# 图形的光栅转化

刘世光
天津大学计算机学院

# 内容

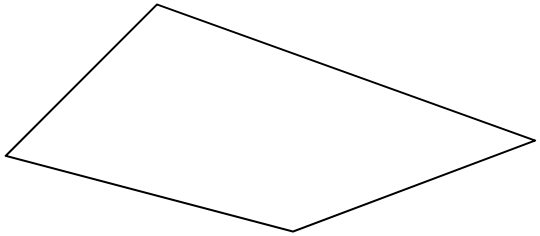
- 基本概念
- 多边形的扫描转换

# 内容

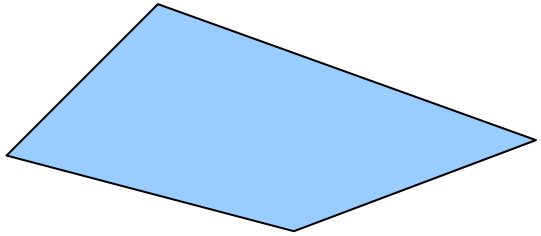
- 基本概念
- 多边形的扫描转换

# 光栅图形的基本概念

- 关于光栅图形
  - 本质：点阵表示
  - 特点：面着色，画面明暗自然、色彩丰富
  - 与线框图相比：更加生动、直观、真实感强

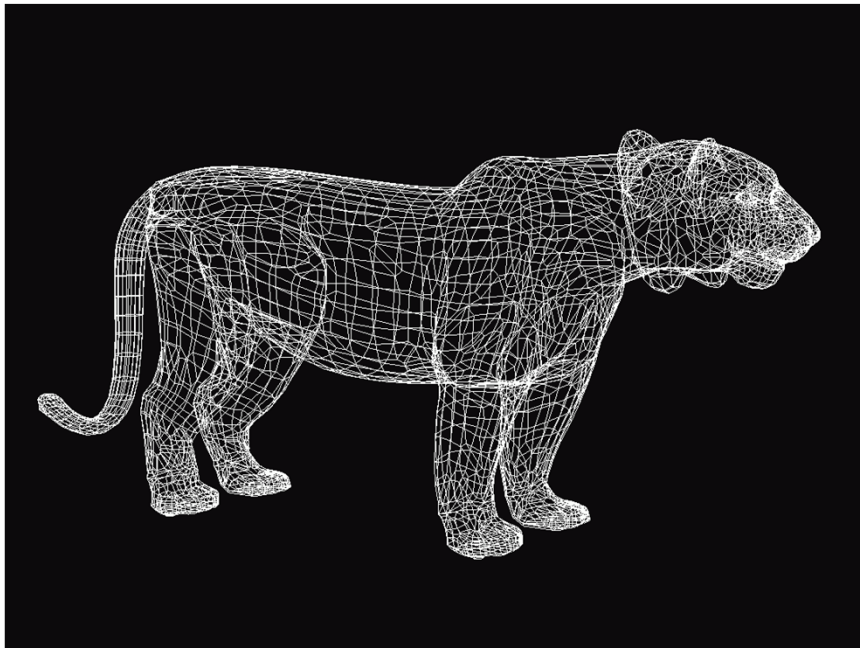


线框平面多边形

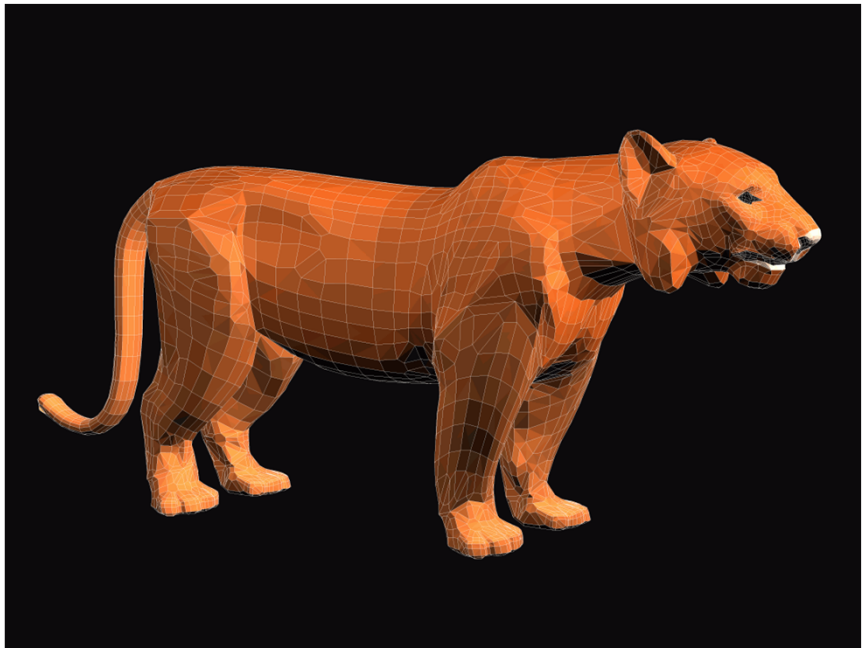


着色的平面多边形

# 光栅图形的基本概念



线框多边形物体



填充多边形物体

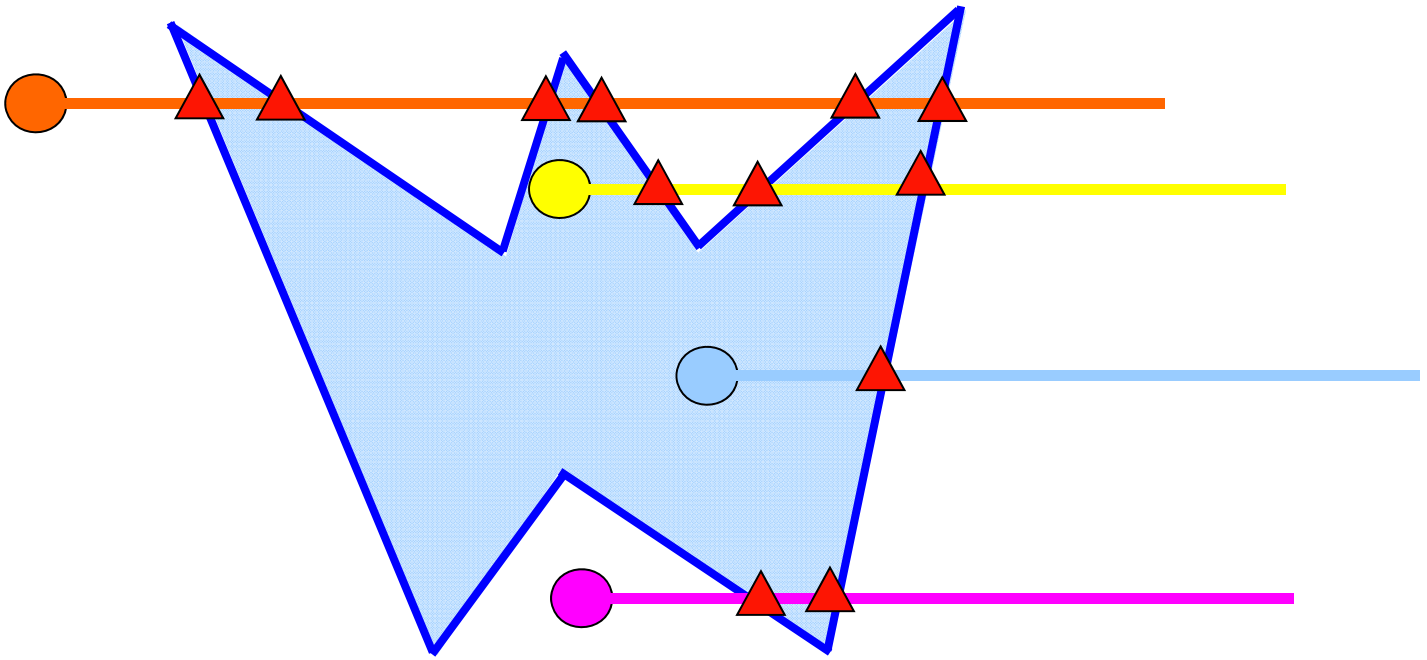
# 内容

- 基本概念
- 多边形的扫描转换
  - 逐点判断算法
  - 扫描线算法
    - 连贯性概念：区域、扫描线、边
    - 奇异点的处理
    - 算法的数据结构与实现

# 逐点判断算法

- 逐点判断算法：逐个像素判别其是否位于多边形内部
- 判断一个点是否位于多边形内部：射线法
  - 从当前像素发射一条射线，计算射线与多边形的交点个数
    - 内部：奇数个交点
    - 外部：偶数个交点

# 逐点判断算法



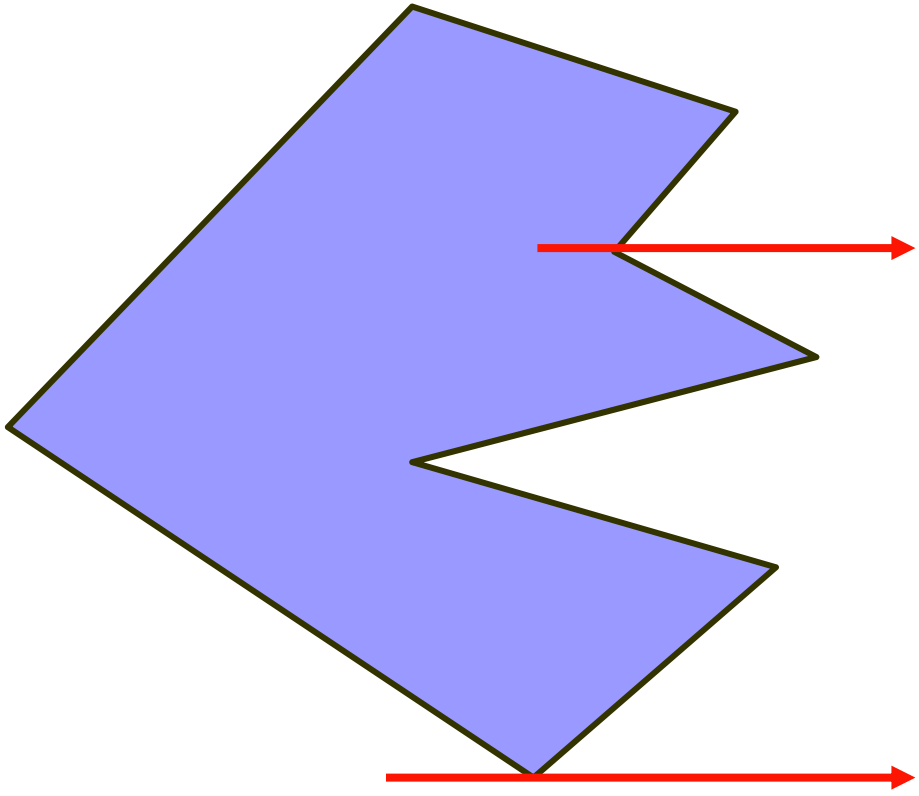
判断一点是否位于多边形内部？

# 逐点判断算法

- 算法描述

```
for(y=0; y<=y_resolution; y++)
for(x=0; x<=x_resolution; x++)
{
    if(inside(polygon, x+0.5, y+0.5))
      setpixel(framebuffer,x,y,polygon_color)
    else
      setpixel(framebuffer,x,y,background_color)
}
```

# 逐点判断算法中的奇异情况



1个或2个交点？

# 逐点判断算法的不足

- 速度慢：几十万甚是几百万像素的多边形内外判断，大量的求交、乘除运算
- 没有考虑像素之间的联系

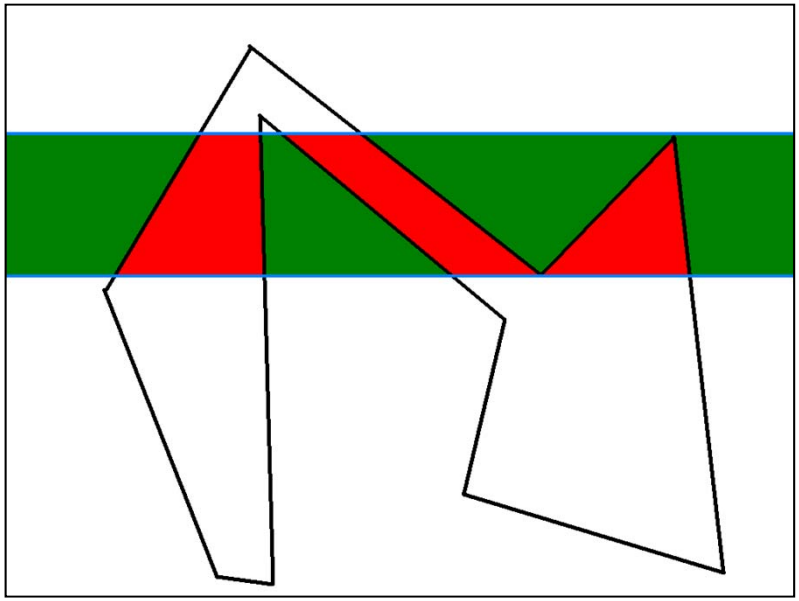
- 结论：逐点判断算法不可取！

# 相邻像素之间的连贯性

- 扫描线算法充分利用了相邻像素之间的连贯性，避免了对像素的逐点判断和求交运算，提高了算法效率
- 各种连贯性的定义
  - 区域连贯性
  - 扫描线连贯性
  - 边的连贯性

