station) ou "INDIVIDUAL" (independant de la station).

BUNDLE INDEX : indice dans une bundle table pour des primitives de sortie. Il définit les aspects de la primitive, dépendant du poste de travail.

BUNDLE TABLE : une table définie pour chaque station, et associée à chaque primitive. Les entrées de la table précisent tous les aspects dépendant d'une station, concernant les primitives. En GKS, il existe des "bundle tables" pour polyline, polymarker, text et fill area.

POLYLINE BUNDLE TABLE : Une table associant des valeurs particulières pour chaque aspect (dépendant de la station) d'une primitive polyline avec un "polyline bundle index". Dans GKS, cette table contient des entrées pour le type, l'épaisseur et la couleur de ligne.

POLYMARKER BUNDLE TABLE : une table associant des valeurs particulières pour chaque aspect (dépendant de la station) d'une primitive polymarker avec un "polymarker bundle index". En GKS, cette table contient des entrées pour le type, le facteur d'échelle et la couleur du polymarker.

TEXT BUNDLE TABLE : une table associant des valeurs particulières pour tout aspects (dépendant de la station) d'une primitive text avec un "text bundle index". En GKS, cette table contient des entrées pour la police, la

precision, le facteur d'expansion, l'espacement et la couleur du texte.

Les entrées dans les "bundle tables" sont définies par le programmeur qui installe GKS, selon les possibilités de la station. Les entrées peuvent être changées par des fonctions GKS. Une propriété remarquable des attributs dépendant de la station est qu'ils peuvent influencer l'image déjà affichée. Les attributs associés aux primitives dans le mode "INDIVIDUAL" ne peuvent pas être changés dans un stade ultérieur.

COLOUR TABLE : une table dépendant de la station, dans laquelle les entrées précisent les intensités du rouge, du vert et du bleu formant une couleur donnée.

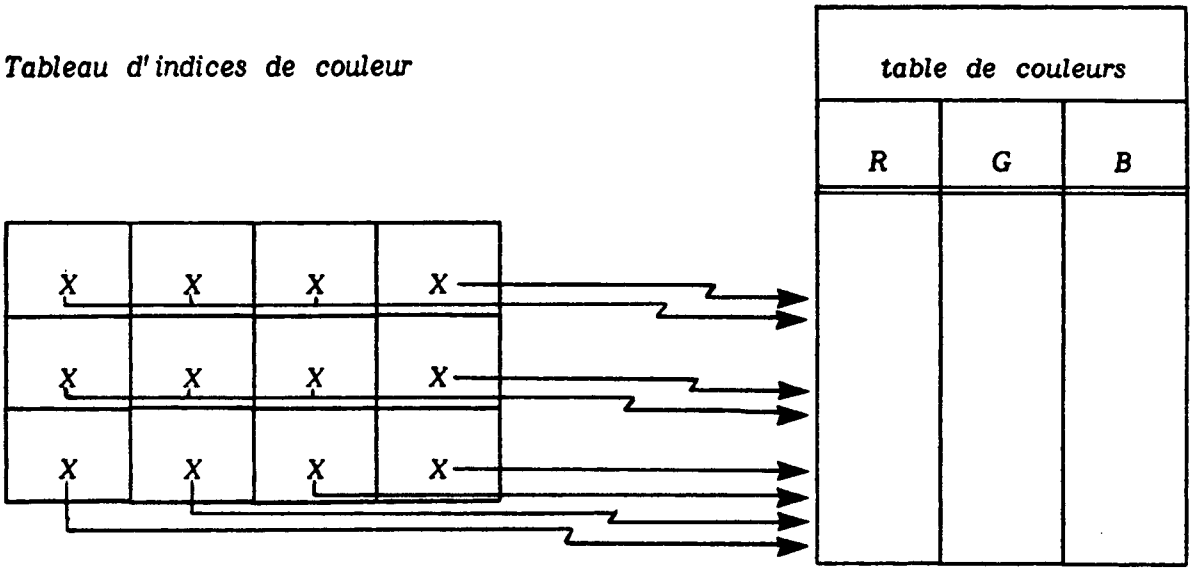
Le schéma de base pour la primitive fill area est semblable, cependant, "fil area bundle index" possède un indice de style qui est utilisé pour la sélection d'un motif ou d'un style de hachure.

FILL AREA BUNDLE TABLE : une table associant des valeurs particulières pour tout aspect (dépendant de la station) d'une primitive fill area avec un fill area bundle index. En GKS, cette table contient des entrées pour le mode de remplissage, le style du motif utilisé, et la couleur de remplissage.

CELL ARRAY : utilise un schéma différent ; un ensemble d'indices de couleur est associé à la fonction primitive cell array. Ces indices de couleur se réfèrent directement à la table de couleur.

La fonction primitive contient des paramètres

Aspect dépendant de la station



4 - SYSTEMES ET TRANSFORMATIONS DE COORDONNES

a) Systèmes de coordonnées

Trois systèmes de coordonnées ont été définis en GKS.

Le programmeur d'application utilise le système workd coordinate (W.C) pour désigner les éléments de l'image. Le système world coordinate est le système standard des utilisateurs.

WORLD COORDINATE (W.C) : c'est un système de coordonnées cartésiennes, indépendant du matériel, utilisé par le programme d'application pour définir les entrées et les sorties graphiques.

Puisque les différents matériels de sortie ont des systèmes de coordonées différents, GKS définit, en tant qu'abstraction de ces systèmes de coordonnées liés au matériel, un unique espace de coordonnées indépendant du matériel et normalisé (NDC).

NORMALIZED DEVICE COORDINATE (N.D.C) : un système de coordonnées intermédiaires, indépendant du matériel, normalisé en GKS, de 0 à 1 sur chaque axe.

L'image normalisée dans l'espace N.D.C peut être stockée et manipulée par un mécanisme de segment ; elle peut aussi être stockée sur un métafichier.

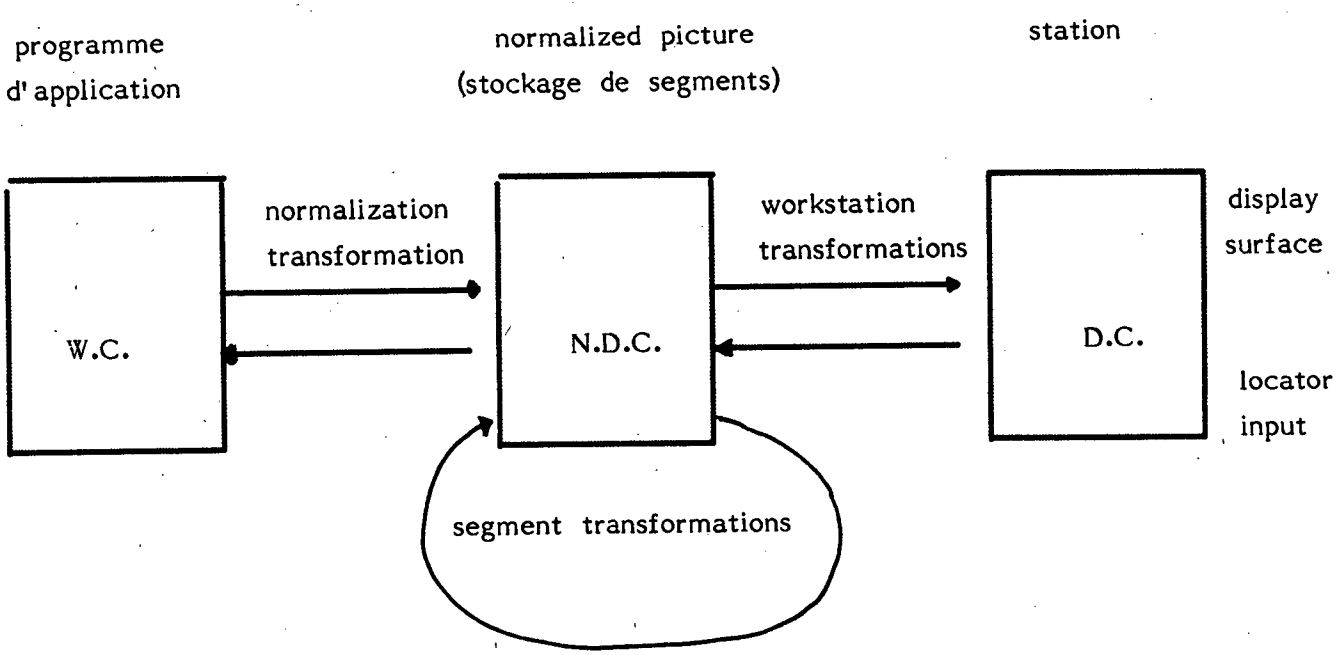
L'espace NDC est projeté sur l'espace des devices coordinates (D.C) de chaque station de travail qui affiche l'image. Chaque type de station peut avoir un espace de device coordinate différent. L'espace device coordinate est toujours un espace limité puisqu'il représente l'extension de la surface de l'écran.

DEVICE COORDINATE (D.C) : un système de coordonnées entièrement dépendant du matériel utilisé.

b) Transformations

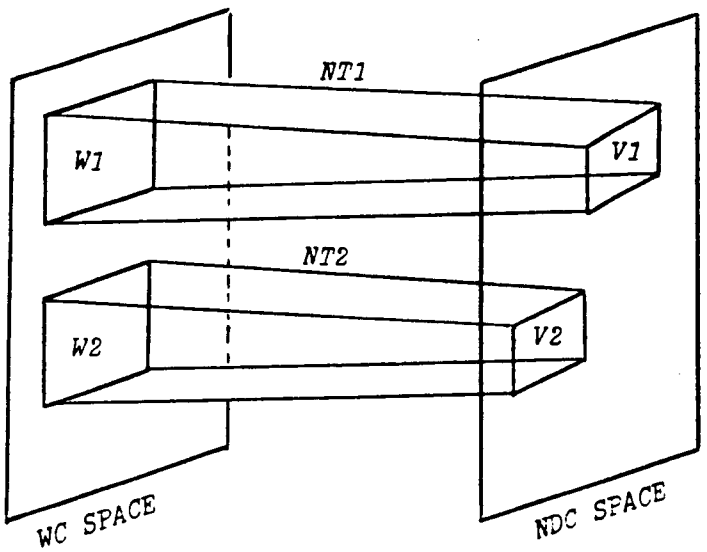
Un ensemble de transformations de normalisation définit les projections entre l'espace W.C et l'unique espace N.D.C. Un ensemble de transformations de station définit les projections entre l'espace N.D.C et la station.

Ces transformations définissent le "viewing pipeline" à partir de la vision du programmeur d'application jusqu'à la surface de l'écran, et le "input pipeline" dans le sens inverse. D'autres types de transformations peuvent être appliquées aux segments ; ces transformations se font dans l'espace N.D.C.



c) Transformation de normalisation

Pour réaliser la sortie graphique, une seule transformation est active. Elle est utilisée pour transformer les coordonnées W.C en coordonnées N.D.C. Une transformation de normalisation est précisée par les limites de l'aire dans le système W.C (on dit que cette aire est une fenêtre "WINDOW") qui doit être transformée dans une aire précisée de l'espace N.D.C (on dit que cette aire est une cloture "VIEWPORT") ; (cf. figure ci-dessous).



W = Window ou fenêtre
V = Viewport ou clôture
NT = Normalization Transformation ou Transformation de Normalisation

WINDOW : fenêtre. Une partie rectangulaire prédéfinie d'un espace virtuel (en GKS, l'espace virtuel est l'espace WC).

VIEWPORT : clôture. Une partie (définie dans le programme d'application) de l'espace NDC. En GKS, cette partie doit être rectangulaire.

NORMALIZATION TRANSFORMATION : Une transformation qui projette les limites et l'intérieur d'une fenêtre sur l'espace NDC pour obtenir les limites et l'intérieur d'une clôture.

Les limites de la fenêtre et de la clôture définissent des rectangles de côtés parallèles aux axes des systèmes W.C et N.D.C.

Plusieurs transformations de normalisation peuvent exister simultanément. L'une des transformations de normalisation définie est choisie pour générer une sortie graphique. Chaque transformation de normalisation est identifiée à un numéro de transformation qui est un entier compris entre 0 et une valeur n dépendant de l'implantation de G.K.S. La transformation associée au numéro 0 est la transformation unitaire qui projette [0,1] x [0,1] dans WC, sur [0,1] x [0,1] dans NDC. Cette transformation ne peut pas être modifiée.

Au début, toutes les transformations de normalisation sont initialisées par défaut à la transformation de numéro 0. Elles peuvent être réinitialisées, à tout moment, lorsque GKS est ouvert. Cette possibilité peut être exploitée par le programme d'application.

d) Clipping

Le programme d'application peut exiger que GKS ne montre que les parties de l'image qui sont à l'intérieur des clôtures qui sont associées aux éléments de l'image.

CLIPPING : supprimer les éléments du graphique qui sont en dehors des limites de la clôture.

5 - STATION DE TRAVAIL GRAPHIQUE

a) Sorties sur une station graphique

Les stations sont identifiées par le programme d'application en utilisant un identificateur de station. Il faut d'abord pour cela que la workstation soit ouverte. Les primitives d'une image sont envoyées à une station ouverte seulement après qu'elle ait été activée. Une fonction est disponible pour effacer l'écran. Après la désactivation, les sorties ne peuvent plus être envoyées sur la station désactivée. Les entrées peuvent être réalisées sur toute station ouverte. La suite suivante de fonctions illustre une sélection de stations.

```
OPEN GSK ;                                          (début du travail)
     OPEN WORKSTATION (N1,...) ;                    (ouverture de stations)
     OPEN WORKSTATION (N2,...) ;
          ACTIVATE WORKSTATION (N1) ;               (sorties autorisées sur N1)
               Output functions ;                   (générées seulement sur N1)
               Input functions ;                    (sur N1, N2)
          ACTIVATE WORKSTATION (N2) ;               (sorties autorisées sur N2)
               Output functions ;                   (générées sur N1 et N2)
          DEACTIVATE WORKSTATION (N1) ;             (plus de sortie sur N1)
               Output functions ;                   (générées seulement sur N2)
               Input functions ;                    (possible sur N1, N2)
          DEACTIVATE WORKSTATION (N2) ;             (plus de sortie)
     CLOSE WORKSTATION (N2) ;                       (fermeture de stations)
     CLOSE WORKSTATION (N1) ;
CLOSE GKS ;                                         (fin du travail)
```

b) Types de stations GKS

Il y a 6 catégories :

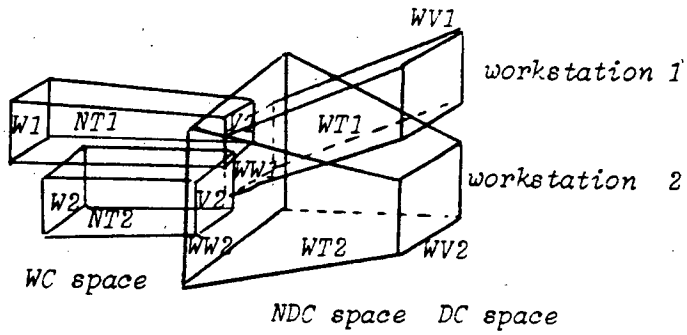
- stations de sortie, ayant une surface d'exposition (papier, écran),
- stations d'entrée, ayant au moins un dispositif d'entrée (clavier),
- stations d'entrée/sortie (on parle de station graphique interactive),
- stokage de segments indépendants de la station
- sortie de méta-fichiers GKS — facilités de GKS
- entrée de méta-fichiers GKS.

c) La transformation de station

Une transformation de station est une projection uniforme de l'espace N.D.C sur l'espace D.C attaché à une station (cf. figure ci-après). Tandis que la transformation de normalisation est utilisée pour la composition d'une image, la transformation de station permet de voir différents aspects de l'image sur différentes stations. La transformation de station peut être réinitialisée à tout moment, après l'ouverture de la station. Elle est définie par les limites d'une aire dans le système N.D.C à l'intérieur du cadre $[0,1] \times [0,1]$ (fenêtre de station) qui est projetée sur une aire spécifiée de l'espace D.C (clôture de station). Les limites de la fenêtre de station, et de la clôture de station définissent des rectangles parallèles aux axes des espaces N.D.C et D.C. Il faut noter que la transformation de station n'effectue que des transformations uniformes ; en d'autre termes GKS rétréciera automatiquement la clôture de station de telle sorte que le rectangle obtenu ait les mêmes proportions (rapport largeur/hauteur) que la fenêtre de station.

Transformation de normalisation et transformation de station

Normalization transformation and workstation transformation



W = Window, V = Viewport, NT = Normalization Transformation, WW = Workstation Window
WV = Workstation Viewport, WT = Workstation Transformation

TRANSFORMATION DE STATION : une transformation qui projette la fenêtre de station (espace N.D.C) sur la clôture de station (espace D.C), en conservant les proportions.

FENETRE DE STATION : région rectangulaire à l'intérieur de l'espace N.D.C qui sera projetée sur l'écran.

CLOTURE DE STATION : partie de l'écran choisie pour la sortie graphique

(cf. schéma plus haut)

d) Etat différé

La sortie sur une station graphique devrait, autant que possible, refléter l'état de l'image tel qu'il est défini à tout instant par le programme d'application. Cependant pour des raisons d'efficacité, GKS permet de retarder pendant un certain temps, les actions exigées par le programme d'application ; Une variable d'état de la station controle l'existence et la durée du délai d'affichage, elle est appelée "état différé de la station". Le délai d'affichage a une valeur par défaut qui dépend du type de station utilisée (écran ou plotter, par exemple). Le différé des images peut être forcé par le programme d'application (en utilisant la fonction UPDATE WORKSTATION).

e) Accès aux possibilités particulières d'un terminal

Les terminaux graphiques peuvent présenter plus de possibilités que celles décrites dans la table de description de la station. Dès lors, ces possibilités ne peuvent pas être utilisées par GKS. Cependant si la terminal est suffisamment "intelligent" les possibilités supplémentaires peuvent être accessibles, depuis GKS, par des fonctions spéciales. Une telle fonction est appelée GENERALIZED DRAWING PRIMITIVE ou "primitive généralisée" (GDP) (elle peut être utilisée pour la création de sorties graphiques sur une station). Une fonction pour accéder à des possibilités de la station autres que des sorties graphiques, est la fonction ESCAPE. Elle peut être utilisée, par exemple, pour déclencher une sonnerie.

