1 Introdução

Sempre que uma imagem contém objectos ou superfícies opacas, aqueles que se encontram mais próximos do observador e directamente na linha de vista de outros objectos, vão bloquear a visão destes últimos. As superfícies ocultas devem ser removidas de modo a que a imagem representada no ecrã seja realista. A identificação e a remoção destas superfícies constituem o problema da **remoção das superfícies ocultas** (RSO) também designado por **cálculo de visibilidade**. A sua solução envolve a determinação da profundidade e da visiblidade para todas as superfícies que constituem a cena.

Existem vários algoritmos de Remoção de Superfícies Ocultas (RSO), sendo difícil dizer qual o melhor. Estes diferentes algoritmos têm a sua origem em determinados requisitos específicos. Os algoritmos que se descrevem de seguida representam as principais estratégias usadas. Todas eles podem ser generalizados e combinados, originando novos algoritmos apropriados para diferentes aplicações.

2 Estratégias Algoritmicas

São conhecidos vários algoritmos de visibilidade (hidden surface ou hidden line algorithms), a maioria dos quais data dos anos 70 e admitem objectos constituídos por faces planas. I. Sutherland *et al*., em [Sutherland74], apresentam dez algoritmos de visibilidade e estabelecem a seguinte taxonomia:

Algoritmos no Espaço Objecto
- À aresta
- Ao volume

Algoritmos no espaço Imagem
- À area
- Ao ponto (ou à linha de varrimento)

Algoritmos do tipo Lista de Prioridades
- À priori
- Lista dinâmica

Um algoritmo do tipo Espaço Objecto concentra-se na relação geométrica entre objectos no modelo de descrição da cena, com o fim de determinar que partes desses objectos são visíveis. Pelo contrário, um algoritmo no Espaço Imagem determina, para cada ponto da imagem, qual o objecto que é visível. Os algorimtos no Espaço Objecto trabalham com uma precisão arbitrária, podendo a imagem final ser aumentada várias vezes sem perda de qualidade, enquanto que os algoritmos no Espaço Imagem se vêem limitados a uma precisão bastante inferior, vulgarmente a precisão do próprio dispositivo de visualização. Os algoritmos no Espaço Objecto eram inicialmente orientados para dispositivos de visualização vectoriais ao passso que os do Espaço Imagem são, naturalmente, orientados para dispositivos *raster* e, portanto mais susceptíveis ao fenómeno do *aliasing*. Outro aspecto relevante na diferenciação destas duas classes de algoritmos reside no tipo de operações elementares e na sua complexidade algorítmica. Na realidade, as operações elementares no caso de um algoritmo do tipo Espaço Objecto

são complexas devido aos cálculos geométricos envolvidos e portnato de elevada carga computacional; ao contrário do que acontece com um algoritmo do tipo Espaço Imagem em que as operações elementares envolvem apenas simples manipulações das coordenadas espaciais dos pontos de imagem (por exemplo, comparação de profundidades entre dois *pixels*) e consequentemente a carga computacional necessária é substancialmente menor. Já no que concerne à complexidade algorítmica o desempenho realizado por estas duas classes se inverte. Assim, para uma cena de n objectos a ser desenhada num dispositivo de visualização com p *pixels*, a complexidade algorítmica de um algoritmo do tipo Espaço Objecto é n x n, ao passo que a do algoritmo Espaço Imagem é n x p.

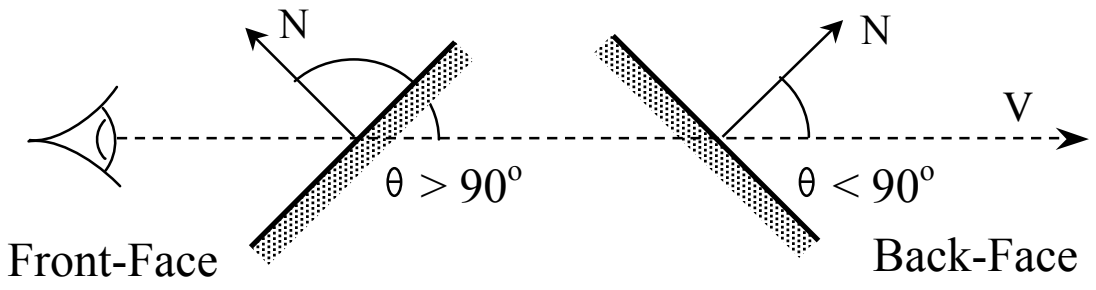
Os algoritmos do tipo Lista de Prioridades ficam situados entre os dois casos anteriores, dado que funcionam parcialmente em cada espaço. Os cálculos de profundidade são efectuados com precisão elevada, enquanto que as imagens são calculadas com a resolução disponível no sistema de visualização.

Do conjunto de algoritmos de visibilidade por I. Sutherland et al., salientam-se, pela ampla utilização que têm integrados em sistemas de Desenho Assistido por Computador (CAD Systems), os do tipo Lista de Prioridades Dinâmica e do tipo Imagem à Linha de Varrimento.