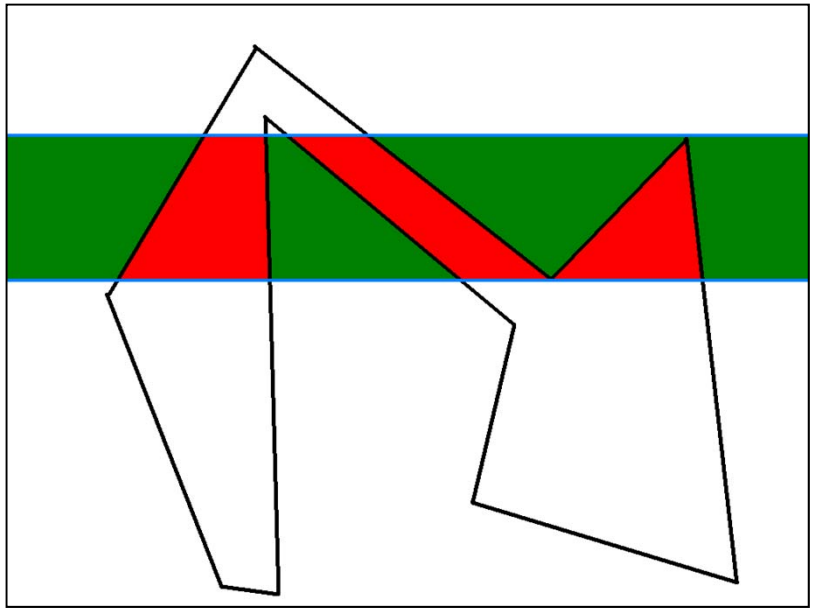
# 区域连贯性

- 区域的连贯性是指多边形定义的区域内部相邻的像素具有相同的性质。例如具有相同的颜色



区域的连贯性

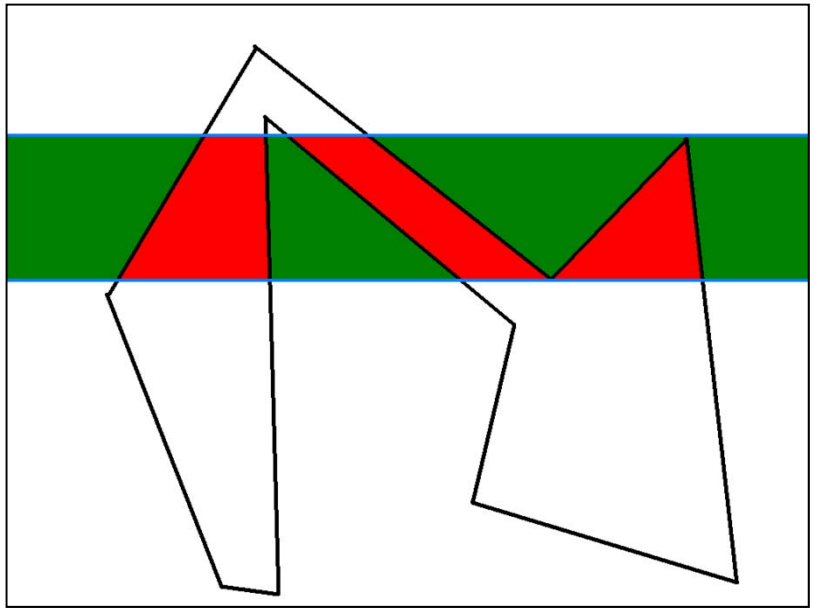
# 区域连贯性



区域的连贯性

- 两条扫描线之间的长方形区域被所处理的多边形分割成若干梯形(三角形可以看作退化梯形)
- 梯形的底边为扫描线，梯形的腰为多边形的边或窗口边缘

# 区域连贯性

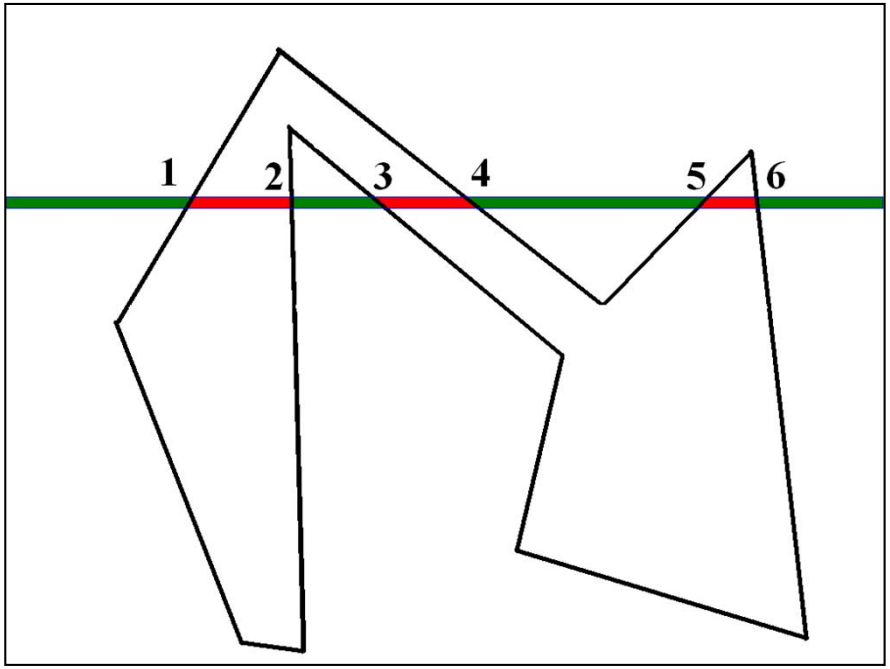


区域的连贯性

- 梯形分为两类：多边形内部和多边形外部
- 两类梯形在多边形内部相间排列(相邻的两个梯形必然有一个位于多边形内部，有一个在多边形外部)

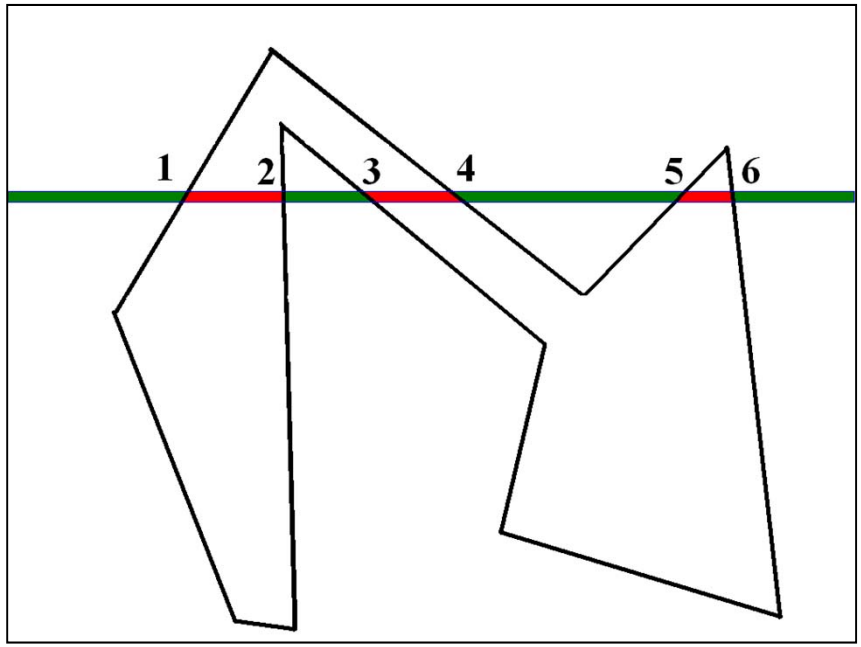
# 扫描线连贯性

- 区域连贯性在一条扫描线上的反映




扫描线的连贯性

# 扫描线连贯性

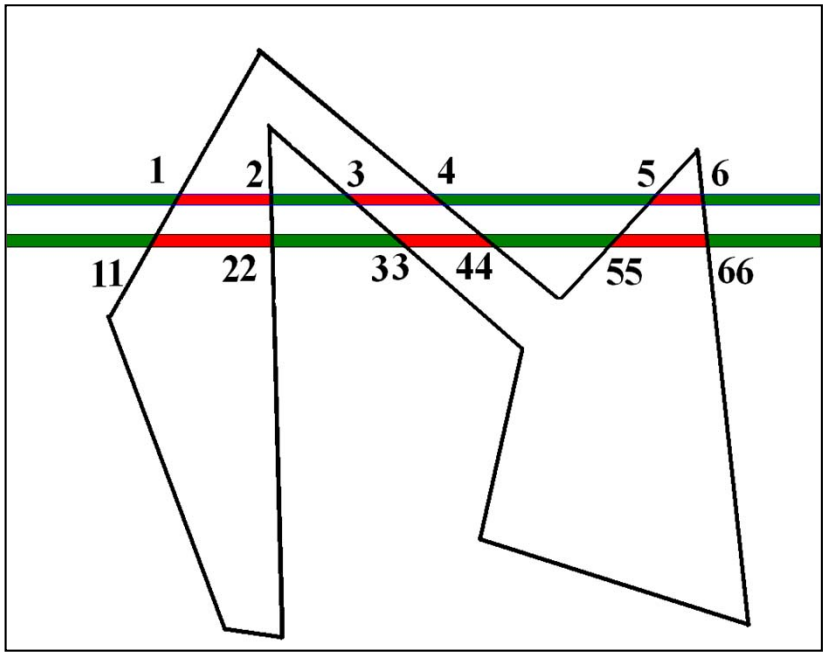



扫描线的连贯性

- 交点序列：扫描线与多边形的交点个数为偶数(1,2,3,4,5,6)
- 红色区间(1,2)、(3,4)、(5,6)位于多边形内部
- 其余绿色区间位于多边形外部
- 两类区间相间排列

# 边的连贯性

- 边的连贯性：直线的线性性质在光栅上的表现




边的连贯性

扫描线与边的交点

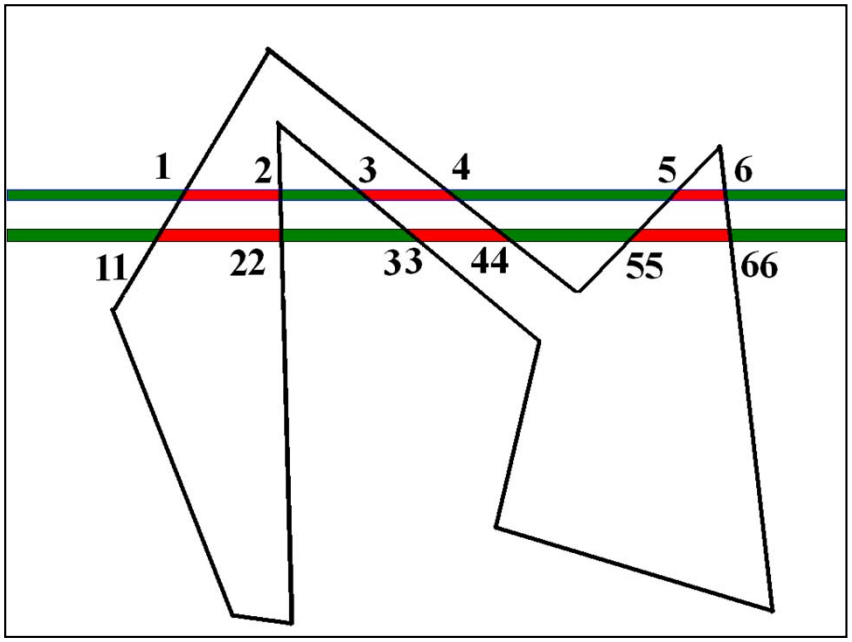
1 $(x_1,y_1)$

2 $(x_2,y_2)$

11 $(x_{11},y_{11})$

22 $(x_{22},y_{22})$

# 边的连贯性




边的连贯性

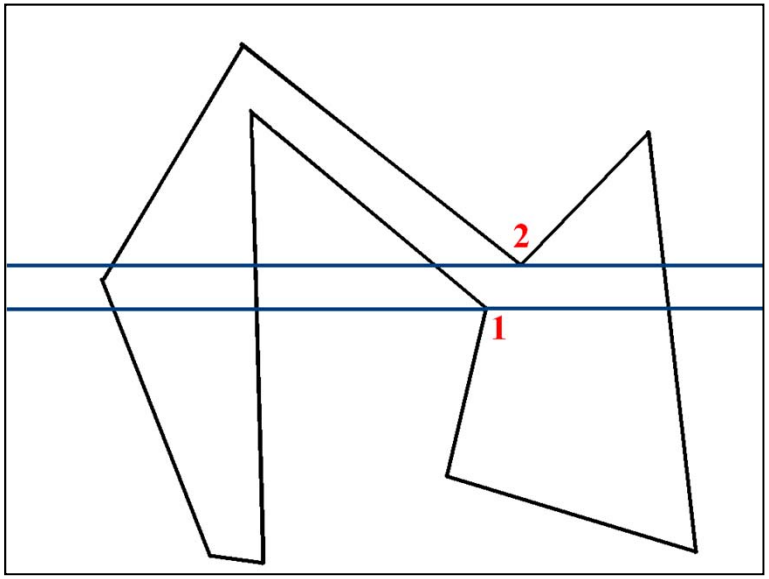
- 相邻扫描线($y_1=y_{11}+1$)与多边形的同一条边的交点存在如下关系：

