Licenciatura em Engenharia Informática
Sistemas Gráficos e Interacção

Época Normal 28-01-2009

N.º_____ Nome _____

Parte Teórica **70% – 75 minutos**

1. A visualização, no ecrã de um vulgar computador, de um gráfico descrito no formato SVG (*Scalable Vector Graphics*)

   a. Não é de todo possível, pois as representações vectoriais não são compatíveis com os dispositivos matriciais
   b. Requer a prévia realização de uma operação de rasterização
   c. Requer técnicas sofisticadas de reconhecimento de padrões
   d. Só é possível em sistemas cuja arquitectura contempla um processador gráfico (GPU)

2. As arquitecturas dotadas de processador gráfico (GPU)

   a. Recorrem a técnicas de paralelismo para aumentar o desempenho
   b. Chamam a si a realização de algumas tarefas tais como as transformações e projecções
   c. Normalmente usam memória separada da do sistema
   d. Todas as anteriores

3. Uma transformação linear afim

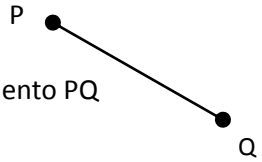
   a. Transforma necessariamente a origem na própria origem
   b. Preserva necessariamente as dimensões e os ângulos dos objectos
   c. Transforma necessariamente os segmentos de recta em segmentos de recta
   d. Nenhuma das anteriores

4. As coordenadas homogéneas $[1, 2, 3, 4]^T$ designam

   a. Um ponto de coordenadas x = 1, y = 2, z = 3
   b. Um vector de componentes x = 1, y = 2, z = 3
   c. Um ponto de coordenadas x = 0.25, y = 0.5, z = 0.75
   d. Um vector de componentes x = 0.25, y = 0.5, z = 0.75
   e. Um ponto de coordenadas x = 4, y = 8, z = 12
   f. Um vector de componentes x = 4, y = 8, z = 12

5. Considere dois pontos genéricos P e Q (não coincidentes) e o ponto R = P + α * (Q - P)

   a. Se $0 < \alpha < 1$ então o ponto R não pertence ao segmento PQ
   b. Se α = 0 então o ponto R coincide com o ponto Q
   c. Se α = 0.5 então o ponto R coincide com o ponto médio do segmento PQ
   d. Se α = 1 então o ponto R coincide com o ponto P



6. 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.

7. A modelação por malha de arame (*wireframe*)

   a. Pode gerar modelos ambíguos
   b. Fornece a descrição matemática das superfícies que delimitam o objecto
   c. Contém informação sobre o fecho e a conectividade dos objectos modelados
   d. Nenhuma das anteriores

8. Para iluminar uma cena com uma fonte de luz pontual que radie em todas as direcções, deverá

   a. Activar o modelo de iluminação do OpenGL
   b. Especificar para a posição um conjunto de coordenadas tal que w ≠ 0
   c. Especificar para o ângulo de *cutoff* o valor de 180º
   d. Todas as anteriores

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

   a. Não 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_LINEAR_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 Normal 28-01-2009

N.º_____ Nome _____

Parte Teorico-Prática **70% – 75 minutos**

1. Supondo que possui um método `paralelepipedo(L, A, P)` que desenha um paralelepípedo centrado na origem com largura *L*, altura *A* e profundidade *P* (figura 2):

