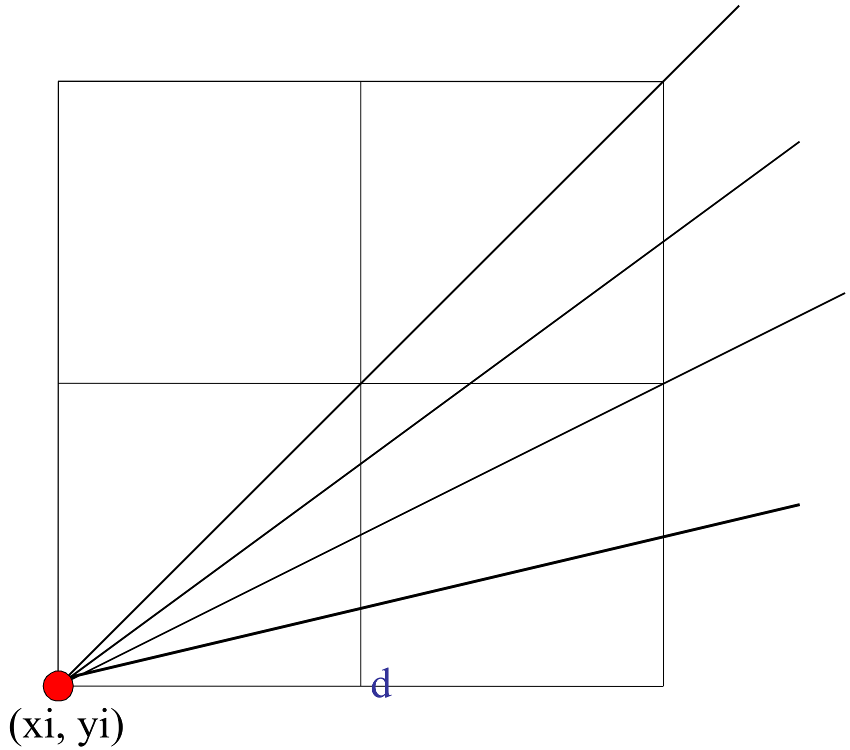
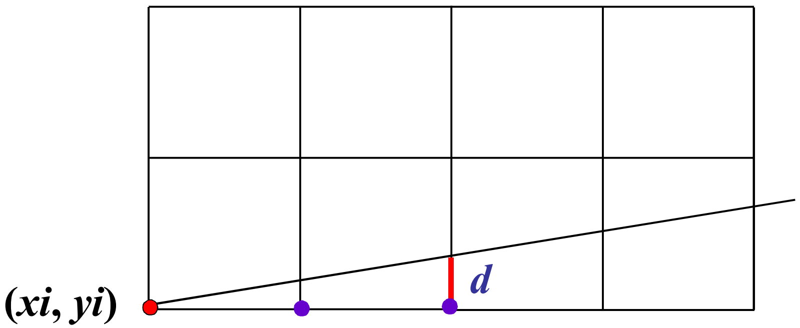
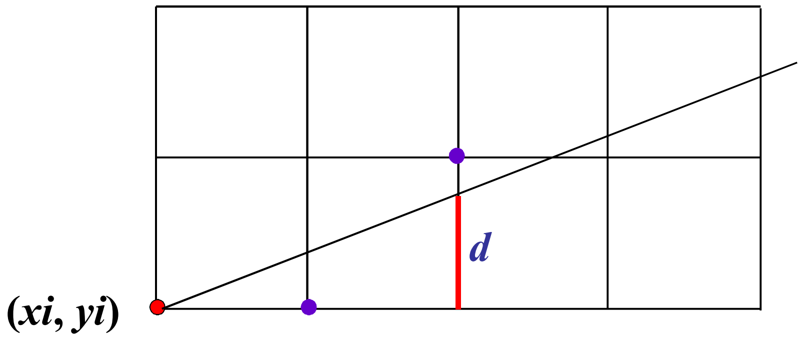
两点的Bresenham方法



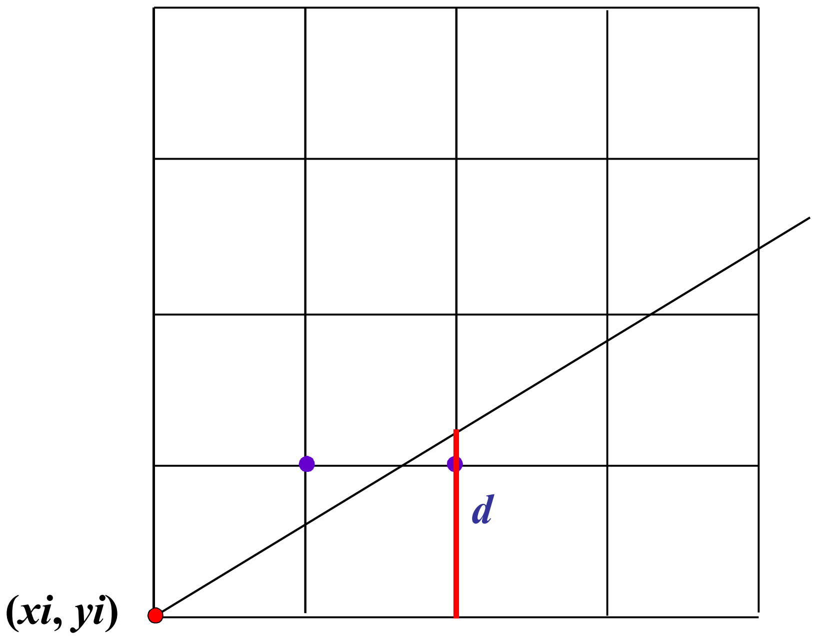
两点的Bresenham方法



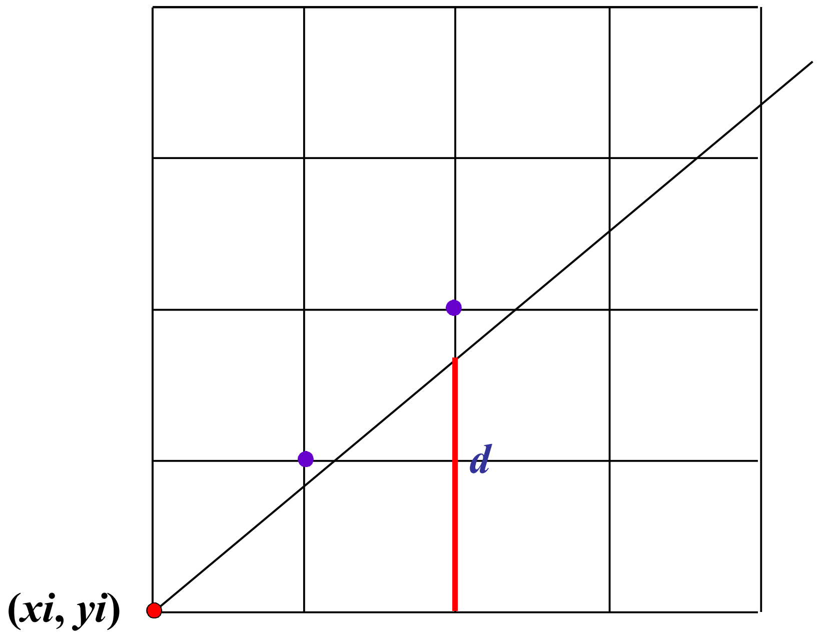
$d \le 0.5$, $d = d + 2k$



$0.5 < d \le 1$, $d = d - 1 + 2k$



$1 < d \le 1.5$, $d = d - 1 + 2k$



$1.5 < d \le 2$, $d = d - 2 + 2k$

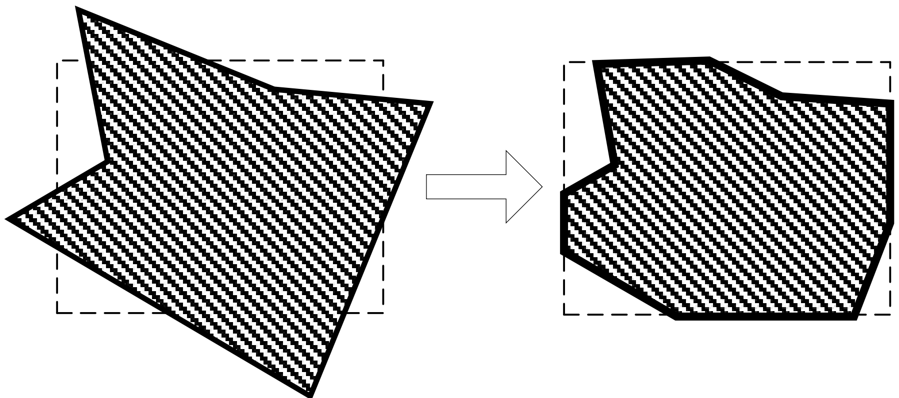
```
d=2k;
if(d>1.5)
{   Draw(Pattern 4);
    y=y+2;   d=d-2;  }
else if(0.5<d <=1.5)
{    if(d>1)  Draw(Pattern 3);
     else Draw(Pattern 2);
     y=y+1;   d=d-1;}
else Draw(Pattern 1);
x=x+2;   d=d+2k;
```

第四章　　裁剪

清华大学软件学院　张慧　2019-09-18

裁剪

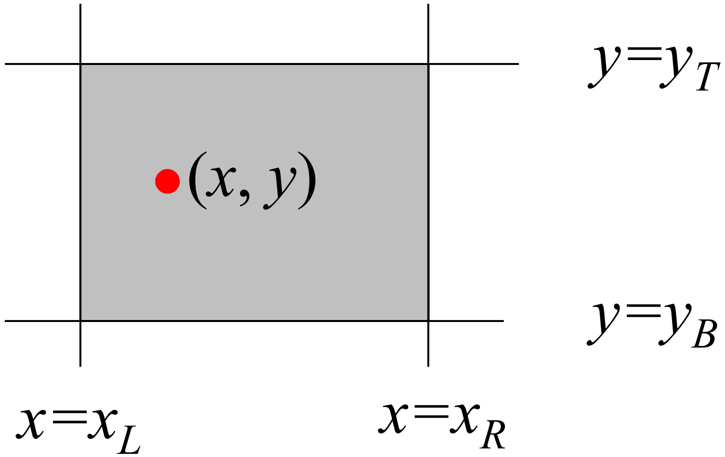
- 裁剪：确定图形中哪些部分落在显示区之内，哪些落在显示区之外, 以便只显示落在显示区内的那部分图形。



  - 裁剪窗口、裁剪对象
  - 裁剪的时机(点阵图形**OR**参数图形)

直线段裁剪

- 直线段裁剪算法是复杂图元裁剪的基础。复杂的曲线可以通过折线段来近似，从而裁剪问题也可以化为直线段的裁剪问题。



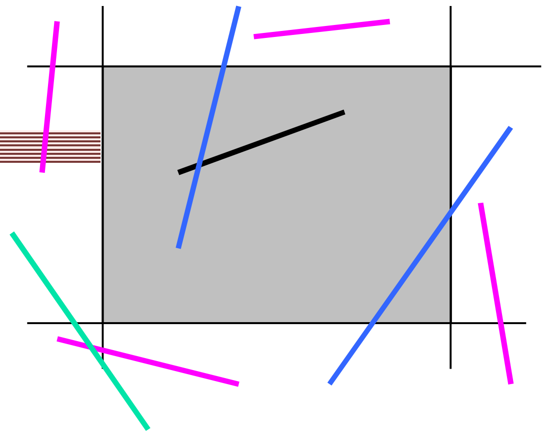
点**(*x*,*y*)**在裁剪窗口内

$\Updownarrow$

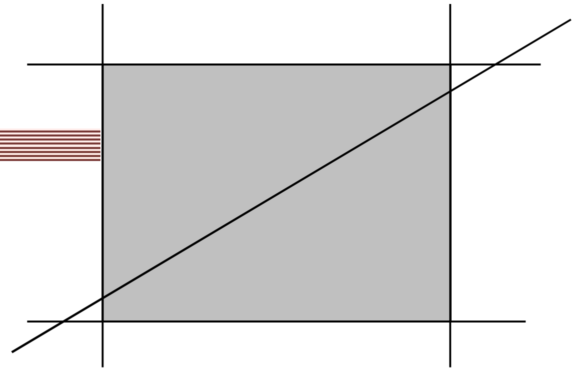
$x_L \le x \le x_R$

$y_B \le y \le y_T$

直接求交法



- 思想：
  - 线段的两个端点都在窗口内，线段必定位于窗口内，即可见；
  - 线段的两个端点同时位于窗口左侧、右侧、上侧或下侧，线段必定位于窗口外，不可见；
  - 线段与窗口各边求交，得到可见部分；



过$P_1(x_1, y_1)$, $P_2(x_1, y_1)$的直线

$y=k(x-x_1)+y_1$ $\quad\quad$ $k=(y_2-y_1)/(x_2-x_1)$

直线与窗口各边所在直线的交点：

左：$x=x_L$，$y=k(x_L-x_1)+y_1$，$k\neq\infty$

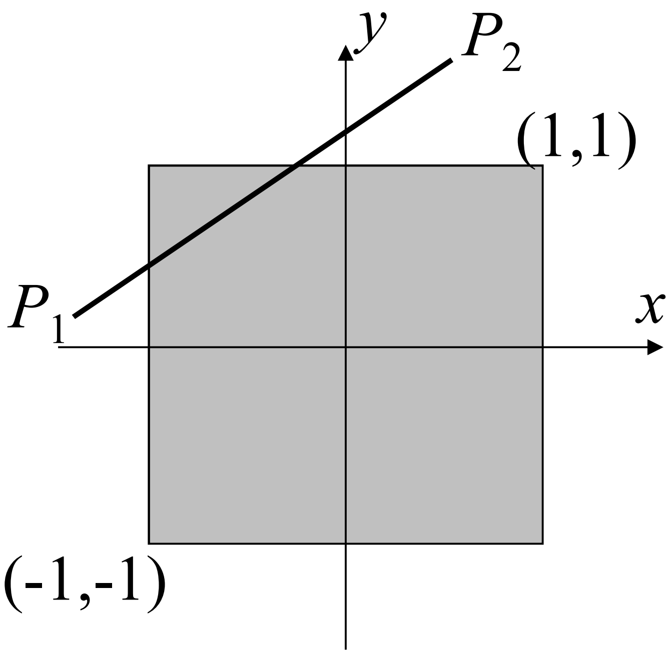
右：$x=x_R$，$y=k(x_R-x_1)+y_1$，$k\neq\infty$

