Experiment 3

Student Name: Pranjal Singh
Branch: BE-CSE
Semester: 6th
Subject Name: Computer Graphics

UID: 22BCS13041
Section/Group: 22BCS_IOT-601/A
Date of Performance: 13/02/2024
Subject Code: 22CSH-352

1. Aim: Apply translation, scaling, and rotation transformations on a given triangle and observe the changes.

2. Objective: To apply geometric transformations such as translation, scaling, and rotation on a given triangle.

3. Algorithm:

a. Translation:

i. Initialize Graphics Mode.
ii. Take Input for Triangle Coordinates
iii. Draw the Original Triangle.
iv. Use the line() function to draw three lines connecting the three given points.
v. Prompt the user to enter translation values tx and ty.
vi. Update the coordinates:
$x1' = x1 + tx$, $y1' = y1 + ty$
$x2' = x2 + tx$, $y2' = y2 + ty$
$x3' = x3 + tx$, $y3' = y3 + ty$

b. Scaling:

i. Initialize Graphics Mode.
ii. Take Input for Triangle Coordinates
iii. Draw the Original Triangle
iv. Prompt the user to enter scaling factors sx and sy
v. Update the coordinates of each vertex by multiplying them with the respective scaling factors:
$x1' = x1 \times sx$, $y1' = y1 \times sy$
$x2' = x2 \times sx$, $y2' = y2 \times sy$
$x3' = x3 \times sx$, $y3' = y3 \times sy$

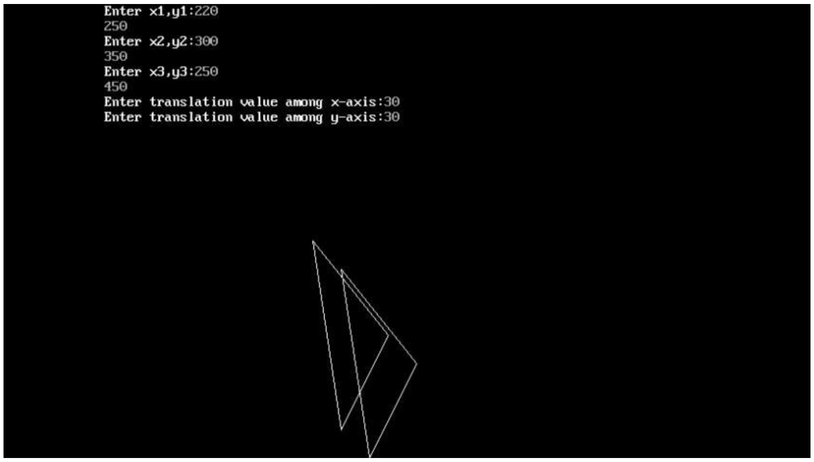
c. Roation:

i. Initialize Graphics Mode.
ii. Take Input for Triangle Coordinates
iii. Draw the Original Triangle.
iv. Take Input for Rotation Angle.
v. Use the rotation transformation formulas
$x' = x \cos(\theta) - y \sin(\theta)$
$y' = x \sin(\theta) + y \cos(\theta)$

4. Implementation/Code:

a) Translation:

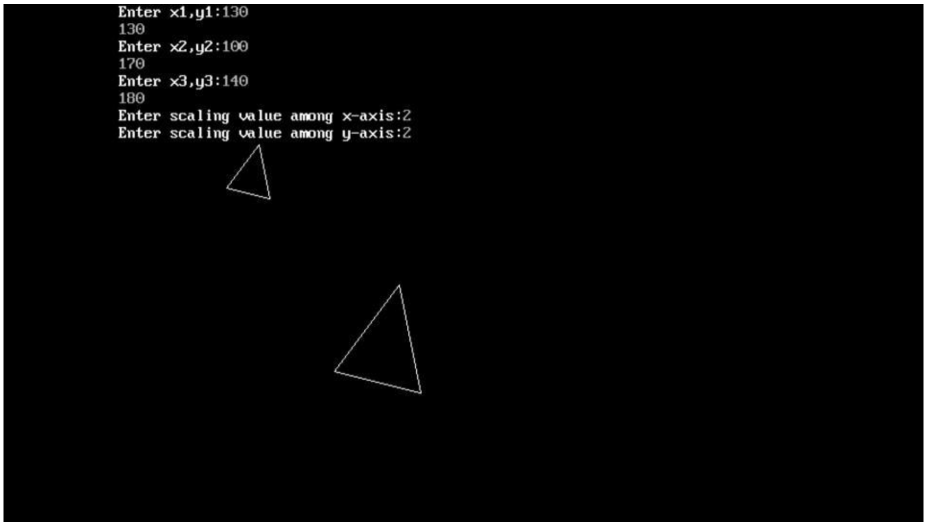
```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
void main() {
    clrscr();
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "c://turboc3//bgi");
    int x1, y1, x2, y2, x3, y3, tx, ty;
    cout << "Enter x1, y1: ";
    cin >> x1 >> y1;
    cout << "Enter x2, y2: ";
    cin >> x2 >> y2;
    cout << "Enter x3, y3: ";
    cin >> x3 >> y3;
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    cout << "Enter translation value among x-axis: ";
    cin >> tx;
    cout << "Enter translation value among y-axis: ";
    cin >> ty;
    x1  +=  tx;
    x2 += tx;
    x3 += tx;
    y1 += ty;
    y2 += ty;
    y3 += ty;
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    getch();
    closegraph();
}
```



```
Enter x1,y1:220
250
Enter x2,y2:300
350
Enter x3,y3:250
450
Enter translation value among x-axis:30
Enter translation value among y-axis:30
```

b) Scaling:

