PREMIER COLLOQUE IMAGE
Traitement, Synthèse, Technologie et Applications
BIARRITZ — Mai 1984 —

LANGAGE DE HAUT NIVEAU POUR IMAGES SONAR
HIGH LEVEL LANGUAGE FOR SONAR IMAGES

Mr LACOSTE Mme GUIMBAL Mr BRIZARD

THOMSON-CSF - Division des Activités Sous-Marines Chemin des travails
B.P. 53 - 06801 Cagnes-Sur-Mer Cédex/France

RESUME

Dans le cadre des applications liées au sonar, qui comportent généralement une partie graphique importante, nous avons éprouvé le besoin d'apporter aux programmeurs une Aide à la Production des Images (API), qui puisse aussi représenter un standard interne pour la conception et les évolutions de ces dernières.

Le système API consiste en un ensemble de logiciels visant à fournir au programmeur un environnement performant et facile à mettre en oeuvre, permettant d'une part de réduire le temps nécessaire à la conception des images et d'autre part de définir une partie commune à toutes les applications, donc de diminuer la part de développement propre à chacune d'elles.

Les logiciels d'aide s'appuient sur un standard graphique qui doit leur assurer une bonne portabilité. Bien qu'orienté plus spécialement sur les besoins en sonar, API pourrait facilement être utilisable dans d'autres domaines.

SUMMARY

Sonar applications generally contain an important part in graphics, as such it is felt there is a need to develope a programmers Aid for Producing Images (API). The aid could also serve as an internal standard for graphics design and modification.

The API will comprise a set of software tools which will provide the following advantages :

. Reduce the design effort

. Define a standard part which will be common for all applications.

The software tools are built on a graphics standard thus providing good portability. Although oriented to sonar needs, API could easily be applied to other fields.

LANGAGE DE HAUT NIVEAU POUR IMAGES SONAR

HIGH LEVEL LANGUAGE FOR SONAR IMAGES

Mr LACOSTE Mme GUIMBAL Mr BRIZARD

1. INTRODUCTION

Le principal effet de l'utilisation des langages graphiques standards dans des systèmes produisant des images a été de résoudre le problème d'indépendance du logiciel d'application vis-à-vis des machines utilisées.

Le progrès amené par ces langages est considérable et permet d'envisager de nombreux échanges entre leurs différents utilisateurs.

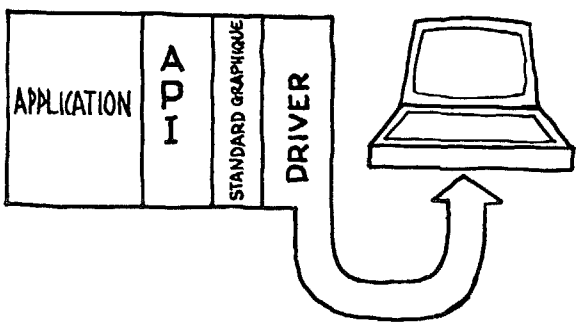
Ils ont le grand mérite de doter les programmes d'application d'un vocabulaire graphique standard.

Néanmoins, la façon de combiner ce vocabulaire pour en faire des phrases cohérentes et structurées, ne semble pas être aujourd'hui la préoccupation majeure des principaux standards.

L'outil que nous développons s'attaque à cet aspect pour aider le programmeur de l'application à composer ses phrases graphiques qui deviendront les images à visualiser.

Le vocabulaire proposé par les standards ne permet aux logiciels d'application qu'une communication par ordres graphiques avec le système chargé de réaliser les images. Cela oblige l'application à prendre en charge des problèmes graphiques qui la détournent de son but initial : programmer un traitement sonar et réaliser une image ne relèvent pas de la même compétence.

Notre outil assure l'interface entre l'application pure et la partie graphique, c'est-à-dire l'exploitation des variables du traitement qui affectent la visualisation afin d'effectuer la mise à jour des images.



2. PRESENTATION D'API

2.1. Principes

Le système d'Aide à la Production d'Images (API) offre au programmeur deux types d'outils :

a) Hors temps réel, un outil de création graphique aboutissant à la génération d'un CODE IMAGE.

b) En temps réel, un outil qui exploite le CODE IMAGE pour traduire les évolutions de l'image demandées par l'application. Cet outil pilote le système de visualisation par l'intermédiaire d'un standard graphique.