上：$y=y_T$，$x= (y_T-y_1)/k+x_1$，$k\neq 0$

下：$y=y_B$，$x= (y_B-y_1)/k+x_1$，$k\neq 0$

$$k=\frac{y_2-y_1}{x_2-x_1}=\frac{3/2-1/6}{1/2+3/2}=\frac{2}{3}$$

$$y=\frac{2}{3}(x+\frac{3}{2})+\frac{1}{6}$$



P_1P_2与窗口各边的交点：

左边：$x=-1, y=\frac{2}{3}\left[-1-\left(-\frac{3}{2}\right)\right]+\frac{1}{6}=\frac{1}{2}$

右边：$x=1, y=\frac{2}{3}\left[1-\left(-\frac{3}{2}\right)\right]+\frac{1}{6}=\frac{11}{6}$

上边：$y=1, x=-\frac{3}{2}+\frac{3}{2}\left[1-\frac{1}{6}\right]=-\frac{1}{4}$

下边：$y=-1, x=-\frac{3}{2}+\frac{3}{2}\left[-1-\frac{1}{6}\right]=-\frac{13}{4}$

$$P_1\left(-\frac{3}{2},\frac{1}{6}\right), P_2\left(\frac{1}{2},\frac{3}{2}\right)$$

判断有效交点

Cohen-Sutherland裁剪

- 基本思想：对于每条线段P_1P_2分为三种情况处理：
  - 若P_1P_2完全在窗口内，则显示(取)该线段；
  - 若P_1P_2显然完全在窗口外，则丢弃(弃)该线段；
  - 若线段不满足“取”或“弃”的条件，则在交点处把线段分为两段。其中一段完全在窗口外，可弃之。然后对另一段重复上述处理。

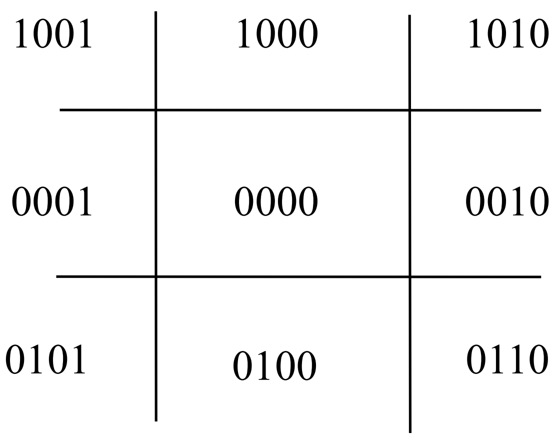
- 为快速判断，采用如下编码方法：
  - 每个区域赋予4位编码$C_tC_bC_rC_l$

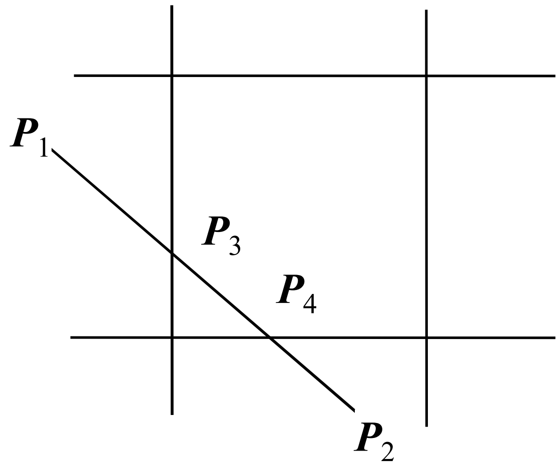
$$C_t=\begin{cases}1, y>y_T\\0, other\end{cases},\qquad C_b=\begin{cases}1, y<y_B\\0, other\end{cases}$$

$$C_r=\begin{cases}1, x>x_R\\0, other\end{cases},\qquad C_l=\begin{cases}1, x<x_L\\0, other\end{cases}$$

1001	1000	1010
0001	0000	0010
0101	0100	0110

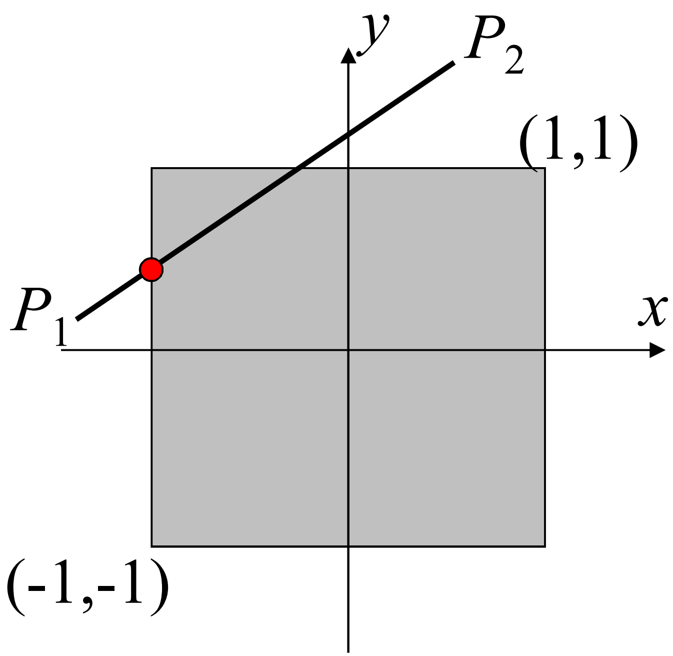
- P_1P_2明显完全在窗口外(code1&code2≠0)，舍弃；
- P_1P_2完全在窗口内(code1=0且code2=0)，绘制；
- 若P_1在窗口内，交换P_1和P_2
- 用P_1P_2和窗口边的交点取代P_1，算法继续





$$P_1\left(-\frac{3}{2},\frac{1}{6}\right),\ P_2\left(\frac{1}{2},\frac{3}{2}\right)\Rightarrow P_1:0001,\ P_2:1000$$

比较两个端点的第一位，…
线段与窗口左边的交点：***P*₁’(-1,1/2)**



$$P_1\left(-\frac{3}{2},\frac{1}{6}\right),P_2\left(\frac{1}{2},\frac{3}{2}\right)$$

$$P_1\left(-1,\frac{1}{2}\right),\ P_2\left(\frac{1}{2},\frac{3}{2}\right)\Rightarrow P_1:0000,\ P_2:1000$$

比较两个端点的第二位，

…

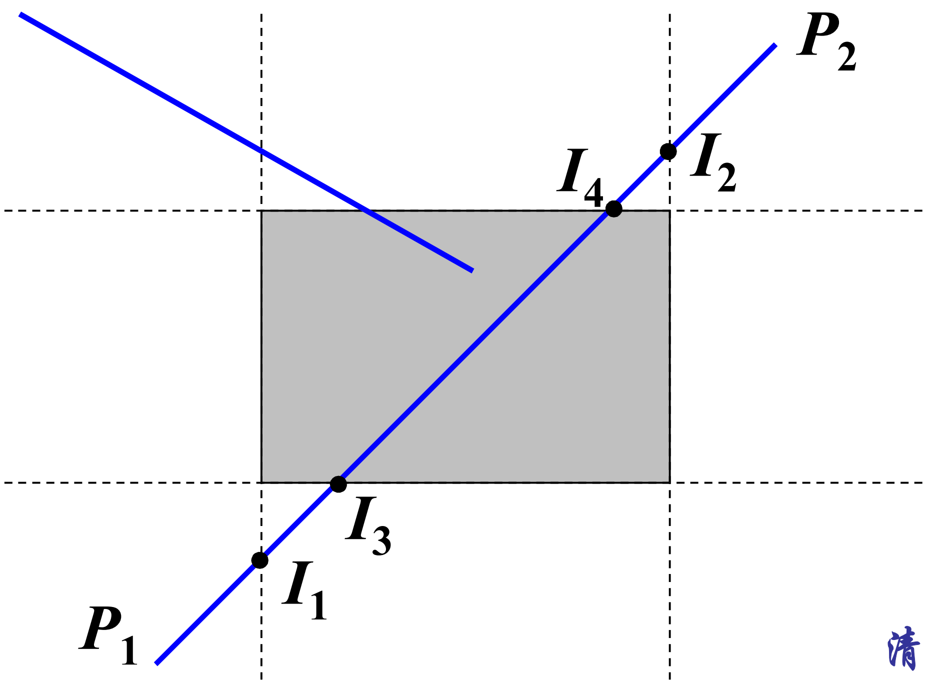
比较两个端点的第四位，…

交换$\boldsymbol{P_1}$和$\boldsymbol{P_2}$

线段与窗口上边的交点：$\boldsymbol{P_1}$**’(-1/4,1)**

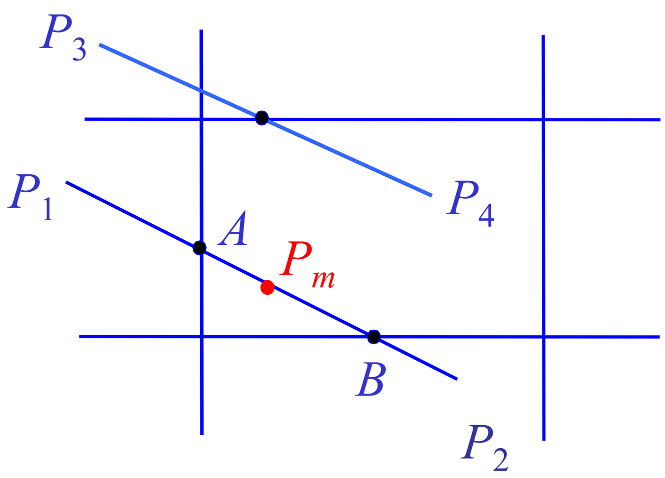
$$P_1\left(-\frac{1}{4},1\right),\ P_2\left(-1,\frac{1}{2}\right)\Rightarrow P_1:0000,\ P_2:0000$$

- 由端点编码可判断线段的可见性；
- 在大窗口和小窗口场合特别高效；
- 求交测试顺序固定；
- 最坏情形，线段求交四次；

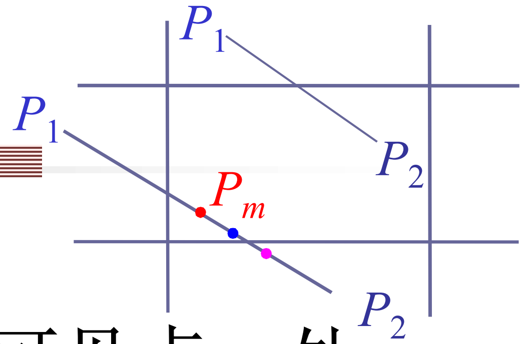


中点分割裁剪算法

- 用二分查找中点代替求交
- 适合硬件实现
- 基本思想：首先对线段端点进行编码，判断完全可见线段和显然不可见线段，对于不能用上述方法判断的线段，由中点将其分割成相等的两段，对每一小段重复上述检查，直到找到线段与窗口的交点或子段长度足够小。



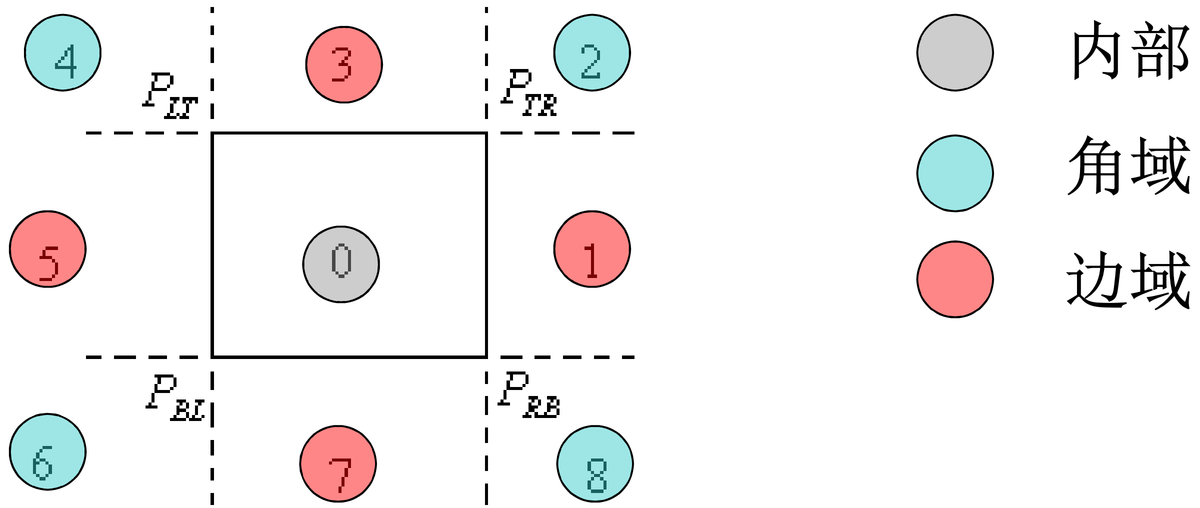
- 可见点：线段落在窗口内的点
- ***B***、***A***分别为距离$\boldsymbol{P_1}$、$\boldsymbol{P_2}$最远的可见点，$\boldsymbol{P_m}$为线段$\boldsymbol{P_1P_2}$的中点



