Computer Graphics

Lab File

Mecklon Fernandes

15

TYBCA

**INDEX**

|  |  |  |
| --- | --- | --- |
| Sr No. | Topic | Pg No. |
| 1 | Polynomial line algorithm | 3-4 |
| 2 | Digital Differential Analyser line algorithm | 5-7 |
| 3 | Bresenham line algorithm | 8-10 |
| 4 | Polynomial circle algorithm | 11-13 |
| 5 | Bresenham circle algorithm | 14-16 |
| 6 | Mid – Point circle algorithm | 17-19 |

Polynomial line algorithm

import matplotlib.pyplot as plt

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

    dx = x2 - x1

    dy = y2 - y1

    slope = float(dy / dx)

    b = y1 - x1 \* slope

    plt.ion()  # turn on interactive mode

    ax = plt.subplot()

    ax.set\_xlim(0, 20)

    ax.set\_ylim(0, 20)

    ax.set\_aspect("equal", adjustable="box")

    start = min(x1, x2)

    end = max(x1, x2)

    for i in range(start, end + 1):

        ax.plot(round(i), round(b + slope \* i), "bo")  # 'bo' means blue color and circle marker

        plt.draw()

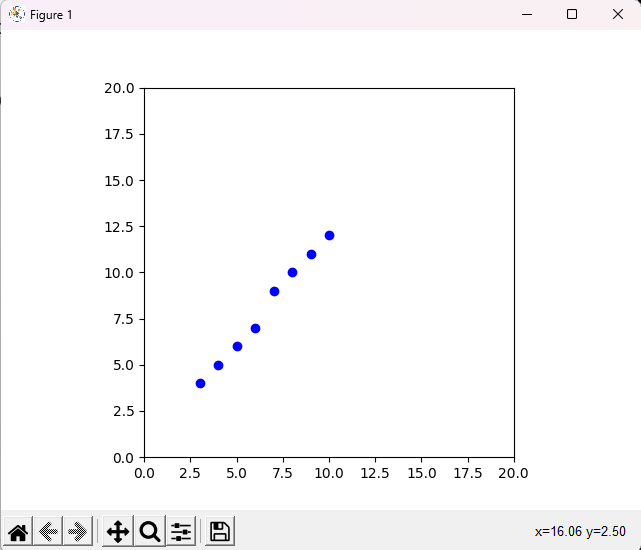
        plt.pause(0.01)  # 1 millisecond delay

    plt.ioff()  # turn off interactive mode

    plt.show()

draw\_line\_equation(3, 10, 4, 12)

**Output:**

****

Digital Differential Analyser line algorithm

import matplotlib.pyplot as plt

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

    dx = x2 - x1

    dy = y2 - y1

    steps = max(abs(dx), abs(dy))

    x\_increment = dx / steps

    y\_increment = dy / steps

    x = x1

    y = y1

    plt.ion()  # turn on interactive mode

    ax = plt.subplot()

    ax.set\_xlim(0, 800)

    ax.set\_ylim(0, 600)

    ax.set\_aspect("equal", adjustable="box")

    for i in range(int(steps) + 1):

        ax.plot(round(x), round(y), "bo")  # 'bo' means blue color and circle marker

        plt.draw()

        plt.pause(0.001)  # 1 millisecond delay

        x += x\_increment

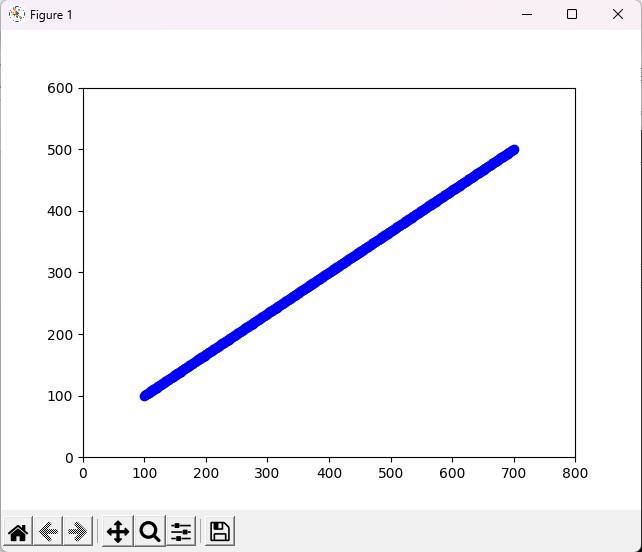
        y += y\_increment

    plt.ioff()  # turn off interactive mode

    plt.show()

draw\_line\_dda(100, 100, 700, 500)

