Delhi Technological University

Deptt. Of Computer Science

Computer Graphics Lab

(CO-313)

Submitted To:-

Mr. Ram Murti Rawat

(Asstt. Professor, COE Department)

Submitted By:-

Umang Ahuja

2K16/CO/337

3rd Year B2-G1

Index

Sr. No.	Name of Experiment	Dated	Signature
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			
10.			
11.			
12.			
13.			
14.			
15.			

Experiment – 1

Aim

To generate a line using DDA (Digital Differential Analyzer) Algorithm.

Description/Theory

In any 2-Dimensional plane if we connect two points (x0, y0) and (x1, y1), we get a line segment. But in the case of computer graphics we cannot directly join any two coordinate points, for that we should calculate intermediate point's coordinate and put a pixel for each intermediate point, of the desired colour with help of functions like putpixel(x, y, K) in C, where (x,y) is our co-ordinate and K denotes some colour.
For using graphics functions, our system output screen is treated as a coordinate system where the coordinate of the top-left corner is (0, 0) and as we move down our x-ordinate increases and as we move right our y-ordinate increases for any point (x, y).
Now, for generating any line segment we need intermediate points and for calculating them we have can use a basic algorithm called DDA(Digital differential analyzer) line generating algorithm.

Algorithm

The algorithm works as follows:-

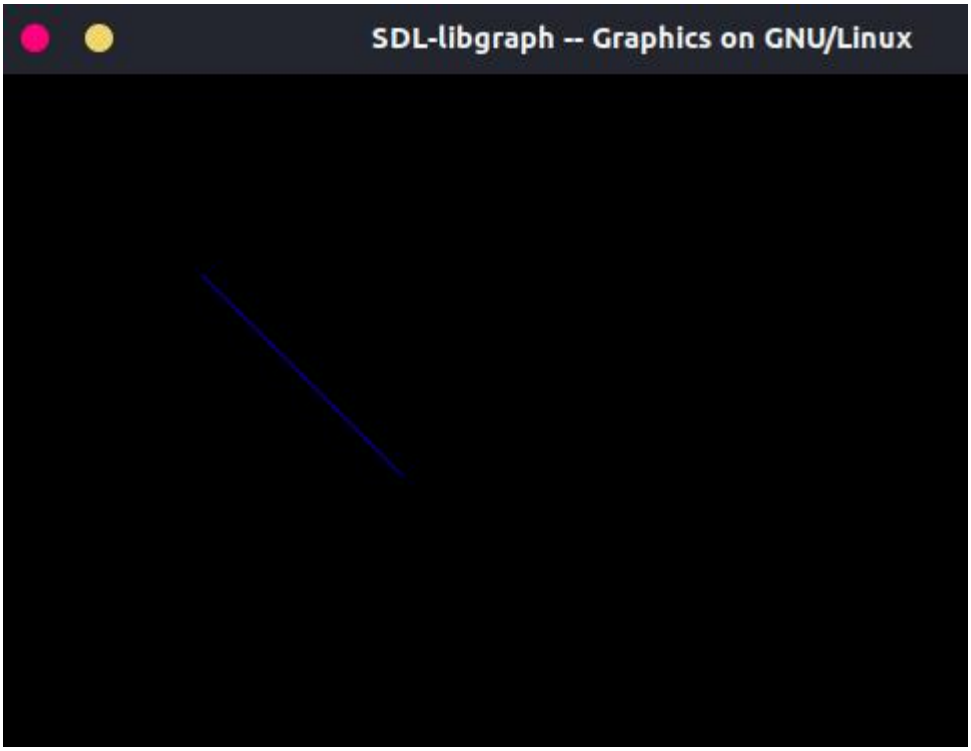
```
dx = X1 - X0;
dy = Y1 - Y0;

Xinc = dx / (float) steps;
Yinc = dy / (float) steps;

X = X0;
Y = Y0;
for (int i = 0; i <= steps; i++)
{
   putpixel (X,Y,WHITE);
   X += Xinc;
   Y += Yinc;
}
```

