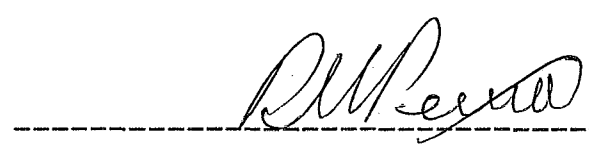
UM ALGORITMO INCREMENTAL PARA REMOÇÃO DE SUPERFÍCIES OCULTAS

Bruno Carlos de Medeiros Câmara

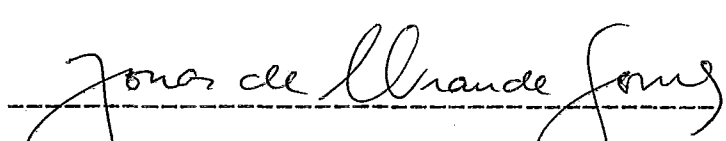
TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por :



Prof. Ronaldo C. M. Persiano, D.Sc.
(Presidente)

Prof. Marcelo Gattass , Ph.D.



Prof. Jonas de Miranda Gomes, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
JUNHO DE 1989

CÂMARA, BRUNO CARLOS DE MEDEIROS

Um algoritmo incremental para remoção de superfícies ocultas [Rio de Janeiro] 1989

VIII, 71 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1989)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Computação Gráfica I. COPPE/UFRJ II. Título (série)

AGRADECIMENTOS

- Ao Professor Ronaldo Cesar Marinho Persiano, por sua valiosa orientação.

- Aos colegas do Laboratório de Computação Gráfica da COPPE/UFRJ, pelo auxílio no decorrer do trabalho, principalmente na fase de implementação.

- Aos familiares e amigos que me apoiaram no decorrer do trabalho, em especial à amiga Patrícia pela cuidadosa elaboração das figuras.

- Ao CNPQ pelo apoio financeiro.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências (M. Sc.).

UM ALGORITMO INCREMENTAL PARA REMOÇÃO DE SUPERFÍCIES OCULTAS

Bruno Carlos de Medeiros Câmara

Junho de 1989

Orientador: Ronaldo Cesar Marinho Persiano
Programa: Engenharia de Sistemas e Computação

Grande parte das aplicações de computação gráfica requerem a visualização de uma cena tridimensional. Para se obter um realismo maior nessa visualização pode-se simular a opacidade dos objetos que compõem a cena, eliminando as superfícies que forem ocultas por outras superfícies da cena. A essa técnica se denomina remoção de superfícies ocultas. Em geral, os algoritmos que tratam essa técnica tem que atualizar toda uma cena já construída, no caso de uma inserção ou remoção de um objeto que a componha. O presente trabalho propõe um algoritmo de remoção de superfícies ocultas, cuja principal característica é o fato de ser incremental. Por incremental entende-se que o algoritmo trata a inserção e remoção de objetos da cena, atualizando a imagem apenas nas regiões em que ela deve sofrer alterações. Esse algoritmo foi implementado de três formas diferentes usando a linguagem Pascal.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M. Sc.).

AN INCREMENTAL HIDDEN-SURFACE REMOVAL ALGORITHM

Bruno Carlos de Medeiros Câmara

June, 1989

Thesis Supervisor: Ronaldo Cesar Marinho Persiano

Department: Systems and Computer Engineering

Many computer graphic applications require the visualization of a tridimensional scene. In order to improve the realism in these visualizations, the opacity of the objects that belong to the scene can be simulated by removing the surfaces that are hidden by other surfaces of the scene. This technique is called hidden surface elimination. Usually, the algorithms that deal with this technique have to update a whole scene already built, when an insertion or removal of an object from this scene occurs. This paper proposes a hidden surface elimination algorithm, whose main characacteristic is the fact of being incremental. As incremental, it can be understood that the algorithm deals with insertions and removals of objects from a scene by updating the image only on the regions which it must be changed. This algorithm was implemented in three different ways, using Pascal language.

ÍNDICE

página

CAPÍTULO I

Representação 3D

Muitas aplicações de computação gráfica envolvem a exibição de imagens tridimensionais. Essas aplicações diferem das aplicações bidimensionais não apenas pela inclusão da terceira coordenada, mas também pelo fato de requererem uma preocupação com o realismo das imagens exibidas. Essa preocupação pode ser traduzida , por exemplo, em como exibir a terceira coordenada na tela bidimensional, ou como eliminar partes da cena escondidas por outras partes, ou ainda como incluir na imagem efeitos como iluminação, cor, sombreamento, textura, etc. A seguir será apresentada uma rápida descrição de algumas dessas técnicas.

I.1- Descrição de técnicas para se obter realismo

As projeções são definidas como uma transformação de pontos de um sistema de dimensão n para um sistema de dimensão menor que n. Aqui serão analisadas as projeções planares de um sistema tridimensional para um sistema bidimensional. Assim a projeção de um objeto tridimensional pode ser definida por raios de projeção retilíneos, os projetores que passando por cada ponto do objeto interceptam o plano de projeção para formar a projeção propriamente dita. Esse tipo de projeção é conhecida como projeção geométrica planar, pois ela é feita num plano, utilizando raios projetores retilíneos e não numa superfície curvilínea ou usando raios projetores curvos. Os diferentes tipos

de projeções planares estão discutidos e ilustrados em CARLBOM et alii [1].

As projeções planares podem ser divididas em duas classes : paralelas e perspectivas. As paralelas se caraterizam pelo fato dos raios projetores serem paralelos a uma dada direção, denominada direção de projeção. Já as perspectivas se caracterizam pelo fato dos raios projetores se cruzarem num ponto, denominado centro de projeção. A figura 1.1 ilustra a diferença entre os dois tipos de projeção, onde CP é o centro de projeção, DP a direção de projeção, PP o plano de projeção e p e q os pontos a serem projetados, e p' e q' os pontos projetados.

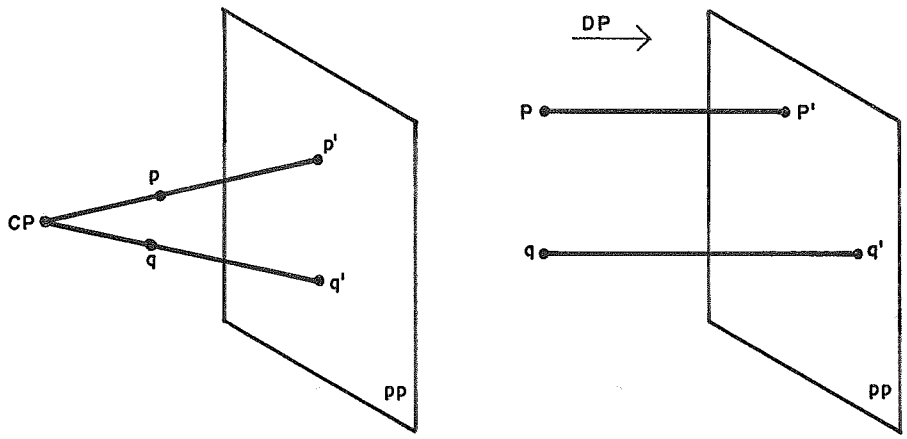


figura 1.1

As projeções paralelas podem ser classificadas como ortogonais ou oblíquas, conforme a direção de projeção e a normal ao plano de projeção sejam paralelas ou não. Algumas das projecões paralelas mais comuns são analisadas a seguir.

Quando o plano de projeção é paralelo a uma face do objeto a ser projetado as arestas paralelas a este plano terão suas dimensões preservadas, enquanto que as paralelas à direção de projeção serão suprimidas. Assim devido a dificuldade de se visualizar o objeto é comum ser apresentada mais de uma projeção desse tipo simultaneamente, com os planos de projeção sendo paralelos a faces diferentes. Essa projeção denomina-se projeção ortogonal multivista. A figura 1.2 apresenta esse tipo de projeção para um cubo com um prisma sobre ele.

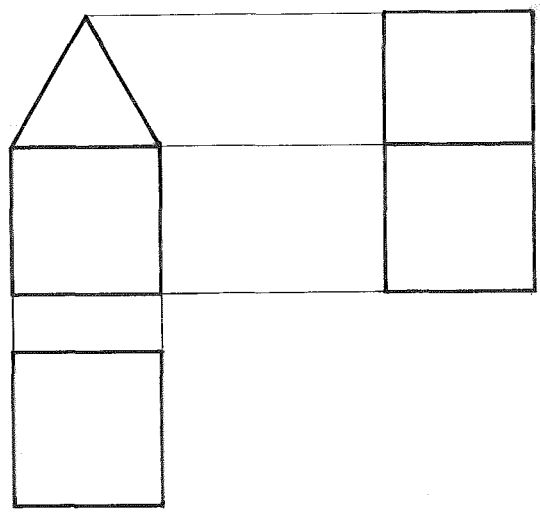


figura 1.2

A projeção ortogonal axonométrica usa um plano de projeção não paralelo a nenhuma das faces do objeto a ser projetado. Assim é capaz de exibir várias faces do objeto simultaneamente, contudo as dimensões e os ângulos entre arestas não são, em geral, preservados. A figura 1.3 apresenta esse tipo de projeção para um prisma.

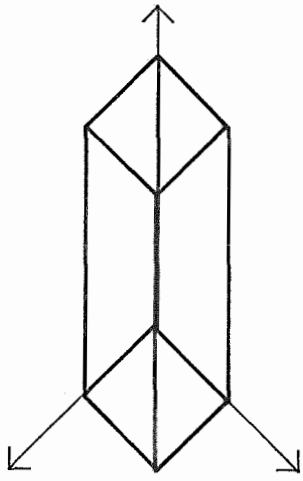


figura I.3

Pode-se ainda conservar o plano de projeção paralelo a uma face, porém tornando a direção de projeção oblíqua a esse plano. Esse é um tipo de projeção paralela oblíqua é exemplificado na figura I.4.

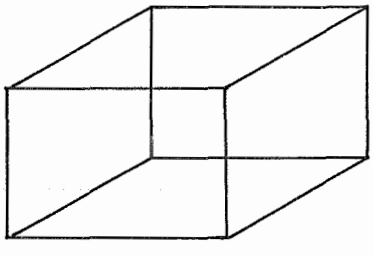


figura I.4

As projeções em perspectiva, ou simplesmente perspecti-

vas, tentam criar o efeito visual caracterizado pelo fato de que as dimensões relativas de um objeto aparentemente diminuem com o aumento da distância deste em relação ao observador. Esse efeito faz com que as perspectivas ofereçam um maior realismo que as projeções paralelas.

Uma propriedade característica das representações em perspectiva é que retas originalmente paralelas terão suas projeções convergindo para um dado ponto, denominado ponto de fuga. Se o plano de projeção for paralelo a duas das três direções principais do objeto, a projeção conterá apenas um ponto de fuga. Porém se esse plano for paralelo a apenas uma direção principal do objeto, a projeção conterá dois pontos de fuga. A figura I.5 apresenta a diferença entre esses dois casos, onde PF1 e PF2 são pontos de fuga.

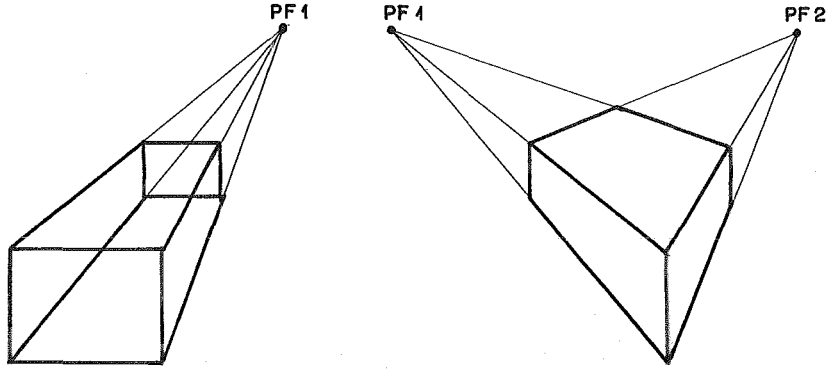


figura I.5

O procedimento para execução de uma projeção ou perspectiva consiste basicamente em se transformar o sistema coordenado original em um novo sistema, de acordo com a transformação desejada. A seguir a terceira coordenada de cada ponto é suprimida para que possa ser feita a exibição. Uma descrição da transformação do sistema coordenado se

encontra em PERSIANO et alii [2].

A eliminação de linhas ou superfícies ocultas de um objeto posicionado espacialmente atrás de outro objeto ou de faces do próprio objeto em relação a um dado observador garante um realismo muito maior, simulando uma opacidade, como pode ser observado na figura I.6. O processo de impedir a exibição dessas linhas e superfícies é conhecido como eliminação de linhas ou superfícies ocultas. Esse assunto será tratado mais extensivamente a partir do capítulo II.

