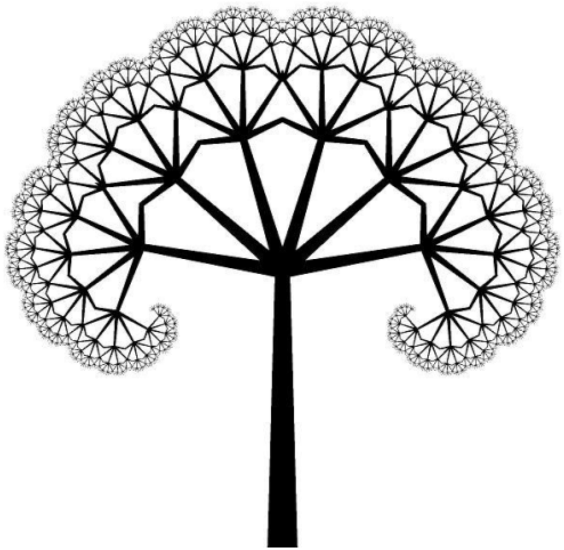
第6讲：二维变换

吴文明
计算机与信息学院

导论

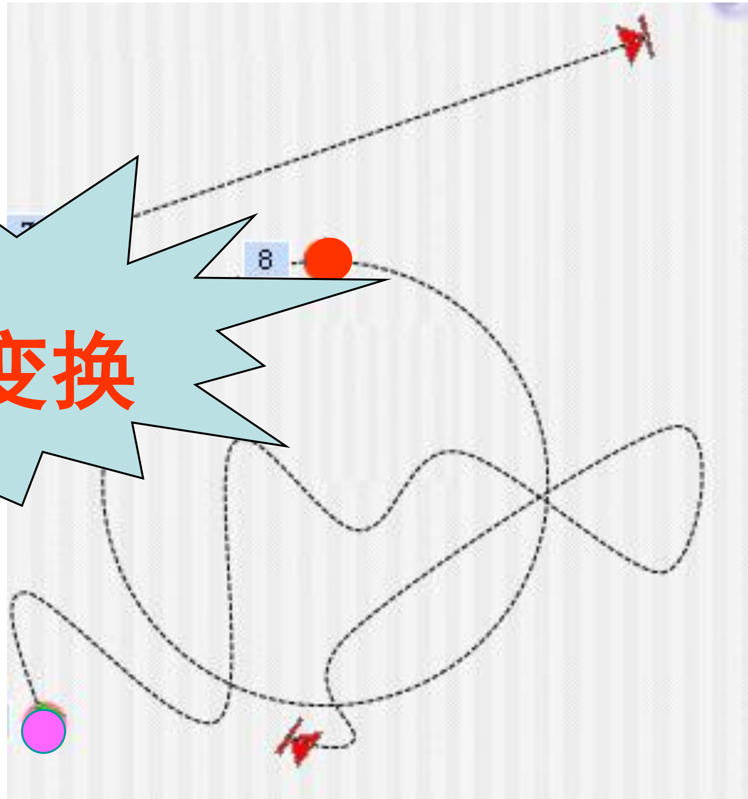


导论

计算机图形学的应用

- 计算机辅助设计与制造
- 计算机辅助绘
- 计算机辅助教
- 计算机动画
- ……

图形变换



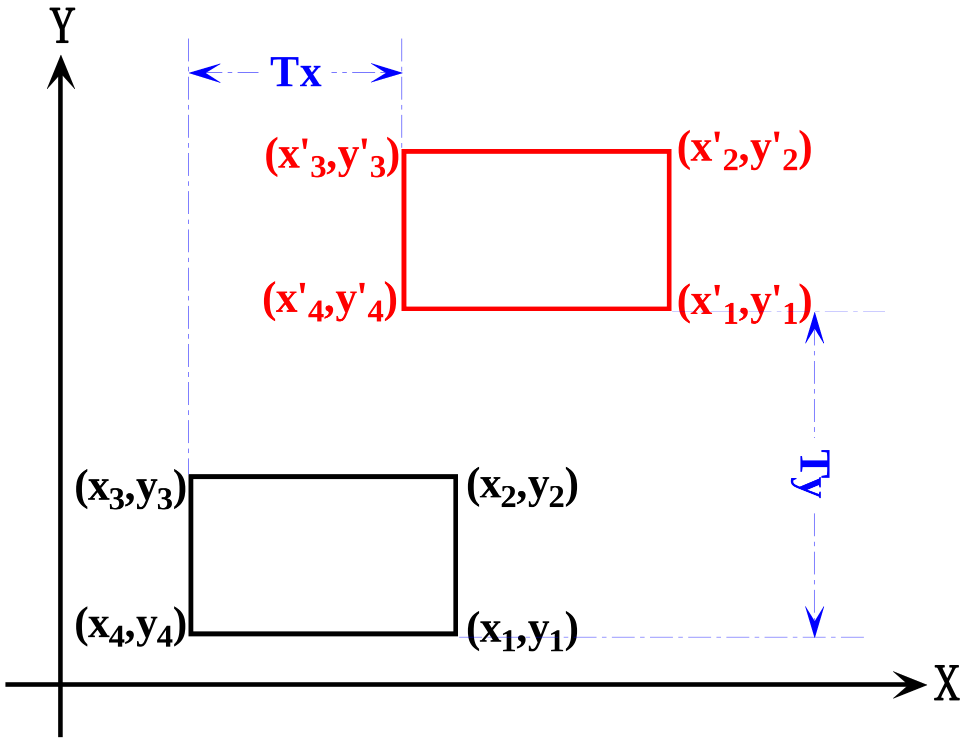
讲义大纲

- 二维变换
- 二维观察
- 裁减

二维变换

节约计算量

$$F\left(x,y\right)?$$



继续推导

$$\begin{cases} x_i' = x_i + Tx \\ y_i' = y_i + Ty \end{cases}$$

$$\begin{cases} x_i' = 1 \cdot x_i + 0 \cdot y_i + Tx \\ y_i' = 0 \cdot x_i + 1 \cdot y_i + Ty \end{cases}$$

$$[x', y'] = [x, y] \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + [Tx \quad Ty]$$

便于计算机处理

$$[x', y'] = [x, y] \cdot A$$

可能吗

齐次坐标

齐次坐标

齐次坐标

用n+1维向量表示n维向量；

- [x， y]的齐次坐标为 [hx， hy， h]

无穷远点：

- [1， 0， 0]
- [0， 1， 0]

齐次坐标不唯一

问题：

- [3，5，1]
- [6，10，2]
- [1.5，2.5，0.5]

（3，5）

规范化齐次坐标：

- 就是h=1的齐次坐标；
- [hx, hy, h]→ [hx/h, hy/h, 1]

齐次坐标

如何求矩阵

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \cdot \begin{bmatrix} a & b & p \\ c & d & q \\ l & m & s \end{bmatrix}$$

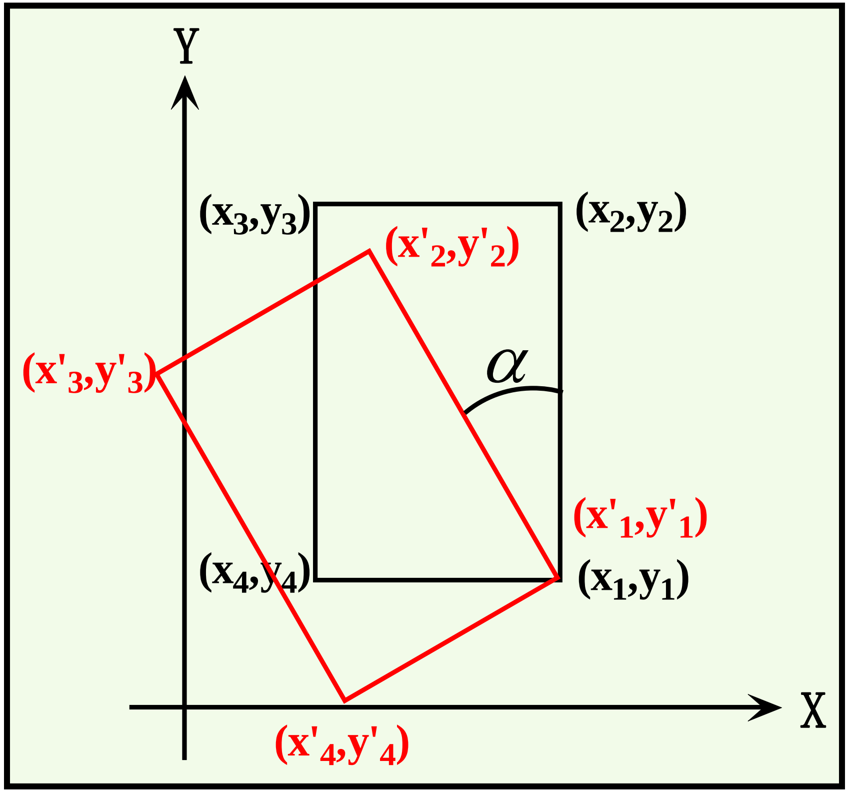
$$[x' \quad y'] = [x \quad y] \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + [Tx \quad Ty]$$

二维几何变换

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix}$$

$$\begin{cases} x' = 1 \cdot x + 0 \cdot y + 1 \cdot Tx \\ y' = 0 \cdot x + 1 \cdot y + 1 \cdot Ty \\ 1 = 0 \cdot x + 0 \cdot y + 1 \cdot 1 \end{cases}$$

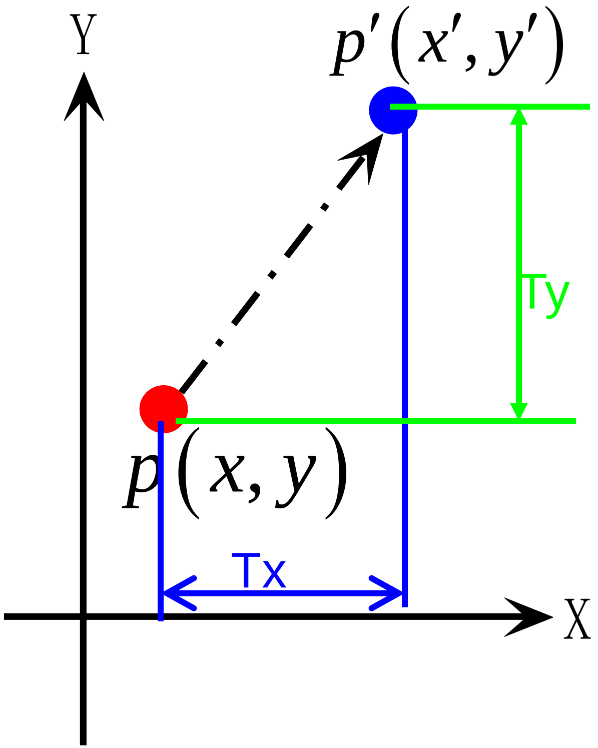
基本几何变换



复杂几何变换可分解为一系列简单几何变换。

- 平移变换
- 旋转变换
- 比例变换
- 对称变换
- 错切变换

平移变换

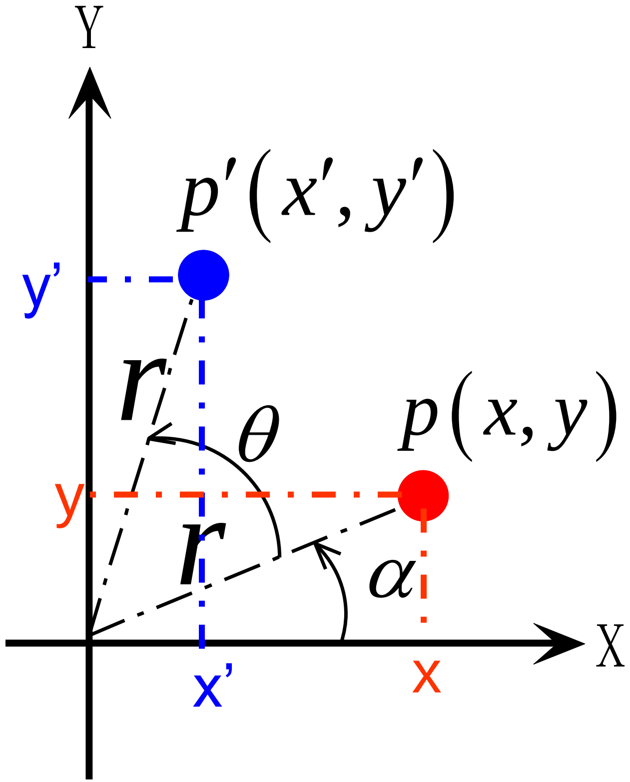


$$\begin{cases} x' = 1\cdot x + 0\cdot y + Tx \\ y' = 0\cdot x + 1\cdot y + Ty \end{cases}$$