ESCAPE : une fonction de GKS qui est l'unique accès au support dépendant de l'implantation ou du matériel, pour des fonctions non graphiques et n'appartenant pas à GKS.

6 - ENTREES

a) Systèmes interactifs

C'est la dimension d'interactivité qui est la cause de l'utilisation de plus en plus poussée des systèmes graphiques sur ordinateur. Les actions de pointage, sélection, d'esquisse, de mise en place et d'effaçage de façon instantanée sont vraiment adaptées au comportement humain dans son environnement.

Les caractéristiques des différents matériels d'entrée sont représentées par des moyens logiques d'entrée. GKS utilise un modèle d'entrée qui décrit les moyens d'entrées en termes logiques et physiques.

b) Classes d'entrée logique

LOGICAL INPUT DEVICE : une entrée logique est une abstraction d'un ou plusieurs dispositifs physiques qui permettent d'entrer une ou plusieurs valeurs logiques dans le programme.

INPUT CLASS : une classe d'entrée est un ensemble de moyens d'entrée qui sont équivalent du point de vue de leur fonction respective.

Nous donnons ci-dessous la définition des différentes classes d'entrée.

LOCATOR (input class) : donne une position dans l'espace W.C. La position est fournie par l'opérateur en actionnant un moyen d'entrée spécifique (par exemple en déplaçant un réticule avec une manette ou levier).

STROKE (input class) : donne une série de positions dans l'espace W.C. Les positions sont fournies par l'opérateur qui déclenche une entrée "locator" à un certain nombre d'endroits différents.

VALUATOR (input class) : donne un nombre réel. Cette valeur est fournie par l'opérateur qui actionne un moyen d'entrée spécifique : par exemple en tournant un potentiomètre ou en entrant un nombre au clavier.

CHOICE (input class) : donne un entier non négatif qui représente une sélection parmi plusieurs choix. Une valeur de choix est obtenue par l'opérateur qui choisit une possibilité, avec un dispositif spécifique (par exemple, en pressant un parmi plusieurs boutons, ou en relevant un item dans un menu).

PICK (input class) : donne un nom de segment et un identificateur "pick". Un segment est identifié par l'opérateur qui pointe un endroit de l'image affichée. La notion d'identificateur "pick" sera expliquée plus loin.

STRING (input class) : donne une chaîne de caractères. Une chaîne de caractères est entrée par l'opérateur au moyen d'un dispositif spécifique (le plus souvent un clavier alphanumérique)

c) Modes opératoires

Chaque dispositif d'entrée logique peut fonctionner selon 3 modes différents. Le mode est choisi dans le programme d'application. Seulement un de ces modes peut être utilisé pour obtenir une entrée avec un dispositif d'entrée logique, à un moment donné.

REQUEST : l'appel d'une fonction d'entrée en mode Request provoque l'arrêt du programme jusqu'à l'entrée d'une valeur logique, à partir d'un dispositif d'entrée logique spécifié. GKS attend jusqu'à ce qu'il y ait eut une entrée ou une action de break. Dans ce mode le programme d'application n'utilise qu'un seul dispositif d'entrée à la fois (comme un READ en fortran).

SAMPLE : l'appel d'une fonction d'entrée en mode Sample provoque le renvoi par GKS de l'actuelle valeur logique d'entrée d'un dispositif d'entrée logique spécifié. Cette valeur (position d'un réticule, par exemple) est la valeur courante du dispositif d'entrée : l'opérateur ne contrôle pas la validité de cette valeur.

EVENT : GKS garde dans une liste d'entrée des résultats ordonnés par leur ordre d'arrivée dans la liste. Ces résultats contiennent l'identification d'un dispositif d'entrée logique et une valeur logique associée. Le programme d'application peut déplacer les plus vieux résultats de la liste pour les examiner, il peut aussi "nettoyer", supprimer de la liste tout résultat, provenant d'un moyen d'entrée spécifié. L'opérateur peut contrôler la validité des valeurs entrées.

d) Echos, Prompts et Initialisation des moyens d'entrée

Quand le programme d'application désire que l'opérateur entre une valeur, il doit l'informer qu'une action est attendue (mode Request) ou possible. Ce type d'information concernant l'opérateur est appelé un prompt (par exemple pour entrer un texte, le prompt se présente sous l'apparence d'un curseur)

PROMPT : sortie indiquant à l'opérateur qu'un dispositif d'entrée logique donné est disponible.
Si l'interaction avec le dispositif d'entrée continue, l'opérateur doit être informé de la valeur actuellement en place. Cette information est appelée l'écho.

ECHO : l'information "en direct" de la valeur en cours.

Pour une entrée en mode request, le transfert de la valeur d'entrée vers le pogramme d'application doit être déclenché par une action de l'opérateur (par exemple, il appuie sur une touche, après avoir positionné le locator). Pour une entrée en mode event, le transfert de la valeur en cours dans la file des résultats, doit également être déclenché par une action de l'opérateur. "L'écho" qui informe l'opérateur qu'une telle action a bien été enregistrée, est appelé "acknowledgement" ou "accusé de réception".

ACKNOWLEDGEMENT : "accusé de réception" pour indiquer qu'une valeur en entrée a bien été enregistrée.

Le programme d'application peut contrôler ces indicateurs (prompt, echo, acknowledgement) : il peut supprimer ou instaurer des échos ou des prompts, ou les choisir à condition qu'ils existent parmi les dispositifs d'entrée. Cependant il ne peut créer un nouveau type d'écho ou de prompt.

Si un écho est créé en affichant une sortie graphique après l'interprétation par le programme d'application, d'une valeur logique entrée on dit que c'est un "feedback".

Quand l'activation d'un descriptif d'entrée commence, toutes les valeurs logiques d'entrée sont initialisées. Ces initialisations peuvent être faites par le programme d'application. Si l'écho ou le prompt fonctionnent, ils reflèteront cet état initial.

7 - SEGMENTS

a) Structure des images

Une image est composée de primitives graphiques. Ces parties d'image peuvent être désignées et manipulées comme un tout. On les appelle segments.

SEGMENT :

Un ensemble de primitives graphiques qui peut être manipulé comme une unité.

Si les primitives ne sont pas structurées en segments, la seule structure possible est de grouper les primitives en différentes images.

Une nouvelle image est créée lorsque GKS est ouvert ou quand l'écran est effacé. Chaque segment est stocké ; stocké signifie qu'il est envoyé à la station et qu'il se comporte sur cette station selon les manipulations de segment exigées par le programme d'application. Si la station n'a pas de possibilités réelles de stockage, le système GKS doit gérer les segments de telle manière que tout se passe comme si le stockage de segment pour la station était possible (possibilité conceptuelle de la station).

Les segments sont identifiés par un unique nom identificateur de segment. Toutes les primitives graphiques sont réunies dans un segment après sa création et jusqu'à sa fermeture. Après la fermeture d'un segment, aucune primitive graphique ne peut être ajoutée ou détruite dans ce segment. Aucun nouveau segment ne peut être créé tant que le segment précédent n'a pas été fermé. Les primitives graphiques à l'intérieur d'un segment peuvent avoir une ou plusieurs identifications supplémentaires. Ce type d'identification est appelée un identificateur "pick". C'est une partie de la valeur entrée lors de l'enregistrement d'un segment.

L'exemple suivant illustre la création et la fermeture de segment aussi bien que la spécification des identificateurs "pick" :

SET PICK IDENTIFIER (pi 4) ;	
CREATE SEGMENT (sega);	
Output functions ;	(segment=sega, pick =pi4)
SET PICK IDENTIFIER (pi 2) ;	
Output functions ;	(segment=sega, pick =pi2)
CLOSE SEGMENT ;	
Output functions ;	(pas de primitives enregistrables) (pick = pi2)
SET PICK IDENTIFIER (pi 5) ;	
Output functions ;	(primitives non enregistrables) (pick = pi5)
CREATE SEGMENT (Seg b) ;	
Output functions ;	(segment=seg b, pick =pi5)
SET PICK IDENTIFIER (pi 3)	
Output functions ;	(segment=seg b, pick =pi3)
CLOSE SEGMENT ;	

b) Manipulation de segments

Les segments peuvent être manipulés comme une entité : transformations géométriques, visibilité, détectabilité, brillance, copie. Un segment peut aussi bien être détruit de toutes les stations sur lesquelles il a été stocké que sur une station spécifiée. Si un segment est détruit de toutes les stations, il n'est plus connu de GKS.
Renommer un segment signifie remplacer son nom par un nouveau nom qui n'a pas déjà été attribué à un segment existant.

c) Attributs des segments

Les attributs des segments sont des valeurs d'état qui concernent toutes les primitives graphiques d'un même segment. Ces valeurs concernent la visibilité, la brillance, la détectabilité, la priorité du segment et la transformation du segment. Ces valeurs peuvent être modifiées sur tout segment existant, y compris un segment ouvert.

SEGMENT ATTRIBUTES : caractéristiques spécifiques aux segments.

HIGHLIGHTING : une manière indépendante du matériel, de mettre en valeur un segment en modifiant ses caractéristiques visuelles (brillance,...).

DETECTABILITY : un attribut de segment indiquant si un segment peut être pris en considération pour la fonction d'entrée : pick.

VISIBILITY : un attribut de segment indiquant si un segment est affiché à l'écran. On ne peut pas utiliser la fonction pick sur des segments invisibles.

SEGMENT PRIORITY : un attribut de segment utilisé pour déterminer qui de plusieurs segments se chevauchant aura la priorité pour les sorties et les entrées.

"Segment priority" ou priorité de segment, concerne uniquement les segments qui sont affichés à l'écran. Si des parties de primitives graphiques chevauchent d'autres appartenant à un segment visible, ayant un ordre de priorité plus grand, alors ces parties peuvent être invisibles. Ceci dépend de la façon dont l'implantation a été effectuée. Quand les figures élémentaires de segment se chevauchant sont pointées par la fonction pick, le segment ayant la plus grande priorité sera choisi.

d) Transformations de segment

Les transformations de segment agissent de l'espace NDC sur l'espace NDC. Elles réalisent des translations, des changements d'échelle et des rotations.

SEGMENT TRANSFORMATION : une transformation opérant sur les éléments affichés d'un segment. Il peut y avoir translation, changement d'échelle ou rotation sur l'écran.

TRANSLATION : en GKS, la possibilité de translation est limitée aux segments.

SCALING : changement d'échelle. En GKS, cette possibilité est limitée aux segments. Pour deux différents changements d'échelle dans deux directions orthogonales, deux valeurs différentes sont nécessaires.

ROTATION : en GKS, cette possibilité est limitée aux segments.

Les transformations de segment sont définies par une matrice de transformation qui est associée au segment. La matrice est de dimension 2x3, dont 2x2 pour le changement d'échelle et la rotation, et le reste de la matrice de dimension 2x1 étant consacré à la translation. Des fonctions utilitaires sont disponibles pour le programme d'application pour établir les matrices de transformation. Initiallement, la transformation est, par défaut, l'identité. La transformation de segment s'effectue après la transformation de normalisation, mais avant tout clipping.

e) Stockage de segments

Le stockage de segments dépendant de la station (WDSS) est le stockage (conceptuel) de segments sur une station (de sortie ou d'entrée-sortie). Ce stokage permet les changements d'attributs de segment, y compris les transformations et les destructions de segments. On définit le stockage de segments indépendant de la station (WISS) afin de permettre à des primitives de segments d'être transferrées d'une station à une autre, ou d'être insérées dans le segment ouvert (utilisation des fonctions COPY et INSERT). Seulement une WISS, est autorisée pour une implantation GKS. Le réalisateur de l'implantation a le choix de réaliser la WISS soit dans la partie de GKS indépendante de la station, soit d'utiliser les possibilités physique d'une station.

Des segments sont stockés dans la WISS aussi longtemps que la WISS est ouverte et active. Ils sont disponibles pour les fonctions copy. La fonction clear peut être employée : elle détruit tous les segments stockés.

Les primitives graphiques sont transformées depuis l'espace W.C jusqu'à l'espace NDC avant d'être enregistrées dans la WISS. Le rectangle de "clipping" (clôture de la transformation de normalisation) est stocké avec les primitives contenues dans un segment. Quand le segment est affiché sur la station le rectangle de clipping est utilisé pour couper les primitives du segment, si le clipping est actif (i.e. ce qui est à l'extérieur de la clôture est supprimé)

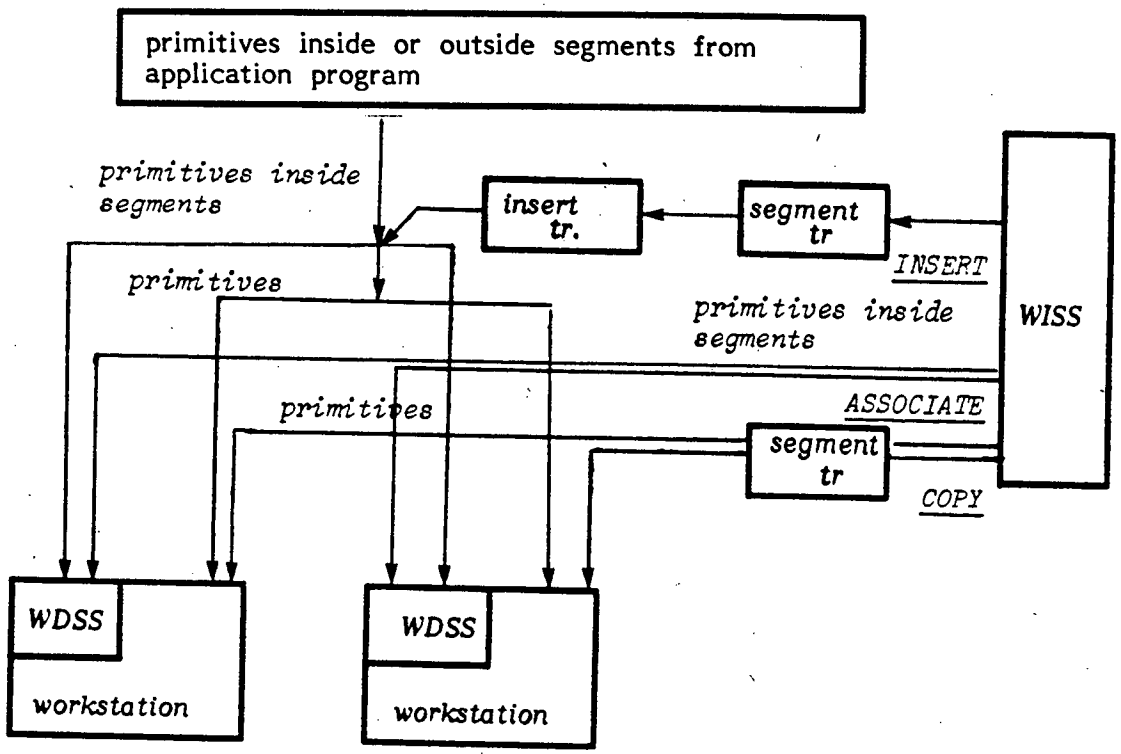
f) Copie de segments

Trois fonctions peuvent utiliser les segments contenus dans une WISS. Aucune de ces fonctions ne modifie le contenu des segments.

COPY SEGMENT TO WORKSTATION : cet ordre fait une copie de chaque primitive et de son rectangle clipping associé dans un segment de la WISS. Puis cet ordre transforme les primitives par la transformation de segment, replace les rectangles de clipping et le segment transformé dans le "viewing pipeline" vers la station spécifiée dans l'appel de la fonction.

ASSOCIATE SEGMENT WITH WORKSTATION : envoie le segment à la station désignée, afin d'obtenir le même effet que si la station avait été active depuis la création du segment.

INSERT SEGMENT : copie les primitives graphiques d'un segment d'une WISS et applique la transformation de segment suivie d'une transformation spéciale donnée dans la fonction Insert (appelée Insert transformation). Les rectangles de clipping ne sont pas associés aux segments traités par la fonction Insert ; on leur assigne un nouveau rectangle standard de clipping : celui de la transformation de normalisation courante.



8 - META-FICHIERS DE GKS

a) Métafichiers graphiques

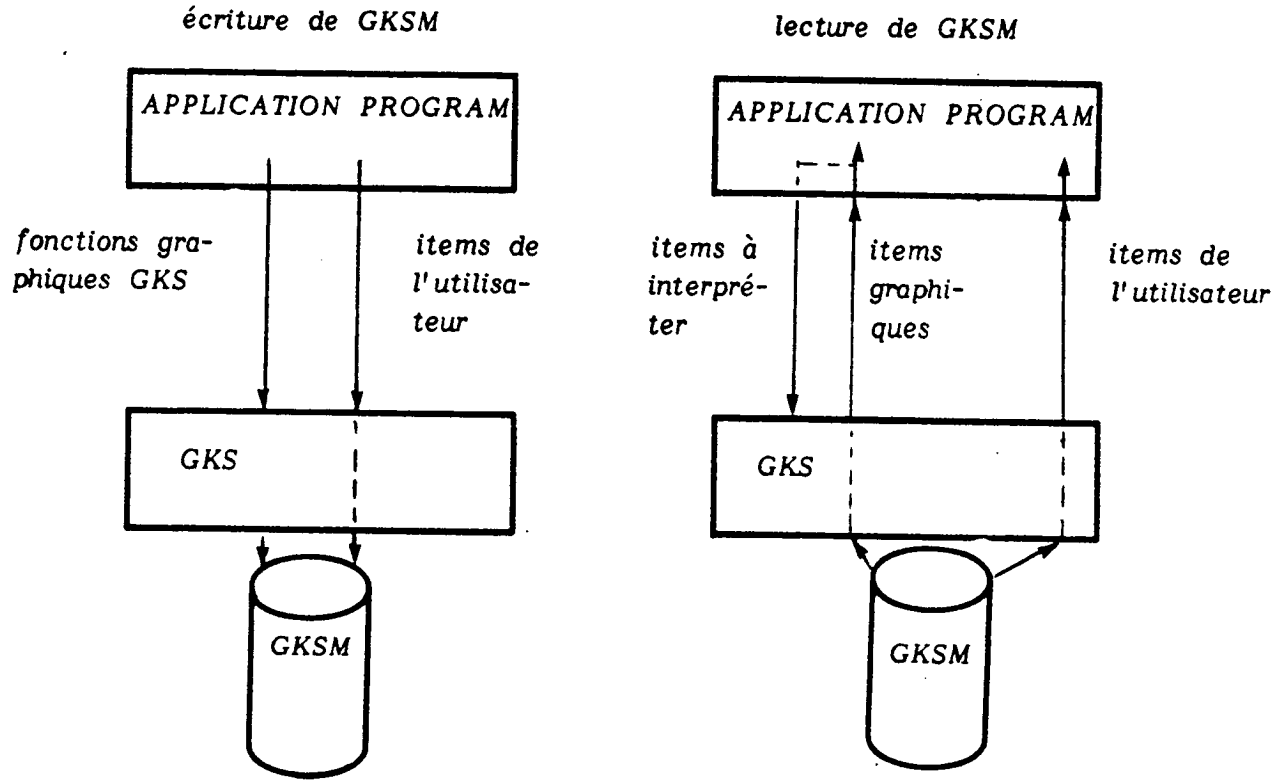
GKS offre des fonctions pour l'écriture d'une image sur un fichier externe et pour la relire de façon intégrale. Les fichiers utilisés pour le stockage d'images sont appelées métafichiers graphiques, de plus, ils sont indépendants du matériel et de l'application.