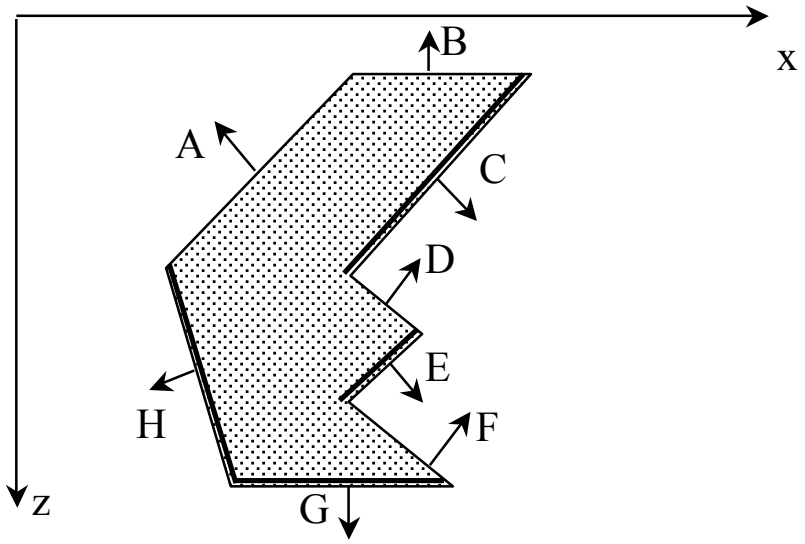
3 Técnicas de aceleração no Cálculo de Visibilidade

3.1 Remoção das Faces Posteriores (*Backface Culling*)

Considere-se o caso de um objecto o qual pode ser aproximado por um poliedro sólido: então as suas faces poligonais envolvem-no completamente, ou seja, possui um volume interior fechado. Assumindo que cada polígono constituinte é definido de forma a que a sua normal aponte para o exterior do objecto, então os polígonos cuja normal aponte na direcção oposta à do observador encontram-se completamente tapados pelos restantes polígonos. Estas faces, denominada de posteriores (*backfaces*), podem ser removidas. A sua determinação realiza-se através da análise do sinal resultante do produto interno entre a normal da face e o vector de visualização (vector VPN, que parte do "olho" do observador na direcção da cena) – ver figura xxxxx. Se o sinal é positivo então a face é posterior pelo que pode ser eliminada. Se o sinal é negativo então trata-se de uma face "voltada para a frente" e portanto vísivel do ponto de vista do observador.



A eliminação das faces posteriores, que poderá rondar cerca de metade do número total de polígonos da cena, constitui um método de aceleração bastante eficaz. No entanto, para a solução final do problema da visibilidade, esta técnica de aceleração deverá ser, quase sempre, acompanhada de um algoritmo que permita remover superfícies (partes de polígonos) ocultas. Mesmo na situação particular de uma cena constituída por um único objecto, o problema da visibilidade pode não ser solucionado só pela remoção das faces posteriores. Veja-se o caso da figura XXXX em que temos um único objecto mas concâvo. Existem faces que o teste de aceleração não as classifica como posteriores (faces D e F) mas que do ponto de vista do utlizador estão ocultos por outras faces.



3.2 Coerência

De modo a reduzir a quantidade de cálculo algorítmico e, consequentemente, obter uma maior eficiência, tenta-se ganhar alguma vantagem das relações de semelhança e dependências, designadas por **coerências**, entre os diferentes elementos que compõem uma cena. O uso das coerências simplifica os cálculos na medida em que estes tornam-se incrementais em vez de serem absolutos.

Apresentam-se, abaixo, alguns exemplos de coerência:

1. Coerência de linha de varrimento: se um pixel numa determinada linha de varrimento pertence a um polígono, muito provavelmente os pixeis adjacentes igualmente pertencerão ao mesmo polígono.
2. Coerência de aresta ou de lado: se uma aresta de um polígono é intersectada pela actual linha de varrimento, então é bastante provável que essa mesma aresta seja intersectada nas próximas linhas de varrimento.
3. Coerência de área: uma pequena área de uma imagem estará contida provavelmente no interior de um único polígono.
4. Coerência espacial: certas propriedades de um objecto podem ser determinadas pela sua **extensão**, isto é, uma figura geométrica que circunscreve o objecto dado. Normalmente, a geometria da extensão é a de um rectângulo ou a de um sólido rectangular (também designado por caixa envolvente – *bounding box*).

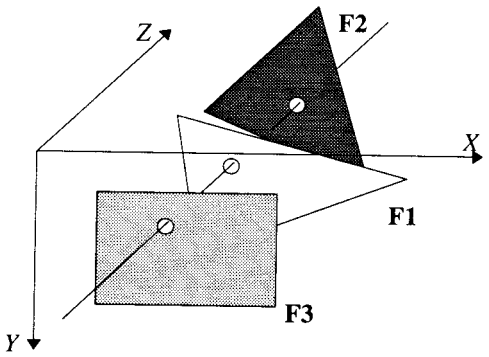
A coerência de linha de varrimento é usada com vantagem na rasterização 2D de polígonos.
A coerência espacial é usada muitas vezes como um passo de pré-processamento. Por exemplo, na verificação de intersecção de polígonos, podem ser eliminados aqueles polígonos cujas extensões não se intersectam, operação esta bastante simples de realizar.