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix}$$

引言中的例题

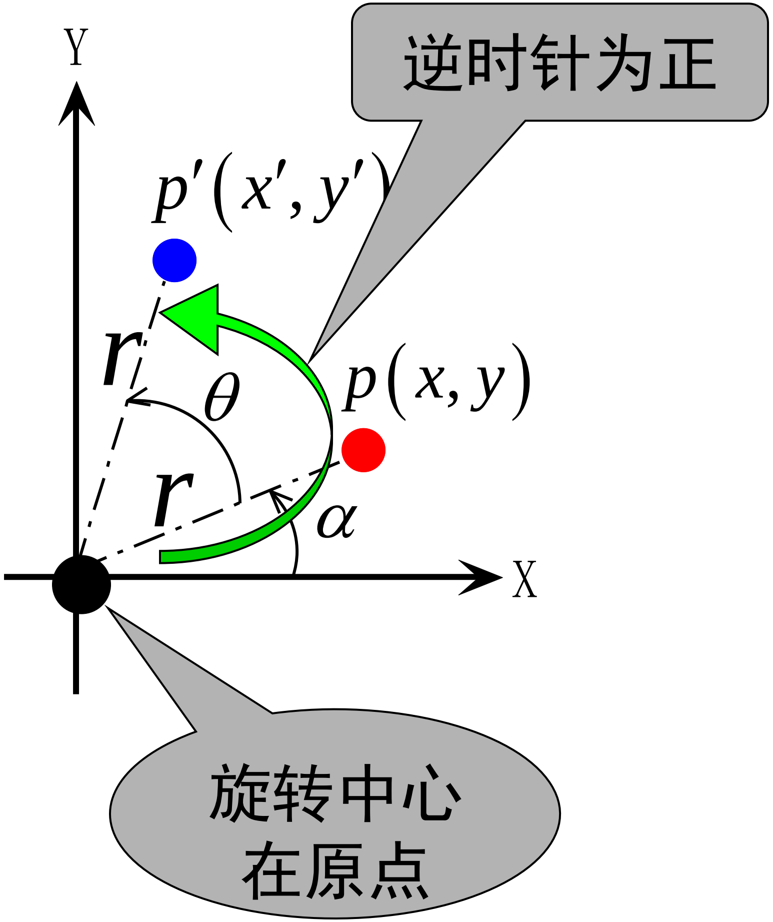
旋转变换



$$\begin{cases} x' = r \cdot \cos(\alpha + \theta) \\ y' = r \cdot \sin(\alpha + \theta) \end{cases}$$

$$x' = r \cdot \cos(\alpha + \theta)$$

旋转变换

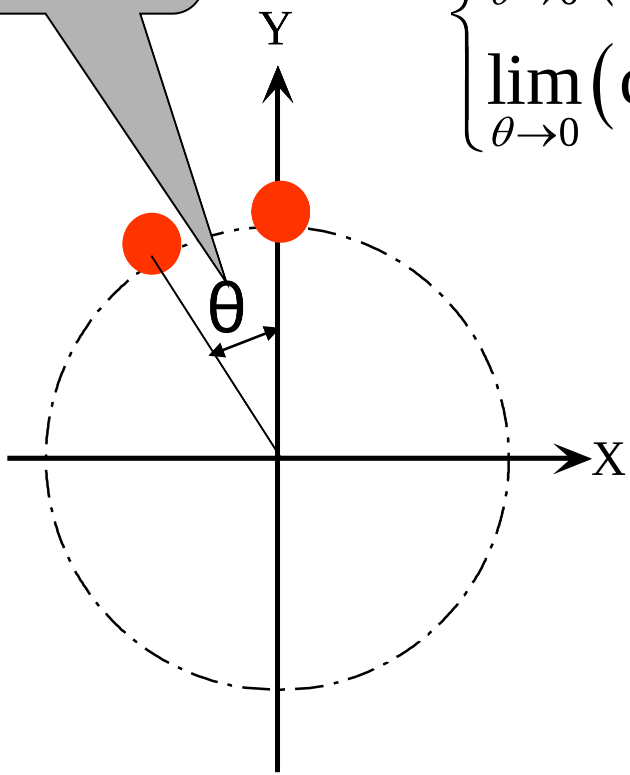


$$\begin{cases} x' = x\cdot\cos\theta - y\cdot\sin\theta \\ y' = x\cdot\sin\theta + y\cdot\cos\theta \end{cases}$$

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

一点小技巧

每次转动小的角度θ



$$\begin{cases} \lim\limits_{\theta\to 0}\left(\sin\theta / \theta\right) = 1 \\ \lim\limits_{\theta\to 0}\left(\cos\theta\right) = 1 \end{cases}$$

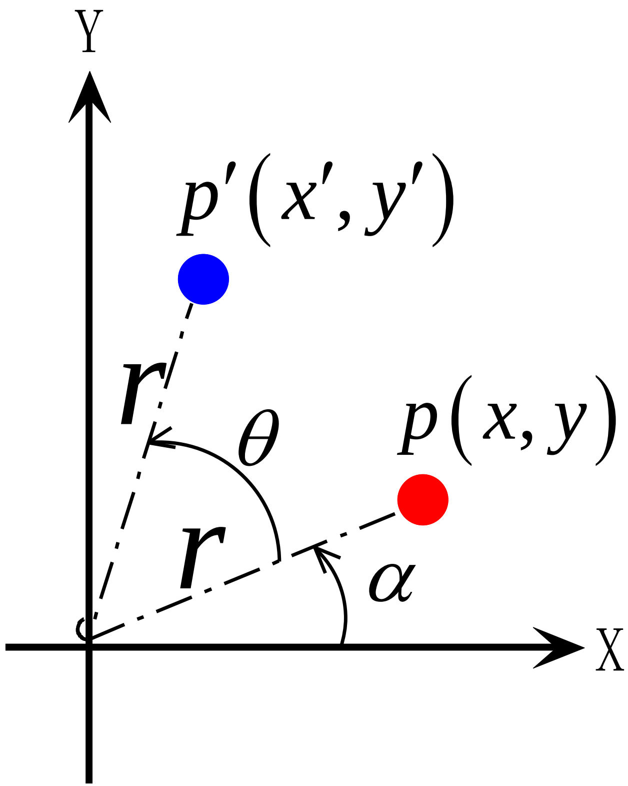
$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & \theta & 0 \\ -\theta & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

减少计算量

旋转变换



要点：

- 绕原点进行旋转
- 逆时针方向为正

提示：

- 理解变换矩阵的推导过程比记住结果更重要

思考题：

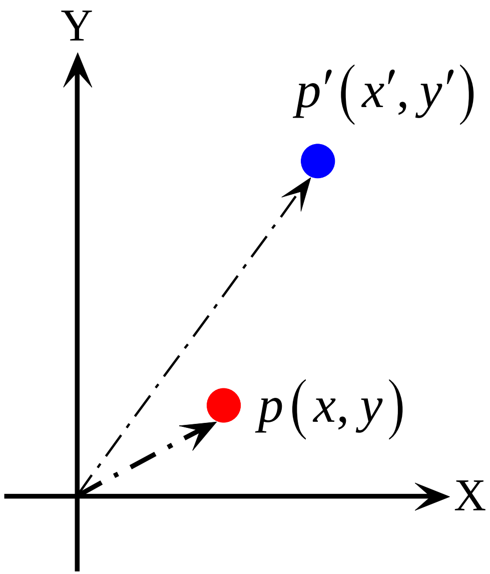
- 如果取顺时针为正，试推导变换矩阵

旋转变换

绕原点顺时针旋转θ角的齐次坐标计算形式可写为：

$$[x^* \quad y^* \quad 1]=[x \quad y \quad 1]\bullet\begin{bmatrix}\cos(-\theta) & \sin(-\theta) & 0\\ -\sin(-\theta) & \cos(-\theta) & 0\\ 0 & 0 & 1\end{bmatrix}$$

比例变换

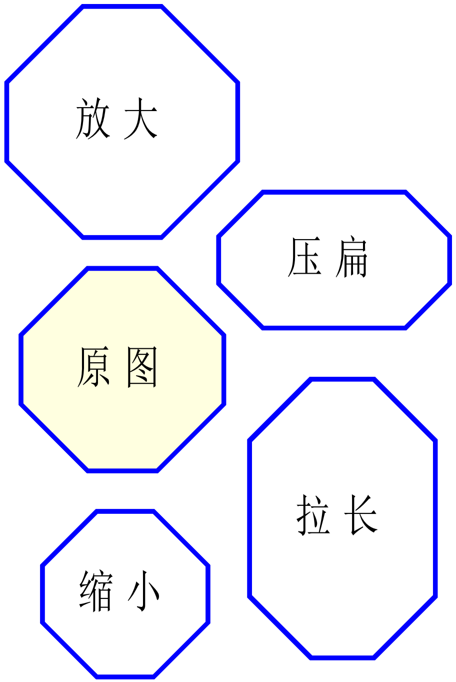


比例变换是指对P点相对于坐标原点沿x方向放缩Sx倍，沿y方向放缩Sy倍，其中Sx和Sy称为比例系数。

$$
\begin{aligned}
\begin{bmatrix} x' & y' & 1 \end{bmatrix} &= \begin{bmatrix} S_x \cdot x & S_y \cdot y & 1 \end{bmatrix} \\
&= \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

比例变换

讨论:S_x、S_y….



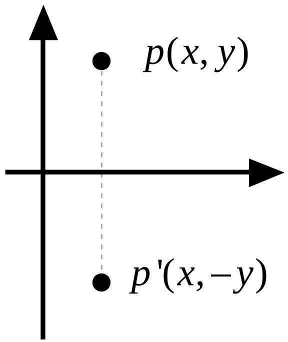
$S_x>0$ && $S_y>0$

- $S_x=S_y>1$：放大
- $S_x=S_y<1$：缩小
- $S_x>S_y$ ：压扁
- $S_x<S_y$ ：拉长

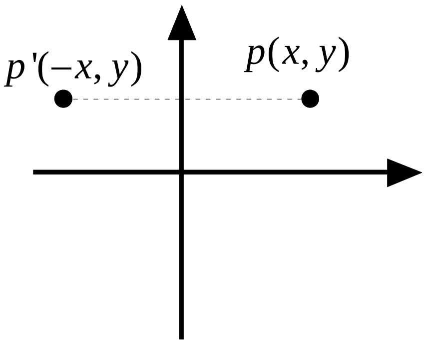
!($S_x>0$ && $S_y>0$)

- 思考题

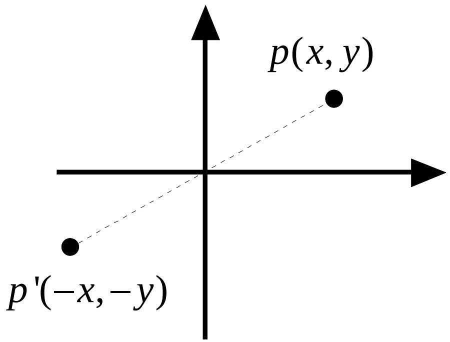
对称变换



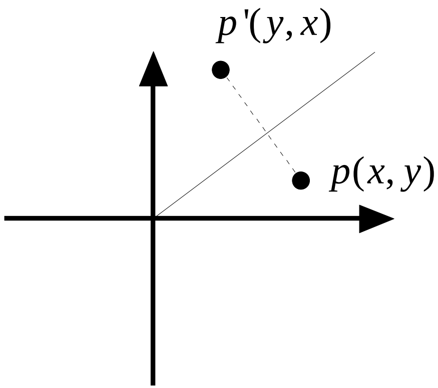
关于x轴对称



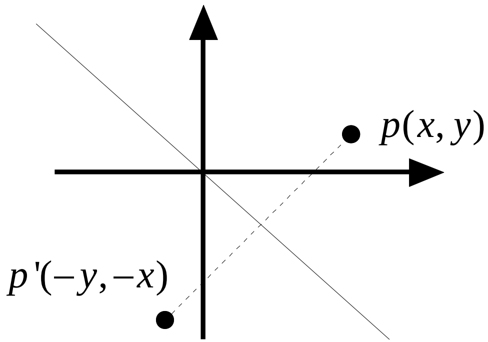
关于y轴对称



关于原点对称

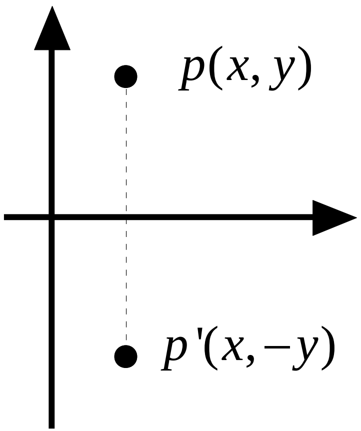


关于y=x对称



关于y=x对称

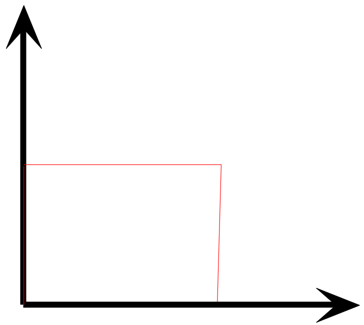
对称变换



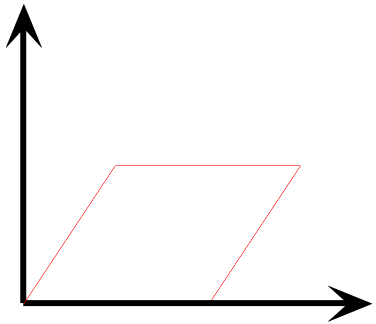
关于x轴对称

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

错切变换

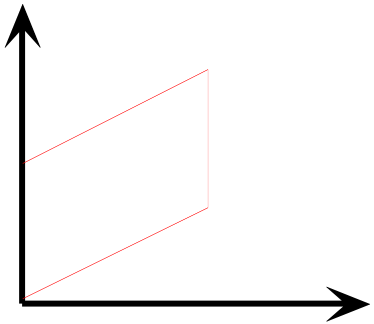


原图



沿x轴错切

$$\begin{cases} x' = x + cy \\ y' = y \end{cases}$$



沿y轴错切

$$\begin{cases} x' = x \\ y' = bx + y \end{cases}$$

错切变换

$$\begin{cases} x' = x + cy \\ y' = bx + y \end{cases}$$

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & b & 0 \\ c & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

二维变换矩阵

$$\left[\begin{array}{cc:c} a & b & p \\ c & d & q \\ \hdashline l & m & s \end{array}\right]$$

二维变换矩阵

比例、旋转、对称、错切

投影

$$
\left[\begin{array}{cc:c}
a & b & p \\
c & d & q \\
\hdashline
l & m & s
\end{array}\right]
$$