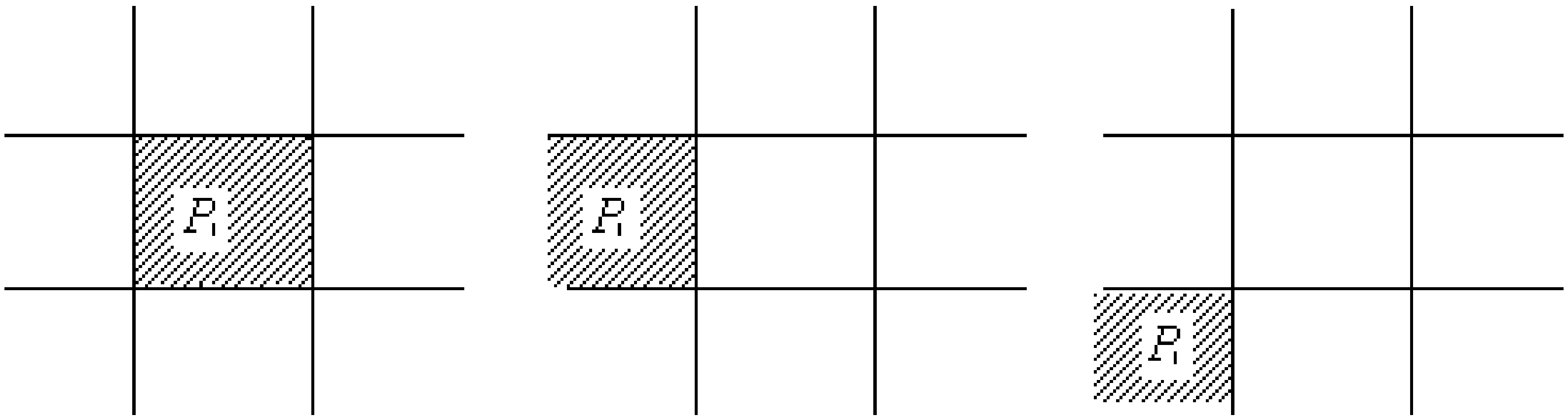
- 找距离端点P_1最远的可见点
  - 若P_2可见，则P_2为距离P_1最远的可见点，处理结束；
  - 在中点P_m处将线段P_1P_2分成两小段；
  - 若P_m可见，用P_mP_2代替P_1P_2，在P_mP_2中找最远的可见点；
  - 若P_m不可见，
    - P_1P_m完全在窗口外，在P_mP_2中找最远的可见点；
    - P_mP_2完全在窗口外，在P_1P_m中找最远的可见点；
  - 重复上述过程，直到线段长度小于给定的控制常数为止，此时P_m收敛于交点。

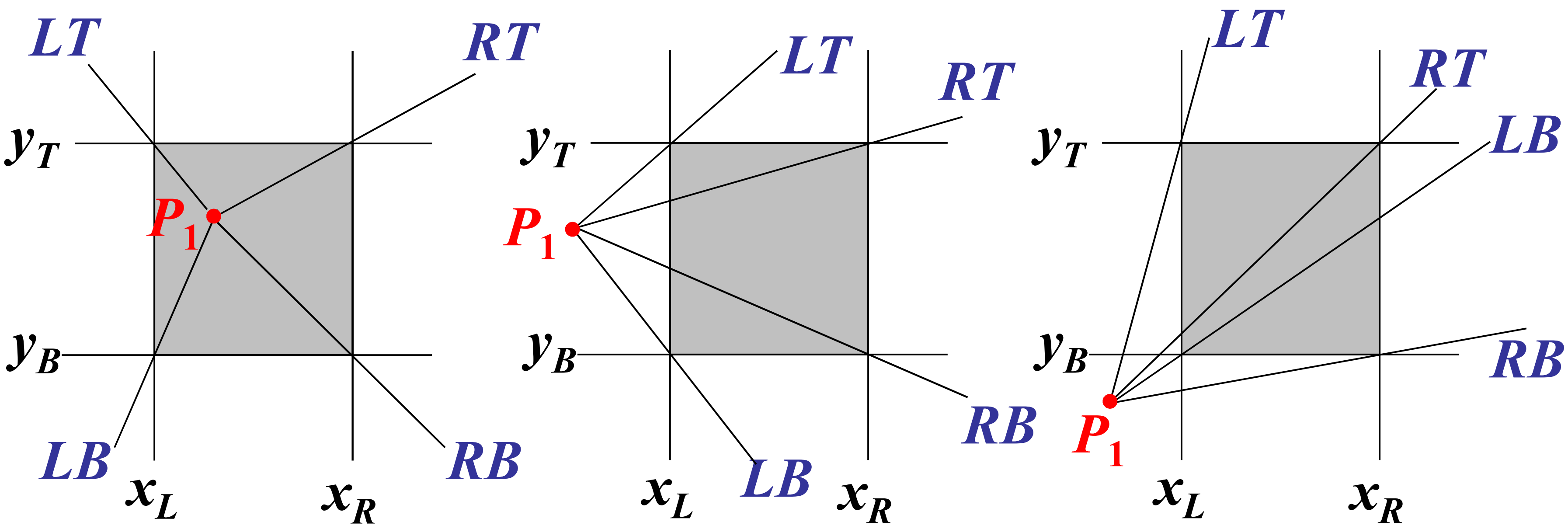
Nicholl-Lee-Nicholl算法

- 消除**C-S**算法中多次求交的情况。
- 基本想法：对**2D**平面的更细的划分。



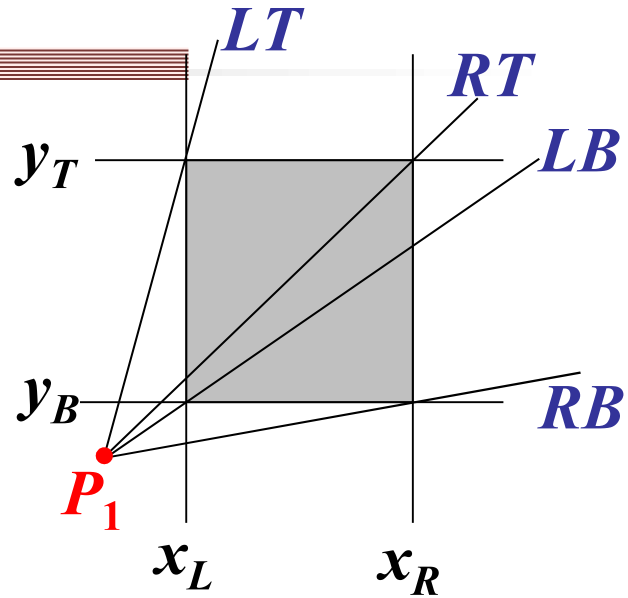
- 算法：
  - 假定P_1点落在区域0，5，6；
  - 若P_1在其他区域，先进行一些变换，使之位于0，5，6区域；





- P_1点向窗口的四角点引射线(*LT*, *RT*, *LB*, *RB*)，把平面区域分成4个区域。
- 由P_2所在区域位置，可判定P_1P_2与窗口哪一条/两条边求交。

- 通过比较P_1P_2与四条过角点的直线的斜率，来确定直线的可见性、交点个数和交点所在的边界。



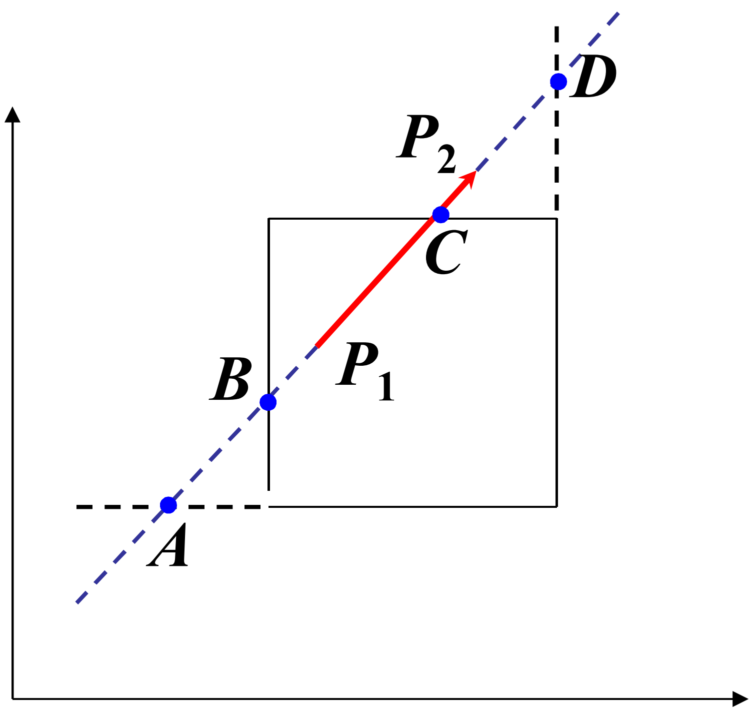
假设P_1位于左下角域，P_2位于裁剪窗口之外的任何区域：

- $k>k_{LT}$：显然不可见；
- $k<k_{RB}$：显然不可见；
- $k=k_{RB}$：一个交点；
- $k_{RB}<k\leq k_{LB}$：与底边、右边相交；
- $k_{LB}<k\leq k_{RT}$：与左边、右边相交；
- $k_{RT}<k<k_{LT}$：与左边、顶边相交；
- $k=k_{LT}$：一个交点；

- 考虑的情况：
  - P_1位于窗口内部/角域/边域，P_2位于窗口内部/外部；
- 特点：效率较高，但仅适合二维矩形窗口。

梁友栋－Barsky算法

- 线段及其延长线与裁剪窗口的四条边所在的直线有**4**个交点。
- 始边(入点)、终边(出点)
- 裁剪后的线段的端点：从A, B, P_1中找距离P_2最近的点；从C, D, P_2中找距离P_1最近的点。



- 线段$P_1(x_1, y_1)$, $P_2(x_2, y_2)$的参数表示：

$$P(t)=P_1+(P_2-P_1)t,\quad 0\le t\le 1$$

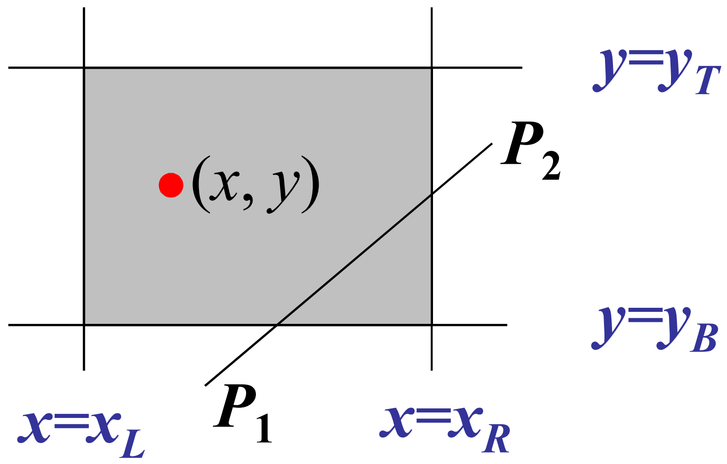
$$x(t)=x_1+(x_2-x_1)t$$

$$y(t)=y_1+(y_2-y_1)t$$

- 裁剪条件：

$$x_L\le x_1+(x_2-x_1)t \le x_R$$

$$y_B\le y_1+(y_2-y_1)t \le y_T$$



- $x_L \le x_1+\Delta x \cdot t \le x_R$, $y_B \le y_1+\Delta y \cdot t \le y_T$
- 即：