GKS Métafile : (GKSM) métafichier de GKS, un fichier séquentiel qui peut être écrit ou lu par GKS, utilisé pour le stockage à long terme et le transfert de l'information graphique.

Le principal intérêt des métafichiers graphiques est qu'ils sont capables de relier des systèmes graphiques différents d'une façon à la fois standard et directe. GKS utilise une interface pour communiquer avec un métafichier GKS.

b) Interface avec un métafichier

Les enregistrements graphiques de GKSM sont produits par les résultats de l'appel des fonctions GKS. Les enregistrements des utilisateurs sont écrits par la fonction : "WRITE USER ITEM TO GKSM". La lecture du métafichier est effectuée par GKS, sous le contrôle de l'utilisateur. Les enregistrements sont communiqués au programme d'application qui peut les générer ou les sauter. L'interprétation des enregistrements graphiques de GKSM peut aussi être laissée à GKS. Dans ce cas, le programme d'application les redonne à GKS, qui effectuera la fonction qui initiallement a créé l'enregistrement GKSM (cf. figure ci-dessous).



c) Format du métafichier

Un métafichier GKS contient des images en deux dimensions. Chaque image est représentée par une série d'enregistrements de données ("items") qui sont le résultat des fonctions GKS appelées.

Un métafichier GKS consiste en une suite d'enregistrements. Chaque enregistrement possède un type, une longueur d'enregistrement et des données. Le type de l'enregistrement indique si l'information qui suit, peut être interprêtée par GKS ou par le terminal.

9 - ETATS ET LISTE D'ETATS

a) Etat de GKS

A chaque instant durant l'exécution d'un programme utilisant GKS, GKS se trouve dans un état défini. Cet état est défini par une variable et par les valeurs des variables d'état contenues dans plusieurs listes présentes dans un système GKS. La variable "état" peut prendre les 5 valeurs suivantes :

GKS fermé
GKS ouvert
Au moins une station ouverte
Au moins une station active
Segment ouvert

La transition d'un état à un autre est provoquée par des fonctions de contrôle appelées par le programme d'application. Selon l'état de GKS, l'appel d'une fonction GKS est autorisée ou non.

b) Listes d'état

Tandis que la variable "état" est une valeur globale qui est disponible même si GKS est fermé, les autres variables d'état sont assignées, initialisées, comme un effet des appels de fonctions GKS. Les listes d'état dans GKS sont les suivantes :

liste d'état de GKS : assignée et initialisée par OPEN GKS ; détruite par CLOSE GKS, elle n'est présente qu'une fois dans l'implantation GKS.

liste d'état de segment : assignée et initialisée par CREATE SEGMENT, détruite quand le segment est détruit, elle existe pour tout segment existant.

liste d'état d'une station : assignée et initialisée par OPEN WORKSTATION, détruite par CLOSE WORKSTATION, elle existe pour toute station ouverte.

liste d'état des erreurs : assignée et initialisée par OPEN GKS, détruite par CLOSE WORKSTATION, elle est en un seul exemplaire pour chaque implantation GKS.

liste des entrées : assignée et initialisée par OPEN GKS, détruite par CLOSE GKS, elle est en un seul exemplaire pour chaque implantation GKS qui autorise le mode d'entrée event.

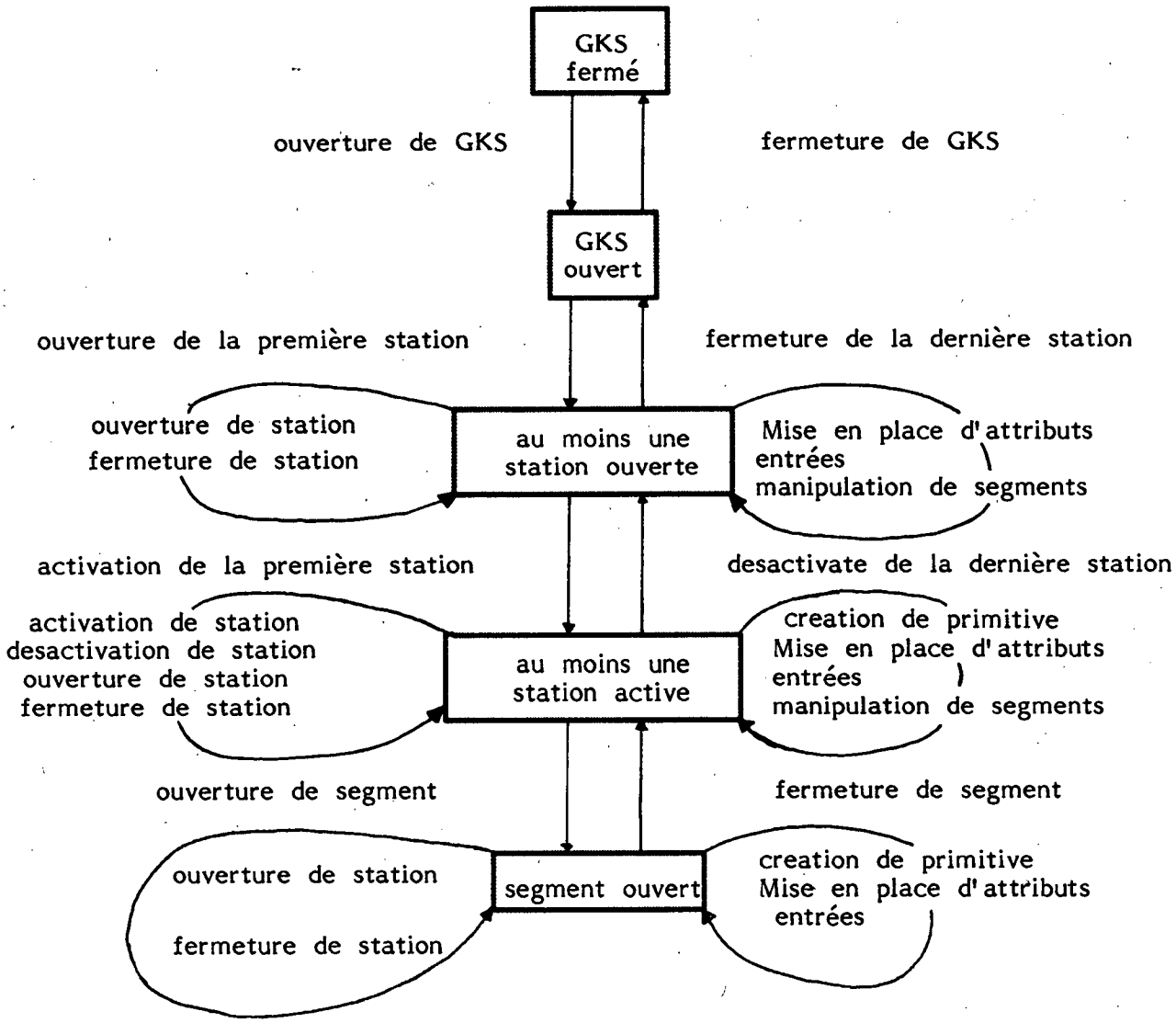
Au cours d'une application GKS, les valeurs de la liste d'état changeront. De plus, GKS contient des tables de description comportant des valeurs statiques. La "GKS description table", table de description de GKS, contient l'information sur l'implantation de GKS (par exemple le niveau de GKS).

La "workstation description table", table de description de la station, contient l'information sur les types de stations disponibles sur une implantation.

c) Interrogation des valeurs situées dans les listes d'état

Toutes les valeurs contenues dans une liste d'état ou dans une table de description peuvent être intérrogées par le programme d'application, par l'intermédiaire de fonctions d'interrogation.

Transitions entre les états, et fonctions autorisées



10 - TRAITEMENT D'ERREUR

Chaque implantation de GKS doit mettre en place une possibilité de traitement d'erreur qui doit pouvoir découvrir les erreurs et enregistrer au moins une des erreurs apparues, sur un fichier de message. L'erreur identifiée et sa fonction GKS associée, sont enregistrées sur fichier. Le programme d'application peut remplacer la procédure de traitement d'erreur standard (GKS) par son propre programme qui peut comporter des réactions aux erreurs, spécifiques.

2ème partie : Utilisation des fonctions de base de GKS

I - ETATS ET LISTES D'ETAT

II - STATIONS

III - TRANSFORMATIONS

I - ETATS ET LISTES D'ETATS

1 - INTRODUCTION

Il y a quelques 100 fonctions de GKS qui ont un effet sur ses états. Les structures des données sont groupées dans les sous-ensembles suivants :

- Etat opératoire,
- liste des états où il y a erreur,
- table de description GKS,
- liste d'état de GKS,
- table de description des stations,
- liste d'état des stations,
- liste d'état des segments.

Les tables de description contiennent des entrées qui décrivent les restrictions de l'implantation et les caractérisques de la station. Les entrées d'une table peuvent varier d'une implantation à une autre : elles sont initialisées par l'installateur de GKS.

Les listes d'état contiennent des variables d'état qui peuvent être initialisées par les fonctions GKS. Au début, les listes d'état sont initialisées par les tables de description ou par l'installateur.

Toutes les valeurs des listes d'états et des tables peuvent être interrogées par un programme d'application (fonctions inquire).

Les tables de description permettent aux programmes d'application d'adapter leurs comportements aux possibilités d'une implantation ou d'une station.

Les listes d'état de GKS reflètent l'état en cours de GKS.

2 - ETATS

5 états de GKS ont été définis :

GKCL = GKS fermé
GKOP = GKS ouvert
WSOP = au moins, une station ouverte
WSAC = au moins, une station active
SGOP = segment ouvert.

Pour chacun de ces états opératoires, il existe des listes d'état. L'état initial de GKS est GKCL.

OPEN GKS (GKOP) créé et initialise la liste d'état GKS et rend accessible les tables de description GKS et des stations.

OPEN WORKSTATION (WSOP) assigne une station particulière. Quand la première station de travail (s.t.) est ouverte, GKS se déplace de l'état GKOP à l'état WSOP. Une fois qu'une s.t. a été ouverte, les appareils d'entrée (réticules, souris) sont disponibles en mode Request.

Les possibilités de sorties graphiques d'une s.t. sont accessibles par ACTIVATE WORKSTATION. Quand la première s.t. est activée, GKS se déplace de l'état WSOP à l'état WSAC.

CREATE SEGMENT déplace GKS de l'état WSAC à l'état SGOP. CREATE SEGMENT créé et initialise la liste d'état de ce segment.

CLOSE SEGMENT déplace GKS de l'état SGOP à WSAC.
Chaque appel de DEACTIVE WORKSTATION interdit toute sortie sur la s.t. en question.

La désactivation de la dernière station déplace GKS de l'état WSAC à l'état WSOP. Chaque appel de CLOSE WORKSTATION interdit à la s.t. toute entrée et détruit la liste d'état de la s.t.

La fermeture de la dernière s.t. déplace GKS de l'état WSOP à l'état GKOP. CLOSE GKS déplace GKS de l'état GKOP à l'état GKCL.

Tableau de la validité des listes d'états et des tables de description

état opératoire : GKCL, GKOP, WSOP, WSAC, SGOP
liste d'état d'erreur : GKCL, GKOP, WSOP, WSAC, SGOP
table de description de GKS : GKOP, WSOP, WSAC, SGOP
table de description d'une w.s. : GKOP, WSOP, WSAC, SGOP
liste d'état de GKS : GKOP, WSOP, WSAC, SGOP
liste d'état des w.s. : WSOP, WSAC, SGOP
liste d'état des segments : WSOP, WSAC, SGOP

3 - FONCTIONS ADMISES POUR CHAQUE ETAT

L'effet d'une même fonction peut être différent dans des états distincts. Par exemple, si des fonctions de sortie sont appelées en WSAC, la sortie graphique ne sera générée qu'une fois ; par contre dans l'état SGOP, la sortie sera mémorisée dans un segment. Dans tout état, les fonctions d'interrogation (fonctions inquire) et les fonctions de traitement d'erreur peuvent être appelées.

Certaines entrées de la liste d'état GKS ont un sens seulement dans certains états : par exemple les noms des segments en cours et la liste d'entrée existent seulement dans les états WSOP, WSAC, SGOP. Dans l'état GKCL, Open GKS peut être appelée. Dans l'état GKOP, Close GKS, open W.S. et quelques fonctions d'initialisation des entrées de la liste d'état de GKS (attributs de primitives indépendantes des stations et transformations de normalisation) peuvent être appelées. La fonction ESCAPE peut aussi être utilisée.

L'ouverture et la fermeture de s.t. sont admises à condition qu'elles n'affectent pas la s.t. active.

ASSOCIATE SEGMENT WITH WORKSTATION et COPY SEGMENT TO WORKSTATION déplacent les données de sortie graphique vers une station.

4 - FONCTIONS ELEMENTAIRES DE CONTROLE

Les fonctions élémentaires de contrôle de GKS sont nécessaires pour générer des sorties graphiques au plus bas niveau de GKS (niveau 0A)

	INTERFACE FORTRAN
OPEN GKS	CALL GOPKS (ERRFIL)
OPEN WORKSTATION	CALL GOPWK (DISPL, CONID, TYPE)

ACTIVATE WORKSTATION	CALL GACWK (DISPL)
CLEAR WORKSTATION	CALL GCLRWK (WKID, CFLAG)
DEACTIVATE WORKSTATION	CALL GDAWK (DISPL)
CLOSE WORKSTATION	CALL GCLWK (DISPL)
CLOSE GKS	CALL GCLKS

II - STATIONS

1 - INTRODUCTION

Les possibilités d'une station de travail (ou s.t.) sont décrites dans la table de description de la s.t. Selon les possibilités d'entrée ou de sortie d'une s.t., on peut distinguer trois catégories de station :

OUTPUT : sortie seulement
INPUT : entrée seulement
OUTIN : entrée et sortie.

De plus, GKS a trois facilités particulières qui permettent un stockage permanent ou temporaire de l'information graphique :

WISS : stockage de segment indépendemment des stations
MO : métafichier de GKS en sortie
MI : métafichier de GKS en entrée.

Ces facilités sont considérées comme des stations pour des raisons de contrôle. Vis à vis de certaines fonctions GKS, elles ont des caractéristiques tout à fait différentes (par exemple une table de description plus réduite).

2 - FONCTIONS GKS S'APPLIQUANT AUX STATIONS

Seulement 4 fonctions GKS s'appliquent à tous les types de station :

OPEN WORKSTATION
CLOSE WORKSTATION
INQUIRE WORKSTATION CONNECTION AND TYPE
ESCAPE

Les stations de sortie (OUTPUT) sont affectées par toutes les fonctions générant ou contrôlant des sorties. Les stations (INPUT) sont affectées par toutes les fonctions générant ou contrôlant des entrées, sauf PICK input.

Les stations d'entrée/sortie sont affectées par les deux groupes de fonctions, et en plus, par toutes les fonctions relatives à PICK input.

Metafifle Output (MO) est affectée par les mêmes fonctions que les stations de sortie, sauf pour INQUIRE TEXT EXTENT. Il existe une fonction qui s'applique uniquement aux métafiles output : il s'agit de WRITE ITEM TO GKSM.

Metafile input (MI) est seulement affectée par deux fonctions particulières : GET ITEM TYPE FROM GKSM et READ ITEM FROM GKSM.

Les WISS se comportent comme des stations de sortie, à la restriction près que les fonctions relatives à l'écran (les transformations de station par exemple) ne s'appliquent pas. D'autre part les fonctions d'interrogation ne s'appliquent pas, car la plupart des entrées de la liste d'état de la station et de la table de description de la station n'existent pas.

3 - CONTROLE DE STATION

Avant que le programme d'application puisse utiliser une station, on doit utiliser la fonction Open Workstation.

Le premier paramètre est l'identificateur de la station. Un deuxième paramètre est le type de station qui sélectionne la table de description de la station et le driver de la station. Un autre paramètre (qui peut être redondant avec le précédent) est l'identificateur de la connection (par exemple 6 pour l'écran en Fortran).

Open Workstation permet la disponibilité de la liste d'état de la station, et l'utilisation des dispositifs d'entrée, ainsi que la manipulation des segments.

Active Workstation permet les sorties graphiques. Une station active est rendue inactive par la fonction "Deactive Workstation".

Une station ouverte est fermée par la fonction "Close Workstation". Dans le cas d'une station graphique qui est un terminal avec moniteur et espace de travail, il est intéressant de construire une fonction GKS qui ouvre et active la station, ainsi qu'une autre fonction qui désactive la station, ferme la station et ferme GKS.

