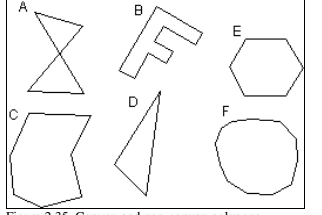


| **TITLE:** Draw the Following polygon/shape/curve.  a. Bresenham Circle Drawing Algorithm.  b. Other shapes |
| --- |

**AIM:**

a. Generate the Circle using Bresenham Circle Drawing Algorithm

b. Draw polygon shown in following figure



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO1: Understand the basic concepts of computer graphics and OpenGL**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

**https://www.gatevidyalay.com/bresenham-circle-drawing-algorithm/**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Algorithm:**

1. **Set initial values of (xc, yc) and (x, y)**
2. **Set decision parameter d to d = 3 – (2 \* r).**
4. **call drawCircle(int xc, int yc, int x, int y) function.**
5. **Repeat steps 5 to 8 until x < = y**
6. **Increment value of x.**
7. **If d < 0, set d = d + (4\*x) + 6**
8. **Else, set d = d + 4 \* (x – y) + 10 and decrement y by 1.**
9. **call drawCircle(int xc, int yc, int x, int y) function**

**drawCircle(int xc, int yc, int x, int y)**

**{**

**putpixel(xc+x, yc+y, RED);**

**putpixel(xc-x, yc+y, RED);**

**putpixel(xc+x, yc-y, RED);**

**putpixel(xc-x, yc-y, RED);**

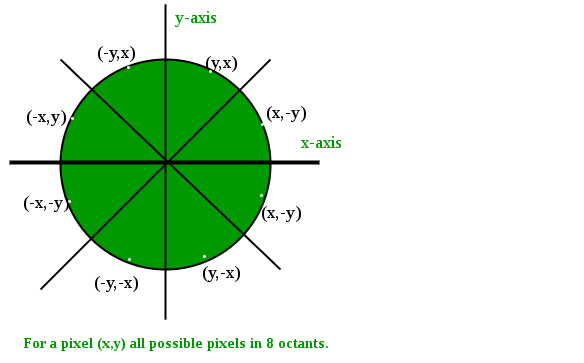
**putpixel(xc+y, yc+x, RED);**

**putpixel(xc-y, yc+x, RED);**

**putpixel(xc+y, yc-x, RED);**

**putpixel(xc-y, yc-x, RED);**

**}**

****

**Implementation details:**

**import** matplotlib.pyplot **as** plt

X**=**[]

Y**=**[]

**def** drawcircle(**cx**,**cy**,**x**,**y**):

X.append(cx**+**x)

Y.append(cy**+**y)

X.append(cx**-**x)

Y.append(cy**+**y)

X.append(cx**+**x)

Y.append(cy**-**y)

X.append(cx**-**x)

Y.append(cy**-**y)

X.append(cx**+**y)

Y.append(cy**+**x)

X.append(cx**-**y)

Y.append(cy**+**x)

X.append(cx**+**y)

Y.append(cy**-**x)

X.append(cx**-**y)

Y.append(cy**-**x)

**import** math

**def** draw(**cx**,**cy**,**r**):

x**=**0

y**=**r

d**=**3**-**(2**\***r)

drawcircle(cx,cy,x,y)

**while**(x**<=**y):

x**=**x**+**1

**if**(d**<**0):

d **=** d **+** (4**\***x) **+** 6

**else**:

d **=** d **+** 4 **\*** (x **-** y) **+** 10

y**=**y**-**1

drawcircle(cx,cy,x,y)

draw(50,50,30)

plt.scatter(X,Y)

plt.show()

**from** OpenGL.GL **import** **\***

**from** OpenGL.GLUT **import** **\***

**from** OpenGL.GLU **import** **\***

**def** draw\_line(**x1**, **y1**, **x2**, **y2**):

glVertex2f(x1, y1)

glVertex2f(x2, y2)

**def** display():

glClear(GL\_COLOR\_BUFFER\_BIT)

glColor3f(1.0, 1.0, 1.0) *#* ***Set color to white (RGB values: 1.0, 1.0, 1.0)***

glBegin(GL\_LINES)

draw\_line(0.0, 0.0, 0.0, 0.5)

draw\_line(0.0, 0.5, 0.5, 0.5)

draw\_line(0.5, 0.5, 0.5, 0.0)

draw\_line(0.5, 0.0, 0.0, 0.0)

glEnd()

glFlush()

