Experiment-2

Student Name: Jatin Mittal
UID: 22BCS15309
Branch: BE-CSE
Section/Group: 605/A
Semester: 6
Date of Performance: 23/01/25
Subject Name: Computer Graphics Lab
Subject Code: 22CSH-352

1. Aim:

Implement and compare the performance of Simple DDA, Symmetrical DDA, and Bresenham's algorithm for positive and negative line slope.

2. Objective:

The objective of this practical is to implement and compare the performance of Simple DDA, Symmetrical DDA, and Bresenham's line-drawing algorithms for lines with both positive and negative slopes. The comparison focuses on computational efficiency, accuracy, and their ability to render lines on a raster display.

3. Algorithm:

I. Initialize Graphics: Start graphics mode using initgraph() and set the background to white.
II. Draw Line Using Simple DDA.
III. Draw Line using Symmetrical DDA.
IV. Draw Line Using Bresenham's Line Algorithm.
V. Wait and Exit: Pause with getch() and close graphics using closegraph().

4. Implementation/Code:

SIMPLE DDA:

```
#include<iostream.h>
#include<dos.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
#define round(a) ((int)(a+0.5))
void dda_line(int x1,int y1,int x2,int y2)
{
        int dx=(x2-x1);
        int dy=(y2-y1);
        int length;
        if(abs(dy)>abs(dx))
            length=abs(dy);
```

```
            else
                length=abs(dx);
            float xinc,yinc,x=x1,y=y1;
            xinc=dx/(float)length;
            yinc=dy/(float)length;
        putpixel(round(x),round(y),15);
        for(int k=1;k<=length;k++)
        {
          x=x+xinc;
          y=y+yinc;
          putpixel(round(x),round(y),15);
          delay(50);
        }
}
void main()
{
        clrscr();
        int x1,x2,y1,y2;
        int gd=DETECT,gm;
        cout<<"Enter the x1 : ";
        cin>>x1;
        cout<<"Enter the y1 : ";
        cin>>y1;
        cout<<endl;
        cout<<"Enter the x2 : ";
        cin>>x2;
        cout<<"Enter the y2 : ";
        cin>>y2;
        getch();
        initgraph(&gd,&gm,"c:\\turboc3\\bgi");
        dda_line(x1,y1,x2,y2);
        setcolor(7);
        outtextxy(150,150,"JATAIN MITTAL");
        outtextxy(200,250,"22BCS15309");
        getch();
        closegraph();
}
```

SYMMETRICAL DDA:

```
#include<conio.h>
#include<iostream.h>
#include<graphics.h>
#include<dos.h>
#include<math.h>
#define ROUND(a)((int)(a+0.5))
void symDDA(int xa,int ya,int xb,int yb)
{
   int dx=xb-xa,dy=yb-ya;
   float length;
   float xinc,yinc,x=xa,y=ya;
   if(abs(dx)>abs(dy))
         length=abs(dx);
```

```
    else
            length=abs(dy);
   float n=log10(length)/log10(2);
   xinc=dx/(pow(2,n));
   yinc=dy/(pow(2,n));
   putpixel(ROUND(x),ROUNDYes,15);
   delay(50);
   for(int i=0;i<length;i++)
   {
    x=x+xinc;
    y=y+yinc;
    putpixel(ROUND(x),ROUNDYes,15);
    delay(50);
   }
}
void main()
{
   int gd=DETECT,gm;
   initgraph(&gd,&gm,"");
   int xa,xb,ya,yb;
   cout<<"enter the points";
   cin>>xa>>xb>>ya>>yb;
   cleardevice();
   symDDA(xa,xb,ya,yb);
   getch();
   closegraph();
        }
```

