Bresenham's Algorithm

Write a C program to draw a line using Bresenham's algorithm.

Syeda Reeha Quasar
14114802719
Group - 3C7

EXPERIMENT – 3

AIM:

Write a C program to draw a line using Bresenham's algorithm.

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.

Advantage:

1. It involves only integer arithmetic, so it is simple.

2. It avoids the generation of duplicate points.

3. It can be implemented using hardware because it does not use multiplication and division.

4. It is faster as compared to DDA (Digital Differential Analyzer) because it does not involve floating point calculations like DDA Algorithm.

Disadvantage:

1. This algorithm is meant for basic line drawing only Initializing is not a part of Bresenham's line algorithm. So to draw smooth lines, you should want to look into a different algorithm.

Bresenham's Line Algorithm:

Step1: Start Algorithm

Step2: Declare variable x_1,x_2,y_1,y_2,d,i_1,i_2,dx,dy

Step3: Enter value of x_1,y_1,x_2,y_2
Where x_1,$y_1$are coordinates of starting point
And x_2,y_2 are coordinates of Ending point

Step4: Calculate dx = x_2-x_1
Calculate dy = y_2-y_1
Calculate i_1=2*dy
Calculate i_2=2*(dy-dx)
Calculate d=i_1-dx

Step5: Consider (x, y) as starting point and x_{end}as maximum possible value of x.
If dx < 0
Then x = x_2
y = y_2
x_{end}=x_1
If dx > 0

