Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Computação Gráfica
AD2 - 1° semestre de 2008.

1) Determine a matriz de transformação 4x4 $\mathbf{M_{obj}}$, em coordenadas homogêneas, que representa 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}.

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

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

$$T^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\langle p,n \rangle \\ o_x & o_y & o_z & -\langle p,o \rangle \\ a_z & a_y & a_z & -\langle p,a \rangle \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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 à direita, que descreve uma função monotônica (verificar o gráfico da função ou se não há troca de sinais na primeira derivada).

$$\boldsymbol{P}' = \begin{bmatrix} \boldsymbol{n} & 0 & 0 & 0 \\ 0 & \boldsymbol{n} & 0 & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$z'' = \frac{z'}{w'} = \frac{\alpha z + \beta}{-z}$$

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 Phong e cada uma de suas componentes: ambiente, difusa e especular (0.5 ponto).

A equação phong é dada por

$I = Ka.Ia + Somatório[i=0,n] (K_dIi(N.L) + K_eI_i(O.R))$

Ka, Kd e Ke são as constantes do material, descrevendo como reagem a cada um dos componentes de iluminação. Ia é o coeficiente de iluminação ambiente. O somatório significa que será efetuado um cálculo de difuso e especular para cada uma das fontes de luz, de um total de n. I_i é a intensidade de iluminação da fonte de luz. N é a normal do ponto que se está calculando, O é o vetor do observador ao ponto da superfície em questão e R é o vetor de reflexo no ponto da superfície.

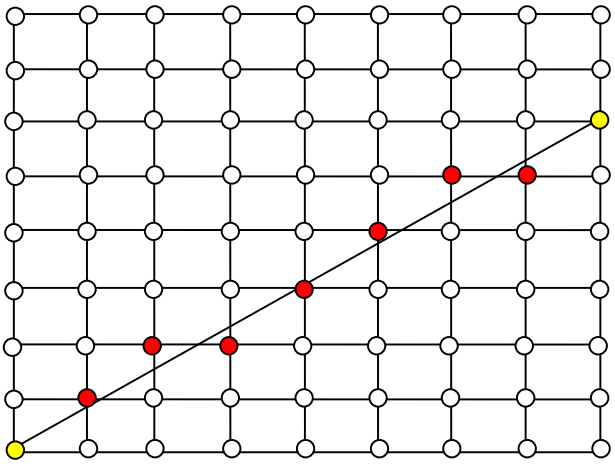
7) No modelo Phong, porque a iluminação ambiente é uma aproximação distante da iluminação ambiente real? (0.5 ponto)

Porque se está multiplicando por um valor que é constante para toda a cena, quando na realidade esta constante deveria ser para cada ponto da cena. Entretanto, é extremamente custoso calcular qual é este coeficiente para cada parte.

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).

O algoritmo Gouraud realiza o calculo de iluminação apenas para os vértices do polígono. A cor do interior do polígono será interpolada, sem realizar cálculo de iluminação algum. O modelo Phong calcula a iluminação para cada ponto visível. Os efeitos de especularidade requerem que sejam computados pixel a pixel do interior do polígono. O algoritmo de Gouraud não irá apresentar um resultado convincente porque interpola linearmente o resultado de uma função não linear calculada nos vértices, no caso da função utilizada pelo modelo de Phonr.

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).



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

O algoritmo se baseia no armazenamento dos valores de profundidade, associados aos pixels gerados pela rasterização de um polígono na tela, em uma área de memória temporária (o Z-Buffer). Inicialmente esta memória está inicializada com valores de profundidade muito distantes para todos os pixels. Durante a rasterização de um polígono no *framebuffer* (ou second Buffer), o algoritmo de Z-Buffer requer que seja feita uma consulta: o fragmento corrente tem um valor de profundidade maior ou menor do que o que está no Z-Buffer? Se for menor, então o pixel é atualizado com as informações do fragmento atual, descartando-se as informações anteriores (por exemplo, cor). Além disso, o valor de profundidade no Z-Buffer é atualizado com o novo valor, a profundidade do fragmento que foi considerado. Se for maior, o fragmento é desconsiderado.

11) Descreva os processos de mapeamento direto e inverso. Quais as vantagens do mapeamento inverso em relação ao mapeamento direto (1.0 ponto).

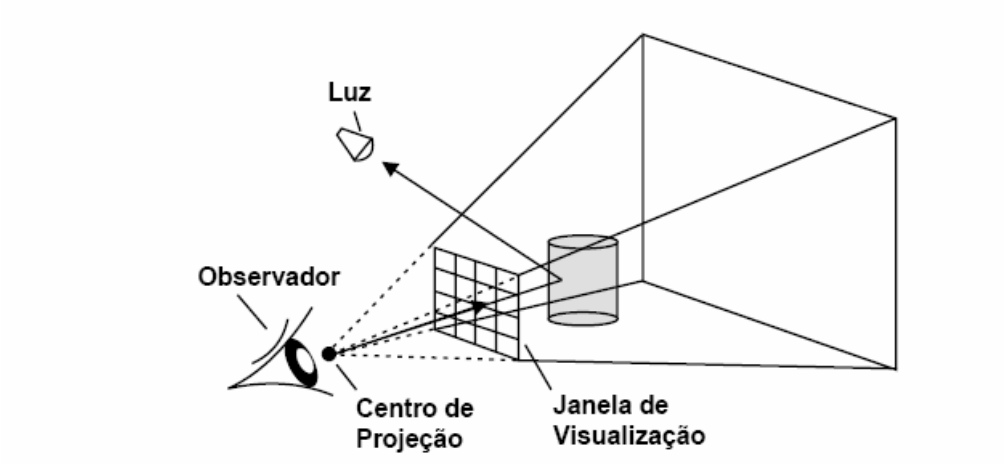
No mapeamento direto, os valores dos pixels em uma imagem destino ID são calculados a partir dos valores da imagem de origem IS através da transformação $T:IS(i,j) \rightarrow ID(i,j)$. No mapeamento inverso, percorrem-se os pixels da imagem destino, buscando os valores na imagem origem através da transformação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

A vantagem do mapeamento inverso é de que não surgem buracos na imagem destino devido a 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 em torno do pixel correspondente na imagem origem, efetuando-se deste modo uma amostragem mais apropriada.

12) Descreva o algoritmo de traçado de raios (*Raytracing*), mostrando onde ocorre a recursão (0.5 ponto) ?

O algoritmo do Raytracing 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 Raytracing 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 reflectiva, 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.



13) Todo algoritmo de recursão deve ter uma condição de parada. Explique qual é esta condição para o *Raytracing* (0.5 ponto).

O material dos objetos possui um índice de reflectividade e de transmissão. Este índice é multiplicado pelo resultado acumulado das recursões anteriores. O algoritmo usa um valor destes índices para limitar as recursões. Estes índices são entre 0 e 1. Se for 1 para cada material, então o algoritmo nunca irá parar, mas se for menor que 1, em algum momento o índice será um valor próximo de zero, o que indica a parada da recursão.

14) O que são *key-frames*. Explique em que consiste 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.

15) 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.