4 Algoritmo Z-Buffer

Este algoritmo classifica-se, na taxonomia de I. Sutherland et al. no grupo dos algoritmos no espaço imagem.

O algoritmo z-buffer, desenvolvido por Catmull [Cat74], tornou-se um dos mais utilizados algoritmos de visibilidade. Este facto deve-se a uma grande simplicidade que lhe permite uma fácil implementação quer em *software* quer em *hardware*.

O algoritmo necessita de duas áreas de memória, uma para a construção da imagem (designada por *frame-buffer*) e outra para armazenamento de profundidades Z (designada por *z-buffer*). Efectua o varrimento linha a linha dos polígonos, um de cada vez, numa ordem arbitrária. Se, num dado ponto, a profundidade Z de um polígono é menor que a profundidade armazenada anteriormente nas mesmas coordenadas , então o polígono é visível nesse ponto, e o algoritmo substitui , nas duas áreas de memória, os valores anteriores pelos valores calculados para o novo polígono, naquele ponto.



O conjunto de passos a realizar neste algoritmo são os seguintes:

```
Inicializa o Zbuffer com a profundidade máxima e Frame-
buffer com a cor de fundo
Para cada polígono
        Para cada ponto do polígono:
                pz = valor de z (x, y)
                Se pz < ReadZ (x, y):
                        WriteZ (x, y, pz);
```

O algoritmo z-buffer apenas lida com comparação de profundidades pelo que a sua realização é integrada no processo de rasterização de primitivas como, por exemplo, o algoritmo 3D de rasterização de polígonos com coerência de aresta.

5 Algoritmos à Linha de Varrimento

Os algoritmos de Romney [Rom70], de Bouknight [Bou69] e de Watkins [Watkins70] (conhecidos vulgarmente pela família de algoritmos de Watkins) são referidos por I. Sutherland et al. como sendo algoritmos de visibilidade no Espaço Imagem, ao ponto ou à linha de varrimento.

O tradicional algoritmo *z-buffer* funciona ao nível do polígono, ou seja, numa base em que os polígonos são rasterizados um a um. Consequentemente, o cálculo do valor final de um pixel tem de recorrer a **uma memória contendo informação de estado (cor e profundidade) referente a todos os pontos do ecrã**. Esta memória tem a vantagem de não depender da complexidade da cena; no entanto, tem o inconveniente da sua dimensão ser imposta pela resolução do ecrã.

Pelo contrário, um algoritmo de rasterização que efectue o seu processamento ao nível da linha de varrimento, ou seja, o cálculo de imagem é feito linha a linha, exige que a **memória para armazenar a informação dos pixels tenha, apenas, a dimensão de uma linha de varrimento** [Rogers85]. Em cada iteração, produz-se uma linha de pixels com a informação correcta pronta a ser enviada para o ecrã.

5.1 Realização algorítmica

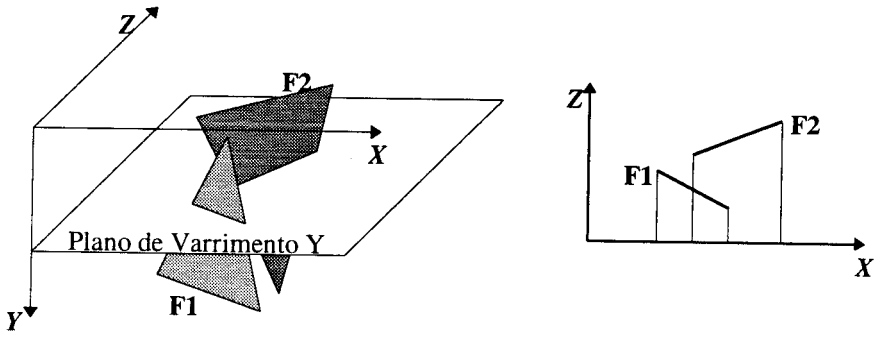
Estrutura Básica

Um algoritmo que funciona com base em linhas de varrimento consiste essencialmente em dois ciclos imbricados: um ciclo de varrimento segundo x contido num ciclo de varrimento segundo y (mudar para três ciclos)

