PURBANCHAL UNIVERSITY

KHWOPA ENGINEERING COLLEGE

An Undertaking of Bhaktapur Municipality

Libali-2, Bhaktapur

Nepal

Report on: Computer Graphics

Lab sheet no: 5

SUBMITTED BY:

Name: Sushant Twayana

Roll no: 760345

Group: B

SUBMITTED TO:

Department of Computer Engineering

Khwopa Engineering College

Purbanchal University

Date of Submission: 2079-05-24

Title : Breshenham Line Drawing Algorithm

Objectives

To learn how to digitize line using DDA Algorithm using C/C++ program.

Theory:

This algorithm is used for scan converting a line. It was developed by Bresenham. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations. These operations can be performed very rapidly so lines can be generated quickly.

In this method, next pixel selected is that one who has the least distance from true line.

The method works as follows:

Assume a pixel P1'(x1',y1'),then select subsequent pixels as we work our may to the night, one pixel position at a time in the horizontal direction toward P2'(x2',y2').

Once a pixel is selected .\

The next pixel is

Either the one to its right (lower-bound for the line)

One top its right and up (upper-bound for the line)

The line is best approximated by those pixels that fall the least distance from the path between P1',P2'.

Algorithm

Step1: Start Algorithm

Step2: Declare variable x1,x2,y1,y2,d,i1,i2,dx,dy

Step3: Enter value of x1, y1, x2, y2

Where x1, y1are coordinates of starting point

And x2,y2 are coordinates of Ending point

Step4: Calculate dx = x2-x1

Calculate dy = y2-y1

Calculate i1=2*dy

Calculate i2=2*(dy-dx)

Calculate d=i1-dx

Step5: Consider (x, y) as starting point and x-end as maximum possible value of x.

If dx < 0

Then x = x2

y = y2

x-end=x1

If dx > 0

Then x = x1

y = y1

x-end=x2

Step6: Generate point at (x, y) coordinates.

Step7: Check if whole line is generated.

If x > = x-end

Stop.

