Sistemas Gráficos e Interacção

Época Normal 2018-01-27

N.º _____ Nome _____

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

Parte Teórica **20%**

a. **[2.5]** Da comparação de uma interface WIMP (*Windows, Icons, Menus and Pointers*) com uma de linha de comandos resulta normalmente que

   i. A primeira está mais vocacionada do que a segunda para ser usada por utilizadores experientes
   ii. O esforço de memorização e a carga cognitiva impostos aos utilizadores são menores na primeira do que na segunda
   iii. A probabilidade de ocorrência de erros de interacção é maior na primeira do que na segunda
   iv. Nenhuma das anteriores

b. **[2.5]** Num sistema gráfico dotado de um *frame buffer* RGBA de 32 bits/píxel

   i. Cada píxel é descrito por 8 bits para a componente vermelha, 8 bits para a verde, 8 bits para a azul e 8 bits para o canal alfa
   ii. É possível reproduzir imagens com 2^8 = 256 níveis de vermelho, 256 níveis de verde e 256 níveis de azul
   iii. É possível reproduzir imagens de objectos transparentes
   iv. Todas as anteriores

c. **[2.5]** O **caso particular** da transformação de escala `glScaled(1.0, 1.0, 1.0);` constitui um exemplo de

   i. Uma transformação identidade
   ii. Uma transformação rígida
   iii. Uma transformação homotética
   iv. Todas as anteriores

d. **[2.5]** Qual das seguintes técnicas de codificação de malhas poligonais permite desenhar a malha sem que cada aresta seja desenhada duas vezes?

   i. Explícita
   ii. Ponteiros para uma lista de vértices
   iii. Ponteiros para uma lista de arestas
   iv. Nenhuma das anteriores

e. **[2.5]** O conhecimento do vector normal é necessário ao cálculo

   i. Das componentes ambiente e difusa de iluminação
   ii. Das componentes difusa e especular de iluminação
   iii. Das componentes especular e ambiente de iluminação
   iv. Nenhuma das anteriores

f. **[2.5]** No modelo de iluminação do OpenGL é possível definir um factor de atenuação da intensidade luminosa

   i. Constante, isto é, que não depende da distância da fonte de luz ao objecto iluminado
   ii. Linear, isto é, proporcional à distância da fonte de luz ao objecto iluminado
   iii. Quadrático, isto é, proporcional ao quadrado da distância da fonte de luz ao objecto iluminado
   iv. Todas as anteriores

g. **[2.5]** De que forma ou formas permite o mecanismo de mapeamento de texturas do OpenGL aplicar uma textura à superfície de um objecto?

   i. Misturando a cor da superfície com uma cor predefinida
   ii. Modulando a cor da superfície com a dos téxeis
   iii. Substituindo a cor da superfície pela dos téxeis
   iv. Todas as anteriores

h. **[2.5]** A correcção perspectiva permite

   i. Corrigir o efeito de discretização (*aliasing*) que decorre da utilização de *frame buffers* de baixa resolução
   ii. Corrigir o efeito de diminuição da dimensão aparente de um objecto quando a distância do mesmo à câmara aumenta
   iii. Corrigir o efeito de deformação que decorre da utilização de técnicas simples de interpolação linear no mapeamento de texturas em polígonos
   iv. Nenhuma das anteriores

Sistemas Gráficos e Interacção

Época Normal 2018-01-27

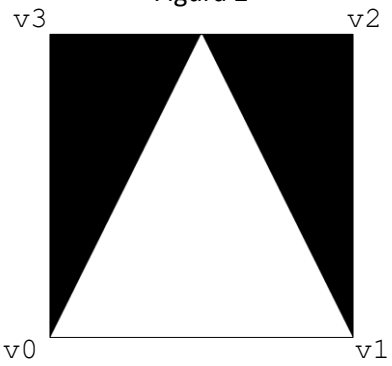
Parte Teórico-Prática **30%**

Resolução: No próprio enunciado
Perguntas de escolha múltipla: cada resposta incorrecta desconta 1/3 do valor da pergunta
Nota: Em todas as perguntas, a menos que algo seja dito em contrário, assuma a posição da câmara por omissão

a. **[3.0]** Aplique a textura apresentada na Figura 1 a um rectângulo, de modo a ficar com o aspecto apresentado na Figura 2.

