Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Lab Report 3: Mid Point Circle & Ellipse Drawing Algorithms

COMP 342

(For partial fulfillment of 3rd Year/ 2nd Semester in Computer Engineering)

Submitted to:

Mr. Dhiraj Shrestha

Department of Computer Science and Engineering

Submitted by:

Neha Malla

C.E.

Roll No. 27

1. **Title:** Implementing Mid Point Circle Drawing Algorithm.

Algorithm:

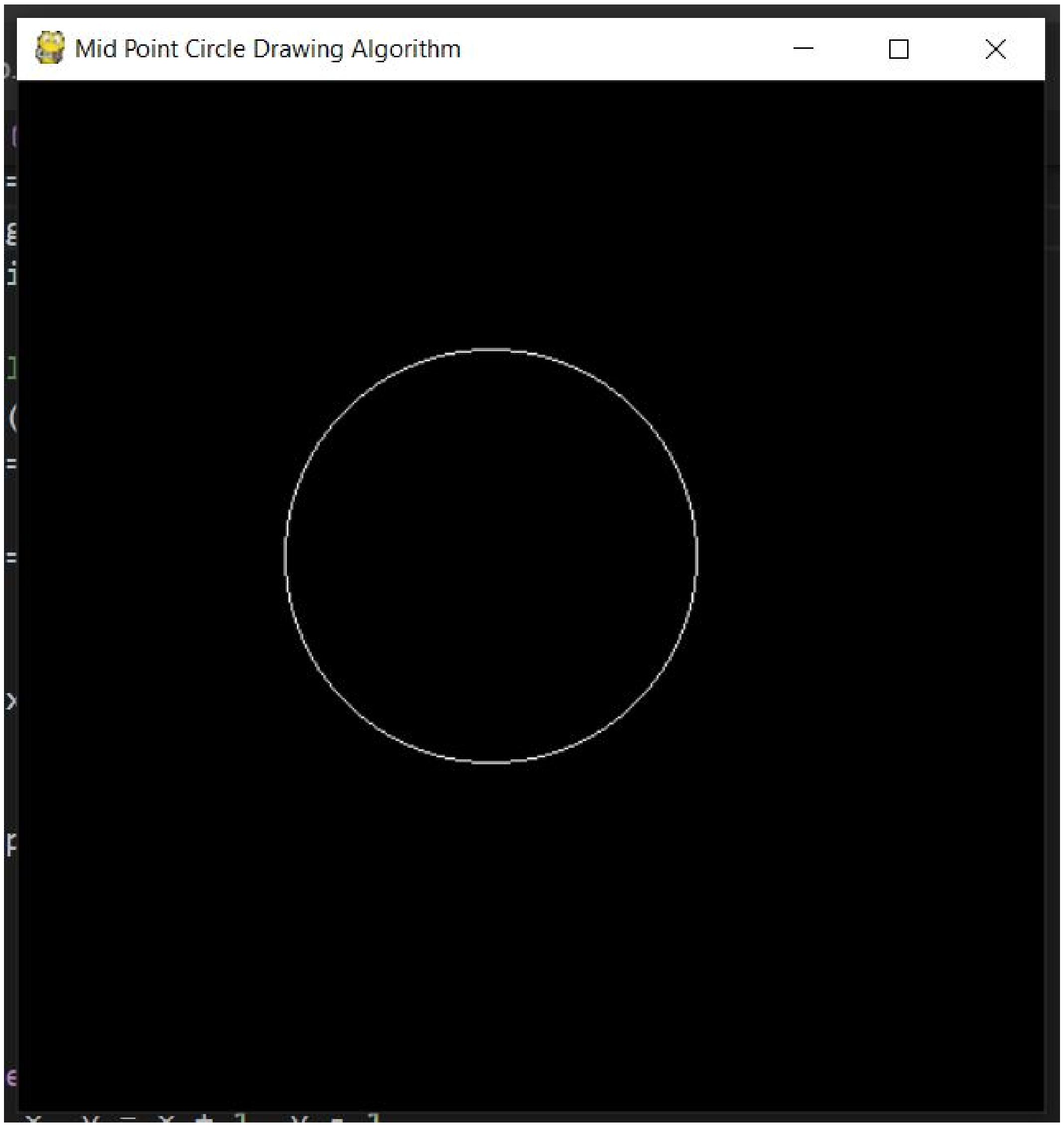
```
1.  Input center (xc, yc) and radius (r) of the circle
2.  First point on the circle: (x0, y0) = (0, r)
3.  Initial decision parameter (p0):
    if type(r) == int:
        p0 = 1 - r
    else:
        p0 = 5/4 - r
4.  At each xk, starting at k = 0
    if pk<0
        x(k+1), y(k+1) = xk + 1, yk
        p(k+1) = pk + 2x(k+1) + 1
    else:
        x(k+1), y(k+1) = xk + 1, yk - 1
        p(k+1) = pk + 2x(k+1) - 2y(k+1) + 1
5.  Determine symmetry points of other 7 octants
6.  Calculate the plotting points i.e. x = x + xc and y = y + yc and plot
7.  Repeat step 4 to 6 until x >= y
```

