Name:Harshada Gopal Rayate
Roll No:19 SEDA
Subject :CGAVR

2D Transformation

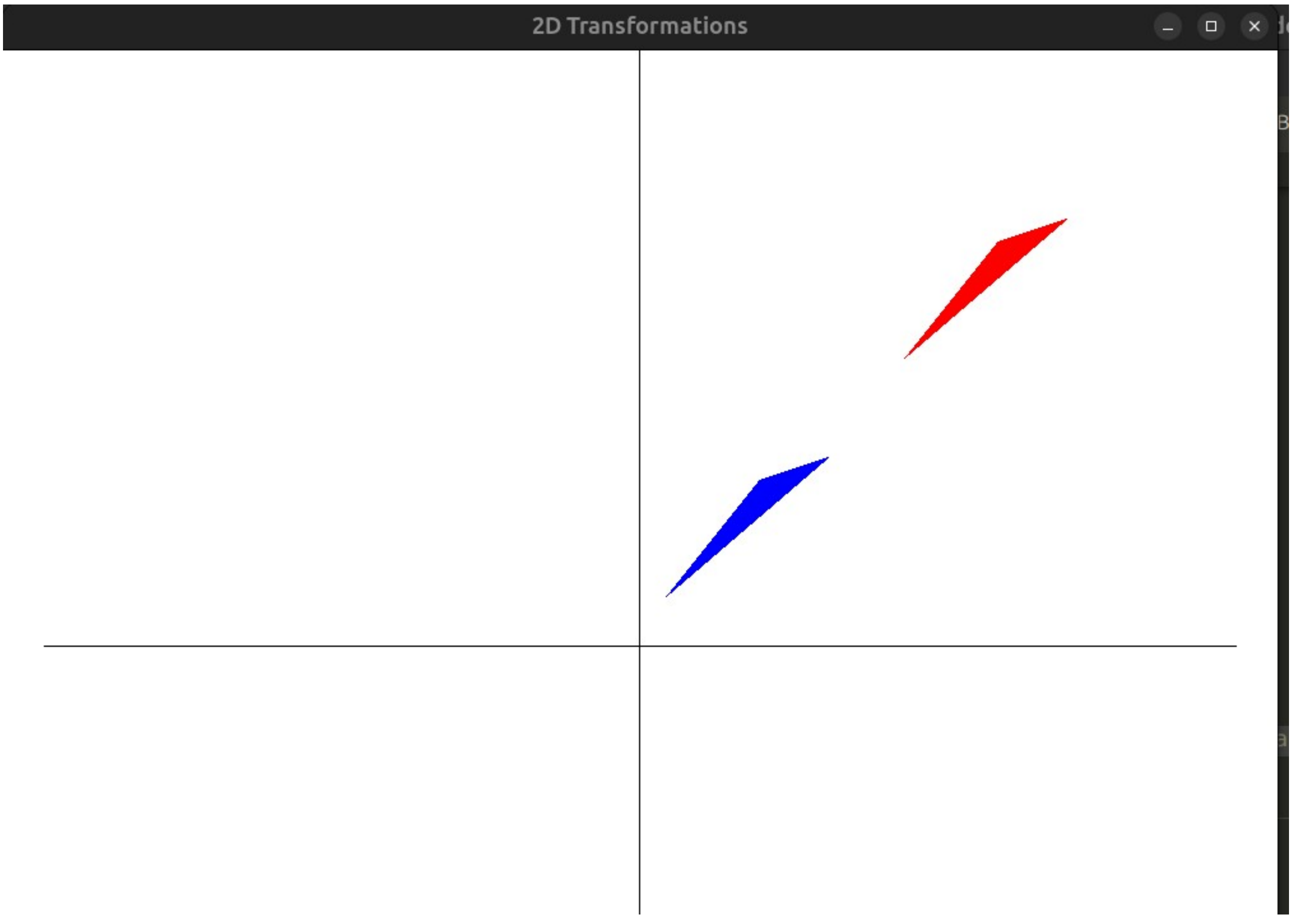
```
#include <GL/glut.h>
#include <stdio.h>
#include <math.h>

#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif

#define MAX_VERTICES 10

GLfloat vertices[MAX_VERTICES][2];
GLfloat transformedVertices[MAX_VERTICES][2];
int numVertices;

GLfloat tx, ty, sx, sy, angle, shearX, shearY;
int choice;
int showTransformed = 0;
int coordinatesDisplayed = 0;

void drawPolygon(GLfloat v[][2], int n) {
glBegin(GL_POLYGON);
for (int i = 0; i < n; i++) {
glVertex2f(v[i][0], v[i][1]);
}
glEnd();
}

void init() {
glClearColor(1.0, 1.0, 1.0, 1.0); // White background
glColor3f(0.0, 0.0, 0.0); // Black drawing color
glMatrixMode(GL_PROJECTION);
gluOrtho2D(-250, 250, -250, 250); // Setting center of the screen as (0,0)
}

void translate() {
for (int i = 0; i < numVertices; i++) {
transformedVertices[i][0] = vertices[i][0] + tx;
