COMPUTAÇÃO GRÁFICA #07

**REMOÇÃO DE FACES ESCONDIDAS**

**MODELOS DE ILUMINAÇÃO GLOBAL**

* Polígonos e Preenchimento de Polígonos (*scan line convertion*)
* Algoritmo de Visibilidade por Prioridade
* Algoritmo de Eliminação de Faces Ocultas pelo Cálculo da Normal
* Algoritmo de Visibilidade por Proximidade (*Z-Buffer*)
* Modelos de Iluminação Global (*Ray-Tracing, Ray-Casting, Radiosity)*
* Modelo *Phong*

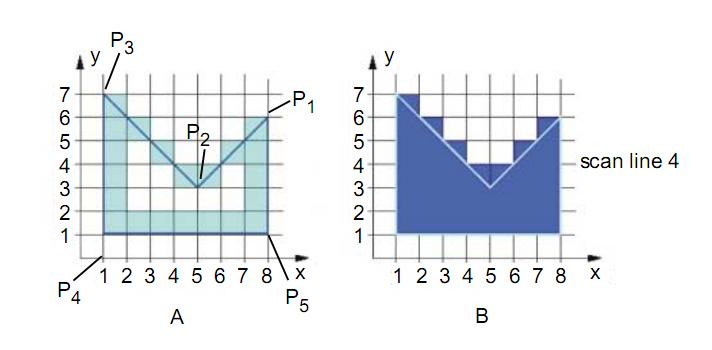
O principal objetivo deste módulo é familiarizar os alunos com os tipos mais usuais de técnicas para remoção de faces escondidas na síntese de imagens de objetos tridimensionais. Para aprofundamento no tem, incluindo a realização dos algoritmos, sugerimos a leitura dos capítulos de:

AMMERRAAL, L e ZHANG, K. *Computação Gráfica para Programadores Java.* Segunda Edição, Rio de Janeiro, LTC, 2008, págs 102-143 (caps. 6 e 7).

XIANG, Z e PLASTOCK, R. *Computer Graphics*. Second Edition, New York, Schaum’s Outline Series, McGraw-Hill. 2000. págs 197-272 (Caps. 10, 11 e 12).

Até aqui vimos o que seria necessário para modelar e projetar uma cena descrita no SISTEMA DE REFERÊNCIA DO UNIVERSO na VIEWPORT do SISTEMA DE REFERÊNCIA DO DISPOSITIVO em modelo ARAMADO (*wire frame*), ou seja, onde todos os vértices e arestas estão visíveis. O próximo passo seria, naturalmente, incluir nos algoritmos a REMOÇÃO DE FACES OCULTAS AO OBSERVADOR (*HSR – Hidden Surface Removal*). Não é uma tarefa muito seimples, pois a maioria dos algoritmos se utiliza de conceitos de GEOMETRIA ANALÍTICA e ÁLGEBRA LINEAR para este fim.

*Grosso modo* podemos subdividir as ideias de gerar algoritmos de remoção de faces ocultas em três ou quatro categorias principais, todas envolvendo o conceito de PREENCHIMENTO DE POLÍGONOS (*scan line convertion*). Uma vez que ainda não abordamos as técnicas mais importantes de RASTERIZAÇÃO, devemos supor que sabemos determinar quais pixels preenchem o espaço delimitado por um conjunto de arestas que formam uma face (figura 1)



**Figura 1.** *Scan line Convertion* (conversão por varredura).

POLÍGONOS são figuras planas fechadas formadas por segmentos de retas concatenadas. O polígono mais simples é o TRIÂNGULO e quase todos os aplicativos de modelagem e jogos utilizam a representação por FACES TRIANGULARES. A razão para isso é simples. Primeiro, é sempre possível descrever qualquer polígono como um conjunto de triângulos e um triângulo com vértices ABC, por sua vez, pode ser visto como dois vetores e . Pela GEOMETRIA ANALÍTICA sabemos que a EQUAÇÃO DO PLANO ABC será dada pelo PRODUTO ESCALAR e o VETOR NORMAL AO PLANO ABC () será dado pelo PRODUTO VETORIAL . Conhecer o VETOR NORMAL de uma superfície é de grande importância para sabermos sua orientação em relação à linha de visada do observador e os parâmetros de incidência de luz sobre ela. Neste caso, é importante termos em conta a orientação dos vértices de um polígono, seja ela horária ou anti-horária. A figura 2 a seguir ilustra dois tipos de POLÍGONOS: um POLÍGONO CONVEXO e um POLÍGONO NÃO CONVEXO. Já a figura 3 ilustra um polígono CONVEXO com 6 vértices e 6 arestas e DOIS VETORES quaisquer definindo uma região triangular do mesmo.

