Sistemas Gráficos e Interacção

Época Normal 2016-01-27

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 **30%**

a. **[2.5]** Num sistema gráfico dotado de um *frame buffer* RGB de 1024 x 1024 píxeis, 8 bits/píxel

   i. Cada píxel é descrito por 3 bits para a componente vermelha, 3 bits para a verde e 2 bits para a azul
   ii. Cada píxel é descrito por 8 bits para a componente vermelha, 8 bits para a verde e 8 bits para a azul
   iii. Cada píxel é descrito por 1024 bits para a componente vermelha, 1024 bits para a verde e 1024 bits para a azul
   iv. Nenhuma das anteriores

b. **[2.5]** A realização de uma translação de um ponto *P* para um ponto *P'* distinto do anterior sem usar coordenadas homogéneas

   i. Não é de todo possível
   ii. É possível mediante uma simples multiplicação de matrizes
   iii. É possível, mas requer uma adição de matrizes
   iv. Nenhuma das anteriores

c. **[2.5**] No *pipeline* OpenGL de transformações

   i. A transformação de *viewport* converte as coordenadas normalizadas de dispositivo em coordenadas de janela
   ii. A matriz de projecção converte as coordenadas de recorte em coordenadas normalizadas de dispositivo
   iii. A matriz de modelação e visualização converte as coordenadas do objecto em coordenadas de recorte
   iv. Todas as anteriores

d. **[2.5]** Numa projecção perspectiva

   i. O volume de visualização tem a forma de um paralelepípedo e as dimensões aparentes dos objectos não dependem da distância à câmara
   ii. O volume de visualização tem a forma de um paralelepípedo e as dimensões aparentes dos objectos diminuem com o aumento da distância à câmara
   iii. O volume de visualização tem a forma de um tronco de pirâmide e as dimensões aparentes dos objectos não dependem da distância à câmara
   iv. O volume de visualização tem a forma de um tronco de pirâmide e as dimensões aparentes dos objectos diminuem com o aumento da distância à câmara

e. **[2.5]** Na representação por fronteira de um objecto (***B-rep***)

   i. É fácil exibir um ponto sobre a superfície do objecto
   ii. É fácil determinar, dado um ponto, se o mesmo se encontra no interior, na fronteira ou no exterior do objecto
   iii. As operações booleanas são avaliadas com facilidade
   iv. Nenhuma das anteriores

f. **[2.5]** No modelo de iluminação do OpenGL, o vector *halfway* é usado para calcular de uma forma simplificada

   i. A componente especular de iluminação
   ii. A componente difusa de iluminação
   iii. A componente ambiente de iluminação
   iv. Todas as anteriores

g. **[2.5]** Para iluminar uma cena com uma fonte de luz posicional do tipo projector, deverá

   i. Desactivar o mecanismo de mapeamento de texturas
   ii. Especificar para a posição um conjunto de coordenadas tal que w = 0
   iii. Especificar para o ângulo de *cutoff* um valor compreendido entre 0^o e 90^o
   iv. Todas as anteriores

h. **[2.5]** A técnica de filtragem de texturas designada por *mipmapping*

   i. É aplicável aos contextos de magnificação
   ii. Permite que texturas de diferentes níveis de resolução sejam aplicadas de forma adaptativa
   iii. Não pode ser usada em conjugação com as projecções perspectivas
   iv. Nenhuma das anteriores

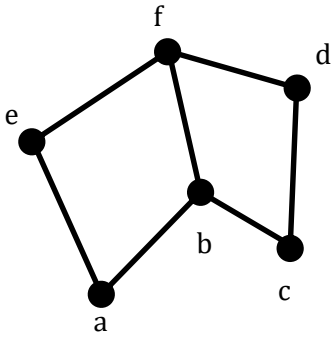
Sistemas Gráficos e Interacção

Época Normal 2016-01-27

Parte Teórico-Prática **40%**

Resolução: Em folhas próprias e separadas (A4). Não se esqueça de indicar, em cada folha, o número de estudante e o nome completo
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. **[2.0]** Suponha que pretende modelar o seguinte objecto com o recurso à primitiva GL_QUAD_STRIP. Transcreva para a folha da prova o seguinte extracto de código, completando as instruções `glVertex3fv()` com as letras que considerar adequadas. A primeira já se encontra preenchida.




```
glBegin(GL_QUAD_STRIP);
      glVertex3fv(  a  );
      glVertex3fv(__e__);
      glVertex3fv(__b__);
      glVertex3fv(__f__);
      glVertex3fv(__c__);
      glVertex3fv(__d__);
glEnd();
```

b. **[2.0]** Qual é a utilidade da função `glEdgeFlag()` do OpenGL?

Permite marcar arestas que começam num determinado vértice como arestas de fronteira ou não. É usado para esconder as linhas interiores resultantes da tesselação.

c. **[2.0]** Qual é a importância do sentido com que se desenham os vértices de um polígono em OpenGL?

O sentido de desenho dos vértices é importante para a orientação da face da frente dos polígonos.

Nota 1: Por omissão, a face da frente (GL_FRONT) fica do lado em que os vértices são dados no sentido contrário ao do movimento dos ponteiros de um relógio (CCW).

Nota 2: Saber orientar um polígono é importante, por exemplo, para a definição de normais e a utilização do mecanismo de "*back face culling*".

