计算机图形学

第二章：光栅图形学算法

随着光栅显示器的出现，为了在计算机上处理、显示图形，需要发展一套与之相适应的算法：

光栅图形学算法

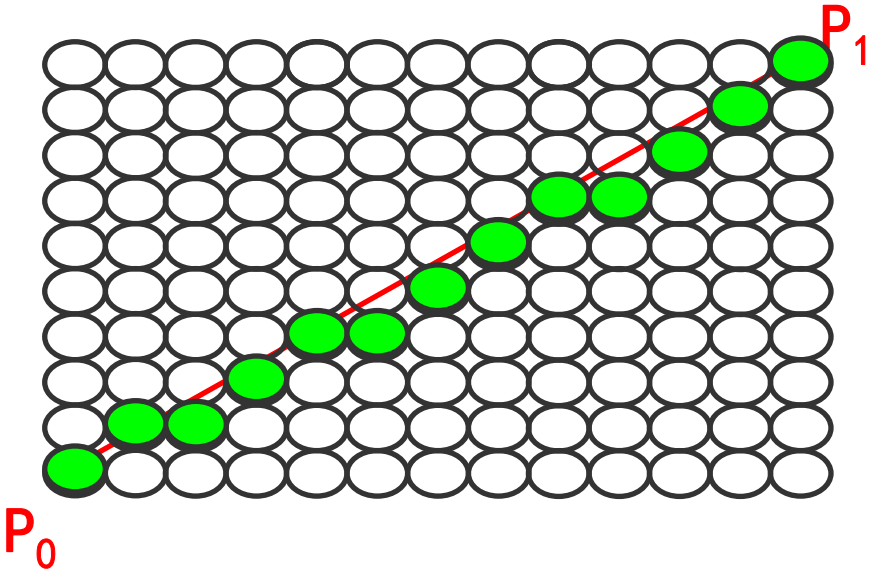
光栅图形算法多数属于计算机图形的底层算法，很多图形学的基本概念和思想都在这一章

光栅图形学算法的研究内容

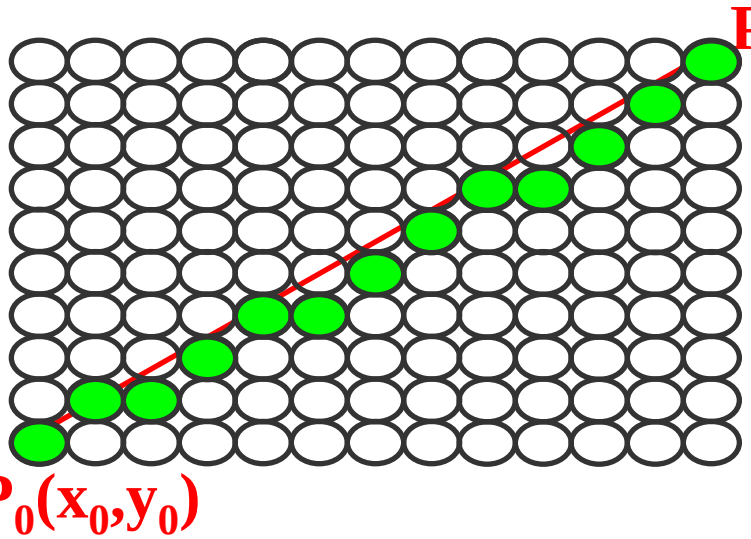
- 直线段的扫描转换算法
- 多边形的扫描转换与区域填充算法
- 裁剪算法
- 反走样算法
- 消隐算法