平移

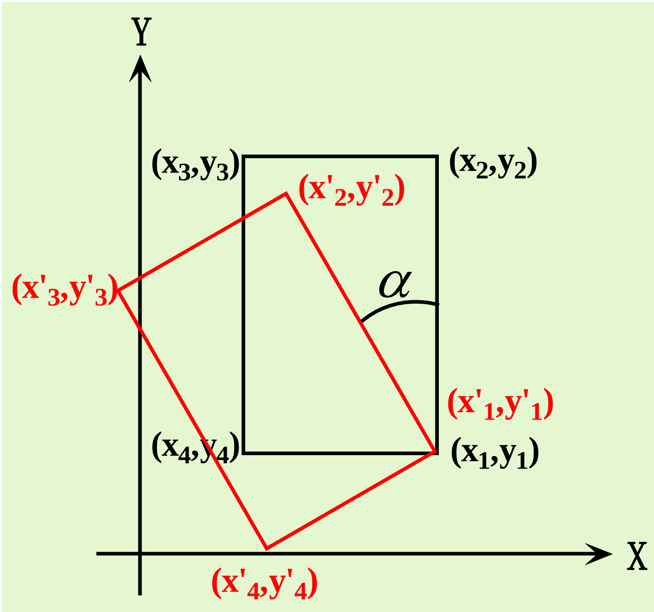
整体比例变换

复合变换

平移变换
- 沿X轴移动Tx;
- 沿Y轴移动Ty;

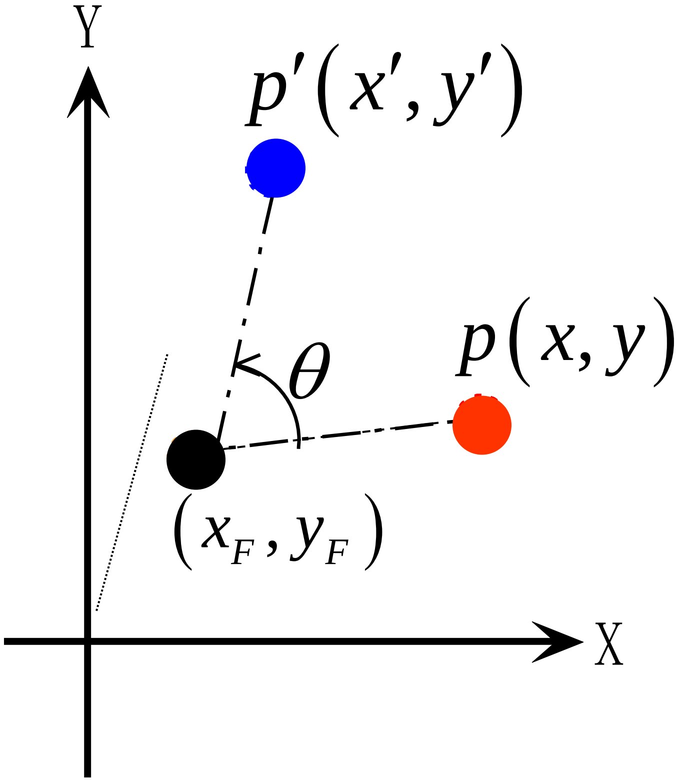
旋转变换
- 绕原点旋转θ

比例变换
- 相对于原点
- 沿X轴放缩S_x
- 沿Y轴放缩S_y



条件不满足
怎么办

相对任一参考点的旋转变换



STEP 1: 进行平移变换

$Tx=-x_F, T_Y=-y_F;$

$(x_1,y_1,1)=(x,y,1)T1$

STEP 2: 绕原点旋转

$(x_2,y_2,1)=(x_1,y_1,1)T2$

STEP 3: 进行平移变换

$Tx=x_F, T_Y=y_F;$

$(x_3,y_3,1)=(x_2,y_2,1)T3$

没有条件，创造条件！

相对任一参考点的旋转变换

STEP 1: 进行平移变换

$T_x = -x_F, T_Y = -y_F$;

$(x_1, y_1, 1) = (x, y, 1)T1$

STEP 2: 绕原点旋转

$(x_2, y_2, 1) = (x_1, y_1, 1)T2$

STEP 3: 进行平移变换

$T_x = x_F, T_Y = y_F$;

$(x_3, y_3, 1) = (x_2, y_2, 1)T3$

$(x', y', 1) = (x_3, y_3, 1)$

$$
\begin{aligned}
\begin{bmatrix} x' & y' & 1 \end{bmatrix} &= \begin{bmatrix} x_3 & y_3 & 1 \end{bmatrix} \\
&= \begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix} \cdot T_3 \\
&= \begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} \cdot T_2 \cdot T_3 \\
&= \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot T_1 \cdot T_2 \cdot T_3
\end{aligned}
$$

$$T = T_1 \cdot T_2 \cdot T_3$$

复合变换

复合变换矩阵

$$T_1=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ -x_F & -y_F & 1\end{bmatrix}\quad T_2=\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\quad T_3=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ x_F & y_F & 1\end{bmatrix}$$

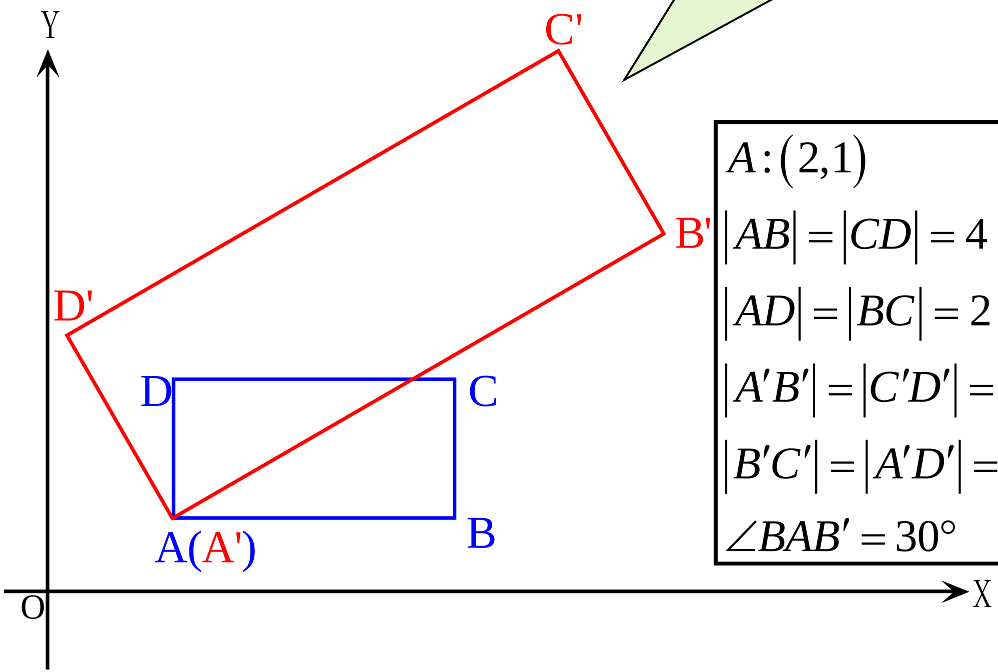
$$T=\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ x_F-x_F\cdot\cos\theta+y_F\cdot\sin\theta & y_F-y_F\cdot\cos\theta-y_F\cdot\sin\theta & 1\end{bmatrix}$$

结果不重要

重要的是方法

思考题

求A’，B’，C’，D’的坐标



$A:(2,1)$

$|AB| = |CD| = 4$

$|AD| = |BC| = 2$

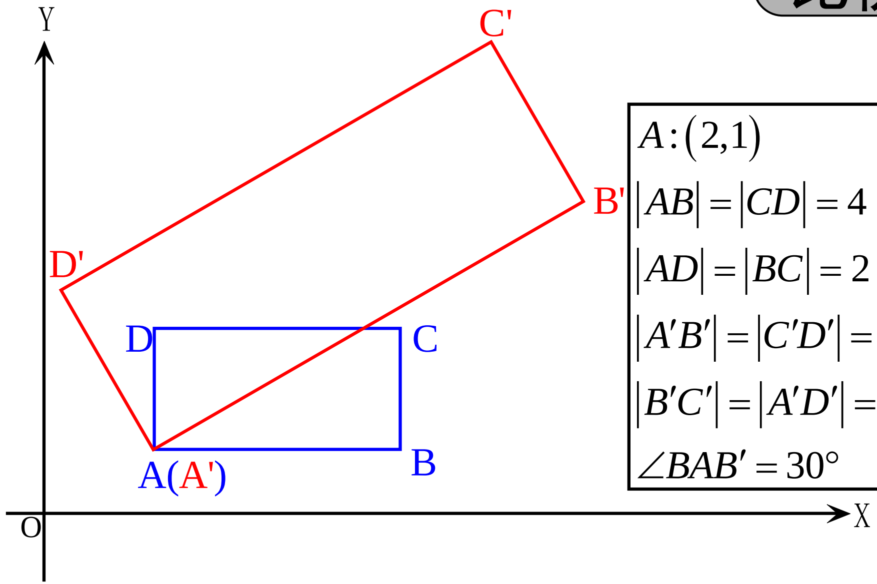
$|A'B'| = |C'D'| = 8$

$|B'C'| = |A'D'| = 3$

$\angle BAB' = 30°$

求变换矩阵

思考题



$A:(2,1)$

$|AB|=|CD|=4$

$|AD|=|BC|=2$

$|A'B'|=|C'D'|=8$

$|B'C'|=|A'D'|=3$

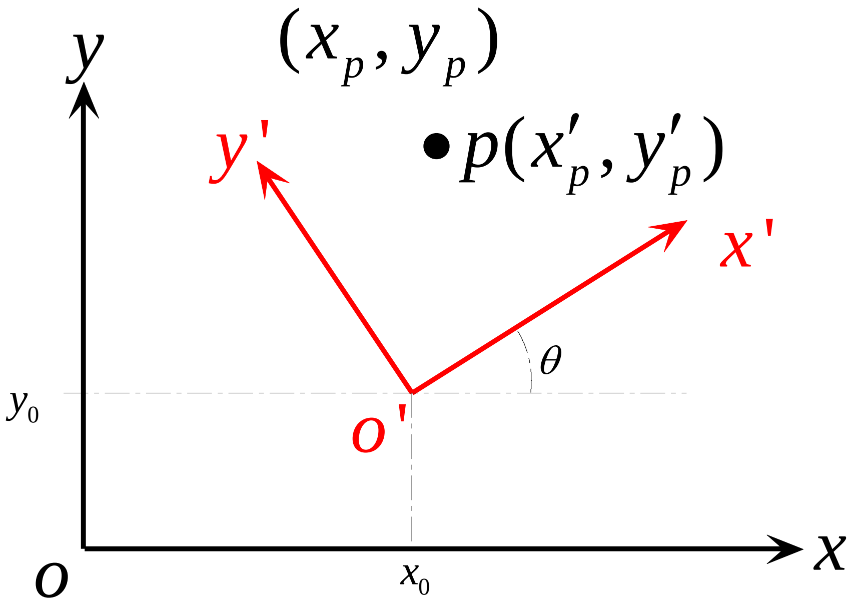
$\angle BAB'=30°$

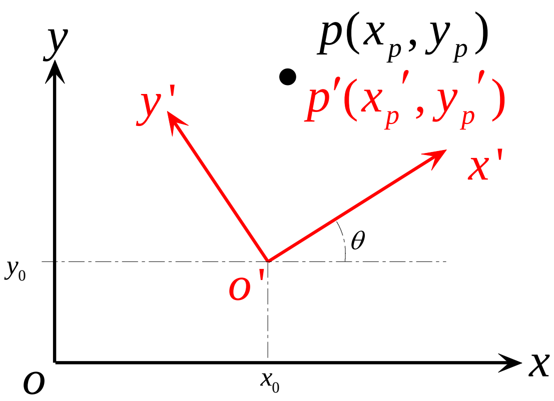
旋转？
比例？

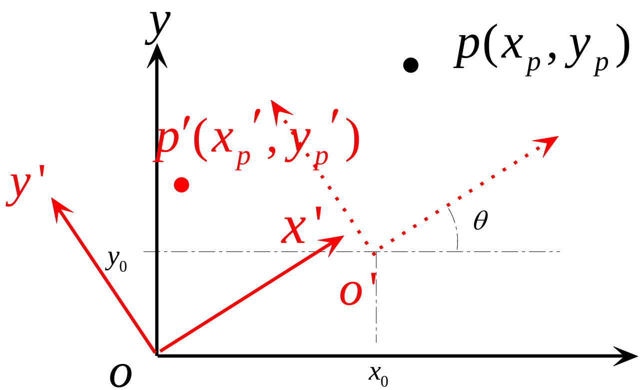
- 进行平移变换，将A点移至原点
- 进行比例变换
- 进行旋转变换
- 再次进行平移，将A点移原始位置

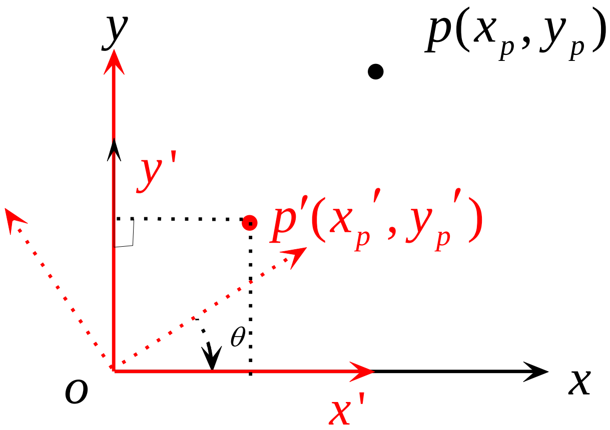
问题分析：
- 相对A点进行了一次比例变换
- 相对A点进行了一次旋转变换

坐标系之间的变换







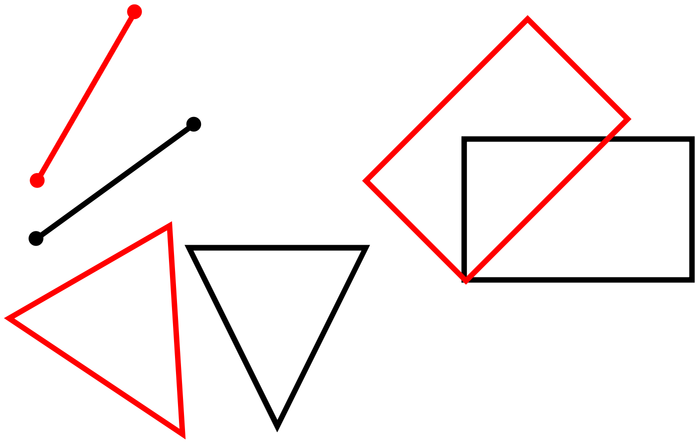


$$T = T_t \cdot T_R$$

$$P' = [x_p', y_p', 1] = [x_p, y_p, 1] \cdot T$$

$$= P \cdot T$$

二维几何图形变换



二维几何图形变换

原始坐标序列

$$P' = P \cdot T$$

新坐标序列

$$\begin{bmatrix} x_1' & y_1' & 1 \\ x_2' & y_2' & 1 \\ x_3' & y_3' & 1 \\ \dots & \dots & \dots \\ x_n' & y_n' & 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ \dots & \dots & \dots \\ x_n & y_n & 1 \end{bmatrix} \cdot T$$

