Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Mini Report

on

"Lab 2"

[Course Code: COMP 342]

(For partial fulfillment of III Year/ I Semester in Computer Science)

Submitted By

Nisham Ghimire (22)

Submitted To

Mr. Dhiraj Shrestha

Department of Computer Science and Engineering

Submission Date

27th October, 2023

1. Implement Digital Differential Analyzer Line drawing algorithm.

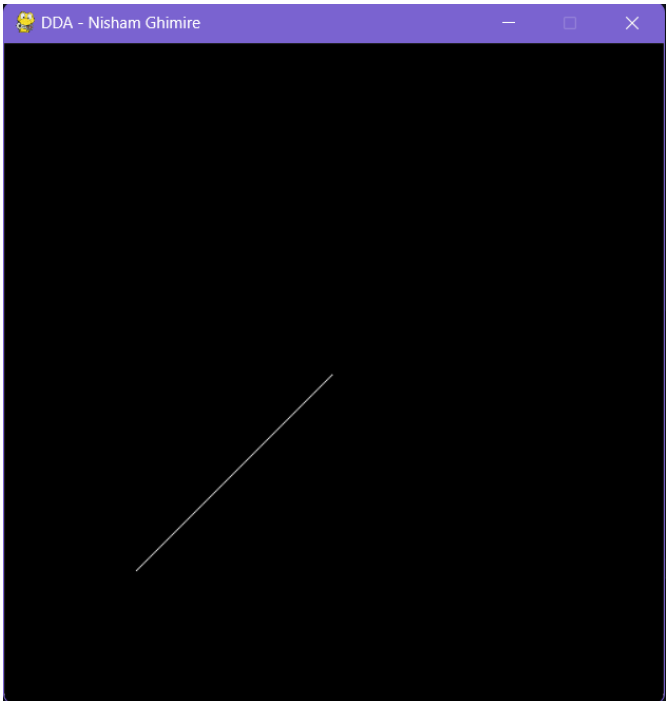
Ans:

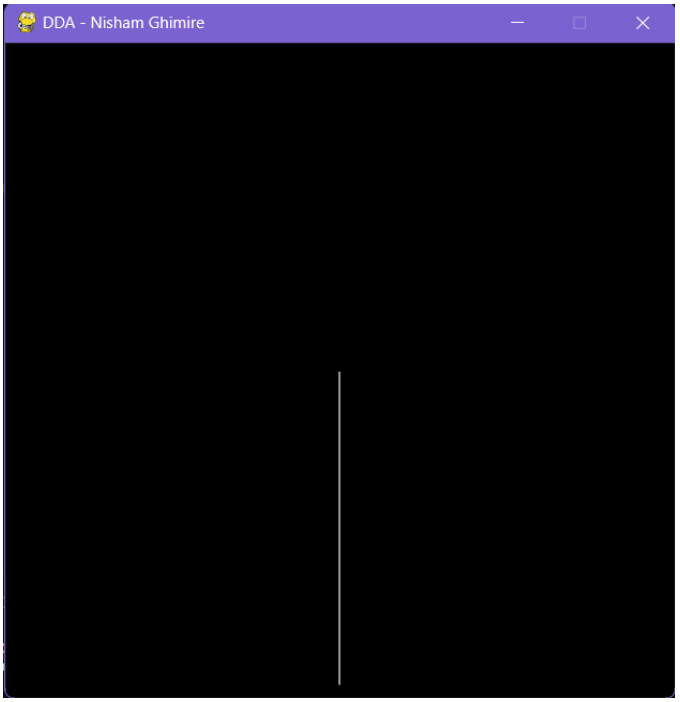
1. *Start Algorithm*
2. *Declare variables:*
   Declare x1, y1, x2, y2, dx, dy, steps, Xinc, Yinc, X, Y, and vertices.
3. *Input coordinates: Assign the values of x1, y1, x2, and y2.*
4. *Calculate dx and dy: Calculate dx as x2 - x1 and dy as y2 - y1.*
5. *Calculate steps: Calculate steps as the maximum of abs(dx) and abs(dy).*
6. *Calculate Xinc and Yinc: Calculate Xinc as dx / steps and Yinc as dy / steps.*
7. *Initialize X and Y: Assign X to x1 and Y to y1.*
8. *Calculate vertices: Repeat steps times:*
   a. Append (X, Y) to vertices.
   b. Add Xinc to X.
   c. Add Yinc to Y.
9. *Append end coordinate: Append (x2, y2) to vertices.*
10. *Draw line: For each vertex in vertices, call glVertex2f with the x and y coordinates of the vertex.*
11. *End Algorithm*

