Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Computação Gráfica
AD2 - 2° semestre de 2007.

1) Determine a matriz de transformação 4x4 $\mathbf{M_{obj}}$, em coordenadas homogêneas, que representação a mudança do sistema de coordenadas do mundo para o sistema de coordenadas da câmera (1.0 ponto).

   A matriz L_{at} que faz a mudança de coordenadas do sistema do mundo para o sistema da câmera é dada pelo produto das matrizes R_{ew} e T_{ew}.

$$\boldsymbol{L_{at} = R_{ew} \circ T_{ew}} = \begin{bmatrix} \boldsymbol{x}_{ex} & \boldsymbol{x}_{ey} & \boldsymbol{x}_{ez} & 0 \\ \boldsymbol{y}_{ex} & \boldsymbol{y}_{ey} & \boldsymbol{y}_{ez} & 0 \\ \boldsymbol{z}_{ex} & \boldsymbol{z}_{ex} & \boldsymbol{z}_{ex} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\boldsymbol{eye}_x \\ 0 & 1 & 0 & -\boldsymbol{eye}_y \\ 0 & 0 & 1 & -\boldsymbol{eye}_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   onde T_{ew} posiciona os sistema que define o referencial da câmera na origem e R_{ew} alinha os eixos do sistema da câmera com os eixos canônicos. Em outras palavras a matriz Lat, faz com que o sistema de referência passe a ser o sistema da câmera.

2) Seja a matriz

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   representando um movimento rígido do espaço em coordenadas homogêneas. Mostre que a transformação inversa é dada por

$$\boldsymbol{T^{-1}} = \begin{bmatrix} \boldsymbol{n}_x & \boldsymbol{n}_y & \boldsymbol{n}_z & -\langle \boldsymbol{p}, \boldsymbol{n} \rangle \\ \boldsymbol{o}_x & \boldsymbol{o}_y & \boldsymbol{o}_z & -\langle \boldsymbol{p}, \boldsymbol{o} \rangle \\ \boldsymbol{a}_z & \boldsymbol{a}_y & \boldsymbol{a}_z & -\langle \boldsymbol{p}, \boldsymbol{a} \rangle \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dica: as três primeiras colunas de T são formadas por vetores ortonormais (1.0 ponto).

A matriz T pode ser escrita como o produto

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{n}_x & \boldsymbol{o}_x & \boldsymbol{a}_x & \boldsymbol{p}_x \\ \boldsymbol{n}_y & \boldsymbol{o}_y & \boldsymbol{a}_y & \boldsymbol{p}_y \\ \boldsymbol{n}_z & \boldsymbol{o}_z & \boldsymbol{a}_z & \boldsymbol{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \boldsymbol{p}_x \\ 0 & 1 & 0 & \boldsymbol{p}_y \\ 0 & 0 & 1 & \boldsymbol{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{n}_x & \boldsymbol{o}_x & \boldsymbol{a}_x & 0 \\ \boldsymbol{n}_y & \boldsymbol{o}_y & \boldsymbol{a}_y & 0 \\ \boldsymbol{n}_z & \boldsymbol{o}_z & \boldsymbol{a}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

logo,

$$\boldsymbol{T}^{-1} = \left(\begin{bmatrix} 1 & 0 & 0 & \boldsymbol{p}_x \\ 0 & 1 & 0 & \boldsymbol{p}_y \\ 0 & 0 & 1 & \boldsymbol{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{n}_x & \boldsymbol{o}_x & \boldsymbol{a}_x & 0 \\ \boldsymbol{n}_y & \boldsymbol{o}_y & \boldsymbol{a}_y & 0 \\ \boldsymbol{n}_z & \boldsymbol{o}_z & \boldsymbol{a}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right)^{-1} = \begin{bmatrix} \boldsymbol{n}_x & \boldsymbol{o}_x & \boldsymbol{a}_x & 0 \\ \boldsymbol{n}_y & \boldsymbol{o}_y & \boldsymbol{a}_y & 0 \\ \boldsymbol{n}_z & \boldsymbol{o}_z & \boldsymbol{a}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & \boldsymbol{p}_x \\ 0 & 1 & 0 & \boldsymbol{p}_y \\ 0 & 0 & 1 & \boldsymbol{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} =$$

