**PRACTICAL NO 2**

**~ YASH MOHADIKAR**

**COMPS 3 07**

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

**Objective:** Draw a line using Bresenham&#39;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 -**

**Step 1:** Except the two end points of Line from User.

**Step 2:** Calculate the slope(m) of the required Line.

**Step 3:** Identify the value of slope(m).

If slope(m) is Less than 1 i.e: m &lt; 1 Calculate the constants dx, dy, 2dy, and (2dy – 2dx) and get the first value for the decision parameter as - p0 = 2dy − dx

**Step 4:** At each Xk along the line, starting at k = 0, perform the following test −

If pk &lt; 0, the next point to plot is (xk + 1, yk) and pk+1 = pk + 2dy

Else plot (xk + 1, yk + 1) pk+1 = pk + 2dy − 2dx

Repeat step 4 (dx - 1) times.

If slope(m) is greater than or equal to 1 i.e: m &gt;= 1

Calculate the constants dx, dy, 2dy, and (2dy – 2dx) and get the first value for the

decision parameter as -

**step 5:**

p0 = 2dx − dy

At each Yk along the line, starting at k = 0, perform the following test −

If pk &lt; 0, the next point to plot is (xk, yk + 1) and

pk+1 = pk + 2dx else plot (xk + 1, yk + 1) pk+1 = pk + 2dx − 2dy Repeat step 5 (dy - 1) times. Exit.

**Program –**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void bsline(int x,int y,int x2,int y2)

{

int dx,dy,p;

dx=x2-x;

dy=y2-x);

p=2\*(dy)-(dx);

while(x<=x2)

{

if(p<0)

{

x=x+1;

y=y;

p=p+2\*(dy);

}

else

{

x=x+1;

y=y+1;

p=p+2\*(dy-dx);

}

putpixel(x,y,RED);

delay(10);

}

}

void main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:\\turboc3\\bgi");

int x1,x2,y1,y2;

cout<<"Enter the x1,y1,x2,y2 values: ";

cin>>x1>>y1>>x2>>y2;

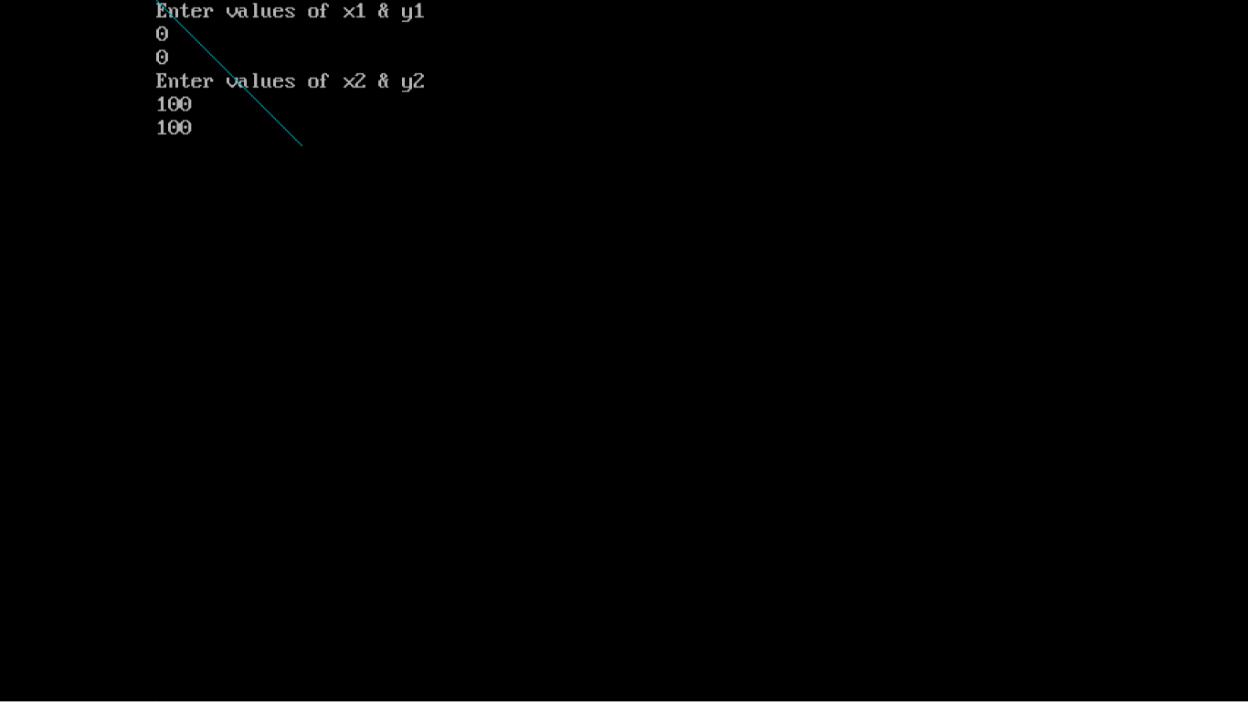
bsline(x1,y1,x2,y2);

getch();

closegraph();

}

**Output –**



**Conclusion:**

**Comment on –**

**Pixel :** Each pixel is chosen based on integer calculations, which makes it highly efficient for drawing lines on digital displays.

**Equation for line :** Y = mx+ c

**Need of line drawing algorithm :** Bresenham's line algorithm is needed primarily for its efficiency and accuracy in drawing straight lines on digital displays.

**Slow or fast :** fast