4 - APPEL DES POSSIBILITES SPECIFIQUES DE LA STATION

Par la mise en place de la liste d'état de la station, le programme peut utiliser beaucoup de caractéristiques de la station. Néanmoins, certaines caractéristiques ne sont pas comprises dans GKS. Des fonctions particulières permettent d'accéder à ces caractéristiques exceptionnelles (leur utilisation réduit la portabilité du programme). Ces fonctions sont :

GDP qui concerne les primitives supplémentaires (par exemple le cercle)
MESSAGE pour converser avec un opérateur d'une autre station
ESCAPE pour appeler une installation spécifique à une station (par exemple un dispositif qui a la possibilité d'exécuter des opérations logiques sur des matrices point par point et de générer l'image résultat)

Plus d'une surface d'exposition peuvent être controllées par GKS en définissant plusieurs stations.

Table of GKS functions and corresponding states

Control Functions	
OPEN GKS	GKCL
CLOSE GKS	GKOP
OPEN WORKSTATION	GKOP,WSOP,WSAC,SGOP
CLOSE WORKSTATION	WSOP,WSAC,SGOP
ACTIVATE WORKSTATION	WSOP,WSAC
DEACTIVATE WORKSTATION	WSAC
CLEAR WORKSTATION	WSOP,WSAC
REDRAW ALL SEGMENTS ON WORKSTATION	WSOP,WSAC,SGOP
UPDATE WORKSTATION	WSOP,WSAC,SGOP
SET DEFERRAL STATE	WSOP,WSAC,SGOP
MESSAGE	WSOP,WSAC,SGOP
ESCAPE	GKOP,WSOP,WSAC,SGOP
Output Functions	WSAC,SGOP
Output Attributes	
Workstation-Independent Primitives Attributes	GKOP,WSOP,WSAC,SGOP
Workstation Attributes (Representations)	WSOP,WSAC,SGOP
Transformation Functions	
Normalization Transformation	GKOP,WSOP,WSAC,SGOP
Workstation Transformation	WSOP,WSAC,SGOP

Segment Functions	
Segment Manipulation Functions	
CREATE SEGMENT	WSAC
CLOSE SEGMENT	SGOP
RENAME SEGMENT	WSOP,WSAC,SGOP
DELETE SEGMENT	WSOP,WSAC,SGOP
DELETE SEGMENT FROM WORKSTATION	WSOP,WSAC,SGOP
ASSOCIATE SEGMENT WITH WORKSTATION	WSOP,WSAC
COPY SEGMENT TO WORKSTATION	WSOP,WSAC
INSERT SEGMENT	WSAC,SGOP
Segment Attributes	WSOP,WSAC,SGOP
Input Functions	WSOP,WSAC,SGOP
Metafile Function	
WRITE ITEM ON GKSM	WSAC,SGOP
GET ITEM TYPE FFROM GKSM	WSOP,WSAC,SGOP
READ ITEM FROL GKSM	WSOP,WSAC,SGOP
INTERPRET ITEM	WSOP,WSAC,SGOP
Inquiry Functions	allowed in all states but meaningful only in :
Inquiry Functions for Operating State Value	GKCL,GKOP,WSOP,WSAC,SGOP
Inquiry Functions for GKS Description Table	GKOP,WSOP,WSAC,SGOP
Inquiry Functions for GKS State List	GKOP,WSOP,WSAC,SGOP
except :	
INQUIRE NAME OF OPEN SEGMENT	SGOP
INQUIRE SET OF SEGMENT NAMES IN USE	WSOP,WSAC,SGOP
INQUIRE MORE SIMULTANEOUS EVENTS	WSOP,WSAC,SGOP
Inquiry Functions for Workstation State List	WSOP,WSAC,SGOP
Inquiry Functions for Workstation Description Table	GKOP,WSOP,WSAC,SGOP
Inquiry Functions for Segment State List	WSOP,WSAC,SGOP
Pixel Inquiries	WSOP,WSAC,SGOP
Inquiry Function for GKS Error State List	WSOP,WSAC,SGOP
Utility Functions	GKOP,WSOP,WSAC,SGOP
Error Handling	GKCL,GKOP,WSOP,WSAC,SGOP

III - TRANSFORMATION

1 - SYSTEMES DE COORDONNEES

Les coordonnées de l'utilisateur doivent être transformées en coordonnées de l'écran de la station. Les coordonnées de l'utilisateur, appelées world coordinates (notées W.C), doivent être cartésiennes. La transformation des coordonnées W.C en coordonnées de la station (Device Coordinates, notées D.C.) se fait en deux étapes. Dans une première étape, les coordonnées WC sont transformées en coordonnées normalisées (notées NDC). Cet espace NDC est utilisé pour le stockage des segments et pour les métafichiers. Théoriquement, l'espace NDC est illimité. Cependant tant que les transformations de segments ne sont pas disponibles, on ne peut afficher à l'écran que la partie de l'image située dans le cadre $[0,1] \times [0,1]$. Sinon, certaines parties de l'image (segments) peuvent être "ramenées" par transfomation de segment, à l'intérieur du cadre $[0,1] \times [0,1]$.

La surface d'écran de chaque station est décrite en coordonnées cartésiennes notées DC (device coordinates). Alors que la surface NDC est carrée, la surface DC de chaque station peut être un rectangle quelconque dont les dimensions sont accessibles dans la table de description de la station.

2 - TRANSFORMATION DE NORMALISATION

Le programme d'application peut composer une image à partir de différentes sources qui ont chacune leur propre système de coordonnées W.C.

La transformation de normalisation est un moyen de relier les différents systèmes de coordonnées WC, en donnant une échelle et en situant les parties de l'image pour former une image dans l'espace NDC. Comme les systèmes WC peuvent avoir des axes avec des dimensions différentes, les changements d'échelle peuvent varier suivant les axes. On dit alors que la transformation de normalisation réalise un changement d'échelle non uniforme. Une transformation de normalisation est définie par deux rectangles :

(window) la fenêtre : dans l'espace WC

(viewport) la clôture : dans le cadre $[0,1] \times [0,1]$ de l'espace NDC.

La transformation de normalisation envoie le contenu de la fenêtre dans la clôture. Les valeurs de la fenêtre et de la clôture sont stockées dans la liste d'état GKS. Les différentes transformations de normalisation sont identifiées par un nombre entier compris entre 0 et n, où n peut être retrouvé dans la table de description de GKS. Par défaut la transformation de normalisation est l'identité (notée 0), les fenêtres et les clôtures sont des carrés unitaires. A un instant donné, il ne peut y a voir qu'une seule transformation de normalisation active.

Différentes transformations de normalisation peuvent être obtenues en redéfinissant la fenêtre ou la clôture de la transformation de normalisation en cours. Toutes les transformations de normalisation (sauf la transformation 0) peuvent être redéfinies à tout moment. Cette facilité est, au niveau 0 (le plus bas niveau), la seule façon de définir différentes transformations de normalisation.

3 - TRANSFORMATION DE STATION

La transformation de station transforme l'espace NDC sur l'espace des coordonnées de la station (ou espace DC). Par défaut, cette transformation est initialisée de telle sorte que le carré $[0,1] \times [0,1]$ de NDC est transformé sur le plus grand carré inclus dans l'écran de la station. Le programme d'application peut changer de transformation de station. Contrairement à la transformation de normalisation, la transformation de station ne peut pas être utilisée pour réunir des images obtenues par différentes transformations de station. De plus, le changement de transformation détruit toutes les primitives en dehors des segments. Il est recommandé de changer de transformation de station, immédiatement après un CLEAR WORKSTATION (effacement de l'écran). On définit la transformation de station par :

- une fenêtre (à l'intérieur du cadre $[0,1] \times [0,1]$ de NDC) appelée fenêtre de station.

- une clôture (à l'intérieur des limites de l'écran) appelée clôture de station.

La fenêtre de station et la clôture de station sont des rectangles qui peuvent être créés de façon indépendante. La transformation de station n'autorise que les changements d'échelle uniformes. La clôture de station est un rectangle situé dans le coin inférieur gauche de l'écran, laissant éventuellement de la place dans le coin supérieur droit. Les principaux objectifs pour définir la transformation de station sont :

- d'utiliser complètement des surfaces d'affichage (écran) non carrées, en choisissant une fenêtre de station ayant les mêmes proportions que ces surfaces d'affichage.

- de faire un zoom sur l'image toute entière, en choisissant une fenêtre de normalisation plus petite.

- de dessiner une image à une échelle correcte en choisissant la taille appropriée de la clôture de station.

4 - DECOUPAGE

GKS fournit un mécanisme qui permet de n'afficher que les primitives à l'intérieur d'un rectangle de découpage (bords compris). Ce mécanisme s'applique à toutes les primitives graphiques. Pour la fonction POLYMARKER, par exemple, le marqueur n'est visible que si la position de ce marqueur se trouve à l'intérieur du rectangle. Pour le texte, il existe un attribut de précision du texte qui précise si :

- le découpage est fait de manière précise (précision STROKE)
- le découpage est fait en respectant l'intégralité de chaque caractère (précision CHAR)
- le découpage est fait de façon dépendante de l'implémentation et de la station (STRING).

En GKS les transformations de normalisation et de station sont associées au découpage. Les rectangles respectifs de découpage sont la clôture de normalisation et la fenêtre de station. Le découpage associé à la transformation de normalisation est différé. Le rectangle de découpage est envoyé (comme un attribut) dans le pipeline de transformation (si nécessaire, dans le stockage de segment). Le découpage est réalisé, lorsqu'il y a affichage. Ceci permet la combinaison de processus de découpage, en ne retenant que l'intersection de la fenêtre de station avec la clôture de normalisation.

Le découpage de la transformation de normalisation est fait au niveau de la clôture. La transformation de normalisation peut être changée de façon arbitraire durant la création d'une image, entraînant ainsi le changement correspondant du rectangle de découpage. Aussi il existe un indicateur de découpage, qui autorise ou

n'autorise pas le découpage par la transformation de normalisation (clipping indicator).

Le découpage de la transformation de station est fait au niveau de la fenêtre de station. Il ne peut pas être supprimé, donc on est sûr qu'aucun graphisme situé à l'extérieur de la fenêtre de station ne sera affiché.

Les transformations de segments (rotation, translation) s'appliquent aux coordonnées et aux attributs géométriques des primitives, mais n'ont aucun effet sur le rectangle de découpage stocké. Ceci permet qu'un segment soit "zoomé" à l'intérieur d'un rectangle de découpage donné et sans interférer avec le reste de l'image. Il y a une fonction de segment qui affecte les rectangles de découpage stockés : il s'agit de INSERT SEGMENT. Quand on appelle cette fonction, tous les rectangles de découpage associés au segment, sont remplacés par le rectangle de découpage en cours (i.e. la clôture de la transformation de normalisation active).

5 - ENTREES PAR RELEVEUR DE COORDONNES (LOCATOR) ET PAR RELEVEUR D'UNE SUITE DE COORDONNEES (STROKE)

Quand on définit des positions par le releveur de coordonnées ou par le releveur d'une suite de coordonnées, l'opérateur désigne des positions à l'intérieur de l'écran.

Ces positions DC sont transformées, par l'intermédiaire de l'espace NDC, dans l'espace WC, avant de les traiter dans le programme d'application. Ceci permet l'usage des entrées "Locator" et "Stroke" pour la création de primitives graphiques, à la position indiquée par l'opérateur.

A chaque instant, il n'y a qu'une seule transformation de station valable pour toutes les primitives visibles d'une station donnée. Par conséquent, la transformation de NDC vers DC est toujours bien définie, ce qui permet de définir son inverse, c'est la transformation de DC vers NDC. Pour être sûr que l'on ne sort pas du cadre $[0,1] \times [0,1]$ de NDC, les données d'entrée sont réservées à l'image de la fenêtre de station qui est toujours plus petite ou égale à la clôture de station.

Entre les espaces WC et NDC, le problème est plus compliqué. Généralement plusieurs transformations de normalisation ont été utilisées pour créer une image. GKS choisit parmi plusieurs transformations de normalisation, celle dont la clôture contient la (ou les) valeurs(s) NDC de la position du locator (resp. les positions strokes).

Cependant ce critère n'est pas suffisant, car les clôtures associées aux différentes transformations de normalisation peuvent se chevaucher. Pour cette raison, GKS a introduit la notion de priorité de clôture en entrée, qui définit une "hiérarchie" stricte entre les clôtures. La priorité d'une transformation de normalisation est déterminée par rapport à une transformation de normalisation. Par défaut, les transformations de normalisation ont des priorités de clôture en entrée, selon le numéro associé à la transformation (la plus haute priorité étant attribuée à la transformation de normalisation 0).

En testant toutes les clôtures par ordre de priorité décroissant, on peut déterminer une clôture contenant toutes les positions d'entrée.

Tant que les entrées sont générées de façon synchrone (modes Request et Sample), il n'y a pas de difficulté pour déterminer les dernières transformations valides.

Ce n'est plus le cas lorsque l'on utilise le mode Event (les positions sont stockées dans une liste d'entrée). Pour cette raison, il est conseillé de ne pas changer les transformations pendant que le locator ou le stroke sont en mode Event, ou pendant qu'il y a encore des données dans la liste d'entrée.

Comme le choix d'une transformation de normalisation pour la transformation permettant l'entrée des données dépend des positions et des priorités, il n'est pas évident pour le programme d'application de connaître quelle transformation a été utilisée. Par conséquent, GKS donne, avec les valeurs WC, le numéro de la transformation correspondante, au programme d'application.

Pour être sûr qu'il existe toujours une clôture valable pour les transformations Locator et Stroke, il existe la transformation de normalisation 0 qui transforme le carré unitaire de WC, en le carré unitaire de NDC. Cette clôture contient toute les valeurs NDC possibles pour l'entrée par Locator ou Stroke.

FONCTIONS GKS - NIVEAU OA

(Interface Fortran)

1 - CONTROLE DE GKS ET DES STATIONS
2 - TRANSFORMATIONS
3 - POLYLINE
4 - POLYMARKER
5 - TEXT
6 - FILL AREA
7 - CELL ARRAY
8 - FONCTIONS D'INTERROGATION

CONTROLE DE GKS ET DES STATIONS

GOPKS (open GKS)	(ERFIL) ouverture de GKS
GOPWK (open workstation)	(WKID, CONID, WTYPE) ouverture de la station WKID
GACWK (activate workstation)	(WKID) la station WKID est activée
GCLRWK (clear workstation)	(WKID, COFL) l'écran de la station WKID est effacé, selon le mode COFL
GDAWK (deactivate workstation)	(WKID) la station WKID est désactivée
GCLWK (close workstation)	(WKID) fermeture de la station WKID
GCLKS (close GKS)	fermeture de GKS
GESC (escape)	(FCTID, LDR, DATAREC) création d'une fonction FCTID faisant appel à des possibilités de la station, non couvertes par GKS

paramètres	signification
ERFIL	numéro du fichier des messages d'erreur
WKID	identificateur de la station (entier)
CONID	identificateur de la connection (n° de fichier)
WTYPE	type de la station
COFL	drapeau de contrôle pour l'effacement (O = condition° ; 1 = toujours)
FCTID	identificateur de la fonction créée (entier)
LDR	longueur de l'enregistrement des données (entier)
DATAREC (LDR)	enregistrement des données

TRANSFORMATIONS

GSELNT (select normalization transformation)	(TNR) la transformation de normalisation, en cours, a pour numéro associé TNR
GSWN (set window)	(TNR, XMIN, XMAX, YMIN, YMAX) définition des limites de la fenêtre de la transformation de normalisation TNR
GSVP (set viewport)	(TNR, XMIN, XMAX, YMIN, YMAX) définition des limites de la clôture de la transformation de la normalisation TNR
GSWKWN (set workstation window)	(WKID, XMIN, XMAX, YMIN, YMAX) définition des limites de la fenêtre de la station WKID
GSWKVP (set workstation viewport)	(WKID, XMIN, XMAX, YMIN, YMAX) définition des limites de la clôture de la station WKID
GSCLIP (set clipping indicator)	(CLSW) choix de l'option de découpage, suivant les valeurs de CLSW
GSVPIP (OB) (set viewport input priority)	(TNR, RTNR, RELPRI) définition de la priorité de la clôture en entrée, par rapport à une transformation de normalisation de référence

paramètres	
TNR	numéro de la transformation de normalisation (entier de 0 à n)
XMIN,XMAX YMIN, YMAX	limites d'une fenêtre, d'une clôture, d'une fenêtre de station, ou d'une clôture de station
WKID	identificateur de la station (entier)
CLSW	vaut 1 s'il y a découpage, 0 sinon
RTNR	numéro de la transformation de normalisation de référence
RELPRI	vaut 0 si la priorité relative est au dessus de la transformation de normalisation de référence, vaut 1 sinon

POLYLINE

GPL
(polyline)

(N, PX, PY)
une suite de lignes droites joint les N points de l'espace WC dont les coordonnées sont indiquées dans les vecteurs PX et PY (fonction polyline)

GSLN
(set linetype)

(LTYPE)
définit le type de ligne (plein, tireté, ...) que la fonction utilise

GSLWSC
(set polyline scale factor)

(LWIDTH)
définit l'épaisseur de ligne qu'utilise la fonction polyline

GSPLCI
(set polyline colour index)

(COLI)
définit la couleur des lignes qu'utilise la fonction polyline

GSPLI
(set polyline index)

(INDEX)
définit l'indice de la fonction polyline, qui est utilisé pour créer une nouvelle forme de fonction polyline en mode bundled (dépendant)

paramètres	signification
PX	vecteur de dimension N précisant les coordonnées en x des points à joindre
PY	vecteur de dimension N précisant les coordonnées en y des points à joindre
LTYPE	1 = ligne pleine ; 2 = ligne tiretée ; 3 = ligne pointillée ; 4 = ligne tiretée-pointillée ; > 4 = dépend de la station
LWIDTH	réel définissant le facteur multiplicatif de l'épaisseur de la ligne
COLI	indice définissant la couleur (entier de 0 à n)
INDEX	indice de la fonction polyline (entier de 1 à n)

