第三章 基本图形的生成——直线、圆、椭圆生成算法

扫描转换直线段

- DDA算法
- 中点画线法
- Bresenham画线算法

圆弧、椭圆弧扫描转换

- 中点算法
- 内接正多边形逼近法
- 等面积正多边形逼近法
- 生成圆弧的正负法

直线段的扫描转换算法

- **直线的扫描转换：**拟定最佳逼近于该直线的一组象素，而且按扫描线顺序，对这些象素进行写操作。
- 三个常用算法：

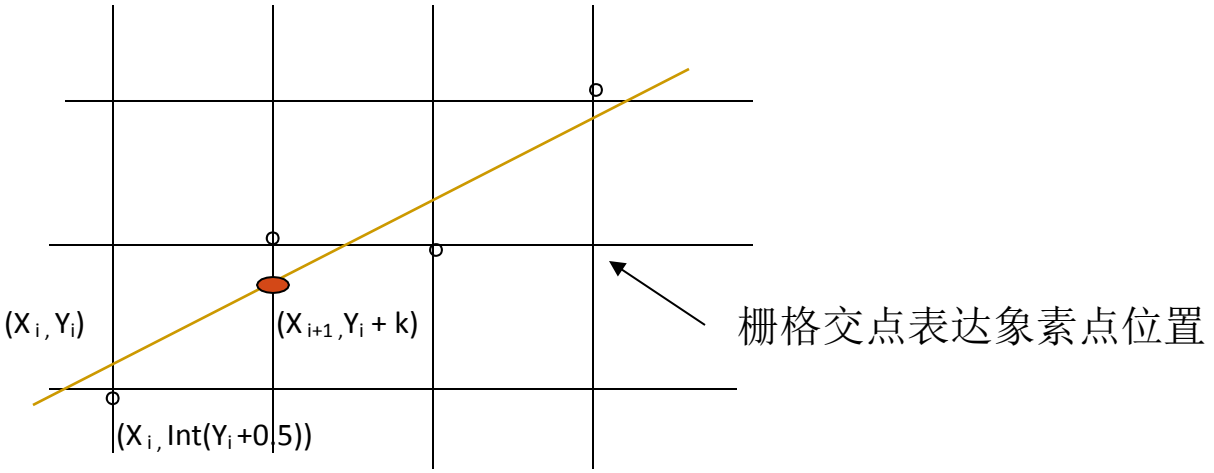
  数值微分法（DDA）

  中点画线法

  Bresenham算法。

数值微分法（ＤＤＡ）

假定直线的起点、终点分别为：（x0, y0），(x1, y1)，且都为整数。



数值微分(DDA)法

- 基本思想

已知过端点P0 (x0, y0), P1(x1, y1)的直线段L

y=kx+b

直线斜率为 $k = \dfrac{y_1 - y_0}{x_1 - x_0}$

$$令 x = x_0 \rightarrow x_1; x = x + stepx$$

$$y = kx + b$$

$$\therefore (x, round(y))$$

这种措施直观，但效率太低，因为每一步需要一次浮点乘法和一次舍入运算。

数值微分(DDA)法

计算$y_{i+1}= kx_{i+1}+b$

$= kx_i+b+k\Delta x$

$= y_i+k\Delta x$

当$\Delta x = 1$；　　$y_{i+1} = y_i+k$

- 即：当x每递增1，y递增k(即直线斜率)；
- 注意上述分析的算法仅合用于$|k| \leqslant 1$的情形。在这种情况下，x每增长1,y最多增长1。
- 当 $|k| > 1$时，必须把x，y地位互换

