Um Sistema Gráfico Hierárquico sobre X-Windows

M. J. Próspero
J. J. Godinho
J. Cruz

Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Quinta da Torre, 2825 Monte de Caparica

RESUMO

Com vista a uma aplicação gráfica no âmbito da Programação em Lógica, em que se pretendia um estilo declarativo de programação, concebeu-se e implementou-se um Sistema Gráfico 2D em X-Windows (SGX). A funcionalidade inclui características encontradas nas actuais normas gráficas, nomeadamente em GKS e em PHIGS. Assim, é possível a construção directa de modelos de segmentação hierárquicos mas cujas propriedades (herdadas ou não) são atributos de cada segmento individual abrangendo todas as suas primitivas. Também a entrada gráfica foi contemplada com a introdução de dispositivos lógicos de acordo com as classes que se encontram nas referidas normas gráficas.

1. Âmbito

Numa tentativa de libertar o programador de aplicação das características estritamente procedimentais relacionadas com a descrição dos modelos ao sistema gráfico usado numa dada instalação, concebeu-se um sistema da Modelação Gráfica em Lógica (MGL) inteiramente escrito em Prolog [PR88b]. A sua interface com o programador, além de preservar a funcionalidade que é habitual nos sistemas gráficos 2D (por exemplo, ao incluir as transformações geométricas elementares), permite deduções com a informação geométrica ou topológica disponível, mesmo que incompleta [PRÓ90]. Estas deduções são conclusões que se extraem automaticamente dos factos e regras declarados como existentes num certo domínio, sendo essa capacidade uma das virtudes das denominadas linguagens de programação em lógica [BRA86]. Como ilustração do que acaba de ser referido, sugere-se a leitura da comunicação referida em [PR88a], onde se analisa um caso de aplicação.

É claro que qualquer sistema cuja execução se baseie numa semântica declarativa tem a sua concretização, ao fim e ao cabo, numa linguagem dita procedimental. Em último recurso, esta será a linguagem da "máquina de Von Neuman" que a tecnologia do presente ainda não ultrapassou. Tal afirmação inclui o caso do sistema de MGL. Quando, então, se pensa na implementação das camadas de baixo nível, há que fazer as opções que melhor se ajustem à funcionalidade pretendida no nível mais alto sem, contudo, degradar em excesso a eficiência do produto final. Como se sabe, existe sempre um compromisso entre aquilo que se pode calcular em tempo de execução (poupando memória) e o que pode ser armazenado para uso posterior (poupando tempo de cálculo).

Assim, para o sistema de MGL resolveu-se conceber e escrever um sistema gráfico dedicado, denominado SGX, por sua vez assente no conhecido X-Window System [JON89]. As opções que foram tomadas tinham, como objectivo fundamental, a implementação eficiente dos predicados Prolog acessíveis ao programador [PR88b]. Não se trata, portanto, de um Sistema Gráfico de uso geral, como o são GKS [END87] e PHIGS [BRO85], embora tenha com eles muitas afinidades por terem sido estes a sua principal fonte de inspiração.

Os aspectos mais salientes de SGX envolvem a concepção e implementação de segmentos permitindo a sua posterior edição e, simultaneamente, possibilitando o seu relacionamento de forma hierárquica. Para se ter uma ideia da importância do papel dos segmentos neste sistema gráfico, bastará referir que um programa de aplicação que o use não poderá chamar qualquer primitiva de saída gráfica fora de um segmento.

Por tais motivos, o texto da presente comunicação incidirá especialmente sobre os principais aspectos ligados à segmentação em SGX. Quando, nas secções seguintes, for necessário apresentar detalhes da implementação, a linguagem de programação C será a utilizada [GET88].

