

GIS专业主干课 : 21905001

计算机图形学

Computer Graphics

林伟华

中国地质大学（武汉）信息工程学院

lwhecug@163.com

目录

- DDA画圆算法
- 中点画圆算法
- Bresenham画圆算法
- Bresenham椭圆算法

圆/椭圆绘制方法

– VC++

- `CDC::Ellipse(int x1, int y1, int x2, int y2);`

– OpenGL

- `void gluDisk(
• GLUquadricObj * qobj,
• GLdouble innerRadius,
• GLdouble outerRadius,
• GLint slices,
• GLint loops);`

– MapGIS

- `void _Circle(MyDC mdc,double rad);`

– 绘点方式

- `CDC::SetPixel(POINT point,COLORREFcrColor);` **//vc++**
- `setPixel(int x, int y);` **//OpenGL**

简单画圆算法

圆的方程: $(x - x_0)^2 + (y - y_0)^2 = r^2$

y与x关系: $y = y_0 \pm \sqrt{r^2 - (x - x_0)^2}$

画圆算法:

```
void circleSimple(int xCenter, int yCenter, int  
    radius, Color c)
```

```
{
```

```
    int x, y, r2;
```

```
    r2 = radius * radius;
```

```
    for (x = -radius; x <= radius; x++)
```

```
{
```

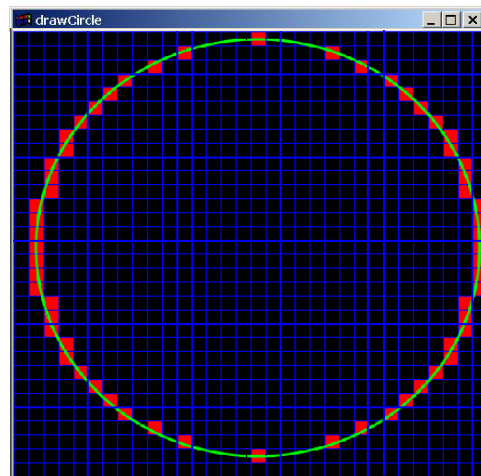
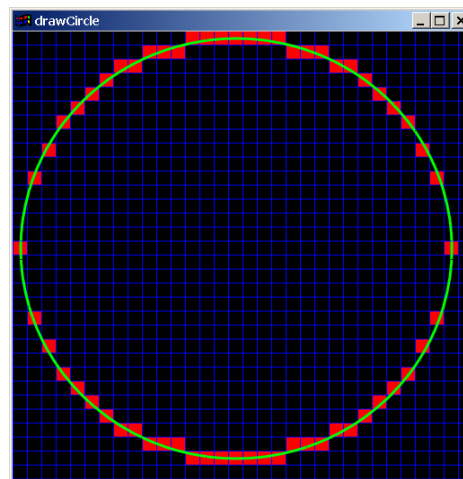
```
        y = (int)(sqrt(r2 - x*x) + 0.5);
```

```
        setPixel(xCenter + x, yCenter + y, c);
```

```
        setPixel(xCenter + x, yCenter - y, c);
```




```
    }
```

```
}
```



DDA画圆算法

■ 1、角度DDA法

- 若已知圆的方程： $x^2 + y^2 = R^2$
- $$\begin{cases} x = x_0 + R\cos\theta \\ y = y_0 + R\sin\theta \end{cases}$$
- 
$$\begin{cases} dx = -R\sin\theta d\theta \\ dy = R\cos\theta d\theta \end{cases}$$
- 
$$\begin{cases} x_{n+1} = x_n + dx \\ y_{n+1} = y_n + dy \end{cases}$$
- 
$$\begin{cases} x_{n+1} = x_n - (y_n - y_0) d\theta \\ y_{n+1} = y_n + (x_n - x_0) d\theta \end{cases}$$

DDA画圆算法

- 角增量 θ (弧度)的选取： $d\theta = \theta/(n-1)$
- n 越大，点越多，速度越慢。所以在不同的精度下，对于不同的半径给定不同的 $d\theta$ ：
- 使 $\max(|\Delta x|, |\Delta y|) \leq 1$
- 因： $\Delta x = - (y_n - y_0) d\theta$
- $\Delta y = (x_n - x_0) d\theta$
- 即 $\max(|(y_n - y_0) d\theta|, |(x_n - x_0) d\theta|) \leq 1$
- 又因为 $|y_n - y_0|, |x_n - x_0|$ 最大是 R
- 所以： $R|d\theta| \leq 1 \longrightarrow |d\theta| \leq 1/R$
- 绝对值表示画圆弧有顺时针和逆时针之别，
- 为精度计，取 $|d\theta| = 1/(R+1)$

DDA画圆算法

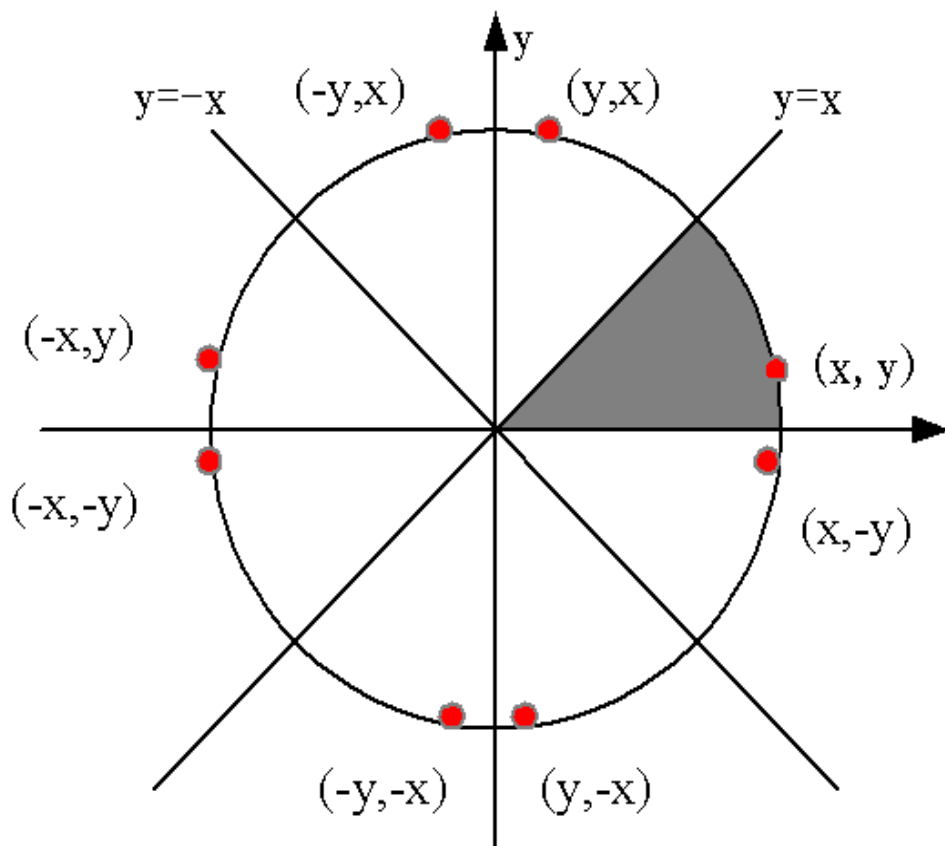
Arc-dda (xc, yc, r, a1, a2, color)

- **int xc, yc, r, color;**
- **double a1, a2;**
- **{ int i, steps, x, y;**
- **double da, radin;**
- **da = 1/(r+1) ;**
- **radin = a2 - a1; steps = radin/da;**
- **x = r*cos(a1); y = r * sin(a1);**
- **for(i = 0; i ≤ steps; i++)**
- **{ drawpixel(x+xc , y+ yc , color);**
- **x = x - y*da ; y = y+ x*da ;**
- **}**
- **}**

圆扫描转换

■ 圆的对称性

- 八分圆的对称性特点。
- 八分圆的圆周上的某点 (x, y) 可计算出其他七个八分圆圆周上对应的点的坐标。
- 圆可由图中阴影的圆周复制而成。



圆扫描转换

■ 算法如下：

- `int Circle_Points(int x,int y,int value)`
- `{ drawpixel(x,y,value);`
- `drawpixel(x,-y,value);`
- `drawpixel(-x,y,value);`
- `drawpixel(-x,-y,value);`
- `drawpixel(y, x, value);`
- `drawpixel(-y, x, value);`
- `drawpixel(y, -x, value);`
- `drawpixel(-y, -x, value);`
- `}`

