Esqueleto 2D

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

Esqueleto para 2D

```
// main.cpp
#include <Windows.h>
#include <gl/GL.h>
#include <gl/GLU.h>

#include <GL/freeglut.h>

// Freeglut parameters
int WIDTH= 500, HEIGHT= 250;

// OpenGL parameters
GLdouble xLeft= 0.0, xRight= 500.0;
GLdouble yBot= 0.0, yTop= 250.0;
```

Esqueleto para 2D

```
int main(int argc, char *argv[]){

  int my_window; //identificador de la ventana

  // Inicialización
  glutInitWindowSize(WIDTH, HEIGHT);
  glutInitWindowPosition (140, 140);
  glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE );
  glutInit(&argc, argv);

  //Construcción de la ventana
  my_window = glutCreateWindow( "Freeglut 2D-project" );

  //Registro de callbacks
  glutReshapeFunc(resize);
  glutKeyboardFunc(key);
  glutDisplayFunc(display);
```

Esqueleto para 2D

```
  intitGL();

  //Main loop can be stopped after X-closing the
  //window using the following Freeglut's setting
  glutSetOption (GLUT_ACTION_ON_WINDOW_CLOSE,
                 GLUT_ACTION_CONTINUE_EXECUTION);

  //Main loop
  glutMainLoop();

  // We never reach this point using classic Glut
  system("PAUSE");

  return 0;
} //main
```

Display callback

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

  // Scene rendering
  // xTriangle, yTriangle
  // triangleWidth, triangleHeight are scene variables
  glBegin (GL_TRIANGLES);
       glVertex2d(xTriangle, yTriangle);
       glVertex2d(xTriangle + triangleWidth, yTriangle);
       glVertex2d(xTriangle + triangleWidth,
                  yTriangle + triangleHeight);
  glEnd () ;

  glFlush();
  glutSwapBuffers();
}
```

Resize callback

```
void resize(int newWidth, int newHeight){

  //Resize Viewport
  WIDTH= newWidth;
  HEIGHT= newHeight;
  GLdouble RatioViewPort= (float)WIDTH/(float)HEIGHT;
  glViewport(0, 0, WIDTH, HEIGHT);

  //Resize Scene Visible Area
  //Se actualiza el área visible de la escena
  //para que su ratio coincida con ratioViewPort
  GLdouble SVAWidth= xRight-xLeft;
  GLdouble SVAHeight= yTop-yBot;
  GLdouble SVARatio= SVAWidth/SVAHeight;
```

Resize callback

```
if (SVARatio >= RatioViewPort) {
   // Increase SVAHeight
   GLdouble newHeight= SVAWidth/RatioViewPort;
   GLdouble yMiddle= ( yBot+yTop )/2.0;
   yTop= yMiddle + newHeight/2.0;
   yBot= yMiddle - newHeight/2.0;
}
else {
   //Increase SVAWidth
   GLdouble newWidth= SVAHeight*RatioViewPort;
   GLdouble xMiddle= ( xLeft+xRight )/2.0;
   xRight= xMiddle + newWidth/2.0;
   xLeft=  xMiddle - newWidth/2.0;
}
```

Resize callback

```
 //Resize Scene Visible Area
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(xLeft, xRight, yBot, yTop);

} //resize
```

Key callback

```
void key(unsigned char key, int x, int y){
  bool need_redisplay = true;
  switch (key) {
  case 27:  /* Escape key */
    glutLeaveMainLoop(); //Freeglut's sentence for stopping main loop
    break;

  case '+' :
    xTriangle += 10.0; yTriangle += 10.0;  break ;

  case '-' :
    xTriangle -= 10.0; yTriangle -= 10.0;  break ;

  default:
    need_redisplay = false;  break;
  }//switch

  if (need_redisplay) glutPostRedisplay();
}
```

Inicialización de OpenGL

```
void intitGL(){

  glClearColor(1.0,1.0,1.0,1.0);
  glColor3f(1.0,0.0,0.0);

  glPointSize(4.0); //Point size
  glLineWidth(2.0); //Line width

  //Viewport setting
  glViewport(0, 0, WIDTH, HEIGHT);

  //Model transformation
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();

  //Scene Visible Area setting
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(xLeft, xRight, yBot, yTop);
}
```

Las matrices de OpenGL

- OpenGL trabaja internamente con **3 matrices**:
  - La **matriz de proyección** almacena la forma en la que debe proyectarse el gráfico en el plano de visión.
  - La de **modelado y vista** almacena las transformaciones de *rotación, traslación*, etc. (y en 3D, la información de la vista)
  - La de **puerto de vista** almacena cómo presentar la proyección realizada en el puerto de vista.

Las matrices de OpenGL

- Antes de definir el área visible de la escena, es necesario cargar la matriz de proyección, y hacerla la identidad.

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(xLeft, xRight, yBot, yTop);
```

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

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

- 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);
```