二维线画图元的生成

刘世光

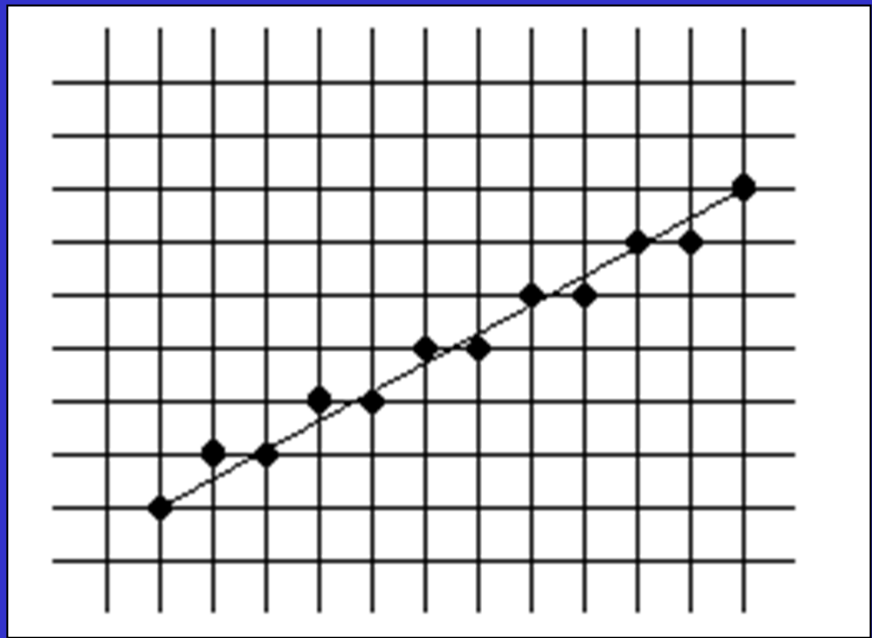
天津大学计算机学院

二维线画图元的生成

- 直线段的生成
- （DDA画线算法，Bresenham画线算法）
- 圆的生成
- （中点画圆算法）
- 椭圆的生成

直线生成算法

- 直线生成: 求与直线段充分接近的像素集



当直线作为一系列像素位置生成时产生的阶梯效果

直线方程

- 直线的斜率截矩方程:

$$y = m \bullet x + B$$

- 给定线段的两个端点 (x_0, y_0) (x_{end}, y_{end})，可以计算斜率**m**和**y**轴截矩**b:**

$$m = \frac{y_{end} - y_0}{x_{end} - x_0} \qquad b = y_0 - m \bullet x_0$$

直线方程

- 对于任何沿直线给定的**x**增量 δ_x ,对应的**y**增量 δ_y 为:

$$\delta_y = m \bullet \delta_x$$

同样,对应于指定的 δ_y 的**x**增量 δ_x 为:

$$\delta_x = \frac{\delta_y}{m}$$

DDA算法

- 数字微分分析仪算法(Digital Differential Analyzer)
  - 条件：
    - 待扫描转换的直线段：$P_0(x0, y0)P_1(x1, y1)$
    - 斜率：$m = \Delta y / \Delta x$

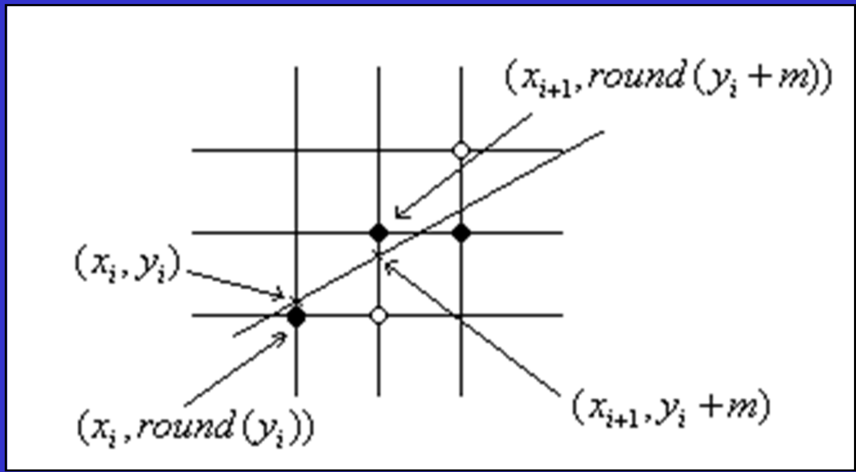
      $\Delta x = x1 - x0, \Delta y = y1 - y0$
    - 直线方程：$y = m \bullet x + B$，$0 < m < 1$
  - 算法步骤：
    - 划分区间**[x0,x1]**：$x_0, x_1, \cdots, x_n$, 其中$x_{i+1} = x_i + 1$
    - 计算纵坐标：$y_{i+1} = m \bullet x_{i+1} + B = m \bullet (x_i + 1) + B$