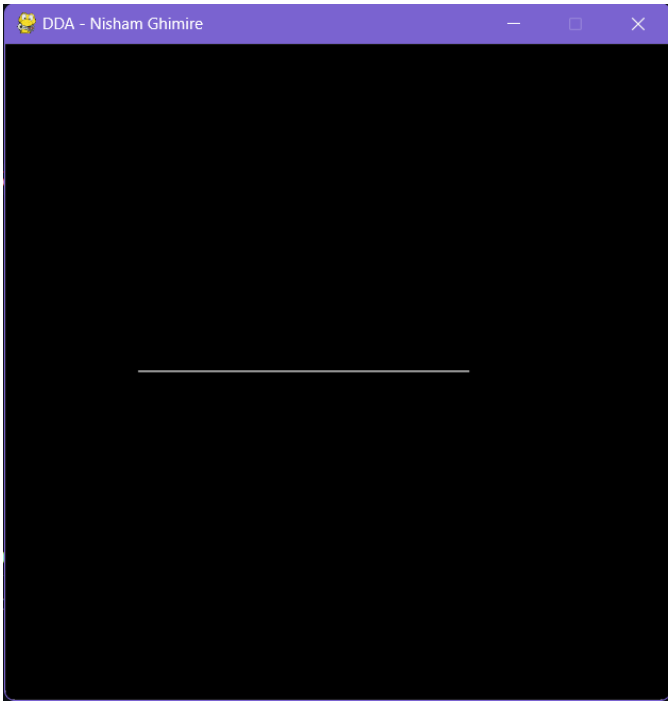
```
from typing import Tuple
import pygame as pg
from pygame import display, event
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLU import *

def DDA(start_coordinate: Tuple[int, int], end_coordinate: Tuple[int,
int]) -> list[Tuple[float, float]]:
    x1, y1 = start_coordinate
    x2, y2 = end_coordinate

    dx = x2 - x1
    dy = y2 - y1

    steps = max(abs(dx), abs(dy))

    Xinc = dx / steps
    Yinc = dy / steps

    X = x1
    Y = y1
    vertices: list[Tuple[float, float]] = []

    for i in range(steps):
        vertices.append((X, Y))
        X = X + Xinc
```

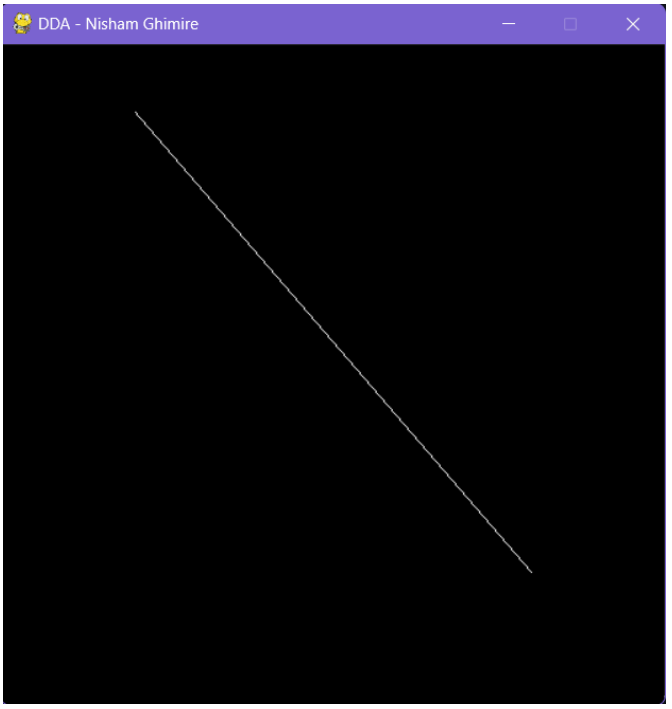
```
        Y = Y + Yinc
    return vertices

def drawDDA():
    vertices = DDA((100, 250), (350, 450))
    glClear(GL_COLOR_BUFFER_BIT)
    glBegin(GL_LINE_STRIP)
    glColor3f(1.0, 1.0, 1.0)
    for v in vertices:
        x, y = v
        glVertex2f(x, y)
    glEnd()
    glFlush()

def main():
    pg.init()
    display.set_mode((500, 500), DOUBLEBUF | OPENGL | GL_RGB)
    display.set_caption("DDA - Nisham Ghimire")
    gluOrtho2D(0, 500, 0, 500)

    while True:
        for ev in event.get():
            if ev.type == pg.QUIT:
                pg.quit()
                quit()
        drawDDA()
        display.flip()

main()
```

