Experiment 2

Student Name: Pranjal Singh **UID: 22BCS13041**
Branch: BE/CSE **Section/Group: 22BCS_FL_601/A**
Semester: 6^{TH} **Date of Performance: 22/01/25**
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:

1. Simple DDA Algorithm:

- Input: (x1,y1)(x1,y1), (x2,y2)(x2,y2).
- Calculate slope m=y2−y1x2−x1m=x2−x1y2−y1.
- If |m|≤1|m|≤1, increment xx and compute y=y+my=y+m.
  Else, increment yy and compute x=x+1mx=x+m1.
- Plot the points.

2. Symmetrical DDA Algorithm:

- Input: (x1,y1)(x1,y1), (x2,y2)(x2,y2).
- Compute dx=x2−x1dx=x2−x1, dy=y2−y1dy=y2−y1.
- Set steps L=max(|dx|,|dy|)L=max(|dx|,|dy|).
- Increment by Δx=dxLΔx=Ldx, Δy=dyLΔy=Ldy.
- Plot (round(x),round(y))(round(x),round(y)).

3. Bresenham's Algorithm:

- Input: (x1,y1)(x1,y1), (x2,y2)(x2,y2).
- Compute dx=x2−x1dx=x2−x1, dy=y2−y1dy=y2−y1.
- Initialize p=2dy−dxp=2dy−dx.
- For each xx, if p<0p<0, adjust pp; else, increment yy and update pp.
- Plot (x,y)(x,y).

4. Implementation:

- **Simple DDA Algorithm**

```
#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);
// Pranjal Singh
               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(100);
               }
}
void main()
{
               clrscr();
               int x1,x2,y1,y2;
               int gd=DETECT,gm;
               cout<<"Enter the x-coordinate of starting point : ";
               cin>>x1;
               cout<<"Enter the y-coordinate of ending point : ";
               cin>>y1;
               cout<<endl;
               cout<<"Enter the x-coordinate of starting point : ";
               cin>>x2;
               cout<<"Enter the y-coordinate of ending point : ";
               cin>>y2;
               getch();
               initgraph(&gd,&gm,"c:\\turboc3\\bgi");
```

```
                dda_line(x1,y1,x2,y2);
                setcolor(4);
                getch();
                closegraph();
}
```

- **Symmetric DDA Algorithm**

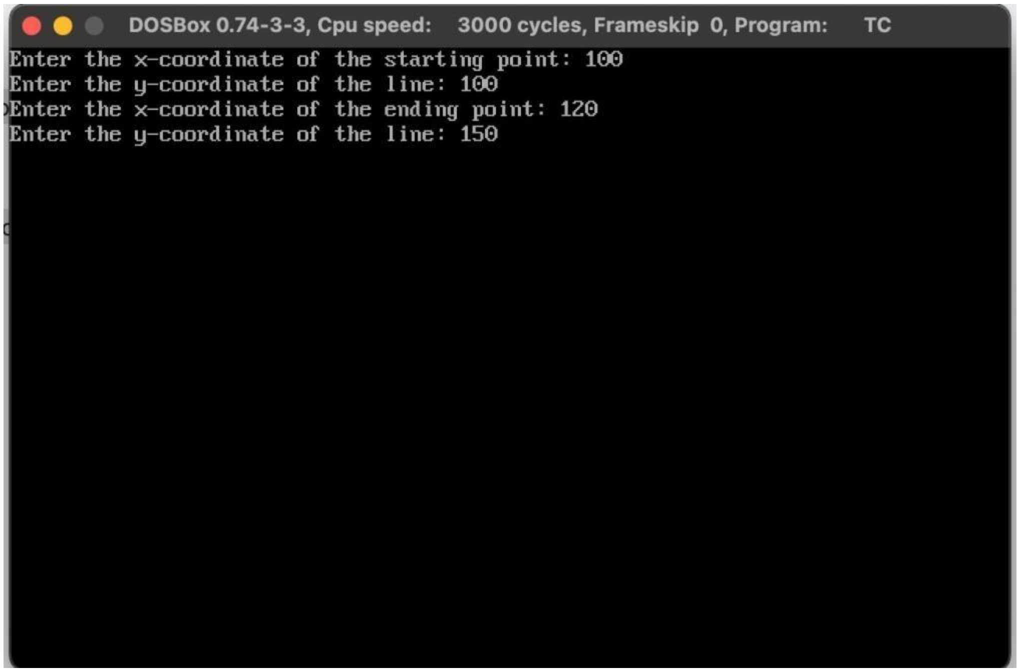
```
#include<conio.h>
#include<iostream.h>
#include<graphics.h> #include<dos.h>
#include<math.h>
#define ROUND(a)((int)(a+0.5))
void symDDA(int xa,int ya,int xb,int yb)
{
    int dx=xb-xa,dy=yb-ya;float length; float
    xinc,yinc,x=xa,y=ya; if(abs(dx)>abs(dy))
            length=abs(dx);
    else
            length=abs(dy);
   float n=log10(length)/log10(2);
   xinc=dx/(pow(2,n));
   yinc=dy/(pow(2,n));
   putpixel(ROUND(x),ROUND(y),15); delay(50);
   for(int i=0;i<length;i++)
   {
    x=x+xinc; y=y+yinc;
    putpixel(ROUND(x),ROUND(y),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 DDA 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
{
```

```
            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. Result:


DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TC