解决的问题：

如何确定8分圆圆弧的坐标，最终画出圆。

中点画圆法

- 利用圆的对称性，只须讨论1/8圆(第一象限中的第二个八分圆)。
- **解决问题：**

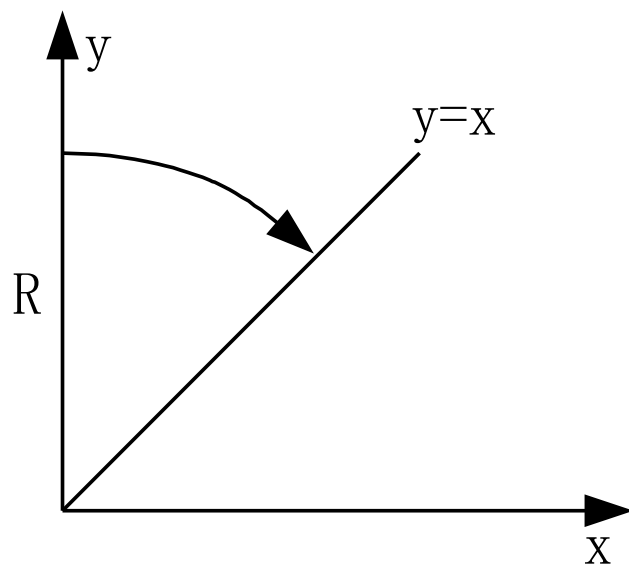
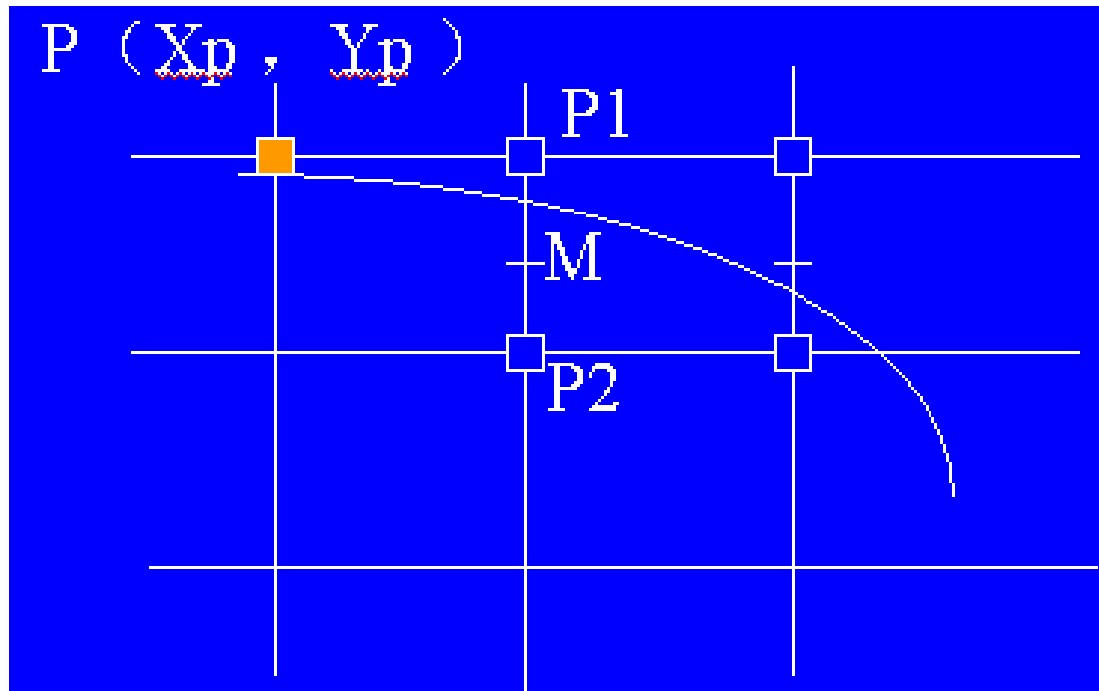


图5-10 1/8圆弧

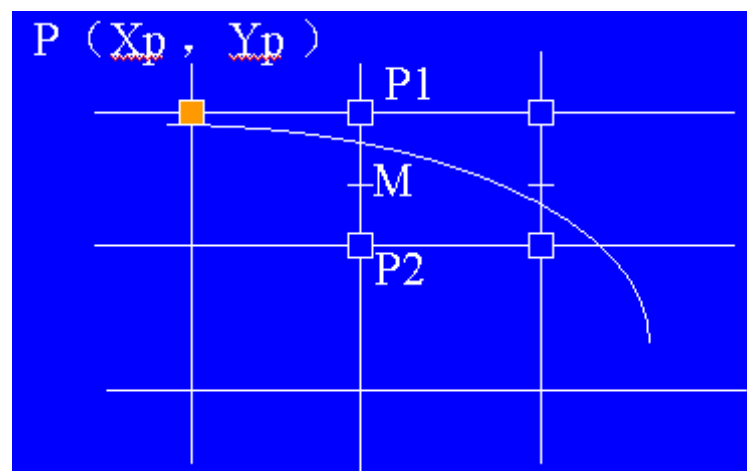
中点画圆法

- **基本原理**：假设 $P(X_p, Y_p)$ 为当前点亮像素，那么，下一个点亮的像素可能是 $P1(X_p+1, Y_p)$ 或 $P2(X_p+1, Y_p+1)$ 。



中点画圆法

- 推导过程：
- ① 构造一函数：
- $F(X, Y) = X^2 + Y^2 - R^2$
- $F(X, Y) = 0$ (X, Y) 在圆上；
- $F(X, Y) < 0$ (X, Y) 在圆内；
- $F(X, Y) > 0$ (X, Y) 在圆外。
- M为P1、P2间的中点，
- $M = (X_p + 1, Y_p - 0.5)$
- 有如下结论：
- $F(M) < 0$ 取P1
- $F(M) \geq 0$ 取P2



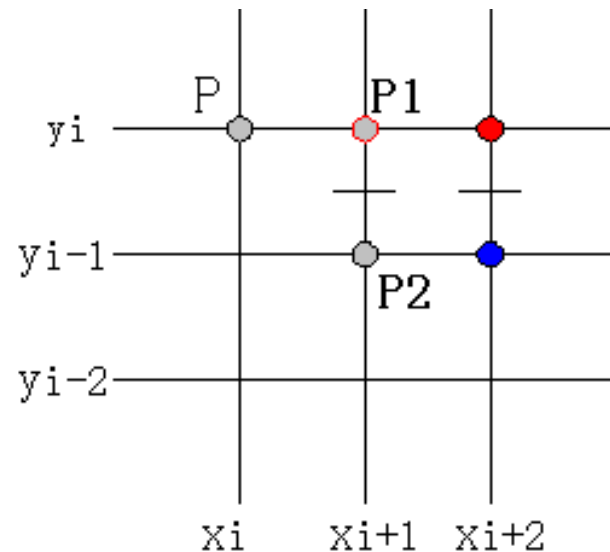
中点画圆法

- ② 构造判别式 $d = F(M) = F(x_p + 1, y_p - 0.5)$
- $$= (x_p + 1)^2 + (y_p - 0.5)^2 - R^2$$

若 $d < 0$, 则 P1 为下一个像素, 那么再下一个像素的判别式为:

$$\begin{aligned} d' &= F(x_p + 2, y_p - 0.5) \\ &= (x_p + 2)^2 + (y_p - 0.5)^2 - R^2 \\ &= d + 2x_p + 3 \end{aligned}$$

即 d 的增量为 $2x_p + 3$



(a) $d \leq 0$ 的情况

中点画圆法

若 $d \geq 0$, 则P2 为下一个像素, 那么再下一个像素的判别式为:

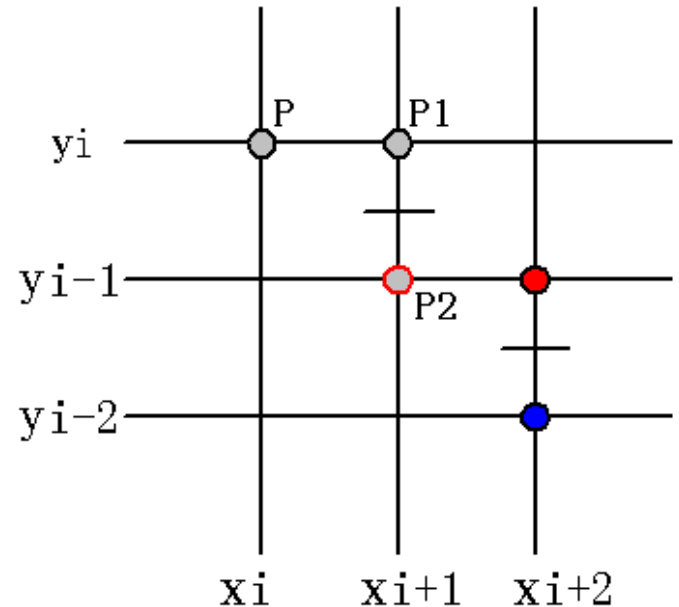
$$\begin{aligned}d' &= F(x_p + 2, y_p - 1.5) \\&= (x_p + 2)^2 + (y_p - 1.5)^2 - R^2 \\&= d + 2(x_p - y_p) + 5\end{aligned}$$

即**d 的增量为 $2(x_p - y_p) + 5$**

③ 计算d的初值:

$$(x_0, y_0) = (0, R)$$

$$\begin{aligned}d_0 &= F(x_0 + 1, y_0 - 0.5) = F(1, R - 0.5) \\&= 1 + (R - 0.5)^2 - R^2 \\&= \mathbf{1.25 - R}\end{aligned}$$



(b) $d > 0$ 的情况

中点画圆法

- 算法步骤：
 1. 输入圆的半径 R 。
 2. 计算初始值 $d=1.25-R$ 、 $x=0$ 、 $y=R$ 。
 3. 绘制点 (x,y) 及其在八分圆中的另外七个对称点。
 4. 判断 d 的符号。若 $d \leq 0$ ，则先将 d 更新为 $d+2x+3$ ，再将 (x,y) 更新为 $(x+1,y)$ ；否则先将 d 更新为 $d+2(x-y)+5$ ，再将 (x,y) 更新为 $(x+1,y-1)$ 。
 5. 当 $x < y$ 时，重复步骤3和4。否则结束。

