LE LANGAGE GRAPHIQUE GASTON

Guy FERRAN

Université de Montréal

RÉSUMÉ

Ce papier décrit très sommairement un langage destiné à la programmation d'un terminal graphique évolué. Un répertoire d'instructions très spécialisées a été défini, qui dégage l'utilisateur de la technologie du terminal. Le langage possède de nombreux types de variables et des outils pour les structurer. Une méthode originale de déclaration et d'appel de procédure est présentée, qui fournit un moyen d'étendre le langage et de compiler toutes les instructions graphiques considérées comme des cas particuliers de telles procédures. Le langage repose sur un système graphique permettant la structuration des images en figures indépendantes chacune contenant des objets graphiques repérables et modifiables à la volonté du programmeur.

THE GRAPHIC LANGUAGE GASTON

ABSTRACT

This paper briefly describes a digigraphic oriented language. A set of highly specialized instructions has been defined freeing the programmer from the low level particularities of the graphic terminal. Types of variables are numerous and data structuring tools are provided. A means of extending the language and compiling all graphical instructions as particular cases of procedures is made possible by use of special declaration and procedure calls. The language is based on the existence of a graphic system capable of structuring pictures into independent subpictures each in turn made up of groups of picture atoms (points, lines, characters). Thus, making every picture entity selectively accessible and modifiable by the programmer.

LE LANGAGE GRAPHIQUE GASTON

Guy FERRAN

Université de Montréal

Introduction.

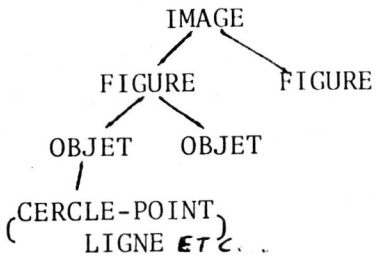
La programmation d'un écran graphique se fait toujours par l'intermédiaire d'un support software, appelé système graphique, qui consiste en un ensemble de modules, ou sous-programmes, fournis en général par la maison qui a vendu ou loué le terminal. Bien souvent, l'utilisateur doit s'astreindre à écrire ses programmes dans un langage comme FORTRAN, quand ce n'est pas en langage machine : le jeu consistant pour lui à faire des appels aux bons sous-programmes en leur fournissant la liste, parfois monstrueuse, de paramètres dont ils ont besoin et malheur à lui si le 17ième paramètre n'est pas de type réel et ne représente pas l'échelle des abscisses de la figure qu'il est en train de construire ! FORTRAN est bien compris de la machine, soit, mais est-ce un langage naturel pour un "programmeur graphique" dont la spécialité, au demeurant, n'est pas l'informatique mais le dessin, l'architecture ou une discipline d'ingénieur ?
A l'Université de Montréal, le groupe de recherche en systèmes et langages de programmation a non seulement défini un système graphique, nommé GUIGNOL (1), mais aussi un langage qui, nous l'espérons, améliorera la communication entre "l'homme et la machine", puisque c'est de cela qu'il s'agit, bien que notre modestie nous inclinerait seulement à dire que le langage GASTON (2) est, au moins, un langage un peu moins barbare.

Les Outils fondamentaux à donner au programmeur.

Les outils minima qu'on devrait fournir au programmeur nous semblent les suivants :

1) Un écran étant un plan dont les points peuvent être repérés par des coordonnées, le langage doit posséder des variables de type "point", c'est-à-dire, en fait, pouvoir manipuler les nombres complexes (couple d'entiers ou de réels).

2) Le but du programme étant de dessiner des figures, il faut des variables de type graphique qui ne servent qu'à désigner des parties d'images. Il est bon de pouvoir créer des figures indépendantes, mais aussi de réunir dans une même figure un ensemble d'objets ayant des caractéristiques communes. Notre système prévoit donc une structuration de l'image à 2 niveaux :



Un objet contient une suite d'éléments graphiques qui sont les "primitives" du système : point, ligne, segment, cercle.
Par exemple, voici un programme qui construit une image formée d'une seule

figure. Cette figure contient 2 objets : le premier est constitué d'un cercle, le deuxième de deux lignes, formant un angle. Les variables ARRIVEE et CENTRE sont des complexes (couple d'entiers). L'instruction ARC génère un cercle de rayon R. L'instruction POSITION place le faisceau au point de coordonnées (20,20). L'instruction LIGNE est répétitive et s'exécute 2 fois : la première ligne part du point courant et aboutit à ARRIVEE, la deuxième part de ce point et aboutit à ARRIVEE + (5,5).

```
ZENTIER ARRIVEE = 10&10, CENTRE = 0&0  ENTIER  R = 10
GRAPHIQUE  GEOMETRIE, ANGLE, CERCLE ;
IMAGE ;
   DEBUT  GEOMETRIE ;
          OBJET  CERCLE ;
               ARC → CENTRE  RAYON R ;
          OBJET  ANGLE ;
               POSITION → CENTRE + 20&20 ;
               LIGNE REPETER 2 → ARRIVEE INC  5&5 ;
   AFFICHER GEOMETRIE ;
```