A principal vantagem da representação triangular é que ela garante uma face (superfície) plane e convexa.

1. Polígono CONVEXO
2. Polígono NÃO CONVEXO

A

B

C

D

E

F

A

B

C

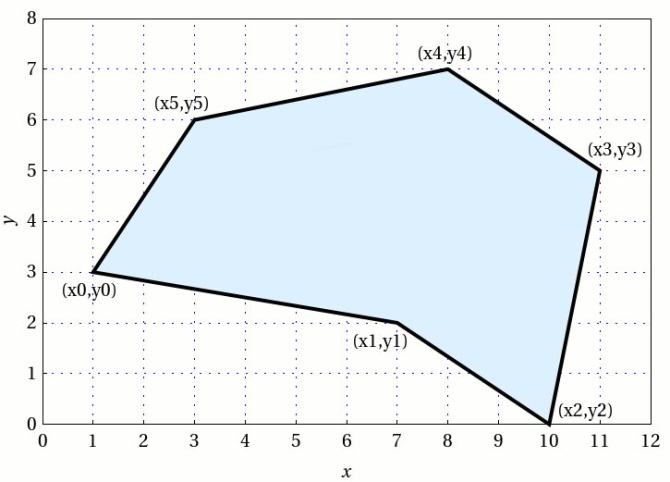
D

E

F

**Figura 2.** Polígono CONVEXO e Polígono NÃO CONVEXO. Observe a orientação dos vértices.

**Figura 3.** Um POLÍGONO em 2D com 6 vértices e 6 arestas:



**ALGORITMOS DE REMOÇÃO DE SUPERFÍCIES ESCONDIDAS**

**(*HIDDEN SURFACE REMOVAL*)**

Os algoritmos para remoção de superfícies escondidas devem ser capazes de:

1. Ler as coordenadas tridimensionais do objeto e armazená-las em forma de matriz.
2. Localizar no espaço 3D a posição do observador, através da qual definirá os parâmetros de visibilidade.
3. Calcular o vetor normal 3D de cada face do objeto.
4. Calcular o vetor da linha de visibilidade para cada face do objeto. Esse vetor é definido pela ligação de algum ponto da face ao observador.

**ALGORITMOS DE VISIBILIDADE POR PRIORIDADE**

Um primeiro algoritmo para esta finalidade desenha as faces na ordem inversa em que aparecem na linha de visada do observador (câmera). Algoritmos assim são denominados de “Algoritmos de Visibilidade por Prioridade” ou ALGORITMO DO PINTOR.

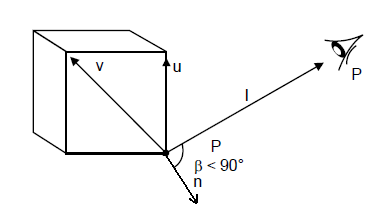
Em linhas gerais, se um objeto “A” bloqueia a visão de um objeto “B” e ambos os objetos encontram-se na mesma linha de visão do observador, então o objeto “B”, mais distante, será desenhado primeiro. Quando “A” for desenhado, este apagará a parte de “B” que tiver que ser ocultada por “A”. Então o algoritmo deverá:

1. Calcular a distância D de todas as faces poligonais da cena até o observador;
2. Ordenar todos os polígonos pelo valor de D;
3. Resolver eventuais ambiguidades no caso de duas faces com mesmo valor de D;
4. Desenhar primeiro os polígonos que tiverem mais distantes do observador.



**Figura 4.** Algoritmo do Pintor. Da esquerda para a direita: Primeiro se pinta as montanhas distantes, depois o prado e, finalmente, as árvores, que são os objetos mais próximos.

**ELIMINAÇÃO DE FACES PELO CÁLCULO DA NORMAL**

Para realizar o teste de visibilidade deve-se calcular o ÂNGULO ENTRE A NORMAL À FACE DO POLÍGONO E A LINHA DE VISADA DO OBSERVADOR. Se o valor absoluto desse ângulo o ângulo entre 90° e 180°, a superfície estará invisível. Por outro lado, a superfície estará visível se esse ângulo estiver entre –90° e 90° (figura 5, ao lado).

