Licenciatura em Engenharia Informática
Sistemas Gráficos e Interacção

Época de Recurso 19-02-2010

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.**

a. Considere uma situação em que o utilizador, ao preencher o campo numérico de um formulário (o código de um item em stock, por exemplo), excede a gama permitida para o valor desse campo. Qual das seguintes mensagens de erro lhe parece mais adequada à situação?

   i. "O código do item é inválido".
   ii. "O código do item é inválido. Consulte o manual de instruções".
   iii. "O código do item encontra-se fora da gama permitida".
   iv. "O código do item deverá estar compreendido entre 0 e 9999".

b. Um sistema gráfico com um *frame buffer* RGB de 24 bits tem capacidade para reproduzir imagens com

   i. Uma resolução de 1024 x 768 píxeis
   ii. 24 níveis de vermelho
   iii. 2^{24} cores
   iv. Nenhuma das anteriores

c. Suponha que pretende transformar a geometria de um objecto de modo a simular uma reflexão numa superfície plana (num espelho, por exemplo; ou na superfície de um lago). Deverá usar uma transformação de

   i. Translação
   ii. Rotação
   iii. Escalamento
   iv. Nenhuma das anteriores

d. Numa projecção perspectiva

   i. As dimensões aparentes dos objectos aumentam com o aumento da distância à câmara
   ii. As dimensões aparentes dos objectos não dependem da distância à câmara
   iii. O volume de visualização tem a forma de um tronco de pirâmide
   iv. Nenhuma das anteriores

e. As *octrees*

   i. Dividem o espaço em cubos de igual dimensão
   ii. Dividem o espaço em cubos cujos lados são potências de 2
   iii. Dividem o espaço em poliedros convexos, não necessariamente cúbicos
   iv. Nenhuma das anteriores

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

   i. Não é compatível com as fontes de luz posicionais
   ii. É característica dos materiais transparentes
   iii. Depende do ângulo de incidência da fonte de luz, mas não depende da posição do observador
   iv. Nenhuma das anteriores

g. O mecanismo de mapeamento de texturas do OpenGL suporta

   i. *Mipmaps*
   ii. Repetição de padrões de texturas
   iii. A modulação da cor da superfície do objecto com a dos téxeis
   iv. Todas as anteriores

h. O algoritmo do pintor de determinação de visibilidade

   i. Pinta os objectos mais próximos antes de pintar os mais distantes
   ii. Pinta os objectos mais distantes antes de pintar os mais próximos
   iii. Só é aplicável a cenas compostas por objectos poliédricos fechados
   iv. Nenhuma das anteriores

Época de Recurso 19-02-2010

N.º_____ Nome _____

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

Notas:

Em todas as perguntas, a menos que algo seja dito em contrário, assuma a orientação dos eixos e a posição da câmara por omissão do OpenGL, considerando o plano XY como o "solo".

Fórmula paramétrica da circunferência:

$x = r \cdot cos(t)$

$y = r \cdot sen(t)$

a. Se tem uma única luz definida unicamente com `glLigthfv(GL_LIGTH0, GL_DIFFUSE, ld)`, em que `ld` é o vector {1, 1, 1, 1}, e pretende que um objecto apareça no ecrã com um tom de verde "claro", como deve configurar o material desse objecto?

```
float md[] = {0.5, 1.0, 0.5, 1.0};
glMaterial(GL_FRONT_AND_BACK, GL_DIFFUSE, md)
```

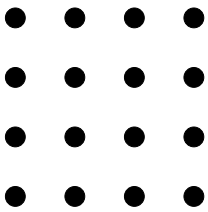
b. Preencha o seguinte vector de modo a definir uma luz direccional que forme um ângulo de 135° com o solo (plano XY) e seja co-planar com o plano XZ.

```
float lpos[] = {-1.0, 0.0, 1.0, 0.0};
glLigthfv(GL_LIGTH0, GL_POSITION, lpos);
```

c. Pretende definir uma vista em que a câmara está atrás do seu personagem. As variáveis `obj.x`, `obj.y` e `obj.z` representam a posição do personagem e `obj.dir` a orientação. Pretende que a câmara esteja a uma altura correspondente a 90% da altura do personagem (representada pela constante `ALTURA_OBJECTO`). Como deve configurar a câmara usando `gluLookAt`?

```
eyex = obj.x - DISTANCIA_CAMARA * cos(obj.dir);

eyey = obj.y - DISTANCIA_CAMARA * sin(obj.dir);

eyez = obj.z + 0.90 * ALTURA_OBJECTO;

gluLookAt( eyex, eyey, eyez,
           obj.x, obj.y, eyez,
           0.0, 0.0, 1.0);
```

d. Suponha que pretende modelar um terreno usando uma malha poligonal. Possui uma variável `Ponto3D terreno[N][M]` contendo vértices que se encontram distribuídos no espaço de forma semelhante à ilustrada na figura.



i. Se os vértices não possuírem a mesma cota, que tipo de polígono deve ser usado para representar a malha formada por estes vértices?