3) Options par défaut : Le programme précédent dit le strict minimum pour générer la figure GEOMETRIE. En effet, on aurait pu allouer simplement une portion de l'écran à la figure et définir une option de coupage : par défaut, la zone allouée est l'écran tout entier et il n'y a pas de coupage. On aurait pu définir le système de coordonnées utilisées par la figure et des échelles sur les abscisses ou les ordonnées : par défaut, ce sont des coordonnées-écran et les échelles sont 1. Enfin, on aurait pu définir des options générales pour tous ses objets : par défaut, ceux-ci ne seront pas sensibles au pointeur optique, ils ne seront pas clignotants, ils seront allumés et moyennement lumineux. De même on aurait pu donner des numéros aux objets au lieu de (ou en plus de) leur donner un nom, et dire à quelle figure ils appartiennent. Chacun d'eux aurait pu avoir ses options propres : sensibilité au pointeur, luminosité etc... Ainsi l'instruction OBJET pourrait s'écrire aussi :

```
OBJET CERCLE @ GEOMETRIE NON SENSIBLE INCLUS NON CLIGNOTANT ;
```

4) Utiliser toutes les possibilités du terminal.
Notre terminal permet l'appel de sous-programme, dans sa propre mémoire. Notre langage possède donc des figures particulières, nommées ALPHABETS, qu'on génère de la même manière que les figures (ils contiennent des objets). Ces alphabets sont, en somme, des ensembles d'objets graphiques auxquels les figures ordinaires peuvent se référer : l'objet en question n'étant alors défini qu'une fois. Dans l'exemple ci-dessous, on définit un alphabet contenant un objet numéroté 1. La figure écriture se réfère ensuite à lui (instruction CARACTERE).

```
DEBUT  lettres  ALPHABET ;
     OBJET # 1 ;
          LIGNE REPETER 2 → Z  INC DZ ;
          SEGMENT  Z1  ;
AFFICHER lettres ; % transmet l'alphabet au terminal %
DEBUT écriture ;
     OBJET  texte  SENSIBLE ;
          CARACTERE → Z2 # 1 @ lettres ;
          TEXTE → Z2 + DZ2   DANS  "DISCOURS" ;
```

Le caractère #1 apparaîtra au point Z2. L'instruction TEXTE écrit une suite de

caractères de l'alphabet standard (tout alphabet créé par l'usager peut d'ailleurs être déclaré standard) ; ce texte se trouve dans une chaîne de caractères (le langage possède des constantes et des variables de type chaîne de caractères).

5) Un langage algorithmique simple, dont la compilation est conversationnelle. Le langage possède les instructions classiques : itération, condition, affectation, etc... De plus des structures d'informations peuvent être crées dynamiquement, grâce au système d'allocation emprunté à KNOWLTON (3). On peut ainsi déclarer des champs de mémoire contenant des variables quelconques et auxquelles on peut se référer par une variable de type référence. Voici un exemple :

```
CHAMP  0  ENTIER I, 1  GRAPHIQUE figure, 2 REFERENCE suivant ;
BLØC  informe  REFERENCE P ;
    ALLOUER informe LONGUEUR 4 ; % demande un bloc %
    METTRE  informe DANS  P ; % POUVOIR s'y référer %
    SI  P.I = 1 ;  DETRUIRE P.figure ;
    SINØN ; P := P. suivant ;
    FINSI ;
```

6) Des instructions de transmission et communication.
Après avoir défini la figure GEOMETRIE (voir 2) on a donné l'ordre de l'AFFICHER. Elle apparaît alors sur l'écran. On pourra aussi la DETRUIRE (définitivement) ou l'ETEINDRE (provisoirement). Pour utiliser les outils de communication (pointeur optique, stylet, croix, manche à balai etc...), on doit les autoriser. Supposons, par exemple, qu'on veuille désigner le cercle de la figure géométrie par le pointeur, comme cet objet n'a pas été déclaré sensible il faut le sensibiliser. Puis on attendra que l'utilisateur le désigne (instruction ETAT), pour récolter les informations à son sujet.

```
OBJET  CERCLE  @  GEOMETRIE DEVIENT SENSIBLE ;
AUTORISER  POINTEUR ;
ETAT POINTEUR ATTENTE  @  NOMFIGURE, NOMOBJET ;
SI  NOMOBJET = CERCLE ;  % ETC... %
```

7) Un moyen simple de définir des nouvelles instructions du langage.
Un langage graphique, comme tout langage spécialisé, ne doit pas être figé dans une forme définitive. Toute adjonction au système d'un nouveau module doit pouvoir être reflétée dans le langage. GASTON est, en ce sens, un langage extensible. Supposons qu'on veuille définir une instruction construisant n segments dont les coordonnées entières des extrémités sont dans deux tableaux : on déclarerait alors une procédure ainsi :

```
PROCEDURE  segment2  ENTIER REPETER = 1
                     2TABLEAU  ZENTIER  XY ;
        ENTIER J ;
            FAIRE  J  DE  1  A  REPETER ;
                   SEGMENT → XY'1 [J], XY'2 [J]
            FINFAIRE ;
FINPROCEDURE
```

Des appels à la nouvelle instruction peuvent s'écrire :

```
SEGMENT2  REPETER   N   XY  TX, TY ;
SEGMENT2  XY  TX1, TX2 ; % par défaut répéter = 1%
```

Toutes les instructions graphiques sont définies par des procédures. L'instruction ASSEM permet d'appeler les sous-programmes du système dans le corps de procédure.

Conclusion.

Un effort de simplification et de clarification a été fait dans la communication du programmeur et de son terminal graphique. Les caractéristiques essentielles du langage sont : extensibilité, instructions avec mots-clés significatifs, nombreuses options par défaut, liberté dans le choix des mots-clé et leur emplacement dans les instructions, types de variables nombreux et en liaison avec la programmation graphique.

(1) GUIGNOL, un système graphique par Normand Lebel

(2) GASTON, un langage graphique par Guy Ferran

Thèses de maîtrise de l'Université de Montréal 1972.

(3) A fast storage allocator par K,C. Knowlton
comm. A.C.M., 8, 10 (october-965), 623-625.