#include <windows.h>

#include <gl/gl.h>

#include <gl/glut.h>

#pragma comment(lib, "opengl32.lib")

#pragma comment(lib, "glu32.lib")

#pragma comment(lib, "glut32.lib")

/\*\*

Creates the main window, registers event handlers, and

initializes OpenGL stuff.

\*/

void InitGraphics(int argc, char \*argv[]);

/\*\*

Sets the logical coordinate system we will use to specify

our drawings.

\*/

void SetTransformations();

/\*\*

Handles the paint event. This event is triggered whenever

our displayed graphics are lost or out-of-date.

ALL rendering code should be written here.

\*/

void OnDisplay();

/\*\*

Handles the key press. This event is whenever

a normal ASCII character is being pressed.

\*/

void OnKeyPress(unsigned char key, int x, int y);

/\*\*

Handles the special key press. This event is whenever

a special key is being pressed.

\*/

void OnSpecialKeyPress(int key, int x, int y);

// you can find more details about key handling on this link

// http://www.lighthouse3d.com/opengl/glut/index.php?5

float fXPos, fYPos, fRot, fScale;

int main(int argc, char\* argv[])

{

fXPos = fYPos = fRot = 0;

fScale = 1;

InitGraphics(argc, argv);

return 0;

}

/\*\*

Creates the main window, registers event handlers, and

initializes OpenGL stuff.

\*/

void InitGraphics(int argc, char \*argv[]) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA);

//Create an 800x600 window with its top-left corner at pixel (100, 100)

glutInitWindowPosition(100, 100); //pass (-1, -1) for Window-Manager defaults

glutInitWindowSize(800, 600);

glutCreateWindow("OpenGL Lab");

//OnDisplay will handle the paint event

glutDisplayFunc(OnDisplay);

// here is the setting of the idle function

glutIdleFunc(OnDisplay);

// here is the setting of the key function

glutKeyboardFunc(OnKeyPress);

glutSpecialFunc(OnSpecialKeyPress);

SetTransformations();

glutMainLoop();

}

/\*\*

Sets the logical coordinate system we will use to specify

our drawings.

\*/

void SetTransformations() {

//set up the logical coordinate system of the window: [-100, 100] x [-100, 100]

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-100, 100, -100, 100);

glMatrixMode(GL\_MODELVIEW);

}

/\*\*

Handles the paint event. This event is triggered whenever

our displayed graphics are lost or out-of-date.

ALL rendering code should be written here.

\*/

void OnDisplay()

{

// pushes the current matrix stack down by one,

// duplicating the current matrix.

// glPushMatrix and glPopMatrix are used here instead of glLoadIdentity.

glPushMatrix();

// clear the transformation matrix

//glLoadIdentity();

//set the background color to white

glClearColor(1, 1, 1, 1);

//fill the whole color buffer with the clear color

glClear(GL\_COLOR\_BUFFER\_BIT);

glTranslatef(fXPos,fYPos,0);//0

glRotatef(fRot, 0, 0, 1);// 0

glScalef(fScale, fScale, 1);//1

//drawing code goes here

glBegin(GL\_QUADS);

glColor3f(1, 0, 0);

glVertex3f(-50, -50, 0);

glVertex3f(50, -50, 0);

glVertex3f(50, 50, 0);

glVertex3f(-50, 50, 0);

glEnd();

// pops the current matrix stack, replacing the

// current matrix with the one below it on the stack.

glPopMatrix();

// swapping the buffers causes the rendering above to be

// shown

glFlush();

glutSwapBuffers();

glutPostRedisplay();

}

/\*\*

Handles the key press. This event is whenever

a normal ASCII character is being pressed.

\*/

void OnKeyPress(unsigned char key, int x, int y)

{

if (key == 27)

exit(0);

switch(key)

{

case 'a':// a key

case 'A':

fXPos -= 2;

break;

case 'd':// d key

case 'D':

fXPos += 2;

break;

case 'w':// w key

case 'W':

fYPos += 2;

break;

case 's':// s key

case 'S':

fYPos -= 2;

break;

case 'e':

case 'E':

fRot += 0.9;

break;

case 'q':

case 'Q':

fRot -= 0.9;

break;

case 'M':

fScale += 0.1;

break;

case 'm':

fScale -= 0.1;

break;

};

}

/\*\*

Handles the special key press. This event is whenever

a special key is being pressed.

\*/

void OnSpecialKeyPress(int key, int x, int y)

{

switch(key)

{

case GLUT\_KEY\_LEFT:// Left function key

fXPos -= 2;

break;

case GLUT\_KEY\_RIGHT:// Right function key

fXPos += 2;

break;

case GLUT\_KEY\_UP:// Up function key

fYPos += 2;

break;

case GLUT\_KEY\_DOWN:// Down function key

fYPos -= 2;

break;

};

}