- Varrimento segundo y

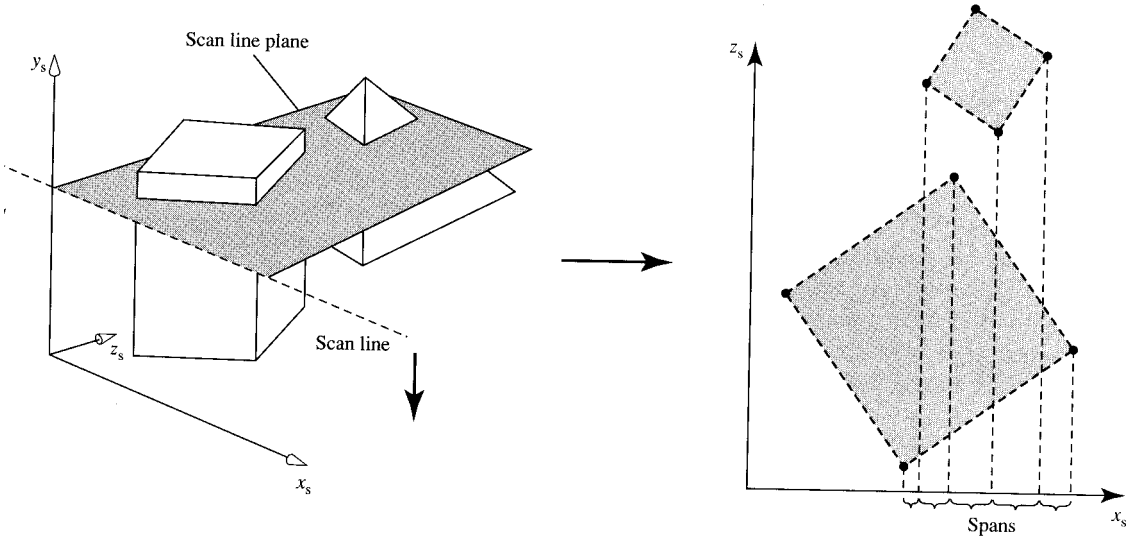
Para cada valor de y, por exemplo, $y = \alpha$, intersectam-se os polígonos da cena com o plano $y = \alpha$, o qual é paralelo ao plano xOz. As intersecções resultantes são segmentos de recta pertencentes a esse plano, os quais designaremos neste contexto de **segmentos**

de face. Atente-se que as extremidades de cada um destes segmentos de face representam as intersecções das arestas do correspondente polígono(ver fig ..) com o plano y =α. O troço de linha de varrimento compreendido entre duas arestas consecutivas denomina-se de *span* (fig…)



- Varrimento segundo x

No plano xOz correspondente à linha de varrimento y =α, para cada valor de x, por exemplo, x=β, intersectam-se os segmentos de face, calculados anteriormente em y=α, com a recta x=β. O resultado é uma sequência de pontos sobre a recta x=β, os quais terão de ser ordenados de acordo com os valores das suas coordenadas z. O ponto (β,α,z) com menor valor em z implica ser aquele que está mais perto do observador e portanto visível (relembre-se a adopção de um referencial de visualização orientado segundo a regra da mão esquerda). Nestas condições, o correspondente pixel será desenhado com a cor do polígono a que o ponto pertence.



Estruturas de Dados

As coerências de linha de varrimento e de aresta são usadas para potenciar o processamento no ciclo segundo a direcção vertical. Dado que esse ciclo em y constrói a lista das arestas potencialmente visíveis, em vez de reconstruir esta lista sempre que se procede a uma nova iteração ao nível das linhas de varrimento (cálculo absoluto), mantém-se a lista e é feita a sua actualização de acordo com a nova configuração espacial entretanto surgida (cálculo incremental), utilizando, precisamente, o conceito de coerência de aresta: actualização das arestas que permanecem na lista, inclusão de novas arestas e remoção das arestas que entretanto se esgotaram. À semelhança do que acontecia com o algoritmo de Rasterização de Polígonos 2D com Coerência de Aresta, utiliza-se uma tabela denominada de Tabela de Aresta Activas (TAA). Para auxiliar a construção e a manutenção da TAA recorre-se uma estrutura de dados designada de Tabela de Arestas (TA) que constitui um repositório das arestas dos polígonos que constituem a cena. O significado desta estrutura é diferente daquele que caracteriza a TA de um simples algoritmo de Rasterização de Polígonos 2D com Coerência de Aresta: enquanto que, neste último, a TA contém somente as arestas do polígono a ser processado, a **TA de um algoritmo à linha de varrimento contém informação de todas as arestas de todos os polígonos**. Deste modo, ao ser inserida uma aresta nesta TA de acordo com o valor da ordenada mínima, há que providenciar, para além dos tradicionais parâmetros (a abcissa de intersecção com a linha de varrimento, a ordenada máxima da aresta, o inverso do declive e os incrementos na cor), um outro campo que contenha um identificador do polígono a que a aresta pertence.
Analisemos agora com mais detalhe quais as estruturas de dados necessárias à realização de um algoritmo de rasterização à linha de varrimento.

1. Tabela de Arestas (TA) – contém todas as arestas não horizontais para todos os polígonos pertencentes à cena.
A TA é um vector de células cuja dimensão é determinada pela resolução vertical do viewport dispositivo de visualização. Cada uma destas células é uma lista de arestas. A colocação das arestas nas referidas células é realizada com base na menor coordenada y (ymin) de cada aresta. A aresta é, por sua vez, uma estrutura de dados que deverá conter a seguinte informação:
a) uma variável x_A que guarde o valor da abcissa de intersecção da aresta com a linha de varrimento corrente. Esta variável é inicializada com o valor da abcissa do extremo da aresta com a menor coordenada y (ymin).
b) O valor da ordenada máxima da aresta (ymáx).
c) O valor do inverso do declive da aresta (1/m).
d) Um identificador do polígono a que a aresta pertence.

