Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação

Disciplina: Computação Gráfica

AD2 - 2° semestre de 2014.

1) Escreva o corpo de uma aplicação típica do OpenGL para desenhar um triangulo, destacando em comentários alguns dos estágios do pipeline gráfico.

A seguir a CALLBACK responsável por desenhar um cubo sem o fundo. Aplicando uma textura nas paredes.

```
void desenhar(void)
{
        //Define os parâmetros da câmera que serão usados no estágio de culling
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(-300.0f, 420.0f, -80.0f, 15.0f, -1.0f, 2000.0f);
        gluLookAt(100.0f, 100.0f, -100.0f,  0.0f, 0.0f, 0.0f,  0.0f, 1.0f,0.0f);

        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();


        glPushMatrix();
        glScale(2.0f, 2.0f, 2.0f); //Aplica a escala para dobrar o tamanhos do quads. Este
comando é processado pelo estágio de processamento de vértice

        glTranslatef(72.0f , 0.0f, 72.0f );  //Desloca os quads para a posição 72, 0, 72.
Este comando é processado pelo estágio de processamento de vértice

        glColor3f(1.0f, 1.0f, 1.0f);   //Define cor branca para não afetar a cor da textura.

        glBindTexture(GL_TEXTURE_2D, 1);  //Efetua o bind da textura – a textura é
processada no estágio de processamento de pixel

        glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 0.0f);
        glVertex3f(0.0f, 0.0f, 0.0f);

        glTexCoord2f(0.0f, 1.0f);
        glVertex3f(0.0f, fHeight, 0.0f);

        glTexCoord2f(1.0f, 1.0f);
        glVertex3f(fWidth, fHeight,0.0f);


        glTexCoord2f(1.0f, 0.0f);
```

```
        glVertex3f(fWidth, 0.0f, 0.0f);//Aplica a textura que é processada no estágio de processamento de pixel


        glutSwapBuffers();  //Troca o framebuvver enviando o buffer construído para a saída de vídeo
}
```

2) O OpenGL é uma API que segue o modelo de iluminação de razterização. Explique.

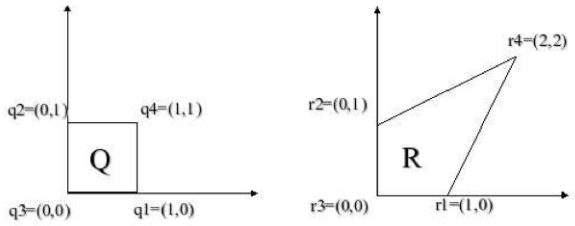
O modelo de iluminação baseado em rasterização calcula propriedades (cor, normal, iluminação, coordenadas de textura) para os vértices. No restante do pipeline os valores são interpolados. Pixel Shaders podem interferir nesta interpolação, mas os valores iniciais são baseados nos calores calculados para cada vértice.

3) O que é o Culling de Polígonos? Diferencie Frustrum Culling e Oclusion Culling (1.0 ponto).

Certos objetos e polígonos não são visíveis num determinado frame. Isto ocorre porque estão fora do cone de visão ou porque estão oclusos por outros que se sobrepõem a eles. Algoritmos de Frustrum Culling são métodos que tentam evitar o envio de polígonos que estão fora do cone de visão para o pipeline. Algoritmos de Oclusion Culling são técnicas que tentam evitar o envio de polígonos que estão oclusos por outros.

4) Escreva a matriz de projeção perspectiva e demonstre porque ela é assim.

A matriz de projeção perspectiva é uma matriz que representa uma transformação: $T : RP^2 \rightarrow RP^2$ no plano projetivo. Assim, uma transformação projetiva P em RP^2 fica caracterizada quando são conhecidas as imagens por P de 4 pontos em "posição geral" (3 quaisquer não estão em linha reta). Portanto, para transforma o quadrilátero A no quadrilátero B.



Neste caso, o que se deseja é achar a matriz que faz a transformação da Q em R na forma: $R = QT$

$$T = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ -1 & -1 & 3 \end{bmatrix}$$

A prova pode ser observada por:

$$T(1,0)=T(1,0,1)=\begin{pmatrix}2&0&0\\0&2&0\\-1&-1&3\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix}=\begin{pmatrix}2\\0\\2\end{pmatrix}=\begin{pmatrix}1\\0\\1\end{pmatrix}=\begin{pmatrix}1\\0\end{pmatrix}$$

$$T(0,1)=T(0,1,1)=\begin{pmatrix}2&0&0\\0&2&0\\-1&-1&3\end{pmatrix}\begin{pmatrix}0\\1\\1\end{pmatrix}=\begin{pmatrix}0\\2\\2\end{pmatrix}=\begin{pmatrix}0\\1\\1\end{pmatrix}=\begin{pmatrix}0\\1\end{pmatrix}$$

$$T(0,0)=T(0,0,1)=\begin{pmatrix}2&0&0\\0&2&0\\-1&-1&3\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}=\begin{pmatrix}0\\0\\3\end{pmatrix}=\begin{pmatrix}0\\0\\1\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix}$$

$$T(1,1)=T(1,1,1)=\begin{pmatrix}2&0&0\\0&2&0\\-1&-1&3\end{pmatrix}\begin{pmatrix}1\\1\\1\end{pmatrix}=\begin{pmatrix}2\\2\\1\end{pmatrix}=\begin{pmatrix}2\\2\end{pmatrix}$$