一、直线段的扫描转换算法

在数学上，直线上的点有无穷多个。但当在计算机光栅显示器屏幕上表示这条直线时需要做一些处理。



为了在光栅显示器上用这些离散的像素点逼近这条直线，需要知道这些像素点的x，y坐标。



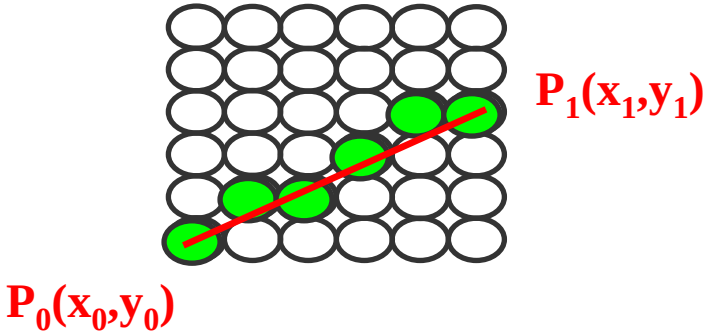
求出过P_0, P_1的直线段方程：

$$y = kx + b$$

$$k = \frac{(y_1 - y_0)}{(x_1 - x_0)} \qquad (x_1 \neq x_0)$$

$$y = kx + b \qquad k = \frac{(y_1 - y_0)}{(x_1 - x_0)} \qquad (x_1 \neq x_0)$$

假设x已知，即从x的起点x_0开始，沿x方向前进一个像素（步长 = 1），可以计算出相应的y值。



因为像素的坐标是整数，所以y值还要进行取整处理

如何把数学上的一个点扫描转换一个屏幕像素点?