Figura 1

Figura 2




```
glTexCoord2f(1.0, 0.0);
glVertex3fv(v0);
glTexCoord2f(1.0, 1.0);
glVertex3fv(v1);
glTexCoord2f(0.5, 1.0);
glVertex3fv(v2);
glTexCoord2f(0.5, 0.0);
glVertex3fv(v3);
```

ou

```
glTexCoord2f(1.0, 1.0);
glVertex3fv(v0);
glTexCoord2f(1.0, 0.0);
glVertex3fv(v1);
glTexCoord2f(0.5, 0.0);
glVertex3fv(v1);
glTexCoord2f(0.5, 1.0);
glVertex3fv(v1);
```

Observações:

- Resposta (1.0, 0.0); (1.0, 1.0); (0.0, 1.0); (0.0, 0.0): 2.0 valores
- Valores incorrectos, mas seguindo a lógica de só alterar uma coordenada por vértice: 1.0 valores

b. **[3.0]** Considere um cubo com material cor-de-laranja (1.0, 0.5, 0.0) iluminado por uma única fonte de luz azul-clara (0.0, 0.5, 1.0). Quais as componentes primárias (R, G, B) da cor resultante? **Indique os cálculos realizados**.

R = `1.0 * 0.0 = 0.0`

G = `0.5 * 0.5 = 0.25`

B = `0.0 * 1.0 = 0.0`

Observações:

- Cálculos correctos: 2.0 valores
- Resultado correcto: 1.0 valores
- 1 ou 2 resultados errados: 0.25 valores

c. **[3.0]** Considerando as definições por omissão do OpenGL, pretende-se definir a normal para o quadrilátero desenhado pelo seguinte extracto de código. Qual a normal unitária perpendicular ao quadrilátero?

```
glBegin(GL_QUADS);
     glNormal3f(0.0, 1.0, 0.0);
     glVertex3f(0, 1, 1);
     glVertex3f(1, 1, 1);
     glVertex3f(1, 1, 0);
     glVertex3f(0, 1, 0);
glEnd();
```

Observações:

- Cada coordenada correcta vale 1.0 valores
- Se for indicado sin()/cos() correcto, mas não o resultado, então vale 0.5 valores

d. **[3.0]** Pretende-se simular uma câmara em cima de um piloto de um carro de Fórmula 1 a olhar para a frente do carro. A posição do piloto é dada por `modelo.x`, `modelo.y` e `modelo.z`, a direcção que o carro está a seguir é dada por `modelo.dir`, e a altura a que a câmara fica da posição do piloto é dada por `ALTURA_CAMARA`. Complete a instrução seguinte de modo a conseguir a câmara pretendida, considerando como eixo vertical o eixo dos Z (positivo para cima).
Nota: A orientação da câmara é similar à visão normal do piloto durante a condução, olhando para a frente do carro.

```
gluLookAt(modelo.x, modelo.y, modelo.z + ALTURA_CAMARA,
          modelo.x + cos(modelo.dir),
          modelo.y + sin(modelo.dir),
          modelo.z + ALTURA_CAMARA,
          0.0, 0.0, 1.0);
```

Observações:

- 1.0 valores para cada linha
- **Linha 1:**
  - Falta “modelo…”: 0.25 valores
  - Falta apenas “+ ALTURA_CAMARA” em Z: 0.75 valores
- **Linha 2:**
  - Falta “modelo….” ou sin()/cos(): 0.25 valores
  - Falta apenas “+ ALTURA_CAMARA” em Z: 0.75 valores
  - Usar “x * cos()” em vez de “x + cos()”: 0.75 valores
  - Usar “DIST + cos()” em vez de “DIST * cos()”: 0.75 valores
- **Linha 3:**
  - (1.0, 0.0 ,0.0) ou (0.0, 1.0, 0.0): 0.25 valores
  - Trocar primeira e segunda linhas: 75%
  - Trocar Y e Z: 66%

e. Considere o objecto representado na Figura 3 e a existência da função `caixa()` que desenha um cubo com 1 unidade de lado, centrado na origem.

As dimensões dos elementos são *Lx*, *Ax* e *Px*, em que *x* designa o nome do elemento.

Considere que:

- O elemento A se desloca linearmente sobre o plano XZ;
- O elemento B roda em torno do vértice do elemento A;
- O elemento C roda em torno do ponto médio da lateral do elemento B.