数值微分(DDA)法

- 增量算法：在一种迭代算法中，假如每一步的x、y值是用前一步的值加上一种增量来取得，则称为增量算法。
- DDA算法就是一种增量算法。

数值微分(DDA)法

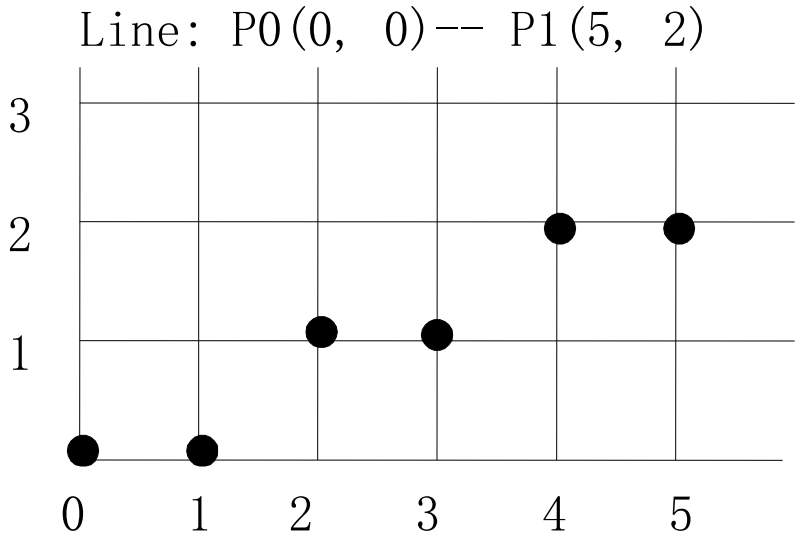
```
void DDALine(int x0,int y0,int x1,int y1,int color)
 {  int x;
  float dx, dy, y, k;

  dx, = x1-x0, dy=y1-y0;
  k=dy/dx, y=y0;
  for (x=x0; x≤x1, x++)
  {  drawpixel (x, int(y+0.5), color);
     y=y+k;
     }
  }
```

数值微分(DDA)法

- 例：画直线段$P_0(0,0)$--$P_1(5,2)$

x	int(y+0.5)	y+0.5
0	0	0+0.5
1	0	0.4+0.5
2	1	0.8+0.5
3	1	1.2+0.5
4	2	1.6+0.5
5	2	2.0+0.5