中点画圆法

- MidpointCircle(r, color)
- int r, color;
- {
- int x,y;
- float d;
- x=0; y=r; d=1.25-r;
- drawpixel(x,y,color);
- while(x<y)
- {
- if(d<0)
- {
- d+ = 2*x+3;
- x++;
- }

else

- {
- d+ = 2*(x-y) + 5;
- x++;y--;
- }

}

中点画圆法

- **改进1：用 $d-0.25$ 代替 d**

令 $e = d - 0.25 \longrightarrow e = 1 - R$

- **算法步骤：**

则 $d < 0 \longrightarrow e < -0.25$

而 e 为整数，则 $e < -0.25$ 等价于 $e < 0$ 。再将 e 仍用 d 来表示

- 1.输入圆的半径 R 。
- 2.计算初始值 $d=1-R$ 、 $x=0$ 、 $y=R$ 。
- 3.绘制点 (x,y) 及其在八分圆中的另外七个对称点。
- 4.判断 d 的符号。若 $d \leq 0$ ，则先将 d 更新为 $d+2x+3$ ，再将 (x,y) 更新为 $(x+1,y)$ ；否则先将 d 更新为 $d+2(x-y)+5$ ，再将 (x,y) 更新为 $(x+1,y-1)$ 。
- 5.当 $x < y$ 时，重复步骤3和4。否则结束。

中点画圆法

- **改进2**：因判别式 d 的增量是 x ， y 的线性函数。
- 每当 x 递增1， d 递增 $\Delta x = 2$ ；
- 每当 y 递增1， d 递减 $\Delta y = -2$ ；
- 由于初始像素为 $(0, r)$ ，所以 Δx 的初值为3， Δy 的初值为 $-2r + 2$ 。再注意到乘2运算可以改用加法实现，至此我们可写出不含乘法，仅用整数实现的中点画圆算法。

中点画圆法

- MidpointCircle(r, color)
- int r, color;
- {
- int x,y, deltax, deltay, d;
- x=0; y=r; d=1-r;
- deltax=3; deltay=2-r-r;
- drawpixel(x,y,color);
- while(x<y)
- {

```
if(d<0)
{
    d+ = deltax;
    deltax+=2; x++;
}
else
{
    d+ = (deltax+deltay);
    deltax+=2; deltay -=2;
    x++; y--;
}
}
```

中点画圆法

例题：画第一象限中，半径 $R = 10$ ，圆心在原点的圆弧

若 $e_0 = 1 - R$

$$e_i < 0, \quad \rightarrow e_{i+1} = e_i + 2x_i + 3$$
$$e_i \geq 0, \quad \rightarrow e_{i+1} = e_i + 2(x_i - y_i) + 5$$

解：起点为 $(x_0, y_0) = (0, 10)$

$$e_0 = 1 - R = -9; \quad (x_1, y_1) = (1, 10)$$

$$e_1 = e_0 + 2x_0 + 3 = -6; \quad (x_2, y_2) = (2, 10)$$

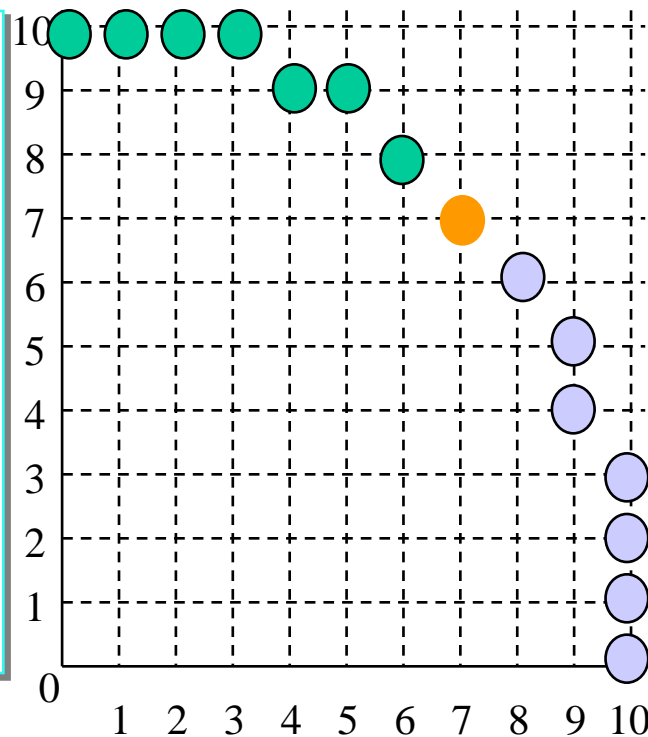
$$e_2 = e_1 + 2x_1 + 3 = -1; \quad (x_3, y_3) = (3, 10)$$

$$e_3 = e_2 + 2x_2 + 3 = 6; \quad (x_4, y_4) = (4, 9)$$

$$e_4 = e_3 + 2(x_3 - y_3) + 5 = -3; \quad (x_5, y_5) = (5, 9)$$

$$e_5 = e_4 + 2x_4 + 3 = 8; \quad (x_6, y_6) = (6, 8)$$

$$e_6 = e_5 + 2(x_5 - y_5) + 5 = 5; \quad (x_7, y_7) = (7, 7)$$



Bresenham画圆法

- 为讨论方便，仅考虑圆心在原点，半径为 R 的第一象限上的一段圆弧。且取 $(0, R)$ 为起点，按顺时针方向绘制该1/4圆弧。

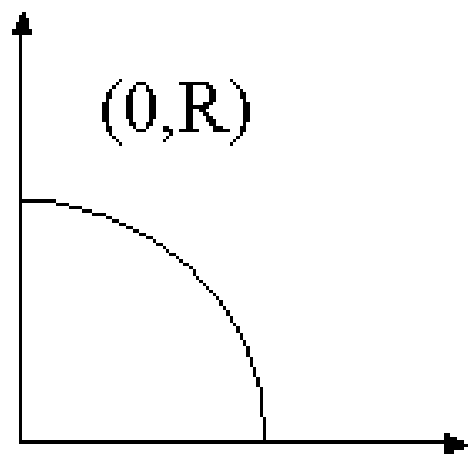


Figure 1

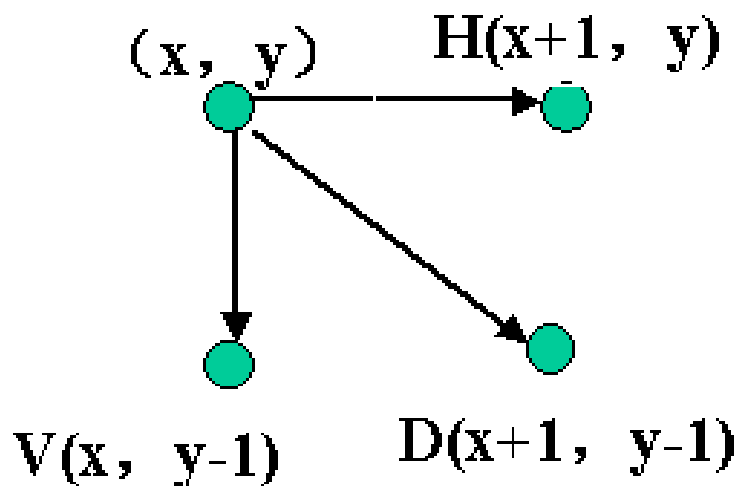


Figure 2

Bresenham画圆法

- **原理：**
- 如图，从当前点亮像素出发，按顺时针方向生成圆时，最佳逼近该圆的下一个像素只可能为H、D、V三像素之一。H、D、V中距圆周边界距离最小者，即为所求的像素点。

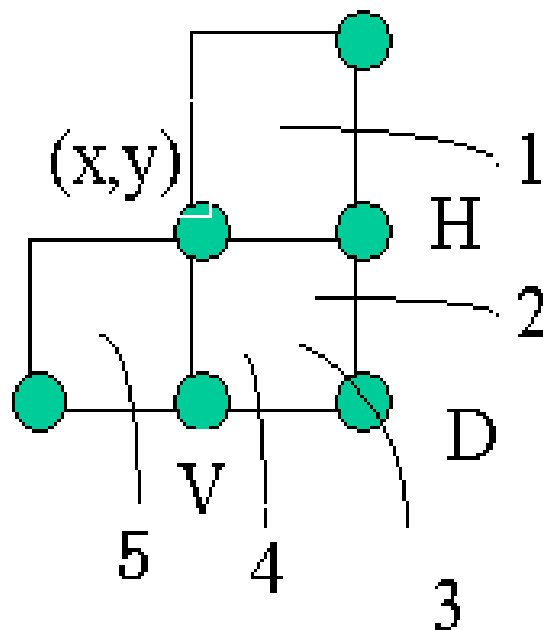


Figure 3

Bresenham画圆法

- 算法：

- H、D、V三点到圆心的距离平方与圆的半径平方差，即为H、D、V到圆弧距离的一种度量：

- $\Delta_H = (x+1)^2 + y^2 - R^2;$
- $\Delta_D = (x+1)^2 + (y-1)^2 - R^2;$
- $\Delta_V = x^2 + (y-1)^2 - R^2;$