POLYMARKER

GPM (polymarker)	(N,PX,PY) une suite de N marqueurs est créée. Les marqueurs occupent les positions indiquées par les vecteurs PX et PY dans l'espace WC (fonc. polymarker)
GSMK (set marker type)	(MTYPE) définit le type de marqueurs que la fonction polymarker utilise
GSMKSC (set marker size scale factor	(LWIDTH) définit la taille du marqueur que la fonction polymarker utilise (le type 1 n'est pas affecté par cet attribut)
GSPMCI (set polymarker coulor index)	(COLI) définit la couleur des marqueurs qu'utilise la fonction polymarker
GSPMI (set polymarker index)	(INDEX) définit l'indice de la fonction polymarker, utilisé en mode bundled (asf = 0) pour établir les attributs de la fonction polymarker
GSPMR niveau 1A (set polymarker repre-sentation)	(WKID, PMI, MTYPE, MSZSF, COLI) chaque valeur de l'indice PMI définit les attributs MTYPE, MSZSF et COLI de la fonction polymarker à condition d'être en mode bundled et que les drapeaux asf aient les valeurs nécessaires. Par défaut, PMI vaut 1.

paramètres	signification
PX	vecteur de longueur N, précisant les coordonnées en x des positions des marqueurs
PY	vecteur de longueur N, précisant les coordonnées en y des positions des marqueurs
MTYPE	1 = . ; 2 = + ; 3 = * ; 4 = 0 ; 5 = X ; > 5 = dépend de la station
LWIDTH ou MSZSF	réel définissant le facteur multiplicatif de la taille du marqueur
COLI	indice définissant la couleur (entier de 0 à n)
INDEX ou PMI	indice de la fonction polymarker (entier de 1 à n)
WKID	identificateur de la station (entier)

TEXT

GTX (text)	(XO, YO, CHARS) une chaîne de caractères est créée. La position de départ est (XO,YO) dans l'espace WC (fonction text)
GSCHH (set character height)	(CHH) définit la hauteur des caractères dans l'espace WC. Le rapport largeur/hauteur reste constant
GSCHUP (set character up vector)	(CHUX, CHUY) définit la direction selon laquelle le texte va être écrit ; le vecteur (CHUX, CHUY), est orthogonal à cette direction
GSTXCI (set text colour index)	(COLI) définit la couleur des caractères que génère la fonction text
GSTXCI (set text index)	(INDEX) définit l'indice de la fonction text, utilisé en mode bundled (asf = 0) pour préciser les attributs de la fonction text employés
GSTXP 1A (set text path)	(TXP) précise l'orientation selon laquelle le texte va être écrit (vers la droite, vers la gauche, vers le haut, vers le bas)
GSTXAL (set text alignement)	(TXALH, TXALV) précise l'intersection de quelles lignes, servant à définir l'architecture des caractères, sera employée en tant que position de départ utilisée dans la fonction GTX
GSTXFP (set text font and precision)	(FONT, PREC) définit le style du texte (font) généré par GTX (par défaut, on prend la font 1 définie par l'ensemble des caractères ASCII). La précision du texte (prec) détermine la fidélité avec laquelle les autres aspects du texte (GSCHH, GSCHUP, ..., GSCHSP) sont employés
GSCHXP (set character expansion factor)	(CHXP) définit le rapport largeur/hauteur des caractères générés par la fonction text
GSCHSP (set character spacing)	(CHSP) précise l'espace entre deux caractères générés par la fonction text
GSTXR 1A (set text representation)	(WKID, INDEX, FONT, PREC, CHXP, CHSP, COLI) cette fonction permet la sélection des attributs text font and precision, character expansion factor, character spacing, et text colour index, pour chaque station individuellement. Ces valeurs sont accessibles par text index, si les valeurs respectives des asf sont 0.

PARAMETRES DU TEXTE

paramètres	signification
(XO, YO)	coordonnées du point de départ de la chaîne de caractères créée par la fonction text
CHARS	chaîne de caractères
CHH	réel > 0, précisant la hauteur des caractères dans WC ; la valeur par défaut est 0.01 : soit 1/100 de la fenêtre
(CHUX, CHUY)	coordonnées du vecteur perpendiculaire à la direction selon laquelle le texte va être écrit, dans l'espace WC
COLI	indice définissant la couleur (entier de 0 à n)
INDEX	indice de la fonction text (entier de 1 à n)
TXP	0 = right ; 1 = left ; 2 = up ; 3 = down
(TXALH, TXALV)	pour TXALH : 0 = normal ; 1 = left ; 2 = centre ; 3 = right pour TXALV : 0 = normal ; 1 = top ; 2 = cap ; 3 = half ; 4 = base ; 5 = bottom
FONT	entier variant de 1 à n : 1 = caractères ascii
PREC	0 = string ; 1 = char ; 2 = stroke valeurs
CHXP	réel > 0 ; la valeur par défaut est 1, on a alors largeur = hauteur
CHSP	réel ; la valeur par défaut est 0, dans ce cas les rectangles d'extension des caractyères se touchent. Une valeur négative entraîne un chevauchement des caractères
WKID	identificateur de la station (entier)

FILL AREA

GFA (fill area)	(N, PX, PY) le polynôme défini par les N points de coordonnées PX, PY est rempli selon le style de remplissage choisi. La frontière n'est tracée que pour le style Hollow (vide)
GSFAIS (set fill area interior style)	(INTS) définit le mode de remplissage de l'aire. Par défaut, on prend le style Hollow (vide)
GSFASI (set fill area style index)	(STYLI) établit la valeur de l'indice de style. Si le style est holow ou solid, cet indice n'est pas utilisé. Pour les styles pattern et hatch, il précise quel motif ou quelle hachures seront utilisées
GSFACI (set fill area colour index)	(COLI) définit la couleur qu'utilise la fonction area (sauf pour le style pattern)
GSFAI (set fill area index)	(INDEX) définit l'indice de la fonction area, utilisé en mode bundled (c'est à dire dépendant de la station : asf = 0) pour préciser les attributs de la fonction fill area
GSPA (set pattern size)	(SX, SY) donne les dimensions des côtés du motif utilisé, selon les axes x et y de l'espace WC
GSPARF (set pattern reference point)	(RFX, RFY) donne le point de référence, c'est-à-dire la position du coin inférieur gauche, dans l'espace WC, du motif (ou pattern)
GSFAR niveau 1A (set fill area representation)	(WKID, INDEX, INTS, STYLI, COLI) cette fonction autorise le choix de interior style, style index, et colour index, pour chaque station individuellement (mode bundled). Les valeurs de ces choix sont accessibles par fill area index, si les valeurs des asf des attributs correspondants sont mis à 0. Cette function permet de définir les entrées de fill area bundle table, table qui se trouve dans la liste d'état de la station
GSPAR niveau 1A (set pattern representation)	(WKID, PAI, N, M, DIMX, COLIA) si une station, possède le style pattern, il existe une table de patterns sur, cette station décrivant les patterns disponibles pour fill arera. Les entrées de cette table peuvent être redéfinies par la fonction set pattern representation (gspar)

PARAMETRES DE FILL AREA

paramètres	signification
(PX, PY)	coordonnées réelles dans WC, des N sommets du polygone de fill area
INTS	0 = hollow ; 1 = solid ; 2 = pattern ; 3 = hatch ; définit le style de fill area
STYLI	entier de 1 à n, pour définir le style de hachure ou le style de pattern (donc dépend de la valeur de INTS : une même valeur de STYLI peut définir un style de hachure ou un style de pattern)
COLI	indice définissant la couleur (entier de0 à n)
INDEX	indice de la fonction fill area (entier de 1 à n)
(SX, SY)	couple de réels précisant la longueur des côtés du motif (ou pattern) dans l'espace WC selon les axes x et y
(RFX, RFY)	coordonnées du coin inférieur gauche du pattern dans l'espace WC
WKID	identificateur de la station
PAI	indice de pattern (entier de 1 à n) sert à mémoriser le choix de la grille du motif (i.e. nombre de colonnes et nombres de ligne du motif)
N = DIMX M	dimensions du tableau du motif (pattern). Il y a NxM cellules dans le motif : N suivant l'axe des x (c'est-à-dire N colonnes) et M suivant l'axe des y (c'est-à-dire M lignes) - couple d'entiers
COLIA	nom du tableau du motif : COLIA (N,M), il est représenté par un entier variant de 0 à n

CELL ARRAY

GCA

(Cell Array)

(PX,PY,QX,QY,DIMX,DIMY,DX,DY,COLIA)

Cell array permet de passer des images rectangulaires définies par une matrice d'indices de couleur, image dont la taille et la position sont définies dans WC. Le rectangle de cell array a ses côtés parallèles aux axes de WC. Ce rectangle est divisé en lignes et colonnes de telle sorte qu'à chaque cellule obtenue corresponde un indice de couleur de la matrice des indices de couleur. Le nombre de colonnes et de lignes du rectangle correspond aux dimensions de la matrice d'indices de couleur.
Il ne faut pas oublier que la grille ainsi définie dans l'espace WC, est sensible à toutes les transformations (rotations, transformations des rectangles en parallélogrammes, clipping...). Cell array n'a pas d'autre attribut que Pick Identifier.

paramètres	signification
(PX,PY)	coordonnées dans l'espace World Coordinates du point inférieur gauche du rectangle cell array.
(QX,QY)	coordonnées dans l'espace WC du point supérieur droit du rectangle cell array
(DIMX,DIMY)	(entiers) dimensions de la matrice des indices de couleur
DX	nombre de colonnes du rectangle cell array
DY	nombre de lignes du rectangle cell array
COLIA	matrice des indices de couleur

FONCTIONS D'INTERROGATION

Nous présentons quelques exemples de fonctions d'interrrogation (il y en a une centaine)

GQCNTN (numéro de transformation de normalisation)	(IERR, TNR) renvoie le numéro de la transformation courante de normalisation notée TNR.
GQMNTN (numéro maximum de tr. de normalisation)	(IERR, MAXTNR) renvoie le plus grand numéro de transformation de normalisation MAXTNR
GQNT (transformation de normalisation)	(TNR, IERR, RW, RV) renvoie les limites de la fenêtre (RW) et de la clôture (RV) pour la transformation de normalisation TNR
GQWKT (transformation de station)	(WKID, IERR, TUS, RW, CW, RV, CV) Pour la station WKID, cette fonction renvoie les limites de la fenêtre actuelle (RW) et par défaut (CW) ainsi que les limites de la clôture actuelle (RV) et par défaut (CV) concernant la transformation de station.
GQTXX (text extent)	(WKID, PX, PY, STR, IERR, CPX, CPY, TEX, TEY) Pour l'écriture de la chaîne de caractères STR au point de coordonnées (PX, PY) sur la station WKID, cette fonction renvoie le point de concaténation de la chaine (CPX, CPY) et l'étendue de cette chaine (TEX, TEY).

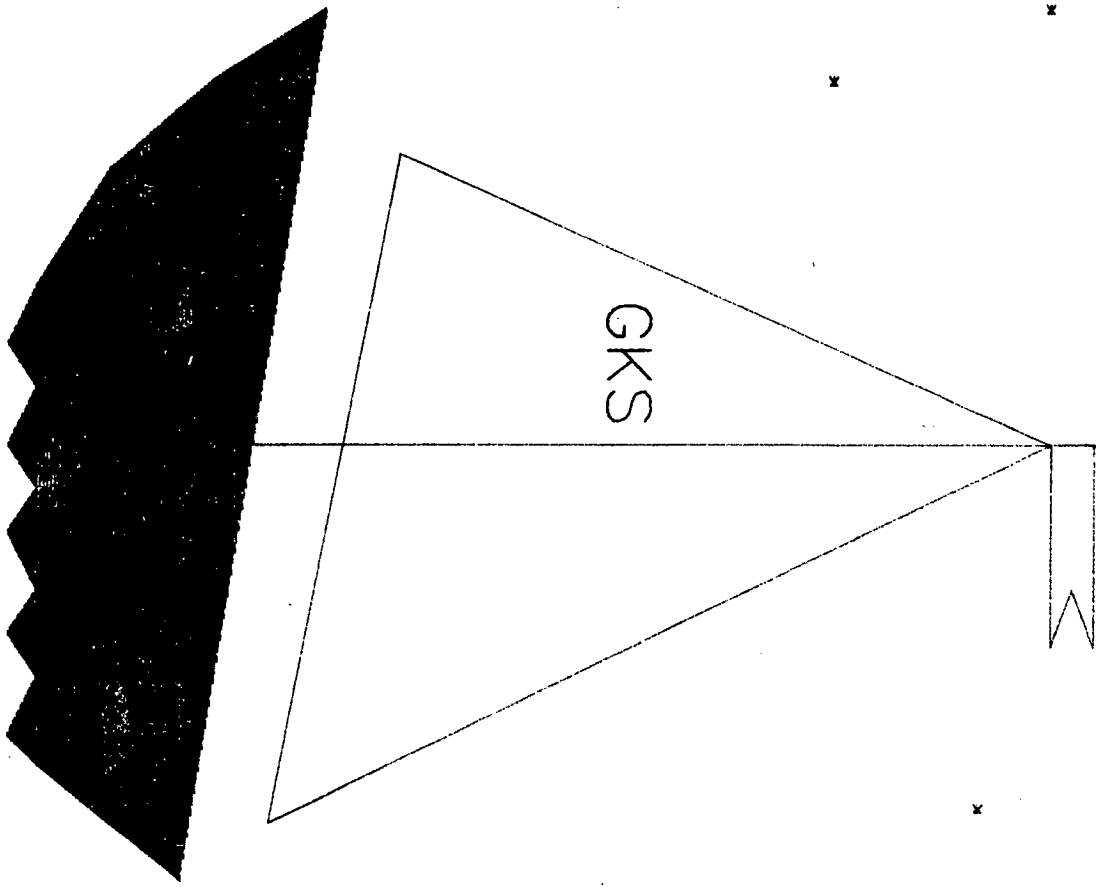
PARAMETRES DES FONCTIONS D'INTERROGATION

paramètres	signification
IERR	indice d'erreur (entier)
TNR	numéro de transformation de normalisation
MAXTNR	plus grand des numéros de transformation de normalisation
RW ou CW	tableau de dimension 4 contenant les dimensions d'une fenêtre de transformation. Plus précisément, RW(1) ou CW(1) désigne l'abscisse du bord gauche, RW(2) ou CW(2) celle du bord droit, RW(3) ou CW(3) l'ordonnée du bord inférieur, et enfin RW(4) ou CW(4) l'ordonnée du bord supérieur
RV ou CV	tableau de dimension 4 contenant les dimensions d'une clôture de transformation.
WKID	identificateur (entier) d'une station de travail
TUS	valeur binaire (0 ou 1) indiquant si le dernier changement de transformation de station demandé a bien été réalisé
(PX, PY) ou (CPX, CPY)	coordonnées dans l'espace W.C.
STR	chaine de caractères
TEX	tableau de dimension 4 indiquant les abscisses des bords du plus petit rectangle contenant la chaine de caractères STR
TEY	tableau de dimension 4 indiquant les ordonnées des bords du plus petit rectangle contenant la chaine de caractères STR

EXEMPLE DE PROGRAMME UTILISANT LES NORMES GKS

```
c ************************************************************
c
c     test 1 -   utilisation des fonctions de GKS
c
c          polyline, polymarker, text, fill area, cell area
c ************************************************************
c
      integer wkid,wtype,in,out,conid
c
c ************************************************************
c     wkid = numero de la station (1 a 10)
      wkid=1
c     wtype = type de la station (8= Traceur)
      wtype=8
c     in = numero console input
      in=5
c     out = numero console output
      out=6
c     conid = numero du fichier d'ecriture des ordres graphiques
      conid=66
c ************************************************************
c
c   ouverture de GKS
c
      call gopks(out)
      call gopwk(wkid,conid,wtype)
      call gacwk(wkid)
c
c ********************************************************
c   image 1
c          UN TRES JOLI BATEAU .....
c ********************************************************
c
      write(out,1)
    1 format(1x,'test started')
c
c      le bateau est dessine dans WC (espace utilisateur ou virtuel)
c     dans une fenetre 0,100 en X et 0,100 en Y
c      L'espace NDC (espace reel GKS) est un carre de dimension 1
c     (fenetre 0 a 1 en X ,0 a 1 en Y)
c
c ****************************************************
c
c       DEFINITION DE LA TRANSFORMATION
c            de WC dans NDC
c ****************************************************
c    transformation no 1 (ntr =1)
c      dans l'espace virtuel xmin=0,ymin=0.
c     xmax=100.,ymax=100.
c      le carre le plus grand de NDC ( 0 a 1 ,, 0 a 1)
c
      xmin=0.
      xmax=100.
      ymin=0.
      ymax=100.

      ntr=1
      call gswn(ntr,xmin,xmax,ymin,ymax)
      call gsvp(ntr,0.,1.,0.,1.)
c   appel de la transformation 1
      call gselnt(ntr)
      call bateau
      call guwk(1,1)
c
      write(out,4)
    4 format(1x,'test completed')
      call clgks(wkid)
      stop
      end
      subroutine bateau
c
c    dessine un bateau dans un cadre 0,100 et 0,100
c
      real x1(5),y1(5),x2(6),y2(6),x3(16),y3(16)
      real xc(5),yc(5)
      integer n1,n2,n3,nc
      character text*3
```

```
      data n1/5/, n2/6/, n3/16/,
     *   x1/50.,50.,30.,76.,50./,      y1/25.,80.,35.,26.,80./,
     *   x2/10.,20.,25.,75.,85.,90./, y2/70.,80.,65.,75.,85.,60./
      data x3/20.,25.,31.,39.,43.,46.,50.,53.,56.,60.,63.,66.,70.,72.,
     *76.,80./
      data y3/30.,22.,15.,10.,08.,10.,08.,10.,08.,10.,08.,10.,08.,10.,
     *15.,20./
      data  x0/40./,y0/50./, nc/3/, text/'GKS'/
      data xc/50.,64.,60.,64.,50./, yc/80.,80.,81.5,83.,83./
c
c  trace le mat du bateau par polylines
c
      call gpl(n1,x1,y1)
c
c  trace les etoiles par polymarkers
c
      call gpm(n2,x2,y2)
c
c trace un texte "GKS" sur le bateau
c
c   choix grandeur caracteres texte
c    xh = grandeur est 3./100 de la fenetre en hauteur (axe Y)
c    100. est la longueur de la fenetre virtuelle en Y
c
      xh=3.0
      call gschh(xh)
c    choix de la matrice de caracteres
c    numero ce fonte est 1 (la plus rapide)
c    precision d'ecriture 2 (bon choix ??)
c
      call gstxfp(1,2)
c    choix de la couleur (de 1 a 7)
c      0 couleur du fond
      call gstxci(4)









      call gtx(x0,y0,text)
c
c    trace le contour du drapeau
c
      call gsfais(0)
      call gsfaci(1)
      call gfa(5,xc,yc)
c
c trace la coque du bateau et colorie cette coque
c   par fill area
      call gsfais(1)
      call gsfaci(1)
      call gfa(n3,x3,y3)
c
      return
      end
```


GKS
W

3ème partie : Segments et interactivité graphique

I. SEGMENTS

II. ENTREES GRAPHIQUES

III. MODES D'INTERACTION

SEGMENTS

1 - INTRODUCTION

Dans beaucoup d'applications, il est intéressant d'afficher (sur un écran ou une table traçante) la même image, plusieurs fois, et à différents endroits.