2. Filosofia de concepção

Cada segmento é composto por um conjunto de valores de atributos (que definem o estado do segmento) e por uma sequência de, digamos, instruções gráficas (ou elementos, na terminologia de PHIGS). Estas, normalmente, são primitivas de saída gráfica. A referida sequência descreve, pois, a ordem pela qual essas primitivas são executadas, tendo sempre em conta os valores correntes dos atributos em tempo de execução. A referência a qualquer outro segmento, desde que a recursividade (mesmo a indirecta) seja evitada, é também uma instrução admissível. Permite-se, assim, a criação de segmentos de forma hierárquica, relações essas que podem representar-se por grafos orientados e sem ciclos.

Para uniformização do sistema e adequação ao modelo em lógica, à raiz de uma dada hierarquia fica associado um segmento. Contudo, ele possui caraterísticas muito próprias: não contém primitivas de saída gráfica, não pode ser descendente de qualquer outro segmento (ou

seja, não pode ser referenciado) e é por ele que se define a transformação de normalização que será aplicada a toda a hierarquia. Em SGX é possível a existência de hierarquias múltiplas, cada qual com a sua raiz. Por omissão, existirá sempre uma raiz pré-definida e cuja transformação de normalização é inalterável.

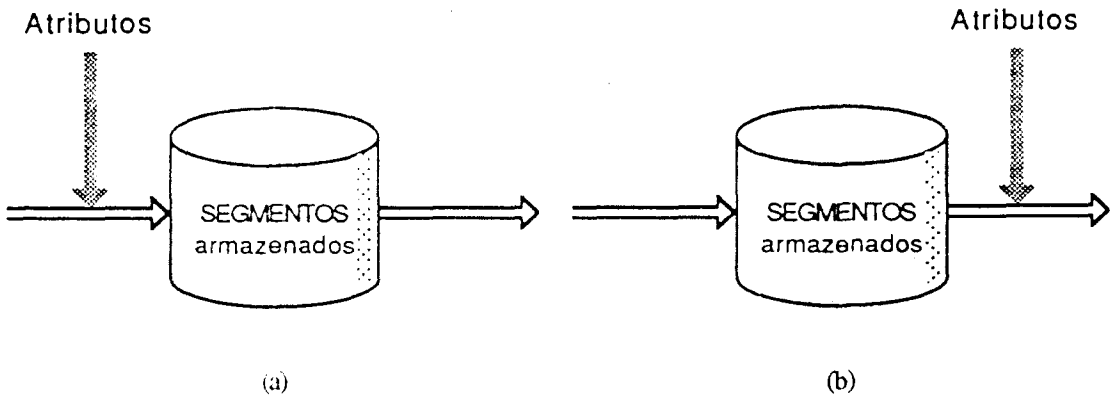

Figura 1 — Ligação dos atributos em GKS (a) e em SGX (b).

A definição de cada segmento é independente da especificação de uma raiz. No entanto, aquele só poderá ser visível no ecrã se pertencer a uma hierarquia existente. Por isso, o processo de visualização tem sempre início na(s) raiz (raízes) existente(s) e propaga-se pelos seus descendentes.

Quando se inicia a execução do programa de aplicação e existe informação gráfica associada, na forma de estruturas de dados apropriadas, é gerada a imagem correspondente no ecrã. É sobre estas estruturas de dados, parte essencial do modelo da aplicação, que se irão processar todas as transformações que são essenciais num programa interactivo com gráficos. Assim, o processo de visualização pode ser decomposto em duas fases: a primeira é a da geração das estruturas de dados a partir da informação prestada pelo programa de aplicação e a segunda corresponde à travessia da(s) hierarquia(s) com a correspondente execução das instruções gráficas. No fundo, esta última fase é uma especialização da descrição dos objectos até se atingir uma representação directamente em X-Windows (por exemplo: segmentos de recta, pontos e rectângulos), tendo em conta os valores efectivos dos atributos que se reflectem na imagem no ecrã (Fig. 1).

3. Entrada Gráfica

SGX apresenta as seguintes classes de dispositivos lógicos de entrada gráfica: *locator*, *pick*, *valuator*, *keyboard* e *choice*. A cada dispositivo lógico está associado um dispositivo físico, de acordo com as estratégias adoptadas para a implementação. Na actual, apenas

A Fig. 5, mais complexa, surge da criação de um terceiro segmento, denominado *"B"*, e da introdução de primitivas gráficas, tanto em *"A"* como em *"B"*. É de notar o facto do campo *num_inst* de qualquer dos segmentos existentes ser zero, que se explica por nenhum deles ter sido ainda referenciado.