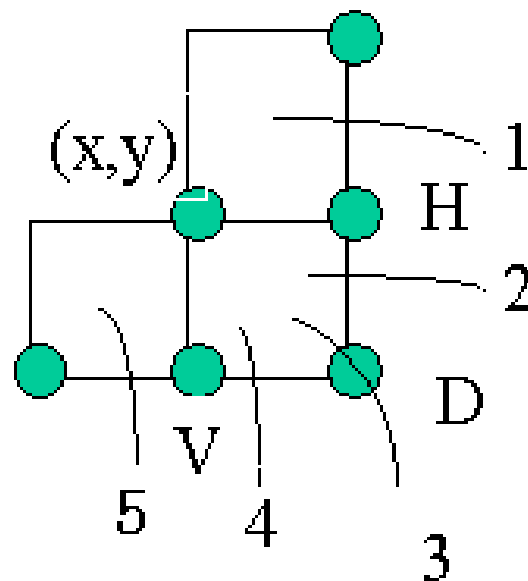


Figure 3

Bresenham画圆法

- **H、D、V与理想圆弧的关系：**
 - 1) H、D、V全在圆内；
 - 2) H在圆外，D、V在圆内；
 - 3) D在圆上，H在圆外，V在圆内；
 - 4) H、D在圆外，V在圆内；
 - 5) H、D、V全在圆外。

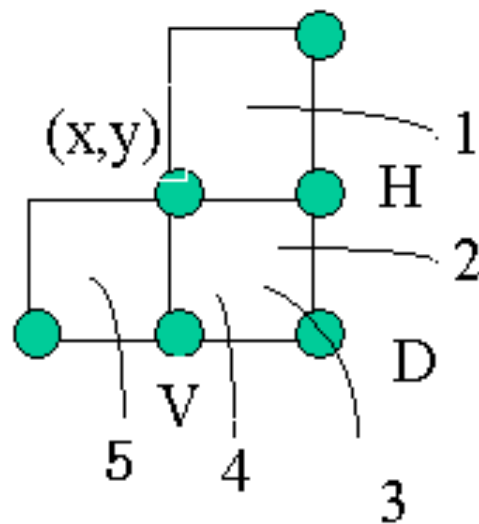


Figure 3

与Bresenham画线算法一样，按照上述不同类型，找出误差度量的递推公式，然后判别它的正、负性即可确定最佳逼近的像素点。

Bresenham画圆法

- 当 $\Delta_D < 0$ ，只可能为1或2种情况。为了确定是H还是D，可用如下判别式：

$$\delta_{HD} = |\Delta_H| - |\Delta_D|$$

$\delta_{HD} \leq 0$ 则应选H，否则选D。

- 1) H、D、V全在圆内；
- 2) H在圆外，D、V在圆内；

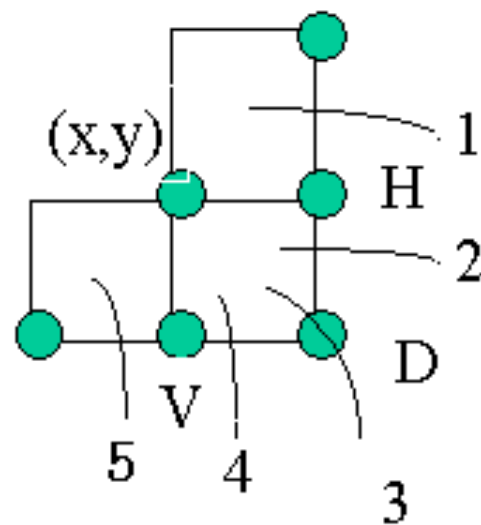


Figure 3

Bresenham画圆法

- 对于第2种情况：
- $\delta_{HD} = |\Delta_H| - |\Delta_D| = \Delta_H + \Delta_D$
- $= (x+1)^2 + y^2 - R^2 + (x+1)^2 + (y-1)^2 - R^2$
- $= 2\Delta_D + 2y - 1$
- 对于第1种情况：
- $\therefore y$ 是 x 的单调递减函数
- $\therefore H$ 为下一点亮象素。
- 另，此时 $\Delta_H < 0$ 和 $\Delta_D < 0$
- $\Delta_H + \Delta_D = 2\Delta_D + 2y - 1 < 0$
- 综上所述两种情况可得如下结论：
- **在 $\Delta_D < 0$ 时，若 $2(\Delta_D + y) - 1 \leq 0$ ，则取H，否则取D**

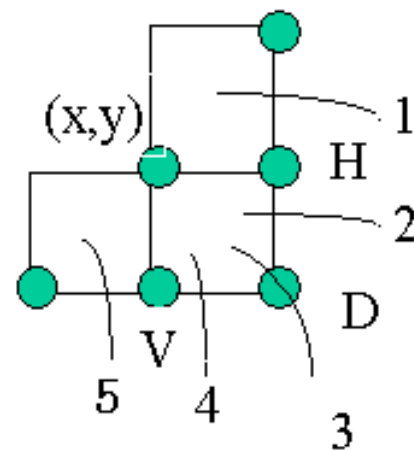


Figure 3

Bresenham画圆法

- 当 $\Delta_D > 0$ ，只可能有4、5两种情况。且最佳象素点为D或V，可用如下判别式：

$$\delta_{DV} = |\Delta_D| - |\Delta_V|$$

$\delta_{DV} \leq 0$ 则应选D，否则选V。

对于第4种情况：

- $\delta_{DV} = \Delta_D + \Delta_V$ ($\Delta_D > 0, \Delta_V < 0$)
- $= (x+1)^2 + (y-1)^2 - R^2 +$
- $(x)^2 + (y-1)^2 - R^2$
- $= 2(\Delta_D - x) - 1$

对于第5种情况：

D,V都在圆外，显然V为所选象素。

注意： $\because \Delta_D > 0, \Delta_V > 0$

$$\therefore \Delta_D + \Delta_V = 2(\Delta_D - x) - 1 > 0$$

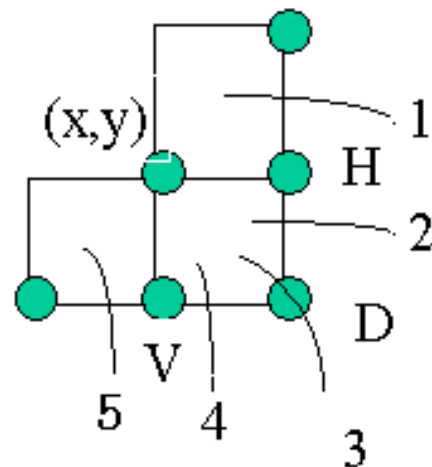


Figure 3

Bresenham画圆法

- 综上两种情况可得如下结论：
- 在 $\Delta_D > 0$ 时，若 $2(\Delta_D - x) - 1 \leq 0$ ，则取D，否则取V
- 当 $\Delta_D = 0$ 此时D是最佳像素。

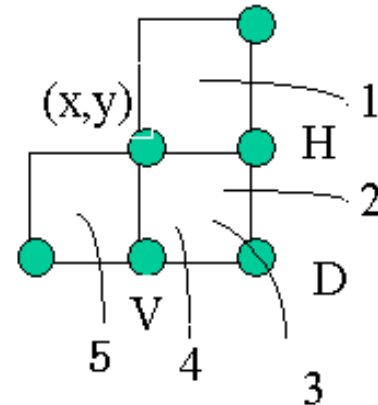


Figure 3

- 总结上述分析结果：
- 当 $\Delta_D > 0$ ，若 $2(\Delta_D - x) - 1 > 0$ ，取V，否则取D
- 当 $\Delta_D < 0$ ，若 $2(\Delta_D + y) - 1 \leq 0$ ，取H，否则取D
- 当 $\Delta_D = 0$ ，取D。

关键的问题就是计算 Δ_D ？

Bresenham画圆法

- **计算 Δ_D :**
- 采用增量法，获得 Δ_D 的计算公式。
- 分三种情况：
- 下一像素为H时,则
- $H=(x', y')=(x+1,y)$
- $\Delta_D' = ((x+1)+1)^2 + (y-1)^2 - R^2$
- $= (x+1)^2 + (y-1)^2 - R^2 + 2(x+1) + 1$
- $= \Delta_D + 2(x+1) + 1$
- $= \Delta_D + 2x' + 1$

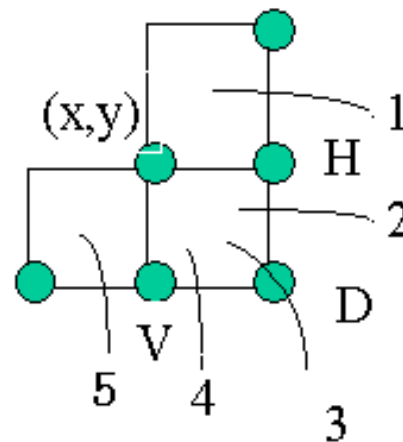


Figure 3

Bresenham画圆法

- 下一像素为D时
- $D=(x',y')=(x+1,y-1)$
- $\Delta_D' = ((x+1)+1)^2 + ((y-1)-1)^2 - R^2$
- $= (x+1)^2 + (y-1)^2 - R^2 + 2(x+1) - 2(y-1) + 1$
- $= \Delta_D + 2(x+1) - 2(y-1) + 2$