Dans ce but, GKS a introduit le concept de segment. Des parties d'images peuvent être stockées dans des segments pendant l'exécution d'un programme et être utilisées par la suite. Les segments ont des attributs qui permettent à l'utilisateur de modifer l'apparence de tout le segment.

2 - CREATION ET DESTRUCTION DE SEGMENTS

La création d'un segment de nom ID se fait grâce à la fonction :

```
CREATE SEGMENT(ID)
```

Le segment ID est alors un segment ouvert et les appels suivants de primitives graphiques concerneront ce segment, jusqu'à ce que l'on rencontre l'instruction :

```
CLOSE SEGMENT
```

Il faut savoir qu'il ne peut y avoir, au même moment, plus d'un segment ouvert. Ainsi la suite d'instructions :

```
CREATE SEGMENT(1)
primitives graphiques
CREATE SEGMENT(2)
primitives graphiques
CLOSE SEGMENT
CLOSE SEGMENT
```

est interdite et générera une erreur.

Une fois qu'un segment est fermé, on ne peut plus lui ajouter les effets de primitives graphiques ultérieures. Plusieurs opérations peuvent être effectuées sur un segment. La plus simple est la destruction :

DELETE SEGMENT(ID)

où ID est le nom du segment à détruire.

3 - ATTRIBUTS DE SEGMENTS

a) Transformations de segment

Les transformations de segment permettent de déplacer des objets (graphiques) ou des parties d'objets sur la surface d'affichage. La transformation de segment agit sur toutes les coordonnées ayant servi à définir le segment. Quand un segment est créé, la transformation de segment est initialisée à l'identité. Cette transformation de segment peut ensuite être modifiée grâce à la fonction :

SET SEGMENT TRANSFORMATION(ID,MATRIX)

où ID est le nom du segment, et MATRIX désigne une matrice de dimensions 2 x 3. Il existe deux fonctions utilitaires qui permettent de construire les matrices de transformation. La première de ces fonctions est :

EVALUATE TRANSFORMATION MATRIX(FX,FY,TX,TY,R,SX,SY,SWITCH,MATRIX)

avec :

(FX,FY)	=	coordonnées d'un point fixé
(TX,TY)	=	coordonnées du vecteur de translation
R	=	angle de rotation (en radiens)
(SX,SY)	=	facteurs d'échelle
SWITCH	=	indice indiquant les valeurs WC ou NDC
MATRIX	=	matrice de transformation retournée (paramètre de sortie)

Les facteurs d'échelle et d'angle de rotation sont définis par rapport au point de coordonnées (FX,FY). L'indice de coordonnées, SWITCH, précise dans quel espace (WC ou NDC) se situent le point fixe (FX,FY) et le vecteur de translation (TX,TY). La matrice de transformation est calculée de telle sorte que l'ordre des opérations effectuées sur les coordonnées soient : changement d'échelle, rotation et translation.

Donnons des exemples de l'utilisation de EVALUATE TRANSFORMATION MATRIX. Le rectangle dessiné ci-dessous est défini par un polyline commençant au point (0,10) :

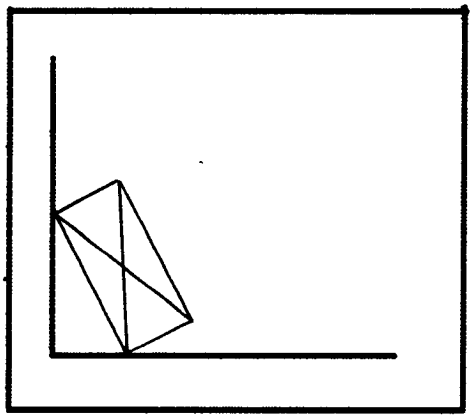


fig. 1

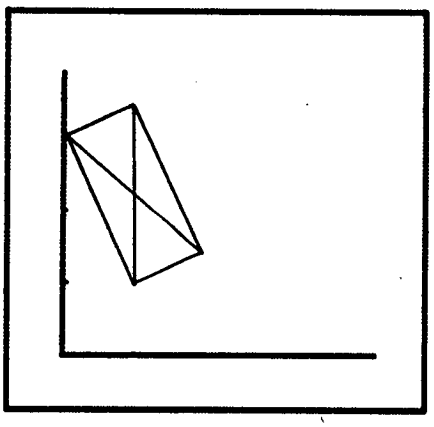


fig. 2

Supposons que nous voulions déplacer le rectangle de telle sorte qu'il commence au point (10,10) dans l'espace W.C. Nous devons préciser tous les arguments de la fonction EVALUATE TRANSFORMATION MATRIX, même si ces arguments (comme R, (FX,FY) par exemple ici) sont inutiles pour la définition de la matrice de transformation. Ici le point (0,10) doit être translaté, par le vecteur (10,10), pour obtenir le point (10,20). Aucune rotation et aucun changement d'échelle ne sont nécessaire. Ainsi, le programme complet pour, créer un segment contenant le rectangle, l'afficher au point (0,10), et le déplacer au point (10,20), sera :

```
CREATE SEGMENT(1)
RECTANGLE
CLOSE SEGMENT
FX = 0
FY = 10
TX = 10
TY = 10
R = 0
SX = 1
SY = 1
SWITCH = WC
EVALUATE TRANSFORMATION MATRIX(FX,FY,TX,TY,R,SX,SWITCH,MAT)
SET SEGMENT TRANSFORMATION(1,MAT)
```

La deuxième fonction utilitaire servant à construire les matrices de transformation, est :

ACCUMULATE TRANSFORMATION MATRIX(MATIN,FX,FY,TX,TY,R,SX,SY,SWITCH,MATOUT)

Les paramètres FX,FY,TX,TY,R,SX,SY, et SWITCH ont la même signification que pour la fonction EVALUATE TRANSFORMATION MATRIX. La transformaion définie par ces paramètres est comparée avec la transformation précisée par la matrice MATIN, et le résultat est la matrice des transformation MATOUT. L'ordre dans lequel les transformations sont appliquées est : matrice d'entrée (MATIN), facteur d'échelle, rotation et translation. Un usage courant de cette fonction est de changer l'ordre dans lequel changement d'échelle, rotation et translation sont appliqués. Pour cela, il suffit de préciser chacune de ces transformations élémentaires séparément et d'appeler d'abord une fois la fonction EVALUATE TRANSFORMATION MATRIX, puis deux fois la fonction ACCUMULATE TRANSFORMATION MATRIX, afin d'obtenir la matrice adéquate.

b) Transformation de segment et clipping

La transformation de segment est effectuée après la transformation de normalisation, mais avant tout clipping. Les coordonnées stockées dans les segments sont exprimées dans l'espace NDC. D'ailleurs les éléments de la matrice de transformation qui expriment la translation, sont exprimées en unités de l'espace NDC.

Une primitive graphique (polyline, fill area, etc...) dans un segment, est transformée par les transformations de normalisation et de segment, puis, si l'indicateur de clipping a la valeur CLIP, il y a clipping (c'est à dire choix d'une sous-fenêtre) au niveau de la cloture de normalisation. Cependant les rectangles de clipping ne sont pas affectés par la transformation de segment. Voici un exemple :

```
SET WINDOW(1,0,30,0,30)
SET VIEWPORT(1,0,0.5,0,0.5)

SELECT NORMALISATION TRANSFORMATION(1)
SET CLIPPING INDICATOR(NOCLIP)
CREATE SEGMENT(1)
CANARD
CLOSE SEGMENT

EVALUATE TRANSFORMATION MATRIX(0,0,0.25,0.25,0,1,1,NDC,MAT)
SET SEGMENT TRANSFORMATION(1,MAT)
```

Si l'indice de clipping a pour valeur NOCLIP, quand une primitive est introduite dans un segment, le rectangle de clipping [0,1] x [0,1] (en unités NDC) est stocké avec la primitive. Après la translation par le vecteur (0.25,0.25) (en unités NDC), le programme précédent a pour effet dans NDC le résultat exprimé fig. 3.

Si l'indice de clipping avait eu pour valeur CLIP, le rectangle de clipping stocké avec la primitive aurait été la cloture de la transformation de normalisation n°1 (i.e. [0,0.5] x [0,0.5]). Le canard aurait été "découpé" suivant cette sous fenêtre, dans l'espace NDC, comme le montre la figure 4.

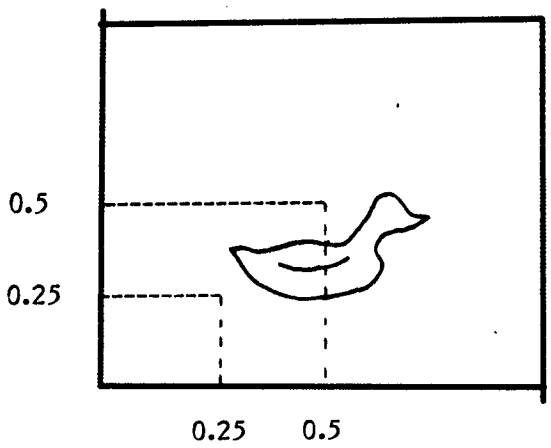


fig. 3

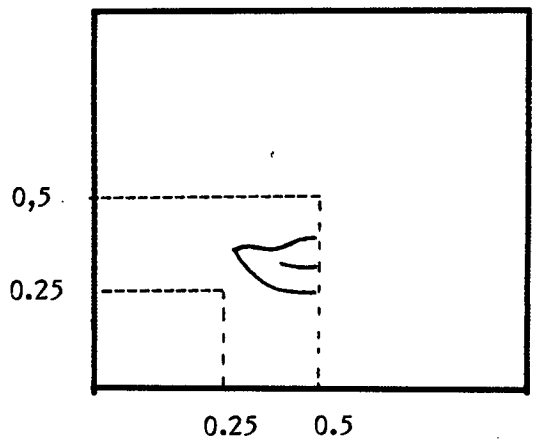


fig. 4

c) **Visibilité d'un segment**

L'attribut précise si un segment sera affiché ou non. Par défaut, quand un segment est créé, il est visible ; ainsi :

```
CREATE SEGMENT(1)
CANARD
CLOSE SEGMENT
```

aura pour résultat l'affichage d'un canard. L'attribut de visibilité est déterminé par la fonction :

```
SET VISIBILITY(ID,VIS)
```

où VIS indique la visibilité du segment ID. VIS peut prendre les valeurs VISIBLE et INVISIBLE. Si, dans les instructions précédentes, on avait inséré :

```
SET VISIBILITY(1,INVISIBLE)
```

après l'ordre CREATE SEGMENT(1), alors le canard n'aurait pas été affiché. L'attribut de visibilité est notamment utile pour contrôler l'affichage des messages et des menus. Par exemple chaque message est mis dans un segment, initialement invisible. Quand un message est demandé, le segment le contenant est rendu visible.

d) **Mise en évidence d'un segment**

Certains terminaux graphiques ont la possibilité de modifier la mise en valeur (ou mise en évidence) d'un segment, par exemple, en le faisant clignoter. Le principal intérêt est de pouvoir attirer l'attention de l'opérateur sur un aspect de l'image. Par exemple, si l'opérateur choisit une nouvelle position pour afficher un objet, on peut faire clignoter le segment contenant cet objet dans sa nouvelle position, et demander à l'opérateur de confirmer que cette position est bien celle qu'il désire.
L'attribut est établi par la fonction :

```
SET HIGHLIGHTING(ID,HIGH)
```

où HIGH précise le degré de mise en valeur du segment ID ; HIGH peut prendre pour valeur HIGHLIGHTED et NORMAL. Par défaut, quand un segment est créé, la valeur de HIGH est NORMAL.

c) Priorités entre segments

Supposons que nous voulions afficher un paysage comprenant : un ciel, un soleil, un sol, un étang, un arbre et un canard (cf. Fig. 5)

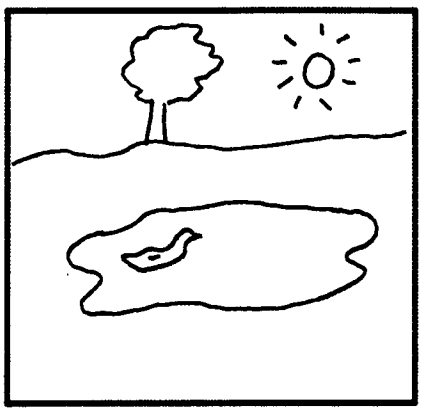


Fig. 5

Nous supposons que nous avons un écran qui affiche des images en couleurs, point par point. Pour réaliser ce paysage, nous couvrons la surface entière par un fond de couleur bleu. Puis, délimitant l'horizon, on utilise un fill area, pour représenter la couleur verte du paturage, ceci étant fait en changeant la couleur bleue de certains points en verte. En procédant de même, on affiche l'arbre, le soleil, l'étang et le canard. Il est clair que l'apparence de l'image globale dépend de l'ordre dans lequel on a exécuté les primitives de remplissage de forme.

Supposons maintenant que nous voulions à l'aide d'une série d'images décrire un coucher de soleil. Le soleil étant un segment, et l'opérateur précisant les positions successives du soleil, il n'y a pas de problème jusqu'à ce que le soleil rencontre le vert du paturage : car

le soleil recouvrira une partie de la couleur verte du sol. Si le sol avait une "priorité plus grande" que le soleil, plus précisément si les points du paturage n'étaient pas recouverts par le soleil couchant, nous aurions exactement l'effet voulu.

Pour réaliser cela, GKS fournit le mécanisme de priorité entre segments, qui est appelé grâce à la fonction :

SET SEGMENT PRIORITY(ID,PRIORITY)

où PRIORITY est un réel compris entre 0 et 1, qui précise la priorité du segment ID. Dans l'exemple traité ici, une suite correcte de priorité serait :

segments	priorités
ciel	0
soleil	0.25
paturage	0.5
étang	0.75
arbre	0.75
canard	1

Les priorités entre segments controlent l'ordre dans lequel les segments sont ré-affichés, quand l'image est changée.

En pratique, les priorités entre segments ne sont valables qu'avec des écrans couleurs à affichage point par point.

D'autre part, dans la pratique, les priorités entre segments ne peuvent prendre qu'un nombre fini de valeurs, les écrans disponibles ne pouvant distinguer qu'un nombre fini de priorités.

4 - CHANGEMENT DE NOM D'UN SEGMENT

On peut renommer un segment grâce à la fonction :

RENAME SEGMENT(OLD,NEW)

où OLD et NEW sont les anciens et nouveaux noms du segment. Il ne doit pas exister de segment ayant pour nom NEW, lors de l'appel de la fonction. Si c'est le cas, un message d'erreur apparaîtra.

On peut renommer le segment ouvert, par exemple :

```
CREATE SEGMENT(1)
primitives graphiques
RENAME SEGMENT(1,2)
primitives graphiques
CLOSE SEGMENT
```

est une suite d'instructions correcte.

ENTREES GRAPHIQUES

1 - INTRODUCTION

Beaucoup de terminaux graphiques ont des dispositifs leur permettant d'entrer l'information graphique directement, sans passer par l'intermédiaire du clavier (par exemple à l'aide d'un réticule). La grande variété des systèmes physiques d'entrée a conduit à ce que l'on définisse pour GKS, six types de systèmes logiques d'entrée.

Ces six classes sont :

(1) locator : entrée d'une position (une valeur (X,Y))
(2) pick : identification d'un objet affiché
(3) choice : sélection parmi un ensemble d'alternatives
(4) valuator : entrée d'une valeur
(5) string : entrée d'une chaîne de caractères
(6) stroke : entrée d'une séquence de positions (X,Y)

Les systèmes physiques souvent associés sont, dans l'ordre :

(1) locator : réticule
(2) pick : crayon optique pointant un objet affiché à l'écran
(3) choice : ensemble de touches ou sélection dans un menu, grâce à un crayon optique.
(4) valuator : potentiomètre ou entrée d'une valeur par le clavier
(5) string : entrée d'une ligne de texte par le clavier
(6) stroke : entrée à la tablette.

Avant de décrire les différents systèmes logiques d'entrée, nous allons décrire les différents modes dans lesquels ils peuvent être utilisés. Ici, nous ne présenterons que le mode request.

2 - MODE REQUEST

Le mode request correspond à peu près au type d'entrée utilisée en Fortran.

En mode request, un programme attend des "données" au niveau du système d'entrée logique, puis attend jusqu'à ce que le système d'entrée logique ait obtenu ses informations. Ceci étant terminé, le système d'entrée retourne l'information au programme qui continue alors à fonctionner. En mode request, ou bien le programme GKS est actif, ou bien le système d'entrée l'est, mais jamais les deux à la fois. Il ne peut y avoir plus d'un système d'entrée actif en même temps. La fonction GKS faisant appel à une entrée en mode request, est :

REQUEST XXX(WS,DV,ST,...)

XXX désigne le système d'entrée logique (locator, pick, etc...)
WS = station sur laquelle le système d'entrée se situe
DV = système physique d'entrée
ST = indicateur qui indique si l'information est bien entrée.

