/\*

\* OGL02Animation.cpp: 3D Shapes with animation

\*/

#include <windows.h> // for MS Windows

#include <GL/glut.h> // GLUT, include glu.h and gl.h

/\* Global variables \*/

char title[] = "3D Shapes with animation";

**GLfloat anglePyramid = 0.0f; // Rotational angle for pyramid [NEW]**

**GLfloat angleCube = 0.0f; // Rotational angle for cube [NEW]**

**int refreshMills = 15; // refresh interval in milliseconds [NEW]**

/\* Initialize OpenGL Graphics \*/

void initGL() {

glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque

glClearDepth(1.0f); // Set background depth to farthest

glEnable(GL\_DEPTH\_TEST); // Enable depth testing for z-culling

glDepthFunc(GL\_LEQUAL); // Set the type of depth-test

glShadeModel(GL\_SMOOTH); // Enable smooth shading

glHint(GL\_PERSPECTIVE\_CORRECTION\_HINT, GL\_NICEST); // Nice perspective corrections

}

/\* Handler for window-repaint event. Called back when the window first appears and

whenever the window needs to be re-painted. \*/

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); // Clear color and depth buffers

glMatrixMode(GL\_MODELVIEW); // To operate on model-view matrix

// Render a color-cube consisting of 6 quads with different colors

glLoadIdentity(); // Reset the model-view matrix

glTranslatef(1.5f, 0.0f, -7.0f); // Move right and into the screen

**glRotatef(angleCube, 1.0f, 1.0f, 1.0f); // Rotate about (1,1,1)-axis [NEW]**

glBegin(GL\_QUADS); // Begin drawing the color cube with 6 quads

// Top face (y = 1.0f)

// Define vertices in counter-clockwise (CCW) order with normal pointing out

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex3f( 1.0f, 1.0f, -1.0f);

glVertex3f(-1.0f, 1.0f, -1.0f);

glVertex3f(-1.0f, 1.0f, 1.0f);

glVertex3f( 1.0f, 1.0f, 1.0f);

// Bottom face (y = -1.0f)

glColor3f(1.0f, 0.5f, 0.0f); // Orange

glVertex3f( 1.0f, -1.0f, 1.0f);

glVertex3f(-1.0f, -1.0f, 1.0f);

glVertex3f(-1.0f, -1.0f, -1.0f);

glVertex3f( 1.0f, -1.0f, -1.0f);

// Front face (z = 1.0f)

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex3f( 1.0f, 1.0f, 1.0f);

glVertex3f(-1.0f, 1.0f, 1.0f);

glVertex3f(-1.0f, -1.0f, 1.0f);

glVertex3f( 1.0f, -1.0f, 1.0f);

// Back face (z = -1.0f)

glColor3f(1.0f, 1.0f, 0.0f); // Yellow

glVertex3f( 1.0f, -1.0f, -1.0f);

glVertex3f(-1.0f, -1.0f, -1.0f);

glVertex3f(-1.0f, 1.0f, -1.0f);

glVertex3f( 1.0f, 1.0f, -1.0f);

// Left face (x = -1.0f)

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex3f(-1.0f, 1.0f, 1.0f);

glVertex3f(-1.0f, 1.0f, -1.0f);

glVertex3f(-1.0f, -1.0f, -1.0f);

glVertex3f(-1.0f, -1.0f, 1.0f);

// Right face (x = 1.0f)

glColor3f(1.0f, 0.0f, 1.0f); // Magenta

glVertex3f(1.0f, 1.0f, -1.0f);

glVertex3f(1.0f, 1.0f, 1.0f);

glVertex3f(1.0f, -1.0f, 1.0f);

glVertex3f(1.0f, -1.0f, -1.0f);

glEnd(); // End of drawing color-cube

// Render a pyramid consists of 4 triangles

glLoadIdentity(); // Reset the model-view matrix

glTranslatef(-1.5f, 0.0f, -6.0f); // Move left and into the screen

**glRotatef(anglePyramid, 1.0f, 1.0f, 0.0f); // Rotate about the (1,1,0)-axis [NEW]**

glBegin(GL\_TRIANGLES); // Begin drawing the pyramid with 4 triangles

// Front

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex3f( 0.0f, 1.0f, 0.0f);

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex3f(-1.0f, -1.0f, 1.0f);

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex3f(1.0f, -1.0f, 1.0f);

// Right

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex3f(0.0f, 1.0f, 0.0f);

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex3f(1.0f, -1.0f, 1.0f);

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex3f(1.0f, -1.0f, -1.0f);

// Back

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex3f(0.0f, 1.0f, 0.0f);

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex3f(1.0f, -1.0f, -1.0f);

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex3f(-1.0f, -1.0f, -1.0f);

// Left

glColor3f(1.0f,0.0f,0.0f); // Red

glVertex3f( 0.0f, 1.0f, 0.0f);

glColor3f(0.0f,0.0f,1.0f); // Blue

glVertex3f(-1.0f,-1.0f,-1.0f);

glColor3f(0.0f,1.0f,0.0f); // Green

glVertex3f(-1.0f,-1.0f, 1.0f);

glEnd(); // Done drawing the pyramid

glutSwapBuffers(); // Swap the front and back frame buffers (double buffering)

**// Update the rotational angle after each refresh [NEW]**

**anglePyramid += 0.2f;**

**angleCube -= 0.15f;**

}

**/\* Called back when timer expired [NEW] \*/**

**void timer(int value) {**

**glutPostRedisplay(); // Post re-paint request to activate display()**

**glutTimerFunc(refreshMills, timer, 0); // next timer call milliseconds later**

**}**

/\* Handler for window re-size event. Called back when the window first appears and

whenever the window is re-sized with its new width and height \*/

void reshape(GLsizei width, GLsizei height) { // GLsizei for non-negative integer

// Compute aspect ratio of the new window

if (height == 0) height = 1; // To prevent divide by 0

GLfloat aspect = (GLfloat)width / (GLfloat)height;

// Set the viewport to cover the new window

glViewport(0, 0, width, height);

// Set the aspect ratio of the clipping volume to match the viewport

glMatrixMode(GL\_PROJECTION); // To operate on the Projection matrix

glLoadIdentity(); // Reset

// Enable perspective projection with fovy, aspect, zNear and zFar

gluPerspective(45.0f, aspect, 0.1f, 100.0f);

}

/\* Main function: GLUT runs as a console application starting at main() \*/

int main(int argc, char\*\* argv) {

glutInit(&argc, argv); // Initialize GLUT

glutInitDisplayMode(GLUT\_DOUBLE); // Enable double buffered mode

glutInitWindowSize(640, 480); // Set the window's initial width & height

glutInitWindowPosition(50, 50); // Position the window's initial top-left corner

glutCreateWindow(title); // Create window with the given title

glutDisplayFunc(display); // Register callback handler for window re-paint event

glutReshapeFunc(reshape); // Register callback handler for window re-size event

initGL(); // Our own OpenGL initialization

**glutTimerFunc(0, timer, 0); // First timer call immediately [NEW]**

glutMainLoop(); // Enter the infinite event-processing loop

return 0;

}