Output:


DDA - Nisham Ghimire


DDA - Nisham Ghimire


DDA - Nisham Ghimire


DDA - Nisham Ghimire

2. Implement Bresenham Line Drawing algorithm for both slopes(|m|<1 and |m|>=1).
Ans:

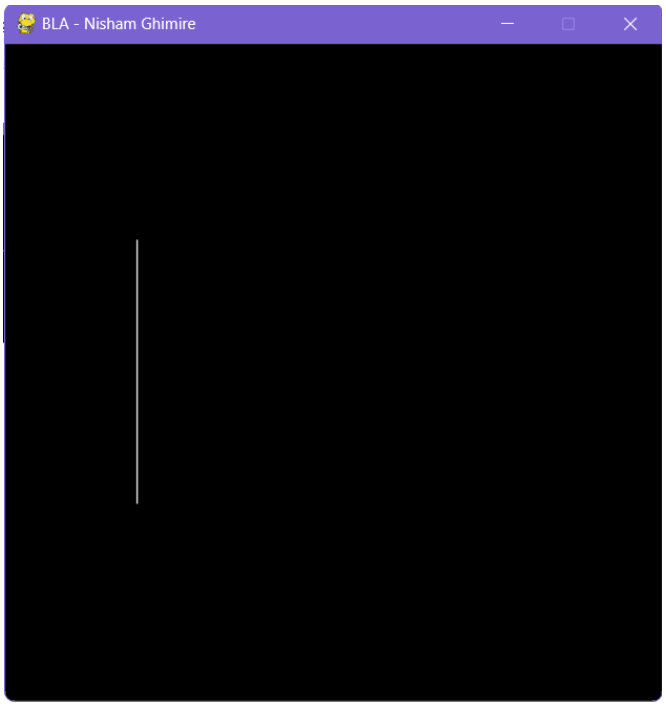
1. **Start Algorithm**
2. **Declare variables:** Declare $x1$, $y1$, $x2$, $y2$, dx, dy, M, p, x, and y.
3. **Input coordinates:** Assign the values of $x1$, $y1$, $x2$, and $y2$.
4. **Calculate slope M:**
   a. If $x2 - x1$ equals 0, set $M = y2 - y1$.
   b. Else, calculate $M = \frac{y2-y1}{x2-x1}$.
5. **Check slope magnitude:**
   a. If $|M| < 1$:
   i. Swap points if $x1 > x2$ to ensure $x1$ is always smaller.
   ii. Calculate $dx = |x2 - x1|$ and $dy = |y2 - y1|$.
   iii. Initialize decision parameter $p = 2 \cdot dy - dx$.
   iv. Initialize x and y to $x1$ and $y1$ respectively.
   v. Draw points using GL_POINTS and the following loop:
   1. While $x \leq x2$:
   a. Plot (x, y) as a point.
   b. Increment x by 1.
   c. If $p \geq 0$:
   i. If $M < 1$, increment y by 1.
   ii. If $M \geq 1$, decrement y by 1.
   iii. Update $p = p + 2 \cdot dy - 2 \cdot dx$.
   d. Else, update $p = p + 2 \cdot dy$.
   b. If $|M| \geq 1$:
   i. Swap points if $y1 > y2$ to ensure $y1$ is always smaller.
   ii. Calculate $dx = |x2 - x1|$ and $dy = |y2 - y1|$.
   iii. Initialize decision parameter $p = 2 \cdot dx - dy$.
   iv. Initialize x and y to $x1$ and $y1$ respectively.
   v. Draw points using GL_POINTS and the following loop:
   1. While $y \leq y2$:
   a. Plot (x, y) as a point.
   b. Increment y by 1.
   c. If $p \geq 0$:
   i. If $M \geq 1$, increment x by 1.
   ii. If $M < 1$, decrement x by 1.
   iii. Update $p = p + 2 \cdot dx - 2 \cdot dy$.
   d. Else, update $p = p + 2 \cdot dx$.
6. **End Algorithm**