```
Enter the x-coordinate of the starting point: 100
Enter the y-coordinate of the line: 100
Enter the x-coordinate of the ending point: 120
Enter the y-coordinate of the line: 150
```

FIG 1.1 (Simple DDA coordinates)

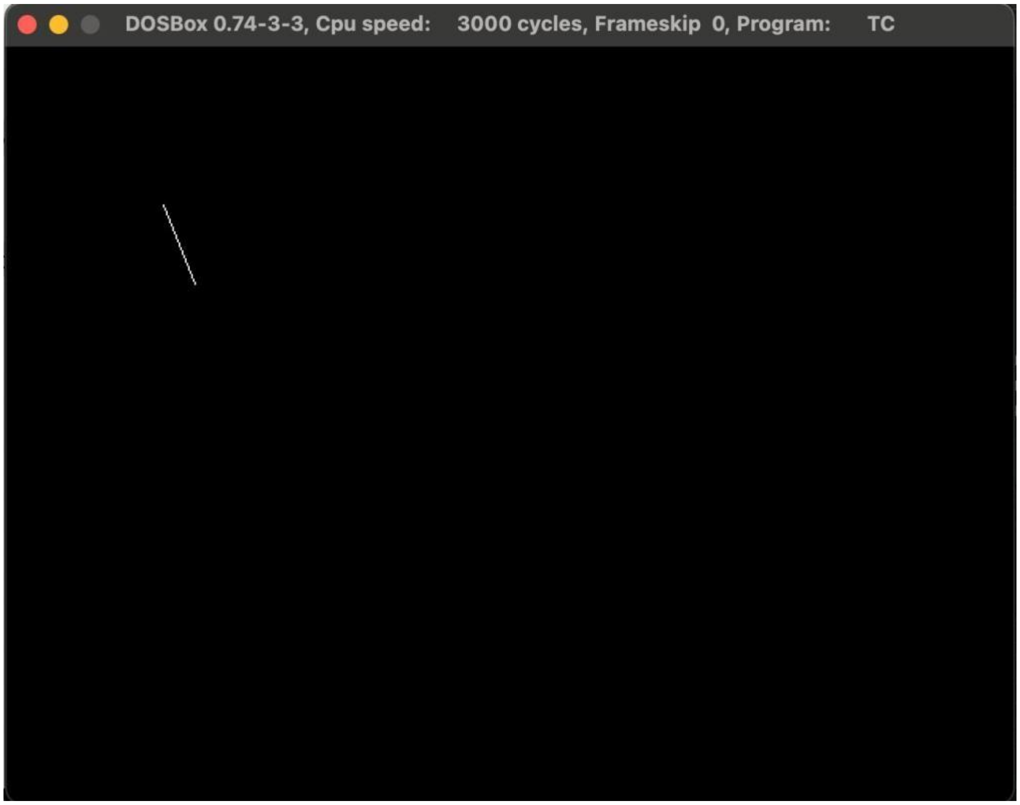
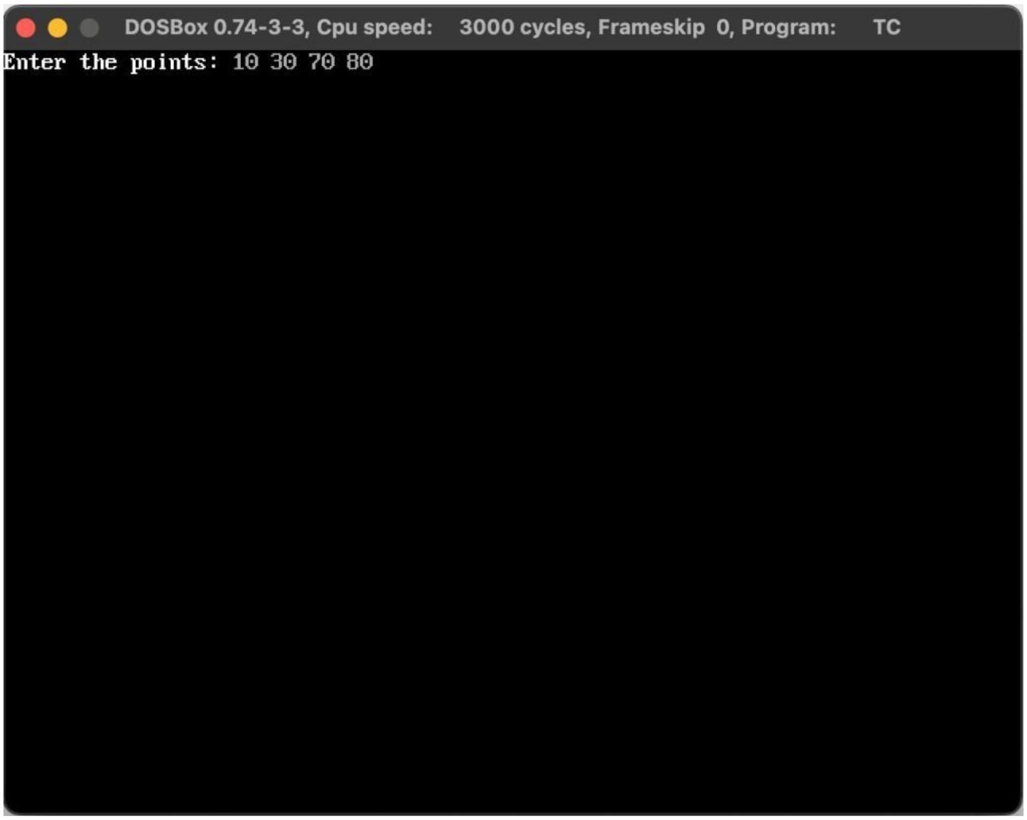

DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TC

FIG 1.2 (Simple DDA algorithm)


DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TC