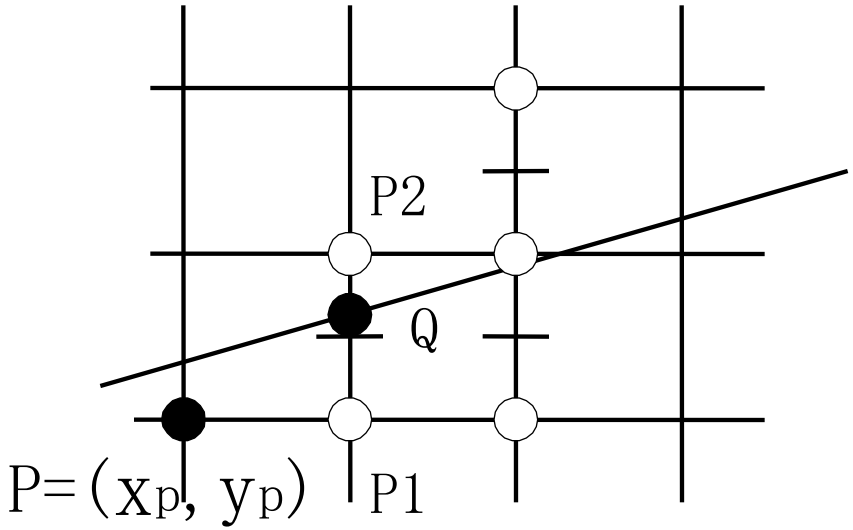


数值微分(DDA)法

- 缺陷：　　在此算法中，y、k必须是float，且每一步都必须对y进行舍入取整，不利于硬件实现。

中点画线法

- 原理：

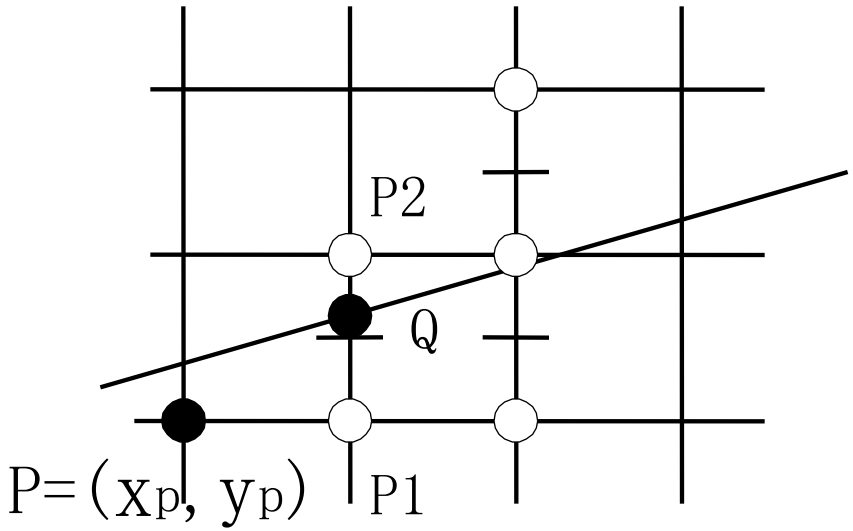


假定直线斜率0<K<1，且已拟定点亮象素点P（X_p ,Y_p ），则下一种与直线最接近的像素只能是P_1点或P_2点。设M为中点，Q为交点

现需拟定下一种点亮的象素。

中点画线法

- 当M在Q的下方-> P_2离直线更近更近->取P_2 。
- M在Q的上方-> P_1离直线更近更近->取P_1
- M与Q重叠， P_1、P_2任取一点。



- 问题：怎样判断M与Q点的关系？

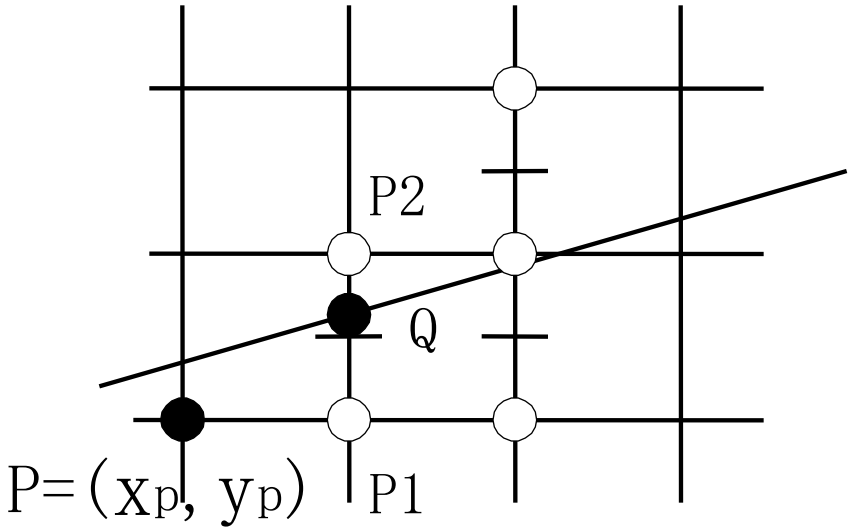
中点画线法

假设直线方程为：$ax+by+c=0$

其中$a=y_0-y_1$，$b=x_1-x_0$，$c=x_0y_1-x_1y_0$

由常识知：

$$\begin{cases} F(x,y)=0 & \text{点在直线上面} \\ F(x,y)>0 & \text{点在直线上方} \\ F(x,y)<0 & \text{点在直线下方} \end{cases}$$



∴欲判断中点M点是在Q点上方还是在Q点下方，只需把M代入$F(x, y)$，并检验它的符号。

中点画线法

构造鉴别式：

$d=F(M)=F(x_p+1, y_p+0.5)$

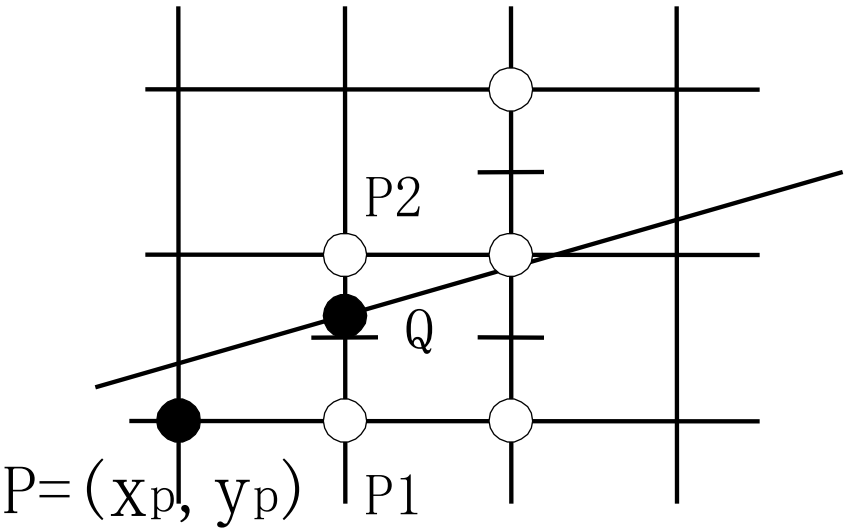
$(x_p+1)+b(y_p+0.5)+c$

当$d<0$，M在直线(Q点)下方，取右上方P_2；

当$d>0$，M在直线(Q点)上方，取右方P_1；

当$d=0$，选P_1或P_2均可，约定取P_1；

能否采用增量算法呢？



中点画线法

若d≥0 ->M在直线上方->取P_1;

