Experiment 5

Student Name: Rohit Kumar **UID:22BCS11093**
Branch: CSE **Section/Group:22BCS640/B**
Semester: 6 **Date of Performance:18-02-25**
Subject Name: Computer Graphics 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. Algorithm:

a) To Rotate Clockwise :

1. Initialize Graphics Mode:-
   Detect and initialize the graphics driver and mode using initgraph().

2. **Input Triangle Coordinates:-**
   Take user input for the three vertices *(x1,y1)(x1, y1)*(x1,y1), *(x2,y2)(x2, y2)*(x2,y2), *(x3,y3)(x3, y3)*(x3,y3).

3. **Draw Original Triangle:-**
   Use drawpoly() to display the initial triangle.

4. **Compute Centroid:-**
   Calculate the centroid *(xc,yc)(xc, yc)*(xc,yc) using:
   xc= x1+x2+x2 / 3, yc= y1+y2+y3 / 3

5. **Input Rotation Angle:-**
   Accept the rotation angle from the user and convert it to radians:
   rad= angle x 3.14/180

6. **Compute New Rotated Coordinates (Clockwise Rotation Formula):-**
   For each vertex *(X,Y)(X, Y)*(X,Y), calculate:
   X’= xc+ (X-xc) . cos(rad) + (Y-yc) . Sin(rad)
   Y’= yc- (X-xc) . sin(rad) + (Y-yc) . Cos(rad)

7. **Draw Rotated Traingle:-**
   Use drawpoly() to display the rotated triangle in a different color.

8. **Wait for User Input & Close Graphics Mode:-**
   Use getch() to pause, then closegraph() to exit.

b) <u>To Rotate Anti-clockwise:</u>

1. **Initialize Graphics Mode:-**
   Detect and initialize the graphics driver and mode using initgraph().

2. **Input Triangle Coordinates:-**
   Take user input for the three vertices (x1,y1), (x2,y2), (x3,y3).

3. **Draw Original Triangle:-**
   Use drawpoly() to display the initial triangle.

4. **Compute Centroid:-**
   Calculate the centroid (xc,yc) using:
   xc= x1+x2+x2 / 3, yc= y1+y2+y3 / 3

5. **Input Rotation Angle:-**
   Accept the rotation angle from the user and convert it to radians:
   rad= angle x 3.14/180

6. **Compute New Rotated Coordinates (Clockwise Rotation Formula):-**
   For each vertex (X,Y), calculate:
   X'= xc+ (X-xc) . cos(rad) - (Y-yc) . Sin(rad)
   Y'= yc- (X-xc) . sin(rad) + (Y-yc) . Cos(rad)

7. **Draw Rotated Traingle:-**
   Use drawpoly() to display the rotated triangle in a different color.

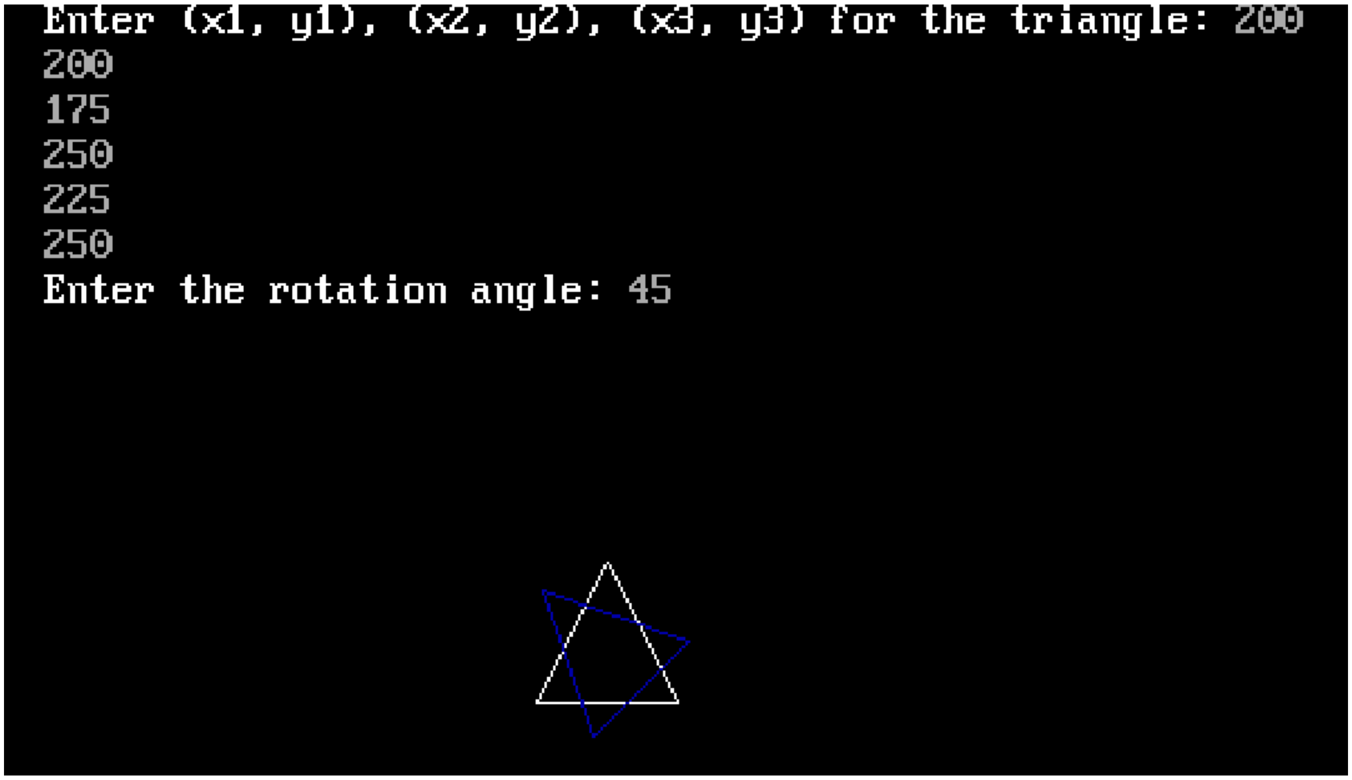
8. **Wait for User Input & Close Graphics Mode:-**
   Use getch() to pause, then closegraph() to exit.

