Midpoint Circle Generation Algorithm

To implement midpoint circle generation algorithm or bresenham’s circle algorithm for drawing a circle of given center (x, y) and radius r

Experiment - 3

COMPUTER GRAPICS AND MULTIMEDIA

Syeda Reeha Quasar

14114802719

Group - 3C7

# **EXPERIMENT – 2**

## **AIM:**

To implement midpoint circle generation algorithm or bresenham’s circle algorithm for drawing a circle of given center (x, y) and radius r.

## **THEORY:**

Circles have the property of being highly symmetrical, which is handy when it comes to drawing them on a display screen.

* We know that there are 360 degrees in a circle. First, we see that a circle is symmetrical about the x axis, so only the first 180 degrees need to be calculated.
* Next, we see that it's also symmetrical about the y axis, so now we only need to calculate the first 90 degrees.
* Finally, we see that the circle is also symmetrical about the 45-degree diagonal axis, so we only need to calculate the first 45 degrees.
* We only need to calculate the values on the border of the circle in the first octant. The other values may be determined by symmetry Bresenham's circle algorithm calculates the locations of the pixels in the first 45 degrees. It assumes that the circle is centered on the origin. So, for every pixel (x, y) it calculates, we draw a pixel in each of the eight octants of the circle. This is done till when the value of the y coordinate equals the x coordinate. The pixel positions for determining symmetry are given in the below algorithm.
* Assume that we have just plotted point (xk, yk)
* The next point is a choice between (xk+1, yk) and (xk+1, yk-1)
* We would like to choose the point that is nearest to the actual circle.
* So, we use decision parameter here to decide

### **Decision parameters:**

1. In this the input radius r is there with a center (xc , yc ). To obtain the first point m the circumference of a circle is centered on the origin as (x0 ,y0 ) = (0,r).
2. Calculate the initial decision parameters which are: p0 =5/4-r or 1-r
3. Now at each xk position starting k=0, perform the following task. if pk < 0 then plotting point will be ( xk+1 ,yk ) and Pk+1=pk +2(xk+1) +1 else the next point along the circle is (xk+1, yk-1 ) and Pk+1=pk+2(xk+1) +1-2(yk+1)
4. Determine the symmetry points in the other quadrants.
5. Now move at each point by the given center that is: x=x+xc y=y+yc
6. At last repeat steps from 3 to 5 until the condition x>=y

## **Midpoint Circle Generation Algorithm:**

**Given -** Centre point of Circle = (X0 , Y0 )

Radius of Circle = R

The points generation using Mid Point Circle Drawing Algorithm involves the following steps-

**Step-01:**

**Assign the starting point coordinates (X0, Y0) as –**

**X0 = 0**

**Y0 = R**

**Step-02:**

**Calculate the value of initial decision parameter P0 as –**

**P0 = 1 – RStep3:**

Suppose the current point is (Xk , Yk ) and the next point is (Xk+1, Yk+1).

Find the next point of the first octant depending on the value of decision parameter Pk .

**Step4:** If the given centre point (X0 , Y0 ) is not (0, 0), then do the following and plot the point –

Xplot = Xc + X0

Yplot = Yc + Y0

Here, (Xc , Yc ) denotes the current value of X and Y coordinates.

**Step5:**

Keep repeating Step-03 and Step-04 until Xplot >= Yplot.

**Step6:**

Step-05 generates all the points for one octant. To find the points for other seven octants, follow the eight symmetry property of circle

**Step7:** End Algorithm

# Algorithm implementation:

# **Source Code:**

# #include<stdio.h>

# #include<conio.h>

# #include<graphics.h>

# circle\_func(int xc,int yc,int r, int clr )

# {

# int i,y,x=0;

# float d;

# 

# d=1.25-r;

# y=r;

# do{

# if(d<0.0){

# x=x+1;

# d=d+2\*x+1;

# }

# else{

# x=x+1;

# y=y-1;

# d=d+2\*x-2\*y+1;

# }

# putpixel(xc+x,yc+y,clr);

# putpixel(xc-y,yc-x,clr);

# putpixel(xc+y,yc-x,clr);

# putpixel(xc-y,yc+x,clr);

# putpixel(xc+y,yc+x,clr);

# putpixel(xc-x,yc-y,clr);

# putpixel(xc+x,yc-y,clr);

# putpixel(xc-x,yc+y,clr);

# }

# while(x<y);

# }

# main()

# {

# initwindow(800,800);

# int i = 100;

# while (i <= 700){

# circle\_func(i,200,10, 1);

# circle\_func(i,200,100, 7);

# circle\_func(i,180,50, 2);

# circle\_func(i,270,100, 3);

# i += 100;

# delay (100);

# }

# 

# i = 200;

# while (i < 700){

# circle\_func(i,400,10, 1);

# circle\_func(i,400,100, 7);

# circle\_func(i,380,50, 2);

# circle\_func(i,370,100, 3);

# i += 100;

# delay (100);

# }

# 

# for (i = 1; i <= 100; i++)

# {

# circle\_func(400,500,i, 4);

# }

# delay (100);

# for (i = 1; i <= 50; i++)

# {

# circle\_func(400,300,i, 1);

# circle\_func(400,400,i, 1);

# }

# getch();

# }

# **OUTPUT**

# **VIVA QUESTIONS:**

**Q1. Define getpixel and putpixel functions in graphics.h?**

Ans. **getpixel():** Function getpixel returns the color of pixel present at point(x, y).

Declarations: int getpixel(int x, int y);

**Putpixel():** Function putpixel plots a pixel at a point(x, y) of the specified color.

Declarations: void putpixel(int x, int y, int color);

**Q2. What is the difference between DDA algorithm and Bresenham’s Algorithm?**

Ans.

|  |  |
| --- | --- |
| DDA Algorithm | Bresenham's Line Algorithm |
| 1. DDA Algorithm use floating point, i.e., Real Arithmetic. | 1. Bresenham's Line Algorithm use fixed point, i.e., Integer Arithmetic |
| 2. DDA Algorithms uses multiplication & division its operation | 2.Bresenham's Line Algorithm uses only subtraction and addition its operation |
| 3. DDA Algorithm is slowly than Bresenham's Line Algorithm in line drawing because it uses real arithmetic (Floating Point operation) | 3. Bresenham's Algorithm is faster than DDA Algorithm in line because it involves only addition & subtraction in its calculation and uses only integer arithmetic. |
| 4. DDA Algorithm is not accurate and efficient as Bresenham's Line Algorithm. | 4. Bresenham's Line Algorithm is more accurate and efficient at DDA Algorithm. |
| 5.DDA Algorithm can draw circle and curves but are not accurate as Bresenham's Line Algorithm | 5. Bresenham's Line Algorithm can draw circle and curves with more accurate than DDA Algorithm. |

**Q3. Give disadvantages of DDA Algorithm?**

Ans. It involves floating point additions rounding off is done. Accumulations of round off error cause accumulation of error.

Rounding off operations and floating point operations consumes a lot of time.

It is more suitable for generating line using the software. But it is less suited for hardware implementation.

**Q4. Give disadvantages of Bresenham’s algorithm?**

Ans. This algorithm is meant for basic line drawing only Initializing is not a part of Bresenham's line algorithm. So to draw smooth lines, you should want to look into a different algorithm.