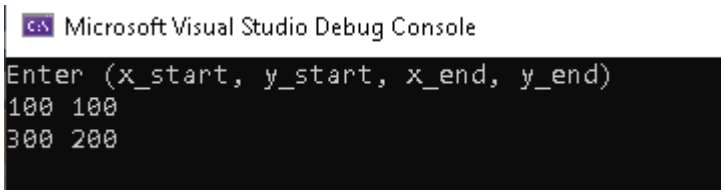
# ASSIGNMENT -2
# COMPUTER GRAPHICS

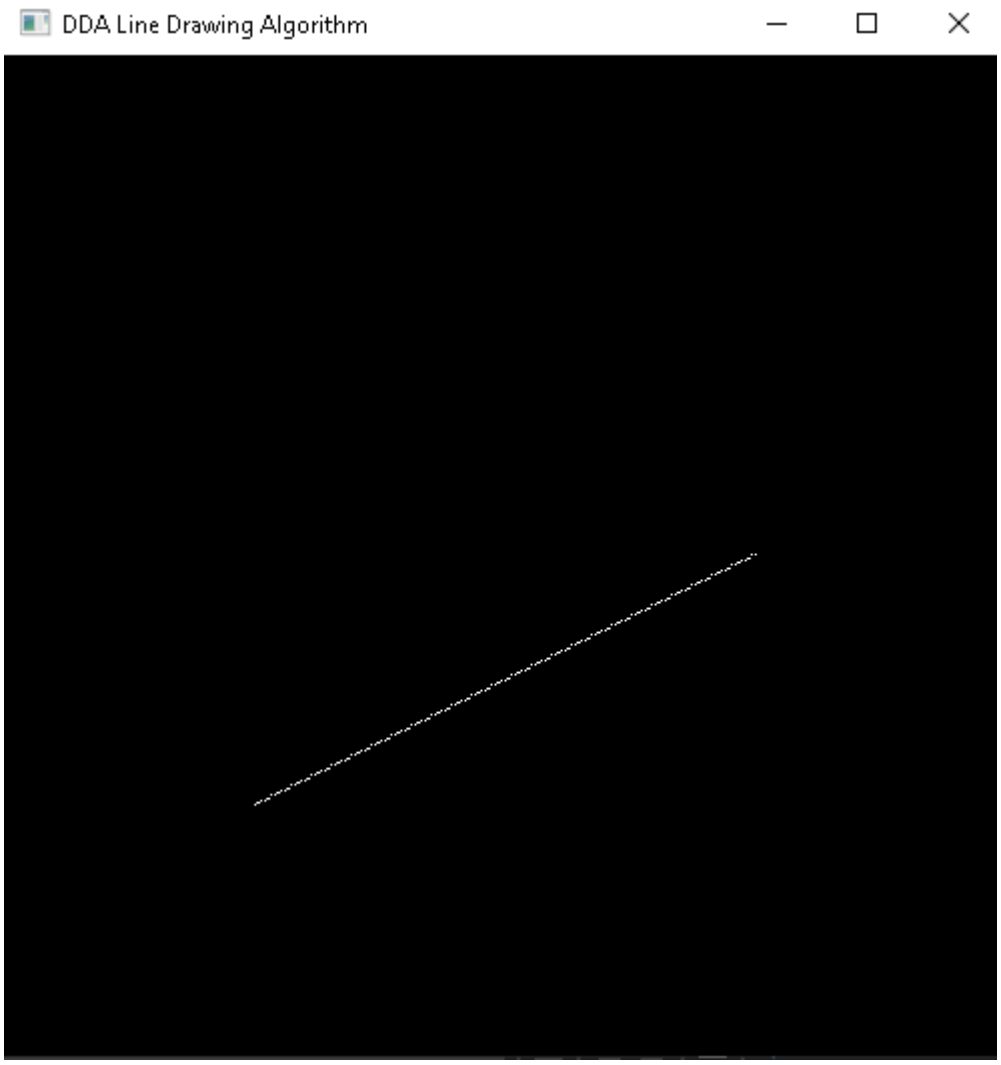
- Shubham Tiwari
101916126
3CS12

## 1.DDA Algorithm

```
#include <GL\glut.h>
#include <iostream>
#include <windows.h>

using namespace std;
int x_start = 10;
int y_start = 20;
int x_end = 200;
int y_end = 100;

void myInit()
{
	glClearColor(1.0, 0.5, 0.0, 1.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0, 400.0, 0.0, 400.0);
}

void draw_pixel(int x, int y)

{
	glBegin(GL_POINTS);
	glVertex2i(x, y);
	glEnd();
}
void lineDDA(int x_start, int y_start, int x_end, int y_end)
{
	int dx = x_end - x_start, dy = y_end - y_start, steps, k;
	float xIncrement, yIncrement, x = x_start, y = y_start;
	if (abs(dx) > abs(dy))
		steps = abs(dx);
	else
		steps = abs(dy);

	xIncrement = dx / (float)steps;
	yIncrement = dy / (float)steps;

	draw_pixel(round(x), round(y));

	for (k = 0; k < steps; k++) {
		x += xIncrement;
		y += yIncrement;
		draw_pixel(round(x), round(y));
	}

}

void myDisplay()
{
```

```
        lineDDA(x_start, y_start, x_end, y_end);
        glFlush();
}

int main(int argc, char** argv)
{
        cout<<"Enter (x_start, y_start, x_end, y_end)\n";
        cin >> x_start >> y_start >> x_end >> y_end;

        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500, 500);

        glutInitWindowPosition(0, 0);

        glutCreateWindow("DDA Line Drawing Algorithm");

        glutDisplayFunc(myDisplay);
        myInit();
        glutMainLoop();

        return 0;
}
```