Nota: mude para uma nova folha de respostas

d. **[2.0]** Considere um programa gráfico com uma janela principal e duas subjanelas. Indique o que terá de fazer para que o conteúdo das três janelas seja actualizado ao mesmo tempo.

É necessário invocar a função `glutSetWindow()`, seguida da chamada à função `glutPostRedisplay()`, para cada uma das janelas/subjanelas.

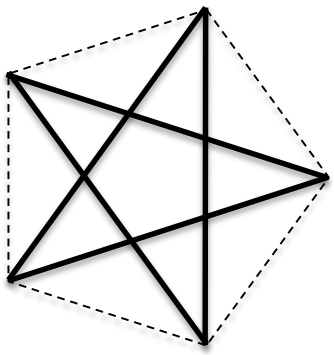
e. **[2.0]** Pretende-se simular a vista de uma câmara a seguir um automóvel de competição. Este está localizado em (`obj.x`, `obj.y`, `obj.z`) e desloca-se na direcção `obj.dir`. A câmara está situada na posição (`cam.x`, `cam.y`, `cam.z`). Escreva o extracto de código que implementa a referida vista.

```
gluLookAt(  cam.x  ,  cam.y  ,  cam.z  ,
            obj.x  ,  obj.y  ,  obj.z  ,
             0.0   ,   0.0   ,   1.0   );
```

Nota: mude para uma nova folha de respostas

f. **[4.0]** Crie a função `pentagrama()` que desenha o objecto representado com linhas contínuas na figura que se segue. O objecto deve ser desenhado com linhas.

Notas: O pentagrama está inscrito num círculo com o mesmo raio.
Não é necessário desenhar o pentágono ilustrado a tracejado.



`void pentagrama(GLfloat xCentro, GLfloat yCentro, GLfloat raio)`

```
void pentagrama(GLfloat xCentro, GLfloat yCentro, GLfloat raio)
{
  GLint i, n = 5;
  GLfloat ang = 2.0 * M_PI / (GLfloat) n;

  glBegin(GL_LINE_LOOP);
  for(i = 0; i < n; i++)
  {
    glVertex2f(xCentro + raio * cos(2.0 * i * ang),
               yCentro + raio * sin(2.0 * i * ang));
  }
  glEnd();
}
```

ou

```
void pentagrama(GLfloat xCentro, GLfloat yCentro, GLfloat raio)
{
  GLint i, n = 5;
  GLfloat ang = 2.0 * M_PI / (GLfloat) n, t = 0;

  glBegin(GL_LINE_LOOP);
  for(i = 0; i < n; i++)
  {
    glVertex2f(xCentro + raio * cos(t),
               yCentro + raio * sin(t));
    t += 2.0 * ang;
  }
  glEnd();
}
```

ou (mais complexo...)

```
void pentagrama(GLfloat xCentro, GLfloat yCentro, GLfloat raio)
{
  GLint i, n = 5;
  GLfloat ang = 2.0 * M_PI / (GLfloat) n;

  glPushMatrix();
    for(i = 0; i < n; i++)
    {
      glBegin(GL_LINES);
        glVertex2f(xCentro + raio, yCentro);
        glVertex2f(xCentro + raio * cos(2.0 * ang),
                   yCentro + raio * sin(2.0 * ang));
      glEnd();
      glRotatef(GRAUS(ang), 0.0, 0.0, 1.0);
    }
  glPopMatrix();
}
```

Nota: mude para uma nova folha de respostas

g. Pretende-se realizar uma animação contínua do pentagrama na seguinte sequência (use as constantes e variáveis do modelo que entender necessárias):

- Aumenta o raio de 5 unidades em pequenos incrementos;
- Roda continuamente 90º no sentido dos ponteiros do relógio;
- Diminui o raio de 5 unidades em pequenos decrementos;
- Roda continuamente 90º no sentido dos ponteiros do relógio;
- Continua com o primeiro passo...

i. **[3.0]** Implemente a função `desenhaCena()` que desenhe o pentagrama e permita realizar esta animação.

```
void desenhaCena()
{
  glClear(GL_COLOR_BUFFER_BIT);
  glPushMatrix();
    glRotatef(modelo.angulo, 0.0, 0.0, 1.0);
    pentagrama(0.0, 0.0, modelo.raio);
  glPopMatrix();
  glFlush();
  if (estado.doubleBuffer)
    glutSwapBuffers();
}
```

ii. **[3.0]** Implemente a função `timer()` para automatizar a animação.

```
void timer(int value)
{
  glutTimerFunc(estado.delay, timer, 0);
  switch (modelo.anim) {
    case 1:
    case 3:
       modelo.raio += modelo.inc;
       if (modelo.raio <= RAIO_INI ||
           modelo.raio >= RAIO_INI + 5.0) {
         modelo.inc *= -1.0;
         modelo.anim++;
       }
      break;
    case 2:
    case 4:
          modelo.angulo++;
          if (modelo.angulo >= modelo.anguloInicial + 90.0) {
            modelo.anguloInicial = modelo.angulo;
            if (modelo.anim < 4)
              modelo.anim++;
            else
              modelo.anim = 1;
          }
      break;
  }
  glutPostRedisplay();
}
```