P1 CG 2013.2

	Computação Gráfica 1 Prof. Rodrigo de Toledo	Data: 7/10/2013 P1 2013.2

1) (1,5 pontos) Ao se aplicar uma textura com anti-aliasing (ou mipmapping) a placa de vídeo faz uso de interpolação baricêntrica e trilinear. (a) O que é interpolação baricêntrica? (b) O que é interpolação trilinear? (c) Liste ordenadamente os passos que acontecem dentro da placa de vídeo para a aplicação de textura.

2) (2,5 pontos)

Suponha que um dado triângulo, após ser transformado para o espaço de clipping $[-1,1]^3$, possua seus vértices com coordenadas { (0,0,0), (0,1,1), (2,2,0) }:
(a) Descreva a transformação que deverá acontecer com o 3o vértice para que o triângulo seja desenhado. (b) Em qual pixel será desenhado o 1o vértice caso o buffer de renderização tenha dimensão 201x201? (c) Esse triângulo é paralelo ao plano near? Justifique. (d) Caso os vértices tenham sido passados na ordem em que foram apresentados, este triângulo está sendo desenhado de frente ou de costas? Justifique. (e) Qual a importância disso (frente/verso)?

3) (2 pontos)
Ray-casting (ou ray-tracing) de mapa de altura para realizar o parallax mapping.
(a) Qual o problema de fazer apenas linear? (b) Qual o problema de fazer apenas binário? (c) Como usar ambos para que fique melhor? (d) Ao usar ambos, em que situação deveriamos dar mais ênfase ao linear e ao binário.

4) (2 pontos)
(a) Por que renderizar espelho em tempo real é difícil? (b) Cite uma possível maneira de contornar essa limitação.

5) (2 pontos) (P2 2011.2) O que faz o vertex shader abaixo, se aplicado à malha de uma esfera, centrada em zero e raio um (como a "ST Sphere" do ShaderLabs)? Qual o resultado visual? O que deve ser alterado para que a normal funcione em todos os casos?

```
void main() {
vec4 v = gl_Vertex;
vec3 normal = vec3(0.0,0.0,0.0);
vec3 color;
float lado = 0.5;
if(v.x > lado){
        v.x = lado;
       color = vec3(0.0,0.0,1.0);
       normal += vec3(1.0,0.0,0.0);
} else if(v.x < -lado){
       v.x = -lado;
       color = vec3(0.0,1.0,0.0);
}
if(v.y > lado){
        v.y = lado;
       color = vec3(1.0,0.0,1.0);
} else if(v.y < -lado){
       v.y = -lado;
       color = vec3(1.0,1.0,1.0);
}
```

```
if(v.z > lado){
        v.z = lado;
       color = vec3(1.0,0.0,0.0);
} else if(v.z < -lado){
       v.z = -lado;
       color = vec3(1.0,1.0,0.0);
}

normal = normalize(normal);
vec3 dir_luz = gl_LightSource[0].position.xyz;
dir_luz = normalize(dir_luz);
float difusa = max(dot(normal, dir_luz),0.0);;
gl_FrontColor.xyz = color*difusa + color*0.2;
gl_FrontColor.w = 1.0;
gl_Position = gl_ModelViewProjectionMatrix * v;
}
```

Gabarito:

1)
2)
3)
4)
Suponha que um dado triângulo, após ser transformado para o espaço de clipping, possua seus vértices com coordenadas { (0,0,0), (0,1, 1), (2,2,0) }:

a) Descreva a transformação que deverá acontecer com o 3o vértice para que o triângulo seja desenhado.
R: Como um pedaço do triângulo está fora do espaço a ser visualizado ([-1,1]^3) ele deverá ser cortado. Nesse caso, o 3o vértice será substituído pelo vértice (1,1,0).

b) Em qual pixel será desenhado o 1o vértice caso o buffer de renderização seja 201x201 ?
R: O 1o vértice está bem no centro do espaço a ser visualizado, portanto ele estará no pixel do do centro do buffer (100,100).

c) Esse triângulo é paralelo ao plano near? Justifique.
R: Não. Ele só seria paralelo ao plano near caso a profundidade fosse igual em todos os vértices do triângulo.

d) Cite um importância do buffer de profundidade (z-buffer) para o algorítimo de rasterização utilizado na placa de vídeo.
R: Como a placa de vídeo processa os fragmentos em paralelo, o z-buffer é muito importante para organizar a escrita concorrente ao buffer de cor, selecionando somente os fragmentos que possuam profundidade menor que os fragmentos que já foram selecionados.

e) Caso os vértices tenham sido passados na ordem em que foram apresentados, este triângulo está sendo desenhado de frente ou de costas? Justifique.
R: De Frente. Na ordem em que foram passados, a regra da mão direita aponta para "fora do monitor" indicando que ele está de frente. O triângulo pode ter diferentes propriedades na frente e atrás (cor, normal, textura etc…). Além disso, com o propósito de otimizar, a placa pode estar com o faceculling ligado, eliminando do rendering os triângulos que estiverem de costas.