

Name - Juhi Sharma .

Course - BCA

Section - 'C'

Roll No - 1121068.

Subject - Computer Graphics Practical (PBC-602)

Sem - VI

P1. Bresenham Line Drawing Algorithm.

Algo \rightarrow Step 1: Start Algorithm.

Step 2: Declare variable $x_1, x_2, y_1, y_2, d, i_1, i_2, dx, dy$

Step 3: Enter value of x_1, y_1, x_2, y_2
where x_1, y_1 are co-ordinates of starting point
And x_2, y_2 are co-ordinates of ending point

Step 4: Calculate $dx = x_2 - x_1$

Calculate $dy = y_2 - y_1$

Calculate $i_1 = 2 * dy$

Calculate $i_2 = 2 * (dy - dx)$ & $d = i_1 - dx$

Step 5: 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 $x = x_1$

$y = y_1$, $x_{end} = x_2$.

Step 6: Generate point at (x, y) coordinates

Step 7: Check if whole line is generated.

Name - Juhi Sharma

Course - BCA

Sec - 'C'

Roll No - 1121068

Subject - Computer Graphics Practical (PBC-602)

Sem - VI

if $x \geq x_{end}$
Stop.

Step 8: Calculate co-ordinates of the next pixel

if $d < 0$

then $d = d + 1$

If $d \geq 0$, then $d = d - 1$

Increment $y = y + 1$

Step 9: Increment $x = x + 1$

Step 10: Draw a point of latest (x, y) coordinates

Step 11: Go to step 7.

Step 12: End of Algorithm.

// CODING //

#include <graphics.h>

void main()

{

float $x, y, x_1, y_1, x_2, y_2, dx, dy, steps, p;$

int $i = 1, gd = DETECT, gm;$

③

Name - Juhi Sharma

Course - BCA

Sec - 'C'

Roll No - 1121068

Subject - Computer Graphics Practical (PBC-602)

Sem - VI

— x — x —

```
printf("Enter (x1,y1): ");
scanf("%f%f", &x1,&y1);
printf("Enter (x2,y2): ");
scanf("%f%f", &x2,&y2);
initgraph(&gd,&gm);
```

```
dx = x2 - x1;
```

```
dy = y2 - y1;
```

```
steps = dx - 1;
```

```
int pk = (2 * dy) - dx;
```

```
p = pk;
```

```
x = x1;
```

```
y = y1;
```

```
while (i < steps)
```

```
{
```

```
    if (p < 0)
```

```
    {
```

```
        putpixel(x,y, BLUE);
```

```
        x = x + 1;
```

```
        y = y;
```

```
        p = p + (2 * dy);
```

```
        delay(50);
```

Juhi

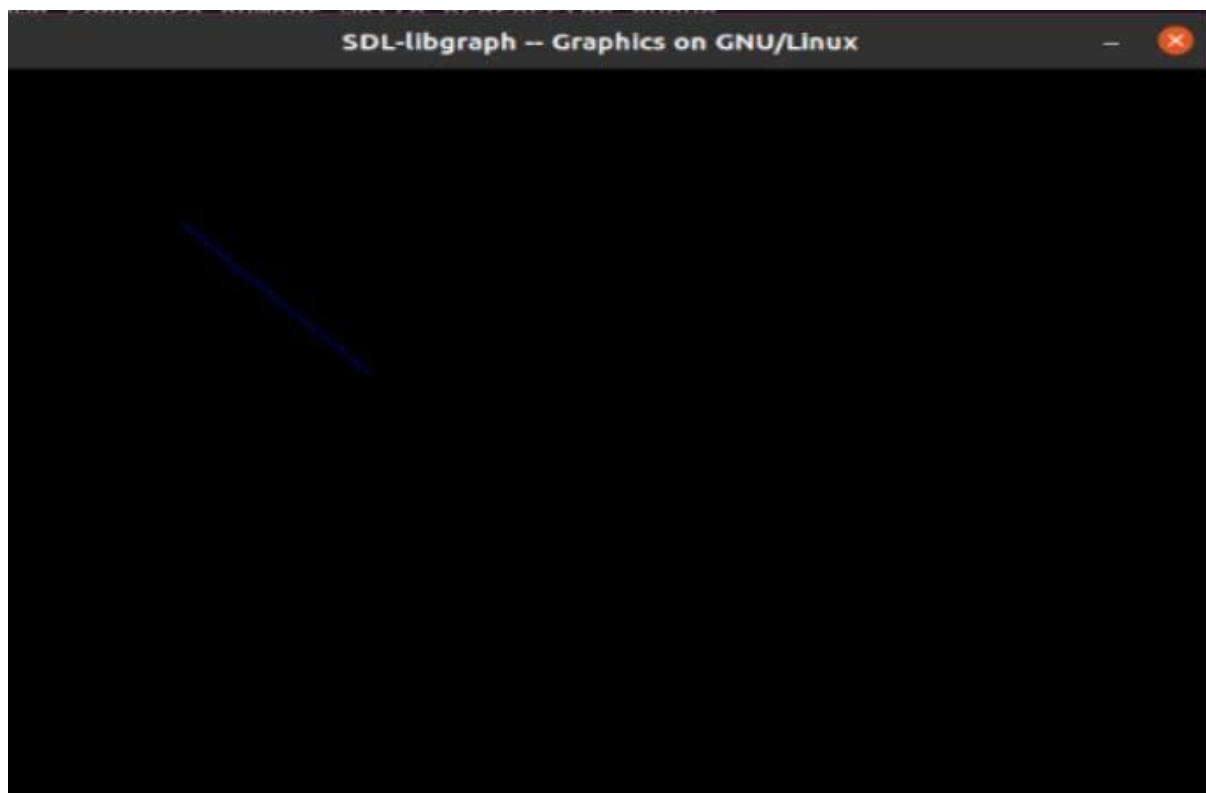
```

else
{
    putpixel(x, y, BLUE);
    x = x + 1;
    y = y + 1;
    p = p + (2 * dy) - (2 * dx);
    delay(50);
}
i++;
}
getch();
closegraph();
}

