Artigo

SIBGRAPI'91
IV Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens

UM EDITOR GRÁFICO INTELIGENTE

Roberto de Beauclair Seixas
(LNCC/CNPq)

Lilia de Assunção Hess
(IME-RJ)

Página em branco na versão original impressa.

Um Editor Gráfico Inteligente

Roberto de Beauclair Seixas

CNPq/LNCC

Rua Lauro Müller, 455

CEP 22290 - Rio de Janeiro - RJ

e-mail: usertron@lncc

Lilia de Assunção Hess

IME

Praça General Tibúcio, 80

CEP 22290 - Rio de Janeiro - RJ

e-mail: s9com01@imerj

Abstract

The Intelligent Graphical Editor (EGI) enables Artificial Intelligence environments with graphic resources and allows the construction in geometric form of objects, with the same language of fact description.

The EGI is a graphic tool with a menu of basic operations. From these operations, and operands furnished by the user, one can construct new operations and more complex objects, through one or more knowledge basis.

1 Introdução

Normalmente, um editor gráfico é uma interface interativa e amigável, entre o usuário e um sistema gráfico [QUEIROZ]. Nos sistemas de CAD, ainda existe um banco de

dados associado ao desenho, no qual se detalham características técnicas dos objetos. Este banco de dados é útil quando o objetivo final é a integração com um sistema de manufatura CAD/CAM [MATSUO].

No entanto, em Inteligência Artificial, seria interessante a interface homem/máquina permitir a aquisição de conhecimento, através de um acesso aos métodos e à inferência na base de conhecimento. Por associar Computação Gráfica e Inteligência Artificial, o EGI permite inclusive a aquisição de conhecimento gráfico.

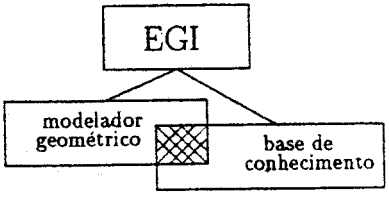

Figura 1: Sistema do Editor Gráfico Inteligente

2 Objetivo

Combinar metodologias de Computação Gráfica e de Inteligência Artificial.

Utilizar, de forma integrada e dependente, o editor gráfico e a base de conhecimento associada, permitindo o desenvolvimento de aplicativos em áreas que utilizem técnicas de Inteligência Artificial e necessitem de recursos gráficos interagindo com a base de conhecimento.

Desta forma, podem ser obtidas transformações nos objetos pela utilização de regras e/ou pela modificação interativa dos fatos contidos na base de conhecimento, possibilitando uma representação independente das transformações sobre os objetos e dos objetos gerados pelo editor gráfico.

3 Modelo de Representação

Na implementação de um modelador geométrico [FOLEY], componente básico de um editor gráfico, é ideal que a manipulação de objetos seja feita de forma rápida e eficiente. Para isto foi desenvolvido um modelo de representação interno, que não interfere no modelo conceitual utilizado pelo usuário quando da especificação dos objetos. Além disto, podemos utilizar gráficos como entrada de dados, como por exemplo grafos, árvores, etc. A representação do objeto, dada pelo usuário de acordo com o seu modelo conceitual, é passada ao EGI através de *fatos gráficos*, que são então convertidos numa estrutura de dados manipulável pelo modelador geométrico.

Denominam-se fatos gráficos aqueles que representam as primitivas de desenho, como uma forma de distinguí-los dos fatos relativos ao problema.

O modelo utilizado internamente no EGI é transparente ao usuário, e é dividido em três camadas, as quais, da mais interna à mais externa, são detalhadas a seguir:

A camada mais interna (Estrutura de Dados do Modelador Geométrico) é aquela que trata da forma de armazenamento das figuras na memória. É através dela que os fatos gráficos serão convertidos em uma estrutura de dados compatível com o editor gráfico, evitando com isto que os algoritmos de manipulação das primitivas tenham que ser muito alterados.

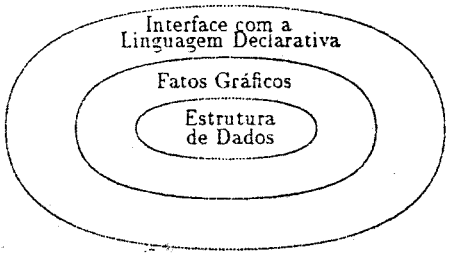

Figura 2: Modelo em camadas

A camada do meio é aquela que trata diretamente com os fatos que se referem às primitivas do desenho, os fatos gráficos. Nesta camada é que os fatos podem ser manipulados pelo usuário de forma que qualquer alteração, por menor que seja, tenha os seus fatos alterados, resultando numa modificação na estrutura de dados associada, através da

conversão realizada pela camada mais interna. Isto permitirá uma separação mais eficiente e mais interativa para o usuário, entre o editor gráfico e a base de conhecimento usada pelo motor de inferência. Porque o usuário, independente da estrutura de dados associada, só terá que se preocupar com os fatos.