Then $x = x_1$

$y = y_1$

$x_{end}=x_2$

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 $d = d + i_1$

If $d \geq 0$

Then $d = d + i_2$

Increment $y = y + 1$

Step9: Increment $x = x + 1$

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

Step11: Go to step 7

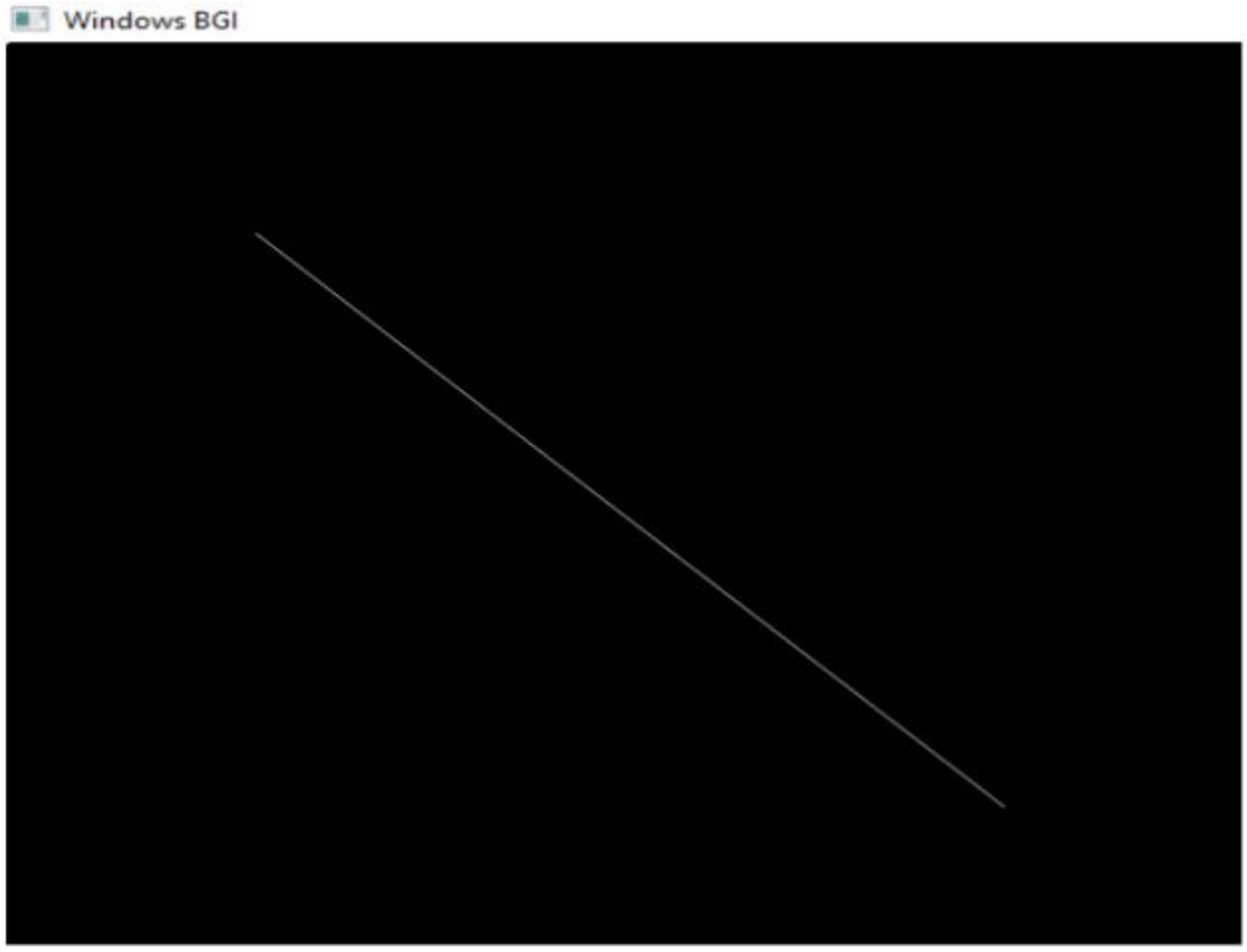
Step12: End of Algorithm

Drawing Line using Bresenham's Algorithm

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);
                          y=y+1;
```

```
                p=p+2*dy-2*dx;
            }
            else
            {       putpixel(x,y,7);
                        p=p+2*dy;
            }
            x=x+1;
        }
}
int main()
{
    initwindow(800,800);
    drawline(100,100,400,400);
    getch();
}
```



Source Code:

```
#include <stdio.h>

#include <conio.h>

#include <graphics.h>


bresenham(int x0, int y0, int x1, int y1, int clr)

{

  float dx, dy, p, x, y, t, m;


  if ((x1 - x0) == 0)

  {

        m = y1 - y0;

  }

  else{

        m = (y1 - y0) / (x1 - x0);

  }

if (abs(m) < 1)

  {


                if (x0 > x1)

                {

                        t = x0;

                        x0 = x1;

