IntMu.Lab5

0.

Importe os ficheiros disponibilizados em http://www.dee.isep.ipp.pt/~jml/intmu/lab5:

```
wget http://ave.dee.isep.ipp.pt/~jml/intmu/lab5/Makefile
make getall
```

Analise detalhadamente o programa **moinho.c**. Compile-o e execute o programa **moinho**

```
make
./moinho
```

Deve obter uma janela vazia.

0.1. Base

A função **Display()** é chamada automaticamente de cada vez que a janela necessita de ser redesenhada. Nesta função acrescente o desenho de um quadrado horizontal (no plano ***xy***) com 6 unidades de lado.

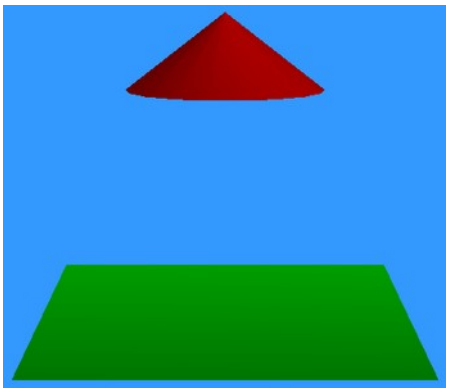
```
glColor3f(0., 0.7, 0.);
glBegin(GL_QUADS);
    glVertex3f(-3., -3., 0.);
    glVertex3f(-3.,  3., 0.);
    glVertex3f( 3.,  3., 0.);
    glVertex3f( 3., -3., 0.);
glEnd();
```

1. Moinho

A função **moinho()** está a ser chamada durante a execução da função de desenho **Display()**. Pretende-se que a função **moinho()** apresente uma representação gráfica de um moinho de vento tradicional.

1.1. Cone

Na função **moinho()**, utilize a funcionalidade **glutSolidCone** para criar um cone assente no plano ***xy***, com raio da base de uma unidade e altura de 1 unidade, com cor vermelha.



```
glColor3f(0.9, 0., 0.);
glutSolidCone(1., 1., 32, 1);
```

Altere a posição do cone através de uma translação de 3 unidades na vertical.

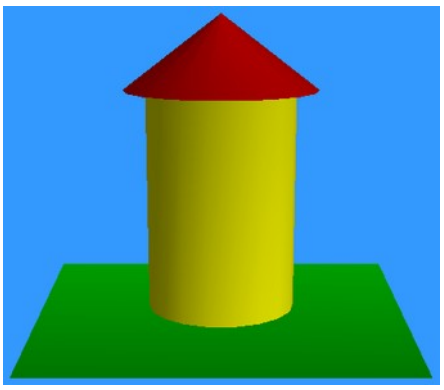
```
glTranslatef(0., 0., 3.);
```

Aplique ao cone um escalamento de forma a ampliar o raio da base em 20%.

```
glScalef(1.2, 1.2, 1.);
```

1.2. Cilindro

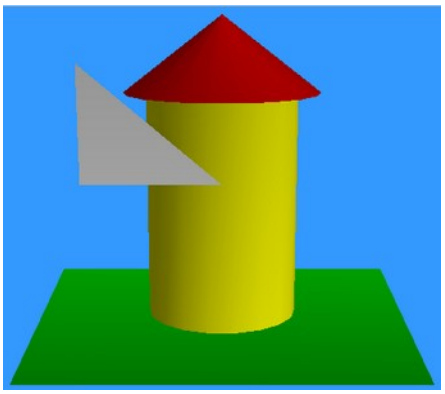
Acrescente um cilindro vertical para representar o corpo do moinho.



```
glColor3f(0.9, 0.9, 0.);
glPushMatrix();
    glScalef(1., 1., 3.);
    glutSolidCylinder(1., 1., 32, 1);
glPopMatrix();
```

1.3. Pá

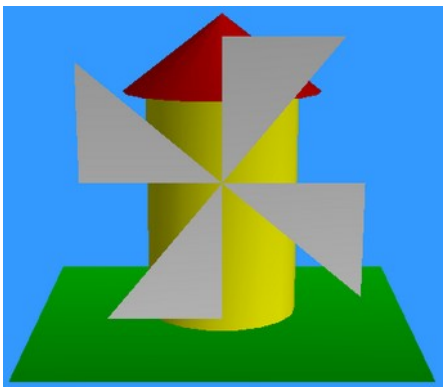
Acrescente um elemento triangular para representar uma das pás do moinho.



```
glColor3f(1.,1.,1.);
glPushMatrix();
  glTranslatef(0., -1.25, 2.4);
  glNormal3f(0., -1., 0.);
  glBegin(GL_TRIANGLES);
     glVertex3f( 0.,  0.,  0.);
     glVertex3f( -1.8, -0.1, 1.3);
     glVertex3f( -1.8, 0.,  0.);
  glEnd();
glPopMatrix();
```

1.4. Pás

Reutilize a especificação da pá para gerar as restantes pás através de rotações.



```
for( i=0 ; i<4 ; i++ )
{
  glBegin(GL_TRIANGLES);
     glVertex3f( 0., 0.,  0.);
     glVertex3f( 1.8, 0., 1.5);
     glVertex3f( 1.8, 0.,  0.);
  glEnd();
  glRotatef(90., 0., 1., 0.);
}
```