$$\begin{bmatrix} \boldsymbol{n}_x & \boldsymbol{n}_y & \boldsymbol{n}_z & 0 \\ \boldsymbol{o}_x & \boldsymbol{o}_y & \boldsymbol{o}_z & 0 \\ \boldsymbol{a}_z & \boldsymbol{a}_y & \boldsymbol{a}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\boldsymbol{p}_x \\ 0 & 1 & 0 & -\boldsymbol{p}_y \\ 0 & 0 & 1 & -\boldsymbol{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{n}_x & \boldsymbol{n}_y & \boldsymbol{n}_z & -(\boldsymbol{n}_x \boldsymbol{p}_x + \boldsymbol{n}_y \boldsymbol{p}_y + \boldsymbol{n}_z \boldsymbol{p}_z) \\ \boldsymbol{o}_x & \boldsymbol{o}_y & \boldsymbol{o}_z & -(\boldsymbol{o}_x \boldsymbol{p}_x + \boldsymbol{o}_y \boldsymbol{p}_y + \boldsymbol{o}_z \boldsymbol{p}_z) \\ \boldsymbol{a}_z & \boldsymbol{a}_y & \boldsymbol{a}_z & -(\boldsymbol{a}_x \boldsymbol{p}_x + \boldsymbol{a}_y \boldsymbol{p}_y + \boldsymbol{a}_z \boldsymbol{p}_z) \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \boldsymbol{n}_x & \boldsymbol{n}_y & \boldsymbol{n}_z & -\langle \boldsymbol{p}, \boldsymbol{n} \rangle \\ \boldsymbol{o}_x & \boldsymbol{o}_y & \boldsymbol{o}_z & -\langle \boldsymbol{p}, \boldsymbol{o} \rangle \\ \boldsymbol{a}_z & \boldsymbol{a}_y & \boldsymbol{a}_z & -\langle \boldsymbol{p}, \boldsymbol{a} \rangle \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3) Determine a matriz de projeção 4x4 que preserva a ordem relativa dos pontos em coordenadas de tela 3D (1.0 ponto).

A matriz de projeção $\boldsymbol{P} = \begin{bmatrix} \boldsymbol{n} & 0 & 0 & 0 \\ 0 & \boldsymbol{n} & 0 & 0 \\ 0 & 0 & \boldsymbol{n} & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$

leva as coordenadas de profundidade de todos os pontos visíveis dentro do volume de visualização no plano *near*, o que faz com que percamos a ordem relativa entre os elementos projetados.

Entretanto, é possível corrigir este problema determinando-se uma matriz tal que a profundidade z''= z'/w, resultante da coordenada projetada z', seja uma função monotônica da profundidade z original do ponto. Para isto precisamos determinar os coeficientes α e β na matriz abaixo, cujo produto da terceira linha pelo vetor de coordenadas especifica a equação a direita, que descreve uma função monotônica (verificar o gráfico da função ou se não há troca de sinais na primeira derivada).

$$\boldsymbol{P}' = \begin{bmatrix} \boldsymbol{n} & 0 & 0 & 0 \\ 0 & \boldsymbol{n} & 0 & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{bmatrix} \qquad z'' = \frac{z'}{w'} = \frac{\alpha z + \beta}{-z}$$

Para determinar α e β lembremos que os valores de profundidade dos pontos no plano *near* e *far* devem ser preservados (-n e –f, respectivamente), assim chegamos ao seguinte sistema:

$$\frac{\alpha(-\boldsymbol{n})+\beta}{-(-\boldsymbol{n})} = -\boldsymbol{n}, \text{ valor de z'' para o plano near}$$
$$\frac{\alpha(-\boldsymbol{f})+\beta}{-(-\boldsymbol{f})} = -\boldsymbol{f}, \text{ valor de z'' para o plano far}$$

A solução do sistema acima indica que α= n+f e β = nf, donde chegamos a matriz:

$$\boldsymbol{P}' = \begin{bmatrix} \boldsymbol{n} & 0 & 0 & 0 \\ 0 & \boldsymbol{n} & 0 & 0 \\ 0 & 0 & \boldsymbol{n}+\boldsymbol{f} & \boldsymbol{nf} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

4) Descreva o algoritmo de *Sutherland* e *Hodgman* para recorte de polígonos. Considere que a janela de recorte é dada por uma região retangular (0.5 ponto).

