Licenciatura em Engenharia Informática
Sistemas Gráficos e Interacção

Época Especial 09-10-2009

N.º_____ Nome _____

Duração da prova: 75 minutos
Perguntas de escolha múltipla: cada resposta incorrecta desconta 1/3 do valor da pergunta

Parte Teórica **30% – 8 valores mín.**

1. A utilização de códigos de cores em interfaces com o utilizador
   a. Pode ser usada livremente, desde que no *splash screen* da aplicação apareça um aviso a desaconselhar o seu uso por utilizadores daltónicos
   b. Deve ser evitada a todo o custo, por causa dos utilizadores daltónicos
   c. Deve ser redundante, de modo a ter em conta as limitações dos utilizadores daltónicos
   d. Nenhuma das anteriores

2. A visualização, no ecrã de um vulgar computador, de um gráfico descrito no formato SVG (*Scalable Vector Graphics*)
   a. Não é de todo possível, dada a incompatibilidade do dispositivo e da representação
   b. É possível, mas requer técnicas sofisticadas de reconhecimento de padrões
   c. É possível, mas requer a realização prévia de uma operação de rasterização
   d. É possível, mas requer uma conversão prévia para o formato JPEG

3. Em que circunstâncias é que a ordem de composição de uma translação com outra translação é irrelevante?
   a. Em circunstância nenhuma
   b. Só quando os vectores que caracterizam as translações forem paralelos
   c. Só quando os vectores que caracterizam as translações forem perpendiculares
   d. Em todas as circunstâncias

4. As coordenadas homogéneas $[1, 0, 1, 0]^T$ designam
   a. Um ponto de coordenadas x = 1, y = 0, z = 1
   b. Um vector de componentes x = 0, y = 1, z = 0
   c. Um ponto de coordenadas x = 0, y = 1, z = 0
   d. Um vector de componentes x = 1, y = 0, z = 1

5. Complete as seguintes frases: na metáfora da câmara virtual adoptada pelo OpenGL
   a. As transformações de modelação servem para posicionar/orientar/deformar os objectos da cena
   b. As transformações de visualização servem para posicionar/orientar a câmara
   c. A transformação de projecção corresponde a ajustar a lente/objectiva da câmara
   d. A transformação de *viewport* corresponde a revelar as fotografias tiradas pela câmara

6. Na representação de sólidos com base em operações de conjuntos (CSG – *Constructive Solid Geometry*)

    a. O modelo é representado através da superfície que o delimita
    b. O modelo é representado por uma árvore em que os nós internos designam operações de conjuntos ou transformações lineares afim e as folhas denotam objectos primitivos
    c. O modelo é representado com base na divisão do espaço em cubos cujos lados são potências de base 2
    d. Há ambiguidade, pois a uma mesma representação podem corresponder vários modelos

7. Na codificação de sólidos com base em ponteiros para uma lista de arestas

    a. Não há uma lista de vértices
    b. As faces referenciam os seus vértices através de ponteiros
    c. Na descrição de cada aresta introduzem-se referências para as duas faces que lhe são adjacentes
    d. Nenhuma das anteriores

8. O modelo de iluminação do OpenGL

    a. Requer a especificação de normais
    b. É um modelo local de iluminação
    c. Contempla apenas os caminhos para a luz do tipo fonte luminosa → superfície → observador
    d. Todas as anteriores

9. No modelo de Phong, a contribuição dada

    a. Pela componente ambiente depende da geometria do objecto iluminado
    b. Pela componente difusa não depende do ângulo de incidência da luz
    c. Pela componente especular depende da posição do observador
    d. Todas as anteriores

10. No mapeamento de texturas em OpenGL, o processo de filtragem designado por GL_NEAREST_MIPMAP_LINEAR

    a. Escolhe o *texel* que mais se aproxima do centro do pixel no *mipmap* que melhor se adequa ao contexto de minificação existente
    b. Calcula uma média pesada da matriz de 2 x 2 *texels* que mais se aproxima do centro do pixel no *mipmap* que melhor se adequa ao contexto de minificação existente
    c. Escolhe o *texel* que mais se aproxima do centro do pixel em cada um dos dois *mipmaps* que melhor se adequam ao contexto de minificação existente; em seguida, efectua uma interpolação linear destes dois valores
    d. Calcula uma média pesada da matriz de 2 x 2 *texels* que mais se aproxima do centro do pixel em cada um dos dois *mipmaps* que melhor se adequam ao contexto de minificação existente; em seguida efectua uma interpolação linear destes dois valores

Época Especial 09-10-2009

N.º_____ Nome _____

Parte Teórico-Prática **40% – 10 valores mín.**

1. Suponha que possui um método `cubo(L)` que desenha um paralelepípedo com **o canto inferior esquerdo na origem** e largura e altura *L* (figura 2).