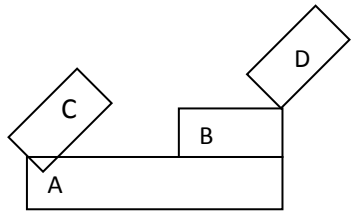


Figura 1

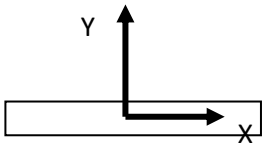
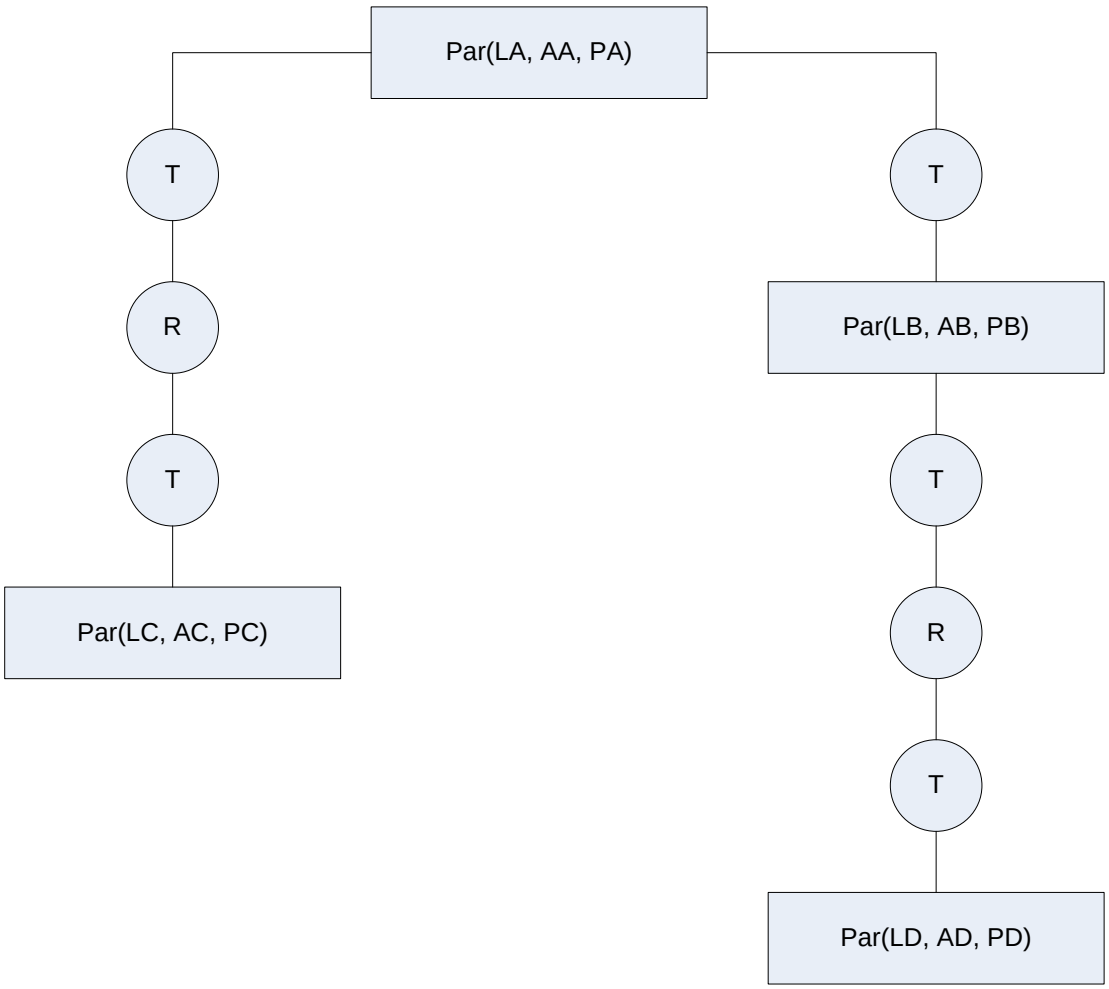