O algortimo de *Sutherland* e *Hodgman* é semelhante ao algoritmo de *Cohen-Sutherland*. O polígono é recortado sucessivamente contra todos os lados da figura de recorte, ou seja, a janela de recorte retangular. O algoritmo trabalha sobre uma lista circular de vértices, sendo que os vértices, e também as arestas que os conectam, são processados em seqüência e classificados contra o lado corrente do polígono de recorte.

```
Criar um lista circular le contendo os vértices do polígono.

PARA (cada lado l da janela)

    PARA (cada vértice v de le)

        SE (v está dentro da janela de recorte) ENTÃO
            Copiar v para a lista de saída ls
        FIM_SE

        SE (a aresta formada por v e o sucessor v' de v intersecta l) ENTÃO
            Calcular ponto de interseção v".
            Copiar v" para a lista de saída ls.
        FIM_SE

    FIM_PARA

    le ← ls

FIM_PARA
```

5) Escreva um algoritmo que determina a classificação de um ponto, através de um código de 4 bits, com relação às nove regiões determinadas por uma janela retangular (0.5 ponto).

```
Função classifica (x, y, xMin, xMax, yMin, yMax) retorna inteiro
{
   Declare classifica como inteiro

    classifica = 0

    SE ( y > yMax ) ENTÃO classifica ← (classifica ou 8)

    SE ( y < yMin ) ENTÂO classifica ← (classifica ou 4)

    SE ( x > xMax ) ENTÂO classifica ← (classifica ou 2)

    SE ( x < xMin ) ENTÂO classifica ← (classifica ou 1)

    retorne  classifica

}
```

Observação: o operador **ou** é um **ou lógico**.

6) Descreva em detalhes a equação de iluminação local e cada uma de suas componentes: ambiente, difusa e especular (0.5 ponto).

A equação da iluminação é expressa da seguinte forma:

iluminação total = iluminação ambiente + iluminação difusa + iluminação especular,

onde iluminação difusa + iluminação especular é o somatório destas iluminações de cada ponto de luz do cenário.

$$I_{Total} = I_a K_a C_a + f_{at} I_{luz} \left[K_d C_d (N \cdot L) + K_s C_s (R \cdot V)^{n_s} \right]$$

Assim temos:

I_a : intensidade de iluminação ambiente.

K_a : coeficiente de reflexão do ambiente.

C_a componente ambiental do material.

f_{at} : somatório de cada ponto de iluminação da cena.

I_{luz} : intensidade de cada ponto de iluminação da cena (intensidade difusa + intensidade especular).

K_d : coeficiente de reflexão especular

C_d : componente difuso do material.

