P1 CG 2012.2

	Computação Gráfica 1 Prof. Rodrigo de Toledo	Data: 9/12/2012 P1 2012.2

1) (2,5 pontos) O propósito de uma *Game Engine* é facilitar o trabalho do programador de jogo. Por exemplo, na URGE, sabemos que a *Terrain* (terreno do jogo) é criada a partir de apenas duas texturas: seus mapas de cor e de altura. O efeito final é de um terreno com acentuações e com iluminação detalhada. Explique como uma *Game Engine* pode produzir tal resultado sem precisar de um mapa de normais. (autor: Matheus Lessa)

2) (2,5 pontos) Qual a ordem correta no pipeline gráfico entre as seguintes etapas?
(a) Clipping; **(b)** Display/Visibility; **(c)** Illumination (shading); **(d)** Modeling Transformations; **(e)** Projection; **(f)** Scan Conversion (rasterization); **(g)** Viewing Transformation.
Continuação: Onde os vertex e fragment shaders se posicionam ao longo desse pipeline?

3) (2,5 pontos) Qual a interpolação usada em cada uma das operações abaixo?
(I) Acesso a textura 3D; **(II)** Alteração de resolução de uma imagem; **(III)** Aplicação de textura; **(IV)** Cálculo da normal no *Phong Shading*; **(V)** Geração de vértices no *tessellator* (*hull* e *domain shaders*); **(VI)** *Mip-mapping*; **(VII)** Projeção em perspectiva; **(VIII)** Rasterização de triângulos sem textura; **(IX)** Rotação 3D; **(X)** Transparência entre duas imagens de mesma resolução. As opções são: **(a)** linear, **(b)** bilinear, **(c)** trilinear, **(d)** baricêntrica, **(e)** nenhuma.

4) (2,5 pontos) Em algumas superfícies a contribuição da componente especular não é contínua. É o caso da pele humana: onde há mais oleosidade a especular é maior. Altere o programa abaixo (*vertex shader*) para que leve em consideração também a componente especular. Porém, a intensidade da especular (que normalmente é uma constante) deve ser obtida de uma segunda textura (em tons de cinza, como o *heightmap*). Use os comandos `Reflect` e `Pow`.

```
vec3 refletido = reflect(vec3 original, vec3 normal);
float result = pow(float base, float exponent);
```

Continuação: Esse programa teria um resultado melhor se usasse o *fragment shader*, destaque na sua resposta, quais linhas você passaria do *vertex* para o *fragment*.

```
uniform sampler2D sampler2d0;
void main() {
   vec4 cor = texture2D(sampler2d0,gl_MultiTexCoord0.xy);
   vec4 verticeModelo = gl_Vertex;
   vec4 verticeTransformadoEye = gl_ModelViewMatrix*verticeModelo;
   gl_Position = gl_ProjectionMatrix * verticeTransformadoEye;
```

```
   vec3 verticeNormal = gl_NormalMatrix*gl_Normal;
   verticeNormal = normalize(verticeNormal);

   vec3 dirLuz = gl_LightSource[0].position.xyz -
                 verticeTransformadoEye.xyz;
   dirLuz = normalize(dirLuz);

   float difusa = dot(dirLuz,verticeNormal);
   vec3 corDifusa = difusa*cor.rgb;
   gl_FrontColor = corDifusa.rgb;
}
```

Gabarito:

1) As acentuações são possíveis graças ao *displacement*, ao deslocar para cima vértices ao longo de um plano (quanto mais branca a cor do mapa de alturas maiores são os deslocamentos).
A inluminação é detalhada pois as normais são calculadas a partir do mapa de alturas.
((Extra: O cálculo da normal é feito da seguinte maneira: cria-se um vetor com a diferença entre os vizinhos horizontais e outro com a diferença entre os vizinhos verticais. A normal é o produto vetorial desses dois vetores.))
2) (1) D C G E A F B, ou: (2) D C G A E F B
Vertex shader: substitui DCGE para sequência (1) ou DCG para (2)
Fragment shader: entre F e B ((respota correta também se for F))

3) C B B D B C E D E A

4) Substituir `gl_FrontColor = corDifusa;` Por:

```
vec3 dirEye = normalize(-verticeTransformadoEye.xyz);
vec3 dirLuzRefletida = normalize(reflect(-dirLuz, verticeNormal));
float espec = max(0.0, dot(dirLuzRefletida,dirEye ));
float contanteEspecular = 64.0*texture2D(sampler2d1, gl_MultiTexCoord0.xy).r;
espec = pow(espec, contanteEspecular);
gl_FrontColor.rgb = corDifusa + espec;
```

As linhas:

```
vec4 cor = texture2D(sampler2d0,gl_MultiTexCoord0.xy);
.
.
.
float difusa = dot(dirLuz,verticeNormal);
vec3 corDifusa = difusa*cor.rgb;
gl_FrontColor = corDifusa.rgb;
```

Os parâmetros necessários devem ser passados para o fragment shader por interpolação.