WORKSHEET 5

Student Name: Sumit Kumar **UID: 22BCS10048**

Section/Group: KRG_IOT-1-A **Semester: 6th Semester**

Branch: BE-CSE **Date of Performance: 13/02/2025**

Subject Name: Computer Graphics with Lab **Subject Code: 22CSH-352**

1. **Aim:** Implement clockwise and anticlockwise rotation of a triangle about a specified point and evaluate the results.

2. **Objective:** To perform and visualize clockwise and anticlockwise rotations of a triangle about a specified point.

3. **Implementation Code:**

```
#include <iostream>
#include <graphics.h>
#include <cmath>

using namespace std;

void rotateTriangle(float x[], float y[], int pivotX, int pivotY, float angle, bool clockwise) {
   float rad = (3.14159 / 180) * angle;
   if (clockwise) rad = -rad;

   float newX[3], newY[3];

   for (int i = 0; i < 3; i++) {
      newX[i] = pivotX + (x[i] - pivotX) * cos(rad) - (y[i] - pivotY) * sin(rad);
      newY[i] = pivotY + (x[i] - pivotX) * sin(rad) + (y[i] - pivotY) * cos(rad);
   }

   setcolor(RED);
   line(newX[0], newY[0], newX[1], newY[1]);
   line(newX[1], newY[1], newX[2], newY[2]);
   line(newX[2], newY[2], newX[0], newY[0]);
}
```

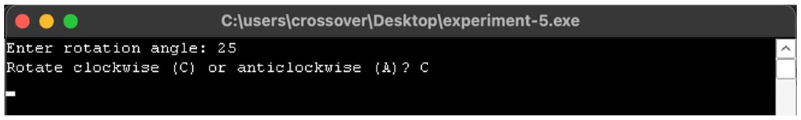
```
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    float x[] = {200, 150, 250}, y[] = {100, 200, 200};
    int pivotX = 200, pivotY = 150, angle;
    char choice;

    setcolor(WHITE);
    line(x[0], y[0], x[1], y[1]);
    line(x[1], y[1], x[2], y[2]);
    line(x[2], y[2], x[0], y[0]);

    cout << "Enter rotation angle: ";
    cin >> angle;
    cout << "Rotate clockwise (C) or anticlockwise (A)? ";
    cin >> choice;

    rotateTriangle(x, y, pivotX, pivotY, angle, (choice == 'C' || choice == 'c'));

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

4. Explanation:

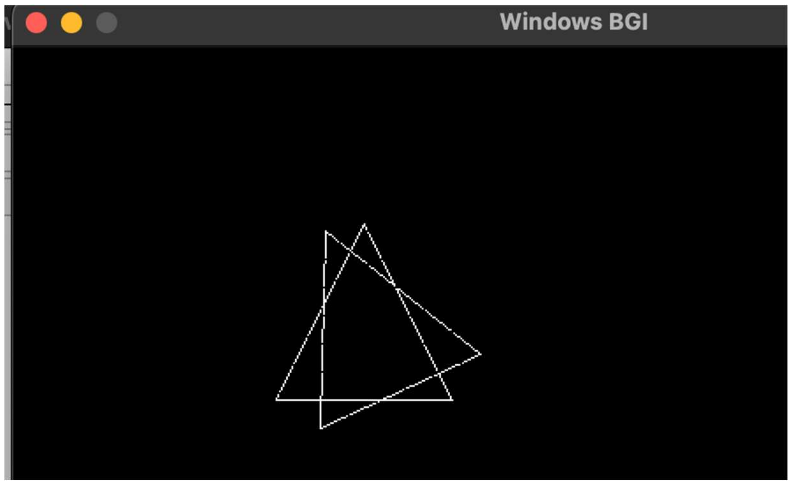
This program demonstrates the clockwise and anticlockwise rotation of a triangle about a specified pivot point. It first initializes a triangle and plots it using the graphics.h library. The user inputs a rotation angle and chooses the direction of rotation. Using transformation equations, the program recalculates and plots the new triangle after applying the rotation matrix. The pivot point remains fixed, ensuring accurate rotation. Clockwise and anticlockwise rotations are determined by adjusting the sign of the rotation angle before applying the transformation.

By visualizing the rotated triangle, the program helps evaluate how rotation affects object positioning. The graphical representation confirms the correctness of the transformation equations. The use of trigonometric functions ensures smooth and precise rotations. This implementation effectively demonstrates 2D transformations in computer graphics, reinforcing concepts such as coordinate transformation and rotational matrices.The second program uses the Midpoint Circle Algorithm, an optimized integer-based approach.

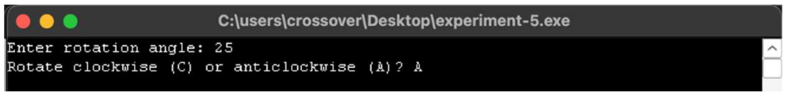
5. Output:



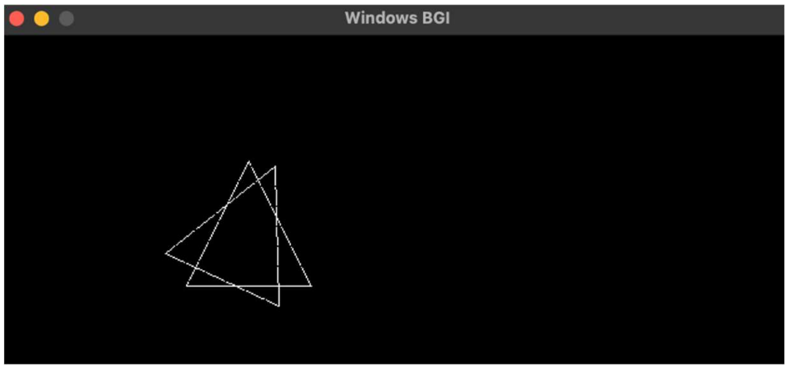
```
Enter rotation angle: 25
Rotate clockwise (C) or anticlockwise (A)? C
```



(a) Rotated Clockwise



```
Enter rotation angle: 25
Rotate clockwise (C) or anticlockwise (A)? A
```



(b) Circle Drawing Using the Midpoint Circle Algorithm

6. Learning Outcomes:

- Understanding 2D rotation transformations in computer graphics.
- Applying trigonometric functions for accurate geometric transformations.
- Implementing clockwise and anticlockwise rotations using matrices.
- Using `graphics.h` to visualize transformations dynamically.
- Evaluating the impact of pivot-based transformations on object positioning.