N . L: Ângulo entre os vetores Normal e Fonte de luz

K_s : coeficiente de reflexão difusa

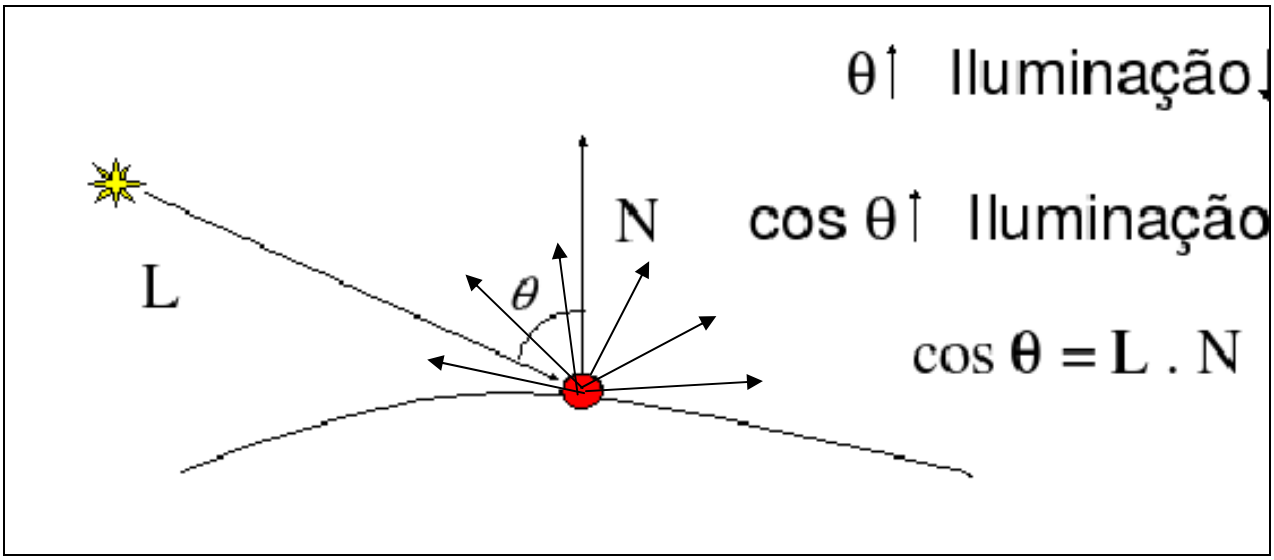
C_s : componente especular do material.

(R .V): Ângulo entre os vetores reflexão e do observador.

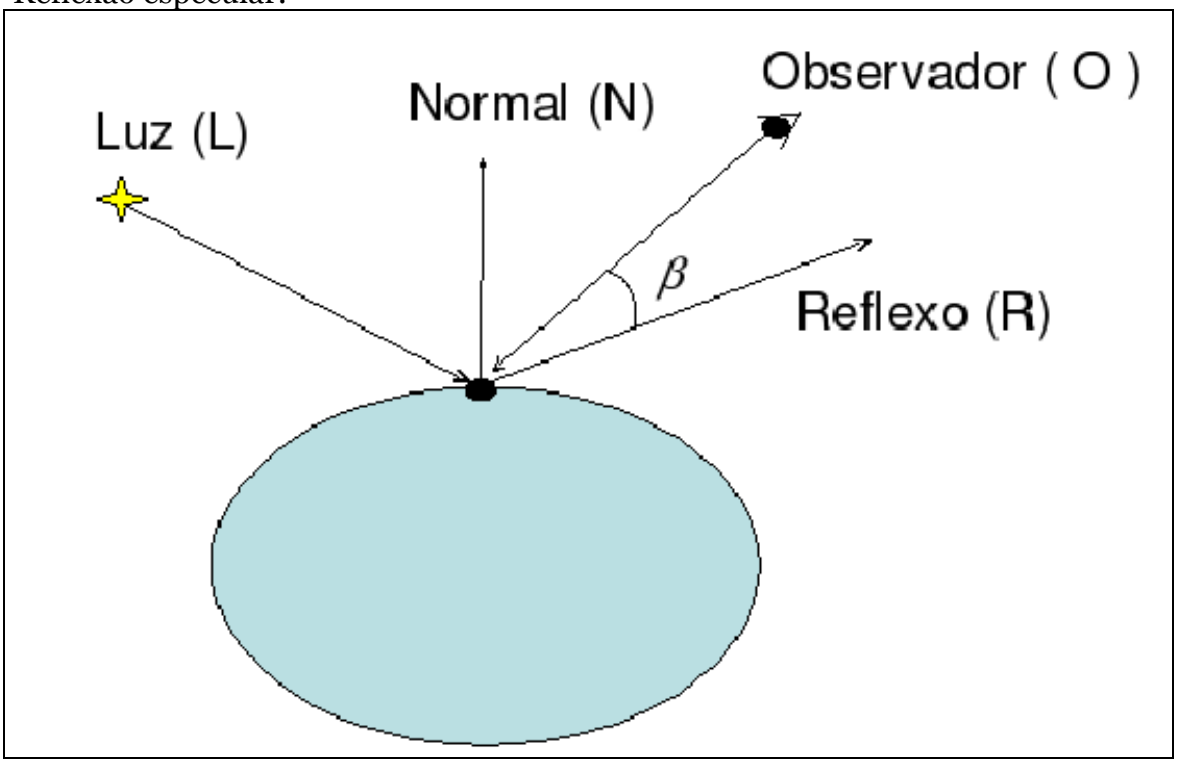
nv: expoente da componente especular (caracteriza o espalhamento do *highlight*)

