EXPERIMENT NO: 3 **DATE:08/09/2021**

CIRCLE DRAWING ALGORITHMS

AIM

Write a menu driven program to draw a circle using

A) Mid point circle drawing algorithm

B) Polar circle generation algorithm

C) Non-Polar circle generation algorithm

ALGORITHM

```
1. Display a menu and get user input to create circle according to any of the three
   algorithms
2. Input the center of the circle and radius of the circle and set it to xc, yc and r
   respectively
3. If choice = 1 (Midpoint circle drawing algorithm)
   o Set x,y as 0 and radius respectively.
   o Set p = 1 – r
   o Loop till x < y
       ▪ Increment x by 1
       ▪ If p < 0 add 2*x + 1 to p
       ▪ Else decrement y by 1 and add 2*x – 2*y + 1 to p
       ▪ Plot the point (x+xc, y+yc)
4. If choice = 2 (Polar Circle Generation Algorithm)
   o Set theta as 0 and target as (pi)/4
   o Loop till theta < target
       ▪ Set x as r*cos(theta) and y as r*sin(theta)
       ▪ Plot the 8 points according to symmetry
       ▪ Add some small value to theta (<0.2)
5. If choice = 3 (Non-polar Circle Generation Algorithm)
   o Set x as xc - r and target as xc + r
   o Loop till x < target
       ▪ Compute offset as sqrt(r^2 – (x-xc)^2)
       ▪ Plot points (x, yc + offset) and (x, yc – offset)
       ▪ Increment x by small value (<0.2)
```

PROGRAM

```
# Importing dependencies

import OpenGL # Standard interface for displaying

from OpenGL.GL import *

from OpenGL.GLU import *

from OpenGL.GLUT import *
```