$-\Delta x \cdot t \le x_1 - x_L$, $-\Delta y \cdot t \le y_1 - y_B$

$\Delta x \cdot t \le x_R - x_1$, $\Delta y \cdot t \le y_T - y_1$

- 可以统一表示为形式： $p_i \cdot t \le q_i$, $i=1,2,3,4$

其中：

$$p_1 = -\Delta x, q_1 = x_1 - x_L; \quad p_2 = \Delta x, q_2 = x_R - x_1$$

$$p_3 = -\Delta y, q_3 = y_1 - y_B; \quad p_4 = \Delta y, q_4 = y_T - y_1$$

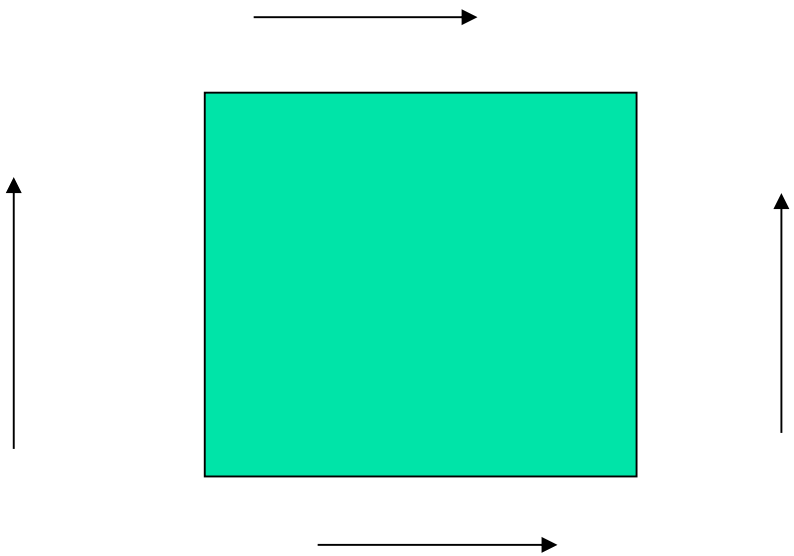
- p_i=0
  - q_i<0，该线段完全在边界外；
  - q_i≥0，该线段平行于裁剪边界并在窗口内。

$$p_1 = -\Delta x, q_1 = x_1 - x_L$$

$$p_2 = \Delta x, q_2 = x_R - x_1$$

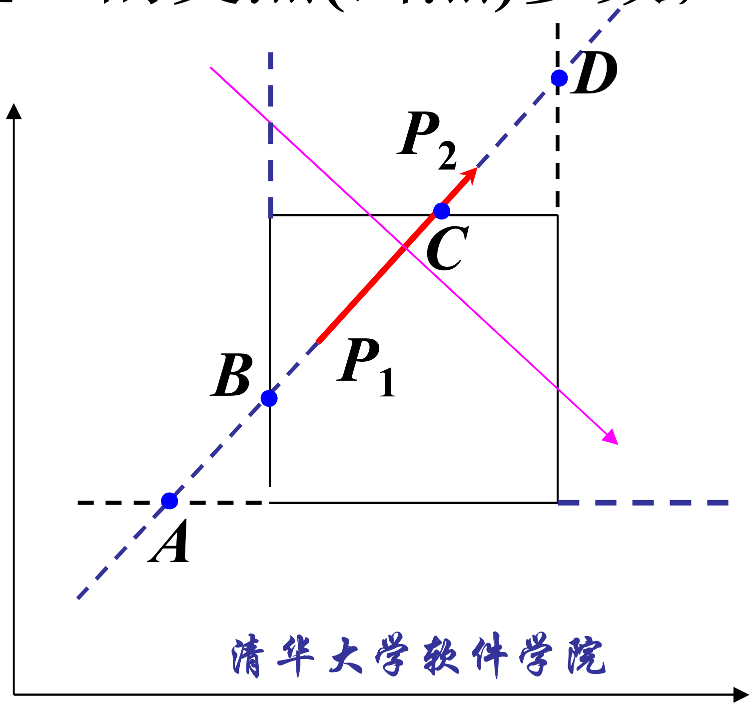
$$p_3 = -\Delta y, q_3 = y_1 - y_B$$

$$p_4 = \Delta y, q_4 = y_T - y_1$$



- 当$p_i \neq 0$， $t_i = q_i / p_i$,i=1,2,3,4
  - 当$p_i<0$，线段从裁剪边界延长线的外部延伸到内部，即t_i为始边上的交点(入点)参数;
  - 当$p_i>0$，线段从裁剪边界延长线的内部延伸到外部，即t_i为终边上的交点(出点)参数;

$$p_1 = -\Delta x, q_1 = x_1 - x_L$$
$$p_2 = \Delta x, q_2 = x_R - x_1$$
$$p_3 = -\Delta y, q_3 = y_1 - y_B$$
$$p_4 = \Delta y, q_4 = y_T - y_1$$

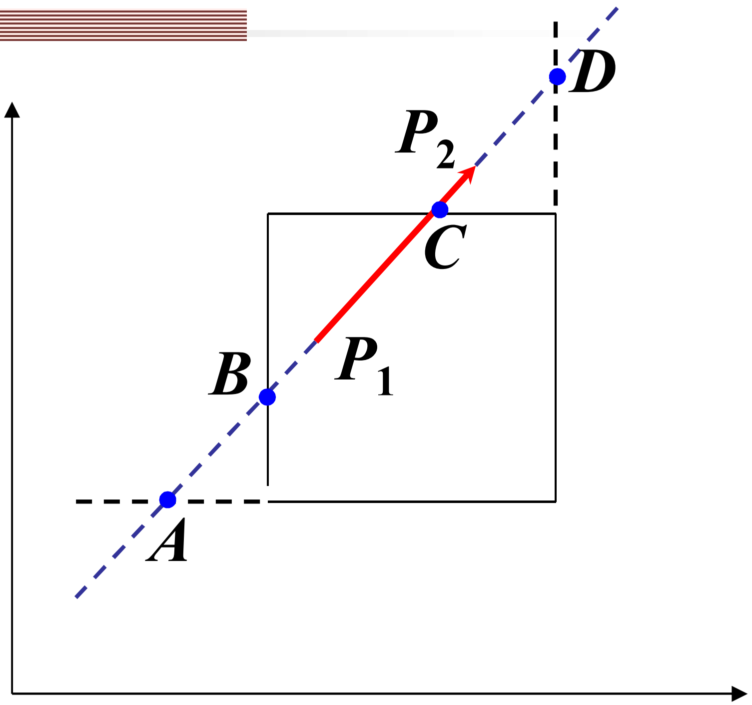


- 假设入点参数为t_1', t_1''，距离P_2最近的点的参数t_1为：

$$t_{min} = \max(t_1', t_1'', 0)$$

- 假设出点参数为t_2', t_2''，距离P_1最近的点的参数t_2为：

$$t_{max} = \min(t_2', t_2'', 1)$$

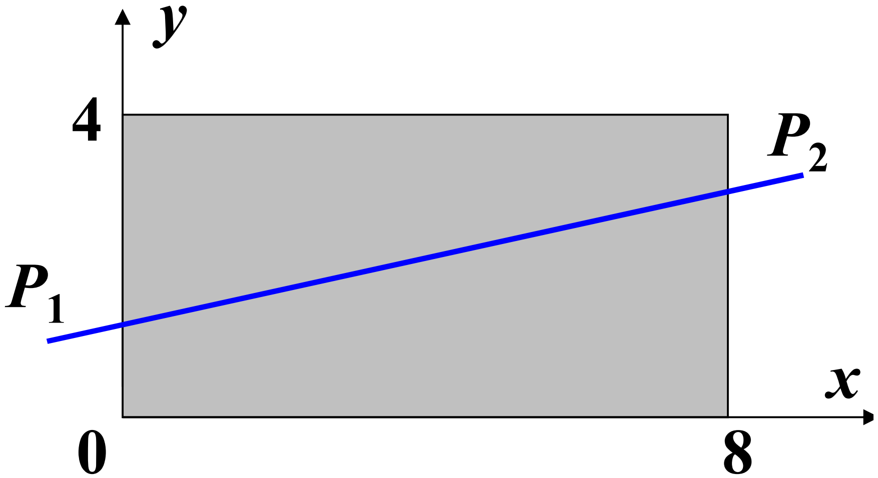


- 对于被裁剪直线，可计算出参数t_1和t_2，它们定义了在裁剪窗口内的线段部分
  - t_1的值由线段与裁剪窗口始边($p_i<0$)的交点决定。对始边计算$r_i=q_i/p_i$，t_1取0和各个r_i值之中的最大值。
  - t_2的值由线段与裁剪窗口终边($p_i>0$)的交点决定。对终边计算$r_i=q_i/p_i$，t_2取1和各个r_i值之中的最小值。
  - 如果$t_1>t_2$，则线段完全落在裁剪窗口之外，被舍弃。
  - 否则由t_1,t_2计算裁剪后可见线段的端点。

```
void LB_LineClip(x1,y1,x2,y2,XL,XR,YB,YT)
float x1,y1,x2,y2,XL,XR,YB,YT;
{   float dx,dy,u1,u2;
    u1=0;u2=1;          dx =x2-x1;dy =y2-y1;
     if(ClipT(-dx,x1-XL,&u1,&u2)
     if(ClipT(dx,XR-x1, &u1,&u2)
     if(ClipT(-dy,y1-YB, &u1,&u2)
     if(ClipT(dy,YT-y1, &u1,&u2)
     { displayline(x1+u1*dx,y1+u1*dy, x1+u2*dx,y1+u2*dy);
        return;     }
 }
```

```
bool ClipT(p,q,u1,u2)
float p,q,*u1,*u2;
{  float r;
     if(p<0)
     {       r=q/p;
             if(r>*u2)  return FALSE;
             else if(r>*u1)
             { *u1=r;return TRUE;}
     }
     else if(p>0)
     {       r=q/p;
             if(r<*u1)return FALSE;
             else if(r<*u2)
            { *u2=r;return TRUE;}
     }
     else if(q<0) return FALSE;
     return TRUE;
}
```

$$P(t)=\begin{bmatrix}-1\\1\end{bmatrix}+\begin{bmatrix}10\\2\end{bmatrix}t$$



P_1(-1,1), P_2(9,3)

$$p_1=-10,\ q_1=-1$$

$$p_2=10,\quad q_2=9$$

$$p_3=-2,\quad q_1=1$$

$$p_4=2,\quad q_4=3$$

$$p_1=-\Delta x, q_1=x_1-x_L$$

$$p_2=\Delta x, q_2=x_R-x_1$$

$$p_3=-\Delta y, q_3=y_1-y_B$$

$$p_4=\Delta y, q_4=y_T-y_1$$

始边上的交点参数：$t_1=\frac{1}{10}$，$t_3=-\frac{1}{2}$

终边上的交点参数：$t_2=\frac{9}{10}$，$t_4=\frac{3}{2}$

$$t_{\min}=\max\left(0,\frac{1}{10},-\frac{1}{2}\right)=\frac{1}{10}$$

$$t_{\max}=\min\left(1,\frac{9}{10},\frac{3}{2}\right)=\frac{9}{10}$$

交点：$\left(0,\frac{6}{5}\right),\left(8,\frac{14}{5}\right)$

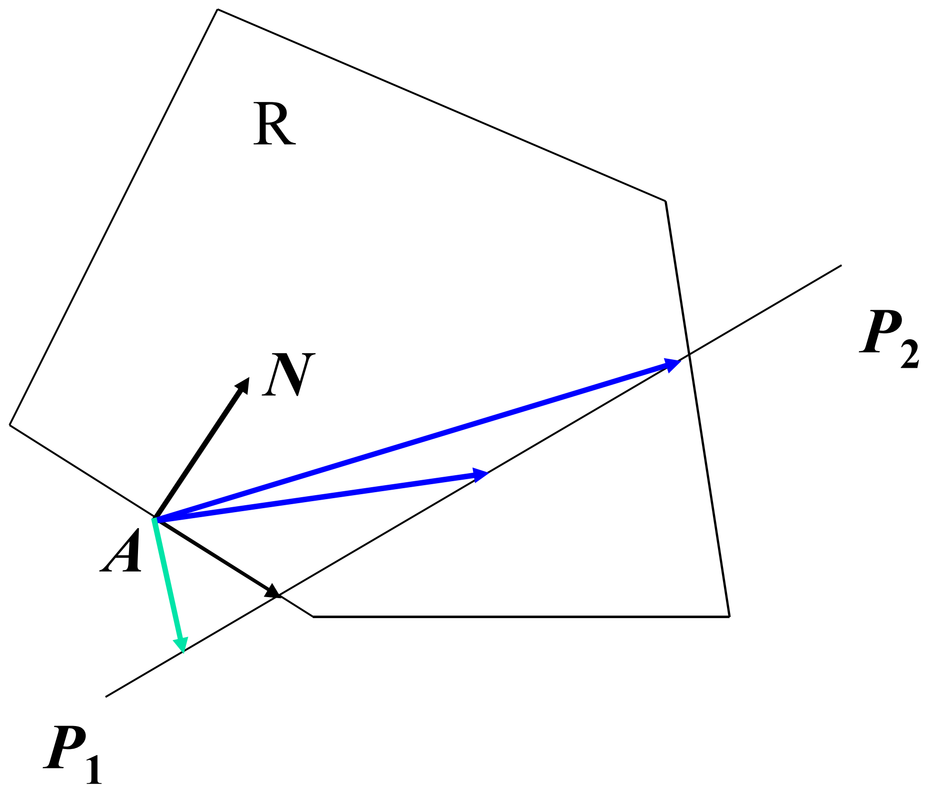
参数化算法(Lyrus-Beck算法)

- 考虑凸多边形区域R和直线段P_1P_2

$$P(t)=(P_2-P_1)\cdot t+P_1$$

- 凸多边形的性质：$P(t)$在凸多边形内的充要条件是，对于凸多边形边界上任意一点A和该点处内法向N，都有：

$$N\cdot(P(t)-A)>0$$

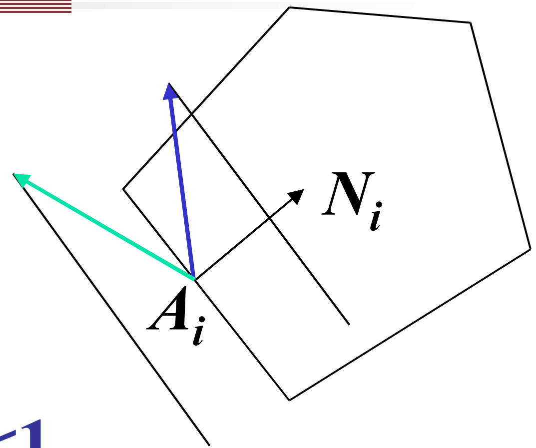

R
N
A
P_2
P_1

- 对多边形各边：

$$N_i \cdot (P(t) - A_i) \geqslant 0 \quad \rightarrow$$

$$N_i \cdot [(P_2 - P_1) \cdot t + P_1 - A_i] \geqslant 0$$

$$N_i \cdot (P_1 - A_i) + N_i \cdot (P_2 - P_1) \cdot t \geqslant 0, \quad 0 \le t \le 1$$



