Uma Estratégia para Tratamento de Áreas Obscurecidas em Sistemas de Janelas

WALDEMAR CELES FILHO [1]
ENIO EMANUEL RAMOS RUSSO [1,2]
MARCELO GATTASS [1]

[1] Grupo de Tecnologia em Computação Gráfica da PUC-Rio
Departamento de Informática, PUC-Rio
Rua Marquês de São Vicente, 225
22453 Rio de Janeiro, RJ, Brasil
celes@icad.puc-rio.br

[2] PETROBRÁS - Petróleo Brasileiro S.A.
CENPES - Centro de Pesquisas e Desenvolvimento Leopoldo A. Miguez de Mello
Cidade Universitária, quadra 7, sala 9020
21949-900 Rio de Janeiro, RJ, Brasil
seproc@c53000.petrobras.anrj.br

Abstract. Window systems, graphical libraries and application programs share the responsibility to restore exposed areas of canvas in an interactive graphical program. This paper presents a versatile and efficient strategy for restoring obscured areas managed by a window system.

1. Introdução

Atualmente quase todos os sistemas interativos tem uma interface com o usuário baseada em um sistema de janelas, especialmente se eles admitem processos concorrentes. A popularidade desses sistemas aumentou substancialmente após o surgimento dos sistemas de janelas do MacIntosh, o X Window e o Windows da Microsoft.

Apesar desta tendência, o desenvolvimento de sistemas portáteis de interface com o usuário não pode pressupor a existência de um sistema de janelas nativo. Sistemas como o DOS, com ambientes de memória protegida, ainda formam a base de muitas aplicações importantes na área de engenharia. Por isto, novas versões de sistemas importantes, como o *MicroStation* da *Intergraph*, têm o seu própio sistema de janelas.

Esta solução de densenvolver um sistema de janelas próprio também faz sentido no desenvolvimento de ferramentas que promovam a portabilidade de programas. Assim, por exemplo, um sistema de interface com o usuário que pretenda ser portátil deve traduzir a especificação do diálogo feita pelo programador de aplicação para os diversos *toolkits* nativos dos sistemas de janelas existentes e, quando estes não existirem, prover o conjunto de *toolkit* do sistema de janela próprio. Esta decisão foi adotada no desenvolvimento da IUP/LED [Figueiredo et al. (1993)].

A implementação de sistemas de janelas mostra, entretanto, algumas questões ainda não resolvidas. A interação dos sistemas de janelas com os sistemas gráficos que geram os desenhos nos *canvas* é a principal delas. O problema central reside na divisão de responsabilidades, entre a aplicação, o sistema gráfico e o sistema de janelas, de restaurar áreas dos *canvas* previamente obscurecidas por outras janelas.

Este trabalho discute diversas estratégias para o tratamento das superposições de janelas e apresenta uma nova estrutura de dados que oferece suporte a algoritmos eficientes e flexíveis, para as operações de **mostrar** e **esconder** janelas que contenham *canvas*.

2. Estratégias de Tratamento de Áreas Ocultas

Um problema básico a ser tratado por um sistema de janelas é como gerenciar o espaço da tela onde são exibidas as diversas janelas. Existem três estratégias principais para gerenciar a alocação deste espaço. Estas estratégias variam desde deixar toda a responsabilidade de restauração das áreas obscurecidas para a aplicação até aquelas em que esta responsabilidade é do próprio sistema de janelas

A primeira estratégia é implementada nos sistemas de janelas mínimos [Foley et al. (1990)], que não têm nenhuma responsabilidade de restaurar áreas obscurecidas. Nestes sistemas, envia-se um evento de

"janela-exposta" para a aplicação, que, então, se encarrega de atualizar a tela. O sistema de janelas simplesmente efetua o *clipping* das primitivas gráficas contra as demais janelas existentes. Neste contexto, a aplicação tanto pode redesenhar todas as primitivas que formam a imagem quanto ter salvo antes os *pixmaps* das janelas de forma a evitar o esforço de regeração das primitivas. Este parece ser um ponto importante, especialmente em aplicações científicas, onde se tem uma imagem complexa, pois isto exigiria ou um tempo de redesenho muito grande ou a capacidade do sistema gráfico em gerenciar *pixmaps*, o que foge aos padrões usuais, como o GKS e o PHIGS. Esta é a estratégia adotada pelos sistemas comerciais de maior penetração no mercado, como o do MacIntosh, o X Window e o Windows da Microsoft.