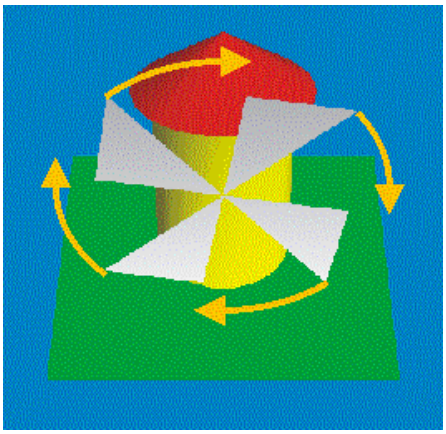
2. Animação

Aplique ao conjunto das pás, uma rotação em torno do eixo y com velocidade constante.

```
glRotatef(0.05*timenow, 0., 1., 0.);
```

Verifique que, sempre a janela é redesenhada, a posição das pás é alterada.

Para impor o redesenho constante da janela, crie uma nova função de *callback*:



```
void  Redisplay(void)
{
    glutPostRedisplay();
}
```

Defina esta nova função como o *callback* a utilizar durante estado **idle**.

```
glutIdleFunc(Redisplay);
```

3. Iluminação

Os seus objetos devem estar representados com as cores uniformes (*flat shade*) especificadas através da função **glColor3f()**. Para obter um efeito mais realista pode ser utilizado o cálculo de iluminação implementado na *pipeline* gráfica.

Ative o cálculo de iluminação:

```
glEnable (GL_LIGHTING);
```

Verifique que os objetos perderam a cor.

3.1 Fonte de luz

Crie uma fonte de luz pontual (`GL_LIGHT0`) na função **Init()**:

```
float light_ambient[] = { 0.2, 0.2, 0.2, 1.0 };
float light_diffuse[] = { 0.35, 0.35, 0.35, 1.0 };
float light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
float light_position[] = { 30.0, -10.0, 30.0, 1.0 };

glLightfv (GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv (GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv (GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv (GL_LIGHT0, GL_POSITION, light_position);
```

Ative a luz definida:

```
glEnable (GL_LIGHT0);
```

Verifique que os objetos já apresentam sinais de iluminação.

3.2 Materiais

Prepare uma função para definir as propriedades óticas dos materiais.

```
void SetMaterial(float r, float g, float b)
{
  float mat_dif[] = { r, g, b, 1.0 };
```

```
    float mat_amb[] = { r, g, b, 1.0 };
    float mat_spe[] = { 0.2, 0.2, 0.2, 1.0 };
    float mat_shi = 10.;

    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_amb);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_dif);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_spe);
    glMaterialf( GL_FRONT, GL_SHININESS, mat_shi);
}
```

Aplique esta função para definir as propriedades dos materiais utilizados, substituindo as chamadas à função **glColor3f():**

```
SetMaterial(0.9, 0.,  0.);
```

4. Texturas

Utilize a biblioteca SOIL (*Simple OpenGL Image Loader*) para carregar uma imagem para mapear sobre o polígono de terreno.

```
#include <SOIL/SOIL.h>
...
int width, height;
unsigned char* image =
    SOIL_load_image("grass.jpg", &width, &height, 0, SOIL_LOAD_RGB);
glBindTexture(GL_TEXTURE_2D, 0);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB,
    GL_UNSIGNED_BYTE, image);
SOIL_free_image_data(image);
```

Na função de desenho (Display), ative a textura carregada para decorar o polígono do solo.

```
glEnable(GL_TEXTURE_2D);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
glBindTexture(GL_TEXTURE_2D, 0);
```

Para cada um dos 4 vértices do polígono de terreno, especifique as coordenadas de textura correspondentes a um dos cantos da imagem

```
glBegin(GL_QUADS);
    glTexCoord2f(0.0, 0.0); glVertex3f(-6., -6., 0.);
    glTexCoord2f(0.0, 1.0); glVertex3f(-6., 6., 0.);
    glTexCoord2f(1.0, 1.0); glVertex3f( 6., 6., 0.);
    glTexCoord2f(1.0, 0.0); glVertex3f( 6., -6., 0.);
glEnd();
```

Após o desenho do polígono de solo, desligue o mapeamento de texturas para não afetar os outros elementos da cena.

```
glDisable(GL_TEXTURE_2D);
```

5. Entrega

Para entregar o trabalho, submeta os ficheiros **moinho.c** e **moinho** através da atividade Lab5 de área de INTMU do Moodle. Se utilizou ficheiros adicionais (por exemplo imagens) diferentes dos propostos, submeta-os também.

6. Bibliografia

- **The OpenGL Programming Guide**: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1 (7th Edition) **,** Dave Shreiner, ISBN 978-0-321-55262-4
- **OpenGLBook.com**, http://openglbook.com/

7. Requisitos

- SOIL (Sample OpenGL Image Loader), Source: http://www.lonesock.net/soil.html
- freeGLUT, http://freeglut.sourceforge.net/

Instalação (em Fedora):

```
su -c "dnf install SOIL SOIL-devel freeglut freeglut-devel"
```

ou:

```
make inst_reqs
```

8. Histórico

- V1, 2000, jml
- V2, 2004, jml
- V3, 2013, jml
- V3.1, 2014, jml
- V4.1, 2016, jml
- V4.2, 2017, jml
- V4.3, 2020, jml
- V4.4, 2021, jml