- 如果 $N_i \cdot (P_2 - P_1) = 0 \Rightarrow N_i \perp (P_2 - P_1)$

  当 $N_i \cdot (P_1 - A_i) > 0$ 时，线段在区域内侧

  当 $N_i \cdot (P_1 - A_i) < 0$ 时，线段在区域外侧

- 如果$N_i\cdot(P_2-P_1)\neq 0$

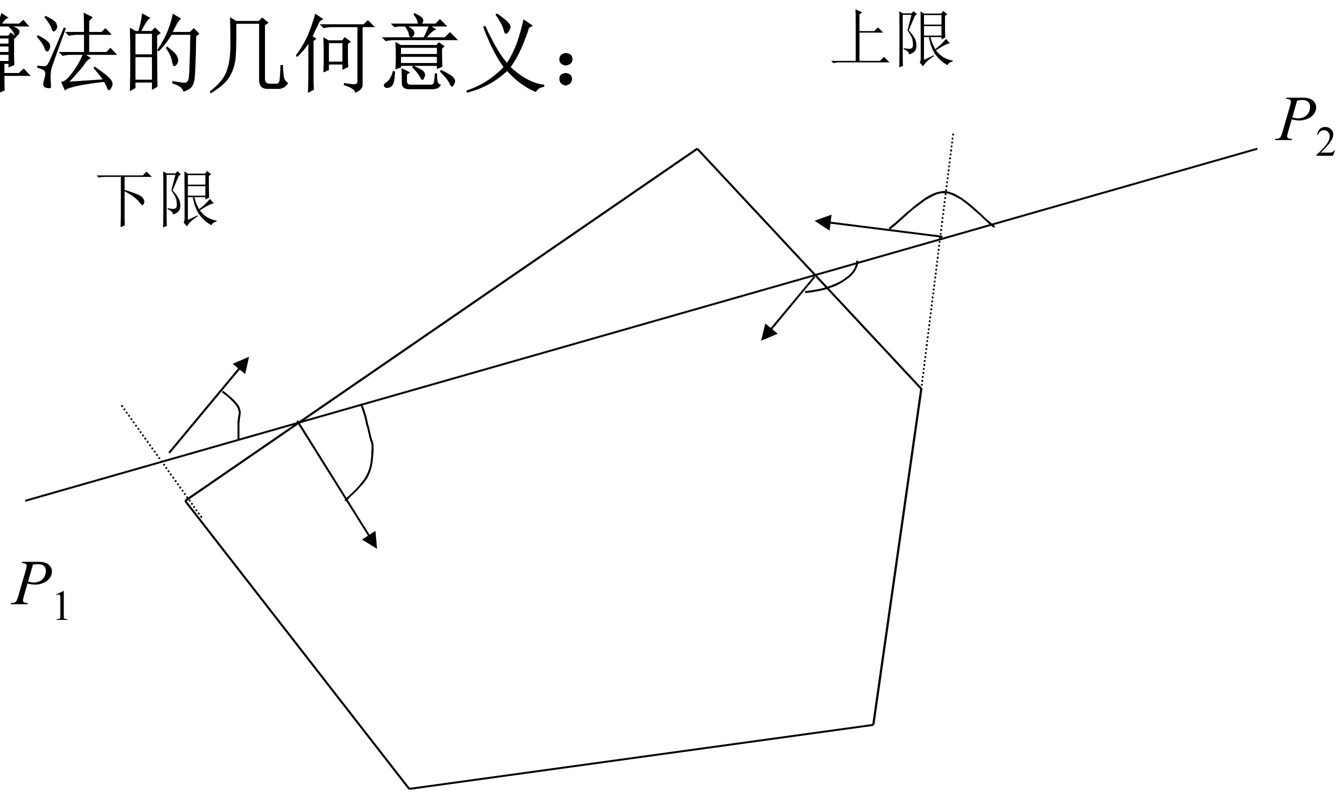
由$N_i\cdot(P_1-A_i)+N_i\cdot(P_2-P_1)\cdot t\geqslant 0$，　$0\leq t\leq 1$

$$当N_i\cdot(P_2-P_1)>0时，\quad t\geq -\frac{N_i\cdot(P_1-A_i)}{N_i\cdot(P_2-P_1)}$$

$$当N_i\cdot(P_2-P_1)<0时，\quad t\leq -\frac{N_i\cdot(P_1-A_i)}{N_i\cdot(P_2-P_1)}$$

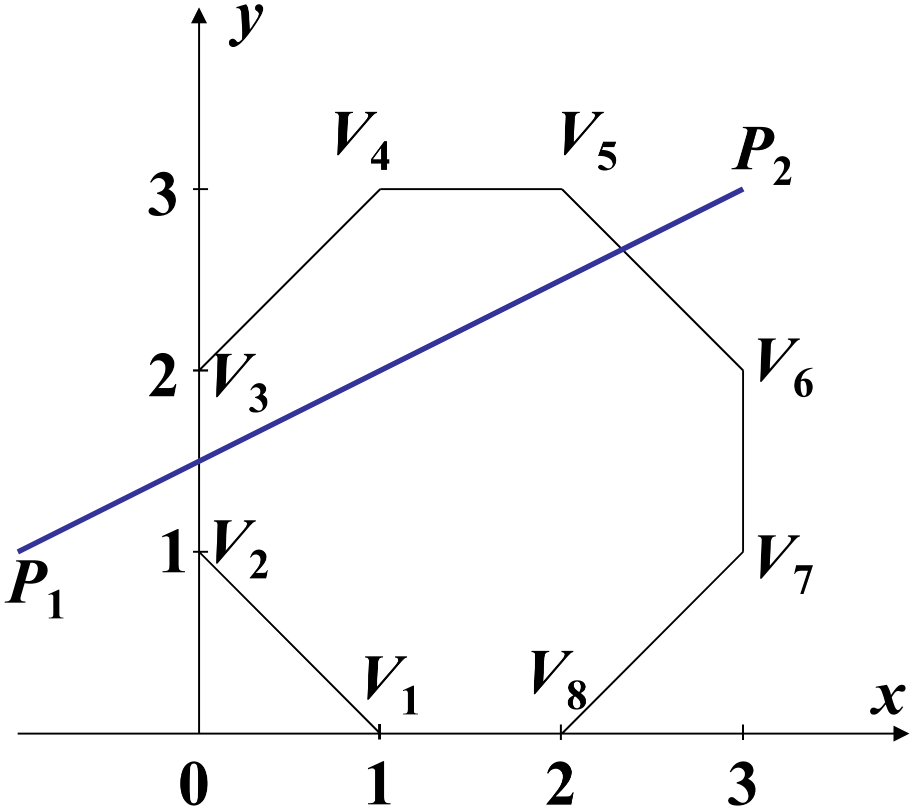
- 线段可见的交点参数：
  - $t_{min}=\max\{0, \max\{t_i: N_i\cdot(P_2-P_1)>0\}\}$
  - $t_{max}=\min\{1, \min\{t_i: N_i\cdot(P_2-P_1)<0\}\}$
- 若 $t_{min}\leqslant t_{max}$, $[t_{min}, t_{max}]$是可见线段的交点参数区间，否则，线段不可见。

- 算法的几何意义：



当凸多边形是矩形窗口且矩形的边与坐标轴平行时，算法退化为**Liang-Barsky**算法。

计算题



$P_1(-1,1)$, $P_2(3,3)$

$$P(t)=\begin{bmatrix}-1\\1\end{bmatrix}+\begin{bmatrix}4\\2\end{bmatrix}t$$

当$N_i\cdot(P_2-P_1)>0$时，

$$t\ge-\frac{N_i\cdot(P_1-A_i)}{N_i\cdot(P_2-P_1)}$$

当$N_i\cdot(P_2-P_1)<0$时，

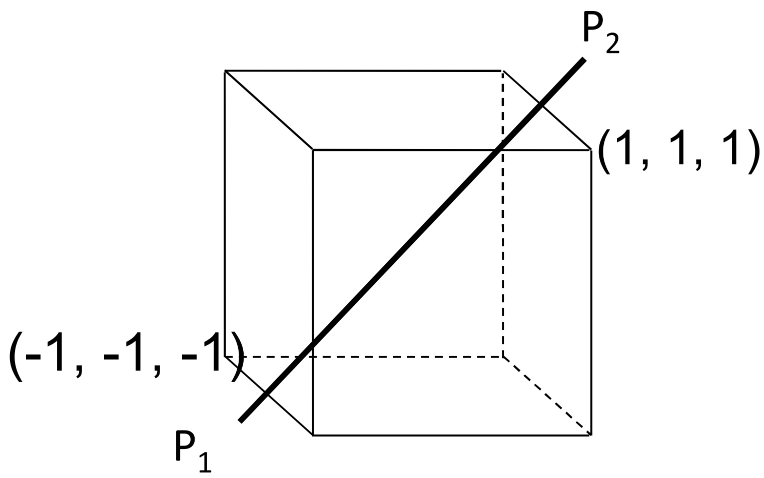
$$t\le-\frac{N_i\cdot(P_1-A_i)}{N_i\cdot(P_2-P_1)}$$

思考题

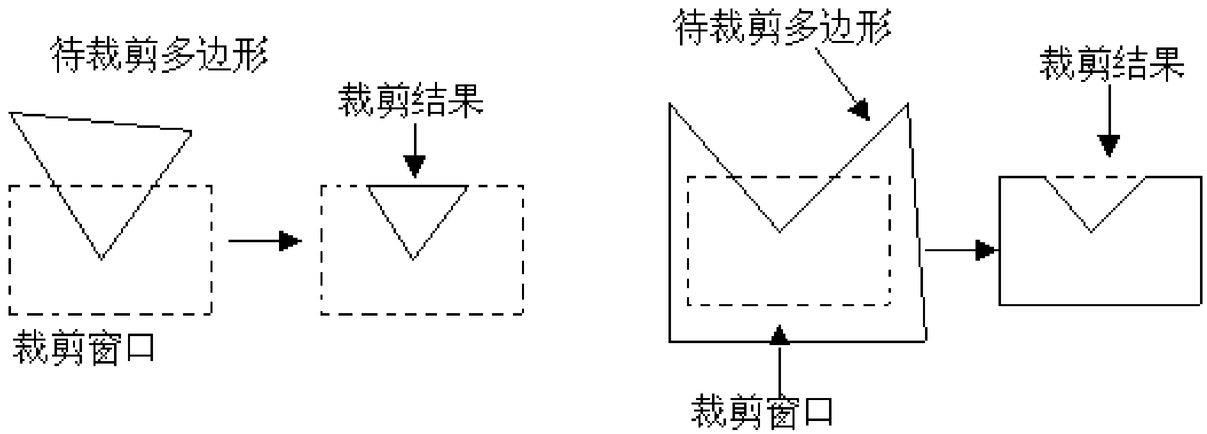
- 直线段的三维裁剪
  - 直接求交、**Cohen-Sutherland**、中点分割、 **Nicholl-Lee-Nicholl**、**Liang- Barsky**、**Lyrus-Beck**

$P_1(-2, -1, 1/2)$

$P_2(3/2, 3/2, -1/2)$



多边形裁剪



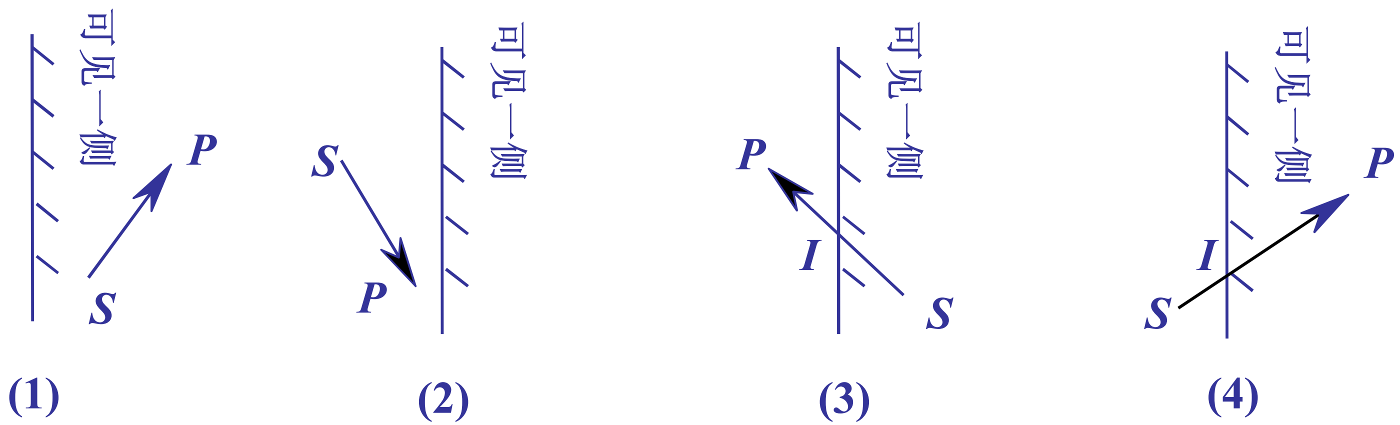
- 把多边形分解成线段逐段进行裁剪
  - 边界的封闭性如何保证?