A segunda estratégia principal é o sistema de janelas armazenar, para cada janela, um *pixmap off-screen* contendo toda a área da janela. Toda vez que uma parte da janela é reexposta, a subárea apropriada do *pixmap* é copiada para a tela. Esta estratégia exige sistemas com mais memória e pode apresentar ineficiência na geração da imagem, isto porque as primitivas gráficas devem ser desenhadas tanto na área *off-screen* quanto nas áreas visíveis da tela. Existem vários sistemas dedicados de *hardware* que evitam a necessidade de se copiar *pixmap off-screen* para o *pixmap* da tela. De qualquer forma, necessita-se de alta capacidade de memória e de *hardware* específicos, se a queda de desempenho ao se copiar *pixmaps* for significante.

Já a terceira estratégia consiste em armazenar *off-screen* somente as áreas obscurecidas das janelas. Esta estratégia foi inicialmente proposta por Pike [Pike (1983)] e evita duplicidade no armazenamento das informações. Um algoritmo simples apresentado por Flerackers e Van Reeth [Flerackers e Van Reeth (1988)] implementa esta estratégia, subdividindo a tela num *grid* de células, com a restrição de somente permitir o posicionamento e o dimensionamento de janelas neste *grid*.

O algoritmo e a estrutura de dados aqui propostos seguem a linha da terceira estratégia, sem restrições de posicionamento, procurando reduzir ao mínimo o salvamento de áreas (redução da memória necessária) e o esforço de restauração de áreas obscurecidas, dentro da filosofia de que é responsabilidade do sistema de janelas redesenhar novas partes expostas de uma janela.

3. Algoritmo Proposto

O sistema de janelas deve ser capaz de suportar operações sobre as janelas gerenciadas. Identificam-se quatro operações básicas: mostrar, esconder, trazer para frente e levar para trás. A operação de **mostrar** é executada quando uma janela, inicialmente não representada graficamente, é exposta na tela. A operação **esconder** é oposta, elimina a representação gráfica de uma janela. As outras duas operações trocam a prioridade de exposição das janelas. Como as janelas podem ser superpostas, existe uma ordem de prioridade entre elas. Inicialmente, a ordem de prioridade corresponde à ordem em que as janelas são mostradas. A operação de **trazer para frente** coloca uma determinada janela no topo da pilha, diz-se que ela tem prioridade máxima. A operação inversa, **levar para trás**, dispõe a janela na base da pilha, com prioridade mínima.

As demais operações, tais como mover, maximizar e iconizar, podem ser realizadas através de uma composição das quatro operações básicas. Portanto, é necessário que estas operações sejam realizadas eficientemente, de forma a possibilitar a obtenção de um ambiente interativo eficaz.

O problema principal que está por detrás destas operações é permitir a manipulação das janelas com um eficiente mecanismo para restauração da imagem na tela. A estrutura proposta independe da forma em que será realizada a restauração, ou através de eventos para redesenhar, ou através de reexposição de imagens armazenadas em memória. Basicamente, o sistema identificará que regiões devem ser atualizadas, e a forma de atualização dependerá da forma adotada. Se for adotado o armazenamento das imagens obscurecidas, deve-se prover mecanismos para subdividir uma imagem já armazenada.

Identifica-se, basicamente, duas estratégias para o gerenciamento das janelas, tendo em vista o suporte às operações básicas de manipulação das mesmas. Na primeira, cada janela guarda informações da área que ela obscurece. Esta estratégia favorece a operação **levar para trás**, já que é fácil restaurar as áreas obscurecidas pela janela, pois todas as informações necessárias encontram-sa na estrutura da janela. A segunda estratégia armazena em cada janela informações a respeito das áreas obscurecidas da própria janela. Com isto, é fácil realizar uma operação do tipo **trazer para frente**, pois uma janela "sabe" se mostrar (ela contém todas as informações necessárias para se restaurar). Portanto, ambas as estratégias têm pontos positivos e negativos. A solução então passa por uma combinação das duas formas, sem que com isto haja duplicidade de informações.

A estrutura proposta dispõe as janelas numa lista encadeada. A ordem das janelas nesta lista retrata as suas prioridades. Assim, uma janela no início da lista aparece com prioridade mínima e então pode ser superposta pelas demais. A janela no final da lista, por

apresentar prioridade máxima, obrigatoriamente está totalmente exposta, sem áreas obscurecidas. Com isto, redesenhando todas as janelas na ordem em que aparecem na lista, obtém-se a imagem correta, pois as janelas de menor prioridade serão obscurecidas pelas de maior prioridade, se elas se superpuserem.

Além disso, cada janela da lista contém uma lista das áreas obscurecidas por outras janelas. Assim, supondo a ocorrência da Figura 1, onde a janela A é parcialmente obscurecida pela janela B (com maior prioridade que A), obtém-se a estrutura de listas ilustrada na figura.