                        x1 = t;
```

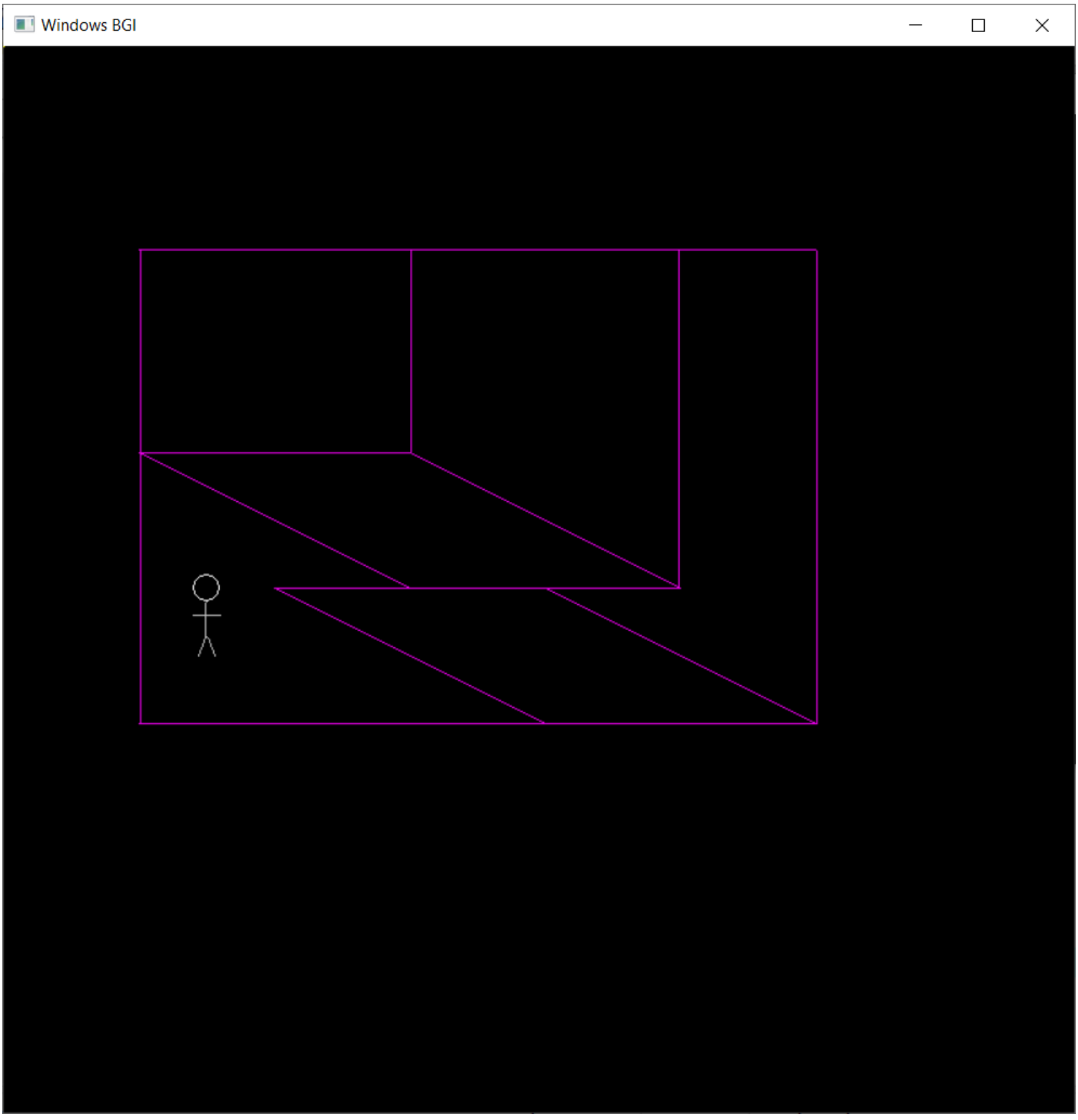
```
        t = y0;
        y0 = y1;
        y1 = t;
}
dx = abs(x1-x0);
dy = abs(y1-y0);

x = x0;
y = y0;
p = (2*dy)-dx;

while(x<=x1)
{
        putpixel(x, y, clr);
        x = x + 1;
        if (p >= 1)
        {
                if (m < 1)
                {
                        y = y + 1;
                }
                else{
                        y = y - 1;
                }
                p = p + (2 * dy) - (2 * dx);
        }
        else
```

```
                {
                        p = p + (2*dy);
                }
        }
    }
if (abs(m) > 1)
  {
        if (y0 > y1)
        {
                t = x0;
                x0 = x1;
                x1 = t;

                t = y0;
                y0 = y1;
                y1 = t;
        }
        dx = abs(x1-x0);
        dy = abs(y1-y0);

        x = x0;
        y = y0;

        p = (2*dy)-dx;

        while(y<=y1)
        {
```

```
                putpixel(x, y, clr);
                y = y + 1;
                if (p >= 1)
                {
                        if (m < 1)
                        {
                                x = x + 1;
                        }
                        else{
                                x = x - 1;
                        }
                        p = p + (2 * dx) - (2 * dy);
                }
                else
                {
                        p = p + (2*dx);
                }
        }
    }
}
main()
{
        //pattern
        initwindow(800, 800);
        // diamond
        bresenham(200, 400, 400, 400, 5);
        bresenham(200, 400, 400, 500, 5);
```