$$\frac{y_1 - y_{11}}{x_1 - x_{11}} = k$$

$$x_1 = x_{11} + \frac{1}{k}$$

- 当知道扫描线与一条边的一个交点之后，通过上述公式可以通过增量算法迅速求出其他交点

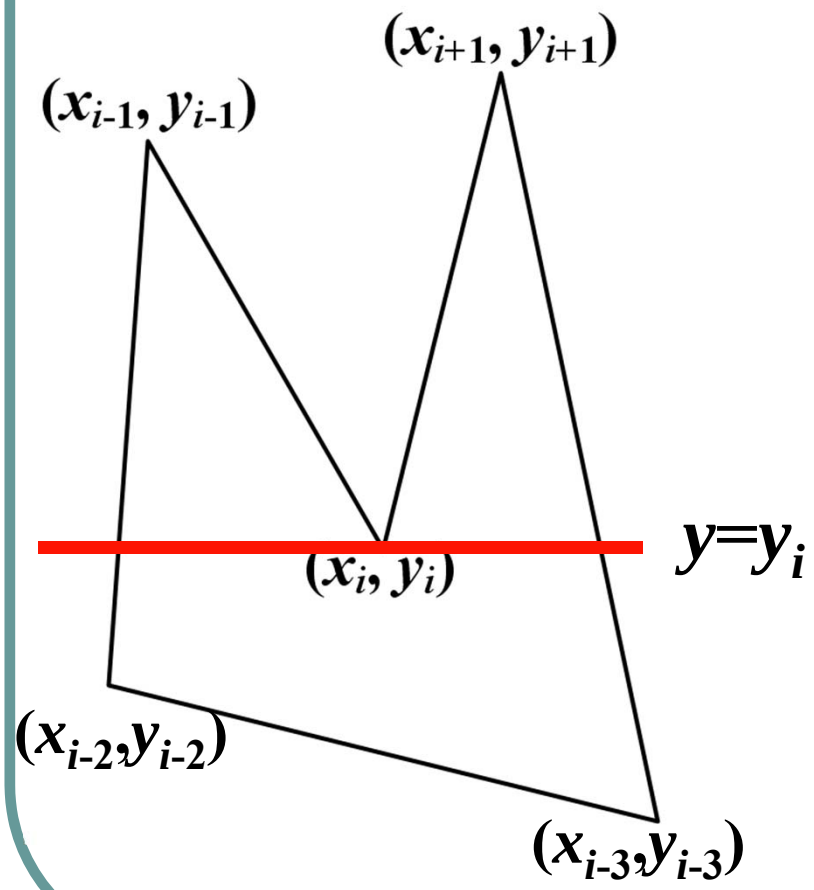
# 奇异点




奇异点示意图

- 奇异点：扫描线与多边形交交于多边形的顶点
- 奇异点计为几个交点？
  - 扫描线1：一个交点
  - 扫描线2：两个交点

# 奇异点的分类

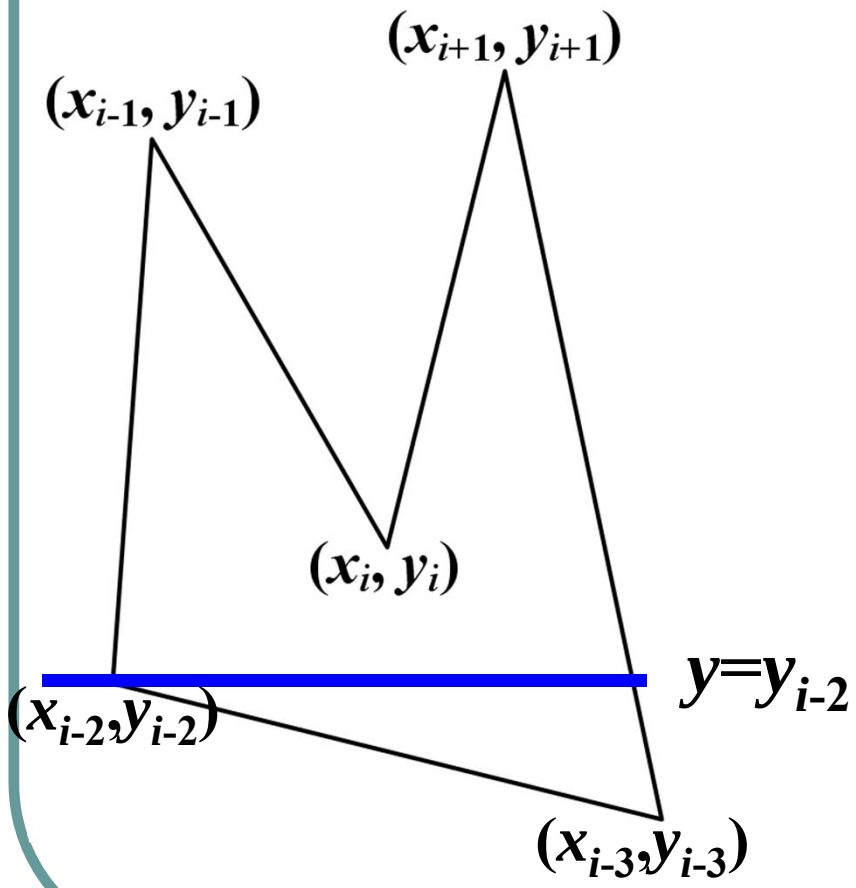



- 极值点：相邻三个顶点的$y$坐标满足如下条件：

$$(y_{i-1}-y_i)(y_{i+1}-y_i)\geq 0$$

即相邻三个顶点位于扫描线的同一侧

# 奇异点的分类




- 非极值点：相邻三个顶点的$y$坐标满足如下条件：

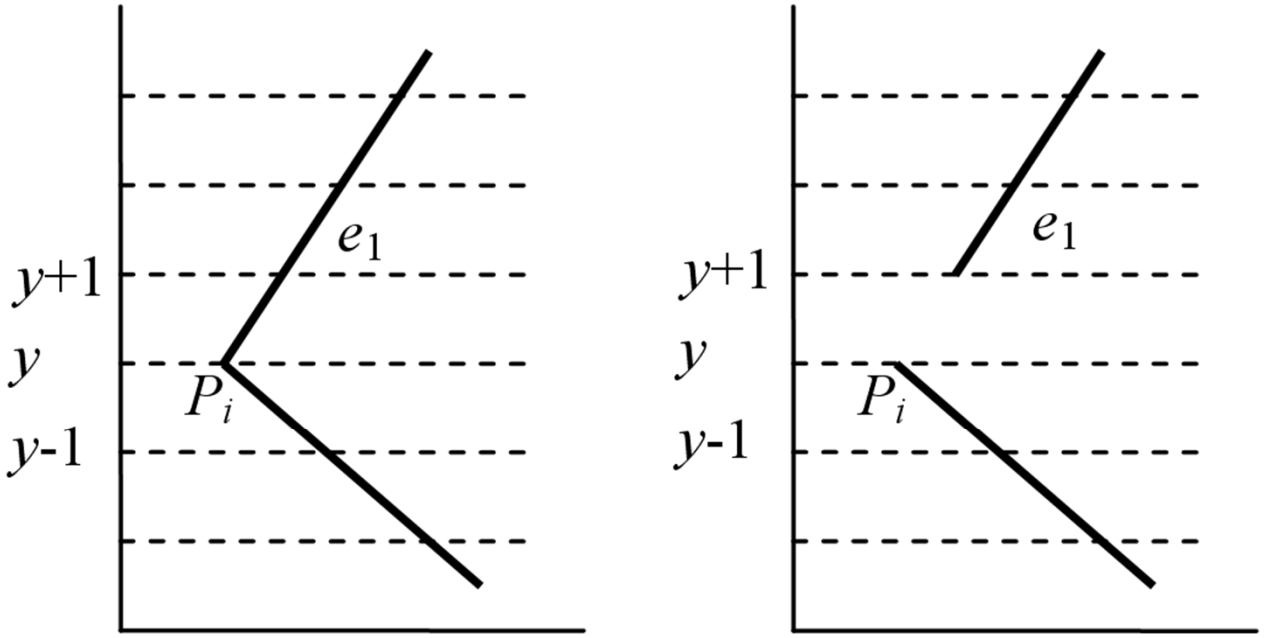
  $(y_{i-3}-y_{i-2})(y_{i-1}-y_{i-2})<0$

  即相邻三个顶点位于扫描线的两侧

# 奇异点的处理

- 奇异点的处理
  - 在**极值点**处，按**两个交点**计算
  - 在**非极值点**处，按**一个交点**计算
- 实际计算前，奇异点(非极值点)的预处理
  - 将扫描线上方线段截断一个单位，这样扫描线就只与多边形有一个交点。

# 奇异点的处理




扫描线 $y$ 通过非极值顶点 $P_i$，在进行扫描转换时，以 $P_i$ 为下端点的边 $e_1$ 在 $P_i$ 处截掉一个像素的宽度。

# 多边形扫描转换算法

- 核心思想(从下到上扫描)
  - 计算扫描线 $y = y_{\min}$ 与多边形的交点，通常这些交点由多边形的顶点组成
  - 根据多边形边的连贯性，按从下到上的顺序求得各条扫描线的交点序列
  - 根据区域和扫描线的连贯性判断位于多边形内部的区段
  - 对位于多边形内的直线段进行着色

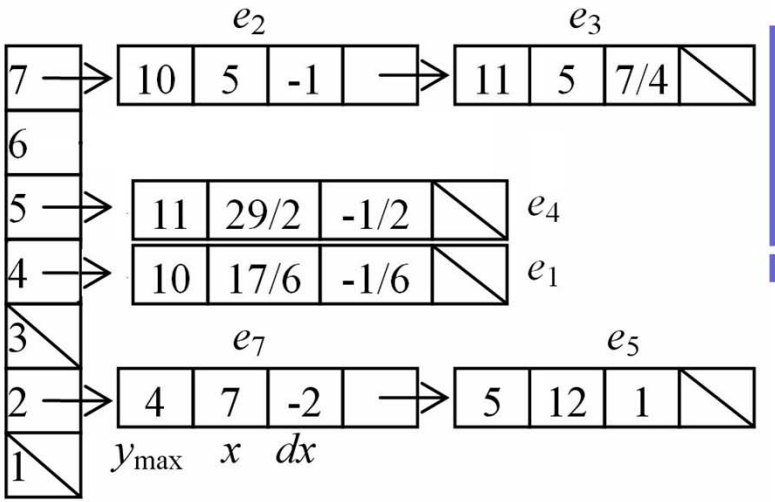
# 多边形扫描转换算法

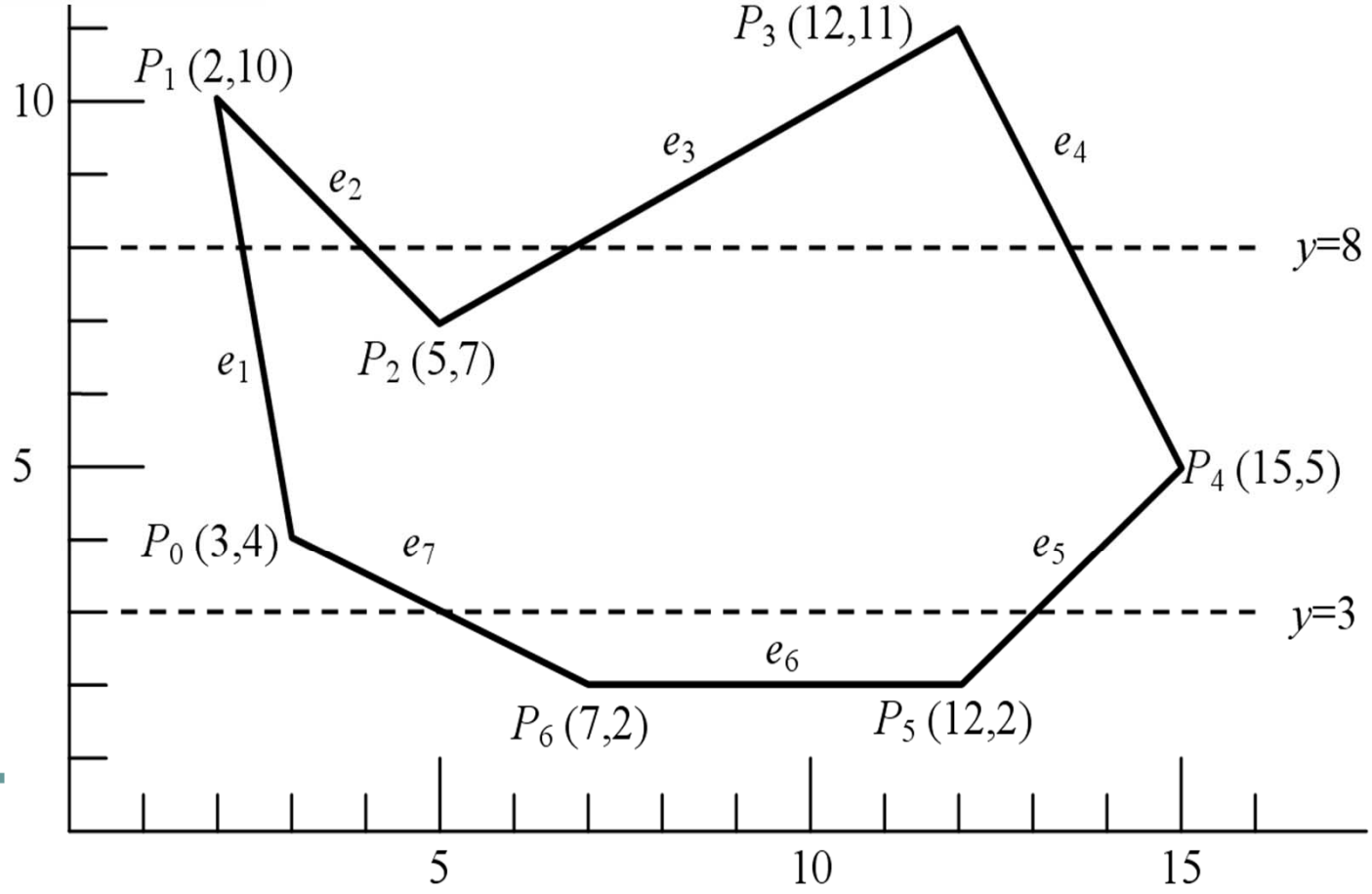
- 为了实现上述思想，算法中需要采取灵活的数据结构。
  - 分类的边表ET (Sorted Edge Table)：记录多边形信息
  - 活化边链表AEL (Active Edge List)：记录当前扫描线信息
- 它们共同基础是边的数据结构

# 边的数据结构

- 边的数据结构
  - $y_{\max}$：边的上端点的$y$坐标
  - $x$：边的下端点$x$坐标，在活化边链表中，表示扫描线与边的交点的$x$坐标
  - $dx$：边的斜率的倒数
  - next：指向下一条边的指针