**Output:**

****

Bresenham line algorithm

import matplotlib.pyplot as plt

import time

def bresenham(x1, y1, x2, y2):

    dx = x2 - x1

    dy = y2 - y1

    diff = 2\*(dy-dx)

    p0  = 2\*dy-dx

    steps = dx+1

    plt.ion()  # turn on interactive mode

    ax = plt.subplot()

    ax.set\_xlim(0, 40)

    ax.set\_ylim(0, 20)

    ax.set\_aspect("equal", adjustable="box")

    for i in range(int(steps)):

        ax.plot(round(x1), round(y1), "bo")  # 'bo' means blue color and circle marker

        plt.draw()

        plt.pause(0.01)  # 1 millisecond delay

        x1+=1

        if(p0<0):

            p0+=2\*dy

        else:

            p0+=diff

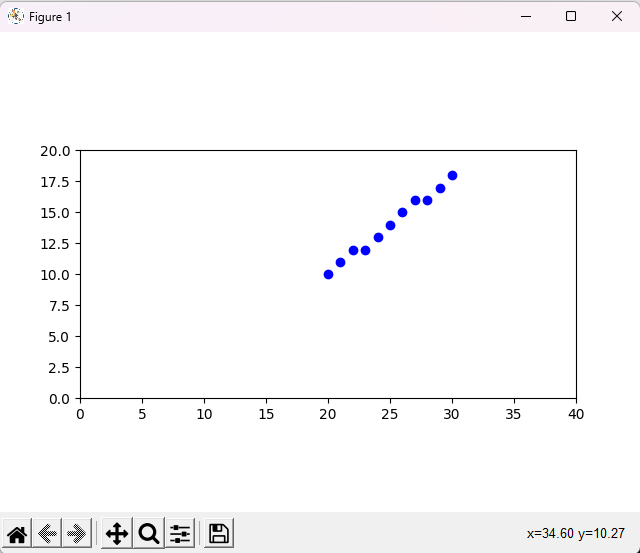
            y1+=1

    plt.ioff()  # turn off interactive mode

    plt.show()

bresenham (20,10,30,18)

**Output:**

****

Polynomial circle algorithm

import matplotlib.pyplot as plt

import math

def circle\_poly(h,k,r):

    x = 0

    y = 0

    xend = r/1.41

    plt.ion()  # turn on interactive mode

    ax = plt.subplot()

    ax.set\_xlim(-10, 10)

    ax.set\_ylim(-10, 10)

    ax.set\_aspect("equal", adjustable="box")

    while x<xend:

        y = int(round(math.sqrt(r\*r-x\*x)))

        ax.plot(x+h,y+k,"bo")

        ax.plot(-x+h,y+k,"bo")

        ax.plot(x+h,-y+k,"bo")

        ax.plot(-x+h,-y+k,"bo")

        ax.plot(y+h,x+k,"bo")

        ax.plot(-y+h,x+k,"bo")

        ax.plot(y+h,-x+k,"bo")

        ax.plot(-y+h,-x+k,"bo")

        plt.draw()

        plt.pause(0.01)

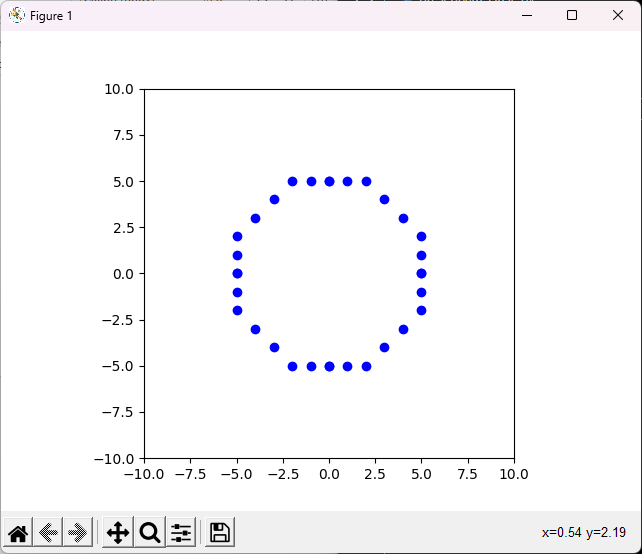
        x+=1

    plt.ioff()  # turn off interactive mode

    plt.show()

circle\_poly(0,0,5)