7) Faça um desenho esquemático que ilustre como um raio incidente é refletido em um ponto de uma superfície correspondente a um difusor *lambertiano* (reflexão difusa). Faça um desenho análogo para o caso em que a superfície é composta por um material que possui reflexão especular (0.5 ponto).

Reflexão difusa:



Reflexão especular:



8) Descreva as diferenças entre os algoritmos de tonalização de *Gouraud* e *Phong*. Explique porque o algoritmo de *Gouraud* não modela bem os efeitos de reflexão especular (1.0 ponto).

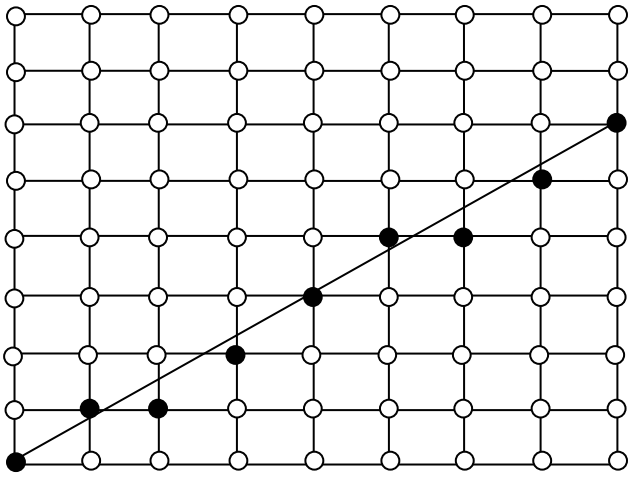
A primeira das diferenças é a iluminação especular. O algoritmo de Gouraud usa apenas a iluminação difusa, por outro lado, o algoritmo de Phong usa ambas iluminações: difusa e especular.

A outra diferença está no processo de reconstrução da função de iluminação. No algoritmo de Gouraud a função de iluminação, sem a componente especular, é calculada nos vértices usando a informação da normal definida em cada um deles. Em seguida, a função de iluminação é reconstruída no interior do polígono através de interpolação.

No algoritmo de Phong a normais dos vértices são usadas para reconstruir a normal sobre cada ponto no interior do polígono e, a partir destas normais reconstruídas, a função de iluminação é calculada.

O algoritmo de Gouraud não modela corretamente os efeitos de reflexão especular pois a iluminação especular não é utilizada. Caso fosse utilizada, surgiriam artefatos indesejáveis decorrentes da interpolação linear de uma função não-linear, no caso a componente especular da função de iluminação local.

9) Considerando o algoritmo do ponto-médio para rasterização de retas, determine quais *pixels* devem ser acessos no reticulado abaixo. Considere os *pixels* como sendo os elementos na interseção das retas do reticulado conforme ilustrado na figura abaixo (0.5 ponto).



Observação: no caso da reta passar pelo ponto médio foi escolhido o pixel inferior.

10) Descreva o algoritmo Z-buffering (0.5 ponto).

O Algoritmo *z-buffer* é o algoritmo mais utilizado para solucionar o problema de remoção de superfícies escondidas. O algoritmo armazena para cada *pixel* a distância existente entre o observador e a superfície mais próxima, a qual é a superfície visível. O *Z-Buffer* cria um array bidimensional contendo informação sobre a profundidade de cada *pixel* em um dado instante.

