Licenciatura em Engenharia Informática
Sistemas Gráficos e Interacção

Época de Recurso 19-02-2009

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. O design de interfaces do tipo WIMP (*Windows, Icons, Menus and Pointers*)

   a. Não deve contemplar o uso de teclas aceleradoras, pois o mesmo implica um esforço de memorização considerável por parte do utilizador
   b. Deve ser centrado no sistema e não no utilizador, pois o comportamento deste último é imprevisível
   c. Não deve basear-se em analogias com o mundo real, pois pode suscitar confusão na mente do utilizador
   d. Nenhuma das anteriores

2. A visualização, no ecrã de um vulgar computador, de um gráfico descrito no formato BMP (*Bitmap*)

   a. Não é de todo possível, dada a incompatibilidade do dispositivo e da representação
   b. Requer a prévia realização de uma operação de rasterização
   c. Dispensa o recurso a técnicas sofisticadas de reconhecimento de padrões
   d. Só é possível em sistemas cuja arquitectura contempla um processador gráfico (GPU)

3. As translações e as rotações

   a. São exemplos de transformações lineares afim
   b. São exemplos de transformações rígidas
   c. Podem ser representadas na forma matricial
   d. Todas as anteriores

4. As coordenadas homogéneas $[6, 12, 8, 0]^T$ designam

   a. Um ponto de coordenadas x = 6, y = 12, z = 8
   b. Um vector de componentes x = 6, y = 12, z = 8
   c. Um sólido com 6 faces, 12 arestas e 8 vértices
   d. Nenhuma das anteriores

5. Complete a seguinte frase:

No *pipeline* de transformações do OpenGL as coordenadas correspondentes aos vértices dos objectos são multiplicadas pela matriz de modelação e visualização, dando origem às coordenadas de olho. Segue-se a multiplicação pela matriz de projecção, da qual resultam as correspondentes coordenadas de recorte. Em seguida, realiza-se a operação de divisão perspectiva, obtendo-se assim as coordenadas normalizadas de dispositivo. Por último, efectua-se a transformação *viewport* e a consequente determinação das coordenadas de janela.

6. A codificação *Winged-Edge* de sólidos

    a. É usada na representação de sólidos por fronteira (B-Rep)
    b. Armazena informação numa estrutura associada às arestas
    c. Permite determinar em tempo constante os 9 tipos de adjacência de vértices, arestas e faces
    d. Todas as anteriores

7. Na representação de sólidos por fronteira (B-Rep – *Boundary Representation*)

    a. Há ambiguidade, pois a uma mesma representação podem corresponder vários modelos
    b. O modelo é representado através da superfície que o delimita
    c. 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
    d. O modelo é representado com base na divisão do espaço em cubos cujos lados são potências de base 2

8. Para iluminar uma cena com uma fonte de luz posicional do tipo projector, deverá

    a. Activar o modelo de iluminação do OpenGL
    b. Especificar para a posição um conjunto de coordenadas tal que $w \neq 0$
    c. Especificar para o ângulo de *cutoff* um valor compreendido entre 0^o e 90^o
    d. Todas as anteriores

9. A contribuição dada pela componente de iluminação ambiente do modelo de Phong

    a. Depende da geometria do objecto iluminado
    b. Depende do co-seno do ângulo de incidência da luz
    c. Depende da posição do observador
    d. Nenhuma das anteriores