2. Uma Tabela de Polígonos (TP) em que cada elemento desta lista possui:
   a) os coeficientes da equação do plano a que o polígono pertence; esta informação é fundamental para a determinação da profundidade z no pixel de posição (x,y).

b) Uma variável booleana IN_OUT, inicializada com o valor Falso (esta variável é activada consoante o facto de o *span* em causa pertencer ao polígono ou não);
c) Informação de côr do polígono (assume-se um sombreamento constante para simplificar)

3. A Tabela das Arestas Activas (TAA) que contém para cada valor y de varrimento, o conjunto das arestas intersectadas pelo correspondente plano de varrimento. Estas arestas são mantidas ordenadas por ordem dos valores crescentes de x_A

4. A Tabela de Polígonos Activos (TPA) que contém para cada x de varrimento:
    a) Apontadores para todos os polígonos da TP cuja variável IN_OUT esteja a verdadeiro;
    b) O número de polígonos nesta lista.

Programação do Algoritmo

1) Inicialização
    a) Inicialização da TAA como vazia
    b) Inicializar a memória de imagem com a cor de fundo
    c) Atribuir a y o índice da primeira célula não-vazia da TA

Repetir os passos 2 e 3 até que ambas as listas, TA e TAA, estejam vazias:

2) Ciclo de Varrimento segundo y
    a) Actualizar a TAA com a informação contida na célula y da TA
    b) Ordenar a TAA de acordo com os valores crescentes x_A

3) Ciclo de Varrimento segundo x: processar, da esquerda para a direita, cada aresta da lista TAA (ou dito de outro modo, determinar a cor de cada um dos *spans*) do seguinte modo:

a) Inverter o valor da variável booleana IN_OUT do polígono da TP que contém a aresta.

b) Verifica-se o número de elementos da TPA:

i) se igual a 1, atribui-se ao *span* corrente (conjunto de pixeis desde a aresta em questão até a próxima aresta) a cor do respectivo polígono activo;

ii) se maior do que 1, determinar o polígono visível a partir do menor valor de z, para cada polígono e para o pixel em causa. No caso de se impor que não existem polígonos inter-penetrantes, utiliza-se a coerência de *span*: a determinação dos valores z é realizada apenas num pixel (ínicio do *span*) pois garante-se que ao longo do *span* a relação entre profundidades mantém-se inalterável.

iii) se igual a zero, os pixels do *span* corrente não sofrem alteração

c) Actualiza-se a TAA após a última aresta ter sido processada:

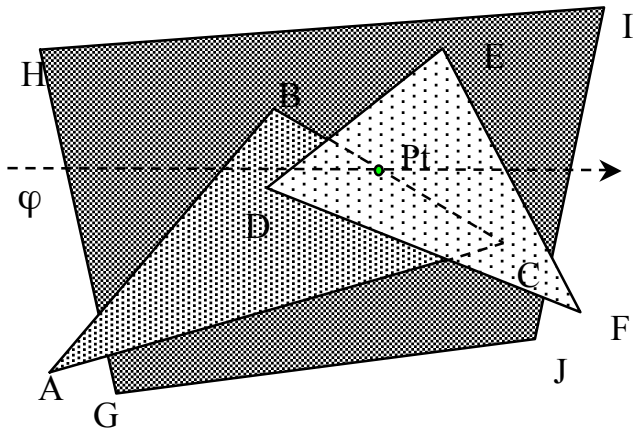
i) remover as arestas que se esgotam, ou seja arestas cujo ymáx é igual ao valor y da linha de varrimento menos uma unidade (devido ao problema dos vértices); se a lista ficar vazia e TA se encontrar igualmente vazia o algoritmo termina;

ii) para cada aresta que permanece na TAA, incrementar o valor da variável x_A de 1/m: esta é ovalor da abcissa de intersecção da aresta na próxima linha de varrimento;

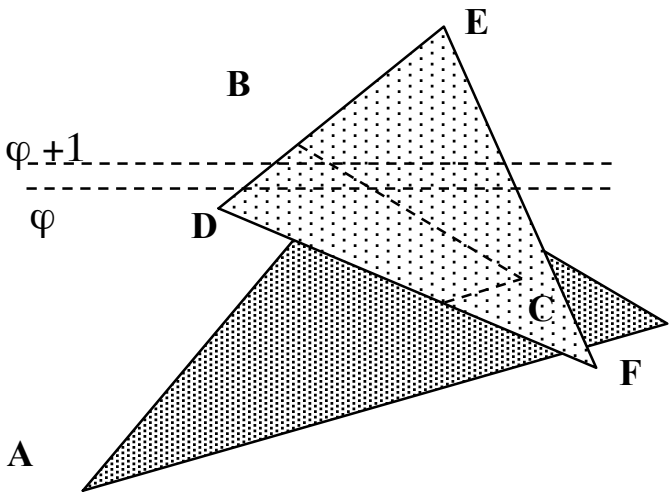
iii) ordenar a TAA de acordo com os valores crescentes x_A;

iv) incrementar y de uma unidade, que indica a próxima linha de varrimento e repetir o passo 2.