变换的性质

- 仿射变换（Affine Transform）：
  - x’= ax + by +m
  - y’= cx + dy + n

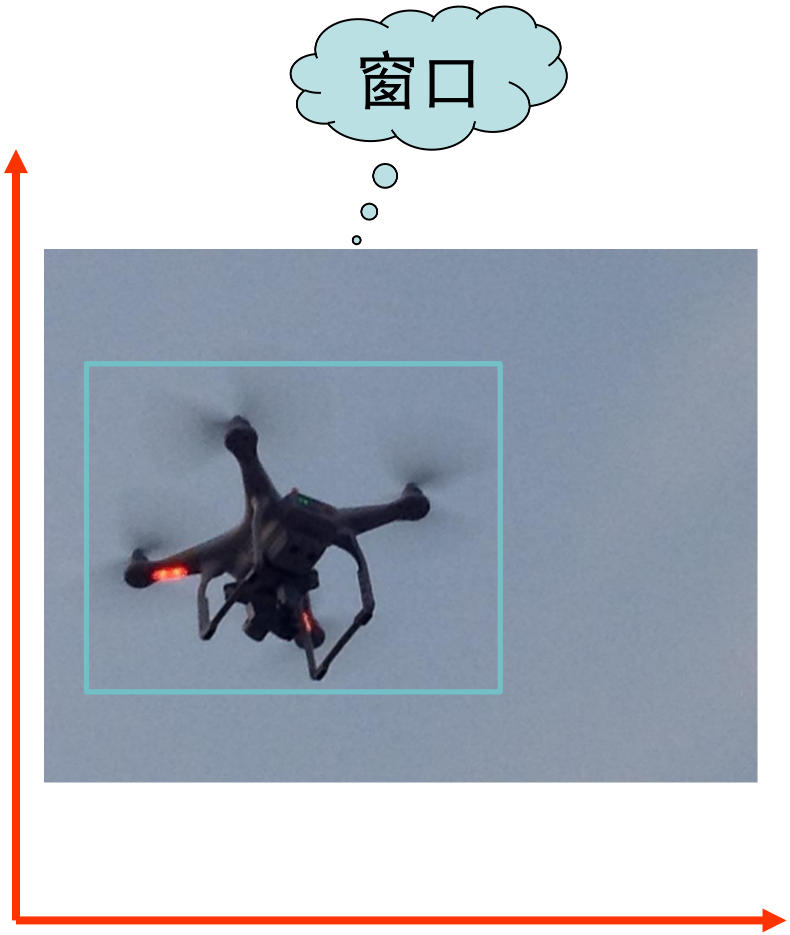
- 可通过平移、缩放、旋转、错切复合实现

变换的性质

- 直线的中点不变性；
- 平行直线不变性；
- 相交不变性；
- 仅包含旋转、平移和反射的仿射变换维持角度和长度的不变性；
- 比例变化可改变图形的大小和形状；
- 错切变化引起图形角度关系的改变，甚至导致图形发生畸变。

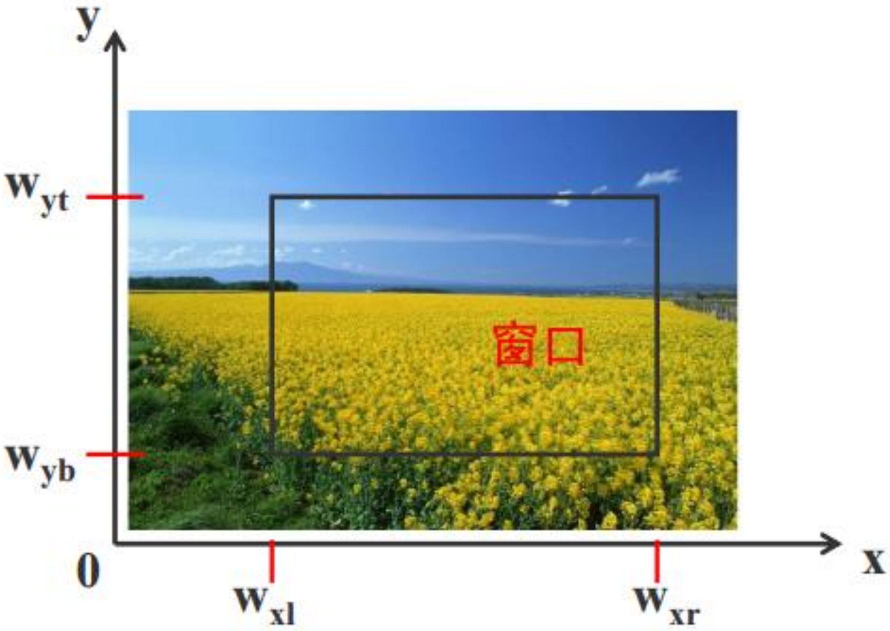
二维观察

二维观察

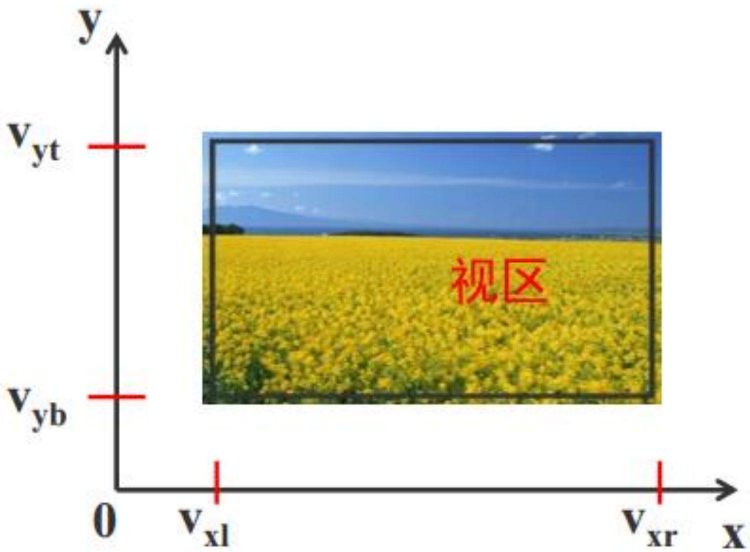

窗口


视区
计算机
UltraEdit
SSH Secure Shell Client
Lenovo Soluti...
模型
PreInterfa...
Structure...
回收站
WinEdt
Microsoft Office 2010
Shortcuts...
Edit Pictures...
Maxthon... - 快捷方式
KMPlayer
Skype
Maple 16 (32-bit)
Mozilla Firefox
Maple 16 (32-bit) ...
腾讯QQ
Reading Material
个人数字图书馆2.1
MeshLab
DJI_2.JPG
Zoom:

二维观察

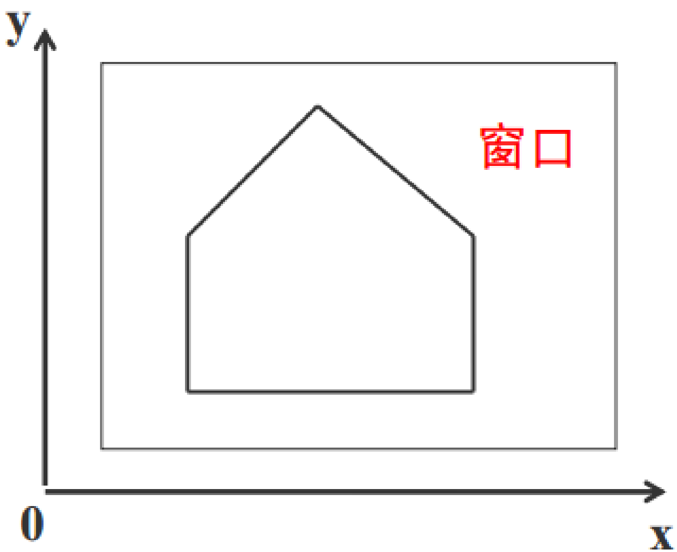


世界坐标系

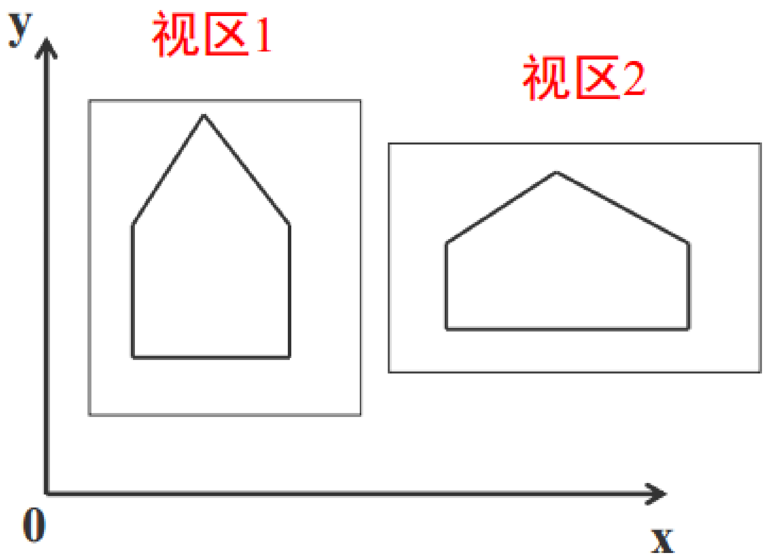


设备坐标系

二维观察



世界坐标系



屏幕坐标系

“窗口—视区”变换

窗口：在用户坐标系中需要进行观察和处理的一个坐标区域；定义显示什么

视区：显示设备上用于显示窗口的坐标区域；定义怎么显示

窗口(用户坐标系)

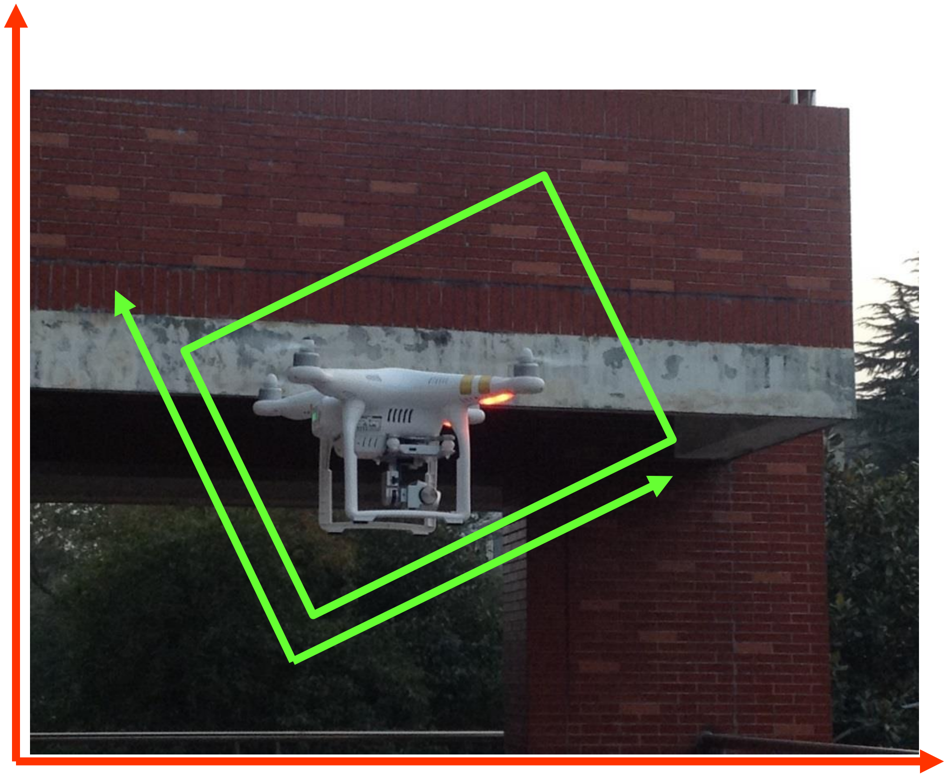
$$(wxl \quad wyb),(wxr \quad wyt)$$

视区（设备坐标系）

$$(vxl \quad vyb),(vxr \quad vyt)$$

$$\begin{bmatrix} (vxr-vxl)/(wxr-wxl) & 0 & 0 \\ 0 & (vyt-vyb)/(wyt-wyb) & 0 \\ (vxl\cdot wxr-wxl\cdot vxr)/(wxr-wxl) & (vyb\cdot wyt-wxb\cdot vyt)/(wyt-wyb) & 1 \end{bmatrix}$$