```
bresenham(400, 400, 600, 500, 5);
bresenham(100, 500, 600, 500, 5);
// diamond
bresenham(100, 300, 300, 300, 5);
bresenham(100, 300, 300, 400, 5);
bresenham(300, 300, 500, 400, 5);
bresenham(300, 400, 500, 400, 5);
// joints
bresenham(100, 150, 250, 150, 5);
bresenham(100, 500, 100, 150, 5);
bresenham(600, 500, 600, 150, 5);
bresenham(400, 150, 600, 150, 5);
bresenham(250, 150, 500, 150, 5);
bresenham(500, 150, 500, 400, 5);

bresenham(300, 300, 300, 150, 5);
//man
circle(150,400,10);
bresenham(150,410,150,436, 7);
bresenham(150,436,145,450, 7);
bresenham(150,436,155,450, 7);
bresenham(150,436,155,450, 7);
bresenham(140,420,160,420, 7);
bresenham(160,420,170,402, 7);
getch();
closegraph();}
```

OUTPUT



VIVA QUESTIONS:

Q1. What is the difference between DDA algorithm and Bresenham's Algorithm?

Ans.

DDA Algorithm	Bresenham's Line Algorithm
1. DDA Algorithm use floating point, i.e., Real Arithmetic.	1. Bresenham's Line Algorithm use fixed point, i.e., Integer Arithmetic
2. DDA Algorithms uses multiplication & division its operation	2.Bresenham's Line Algorithm uses only subtraction and addition its operation
3. DDA Algorithm is slowly than Bresenham's Line Algorithm in line drawing because it uses real arithmetic (Floating Point operation)	3. Bresenham's Algorithm is faster than DDA Algorithm in line because it involves only addition & subtraction in its calculation and uses only integer arithmetic.
4. DDA Algorithm is not accurate and efficient as Bresenham's Line Algorithm.	4. Bresenham's Line Algorithm is more accurate and efficient at DDA Algorithm.
5.DDA Algorithm can draw circle and curves but are not accurate as Bresenham's Line Algorithm	5. Bresenham's Line Algorithm can draw circle and curves with more accurate than DDA Algorithm.

Q2. What is c:\\tc\\bgi?

Ans. It specifies the directory path where initgraph looks for graphics drivers (*. BGI) first. If files are not there then initgraph will look for the current directory of your program.

Q3. Define closegraph()?

Ans. The header file graphics.h contains closegraph() function which closes the graphics mode, deallocates all memory allocated by graphics system and restores the screen to the mode it was in before you called initgraph.

Syntax : void closegraph();

Q4. What is DETECT graphics driver?

Ans. DETECT is an enumeration type that identifies the proper graphics driver. It tells the compiler that what graphics driver to use or to automatically detect the drive. If graphic driver is set to DETECT ,

then initgraph sets graphic mode to the highest resolution available for the detected driver. Use - intgd = DETECT, gm;

Q5. In Bresenham's line algorithm, if the distance d1 < d2 then decision parameter Pk is _____

a) **Positive**
b) **Equal**
c) **Negative**
d) **Option a or c**

Ans. C) Negative

Q6. The Algorithm which uses multiple processors to calculate pixel positions is

a) **Midpoint algorithm**
b) **Parallel line algorithm**
c) **Bresenham's line algorithm**
d) **All of the above mentioned**

Ans. b) Parallel line algorithm

Q3. Which algorithm is a faster method for calculating pixel position?

a) **Bresenham's line algorithm**
b) **Parallel line algorithm**
c) **Mid-point algorithm**
d) **DDA line algorithm**

Ans. d) DDA line algorithm