Source code:

```python
import pygame
import pygame.gfxdraw

# Initializing the game engine
pygame.init()
```

```python
# Colors in RGB format
white = (255, 255, 255)
blue = (0, 56, 147)
black = (0, 0, 0)
red = (255, 0, 0)

# height and width of the screen
width = 500
height = 500
screen = pygame.display.set_mode([width, height], pygame.RESIZABLE)
pygame.display.set_caption("Mid Point Circle Drawing Algorithm")

# Function for mid point circle drawing algo
def MidPointCircle(xc, yc, radius):
    x0, y0 = 0, radius
    pygame.gfxdraw.pixel(screen, xc + x0, yc + y0, white)
    EightPointSymmetricity(x0 ,y0, xc, yc)

    # initial decision parameter
    if type(radius) == float:
        p0 = (5/4)  - radius
    else:
        p0 = 1 - radius

    pk = p0
    x, y = x0, y0

    while x <= y:
        if pk < 0:
            x, y = x + 1, y
            pk = pk + 2 * x + 1
            pygame.gfxdraw.pixel(screen, x + xc, y + yc, white)
            EightPointSymmetricity(x, y, xc, yc)
        else:
            x, y = x + 1, y - 1
            pk = pk + 2 * x - 2 * y + 1
            pygame.gfxdraw.pixel(screen, x + xc, y + yc, white)
            EightPointSymmetricity(x, y, xc, yc)

# Drawing pixels using Circle's 8 point symmetric property
def EightPointSymmetricity(x,y,xc,yc):
    pygame.gfxdraw.pixel(screen, x + xc, -y + xc, white)
    pygame.gfxdraw.pixel(screen, -x + xc, y + yc, white)
    pygame.gfxdraw.pixel(screen, -x + xc, -y + yc, white)
    pygame.gfxdraw.pixel(screen, y + xc, x + yc, white)
    pygame.gfxdraw.pixel(screen, y + xc, -x + yc, white)
    pygame.gfxdraw.pixel(screen, -y + xc, -x + yc, white)
    pygame.gfxdraw.pixel(screen, -y + xc, x + yc, white)



done = False

while not done:
    # If user clicks close
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    # The screen background as white
    screen.fill(black)

    # circle with radius 100 & center at 230, 230
    MidPointCircle(230, 230, 100)

    # Update the contents of the entire display
    pygame.display.flip()

pygame.quit()
```

Output:



2. **Title:** Implementing Mid Point Ellipse Drawing Algorithm.

Algorithm:

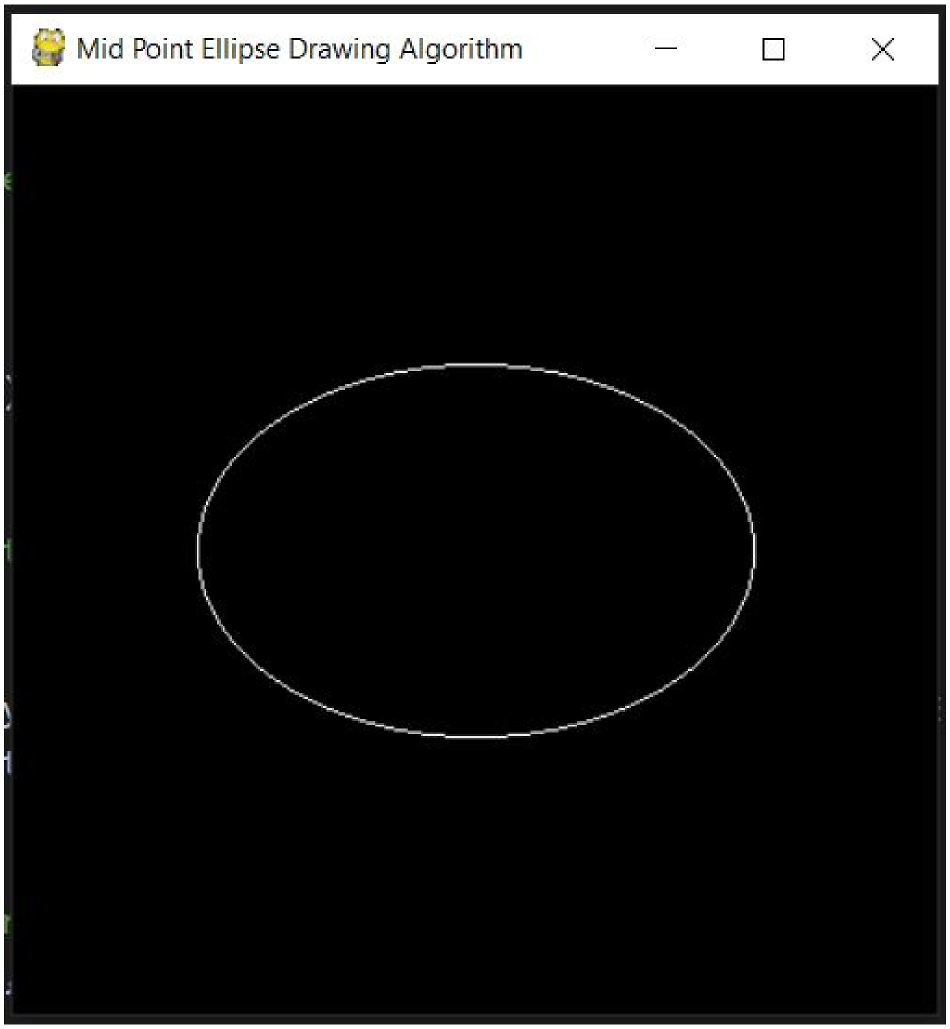
```
1. Input center (xc, yc) and radius rx & ry
2. First point on the ellipse: (0, ry)
3. Initial decision parameter in Region 1:
   p10 = ry^2 - rx^2.ry + ¼rx^2
4. At each xk, starting at k = 0, in region 1:
   a. if p1k < 0:
              x(k+1), y(k+1) = xk + 1, yk
              p1(k+1) = p1k + 2ry^2.x(k+1) + ry^2
      else:
              x(k+1), y(k+1) = xk + 1, yk - 1
              p1(k+1) = p1k + 2ry^2.x(k+1) - 2rx^2.y(k+1) + ry^2
   b. Repeat until 2ry^2.x >= 2rx^2.y
5. Initial decision parameter in region 2 with last point (x0, y0) in region 1:
   p20 = ry^2(x0 + 1/2)^2 + rx^2(y0 - 1)^2 - rx^2.ry^2
6. At each yk, starting at k = 0, in region 2:
   a. if p2k <= 0:
              x(k+1), y(k+1) = xk + 1, yk - 1
              p2(k+1) = p2k + 2ry^2.x(k+1) - 2rx^2.y(k+1) + rx^2
      else:
              x(k+1), y(k+1) = xk, yk - 1
              p2(k+1) = p2k - 2rx^2.y(k+1) + rx^2
   b. Repeat until y = 0
```