BRESENHAM'S LINE-DRAWING ALGORITHM

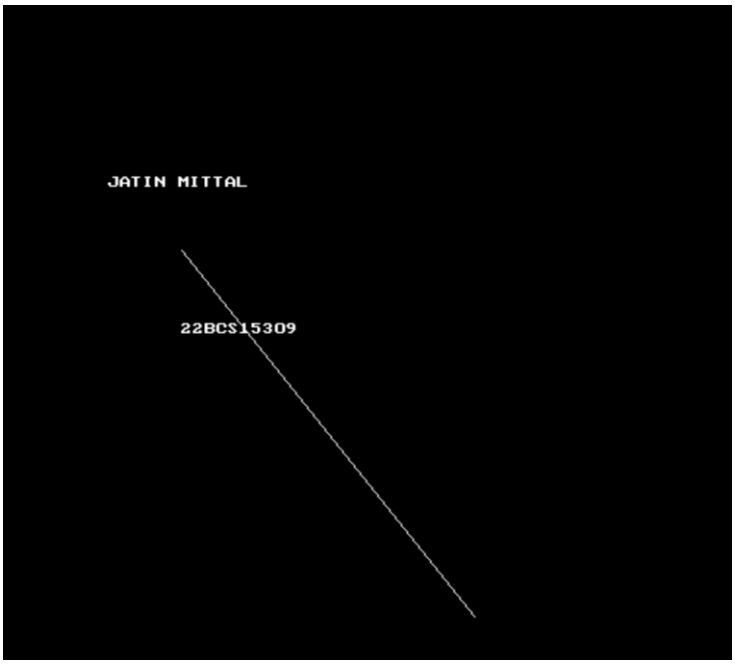
```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
int sign(int x)
{
            if(x<0)
            return(-1);
            if(x>0)
            return(1);
            else
            return(0);
}
 void lineBres(int xa,int ya,int xb,int yb)
{
            int sx,sy,t,length,flag;
            int x=xa;
            int y=ya;
            int dx=abs(xa-xb),dy=abs(ya-yb);
            sx=sign(xb-xa);
            sy=sign(yb-ya);
            if(dy>dx)
```

```
        {
            t=dx;
            dx=dy;
            dy=t;
            length=dy;
            flag=1;
        }
        else
        {
            length=dx;
            flag=0;
        }
            int p=(2*dy)-dx;
            int twoDx=2*dx,twoDy=2*dy;
            putpixel(x,y,15);
            delay(50);
        for(int i=0;i<length;i++)
        {
            while(p>0)
            {
                    if(flag==1)
                    x=x+sx;
                    else
                    y=y+sy;
                    p=p-twoDx;
            }
            if(flag==1)
            y=y+sy;
            else
            {
            x=x+sx;
            p=p+twoDy;
            putpixel(x,y,15);
            delay(50);
            }
        }
}
void main()
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"c://turboc3//bgi");
        int xa,ya,xb,yb;
        cout<<"Enter the starting point of x :";
        cin>>xa;
        cout<<"Enter the starting point of y :";
        cin>>ya;
        cout<<"Enter the ending point of x :";
        cin>>xb;
        cout<<"Enter the ending point of x :";
        cin>>yb;
        cleardevice();
        lineBres(xa,ya,xb,yb);
        getch();
        closegraph();
```

