P3 CG 2012.1

	Computação Gráfica 1 Prof. Rodrigo de Toledo	Data: 3/07/2012 P3

1) (2 pontos) Filtro de imagem

Dado o filtro Δ a ser aplicado na imagem $\mathcal{I}$ (valores entre 0 e 1):

a) qual é o resultado esperado, considerando que a borda resolvida por espelhamento?

b) qual é o resultado esperado, considerando que a borda resolvida por repetição?

c) qual é a conta a que deve ser feita para que o resultado em (a) fique normalizado entre 0 e 1?

Δ =

-1	-2	-1
0	0	0
1	2	1

$\mathcal{I}$ =

1	0.5	0.2	0.2
0.6	0.4	0.1	0.1
0.4	0.3	0	0
0.3	0.2	0.1	0

2) (2 pontos) Coordenadas Homogêneas

Diga ao menos três utilidades para as coordenadas homogêneas.

3) (2 pontos) Bresenham (qualquer semelhança com a P2 não é mera coincidência)

a) Dado dois pontos (X1,Y1) e (X2,Y2), calcule quantos SUBPIXELS serão acesos pelo algoritmo de Bresenham, onde cada pixel tem 16 subpixels (4x4) para realizar o anti-aliasing. Pode usar pseudo-código apenas com estruturas se-senão. Explique como você está tratando o primeiro e o último pixels!

4) (2 pontos) Ray-tracing (qualquer semelhança com a P2 não é mera coincidência)

A função `bool = intersection(vec3 ray_origin, vec3 ray_direction, float radius, vec3 center, float &t1, float &t2, vec3 &normal1, vec3 &normal2)` retorna `verdadeiro/falso` se houve interseção entre uma esfera e um raio, assim como guarda os parâmetros dos pontos de interseção em `t1` e `t2` e suas normais. Implemente a função `bool = intersection1-2 (vec3 ray_origin, vec3 ray_direction, float radius1, vec3 center1, float radius2, vec3 center2, float &t1, float &t2, vec3 &normal1, vec3 &normal2);` que retorna se houve interseção entre o raio e o modelo CSG resultante da operação de subtração: `esfera1-esfera2`. A função também deve calcular o parâmetro do ponto de entrada e saída do raio, assim como as normais nesses pontos.

5) (2 pontos) GPU programming (qualquer semelhança com a P2 não é mera coincidência)

vertex:

```
varying vec3 v_V;
varying vec3 v_N;

void main () {
    gl_Position = ftransform();
    v_V = (gl_ModelViewMatrix * gl_Vertex).xyz;
    v_N = gl_NormalMatrix * gl_Normal;
}
```

fragment:

```
varying vec3 v_V;
varying vec3 v_N;

void main () {
  vec3 color = vec3(1.0, 0.0, 0.0);
  vec3 N = normalize(v_N);
  vec3 V = normalize(v_V);
  vec3 L = normalize(vec3(gl_LightSource[0].position));
  vec3 R = L - 2*dot(L,N)*N;

  vec3 a = color * 0.1;
  vec3 b = color * (1.0 - a) * max(dot(L, N), 0.0);
  vec3 c = vec3(1.0, 1.0, 1.0) * pow(max(dot(R, V), 0.0), 8.0);

  gl_FragColor = vec4(a+b+c, 1.0);
}
```

Quais modificações devem ser feitas para aplicar o modelo de Gouraud ao invés de Phong?

Gabarito:

1)

0 0 0 0

-1.6 -1.2 -0.8 -0.8

-1 -0.7 -0.3 -0.2
0 0 0 0

-1.3 -0.7 -0.4 -0.4
-2 -1.2 -0.8 -0.8
-1.1 -0.7 -0.3 -0.3
-0.4 -0.2 0.1 0.1

2)

- Permitir a operação de translação via multiplicação de matrizes
- Representar uma direção no espaço
- Perspectiva

3)

$\Delta X = X2 - X1$
if ($\Delta X < 0$) $\Delta X = - \Delta X$
$\Delta Y = Y2 - Y1$
if ($\Delta Y < 0$) $\Delta Y = - \Delta Y$
if ($\Delta Y > \Delta X$)
  pixels = $\Delta Y + 1$
else
  pixels = $\Delta X + 1$
subpixels = pixels * 4