# 边的数据结构实例

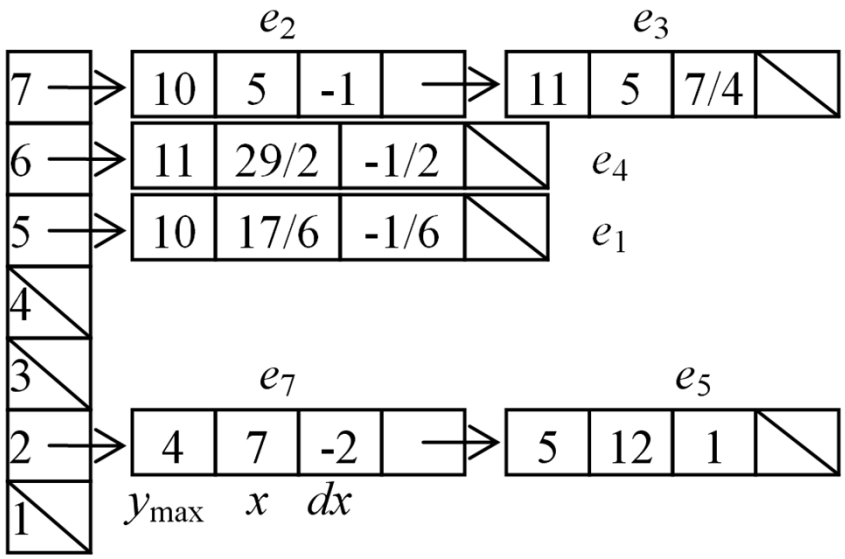

$e_2$
$e_3$
7
10 5 -1
11 5 7/4
6
5
11 29/2 -1/2
$e_4$
4
10 17/6 -1/6
$e_1$
3
$e_7$
$e_5$
2
4 7 -2
5 12 1
1
$y_{max}$ $x$ $dx$

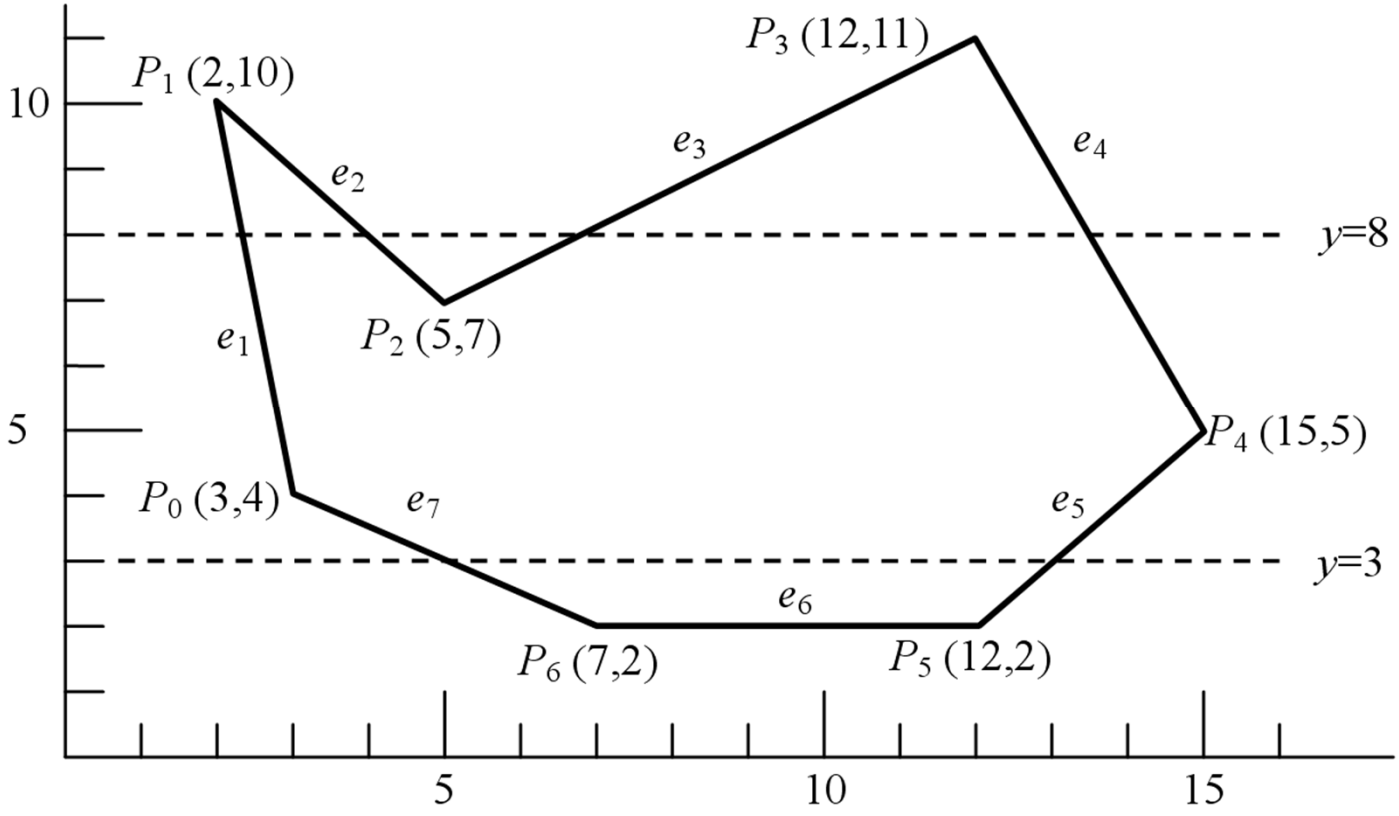

$P_1$ (2,10)
$P_3$ (12,11)
10
$e_2$
$e_3$
$e_4$
$y$=8
$e_1$
$P_2$ (5,7)
5
$P_4$ (15,5)
$P_0$ (3,4)
$e_7$
$e_5$
$y$=3
$e_6$
$P_6$ (7,2)
$P_5$ (12,2)
5
10
15

# 分类的边表 (ET)

- 分类的边表是按边的下端点的纵坐标$y$对非水平边进行分类的指针数组
  - 下端点的纵坐标$y$值等于$i$的边，归入第$i$类
  - 同一类中，各边按$x$值( $x$值相等时，按$dx$的值)递增的顺序排成行
  - 水平边不加入分类边表中

# 分类的边表实例

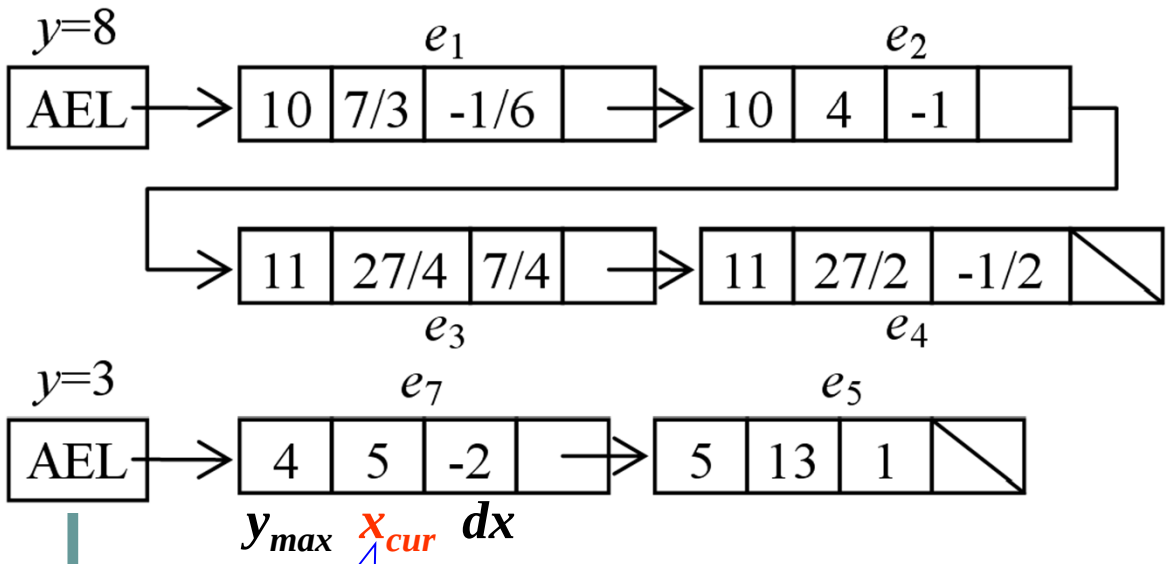

$e_2$
$e_3$
7
10 5 -1
11 5 7/4
6
11 29/2 -1/2
$e_4$
5
10 17/6 -1/6
$e_1$
4
3
$e_7$
$e_5$
2
4 7 -2
5 12 1
1
$y_{max}$ $x$ $dx$


$P_1$ (2,10)
$P_3$ (12,11)
10
$e_2$
$e_3$
$e_4$
$y$=8
$e_1$
$P_2$ (5,7)
5
$P_4$ (15,5)
$P_0$ (3,4)
$e_7$
$e_5$
$y$=3
$e_6$
$P_6$ (7,2)
$P_5$ (12,2)
5
10
15

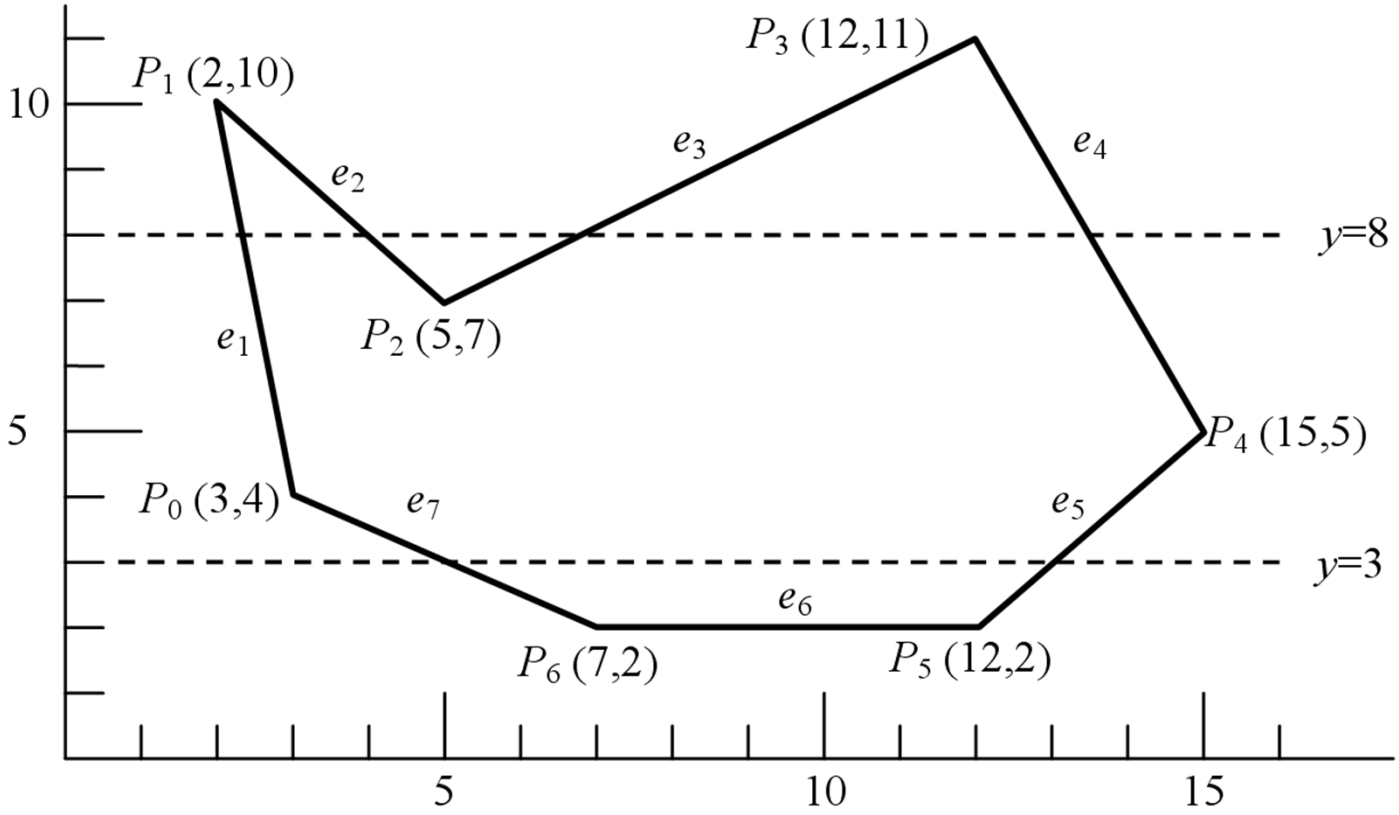
# 活化边链表(AEL)

- 活化链表由与当前扫描线相交的边组成
  - 记录了多边形的边沿扫描线的交点序列
  - 根据边的连贯性不断刷新交点序列
- 基本单元是边(与扫描线相交的边)
- 与分类边表不同
  - 分类边表记录初始状态
  - 活化边表随扫描线的移动而动态更新