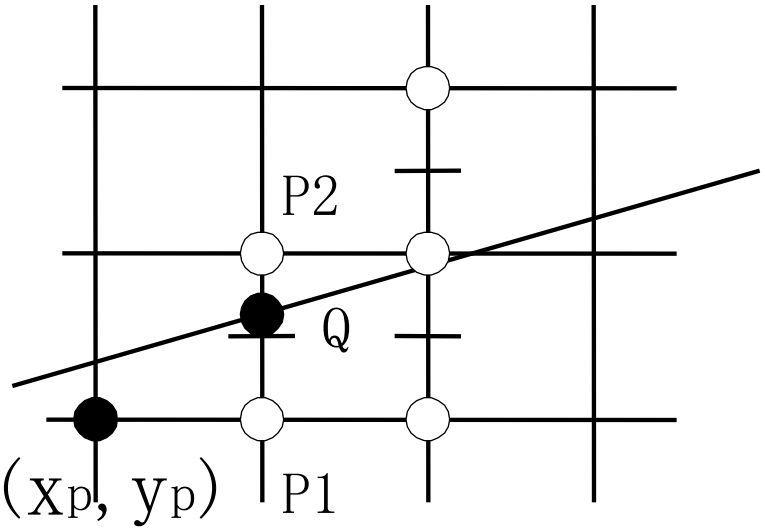
此时再下一种象素的鉴别式为

$$d_1=F(x_p+2,\ y_p+0.5)$$

$$=a(x_p+2)+b(y_p+0.5)+c$$

$$=a(x_p+1)+b(y_p+0.5)+c+a=d+a;$$

增量为a



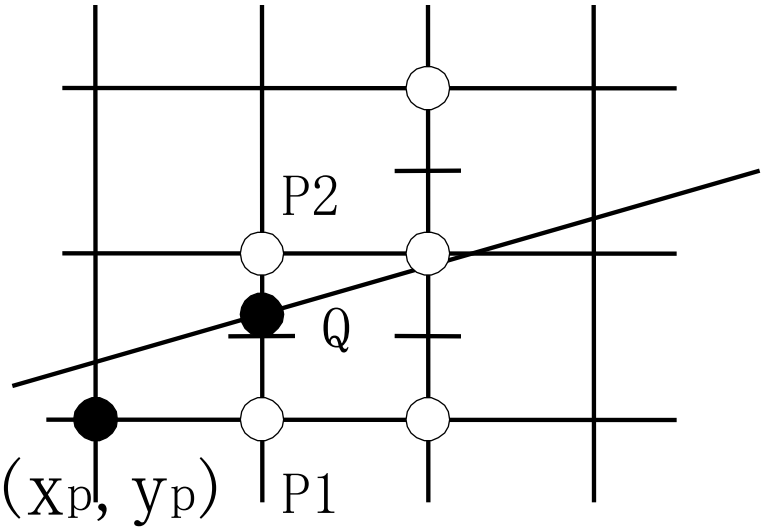
中点画线法

- 若d<0->M在直线下方->取P_2;
- 此时再下一种象素的鉴别式为

$$d_2 = F(x_p+2,\ y_p+1.5)$$

$$=a(x_p+2)+b(y_p+1.5)+c$$

$$= a(x_p+1)+b(y_p+0.5)+c +a +b = d+a+b$$ ； 增量为a＋b



中点画线法

- 画线从(x_0, y_0)开始，d的初值

$$d_0=F(x_0+1, y_0+0.5)= a(x_0+1)+b(y_0+0.5)+c$$
$$= F(x_0, y_0)+a+0.5b =$$
$$a+0.5b$$

因为只用d 的符号作判断，为了只包括整数运算，能够用2d替代d来摆脱小数，提升效率。

中点画线法

```
void Midpoint Line (int x0,int y0,int x1, int y1,int color)
   {   int a, b, d1, d2, d, x, y;
    a=y0-y1, b=x1-x0, d=2*a+b;
    d1=2*a, d2=2* (a+b);
    x=x0, y=y0;
    drawpixel(x, y, color);
    while (x<x1)
    { if (d<0)        {x++; y++; d+=d2; }
     else        {x++; d+=d1;}
     drawpixel (x, y, color);
     }  /* while */
 } /* mid PointLine */
```