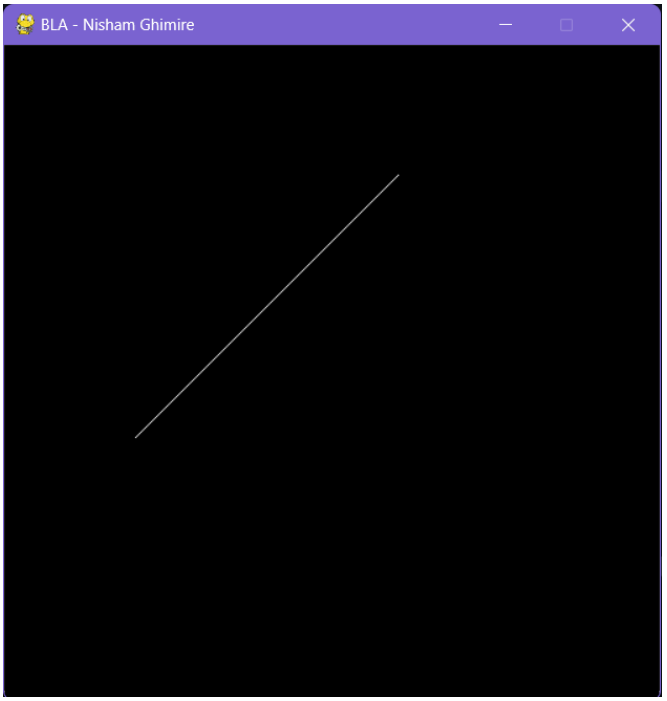
```
import pygame
from pygame import display,event
from OpenGL.GL import *
from OpenGL.GLU import *
import math

def Draw():
    x1, y1, x2, y2 = 100, 200 , 300, 400
    glClear(GL_COLOR_BUFFER_BIT)

    if (x2-x1) == 0:
```