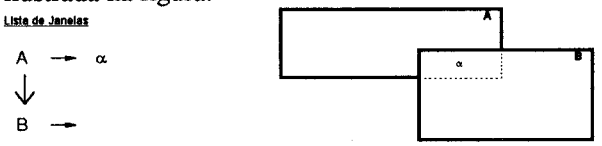

Figura 1: superposição simples

Nesta figura, α representa a área de A obscurecida pela janela B. Assim, a estrutura implementa uma simples estratégia onde cada janela armazena a lista de áreas obscurecidas por outras janelas. Conforme discutido, esta estrutura é insuficiente para a realização de operações **levar para trás** de maneira eficiente, já que somente a operação de **trazer para frente** está sendo adequadamente suportada. Na verdade, a operação **levar para trás** pode ser substituída por uma seqüência de operações **trazer para frente**. Isto é, para minimizar a prioridade de uma dada janela, bastaria maximizar a prioridade de cada uma das outras janelas, respeitando a ordem de prioridade já existente. No entanto, seria uma solução ineficiente e produziria um aspecto desagradável com as sucessivas exposições de janelas na tela.

Propõe-se então que a cada área obscurecida adicione-se uma lista das janelas que obscurecem aquela área. A Figura 2 mostra como dispor-se-ia a estrutura com a introdução de uma nova janela C, com prioridade superior às demais.

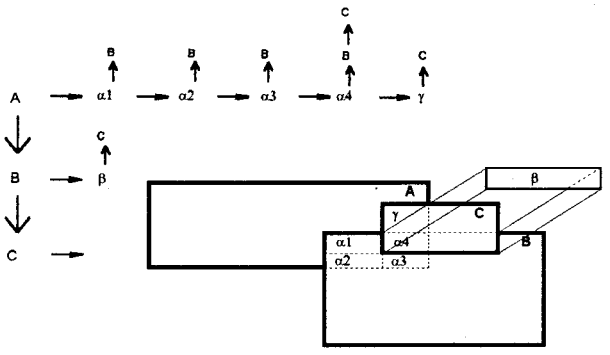

Figura 2: efeito de mostrar uma nova janela

Com a introdução da janela C, parte da área α, que era obscurecida por B, passa a ser também obscurecida por C. Com isto, há necessidade de se subdividir a área α em subáreas, tal que cada subárea tenha sua própria lista das janelas que a obscurecem. Além da subdivisão da área α, a janela C criou uma área γ em A e outra β em B. Observa-se que a área β superpõe-se à área α4, no entanto isto não representa duplicidade de informações, já que na área α4 armazenam-se informações que dizem respeito à janela A e em β armazenam-se informações da janela B.

Analisam-se, a seguir, as operações básicas e os procedimentos associados a cada uma delas. Deve-se ter em mente que, como se trata de operações que envolvem atualização de imagens, se tem como objetivo principal realizar o mínimo de acesso às rotinas gráficas, mesmo que seja necessário efetuar cálculos geométricos. Isto porque os cálculos envolvidos referem-se apenas à interferência entre retângulos, que pode ser eficientemente implementada através da *shape algebra* [Steinthart (1991)].

A operação **mostrar** já foi exemplificada com as ilustrações mostradas acima. Para cada janela que se mostra, deve-se checar a interferência da sua área com as demais janelas e com as áreas obscurecidas dessas janelas. A checagem de interferência, conforme observado, pode acarretar em subdivisões de áreas obscurecidas já existentes e criação de novas áreas obscurecidas.

A operação **trazer para frente** é análoga. Para que uma janela parcialmente obscurecida se torne totalmente visível, basta que sejam restauradas todas as áreas obscurecidas da janela. Portanto, para realizar a operação, deve-se checar a interferência de cada uma das áreas obscurecidas com as demais janelas. Deve-se também reposicionar a janela no final da lista, pois ela passa a ter prioridade máxima. E por fim, libera-se a lista de áreas, já que a janela passa a ser totalmente exposta.

A operação **levar para trás** merece uma análise mais detalhada. Sabe-se que uma janela com prioridade mínima não pode obscurecer nenhuma outra. Ela apenas pode conter áreas próprias obscurecidas por outras janelas. Portanto, nesta operação deve-se eliminar todas as possíveis áreas obscurecidas pela janela em questão nas listas das outras janelas (sabendo-se que uma janela só pode ter área obscurecida por outra de maior prioridade).