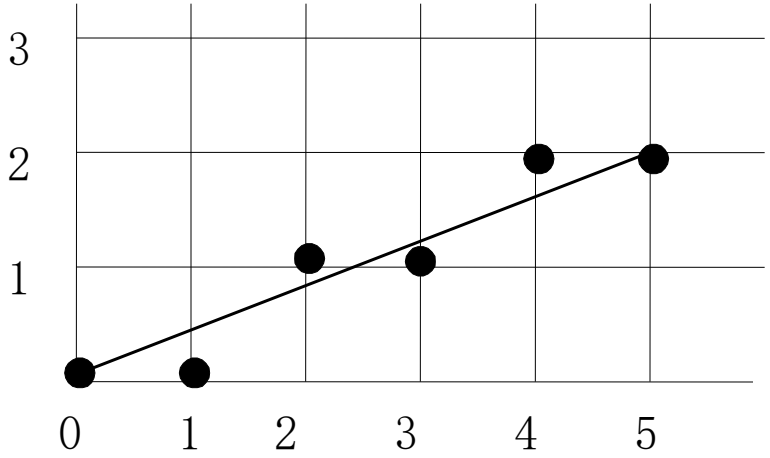
中点画线法

例：用中点画线法P0(0,0) P1(5,2)

a=y0-y1=-2　b=x1-x0=5

d0=2a+b=1　d1=2a=-4　d2=2(a+b)=6

i	xi	yi	d
1	0	0	1
2	1	0	-3
3	2	1	3
4	3	1	-1
5	4	2	5



Bresenham画线算法

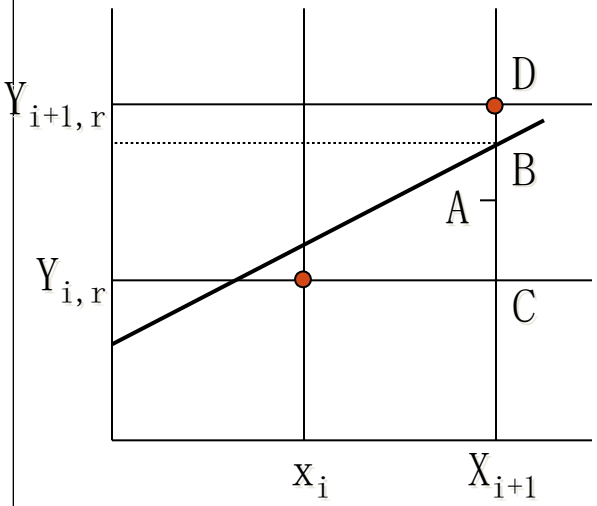
在直线生成的算法中Bresenham算法是最有效的算法之一。令 $k=\Delta y/\Delta x$，就$0\leqslant k\leqslant 1$的情况来阐明Bresenham算法。由DDA算法可知：

$$y_{i+1}=y_i+k \qquad (1)$$

因为k不一定是整数，由此式求出的y_i也不一定是整数，所以要用坐标为（x_i, y_{ir}）的象素来表达直线上的点，其中y_{ir}表达最接近y_i的整数。

Bresenham画线算法

设图中x_i列上已用（x_i, y_{ir}）作为表达直线的点，又设B点是直线上的点，其坐标为（x_{i+1}, y_{i+1}），显然下一种表达直线的点（ $x_{i+1}, y_{i+1,r}$）只能从图中的C或者D点中去选。设A为CD边的中点。 若B在A点上面则应取D点作为（ $x_{i+1}, y_{i+1,r}$），不然应取C点。