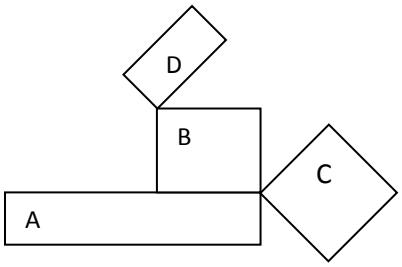


Figura 1

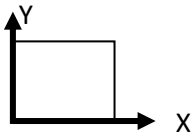
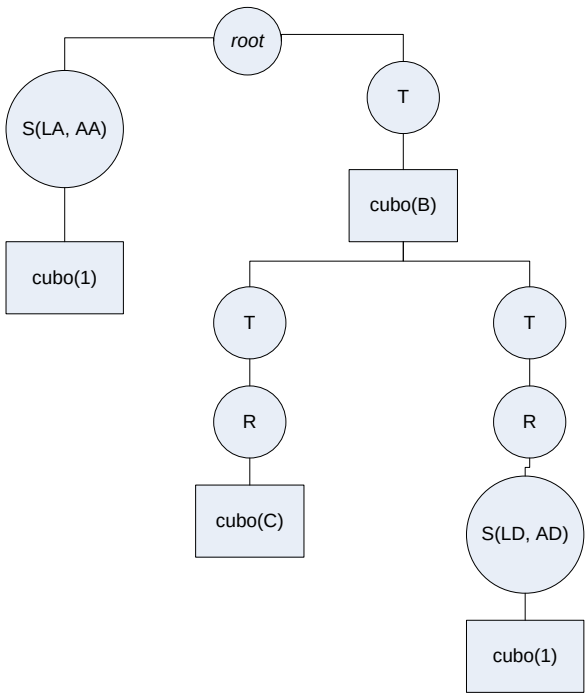


Figura 2

   a. Desenhe a árvore de cena com os nós correspondentes aos objectos e às transformações para desenhar o objecto da figura 1. Considere as seguintes dimensões dos objectos:
      - Objecto A: paralelepípedo de largura *LA* e altura *AA*
      - Objecto B: cubo de lado *B*
      - Objecto C: cubo de lado *C*
      - Objecto D: paralelepípedo de largura *LD* e altura *AD*

Árvore de cena



b. Escreva a sequência de instruções OpenGL necessárias para desenhar o modelo da figura 1.

```
glPushMatrix();
      glScalef(LA, AA, 1.0);
      cubo(1.0);
glPopMatrix();
glPushMatrix();
      glTranslatef(LA - B, AA, 0.0);
      cubo(B);
      glPushMatrix();
            glTranslatef(B, 0.0, 0.0);
            glRotatef(-45, 0.0, 0.0, 1.0);
            cubo(C);
      glPopMatrix();
      glPushMatrix();
            glTranslatef(0.0, B, 0.0);
            glRotatef(45.0, 0.0, 0.0, 1.0);
            glScalef(LD, AD, 1.0);
            cubo(1.0);
      glPopMatrix();
glPopMatrix();
```

2. Complete a seguinte frase:

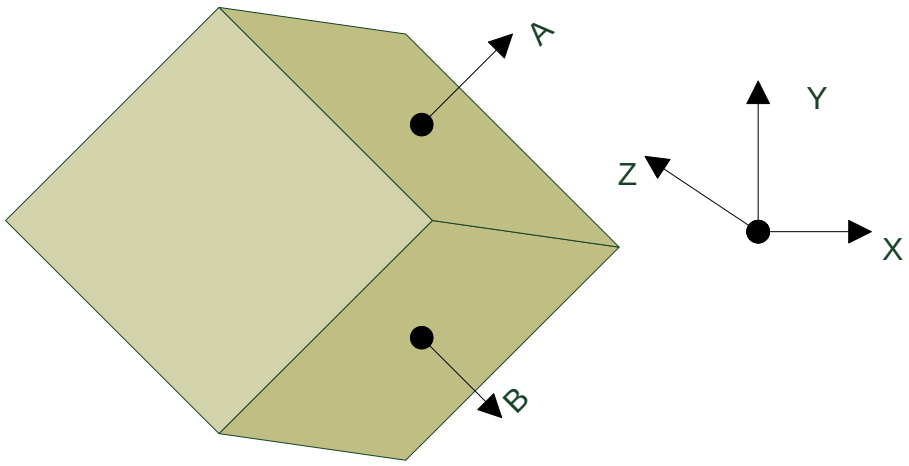
   e. Dos 4 blocos fundamentais da arquitectura de um programa gráfico, o renderer é "suportado" pela *callback* GLUT glutDisplayFunc, enquanto o bloco input está associado às *callbacks* GLUT glutKeyboardFunc e glutMouseFunc. O bloco lógica (regras de funcionamento) é normalmente implementado em funções separadas e invocadas pelas *callbacks* GLUT glutTimerFunc e/ou glutKeyboardFunc e/ou glutMouseFunc.