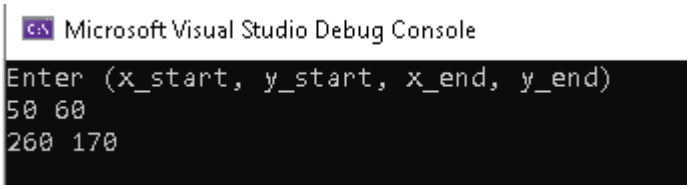


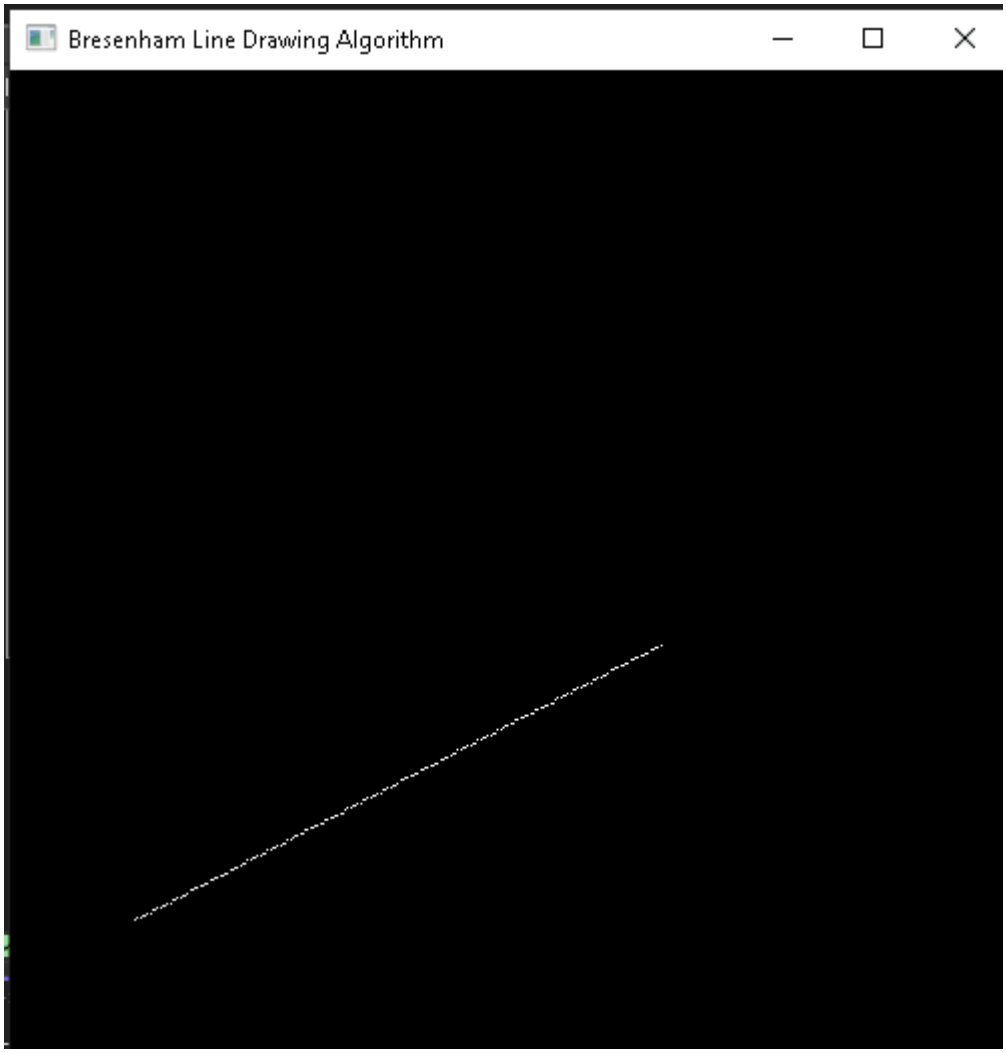

DDA Line Drawing Algorithm

# 2. Bresenham’s line algorithm

```
#include <GL\glut.h>
#include <iostream>
#include <windows.h>

using namespace std;
int x_start = 10;
int y_start = 20;
int x_end = 200;
int y_end = 100;

void myInit()
{
	glClearColor(1.0, 0.5, 0.0, 1.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0, 400.0, 0.0, 400.0);
}

void draw_pixel(int x, int y)

{
	glBegin(GL_POINTS);
	glVertex2i(x, y);
	glEnd();
}
void bresenham(int x_start, int y_start, int x_end,  int y_end)
{
	int dx, dy, i, e;
	int incx, incy, inc1, inc2;
	int x, y;
	dx = x_end - x_start;
	dy = y_end - y_start;
	if (dx < 0)
		dx = -dx;
	if (dy < 0)
		dy = -dy;
	incx = 1;
	if (x_end < x_start)
		incx = -1;
	incy = 1;
	if (y_end < y_start)
		incy = -1;
	x = x_start;
	y = y_start;
	if (dx > dy)
	{
		draw_pixel(x, y);
		e = 2 * dy - dx;
		inc1 = 2 * (dy - dx);
		inc2 = 2 * dy;
		for (i = 0; i < dx; i++)
		{
			if (e >= 0)
			{
				y += incy;
				e += inc1;
			}
			else
				e += inc2;
			x += incx;
```

```
                draw_pixel(x, y);
            }
        }

        else
        {
            draw_pixel(x, y);
            e = 2 * dx - dy;
            inc1 = 2 * (dx - dy);
            inc2 = 2 * dx;
            for (i = 0; i < dy; i++)
            {
                if (e >= 0)
                {
                    x += incx;
                    e += inc1;
                }
                else
                    e += inc2;
                y += incy;
                draw_pixel(x, y);
            }
        }
}


void myDisplay()
{
        bresenham(x_start, y_start, x_end, y_end);
        glFlush();
}

int main(int argc, char** argv)
{
        cout<<"Enter (x_start, y_start, x_end, y_end)\n";
        cin >> x_start >> y_start >> x_end >> y_end;
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500, 500);

        glutInitWindowPosition(0, 0);

        glutCreateWindow("Bresenham's Line Drawing");

        glutDisplayFunc(myDisplay);
        myInit();
        glutMainLoop();

        return 0;
}
```




Bresenham Line Drawing Algorithm