# 活化边链表实例

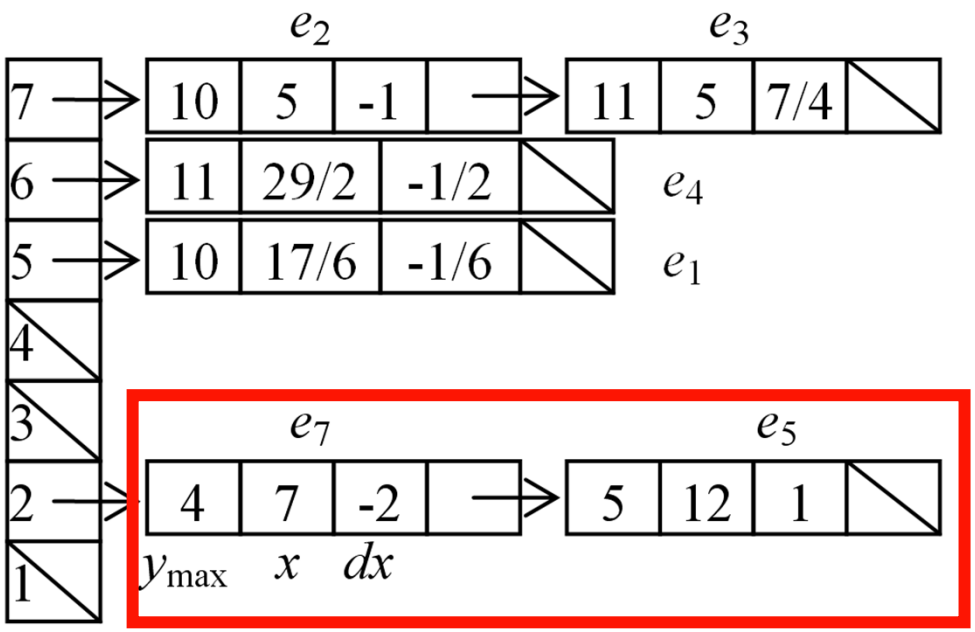



与分类边表的区别



# 多边形扫描转换算法

1. (*y*初始化) 取扫描线纵坐标*y*的初始值为ET中非空元素的最小序号 (*y*=2)



# 多边形扫描转换算法

2. (AEL初始化) 将边的活化链表AEL设置为空
3. 按从下到上的顺序对纵坐标值为*y*的扫描线(当前扫描线)执行如下步骤，直到分类边表ET和边的活化链表AEL都变成空为止

# 多边形扫描转换算法

a) 如果分类边表ET中的第*y*类元素非空，则将属于该类的所有边从ET中取出并插入边的活化链表AEL中(同时将ET中相应的边表删除)，AEL中的各边按照*x*值(*x*值相等时，按*dx*值)递增方向排序；

b) 若对于当前扫描线，边的活化链表非空，则将AEL中的边交点两两依次配对。每一对边与当前扫描线的交点区间位于多边形内部，依次对这些区间上的像素按多边形属性着色；

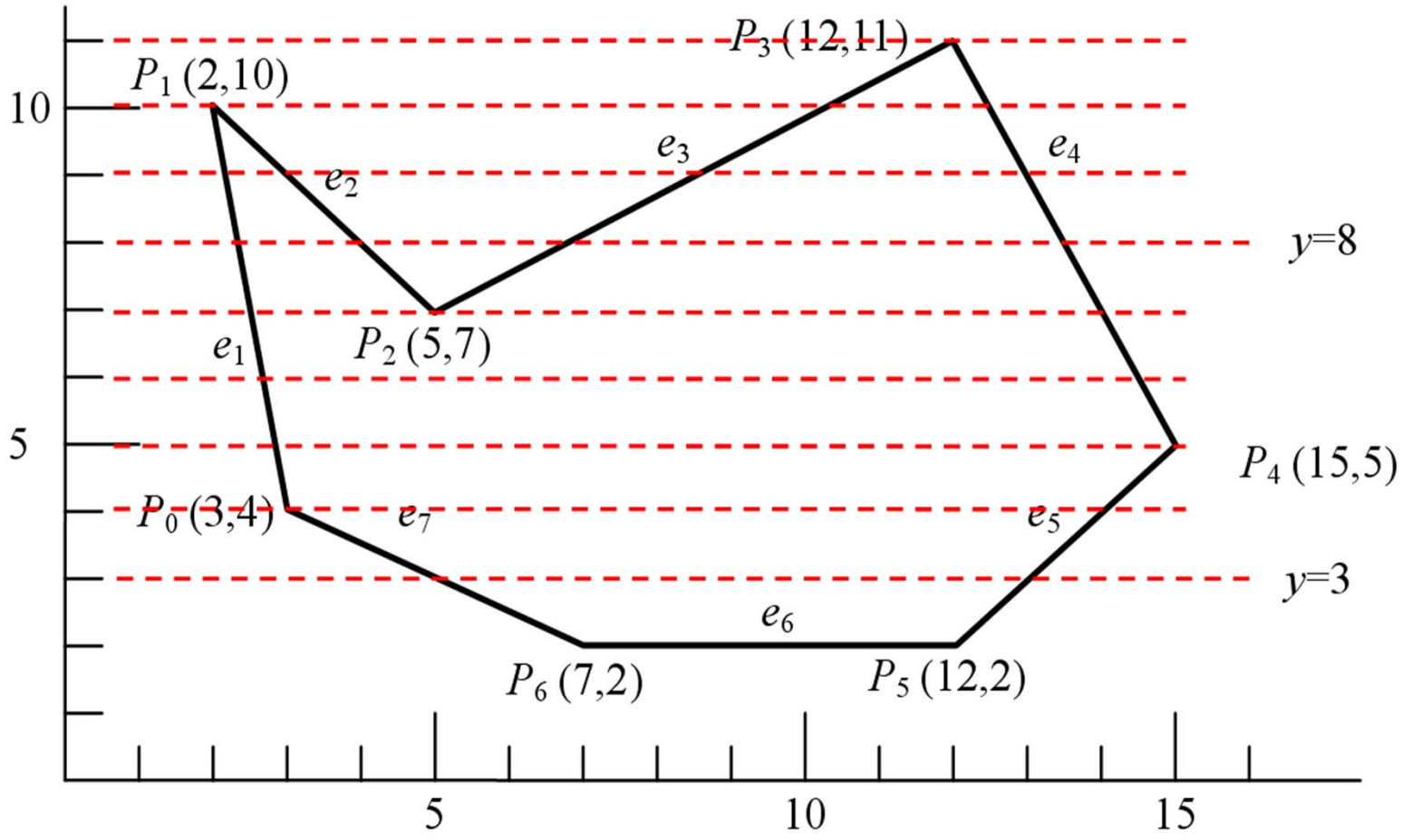
# 多边形扫描转换算法

c) 将边的活化链表AEL中满足$y=y_{max}$的边删除；

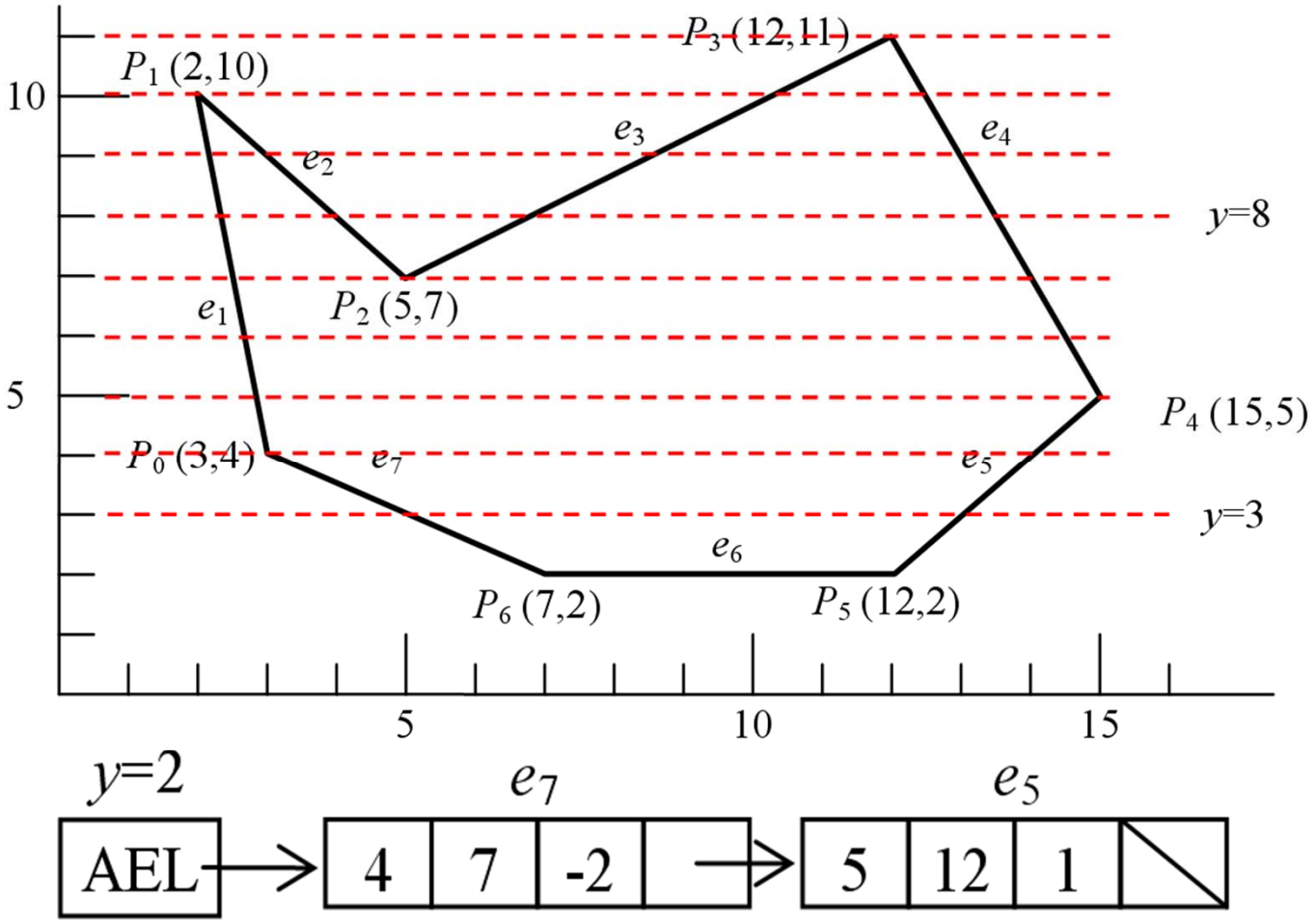
d) 将边的活化链表AEL中剩下的每一条边的$x$累加$dx$，即：$x=x+dx$;

e) 将当前扫描线的纵坐标值$y$累加，即$y=y+1$。

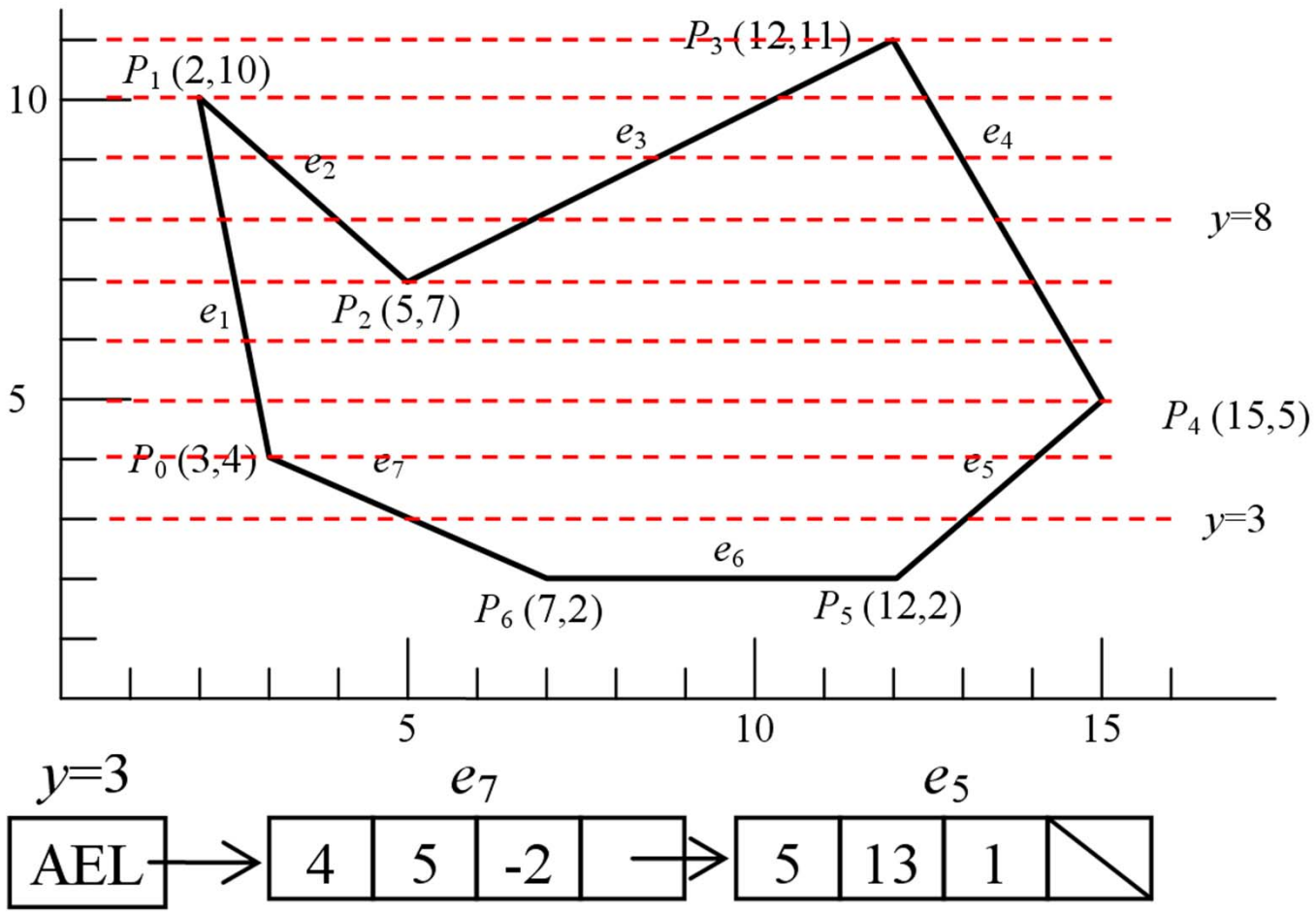
# 多边形扫描转换实例


$P_3$ (12,11)
$P_1$ (2,10)
10
$e_3$
$e_4$
$e_2$
$y$=8
$e_1$
$P_2$ (5,7)
5
$P_4$ (15,5)
$P_0$ (3,4)
$e_7$
$e_5$
$y$=3
$e_6$
$P_6$ (7,2)
$P_5$ (12,2)
5
10
15