Para facilitar a explanação, considera-se aplicar a operação **levar para trás** sobre uma janela W qualquer. O algoritmo que realiza esta tarefa percorre a lista de janelas, em ordem crescente de prioridade, até alcançar a janela W. Para cada janela, verifica-se a existência de áreas obscurecidas por W. A cada área obscurecida

detetada, deve-se retirar a janela W da lista de janelas da área e checar a interferência desta área sobre esta mesma janela W. Assim, inverte-se a situação e é a janela corrente que passa a obscurecer a janela W. Este cálculo de interferência pode alterar a lista de áreas obscurecidas de W, subdividindo áreas anteriormente existentes e criando novas áreas.

Ainda, para cada área que era obscurecida por W, deve-se verificar se W era a única janela que a obscurecia, pois, neste caso, deve-se restaurar a imagem desta área e, então, retirá-la da lista de áreas da respectiva janela. Ao final do processo, reposiciona-se a janela W no início da lista de janelas, com prioridade mínima.

A operação **esconder** é quase uma repetição do procedimento explicado acima. É necessário apenas limpar as áreas visíveis da janela antes de escondê-la. Outro aspecto é que deixa de ser necessário realizar os cálculos de interferência, já que a representação gráfica da janela desaparecerá.

Pike [Pike (1983)] propõe uma estrutura para gerenciamento de janelas com muitos pontos em comum com a apresentada aqui. Basicamente, identifica-se duas diferenças. A primeira é que a estrutura proposta por Pike armazena para cada área obscurecida apenas a janela de maior prioridade que está obscurecendo a área. Com isto, a operação **levar para trás** só pode ser realizada através de uma combinação de operações **trazer para frente**. E isto, como anteriormente mencionado, torna-se ineficiente do ponto de vista computacional, resultando numa seqüência de exposições de imagens que poderiam ser evitadas. A introdução das listas de janelas associadas a cada área obscurecida possibilita a realização da operação **levar para trás** com o mínimo de atualização gráfica na tela.

Segundo, Pike cria todas as subdivisões das áreas obscurecidas quando uma determinada janela é mostrada. Com isto, no exemplo ilustrado anteriormente, quando a janela C fosse mostrada, a área β seria subdividida em duas, já que parte dela tem interseção com a janela A. A estrutura aqui proposta procura minimizar o número de subdivisões, mantendo sempre que possível as áreas contíguas, sem subdividi-las. No exemplo ilustrado, a área β só necessita ser subdividida se a janela A passar a ter prioridade maior que a janela B, e, somente se isto ocorrer, a subdivisão será feita.

Por fim, salienta-se que, após diversas operações de manipulação das janelas, as áreas obscurecidas ficam muito subdivididas. Portanto, pode ser mais eficiente implementar um algoritmo adicional que realize a união de retângulos sempre que possível, lembrando que somente retângulos com a mesma lista de janelas podem ser unidos. Com isto, é importante manter uma ordem na disposição das janelas nas listas das áreas (por exemplo, armazená-las sempre em ordem crescente de prioridade).

4. Conclusões

A flexibilidade de deixar a cargo da aplicação restaurar as áreas obscurecidas ou a cargo do próprio sistema de janelas parece ser a estratégia mais adequada, especialmente em aplicações que geram imagens complexas. Dentro deste contexto, propõe-se uma estrutura que permite o gerenciamento das áreas obscurecidas de um sistema de janelas.

A estrutura apresentada suporta as operações básicas para manipulação de janelas. A restauração das áreas obscurecidas é feita com o mínimo de atualização gráfica.

Agradecimentos

A motivação deste estudo teve origem no âmbito do convênio TeCGraf-PETROBRÁS/CENPES. Agradecemos a Carlos Henrique Levy pelas discussões e implementações, que muito contribuíram para o desenvolvimento deste trabalho. Agradecemos também à CAPES, pelo apoio financeiro ao programa de doutoramento do primeiro autor.

Referências

R. Pike, Graphics in Overlapping Bitmap Layers, ACM Transactions on Graphics, Vol. 2, No. 2 (1983), 135-160.

E. Flerackers and F. Van Reeth, Algorithms and Data Structures for Windowing on Raster Graphics Devices, NATO ASI series, Vol. F40, 1099-1107, Springer-Verlag Berlim Heidelberg, 1988.

J. D. Foley, A. Van Dam, S. K. Feiner and J. F. Hughes, Computer Graphics: Principles and Practices, Addison-Wesley, Reading, 1990.

J. E. Steinhart, Scanline Coherent Shape Algebra, Graphics Gems II, Academic Press, Inc., Boston, 1991.

L. H. de Figueiredo, M. Gattass e C. H. Levy, Estratégias de portabilidade para aplicações gráficas interativas, artigo submetido para publicação no SIBGRAPI (1993).