观察坐标系



用户坐标系

↓

观察坐标系

↓

设备坐标系

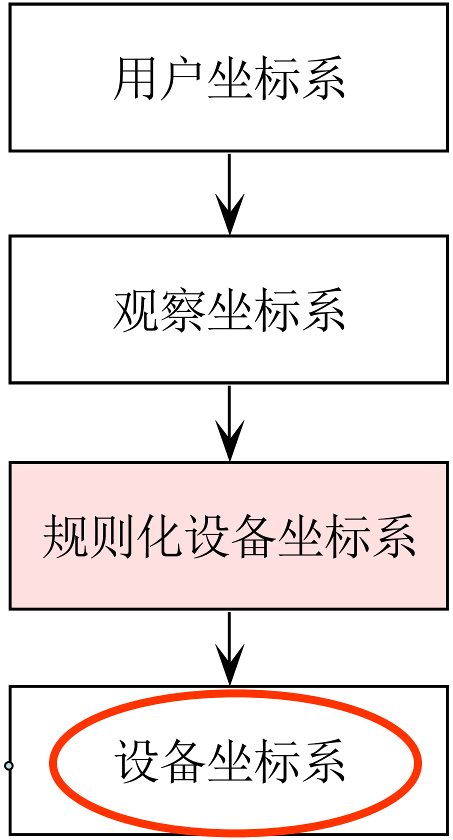
规则化设备坐标系

问题：已知窗口(0,0),(100,100),现在需要进行全屏显示，请给出“窗口—视区”变换矩阵

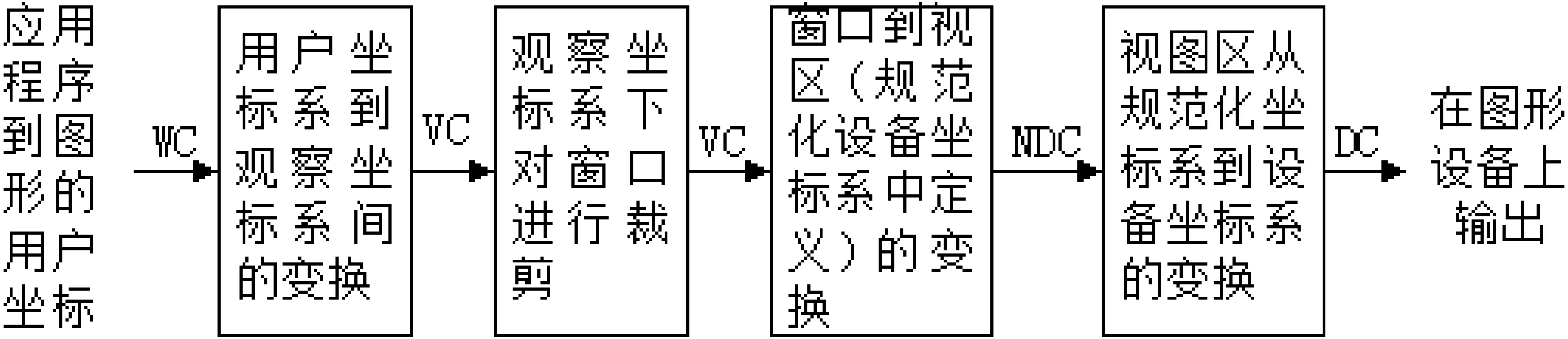
规则化设备坐标系

- [0,1],[0,1]

操作系统负责

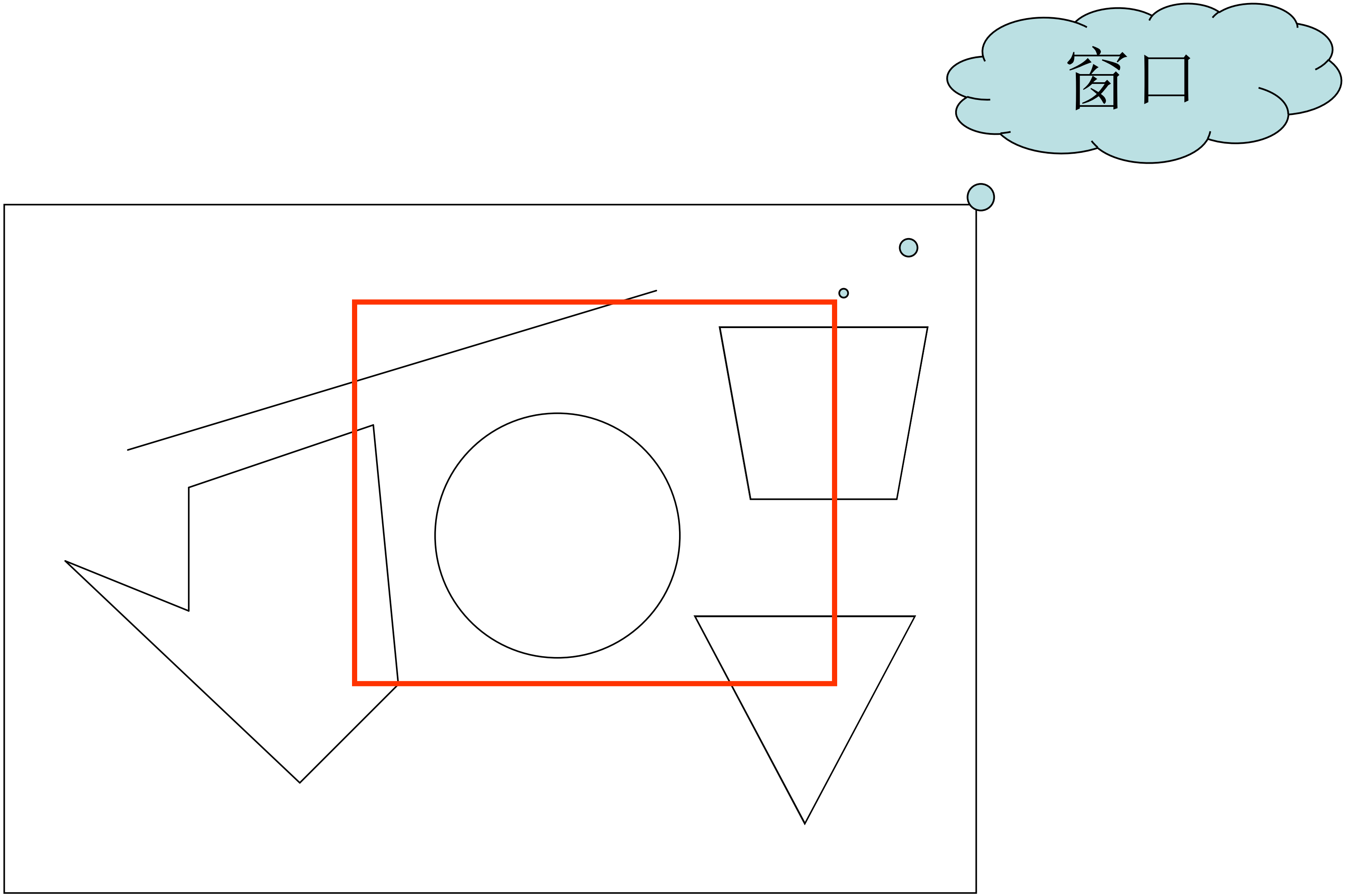


二维观察流程


应用程序到图形的用户坐标
WC
用户坐标系到观察坐标系间的变换
VC
观察坐标系下对窗口进行裁剪
VC
窗口到视区（规范化设备坐标系中定义）的变换
NDC
视图区从规范化坐标系到设备坐标系的变换
DC
在图形设备上输出

裁减

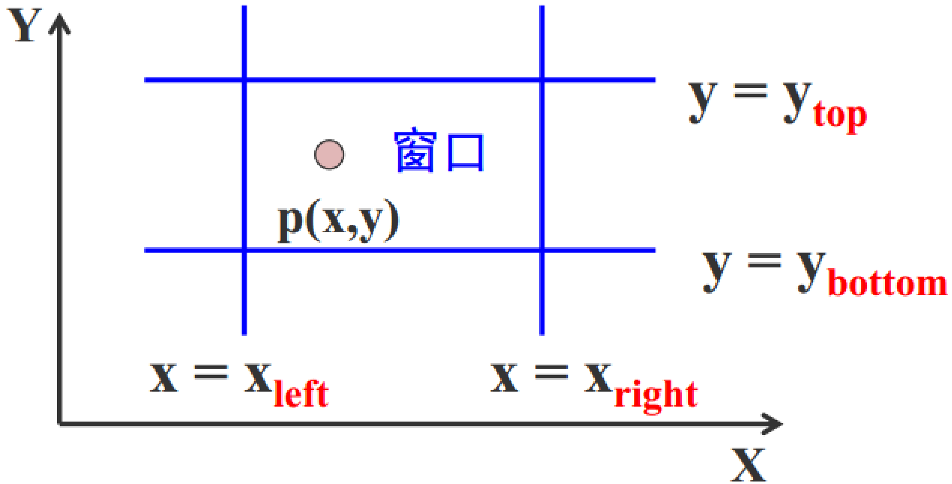
裁减



裁减

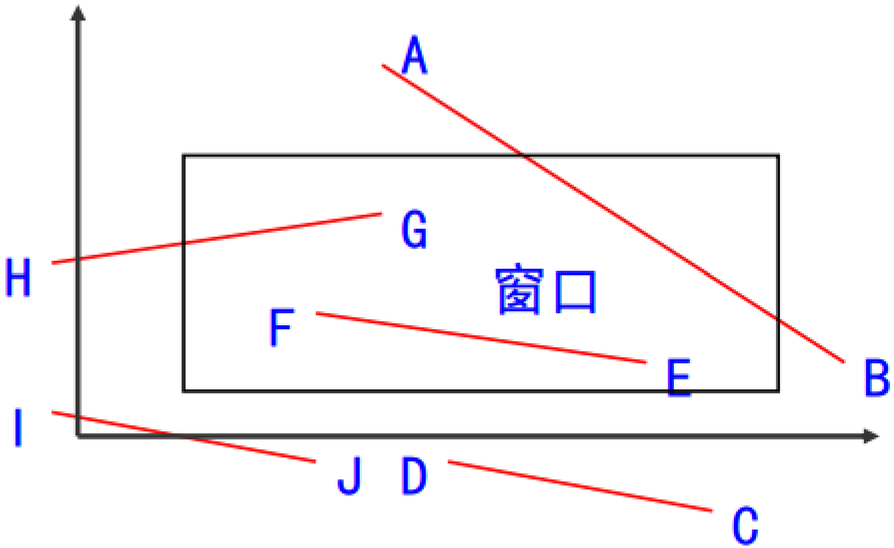
- 点的裁减
- 直线段的裁减
- 多边形的裁减

点的裁减



$$
\begin{cases}
x_{left} \le x \le x_{right} \\
y_{bottom} \le y \le y_{top}
\end{cases}
$$

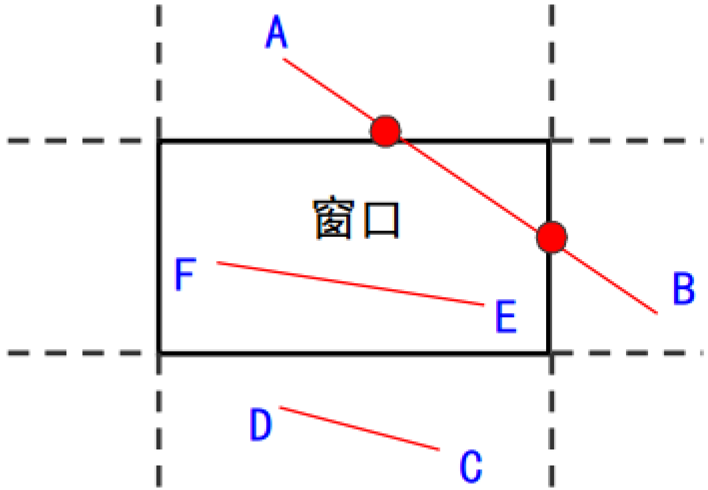
直线段的裁减



常用算法

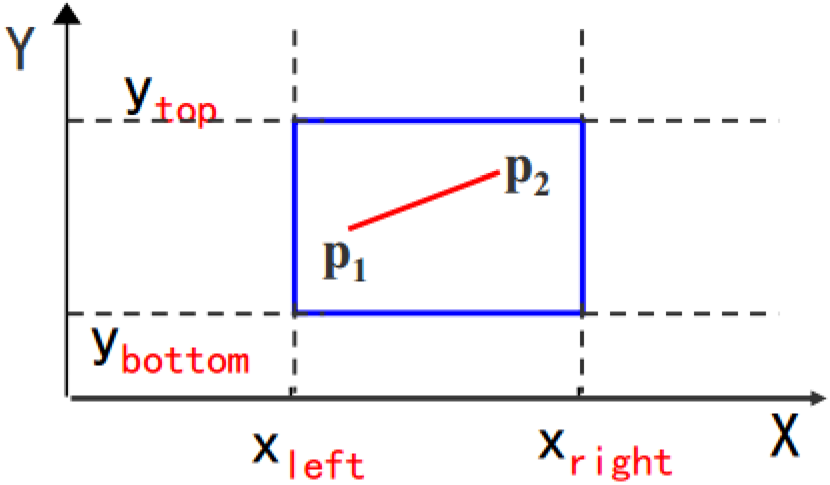
- Cohen-Sutherland
- 中点分割算法
- Liang-Barsky算法

直线段的裁减



Cohen-Sutherland算法

（1）若点p_1和p_2完全在裁剪窗口内



“简取”之

Cohen-Sutherland算法

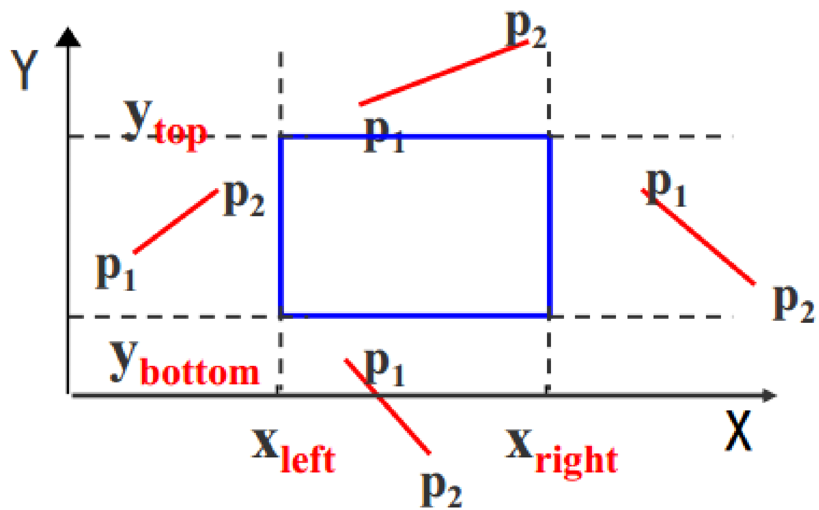
(2) 若点$p_1(x_1, y_1)$和$p_2(x_2, y_2)$均在窗口外，且满足下列四个条件之一：