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

    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 de Recurso 19-02-2009

N.º_____ Nome _____

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

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

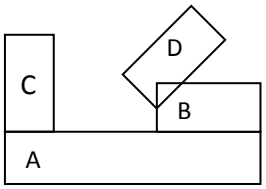


Figura 1

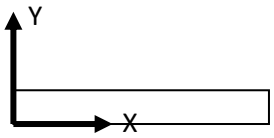
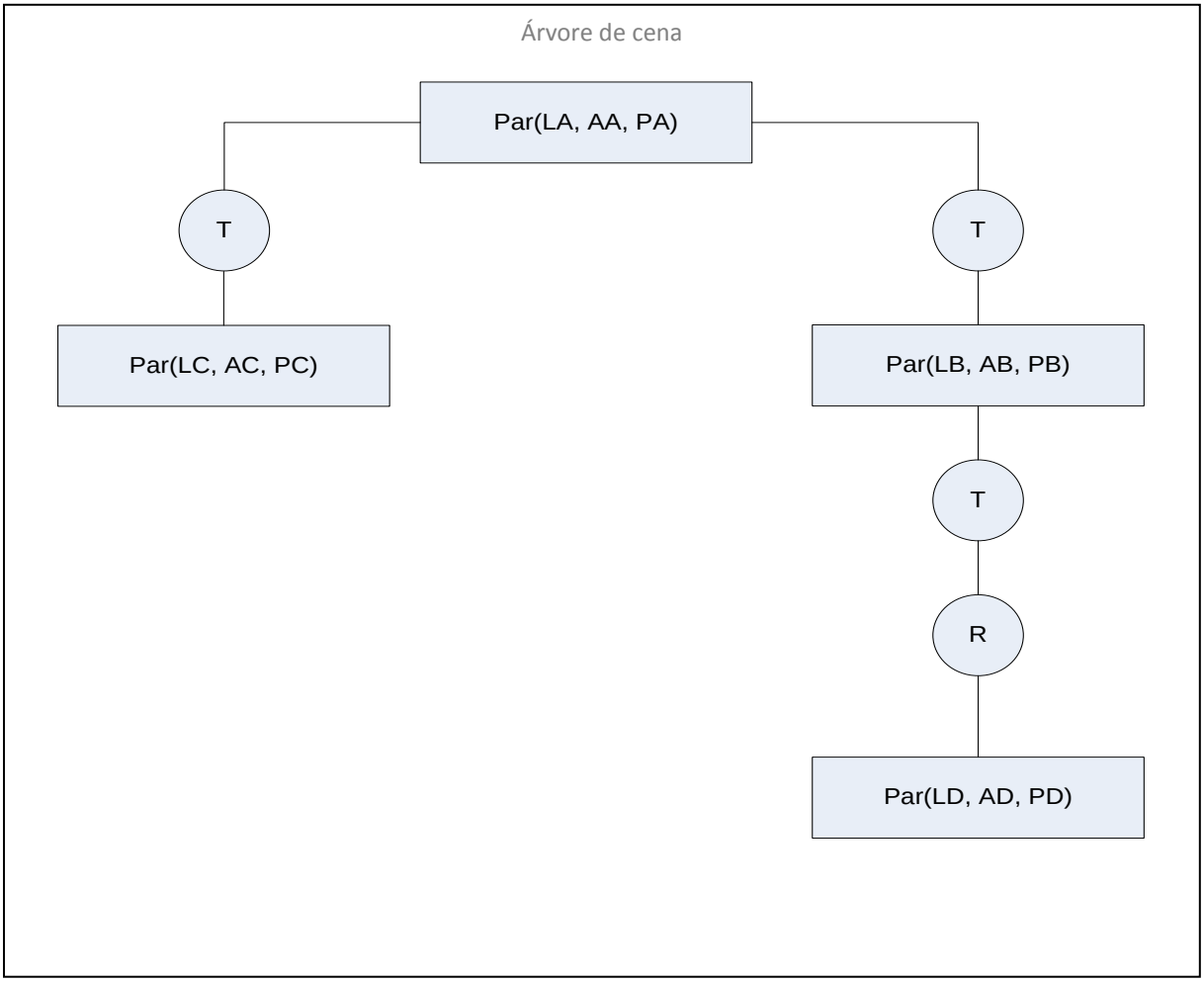


Figura 2

Dimensões dos objectos
A: L_A, A_A, P_A
B: L_B, A_B, P_B
C: L_C, A_C, P_C
D: L_D, A_D, P_D

a. Desenhe a árvore de cena com os nós correspondentes aos objectos e às transformações necessárias para desenhar o modelo da figura 1.