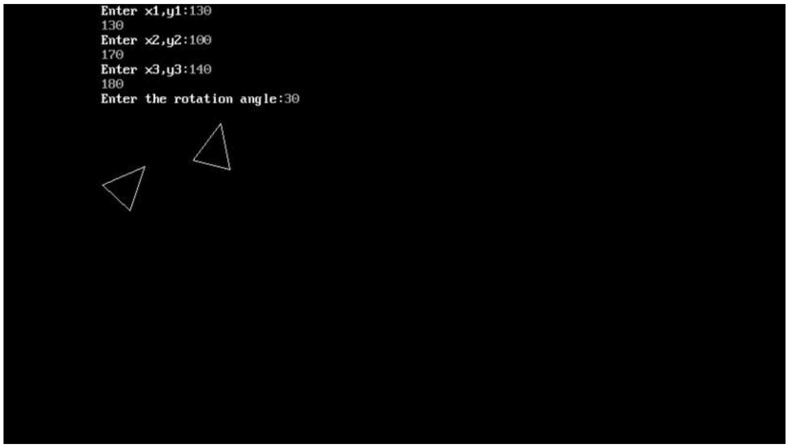
```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
void main() {
    clrscr();
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "c://turboc3//bgi");
    int x1, y1, x2, y2, x3, y3, sx, sy;
    cout << "Enter x1, y1: ";
    cin >> x1 >> y1;
    cout << "Enter x2, y2: ";
    cin >> x2 >> y2;
    cout << "Enter x3, y3: ";
    cin >> x3 >> y3;
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    cout << "Enter scaling value among x-axis: ";
    cin >> sx;
    cout << "Enter scaling value among y-axis: ";
    cin >> sy;
    x1  *=  sx;
    x2  *=  sx;
    x3  *=  sx;
    y1  *=  sy;
    y2  *=  sy;
    y3 *= sy;
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    getch();
    closegraph();
}
```



```
Enter x1,y1:130
130
Enter x2,y2:100
170
Enter x3,y3:140
180
Enter scaling value among x-axis:2
Enter scaling value among y-axis:2
```

c) Rotation:

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
void main() {
    clrscr();
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "c://turboc3//bgi");
    int x1, y1, x2, y2, x3, y3;
    float angle;
    cout << "Enter x1, y1: ";
    cin >> x1 >> y1;
    cout << "Enter x2, y2: ";
    cin >> x2 >> y2;
    cout << "Enter x3, y3: ";
    cin >> x3 >> y3;
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    cout << "Enter the rotation angle: ";
    cin >> angle;
    angle = angle * 3.1428 / 180;
    int tempX, tempY;
    tempX = x1; tempY = y1;
    x1 = tempX * cos(angle) - tempY * sin(angle);
    y1 = tempX * sin(angle) + tempY * cos(angle);
    tempX = x2; tempY = y2;
    x2 = tempX * cos(angle) - tempY * sin(angle);
    y2 = tempX * sin(angle) + tempY * cos(angle);
    tempX = x3; tempY = y3;
    x3 = tempX * cos(angle) - tempY * sin(angle);
    y3 = tempX * sin(angle) + tempY * cos(angle);
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    getch();
    closegraph();
}
```



```
Enter x1,y1:130
130
Enter x2,y2:100
170
Enter x3,y3:140
180
Enter the rotation angle:30
```

5. **Learning Outcome:**

- Understanding Basic Graphics Programming.
- Understanding 2D Transformations.
- Understood the concept of coordinate transformation using trigonometric functions

Experiment 4

Student Name: Pranjal Singh
UID: 22BCS13041
Branch: BE-CSE
Section/Group: 22BCS_IOT-601/A
Semester: 6th
Date of Performance: 20/02/2024
Subject Name: Computer Graphics
Subject Code: 22CSH-352

1. **Aim:** Develop a program to draw a circle using the circle generator algorithm and mid-point circle algorithm for a given center and radius.

2. **Objective:** To develop and implement the circle generator and midpoint circle generator algorithm to draw a circle with a given center and radius.

3. **Algorithm:**

a) Circle Generator Algorithm