```

Output: ↓

[Signature]



5

Name - Juhi Sharma

Course - BCA

Sec - 'C'

Roll No - 1121068

— e — a —

P2 Algo for Mid Point Circle.

Step 1: Start.

Step 2: Allot the center coordinates (p_0, q_0) as follows -

$$p_0 = 0, q_0 = r.$$

Step 3: Now, calculate the initial decision parameter $d_0 = 1 - r$;

Step 4: Assume the starting coordinates (p_k, q_k)

The next co-ordinates will be (p_{k+1}, q_{k+1})

Find the next point of first octant according to d_k .

Step 5: Follow these 2 cases -

Case 1: If $d_k < 0$, then

Case 2: If $d_k \geq 0$, then

$$p_{k+1} = p_k + 1$$

$$p_{k+1} = p_k + 1$$

$$q_{k+1} = q_k$$

$$q_{k+1} = q_k - 1$$

$$d_{k+1} = d_k + 2p_{k+1} + 1$$

$$d_{k+1} = d_k - 2(q_{k+1} + 2p_{k+1}) + 1$$

Step 6: If center not $(0,0)$ points will be

$$x_{\text{coordinate}} = x_c + p_0$$

$$y_{\text{coordinate}} = y_c + q_0$$

Step 7: Repeat steps 5 & 6 until $x \geq y$

Step 8: Stop.

Juhi

⑥

Name - Juhi Sharma

Course - BCA

Sec - 'C'

Roll No - 1121068

Sem - VI

Pa

Mid Point Circle

// Coding .

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

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

```
int main()
```

```
{ int gd = DETECT, gm;
```

```
int r, x, y, p, xc = 200, yc = 200;
```

```
printf ("Enter Radius ");
```

```
scanf ("%d", &r);
```

```
initgraph (&gd, &gm, " ");
```

```
x = 0;
```

```
y = r;
```

```
p = 1 - r;
```

```
for (x = 0; x <= y; x++)
```

```
{ if (p < 0)
```

```
{ y = y;
```

```
p = p + (2 * x) + 1;
```

```
}
```

```
else
```

```
{ x = x + 1; y = y - 1;
```

$p = p + (2 * x) - (2 * y) + 1;$

}

putpixel($x_c + x, y_c + y, 7$);

putpixel($x_c + y, y_c + x, 7$);

putpixel($x_c - x, y_c + y, 7$);

putpixel($x_c - y, y_c + x, 7$);

putpixel($x_c - x, y_c - y, 7$);

putpixel($x_c - y, y_c - x, 7$);

putpixel($x_c + x, y_c - y, 7$);

putpixel($x_c + y, y_c - x, 7$);

}

getch();

closegraph();

return 0;

}

[Signature]

SDL-libgraph -- Graphics on GNU/Linux