Figura 2

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



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

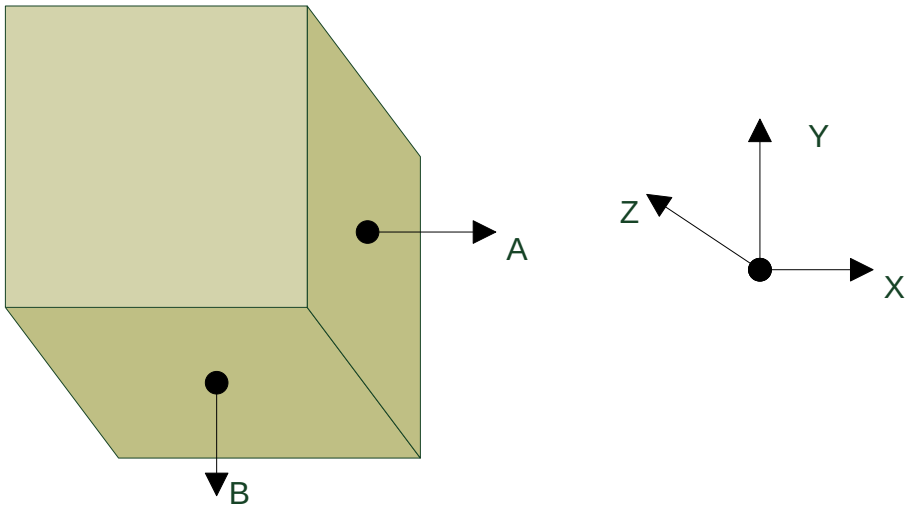
```
paralelepipedo(LA, AA, PA);

glPushMatrix();
    glTranslatef(-LA / 2.0, AA / 2.0, 0.0);
    glRotatef(45.0, 0.0, 0.0, 1.0);
    glTranslatef(LC / 2.0, 0.0, 0.0);
    paralelepipedo(LC, AC, PC);
glPopMatrix();

glPushMatrix();
    glTranslatef(LA / 2.0 - LB / 2.0, AA / 2.0 + AB / 2.0, 0.0);
    paralelepipedo(LB, AB, PB);

    glTranslatef(LB / 2.0, AB / 2.0, 0.0);
    glRotatef(45.0, 0.0, 0.0, 1.0);
    glTranslatef(LD / 2.0, AD / 2.0, 0.0);
    paralelepipedo(LD, AD, PD);
glPopMatrix();
```

2. Complete a seguinte frase:

A arquitectura de um programa gráfico pode ser resumidamente descrita em 4 blocos: o renderer que trata do desenho, o input que trata dos eventos de entrada, o modelo que descreve as estruturas de dados (os objectos) do programa e a lógica que define as regras de funcionamento do programa.

N.º_____ Nome _____

3. Indique as normais (não unitárias) identificadas pelos vectores A e B dos polígonos descritos na seguinte figura:



A: (1, 0, 0)

B: (0, -1, 0)

4. Se estiver a usar uma projecção do tipo perspectiva e pretender simular um efeito de *zoom* na cena, que função do OpenGL deve usar e qual o parâmetro que deve variar?

Devo usar a função `gluPerspective()`, variando o parâmetro *`field_of_view`*.

5. O seguinte conjunto de instruções costuma ser colocado em que *callback* do GLUT?

```
glViewport(0, 0, (GLint) width, (GLint) height);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0, LARGURA_CAMPO, 0, ALTURA_CAMPO);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

a. glutReshapeFunc
b. glutDisplayFunc
c. glutTimerFunc
d. glutKeyboardFunc

6. Qual dos seguintes vectores define uma luz direccional perpendicular ao solo se este for definido no plano XZ?

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

7. Complete o seguinte programa de forma a animar a rotação de um cubo em torno do eixo dos YY. 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 ser preenchidas*).

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

typedef struct {
     float angulo;
     _____________________________________________
     _____________________________________________
}Modelo_t;

Modelo_t modelo;

void Init(void)
{
     ...
     modelo.angulo = 0.0f;
     _____________________________________________
     _____________________________________________
}

void Reshape(int width, int height)
{
     ...
     _____________________________________________
     _____________________________________________
}

void desenhaCubo(void) { ... }
```

N.º_____ Nome _____

```
void Draw(void)
{
     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
     glLoadIdentity();
     glRotatef(modelo.angulo, 0.0, 1.0, 0.0);
     desenhaCubo();
     _____
     _____
     glFlush();
}

void Timer(int value)
{
     glutTimerFunc(estado.delay, Timer, 0);
     modelo.angulo += 5.0f;
     _____
     // redesenhar o ecra
     glutPostRedisplay();
}

void main(int argc, char **argv)
{
      ...
      Init();
      ...
     _____
     _____
     glutMainLoop();
}
```