3 - LOCATOR

Le rôle du locator est de fournir une position (unique) au programme d'application. En mode request, la fonction d'appel du locator est :

REQUEST LOCATOR(WS,DV,ST,NORMTR,XPOS,YPOS)

avec :

(XPOS,YPOS) = position retournée par la fonction locator (espace WC)

NORMTR = transformation de normalisation utilisée pour changer les coordonées NDC en coordonnées WC (i.e. XPOS et YPOS)

La fonction locator retourne à la fois (XPOS,YPOS) et NORMTR.

a) Plusieurs clôtures de normalisation

On utilise souvent plusieurs transformations de normalisation dans le même programme. A l'écran, on trouve plusieurs clôtures correspondant aux différentes clôtures de normalisation. Lorsque l'opérateur utilise une entrée locator, il doit choisir entre les différentes clôtures. Le programme d'application, grâce à l'argument NORMTR retourné par la fonction REQUEST LOCATOR, pourra contrôler ou vérifier le choix de clôture effectué par l'opérateur.

b) Clôtures se chevauchant

Lorsque deux clôtures (de normalisation) se chevauchent, on doit définir quelle transformation de normalisation est utilisée pour changer les coordonnées NDC en coordonnées WC. Ce cas se produit fréquemment, puisque implicitement, la transformation de normalisation 0 envoie le carré unitaire de WC sur l'espace NDC tout entier.
Considérons l'exemple de la Figure 1 :

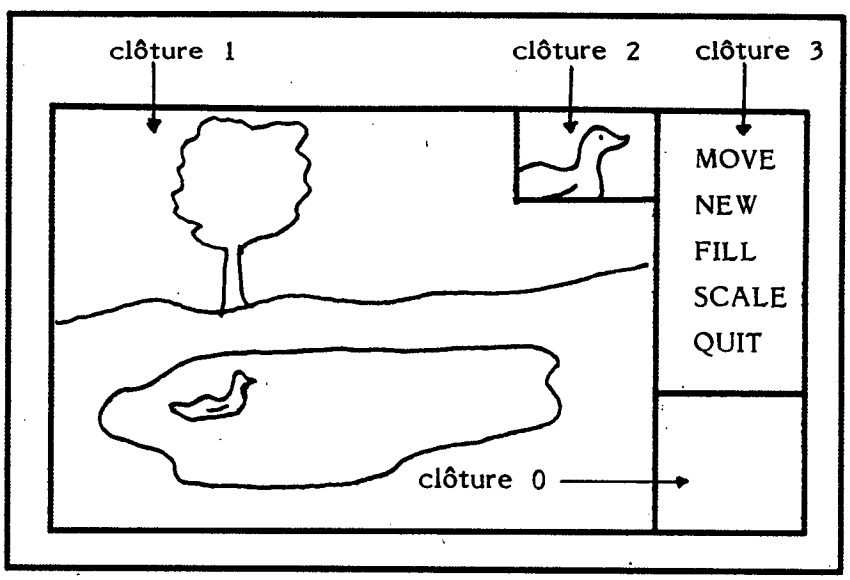
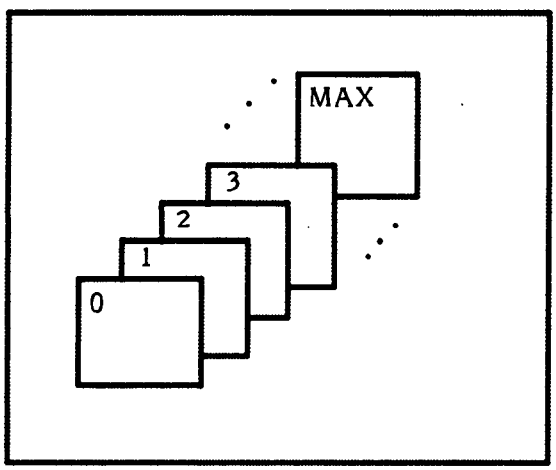


Figure 1

Dans cet exemple, la clôture 1 (i.e. la clôture de la transf. de normalisation n°1) est le rectangle où l'image principale dessinée. On voit que l'image de détail (clôture 2) est superposée sur l'image principale, si bien qu'il y a ambiguité sur le choix de la transformation de normalisation qu'il faut utiliser pour changer les coordonnées NDC en coordonnées WC.

Pour résoudre cette difficulté, on associe à chaque transformation une priorité sur les clôtures pour les entrées : VIEWPORT INPUT PRIORITY. C'est la clôture qui a la plus haute priorité qui est choisie pour réaliser le retour aux coordonnées dans l'espace WC.

Les priorités de clôtures pour les entrées sont initialisées dans l'ordre décrit par la Figure 2 (avec MAX = numéro maximal autorisé pour désigner une transf. de normalisation) :



- A l'initialisation, c'est la transformation de normalisation n°0, qui a la plus haute priorité d'entrée.

Figure 2

Pour changer les priorités, on utilise la fonction :

SET VIEWPORT INPUT PRIORITY(TR1,TR2,HILO)

Si le paramètre HILO prend la valeur HIGHER, alors cette instruction établit que la transformation TR1 a une priorité plus haute que la transformation TR2. Si le paramètre HILO prend la valeur LOWER, alors c'est TR2 qui a une priorité plus haute que TR1.

Dans l'exemple de la Figure 1, pour obtenir l'ordre correct des priorités en entrée (c'est à dire dans l'ordre décroissant 2,1,3,0), nous pouvons utiliser la suite d'instructions :

```
SET VIEWPORT INPUT PRIORITY (1,0,HIGHER)
SET VIEWPORT INPUT PRIORITY (2,1,HIGHER)
SET VIEWPORT INPUT PRIORITY (0,3,LOWER)
```

De la sorte, si la position du locator est sur l'image principale, sur le canard, par exemple, alors ce seront des coordonnées WC associées à la transformation de normalisation n°1 qui seront envoyées au programme d'application.

4 - PICK

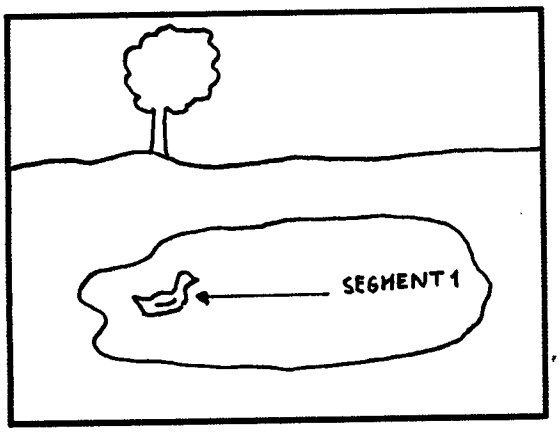
L'entrée logique Pick permet de retourner le nom d'un segment au programme d'application en identifiant l'objet indiqué par le système physique pick. En mode request, la fonction GKS qui effectue cette tâche, est :

```
REQUEST PICK(WS,DV,ST,SEG,PICKID)
```

La fonction REQUEST PICK renvoie le nom d'un segment : SEG, et une identification plus précise à l'intérieur du segment : PICKID.

Supposons que nous soyons dans l'exemple défini par la Figure 3, et qu'on ait défini le segment n°1 comme le segment contenant le canard. L'opérateur désire déplacer le canard sur la mare. Pour cela, il suffit d'écrire les instructions :

```
100   SEG = 1
      REQUEST PICK(WS,DV1,ST1,SEG,PICKID)
      REQUEST LOCATOR(WS,DV2,ST2,NT,X,Y)
      EVALUATE TRANSFORMATION MATRIX(0,0,X,Y,0,1,1,NDC,MAT)
      SET SEGMENT TRANSFORMATION(SEG,MAT)
      GOTO 100
```



L'entrée PICK définit l'objet qui va être déplacé. La fonction REQUEST LOCATOR renvoie une position (en coordonnées NDC puisque aucune priorité de clôture en entrée n'a été réinitialisée). L'instruction EVALAUTE TRANSFORMATION MATRIX définit le mouvement que doit effectuer le segment grâce à l'ordre SET SEGMENT TRANSFORMATION. On remarque que c'est le point (0,0) du segment dans NDC qui est déplacé de la position initiale à la nouvelle position.

Figure 3

a) Identificateur associé à pick

Le paramètre PICKID retourné par la fonction REQUEST PICK précise l'action de l'opérateur à l'intérieur du segment.

En effet, la place disponible poour stocker les segments étant limitée, on utilise pick identifier pour avoir un second niveau d'identification, à l'intérieur de chaque segment. Si par exemple un segment est composé de 3 parties bien distinctes, chacune de ces 3 parties pourra être identifiée par PICK IDENTIFIER (ce qui évitera de créer 3 segments, opération plus coûteuse en place). L'ordre PICK IDENTIFIER est donné sous la forme :

SET PICK IDENTIFIER(N)

Les primitives graphiques qui suivront cette instruction, auront un pick identifier (PICKID) égal à N, et ce jusqu'à ce que l'on rencontre une autre instruction SET PICK IDENTIFIER.

b) Détection d'un segment

On peut empêcher ou permettre qu'un segment soit détecté par un système d'entrée Pick, grâce à la fonction :

SET DETECTABILITY(ID,DET)

où ID est le nom du segment, et DET peut prendre les valeurs DETECTABLE et UNDETECTABLE. Un segment ne peut être reconnu par l'ordre PICK, que si DET a pour valeur DETECTABLE. De plus, la valeur par défaut de DET est UNDETECTABLE. Cependant, rendre un segment "UNDETECTABLE", n'a aucun effet sur son apparence.

5 - CHOICE

Le système d'entrée CHOICE renvoie un entier au programme d'application, entier qui précise un choix parmi plusieurs possibilités. En mode REQUEST, la fonction GKS associée est :

REQUEST CHOICE(WS,DV,ST,CH)

où CH est un entier représentant le choix. Les systèmes physiques d'entrée CHOICE peuvent être très variés : menu utilisé avec un crayon optique, ensemble de touches à manipuler, nom que l'on doit entrer au clavier. La méthode exacte pour préciser le choix associé à chaque entier CH, sera précisée plus loin.

6 - VALUATOR

Un système d'entrée Valuator envoie une valeur réelle au programme d'application. En mode REQUEST, la fonction GKS qui déclenche une entrée valuator, est :

REQUEST VALUATOR(WS,DV,ST,VAL)

où VAL est la valeur réelle du réel qui a été entré.

Evidemment un nombre peut être enregistré, chiffre par chiffre, depuis un clavier sans utiliser de système graphique. Cependant, si on utilise un système graphique interactif, l'opérateur peut examiner l'effet d'une valeur entrée sans délai. Si l'opérateur ne sait pas à l'avance quelle est la meilleure valeur, plusieurs valeurs doivent être

entrées, l'opérateur examinant à chaque fois le résultat, jusqu'à ce que la meilleure valeur soit obtenue.

D'autre part, l'instruction REQUEST VALUATOR lit des valeurs depuis n'importe quel système physique d'entrée. Des systèmes comme des boutons que l'on tourne, peuvent générer de façon continue des valeurs et l'écho graphique peut ainsi être constamment à jour.

Une application fréquente du système d'entrée valuator, est la modification de la taille ou échelle, d'un segment ou d'une image.

Exemple de programme :

```
100   CONTINUE
      REQUEST VALUATOR(WS,DV,ST,SIZE)
      EVALUATE TRANSFORMATION MATRIX(FX,FY,0,0,0,SIZE,SIZE,WC,MAT)
      SET SEGMENT TRANSFORMATION(SEG,MAT)
      GOTO 100
```

7 - STRING

Un système logique d'entrée String envoie une chaîne de caractères au programme d'application. Dans la plupart des cas, la chaîne est enregistrée avec un écho après chaque caractère. En mode REQUEST, la fonction GKS qui déclenche une entrée String, est :

```
REQUEST STRING(WS,DV,ST,NCHARS,STR)
```

où STR est la chaîne de caractères qui a été entrée et NCHARS est le nombre de caractères qu'elle contient.

L'entrée String est le plus souvent réservée pour enregistrer un titre ou un nom de fichier.

8 - STROKE

Le système logique d'entrée Stroke, transmets une suite de points au programme d'application, un peu comme le ferait un "locator multiple".

En mode REQUEST, la fonction GKS qui entraîne une entrée STROKE, est :

REQUEST STROKE(WS,DV,PTSMAX,ST,NTnNPTS,X,Y)

Le paramètre PTSMAX précise le nombre maximal de points que l'on peut entrer à l'aide du système STROKE. Une entrée Stroke transmets une suite de points dans les tableaux X et Y, qui sont les coordonnées en X et Y de ces points. Le nombre de points est NPTS (argument de sortie). Stroke transmet aussi la transformation de normalisation NT (argument de sortie) utilisée pour revenir dans l'espace WC. Tous les points sont transformés en utilisant la même transf. de normalisation. On utilise les priorités d'entrée entre clôtures pour choisir parmi plusieurs clôtures empiétantes l'une sur l'autre, et ce de la même façon qu'avec un système locator.

Certaines installations de l'entrée STROKE peuvent procurer un moyen d'éliminer des points indésirables. Par exemple, un système STROKE peut dans certains cas ne retenir que les points se présentant à des intervalles de temps choisis. Dans ce cas, cette idée permettra d'enregistrer des données décrivant un mouvement entré par l'opérateur, pas seulement tous les points enregistrés. Cela est utile en reconnaissance de signature (mais, il est impossible de réaliser cela avec une entrée locator).

MODES D'INTERACTION

1 - INTRODUCTION

Nous décrivons ici les différents modes d'entrée disponibles sous GKS, et montrons comment ils permettent différents styles d'interaction.

L'interaction que nous présentons se situe entre deux processus ; le premier est le programme d'application, et le second est le processus d'entrée qui comprend le système d'entrée et la livraison des données entrées au programme d'application. Le lien entre ces deux processus peut varier, et par conséquent produire différents styles d'interaction.
Les trois modes d'interaction sont :

(1) mode REQUEST :

Le programme d'application et le processus d'entrée travaillent alternativement. D'abord le programme d'application demande une donnée en entrée, et attend une réponse. Le processus d'entrée est déclenché par cette demande, délivre la donnée au programme d'application, puis retourne à l'état d'attente. L'un ou l'autre des processus est actif, mais jamais les deux à la fois.

(2) mode SAMPLE :

Le programme d'application et le processus d'entrée sont actifs ensemble. Le processus d'entrée travaille en fournissant la dernière donnée entrée qui peut être ou ne pas être utilisée par le programme d'application. De l'autre côté, le programme d'application continue à s'exécuter, prenant et utilisant la dernière donnée entrée uniquement quand il en a besoin.

(3) mode EVENT :

Le programme d'application et le processus d'entrée sont actifs ensemble, mais contrairement au mode SAMPLE, c'est le processus d'entrée qui a le rôle le plus dominant. Il fournit au programme d'application les données entrées et attend que le programme agisse selon les données reçues. L'opérateur contrôle l'interaction.

2 - MISE EN PLACE D'UN MODE

Chaque système d'entrée ne peut être dans plus d'un mode à la fois. On choisit le mode associé à un système d'entrée (ici LOCATOR), grâce à la fonction :

SET LOCATOR MODE(WS,DV,MODE,EC)

Les paramètres WS, DV, déjà vus auparavant, ont la même signification. Le paramètre MODE définit le mode d'interaction, et peut prendre les valeurs : REQUEST, SAMPLE et EVENT. Le dernier paramètre EC peut prendre deux valeurs ECHO et NOECHO. Si EC a pour valeur ECHO, l'opérateur obtiendra une indication (un signe), lorsque la donnée entrée aura été acquise par le programme.

Par défaut, le mode opérant pour tous les systèmes d'entrée, est le mode REQUEST, et par défaut également, le paramètre EC a pour valeur ECHO.

3 - Mode REQUEST

Pour chaque système d'entrée, la fonction d'appel d'un système en mode REQUEST, est comme nous l'avons déjà vu :

REQUEST XXX(WS,DV,STATUS,NORHTR,XPOS,YPOS)

où XXX désigne le système logique d'entrée. Comme précédemment WS et DV précisent le système physique d'entrée. Le paramètre STATUS signale si le système logique d'entrée a réussi à acquérir la donnée demandée.

Si EC a pour valeur ECHO, l'appel de la fonction REQUEST LOCATOR fera apparaître un écho initial sur l'écran. Cet écho, dépendant du système physique utilisé, peut être par exemple, un réticule. Puis l'opérateur actionne le système physique pour placer de façon adéquate l'écho, ici un réticule. Ceci fait, il actionne une commande (bouton, touche, etc...), pour signaler que l'entrée est terminée.

