Computação Gráfica e Interfaces (LEIC)
Sistemas Gráficos (LEEC)

Enunciados de exercícios para as aulas práticas

2. Shading e Smooth-shading

O objectivo deste grupo de exercícios é aplicar os conceitos de Iluminação Constante (*Flat Shading*), Iluminação Suavizada (*Smooth Shading*) pelo método de Gouraud e de Textura, em superfícies originalmente planas e em superfícies curvas aproximadas por malhas poligonais.

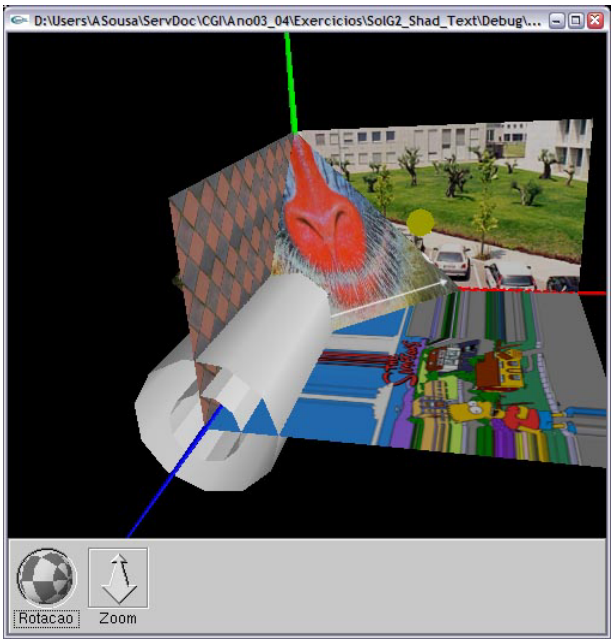

Fig. 1 - Cena 3D em OpenGL

A cena 3D da Fig. 1 constitui-se de, além dos três eixos e da fonte de luz (com respectiva esfera simbólica), de três "paredes" (rectângulos coincidentes com os planos do sistemas de coordenadas), um prisma e um cilindro coincidentes com o eixo dos ***z*** (o cilindro é o que possui maior raio) e uma pirâmide quadrangular sem base. Vários dos objectos possuem texturas.

Para este conjunto de problemas (**2. Shading e Smooth-shading**), só serão tratados o prisma, o cilindro e a "parede" do plano ***xy***.

No arquivo G2_Shad_Text é fornecido um conjunto de ficheiros com parte do código para a criação daquela cena. Para este conjunto de problemas, o ficheiro RGBpixmap.cpp e respectivo *header file* não são necessários. Na zona de declarações globais do programa encontram-se as principais grandezas utilizadas, nomeadamente nas funções `display()` e `inicializacao()`.

Compile e faça correr o respectivo código. Como se pode observar, a cena inicial é constituída pelos eixos, fonte de luz e parede do plano ***xy***.

Nota: Para uma boa percepção dos exercícios seguintes, leia o manual *OpenGL Programming Guide*, designadamente a secção ***Specifying a Shading Model*** (pág. 109). Para esclarecimento de algumas funções particulares, pode consultar-se o manual **openglman.html**. Deve também ter alguns conhecimentos teóricos sobre *Shading* e *Smooth Shading*, nomeadamente sobre o método de Gouraud.

1. Crie a função `myCylinder()`, cujo cabeçalho já existe no código, e utilize-a para desenhar o prisma da imagem (as dimensões encontram-se definidas na zona de declarações do programa, sob o nome cilindro 1). Nesta fase, ignore o parâmetro `smooth`.

2. Acrescentando o processamento do parâmetro `smooth` (Booleano) na função `myCylinder()`:
   a) Modifique a função de forma permitir desenhar prismas (superfície em malha de polígonos planos) e cilindros (superfície arredondada por efeito do método de Gouraud para *Smooth Shading*).
   b) Utilizando a função assim obtida, faça desenhar o cilindro 2 com as dimensões que constam nas declarações do programa.
   c) Experimente vários valores do parâmetro `slices`.
   d) Compare com o resultado obtido pela função `gluCylinder()`.

3. Qual é o efeito da iluminação suavizada em cada um dos 3 objectos assim constituídos (dois cilindros e "parede" no plano ***xy***)?

4. Altere, na função `inicializacao()`, a instrução `glShadeModel(GL_SMOOTH)` para `glShadeModel(GL_FLAT)`.
   a) Que modificações encontra nos mesmos objectos?
   b) Dependendo da implementação do OpenGL, os rectângulos podem aparecer como conjuntos de dois triângulos. Relacione as Iluminações Constante e Suavizada com estes rectângulos/triângulos.