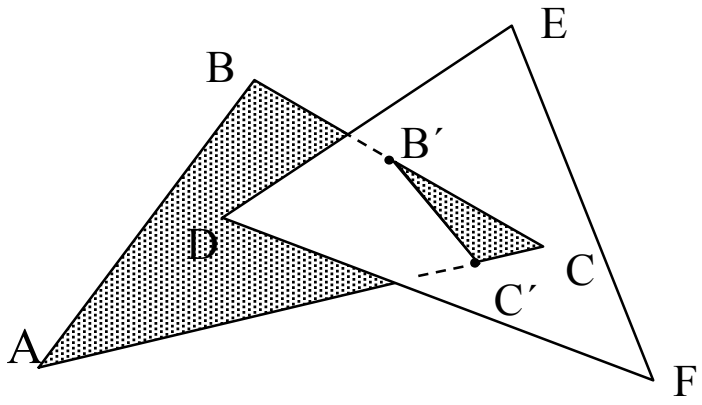
Se os polígonos não se inter-penetram, ver figura XXXX, alguns cálculos de profundidade podem ser evitados. Em pt, quando se abandona (ABC),pode assumir-se que a posição relativa de (DEF)e (GHIJ) não se alterou.



Por outro lado, com base na coerência de profundidade, se estão presentes na TAA de uma linha de varrimento (j+1) os mesmos lados que estavam na TAA da linha anterior (j), e pela mesma ordem, então as relações de profundidade entre polígonos permanecem inalteradas em todos os intervalos da linha (*spans*), ver figura XXXX.



Para polígonos interpenetrantes deve calcular-se a intersecção entre polígonos, criar um falso lado (B´C´), e com ele fragmentar um dos polígonos originais (ABC) em dois novos polígonos (ABB´C´e B´CC´) não penetrantes.



O plano de fundo da cena ("background") pode ser tratado de uma das seguintes formas:
a) Inicializar o "frame-buffer" com a cor de fundo
b) Modificar o algoritmo para atribuir cor de fundo a todos os pixels tratados nas situações em que todos os polígonos estão "out".
) Introduzir na cena um polígono de grandes dimensões, paralelo ao plano de projecção e mais afastado do observador que qualquer outro polígono (atribuir cor e sombreamento desejados).

5.2 Algoritmo de Atherton & Weiler

O algoritmo de Atherton & Weiler é posterior à publicação referida de I. Sutherland et al.. Classifica-se no grupo dos algoritmos no espaço objecto.

Como dados de entrada recebe a lista de polígonos da cena procede à sua ordenação em termos de profundidade. O polígono mais próximo do observador é utilizado para recortar todos os outros e produzir, deste modo, duas listas contendo os polígonos dentro e fora do polígono de recorte, respectivamente a lista interior e a lista exterior. Todos os polígonos da lista interior encontram-se situados atrás do polígono de recorte são eliminados pois são invisíveis. Cada um dos elementos da lista interior que estiver atrás do polígono de recorte deverá ser processado recursivamente de forma a recortar os elementos da lista interior. Quando esta subdivisão recursiva terminar, a lista interior deverá ser visualizada e o algoritmo deverá processar a lista exterior.

6 Algoritmos tipo Lista de Prioridades

A estratégia utilizada nos algoritmos tipo Lista de Prioridades consiste em estabelecer uma ordem de prioridades para o desenho dos polígonos. As prioridades são atribuídas aos polígonos baseadas nas suas profundidades. Uma vez determinada essa lista de polígonos, os seus elementos são "pintados" no ecrã pela respectiva ordem. A ideia fundamental que preside a esta estratégia é que os polígonos mais distantes são desenhados em primeiro lugar; polígonos mais próximos são pintados por cima de polígonos mais distantes, ocultando-os parcial ou totalmente.

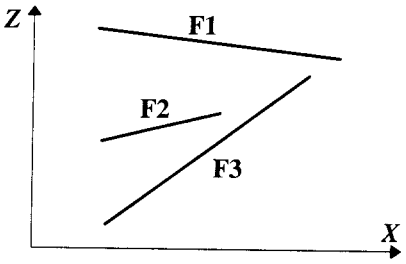
6.1 Algoritmo de Newell et al.

O algoritmo de Newell et al. [Newell72], é classificado por I. Sutherland et al. como um algoritmo do tipo Lista de Prioridades Dinâmica. Baseia-se na construção de uma lista de faces, ordenada segundo um critério de prioridade: uma face P tem prioridade sobre outra face Q, se P oculta total ou parcialmente Q. A extensão z de um polígono é a região entre os planos z=zmin e z=zmáx, em que zmin é a menor das coordenadas z de todos os vértices do polígono e zmáx é a maior.