**ALGORITMO DE VISIBILIDADE POR PROXIMIDADE (*Z-BUFFER*)**

O algoritmo *Z-BUFFER* é um dos algoritmos de determinação de visibilidade de superfícies mais simples de se implementar, tanto em software quanto em hardware, e hoje é o mais popular dentre os algoritmos de HSR. É o algoritmo utilizado na grande maioria dos jogos e sistemas de renderização em tempo real. Embora simples e rápido, o algoritmo demanda um grande esforço computacional (memória e processamento). Requer a alocação de até dois buffers, ou matrizes, de em memória, com dimensões idênticas à tela de apresentação (*viewport*), normalmente denominados BUFFER DE IMAGEM e BUFFER DE PROFUNDIDADE.

Em linhas gerais, para cada pixel da VIEWPORT dispara-se um vetor paralelo às linhas de projeção interceptando cada face de cada objeto em cena. O ponto que tiver o valor de Z mais próximo do observador será o ponto desenhado.

*PSEUDOCÓDIGO PARA O Z-BUFFER*

Lista of polígonos {P1, P2, ..., Pn}

Matriz z-buffer[x,y] inicializado com menos “infinito”

Matriz Intensidade[x,y]

Início

para cada polígono P na lista de polígonos faça

{

para cada pixel (x,y) que intercepta P faça

{

calcule profundidade-z de P na posição (x,y)

se prof-z < z-buffer[x,y] então

{

Intensidade[x,y] = intensidade de P em (x,y)

z-buffer[x,y] = prof-z

}

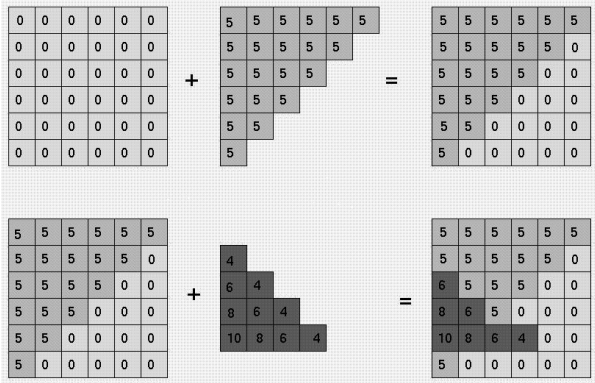
}

}

Desenhe Intensidade

fim

Na figura 6 é mostrado um exemplo da evolução do BUFFER DE IMAGEM quando dois polígonos se superpõem



**Figura 6: O Z-BUFFER.** A cor do pixel (BUFFER DE IMAGEM) é apresentada em TONS DE CINZA e seu valor Z (BUFFER DE PROFUNDIDADE) é representado pelo número. a) Adição ao z-buffer de um polígono com valor constante de profundidade. b) Adicionando outro polígono que intersecta o primeiro.

**MODELOS DE ILUMINAÇÃO GLOBAL (*RAY-TRACING, RAY-CASTING, RADIOSITY)***

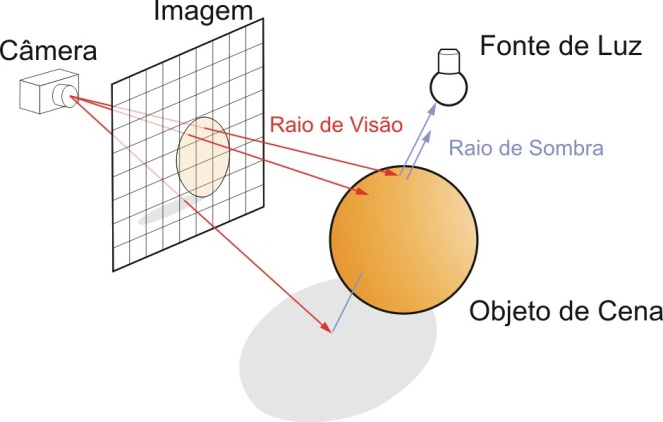
Por ser essencialmente geométrica, a principal limitação da técnica do Z-BUFFER está na dificuldade para se computar efeitos luminosos como REFRAÇÃO, REFLEXÃO e SOMBRAS.

Em cenas realistas, não são apenas as faces, arestas e vértices que devem ser mostrados, com suas cores calculadas a partir da iluminação direta de uma fonte luminosa, ou ocultados. Tais modelos podem ser referidos como MODELOS DE ILUMINAÇÃO LOCAL.

Os modelos que consideram todos os raios de luz emitidos, mesmo por fontes indiretas, são denominados MODELOS ILUMINAÇÃO GLOBAL e, desses, os mais populares são o *RAY TRACING* e a RADIOSIDADE. Esses modelos também permitem a inclusão de efeitos mais realistas com sombreamento parcial (penumbras) e refração (desvio da luz ao atravessar um objeto transparente).

