Transformation of a given shape

1 Objective

To transform a given shape using the following transformations:

- Reflection
- Rotation
- Translation

2 Theory

2.1 Reflection

Reflection is a transformation that produces a mirror image of an object. The mirror image is generated by reflecting the object across an axis. The axis is called the line of reflection. In two dimensions, the line of reflection is a line in a plane and is perpendicular to the line containing the points and its image. Mathematically, the reflection of a point (x, y) across the x-axis is $(x, -y)$ and the reflection across the y-axis is $(-x, y)$.

2.2 Rotation

Rotation is a transformation that turns a figure about a fixed point called the center of rotation. An object and its rotation are the same shape and size, but the figures may be turned in different directions. Mathematically, the rotation of a point (x, y) about the origin through an angle θ in the counter-clockwise direction is given by:

$$(x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$$

A more general form of rotation is about a point (x_r, y_r), which is given by:

$$(x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta, y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta)$$

The above formulaes support rotation in both the clockwise and counter-clockwise direction. For clockwise rotation, the angle θ is negative. An even more general approach to rotate coordinates is by using the following rotation matrix. Objects can also be rotated by an angle theta about an axis (x_r, y_r) using the rotation matrix:

$$\begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta \\ y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta \\ 1 \end{bmatrix}$$

2.3 Translation

Translation is a transformation that moves a figure in a straight line without turning it or making it bigger or smaller. Mathematically, the translation of a point (x, y) by a vector (a, b) is given by $(x + a, y + b)$.

3 Algorithm

3.1 Reflection

1. Start
2. Read the coordinates of the vertices of the shape
3. Read the axis of reflection
4. If the axis of reflection is x-axis, then
   (a) For each vertex, negate the y-coordinate
5. Else if the axis of reflection is y-axis, then
   (a) For each vertex, negate the x-coordinate
6. Else
   (a) For each vertex, negate both the x and y coordinates
7. Print the coordinates of the new vertices or draw the reflected shape
8. Stop

3.2 Rotation

1. Start
2. Read the coordinates of the vertices of the shape
3. Read the angle of rotation
4. Read the center of rotation
5. For each vertex, rotate it about the center of rotation by the angle of rotation using the rotation matrix
6. Print the coordinates of the vertices of the rotated shape or display the new shape.
7. Stop

3.3 Translation

1. Start
2. Read the coordinates of the vertices of the shape
3. Read the vector of translation
4. For each vertex, do
   (a) Add the vector of translation to the coordinates of the vertex
5. Print the coordinates of the new vertices or display the translated shape
6. Stop

4 Source Code

```
#include <stdio.h>
#include <math.h>
#include <graphics.h>

#define width 640
#define height 480

#define PI 3.1415926535889

void axes()
{
    // print axes
    line(width / 2, 0, width / 2, height);
    line(0, height / 2, width, height / 2);
}

void translate(unsigned* x, unsigned* y)
{
    // translates the coordinates to the new axes
    *x = (*x + width / 2) % width;
    *y = (-*y + height / 2) % height;
}

void drawTriangle(unsigned coordinates[3][2])
{
    // draw functions guarantee that the data received is not modified
    // so work with a copy of the coordinates; it's a pointer
    unsigned copy[3][2];
    for (unsigned i = 0; i < 3; i++)
    {
        copy[i][0] = coordinates[i][0];
        copy[i][1] = coordinates[i][1];
    }
    // only needs to be done here; translate to the axes defined by the computer
    for (unsigned i = 0; i < 3; i++) translate(&copy[i][0], &copy[i][1]);
    // plot the translated points
    for (unsigned i = 0; i < 3; i++)
        line(copy[i][0], copy[i][1], copy[(i + 1) % 3][0], copy[(i + 1) % 3][1]);
}

void reflectXaxis(unsigned num_points, unsigned coordinates[][2])
{
    // make the y coordinates negative
    for (unsigned i = 0; i < num_points; i++) coordinates[i][1] = -coordinates[i][1];
}

void reflectYaxis(unsigned num_points, unsigned coordinates[][2])
{
    // make the x coordinates negative
    for (unsigned i = 0; i < num_points; i++) coordinates[i][0] = -coordinates[i][0];
```

```
}

void rotate(unsigned num_points, unsigned coordinates[][2], float theta, unsigned axis[2])
{
    // buffer the coordinates because the updates are simultaneous
    unsigned copy[2];

    // rotate the coordinates by theta degrees
    for (unsigned i = 0; i < num_points; i++)
    {
        copy[0] = axis[0] + (coordinates[i][0] - axis[0]) * cos(theta * PI / 180)
                - (coordinates[i][1] - axis[1]) * sin(theta * PI / 180);
        copy[1] = axis[1] + (coordinates[i][0] - axis[0]) * sin(theta * PI / 180)
                + (coordinates[i][1] - axis[1]) * cos(theta * PI / 180);

        // copy the rotated coordinates back to the original array
        coordinates[i][0] = copy[0];
        coordinates[i][1] = copy[1];
    }
}

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    setcolor(GREEN);

    axes();
    // coordinates of a triangle in the first quadrant
    unsigned triangle[3][2] =
    {
        {50, 50},
        {50, 200},
        {200, 50}
    };
    // draw the triangle in the first quadrant
    drawTriangle(triangle);

    // REFLECTION stuff reflect the triangle with respect to the x-axis
    reflectXaxis(3, triangle);
    drawTriangle(triangle);
    // reflect the above triangle with respect to the y-axis
    reflectYaxis(3, triangle);
    drawTriangle(triangle);

    // reflect the above triangle with respect to the x-axis
    reflectXaxis(3, triangle);
    drawTriangle(triangle);

    // reflect once again to get the original triangle
    reflectYaxis(3, triangle);
    setcolor(RED);
```

```
    drawTriangle(triangle);

    // ROTATION stuff
    setcolor(BLUE);
    unsigned axis[2] = { 50, 50 };
    printf("%u %u\n", axis[0], axis[1]);
    float theta = -60.f;
    rotate(3, triangle, theta, axis);
    drawTriangle(triangle);

    getch();
    closegraph();
    return 0;
}
```