O algoritmo divide-se em duas fases principais:

1. Pré-ordenação de todas as faces por ordem decrescente da profundidade do vértice mais afastado do ponto de observação, zmáx (figura 1);

2. Cálculo de prioridades finais entre faces.

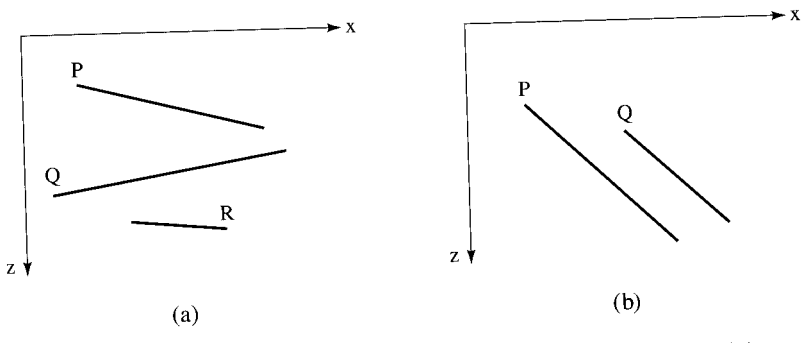


Pré-ordenação de Faces:
F2→F3→F1

Observador

Não sendo detectada, durante a primeira fase, qualquer sobreposição das extensões z entre faces, então a ordem resultante corresponde à ordem de prioridades finais desejada. Caso contrário, recorre-se a um algorito mais complexo que decide, na segunda fase quais as faces que têm prioridade superior, recorrendo, quando necessário, a operações de corte de uma face em outras menores. Na figura 1, existe sobreposição das extensões z entre as faces F2 e F3, e neste caso estão incorrectamente ordenadas.

Uma vez definida a prioridade entre faces, estas são enviadas para o dispositivo de visualização, começando pelas de menor prioridade.



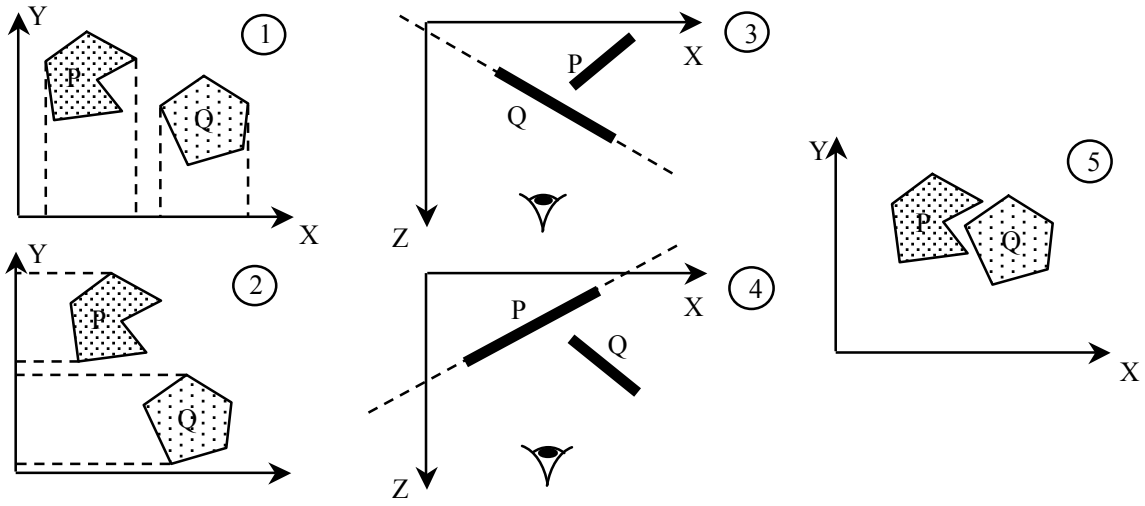
Resolução de Ambiguidades

Após a pré-ordenação podem ocorrer situações ambíguas, em que os intervalos de ocupação dos polígonos segundo zz´ se sobrepõem. Por vezes, a resolução dessas ambiguidades passa pela fragmentação dos polígonos.

Antes de rasterizar o primeiro polígono (P) da lista, ou seja, o mais afastado do observador, deve-se compará-lo com todos os polígonos (Q) cujos intervalos de ocupação segundo z sobreponham o intervalo z do primeiro. O processo de comparação tem por objectivo provar que P não obstrui a visibilidade de Q. Basta que uma das 5 condições seguintes se verifique (por ordem crescente de complexidade):

1. Os intervalos de ocupação de P e Q, segundo xx´ , não se sobrepõem
2. Os intervalos de ocupação de P e Q, segundo yy´ , não se sobrepõem
3. P encontra-se totalmente contido no semi-espaço definido pelo plano de Q, oposto àquele em que se encontra o ponto de observação da cena.
4. Q encontra-se totalmente contido no mesmo semi-espaço, definido pelo plano de P, em que se encontra o ponto de observação da cena.
5. As projecções de P e Q no plano (x,y) não se sobrepõem.

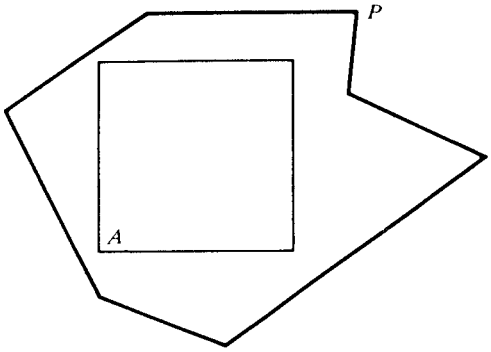
Se P passa no teste de não obstrução com todos os polígonos Q, pode ser rasterizado e o próximo polígono da lista passa a ser o novo polígono P. Se o teste falha com um dos polígonos Q, deve fragmentar-se P segundo um plano de corte coplanar com Q (ou vice-versa), descartar o polígono original, inserir na lista, por ordem decrescente de zmáx, os fragmentos resultantes e repetir o algoritmo.