Sutherland-Hodgeman算法

- 多边形用顶点序列表示
- 逐次多边形裁剪
- 基本思想：一次用窗口的一条边裁剪多边形，裁剪结果传给下一条窗口边继续裁剪。
- 考虑窗口的一条边及延长线构成的裁剪线

  该线把平面分成两个部分:可见/不可见

- 多边形各边*SP*与裁剪线的位置关系：

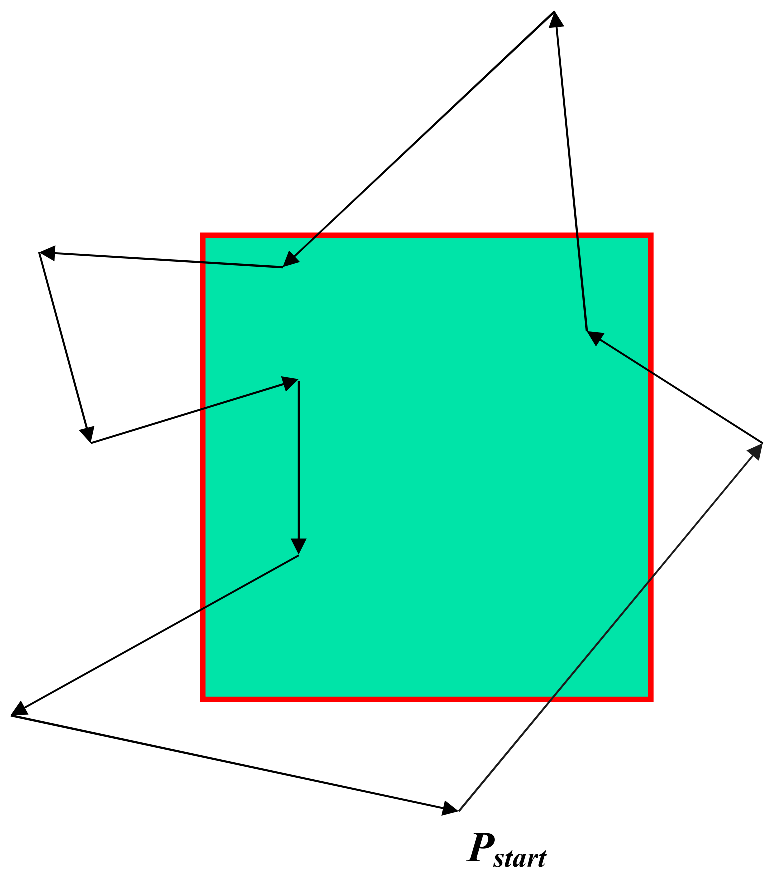


- (1)：仅输出顶点*P*；
- (2)：输出0个顶点；
- (3)：输出线段*SP*与裁剪线的交点*I*；
- (4)：输出线段*SP*与裁剪线的交点*I*和终点*P*

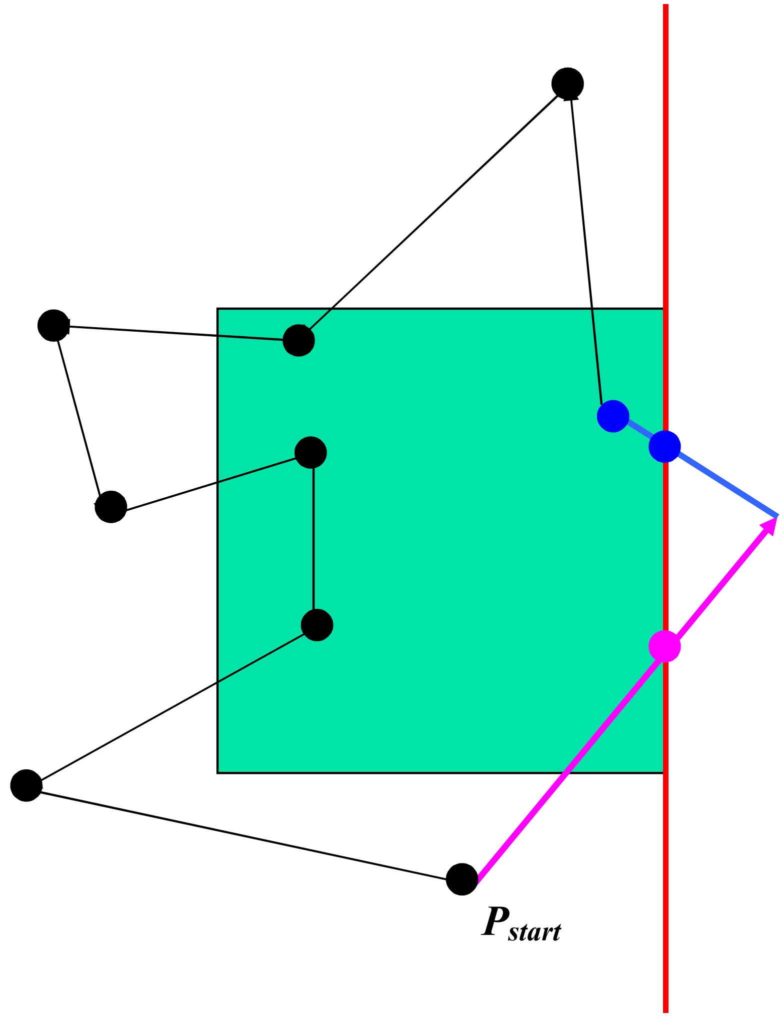
- 上述算法仅用一条裁剪边对多边形进行裁剪，得到一个顶点序列，作为下一条裁剪边处理过程的输入。

- 对于每一条裁剪边，只是判断点在窗口哪一侧以及求线段***SP***与裁剪边的交点算法应随之改变。
  - 位置关系判断
  - 交点求解

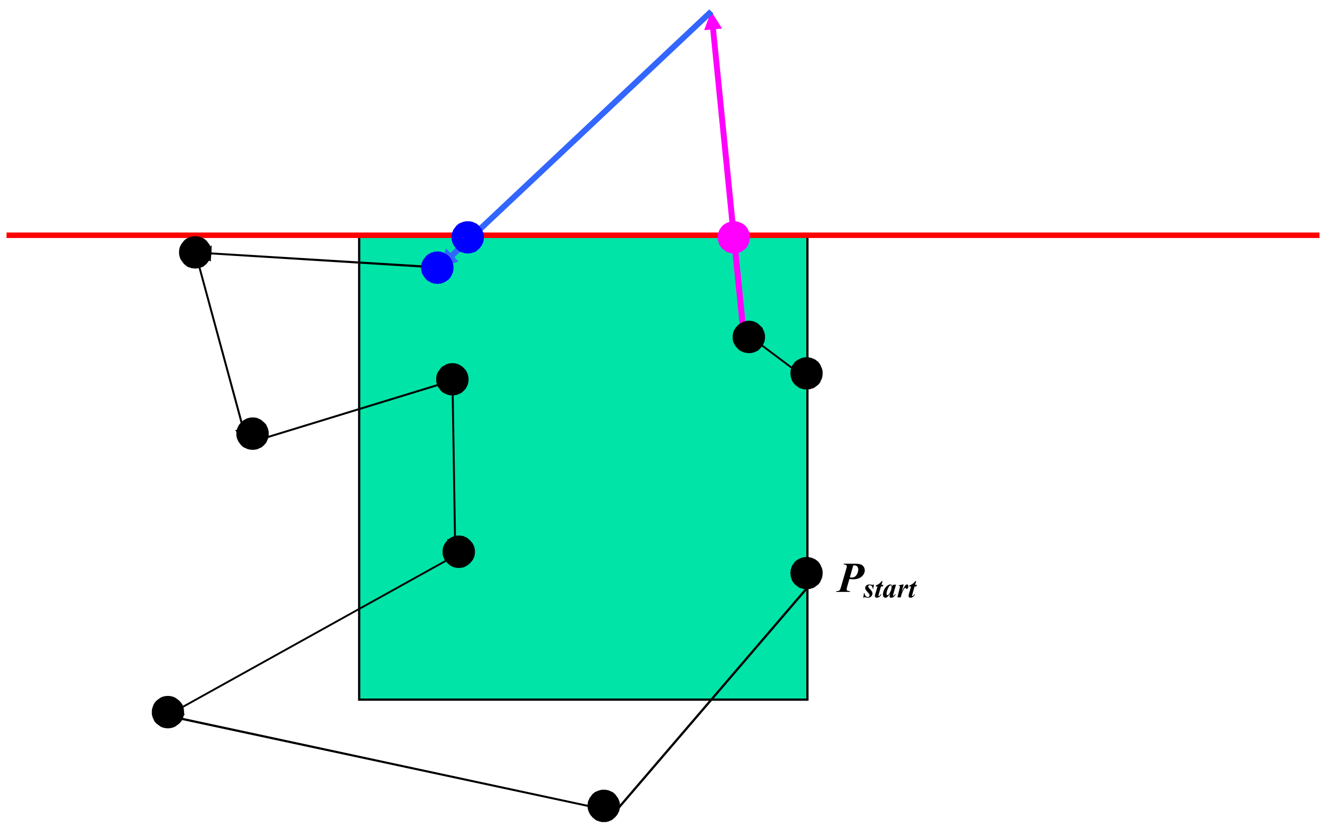
- 例子：



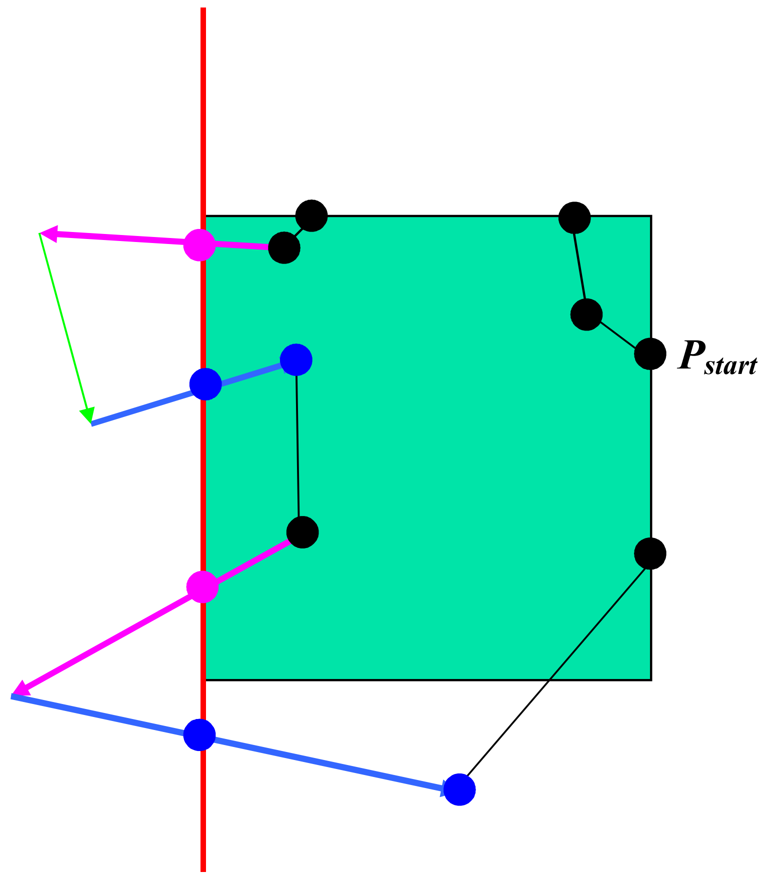
■ 示意图(5之1)



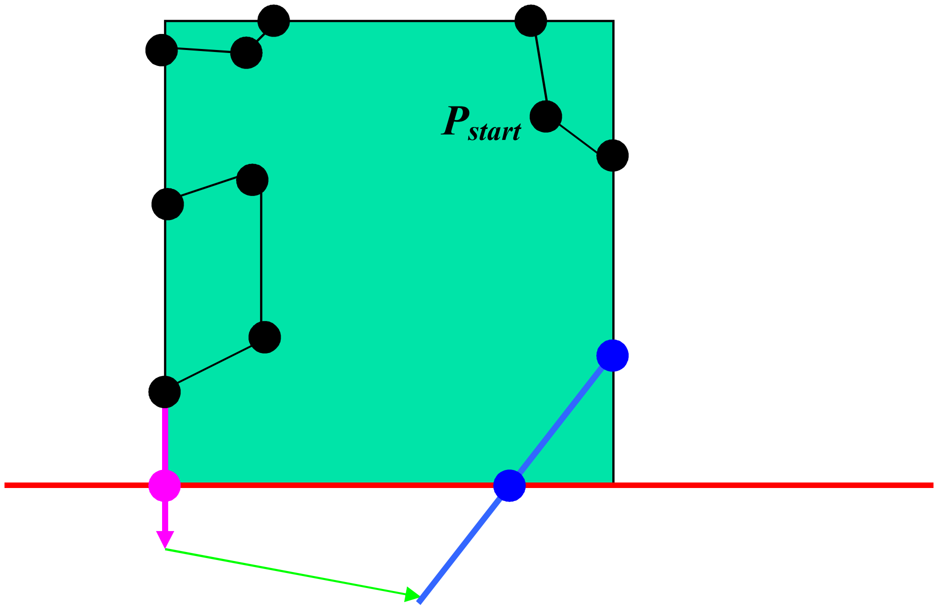
- 示意图(5之2)



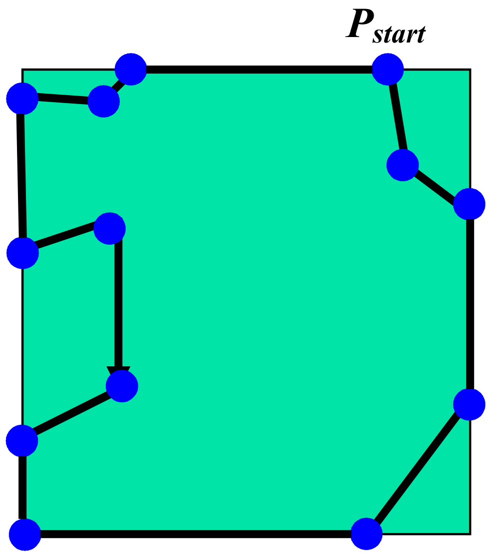
- 示意图(5之3)