      $= m \bullet x_i + B + m = y_i + m$

DDA算法

- 取整：

$$y_{i+1} = round(y_{i+1}) = (\text{int})(y_i + m + 0.5)$$



– 算法复杂度: 加法+取整
– 比较: 直接求交算法

$$y_i = m \bullet x_i + B$$

$$\{(x_i, y_i)\}_{i=0}^{n} \longrightarrow \{(x_i, y_{i,r})\}_{i=0}^{n}$$

$$y_{i,r} = round(y_i) = (\text{int})(y_i + 0.5)$$

– 算法复杂度: 乘法+加法+取整

DDA算法

- 其他斜率情况:

$m > 1$ 交换**x**和**y**的位置

$m < 0$ 步长**dx**或**dy**取**-1**

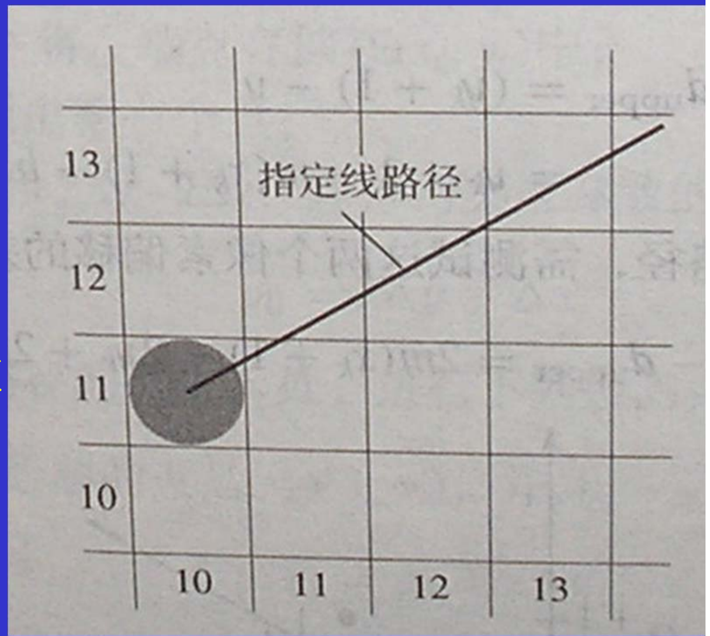
不足之处：取整操作和浮点运算仍十分耗时

Bresenham画线算法

- 由**Bresenham**提出的一种精确而有效的光栅线生成算法，该算法仅仅使用增量整数计算，可用于显示圆和其他曲线。

基本原理：

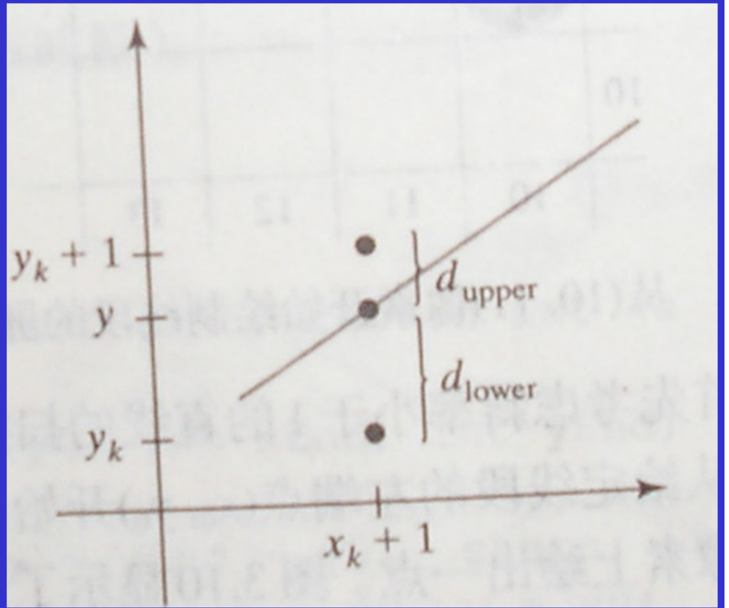
从给定线段的左端点开始，逐步处理每个后继列，并在其扫描线y值最接近线段的像素上绘出一点。



Bresenham画线算法

算法分析：

在取样位置 x_k+1，使用 d_{lower}, d_{upper} 来标识两个像素与数学路径上的垂直偏移。



$$y = m(x_k+1)+b$$

$$d_{lower} = y - y_k = m(x_k+1)+b-y_k$$

$$d_{upper} = (y_k+1)-y = y_k+1-m(x_k+1)-b$$