ε (x)的几何意义

为能拟定B在A点上面或下面，令

$$\varepsilon(x_{i+1}) = y_{i+1} - y_{ir} - 0.5 \qquad (2)$$

若B在A的下面，则有 $\varepsilon(x_{i+1}) < 0$，反之，则 $\varepsilon(x_{i+1}) > 0$。由图可知

$$y_{i+1,r} = y_{ir} + 1, \quad 若\ \varepsilon(x_{i+1}) \geqslant 0 \qquad (3)$$

$$y_{i+1,r} = y_{ir}, \quad 若\ \varepsilon(x_{i+1}) \leqslant 0$$

Bresenham画线算法

由式（2）和式（3）可得到

$$\varepsilon(x_{i+2}) = y_{i+2} - y_{i+1,r} - 0.5$$
$$= y_{i+1} + k - y_{i+1,r} - 0.5 \quad (4)$$

$$y_{i+1} - y_{ir} - 0.5 + k - 1, \text{当} \ \varepsilon(x_{i+1}) \geqslant 0$$
$$y_{i+1} - y_{ir} - 0.5 + k, \quad \text{当} \ \varepsilon(x_{i+1}) \leqslant 0$$

$$\varepsilon(x_{i+2}) = \varepsilon(x_{i+1}) + k - 1 \ , \text{当} \ \varepsilon(x_{i+1}) \geqslant 0$$
$$\varepsilon(x_{i+2}) = \varepsilon(x_{i+1}) + k \ , \quad \text{当} \ \varepsilon(x_{i+1}) \leqslant 0$$

由式（1）和式（2）可得到

$$\varepsilon(x_2) = k - 0.5 \quad (5)$$

Bresenham画线算法

程序如下：

```
BresenhamLine(x0,y0,x1,y1,color)
int x0,y0,x1,y1,color;
{
    int x,y,dx,dy;
    float k,e;          int e;
    dx = x1-x0;
    dy = y1-y0;
     k = dy/dx;
     e = -0.5; x=x0; y=y0;    e1 = -
dx;

     for( i=0; i<=dx; i++){
        drawpixel(x,y,color);
         x++; e=e+k; e1=e-0.5;
         e=e+2*dy; e1= e-dx;
         if(e1 > 0) e = e - 1;   e = e
- 2*dx;

         if(e >=0) y++;
     }
}
```

圆的扫描转换算法

下面仅以圆心在原点、半径R为整数的圆为例，讨论圆的生成算法。

假设圆的方程为：

$$X^2 + Y^2 = R^2$$

圆弧扫描算法

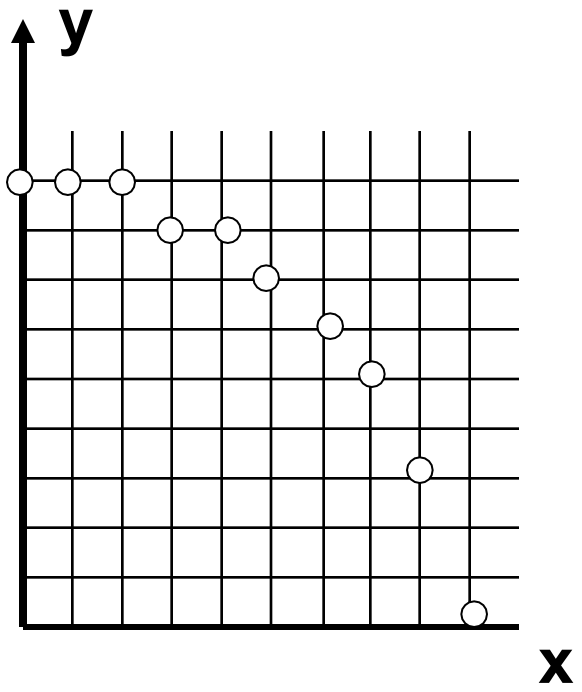
- $X^2 + Y^2 = R^2$

$Y = \pm Sqrt(R^2 - X^2)$

在一定范围内，每给定一X值，可得一Y值。

当X取整数时，Y须取整。

缺陷：浮点运算，开方，取整，不均匀。



角度DDA法

$$x = x_0 + R\cos\theta$$

$$y = y_0 + R\sin\theta$$

$$dx = -R\sin\theta d\theta$$

$$dy = R\cos\theta d\theta$$

$$x_{n+1} = x_n + dx$$

$$y_{n+1} = y_n + dy$$

$$x_{n+1} = x_n + dx = x_n - R\sin\theta d\theta = x_n - (y_n - y_0)d\theta$$

$$y_{n+1} = y_n + dy = y_n + R\cos\theta d\theta = y_n + (x_n - x_0)d\theta$$

显然，拟定x,y的初值及dθ值后，即能够增量方式取得圆周上的坐标，然后取整可得象素坐标。但要采用浮点运算、乘法运算、取整运算。