```
i.   Start
ii.  Input the center (xc, yc) and radius r.
iii. Loop x from -r to r and compute y
     as: y=yc±r2-(x-xc)2y = yc \pm
     \sqrt{r^2- (x - xc)^2}y=yc±r2-(x-
     xc)2
iv.  Plot the points (x, y).
v.   End
```

b) Mid-point Circle Drawing Algorithm

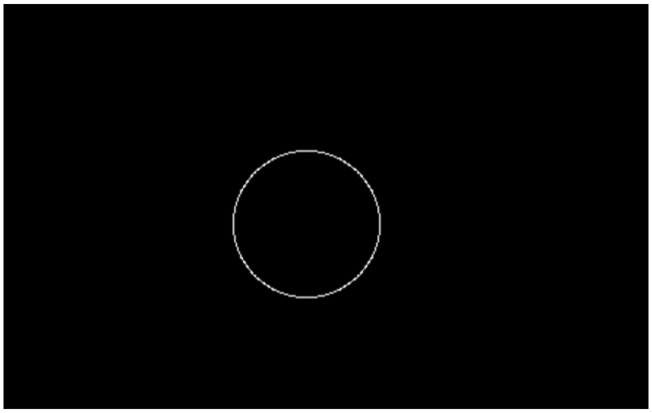
```
i.   Start
ii.  Input center (xc, yc) and radius r.
iii. Set x = 0, y = r, and decision
     parameter p = 1 - r.
iv.  Repeat while x ≤ y:
     Plot the points using symmetry.
     If p < 0, update p = p + 2x + 1.
     Else, update p = p + 2x - 2y + 1 and
     decrement y.
     Increment x.
v.   End
```

4. **Implementation/Code:**

a) Circle Generator Algorithm:

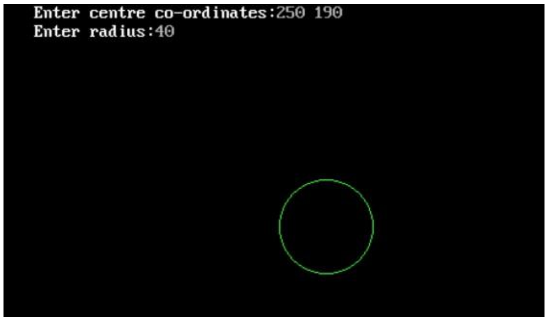
```
#include<iostream.h>
#include<conio.h>
#include<dos.h>
#include<graphics.h>
#include<math.h>
#define round(a) ((int)(a+0.5))
void main()
{
    clrscr();
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

    if (graphresult() != grOk) {
        cout << "Graphics initialization failed." << endl;
        cin.get();
        return;
    }
    int xc = 100, yc = 150, r = 50;
    float x = 0, y = 0;
    for(int i = 0; i <= 45; i++)
    {
        double ang = double(i) * (3.142 / 180);
        x = r * cos(ang);
        y = r * sin(ang);
        putpixel(xc + round(x), yc + round(y), 15);
        putpixel(xc - round(x), yc + round(y), 15);
        putpixel(xc + round(x), yc - round(y), 15);
        putpixel(xc - round(x), yc - round(y), 15);
        putpixel(xc + round(y), yc + round(x), 15);
        putpixel(xc - round(y), yc + round(x), 15);
        putpixel(xc + round(y), yc - round(x), 15);
        putpixel(xc - round(y), yc - round(x), 15);
        delay(100);
    }
    cin.get();
    closegraph();
}
```



b) Mid-point Circle Algorithm:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
#define round(a) ((int)a + 0.5)
void putcircle(int xc, int yc, int x, int y)
{
    putpixel(xc+x, yc+y, 1);
    putpixel(xc-x, yc+y, 2);
    putpixel(xc+x, yc-y, 3);
    putpixel(xc-x, yc-y, 4);
    putpixel(xc+y, yc+x, 5);
    putpixel(xc-y, yc+x, 6);
    putpixel(xc+y, yc-x, 7);
    putpixel(xc-y, yc-x, 8);
}
void circlemid(int xc, int yc, float r)
{
    float x = 0, y = r;
    int p = 1 - r;
    while(x < y)
    {
        x++;
        if(p < 0)
            p = p + (2*x) + 1;
        else
        {
            y--;
            p = p + (2*(x - y) + 1);
        }
        putcircle(xc, yc, round(x), round(y));
        delay(50);
    }
}
void main()
{
    clrscr();
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI"); // Provide correct path for BGI
    int xc, yc, r;
    cout << "Enter centre co-ordinates:";
    cin >> xc >> yc;
    cout << "Enter radius:";
    cin >> r;
    circlemid(xc, yc, r);
    setcolor(10);
    circle(xc, yc, r);
    getch();
    closegraph();
}
```



```
Enter centre co-ordinates:250 190
Enter radius:40
```

5. Learning Outcome:

- **Circle Drawing**: Displays a circle using the midpoint algorithm, incrementally plotting points.
- **Geometric Shapes**: Draws multiple shapes (circle, rectangle, line, arc, ellipse) with labels and colors.
- **Circle with Trigonometry**: Plots a circle using trigonometric functions (cos and sin) to calculate points.