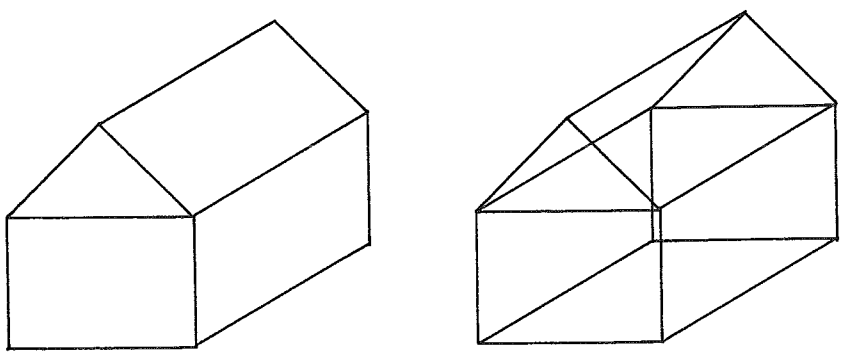


figura I.6

Outras técnicas podem acrescentar um realismo muito maior à imagem tais como : iluminação, onde são exibidas as variações da intensidade da cor na face de um objeto devido às propriedades da luz e do material do objeto; sombreamento, onde as sombras geradas pela posição do objeto e das fontes de luz são exibidas; textura, onde é tentado simular o material que compõe o objeto, a partir de suas propriedades luminosas. Contudo, elas requerem, em geral, um processamento muito maior. Indicações e referências dessas e

outras técnicas podem ser encontradas em NEWMAN et alii [3].

O presente trabalho propõe-se a apresentar uma técnica para a solução do problema de eliminação de superfícies ocultas. No capítulo II serão abordados os métodos mais conhecidos para resolução desse problema e como o método proposto se enquadra entre eles. No capítulo III será discutido o método desenvolvido neste trabalho. No capítulo IV será apresentada uma implementação em computador do método discutido no capítulo anterior. Finalmente no capítulo V serão tecidos alguns comentários a cerca do método proposto e da implementação realizada.

I.2- Sequência de processos

A sequência normalmente utilizada dos processos ou técnicas descritos anteriormente é a mostrada na figura I.7. Inicialmente os objetos fornecidos recebem uma transformação, como rotação ou translação, para serem posicionados na posição desejada no espaço. A seguir há normalmente um recorte, onde as porções da cena fora da área de exibição são eliminadas. A seguir a cena sofre a transformação de projeção, ou uma projeção paralela ou uma perspectiva, sem a eliminação da terceira coordenada. A etapa seguinte envolve a eliminação de superfícies ocultas e outros processos mais elaborados, como iluminação e sombreamento. A cena resultante ainda está em três dimensões. Por fim é feita a projeção, propriamnete dita, na tela de exibição, onde a coordenada z é simplesmente eliminada da cena resultante.

Essa coordenada não foi eliminada na etapa de projeção, como descrito na seção 1.1, pois ela foi necessária nas etapas posteriores. É comum que alguns desses processos se encontrem agrupados em um único.

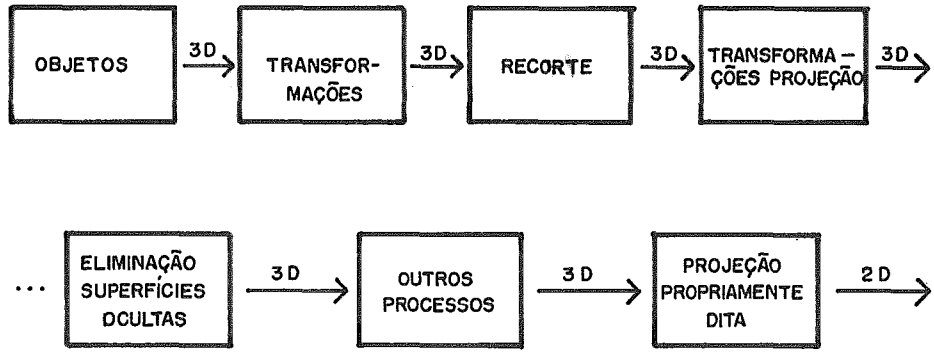


figura 1.7

CAPÍTULO II

Algoritmos para eliminação de superfícies ocultas

A necessidade da eliminação de linhas ou superfícies ocultas ocorre porque o material dos objetos se opaco obstrue os raios de luz das partes que estarão ocultas, e essa eliminação obviamente não ocorre automaticamente na geração de uma imagem em computador. Assim é necessário que haja um processamento para que essas linhas ou superfícies sejam eliminadas quando da construção da imagem. A seguir serão apresentados os principais tipos de algoritmos e suas características e como o algoritmo proposto se enquadra entre eles. Uma análise completa dos diversos tipos de algoritmos para eliminação de linhas ou superfícies ocultas, com comparações e referências encontra-se em SUTHERLAND et alii [4].

Apesar de existirem diversos algoritmos para resolução desse problema, todos eles fazem uso de alguma forma de ordenação geométrica, porque afinal o que se quer é que somente as superfícies mais próximas do observador sejam exibidas, e por isso elas necessitam ser ordenadas. Uma ordenação geométrica é muito mais complexa que uma ordenação linear, pois os objetos geométricos não estão sempre necessariamente totalmente a frente de outros. Isso significa que a ordenação geométrica não é completa, ao contrário da ordenação linear. Além disso os algoritmos para eliminação de superfícies ocultas, em geral, fazem uso de algum tipo de coerência ou similaridade com o propósito de reduzir o processamento necessário para gerar a imagem. As

diferentes formas de similaridade estão relacionadas com as diferentes formas de ordenação ou regularidade da imagem.

II.1- Tipos de algoritmos

Os algoritmos para eliminação de linhas ou superfícies ocultas podem trabalhar no espaço de objetos ou no espaço da imagem. Os algoritmos que trabalham no espaço de objetos exploram as relações geométricas que existem entre os objetos para determinar quais as partes visíveis de cada um. Já os algoritmos que trabalham no espaço da imagem determinam para cada pixel da tela qual o objeto que deve fornecer seus atributos. Um algoritmo que trabalhe no espaço de objetos processa os cálculos necessários numa precisão arbitrária, geralmente a máxima disponível no computador que executa o algoritmo, pois o objetivo do algoritmo é determinar "exatamente" como a imagem será. Já os algoritmos trabalhando no espaço da imagem processam os cálculos necessários com uma precisão menor, normalmente apenas a precisão da tela de exibição. Em outras palavras, os algoritmos que operam no espaço de objetos determinam se cada item potencialmente visível no espaço é realmente visível pelo observador; enquanto os algoritmos no espaço da imagem determinam o que é visível em cada pixel da tela. Essa diferença de atitude leva a uma diferença de desempenho dos algoritmos : o tempo de processamento de um algoritmo no espaço de objetos cresce com o número de objetos a serem processados, porém o tempo de processamento de um algoritmo que opera no espaço da imagem é mais limitado, devido à

limitação do número de pixels da tela de exibição. Assim, em geral, o custo computacional de uma algoritmo no espaço da imagem cresce mais lentamente com a complexidade da cena que um algoritmo no espaço de objetos. Algoritmos trabalhando no espaço de objetos são usados, em geral, para eliminação de linhas ocultas, enquanto aqueles operando no espaço da imagem são usados,em geral, para eliminação de superfícies ocultas.

O algoritmo mais simples que trabalha no espaço da imagem é o chamado algoritmo de memória com profundidade ou "z-buffer". Esse algoritmo utiliza uma matriz, o "z-buffer", em que cada posição corresponderá a um pixel da tela. Essa matriz conterá para cada ponto a profundidade e a cor do objeto que naquele ponto está mais próximo do observador. Para a execução do algoritmo deve ser calculada a profundidade de cada ponto do polígono que se projeta em um ponto da tela, logo esse ponto representa uma posição na matriz. A profundidade do polígono no ponto deve ser comparada então com a profundidade armazenada na matriz na posição correspondente. Nos pontos em que a profundidade do novo polígono for menor que a armazenada na matriz, sua cor pinta o pixel correspondente e sua profundidade passa a ocupar essa posição na matriz. O grande problema desse algoritmo é a enorme quantidade de memória requerida, pois é necessária uma posição para cada ponto da tela. Uma solução alternativa seria dividir a área da tela em porções menores e aplicar o algoritmo em cada uma delas, isso reduziria a memória necessária, porém faria o tempo de processamento crescer. O algoritmo poderia ser executado, por exemplo, para cada linha da tela.

Existem ainda dois grupos de algoritmos trabalhando no espaço da imagem : aqueles que fazem a varredura por áreas e aqueles que fazem a varredura por linha. O objetivo dos algoritmos de varredura por área é dividir a tela em áreas disjuntas em que a análise passe a ser feita isoladamente para cada uma. Assim o problema pode ser dividido em problemas menores, cuja solução é mais simples pois a análise pode-se ater aos polígonos que se projetam na área considerada. Esse tipo de filosofia foi proposto originalmente por WARNOCK [5] que propôs que se a área da tela a ser analisada não for simples o suficiente para ser resolvida de uma só vez, ela deve ser particionada e a análise repetida para cada uma das áreas resultantes da partição. Esse procedimento recursivo só é encerrado quando é alcançada uma área simples o suficiente para ser resolvida ou se a área resultante da partição corresponder a um único pixel da tela.

Os algoritmos de varredura por linha ou "scan-line" resolvem o problema das superfícies ocultas em uma linha de varredura por vez, geralmente processando essas linhas de cima para baixo na tela. O algoritmo computa a interseção do plano horizontal que contem a linha de varredura com cada face dos objetos da cena. Essas interseções serão segmentos de reta num plano xz. Assim o problema tridimensional é reduzido em cada linha de varredura para um problema bidimensional. Para se reduzir o procesamento pode-se fazer uso de similaridade para, por exemplo, facilitar a determinação das interseções das faces com cada plano que contém a linha de varredura, sabendo-se que o conjunto de faces do polígono interceptadas terá uma pequena alteração

quando se passa de uma linha de varredura para a linha seguinte. Em WATKINS [6] é apresentado o método "scan-line" tradicional.

II.2- A motivação para o algoritmo incremental

Existem certas situações de geração de imagens como animação ou em sistemas interativos, em que entre uma etapa e outra se observam alterações em poucas regiões da cena. Num processo de animação com observador fixo alguns objetos da cena se movem, outros ficam estáticos. A imagem desses objetos que se movem deve ser removida da antiga posição e inserida na nova, entre uma cena e outra. Em um sistema interativo alguns objetos da cena podem sofrer alguma transformação ou serem inseridos em uma nova posição, fazendo com que seja necessário remover sua imagem anterior e criar a nova imagem de acordo com a nova situação. Enquanto isso, outros objetos da cena podem não sofrer alteração alguma quanto a sua posição.

Nos algoritmos tradicionais para resolução do problema de superfícies ocultas, à exceção do método do "z-buffer", não é possível fazer uma alteração em apenas uma parte da cena sem que se tenha de processar todo o resto dela. Assim numa situação como a de animação, por exemplo, onde um objeto deve ser removido de uma posição e inserido em outra, a cena inteira teria que ser reprocessada, incluindo as regiões onde não há nenhuma possibilidade de alteração na imagem. A exceção é o método do "z-buffer", pois se for mantida a matriz de profundidade, qualquer novo polígono ao ser inserido causará um processamento apenas nas regiões

onde a imagem da cena pode possivelmente ser alterada. Contudo, o método do "z-buffer" não permite remoções, pois a matriz de profundidade armazena apenas uma profundidade em cada ponto, não armazenando a cor e a profundidade de polígonos que também possuem projeção naquele ponto e que estejam àquela altura atrás do polígono cuja imagem está sendo exibida.

No algoritmo proposto procurou-se possibilitar a inserção e remoção incrementais, isto é, quando um ou mais objetos são inseridos e removidos da cena, esta é reprocessada somente nas regiões em que há possibilidade de alteração da imagem. A metodologia se baseia na idéia do algoritmo "scan-line" tradicional, pois este é, dentre os métodos tradicionais, aquele cuja filosofia se encaixa mais facilmente num procedimento incremental.

CAPÍTULO III

Descrição do algoritmo

III.1- Introdução

O algoritmo aqui proposto não vai utilizar nenhum modelo topológico especial dos objetos da cena. Ele assume que os objetos (ou partes) da cena nada mais são que conjuntos de polígonos. A vantagem disso está no fato de que o algoritmo pode, em princípio, ser utilizado em qualquer modelagem topológica. Contudo, uma modelagem topológica mais elaborada como a descrita em HORNUNG [7], poderia trazer resultados mais eficientes num algoritmo de eliminação de superfícies ocultas.

A técnica a ser utilizada no algoritmo é a "scan-line", que é caracterizada por transpor para cada linha horizontal do plano de projeção a análise de quais regiões estão à frente de outras. Esse tipo de análise é feita para toda a cena, partindo do seu extremo superior para o inferior ou vice-versa. Já num algoritmo "scan-line" incremental a análise precisa ser feita apenas entre o extremo superior e o inferior do objeto que entra ou sai da cena, já que somente nesse espaço pode haver alguma alteração na cena.

A partir de agora será adotado um sistema de coordenadas, em que a tela de exibição corresponde ao plano xy. A origem corresponde ao extremo inferior esquerdo, com o eixo y crescendo para cima e o x para a direita. O eixo z cresce no sentido em que se afasta do observador.