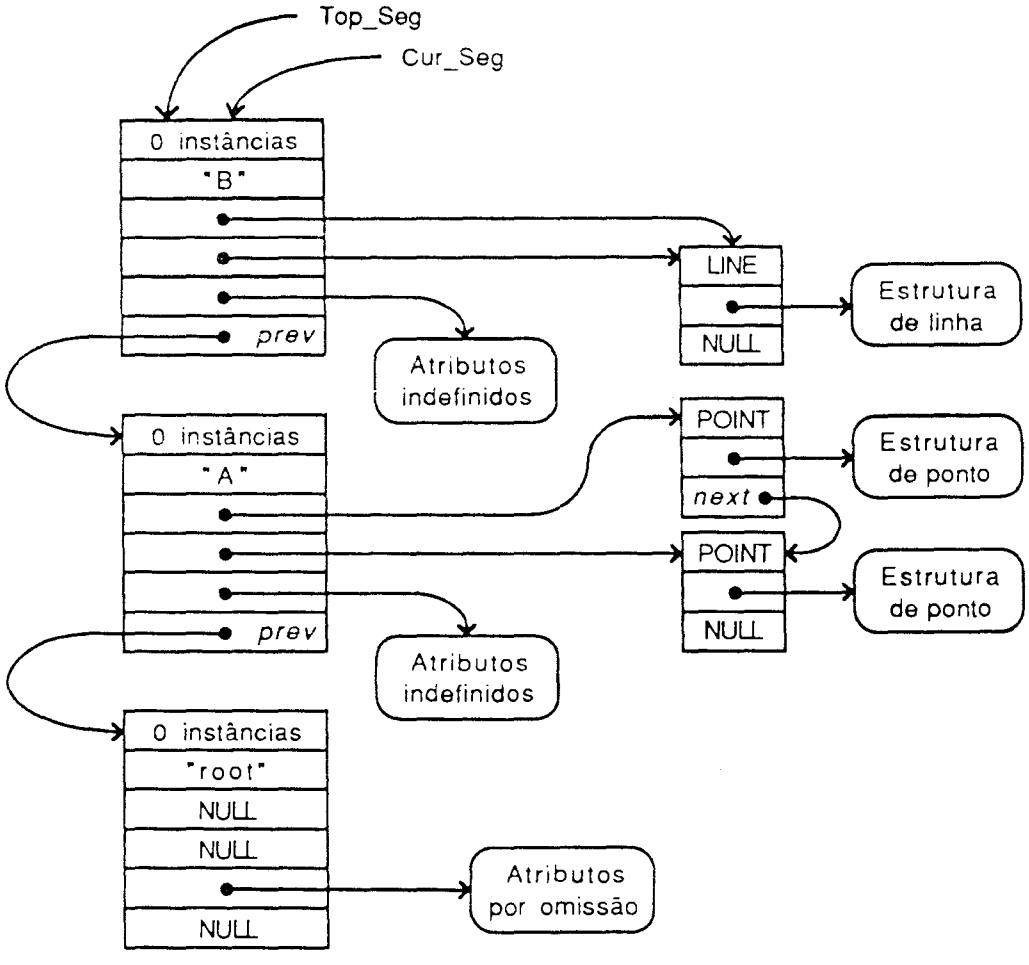

Figura 5 — Segmentos com algumas primitivas de saída gráfica.

A necessidade de níveis de segmentação está intimamente associada, do ponto de vista gráfico, ao facto da generalidade das entidades passíveis de modelação não serem monolíticas, mas, antes, conceptualmente decomponíveis noutras. Existem sistemas gráficos no mercado que obrigam o programador a usar identificadores distintos para segmentos, mesmo que eles reflictam instâncias da mesma entidade. Isto acontece sempre que haja uma relação unívoca entre segmento e objecto gráfico representado no ecrã.