5) O Ray tracing é um ótimo método para renderizar funções implícitas. Explique (1.0 ponto).

O Ray tracing, por realizar um cálculo exaustivo de interseções com a geometria da cena, pode usar neste cálculo equações que não sejam necessariamente de polígonos, mas de outras superfícies. Na rasterização isto é praticamente impossível, devido ao pipeline baseado em vértices e interpolação de polígonos.

6) Escreva em alto nível um programa de Ray-tracing, destacando a sua chamada recursiva

```
Ray_Tracing(VETOR)
Para cada Pixel da Imagem faça
        Objeto_mais_próximo = Nenhum
        Distância_mínima = infinito
        Crie um raio do observador ao pixel
        Para cada objeto da cena faça
                Se o raio tem interseção com este objeto
                        Se Distancia_mínima < distancia (câmera até este objeto)
                                Objeto_mais_próximo = este objeto
                        Fim se
                        Se Objeto_mais_próximo = Nenhum
                                Pixel = Cor_de_fundo
                        Senão
                                Reflexo  = Calcula_reflexo(Objeto_mais_próximo, luz)
                                Transmissão = Calcula_transmissão (Objeto_mais_próximo, N)

                                Pixel = Phong(Objeto) + Ray_Tracing(Reflexo) +
                                Ray_Tracing(Transmissão)
                        Fim Se
                Fim se
        Fim faça
Fim faça
```

As chamadas recursivas estão em negrito.

7) O que é o algoritmo de radiosidade? Explique sucintamente seu funcionamento (1.0 ponto)

É um modelo de iluminação que leva em conta iluminação indireta, além da direta. Seu funcionamento se baseia em dividir a malha em patches, de onde se estimará quanto de energia luminosa é emitida por cada um, levando esta energia em conta no cálculo da iluminação e cada polígono ou ponto.

8) Qual a diferença do algoritmo de radiosidade para o algoritmo de ray-tracing

Os raios recursivos calculados pelo ray-tracing são individuais. No caso da reflexão, são os apenas os raios que seguem o modelo especular. No caso da transmissão, apenas o raio que segue a lei de Snell. Assim, o ray-tracing não calcula a iluminação global como um todo, uma vez que uma superfície reflete luz de maneira difusa também. Os algoritmos de radiosidade fazem uma estimativa da energia emitida por uma parte da geometria local, simulando com maior aproximação a iluminação ambiente global.

9) Porque na equação de Ray-tracing, em cada etapa a contribuição de cor vinda da recursão é menor? Porque depois de algumas recursões o efeito passa a ser praticamente imperceptível? (1.0 ponto)

A equação do Ray-tracing é dada por:

Cor = Iluminacao (P) + Kr* RayTracing(Reflexo) + Kt*RayTracing(Transmissao)

Kr é o coeficiente de reflexo, que varia entre 0 (sem reflexo) a 1 (espelho). Assim, em cada etapa o resultado da recrusão é multiplicado por um coeficiente menor ou igual a um, fazendo com que diminua de intensidade. Ao fazer isto recursivamente este coeficiente é espalhado e tem-se que o valor fica pequeno, podendo num determinado momento chegar a ser menor do que o valor de representatividade do RGB.

10) Como se faz para resolver o problema de profundidade na rasterização de polígonos? (Expliquem detalhadamente) (1.0 ponto)

Utiliza-se o algoritmo de Z-Buffer. Este algoritmo consiste em criar uma memória de tamanho equivalente ao frame-buffer. Sempre que um pixel referente a um triangulo for pintado no frame-buffer, será escrito no frame-buffer a profundidade do mesmo. Caso já haja uma profundidade escrita anteriormente no Z-Buffer, antes de pintar o pixel será feita uma consulta se este novo pixel possui profundidade maior ou menor. Caso seja menor, permite-se a sua escrita, em cima

do anterior. Neste caso atualiza-se o valor do Z-Buffer com o valor deste novo pixel. Caso o valor de profundidade seja menor, impede-se a escrita deste pixel no frame-buffer, pois já há um pixel mais próximo pintado previamente.