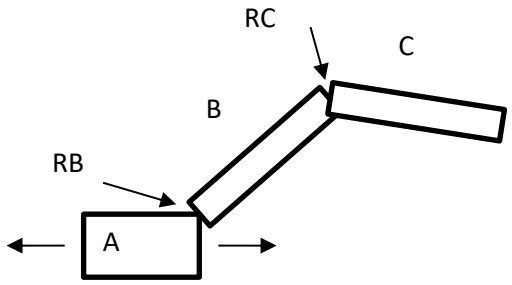
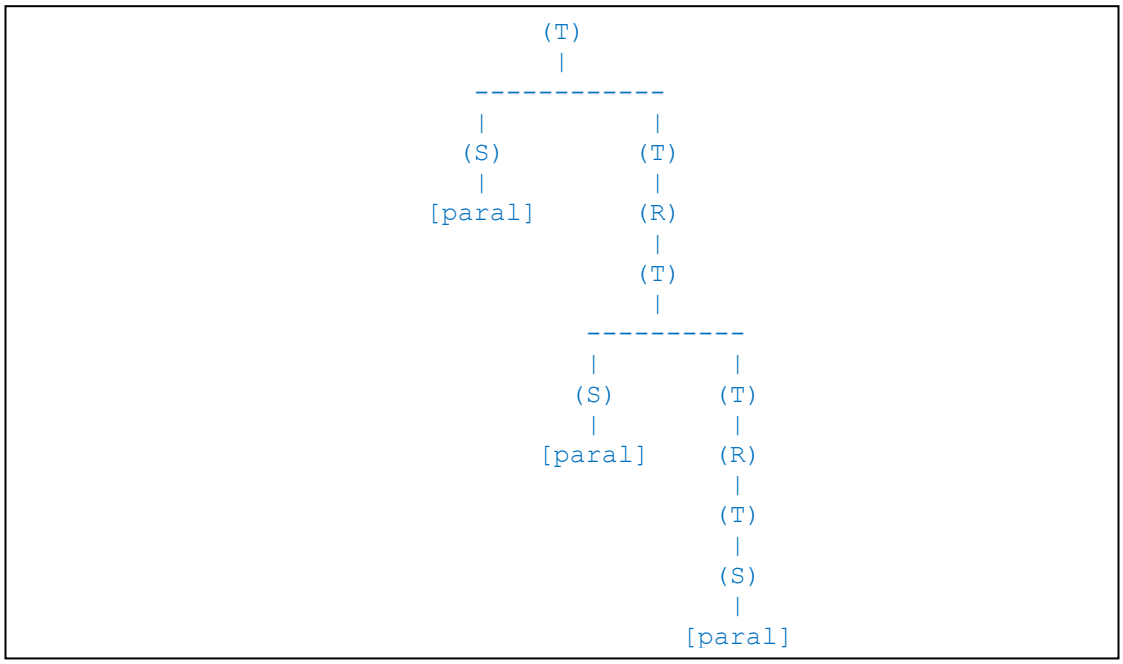


Figura 3

i. **[4.0]** Construa a árvore de cena do objecto apresentado na Figura 3. Não se esqueça de colocar as transformações que garantam o movimento dos elementos A, B e C.



Observações:

- Tudo em linha – 1.0 valores
- Tudo em paralelo – 1.0 valores
- Troca ou falta de 1 a 3 T/S – 0.25 valores cada um

ii. **[4.0]** Pretende-se controlar o deslocamento do elemento **A** com as teclas do cursor (`GLUT_KEY_LEFT` e `GLUT_KEY_RIGHT`). Complete o código que se segue para implementar o controlo do movimento pretendido, directamente no *callback* `glutSpecialFunc()`. Considere que as constantes `LEFT_LIMIT` e `RIGHT_LIMIT` definem os limites do deslocamento.

```
void SpecialKey(int key, int x, int y)
{
   switch(key) {
       case GLUT_KEY_RIGHT:
                   if(modelo.translacao < RIGHT_LIMIT)
                       modelo.translacao += INC_TRANSLACAO;
                   break;
       case GLUT_KEY_LEFT:
                   if(modelo.translacao > LEFT_LIMIT)
                       modelo.translacao -= INC_TRANSLACAO;
                   break;
   }
}
```

Observações:

- 1.0 valores: switch (key) + break ou if (key…)
- Falta o break: -0.5 valores
- Falta o switch (key): -0.5 valores
- 1.0 valores: case GLUT_KEY_LEFT/RIGHT ou usar GLUT_KEY_LEFT/RIGTH no if ()
- 1.0 valores: if (x > LEFT_LIMIT) … / RIGHT_LIMIT
- 1.0 valores: LEFT_LIMIT: x++ / RIGHT_LIMIT: x--
- Trocar sentido: -0.5 valores
- Usar variáveis diferentes para o if () e o ++/--: -0.5 valores