- 下一像素为V时
- $V=(x',y')=(x,y-1)$
- $\Delta_D' = (x+1)^2 + ((y-1)-1)^2 - R^2$
- $= (x+1)^2 + (y-1)^2 - R^2 - 2(y-1) + 1$
- $= \Delta_D - 2(y-1) + 1$

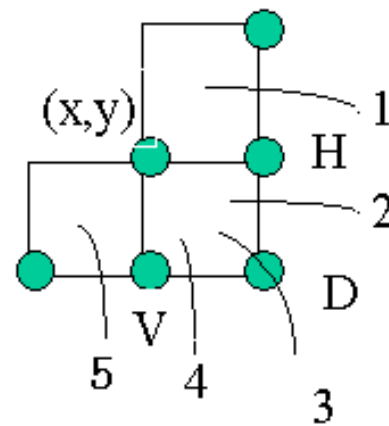


Figure 3

Bresenham画圆法

- 有了上述 Δ_D 的递推计算公式，还需计算出 Δ_D 的初值。
- \therefore 圆弧的起点为 $(0, R)$
- $\therefore \Delta_D$ 的初值为：
- $$\Delta_D = (0+1)^2 + (R-1)^2 - R^2$$
- $$= 2(1-R)$$

Bresenham画圆法

BresenhamCircle(r, color)

```
int r, color;
{  int x,y,delta, d1, d2, dir
    x=0; y=r;
    delta = 2*(1-r)
    while(y>=0){
        drawpixel(x,y,color);
        if(delta < 0){
            d1 = 2* (delta + y) -1;
            if(d1 <=0)  dir = 1;
            else      dir = 2;
        }
        else if( delta > 0){
            d2 = 2*(delta-x)-1;
            if(d2 <= 0) dir = 2;
            else      dir = 3;
        } else
            dir = 2;
```

switch(dir){

case 1:

x++;

delta+=2*x + 1;

break;

case 2:

x++; Y--;

delta+=2*(x-y+1) + 1;

break;

case 3:

y--;

delta+=-2*y + 1;

break;

/*end of switch*/

/*end of while*/

/*end of BresenhamCircle*/

椭圆的扫描转换

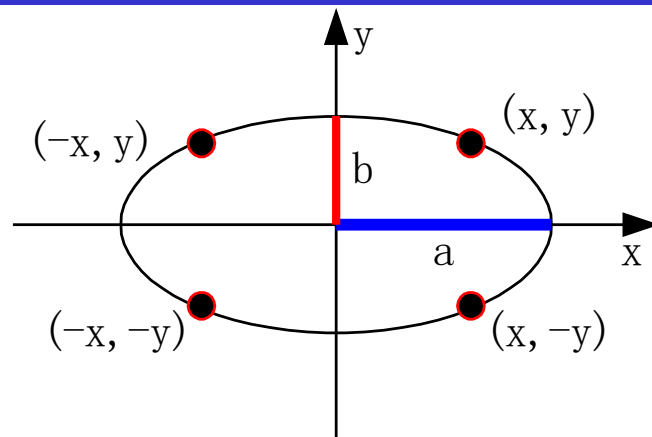


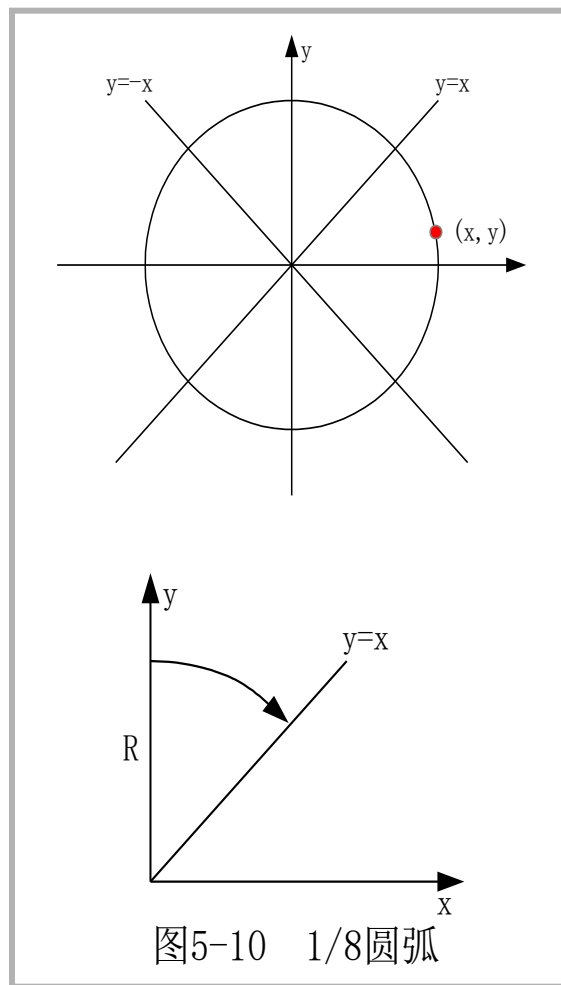
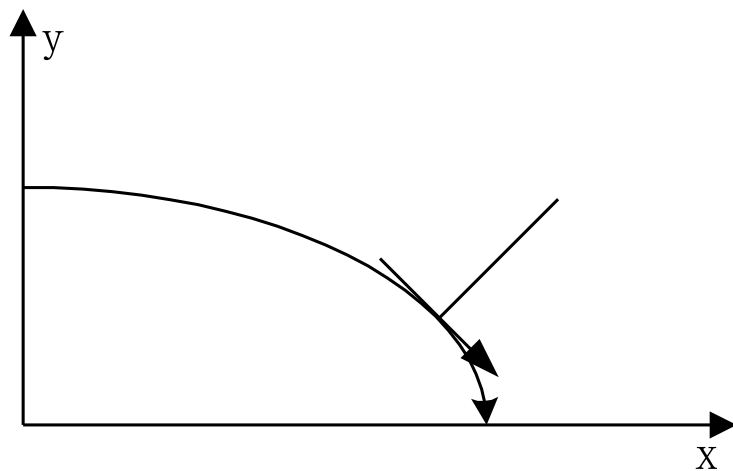
图5-14 长半轴为a，短半轴为b的标准椭圆

$$F(x, y) = b^2 x^2 + a^2 y^2 - a^2 b^2 = 0$$

- ❖ 对于椭圆上的点，有 $F(x, y) = 0$ ；
- ❖ 对于椭圆外的点， $F(x, y) > 0$ ；
- ❖ 对于椭圆内的点， $F(x, y) < 0$ 。

椭圆的扫描转换

解决问题：



椭圆的扫描转换

椭圆方程

$$F(x, y) = b^2 x^2 + a^2 y^2 - a^2 b^2 = 0$$

法向量

$$N(x, y) = \frac{\partial F}{\partial x} i + \frac{\partial F}{\partial y} j = 2b^2 x i + 2a^2 y j$$

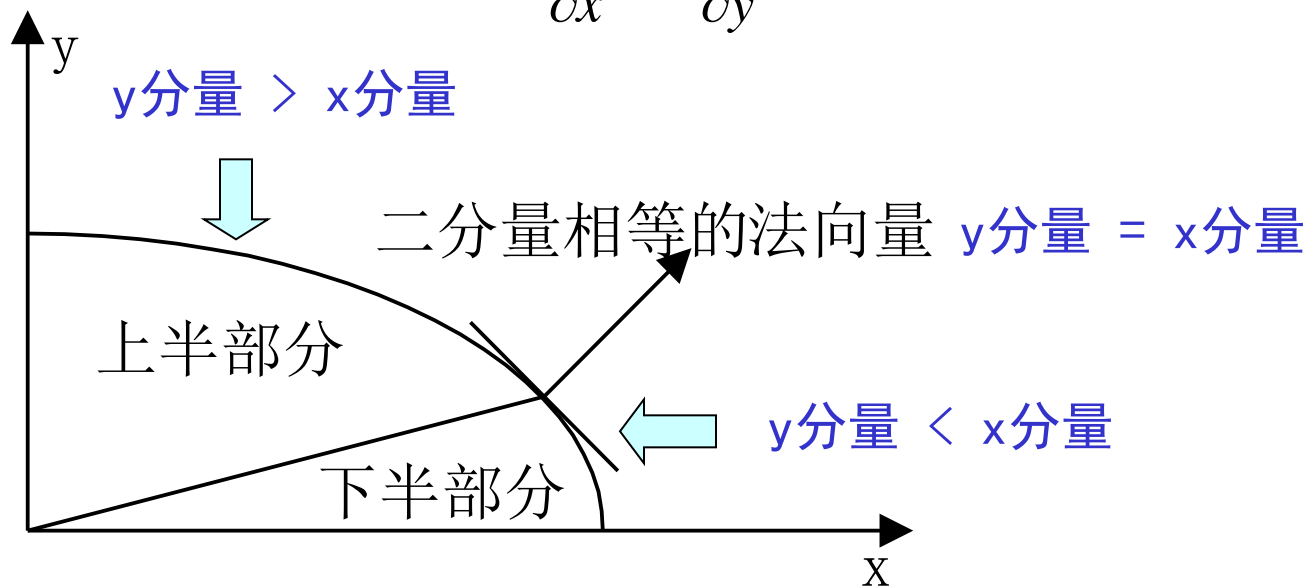


图5-15 第一象限的椭圆弧

分界点：法向量两分量相等的点，以弧上斜率为 - 1 的点

椭圆的扫描转换

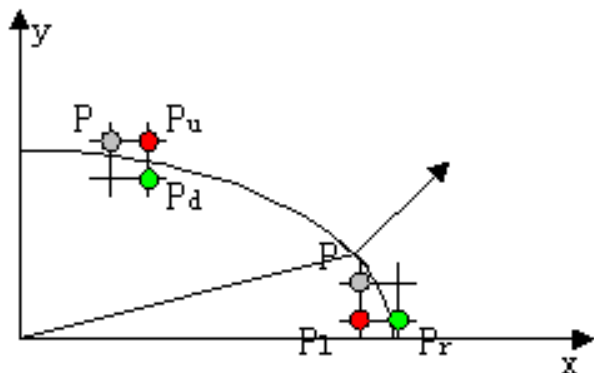


图5-16 中点椭圆绘制算法的原理

当前点 $P(x_i, y_i)$
 候选点 $P_u(x_i+1, y_i)$
 $P_d(x_i+1, y_i-1)$
 中点 $(x_i+1, y_i-0.5)$