transformedVertices[i][1] = vertices[i][1] + ty;
}
}

void scale() {
for (int i = 0; i < numVertices; i++) {
```

```
transformedVertices[i][0] = vertices[i][0] * sx;
transformedVertices[i][1] = vertices[i][1] * sy;
}
}

void rotate() {
GLfloat rad = angle * M_PI / 180.0;

for (int i = 0; i < numVertices; i++) {
transformedVertices[i][0] = vertices[i][0] * cos(rad) - vertices[i][1] * sin(rad);
transformedVertices[i][1] = vertices[i][0] * sin(rad) + vertices[i][1] * cos(rad);
}
}

void drawAxes() {
// Draw X axis
glColor3f(0.0, 0.0, 0.0); // Black color for axes
glBegin(GL_LINES);
glVertex2f(-250, 0);
glVertex2f(250, 0);
glEnd();

// Draw Y axis
glBegin(GL_LINES);
glVertex2f(0, -250);
glVertex2f(0, 250);
glEnd();
}

void reflect() {
for (int i = 0; i < numVertices; i++) {
transformedVertices[i][0] = vertices[i][0] * -1; // Reflect across Y-axis
transformedVertices[i][1] = vertices[i][1]; // Y remains the same
}
}


void shear() {
for (int i = 0; i < numVertices; i++) {
transformedVertices[i][0] = vertices[i][0] + shearX * vertices[i][1]; // Shearing in X direction
transformedVertices[i][1] = vertices[i][1] + shearY * vertices[i][0]; // Shearing in Y direction
}
}

void display() {
glClear(GL_COLOR_BUFFER_BIT);

drawAxes();
glColor3f(0.0, 0.0, 1.0);
drawPolygon(vertices, numVertices);
```

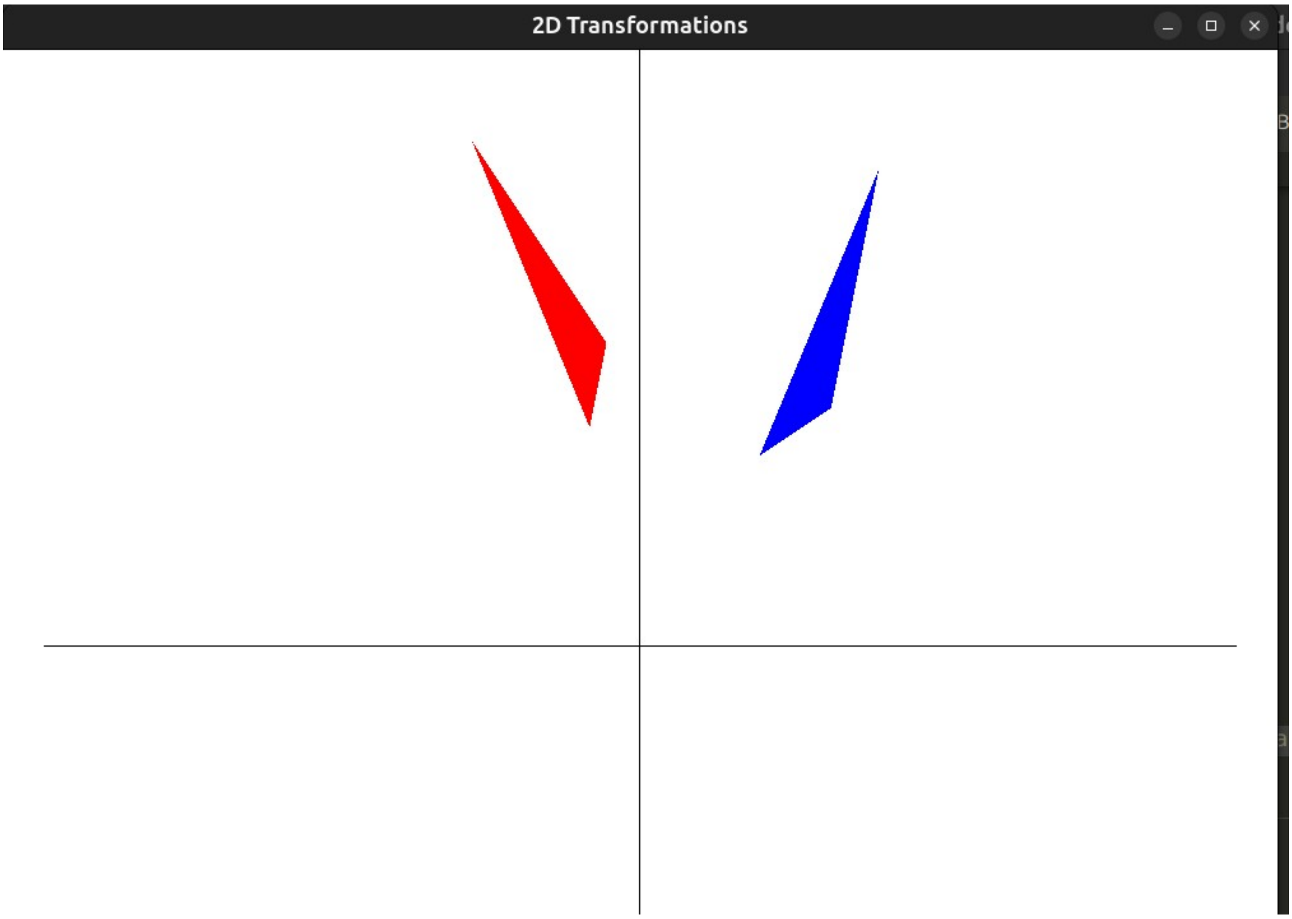
```
if (showTransformed) {
glColor3f(1.0, 0.0, 0.0);
drawPolygon(transformedVertices, numVertices);
}

glFlush();
}

void reshape(int width, int height) {
glViewport(0, 0, width, height); // Set the viewport to cover the new window size

glMatrixMode(GL_PROJECTION);
glLoadIdentity();

// Maintain the aspect ratio of the original coordinate system
if (width <= height) {
gluOrtho2D(-250, 250, -250 * (GLfloat)height / (GLfloat)width, 250 * (GLfloat)height /
(GLfloat)width);
} else {