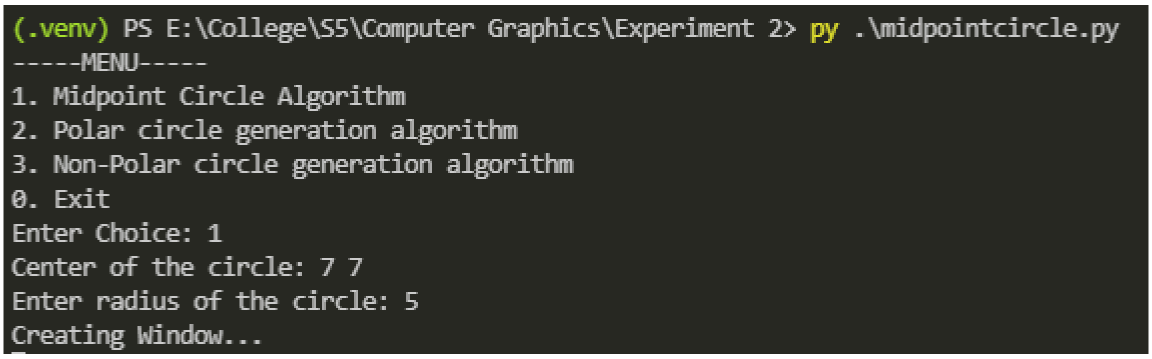
```
import sys
import math

WINDOW_POSITION = 50
POINT_SIZE = 10

def init():
    # Clear screen and set origin
    glClearColor(0.0, 0.0, 0.0, 1.0)                    # Set Background Color
    gluOrtho2D(0, WINDOW_POSITION, 0, WINDOW_POSITION)        # Set the Range of
coordinate system (x1, x2, y1, y2)

def display_menu():
    # Function to display menu
    print("-----MENU-----")
    print(f"1. Midpoint Circle Algorithm")
    print(f"2. Polar circle generation algorithm")
    print(f"3. Non-Polar circle generation algorithm")
    print(f"0. Exit")
    return int(input("Enter Choice: "))

def get_inputs():
    # Function to get input from user
    x, y = map(int, input("Center of the circle: ").split(" "))
    r = int(input("Enter radius of the circle: "))

    return x, y, r

def get_midpoint_circle_points(xc, yc, r):
    # Function to return points to plot
    # Points calculated using Midpoint circle drawing Algorithm
```

```
points = []

x, y = 0, r

points.append((x + xc, y + yc))

if r > 0:
    points.append((x+xc, -y+yc))
    points.append((y+xc, x+yc))
    points.append((-y+xc, x+yc))

p = 1 - r

while x < y:
    x += 1
    if p <= 0:
        p += 2*x + 1
    else:
        y -= 1
        p += 2*x - 2*y + 1
    points.append((x+xc, y+yc))
    points.append((-x+xc, y+yc))
    points.append((x+xc, -y+yc))
    points.append((-x+xc, -y+yc))

    if x != y:
        points.append((y+xc, x+yc))
        points.append((-y+xc, x+yc))
        points.append((y+xc, -x+yc))
        points.append((-y+xc, -x+yc))
    return points

def get_polar_circle_points(xc, yc, r):
    # Function to return points to plot
    # Points calculated using Polar circle generation Algorithm
    points = []

    theta = 0
    increment_by = 1 / 15
    target = math.pi / 4

    while theta <= target:
        x = r * math.cos(theta)
        y = r * math.sin(theta)
        for i in [-1, 1]:
            for j in [-1, 1]:
                points.append((x*i + xc, y*j + yc))
                points.append((y*j + yc, x*i + xc))
        theta += increment_by
    return points

def get_nonpolar_circle_points(xc, yc, r):
    # Function to return points to plot
    # Points calculated using Non-Polar circle generation Algorithm
    points = []
    x = xc - r
    target = xc + r
    increment_value = 1/50
    x += increment_value
    while x < target:
        offset = math.sqrt(r**2 - (x-xc)**2)
        points.append((x, yc + offset))
        points.append((x, yc - offset))
        x += increment_value
    return points



def plot_line(points):
    # Function to plot points
    glClear(GL_COLOR_BUFFER_BIT)
    glColor3f(1.0,0.0,0.0)
    glPointSize(POINT_SIZE)
    glBegin(GL_POINTS)

    # Plot the points
    for x, y in points:
        glVertex2f(x, y)

    glEnd()
    glFlush()

def display_window(x, y, r, choice):
    # Function to display window
    print("Creating Window...")
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_RGB)
    glutInitWindowSize(500,500)
    glutInitWindowPosition(50, 50)
    title = {
        1: "Midpoint Circle",
        2: "Polar Circle Generation",
        3: "Non-Polar Circle Generation",
    }
    glutCreateWindow(f"Plot Circle using {title[choice]} Algorithm")
    if choice == 1:
        points = get_midpoint_circle_points(x, y, r)
    elif choice == 2:
        points  = get_polar_circle_points(x, y, r)
    elif choice == 3:
        points = get_nonpolar_circle_points(x, y, r)
    else:
        points = []
    glutDisplayFunc(lambda: plot_line(points))
    init()
    glutMainLoop()


def main():
    choice = 1
    while choice != 0:
        choice = display_menu()
        if choice == 1 or choice == 2 or choice == 3:
            # Checks if it's a valid input (i.e. present in dictionary)
            x, y, r = get_inputs()
            display_window(x, y, r, choice)
        elif choice == 0:
            # To handle exit from program
            print("Exiting Program...")
        else:
            # To handle invalid choice
        print("Invalid Choice! Try again.")


main()
```