**Output:**

****

Bresenham circle algorithm

import matplotlib.pyplot as plt

def bresenham\_circle (h,k,r):

    x = 0

    y = r

    d = 3 - 2\*r

    print(x,y,r,h,k,d)

    plt.ion()  # turn on interactive mode

    ax = plt.subplot()

    ax.set\_xlim(h-r-5, h+r+5)

    ax.set\_ylim(k-r-5,k+r+5)

    ax.set\_aspect("equal", adjustable="box")

    while x<=y:

        ax.plot(x+h,y+k,"bo")

        ax.plot(-x+h,y+k,"bo")

        ax.plot(x+h,-y+k,"bo")

        ax.plot(-x+h,-y+k,"bo")

        ax.plot(y+h,x+k,"bo")

        ax.plot(-y+h,x+k,"bo")

        ax.plot(y+h,-x+k,"bo")

        ax.plot(-y+h,-x+k,"bo")

        plt.draw()

        plt.pause(0.01)

        if d < 0:

            d = d+4\*x+6

        else:

            d = d+4\*(x-y)+10

            y-=1

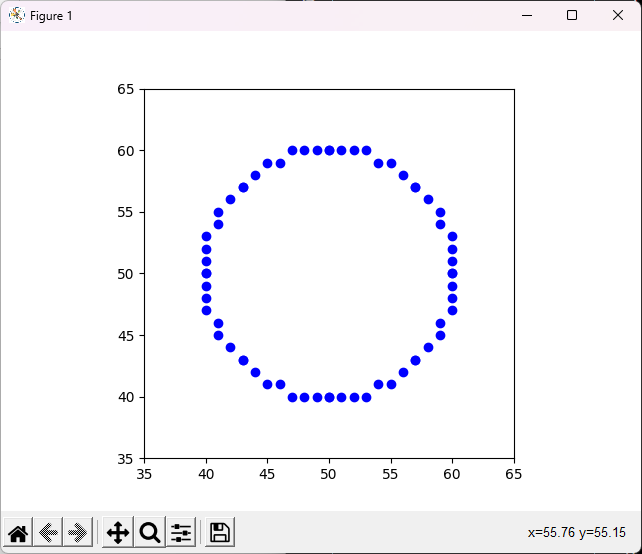
        x+=1

    plt.ioff()  # turn off interactive mode

    plt.show()

bresenham\_circle (50,50,10)

**Output:**

****

Mid – Point circle algorithm

import matplotlib.pyplot as plt

import math

def midPointCircle(h,k,r):

    x = 0

    y = r

    p = 1-r

    plt.ion()  # turn on interactive mode

    ax = plt.subplot()

    ax.set\_xlim(-20, 20)

    ax.set\_ylim(-20, 20)

    ax.set\_aspect("equal", adjustable="box")

    while x<=y:

        ax.plot(x+h,y+k,"bo")

        ax.plot(-x+h,y+k,"bo")

        ax.plot(x+h,-y+k,"bo")

        ax.plot(-x+h,-y+k,"bo")

        ax.plot(y+h,x+k,"bo")

        ax.plot(-y+h,x+k,"bo")

        ax.plot(y+h,-x+k,"bo")

        ax.plot(-y+h,-x+k,"bo")

        plt.draw()

        plt.pause(0.01)

        if p<0:

            p = p + 2\*x+2+1

        else:

            p = p + 2\*x+2 + 1 - 2\*y-2

            y-=1

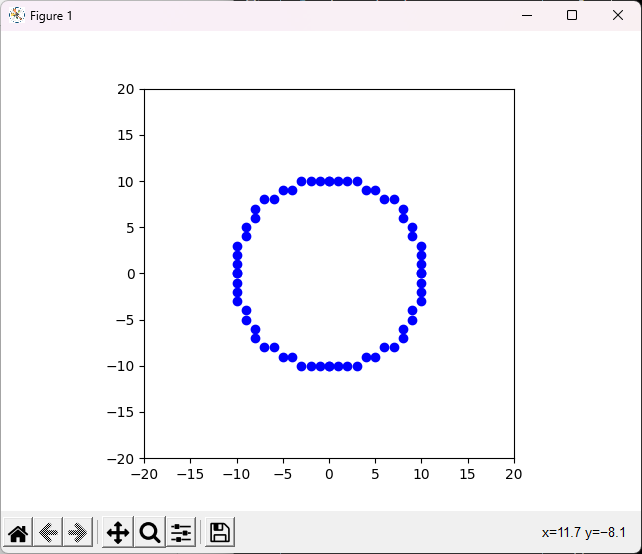
        x+=1

    plt.ioff()  # turn off interactive mode

    plt.show()

midPointCircle(0,0,10)

**Output:**

****