Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação

Disciplina: Computação Gráfica

AD2 - 1° semestre de 2015.

1) O são Vertex Shaders, Pixel Shaders e Geometry Shaders? Dê um exemplo de efeito que pode ser atingido por cada um. (1.0 ponto).
   Vertex Shaders são programas que são programados e carregados para a GPU e interferem de alguma forma o estagio de geometria do pipeline gráfico. Efeitos possíveis com esta estratégia são texturas procedurais e diversos modelos de iluminação por vértice. Geometry shaders são programas de GPU que permitem alterar propriedades da geometria como um todo. Com eles é possível alterar a topologia ou a triangulação de uma malha, por exemplo. Pixel shaders são programas que alteram o estagio de rasterizacao. Efeitos possíveis são bump-mapping e especularidade.

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

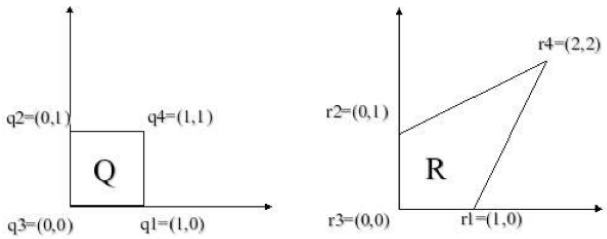
CALLBACKS são funções escritas pelo programador para descrever o comportamento da aplicação quando os eventos forem disparados. Assim, abaixo segue um conjunto de alguns eventos.

```
// 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) O que são algoritmos de Frustrum Culling e Occlusion Culling (1.0 ponto).
Culling de polígonos consiste em descartar polígonos que não são visíveis para a câmera, seja por razoes de oclusao ou por razoes de estarem fora do câmera frustum. Algoritmos de Frustrum Culling eliminam polígonos fora do view frustrum e algoritmos de Occlusion Culling eliminam polígonos que estão ocultos 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).



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 que é a linguagem CUDA para GPUs? O que é um kernel, dentro deste contexto? (1.0 ponto)

As GPUs se tornaram genéricas o suficiente para poderem executar programas que não são necessariamente etapas do pipeline gráfico. A linguagem CUDA foi criada para poder acessar a GPU de forma mais genérica e poder integrar com programas convencionais. Kernels são funções escritas em CUDA e que serão executadas pela GPU.

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) Diferencie o conceito de Iluminação global e iluminação local. (1 ponto)

Iluminação global é aquela que leva em consideração todos os raios de luz do cenário, incluindo aqueles derivados de múltiplas reflexões e difrações. Iluminação local é aquela que apenas considera os raios de luz derivados diretamente da fonte de luz.

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* e é um problema de amostragem. Ocorre porque a textura contém um número finito de valores

discretos. Ao realizar o mapeamento 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, quando o polígono é encolhido devido à perspectiva, o mapeamento irá abranger apenas alguns pixels da imagem original da textura. Isso resultará com que apenas alguns dos pontos de amostragem sejam vistos, causando este efeito indesejado.

9) Porque as texturas procedurais podem ser geradas nas placas gráficas programáveis? (1.0 ponto)

As texturas procedurais precisam gerar dados para cada pixel, que deve receber uma cor de uma textura. Na razterizacao convencional, a simples interpolação não e' capaz de gerar dados individualizados para cada texel. As placas programáveis permitem gerar dados individuais para cada pixel, interferindo no processo de rasterizacao. Pelo fato de poder executar uma função por pixel, estas placas permitem criar dados de textura pixel a pixel.

10) No estágio de rasterização, os triângulos que chegam para ser desenhados já sofreram projeção e são 2D. Além disso, a ordem com que chegam é aleatória e não estão organizados por profundidade de câmera. Explique como se trata o problema de conflito de profundidade gerado por triângulos que chegam para serem desenhados no mesmo local da tela. (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 maior do que o valor que já está escrito, impede-se a escrita deste pixel no frame-buffer, pois já há um pixel mais próximo e que já foi pintado previamente.