2.2. Mécanismes

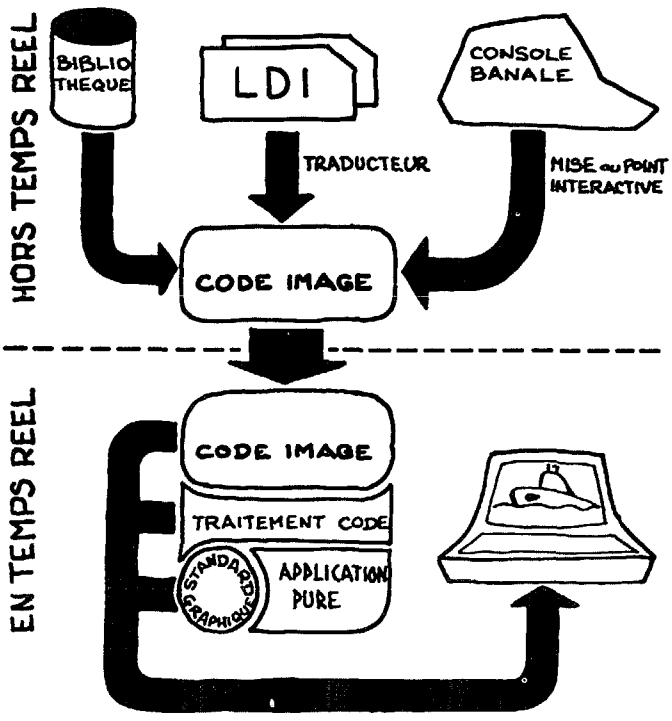
a) L'outil hors temps réel se compose :

. D'un langage évolué assurant la description naturelle et structurée des images (LDI)

. D'un traducteur transformant cette description en code image

. De bibliothèques d'images permettant de réutiliser l'expérience acquise sur des réalisations précédentes

. D'un logiciel de mise au point interactive des images utilisant une console banale par le biais d'un standard graphique (GKS).



b) En temps réel, les modifications des images sont assurées par des modules de traitement du code sur sollicitation de l'application :

. Soit explicite au moyen d'ordres de haut niveau

. Soit implicite sur évolution des valeurs de variables.

3. PRINCIPES DU LANGAGE

Le langage de description des images doit d'abord être un langage naturel, permettant autant que possible de s'imaginer l'image par simple lecture ou d'identifier les objets composant l'image.

Il est facile à mettre en oeuvre, en particulier il doit permettre au programmeur de définir ses propres primitives et de les utiliser ensuite comme des primitives standard. Par exemple, si les images contiennent plusieurs tracés d'axe, l'axe apparaîtra au programmeur comme une primitive standard.

Il est très structuré, en fonction de deux critères :

. Toute partie répétée plusieurs fois dans l'image constitue un objet

. Toute partie donnant lieu à des manipulations par l'application est un objet (l'application signifiant ici la partie d'application pure, et non la partie de traitement du code image).

Les objets sont structurés d'une part en arbres, d'autre part en ensembles. Ils possèdent un nom symbolique qui assure la lisibilité de la description des évolutions de l'image dans l'application.

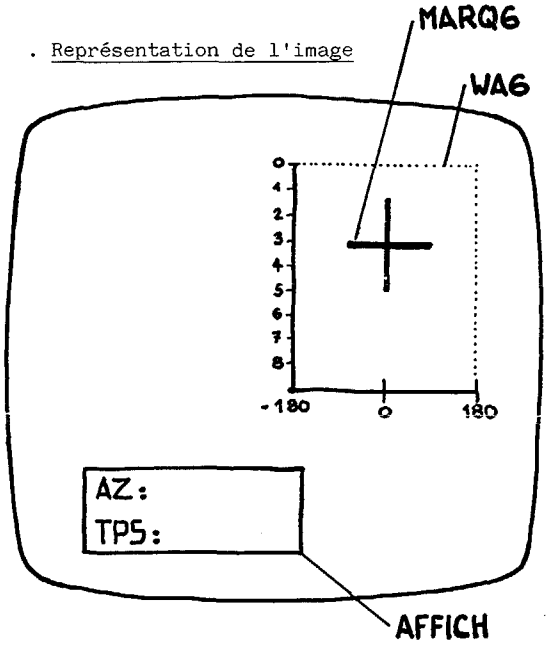
Les spécifications de l'image (Cf figure 3) peuvent être définies :

. Soit par la liste source de description de l'image

. Soit par une représentation de l'image, sur laquelle on fait apparaître les noms des principaux objets.

