Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Lab Report 2: Line 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 Digital Differential Analyzer Line drawing algorithm.

Algorithm:

1. Input (x1, y1) and (x2, y2)
2. dx = x2 - x1
   dy = y2 - y1
3. if abs(dx) > abs(dy):
   stepSize = abs(dx)
   xinc = 1
   yinc = dy/stepSize
   else:
   stepSize = abs(dy)
   xinc = abs(dx/stepSize)
   yinc = 1 if y1 < y2 else -1
4. plot(x1, y1)
5. Repeat until stepSize:
   a. x1 += xinc
   y1 += yinc
   b. plot(Round(x1), Round(y1))

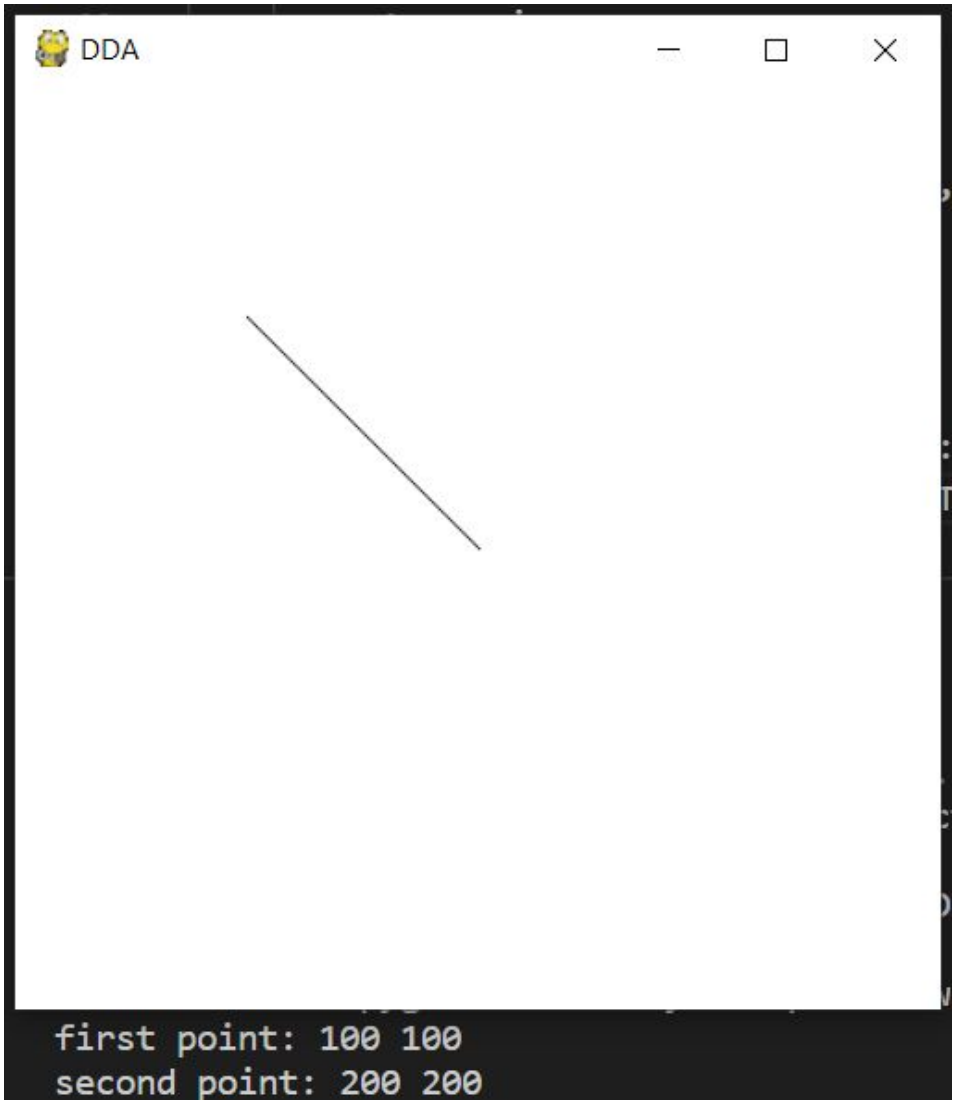
Source code:

```
import pygame
import pygame.gfxdraw

pygame.init()

white = (255, 255, 255)
black = (0, 0, 0)

x1, y1 = map(int,input("first point: ").split())
x2, y2 = map(int,input("second point: ").split())

screen = pygame.display.set_mode([400, 400], pygame.RESIZABLE)
pygame.display.set_caption("DDA")
```

```
done = False
points = []
count = 1

def Round(x):
    if x >= 0:
        return round(x + 0.1)
    else:
        return round(x - 0.1)

def DDA(x1, y1, x2, y2):
    dx = x2 - x1
    dy = y2 - y1

    if abs(dx) > abs(dy):
        stepSize = abs(dx)
        xinc = 1
        yinc = dy/stepSize
    else:
        stepSize = abs(dy)
        xinc = abs(dx/stepSize)
        yinc = 1 if y1 < y2 else -1

    pygame.gfxdraw.pixel(screen, x1, y1, black)
    points.append((x1,y1))

    for i in range(stepSize):
        x1 = x1 + xinc
        y1 = y1 + yinc
        pygame.gfxdraw.pixel(screen, Round(x1), Round(y1), black)
        points.append((Round(x1),Round(y1)))

while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
    screen.fill(white)
```

```
    DDA(x1, y1, x2, y2)

    while count > 0:
        for point in points:
            print(point)
        count -= 1

    pygame.display.flip()
pygame.quit()
```