3. Mapeamento de Texturas

O código inicial disponibilizado inclui um ficheiro RGBpixmap.cpp com uma classe que contém dois métodos principais: `readBMPFile()` e `setTexture()`. Estes dois métodos são evocados na parte final da função `inicializacao()` do ficheiro principal para declarar as quatro texturas a utilizar neste grupo de trabalhos práticos:

> `readBMPFile()`: o ficheiro com a imagem é lido e esta é armazenada no *array "pixel"*;
>
> `setTexture()`: o contéudo de *pixel* é passado para o OpenGL que o armazena em estrutura própria; são efectuadas outras inicializações do OpenGL relacionadas com texturas.

Partindo da cena 3D que se obteve no grupo de exercícios anterior (**2. Shading e Smooth-shading**), pretende-se mapear as quatro texturas fornecidas (`clamp.bmp`, `feup.bmp`, `mandril`, `tile.bmp`) em alguns objectos, de acordo com a Fig. 1.

Comece por retirar de comentários as seguintes instruções da função `inicializacao()`:

```
pixmap.readBMPFile("<nome de ficheiro>");
pixmap.setTexture(<n>);
```

Assim como as instruções seguintes da função `display()`:

```
glEnable(GL_TEXTURE_2D);
....
glDisable(GL_TEXTURE_2D);
```

Substitua o desenho da "parede" do plano XY pelo conjunto de instruções seguinte:

```
// desenha a "parede" do plano XY
glBindTexture(GL_TEXTURE_2D, 1);    // activa a textura 1 (feup)
glBegin(GL_POLYGON);
      glNormal3d(0.0,0.0,1.0);
      glTexCoord2f(0.0,0.0); glVertex3d( 0.0,  0.0,  0.0);
      glTexCoord2f(1.0,0.0); glVertex3d(dimx,  0.0,  0.0);
      glTexCoord2f(1.0,1.0); glVertex3d(dimx, dimy,  0.0);
      glTexCoord2f(0.0,1.0); glVertex3d( 0.0, dimy,  0.0);
glEnd();
```

Ao correr o programa, pode verificar que este conjunto de instruções mapeia a textura com o identificador 1 (ficheiro `feup.bmp`, ver parte final da função `inicialização()`) sobre o polígono em questão.

1. Como se vê pelo código acima, cada vértice do polígono relaciona-se com um vértice da textura. Avalie o que sucederia se, mantendo-se a ordem dos vértices no polígono, se fizesse "rodar" de uma posição os vértices da textura, i.e.:

```
      glTexCoord2f(0.0,1.0); glVertex3d( 0.0,  0.0,  0.0);
      glTexCoord2f(0.0,0.0); glVertex3d(dimx,  0.0,  0.0);
      glTexCoord2f(1.0,0.0); glVertex3d(dimx, dimy,  0.0);
      glTexCoord2f(1.0,1.0); glVertex3d( 0.0, dimy,  0.0);
```

2. Verifique que a relação comprimento/largura da textura (a imagem é quadrada...) não está de acordo com as dimensões do respectivo polígono.

3. Implemente a correcção dessa relação, actuando sobre as coordenadas ***s,t*** da textura, de forma a cortar automaticamente o necessário na altura da textura.

4. Desenhe a "parede" do plano ***yz*** (dimensões na zona de declarações do programa) e preencha-a com a textura 2 (`tile.bmp`), distribuída em formato mosaísta (*tiling*) com 5*5 repetições. Deve, para o efeito, tomar conhecimento com a função `glTexParameter*()`.

a) Verifique que a proporção comprimento/largura da textura assim obtida não é correcta.

b) Corrija o problema anterior, atendendo à relação comprimento/largura do polígono e desenhando, em uma das direcções, o número adequado de réplicas para garantir a correcção da proporção.

5. Desenhe o "chão" e mapeie a textura 3 (`clamp.bmp`) em modo *clamping*, iniciando com:
(*s*, *t*) = (-0.2, -0.2) no vértice ***x*=10.0**, ***y*=0.0**, ***z*=10.0**

6. Desenhe as quatro faces laterais da pirâmide quadrangular.

a) O eixo da pirâmide coincide com o eixo dos ***y***. Os cinco vértices correspondem a pontos sobre os eixos do sistema de coordenadas com a respectiva coordenada em 1. Cada uma das quatro normais pode assim ser definida pelo vector

(±1.0, +1.0, ±1.0)

depois de devidamente normalizado.

b) Mapeie a textura 4 (`mandril.bmp`) sobre a pirâmide assim obtida. O centro da textura deve coincidir com o vértice superior da pirâmide; deve existir coincidência lateral das texturas de duas faces vizinhas.

7. Mapeie a totalidade da textura `feup.bmp` sobre a superfície do cilindro exterior.