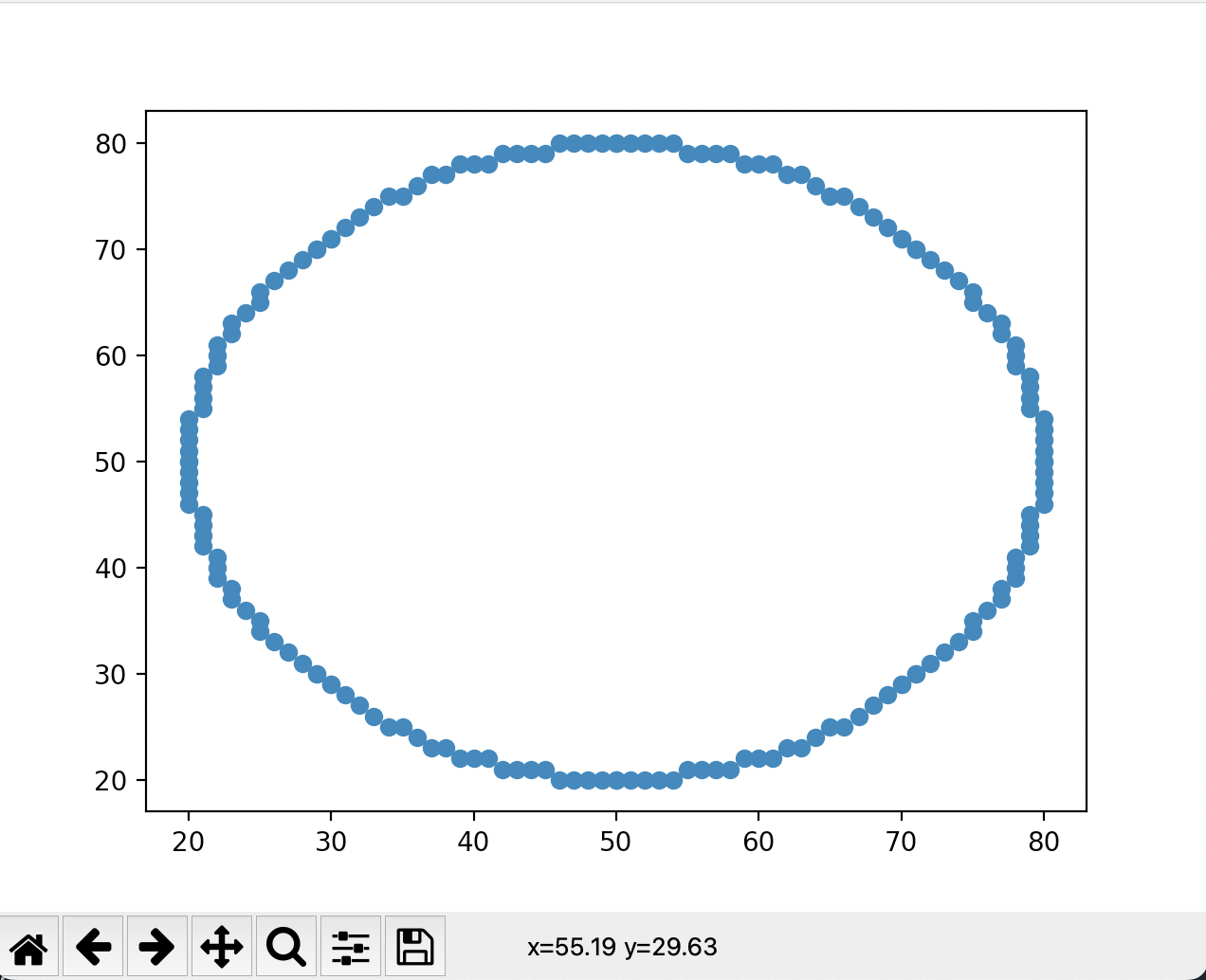
glutInit()

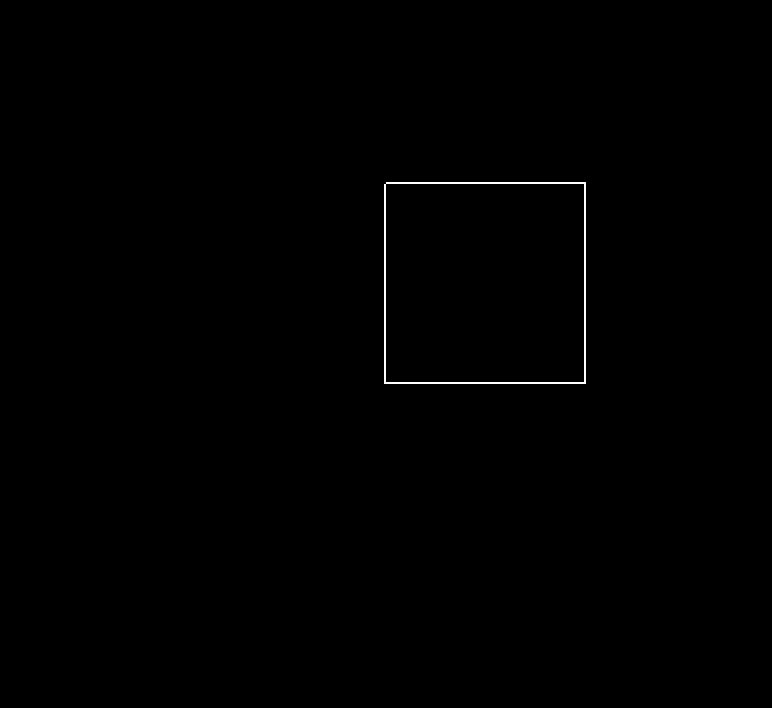
glutInitWindowSize(400, 400)

glutCreateWindow(b"PyOpenGL Line Example")

glutDisplayFunc(display)

glutMainLoop()

**Output(s) (final edited screen shot):**

**Conclusion and discussion:**

**Thus we have understood how to implement Bresenham’s circle drawing algorithm. Also, we used openGL to make polygons and draw them. We used various functions in openGL and understood how openGL works.**

**Date: 5 aug**

**Signature of faculty in-charge**

**Post lab**

**Visit and explore and paste your screenshot**

[**https://cse18-iiith.vlabs.ac.in/exp/rasterization-line/**](https://cse18-iiith.vlabs.ac.in/exp/rasterization-line/)

[**https://cse18-iiith.vlabs.ac.in/exp/rasterization-polygon/**](https://cse18-iiith.vlabs.ac.in/exp/rasterization-polygon/)