



区域填充

王坤峰 教授

信息科学与技术学院



内容

- 区域填充
 - 扫描线填充算法
 - 种子填充算法

■ 什么是区域填充？

- 区域填充即给出一个区域的边界，要求对边界范围内的所有像素单元赋予指定的颜色代码。
- 区域填充中最常用的是**多边形填色**。
- 多边形填色即给出一个多边形的边界，要求对多边形边界范围内的所有像素单元赋予指定的色代码。
- 要完成这个任务，一个首要的问题，是判断一个像素是在多边形内还是外。

■ 填色算法分为两大类：

1. 扫描线(Scanline)填色算法。

- 这类算法建立在多边形边界的矢量形式数据之上，可用于程序填色，也可用于交互填色。

2. 种子(Seed)填色算法。

- 这类算法建立在多边形边界的图像形式数据之上，并还需提供多边形界内一点的坐标。所以，它一般只能用于人机交互填色，而难以用于程序填色。

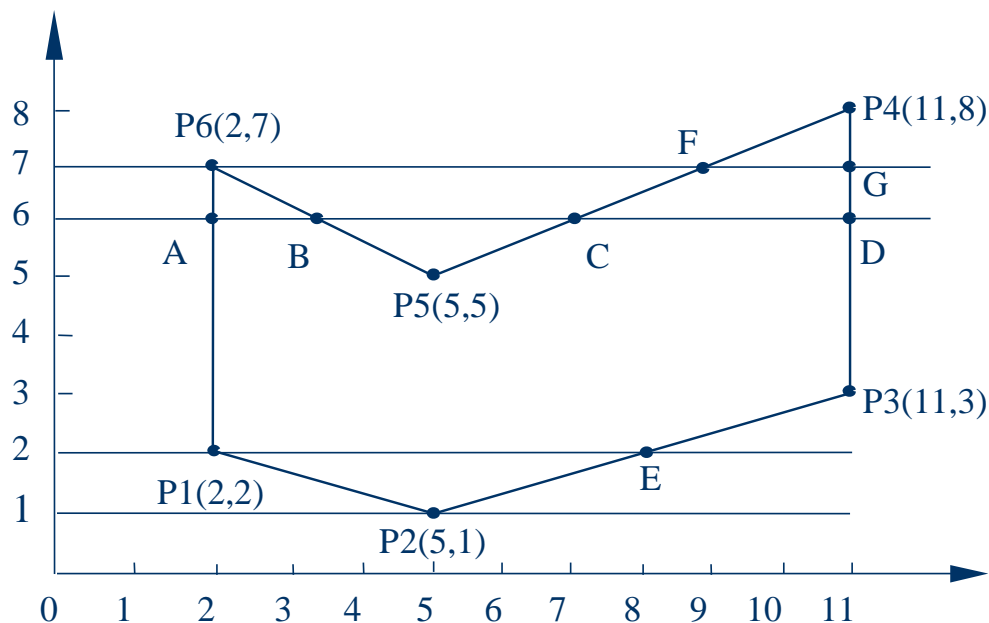


扫描线填充算法

■ 基本思想

- 用水平扫描线从上到下扫描由点、线段构成的多边形。
- 每根扫描线与多边形各边产生一系列交点。将这些交点按照 x 坐标进行分类，将分类后的交点成对取出，作为两个端点，用要求的色彩画水平直线。
- 多边形被扫描完毕后，填色也就完成。

(1) 求交 (2) 排序 (3) 配对 (4) 填色



活性边表算法

■ 活性边表 (AET)