$x_1 < x_{left}$ 且 $x_2 < x_{left}$

$x_1 > x_{right}$ 且 $x_2 > x_{right}$

$y_1 < y_{bottom}$ 且 $y_2 < y_{bottom}$

$y_1 > y_{top}$ 且 $y_2 > y_{top}$

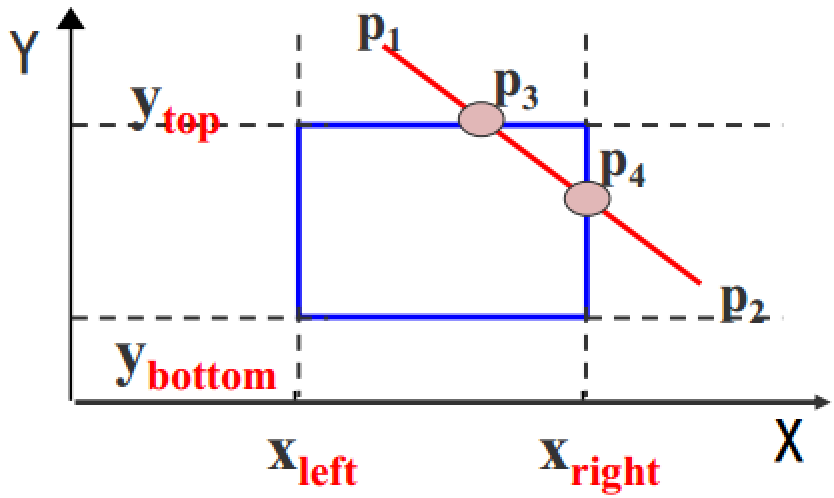


对这四种类型的直线，“简弃”之

Cohen-Sutherland算法

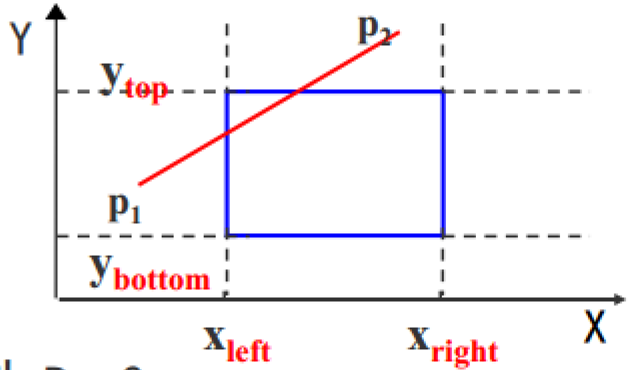
（3）如果直线段既不满足“简取”的条件，也不满足“简弃”的条件？

需要对直线段按交点进行分段，分段后判断直线是“简取”还是“简弃”。



Cohen-Sutherland算法

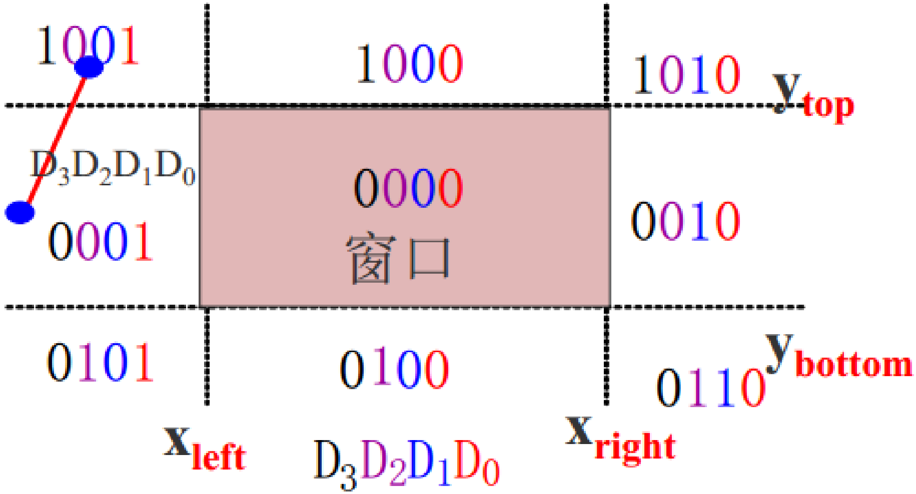
每条线段的端点都赋以四位二进制码$D_3D_2D_1D_0$，编码规则如下：



- 若$x<x_{left}$， 则 $D_0=1$， 否则 $D_0=0$
- 若$x>x_{right}$， 则 $D_1=1$， 否则 $D_1=0$
- 若$y<y_{bottom}$， 则 $D_2=1$， 否则 $D_2=0$
- 若$y>y_{top}$， 则 $D_3=1$， 否则 $D_3=0$

Cohen-Sutherland算法

窗口及其延长线所构成了9个区域。根据该编码规则：



D_0对应窗口左边界

D_1对应窗口右边界

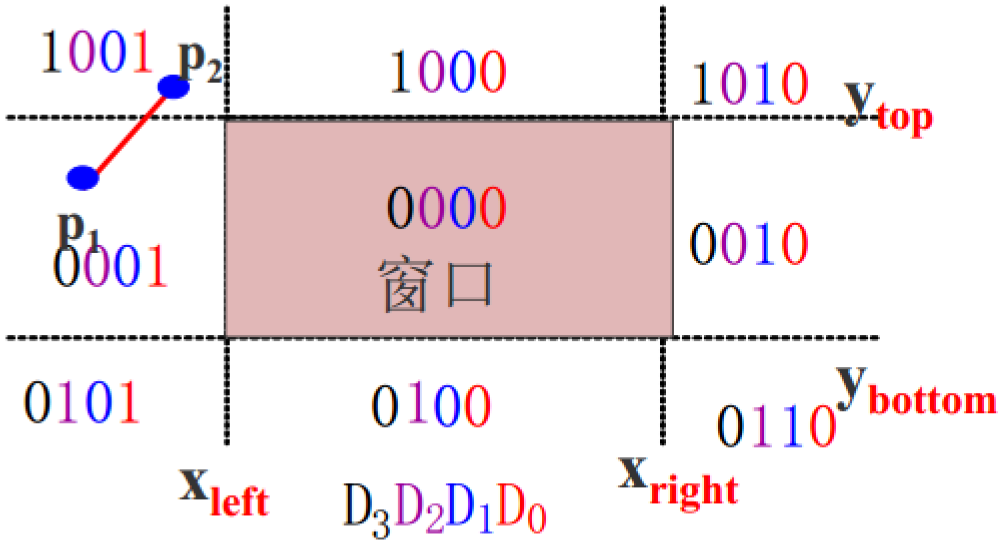
D_2对应窗口下边界

D_3对应窗口上边界

Cohen-Sutherland算法

裁剪一条线段时，先求出端点p_1和p_2的编码$code_1$和$code_2$

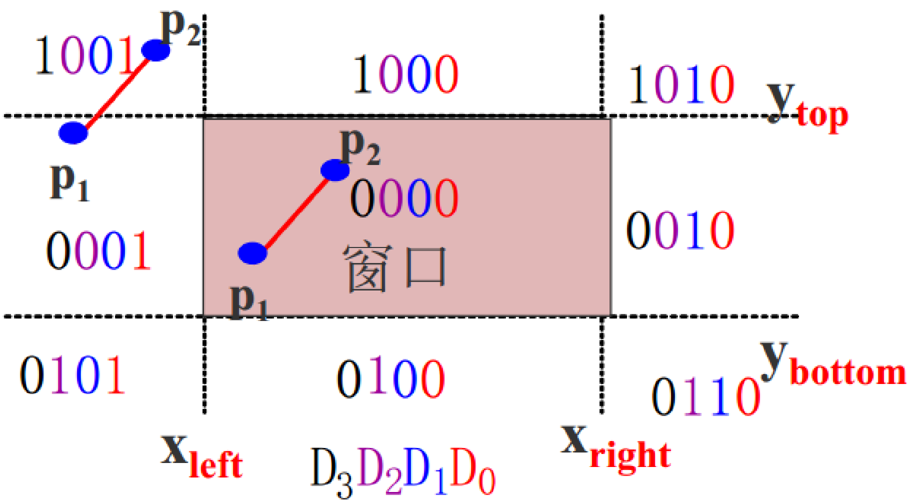
然后进行二进制“或”运算和“与”运算



Cohen-Sutherland算法

（1）若$code_1 | code_2 = 0$，对直线段应**简取**之

	0000
或	0000
	0000



（2）若$code_1 \& code_2 \neq 0$，对直线段可**简弃**之

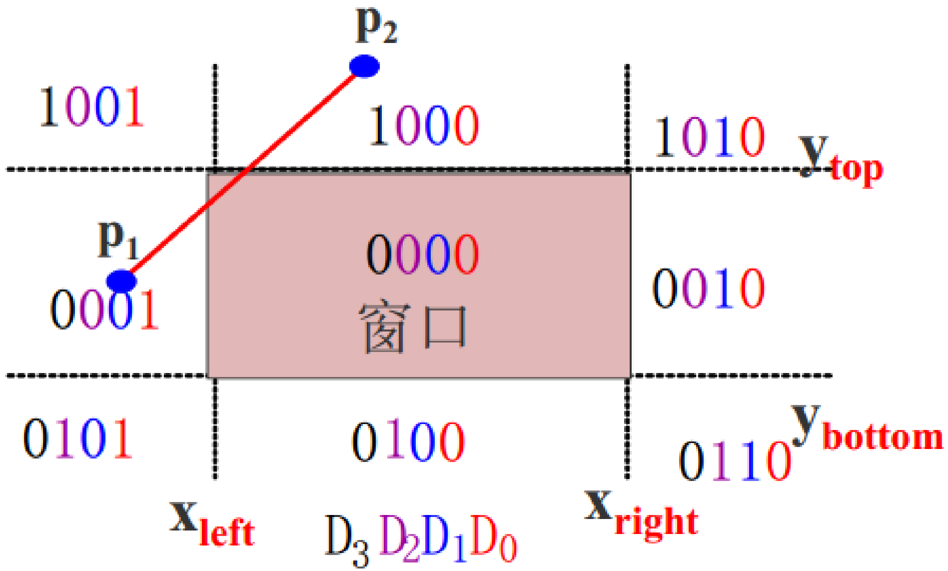
	1001
与	0001
	0001

Cohen-Sutherland算法

若上述两条件均不成立

或	0001
	1000
	1001

与	0001
	1000
	0000



则需求出直线段与窗口边界的交点在交点处把线段一分为二

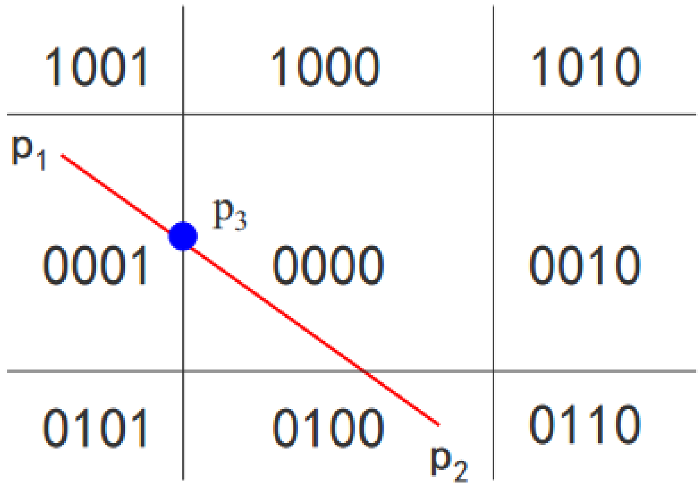
Cohen-Sutherland算法

下面根据该算法步骤来裁剪如图所示的直线段P_1P_2：

首先对P_1P_2进行编码

或	0001
	0100
	0101

与	0001
	0100
	0000



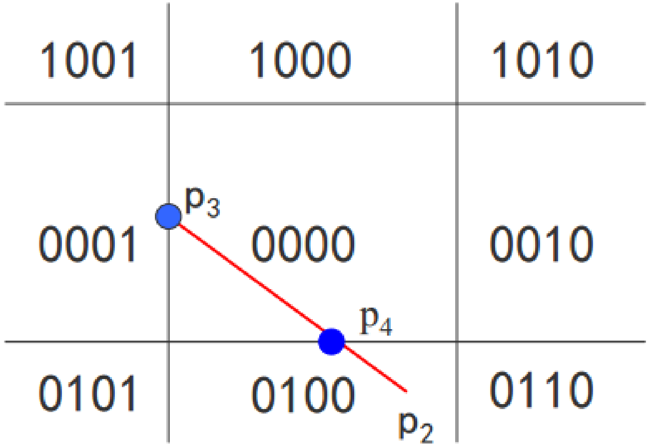
按左、右、下、上的顺序求出直线段与窗口左边界的交点为P_3，P_1P_3必在窗口外，可简弃

Cohen-Sutherland算法

对P_2P_3重复上述处理

	0000	
或	0100	
	0100	

	0000	
与	0100	
	0000	



剩下的直线段（P_3P_4）再进行进一步判断， $code_1|code_2=0$，全在窗口中，简取之。

Cohen-Sutherland算法

- 优点：

  简单，易于实现。可简单地描述为将直线在窗口一侧的部分删去，按左，右，下，上的顺序依次进行，处理之后，剩余部分就是可见的了。用编码方法可快速判断线段的完全可见和显然不可见。