4. Implementation/Code:

a.To Rotate Clockwise:

```
#include <iostream>
#include <conio.h>
#include <graphics.h>
#include <math.h>

using namespace std;
int main() {
clrscr();
int gd = DETECT, gm;
initgraph(&gd, &gm, "C:\\Turboc3\\BGI");  // Use the correct BGI path

int x1, y1, x2, y2, x3, y3;
cout << "Enter (x1, y1), (x2, y2), (x3, y3) for the triangle: ";
cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;

int tri[] = {x1, y1, x2, y2, x3, y3, x1, y1};
setcolor(WHITE);
drawpoly(4, tri);

int xc = (x1 + x2 + x3) / 3;
int yc = (y1 + y2 + y3) / 3;

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

float rad = angle * M_PI / 180.0;

int X1 = xc + (int)((x1 - xc) * cos(rad) - (y1 - yc) * sin(rad));
int Y1 = yc + (int)((x1 - xc) * sin(rad) + (y1 - yc) * cos(rad));

int X2 = xc + (int)((x2 - xc) * cos(rad) - (y2 - yc) * sin(rad));
int Y2 = yc + (int)((x2 - xc) * sin(rad) + (y2 - yc) * cos(rad));

int X3 = xc + (int)((x3 - xc) * cos(rad) - (y3 - yc) * sin(rad));
int Y3 = yc + (int)((x3 - xc) * sin(rad) + (y3 - yc) * cos(rad));

setcolor(RED);
int rotatedTri[] = {X1, Y1, X2, Y2, X3, Y3, X1, Y1};
drawpoly(4, rotatedTri);
getch();
closegraph();
return 0;
}
```