Como exemplo, suponhamos que pretendemos visualizar duas imagens exactamente iguais, mas ocupando lugares distintos no ecrã. Seja a entidade *objecto* a que está em causa. Usando GKS poder-se-ia escrever código da forma

```
create_segment ("objecto1");
```

```
.................
/*  Instruções que especificam objecto  */
.................
close_segment ();
set_segment_transformation ("objecto1",M1);
create_segment ("objecto2");
.................
/*  Instruções que especificam objecto  */
.................
close_segment ();
set_segment_transformation ("objecto2",M2);
```

No caso de SGX ter-se-ia:

```
create_segment ("objecto");
.................
/*  Instruções que especificam objecto
    no espaço de coordenadas de modelação  */
.................
close_segment ();
reference ("objecto",M1,...);
reference ("objecto",M2,...);
```

Assim, a cada instância de um mesmo segmento é que irá corresponder eventualmente uma imagem, ou seja, um objecto gráfico [PR88b].

6. Hierarquias e Objectos Gráficos

As referências a segmentos, traduzidas pelo campo *ref* em *segment_key*, é que definem a estrutura hierárquica de um modelo. Para tal efeito, a implementação de um tipo *rf* é assim realizada:

Campo	Descrição	Tipo
atr	Apontador para uma estrutura de atributos	&ATTR
trans	Matriz de transformação	&FLOAT
instance	Apontador para o segmento referenciado	&SEGMENT

7. Conclusões

Apresentaram-se algumas das características mais significativas de SGX. Como se constata, possuem certas semelhanças com as das actuais normas gráficas. Porém, dada a funcionalidade muito específica que se pretendia, SGX mostra também bastantes diferenças. Muito embora a descrição que constitui as secções anteriores não tenha sido exaustiva, o leitor fica com a sensação que este sistema é de dimensões muitíssimo mais reduzidas que GKS ou PHIGS. É uma verdade que, ao mesmo tempo, se torna numa vantagem, pois exclui da implementação inúmeras funções que caem fora do âmbito previamente estabelecido, resultando um sistema muito menos pesado.

Numa fase inicial, foi escrito um protótipo a funcionar sobre GKS e, mais tarde, com um interpretador para PHIGS [PR88b]. O sistema de MGL era, assim, facilmente transportável, mas muito ineficiente (principalmente com GKS, onde existe apenas um único nível de segmentação).

Com a posterior restrição do sistema de operação ao UNIX, foi possível desenvolver rapidamente SGX com a norma *de facto* que é o X11. Num mercado que se afigura de expansão significativa nos próximos anos, presume-se que foi um compromisso bastante razoável.

A ligação, propriamente dita, de SGX com o interpretador da linguagem Prolog ainda não foi realizada. A implementação do sistema gráfico prossegue, actualmente, numa estação de trabalho VAXstation 3200, equipamento que foi adquirido pelo Departamento de Informática da UNL e totalmente financiado pela Fundação Volkswagen no âmbito do projecto DIGRA ("Distributed Graphics") envolvendo o grupo de Computação Gráfica do Professor Doutor José Encarnação na Universidade Técnica de Darmstadt (RFA).

8. Referências bibliográficas

[BRA86] Bratko, I.
Prolog Programming for Artificial Intelligence, Addison-Wesley (1986)

[BRO85] Brown, M.
Understanding PHIGS, Megatek Corp., San Diego, Calif. (1985)

[END87] Enderle, G.; Kansy, K.; Pfaff, G.
Computer Graphics Programming / GKS — the Graphics Standard, 2nd edition, Springer-Verlag (1987)

[GET88] Gettys, J.; Newman, R.; Schafler, R.
X lib — C Language Interface (Vers. 11), Massachusetts Institute of Technology, Cambridge, Massachusetts (1988)

[JON89] Jones, O.
Introduction to the X Window System, Prentice-Hall (1989)