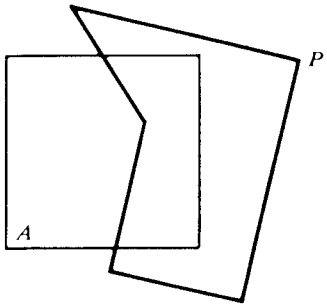
7 Algoritmo de Warnock ou Partição de Área

Os algoritmos de partição de área são procedimentos recursivos baseados numa estratégia a dois passos que primeiro determina quais os polígonos projectados que se sobrepõem numa dada área A do ecrã e, portanto, potencialmente visíveis, nessa área; segundo determina quais os polígonos realmente visíveis nessa área A. Se a decisão quanto à visibilidade não for trivial, esta área A, habitualmente rectangular, é sucessivamente subdividida até que essa decisão possa ser executada ou a área se resuma a um pixel.
Começando com a totalidade do ecrã como área inicial, o algoritmo divide, em cada passo, uma área em quatro áreas menores, gerando, desse modo, uma árvore quaternária, ver figura.XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

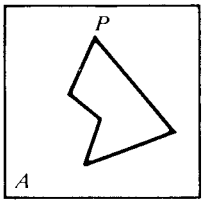
O processamento aproveita a coerência de área, classificando os polígonos P em relação a uma dada área A do ecrã, de acordo com as seguintes categorias:
1 – polígono envolvente – polígono que contém completamente a área, fig ....
2 – polígono intersectante - polígono que intersecta a área
3 - polígono contido - polígono que está completamente contido no interior da área
4 - polígono disjunto - polígono que é disjunto da área



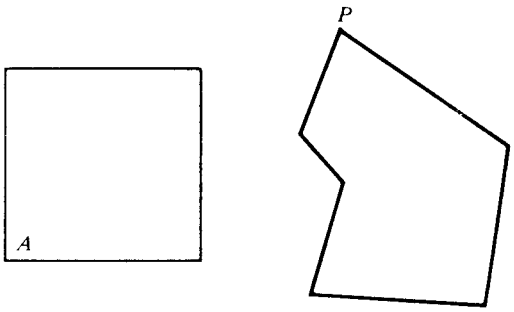
(*a*) *P* envolve *A*.



(*b*) *P* intersecta *A*.



(*c*) *P* está contido em *A*.



(*d*) *P* é disjunto de *A*.

A classificação dos polígonos que fazem parte de uma cena constitui o principal esforço computacional do algoritmo. Para uma dada área do ecrã, mantém-se uma lista dos polígonos potencialmente visíveis (PVPL – potentially visible polygons list), os das categorias 1, 2 e 3. Os polígonos disjuntos são claramente não visíveis. Note-se também que, por subdivisão de uma dada área do ecrã, polígonos envolventes e polígonos disjuntos continuam a ter essa classificação nas áreas recém-formadas. Portanto, apenas os polígonos contidos e os polígonos intersectantes necessitam de ser re-classificados.

A chave para uma computação eficiente da visibilidade reside no facto de um polígono não ser visível se estiver situado na rectaguarda de um polígono envolvente. Portanto pode ser removido da PVPL.

Algoritmo de partição

1. Inicializar a área como a totalidade do ecrã.
2. Criar a lista PVPL associada à área, ordenada de acordo com o zmin (a menor coordenada dos vértices do polígono). Classificar cada polígono. Com o uso de um qualquer algoritmo de recorte, já discutido anteriormente, um polígono de categoria 2 (polígono intersectante) pode ser recortado num polígono contido e num polígono disjunto. Portanto procede-se como se a categoria 2 fosse eliminada. Remover da lista os polígonos disjuntos.
3. Executar os testes de visibilidade:

i. Se todos os polígonos são disjuntos da área (lista PVPL vazia) atribuir a cada pixel a cor de fundo
ii. Se existe apenas um polígono na lista e pertence à categoria 3, contido, então pinta-se o referido polígono e atribui-se à restante àrea a cor de fundo.
iii. Se existe apenas um polígono na lista e é um polígono envolvente, então a área é pintada na sua totalidade com a cor do dito polígono
iv. Se existe um polígono envolvente que está mais próximo do observador que todos os outros polígonos, então isso significa que esses polígonos estão todos ocultos pelo polígono envolvente e portanto a área pode ser pintada com a cor deste último.
v. Se a área é o pixel (x, y) e não se aplicam os passos 1, 2, 3 e 4, determina-se então a coordenada z nesse pixel para todos os polígonos pertencentes à lista PVPL. Ao pixel atribui-se então a cor do polígono com a menor coordenada z.

4. Se não ocorreu nenhum dos casos anteriores, então subdivide-se a área em quatro partes iguais. Para cada área resultante da subdivisão retoma-se o passo 2.