7. Determine symmetry points in other 3 quadrants
8. Calculate the plotting points i.e. x = x + xc and y = y + yc and plot

Source code:

```python
import pygame
import pygame.gfxdraw

# Initializing the game engine
pygame.init()

# Colors in RGB format
white = (255, 255, 255)
black = (0, 0, 0)

# height and width of the screen
width = 500
height = 500
screen = pygame.display.set_mode([width, height], pygame.RESIZABLE)
pygame.display.set_caption("Mid Point Ellipse Drawing Algorithm")

# Function for mid point ellipse drawing algo
def MidPointEllipse(rx, ry, xc, yc):
    # Region 1
    x , y = 0 , ry
    pygame.gfxdraw.pixel(screen, x + xc, y + yc, white)
    OtherThreeQuadrants(x, y, xc, yc)

    # Initial decision parameter
    p1 = ry**2 - (rx**2 * ry) + (1/4)*(rx**2)
    A = 2 * ry**2 * x
    B = 2 * rx**2 * y

    while A < B:
        x = x + 1
        A = A + 2 * ry**2
        if p1 < 0:
            p1 = p1 + A + ry**2
        else:
            y = y -1
            B = B - 2 * rx**2
            p1 = p1 + A - B + ry**2
        pygame.gfxdraw.pixel(screen, x + xc, y + yc, white)
        OtherThreeQuadrants(x, y, xc, yc)

    # Region 2
    # Initial decision parameter
    p2 = ry**2 * (x + (1/2))**2 + rx**2 * (y-1)**2 - (rx**2 * ry**2)

    while y >= 0:
        y = y - 1
        B = B - 2 * rx**2
        if p2 > 0:
            p2 = p2 + rx**2 - B
        else:
            x = x+1
            A = A + 2 * ry**2
            p2 = p2 + A - B + rx**2
        pygame.gfxdraw.pixel(screen, x + xc, y + yc, white)
        OtherThreeQuadrants(x, y, xc, yc)

# Drawing pixels in other 3 quadrants
def OtherThreeQuadrants(x, y, xc, yc):
    pygame.gfxdraw.pixel(screen, x + xc, -y + yc, white)
    pygame.gfxdraw.pixel(screen, -x + xc, y + yc, white)
    pygame.gfxdraw.pixel(screen, -x + xc, -y + yc, white)

done = False

while not done:
    # If user clicks close
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    # The screen background as white
    screen.fill(black)

    # circle with rx = 160, ry = 100 & center at 200, 200
    MidPointEllipse(160, 100, 200, 200)

    # Update the contents of the entire display
    pygame.display.flip()

pygame.quit()
```

Output:



Conclusion:

The Mid Point Circle drawing algorithm and Mid Point Ellipse drawing algorithm were implemented successfully for a given set of center coordinates and radius which are taken as input and the circle & ellipse are generated pixel by pixel using the algorithms.