Step8: Calculate co-ordinates of the next pixel

If d < 0

Then d = d + i1

If d ≥ 0

Then d = d + i2

Increment y = y + 1

Step9: Increment x = x + 1

Step10: Draw a point of latest (x, y) coordinates

Step11: Go to step 7

Step12: End of Algorithm

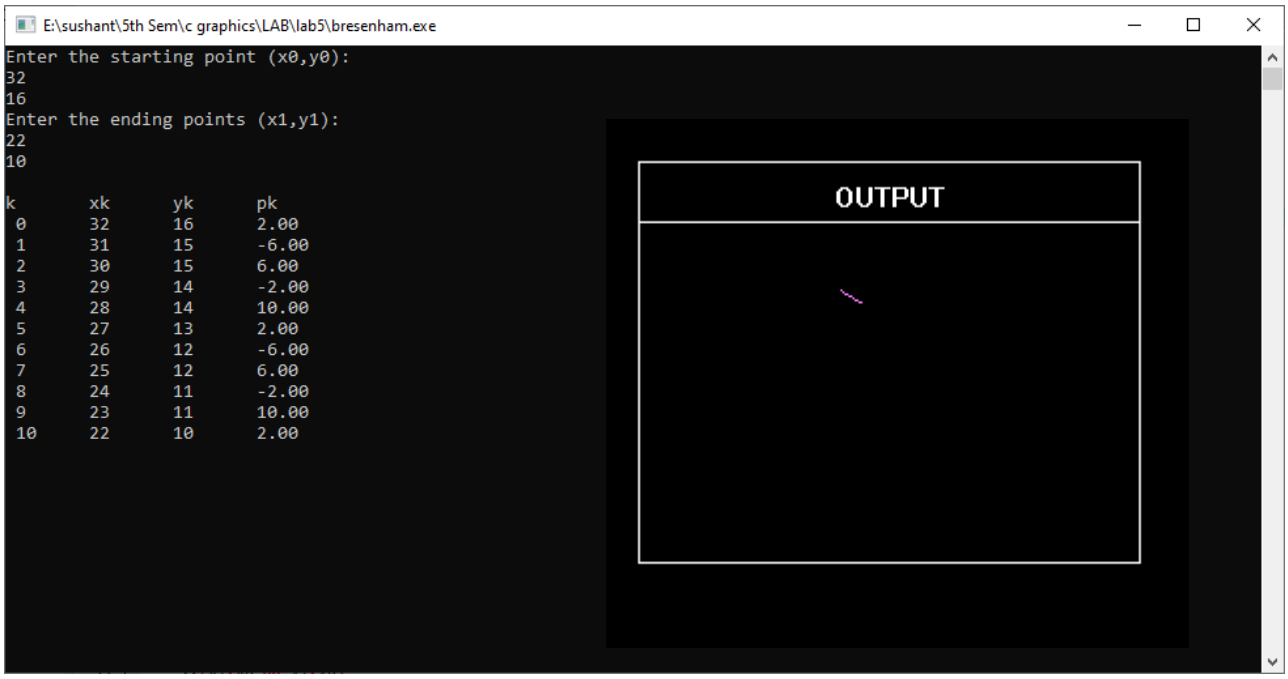
C++ code

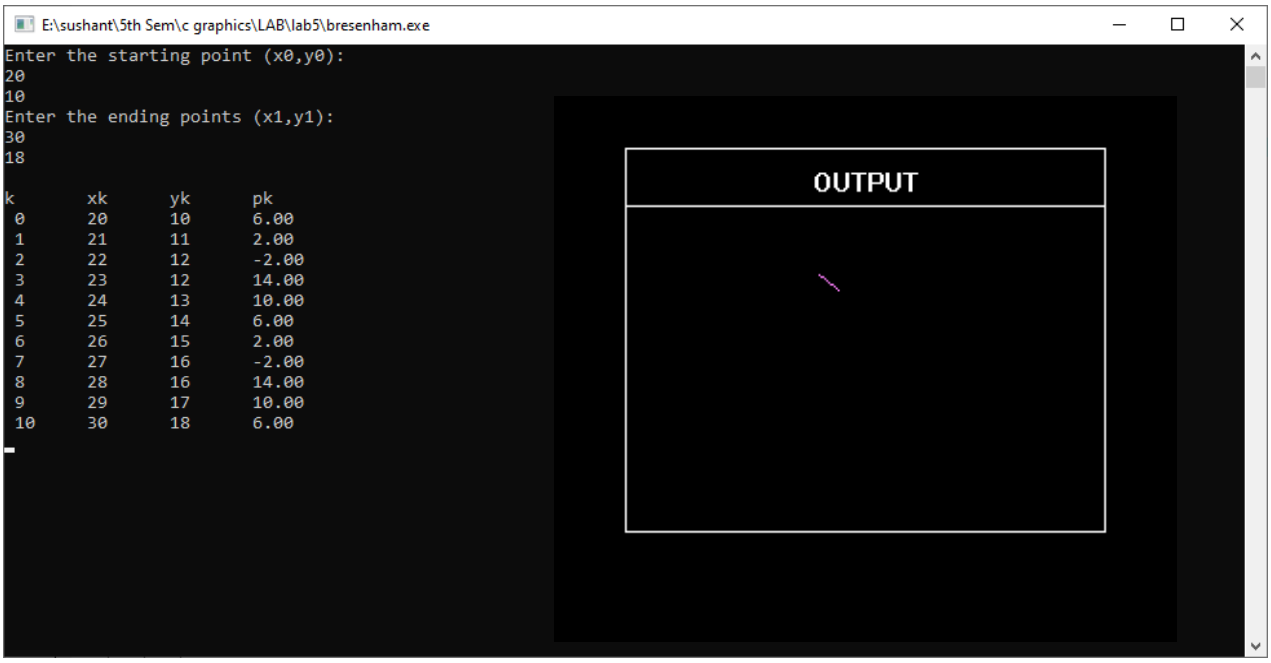
```
#include<iostream>
#include<conio.h>
#include<graphics.h>
#include<time.h>
#include<stdio.h>
#include<math.h>
using namespace std;
int main ()
{
	initwindow(1000,1000);
	float x0,y0,p0,k,pk,pk1,x1,y1,m,dx,dy;
	cout<<"Enter the starting point (x0,y0):\n";
	cin>>x0;
	cin>>y0;
	cout<<"Enter the ending points (x1,y1):\n";
	cin>>x1;
	cin>>y1;
	dx = x1 - x0;
	dy = y1 - y0;
	m = (dy)/(dx);
	putpixel(x0,y0,13);
	printf("\nk\txk\tyk\tpk\n");
	rectangle(150,150,400,350);
```

```
line(150,180,400,180);
outtextxy(248,160,"OUTPUT");
if(x1>x0 && y1>y0)
{
        if(m<1)
                {
                        p0 = 2*dy - dx;
                        pk = p0;
                        for(k=0; k<=dx ; k++)
                        {   printf(" %0.0f\t%0.0f\t%0.0f\t%0.2f\n",k,x0,y0,pk);
                        if (pk < 0)
                                {
                                        x0 = x0 + 1;
                                        putpixel(x0 +230, y0+205,13);
                                        pk = pk + 2*dy;
                                }
                        else
                                {
                                        x0 = x0+1;
                                        y0 = y0+1;
                                        putpixel(x0 +230,y0 +205,13);
                                        pk = pk + 2*dy -2*dx;
                                }
                        }
                }
}
else
{
```

```
                if(m<1)
                {
                        p0 = 2*fabs(dy) - fabs(dx);
                        pk = p0;
                        for (k=0; k<=fabs(dx); k++)
                        {
                        printf(" %0.0f\t%0.0f\t%0.0f\t%0.2f\n",k,x0,y0,pk);
                        if (pk < 0)
                                {
                                        x0 = x0-1;
                                        putpixel(x0 +230, y0+205,13);
                                        pk = pk + 2*fabs(dy);
                                }
                                else    {
                                        x0 = x0-1;
                                        y0 = y0-1;
                                        putpixel(x0 + 230,y0 +205,13);
                                        pk = pk + 2*fabs(dy) -2*fabs(dx);
                                }
                        }
                }
                        else{
                                cout<<"m>1";
                        }
        }
        getch();
        return 0;
}
```

OUTPUT





CONCLUSION

Hence, in this lab we learnt about the Bresenham algorithm, it's uses, advantages over DDA and disadvantages and finally implemented this algorithm to draw a line using C++ in a compiler DEV C++.