La Figure 1 montre un exemple de ce que l'opérateur voit :

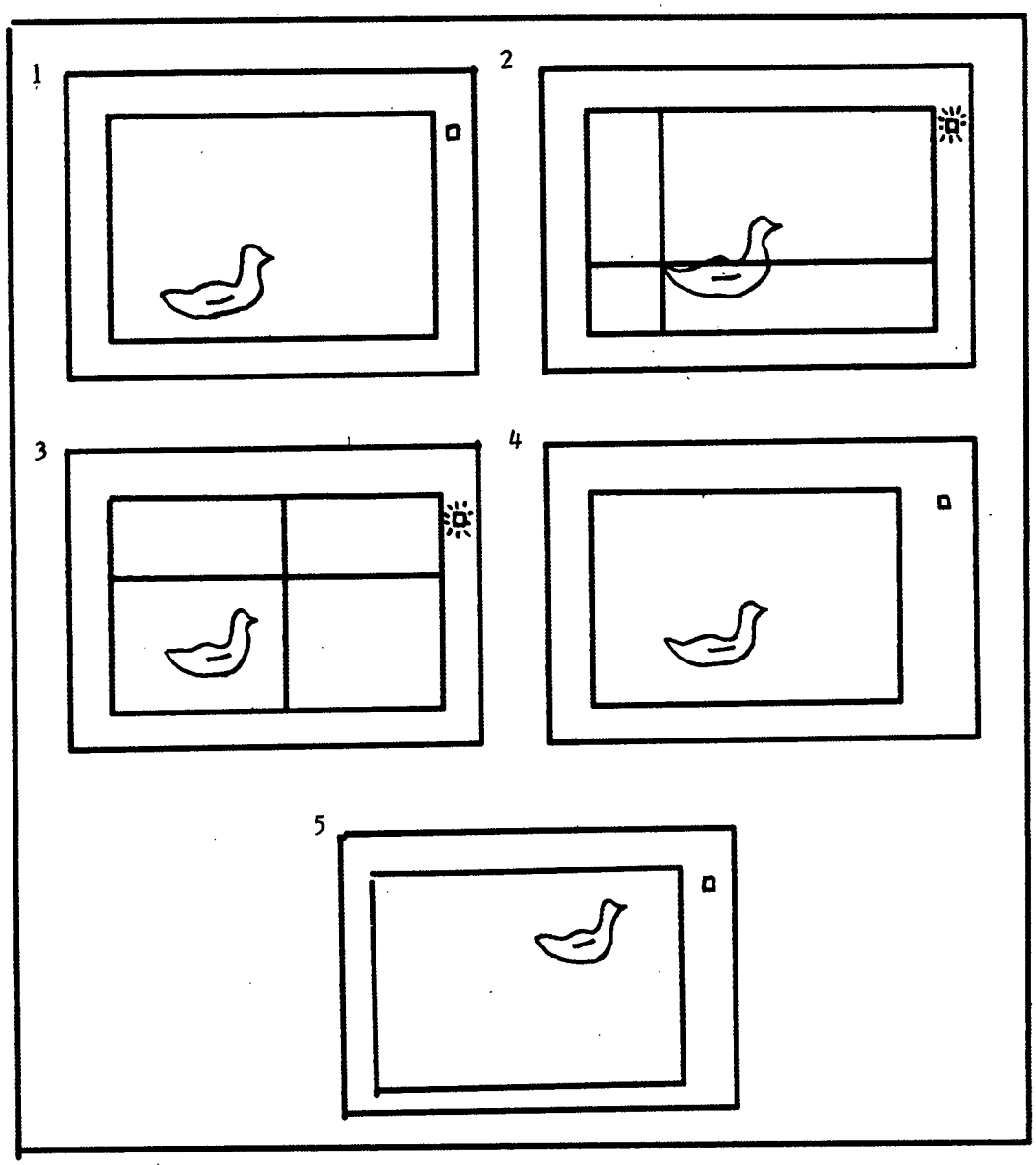


Figure 1

La première image montre ce que voit l'opérateur avant que l'entrée n'ait commencé. La lampe éteinte, à droite de l'écran, signifie que le système ne demande pas qu'une commande soit exécutée (bouton pressé, par exemple).

Si le programme appelle REQUEST LOCATOR, un réticule apparaît comme dans la 2e image, pour montrer à l'opérateur que le système logique d'entrée est disponible. Cette action est appelée "prompt". La position du locator donne les valeurs initiales de XPOS et YPOS. La lampe est allumée pour indiquer à l'opérateur, qu'une fois qu'il aura déplacé le locator à la position désirée, il devra appuyer sur la touche correspondante pour entrer cette position.

La troisième image montre que l'opérateur a actionné le système physique d'entrée, et le réticule s'est déplacé, faisant écho à l'action de l'opérateur.

Si l'opérateur appuie sur la touche adéquate, la position du Locator sera entrée. Le réticule disparaîtra, et la lampe s'éteindra comme dans la 4e image. Cette réaction est un "acknowledgement" (reconnaissance) de l'action d'appuyer sur la touche.

Dans la dernière image, le programme donne le résultat (feedback) en déplaçant le canard à la position précisée par le locator. Il est possible que la 4e image soit si éphémère qu'elle ne puisse être aperçue par l'opérateur.

4 - STATUS

Le paramètre Status est renvoyé par la fonction REQUEST pour indiquer un succès ou un échec. STATUS peut prendre les valeurs OK ou NONE. La valeur OK signifie que l'opérateur a terminé l'entrée des données de façon normale.

La valeur NONE signifie que les données entrées ne sont pas correctes parce que l'opérateur a envoyé un signal d'arrêt (break). Un

signal d'arrêt ou break, peut être interprêté par le programme comme la fin de séries de valeurs entrées.

La façon d'indiquer un break dépend du système d'entrée. Si l'opérateur choisit une position incorrecte du locator, cela pourra être interprêté comme un break. L'opérateur peut aussi indiquer un break en actionnant une commande alternative. Le système peut avoir deux boutons : l'un pour signaler une fin normale, l'autre pour un break.

Exemple de programme utilisant la variable STATUS :

```
100   CONTINUE
      REQUEST LOCATOR(WS,DV,STATUS,NT,X,Y)
      IF(STATUS.EQ.NONE) GOTO 200
      NEW FRAME
      DRAW BACKGROUND
      DRAW DUCK AT(X,Y)
      GOTO 100
200   CONTINUE
```

La fonction REQUEST PICK peut renvoyer une valeur pour STATUS, autre que OK et NONE. Il s'agit de la valeur NOPICK, utilisé lorsque l'opérateur ne pointant aucun segment, veut déclencher une entrée Pick.

5 - MODE SAMPLE

En mode Sample, une valeur à entrer, est envoyée directement au programme, sans attendre que l'opérateur actionne une commande après avoir contrôlé la donnée à entrer (cas du mode REQUEST). Nous illustrons, ici, l'explication de ce mode avec l'entrée Locator, mais le mode SAMPLE s'applique à tous les systèmes d'entrée. Tout d'abord, on doit mettre le système d'entrée en mode SAMPLE, en appelant la fonction :

```
SET LOCATOR MODE(WS,DV,SAMPLE,EC)
```

Contrairement au mode Request, l'écho commence dès la mise en place du mode. Pour utiliser le système d'entrée, on doit alors appeler la fonction :

```
SAMPLE LOCATOR(WS,DV,NT,X,Y)
```

On remarquera que les fonctions SAMPLE n'ont pas de paramètre STATUS. La fonction SAMPLE renvoie la valeur courante du Locator produite par le processus d'entrée. L'exemple précédent s'écrit alors :

```
      SET LOCATOR MODE(WS,DV,SAMPLE,ECHO)
100   CONTINUE
      SAMPLE LOCATOR(WS,DV,NT,X,Y)
      NEW FRAME
      DRAW BACKGROUND
      DRAW DUCK AT(X,Y)
      GOTO 100
```

Avec le mode SAMPLE, il n'y a pas de paramètre STATUS pour signaler qu'une valeur entrée est incorrecte. Donc il est impossible d'indiquer facilement un break (signal d'arrêt). Cependant, on peut utiliser le numéro de la transformation de normalisation pour réaliser un break. Par exemple, par un test sur NT, dès que l'opérateur déplace le Locator, en dehors de la clôture, l'interaction se termine.

Cependant, nous verrons une meilleure méthode pour terminer une interaction avec le mode EVENT.

En mode Sample, on peut utiliser plus d'un système d'entrée en même temps ; ce qui n'était pas possible avec le mode REQUEST, car le programme devait s'arrêter tant qu'une entrée n'était pas terminée.

Exemple où l'on contrôle la taille du canard et son orientation :

```
      SET VALUATOR MODE(WS,DV1,SAMPLE,NOECHO)
      SET VALUATOR MODE(WS,DV2,SAMPLE,NOECHO)
100   CONTINUE
      SAMPLE VALUATOR (WS,DV1,SIZE)
      SAMPLE VALUATOR(WS,DV2,ROT)
      EVALUATE TRANSFORMATION MATRIX(FX,FY,0,0,ROT,SIZE,SIZE,WC,MAT)
      SET SEGMENT TRANSFORMATION(CANARD,MAT)
      GOTO 100
```

6 - MODE EVENT

En mode Event, les valeurs entrées sont placées dans une suite d'entrée ou liste (input queue) que le programme doit lire. Le programme d'application lit chaque valeur dans l'ordre, et la traite, avant de traiter la suivante.

Comme le programme d'application et le processus d'entrée sont tous deux actifs en même temps, il est possible que le programme regarde la liste, alors qu'il n'y a pas eu de nouvelle entrée. D'un autre côté, il est possible que le processus d'entrée remplisse complètement la liste avant que le programme ne l'ait enregistrée. (La condition de "saturation" de la liste sera examinée par la suite).

Par conséquent, GKS a besoin de fonctions qui vérifient l'état de la liste. Il n'y a qu'une seule liste d'entrée pour tous les systèmes. En mode EVENT, il est possible d'avoir plusieurs systèmes actifs en même temps. Depuis chaque système d'entrée, les données sont aussitôt ajoutées à la liste, avec une information indiquant quel système les a générées.
La fonction GKS vérifiant si la liste d'entrée est vide, est :

```
      AWAIT EVENT(TIMEOUT,WS,DVCLASS,DV)
```

Si la liste est vide, le programme s'arrête, attendant jusqu'à ce qu'une donnée soit générée, ou que s'écoulent TIMEOUT secondes. Dans le dernier cas, DVCLASS prend la valeur NONE. Si on a fixé TIMEOUT égal à 0, le programme d'application continue. Si la liste n'est pas vide (soit quand AWAIT EVENT est appelée, soit avant que TIMEOUT secondes ne s'écoulent), alors la première donnée dans la liste est transmise. De plus, certaines informations sont aussi transmises : WS et DV qui précisent le système physique, DVCLASS qui précise le système logique d'entrée.

Les données entrées sont alors transférées dans l'ensemble des données disponibles (current event report). On appelle alors la fonction GET qui permet de lire la donnée entrée ; par exemple si DVCLASS a pour valeur LOCATOR :

```
GET LOCATOR(NT,X,Y)
```

où NT est le numéro de la transformation de normalisation et X, Y la position enregistrée.

L'intérêt du mode EVENT apparaît dans le cas où l'opérateur a plusieurs systèmes à contrôler, ou, dans le cas où le programme a des tâches à réaliser entre l'arrivée de deux données. Supposons, par exemple, que nous voulions représenter à l'écran plusieurs points par des symbôles différents. Le choix des symbôles peut s'effectuer en utilisant un système Choice. Le symbole choisi sera utilisé pour tous les points qui seront entrés, jusqu'à ce qu'un autre symbôle soit choisi. Nous pouvons employer la suite d'instructions :

```
      SET POLYMARKER INDEX(1)
      SET LOCATOR MODE(WS,DV1,EVENT,ECHO)
      SET CHOICE MODE(WS,DV2,EVENT,ECHO)
100   CONTINUE
      AWAIT EVENT(600,WST,CLASS,DEV)
      IF(CLASS.EQ.NONE) STOP
      IF(CLASS.EQ.LOCATOR) GOTO 200
      GET CHOICE(CH)
      SET POLYMARKER(CH)
      GOTO 100
200   GET LOCATOR(NT,X(1),Y(1))
      POLYMARKER(1,X,Y)
      GOTO 100
```

Cet exemple donne une idée de la souplesse du mode EVENT. Le programme se termine, ici, lorsque l'opérateur cesse d'entrer des données pendant TIMEOUT secondes (ici 600 secondes !).

Il existe une 3e fonction GKS qui contrôle ce qui se passe dans la liste d'entrée. Il est possible que l'opérateur entre plus d'une valeur dans un système d'entrée, alors que le programme d'application n'a besoin que d'une seule valeur. La fonction suivante, enlève certaines données indésirables de la liste d'entrée :

FLUSH DEVICE EVENTS(WS,DVCLASS,DV)

Après son appel, les données parvenant du système logique DVCLASS, et du système physique précisé par WS et DV, sont enlevées de la suite d'entrée.

7 - MODES MIXTES

Nous avons déjà vu qu'il était plus facile de déplacer un segment en mode SAMPLE plutôt qu'en mode REQUEST. Néanmoins il est difficile de terminer une interaction en mode SAMPLE (paragraphe 5). On peut donner une solution à ce problème en utilisant un second système d'entrée, en mode EVENT, pour indiquer la fin de l'interaction. Il pourra s'agir d'un bouton que l'on doit presser, ou d'une touche que l'on doit enfoncer. Dans tous les cas, ce système d'entrée sera un système Choice.

Dans l'exemple suivant, plus complexe, nous utilisons un mélange de systèmes d'entrée dans différents modes. Supposons que l'opérateur définisse une aire en utilisant un Locator et qu'il désire avoir un écho graphique qu'il définit après chaque entrée d'un nouveau sommet. Quand l'opérateur a sélectionné un nouveau sommet, avec le Locator, il appuie sur une touche correspondant à la première valeur de Choice. Quand il juge que l'aire entière est terminée, il appuie sur la touche correspondant à la 2e valeur de Choice. Ainsi, le programme s'écrit :

```
      SET LOCATOR MODE(WS,DV1,REQUEST,ECHO)
      REQUEST LOCATOR(NT,XAR(1),YAR(1))
      NP=1

      SET LOCATOR MODE(WS,DV1,SAMPLE,ECHO)
      SET CHOICE MODE(WS,DV1,SAMPLE,NOECHO)
      CREATE SEGMENT (AREA)
      CLOSE SEGMENT

100   CONTINUE
      NP = NP+1
      XAR(NP+1) = XAR(1)
      YAR(NP+1) = YAR(1)

200   CONTINUE
      SAMPLE LOCATOR(WS,DV1,NT,XAR(NP),YAR(NP))
      DELETE SEGMENT(AREA)
      CREATE SEGMENT(AREA)
      FILL AREA(NP+1,XAR,YAR)
      CLOSE SEGMENT
      AWAIT EVENT (0,WST,CLASS,DEVICE)
      IF(CLASS.NE.CHOICE) GOTO 200

      GET CHOICE(CH)
      IF(CH.EQ.1) GOTO 100
      FLUSH DEVICE EVENTS(WS,CHOICE,DV2)
```

La figure 2 illustre la situation quand l'opérateur choisit le 5e point de l'aire. Lorsque nous affichons l'aire en cours, nous mettons cette aire dans un segment. Ceci nous permet de détruire l'ancienne aire avant chaque passage dans l'instruction SAMPLE LOCATOR.

Nous avons utilisé FLUSH DEVICE EVENTS, pour enlever des valeurs Choice qui pourraient rester dans la file d'entrée (dans le cas, par exemple, où l'opérateur appuierait deux fois sur le bouton, par accident)

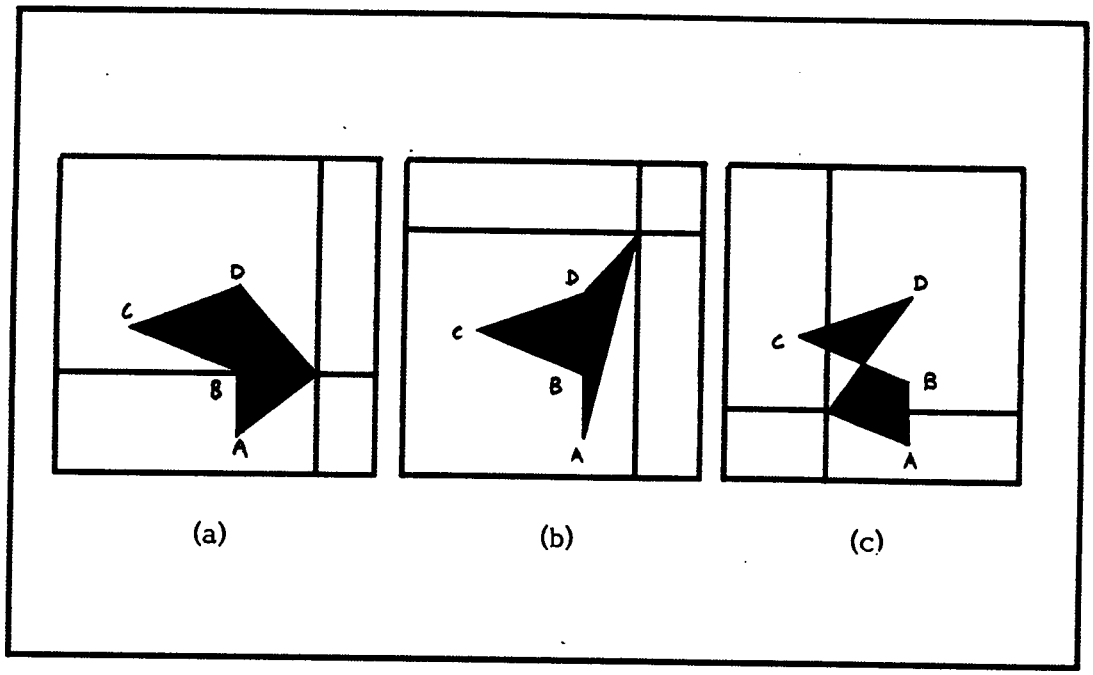


Figure 2

Les images (a), (b), (c) sont différentes figures que l'opérateur peut obtenir en fixant 4 points, à partir du programme lorsque la valeur de CLASS est non égale à CHOICE (renvoi en 200).

REMERCIEMENTS

Ce travail a été effectué à l'Institut National de Recherche en Informatique et en Automatique, à Rocquencourt.

Je remercie particulièrement Monsieur A. DUCROT (membre du groupe de travail ISO "Computer Graphics") qui a signalé plusieurs imprécisions dans la première version de ce texte, et qui m'a expliqué l'organisation d'une interface GKS-écran graphique. Je remercie vivement Monsieur Y. LECHEVALLIER qui m'a encouragé à rédiger cette documentation, et m'a donné l'accès à plusieurs types de terminaux graphiques.

Je remercie également Madame M. CORNELIS qui a effectué toute la mise en page de ce document.

REFERENCES

[1] Computers Graphics Programming (GKS - the Graphics Standard); G. Enderle, K. Kansy, G. Pfaff; Springer-Verlag, 1984;

[2] Introduction to the Graphical Kernel System (GKS); F.R.A Hopgood, D.A. Duce, J.R. Gallop, D.C. Sutcliffe; Academic Press, 1983

[3] FORTRAN Interface of GKS 7.2, ISO TC97/SC5/WG2. Erlangen; november 1983.

[4] Systèmes de traitement de l'information, langages de programmation, GKS graphical kernel system, description formelle, NF Z 65-900, afnor, novembre 1984.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique