Name:- KOMAL TOPWAL

University Roll No:- 1121074

Course:- BCA Semester: 6th

Paper Name:- COMPUTER GRAPHICS

Paper Code:- PBC 602

(P1) Bresenham Line Drawing Algorithm.

Step1 :- Start

Step2 :- Declare x, y, x1, y1, x2, y2, dx, dy, step, p as float type.

Step 3 :- Declare gm and initialize gd = DETECT & i=1

Step 4 :- Enter coordinates of first point for variable x1 & y1.

Step 5 :- Enter coordinates of second point for variable x2 & y2.

Step 6 :- Initialize graph by using initgraph(&gd, &gm, "");

Step 7 :- Calculate, dx = x2 - x1;
dy = y2 - y1;
steps = dx - 1;

Step 8 :- Initialize decision parameter
pR = (2*dy) - dx

Step 9 :- Initialize p = pR, x = x1, y = y1

Step 10 :- Repeat Step 11 to step 13 while i <= steps.

Step 11 :- Check if p < 0, then
putpixel (x, y, WHITE)
x = x + 1;
y = y;
p = p + (2*dy);

Otherwise go to step 12

2021/6/16 10:25

Step 12 :- putpixel (x, y, WHITE)

x = x + 1;

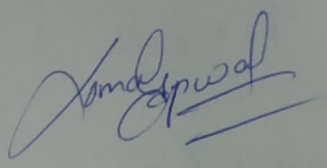
y = y + 1;

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

Step 13 :- Increment p by one

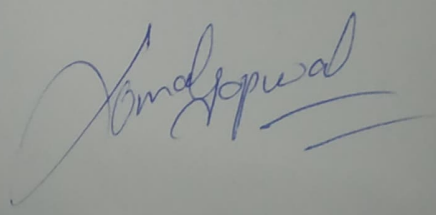
Step 14 :- Close the graph

Step 15 :- Stop



```
#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;
  y = y1;
  while (i <= steps)
  {
    if (p < 0)
    {
      putpixel (x, y, WHITE);
      x = x + 1;
      y = y;
      p = p + (2*dy);
      delay(50);
    }
```

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



P2 Mid point circle drawing Algorithm

Step 1 :- Start

Step 2 :- Initialize gd = DETECT and declare gm;

Step 4 :- Declare r, x, y, p and initialize
xc = 200 , yc = 200;

Step 5 :- Enter the radius for circle as r,

Step 6 :- Initialize graph by using initgraph(&gd, &gm, " ");

Step 7 :- Initialize, x = 0
y = r
and calculate p = 1 - r

Step 8 :- Initialize 0 to x, check if x is less than or equal to y increment x

Step 9 :- Check

Step 8 :- Initialize 0 to x, repeat step 9 to step 12 if x is less than or equal to y and increment x by one.

Step 9 :- Check p is less than 0,

Step 10 :- If step 9 is true,
y = y
p = p + (2 * x) + 1

Step 11 :- If step 9 is false
y = y - 1
p = p + (2 * x) - (2 * y) + 1;

Formula proof

2021/6/16 10:19

Step 12: Plot pixels using putpixel

```
putpixel (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-x, 7);
putpixel (xc+x, yc-y, 7);
putpixel (xc+y, yc-x, 7);
```

Step 13: Close the graph.

Step 14: Stop

```
#Include <stdio.h>
#include <graphics.h>
int main()
{
  int gd = DETECT, gm;
  Pnt 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
      {
          y=y-1
          p= p+(2*x)-(2*y)+1;
      }
    putpixel (xc+x, yc+y, 7);
    putpixel (xc+y, yc+x, 7);
    putpixel (xc-x, yc+y, 7);
    putpixel (xc-y, yc+x, 7);
```

Somdatpwal

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