**#Circle-Polygon**

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

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

#include <math.h>

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

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

void display() {

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

glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer (background)

glLineWidth(7.5);

glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin

for(int i=0;i<200;i++)

{

glColor3f(1.0,0,1.0);

float pi=3.1416;

float A=(i\*2\*pi)/200;

float r=0.85;

float x = r \* cos(A);

float y = r \* sin(A);

glVertex2f(x,y );

}

//glVertex2f(0.3f,0.4f);

//glVertex2f(0.1f,0.4f);

glEnd();

glFlush(); // Render now

}

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

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

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

glutCreateWindow("OpenGL Setup Test");

//gluOrtho2D(-0.1,0.7,-0.1,0.3); // Create a window with the given title

glutInitWindowSize(320, 320);// Set the window's initial width & height

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

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

return 0;

}

**#Line\_Strip**

#include <windows.h>

#ifdef \_\_APPLE\_\_

#include <GLUT/glut.h>

#else

#include <GL/glut.h>

#endif

#include <stdlib.h>

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_LINE\_STRIP);

glVertex3f (10 , 10, 0.0);

glVertex3f (-10, 10, 0.0);

glVertex3f (-10,-10, 0.0);

glVertex3f (10 ,-10, 0.0);

glVertex3f (10 , 10, 0.0);

glEnd();

glFlush();

}

void init(void)

{

glClearColor (0.0, 0.0, 0.0, 0.0);

glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);

}

int main()

{

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize (250, 250);

glutInitWindowPosition (100, 100);

glutCreateWindow ("drawing");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**#Translation**

#include <iostream>

#include<GL/gl.h>

#include <GL/glut.h>

using namespace std;

float \_move = 0.0f;

void drawScene() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3d(1,0,0);

glLoadIdentity(); //Reset the drawing perspective

glMatrixMode(GL\_MODELVIEW);

glPushMatrix();

glTranslatef(\_move, 0.0f, 0.0f);

glBegin(GL\_QUADS);

glVertex2f(0.1f, 0.0f);

glVertex2f(0.5f, 0.0f);

glVertex2f(0.5f, 0.2f);

glVertex2f(0.1f, 0.2);

glEnd();

glPopMatrix();

glutSwapBuffers();

}

void update(int value) {

\_move += .02;

if(\_move > 1.3)

{

\_move = -1.0;

}

glutPostRedisplay();

glutTimerFunc(20, update, 0);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(800, 800);

glutCreateWindow("Transformation");

glutDisplayFunc(drawScene);

gluOrtho2D(-2,2,-2,2);

glutTimerFunc(20, update, 0); //Add a timer

glutMainLoop();

return 0;

}

**#Rotation**

#include <iostream>

#include<GL/gl.h>

#include <GL/glut.h>

#include <windows.h>

using namespace std;

float \_angle1 = 0.0f;

void drawScene() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3d(1,0,0);

glLoadIdentity(); //Reset the drawing perspective

glMatrixMode(GL\_MODELVIEW);

glPushMatrix();

glRotatef(\_angle1, 0.0f, 0.0f,1.0f);

glBegin(GL\_QUADS);

glVertex2f(0.1f, 0.0f);

glVertex2f(0.5f, 0.0f);

glVertex2f(0.5f, 0.2f);

glVertex2f(0.1f, 0.2);

glEnd();

glPopMatrix();

glutSwapBuffers();

}

void update(int value) {

\_angle1+=2.0f;

if(\_angle1 > 360.0)

{

\_angle1-=360;

}

glutPostRedisplay(); //Notify GLUT that the display has changed

glutTimerFunc(20, update, 0); //Notify GLUT to call update again in 25 milliseconds

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(800, 800);

glutCreateWindow("Transformation");

glutDisplayFunc(drawScene);

glutTimerFunc(20, update, 0); //Add a timer

glutMainLoop();

return 0;

}