NAME VIKAS KUMAR

COURSE :- BCA

Roll No :- 1121165 Section - 'C'

Subject - Computer Graphics Practical (PBC-602)

Sem - VI

P1 :- Bresenham Line Drawing Algorithm

Algo → 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-ordinate of starting point
and x^2, y_2 are co-ordinerate of ending point

step 4 :- calculate ~~d~~ $dx = x_2 - x_1$
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 ~~stas~~ starting point and x end as maximum prossible value of x
if $dx < 0$, then $x = x_2$
$y - y_1$, $x_{end} = x_1$
if $dx > 0$ Then $x = x_1$
$y = y_1$, $x_{end} = x_2$

step 6 :- Generate point at ~~key~~ (x,y) Coordinates

step 7 :- check if whole line is generated
if ~~dx~~ ~~...~~ if $x >= x_{end}$
stop

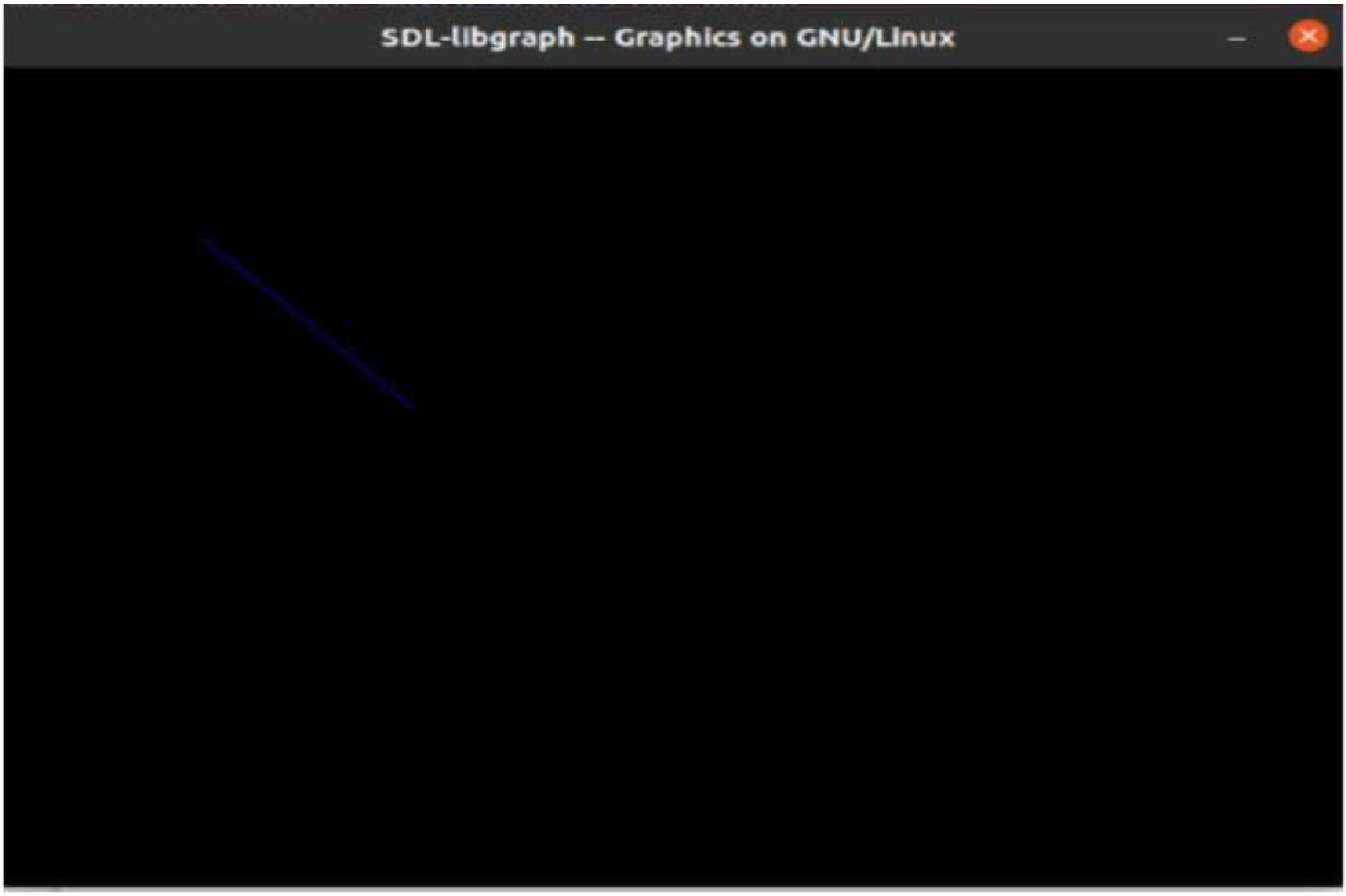
```
Step 8. Calculate co-ordinate of the next points
        if d<0
        then d = d + i1
        if d ≥0 , ~~then~~ then d = d + d + i2
            increment y = y+1
step 9:- Increment x = x+1
step 10:- Draw point of latest (x,y) coordinates
step 11:- go to step 7
step 12:- end of Algorithm.
```

// CODING //

```
# include <graphics.h>
 void main()
 {
float x, y, x1, y1, x2, y2, dx, dy, steps, p;
int i=1, gd = DETECT, gm;
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;
```

Vijay

```
    y=y1;
while  (i<= steps)
  {  if ( p<0)
  {  put pixel ( x,y ,BLUE );
       x = x+1;
        y =y  ;
        P = p+(2* dy );
         ~~dela~~
         delay (50);
         }
       else
  {  put pixel (x,y ,BLUE);
          x = x+1;
          y =y+1 ;
          P = P+ (2*dy) - (2*dx);
            delay (50 );
           }
           i++;
         }
        getch();
        closegraph();
         }
```