Durante o processo de rasterização, são gerados fragmentos que são candidatos a produzirem informações para um pixel. O teste de profundidade compara a coordenada de profundidade do fragmento com a coordenada de profundidade armazenada na posição correspondente *no array bidimensional* (o Z-buffer). Caso a coordenada de profundidade do fragmento seja menor (mais próxima da tela de projeção) que a coordenada já armazenada, escreve-se a informação de cor do fragmento, na posição correspondente no buffer de cores, ao mesmo tempo em que a coordenadas de profundidade (o valor de z) no Z-buffer é atualizada com o valor da profundidade do fragmento.

Abaixo descrevemos sucintamente os principais passos do algoritmo Z-buffering.

1. Criar e inicializar com a cor de fundo um *array* bidimensional (buffer de cor) que conterá as informações de cor de cada *pixel* da tela;

2. Inicializar o *array* de profundidades (Z-buffer) com o valor da profundidade máxima;

3. Calcular a coordenada z de cada fragmento gerado para os polígonos rasterizados;

4. Para cada fragmento, testar se a sua profundidade z é menor do que o valor armazenado na posição correspondente no z-buffer. Em caso positivo, atualizar o valor de profundidade no z-buffer e a armazenar a informação de cor no buffer de cores.

11) Descreva os processo de mapeamento direto e inverso. Quais as vantagens do mapeamento inverso em relação ao mapeamento direto (0.5 ponto).

No mapeamento direto, os valores dos pixels em uma imagem destino ID são calculados a partir dos valores da imagem de origem ID através da transformação $T:ID(i,j)\rightarrow IS(i,j)$. No mapeamento inverso, percorre-se os pixels da imagem destino, buscando os valores na imagem origem através da transformação inversa T^{-1}: $ID(i,j)\rightarrow IS(i,j)$.

A vantagem do mapeamento inverso é de que não surgem buracos na imagem destino transformações que envolvam um fator de ampliação. Além disso, quando houver um fator de redução, é possível utilizar esquemas de filtragem considerando-se uma vizinhança emtorno do pixel correspondente na imagem origem, efetuando-se deste modo uma amostragem mais apropriada.

12) Descreva as técnicas de mapeamento de rugosidade (*bump mapping*), de deslocamento (*displacement mapping*) e de ambiente (*environment mapping*) (0.5 ponto).

As técnicas de mapeamento de rugosidade (*bump mapping*), mapeamento por deslocamento (*displacement mapping*), e mapeamento de ambiente (e*nviroment mapping*), têm como objetivo aumentar o grau de realismo de um ambiente virtual sem exigir um aumento na malha poligonal que compõe o ambiente virtual.

Quando utilizamos uma fotografia de uma superfície áspera como mapa de textura, a superfície renderizada não fica muito correta, porque a direção da fonte de luz utilizada para criar o mapeamento é diferente da direção da iluminação do sólido, não gerando o efeito desejado.

Uma possível solução para este problema está na aplicação da técnica de *bump mapping* ou da técnica de *displacement Mapping*. A técnica de *bump mapping*, ou mapeamento por rugosidade, criada por (James F. Blim, 1978) utiliza duas texturas: a primeira é a textura que se deseja mapear sobre o objeto e a segunda contém um mapa de normais para cada ponto da primeira textura.

Aplicativos como GIMP e Photoshop possuem *plug-ins* para a construção de mapa de normais. A técnica consiste em aplicar uma "perturbação" sobre o vetor normal da superfície, exigindo o uso de uma textura com o mapa das normais. Esta "perturbação" nada mais é do que um deslocamento na posição real dos pontos da superfície, produzindo uma ilusão de relevo. Assim dada uma função imagem dada por $b(u,v)$, uma superfície paramétrica dada por $p(u,v)$ e um vetor normal no ponto p dado por $N(u,v)$, pode-se calcular o novo valor da normal como:

$$q(u,v) = p(u,v) + b(u,v) \times N(u,v)$$

$$N_1 = \frac{\partial q}{\partial u} \wedge \frac{\partial q}{\partial v}$$

$$N = \frac{N_1}{|N_1|}$$

Obtendo o novo valor da normal basta apenas substituir o valor da normal da equação da iluminação por este novo valor. Esta técnica produz um efeito indesejável que é a perda de detalhes da geometria que são mapeados.