Code

```
#include <graphics.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>
int main( )
{
   float x,y,x1=100,y1=100,x2=200,y2=200,dx,dy,step;
   int i,gd=DETECT,gm;
   initgraph(&gd,&gm,"");
   dx=abs(x2-x1);
   dy=abs(y2-y1);
   if(dx>=dy)
      step=dx;
   else
      step=dy;
  dx=dx/step;
```

```
   dy=dy/step;
    x=x1;
    y=y1;

    i=1;
    while(i<=step)
    {
        putpixel(x,y,5);
        x=x+dx;
        y=y+dy;
        i=i+1;
        delay(50);
    }
    closegraph();
}
```

Result/output



Discussion

The DDA Algorithm is a faster method for calculating pixel position than the direct use of equation of a straight line i.e. y=mx + c. It eliminates the multiplication in line equation by making use of faster characteristics, so that the appropriate increments are applied in ‘x’ or ‘y’ direction to step to pixel positions along the line path.

Finding and learning

We found out that the accumulation of round-off error in successive additions of the floating point increment, however, can cause the calculated pixel positions to drift away from the true line path for long line segments. Furthermore, the rounding operations and floating point arithmetic in procedure line DDA are still time-consuming. We can improve the performance of DDA Algorithm by separating the increments ‘m’ and ‘1/m’ into integer and fractional parts so that all calculations are reduced to integer operations.

Experiment – 2

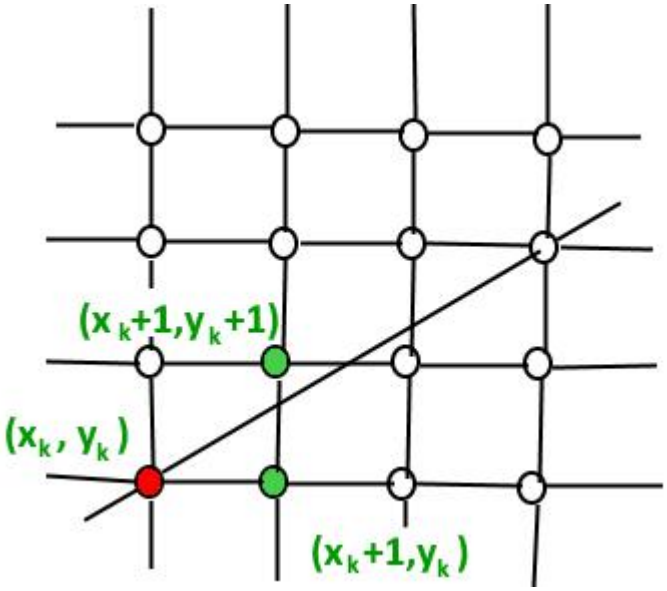
Aim

To generate a line using Bresenham Line Drawing Algorithm.

Dicussion/Theory

Given coordinate of two points A(x1, y1) and B(x2, y2). The task to find all the intermediate points required for drawing line AB on the computer screen of pixels. Note that every pixel has integer coordinates.
The idea of Bresenham's algorithm is to avoid floating point multiplication and addition to compute mx + c, and then computing round value of (mx + c) in every step. In Bresenham's algorithm, we move across the x-axis in unit intervals.

1. We always increase x by 1, and we choose about next y, whether we need to go to y+1 or remain on y. In other words, from any position (X_k, Y_k) we need to choose between (X_k + 1, Y_k) and (X_k + 1, Y_k + 1).



2. We would like to pick the y value (among Y_k + 1 and Y_k) corresponding to a point that is closer to the original line.

We need to a decision parameter to decide whether to pick Yk + 1 or Yk as next point. The idea is to keep track of slope error from previous increment to y. If the slope error becomes greater than 0.5, we know that the line has moved upwards one pixel, and that we must increment our y coordinate and readjust the error to represent the distance from the top of the new pixel – which is done by subtracting one from error.