Qualquer algoritmo do tipo "scan-line" pode ser divi-

dido em três partes básicas : a primeira consiste na seleção de quais polígonos da cena tem a sua projeção no plano de projeção (plano xy) interceptando a linha deste plano que está sendo avaliada. Após isso deve-se determinar quais as porções desses polígonos que se projetam na linha avaliada, essas porções serão sempre segmentos de reta em um plano y=cte, correspondente à linha avaliada, obtidos pela intereseção dos polígonos com esse plano. Finalmente esses segmentos de reta deverão ser ordenados por profundidade para se descobrir quais estão à frente e consequentemente serão pintados na linha de projeção.

A análise para cada linha do plano de projeção é feita comparando-se as interseções dos polígonos com o plano y=cte que contem essa linha. Essas interseções serão segmentos de reta que cobrirão o espaço entre uma aresta e a aresta seguinte interceptada pelo mesmo plano y=cte. Portanto como esses segmentos estão sempre compreendidos entre as interseções do plano com duas arestas, estas devem ser tratadas aos pares. Como o algoritmo é incremental a cada inserção ou retirada a análise pode ter que ser repetida para um certo plano y=cte, não sendo conveniente determinar a cada vez qual aresta faz par com outra. Assim essa informação deve ser armazenada, e uma maneira de se fazer isso é tratar os polígonos como um conjunto de trapézios com lados paralelos ao eixo x do sistema coordenado da tela de exibição. Dessa maneira os pares de arestas que determinam uma interseção do polígono com o plano y=cte ficam armazenados em cada trapézio. De agora em diante, portanto, não se tratará mais de polígonos gerais, mas sim de trapézios, já

que os polígonos serão todos subdivididos em trapézios quando inseridos na cena, sendo também armazenados dessa forma. Denomina-se segmento à interseção de um trapézio com um determinado plano y=cte.

A primeira restrição que se faz é que os polígonos sejam formados por trapézios planos, já que, se isso não ocorrer, não há como determinar-se qual a superfície gerada pelos vértices do trapézio.

A análise num algoritmo incremental em cada linha é feita baseando-se em que sua imagem não será alterada se um segmento de um trapézio da cena estiver localizado à frente (mais próximo do observador) dos segmentos dos trapézios inseridos ou removidos. Contudo se esse segmento estiver atrás, ela poderá sofrer alguma alteração. Dessa maneira, a imagem dos segmentos dos trapézios da cena no plano de projeção só será alterada se esses segmentos estiverem atrás dos segmentos dos polígonos que entram ou saem da cena. Portanto, a cada linha do plano de projeção os segmentos da cena, isto é, provenientes dos trapézios da cena, deverão ser comparados com os segmentos de referência, isto é, aqueles provenientes dos trapézios que estão entrando ou saindo da cena. Na figura III.1, a imagem do polígono 1 na linha correspondente ao plano analisado será alterada com a retirada ou inserção do polígono 2, pois este está mais próximo do observador que o polígono 1. Já o polígono 3 está mais próximo do observador que o 2, portanto sua imagem na linha analisada não será alterada com a inserção ou retirada deste.

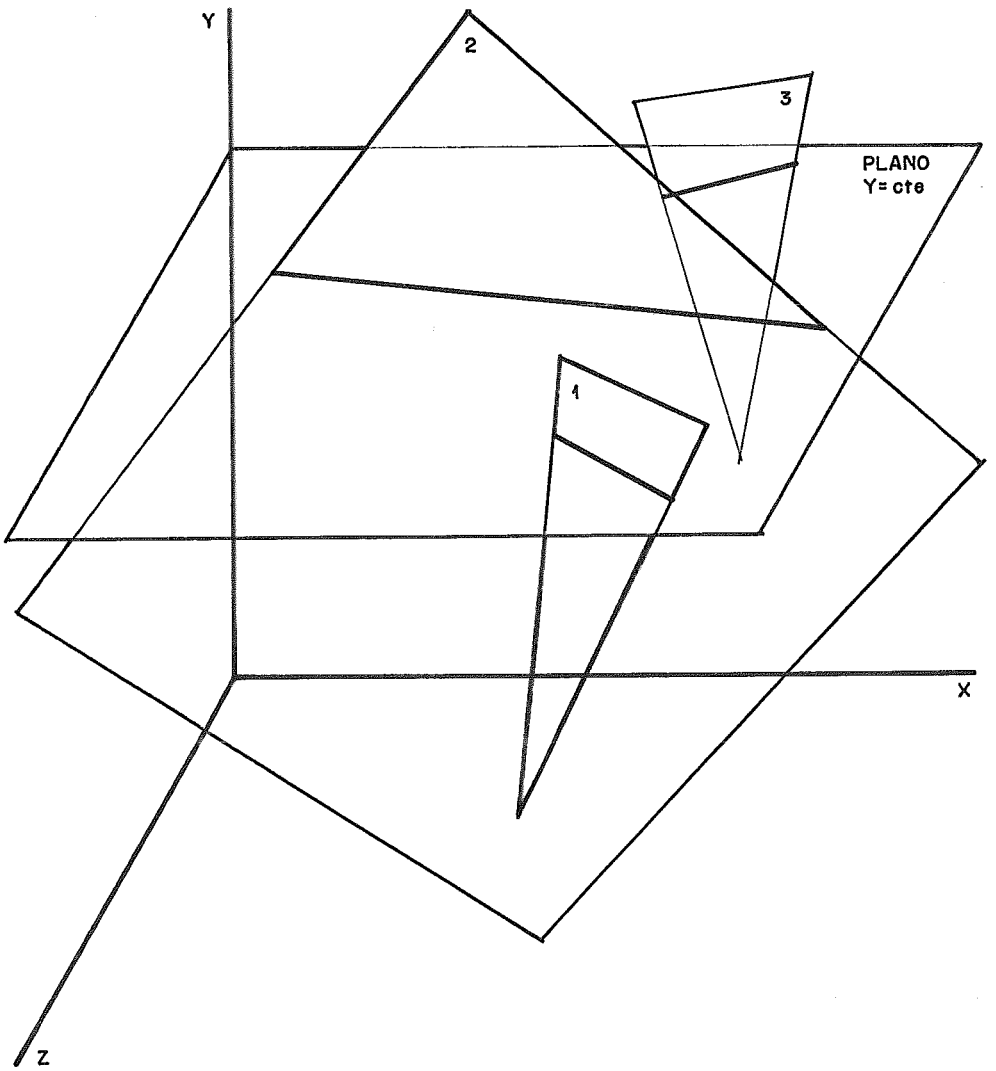


figura III.1

Para que não se tenha que testar cada segmento da cena contra cada segmento de referência, eles devem ser organizados em estruturas em que somente os segmentos mais à frente sejam considerados. Essa estrutura deverá informar em cada ponto, a coordenada do segmento de referência ou da cena, conforme o caso, que estiver mais à frente, já que esse é o único relevante no processo. Como foi mencionado anteriormente, os segmentos são interseções de trapézios com o plano y=cte, assim essa estrutura poderá ser caracte-

rizada como uma função, denominada função de profundidade, que associará a cada x, o valor da coordenada z do segmento mais próximo do observador. Serão geradas duas funções de profundidade : uma para os segmentos de referência, outra para os segmentos da cena. Ao final, quando todos os segmentos de referência já estiverem organizados na função de profundidade dos segmentos de referência e os segmentos da cena na função dos segmentos da cena, essas duas funções podem ser comparadas. Nos pontos em que a função de profundidade da cena tiver um valor maior ou igual ao da função de profundidade de referência haverá uma alteração da imagem. Essa alteração no caso da remoção será uma pintura, isto é, estará sendo pintado um segmento descoberto por algum segmento removido. Na inserção essa alteração consistirá na pintura do segmento que está sendo inserido por cima da imagem existente de outros segmentos da cena. As figuras III.2 e III.3 mostram as funções de profundidade relativas ao plano y=cte e aos polígonos mostrados na figura III.1. Na figura III.4 encontra-se o resultado da comparação das duas funções, indicando as regiões onde haverá alteração da imagem.

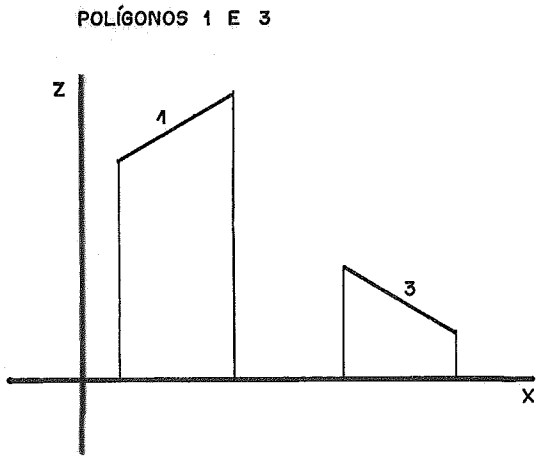


figura III.2

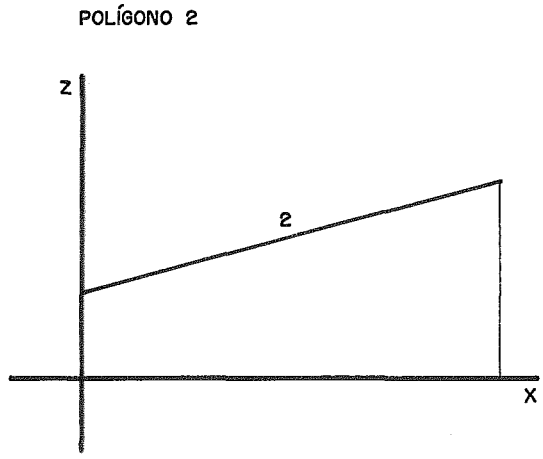


figura III.3

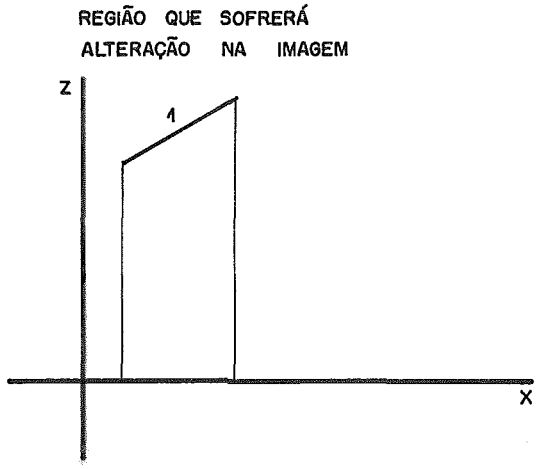


figura III.4

Por fim, deve-se ressaltar que devido às suas características incrementais, com partes da cena sendo restauradas numa análise linha a linha, o algoritmo presta-se apenas para o uso em dispositivos matriciais.

Inicialmente vai-se apresentar o caso mais simples do algoritmo generalizando-se a seguir para os demais. Esse caso será o de remoção da cena de um único polígono, e como já foi visto que os polígonos serão subdivididos em trapé-

zios, vai-se supor que esse polígono único é um trapézio.

III.2- Remoção única

III.2.1- Introdução

O trapézio a ser retirado da cena será denominado trapézio de referência. Assim a primeira providência a ser tomada é a seleção dos trapézios da cena que serão afetados pela retirada do trapézio de referência : os trapézios relevantes. A seguir, como se deseja uma análise linha a linha, deve-se interceptar os trapézios com cada plano y=cte. Fazendo essa interseção vai-se obter uma série de segmentos no plano xz (y=cte) analisado. A análise fica então transposta para esses segmentos no plano xz, de modo que os segmentos mais próximos ao observador, isto é, com coordenada z menor, serão pintados. Essa pintura será feita na projeção do segmento no plano xy. Os segmentos originados dos trapézios relevantes serão denominados segmentos relevantes e o segmento obtido pela interseção do trapézio de referência com o plano y=cte analisado será o segmento de referência.

Como já foi visto na seção III.1, tem-se que, com a retirada do segmento de referência, os segmentos localizados à frente deste não serão alterados; já os localizados atrás deverão ser comparados entre si, para se descobrir qual o mais a frente, e este será pintado. Para se determinar se um segmento está à frente ou atrás do segmento de referência deve-se comparar a coordenada z em ambos segmentos para uma mesma coordenada x. Vai-se apresentar a seguir o procedimento com os segmentos localizados à frente do

segmento de referência e após para os localizados atrás.

III.2.2- Análise para os segmentos localizados à frente do segmento de referência

Ao iniciar-se a análise em um determinado plano y=cte, todo o intervalo de projeção no eixo x do segmento de referência é passível de ser pintado, pois ao ser retirado ele estará descobrindo todos os segmentos localizados atrás dele. Contudo, nos trechos em que há um segmento à frente do segmento de referência, este, ao ser retirado, estará descobrindo segmentos ainda encobertos pelo segmento que está à sua frente, permanecendo portanto a imagem inalterada. Assim em todos os trechos em que houver algum segmento à frente do segmento de referência a imagem não sofrerá alteração. Seria interessante então descobrir quais trechos do segmento de referência, que na retirada deste, causarão alguma alteração na imagem. Esses trechos serão exatamente aqueles em que não há nenhum segmento relevante a sua frente. Assim todo segmento relevante tomado que estiver à frente do segmento de referência deverá reduzir o intervalo deste que interessa para a análise, isto é, aquele em que não há nenhum segmento a sua frente. Exemplificando : se o segmento de referência é (x1,z1)-(x2,z2) e o segmento relevante é (x3,z3)-(x4,z4), há três casos possíveis:

- $x3<x1<x4<x2$ e $z4<z^*$ (z do segmento de referência no ponto x4), nesse caso o segmento relevante recorta o segmento de referência no seu extremo esquerdo, podendo o segmento de referência ser reduzido a (x4,z*)-(x2,z2).