- 缺点：本算法对于其它形状的窗口未必同样有效。
- 另：求交点很重要，决定了算法的速度。

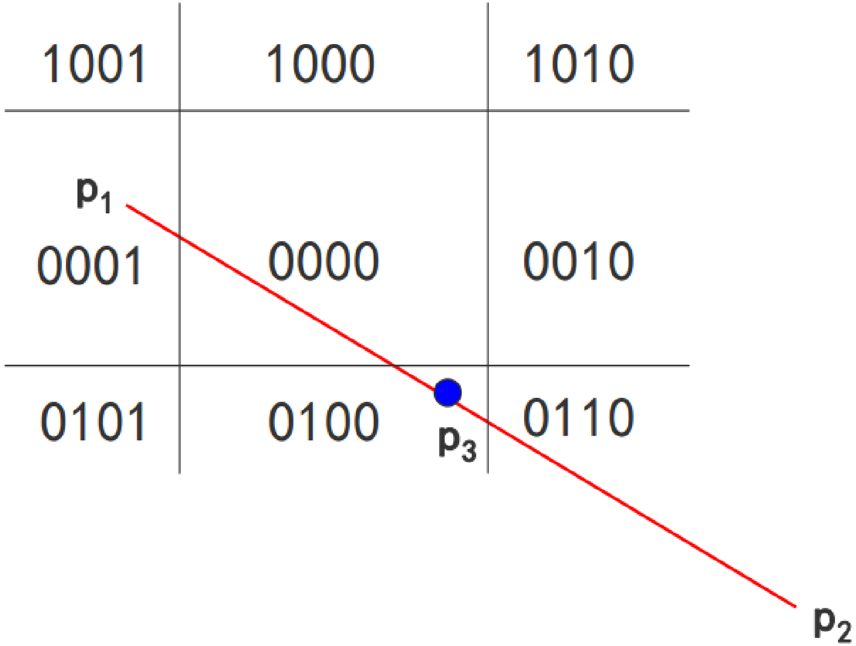
中点分割算法

基本思想：

当对直线段不能简取也不能简弃时，简单地把线段等分为二段，对两段重复上述测试处理，直至每条线段完全在窗口内或完全在窗口外。

中点分割算法

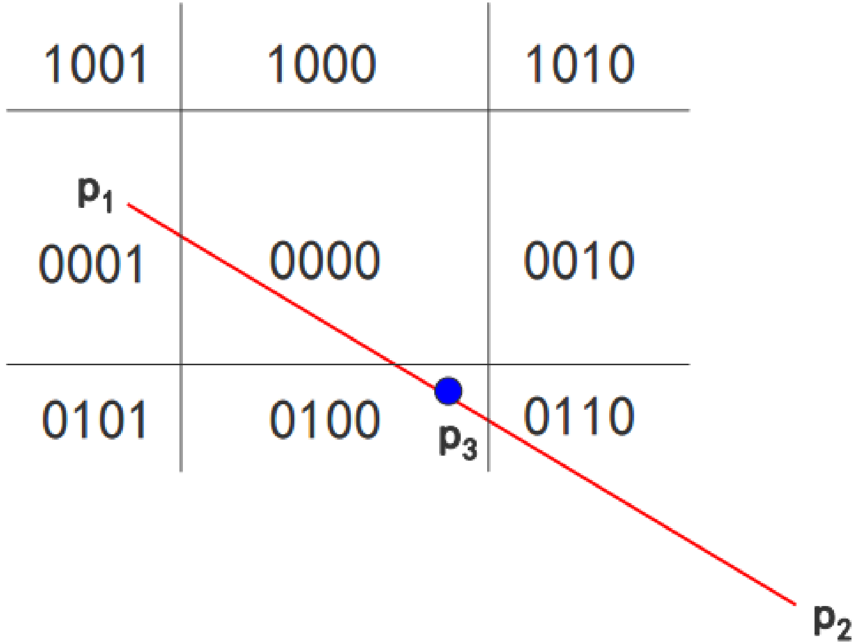
中点分割算法的**核心思想**是通过**二分逼近**来确定直线段与窗口的交点。



中点分割算法

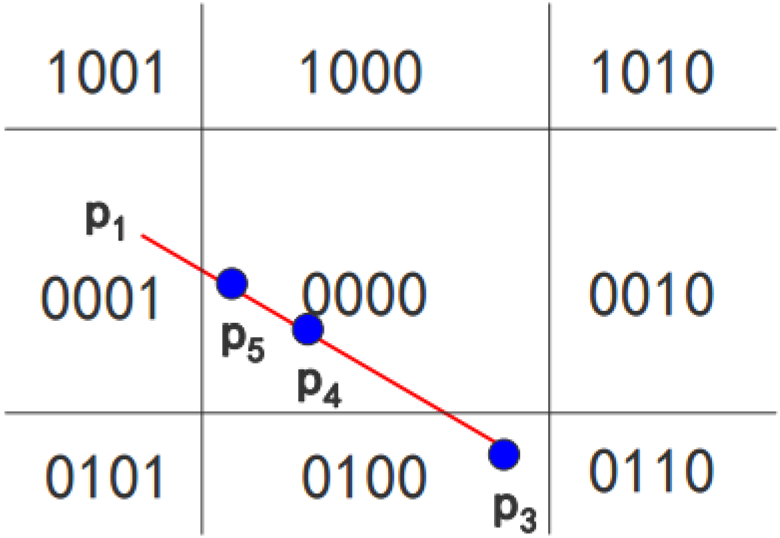
注意：

1、若中点不在窗口内，则把中点和离窗口边界最远点构成的线段丢掉，以线段上的另一点和该中点再构成线段求其中点



中点分割算法

2、如中点在窗口内，则又以中点和最远点构成线段，并求其中点，直到中点与窗口边界的坐标值在规定的误差范围内相等



Liang-Barsky算法

在Cohen-Sutherland算法提出后，梁友栋和Barsky又针对标准矩形窗口提出了更快的Liang-Barsky直线段裁剪算法。

上世纪80年代，梁友栋先生提出了著名的Liang-Barsky算法，至今仍是计算机图形学中最经典的算法之一，也是写进国内外主流《计算机图形学》教科书里的唯一一个以中国人命名的算法。

You-Dong Liang; Barsky, B.A. A new concept and method for line clipping, ACM Transactions on Graphics, Vol.3 1-22,1984

Liang-Barsky算法

RESEARCH CONTRIBUTIONS

A New Concept and Method for Line Clipping

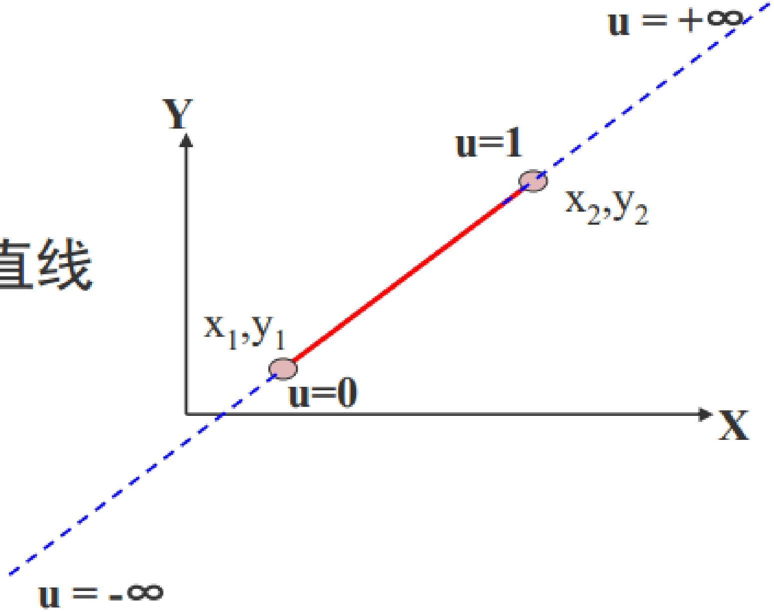
YOU-DONG LIANG and BRIAN A. BARSKY
University of California, Berkeley

A new concept and method for line clipping is developed that describes clipping in an exact and mathematical form. The basic ideas form the foundation for a family of algorithms for two-dimensional, three-dimensional, and four-dimensional (homogeneous coordinates) line clipping. The

Liang-Barsky算法

梁算法的主要思想：

（1）用参数方程表示一条直线

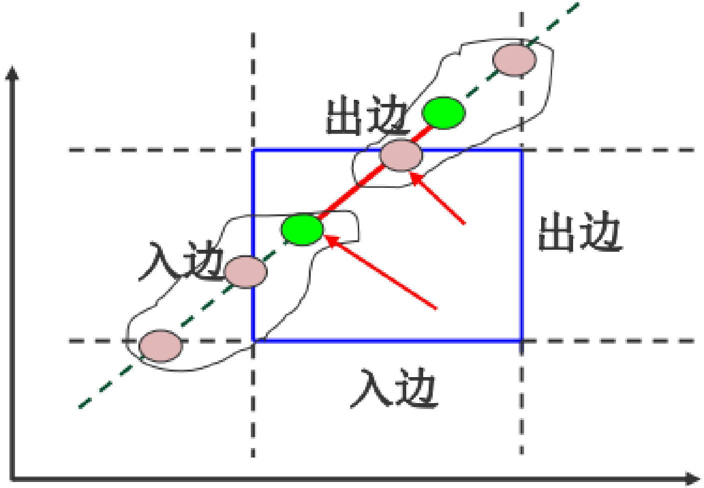


$$x = x_1 + u \cdot (x_2 - x_1) = \underline{x_1 + \Delta x \cdot u}$$
$$y = y_1 + u \cdot (y_2 - y_1) = \underline{y_1 + \Delta y \cdot u} \qquad 0 \le u \le 1$$

Liang-Barsky算法

梁算法的主要思想：

（2）把被裁剪的红色直线段看成是一条有方向的线段，把窗口的四条边分成两类：

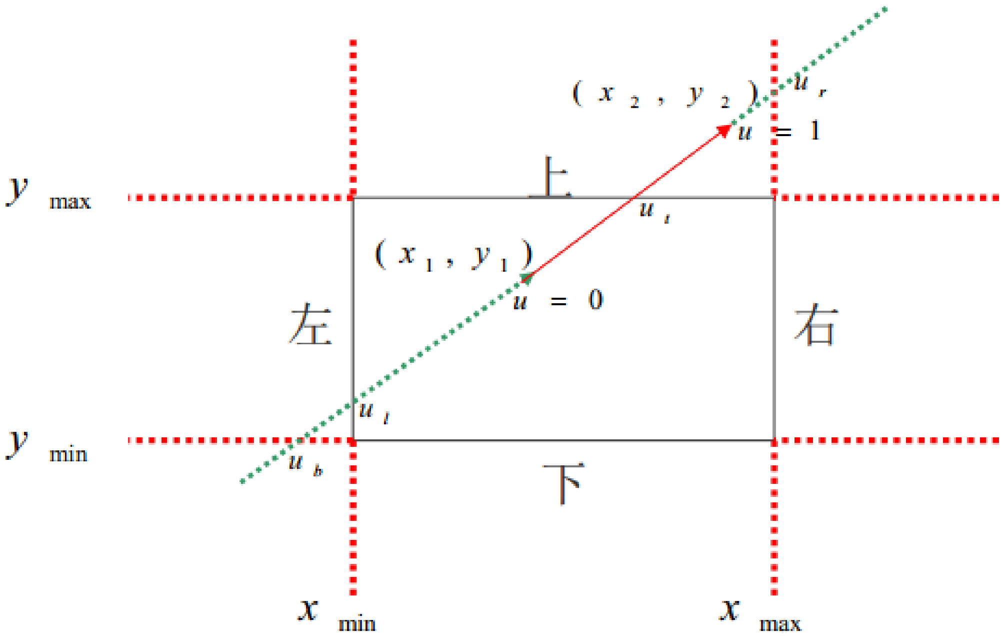


入边和出边

裁剪结果的线段起点是直线和两条入边的交点以及始端点三个点里最前面的一个点，即参数u最大的那个点；

裁剪线段的终点是和两条出边的交点以及端点最后面的一个点，取参数u最小的那个点。

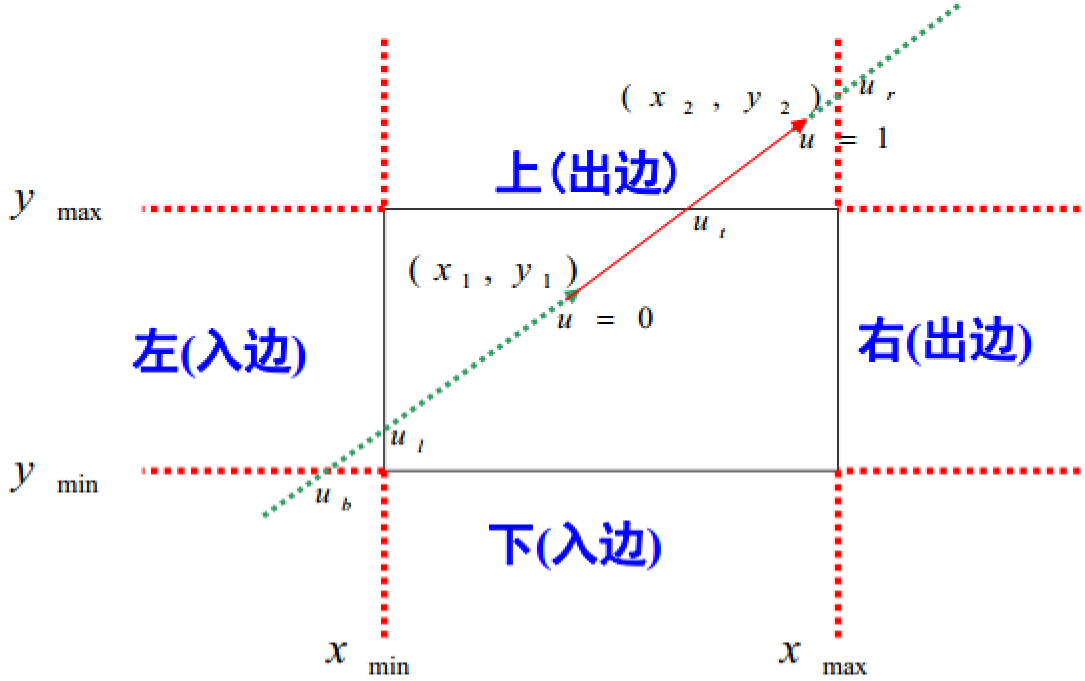
Liang-Barsky算法



值得注意的是，当u从−∞到+∞遍历直线时，首先对裁剪窗口的两条边界直线（下边和左边）从外面向里面移动，再对裁剪窗口两条边界直线（上边和右边）从里面向外面移动。

Liang-Barsky算法

如果用u_1，u_2分别表示线段（$u_1 \leqslant u_2$）可见部分的开始和结束



$$u_1 = \max(0, u_l, u_b)$$

$$u_2 = \min(1, u_t, u_r)$$

Liang-Barsky算法

(1)输入直线段的两端点坐标：(x_1,y_1)和(x_2,y_2)，以及窗口的四条边界坐标：wyt、wyb、wxl和wxr。

(2)若$\Delta x=0$，则$p_1=p_2=0$。此时进一步判断是否满足$q_1<0$或$q_2<0$，若满足，则该直线段不在窗口内，算法转(7)。否则，满足$q_1>0$且$q_2>0$，则进一步计算u_1和u_2。算法转(5)。