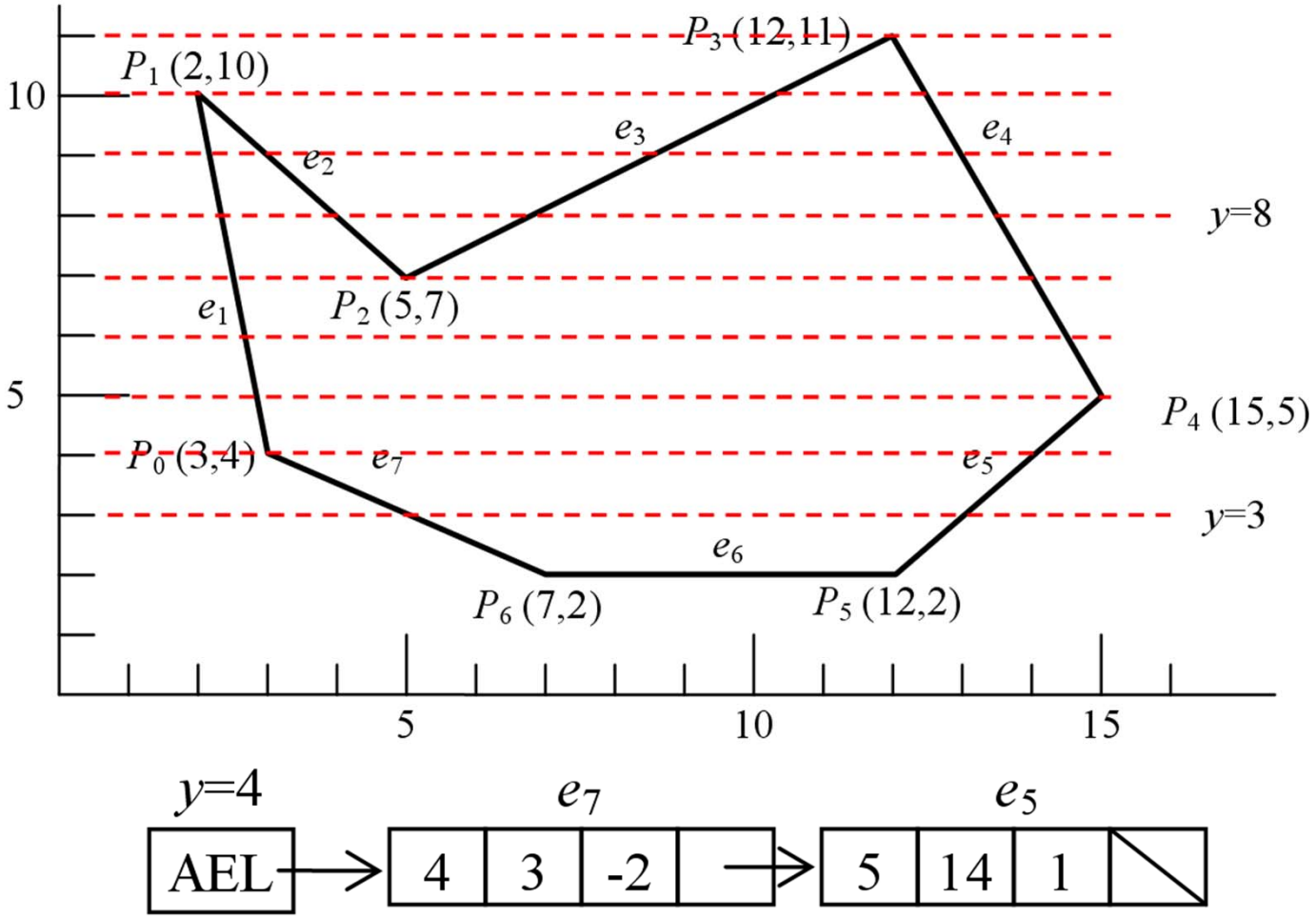
# 多边形扫描转换实例


$P_1$ (2,10)
$P_3$ (12,11)
$e_2$
$e_3$
$e_4$
$y$=8
$e_1$
$P_2$ (5,7)
$P_4$ (15,5)
$P_0$ (3,4)
$e_7$
$e_5$
$y$=3
$e_6$
$P_6$ (7,2)
$P_5$ (12,2)
10
5
5
10
15
$y$=2
$e_7$
$e_5$
AEL
4
7
-2
5
12
1

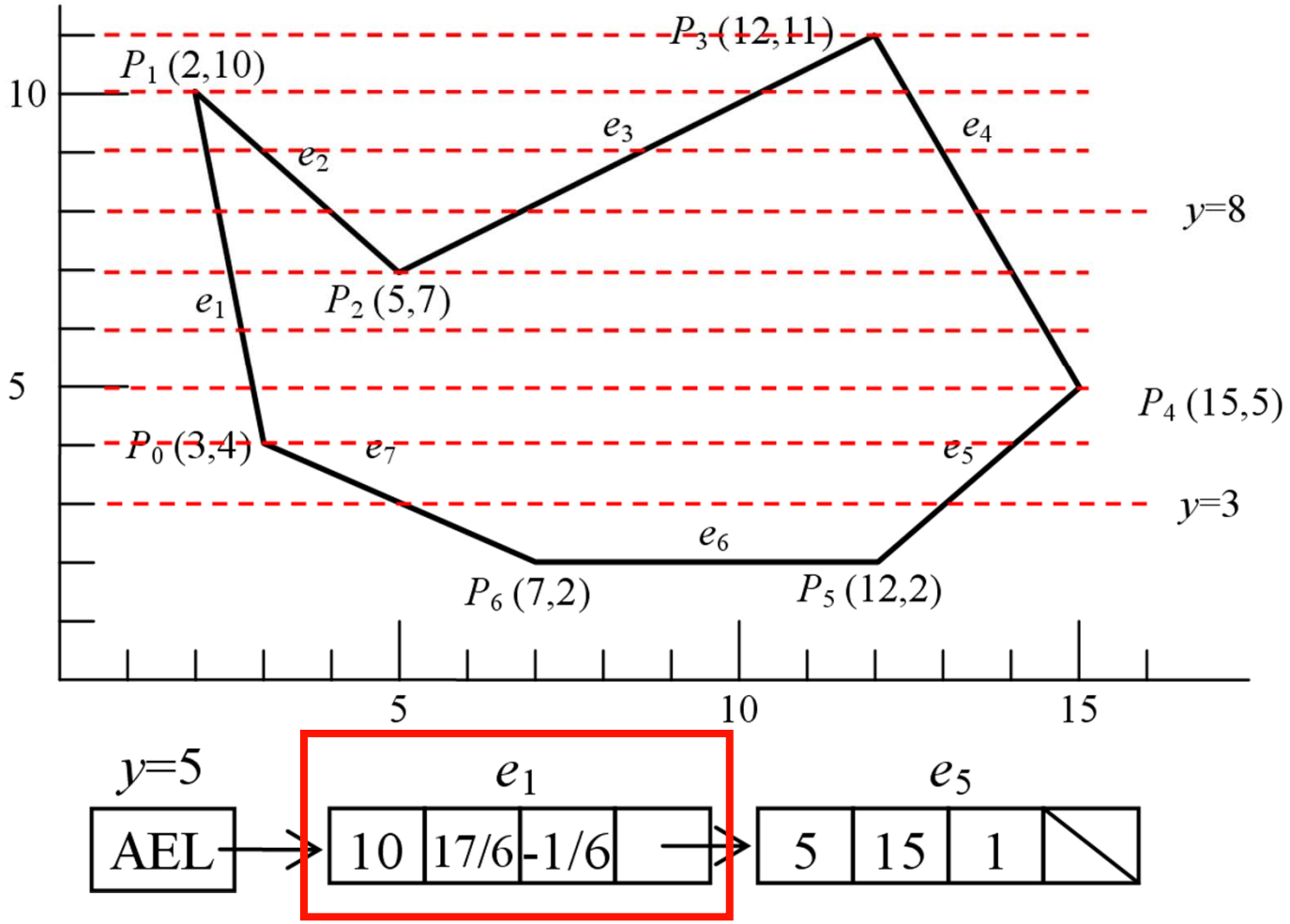
# 多边形扫描转换实例


$P_1$ (2,10)
$P_3$ (12,11)
10
$e_2$
$e_3$
$e_4$
$y$=8
$e_1$
$P_2$ (5,7)
5
$P_4$ (15,5)
$P_0$ (3,4)
$e_7$
$e_5$
$y$=3
$e_6$
$P_6$ (7,2)
$P_5$ (12,2)
5
10
15
$y$=3
$e_7$
$e_5$
AEL
4
5
-2
5
13
1

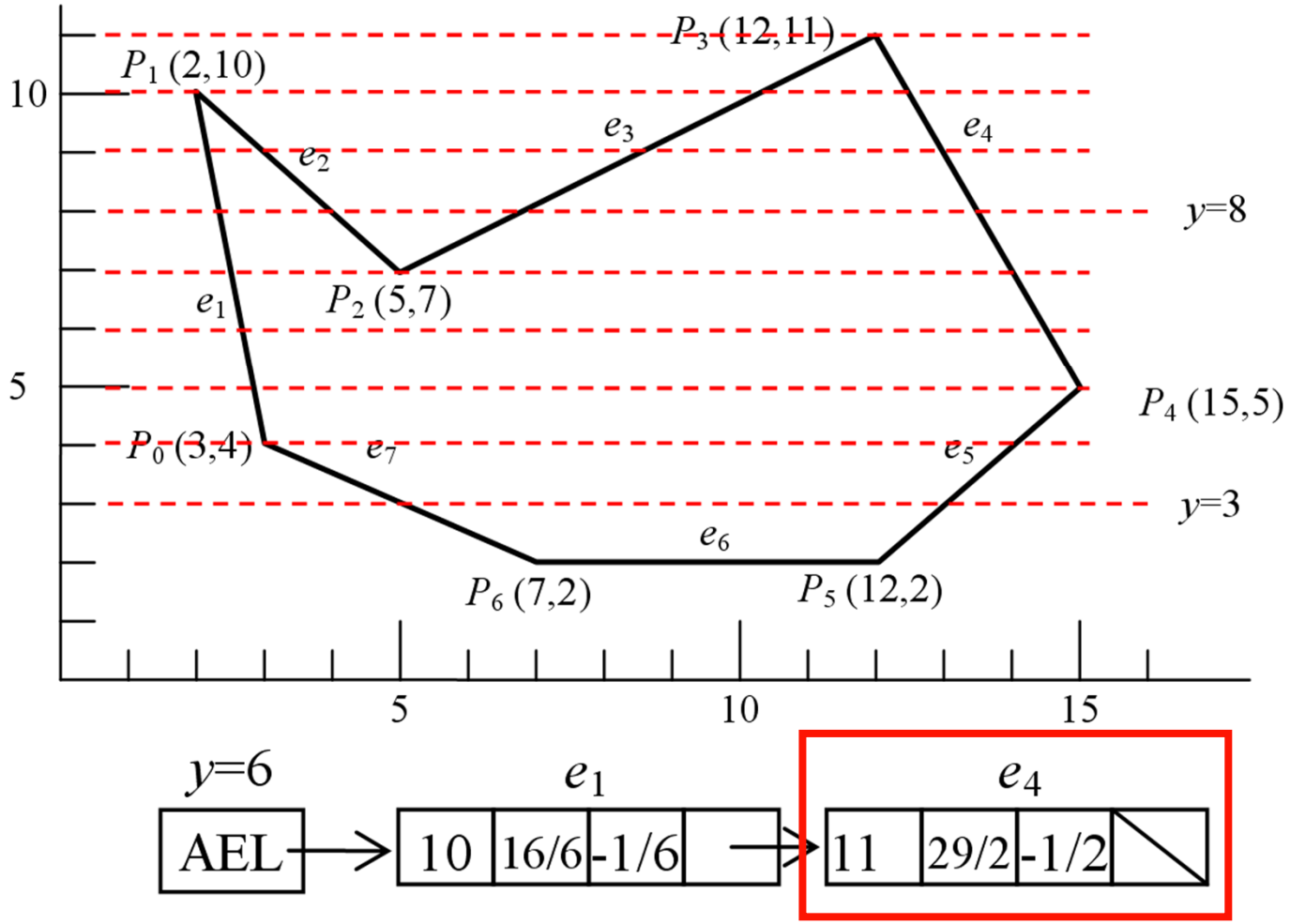
# 多边形扫描转换实例


$P_1$ (2,10)
$P_2$ (5,7)
$P_3$ (12,11)
$P_4$ (15,5)
$P_5$ (12,2)
$P_6$ (7,2)
$P_0$ (3,4)
$e_1$
$e_2$
$e_3$
$e_4$
$e_5$
$e_6$
$e_7$
y=8
y=3
10
5
5
10
15
y=4
AEL
4
3
-2
5
14
1

