



Experiment No. 2
Implement Bresenham's Line Drawing algorithm.
Name: Priyanka N. S. Bhandari
Roll Number: 02
Date of Performance:
Date of Submission:



### Experiment No. 2

**Aim:** To implement Bresenham's algorithms for drawing a line segment between two given end points.

#### Objective:

Draw a line using Bresenham's line algorithm that determines the points of an n-dimensional raster that should be selected to form a close approximation to a straight line between two points

#### Theory:

In Bresenham's line algorithm pixel positions along the line path are obtained by determining the pixels i.e. nearer the line path at each step.

#### Algorithm -

1. Input two endpoints: (x1, y1) and (x2, y2).
2. Calculate the differences in the x and y coordinates:
3.  $dx = x2 - x1$   $dy = y2 - y1$
4. Initialize variables for tracking the current position, decision parameter, and steps:
5.  $x = x1$   $y = y1$   $d = 2 * dy - dx$   $x\_increment = 1$   $y\_increment = 1$
6. If  $dx < 0$ , set  $x\_increment$  to -1.
7. If  $dy < 0$ , set  $y\_increment$  to -1.
8. Start a loop that runs from 1 to  $dx$  (or  $-dx$  if  $dx$  is negative):
9. a. Plot the pixel at the current position (x, y).
10. b. If the decision parameter is greater than or equal to 0, increment y by  $y\_increment$  and update the decision parameter:
11. if  $d \geq 0$ :  $y = y + y\_increment$   $d = d - 2 * dx$
12. c. Increment x by  $x\_increment$ .
13. d. Update the decision parameter:
14.  $d = d + 2 * dy$
15. Repeat the loop until you have plotted all the necessary pixels to draw the line segment.

#### Program -

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

```
int main()
{
int x,y,x1,y1,x2,y2,p,dx,dy;
```



```
int gd=DETECT,gm=0;
initgraph(&gd,&gm, "");
printf("\n Enter x1 cordinate: ");
scanf("%d",&x1);
printf("\n Enter y1 cordinate: ");
scanf("%d",&y1);
printf("\n Enter x2 cordinate: ");
scanf("%d",&x2);
printf("\n Enter y2 cordinate: ");
scanf("%d",&y2);
```

```
x=x1;
y=y1;
dx=x2-x1;
dy=y2-y1;
```

```
putpixel (x,y, RED);
p = (2 * dy-dx);
```

```
while(x <= x2)
{
if(p<0)
{
x = x+1;
p = p + 2*dy;
}
else
{
x = x + 1;
y = y + 1;
p = p + (2 * dy) - (2 * dx);
}
putpixel (x,y, RED);
}
```

```
getch();
closegraph();
}
```



### Output –



### Conclusion: Comment on -

1. Pixel- The "pixel" is represented by the **putpixel** function. It sets the color of individual pixels on the screen.
2. Equation for line- The algorithm calculates and uses the difference in the x and y coordinates (dx and dy) to determine which pixels to color to approximate the line.
3. Need of line drawing algorithm- The need for a line drawing algorithm arises from the discrete nature of digital screens, which represent images using pixels on a grid. To draw a continuous line on such a grid, an algorithm like Bresenham's is necessary to determine which pixels to color to create the appearance of a smooth line.
4. Slow or fast- Bresenham's algorithm is relatively fast and efficient, especially for drawing lines with integer coordinates. It uses integer arithmetic and avoids floating-point calculations