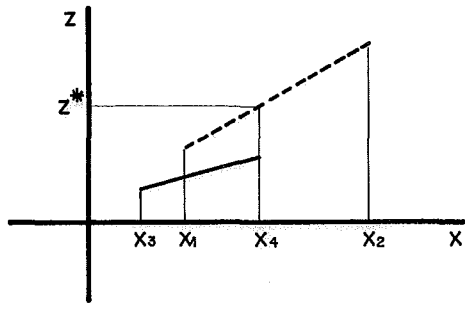


figura III.5

- x1<x3<x2<x4 e z3<z* (z do segmento de referência no ponto x3), nesse caso o segmento relevante recorta o segmento de referência no seu extremo direito, podendo o segmento de referência ser reduzido a (x1,z1)-(x3,z*).

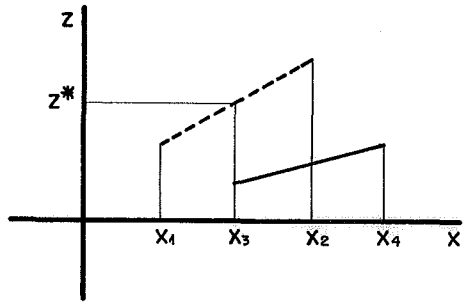


figura III.6

- x1<x3<x4<x2 e z3<z* (z do segmento de referência no ponto x3), assim z4<z** (z do segmento de referência no ponto x4), nesse caso o segmento relevante particiona o segmento de referência em dois : (x1,z1)-(x3,z*) e (x4,z**)-(x2,z2).

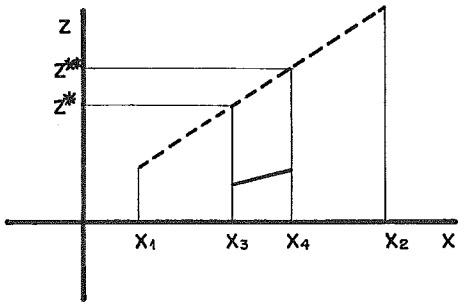


figura III.7

III.2.2.1- A vantagem da ordenação

Portanto, os segmentos relevantes à frente do segmento de referência, à medida que vão sendo tomados, vão reduzindo o intervalo deste podendo inclusive fazê-lo desaparecer. Como já foi visto acima, o segmento de referência pode ir se subdividindo em diversos segmentos menores. Isso seria muito incômodo, pois armazenar múltiplos segmentos é bem mais complexo que armazenar um único; além disso, as comparações mencionadas acima teriam que ser múltiplas, isto é, para cada um dos segmentos de referência gerados. Assim seria interessante evitar a existência de múltiplos segmentos de referência. Isso é conseguido se os segmentos relevantes à frente forem fornecidos ordenados crescentemente pela coordenada x do seu extremo esquerdo. Pois agora, quando se toma um segmento não se tomará depois dele nenhum outro iniciando-se à sua esquerda. Disso decorre o fato de que a porção do segmento de referência localizada antes do extremo esquerdo do segmento relevante tomado não mais será alterada. E assim quando esta porção for retirada certamente causará uma alteração na imagem dentro de seus limites.

Nos limites desse trecho, portanto, já pode ser feita a comparação entre segmentos localizados atrás dele, para se descobrir qual ou quais os mais à frente e que consequentemente serão pintados.

Assim elimina-se a divisão de um segmento em duas partes e os três casos anteriores ficam simplificados. O primeiro caso não irá se alterar, em nenhuma porção do segmento já pode-se fazer a análise dos segmentos localizados atrás,pois o segmento relevante está antes da parte inicial do segmento de referência, e este será apenas reduzido.

No segundo caso, no trecho à esquerda do ponto x3, já pode ser feita a análise para os segmentos localizados atrás, pois esse trecho não terá mais seus limites reduzidos. Já no trecho à direita do ponto x3 observa-se que o segmento de referência está todo encoberto, assim o único trecho do segmento de referência em que haverá alteração da imagem, será a porção antes do ponto x3.

No terceiro caso, na parte anterior ao ponto x3 já pode ser feita a análise para os segmentos localizados atrás, e o segmento de referência reduz-se à sua segunda porção (x4 , x2).

III.2.3- Análise para os segmentos localizados atrás do segmento de referência

Os segmentos localizados atrás do segmento de referência dão origem a duas funções relacionadas com a profundidade e a cor dos segmentos. A função de profundidade associará a cada coordenada x, a coordenada z correspondente ao

segmento mais próximo do observador. A função de coloração associará a esse ponto a cor correspondente a esse segmento. Essas funções serão funções contínuas por partes, pois nos pontos em que um segmento passa a encobrir outro, haverá uma descontinuidade na função de profundidade e consequentemente na função de coloração.

Como foi visto na seção III.2.2.1 só haverá possibilidade de pintura nos trechos em que não há segmentos relevantes à frente do segmento de referência. Ao final da análise dos segmentos localizados atrás do segmento de referência, quando a função de coloração estiver toda avaliada, será feita a pintura nos trechos mencionados acima, com a cor indicada pela função de coloração.

III.2.3.1- Obtenção da função de profundidade e coloração

As funções de profundidade e coloração que foram obtidas após a análise de todos os segmentos localizados atrás, correspondem na realidade à última função de uma sequência de funções. A n-ésima função da sequência corresponde àquela obtida pela consideração dos primeiros n segmentos. Vai-se denominar função do segmento à função de profundidade ou coloração avaliada para um único segmento. Assim a cada coordenada x entre os extremos do segmento, a função de profundidade associa sua coordenada z e a de coloração a sua cor. Fora dos extremos do segmento será associado a profundidade máxima e a cor do fundo. Dessa maneira, a função de profundidade de ordem n pode ser obtida pelo mínimo entre a função de profundidade de ordem n-1 e a função de profundidade do segmento n. A função de coloração de ordem n corresponderá à função de coloração do segmento

n nas partes em que a função de profundidade do segmento n tiver um valor menor que a função de profundidade de ordem n-1. As figuras abaixo ilustram a comparação entre as funções de profundidade, onde Pn(x), Pn-1(x) são as funções de profundidade de ordem n e n-1 e Psn(x) é a função de profundidade do segmento n.

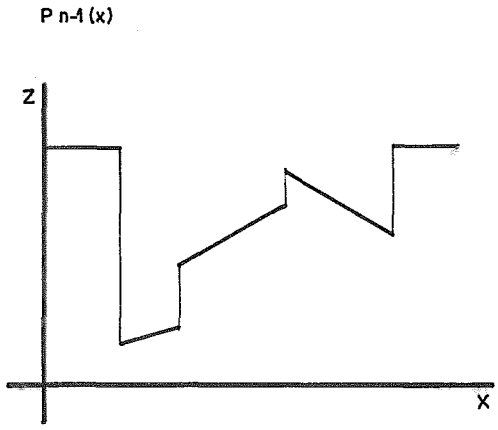


figura III.8

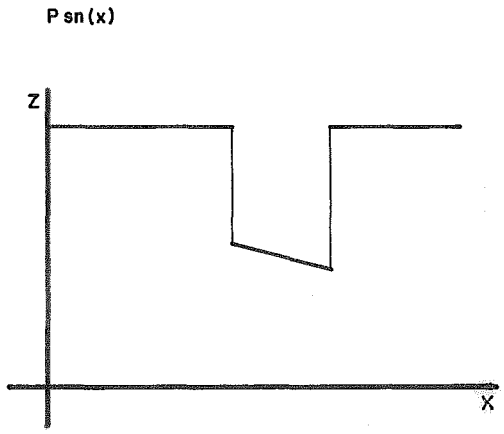


figura III.9

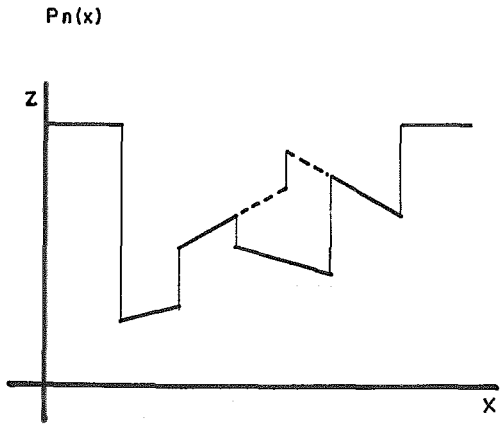


figura III.10

A sequência de funções calculadas dessa maneira conterá,

ao final, a profundidade e a cor dos segmentos mais próximos ao observador (coordenada z menor)

III.2.3.2- A vantagem da ordenação

Os segmentos localizados atrás do segmento de referência podem também ser tomados ordenadamente pela coordenada x do seu extremo esquerdo, como os segmentos localizados à frente. Se isso acontecer, vai ocorrer a seguinte situação :

$$P(x) = Pn(x) \ , \ x < xn$$

$$C(x) = Cn(x) \ , \ x < xn,$$

onde P(x) e C(x) são as funções de profundidade e coloração avaliadas após todos os segmentos, Pn(x) e Cn(x) as funções avaliadas após o n-ésimo segmento e xn a coordenada x do extremo esquerdo do n-ésimo segmento. Isso indica que a porção das funções anteriores ao extremo esquerdo de um segmento tomado serão mantidas nas funções seguintes da sequência, avaliadas para os próximos segmentos. Isso ocorre porque o extremo esquerdo de um próximo segmento terá necessariamente a coordenada x maior ou igual que o atual, e quando for feita a operação para encontrar o mínimo entre a função desse próximo segmento com a função de profundidade atual a nova função não terá nenhuma alteração antes do extremo esquerdo desse próximo segmento, pois antes disso sua função de profundidade conterá o valor zmax. Pelas figuras abaixo, vê-se que a função de profundidade do novo segmento, que tem o extremo esquerdo em x1, só poderá alterar a função Pn(x) na operação de encontrar o mínimo entre as duas, a partir do ponto x1.

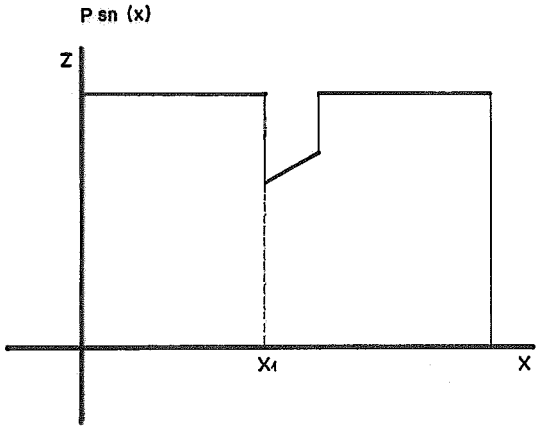


figura III.11

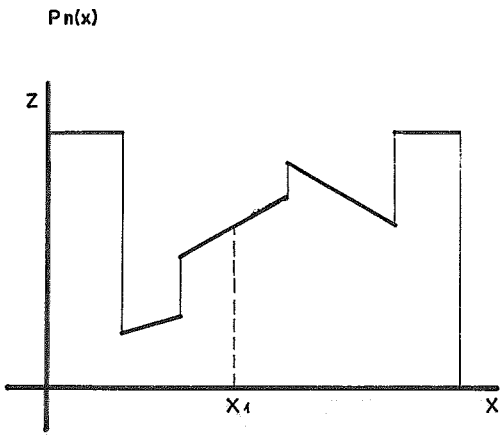


figura III.12

III.2.3.3- Propriedades da função de profundidade

Como foi visto na seção III.2.3.2, quando se analisa um segmento, as funções subsequentes da sequência não sofrerão alteração nas porções anteriores ao extremo esquerdo desse segmento. Como as funções de profundidade não sofrerão mais alterações, as funções de coloração também não. Considerando que a função de profundidade é construída justamente

para que possa ser obtida a função de coloração, e como essa última não se altera, então a função de profundidade não precisa ser avaliada antes desse ponto. Desse modo, ao se construir a função de profundidade e coloração de ordem n, só é preciso considerar a porção após o extremo esquerdo do segmento n. A porção anterior a esse ponto, na função de coloração, pode ser usada para realizar a pintura.

Se esse procedimento acima for adotado a função de profundidade de qualquer ordem só terá descontinuidades crescentes. Entende-se como descontinuidade crescente uma descontinuidade em que o segmento à esquerda tem sua coordenada z menor que a do segmento à direita, no ponto de descontinuidade. As descontinuidades decrescentes não existirão porque a nova função só está sendo considerada após o extremo esquerdo do novo segmento. A figura III.13 ilustra essa propriedade.

