Computação Gráfica

Exame 18/09/007

Duração: 2 horas

Parte I - OpenGL

1. Descreva o mecanismo dos vertex buffer objects considerando a sua definição e utilização. Compare com a utilização do modo imediato (glBegin – glEnd, submissão vértice a vértice).

2. Distinga os modelos de iluminação de Gouraud com interpolação e Phong, e descreva detalhadamente uma situação em que se verifiquem diferenças visíveis no resultado.

3. Enumere e caracterize as diferentes componentes da cor utilizadas nos materiais em OpenGL.

4. Descreva o conceito de *view frustum* e indique como este é definido em OpenGL.

5. Descreva o conceito de *Mipmapping*.

6. Considere um mini sistema solar constituido pelo Sol, planeta Terra e pela Lua. O Sol está fixo no centro do "Universo". A Terra gira em volta do sol e também em torno de si própria. Por sua vez a Lua gira em torno da Terra, com a particularidade de exibir sempre a mesma face para os habitantes da terra. Construa uma rotina para realizar a animação definida. Para simplificar o problema pode definir livremente as velocidades de rotação e translação, assim como as distâncias e dimensões dos corpos celestes. Considere que os corpos celestes têm o seu centro no plano XZ, e assuma que o eixo de rotação da terra em torno de si própria é perpendicular ao plano XZ.

Parte II – Shaders

Vertex Shader	**Fragment shader**

varying vec3 normal:
void main() {
normal = normalize(gl_NormalMatrix * gl_Normal);
...
gl_Position = ftransform();
}
	varying vec3 normal;
	void main () {
	vec3 color;
	vec3 n = normalize(normal);
	...
	gl_FragColor = color;
	}

1. Expanda o *shader* apresentado de acordo com as seguintes funcionalidades.

   a. Acrescente o código necessário para implementar uma luz posicional por pixel considerando somente a componente difusa e assumindo que a geometria tem uma cor fornecida ao shader através de uma variável "uniform" definida pelo programador. O código deve recorrer ao estado do OpenGL (considere a luz 0) para obter os dados da luz.

   b. Assumindo que a cor resultante da expansão acima pedida está armazenada numa variável "color" acrescente um efeito de nevoeiro ao shader baseado na seguinte fórmula:

   ```
   f = d / far
   cFinal =color * (1-f) + vec4(0.5, 0.5, 0.5, 1.0) * f;
   ```

   sendo far um valor passado numa variável uniform e d a distância do fragmento à câmara.

2. Indique quais as diferenças entre os qualificadores "uniform" e "attribute".

3. Diga por que razão a operação de *blending*, que permite a transparência, não pode ser realizada no *fragment shader*.

4. Diga porque é necessária a existência de uma matriz especial para transformar as normais: *gl_NormalMatrix*.

5. No *fragment shader* é comum proceder-se à normalização do vector normal. Indique a justificação para o fazer no caso geral, e em que circunstâncias particulares se poderia evitar a operação de normalização.

6. Diga quais as funções e limitações de um *vertex shader*.