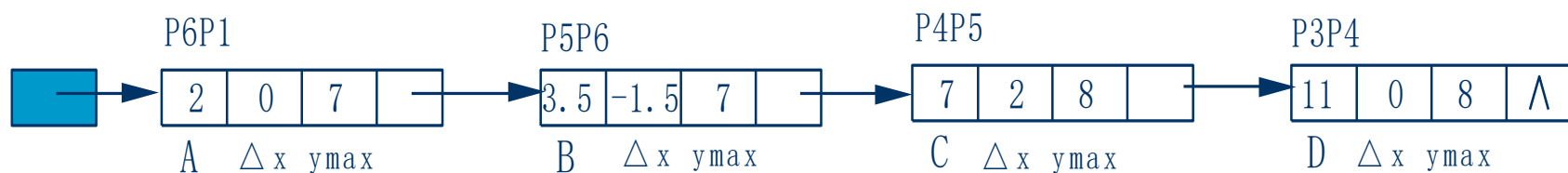
- 把与当前扫描线相交的边称为活性边，并把它们按与扫描线交点x坐标递增的顺序存放在一个链表中

- 结点内容

x: 当前扫描线与边的交点坐标

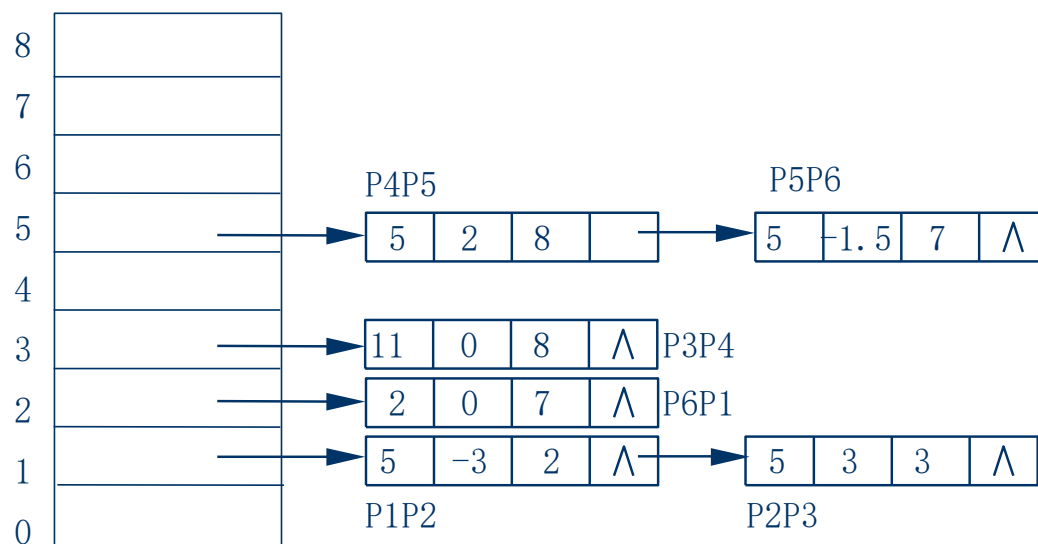
Δx : 从当前扫描线到下一条扫描线间x的增量

y_{max}: 该边所交的最高扫描线号y_{max}



■ 新边表（NET）

- 存放在该扫描线第一次出现的边。若某边的较低端点为 y_{\min} ，则该边就放在扫描线 y_{\min} 的新边表中





算法程序

```
void polyfill (polygon, color)
int color; 多边形 polygon;
{ for (各条扫描线i)
  { 初始化新边表头指针NET [i];
    把 $y_{\min} = i$  的边放进新边表NET [i];
  }
y = 最低扫描线号;
初始化活性边表AET为空;
```

```
for (各条扫描线i )  
{  
    把新边表NET [i] 中的边结点用插入排序法插入活性边  
    表AET，使之按x坐标递增顺序排列；  
    遍历AET，把配对交点区间(左闭右开)上的像素(x, y)，  
    用drawpixel (x, y, color) 改写像素颜色值；  
    遍历AET表，把 $y_{\max}=i$  的结点从AET中删除，并把  
     $y_{\max} > i$  结点的x值递增 $\Delta x$ ；  
}  
} /* polyfill */
```