SDL-libgraph -- Graphics on GNU/Linux

P2

Algo fo mid point Circle

step 1. start

step 2:- Allot the center Coordinate (po - qo) as follows po = 0, qo = r

step 3:- Now, calculate the initial decision parameter do = 1 - r;

step 4:- Assume the starting coordinate = (pk, qk)
The next co-ordinates will be (pk-1, qk+1)
find the next point of first octant according to dk

step 5:- Follow these 2 case:-

case 1 if dk < 0, then

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

$q_k + 1 = q_k$

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

case: if dk >= 0, then

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

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

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

step 6:- if Center not (0,0) points will be
x coordinate = xc + po
y coordinate = yc + qo

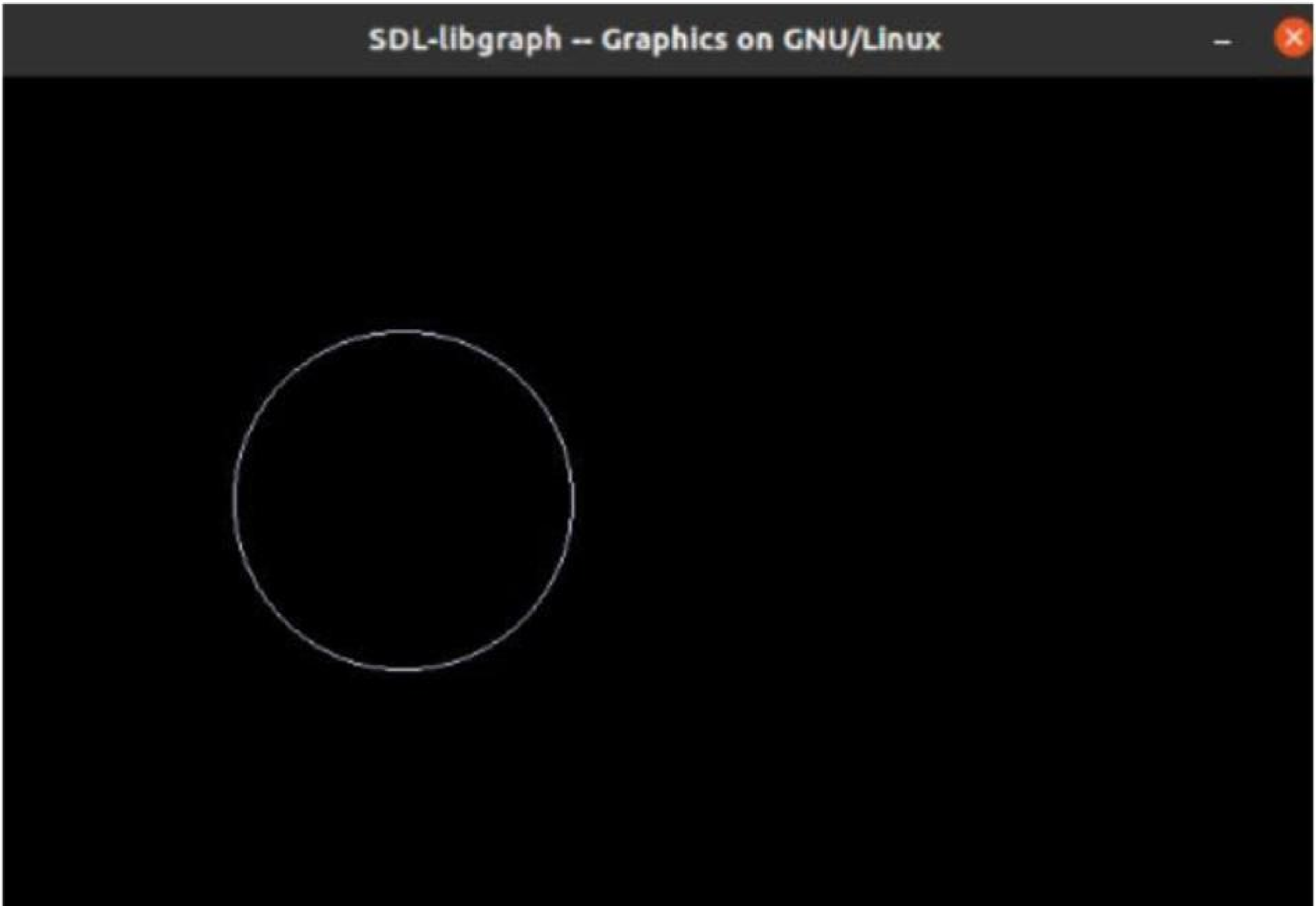
step 7:- Repeats step ~~st~~ 5 and 6 until x >= y

step 8: stop

Q P2 **coding**

```
# include <stdio.h>
# include < geophics.h>
Int main ()
{
int ~~gd~~ gd= DETECT ,gm;
int r,x.y .P, xc= 20 0,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
   { y=y-1;
 P=P+(2*x) -(2*y)+1;
  }
Put pixel (xc+x,yc+y,7);
Put pixel (xc+y ,yc+x ,7)
Put pixel (xc -x ,yc+y ,7)
```

```
Putpixel (xc+-y ,yc+x ,7);
Putpixel xc-x ,yc-y, ,7);
putpixel (xc-y, yc-x, 7)
 putpixel (xc+x, ,yc-y, ,7)
 putpixel (xc+y, yc-y,, 7
 {
  getch ();
  closegraph ();
    return 0;
    }
```



SDL-libgraph -- Graphics on GNU/Linux