Dans l'ensemble de la figure, l'application pourra changer la couleur de l'objet AFFICH, mais pas celle de son cadre séparément. Ainsi lors de la description de l'image, son évolution doit être connue et certaines opérations se trouvent alors être autorisées et d'autres non. La position de l'objet MARQ6 se trouve automatiquement liée à celle d'un marqueur boule : l'application ne devra pas gérer l'évolution de cette position.

. Représentation de l'image



. Source LDI

```
Déclaration de variables
  XB, YB : position de la boule
Définition d'AXE (paramètres)
Définition de WATERFALL (paramètres)
IMAGE VEILLE
  OBJET WA6 (attributs)
    AXE (...)
    AXE (...)
    WATERFALL (...)
    OBJET MARQ6 (attributs)
      POLYLINE (2, XB-L, YB, XB + L, YB)
      POLYLINE (2, XB, YB-L, XB, YB + L)
    FIN
  FIN
  OBJET AFFICH (attributs)
    POLYLINE (5, ...)
    TEXT (XT1, YT1, 'AZ : ')
    AZIMUT = A* (XB - XO)
    EDINUMC (AZIMUT, F6.1) ; position cou-
                             rante
    TEXT (XT2, YT2, 'TPS : ')
```

```
        TEMPS = T* (YB - YO)
        EDINUMC (TEMPS, F4,2)
    FIN
FIN
```

Figure 3.

Le langage permet la description de plusieurs types de symboles objets :

. Les images qui représentent un ensemble d'objets présents simultanément en mémoire de la console de visualisation (allumés ou éteints, c'est-à-dire n'apparaissant pas toujours sur l'écran). Les évolutions à l'intérieur d'une image doivent être rapides ; par contre, le passage d'une image à une autre peut être lent

. Les objets, formés d'une liste de primitives de d'une liste de sous-objets (donc structurés en arbre)

. Les entités, objets de plus bas niveaux, ne comportant que des primitives

. Les modèles qui permettent de générer en temps réel un nombre variable d'objets identiques (en structure, mais pouvant avoir des attributs différents)

. Les primitives étendues qui sont des sous-programmes de primitives, utilisées ensuite dans le langage comme les primitives de base

. Les ensembles qui sont des regroupements d'objets (ou d'ensembles), en dehors de la hiérarchie arborescente, sur lesquels l'application voudra appliquer une même transformation.

Les symboles variables peuvent être de type entier, réel ou chaîne de caractères.

Les attributs des objets sont :

. La visibilité (allumé/éteint)

. La mise en valeur (plusieurs niveaux)

. La détectabilité (oui/non)

. La priorité (plusieurs niveaux)

. La matrice de translation/rotation/échelle.

Les attributs des primitives (différents selon le type de primitives) sont les mêmes que ceux du standard graphique choisi.

Chaque attribut peut être absolu ou relatif : les attributs des objets sont relatifs au sur-objet, les attributs des primitives à une valeur courante.

4. PRINCIPES DU CODE IMAGE

Le code image permet la représentation de la structure de l'image, au moyen de listes chaînées entre elles. Il mémorise toutes les informations contenues dans la source de description d'images. Sa structure permet tout remaniement ou extension des listes.

Tous les paramètres, qu'ils soient liés aux objets (priorité, rotation ...) ou aux primitives (épaisseur de trait, taille des caractères ...) peuvent être soit des constantes, soit des variables, soit des expressions.

Les diverses tables formant le code standard sont (Cf figure 4.) :

. Une table de symboles faisant correspondre à chaque nom de symbole, un type (objet, variable ...) et un numéro

. Une table d'adresses, donnant pour chaque numéro d'objet l'adresse de description de sa structure

. Une table de structure, contenant la description de la structure de chaque symbole de type objet : attributs liés à l'objet, adresse de la liste des primitives de l'objet, liste des sous-objets de l'objet

. Une table de primitives, structurée en listes correspondant à chaque objet

. Une table de valeurs contenant des constantes, des expressions (mémorisées en notation polonaise inversée) et les valeurs de variables

. Une table d'occurrences, formée des listes pour chaque variable des objets dans lesquels cette variable intervient.

Une première table d'implantation permet de connaître les longueurs de chacune des autres tables, afin de permettre le chargement dans l'application. Dans le cas d'une application disposant d'un espace mémoire restreint, cette structure permet de gérer facilement les informations d'image en mémoire virtuelle.