Triângulo

ii. Qual o melhor valor para o parâmetro a usar em `glBegin` nesse caso?

```
glBegin(GL_TRIANGLE_STRIP);
```

iii. E se todos os vértices possuíssem a mesma cota?

```
glBegin(GL_QUAD_STRIP);
```

N.º_____ Nome _____

e. O seguinte programa faz uma animação de um cubo ao longo do eixo dos YY. Suponha que as reticências correspondem ao código OpenGL que normalmente se encontra nessa função. Identifique (com círculos à volta) os pedaços de código correspondentes a cada um dos blocos principais da arquitectura de um programa gráfico.

```
#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;
```

Modelo

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

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

void desenhaCubo(void) { ... }

void Draw(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
        glLoadIdentity();
        glRotatef(angulo, 0, 1, 0);
        desenhaCubo();
        glFlush();
}
```

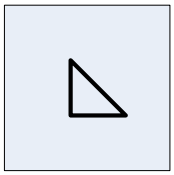
Renderer

```
void Timer(int value)
{
        glutTimerFunc(estado.delay, Timer, 0);

        angulo += 5.0f;

        // redesenhar o ecra
        glutPostRedisplay();
}

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

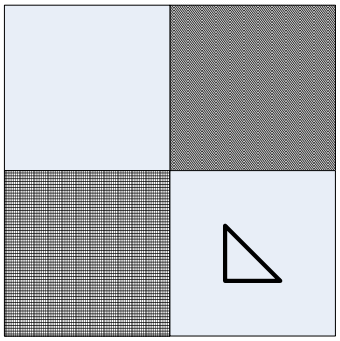
Input

Lógica

f. Suponha que pretende texturizar um quadrado por forma a ficar como na figura (a), usando a imagem de textura (b). Quais as coordenadas de textura que deve usar?



(a)



(b)

```
glBegin(GL_QUADS);
     glTexCoord2f(0.5, 0.0);
     glVertex3f(-2.0, -1.0, 0.0);
     glTexCoord2f(0.5, 0.5);
     glVertex3f(-2.0, 1.0, 0.0);
     glTexCoord2f(1.0, 0.5);
     glVertex3f(0.0, 1.0, 0.0);
     glTexCoord2f(1.0, 0.0);
     glVertex3f(0.0, -1.0, 0.0);
glEnd();
```

g. Considere uma cena constituída por dois objectos de dimensões iguais mas dispostos de forma a que um se encontra perto da câmara e o outro afastado.

   i. Se a cena resultante no ecrã mostrar os dois objectos com tamanho diferente está a usar que tipo de projecção?

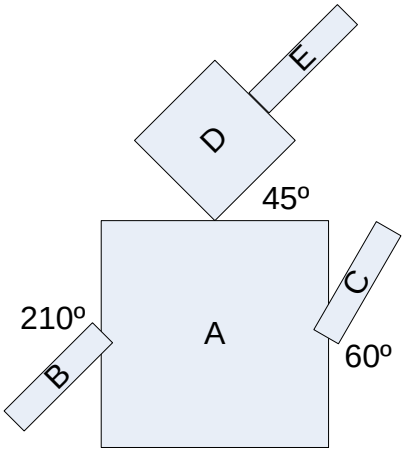
      Perspectiva

   ii. Qual a função da biblioteca principal do OpenGL que permite definir esse tipo de projecção?