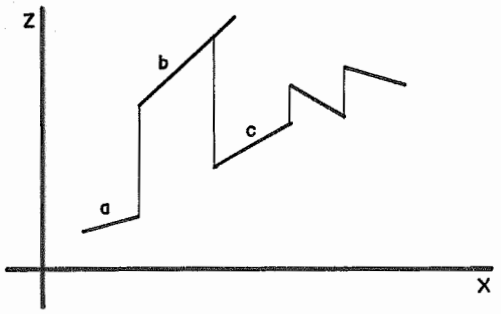


figura III.13

O novo segmento c, a princípio, teria gerado uma descontinuidade decrescente, contudo como a função só é considerada após o extremo esquerdo do novo segmento, ela na

realidade fica como mostrado na figura III.14. Essa propriedade pode ser explorada para uma implementação mais eficiente de uma estrutura de dados que represente a função de profundidade, como será mostrado na seção IV.4.1.1.2.

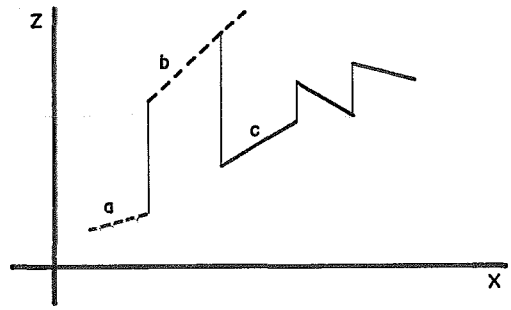


figura III.14

III.2.4- Análise conjunta dos segmentos localizados à frente e atrás do segmento de referência

Como já foi visto é recomendável que os segmentos atrás e à frente do segmento de referência sejam fornecidos ordenadamente e apesar da análise acima ter sido feita separadamente para cada um deles, isso não ocorre, pois os segmentos relevantes estão à princípio, todos misturados. Das duas análises acima decorre que quando um segmento é tomado a porção antes dele na função de profundidade e de coloração não será mais alterada. Assim essa porção das funções não precisa ser avaliada para esse segmento e os subsequentes e já podem ser usadas (função de coloração) para realizar a pintura na linha avaliada. Se o segmento estiver atrás do segmento de referência é feita a operação

de mínimo entre a função desse segmento e a função de profundidade. Caso o segmento esteja à frente do segmento de referência, como foi visto na seção III.2.2.1, no intervalo entre os extremos desse segmento não haverá nenhuma alteração da imagem, assim não há porque avaliar a função de profundidade e coloração nesse intervalo. Portanto, quando se tem um segmento à frente do segmento de referência, as próximas funções de profundidade e coloração só serão avaliadas após o extremo direito desse segmento.

Assim a função Pn+1(x) só será avaliada após o extremo direito do segmento n, caso ele esteja à frente do segmento de referência; e não haverá avaliação da função Pn(x), pois o segmento está à frente do segmento de referência e a função só é avaliada para os segmentos localizados atrás. A função Pn(x) é avaliada após o extremo esquerdo do segmento n, caso ele esteja atrás do segmento de referência.

III.2.5- Generalização

Inicialmente foi feita uma restrição dos polígonos serem todos trapézios. Se isso não ocorrer, eles devem ser divididos em trapézios com dois lados paralelos ao eixo x da janela de exibição. O mesmo vale para o polígono de referência, a análise é feita para cada um de seus trapézios que venham a surgir quando um outro termina. O único problema seria se esse polígono fosse não convexo e contivesse numa ou várias linhas dois ou mais trapézios, nesse caso a análise a cada linha seria feita considerando que o segmento de referência é composto por esses dois ou mais subsegmentos disjuntos.

III.3- Inserção única

No caso da inserção o que se deseja é que o novo polí-gono apareça nas regiões em que não há ninguém à frente dele. Se fosse executado o algoritmo de remoção deste polígono, mas sem retirá-lo da estrutura da cena (ao contrário, tendo já o inserido), o que se obteria seriam exatamente as alterações causadas por ele. Isso porque quando um polígono é retirado, o que é pintado são os segmentos mais próximos e atrás dele, naquelas regiões onde ele não é encoberto por ninguém. No caso analisado o polí-gono que vai ser retirado é coincidente com o polígono que acabou de ser inserido na estrutura da cena, logo esse polígono será sempre o mais próximo do polígono a ser retirado. Assim nas regiões em que este não possui nada à sua frente, o novo polígono será pintado. Dessa maneira a inserção de um polígono pode ser resolvida da mesma maneira que a remoção. A única diferença é que esse polígono, o polígono de referência, no caso da remoção terá seus tra-pézios retirados da estrutura da cena, e no caso da inser-ção eles serão lá inseridos.

III.4- Remoção e inserção em grupo

O caso mais geral que é a inserção ou remoção de um objeto que contenha vários polígonos pode ser resolvido de maneira similar. Será descrito o procedimento para remoção de um grupo de polígonos, o procedimento para inserção, como já foi visto, pode ser resolvido através da remoção.

Se se pudesse garantir que em qualquer plano y=cte as interseções no plano xz dos polígonos do objeto não possuem

projeções que se interceptam no eixo x, teria-se a mesma situação da remoção de um polígono não convexo, em que num mesmo y há diversos segmentos cujas projeções são disjuntas. Nota-se porém, que o que interessa para modificar a cena é somente o segmento de referência que está mais a frente, já que outros segmentos de referência localizados atrás deste em nada afetam a cena, quando da sua retirada. Assim poderia se criar uma função de profundidade também para os segmentos de referência, pois como se viu a função armazena somente uma profundidade em cada ponto (exceto nos pontos de descontinuidade). Assim teria-se para cada coordenada x o valor da coordenada z correspondente ao segmento de referência mais a frente. Se função de profundidade for avaliada da mesma maneira que o descrito na seção 2, poderia se usar as mesmas propriedades descritas. Contudo aqui não há função de coloração, porque o que se deseja não é pintura, mas um conjunto de segmentos disjuntos. Assim ao final da avaliação da função cada um de seus trechos contínuos representará um novo segmento de referência, disjunto de todos os outros e que poderá ser usado na comparação com os segmentos relevantes.

Poderia-se também seguindo a mesma idéia de remoção de um objeto realizar a inserção e remoção de vários objetos antes da cena ser atualizada. Os segmentos pertencentes a esses objetos formam o conjunto de segemntos de referência. No caso da inserção os trapézios que formam o objeto são inseridos na estrutura da cena, no caso da remoção eles são retirados dessa estrutura. A partir daí procede-se como descrito no parágrafo anterior.

CAPÍTULO IV

Descrição da implementação

IV.1- Introdução

A execução do algoritmo de inserção ou remoção incrementais se inicia com os objetos, fornecidos pelo progrma de aplicação, que deverão ser removidos ou inseridos na cena. Os objetos consistem numa coleção de polígonos.

Os polígonos que formam a cena se encontram numa estrutura denominada estrutura da cena, montada com os trapézios desses polígonos. Os polígonos que serão inseridos ou removidos da cena, se encontram também numa estrutura denominada estrutura de referência, também constituida de trapézios. Assim no caso de uma inserção, os polígonos que formam o objeto são inseridos na estrutura da cena e na de referência; no caso de uma remoção esses polígonos são retirados da estrutura da cena e inseridos na estrutura de referência. Esse procedimento de inserir ou retirar da estrutura da cena e inserir na estrutura de referência é feita com todos os objetos que devem ser inseridos ou retirados da cena, até que seja solicitada a atualização desta. Nesse momento encerra-se o grupo de procedimentos denominado pré-processamento e inicia-se o processamento de atualização da imagem.

A fase de atualização da imagem, de acordo com que foi apresentado no capítulo III, consiste numa análise feita em planos y=cte, partindo do topo da cena, fazendo com que cada plano analisado corresponda a uma linha da tela. Assim a diferença entre dois y's constantes analisados é de uma

unidade. A etapa de atualização pode ser dividida em duas partes ou módulos : o módulo de avanço e o módulo de análise de segmentos. O módulo de avanço consiste no conjunto de procedimentos executados ao término da análise com um determinado plano y=cte, antes de se começar a análise no plano seguinte. O módulo de análise de segmentos consiste no conjunto de procedimentos executados num determinado plano y=cte com os segmentos resultantes da interseção dos polígonos da estrutura da cena e da referência num mesmo plano.

A etapa de pré-processamento é solicitada através dos procedimentos INSERE POLÍGONO que insere um polígono no objeto que está sendo inserido; CRIA OBJETO que cria um objeto a ser referenciado pelo índice dado com os polígonos fornecidos pelo procedimento anterior; REMOVE OBJETO que remove o objeto indicado pelo seu índice. A etapa de atualização é solicitada pelo procedimento ATUALIZA CENA. A forma como os dados devem ser passados para eses procedimentos se encontra no Apêndice I.

A seguir serão apresentados maiores detalhes das fases de pré-processamento, avanço e análise de segmentos. Ao final serão apresentados as principais estruturas de dados utilizadas nessa implementação.

IV.2- Pré-processamento

Após terem sido fornecidos os vértices dos polígonos que formam um objeto a ser inserido na cena, é solicitado que esse objeto seja então criado e referenciado por um índice. Como foi visto no capítulo III o algoritmo não usa

polígonos no seu processamento, mas sim trapézios. Assim os polígonos já podem ser armazenados decompostos em trapézios. A decomposição em trapézios é feita de modo que as bases dos trapézios sejam paralelas ao eixo x do sistema coordenado da janela de exibição. Além disso cada base de um trapézio deverá conter pelo menos um vértice do polígono original. As estruturas da cena e de referência que armazenarão os polígonos a serem inseridos, na realidade irão armazenar os trapézios resultantes da decomposição destes. A estrutura da cena e de referência serão do tipo estrutura de trapézios, que contém para cada linha horizontal da tela os trapézios cujas bases superiores encontram-se nela. Dentro de cada linha os trapézios estão ordenados na estrutura crescentemente pela coordenada x do seu extremo superior esquerdo. Assim esse tipo de estrutura fornece para cada linha os trapézios que se iniciam nela.

No caso da remoção de um objeto da cena, não é necessário que sejam fornecidos os polígonos que formam o objeto, pois se eles estão sendo removidos é porque já foram inseridos e encontram-se dispersos na estrutura da cena na forma de trapézios. Uma estrutura de trapézios possui uma outra forma de acesso, além da que permite determinar os trapézios que se iniciam numa determinada linha : dado o índice de referência de um objeto é possível obter todos os trapézios na estrutura que pertençam a esse objeto. Assim na remoção de um objeto, os trapézios que formam esse objeto devem ser retirados da estrutura da cena. A seguir, esses trapézios são inseridos, como no caso da inserção, na estrutura de referência.

Dessa maneira ao ser solicitada a atualização da imagem

da cena, a estrutura de referência conterá todos os trapézios dos objetos a serem inseridos ou removidos e que serão usados nas comparações com os trapézios da estrutura da cena. Os trapézios que se encontram na estrutura de referência serão denominados trapézios de referência e os que se encontram na estrutura da cena, trapézios da cena. Como só interessa a análise dos trapézios da cena que são afetados pelos trapézios de referência, esses trapézios da cena serão denominados trapézios relevantes.

IV.3- Avanço

O módulo de avanço calcula os pontos extremos das interseções dos trapézios relevantes (aqueles trapézios afetados pelos trapézios de referência) e de referência com o plano y=cte analisado. Essas interseções, denominadas segmentos, serão usadas na análise dos segmentos para se determinar as regiões a serem pintadas. Além disso entre um plano y=cte analisado e outro, um novo trapézio pode passar a interceptar os planos analisados, ou um trapézio pode deixar de interceptar os planos analisados.

Os segmentos encontram-se dispostos em listas, denominadas listas de segmentos relevantes e de referência. Os segmentos se encontram ordenados pela coordenada x de seu extremo esquerdo. Na etapa de avanço os extremos desses segmentos são recalculados para o novo plano y=cte analisado. Ocorre que a ordenação das listas pode não se manter após a atualização, assim é necessário que seja verificado se ela foi mantida e se isso não ocorreu ela deve ser refeita.

Como os trapézios tem suas bases paralelas ao eixo x, quando um segmento de interseção com o plano y=cte corresponder à base inferior de um trapézio, os próximos planos y=cte não mais interceptarão esse trapézio. Assim o nó que representava os segmentos desse trapézio deve ser removido da lista. Para detetar se há trapézios que venham a surgir a partir de um determinado plano é necessário que a cada plano analisado seja verificado na estrutura da cena e de referência se há algum trapézio com base superior naquele y. Se isso ocorrer a base superior do trapézio é inserida na lista de segmentos, respeitando sua ordenação. Antes porém, se o trapézio que surgir for um trapézio da estrutura da cena, é necessário verificar se ele é afetado pelos trapézios de referência, isto é, se ele é relevante ou não, e somente se ele for deverá ser inserido na lista de segementos. Para se detetar se um trapézio é relevante seus limites superior, inferior e laterais são comparados com os mesmos limites do conjunto dos polígonos de referência. Se for detetado através dos limites do trapézio que ele se encontra, ao menos parcialmente, dentro dos limites do conjunto dos polígonos de referência, ele é considerado relevante.