Com o objetivo de resolver este problema é que a técnica de mapeamento por deslocamento (*displacement mapping*) foi desenvolvida (Cook 1984). Nesta outra técnica, o problema é contornado alterando geometria do objeto utilizando a intensidade da imagem. Assim como o vetor normal, a geometria da superfície também é "perturbada". Abaixo seque o novo calculo para o novo valor da normal, este valor deve ser substituído na equação da iluminação.

$$\boldsymbol{p}(\boldsymbol{u},\boldsymbol{v}) = \boldsymbol{p}(\boldsymbol{u},\boldsymbol{v}) + \boldsymbol{b}(\boldsymbol{u},\boldsymbol{v}) \times \boldsymbol{N}(\boldsymbol{u},\boldsymbol{v})$$

$$\boldsymbol{N} = \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}} \wedge \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{v}}$$

$$\boldsymbol{N} = \frac{\boldsymbol{N}\boldsymbol{b}}{\left|\boldsymbol{N}\boldsymbol{b}\right|}$$

A técnica de mapeamento de ambiente (e*nviroment mapping*), é uma técnica usada para realizar a renderização de objetos refletivo. O método foi proposto por Blinn & Newell em 1976 e permite simular efeitos de *Ray tracing* a baixo custo. É um mapeamento do ambiente sobre um objeto, no qual o próprio objeto está imerso, ou seja, reflete o cenário. A primeira forma de obter este efeito é tomando um cubo, o qual requer seis imagens de textura, uma para cada direção contendo as informações dos objetos que compõem o ambiente. Para cada vértice do polígono, um vetor de reflexão é calculado, através do qual é indicada uma posição em uma das seis imagens.

13) Descreva o algoritmo de traçado de raios (*Raytracing*) (0.5 ponto)

O algoritmo do *Ray tracing* simula a geometria ótica envolvida no trajeto dos raios de luz que viajam pelo espaço do cenário (algoritmo de iluminação global), embora o modelo funcione ao contrário do modelo físico, pois neste o raio de luz se origina no objeto e viaja até nossos olhos. Na técnica de *Ray tracing* é suposto que o raio se originou em nossos olhos e atinge o objeto (Figura 13.1). A inversão não altera a geometria ótica envolvida. Descrevemos abaixo seu funcionamento:

1. Trace um "raio" a partir do observador até a cena a ser representada através de um *pixel* da tela;
2. Determine qual é o primeiro objeto a interceptar esse raio;
3. Calcule a cor da superfície do objeto no ponto de interseção baseado nas características do objeto e na luz;
4. Se a superfície do objeto for reflexiva, calcule um novo raio a partir do ponto de interseção e na "direção de reflexão";
5. Se a superfície do objeto for transparente, calcule um novo raio (refratado) a partir do ponto de interseção.
6. Considere a cor de todos os objetos interceptados pelo raio até sair da cena ou atingir uma fonte de luz, e use esse valor para determinar a cor do *pixel* e se há sombras.

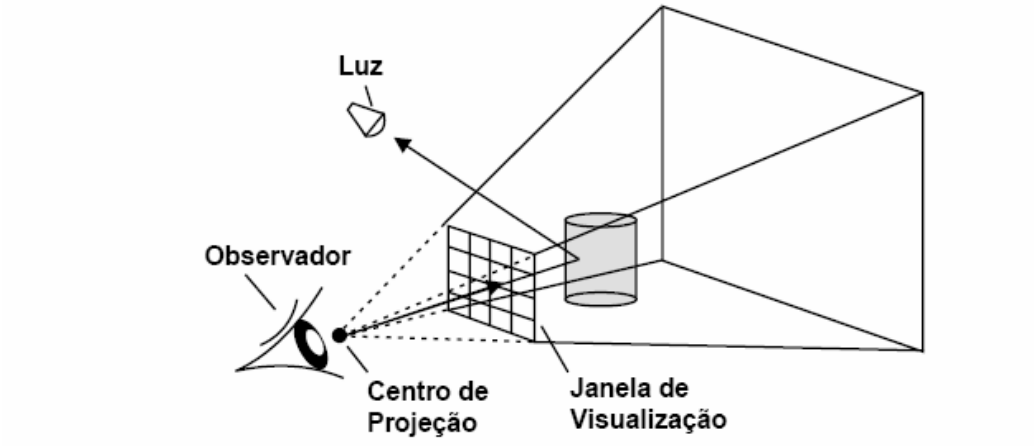


Figura 13.1

14) Descreva sucintamente os processos que compõem o *pipeline* de visualização. (0.5 ponto).

