Fundação CECIERJ - **Vice Presidência de Educação Superior a Distância**

Curso Superior de Tecnologia em Sistemas de Computação

Disciplina: Computação Gráfica

AD2 2° semestre de 2010.

Nome –

Assinatura –

---

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

2) Dê alguns exemplos de callbacks no OpenGL e explique o que fazem.

3) Escreva uma matriz de transoformação geométrica que translade um objeto nos valores de 4 na coordenada x, 3 na coordenada y e 1 na coordenada z, rotacione o mesmo em 30 graus e aumente a sua escala em 50%. (observação: deve ser uma úncia matriz que contenha todas as transformações unificadas).

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

5) Escreva a equação do phong inteira para o caso de um objeto não conter especular e para a presença de duas fontes de luz L1 e L2.

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

7) Em que situações o Ray-tracing cairia num loop infinito?

8) Um jogador de vídeo-game reclama que uma textura de um quadro de uma sala fica piscando quando ele se afasta demais do objeto. Explique tecnicamente para ele o que é que está acontecendo.

9) Como se pode simular um reflexo num rendering em tempo real, onde o Ray-tracing não pode ser usado?

10) Porque a luz ambiente no modelo Phong é extremamente aproximada do que acontece na física real?

Fundação CECIERJ - **Vice Presidência de Educação Superior a Distância**
Curso de Tecnologia em Sistemas de Computação
Disciplina: Computação Gráfica AD1 - 2° semestre de 2010 - GABARITO

1) Escreva o corpo de uma aplicação típica do OpenGL, 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) Dê alguns exemplos de callbacks no OpenGL e explique o que fazem

CALLBACKS são funções escritas pelo programador para descrever o comportamento da aplicação quando os eventos forem disparados. Assim, abaixo segue um conjunto de alguns eventos.

```
// Função callback chamada para fazer o desenho
void renderEvent(void)
{...}

// Função callback chamada quando o tamanho da janela é alterado
void windowResizeEvent(GLsizei w, GLsizei h)
{ ... }

// Função callback chamada para gerenciar eventos de teclado – teclas comuns
void simpleKeysEvent(unsigned char key, int x, int y)
{ ... }

// Função callback chamada para gerenciar eventos do mouse
void mouseEvents(int button, int state, int x, int y)
{ ... }

// Função callback chamada para gerenciar eventos do teclado
// para teclas especiais, tais como F1, PgDn e Home
void specialKeysEvents(int key, int x, int y)
{ ... }
```

3) Escreva uma matriz de transformação geométrica que translade um objeto nos valores de 4 na coordenada x, 3 na coordenada y e 1 na coordenada z, rotacione o mesmo em 30 graus e aumente a sua escala em 50%. (observação: deve ser uma úncia matriz que contenha todas as transformações unificadas)

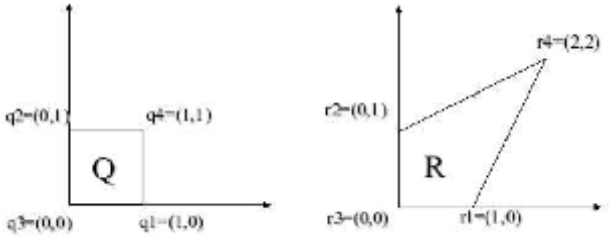
Observação: A rotação foi aplicada ao eixo Z. Entretanto, qualquer outro eixo pode ser usando para responder a questão, desde que a resposta esteja correta e justificada.

$$M = \begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos(30) & -sen(30) & 0 & 0 \\ sen(30) & \cos(30) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1.5 & 0 & 0 & 0 \\ 0 & 1.5 & 0 & 0 \\ 0 & 0 & 1.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 1.5 \times \cos(30) & -1.5 \times sen(30) & 0 & 4 \\ -1.5 \times sen(30) & 1.5 \times \cos(30) & 0 & 3 \\ 0 & 0 & 1.5 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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 ica 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) Escreva a equação do phong inteira para o caso de um objeto não conter especular e para a presença de duas fontes de luz L1 e L2.

A quantidade de luz de incide sobre um objeto é a soma das fontes, portanto a equação pode ser dada por:

$$I_{total} = I_a K_a C_a + f_{at} I_{luz} \left[\left(K_{l1} C_{l1} (NL_{l1}) \right) + \left(K_{l2} C_{l2} (NL_{l2}) \right) \right]$$

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) Em que situações o Ray-tracing cairia num loop infinito?

Uma superfície é capaz de refletir infinitos raios. Desta forma, o algoritmo de Ray-tracing pode entrar em um loop infinito. Para evitar este problema limita-se o número de raios refletidos desse objeto, garantindo que o algoritmo não entre em um estado de loop infinito.

8) Um jogador de vídeo-game reclama que uma textura de um quadro de uma sala fica piscando quando ele se afasta demais do objeto. Explique tecnicamente para ele o que é que está acontecendo.

O problema que acontece é conhecido como *alias*. Acorre porque a textura é armazenada contém um número finito de valores discretos de alguns imagem original, ou seja, a uma perda da informação. Ao ser aplicada sobre uma primitiva do OpenGL, por exemplo. Ao realizar a varredura de conversão da textura para o polígono, as suas coordenadas de textura são mapeados em coordenadas (u, v) e a textura é re-amostrados usando essas coordenadas. Assim, no caso em que o polígono é encolhido devido à perspectiva, que irá abranger apenas alguns pixels. Isso resultará em apenas um dos pontos de amostragem poucos espalhados por toda a imagem de textura e, portanto, causando este efeito indesejado.

9) Como se pode simular um reflexo num rendering em tempo real, onde o Ray-tracing não pode ser usado?

Para simular o reflexo em tempo real a estratégia mais simples consiste no Environment Mapping. Esta técnica se resume em criar uma esfera envolvente a cena e mapear sobre ela uma imagem que será refletida no objeto reflexivo. Esta esfera não será renderizada, apenas o seu reflexo será visto no objeto.

10) Porque a luz ambiente no modelo Phong é extremamente aproximada do que acontece na física real?

O modelo de iluminação Phong, também conhecido como modelo de iluminação local, permite realizar cálculo da intensidade de energia luminosa que é refletida por cada elemento da superfície de um objeto, aproximando-se, desta forma, do que acontece no modelo real.