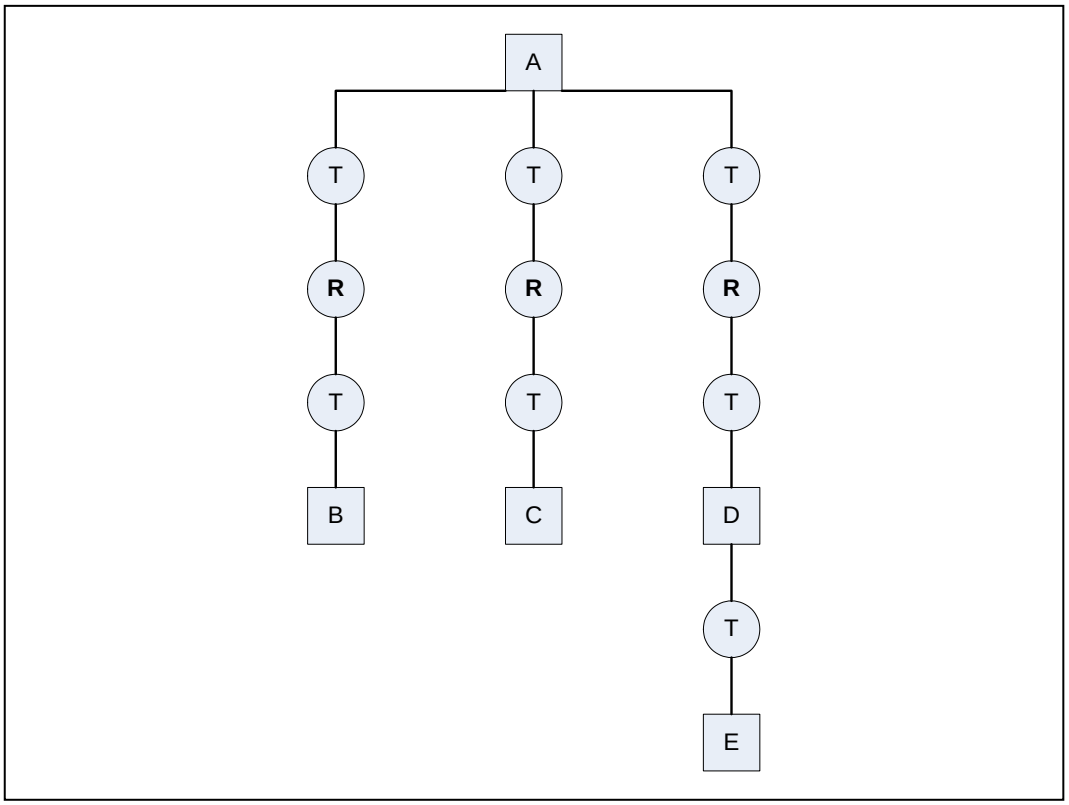
      `glFrustum()`

N.º_____ Nome _____

h. Pretende construir o objecto da figura recorrendo à função `desenhaParalelo(LX, LY, LZ)`, que desenha um paralelepípedo centrado na origem do sistema "local" de eixos, com dimensões LX, LY e LZ ao longo dos eixos X, Y e Z, respectivamente. Suponha que o elemento **A** tem lado *LA*; **B, C** e **E** têm comprimento *COMP* e altura *ALT*; o elemento **D** tem metade do lado de **A**; e todos os elementos têm a mesma profundidade *PROF*. Os ângulos de rotação dos elementos são dados considerando a orientação XY padrão do sistema cartesiano.



i. Desenhe a árvore de cena correspondente.



ii. Escreva a sequência de instruções OpenGL necessárias para desenhar os elementos **A**, **D** e **E** do objecto.

```
// elemento A
desenhaParalelo(LA, LA, PROF);
// elementos D e E
glPushMatrix();
  glTranslatef(0.0, LA / 2.0, 0.0);
  glRotatef(45.0, 0.0, 0.0, 1.0);
  glTranslatef(LA / 4.0, LA / 4.0, 0.0);
  desenhaParalelo(LA / 2.0, LA / 2.0, PROF);
  glTranslatef(LA / 4.0 + COMP / 2.0, 0.0, 0.0);
  desenhaParalelo(COMP, ALT, PROF);
glPopMatrix();
// elemento B (nao foi pedido)
glPushMatrix();
  glTranslatef(-LA / 2.0, 0.0, 0.0);
  glRotatef(210.0, 0.0, 0.0, 1.0);
  glTranslatef(COMP / 2.0, 0.0, 0.0);
  desenhaParalelo(COMP, ALT, PROF);
glPopMatrix();
// elemento C (nao foi pedido)
glPushMatrix();
  glTranslatef(LA / 2.0, 0.0, 0.0);
  glRotatef(60.0, 0.0, 0.0, 1.0);
  glTranslatef(COMP / 2.0, 0.0, 0.0);
  desenhaParalelo(COMP, ALT, PROF);
glPopMatrix();
```