**Practical 6**

Implement following 2D transformations on the object with respect to axis : –

i) Translation ii) Rotation iii) Reflection

**Program Code:-**

#include <stdio.h>

#include <math.h>

#include <iostream>

#include <vector>

#include <GL/glut.h>

using namespace std;

int pntX1, pntY1, choice = 0, edges;

vector<int> pntX;

vector<int> pntY;

int transX, transY;

double angle, angleRad;

char reflectionAxis;

double round(double d)

{

return floor(d + 0.5);

}

void drawPolygon()

{

glBegin(GL\_POLYGON);

glColor3f(1.0, 0.0, 0.0);

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

{

glVertex2i(pntX[i], pntY[i]);

}

glEnd();

}

void drawPolygonTrans(int x, int y)

{

glBegin(GL\_POLYGON);

glColor3f(0.0, 1.0, 0.0);

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

{

glVertex2i(pntX[i] + x, pntY[i] + y);

}

glEnd();

}

void drawPolygonRotation(double angleRad)

{

glBegin(GL\_POLYGON);

glColor3f(0.0, 0.0, 1.0);

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

{

glVertex2i(round((pntX[i] \* cos(angleRad)) - (pntY[i] \* sin(angleRad))), round((pntX[i] \* sin(angleRad)) + (pntY[i] \* cos(angleRad))));

}

glEnd();

}

void drawPolygonMirrorReflection(char reflectionAxis)

{

glBegin(GL\_POLYGON);

glColor3f(0.0, 0.0, 1.0);

if (reflectionAxis == 'x' || reflectionAxis == 'X')

{

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

{

glVertex2i(round(pntX[i]), round(pntY[i] \* -1));

}

}

else if (reflectionAxis == 'y' || reflectionAxis == 'Y')

{

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

{

glVertex2i(round(pntX[i] \* -1), round(pntY[i]));

}

}

glEnd();

}

void myInit(void)

{

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0f, 0.0f, 0.0f);

glPointSize(4.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-640.0, 640.0, -480.0, 480.0);

}

void myDisplay(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0, 0.0, 0.0);

if (choice == 1)

{

drawPolygon();

drawPolygonTrans(transX, transY);

}

else if (choice == 2)

{

drawPolygon();

drawPolygonRotation(angleRad);

}

else if (choice == 3)

{

drawPolygon();

drawPolygonMirrorReflection(reflectionAxis);

}

glFlush();

}

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

{

cout << "Enter your choice:\n\n" << endl;

cout << "1. Translation" << endl;

cout << "2. Rotation" << endl;

cout << "3. Mirror Reflection" << endl;

cout << "4. Exit" << endl;

cin >> choice;

if (choice == 4) {

return choice;

}

cout << "\n\nFor Polygon:\n" << endl;

cout << "Enter no of edges: "; cin >> edges;

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

{

cout << "Enter co-ordinates for vertex " << i + 1 << " : "; cin >> pntX1 >> pntY1;

pntX.push\_back(pntX1);

pntY.push\_back(pntY1);

}

if (choice == 1)

{

cout << "Enter the translation factor for X and Y: ";

cin >> transX >> transY;

}

else if (choice == 2)

{

cout << "Enter the angle for rotation: "; cin >> angle;

angleRad = angle \* 3.1416 / 180;

}

else if (choice == 3)

{

cout << "Enter reflection axis ( x or y ): "; cin >> reflectionAxis;

}

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutInitWindowPosition(100, 150);

glutCreateWindow("Extended Basic Transformations");

glutDisplayFunc(myDisplay);

myInit();

glutMainLoop();

}