gluOrtho2D(-250 * (GLfloat)width / (GLfloat)height, 250 * (GLfloat)width / (GLfloat)height, -250,
250);
}

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}

void printCoordinates() {
if (!coordinatesDisplayed) {
printf("\nOriginal Coordinates:\n");
for (int i = 0; i < numVertices; i++) {
printf("Vertex %d: (%.2f, %.2f)\n", i + 1, vertices[i][0], vertices[i][1]);
}

printf("\nTransformed Coordinates:\n");
for (int i = 0; i < numVertices; i++) {
printf("Vertex %d: (%.2f, %.2f)\n", i + 1, transformedVertices[i][0], transformedVertices[i][1]);
}

coordinatesDisplayed = 1;
}
}

void animate(int val) {
switch (choice) {
case 1: translate(); break;
case 2: scale(); break;
case 3: rotate(); break; // Rotate without arbitrary point
case 4: reflect(); break;
case 5: shear(); break;
default: return;
}

// Trigger display and print coordinates once the transformation is applied
showTransformed = 1;
glutPostRedisplay();
printCoordinates();
}

// Handle mouse click to switch display
void mouse(int button, int state, int x, int y) {
if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
showTransformed = 1;
glutPostRedisplay();
}
}

int main(int argc, char** argv) {
printf("Enter the number of vertices (max %d): ", MAX_VERTICES);
scanf("%d", &numVertices);

printf("Enter the coordinates of the polygon vertices (x y):\n");
for (int i = 0; i < numVertices; i++) {
printf("Vertex %d: ", i + 1);
scanf("%f %f", &vertices[i][0], &vertices[i][1]);
}

printf("Choose a transformation: \n");
printf("1. Translation\n");
printf("2. Scaling\n");
printf("3. Rotation\n");
printf("4. Reflection\n");
printf("5. Shearing\n");
scanf("%d", &choice);

switch (choice) {
case 1:
printf("Enter tx and ty: ");
scanf("%f %f", &tx, &ty);
break;
case 2:
printf("Enter sx and sy: ");
scanf("%f %f", &sx, &sy);
break;
case 3:
printf("Enter the rotation angle: ");