A camada mais externa é aquela que faz, ou possibilita, a interface com uma linguagem declarativa qualquer. Nesta camada, os fatos gráficos vão ter que estar associados aos comandos de entrada, ou melhor, de aquisição de conhecimento, fazendo com que esta linguagem aprenda os fatos gráficos. Pensando agora no caminho inverso, também através desta camada os dados serão lidos para o EGI.

Para que o modelo interno fosse eficiente e flexível o bastante para ser manipulado por uma linguagem declarativa qualquer, foi necessário uma maneira de interfacear o EGI com a linguagem declarativa, ou melhor, trocar informações de forma simples, de modo a não prejudicar as vantagens obtidas com a estrutura de dados.

A forma empregada para suprir esta interface foi a criação de um arquivo, que armazena a estrutura de dados dos objetos como fatos, e que pode ser lido diretamente pela linguagem declarativa como um arquivo comum à linguagem. Assim, os atributos dos objetos se tornam predicados, incorporando um predicado global, para identificar que o fato presente na base de conhecimentos é um objeto.

Isto pode ser melhor entendido com o predicado exemplificado abaixo:

objeto(identificador,grupo(1),cor(azul),linha(10,10,20,20))

Para tornar ideal este interfaceamento, seria necessário que os fatos fossem separados da parte procedimental da linguagem, isto é, os fatos e as regras em um nível de abstração e o programa que se utilizará destes fatos e regras em outro. Com isso, uma mesma série de regras e procedimentos pode ser aplicada a vários objetos, em bases de conhecimento diferentes, resultando em figuras geradas por semânticas diferentes. Este é um dos fatores que o qualificam como inteligente.

A linguagem declarativa escolhida foi o Safo (Sistema de Aquisição e Formalização do Conhecimento) [LINS] que possui os dois níveis necessários ao modelo utilizado. No nível denominado *nível de baixo*, o Safo funciona somente para o armazenamento de fatos e regras

que poderão, ou não, ser usados pelo *nível de cima* quando solicitado, durante a execução, pelo motor de inferência.

Deste modo, pode-se armazenar um certo tipo de conhecimento sobre os objetos no *nível de cima* e aplicá-los a conjuntos de fatos e regras de objetos diferentes, pela utilização de múltiplas bases de conhecimento ou, simplesmente, pela carga de uma base de cada vez no *nível de baixo*. Não obstante, o Safo permite as duas maneiras.

Também foram aproveitadas as facilitades de recursão e *back-tracking* da linguagem declarativa empregada, e conhecimentos diferentes sobre um mesmo conjunto de fatos gráficos ou vice-versa.

A utilização de gráficos através de linguagens declarativas foi proposta de forma clara por Helm & Marriott [MARRIOTT], que diziam que as pessoas se utilizam de uma visão bottom-up na descrição e reconhecimento de figuras, mas se utilizam de uma visão top-down na sua geração. Consequentemente, a programação lógica é bem apropriada para Computação Gráfica [NETO].

Geralmente, a especificação de figuras por linguagens declarativas também pode ser vista através de estruturas hierárquicas de dados gráficos. Isto permite que o usuário consiga manipular esta estrutura para aplicações dinâmicas e interativas. A hierarquia [MARRIOTT] pode ser considerada na própria estrutura de dados, possibilitando que se tenha um algoritmo recursivo e não determinístico que percorra a hierarquia da figura, calculando as transformações geométricas em cada nó. Este algoritmo pode ser usado tanto para o reconhecimento de figuras quanto para a sua geração.

O modelo utilizado no EGI realiza a busca ordenada de fatos na base de conhecimento e embora possibilite a descrição das figuras de forma hierárquica, não constitue uma imposição à escolha do modelo conceitual utilizado pelo usuário.

Portanto, os objetos gráficos não estão vinculados a uma descrição com coordenadas no espaço cartesiano e dependem somente da base de fatos. A manipulação dinâmica dos objetos é dada pela alteração na base de fatos escolhida.

Dito isto, um editor gráfico, construido com as descrições acima, deve ser simples, pequeno e versátil. Esta simplicidade resulta da facilidade na manipulação dos dados, do

acesso à estrutura da figura e ao fato de ser não determinístico.