```
Enter the points: 10 30 70 80
```

FIG 1.3 (Symmetric DDA coordinates)

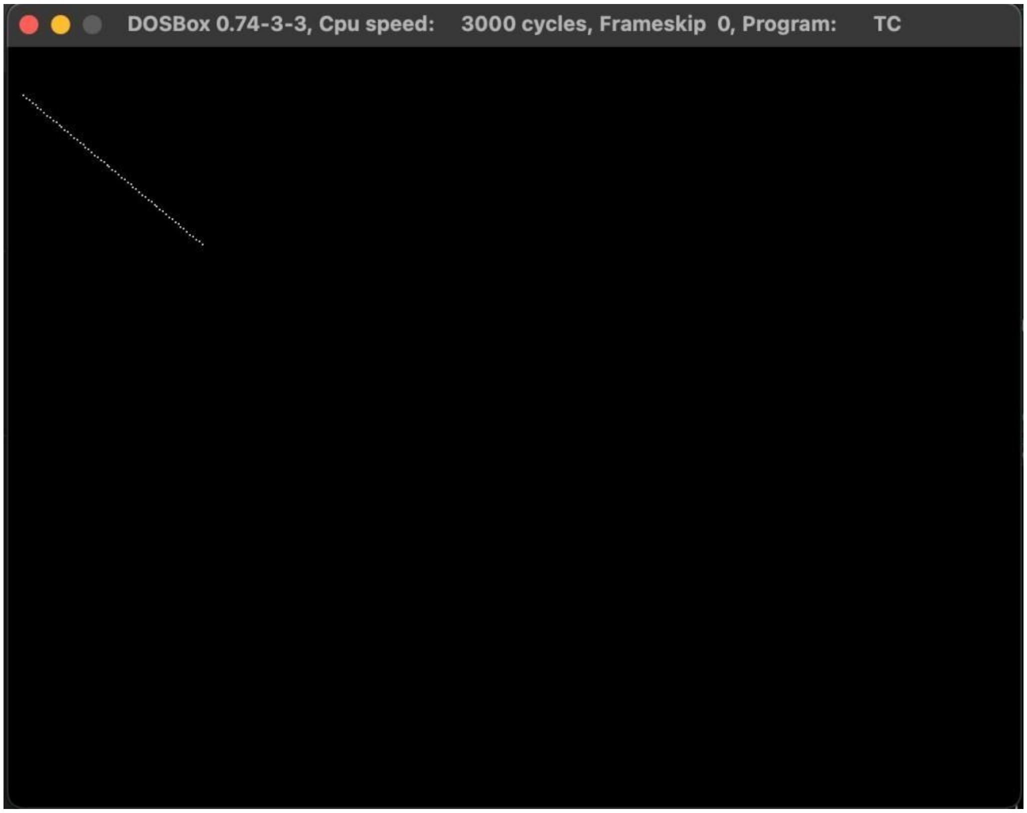
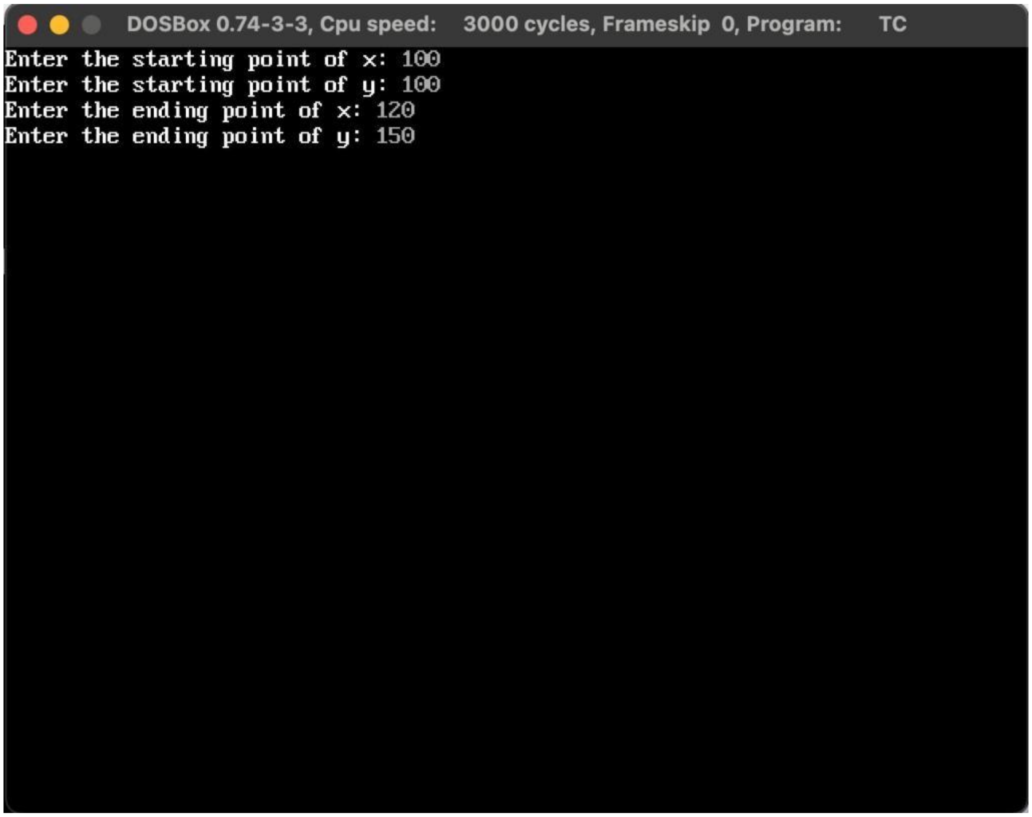

DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TC

FIG 1.4 (Symmetric DDA algorithm)


DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TC

