Computação Gráfica

Exame 03/07/007

Duração: 2 horas

Parte I - OpenGL

1. Descreva *GL_TRIANGLE_STRIP* e *GL_TRIANGLE_FAN* exemplifique e indique quais as vantagens relativamente à especificação de triângulos independentes com *GL_TRIANGLES*.

2. Descreva as diferenças entre a componente difusa e especular.

3. Descreva as características e limitações do modelo de iluminação de Gouraud com interpolação.

4. Considere o seguinte excerto de código e descreva qual o objectivo e a justificação de cada uma das linhas numeradas apresentadas.

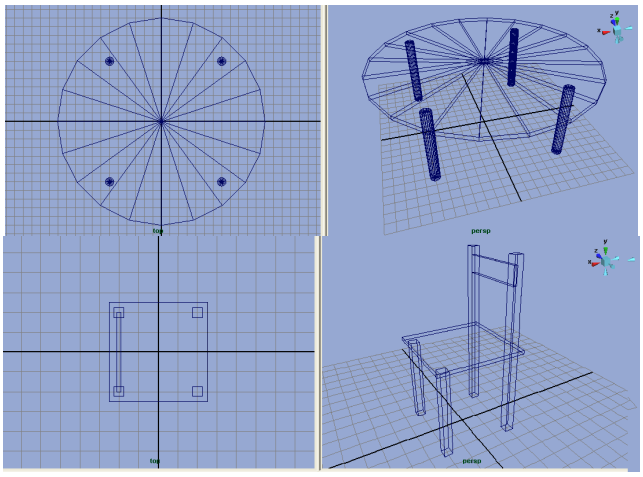
```
void changeSize(int w, int h) {
      ...
      float ratio = 1.0* w / h;
1      glViewport(0, 0, w, h);
2     glMatrixMode(GL_PROJECTION);
3     glLoadIdentity();
4     gluPerspective(45,ratio,1,1000);
5     glMatrixMode(GL_MODELVIEW);
}
```

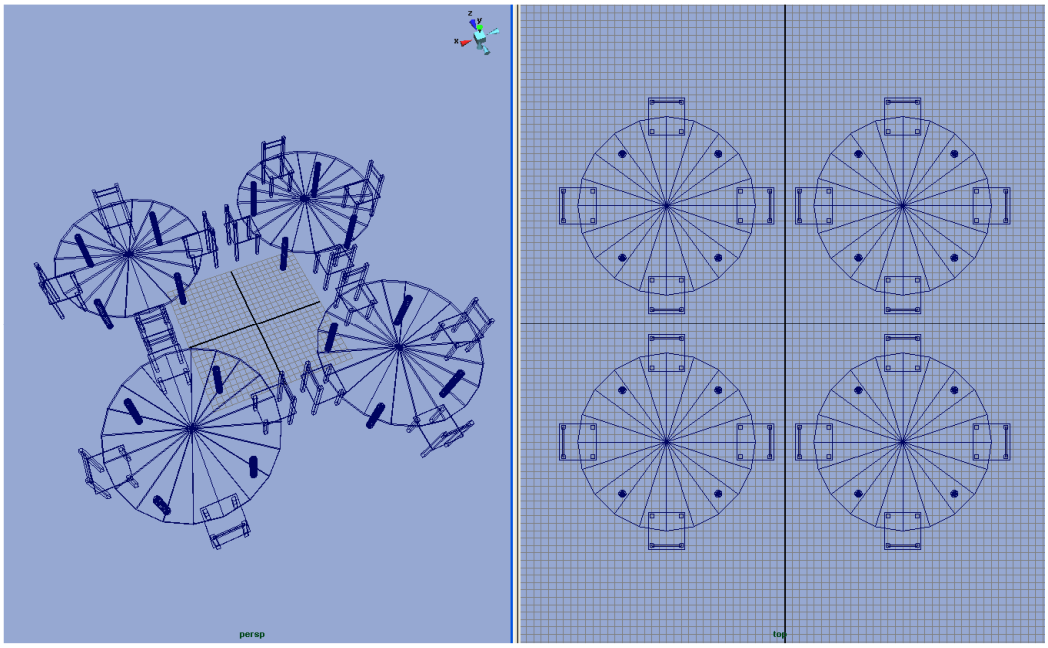
5. Descreve o modo de funcionamento do *Z buffer*

6. Descreva os filtros *GL_LINEAR* e *GL_NEAREST* no contexto da utilização de texturas.

7. Considere a biblioteca gLCC que contem primitivas gráficas para cadeiras e mesas como se ilustra nas figuras. Escreva uma função em C que permita construir em OpenGL um modelo semelhante ao apresentado na figura com a cena das mesas e cadeiras.

   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. Escreva um *shader* para realizar *toon shading* discretizando para cada pixel a sua cor de origem, especificada com *glColor3f*, em três valores, ou seja se um *pixel* for amarelo e receber uma intensidade de luz alta fica amarelo claro, se receber uma intensidade média fica como está e se a sua intensidade for baixa fica amarelo escuro. Considere uma luz posicional com a direcção da luz obtida através do estado do OpenGL na luz 0.

2. Pretende-se criar um *shader* que permita emular a funcionalidade dos *clip planes* do OpenGL, ou seja, para cada *pixel* é determinado de que lado do plano é que este se encontra e caso o resultado seja negativo o *pixel* é descartado. Escreva o código dos *shaders* assumindo que a informação do plano é acessível através de uma variável *uniform*. Considere ainda que o espaço em que o plano actua é o espaço global.

3. Descreva o processo de preparação da utilização de *shaders* numa aplicação em C com OpenGL.

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

5. Distinga entre variáveis do tipo *attribute* e *uniform*.

6. 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.