5 Output

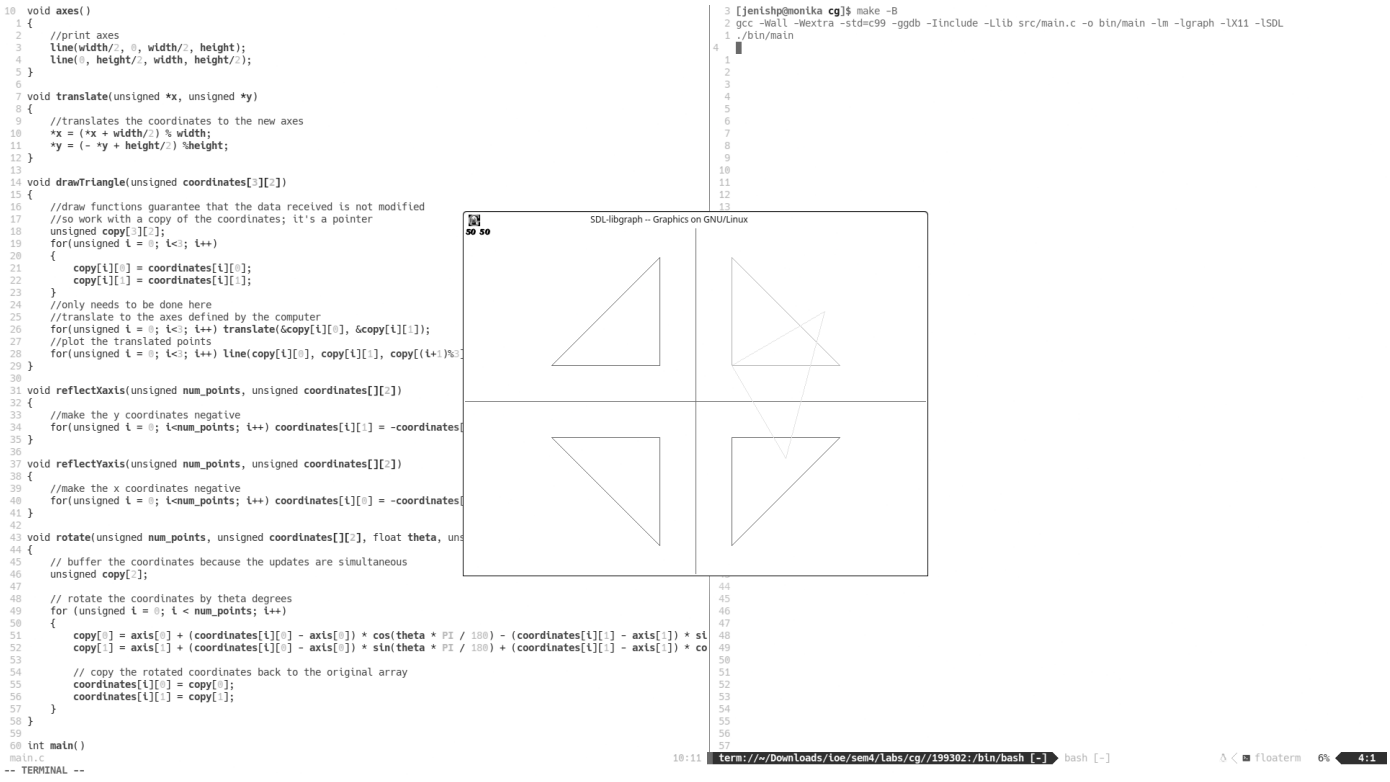

Figure 1:

6 Conclusion

Here, we have seen how to transform a given shape using the Reflection, Rotation and Translation. On transforming the shape, we can see that the shape is reflected about the x-axis, y-axis and to the 3rd quadrant. The shape is also rotated about the origin by an angle of 60 degrees in the clockwise direction about its vertex. To make the transformations easier, we have used a translation function to translate the coordinates to the axes defined by the computer just before plotting the points. This virtually shifts the origin to the center of the screen and makes the transformations easier to understand.