- 引理3-1：若在当前中点，法向量的y分量比x分量大，即

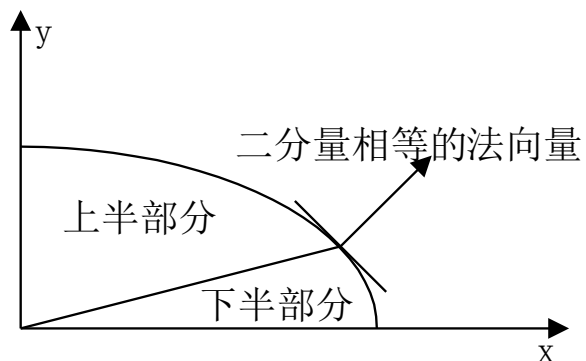


图5-15 第一象限的椭圆弧

$$b^2(x_i + 1) < a^2(y_i - 0.5)$$

而在下一个中点，不等号改变方向，则说明椭圆弧从上部分转入下部分。

中点画椭圆算法

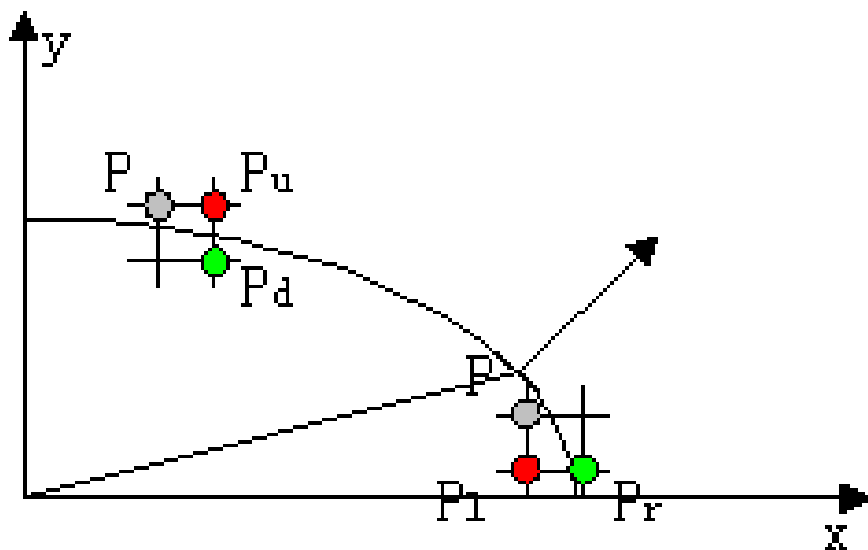


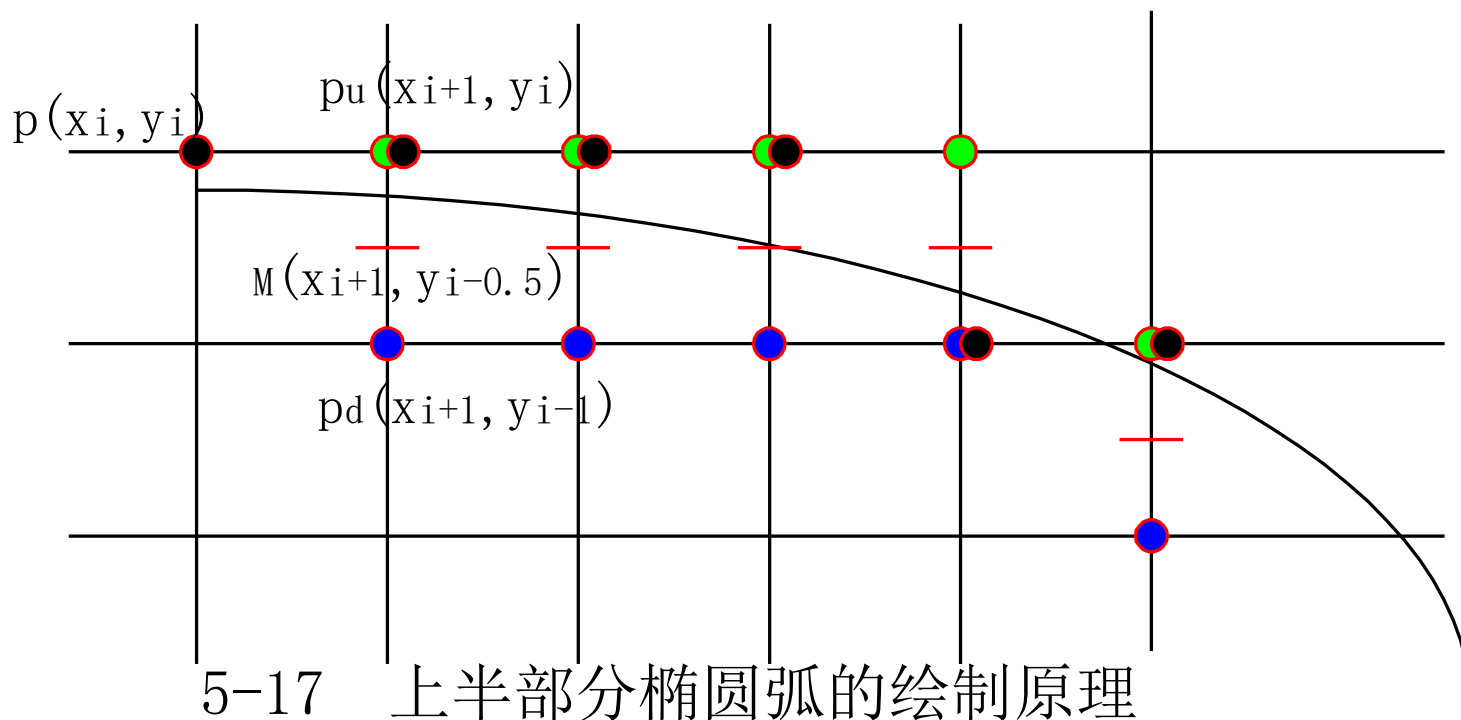
图5-16 中点椭圆绘制算法的原理

先推导上半部分的椭圆绘制公式

再推导下半部分的椭圆绘制公式

中点画椭圆算法

- 先推导上半部分的椭圆绘制公式
- 判别式 误差项的递推 判别式的初值



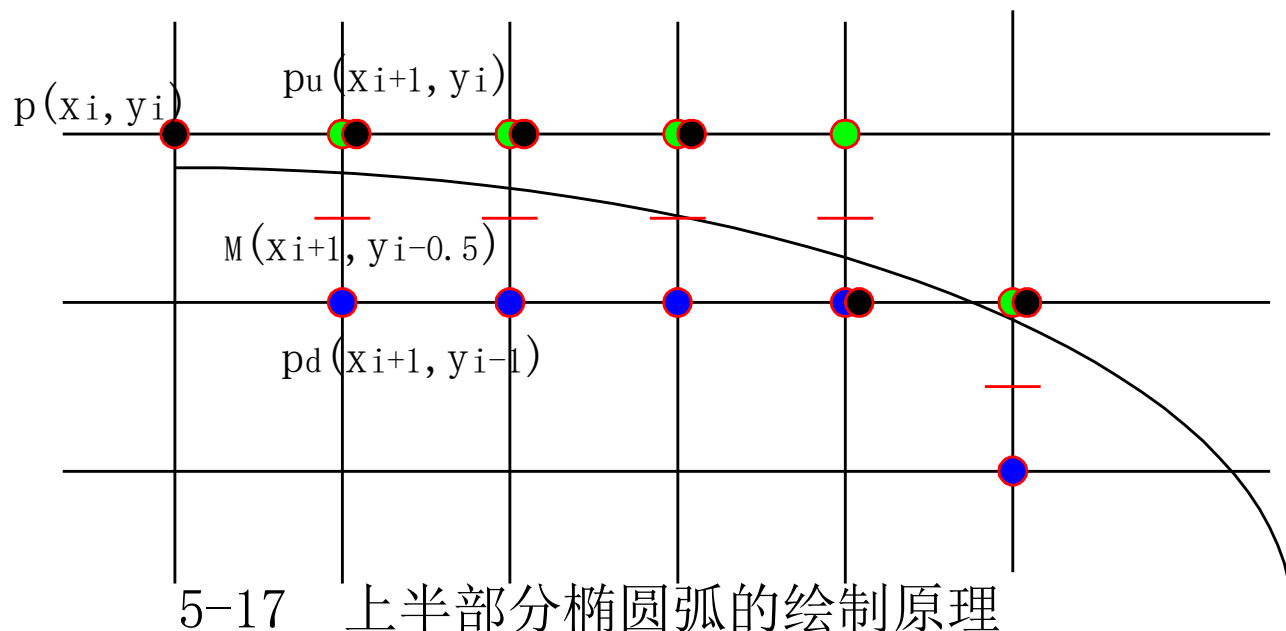
中点画椭圆算法

• 判别式

$$d_1 = F(x_i + 1, y_i - 0.5) = b^2(x_i + 1)^2 + a^2(y_i - 0.5)^2 - a^2b^2$$

❖ 若 $d_1 \leq 0$, 取 $P_u(x_{i+1}, y_i)$

❖ 若 $d_1 > 0$, 取 $P_d(x_{i+1}, y_{i-1})$



中点画椭圆算法

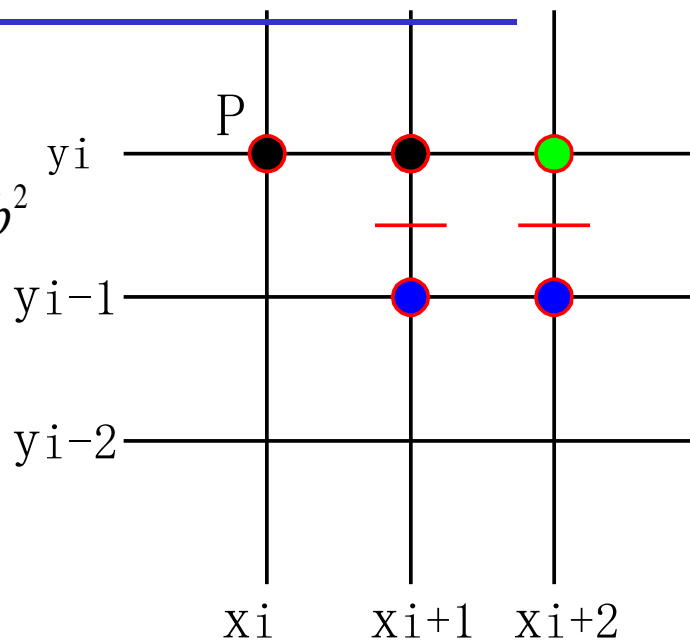
• 误差项的递推

$$d_1 = F(x_i + 1, y_i - 0.5) = b^2(x_i + 1)^2 + a^2(y_i - 0.5)^2 - a^2b^2$$

情况一：

$$d_1 \leq 0 :$$