Output:



2. **Title:** Implementing Bresenham Line Drawing algorithm for both slopes(|m|<1 and |m|>=1).

Algorithm:

1. Input (x1, y1) and (x2, y2)
2. plot(x1, y1)
3. a. dx = abs(x2 - x1)
   dy = abs(y2 - y1)
   b. if dx == 0:
   m = 1
   else:
   m = abs(dy/dx)
4. if m < 1:
   pk = 2 * dy - dx
   plot(x1, y1)

   Repeat until dx:
   if pk < 0:
   x += 1 if x2 > x1 else -1
   y = y
   pk = pk + 2 * dy
   else:
   x += 1 if x2 > x1 else -1
   y += 1 if y2 > y1 else -1
   pk = pk + 2 * (dy - dx)
   plot(x, y)

else if m >= 1:
   pk = 2 * dx - dy
   plot(x1, y1)

   Repeat until dy:
   if pk < 0:
   x = x

```
            y += 1 if y2 > y1 else -1
            pk = pk + 2 * dx
    else:
            x += 1 if x2 > x1 else -1
            y += 1 if y2 > y1 else -1
            pk = pk + 2 * (dx - dy)
    plot(x, y)
```

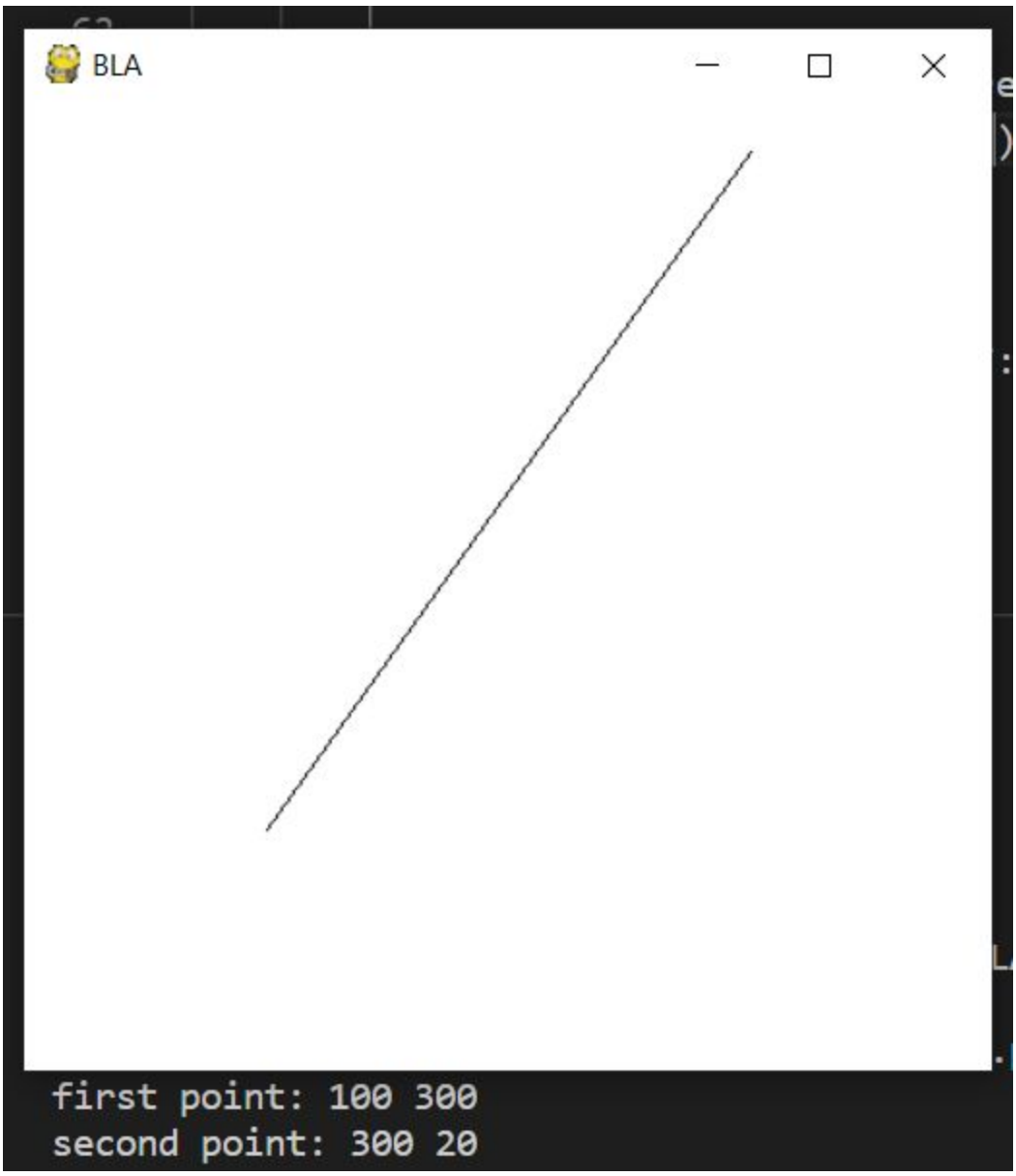
Source code:

```
import pygame
import pygame.gfxdraw

pygame.init()

white = (255, 255, 255)
black = (0, 0, 0)

x1, y1 = map(int,input("first point: ").split())
x2, y2 = map(int,input("second point: ").split())

width = 400
height = 400
screen = pygame.display.set_mode([width, height], pygame.RESIZABLE)
pygame.display.set_caption("BLA")

done = False
points = []
count = 1

def BLA(x1, y1, x2, y2):
    dx = abs(x2 - x1)
    dy = abs(y2 - y1)
    x, y = x1, y1

    if dx == 0:
        m = 0
```

```
    else:
        m = dy / dx