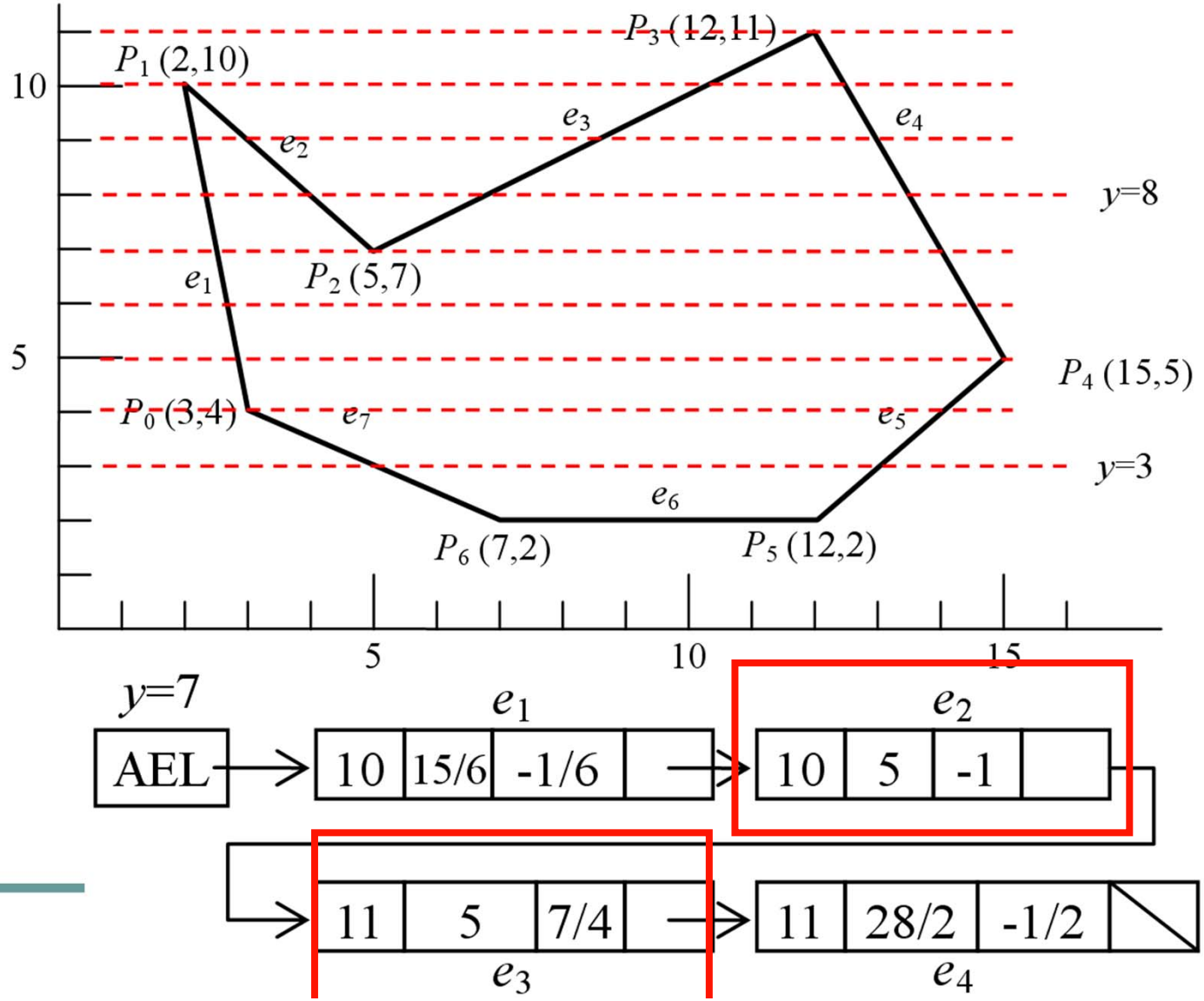
# 多边形扫描转换实例


$P_1$ (2,10)
$P_2$ (5,7)
$P_3$ (12,11)
$P_4$ (15,5)
$P_5$ (12,2)
$P_6$ (7,2)
$P_0$ (3,4)
$e_1$
$e_2$
$e_3$
$e_4$
$e_5$
$e_6$
$e_7$
y=8
y=3
10
5
5
10
15
y=5
AEL
$e_1$
10
17/6
-1/6
$e_5$
5
15
1

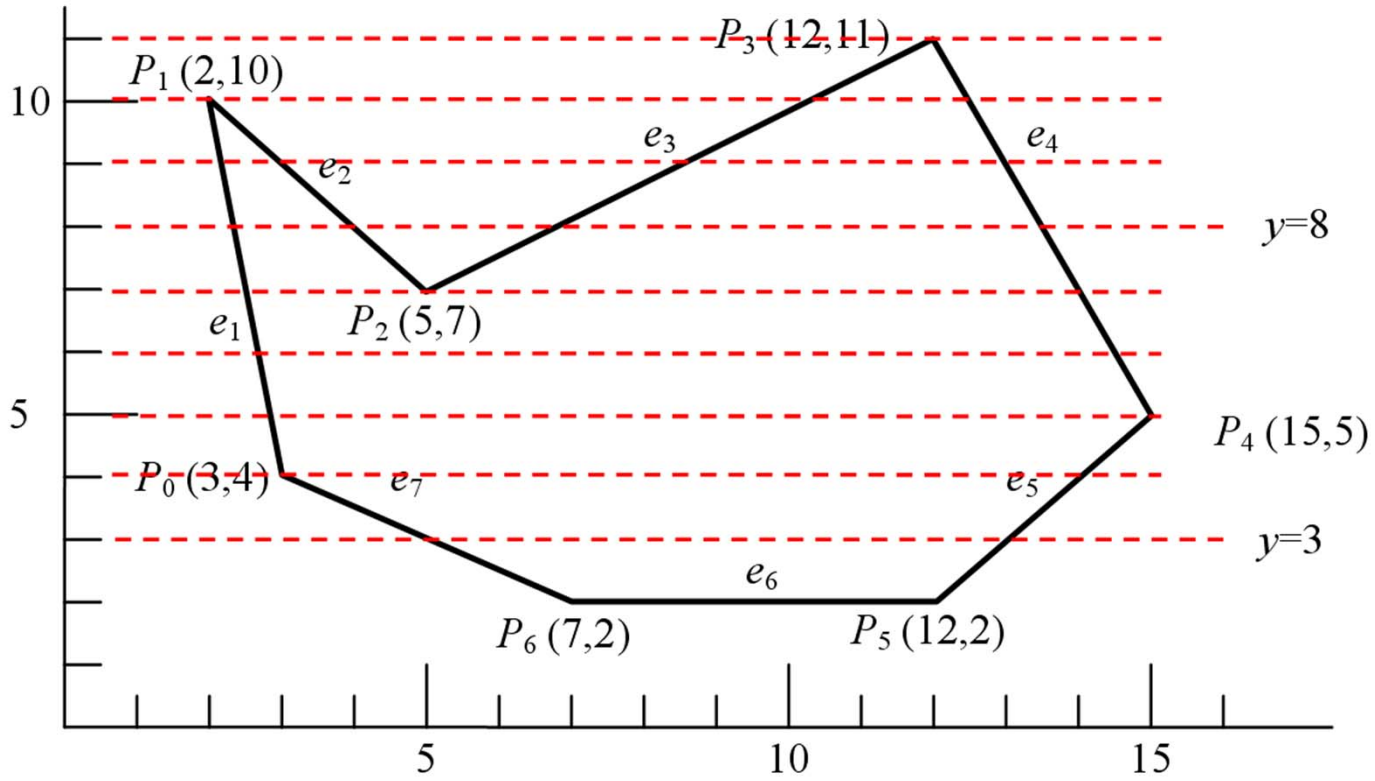
# 多边形扫描转换实例

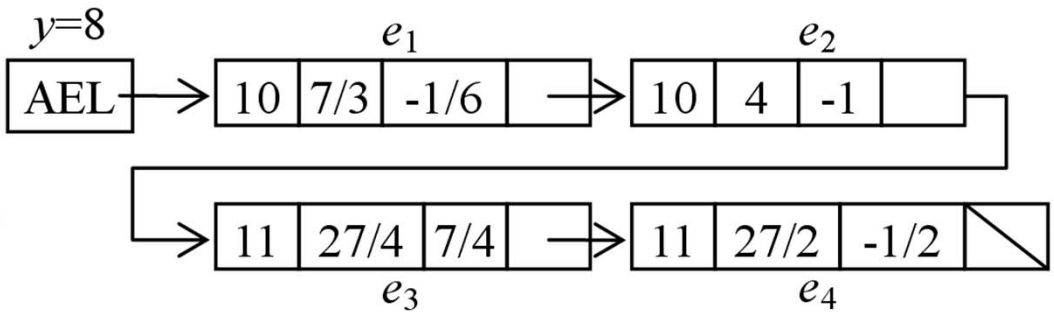

$P_1$ (2,10)
$P_3$ (12,11)
10
$e_2$
$e_3$
$e_4$
$y$=8
$e_1$
$P_2$ (5,7)
5
$P_4$ (15,5)
$P_0$ (3,4)
$e_7$
$e_5$
$y$=3
$e_6$
$P_6$ (7,2)
$P_5$ (12,2)
5
10
15
$y$=6
$e_1$
$e_4$
AEL
10
16/6
-1/6
11
29/2
-1/2

# 多边形扫描转换实例

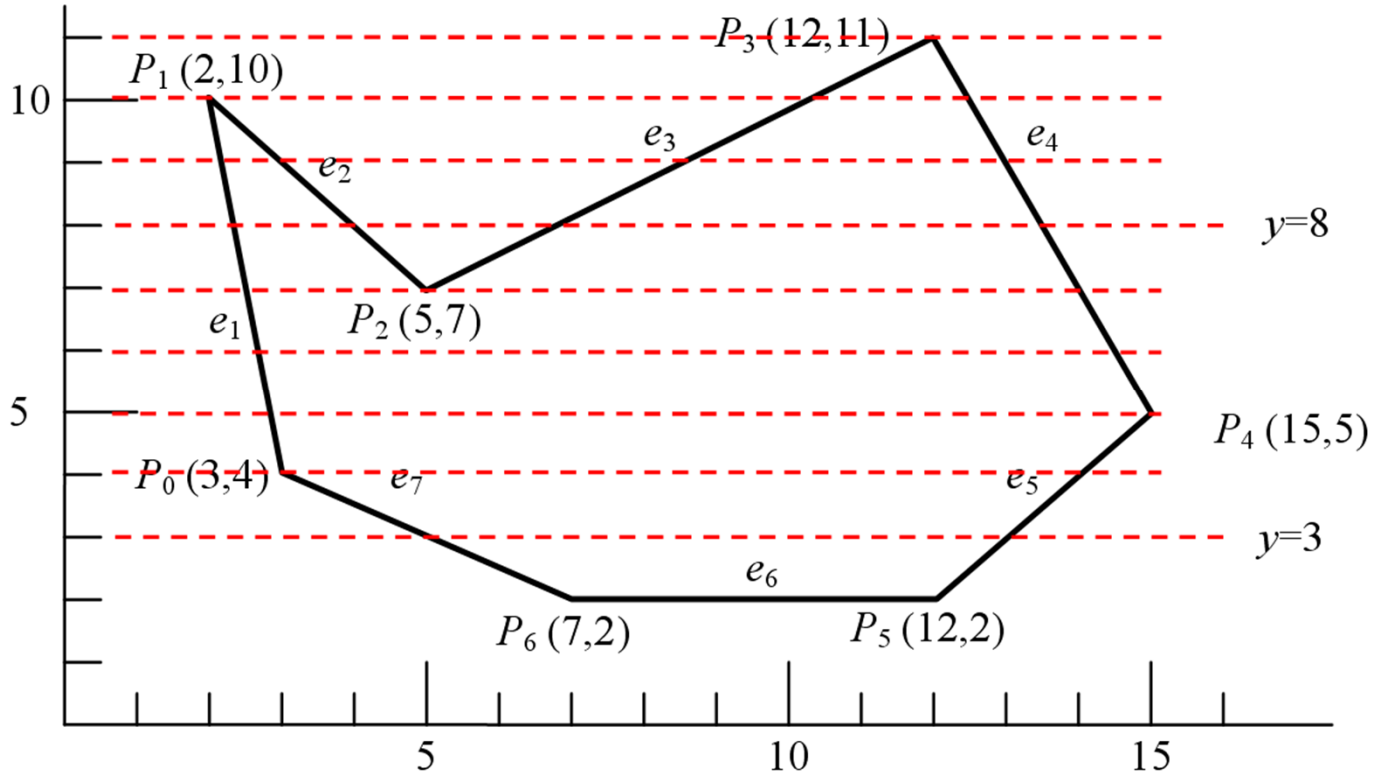

$P_1$ (2,10)
$P_3$ (12,11)
10
$e_2$
$e_3$
$e_4$
$y$=8
$e_1$
$P_2$ (5,7)
5
$P_4$ (15,5)
$P_0$ (3,4)
$e_7$
$e_5$
$y$=3
$e_6$
$P_6$ (7,2)
$P_5$ (12,2)
5
10
15
$y$=7
$e_1$
$e_2$
AEL
10
15/6
-1/6
10
5
-1
11
5
7/4
11
28/2
-1/2
$e_3$
$e_4$

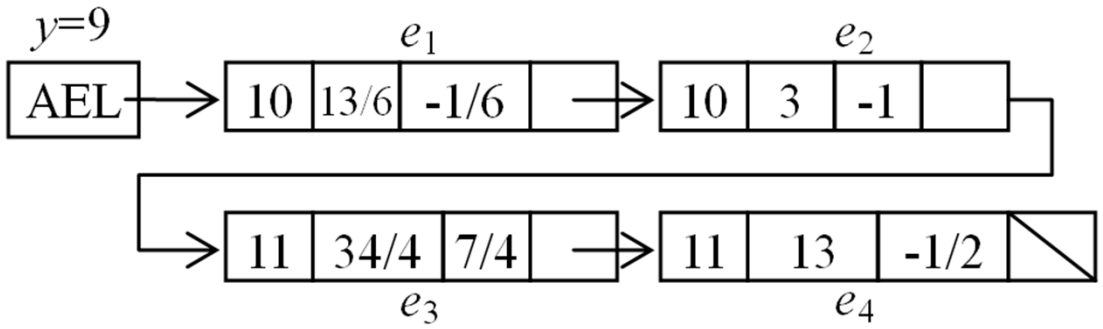
# 多边形扫描转换实例


$P_1$ (2,10)
$P_3$ (12,11)
10
$e_2$
$e_3$
$e_4$
$y$=8
$e_1$
$P_2$ (5,7)
5
$P_4$ (15,5)
$P_0$ (3,4)
$e_7$
$e_5$
$y$=3
$e_6$
$P_6$ (7,2)
$P_5$ (12,2)
5
10
15