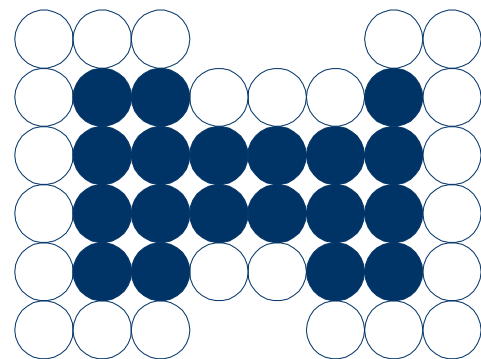
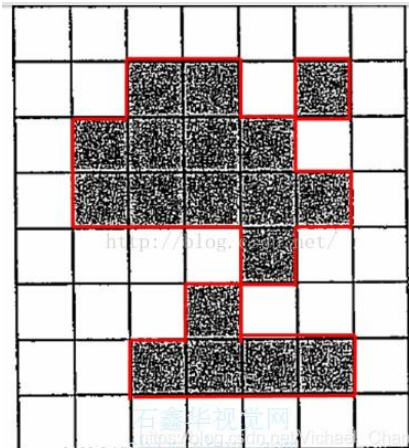
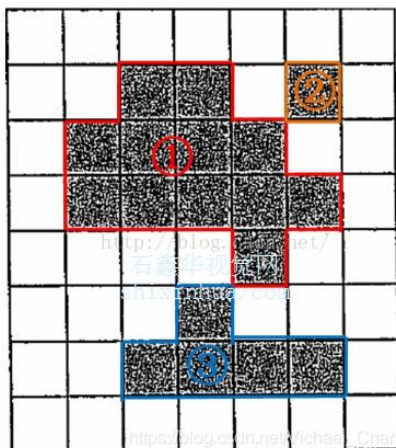
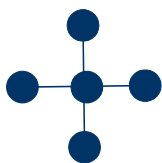


种子填充算法

■ 基本思想

- 指先将区域的一点赋予指定的颜色，然后将该颜色扩展到整个区域的过程。
- 种子填充算法要求区域是连通的。

4向连通和8向连通

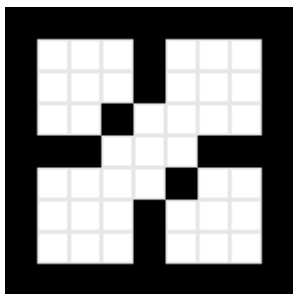


● 表示内点 ○ 表示边界点

种子填充的递归算法

■ 4连通区域的递归填充算法:

```
void BoundaryFill4(int x,int y,int boundarycolor,int newcolor)
{  int color = getpixel (x,y);
   if(color!=newcolor && color!=boundarycolor)
   {  drawpixel(x,y,newcolor);
      BoundaryFill4 (x,y+1, boundarycolor,newcolor);
      BoundaryFill4 (x,y-1, boundarycolor,newcolor);
      BoundaryFill4 (x-1,y, boundarycolor,newcolor);
      BoundaryFill4 (x+1,y, boundarycolor,newcolor);
   }
}
```



种子填充的扫描线算法

■ 算法步骤:

- 1.初始化: 堆栈置空。将种子点 (x, y) 入栈。
- 2.出栈: 若栈空则结束。否则取栈顶元素 (x, y) , 以 y 作为当前扫描线。
- 3.填充并确定种子点所在区段: 从种子点 (x, y) 出发, 沿当前扫描线向左、右两个方向填充, 直到边界。分别标记区段的左、右端点坐标为 x_l 和 x_r 。
- 4.并确定新的种子点: 在区间 $[x_l, x_r]$ 中检查与当前扫描线 y 上、下相邻的两条扫描线上的像素。若存在非边界、未填充的像素, 则把每一区间的最右像素作为种子点压入堆栈, 返回第2步。