3.1 Interface com a Linguagem Declarativa

Esta interface deve possibilitar a troca de informações entre o editor gráfico e a linguagem declarativa. No caso do EGI, a forma de entrada dos dados é através de um arquivo que contém os fatos gráficos. Para este arquivo de fatos gráficos ser manipulado pela linguagem declarativa, tem-se que acrescentar, em cada fato gráfico, o comando específico da linguagem para a aquisição de conhecimento.

O mais interessante deste processo é que o comando incorporado ao fato gráfico, qualquer que seja a sua sintaxe, não interfere na leitura do arquivo pelo EGI. Para isto ser possível, a leitura deste arquivo de fatos gráficos pelo EGI é feita na forma de comparação de predicados, onde as palavras chaves são as primitivas e os seus atributos.

Para evitar que o usuário tenha, por acaso, criado algum fato referente ao seu problema com um nome de primitiva, todos os fatos gráficos foram acrescidos da palavra "egi_". Assim, não restringimos o usuário e conseguimos até uma forma mais padronizada de representação.

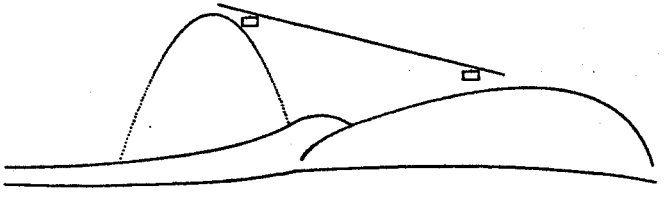

Figura 3: Exemplo de utilização

Um bom exemplo seria a utilização de uma figura em que se reunissem algumas das possibilidades de utilização das primitivas do editor gráfico.

A figura acima foi animada através da interação do motor de inferência com a base de conhecimento, respeitando as regras de deslocamento dos bondinhos e os procedimentos para desenho e apagamento das posições dos bondinhos.

A seguir, o detalhe de um procedimento como se encontra na base de conhecimento, de uma pequena animação de um círculo, que será lido pelo motor de inferência do Safo.

```
anima ::= prove(egi_obj(*id,*grp,*cor,egi_circulo(*x,*y,*raio)))
        & incremente(*x,5)
        & execute(EXIBEEGI.EXE)
        & <(*x,640)
```

Foi realizada a interface com o processador de textos LaTeX possibilitando a importação das figuras. Como exemplo, as figuras contidas neste artigo foram assim feitas.

O ambiente de desenvolvimento foi um Microcomputador IBM-PC/AT compatível, com placa EGA e mouse. A linguagem adotada foi o Turbo Pascal 5.5 da Borland para o desenvolvimento da interface gráfica e o Safo da Intelect como linguagem declarativa.

4 Conclusão

O EGI é uma ferramenta gráfica que inclui um menu de operações básicas. A partir destas operações e de operandos fornecidos pelo usuário, pode-se construir novas operações e objetos mais complexos, através de uma ou mais bases de conhecimento.

Esta abordagem, que carateriza o EGI, implicou em benefícios para as duas áreas envolvidas. No que diz respeito à Inteligência Artificial, o uso de gráfico possibilitou uma melhor visualização, uma melhor representação de certos tipos de conhecimentos, através da mesma linguagem de descrição dos fatos.

No que diz respeito à Computação Gráfica, algoritmos procedimentais foram reescritos na linguagem declarativa Safo, e implementadas novas operações e operandos de objetos gráficos, de forma simples e dinâmica, com a inclusão de regras e fatos na base de conhecimento associada.

Bibliografia

[FOLEY] J. D. Foley, A. Van Dam, **Fundamentals of Interactive Computer Graphics**. Addison-Wesley Publishing Company, 1984.

[QUEIROZ] G. de B. e Queiroz, **Implementação de uma Ferramenta Gráfica de Uso Geral**, Tese de Mestrado - Instituto Militar de Engenharia, 1987.

[NETO] E. C. Oliveira Neto, **Uma Proposta para Programação Lógica**, Tese de Mestrado - Instituto Militar de Engenharia, 1988.

[MARRIOTT] R. Helm, K. Marriott, *Declarative Graphics*, **III International Conference of Logic Programming**, p.513–527, (1986).

[MATSUO] H. Matsuo, J. Shang, R. Sullivan, "A Knowledge-Based System for Stacker Crane Control in a Manufacturing Environment", *IEEE Transactions on Systems, Man, and Cybernetics*, **Volume 19**, Number 5, p.932-945, (1989).

[SEIXAS] R. de Beauclair Seixas, **Editor Gráfico Inteligente**, Tese de Mestrado - Instituto Militar de Engenharia, 1990.

[LINS] R. Lins de Carvalho, **Safo - Manual do Usuário**, 1990.