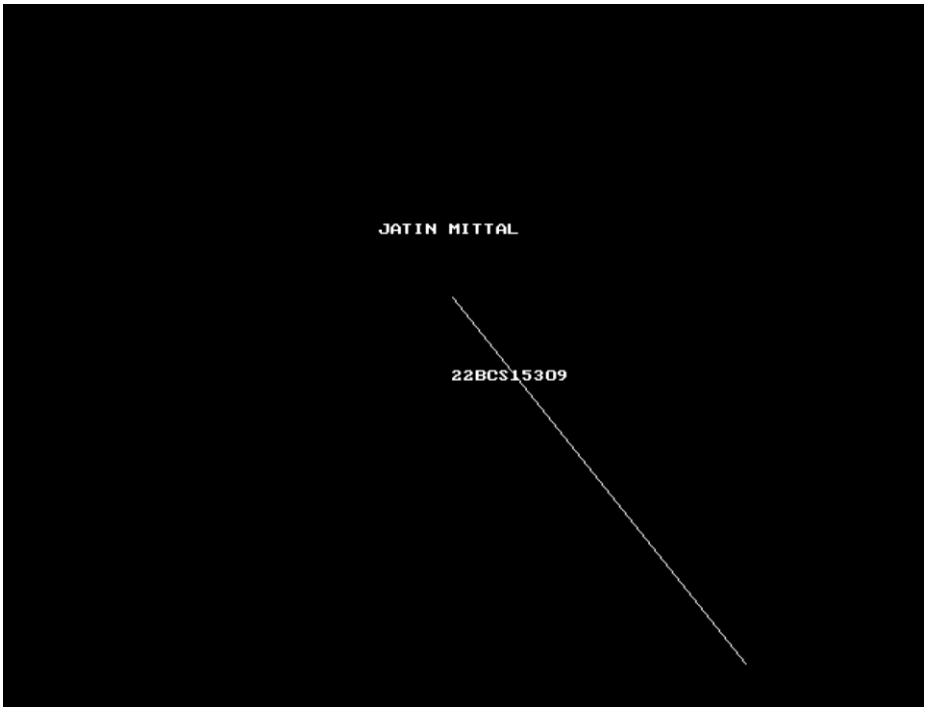
```
}
```

5. Output

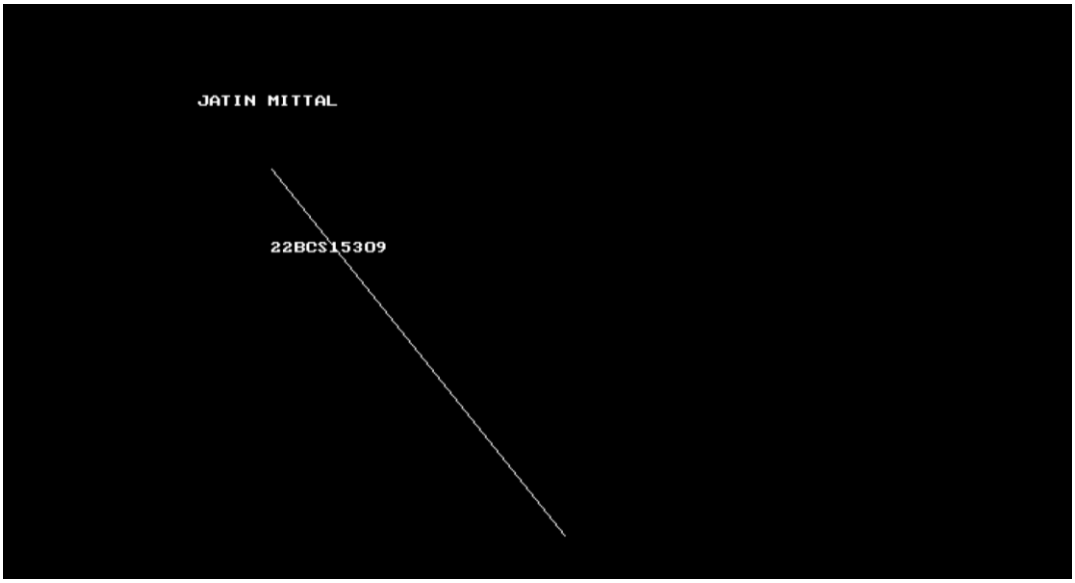
SIMPLE DDA:



SYMMETRICAL DDA:



BRESENHAM'S ALGORITHM:



6. Learning Outcome:

i. Understand basic graphics programming using the graphics.h library.
ii. Understanding Line-Drawing Algorithms
iii. Display text on the graphics screen using the outtextxy() function.
iv. Initialize graphics mode and set the background color in a program.
v. Handle user input and control program flow using getch().