$$\begin{aligned} d_1 &= F(x_i + 2, y_i - 0.5) = b^2(x_i + 2)^2 + a^2(y_i - 0.5)^2 - a^2b^2 \\ &= b^2(x_i + 1)^2 + a^2(y_i - 0.5)^2 - a^2b^2 + b^2(2x_i + 3) \\ &= d_1 + b^2(2x_i + 3) \end{aligned}$$



(a) $d \leq 0$ 的情况

中点画椭圆算法

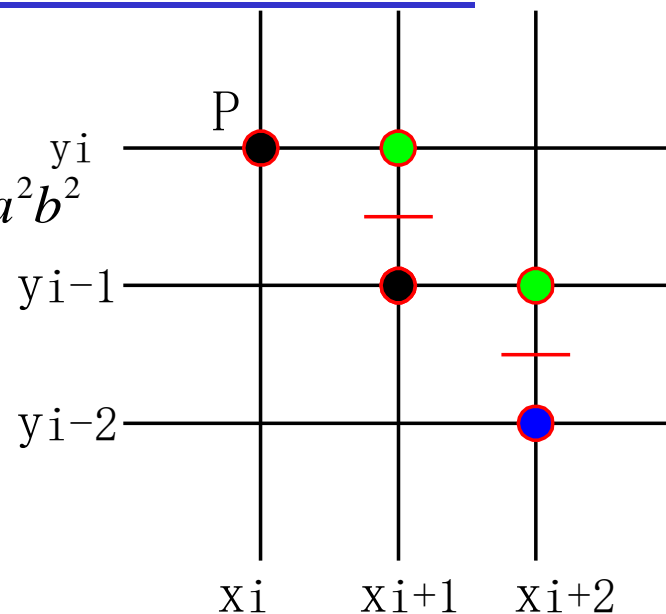
误差项的递推

$$d_1 = F(x_i + 1, y_i - 0.5) = b^2(x_i + 1)^2 + a^2(y_i - 0.5)^2 - a^2b^2$$

情况二：

$d_1 > 0$ ：

$$\begin{aligned} d_1 &= F(x_i + 2, y_i - 1.5) = b^2(x_i + 2)^2 + a^2(y_i - 1.5)^2 - a^2b^2 \\ &= b^2(x_i + 1)^2 + a^2(y_i - 0.5)^2 - a^2b^2 + b^2(2x_i + 3) + a^2(-2y_i + 2) \\ &= d_1 + b^2(2x_i + 3) + a^2(-2y_i + 2) \end{aligned}$$



(b) $d > 0$ 的情况

中点画椭圆算法

上半部判别式的初始值：

初始点 (0 , b)

候选点 (1 , b-1) (1 , b)

中 点 (1 , b-0.5)

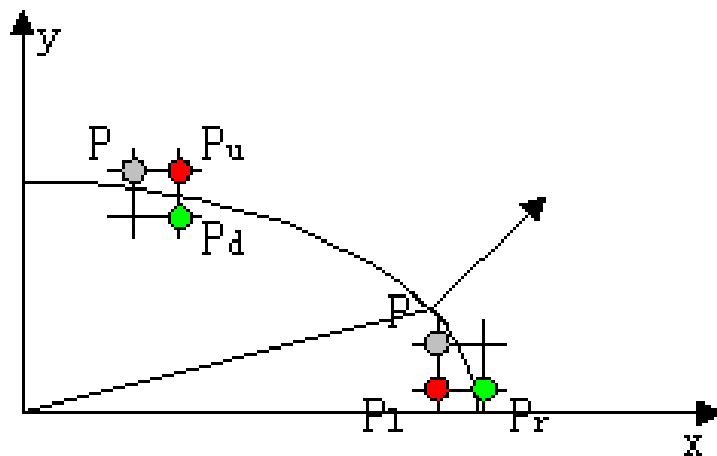


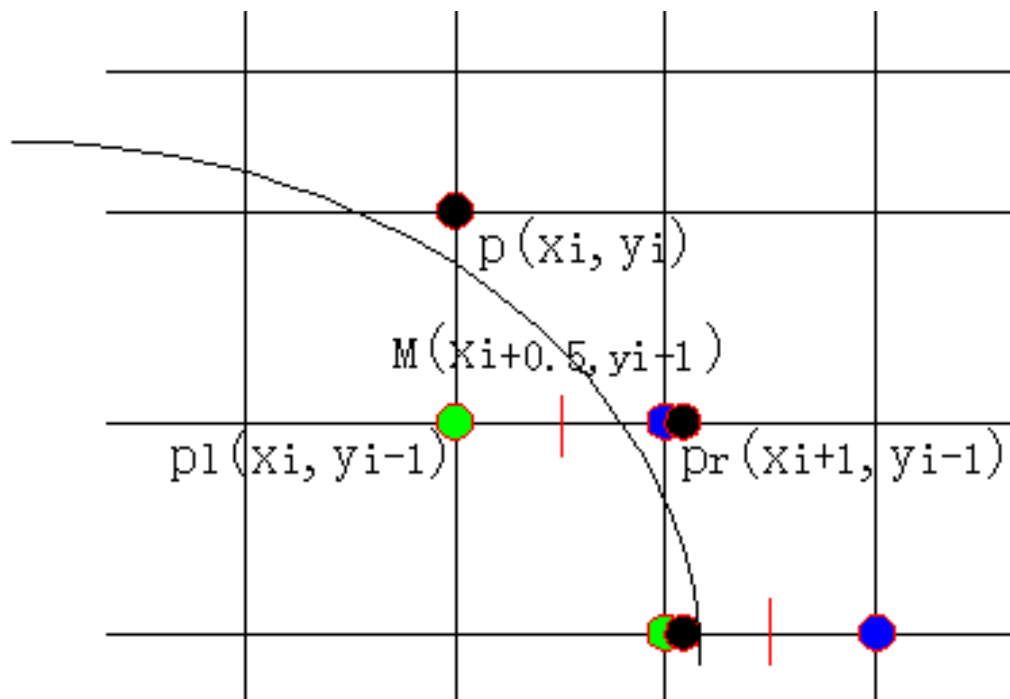
图5-16 中点椭圆绘制算法的原理

$$\begin{aligned}d_{10} &= F(1, b - 0.5) = b^2 + a^2(b - 0.5)^2 - a^2b^2 \\ &= b^2 + a^2(-b + 0.25)\end{aligned}$$