O *pipeline* gráfico é composto por 2 estágios bem definidos, o primeiro trata da geometria e a segunda da rasterização.

O estágio de geometria é responsável pelas operações sobre os vértices. Assim um objeto é transformado por uma seqüência de operações, caracterizando diferentes sistemas de coordenadas.

O objeto está originalmente em coordenadas chamadas de coordenadas do modelo. Cada objeto da cena virtual pode ser associado a uma transformação de modelagem que irá posicioná-lo no mundo virtual. Esta transformação é aplicada a todos os vértices e normais, levando ao espaço do mundo, comum a todos os objetos da cena; a câmera virtual está posicionada no espaço do mundo e para facilitar as etapas de iluminação, projeção e *clipping* esta câmera e todos os objetos são transformados pela transformação de visualização. O objetivo desta transformação é posicionar a câmera na origem e alinhá-la com os eixos ortogonais; este espaço é denominado espaço da câmera ou espaço do olho.

No estágio seguinte, o *pipeline* trata da iluminação, assim são aplicados os algoritmos de iluminação. O Estágio de *clipping* é seguinte ao de iluminação e projeção, e sua função é definir os objetos que vão passar para o estágio seguinte, ou seja, os objetos que estão completamente inseridos no volume de visualização são mantidos, os que estão completamente fora são descartadas e os que estão parcialmente inseridos são recortados. O estágio de mapeamento da tela é o seguinte. Assim os objetos têm suas coordenadas transformadas para as coordenadas da tela, ou seja, uma translação seguida de uma escala. A coordenada z é mapeada para uma coordenada do *z-buffer*, e as novas coordenadas x e y são chamadas coordenadas de tela. Toda esta informação é repassada para o estágio de rasterização.

O estágio de rasteriazação tem o objetivo de converter vértices projetados, com informações de cor, coordenadas de textura, etc., em *pixels* na tela, realizando operações sobre os mesmos. A primeira etapa de rasterização é produzir um conjunto de *pixels* a partir dos objetos renderizados. Sobre todos os *pixels* produzidos são realizadas operações a fim de determinar sua cor final. Entre estas operações está o acesso a texturas e operações aritméticas para combinar as diversas informações do *pixel*. Também é no estágio da rasterização que é executado o principal algoritmo de visibilidade, o *z-buffer*. Além do *z-buff*er há uma diversidade de outros *buffers* que auxiliam a elaboração de algoritmos que precisam armazenar informações de cada *pixel*.

15) O que são *key-frames*. Explique o processo de *in-betweening* (0.5 ponto).

Key-frames são os quadros que especificam o inicio e fim de uma transição suave em uma animação. O processo de *in-betweening* consiste na geração dos quadros intermediários, entre dois *key-frames,* necessários para a criação da ilusão de movimento.

16) O que é quantização e quais são suas aplicações. Descreva de forma sucinta o Algoritmo da Populosidade (0.5 ponto).

Quantização é uma transformação que visa discretizar o gamute de cores de uma imagem, além de reduzir o número de bits necessário para armazenar a informação de cor. As principais aplicações estão na compressão e transmissão de imagens.

Algoritmo da Populosidade:

1. Calcule o *histograma* da imagem.
2. Escolha os *k* níveis de quantização como sendo as *k cores mais freqüentes* (mais populosas).
3. A função de quantização *q(c)* atribui a cada cor c o nível de quantização *mais próximo* de *c* segundo o quadrado da métrica euclidiana.

Obs.: Em caso de empate fazer uma escolha aleatória ou considerar a vizinhança do *pixel*.