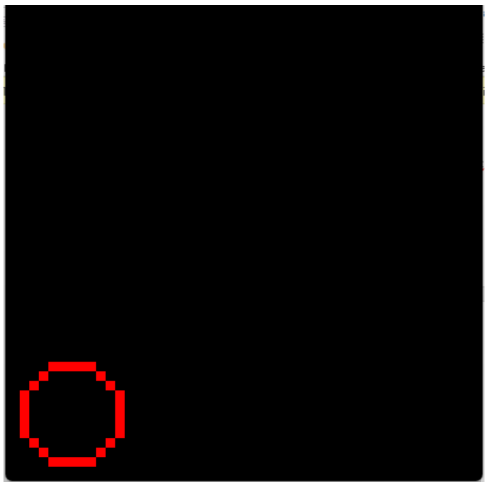
RESULT

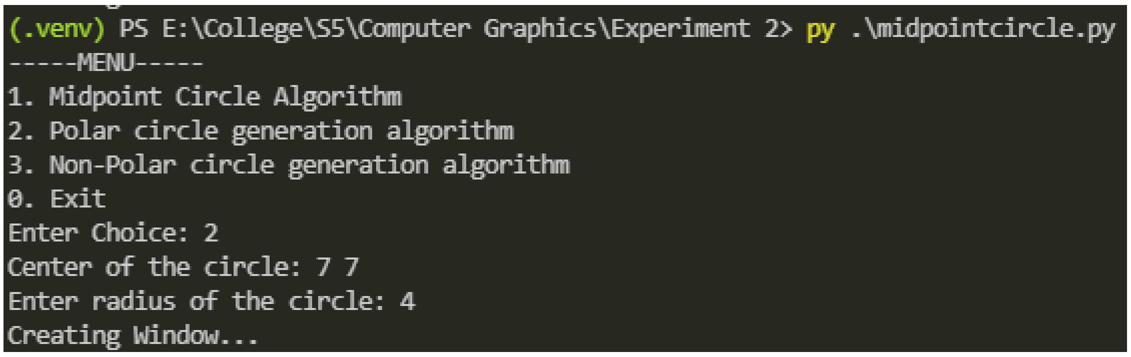
Program to draw a circle using a menu driven program using Mid point circle drawing algorithm, Polar circle generation algorithm, Non-Polar circle generation algorithm

OUTPUT/INPUT

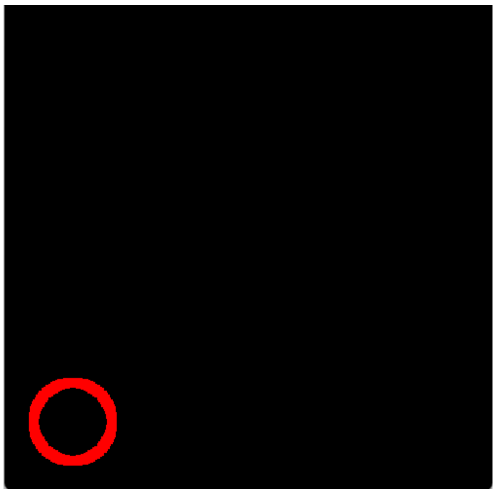


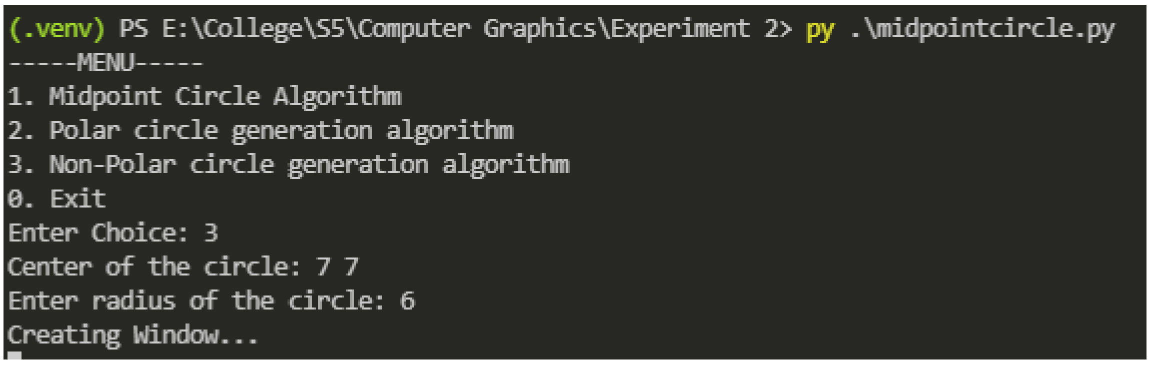
```
(.venv) PS E:\College\S5\Computer Graphics\Experiment 2> py .\midpointcircle.py
-----MENU-----
1. Midpoint Circle Algorithm
2. Polar circle generation algorithm
3. Non-Polar circle generation algorithm
0. Exit
Enter Choice: 1
Center of the circle: 7 7
Enter radius of the circle: 5
Creating Window...
```





```
(.venv) PS E:\College\S5\Computer Graphics\Experiment 2> py .\midpointcircle.py
-----MENU-----
1. Midpoint Circle Algorithm
2. Polar circle generation algorithm
3. Non-Polar circle generation algorithm
0. Exit
Enter Choice: 2
Center of the circle: 7 7
Enter radius of the circle: 4
Creating Window...
```





```
(.venv) PS E:\College\S5\Computer Graphics\Experiment 2> py .\midpointcircle.py
-----MENU-----
1. Midpoint Circle Algorithm
2. Polar circle generation algorithm
3. Non-Polar circle generation algorithm
0. Exit
Enter Choice: 3
Center of the circle: 7 7
Enter radius of the circle: 6
Creating Window...
```