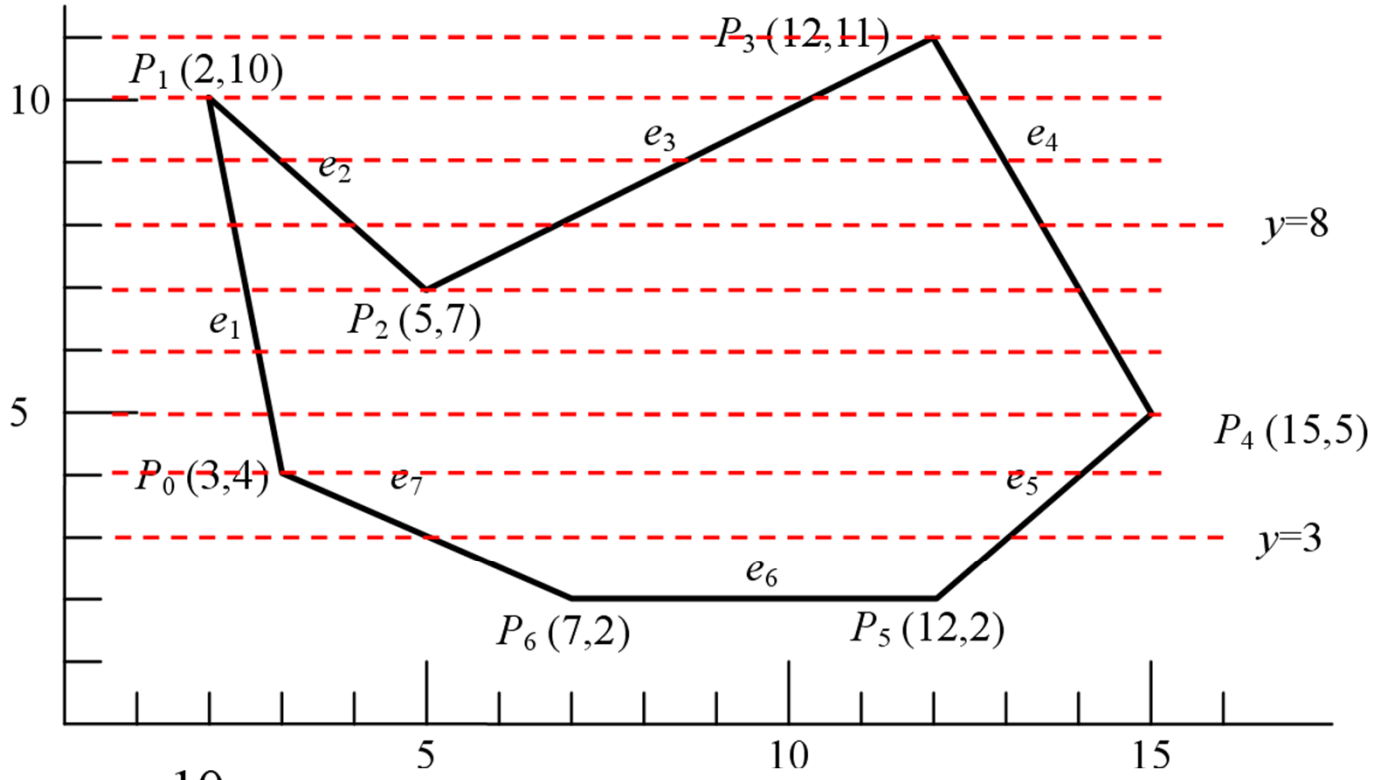

$y$=8
$e_1$
$e_2$
AEL
10
7/3
-1/6
10
4
-1
11
27/4
7/4
11
27/2
-1/2
$e_3$
$e_4$

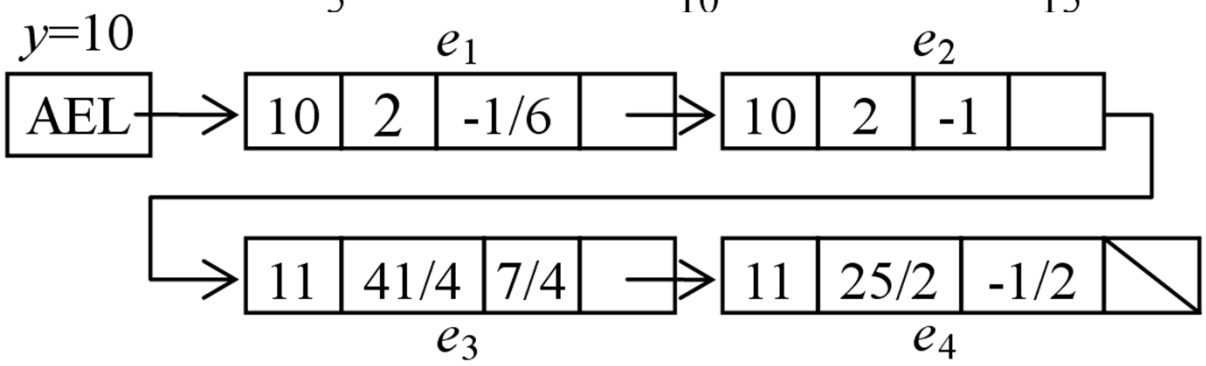
# 多边形扫描转换实例


$P_1$ (2,10)
$P_3$ (12,11)
10
$e_2$
$e_3$
$e_4$
$y$=8
$e_1$
$P_2$ (5,7)
5
$P_4$ (15,5)
$P_0$ (3,4)
$e_7$
$e_5$
$y$=3
$e_6$
$P_6$ (7,2)
$P_5$ (12,2)
5
10
15

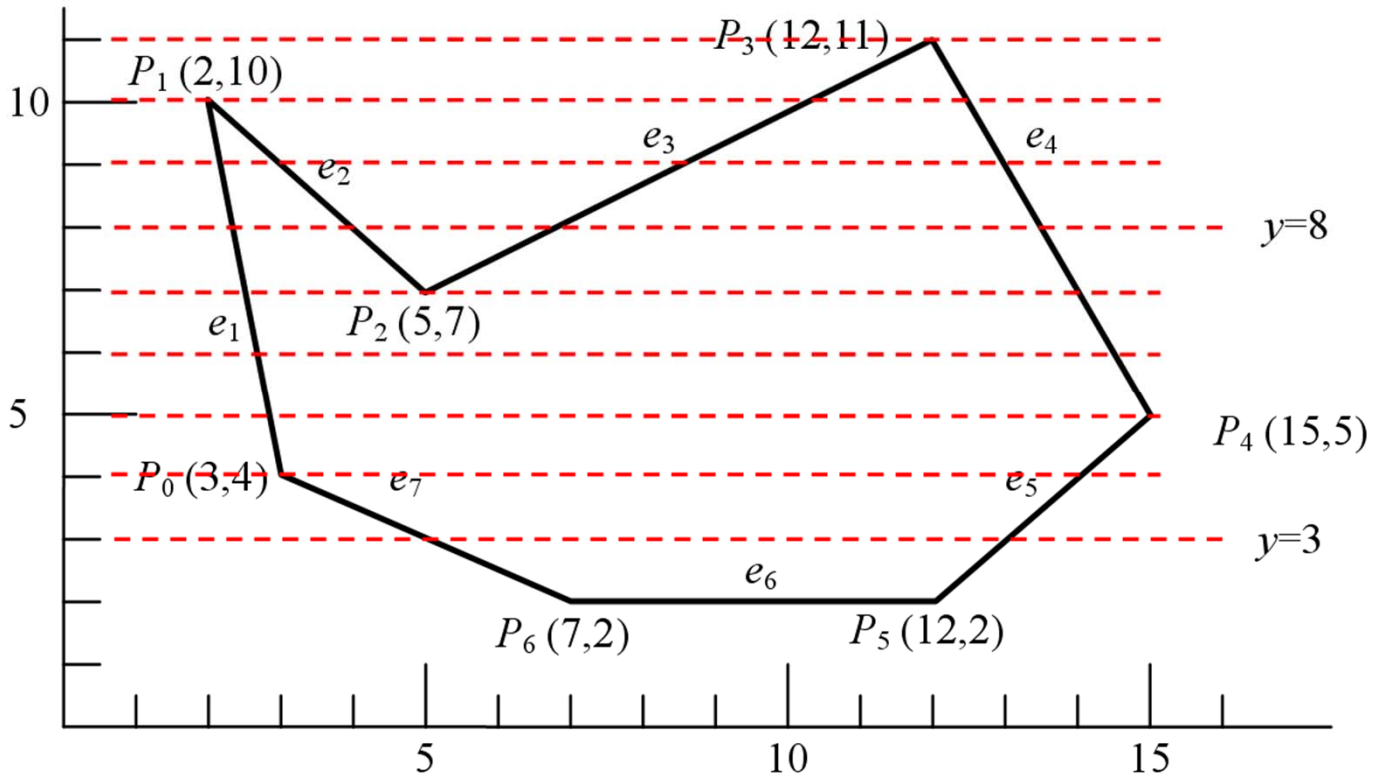

$y$=9
AEL
$e_1$
10
13/6
-1/6
$e_2$
10
3
-1
11
34/4
7/4
$e_3$
11
13
-1/2
$e_4$

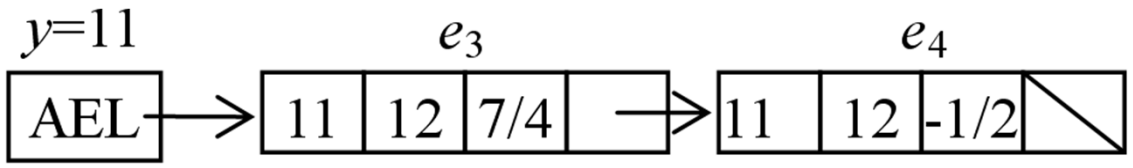
# 多边形扫描转换实例


$P_1$ (2,10)
$P_3$ (12,11)
10
$e_2$
$e_3$
$e_4$
$y$=8
$e_1$
$P_2$ (5,7)
5
$P_4$ (15,5)
$P_0$ (3,4)
$e_7$
$e_5$
$y$=3
$e_6$
$P_6$ (7,2)
$P_5$ (12,2)
5
10
15

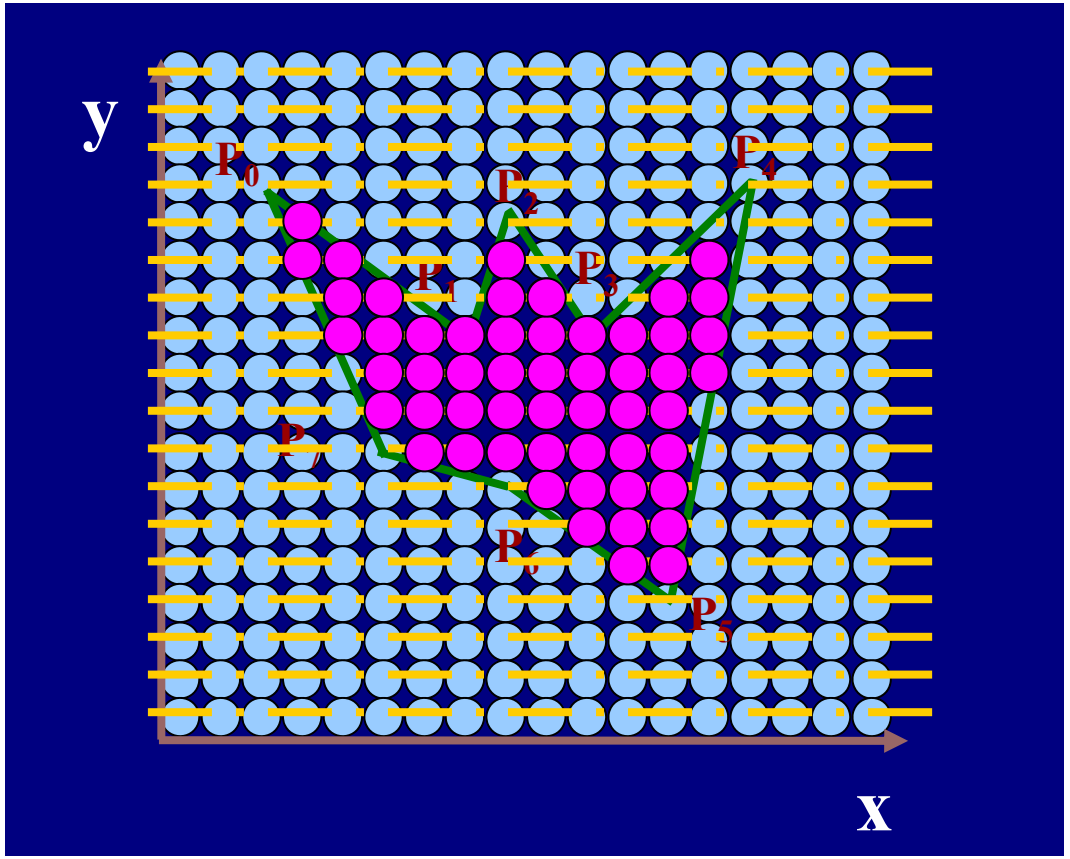

$y$=10
AEL
$e_1$
10
2
-1/6
$e_2$
10
2
-1
11
41/4
7/4
$e_3$
11
25/2
-1/2
$e_4$

# 多边形扫描转换实例


$P_1$ (2,10)
$P_2$ (5,7)
$P_3$ (12,11)
$P_4$ (15,5)
$P_5$ (12,2)
$P_6$ (7,2)
$P_0$ (3,4)
$e_1$
$e_2$
$e_3$
$e_4$
$e_5$
$e_6$
$e_7$
y=8
y=3
10
5
5
10
15


y=11
$e_3$
$e_4$
AEL
11
12
7/4
11
12
-1/2

# 多边形扫描转换算法




扫描转换示意图

# 多边形扫描转换

- 优点
  - 充分利用多边形的区域、扫描线和边的连贯性，避免了反复求交的大量运算
- 不足
  - 算法的数据结构和程序结构复杂
  - 对各种表的维持和排序开销太大，适合软件实现而不适合硬件实现