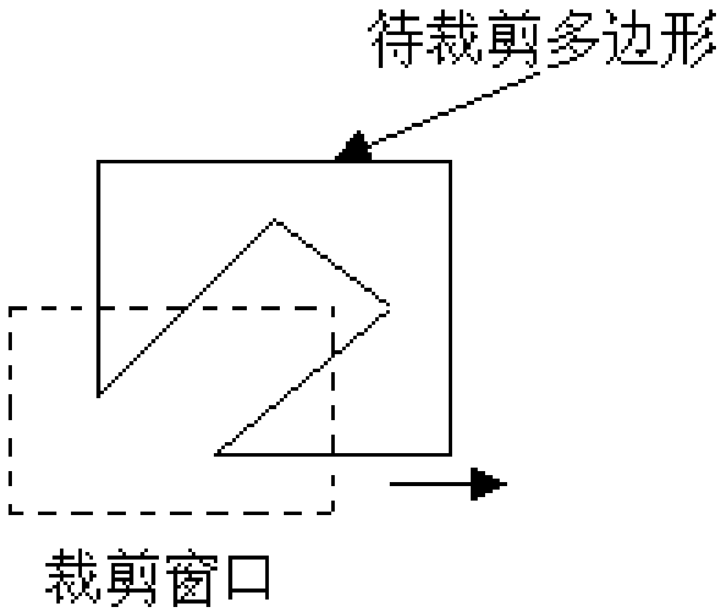
■ 示意图(5之4)



■ 示意图(5之5)

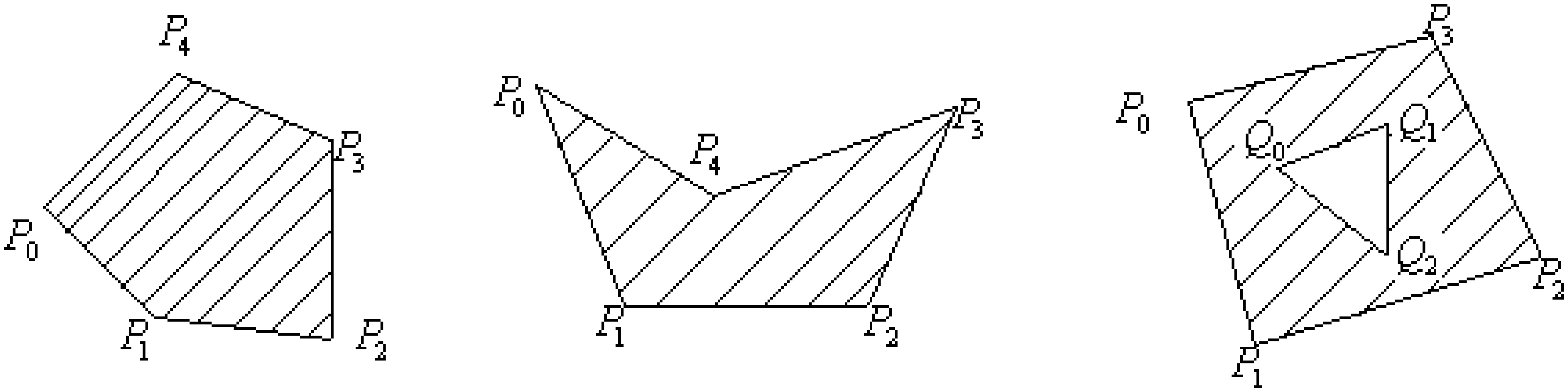


- **S-H算法的问题：**

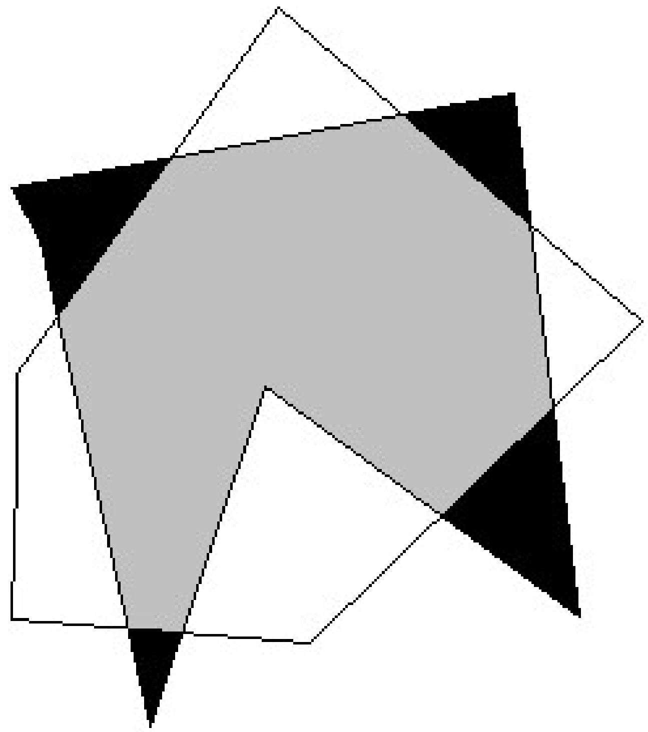


Weiler-Atherton算法

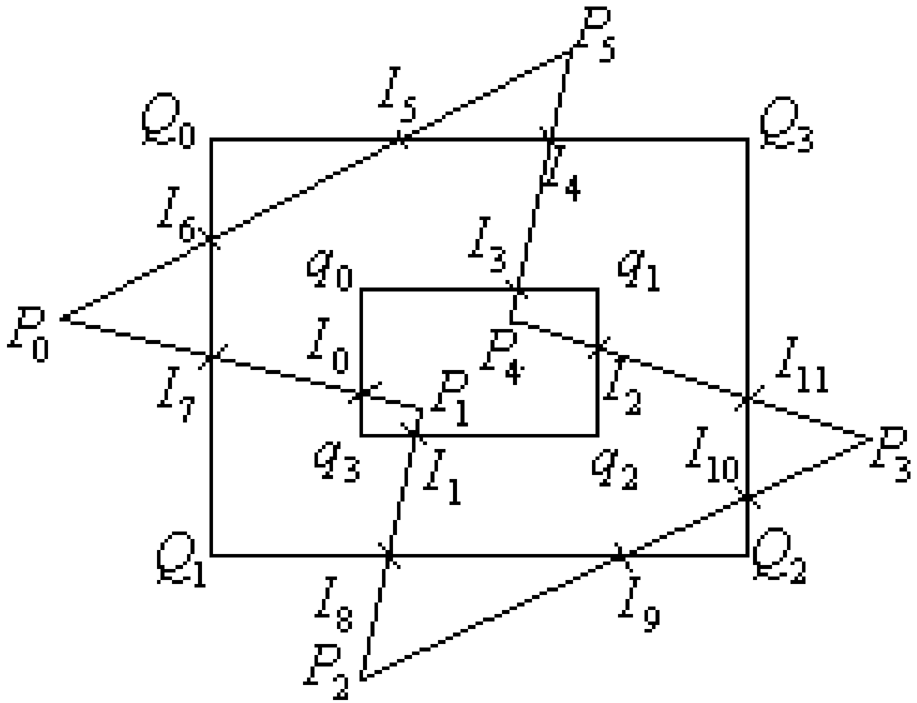
- 裁剪窗口，被裁剪多边形可以是任意多边形：凸、凹、带内环，地位对等。
- 被裁剪多边形为主多边形，裁剪窗口为裁剪多边形；约定顶点序列方向：外环(逆时针)；内环(顺时针)



- 裁剪结果区域由主多边形的部分边界和裁剪多边形的部分边界共同组成；
- 在交点处边界发生交替；
- 交点成对出现；



- 入点：主多边形边界由此进入裁剪多边形区域内部；
- 出点：离开裁剪多边形区域进入主多边形边界内部；



主多边形：$P_0P_1P_2P_3P_4P_5P_0$

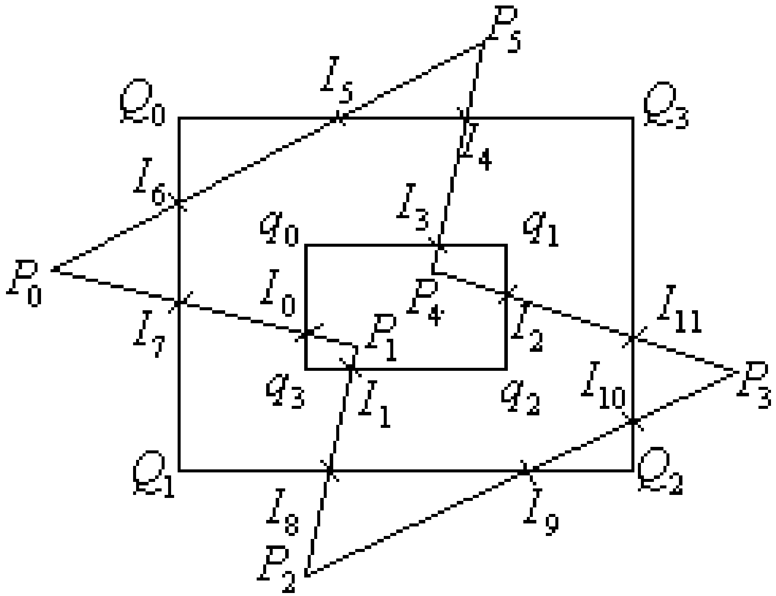
裁剪多边形：

$Q_0Q_1Q_2Q_3Q_0q_0q_1q_2q_3q_0$

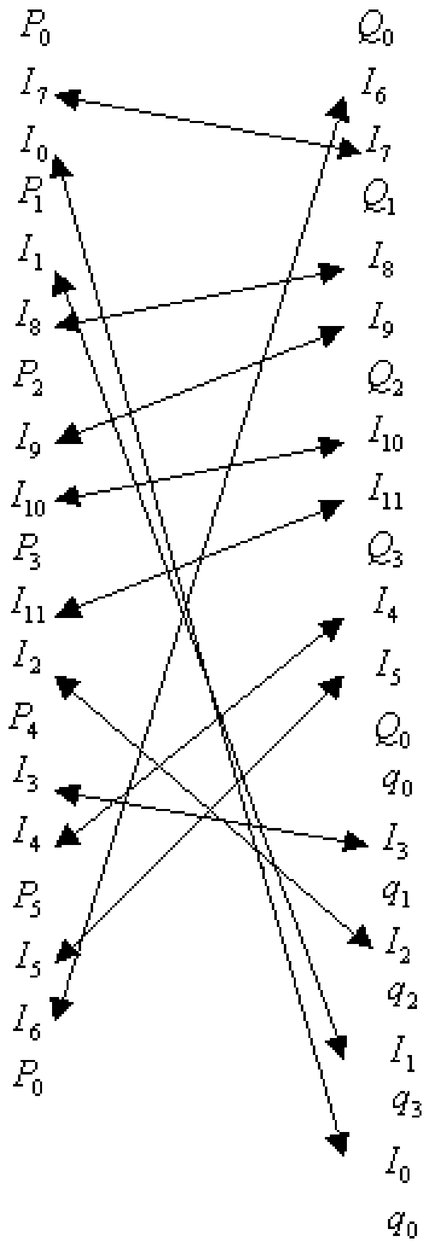
入点：$I_1,I_3,I_5,I_7,I_9,I_{11}$

出点：$I_0,I_2,I_4,I_6,I_8,I_{10}$

- 1建主多边形和裁剪多边形的顶点表；
- 2求交点、归类，并按顺序插入到顶点表中，在两个表的相应顶点间建双向指针；
- 3裁剪：
  - 3.1如果还有未跟踪过的交点，则任取一个作为起点，建空的裁剪结果多边形顶点表，把该交点入结果顶点表。否则算法结束；
  - 3.2如果该交点为入点，在主多边形顶点表内跟踪，否则在裁剪多边形顶点表内跟踪；
  - 3.3如果跟踪到的是多边形顶点，将其加入结果顶点表，继续跟踪，直到遇到新的交点，重复3.2~3.3，直到回到起点。

