Desenvolvimento De Algumas Extensões À Biblioteca VOGL Para Aplicação em Gráficos Moleculares

Pedro A. M. Vazquez

Instituto de Química
Universidade Estadual de Campinas
Caixa Postal 6154
13081-970, Campinas, SP, Brasil
vazquez@iqm.unicamp.br

Abstract. The Iris GL clone VOGL was extended to support a set of functions essential to molecular graphics. The resulting enhanced library was used to sucessfuly port GL applications to Unix/X11 based environments and to develop new applications suitable to compile and run on both GL and VOGL/X11 platforms.

Introdução

A Química situa-se entre as áreas de maior aplicação da computação gráfica e de imagens atualmente. A visualização interativa de estruturas moleculares, iso-superfícies de densidade eletrônica ou a animação de processos de reação, por exemplo, auxiliam de forma notável a interpretação de resultados de cálculos de estrutura eletrônica e o desenvolvimento de uma percepção intuitiva dos mesmos. Devido à natureza tridimensional dos objetos gráficos de interesse químico qualquer biblioteca que se pretenda empregar no desenvolvimento de aplicações deve possuir um conjunto básico de recursos como *Z-buffering*, *depth cueing*, e *shading*. Dentre as bibliotecas existentes para o ambiente Unix a Iris GL da Silicon Graphics acabou tornando-se o padrão de fato para o desenvolvimento de um grande número de programas na área de química. Embora dotada de todos os recursos citados e, em virtude de operar com hardware adequado, possuir um desempenho excelente, a GL está restrita basicamente a equipamentos Silicon Graphics e IBM/RISC6000 nem sempre acessíveis aos pesquisadores.

Com a finalidade de contornar esta limitação e obter uma biblioteca que possa ser utilizada em diversas plataformas Unix, nesta comunicação relatamos as modificações e adições efetuadas na biblioteca VOGL, clone da GL, de forma que esta possa ser utilizada tanto para a compilação de programas escritos para a GL quanto para o desenvolvimento de novos programas em sistemas Unix que possuam ambiente X11.

Descrição das modificações

A VOGL, desenvolvida e distribuida gratuitamente na forma de programa fonte em C pelo *Department Of Engineering Computer Resources* da Universidade de Melbourne, consiste num subconjunto de funções da GL que permitem o desenvolvimento de programas compatíveis com esta. Embora a VOGL possua a capacidade de operar com diversos dispositivos de saida (HPGL, PostScript, X11, etc) as modificações relatadas nesta comunicação aplicam-se apenas ao dispositivo X11 procurando-se manter inalteradas as funções já existentes. Para atingir os objetivos desejados as alterações foram feitas nas funções públicas e nas funções privativas da VOGL. Estas modificações foram realizadas utilizando o compilador C da GNU e a distribuição X11R5 do MIT em equipamentos Sun (Sparcstation 1+/SunOS4.1.1), IBM (RS6000/AIX 3.2) e PC (486 e 386-387/386BSD 0.1). A seguir são descritas estas modificações.

Remoção de linhas e superfícies ocultas

A remoção de linhas e superfícies ocultas foi implementada através de *Zbuffering* pela adição das funções públicas `zbuffer()` e `zclear()` e pela modificação das funções privativas `draw()`, `X11_draw` e `X11_fill`. Caso o *Zbuffer* tenha sido alocado e ativado através de uma chamada a função `zbuffer(yes, no)`, as linhas passam a ser traçadas utilizando o algoritmo padrão de Bresenham [Brown et al, 1991] e realizando o teste de visibilidade ponto a ponto. Nesta etapa também é realizado o *depth cueing* caso habilitado por chamada as funções `depthcue()` e `lsetdepth()`.

A remoção de superfícies ocultas (assim como os efeitos de iluminação descritos abaixo) foi implementada apenas para polígonos triangulares. Para o caso de polígonos com um número maior de vértices estes são reduzidos a fitas de triângulos antes da operação ser realizada [Judd & Nelson, 1988]. Para tanto foram realizadas, também, modificações nas funções privativas que lidam com polígonos. O método empregado em `X11_fill` foi o de *scanline* simples [Ng & Bhouris, 1992] com uma ligeira modificação no algoritmo de interpolação dos valores de Z.

Efeitos de Iluminação

Para a inclusão de efeitos de iluminação à VOGL foram adicionadas as funções públicas `RGBcolor()`, `lmdef()`, `lmbind()` e `shademodel(`*model*`)` (com *model* = `NONE`, `FLAT` ou `INTERP`) necessárias para a definição das fontes de luz, propriedades das superfícies dos objetos e modelo de iluminação a ser calculado. A rotina de inicialização da VOGL também teve de ser modificada para a criação de um mapa de cores indexadas do tipo 3/3/2 (2^3 = 8 níveis de vermelho, 2^3 = 8 níveis de verde e 2^2 = 4 níveis de azul).

Para *model* = `FLAT` e `INTERP` foi empregado um modelo simples de reflexão que leva em conta a iluminação ambiente mais as componentes de reflexão difusa e especular. No caso *model* = `FLAT` este cálculo é feito apenas em relação ao vetor normal a superfície de cada triângulo e a cor obtida é utilizada para todo o triângulo. Quando *model* = `INTERP` a cor de cada pixel é calculada utilizando o vetor normal sobre o mesmo obtido pela interpolação dos vetores normais em cada vértice. Para isto foi necessário a adição da função pública `n3f()` e modificações nas funções privativas de criação de objetos, operações sobre polígonos e na estrutura de armazenamentos de dados (*tokens*) originais.

Resultados e discussão

Utilizando a VOGL com as modificações descritas nosso grupo tem conseguido transferir de forma satisfatória programas escritos com a GL para equipamentos que não a possuem.

Da mesma forma esta biblioteca tem permitido o desenvolvimento de programas de visualização em equipamentos de baixo custo que podem ser transferidos diretamente para equipamentos mais sofisticados e equipados com a GL liberando estes últimos para trabalhos mais intensivos.

Para produção da imagem do modelo de um zeólito com 190 átomos na forma de cilindros e esferas em uma estação de trabalho IBM RS6000 modelo 320h foram necessários 60 segundos utilizando *model*=`FLAT` e 90 segundos com *model*=`INTERP`. A mesma imagem consumiu apenas 5 segundos para ser gerada utilizando a GL e hardware adequado na mesma estação. O desempenho observado, embora cerca de 10 a 20 vezes inferior, em média, ao de equipamentos dotados de hardware apropriado, é notavelmente superior ao de métodos mais precisos de geração de imagens (*raytracing, scanline*) permitindo a operação de forma interativa e/ou a geração de sequências de animação com uma qualidade adequada pra reprodução fotográfica.

Conclusões

Através das adições e modificações realizadas a VOGL passa a contar com a funcionalidade mínima necessária para a transferência de vários programas existentes. Ao mesmo tempo estas alterações permitem que novos programas possam ser desenvolvidos em outras plataformas Unix ampliando o número de pesquisadores com acesso a este tipo de recurso gráfico.

Referências

W. Brown, P. Egbert, D. Griffith, K. Sung,*University of Illinois PEX*,University of Illinois,1991

R. Judd, S. R. Nelson, Evans & Sutherland Computer Corporation , *comp.graphics archives*,1988

J. Ng, G. Bourhis, *XDataSlice*, National Center for Supercomputing Applications,1992

Agradecimento

O autor agradece a B. Kirby do grupo de desenvolvimento e manutenção da VOGL por valiosos esclarecimentos e discussões a respeito da estrutura interna desta biblioteca que muito auxiliaram no desenvolvimento das extensões descritas.