Esqueleto 3D

P. J. Martín, A. Gavilanes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Esqueleto para 3D

```
// main.cpp
…
// Viewport size
int WIDTH= 500, HEIGHT= 500;

// Viewing frustum parameters
GLdouble xRight=10, xLeft=-xRight,
         yTop=10, yBot=-yTop,
         N=1, F=1000;

// Camera parameters
GLdouble eyeX=100.0, eyeY=100.0, eyeZ=100.0;
GLdouble lookX=0.0, lookY=0.0, lookZ=0.0;
GLdouble upX=0, upY=1, upZ=0;
```

Esqueleto para 3D

```
// Scene variables
GLfloat angX, angY, angZ;

void buildSceneObjects() {
    angX=0.0f;
    angY=0.0f;
    angZ=0.0f;
}
```

Esqueleto para 3D

```
void initGL() {
   // Background color
   glClearColor(0.6f,0.7f,0.8f,1.0);

   glEnable(GL_COLOR_MATERIAL);
   glMaterialf(GL_FRONT, GL_SHININESS, 0.9);
   glEnable(GL_DEPTH_TEST);
   glEnable(GL_NORMALIZE);
   // Shading by default
   glShadeModel(GL_SMOOTH);

   buildSceneObjects();
   …
```

Esqueleto para 3D

```
// Light0
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
GLfloat d[]={0.7,0.5,0.5,1.0};
glLightfv(GL_LIGHT0, GL_DIFFUSE, d);
GLfloat a[]={0.3,0.3,0.3,1.0};
glLightfv(GL_LIGHT0, GL_AMBIENT, a);
GLfloat s[]={1.0,1.0,1.0,1.0};
glLightfv(GL_LIGHT0, GL_SPECULAR, s);
GLfloat p[]={0.0, 0.0, 0.0, 1.0};
glLightfv(GL_LIGHT0, GL_POSITION, p);
…
```

Esqueleto para 3D

```
    …
    // Camera set up
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyeX,eyeY,eyeZ,lookX,lookY,lookZ,upX,upY,upZ);

    // Frustum set up
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(xLeft, xRight, yBot, yTop, N, F);

    // Viewport set up
    glViewport(0, 0, WIDTH, HEIGHT);
}
```

Display callback

```
void display(void) {
   glClear(GL_COLOR_BUFFER_BIT |
           GL_DEPTH_BUFFER_BIT);

   glMatrixMode(GL_MODELVIEW);
   glPushMatrix();

   // Rotating the scene
   glRotatef(angX, 1, 0, 0);
   glRotatef(angY, 0, 1, 0);
   glRotatef(angZ, 0, 0, 1);
```

Display callback

```
// Drawing axes
glBegin( GL_LINES );
      glColor3f(1.0,0.0,0.0);
      glVertex3f(0, 0, 0);
      glVertex3f(20, 0, 0);

      glColor3f(0.0,1.0,0.0);
      glVertex3f(0, 0, 0);
      glVertex3f(0, 20, 0);

      glColor3f(0.0,0.0,1.0);
      glVertex3f(0, 0, 0);
      glVertex3f(0, 0, 20);
glEnd();

// Drawing the scene
glColor3f(1.0, 1.0, 1.0);
glutSolidSphere(6, 50, 60); // Sphere: radius=6, meridians=50, parallels=60
glPopMatrix();
```

…

Key callback

```
void key(unsigned char key, int x, int y) {
   bool need_redisplay = true;
   switch (key) {
        case 27:  /* Escape key */
        //continue_in_main_loop = false; // (**)
        //Freeglut's sentence for stopping glut's main loop (*)
               glutLeaveMainLoop ();
               break;
        case 'a': angX=angX+5; break;
        case 'z': angX=angX-5; break;
        case 's': angY=angY+5; break;
        case 'x': angY=angY-5; break;
        case 'd': angZ=angZ+5; break;
        case 'c': angZ=angZ-5; break;
   …
```

Resize callback

```
   …
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(xLeft, xRight, yBot, yTop, N, F);
}
```

Las matrices de OpenGL

- Antes de definir el volumen de vista es necesario cargar la matriz de proyección, y hacerla la identidad.

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(xLeft, xRight, yBot, yTop, N, F);
```

- De igual forma, tras establecer el volumen de vista y antes de dibujar nada, es necesario cargar como matriz actual la de modelado y vista, y hacerla la identidad.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(eyeX, eyeY, eyeZ,
          lookX, lookY, lookZ, upX, upY, upZ);
```

- La matriz del puerto de vista se construye cuando establecemos el puerto de vista dentro de la ventana. Conviene que ocupe todo el área de la ventana.

```
glViewport(0, 0, WIDTH, HEIGHT);
```