中点画椭圆算法

再来推导椭圆弧下半部分的绘制公式

判别式 误差项的递推 判别式的初值



5-19 下半部分椭圆弧的绘制原理

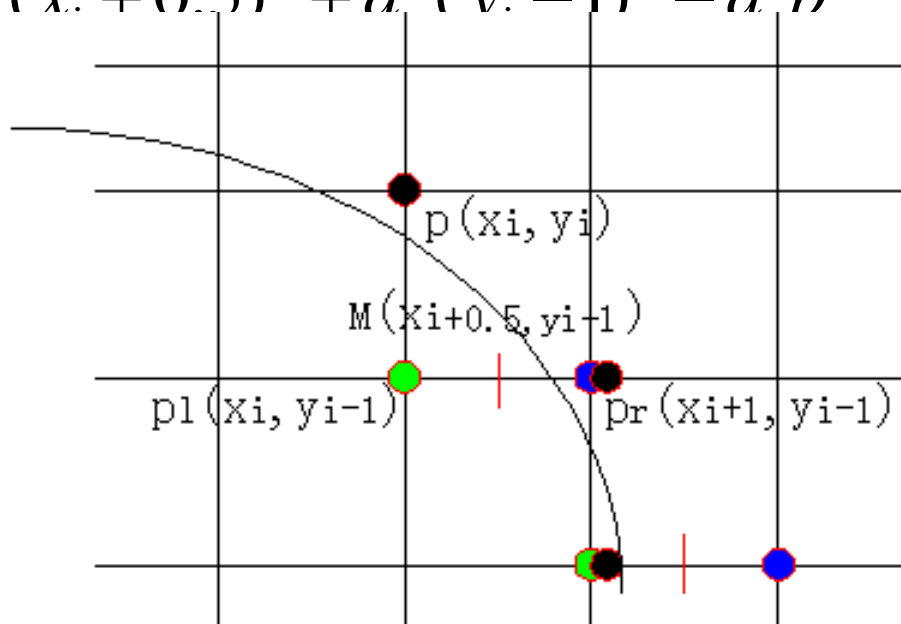
中点画椭圆算法

- 判别式

$$d_2 = F(x_i + 0.5, y_i - 1) = b^2(x_i + 0.5)^2 + a^2(y_i - 1)^2 - a^2b^2$$

❖ 若 $d_2 > 0$, 取 $P_l(x_i, y_i - 1)$

❖ 若 $d_2 \leq 0$, 取 $P_r(x_i + 1, y_i - 1)$



5-19 下半部分椭圆弧的绘制原理

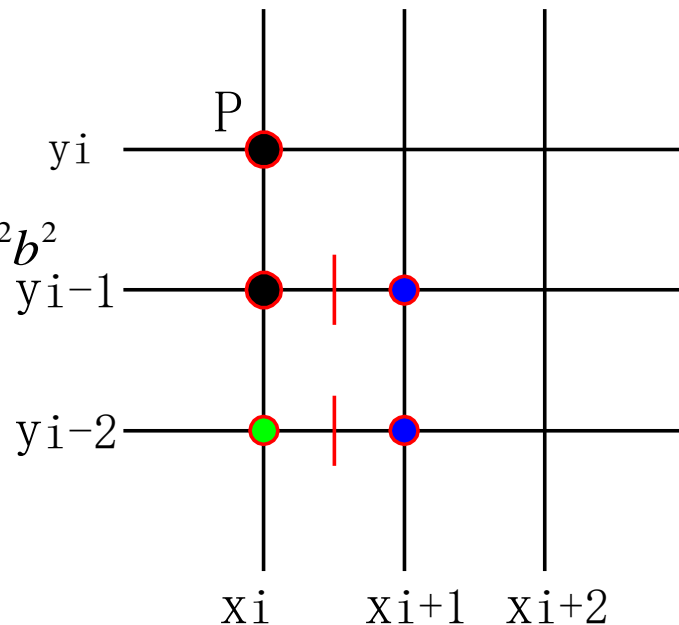
中点画椭圆算法

- 误差项的递推

$$d_2 = F(x_i + 0.5, y_i - 1) = b^2(x_i + 0.5)^2 + a^2(y_i - 1)^2 - a^2b^2$$

第一种情况：

$d_2 > 0$ ：



(a) $d > 0$ 的情况

$$\begin{aligned} d_2 &= F(x_i + 0.5, y_i - 2) = b^2(x_i + 0.5)^2 + a^2(y_i - 2)^2 - a^2b^2 \\ &= b^2(x_i + 0.5)^2 + a^2(y_i - 1)^2 - a^2b^2 + a^2(-2y_i + 3) \\ &= d_2 + a^2(-2y_i + 3) \end{aligned}$$

中点画椭圆算法

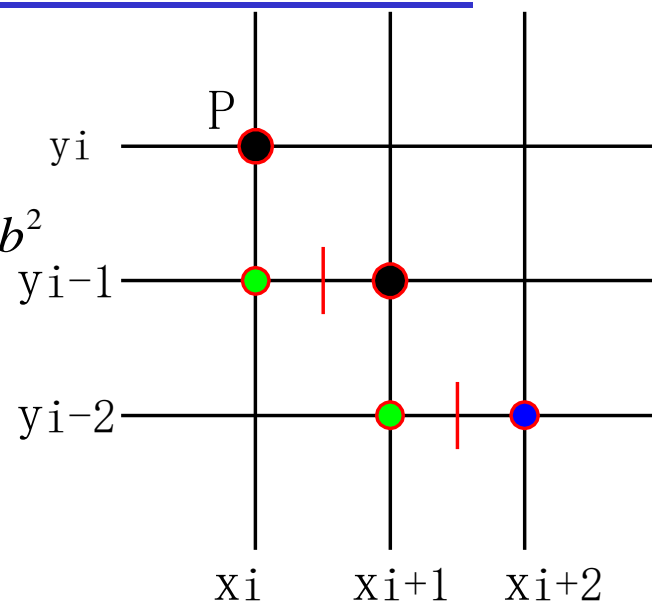
误差项的递推

$$d_2 = F(x_i + 0.5, y_i - 1) = b^2(x_i + 0.5)^2 + a^2(y_i - 1)^2 - a^2b^2$$

第二种情况：

$$d_2 \leq 0 :$$

$$\begin{aligned} d_2 &= F(x_i + 1.5, y_i - 2) = b^2(x_i + 1.5)^2 + a^2(y_i - 2)^2 - a^2b^2 \\ &= b^2(x_i + 0.5)^2 + a^2(y_i - 1)^2 - a^2b^2 + b^2(2x_i + 2) + a^2(-2y_i + 3) \\ &= d_2 + b^2(2x_i + 2) + a^2(-2y_i + 3) \end{aligned}$$



(b) $d \leq 0$ 的情况

中点画椭圆算法

注意：

- ◆ 上半部分的终止判别

$$dy/dx = -2b^2x/2a^2y, \quad dy/dx = -1$$

$$2b^2x \geq 2a^2y$$

- ◆ 下半部分误差项的初值

$$d_{20} = F(x_0 + 0.5, y_0) = b^2(a + 0.5)^2 + a^2(y_0 - 1)^2 - a^2b^2$$

或从 $(a, 0)$ 开始选择像素位置

中点画椭圆算法

■ 算法步骤：

1. 输入椭圆的长半轴 a 和短半轴 b 。
2. 计算初始值 $d = b^2 + a^2(-b + 0.25)$ 、 $x = 0$ 、 $y = b$ 。
3. 绘制点 (x, y) 及其在四分象限上的另外三个对称点。
4. 判断 d 的符号。若 $d \leq 0$ ，则先将 d 更新为 $d + b^2(2x + 3)$ ，再将 (x, y) 更新为 $(x + 1, y)$ ；否则先将 d 更新为 $d + b^2(2x + 3) + a^2(-2y + 2)$ ，再将 (x, y) 更新为 $(x + 1, y - 1)$ 。
5. 当 $b^2(x + 1) < a^2(y - 0.5)$ 时，重复步骤3和4。否则转到步骤6。

中点画椭圆算法

6. 用上半部分计算的最后点 (x,y) 来计算下半部分中 d 的初

值：
$$d = b^2(x + 0.5)^2 + a^2(y - 1)^2 - a^2b^2$$

7. 绘制点 (x,y) 及其在四分象限上的另外三个对称点。

8. 判断 d 的符号。若 $d \leq 0$ ，则先将 d 更新为 $b^2(2x_i + 2) + a^2(-2y_i + 3)$ ，再将 (x,y) 更新为 $(x+1, y-1)$ ；否则先将 d 更新为 $d + a^2(-2y_i + 3)$ ，再将 (x,y) 更新为 $(x, y-1)$ 。

9. 当 $y > 0$ 时，重复步骤7和8。否则结束。

中点画椭圆算法

MidpointEllipse (a, b, color) // 部分代码

```
int a, b, color;
{
    int x,y;
    float d1, d2;
    x=0; y=b;
    d1=b*b+a*(-b+0.25);
    drawpixel(x,y,color);
    while(b*b*(x+1) < a*a*(y-0.5) )
    { if(d1 < 0)
        { d1 += b*b* (2*x+3);
          x++;
        }
      else
        { d1 += b*b* (2*x+3) +
          a*a*(-2*y+2));
          x++;y--;
        }
      drawpixel(x,y,color);
    }
}
```

思考

1、试阐述中点画圆算法与中点画椭圆算法有何区别？

谢 谢 ！