IV.4- Análise dos segmentos

Esse módulo inicia-se de posse da lista de trapézios relevantes representados por seus segmentos e uma lista de trapézios de referência, também representada por segmentos, os segmentos de referência, ambos correspondentes a um plano y=cte. Além disso, esse módulo fará uso das funções de profundidade e coloração que foram definidas na seção

III.2.3. A função de profundidade será representada por uma estrutura de dados de tal modo que a operação de se inserir um segmento na estrutura que representa a função corresponda a fazer o mínimo entre a função de profundidade e a função de profundidade desse segmento. Além disso como foi visto na seção III.2.3.3, a cada novo segmento tomado, a porção da função de profundidade localizada antes de seu extremo esquerdo não sofrerá mais alteração, fazendo com que não haja mais alteração na função de coloração. Assim a função de profundidade nas próximas iterações só precisa ser avaliada a partir desse ponto. Isso significa que na estrutura de dados, se for interessante, a porção anterior a esse ponto pode ser eliminada. Quanto à função de coloração, ela pode ser incluída na mesma estrutura que a função de profundidade, bastando que além da profundidade sejam armazenadas as cores dos segmentos, já que cada segmento possue uma única cor.

A primeira estrutura de dados usada para representar a função de profundidade é uma lista encadeada, denominada lista de profundidade, cujos nós serão a representação dos diversos trechos contínuos da função, além de conter sua cor. A implementação com essa estrutura de dados aproxima-se muito do algoritmo descrito na seção III.2.

Uma outra estrutura usada para representar a função de profundidade é um vetor, denominado z-buffer, em que cada posição representa um pixel da linha da tela analisada, contendo a profundidade e a cor desse ponto. Como essa estrutura é uma representação discreta da função, ela seria pouco eficiente se o algoritmo utilizado fosse o mesmo da

representação com lista, pois esse algoritmo é mais orientado para representações contínuas da função. Com essa estrutura de dados será utilizado a forma mais geral do algoritmo proposto, que se encontra descrito na seção III.1.

A implementação pode também levar ou não em consideração o fato de não haver interpenetrações entre os polígonos. Assim foram desenvolvidas três implementações diferentes : utilizando a lista de profundidade e não considerando interpenetrações ou utilizando z-buffer e considerando ou não interpenetrações. Cada uma das três possibilidades será tratada abaixo.

Esse módulo de análise de segmentos pode ser dividido em duas partes : na primeira serão tratados os segmentos de referência e na segunda os segmentos relevantes. Como foi visto na seção III.4, para que os segmentos de referência possam ser usados na comparação com os segmentos relevantes, eles precisam estar organizados numa estrutura em que só os segmentos de referência mais à frente sejam considerados e que só haja segmentos disjuntos. Isso será conseguido inserindo-se os segmentos de referência iniciais na lista de profundidade ou no z-buffer. Na segunda parte os segmentos relevantes serão testados contra os segmentos de referência tomados e organizados na primeira parte.

IV.4.1- Utilização da lista encadeada sem interpenetração entre polígonos (caso 1)

IV.4.1.1- Procedimentos com os segmentos de referência

São os seguintes os procedimentos com cada um dos

segmentos de referência :

- retira-se da lista de profundidade todos os nós anteriores ao extremo esquerdo do segmento de referência tomado (seção IV.4.1.1.1).

- insere-se o novo segmento na lista de profundidade (seção IV.4.1.1.2).

A lista de profundidade de referência é inicializada com um nó correspondente ao fundo. E vai-se denominar lista de trabalho de referência à lista em que serão colocados os nós retirados da lista de profundidade como em IV.4.1.1.1 . Essa lista só conterá segmentos disjuntos e será utilizada nas comparações com os segmentos de referência. Essa lista conterá os segmentos de referência mais à frente em cada ponto.

Após todos os segmentos de referência terem sido inseridos na lista de trabalho, devem ser retirados da lista de profundidade, da mesma forma que a retirada de IV.4.1.1.1, todos os segmentos que ainda estejam na lista.

IV.4.1.1.1- Retirada da lista de profundidade

Como os segmentos são tomados ordenadamente pela coordenada x do seu extremo esquerdo, quando se toma um segmento toda a porção da lista de profundidade anterior a ele não mais será alterada pelos segmentos subsequentes, como foi ,mostrado no início da seção IV.4. Assim os segmentos que se encontram na lista de profundidade e sejam anteriores a esse ponto, como não sofrerão mais alteração já podem ser retirados da lista e inseridos na lista de trabalho de referência, para serem posteriormente usados na comparação com os segmentos relevantes.O nó da lista de

profundidade que contem o extremo esquerdo do novo segmento será particionado nésse ponto. A parte anterior a esse ponto é removida e inserida na lista de trabalho e a posterior é mantida na lista de profundidade.

IV.4.1.1.2- Inserção na lista de profundidade

Quando um segmento é inserido na lista de profundidade o primeiro nó desta tem a cooordenada x de seu extremo esquerdo exatamente igual ao do segmento a ser inserido. Isso porque tudo anteriormente já foi retirado da lista (seção IV.4.1.1.1 ou IV.4.1.2.4).

Na seção III.2.3.3 foi visto que a função de profundidade só possuirá descontinuidades crescentes. Além disso nessa implementação desse módulo não está previsto interpenetrações entre os polígonos, isso fará com que não haja intercessões entre os segmentos no plano xz. Devido a esses dois fatos decorre uma propriedade muito útil na inserção de um novo segmento na estrutura de dados que representa a função de profundidade. Essa propriedade diz que a partir do primeiro ponto, contado a partir do extremo esquerdo do novo segmento, em que este encobre alguma porção da função, ele encobrirá toda a função até seu extremo direito. Assim para se inserir um novo segmento, a lista de profundidade deve ser percorrida de seu ponto inicial até o nó que contem o extremo direito desse novo segmento, a procura do primeiro nó que ele encobre. Um exemplo da inserção de um segmento encontra-se nas figuras IV.1 e IV.2.

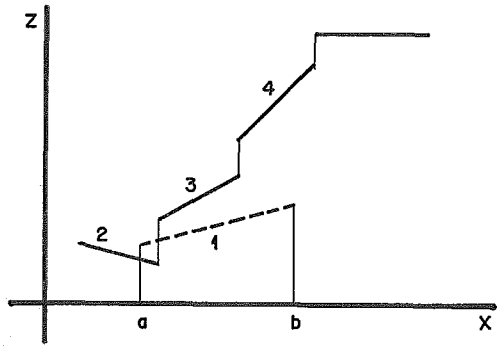


figura IV.1

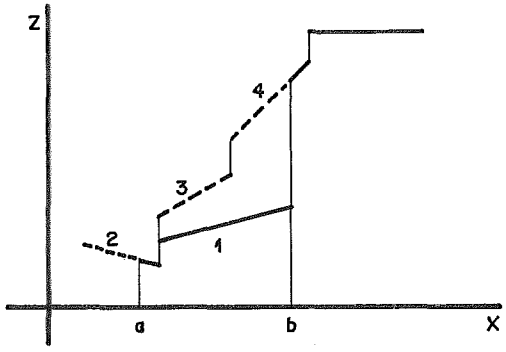


figura IV.2

Naquelas figuras o segmento 1 a ser inserido foi encoberto pelo segmento 2, porém encobriu o segmento 3, assim ele encobrirá todos os demais segmentos até seu extremo direito, que não precisam ser testados. O nó que contem o extremo direito foi particionado e a porção anterior ao ponto de partição, assim como todos os nós intermediários após o primeiro nó encoberto são eliminados da lista de

profundidade e o novo nó é inserido.

IV.4.1.2- Procedimentos com os segmentos relevantes

São os seguintes os procedimentos com cada um dos segmentos relevantes :

- retira-se da lista de profundidade todos os nós anteriores ao segmento tomado e insere-se-os na lista de pintura. Conforme o segmento esteja à frente ou atrás do segmento de referência ativo (explicação adiante) a porção da lista correspondente à sua extensão é retirada, mas não inserida na lista de pintura. A tudo isso denomina-se processar a lista de profundidade (seção IV.4.1.2.4).
- insere-se o segmento na lista de profundidade. Esse procedimento é idêntico ao da seção IV.4.1.1.2 .
- verifica-se se o segmento relevante ultrapassou o segmento de referência ativo (explicação adiante) (seção IV.4.1.2.2).
- compara-se a posição do segmento relevante com o segmento de referência ativo (seção IV.4.1.2.3).

Além disso, a análise deve-se iniciar no primeiro segmento relevante afetado pelos segmentos de referência (seção IV.4.1.2.1).

Para que os segmentos relevantes não tenham que ser comparados contra todos os segmentos de referência, haverá sempre um segmento de referência denominado ativo, contra qual serão feitas as comparações dos segmentos relevantes. Se um segmento relevante ultrapassou os limites do segmento ativo, todos os demais segmentos relevantes o terão ultrapassado, pois a lista de segmentos é ordenada. Nesse caso o próximo segmento de referência da lista de trabalho de

referência é tomado como segmento ativo (IV.4.1.2.2) .

IV.4.1.2.1- Seleção dos segmentos relevantes afetados pelos segmentos de referência

Nenhum dos segmentos relevantes localizados antes do primeiro segmento de referência serão afetados por estes. Como as listas de segmentos são ordenadas, a partir do primeiro segmento afetado todos os demais o serão. Assim a lista de segmentos relevantes é percorrida até que seja encontrada o primeiro segmento cujo os limites interceptam os limites do primeiro segmento de referência. A análise começa a partir daí.

IV.4.1.2.2- Verificação se o segmento relevante tomado ultrapassou o segmento ativo

É verificado se o segmento relevante tomado ultrapassou o segmento ativo, isto é, se o extremo esquerdo do segmento relevante tem o valor da coordenada x maior que o valor da coordenada no extremo direito do segmento ativo. Se só uma parte do segmento relevante ultrapassou o extremo direito do segmento ativo, então esse segmento relevante deve ser particionado em duas partes no ponto com coordenada x igual à coordenada do extremo direito do segmento ativo. A primeira porção resultante da partição está toda contida no intervalo do segmento ativo, logo pode ser comparado com ele (seção IV.4.1.2.3), e a segunda porção será comparada numa próxima iteração, certamente com um outro segmento ativo, já que essa segunda porção será testada posteriormente como um outro segmento da lista.

Um segmento relevante para ser comparado com um segmento ativo deve estar todo contido no intervalo deste, por isso é necessário o passo acima descrito. A figura IV.3 explica o porquê.

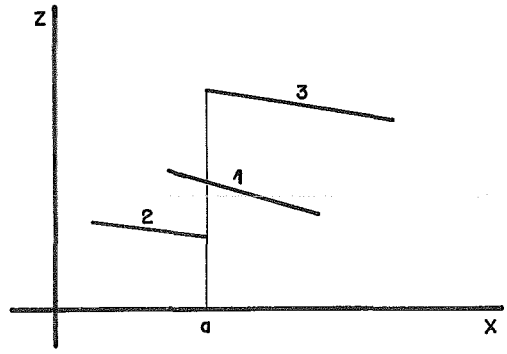


figura IV.3

O segmento relevante 1 está atrás do segmento de referência ativo 2, porém está à frente do segmento de referência 3. Se ele for particionado no ponto a, cada subsegmento poderá ser tratado sem problemas, já que não há possibilidade de interseção entre segmentos, pois não há interpenetrações entre polígonos.

IV.4.1.2.3- Comparação do segmento relevante como segmento ativo

Como não há interseção entre os segmentos para se descobrir se o segmento relevante está à frente ou não do segmento ativo, basta que as coordenadas z de ambos sejam comparadas em uma única coordenada x. Se a coordenada z do segmento relevante tiver um valor menor ou igual que do segmento ativo, o segmento relevante estará atrás do segmento ativo, caso contrário estará à frente.

IV.4.1.2.4- Processamento da lista de profundidade

Quando um novo segmento relevante é tomado, quer ele esteja à frente quer atrás do segmento ativo, toda porção da lista de profundidade localizada antes de seu extremo esquerdo já pode ser retirada da lista, já que não será mais alterada, como foi visto no início da seção IV.4. Essa porção será retirada da lista de profundidade e inserida na lista de pintura : a lista a ser enviada para o módulo de pintura (seção IV.4.1.3), ao final da análise em cada linha. Essa lista conterá os diversos trechos contínuos a serem pintados e as suas cores. O nó da lista de profundidade que contem o extremo esquerdo do novo segmento deverá ser particionado em duas partes. A sua porção esquerda deverá ser inserida também na lista de pintura e retirada da lista de profundidade, enquanto que a porção direita permanece na lista de profundidade.