***RAY TRACING***

Como o próprio nome indica, o *RAY TRACING* baseia-se na simulação de raios de luz que percorrem a cena e iluminam as faces dos objetos refletindo suas cores na direção de um observador (câmera). Contudo, ao contrário do que ocorre no mundo real, onde os raios de luz partem das fontes luminosas até atingirem o observador após refletirem nos objetos de cena, na técnica do *RAY TRACING,* DE CADA PIXEL da *viewport* dispara-se um RAIO DE VISÃO que interceptará todas as faces dos objetos de cena que se encontram pelo caminho e, a partir desses pontos de intersecção, dispara-se um RAIO DE SOMBRA até a(s) fonte(s) de luz que iluminam a cena (figura 7).



**Figura 7.** Raios de Visão e Sombra no Modelo *Ray Tracing*.

Observe que a primeira parte da técnica, a do disparo do RAIO DE VISÃO, é essencialmente a mesma do Z-BUFFER, embora o tratamento algébrico seja distinto. Mas é a segunda parte, a do RAIO DE SOMBRA e a possibilidade de se ver através de objetos com transparências e efeitos de refração, entre outros, é que torna a técnica muito mais exigente do ponto de vista do processamento. Apesar do altíssimo custo de processamento (cerca de 90% do tempo de processamento é gasto na determinação de intersecção do raio de visão com objetos) as cenas resultantes são muito mais realistas (de fato, hiper-realistas), o que torna a técnica ideal para a renderização de FILMES DE ANIMAÇÃO, mas pouco útil para jogos eletrônicos e renderizações em tempo real. Contudo, com o lançamento de *hardwares* mais rápidos e com maior capacidade de memória, já começa a ser viável o uso do *Ray tracing* em tempo real. Procure na internet por “ray tracing em tempo real”.

Quando a técnica (modelo) do *RAY TRACING* é empregada apenas para o cálculo de superfícies ocultas, como no *Z-BUFFER*, ela é denominada de *RAY CASTING*.

**RADIOSIDADE**

A RADIOSIDADE é uma técnica para aumentar o realismo de uma cena povoada de objetos...

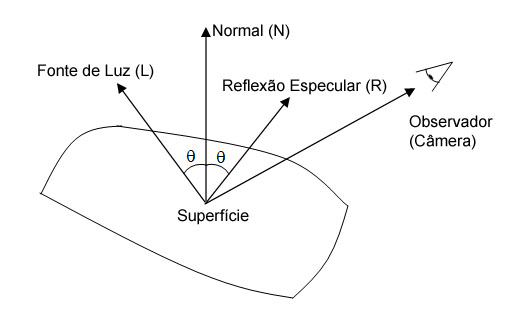
A radiosidade consiste na troca de luminosidade ocorrida entre os objetos da cena, considera a reflexão do raio de luz nos objetos (figura 8).

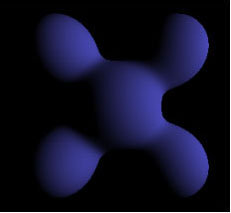
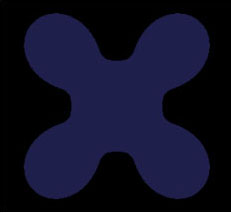


Figura 8. Radiosidade

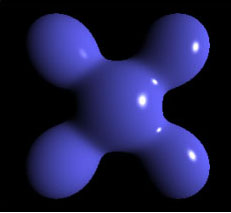
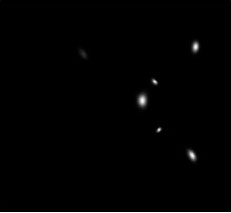
MODELO *PHONG*.

Ao incidir sobre uma superfície opaca a luz pode ser ABSORVIDA, ou REFLETIDA. A luz é refletida em duas formas básicas: a REFLEXÃO DIFUSA e a REFLEXÃO ESPECULAR. O MODELO *PHONG* é uma aproximação bastante boa para contabilizar esses dois tipos de reflexão. Trata-se de um MODELO DE ILUMINAÇÃO LOCAL baseado em três componentes de iluminação de uma superfície: a iluminação AMBIENTE, DIFUSA e ESPECULAR. A primeira componente, a iluminação AMBIENTE contabiliza como seria a luz refletida pela superfície caso o objeto estivesse iluminado indiretamente, pela luz difusa no ambiente em que ele se encontra. Trata-se de abordagem simplista da RADIOSIDADE. A segunda componente, a iluminação DIFUSA assume que a energia luminosa reflete na superfície igualmente em todas as direções, sendo ponderada em função do ponto de vista do observador. A terceira e última componente, a iluminação ESPECULAR é contabilizada segundo a primeira lei da reflexão da luz, que propõe que o ângulo incidente e o ângulo refletido de um raio de luz em relação a normal são iguais. A combinação dessas três componentes produz um gradiente de iluminação que se aproxima muito bem dos reflexos naturais que vemos, sobretudo, em superfícies curvas (figura 9).