N.º_____ Nome _____

3. Indique as diferenças entre o modo *picking* e o modo *feedback* do OpenGL.

Ambos os modos devolvem informação sobre os objectos contidos num volume de projecção de uma cena. No modo *picking* é devolvido o nome simbólico dos objectos que se encontram dentro do volume de projecção de uma cena; normalmente esse volume de selecção é definido com base na coordenada de um dispositivo de entrada (ex., rato). No modo *feedback* a informação devolvida é informação geométrica dos objectos (vértices, cores, texturas).

4. Indique as componentes das normais (não necessariamente unitárias) identificadas pelos vectores A (co-planar com XY) e B (co-planar com XY) do polígono descrito na seguinte figura:



A: (1, 1, 0)

B: (1, -1, 0)

5. Qual dos seguintes vectores define uma luz direccional num ângulo de 45 graus ao solo se este for definido no plano XZ?

a. (1, 1, 0, 0)
b. (0, 0, 1, 0)
c. (0, 1, 0, 1)
d. (0, 0, 1, 1)

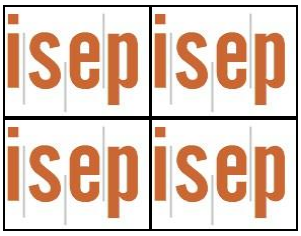
6. Suponha que foi activado o mapeamento de uma textura representativa do logótipo do ISEP. O seguinte conjunto de instruções gera que imagem final no ecrã?

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glBegin(GL_QUADS);
   glTexCoord2f(0.0, 0.0);
   glVertex3f(-2.0, -1.0, 0.0);
   glTexCoord2f(0.0, 1.0);
   glVertex3f(-2.0, 1.0, 0.0);
   glTexCoord2f(2.0, 1.0);
   glVertex3f(0.0, 1.0, 0.0);
   glTexCoord2f(2.0, 0.0);
   glVertex3f(0.0, -1.0, 0.0);
glEnd();
```

a. 

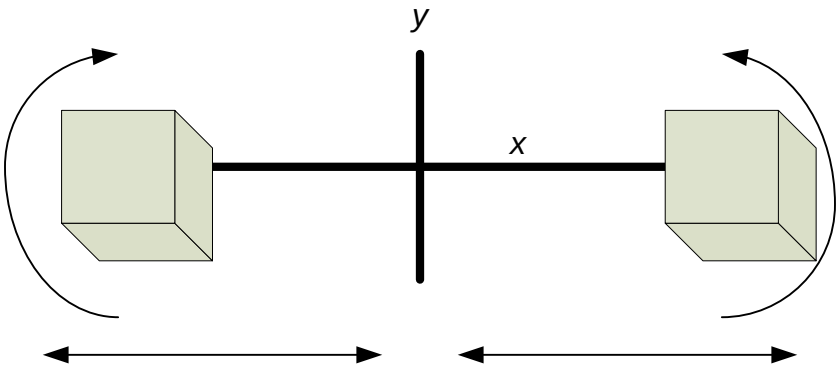
b. 

c. 

d. 

N.º_____ Nome _____

7. Complete o seguinte programa de modo a que a cena final desenhe dois cubos em rotação sobre si próprios no eixo do *x* (cada um em sentidos opostos) e que se desloquem ao longo do eixo do *x* entre [0 e +/-3] (ver figura). Suponha que as reticências correspondem ao código OpenGL que normalmente se encontraria nessa função e preencha apenas as linhas com o código para efectuar a animação (*nem todas as linhas necessitam de ser preenchidas*).




```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <GL/glut.h>

typedef struct {
     double angRot, deltaT;
     int dirT;
     _____
} Modelo_t;

Modelo_t modelo;

typedef struct {
     int delay;
     _____
     _____
} Estado_t;

Estado_t estado;
```

```
void Init(void)
{
     ...
     modelo.angRot = 0.0;
     modelo.deltaT = 0.0;
     modelo.dirT = 1;
}

void desenhaCubo(void) { ... }

void Draw(void)
{
     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
     glLoadIdentity();
     gluLookAt(0, 0, 5, 0, 0, 0, 0, 1, 0);
     glPushMatrix();
          glTranslated(modelo.deltaT, 0.0, 0.0);
          glRotated(modelo.angRot, 1.0, 0.0, 0.0);
          desenhaCubo();
     glPopMatrix();
     glPushMatrix();
          glTranslated(-modelo.deltaT, 0.0, 0.0);
          glRotated(-modelo.angRot, 1.0, 0.0, 0.0);
          desenhaCubo();
     glPopMatrix();

     glFlush();
}

void Timer(int value)
{
     glutTimerFunc(estado.delay, Timer, 0);
     modelo.angRot += 5.0;
     modelo.deltaT += 0.5 * (double)modelo.dirT;
     if (modelo.deltaT > 3.0 || modelo.dirT < -3.0)
          modelo.dirT *= -1;
     glutPostRedisplay();
}
```

N.º_____ Nome _____

```
void main(int argc, char **argv)
{
     ...

     Init();

     ...

     glutTimerFunc(estado.delay, Timer, 0);

     glutMainLoop();
}
```