LANGAGE DE HAUT NIVEAU POUR IMAGES SONAR
HIGH LEVEL LANGUAGE FOR SONAR IMAGES

Mr LACOSTE Mme GUIMBAL Mr BRIZARD

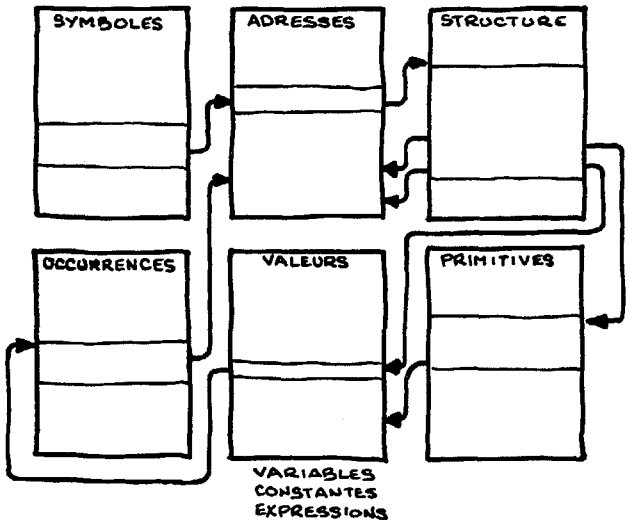


Figure 4.

5. MODE INTERACTIF

Le mode interactif permet soit la création, soit la manipulation (pour la mise au point) d'objets.

Il est utilisé hors temps réel et conduit comme le langage à une bibliothèque d'objets. Tous les objets définis par le langage peuvent aussi l'être en mode interactif.

L'utilisateur peut définir ou modifier des menus composés de textes, ou de symboles représentant des objets. Il peut sélectionner dans ces menus un mode ou un objet. Il peut désigner sur la partie de l'écran où est représentée l'image : soit une primitive, soit le plus petit objet la contenant, soit un objet de rang supérieur la contenant. Dans tous les cas, un feed-back (surbrillance, clignotement ...) permet de vérifier que la partie choisie est bien celle reconnue.

Les primitives sont les mêmes que celles du langage, mais interactives. ELles peuvent être choisies dans un menu, ou tapées au clavier alphanumérique. Les positions peuvent être choisies par positionnement d'un marqueur sur l'écran. Si l'objet est paramétré, le nom du paramètre est tapé au clavier.

Les objets existant en bibliothèque peuvent être tracés sur l'écran. Dans le cas d'objets paramétrés, il existe plusieurs modes possibles : soit on utilise la valeur courante du paramètre, soit on interroge l'opérateur sur les valeurs souhaitées.

Les actions possibles sont les modifications d'attributs sur les objets ou les primitives, la destruction ou l'insertion d'objets ou de primitives, les transformations de rotation, translation, échelle sur les objets ou les primitives, les modifications de texte ... Les rotations et translations peuvent se faire par asservissement de l'objet du marqueur.

6. INTERFACE AVEC L'APPLICATION

Dans le développement classique d'une application, le logiciel opérationnel englobe la partie logiciel de visualisation qui se trouve codée à l'aide de primitives graphiques (GKS ou autre) implémentées dans le langage de l'application.

Avec le système API, l'application communique avec un logiciel de visualisation indépendant de l'application, au moyen d'ordres de haut niveau :

CHARGER < Image > appelle depuis une mémoire de masse, la liste de visualisation et les tables de structure associées à l'image spécifiée.

ALLUMER }<Objet > rend visible ou invisible un
ETEINDRE }
objet dans une image.

MODIFIER < Objet, attribut, valeur > modifie la valeur d'un attribut dans l'objet désigné.

MODIFIER < Variable > Prise en compte de la nouvelle valeur d'une variable pour le recalcul de tous les objets dans la description desquels elle intervient.

D'autres ordres permettront aussi des échanges dans le sens API vers l'application pour donner le moyen à celle-ci de connaître les entrées de l'opérateur : lecture de valeurs numériques ou alphanumériques, désignation d'objets, etc ...

7. CONCLUSION

Le système API est actuellement en cours de spécification et nous comptons rapidement l'utiliser dans nos applications.

Cette démarche :

. Sépare les activités logiciel et image

. Facilite un développement en parallèle de ces deux activités

. Permet de voir suffisemment tôt les images avant même que les moyens de visualisation définitifs ne soient disponibles

. Tolère les évolutions tardives des spécifications d'images

. Et surtout, réduit le coût et le temps de développement des images.

Dans l'avenir, nous pensons développer un système de gestion des dialogues, en parallèle avec l'API, qui permettra de gérer simplement le graphe des actions opérateur autorisées.