Algorithm

The algorithm works as follows:-

```
void bresenham(x1, x2, y1, y2)
{
  m_new = 2 * (y2 - y1)
  slope_error_new = [Some Initial Value]
  for (x = x1, y = y1; x <= x2; x++)
  {
    print(x, y);

    // Add slope to increment angle formed
```

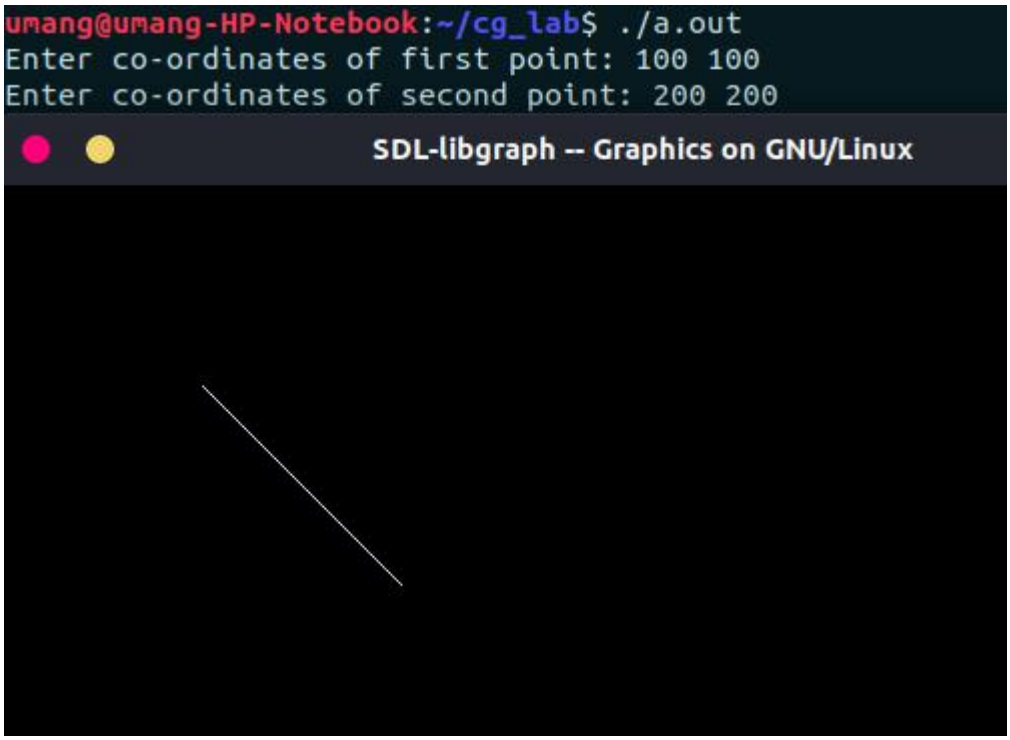
```
    slope_error_new += m_new;

    if (slope_error_new >= 0)
    {
      y++;
      slope_error_new  -= 2 * (x2 - x1);
    }
  }
}
```

Code

```
#include<stdio.h>
#include<graphics.h>
#include <conio.h>

void drawline(int x0, int y0, int x1, int y1)
{
   int dx, dy, p, x, y;

   dx=x1-x0;
   dy=y1-y0;

   x=x0;
   y=y0;

   p=2*dy-dx;

   while(x<x1)
   {
      if(p>=0)
      {
         putpixel(x,y,7);
         delay(20);
         y=y+1;
         p=p+2*dy-2*dx;
      }
      else
      {
         putpixel(x,y,7);
         delay(20);
         p=p+2*dy;
      }
      x=x+1;
   }
}

int main()
{
   int gdriver=DETECT, gmode, error, x0, y0, x1, y1;
   printf("Enter co-ordinates of first point: ");
   scanf("%d%d", &x0, &y0);

   printf("Enter co-ordinates of second point: ");
   scanf("%d%d", &x1, &y1);
```

```
    initgraph(&gdriver, &gmode, "");
    drawline(x0, y0, x1, y1);
          getch();

    return 0;
}
```

Result



Discussion

Bresenham's Line Drawing Algorithm is a highly efficient incremental over Digital Differential Analyzer (DDA).
It produces mathematically accurate results using only integer addition, subtraction and multiplication by 2, which can be accomplished by a simple arithmetic shift operation.

Findings and Learning

Bresenham's Line Drawing Algorithm is more accurate method to produce a line, with given end points, than other line algorithms.
It uses mid-point approach to find the subsequent pixels. Due to this, other curves, such as circle or ellipse, also follow the same approach.