Em seguida, caso o segmento relevante esteja atrás do segmento de referência ele deverá ser inserido na lista de profundidade (seção IV.4.1.1.2), pois esse segmento é passível de ser pintado. Contudo, caso o segmento relevante esteja à frente do segmento de referência ele não precisará ser pintado e os segmentos que estão no espaço correspondente aos seus limites na lista de profundidade também não necessitarão ser pintados, logo eles devem ser simplesmente retirados da lista. Dessa maneira, se o segmento relevante estiver à frente do segmento ativo todos os nós entre seu extremo esquerdo e direito são retirados da lista de profundidade. O nó em que se encaixa o extremo direito do segmento deve ser particionado em duas partes. A primeira

porção é retirada da lista e a segunda, que fica após o término do segmento, é mantida.

IV.4.1.3- Pintura

Ao final da análise para um determinado plano y=cte, o algoritmo estará de posse da lista de pintura, que conterá os diversos trechos da linha da tela, correspondente a esse plano, que deverão ser pintados e a sua cor.

IV.4.2- Utilização do z-buffer, sem interpenetração entre os polígonos (caso 2)

O z-buffer é um vetor que contem em cada posição, que representa uma coordenada x da tela, o valor da coordenada z naquele ponto e a cor deste ponto. Nesse tipo de implementação uma maior eficiência é conseguida se forem feitas comparações ponto a ponto entre os segmentos relevantes e de referência, ao invés de se tentar uma implementação orientada para uma representação contínua da função de profundidade, como foi a anterior. Assim serão geradas as duas funções de profundidade integralmente, a de referência e a dos segmentos relevantes. Os dois z-buffers serão comparados ponto a ponto e naqueles pontos em que o z-buffer de referência tiver um z menor ou igual ao dos segmentos relevantes, será feita uma pintura de acordo com a cor armazenada no z-buffer dos segmentos relevantes. Isso está significando que existe um segmento de referência que ao ser retirado estará descobrindo segmentos relevantes.

Como aqui os segmentos são fornecidos ordenadamente pode-se começar a análise dos segmentos relevantes naquele que é o primeiro afetado pelos segmentos de referência,

exatamente como o descrito na seção IV.4.1.2.1 . O procedimento aqui consiste em inserir os segmentos de referência no z-buffer de referência e os relevantes no dos relevantes, exatamente da mesma maneira (seção IV.4.2.1). Após isso cada um dos z-buffers conterá para cada x o valor da coordenada z do segmento mais próximo. Assim esses z-buffers podem ser comparados ponto a ponto. Os pontos a serem pintados serão agrupados em trechos contínuos com a mesma cor e formarão uma lista de pintura, que será processada (pintada) como mostrado em IV.4.1.3.

IV.4.2.1- Inserção num z-buffer

Como aqui não há interseção entre os segmentos a mesma propriedade utilizada no caso 1 (seção 4.1.1.2) pode ser aproveitada. Assim vai-se percorrendo o z-buffer até se encontrar um ponto em que a coordenada z do novo segmento seja menor que a coordenada z do ponto. O novo segmento ocupará o espaço então desse ponto até seu ponto extremo direito, não necessitando os demais pontos existentes nesse intervalo serem testados. A coordenada z em cada ponto do novo segmento na comparação acima é calculada, assim como nos pontos intermediários, após ter sido determinado o ponto inicial onde ele se encaixa no z-buffer,por um método que gera pontos de uma reta em coordenadas inteiras, como o de BRESENHAM [8].

IV.4.3- Utilização do z-buffer, permitindo intepenetração entre polígonos (caso 3)

Aqui como são permitidas as interpenetrações entre

polígonos a propriedade apresentada na seção IV.4.1.1.2 não pode ser utilizada. Devido a esse fato e ao fato de se estar utilizando z-buffer, vai se dispensar os segmentos de serem fornecidos ordenadamente, pois essa ordenação seria de pouca utilidade. Assim no módulo de avanço da lista de segmentos não se procurará manter a ordenação, quando esse módulo fornecer dados para essa implementação do módulo de análise dos segmentos.

Os segmentos relevantes e de referência são tomados um a um e inseridos no seu z-buffer (seção IV.4.3.1). A seguir os dois z-buffers são comparados ponto a ponto, exatamente como o descrito no caso 2.

IV.4.3.1- Inserção num z-buffer

A inserção aqui não pode se valer da propriedade usada no caso 2, pois além da possibilidade de haver interseções os segmentos não estão ordenados. Assim é criado um vetor que conterá em cada posição correspondente a cada ponto do segmento a ser inserido, o valor da coordenada z do segmento, gerado por um método como o de BRESENHAM [8]. Esse vetor é então comparado com o z-buffer. Nas posições em que a coordenada z desse vetor for menor que a coordenada armazenada no z-buffer, a coordenada do z-buffer é substituída por essa nova coordenada e a cor pela cor do segmento que está sendo inserido.

IV.5 - Estrutura de dados

A seguir serão apresentas as principais estruturas de dados utilizadas pelo algoritmo. Todos os campos numéricos são representados por números inteiros.

- estrutura de trapézios : consiste em um vetor em que cada posição representa uma linha horizontal da tela, apontando para uma lista encadeada em que os nós são trapézios cuja base superior esteja nessa linha. Essas listas estão ordenadas pela coordenada x do extremo superior esquerdo do trapézio. Cada nó contem as coordenadas dos quatro extremos do trapézio, sua cor, um ponteiro para o próximo trapézio da lista e um ponteiro para o próximo trapézio do mesmo objeto que esteja nessa estrutura. Esse último ponteiro é utilizado no caso de uma remoção em que a partir do trapézio com o extremo esquerdo superior de maior coordenada y, pode-se chegar a todos os outros que compõe o objeto.

- lista de segmentos : consiste em uma lista encadeada representando os segmentos que são as interseções dos trapézios com um plano y=cte. Os nós, além dos extremos do segmento e da sua cor, contêm os incrementos necessários para que os extremos possam ser calculados para o próximo y, que será o y atual mais uma unidade. Além disso na implementação com lista encadeada representando a função de profundidade, os nós conterão também coeficientes para que uma coordenada z de um ponto entre os extremos do segmento possa ser interpolada pela coordenada x.

- lista de trabalho de referência : consiste em uma lista encadeada representando segmentos de reta, correspondentes a segmentos de referência disjuntos. E utilizada apenas na implementação com lista encadeada representando a função de profundidade. Cada nó conterá as coordenadas dos extremos e os coeficientes para interpolar uma coordenada z, dada a coordenada x.

- lista de profundidade : consiste em uma lista encadeada representando a função de profundidade. Cada nó representa um trecho da contínuo da função, assim ele conterá as coordenadas dos extremos, a cor e os coeficientes para interpolar uma coordenada z a partir da coordenada x.

- z-buffer : consiste em um vetor representando a função de profundidade e de coloração de forma discreta. As posições do vetor representam os pixels de uma linha da tela. Cada posição conterá a coordenada z associada a ela e a cor.

- lista de pintura : consiste em uma lista encadeada representando os trechos a serem pitados de uma mesma linha da tela. Cada nó contem a coordenada x dos extremos e a cor.

- vetor-vértice : consiste na matriz em que são fornecidos os vértices de um polígono. A primeira dimensão varia de 1 até o número de vértices do polígono, enquanto a segunda dimensão varia de 1 a 3, representando as coordenadas x, y, e z do vértice.

A mais crítica entre as estruturas utilizadas é a estrutura de trapézios, já que é essa estrutura que armazena os trapézios da cena, quer estejam visíveis ou não. De acordo com ASANO et alii [9] o número de trapézios com bases paralelas o eixo x, resultantes da decomposição de um polígono é n + w - h - d - 1, onde n, w e h são o número de vértices, janelas e arestas horizontais do polígono e d é relacionado com o número de vértices que estão na mesma linha do polígono. Assim um polígono geral sem janelas, arestas horizontais e vértices alinhados será decomposto em

n-1 trapézios, onde n é o número de vértices. A estrutura de trapézios precisa de 22 bytes para armazenar as coordenadas dos extremos de um trapézio. Assim um polígono geral precisará de 22 x (n-1) bytes para armazenar as coordenadas. Dessa maneira a memória requerida para a estrutura será diretamente proporcional ao número de polígonos da cena e ao número de vértices desses polígonos.

CAPÍTULO V

Resultados e conclusões

Das três implementações executadas observa-se que a implementação 1, cuja função de profundidade é representada por uma lista encadeada, foi a mais eficiente. Isso era de se esperar, pois essa implementação trabalha com os extremos dos segmentos, não interpolando e nem comparando valores intermediários, como nas outras duas implementações. A implementação 2 foi um pouco mais eficiente que a 3, pois como não foram permitidas interpenetrações entre polígonos, algumas propriedades da implementação 1 puderam ser utilizadas, evitando algumas comparações. A implementação 3 por ser a mais geral, foi a menos eficiente. Essa implementação se aproxima muito do método do "z-buffer" ou memória com profundidade descrito no capítulo II, em que o algoritmo é executado para cada linha da tela.

As implementações foram desenvolvidas em ambiente Turbo Pascal 4.0, em um microcomputador tipo PC-XT, equipado com placa gráfica CGA: O código para as três implementações fartamente documentadas ocupam a memória mostrada abaixo, em kbytes. O total de memória ocupada pelas três implementações é menor que a soma de cada uma, pois como foi visto há módulos comuns às três.

	código fonte	código executável
implementação 1	106	59
implementação 2	84	49
implementação 3	70	44
total	175	

Como teste para o programa foi utilizada a cena de uma sala em projeção. Os objetos inseridos separadamente foram o piso (quadrilátero), árvores (dois quadriláteros e dois triângulos), uma mesa (quadrilátero e quatro retas), uma cadeira (dois quadriláteros e quatro retas), paredes traseiras com janelas (cinco quadriláteros), paredes frontais com porta e janela (oito quadriláteros) e o teto (um quadrilátero). Esses objetos testam fartamente a eliminação de superfícies ocultas e ocupam aproximadamente um terço da área da tela. Após a inserção cada objeto foi retirado separadamente. O tempo de execução, em minutos, das três implementações foi o mostrado abaixo, sendo que o tempo de inserção inclui também a transformação de visualização em cada objeto.

	inserção	remoção
implementação 1	1:35	1:22
implementação 2	3:10	2:47
implementação 3	5:25	4:45

Uma outra abordagem para resolução do problema de eliminação de superfícies ocultas de forma incremental foi inicialmente considerada, mas depois abandonada. Ela consistia na utilização de um grafo planar para representar a imagem visível na tela. Os vértices do grafo corresponderiam ou a vértices de um polígono ou a pontos em que as projeções no plano xy de dois polígonos se cruzassem, fazendo um encobrir outro. O grafo teria que armazenar nos seus vértices e arestas informações de profundidade para que novos polígonos pudessem ser inseridos. A primeira

desvantagem é a quantidade de processamento necessária para localizar os vértices do polígono a ser inserido nos nós do grafo. Além disso para que a estrutura pudesse suportar remoções seria necessário que contivesse informações de profundidade dos polígonos não visíveis em todos os vértices e arestas do grafo contidos no interior de sua projeção no plano xy. Isso faria com que a estrutura ficasse excessivamnete carregada, inviabilizando-a.

Quanto a comparações do algoritmo proposto com outros algoritmos deve-se primeiro ressaltar a finalidade de cada um, pois esse algoritmo foi desenvolvido visando a eficiência em aplicações incrementais e não a eficiência na geração da imagem final da cena em uma só execução do algoritmo. Assim é provável que um algoritmo "scan-line" convencional traga melhores resultados em uma aplicação em que não se deseja inserções ou remoções incrementais na cena, apenas que ela seja gerada de uma só vez. Contudo, nas aplicações em que essas inserções e remoções incrementais forem necessárias é natural que um algoritmo que processe apenas a parte da cena afetada por essas operações seja mais eficiente que um outro que tenha que processar a cena inteira a cada operação. Logicamente essa diferença de eficiência será tão maior quanto mais complexa for a cena e quanto menor for a porção a ser inserida ou removida.

Outro fato a se considerar na eficiência do algoritmo é que houve a preocupação com que as operações de inserção e remoção fossem resolvidas da mesma maneira, para que elas pudessem ser combinadas antes de uma atualização da imagem, com isso conseguiu-se um código mais compacto e simples.

Como a idéia inicial do algoritmo foi desenvolvida para remoção e a seguir adaptada para inserção, é provável que se fosse desenvolvido um algoritmo para inserção visando só essa operação obteria-se melhores resultados com ela.

No desenvolvimento da implementação 1 adotou-se que as coordenadas deveriam ser expressas em números inteiros e assim conseguiu-se que todos os cálculos repetitivos fossem feitos sem o uso de ponto flutuante. A vantagem disso está na eficiência muito maior dos cálculos inteiros frente a cálculos com ponto flutuante. Assim todos os incrementos de segmentos e interpolações foram ajustados para que pudessem ser feitos só com cálculos inteiros. Além disso as coordenadas na estrutura da cena, que deve ser mantidas na memória durante toda a execução do algoritmo, e nas demais estruturas sendo em números inteiros, ocupam pelo menos metade do espaço ocupado, caso fossem utilizadas coordenadas reais. Contudo no uso de coordenadas inteiras são feitas algumas aproximações, que eventualmente podem levar a resultados errôneos em comparações. Por exemplo, interpolações e incrementos de segmentos de reta são feitos utilizando o método de BRESENHAM [8] que, por vezes, para uma mesma coordenada x faz corresponder vários pontos com coordenadas z diferentes ou vice-versa. Isso pode gerar problemas como o mostrado abaixo.