(3)若$\Delta y=0$，则$p_3=p_4=0$。此时进一步判断是否满足$q_3<0$或$q_4<0$，若满足，则该直线段不在窗口内，算法转(7)。否则，满足$q_1>0$且$q_2>0$，则进一步计算u_1和u_2。算法转(5)。

(4)若上述两条均不满足，则有$p_k\neq 0$（k=1,2,3,4）。此时计算u_1和u_2。

(5)求得u_1和u_2后，进行判断：若$u_1>u_2$，则直线段在窗口外，算法转(7)。若$u_1<u_2$，利用直线的参数方程求得直线段在窗口内的两端点坐标。

(6)利用直线的扫描转换算法绘制在窗口内的直线段。

(7)算法结束。

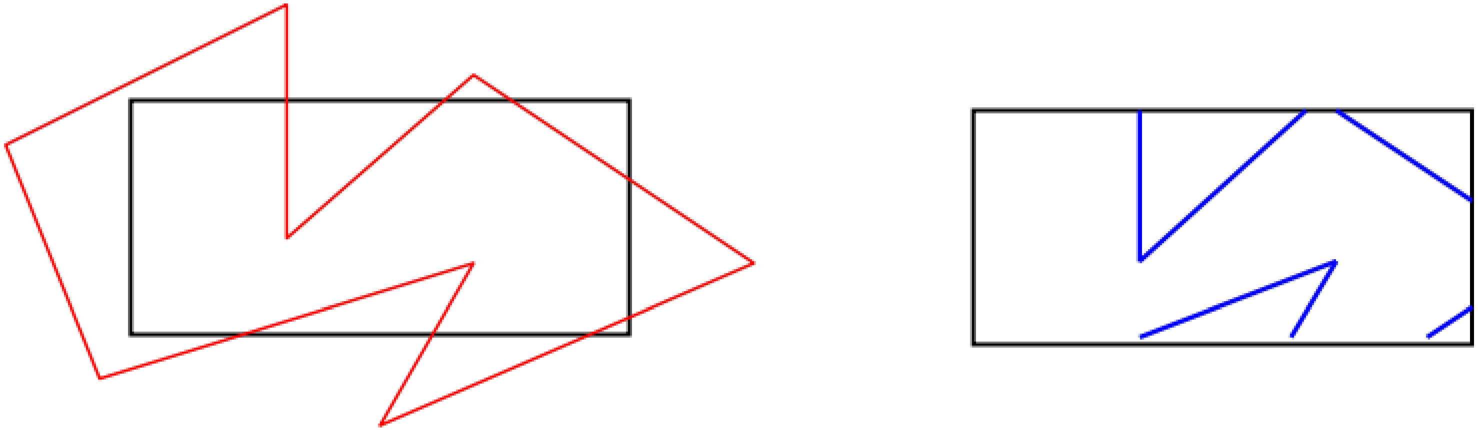
Liang-Barsky算法

梁算法引进了一种新的思想，把窗口的4条边根据线段的方向走向分成入边和出边，裁剪的算法就变得比较简单。

它首先把线段的裁剪变成了射线的裁剪，也就是把线段赋以方向。发现最终裁剪结果的起始端点肯定是入边的两个交点和线段的起始点里面的一点；而裁剪结果的终点肯定是出边的两个交点和线段的终点里面的一点；前三个点取最大，后三个点取最小。这样就把一个经典的裁剪问题变成了解不等式。

这也是中国人的算法第一次出现在了所有图形学教科书都必须提的一个算法。这就叫原始创新！

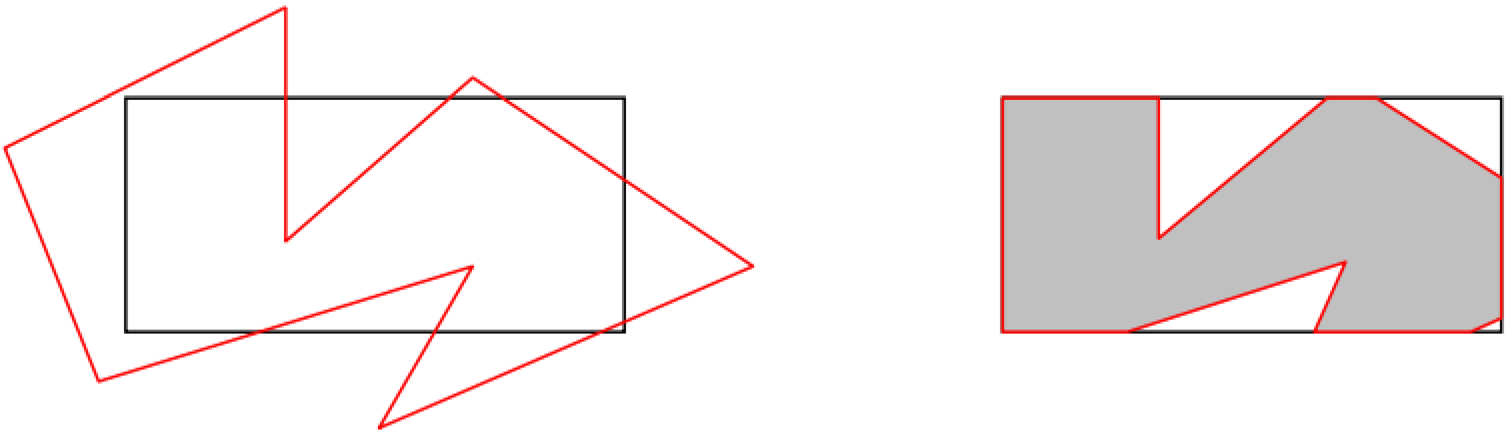
多边形的裁剪



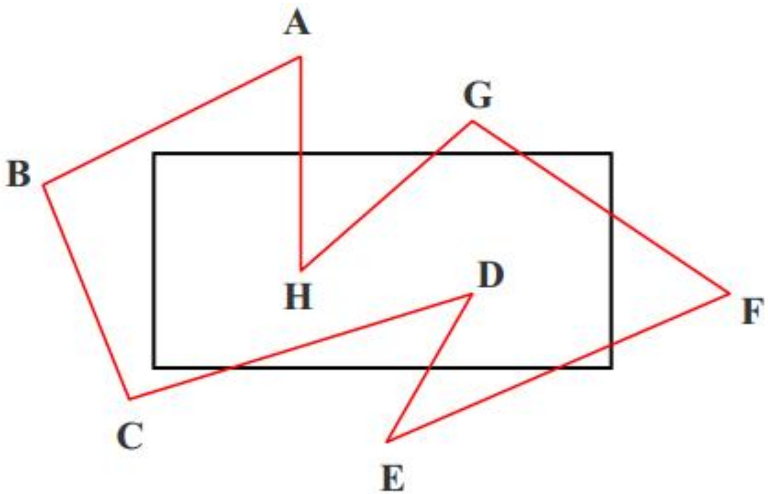
即得到一系列不连续的直线段！

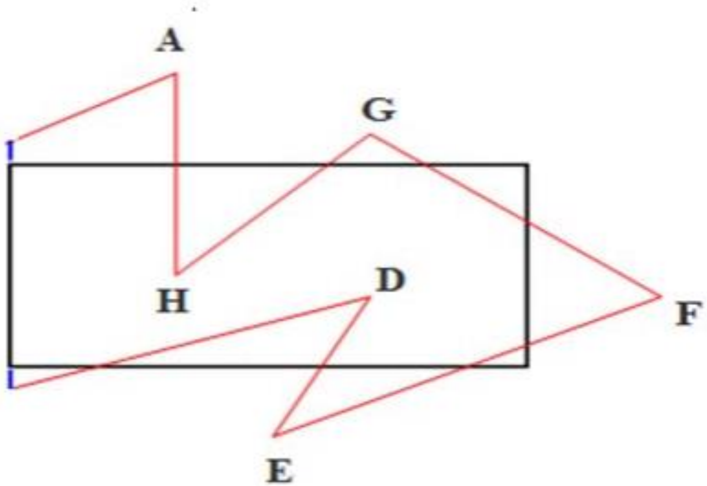
多边形的裁剪

而实际上，应该得到的是下图所示的有边界的区域：



Sutherland-Hodgeman多边形裁剪

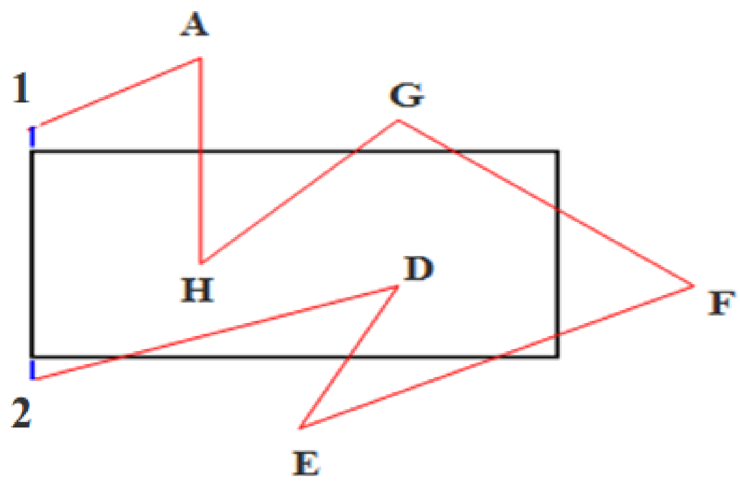


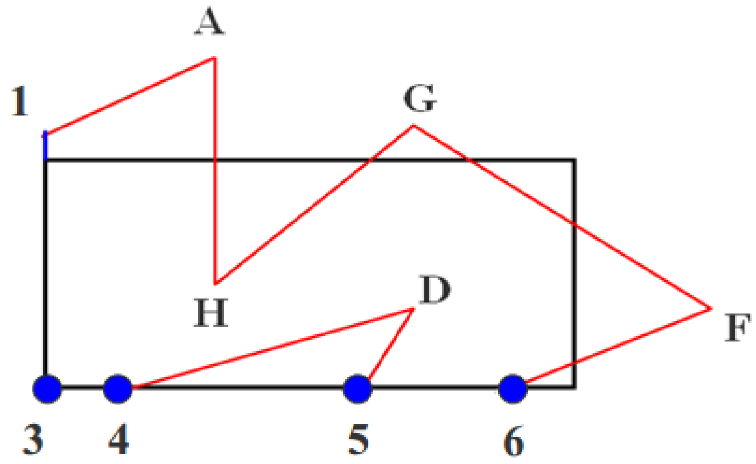


算法的输入：ABCDEFGH　　　　输出：A12DEFGHA

把平面分为两个区域：包含有窗口区域的一个域称为可见侧；不包含窗口区域的域为不可见侧

Sutherland-Hodgeman多边形裁剪





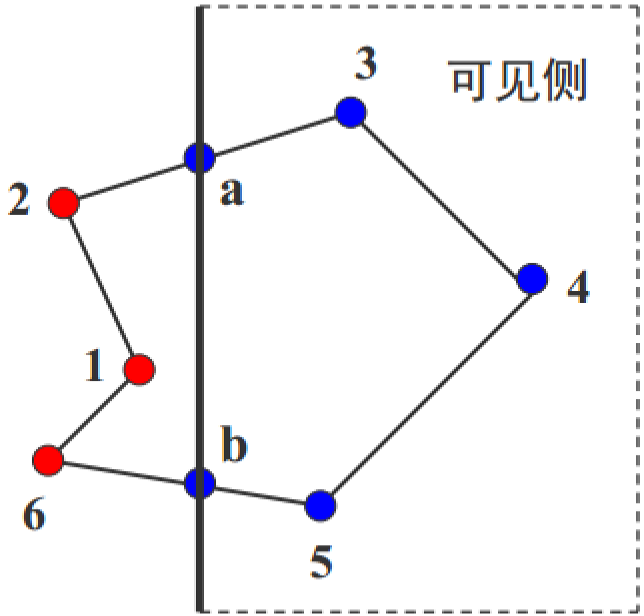
输入：A12DEFGH　　**输出：A134D56FGH**

裁剪得到的结果多边形的顶点有两部分组成：

（1）落在可见一侧的原多边形顶点

（2）多边形的边与裁剪窗口边界的交点

Sutherland-Hodgeman多边形裁剪



输入：1 2 3 4 5 6

输出：

a 3 4 5 b

Sutherland-Hodgeman多边形裁剪

```
while对于每一个窗口边或面 do
  begin
  if P1 在窗口边的可见一侧 then 输出P1
  for i=1 to n do
      begin
      if P1 在窗口边的可见一侧 then
         if P1+1 在窗口边的可见一侧 then 输出 P1+1
         else 计算交点并输出交点
      else if Pi+1 在窗口可见一侧, then 计算交点
           并输出交点, 同时输出Pi+1
      end
  end
end
```

谢 谢