AMBIENTE DIFUSA



ESPECULAR *PHONG*

**Figura 10.** Modelo de Iluminação *PHONG:* AMBIENTE+DIFUSA+ESPECULAR.

Há, ainda, uma quarta componente que é a luz EMITIDA pela superfície, mas esta pode ser incluída como parcela da luz ambiente.

EXERCÍCIOS RESOLVIDOS

I

II

III

Fonte de Luz

Plano de Projeção

(tela)

Plano

Objeto

R1.O desenho ao lado representa uma cena, composta de uma fonte de luz, um objeto (esfera) e um plano, que será renderizada com a técnica do *ray tracing* usando uma projeção cilíndrica ortogonal, ou ortográfica. Assinale a alternativa que indica corretamente quais, ou qual, dos 3 pontos marcados no plano de projeção (*pixel*) correspondem a um ponto de SOMBRA.

(a) I (b) II (c) III (d) I e II (e) I e III

*RESP.: Alternativa (b). Traçando RAIOS DE VISÃO partindo de cada pixel (I, II e III), PARALELOS ENTRE SI (cilíndrica) e PERPENDICULARES ao plano de projeção (ortográfica) e em seguida os RAIOS DE SOMBRA até a fonte de luz, mapeia-se os pixels que corresponde aos pontos iluminados diretamente pela fonte.*

I

II

III

Fonte de Luz

Plano de Projeção

(tela)

Plano

Objeto

R2. Assinale a alternativa errada

(a) Um *pipeline* é um conjunto de processos encadeados de forma que a saída de um processo é utilizada como entrada do processo seguinte. A ordem correta do *pipeline* de visualização de objetos 3D corresponde à: transformação de câmera, recorte 3D, projeção, mapeamento para coordenadas de tela e rasterização de polígonos.

(b) Um *pipeline* é um conjunto de processos encadeados de forma que a saída de um processo é utilizada como entrada do processo seguinte. No *pipeline* de visualização de objetos 3D um volume de visualização correspondente a um paralelepípedo determinado pela adoção de projeção perspectiva.

(c) A técnica de iluminação denominada *ray-tracing* utiliza o modelo de iluminação *Phong* no cálculo parcial da iluminação.

(d) Na técnica do *ray-tracing* o número de raios lançados depende do número de objetos da cena.

(e) Na técnica do *z-buffer* é possível realizar o cômputo das variáveis envolvidas de forma incremental.

*RESP.: Alternativa (b) pois é INCORRETO afirmar que um PARALELEPÍPEDO, cujos vértices são PARALELOS, corresponde a um volume de visualização determinado pela adoção de projeção perspectiva, ou CÔNICA.*

EXERCÍCIOS PROPOSTOS

P1. Considere as afirmativas abaixo e assinale a alternativa correta.

I. O *Z-Buffer* é um dos principais algoritmos de remoção de superfícies ocultas (*HSR - hidden surface removal*)

II. O algoritmo *Z-Buffer* serve com um algoritmo para iluminação global

III. O *Ray-tracing* é também usado simplesmente como um algoritmo de determinação da visiblidade dos elementos de cena. Neste caso, é denominado *Ray-casting*.

(a) Apenas I é correta (b) Apenas II é correta (c) Apenas I e III são corretas   
(d) Todas são corretas. (e) Todas são incorretas.

*RESP. (c)*

P2. Julgue se as afirmativas abaixo são verdadeiras (V) ou Falsas (F) e assinale a alternativa que corresponde à sequência V/F correspondente.

( ) Sobre a técnica conhecida como *Z-buffer* não se pode afirmar que seja possível realizar o cômputo das variáveis envolvidas de forma incremental.

( ) No algoritmo *Z-buffer* as primitivas geométricas não precisam estar ordenadas de acordo com a distância em relação ao observador.

( ) O algoritmo *Z-Buffer* serve com técnica para detecção de colisão.

( ) As dimensões do *Z-buffer* são dependentes das dimensões do *frame buffer,* ou *buffer* de imagem.

( ) A técnica de *Z-buffer* utiliza ordenação de primitivas para determinação dos pixels visíveis.

(a) V,V,V,V,V (b) F,F,F,F,F (c) V,F,V,F,V (d) F,V,F,V,F (e) F,F,F,V,V.

*RESP.: (d)*