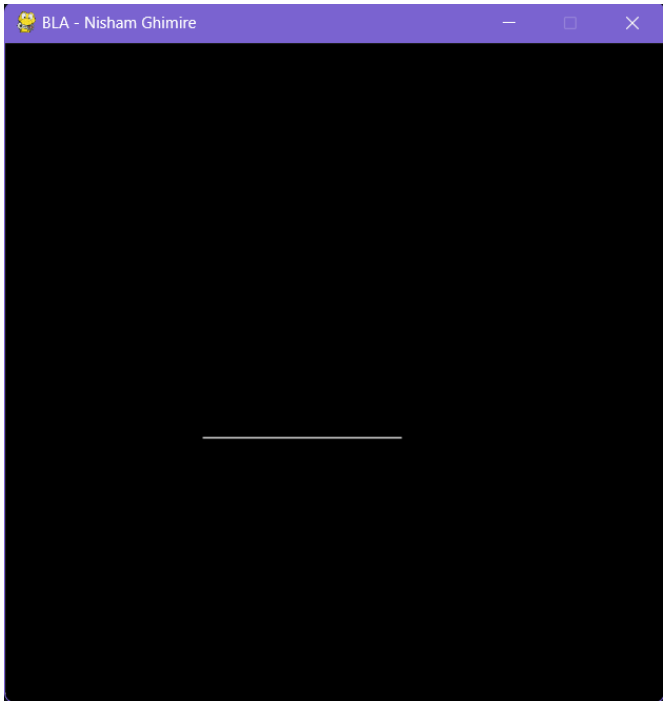
```
        M = (y2-y1)
    else:
        M = (y2-y1)/(x2-x1)

    if abs(M) < 1:
        if x1 > x2:
            t = x1
            x1 = x2
            x2 = t

            t = y1
            y1 = y2
            y2 = t

        dx = abs(x2-x1)
        dy = abs(y2-y1)

        p = 2*dy-dx

        x = x1
        y = y1

        glBegin(GL_POINTS)
        while x <= x2:
            glVertex2f(x, y)
            x = x+1

            if p >= 0:
                if M < 1:
                    y = y+1
                else:
                    y = y-1
                p = p+2*dy-2*dx
            else:
                y = y
                p = p+2*dy
        glEnd()

    if abs(M) >= 1:
        if y1 > y2:
            t = x1
            x1 = x2
            x2 = t

            t = y1
            y1 = y2
            y2 = t

        dx = abs(x2-x1)
        dy = abs(y2-y1)
```

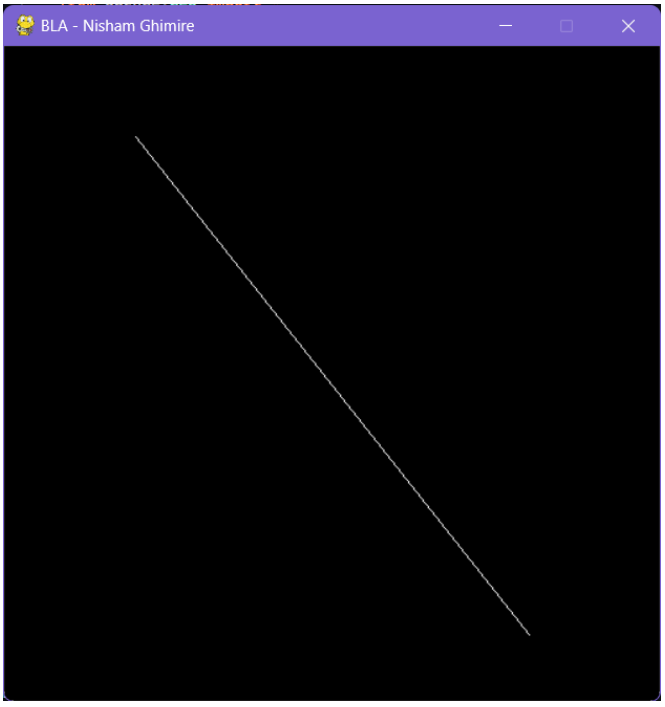
```
        p = 2*dx-dy

        x = x1
        y = y1

        glBegin(GL_POINTS)
        while y <= y2:
            glVertex2f(x, y)
            y = y+1

            if p >= 0:
                if M >= 1:
                    x = x+1
                else:
                    x = x-1
                p = p+2*dx-2*dy
            else:
                x = x
                p = p+2*dx
        glEnd()

    glFlush()

def main():
    pygame.init()
    display.set_mode((500, 500), pygame.DOUBLEBUF | pygame.OPENGL )
    display.set_caption("BLA - Nisham Ghimire")

    gluOrtho2D(0, 500, 0, 500)

    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()

        Draw()

        pygame.display.flip()

if __name__ == '__main__':
    main()
```

Output:









Conclusion:

After completing this lab work, I have learned how to draw lines using DDA and Bresenham algorithms in Python. I utilized Python, OpenGL APIs, and pygame for window creation. This practical experience enhanced my understanding of computer graphics and equipped me with valuable programming skills.