    if m < 1:
        pk = 2 * dy - dx
        pygame.gfxdraw.pixel(screen, x1, y1, black)
        points.append((pk, x, y))

        for i in range(dx):
            if pk < 0:
                x += 1 if x2 > x1 else -1
                y = y
                pk = pk + 2 * dy
            else:
                x += 1 if x2 > x1 else -1
                y += 1 if y2 > y1 else -1
                pk = pk + 2 * (dy - dx)

            pygame.gfxdraw.pixel(screen, x, y, black)
            points.append((pk, x, y))
    elif m >= 1:
        pk = 2 * dx - dy
        pygame.gfxdraw.pixel(screen, x1, y1, black)
        points.append((pk, x, y))

        for i in range(dy):
            if pk < 0:
                x = x
                y += 1 if y2 > y1 else -1
                pk = pk + 2 * dx
            else:
                x += 1 if x2 > x1 else -1
                y += 1 if y2 > y1 else -1
                pk = pk + 2 * (dx - dy)

            pygame.gfxdraw.pixel(screen, x, y, black)
            points.append((pk, x, y))
```

```
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    screen.fill(white)

    BLA(x1, y1, x2, y2)

    while count > 0:
        for point in points:
            print(point)
        count -= 1

    pygame.display.flip()

pygame.quit()
```

Output:



3. **Title:** Implementing Midpoint algorithm Line drawing for both slopes(|m|<1 and |m|>=1).

Algorithm:

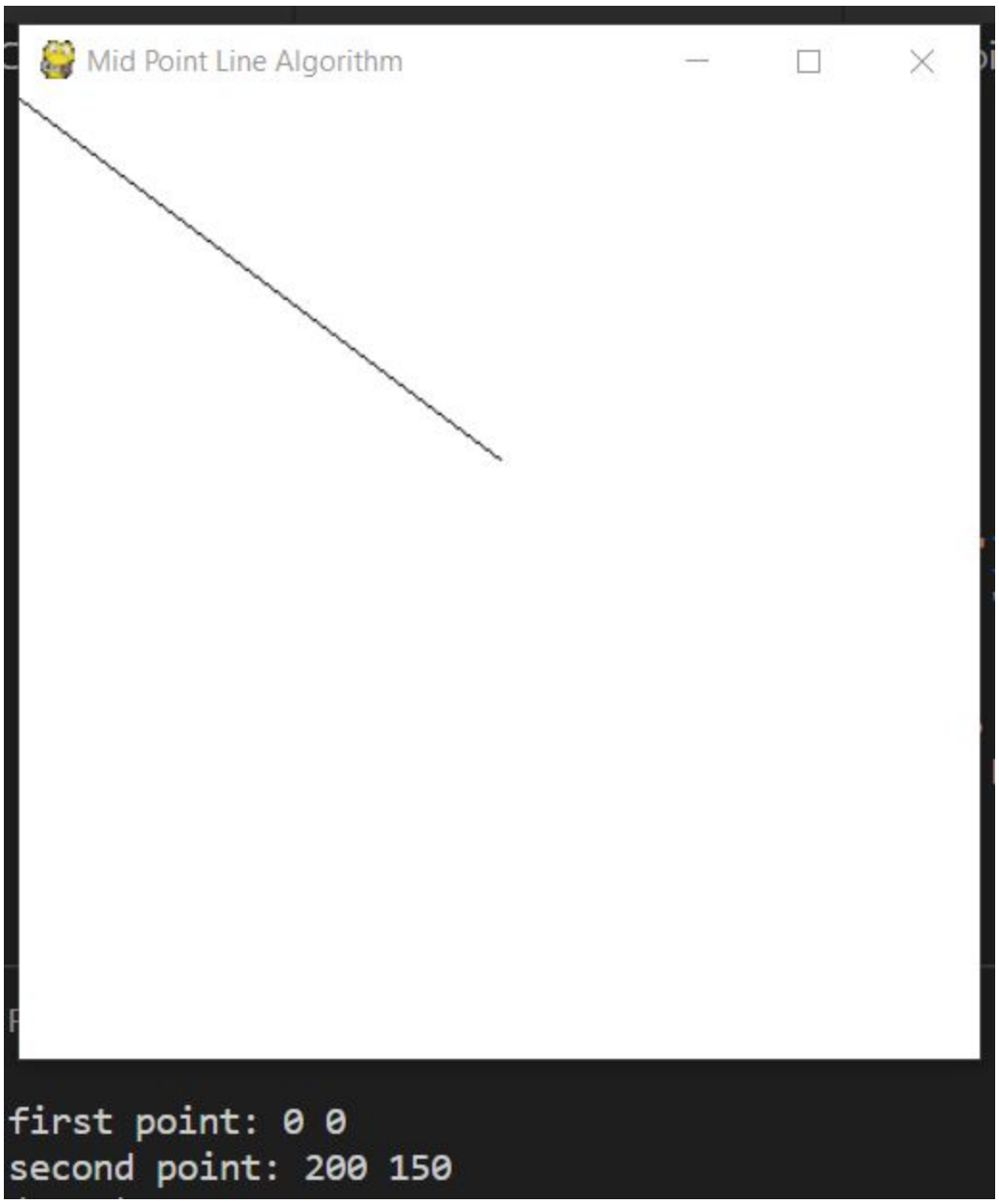
1. Input (x1, y1) and (x2, y2)
2. plot(x1, y1)
3. dx = abs(x2 - x1)
   dy = abs(y2 - y1)
4. pk = 2 * dy - dx
   east = 2 * dy
   northeast = 2 * (dy - dx)
5. x, y = x1, y1
   if dx > dy:
       step = dx
   else:
       step = dy
6. Repeat until step:
   a. if pk < 0:
           x += 1 if x2 > x1 else -1
           y = y
           pk = pk + east
       else:
           x += 1 if x2 > x1 else -1
           y += 1 if y2 > y1 else -1
           pk = pk + northeast
   b. plot(x, y)

Source code:

```
import pygame
import pygame.gfxdraw

pygame.init()

white = (255, 255, 255)
black = (0, 0, 0)

x1, y1 = map(int,input("first point: ").split())
x2, y2 = map(int,input("second point: ").split())

screen = pygame.display.set_mode([400, 400], pygame.RESIZABLE)
pygame.display.set_caption("Mid Point Line Algorithm")

done = False
points = []
count = 1

def midpoint_line_drawing(window, x1, y1, x2, y2):
    dx = abs(x2 - x1)
    dy = abs(y2 - y1)

    pk = 2 * dy - dx

    east = 2 * dy
    northeast = 2 * (dy - dx)

    pygame.gfxdraw.pixel(window, x1, y1, black)
    x, y = x1, y1
    points.append((x, y))

    if dx > dy:
        step = dx
    else:
        step = dy
```

```
    while x < x2 and y < y2:
        if pk < 0:
            x += 1 if x2 > x1 else -1
            y = y
            pk = pk + east

        else:
            x += 1 if x2 > x1 else -1
            y += 1 if y2 > y1 else -1
            pk = pk + northeast

        pygame.gfxdraw.pixel(window, x, y, black)
        points.append((x, y))

while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    screen.fill(white)

    midpoint_line_drawing(screen, x1, y1, x2, y2)

    while count > 0:
        for point in points:
            print(point)
        count -= 1

    pygame.display.flip()

pygame.quit()
```

Output:



Conclusion:

The three line drawing algorithms: Digital Differential Analyzer(DDA), Bresenham's Line Algorithm(BLA) and Mid Point Line Algorithm were implemented successfully for drawing lines with positive and negative slopes of any magnitude. The starting and ending points of the line are taken as input and the line is generated pixel by pixel using the algorithms.