Computação Gráfica

Exame 23/06/007

Duração: 2 horas

Parte I - OpenGL

1. As *Display Lists* permitem aninhamento, i.e. uma *Display List* pode conter referências a outras *Display Lists*. Compare a utilização desta estratégia vs. a utilização de uma só *Display List* para a cena toda.

2. Enumere e descreva as componentes da iluminação utilizadas em OpenGL.

3. Distinga os modelos de iluminação de Gouraud e Phong.

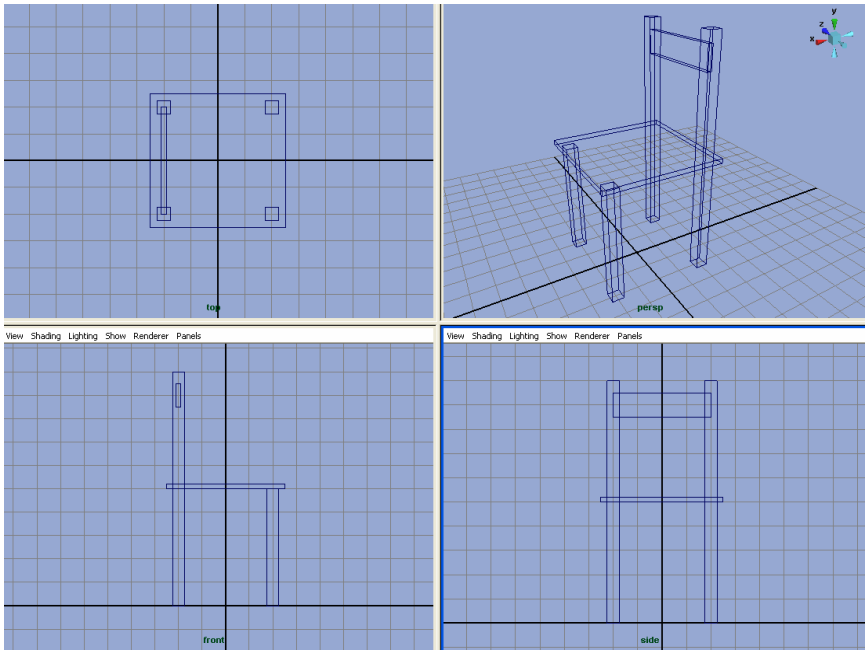
4. Enumere e relacione os diferentes sistemas de coordenadas utilizados em aplicações gráficas. Indique também quais as matrizes do estado do OpenGL que devem ser utilizadas para realizar as transformações entre os sistemas de coordenadas apresentados.

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

6. Descreva o conceito de *Mipmapping*.

7. Considere a seguinte figura onde são apresentadas diferentes vistas de uma "cadeira". Escreva uma função em C que permita construir um modelo semelhante em OpenGL, utilizando somente a primitiva Box (glutSolidBox(float size))

   Como referência, em termos de medidas, o lado dos quadrados da grelha vale uma unidade.



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 para implementar uma luz direccional 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.

2. Assuma que se pretende utilizar um *shader* que atribua cor a um pixel baseado na sua altura em coordenadas do mundo. São considerados três níveis de altura: alto (cor branca), médio (cor verde) e baixo (cor de areia). Considere que os valores que definem os níveis e as cores são passadas ao *shader* através de variáveis *uniform*.

3. Apresente o esquema do *pipeline* simplificado e descreva brevemente cada unidade do *pipeline*.

4. Diga por que razão a operação de *backface culling* não pode ser realizada no *vertex shader*.

5. Descreva a utilização de variáveis do tipo *varying*.

6. No *vertex shader* é comum proceder-se à normalização do vector normal. Enumere as circunstâncias em que é possível evitar esta operação.