如：p（1.7，0.8） —取整→ p（1，0）

p（1.7，0.8） —+0.5→ p（2.2，1.3）

p（2.2，1.3） —取整→ p（2，1）

$$y = kx + b$$

直线是最基本的图形，一个动画或真实感图形往往需要调用成千上万次画线程序，因此直线算法的好坏与效率将直接影响图形的质量和显示速度。

回顾一下刚才的算法：

$$y = kx + b$$

为了提高效率，把计算量减下来，关键问题就是如何把**乘法**取消？

二、直线绘制的三个著名的常用算法

1、数值微分法（DDA）

2、中点画线法

3、Bresenham算法

1、数值微分DDA(Digital Differential Analyzer)法

引进图形学中一个很重要的思想---增量思想

$$y_i = kx_i + b$$

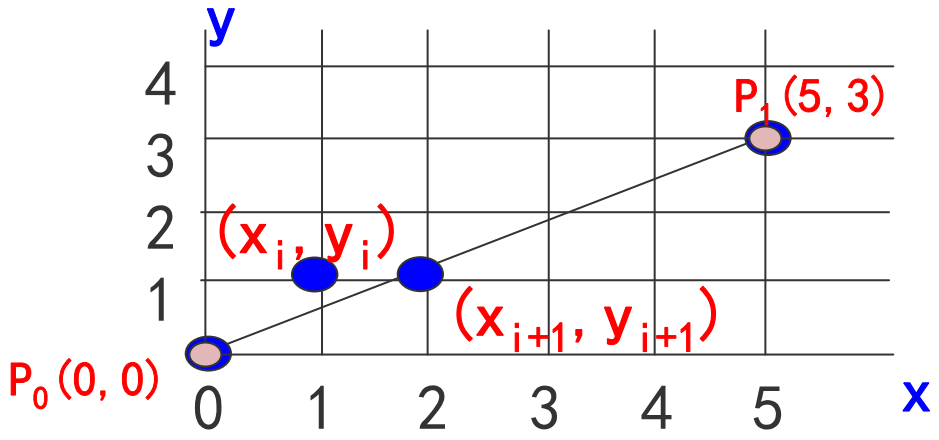
$$y_{i+1} = kx_{i+1} + b$$

$$= k(x_i + 1) + b$$

$$= kx_i + k + b$$

$$= kx_i + b + k$$

$$= y_i + k$$



$$y_{i+1} = y_i + k$$

$$y_{i+1} = y_i + k$$

增量

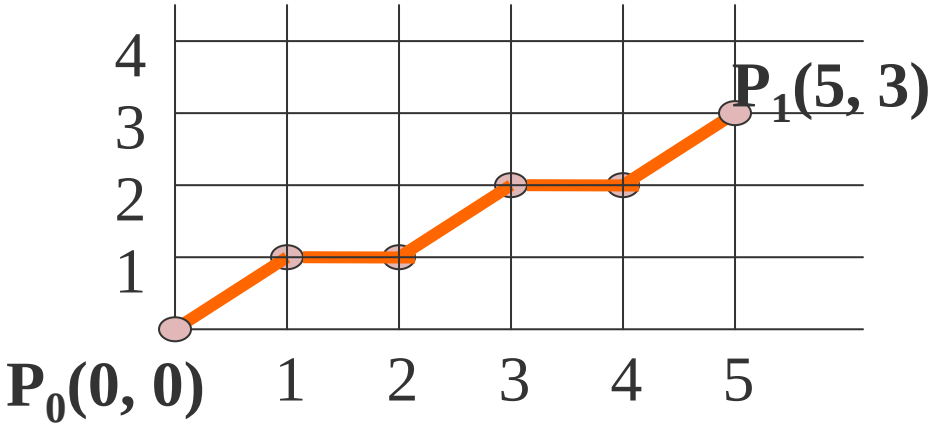
这个式子的含义是：当前步的y值等于前一步的y值加上斜率k

这样就把原来一个乘法和加法变成了一个加法！

用DDA扫描转换连接两点$P_0(0,0)$和$P_1(5,3)$的直线段。

$$k=\frac{y_1-y_0}{x_1-x_0}=\frac{3-0}{5-0}=0.6<1 \qquad y_{i+1}=y_i+k$$

x	y	int(y+0.5)
0	0	0
1	0+0.6	1
2	0.6+0.6	1
3	1.2+0.6	2
4	1.8+0.6	2
5	2.4+0.6	3



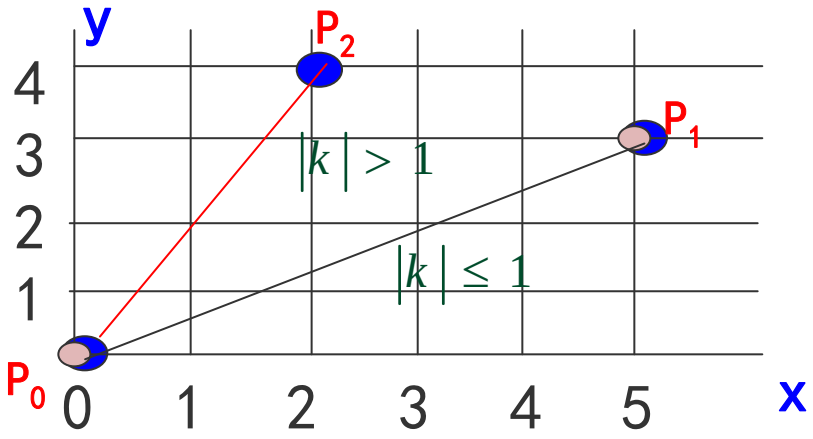
给大家留二个思考题

（1）DDA画直线算法：x每递增1，y递增斜率k。是否适合任意斜率的直线？

$$|k| \le 1$$

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i + k$$

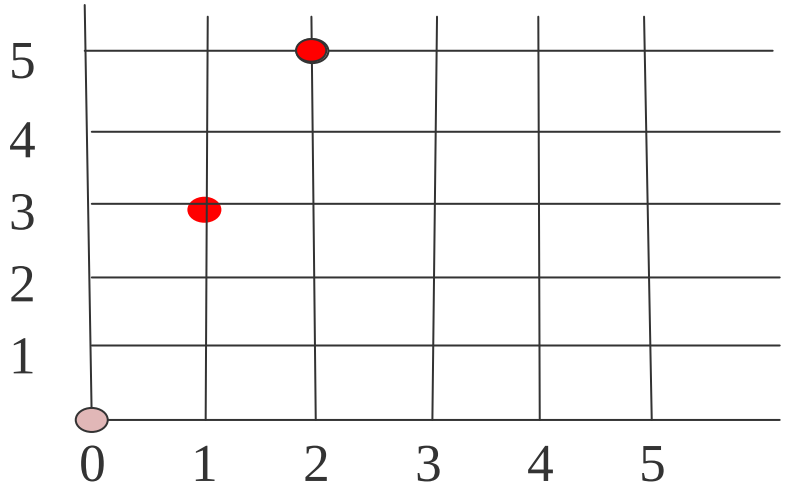


$|k| > 1$

用DDA方法画连接两点P_0（0,0）和P_1（2,5）的直线段

K=5/2=2.5>1　　$y_{i+1} = y_i + k$

x	y	int(y+0.5)
0	0	0
1	2.5	3
2	5	5



再比如直线点从（0，0）到（2，100），也只用3个点来表示

（1） $|k| > 1$

$$x_{i+1} = x_i + ?$$

$$y_{i+1} = y_i + ?$$

（2）DDA画直线算法是否最优呢？若非，如何改进？