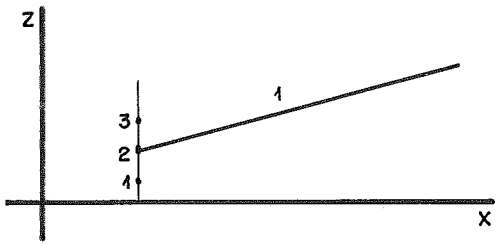


figura V.1

Suponha que se deseja comparar a reta 1 no seu extremo esquerdo, o ponto a, com a reta 2, cujo resultado da aplicação do método de BRESENHAM [8] no ponto a é o mostrado pelos pontos marcados. Assim o resultado da interpolação dessa reta no ponto a poderia ser o ponto 1, 2 ou 3. A utilização de cada um desses três pontos traria resultados diferentes, em relação a uma reta estar à frente ou atrás de outra. Esse problema não ocorreria se fossem utilizadas interpolações reais, ao invés de inteiras, pois haveria somente uma coordenada z para cada coordenada x.

Além disso, os problemas de aproximação causados pelo uso de coordenadas inteiras inviabilizam uma implementação usando lista encadeada e permitindo interpenetrações entre polígonos. Nesse tipo de implementação tem que se testar a existência e calcular as interseções entre os segmentos relevantes e os segmentos de referência. Os segmentos são então particionados de modo que os segmentos resultantes não se interceptem. Esse tipo de implementação foi testada e verificou-se a ocorrência de muitos erros, principalmente quando havia interseções nas proximidades das bordas dos polígonos; isso levava o algoritmo tirar conclusões erradas quanto a um segmento estar à frente de outro. Essa imple-

mentação, contudo seria completamente realizável se fossem utilizadas coordenadas reais.

O algoritmo proposto tem uma limitação quanto à memória necessária para sua execução. Toda a estrutura da cena tem que ser mantida na memória, mesmo que ela não esteja sendo totalmente exibida, pois com uma remoção qualquer parte pode passar a ser visível e então exibida. Como cada trapézio da estrutura armazena 10 coordenadas inteiras de seus extremos, sua cor e também dois ponteiros, então só em coordenadas e cor cada trapézio requer 22 bytes. De acordo com a seção IV.5 um polígono sem arestas horizontais, sem vértices na mesma linha e sem janelas, com n vértices, terá n-1 trapézios. Portanto, um polígono de n vértices, de modo geral, necessitará de (n-1) * 22 bytes para armazenar as coordenadas. Com isso uma cena complexa, pode exigir uma memória que torne a execução proibitiva em equipamentos menores. Um procedimento para minimizar um pouco esse problema seria pré-processar os objetos, de modo que trapézios que sejam totalmente encobertos pelos outros trapézios do objeto sejam eliminados. Como os objetos são inseridos ou removidos sempre integralmente, os trapézios do objeto encobertos por outros do próprio objeto, nunca tem chance de serem exibidos, podendo portanto ser eliminados da estrutura da cena. Esse procedimento pode ser muito vantajoso se os objetos participantes da cena forem muito complexos. Contudo, em qualquer caso, esse procedimento iria requerer um processamento extra para eliminação de superfícies ocultas dentro de cada objeto.

A principal idéia do algoritmo proposto é a de que a análise em cada linha é transformada numa comparação de

duas funções, uma representando os polígonos que entram ou saem da cena e outra os polígonos que estão na cena e podem ter a sua imagem afetada por esses outros. Essa comparação como foi vista nos capítulos III e IV pode ser feita de várias maneiras, aproveitando ou não algumas propriedades. Essa idéia, assim como as propriedades, podem ser levadas para outras aplicações que envolvam análise de profundidade linha a linha. Uma aplicação similar é, por exemplo, a eliminação de superfícies ocultas em animação de objetos rígidos com observador móvel, ao contrário da aplicação do algoritmo proposto que é para animação com observador fixo. Nessa aplicação poderia ser analisada alguma forma de se aproveitar o conceito de função de profundidade, já que ela tem muitas semelhanças com a aplicação do algoritmo proposto.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] CARLBOM, I. e PACIOREK, J. , "Geometric Projection and Viewing Transformations", Computing Surveys, 1(4), 1978, pp 465-502

[2] PERSIANO, R. e OLIVEIRA, A. , Introdução à Computação Gráfica, Belo Horizonte, UFMG, 1986

[3] NEWMAN, W. e SPROULL, R. , Principles of Interactive Computer Graphics, 2nd ed. Mc Graw-Hill, New York, 1979

[4] SUTHERLAND, I. , SPROULL, R. e SCHUMACKER, R. , "A Characterization of Ten Hidden-Surface Algorithm", Computing Surveys, Vol. 6, No. 1, Mar. 1974, pp 1-55

[5] WARNOCK, J. , "A Hidden-Surface Algorithm for Computer Generated Half-Tone Pictures", University of Utah Computer Science Departament, TR 4-15, 1969, NTIS AD 753 671

[6] WATKINS, G. , "A Real Time Visible Surface Algorithm", University of Utah Computer Science Departament, UTEC - CSc - 70 - 101, 1970, NTIS AD 762 004

[7] HORNUNG, C. , "A Method for Solving the Visibility Problem", IEEE CG&A, Jul. 1984, pp 26-33

[8] BRESENHAM, J. , "Algorithm for Computer Control of a Digital Plotter", IBM System Journal, Vol.4, No.1, Jan.1965, pp 25-30

[9] ASANO, T. , ASANO, T. e IMAI, H. , "Partitioning a Polygonal Region into Trapezoids", University of Tokio Research Memorandum, RMI 84-03, Fev. 1984

APÊNDICE I

Descrição da entrada de dados para as implementações

Um programa de aplicação dispõe de cinco procedimentos de acesso às três implementações do algoritmo. São elas : INICIALIZACAO, INSERE_POLIGONO, CRIA_OBJETO, REMOVE_OBJETO e ATUALIZA_CENA. O procedimento INICIALIZACAO recebe como parâmetros o driver e o modo relativos à placa gráfica utilizada e o diretório em que se encontra o arquivo de interface gráfica. O procedimento INSERE_POLIGONO recebe como parâmetros o número de vértices e os vértices do polígono, além de sua cor. Os vértices devem estar sempre representados por coordenadas da tela, isto é, coordenadas inteiras, limitadas às coordenadas máximas da placa gráfica utilizada. O procedimento CRIA_OBJETO é usado após terem sido fornecidos todos os polígonos que compõe o objeto, através do procedimento INSERE_POLIGONO. O parâmetro recebido indica o índice que o objeto criado passará a ter. O procedimento REMOVE_OBJETO remove da cena o objeto indicado pelo índice fornecido. O procedimento ATUALIZA_CENA faz as modificações na cena que foram causadas pelas operações CRIA_OBJETO e REMOVE_OBJETO que foram feitas desde a última operação ATUALIZA_CENA ou do procedimento INICIALIZACAO, se a operação ATUALIZA_CENA for a primeira.

Para que o programa de aplicação execute uma das implementações, deve ser incluído no seu início um dos arquivos de inclusão UNITS1.INC, UNITS2.INC ou UNITS3.INC, conforme a implementação desejada. A seguir o exemplo de um programa de aplicação em Turbo Pascal, onde são criados e depois

eliminados dois objetos com índice mesa e piso. O objeto piso é formado por um quadrilátero e o objeto mesa por um quadrilátero e quatro retas.

```
{$I units1.inc}

const mesa = 1;
      piso = 2;

type vetor = array[1..3] of real;
     matriz = array [ 1..4 , 1..4 ] of real;

var
      t : matriz;
      aux , aux1 : vetor;

{ tipo definido pelas 3 implementacoes }
      v , w : vetor_vertice;

      driver,modo,i,j : integer;
      arq : file of real;

{ multiplicacao de vetor por matriz }
procedure mult ( a: vetor ; b : matriz ; var c : vetor );
var i,j : integer;
begin
for i:=1 to 3 do
      begin
      c[i]:=0;
      for j:=1 to 3  do
               c[i] := c[i] + a[j]*b[j,i];
      c[i] := c[i] + b[4,i];
```

```
        end;
end;
  begin
  assign( arq , 'transf.dat');
  reset ( arq );

  { leitura da matriz com a transformacao de
    visualizacao }
  for i:=1 to 4 do
        for j:=1 to 4 do
                read ( arq , t[i,j] );

  { inicializacao para a placa grafica a ser
    utilizada }
  driver:= CGA ;
  modo:= CGAC1 ;
  inicializacao(driver,modo,'c:\bruno');

  { criacao do objeto piso }
  v[1,1]:=-10;
  v[1,2]:=-10;
  v[1,3]:=30;
  v[2,1]:=-10;
  v[2,2]:=120;
  v[2,3]:=30;
  v[3,1]:=120;
  v[3,2]:=120;
  v[3,3]:=30;
  v[4,1]:=120;
  v[4,2]:=-10;
  v[4,3]:=30;
```

```
{ transformacao das coordenadas dadas de acordo
  com a matriz de visualizacao e volta para
  coordenadas inteiras }
for J:=1 to 4 do
      begin
      for I:=1 to 3 do
               aux[I] := v[J,I];
      mult (aux , t , aux1 );
      for I:=1 to 2 do
               w[J,I] := round ( aux1[I] + 105);
      w[J,3]:=round(aux1[3] + 200 );
      end;

insere_poligono(w,4,2);

cria_objeto(piso);

atualiza_cena;

{ criacao do objeto mesa }
v[1,1]:=50;
v[1,2]:=50;
v[1,3]:=50;
v[2,1]:=50;
v[2,2]:=100;
v[2,3]:=50;
v[3,1]:=100;
v[3,2]:=100;
v[3,3]:=50;
v[4,1]:=100;
v[4,2]:=50;
```

```
   v[4,3]:=50;

   { transformacao das coordenadas dadas de acordo
     com a matriz de visualizacao e volta para
     coordenadas inteiras }

for j:=1 to 4 do
         begin
         for i:=1 to 3 do
                  aux[i] := v[j,i];
         mult (aux , t , aux1 );
         for i:=1 to 2 do
                  w[j,i] := round ( aux1[i] + 105);
         w[j,3]:=round(aux1[3] + 200 );
         end;

   insere_poligono(w,4,1);

   v[1,1]:=50;
   v[1,2]:=50;
   v[1,3]:=50;
   v[2,1]:=50;
   v[2,2]:=50;
   v[2,3]:=30;

   { transformacao das coordenadas dadas de acordo
     com a matriz de visualizacao e volta para
     coordenadas inteiras }

   for j:=1 to 2 do
         begin
         for i:=1 to 3 do
```

```
                aux[i] := v[j,i];
        mult (aux , t , aux1 );
        for i:=1 to 2 do
                w[j,i] := round ( aux1[i] + 105);
        w[j,3]:=round(aux1[3] + 200 );
        end;

insere_poligono(w,2,1);

v[1,1]:=50;
v[1,2]:=100;
v[1,3]:=50;
v[2,1]:=50;
v[2,2]:=100;
v[2,3]:=30;

{ transformacao das coordenadas dadas de acordo
  com a matriz de visualizacao e volta para
  coordenadas inteiras }

for j:=1 to 4 do
        begin
        for i:=1 to 3 do
                aux[i] := v[j,i];
        mult (aux , t , aux1 );
        for i:=1 to 2 do
                w[j,i] := round ( aux1[i] + 105);
        w[j,3]:=round(aux1[3] + 200 );
        end;

insere_poligono(w,2,1);

v[1,1]:=100;
```

```
v[1,2]:=100;
v[1,3]:=50;
v[2,1]:=100;
v[2,2]:=100;
v[2,3]:=30;

{ transformacao das coordenadas dadas de acordo
  com a matriz de visualizacao e volta para
  coordenadas inteiras }

for j:=1 to 4 do
      begin
      for i:=1 to 3 do
            aux[i] := v[j,i];
      mult (aux , t , aux1 );
      for i:=1 to 2 do
            w[j,i] := round ( aux1[i] + 105);
      w[j,3]:=round(aux1[3] + 200 );
      end;

insere_poligono(w,2,1);

v[1,1]:=100;
v[1,2]:=50;
v[1,3]:=50;
v[2,1]:=100;
v[2,2]:=50;
v[2,3]:=30;

{ transformacao das coordenadas dadas de acordo
  com a matriz de visualizacao e volta para
```

```
  coordenadas inteiras }

for J:=1 to 4 do
      begin
      for I:=1 to 3 do
            aux[I] := v[J,I];
      mult (aux , t , aux1 );
      for I:=1 to 2 do
            w[J,I] := round ( aux1[I] + 105);
      w[J,3]:=round(aux1[3] + 200 );
      end;

insere_poligono(w,2,1);

cria_objeto(mesa);

remove_objeto(piso);

atualiza_cena;

remove_objeto(mesa);

atualiza_cena;

Closegraph;
end.
```