scanf("%f", &angle);
break;
case 4:
```

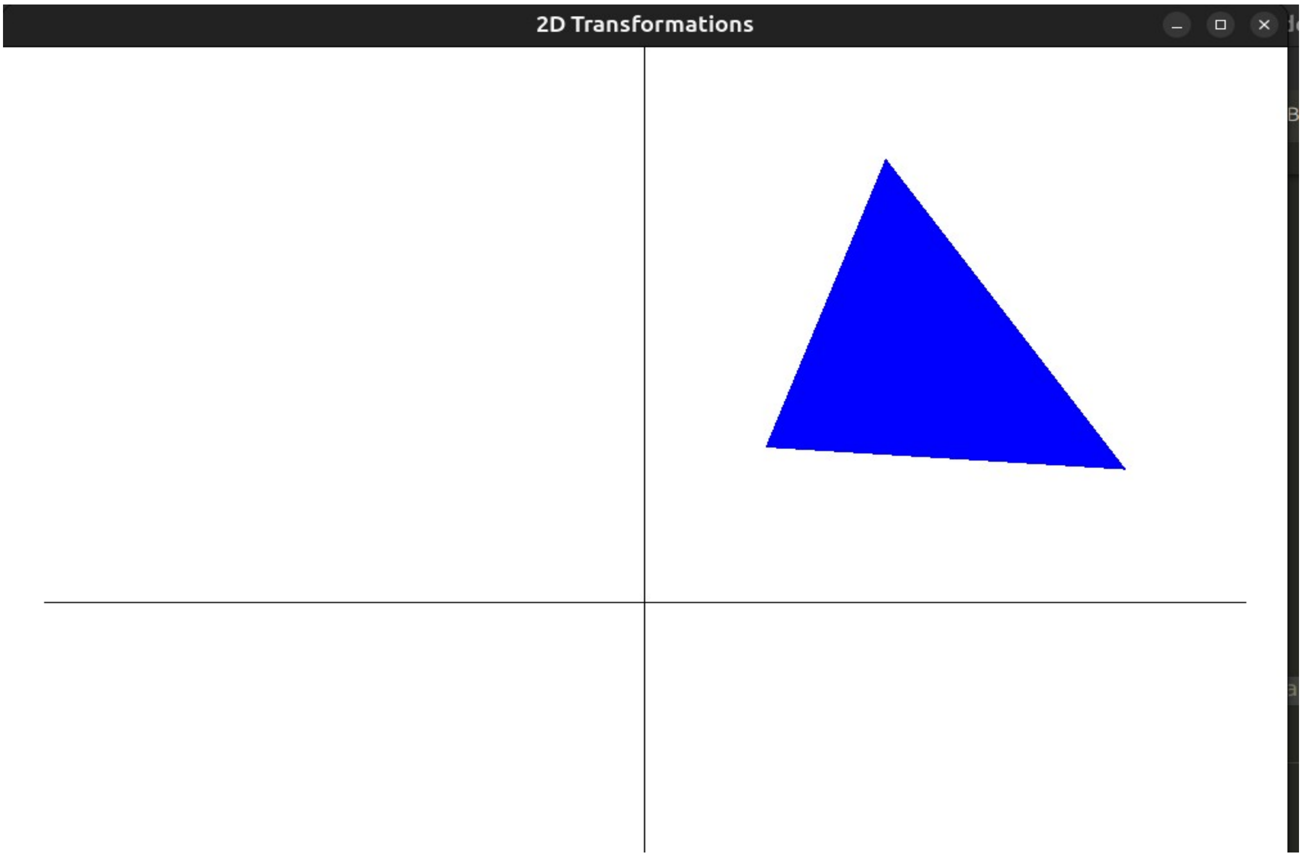
```
printf("Choose axis for reflection:\n1. X-axis\n2. Y-axis\n");
int axis;
scanf("%d", &axis);
if (axis == 1) {
sx = 1;
sy = -1;
} else if (axis == 2) {
sx = -1;
sy = 1;
} else {
printf("Invalid choice!\n");
return -1;
}
break;
case 5:
printf("Enter shearX and shearY: ");
scanf("%f %f", &shearX, &shearY);
break;
default:
printf("Invalid choice!\n");
return -1;
}

glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(1000, 1000);
glutInitWindowPosition(100, 100);
glutCreateWindow("2D Transformations");

init();
glutDisplayFunc(display);
glutReshapeFunc(reshape); // Set the reshape callback
glutMouseFunc(mouse);
glutTimerFunc(1000, animate, 0);
glutMainLoop();

return 0;
}
```

1.Translate

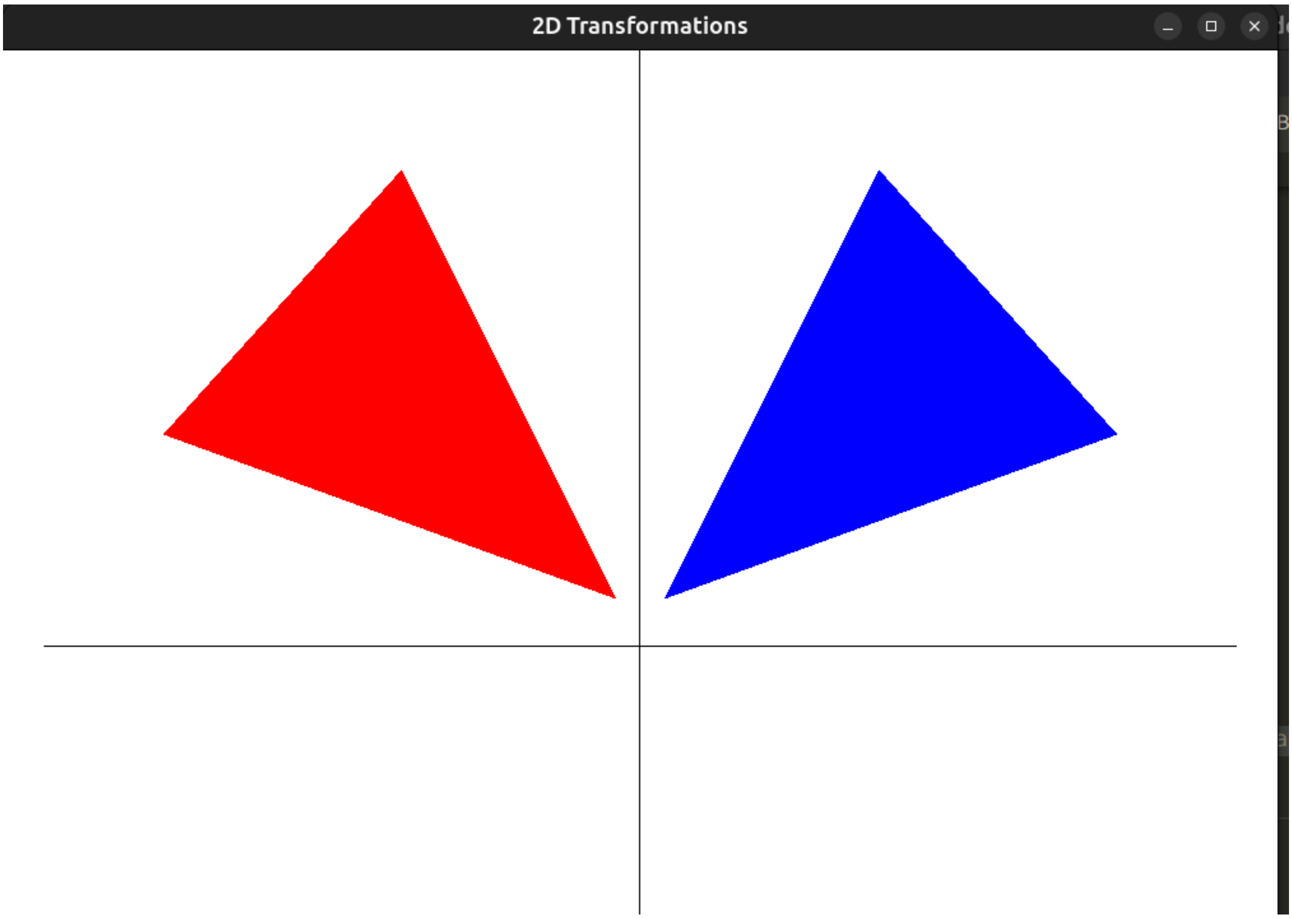


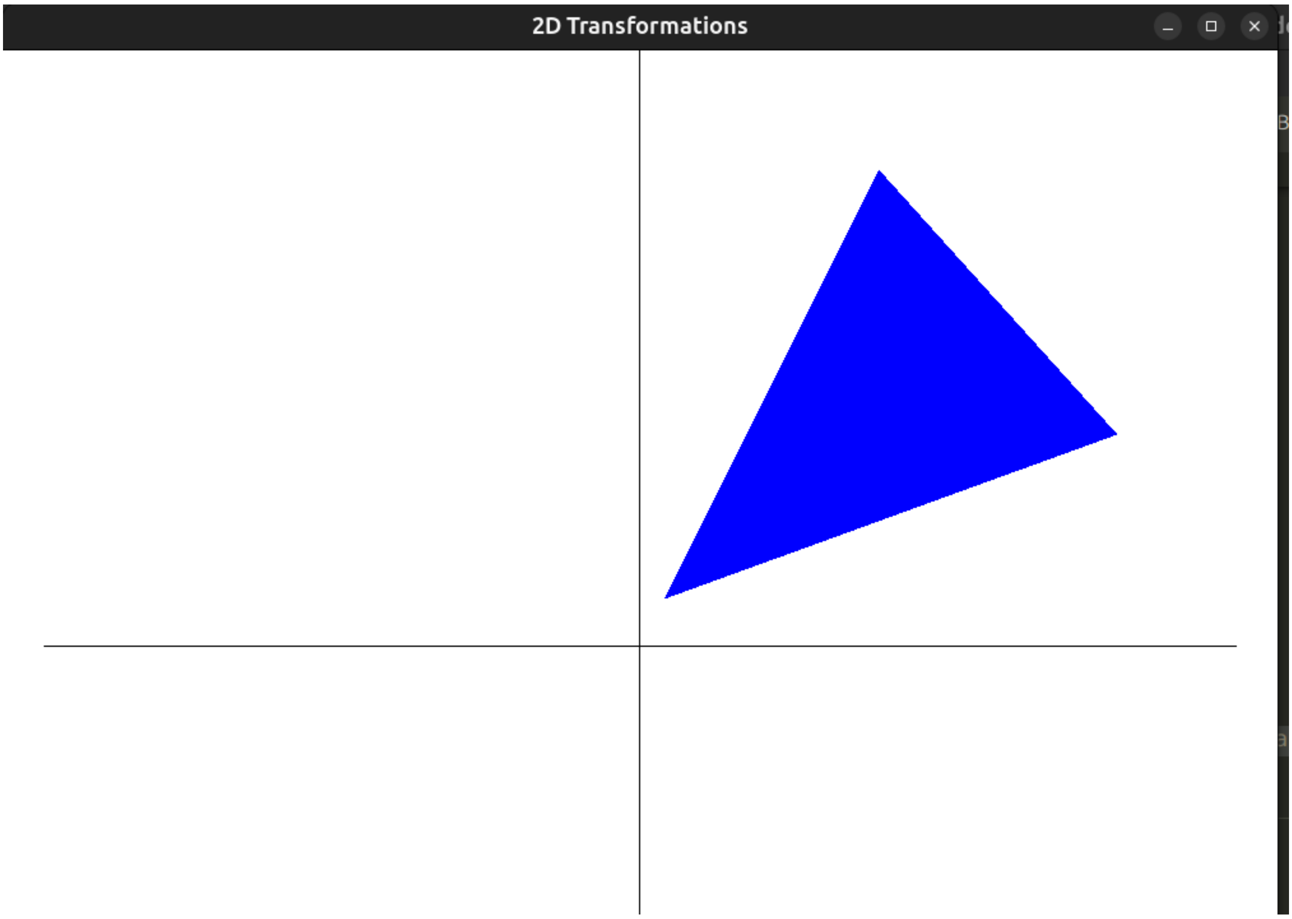
2.Rotation





4) Reflection





2D Transformations