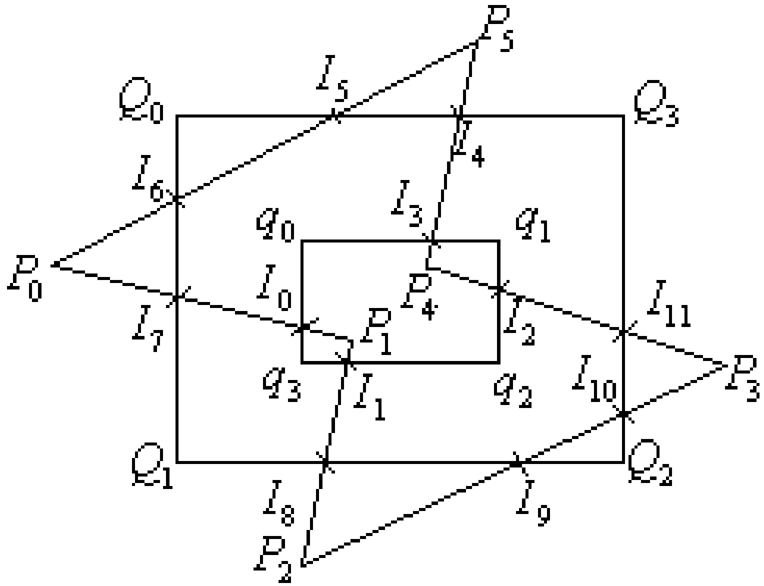


主多边形顶点表　裁剪多边形顶点表

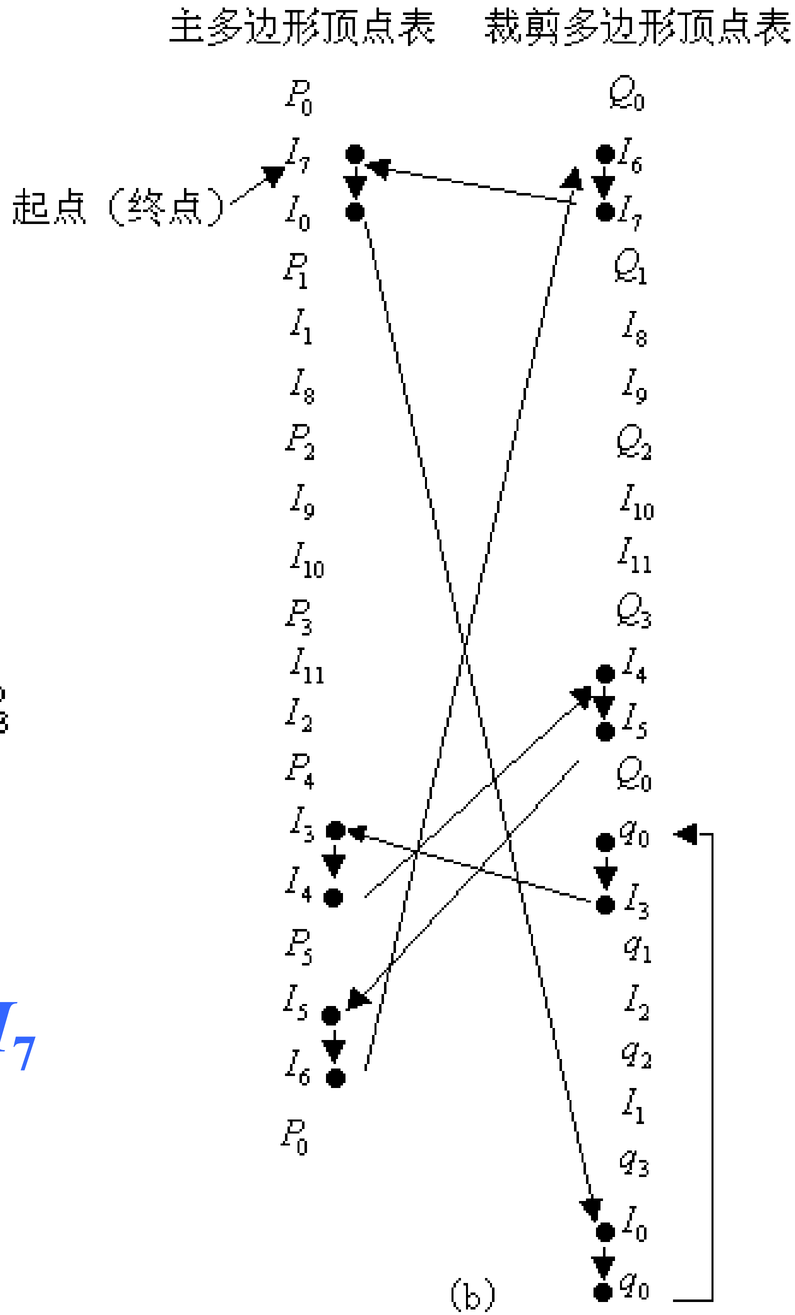


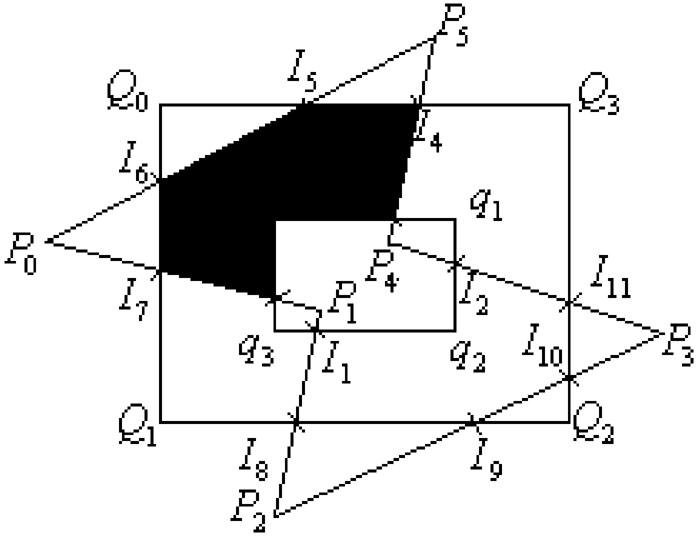
入点：$I_1,I_3,I_5,I_7,I_9,I_{11}$

出点：$I_0,I_2,I_4,I_6,I_8,I_{10}$



I_7	I_0	q_0	I_3	I_4	I_5	I_6	I_7
入	出	顶	入	出	入	出	
主	裁		主	裁	主	裁	



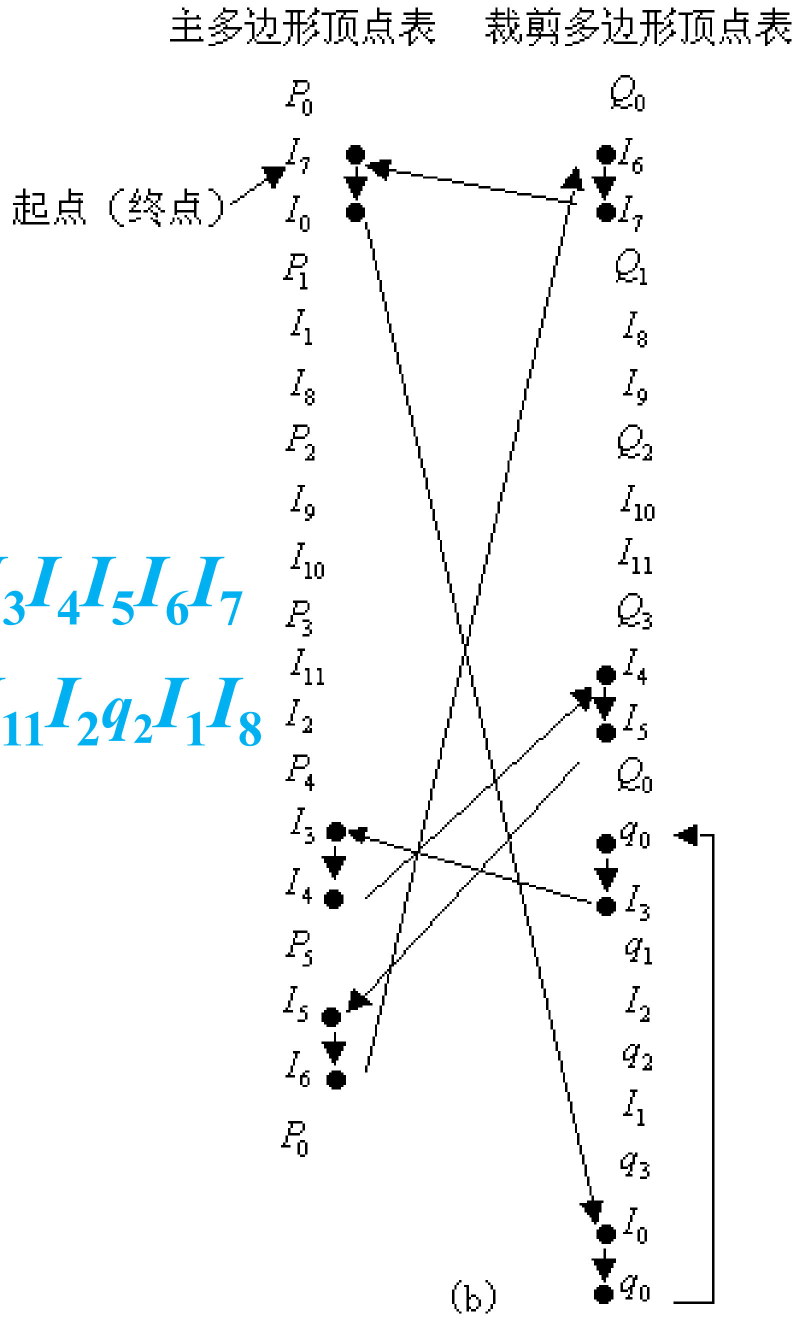


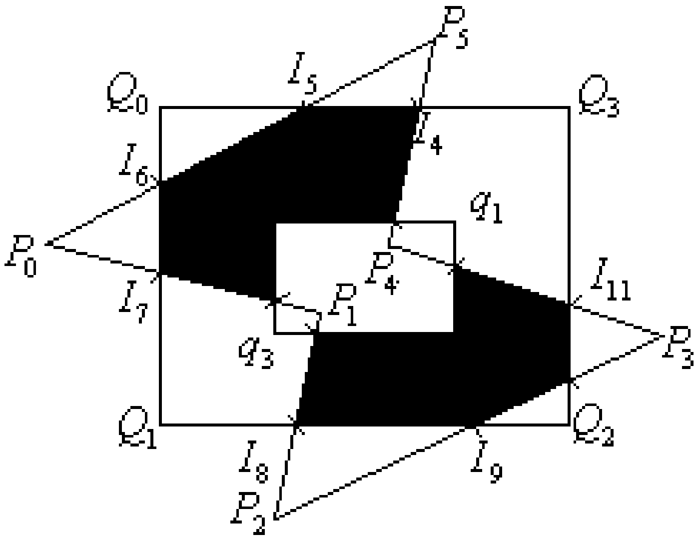
入点：$I_1, I_3, I_5, I_7, I_9, I_{11}$

出点：$I_0, I_2, I_4, I_6, I_8, I_{10}$

$I_7 I_0 q_0 I_3 I_4 I_5 I_6 I_7$

$I_8 I_9 I_{10} I_{11} I_2 q_2 I_1 I_8$





字符

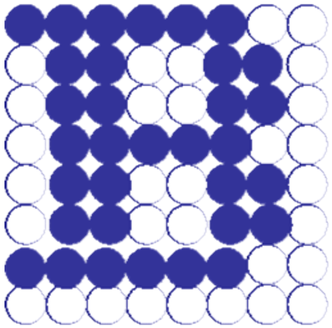
- 字符指数字、字母、汉字等符号。
- 计算机中字符由一个数字编码唯一标识。
- 国际上最流行的字符集：“美国信息交换标准代码集”，简称**ASCII**码。它是用**7**位二进制数进行编码表示**128**个字符；包括字母、标点、运算符以及一些特殊符号。

ASCII码表

L \ H	0000	0001	0010	0011	0100	0101	0110	0111
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	,	7	G	W	g	w
1000	BS	CAN	)	8	H	X	h	x
1001	HT	EM	(	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	’	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

- 汉字编码的国家标准字符集：**GB2312－80**。该字符集分为**94**个区，**94**个位，每个符号由一个区码和一个位码共同标识。区码和位码各用一个字节表示。
- 为了能够区分**ASCII**码与汉字编码，采用字节的最高位来标识：最高位为**0**表示**ASCII**码；最高位为**1**表示表示汉字编码。

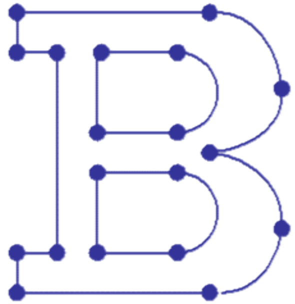
- 字库：为了在显示器等输出设备上输出字符，系统中必须装备有相应的字库。字库中存储了每个字符的形状信息，字库分为矢量型和点阵型两种。



点阵字符

1	1	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0

点阵字库中的位图表示



矢量轮廓字符

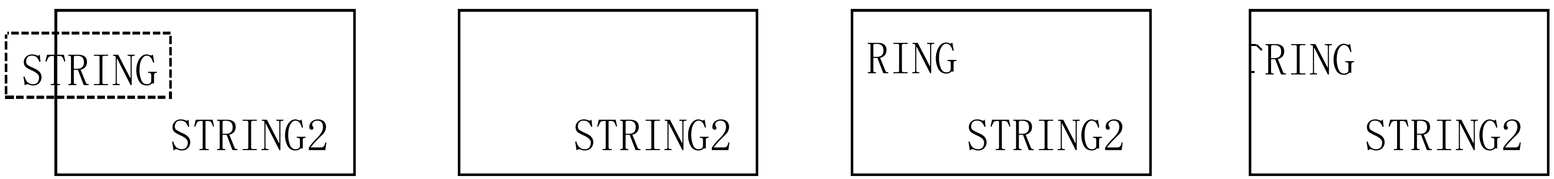
- 点阵字符：每个字符由一个位图表示，某位为**1**表示字符的笔划经过此位，对应于此位的象素应置为字符颜色。某位为**0**表示字符的笔划不经过此位，对应于此位的象素应置为背景颜色。
  - 采用压缩技术解决字库存储空间庞大的问题。
  - 点阵字符的显示分为两步：首先从字库中将它的位图检索出来，然后将检索到的位图写到帧缓冲器中。

- 矢量字符：记录字符的笔划信息，而不是整个位图，具有存储空间小，美观、变换方便等优点。
- 矢量字符的显示：首先从字库中取出它的字符信息，然后得到端点坐标，对其进行适当的几何变换，再扫描转换，显示出字符。

- 对于字符的旋转、缩放等变换
  - 点阵字符的变换需要对表示字符位图中的每一象素进行；
  - 矢量字符的变换只需要对其笔划端点进行变换。
- 特点：
  - 点阵字符：存储量大，易于显示；
  - 矢量字符：存储量小，美观，变换方便，需要光栅化后才能显示。

字符裁剪

- 串精度：当整个字符串完全落在窗口之内才显示；
- 字符精度：当一个字符完全落在窗口之内才显示；
- 笔划\象素精度：按象素或将笔划分解成直线段对窗口作裁剪；



待裁剪字符串　串精度裁剪　字符精度裁剪　象素精度裁剪

《图形与动画》实验要求

- 不准抄袭！
- 准时提交源程序和实验报告(2~3页即可)。
- 实验报告包括：实验目的、使用的算法(名称或简述)、实验结果以及问题分析、交互方式、编译环境。不要粘贴源代码！
- 鼓励创新，可以酌情加分。

《图形与动画》实验(1)

- 题目：多边形的几何操作
- 内容：
  - 实现任意多边形（凹、凸、带内环）的输入及显示，可自定义多边形（边界、内部）的颜色；
  - 实现任意多边形的平移、旋转、缩放；
  - 实现两个任意多边形的裁剪，并支持多步裁剪；
  - 只能使用**SetPixel**、**MoveTo**、**LineTo**函数。
- 时间：**3**周**(10.9**日**23:59**以前提交**)**