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

```
paralelepipedo(LA, AA, PA);

glPushMatrix();
    glTranslatef(0.0, AA, 0.0);
    paralelepipedo(LC, AC, PC);
glPopMatrix();

glPushMatrix();
    glTranslatef(LA - LB, AA, 0.0);
    paralelepipedo(LB, AB, PB);
    glTranslatef(0.0, AB / 2.0, 0.0);
    glRotatef(45.0, 0.0, 0.0, 1.0);
    paralelepipedo(LD, AD, PD);
glPopMatrix();
```

2. Complete a seguinte frase:

O GLUT define um conjunto de funções de registo de *callbacks*, entre as quais:

(i) glutDisplayFunc, que trata do desenho da cena;

(ii) glutKeyboardFunc e glutMouseFunc, que tratam dos eventos de dispositivos de entrada;

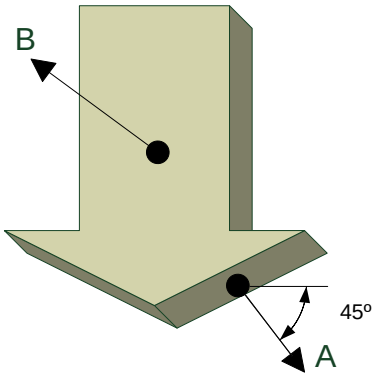
(iii) glutTimerFunc, que permite criar acções cíclicas no programa, desde que reactivadas em cada *callback* do timer.

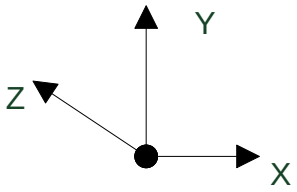
N.º_____ Nome _____

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

No modo selecção é devolvida informação (nome simbólico) referente a quais os objectos que se encontram dentro do volume de projecção de uma cena. O modo *picking* é, na realidade, uma variante do modo selecção, ao definir o volume de selecção com base na coordenada de um dispositivo de entrada (por exemplo, o rato).

4. Indique as componentes das normais (não necessariamente unitárias) identificadas pelos vectores A (complanar com XY) e B (perpendicular a XY) dos polígonos descritos na seguinte figura:





A: (1, -1, 0)

B: (0, 0, 1)

5. Supondo que a sua cena tem apenas um fonte de iluminação cuja componente difusa é (0.5, 1, 0.5) que incide sobre um objecto cujo material tem apenas componente difusa com características (0.5, 0, 1), qual a cor resultante no ecrã para esse objecto?

a. (0.5, 1, 0.5)
b. (0.25, 0, 0.5)
c. (0.5, 0, 1)
d. (0.25, 0, 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?

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(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(1.0, 1.0);
   glVertex3f(0.0, 1.0, 0.0);
   glTexCoord2f(1.0, 0.0);
   glVertex3f(0.0, -1.0, 0.0);
glEnd();
```

a. 

b. 

c. 

d. 

N.º_____ Nome _____

7. Complete o seguinte programa de forma que o utilizador possa viajar pela cena, observando um cubo de diferentes ângulos, utilizando as teclas X, x, Y, y, Z e z. O cubo encontra-se centrado na origem. Suponha que as reticências correspondem ao código OpenGL que normalmente se encontraria nessa função e preencha apenas as linhas com o código necessário para efectuar a navegação (*nem todas as linhas necessitam ser preenchidas*).

   Nota: este exercício pode ser resolvido usando o posicionamento da câmara ou a rotação do objecto. A solução apresentada utiliza o posicionamento da câmara.

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

typedef struct {

     _____

     _____

} Modelo_t;

Modelo_t modelo;

typedef struct {

     float eyex, eyey, eyez;

} Estado_t;

Estado_t estado;

void Init(void)
{
     ...

     estado.eyex = estado.eyey = estado.eyez = 3.0;

}

void desenhaCubo(void) { ... }

void Draw(void)
{
     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

     glLoadIdentity();

     gluLookAt(estado.eyex, estado.eyey, estado.eyez,
```

```
        0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

        desenhaCubo();

        ---

        ---

        glFlush();
}

void Key(unsigned char key, int x, int y)
{
        switch (key) {
                ...
                case 'X' : estado.eyex += 0.4;
                        break;
                case 'x' : estado.eyex -= 0.4;
                        break;
                case 'Y' : estado.eyey += 0.4;
                        break;
                case 'y' : estado.eyey -= 0.4;
                        break;
                case 'Z' : estado.eyez += 0.4;
                        break;
                case 'z' : estado.eyez -= 0.4;
                        break;
        }

        // redesenhar o ecra
        glutPostRedisplay();
}

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

        Init();

        ...

        ---

        ---

        glutMainLoop();
}
```