Output:-

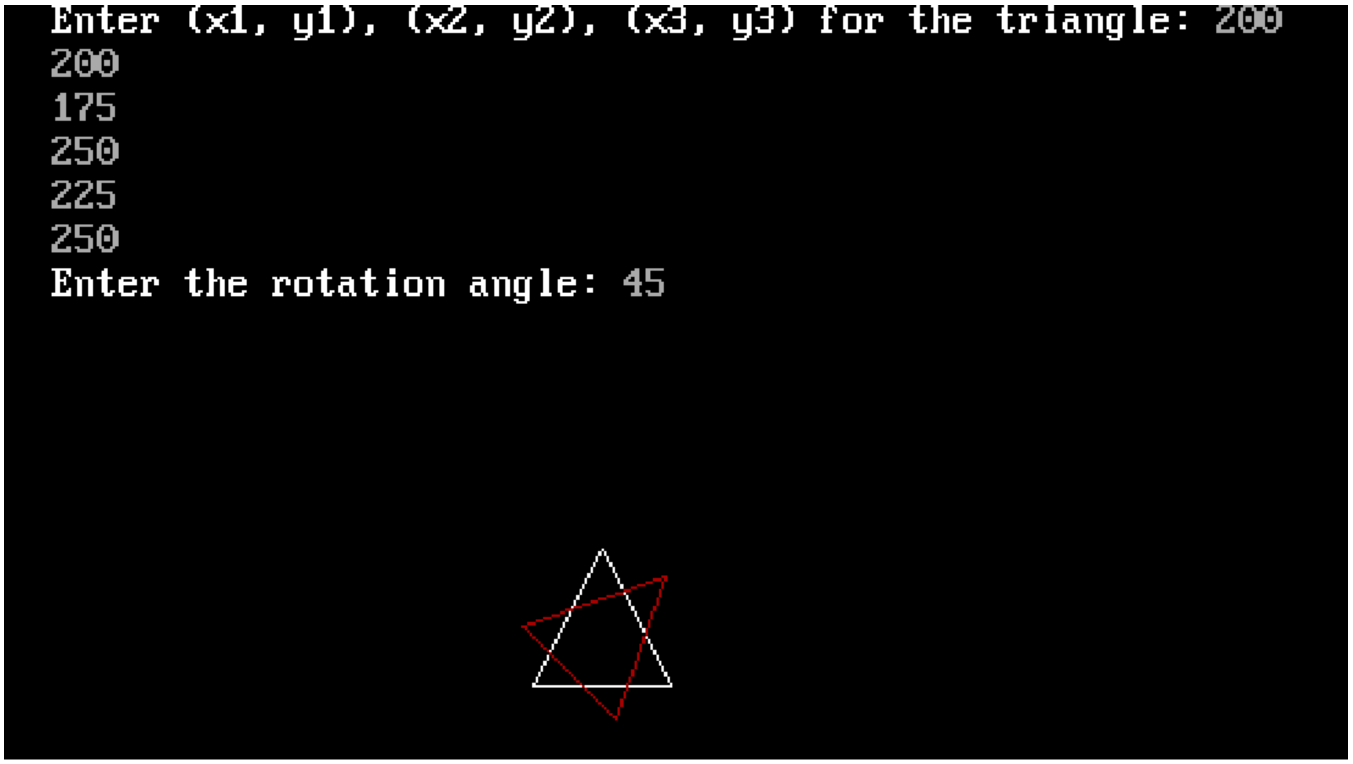


```
Enter (x1, y1), (x2, y2), (x3, y3) for the triangle: 200
200
175
250
225
250
Enter the rotation angle: 45
```

b. To Rotate Anti-Clockwise:

```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>

void main() {
clrscr();
// Initialize graphics mode
int gd = DETECT, gm;
initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

// Input triangle coordinates
int x1, y1, x2, y2, x3, y3;
cout << "Enter (x1, y1), (x2, y2), (x3, y3) for the triangle: ";
cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
// Draw original triangle
int tri[] = {x1, y1, x2, y2, x3, y3, x1, y1};
setcolor(WHITE);
drawpoly(4, tri);

// Compute centroid
int xc = (x1 + x2 + x3) / 3;
int yc = (y1 + y2 + y3) / 3;

// Input rotation angle
float angle;
cout << "Enter the rotation angle: ";
cin >> angle;

// Convert angle to radians
float rad = angle * M_PI / 180;

// Compute new rotated coordinates (Anti-clockwise rotation formula)
int X1 = xc + (int)((x1 - xc) * cos(rad) - (y1 - yc) * sin(rad));
int Y1 = yc + (int)((x1 - xc) * sin(rad) + (y1 - yc) * cos(rad));

int X2 = xc + (int)((x2 - xc) * cos(rad) - (y2 - yc) * sin(rad));
int Y2 = yc + (int)((x2 - xc) * sin(rad) + (y2 - yc) * cos(rad));

int X3 = xc + (int)((x3 - xc) * cos(rad) - (y3 - yc) * sin(rad));
int Y3 = yc + (int)((x3 - xc) * sin(rad) + (y3 - yc) * cos(rad));

// Draw rotated triangle
setcolor(BLUE);
int rotatedTri[] = {X1, Y1, X2, Y2, X3, Y3, X1, Y1};
drawpoly(4, rotatedTri);

getch();
closegraph();
}
```

Output:-



```
Enter (x1, y1), (x2, y2), (x3, y3) for the triangle: 200
200
175
250
225
250
Enter the rotation angle: 45
```

5. Learning Outcomes:-

- Learned how **rotation transformation** works in computer graphics using **trigonometric functions** (sin, cos).
- Explored the difference between **clockwise** and **anti-clockwise** rotation by modifying the **sine term signs** in the rotation formula.
- Understood the importance of **centroid (xc, yc)** in rotating a shape **around its center** rather than the origin.
- Gained hands-on experience in using **C++ graphics.h** library functions like initgraph(), drawpoly(), and setcolor() for visual representation.
- Learned the necessity of converting **degrees to radians** using the formula: rad= angle x 3.14/180