Bresenham画线算法

决策函数：

$$p_k = \Delta x(d_{lower} - d_{upper})$$
$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

$$p_{k+1} - p_k = 2\Delta y \cdot (x_{k+1} - x_k) - 2\Delta x \cdot (y_{k+1} - y_k)$$

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x \cdot (y_{k+1} - y_k)$$

取值0或1

$$p_0 = 2\Delta y - \Delta x$$

常量 $2\Delta y$ 和 $2\Delta y - 2\Delta x$ 对每条直线只计算一次，因此该系统仅进行两个常量之间的整数加减法。

Bresenham画线算法

- $|m|<1$ 时的Bresenhan算法流程：

1、输入线段的两个端点，将左端点存储在 (x_0, y_0) 中；

2、将 (x_0, y_0) 装入帧缓存，画出第一个点；

3、计算常量 Δx、Δy、$2\Delta y$、$2\Delta y-2\Delta x$，并得到决策函数的第一个值；

4、从k=0开始，在沿线路径的每个 x_k 处，进行决策函数检测；

5、重复步骤4，共 $\Delta x-1$ 次。

圆生成算法

- 圆的数学方程：

$$(x-x_c)^2+(y-y_c)^2=r^2$$

沿**x**轴从 x_c-r 到 x_c+r 以单位步长计算对应的**y**值，从而得到圆周上每点的位置：

$$y=y_c\pm\sqrt{r^2-(x_c-x)^2}$$

计算量大，所画像素的间距不一致

圆生成算法

- 圆的极坐标方程:

$$x = x_c + r\cos\theta$$
$$y = y_c + r\sin\theta$$

可以消除所画像素的间距不一致性，但三角函数的计算十分耗时。

圆生成算法

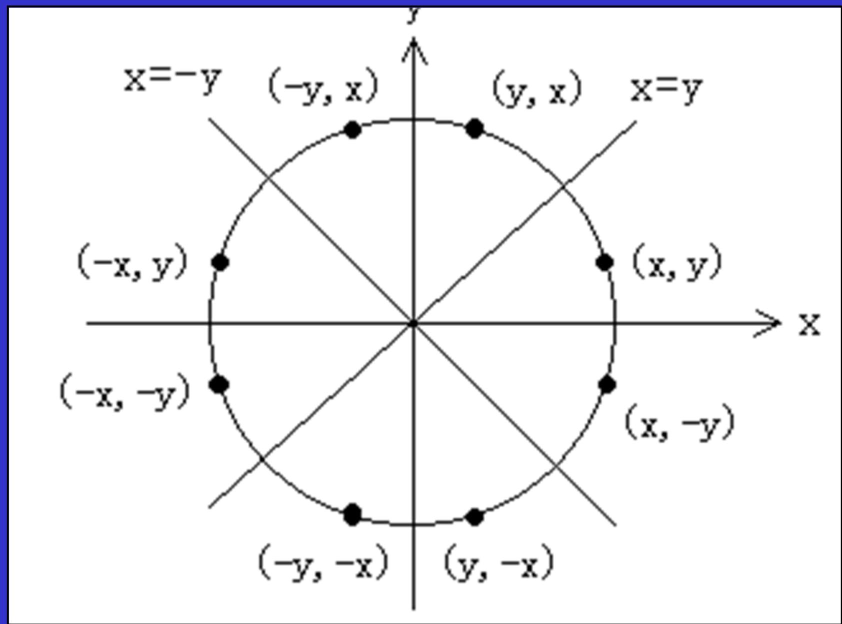
- **Bresenham画圆算法**

通过设定在每一取样步骤中寻找最接近圆周像素的决策函数，可以将**Bresenham**画线算法移植为画圆算法。

圆方程是非线性的。

圆生成算法

策略：考虑圆的对称性；尽量避免平方根、三角函数等复杂运算



中点画圆算法

- 基本思想：

  检测两像素间的中间位置以确定该中点是在圆边界之内还是之外。

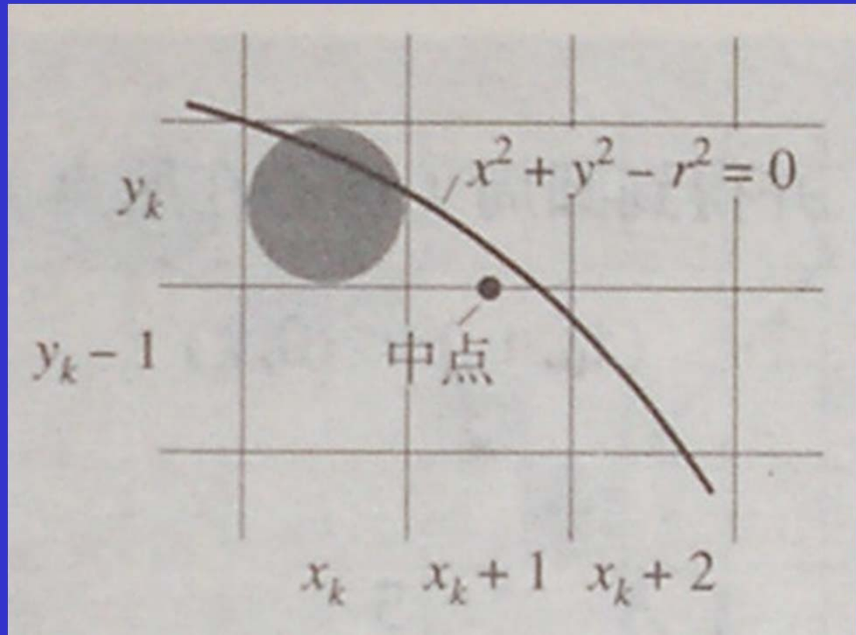
中点画圆算法

- 算法分析：

采用圆函数决策函数：

$$f_{circ}(x, y) = x^2 + y^2 - r^2$$

$$p_k = f_{circ}(x_k + 1, y_k - \frac{1}{2})$$



两个候选像素的中点，**pk<0时取yk, 反之取yk-1**

中点画圆算法

- 算法分析：

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

y_{k+1} 是**y_k（p_k<0时）或者y_k-1，增量是** $2x_{k+1}+$**或者**

$2x_{k+1}+1-2y_{k+1}$

$$p_0 = f_{circ}(1, r - \frac{1}{2}) = \frac{5}{4} - r$$

中点画圆算法

- 算法步骤：

1、输入圆半径和圆心，并得到圆周（圆心在原点）上的第一个点；

2、计算决策函数的第一个值；

3、在每个x_k位置，完成决策函数测试；

4、确定在其他七个八分圆中的对称点；

5、将每个计算出的像素位置移动到圆心在(x_c, y_c)的圆路径上，并画坐标值；

6、重复步骤3~5，直至 x>=y。

椭圆生成算法

- 椭圆可以看作经过修改的圆，**Bresenham**算法和中点算法也适用于椭圆生成。