```
Enter the starting point of x: 100
Enter the starting point of y: 100
Enter the ending point of x: 120
Enter the ending point of y: 150
```

FIG 1.5 (Bresenham's DDA coordinates)

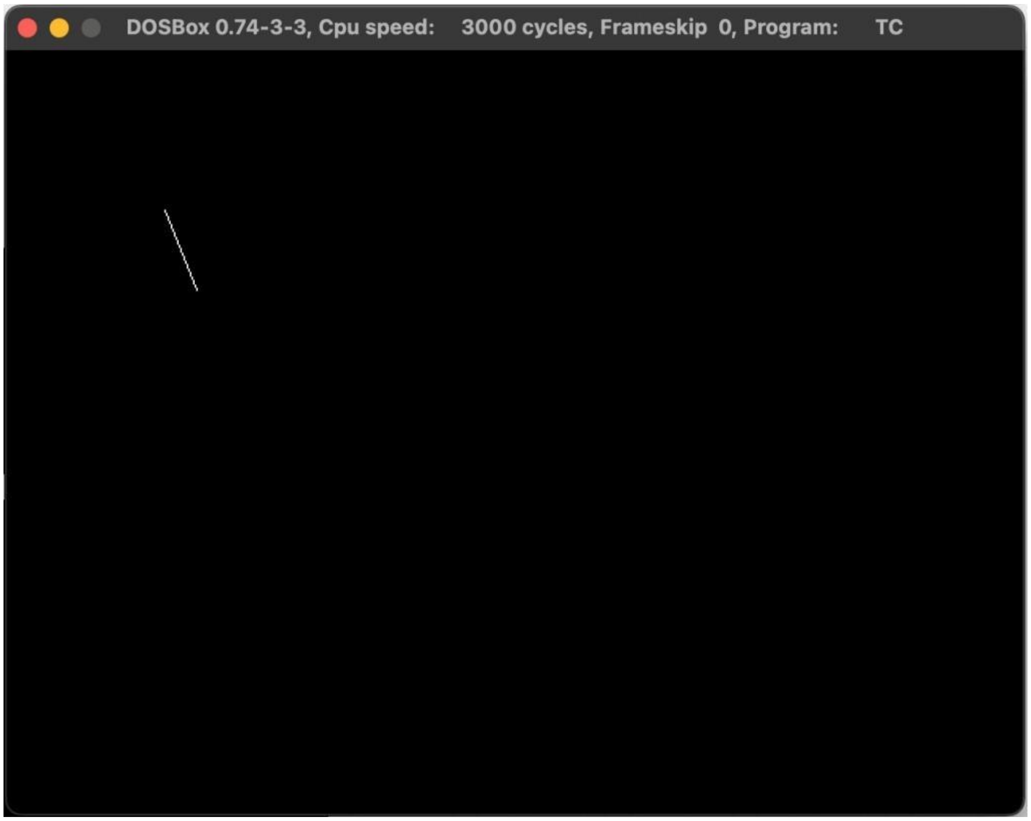

DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TC

FIG 1.6 (Bresenham's DDA algorithm)

6. Learning Outcomes:

- Understand Simple DDA, Symmetrical DDA, and Bresenham’s algorithms.
- Compare efficiency and precision (floating-point vs. integer).
- Learn how line slope impacts drawing.
- Recognize optimization in line drawing algorithms.