上节要点

- 消隐
  - 概念、分类
  - 三维实体表示方法简介
  - 提高消隐效率的方法
    - 排序、连贯性、包围盒、后向面删除、空间分割
  - 消隐算法
    - 线消隐算法
    - 画家算法
    - **Z-Buffer算法**

Z-Buffer算法()

```
{   帧缓存全置为背景色
    深度缓存全置为最小Z值
    for(每一个多边形)
    {
       for(该多边形在投影平面上投影内的每个象素(x,y) )
      {   计算该多边形在该象素的深度值Z(x,y);
          if(Z(x,y)大于Z缓存在(x,y)的值)
          {   把Z(x,y)存入Z缓存中(x,y)处;
              把多边形在(x,y)处的颜色值存入帧缓存的(x,y)处;
          }
       }
    }
}
```

- 优点
  - **Z-Buffer**算法在象素级上以近物取代远物，实现起来远比总体排序灵活简单；
  - 消隐顺序与形体的遮挡关系无关；
  - 有利于硬件实现；
- 缺点
  - 占用空间大；
  - 对每个多边形覆盖的象素都要计算深度值，没有利用图形的相关性与连续性；

- 解决空间占用大的问题
  - 将整个区域分成若干子区域，一区一区显示，Z缓冲器的单元数即为子区域的象素个数；
  - 以屏幕一行为区域单位处理显示——扫描线Z-Buffer算法。
  - 只用一个深度缓存变量Z-Buffer的改进算法。

一个深度缓存变量Z-Buffer改进算法

```
{   帧缓存全置为背景色
    for(屏幕上的每个象素(i,j))
     {     深度缓存变量zb置最小值MinValue
           for(多面体上的每个多边形Pk)
           {
                if(象素点(i,j)在Pk的投影多边形之内)
                     {
                       计算Pk在(i,j)处的深度值depth;
                        if(depth大于zb)
                       {       zb = depth;
                               indexp = k;
                       }
                     }
           }
            if(zb != MinValue)
                     计算多边形Pindexp在象素(i,j) 处的颜色并显示
    }
}
```

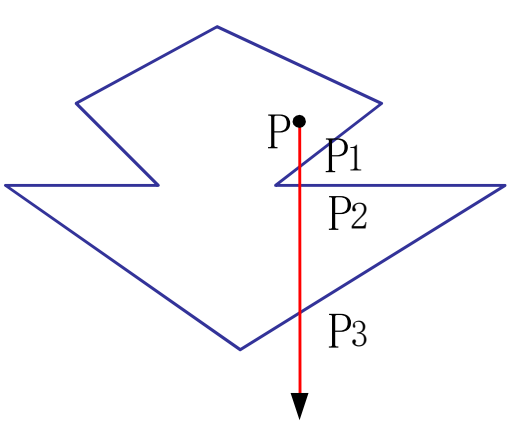
- 关键问题：
  - 判断象素点(i, j)是否在$\mathbf{P}_k$的投影多边形之内
  - 计算多边形$\mathbf{P}_k$在点(i, j)处的深度。设多边形$\mathbf{P}_k$的平面方程为：

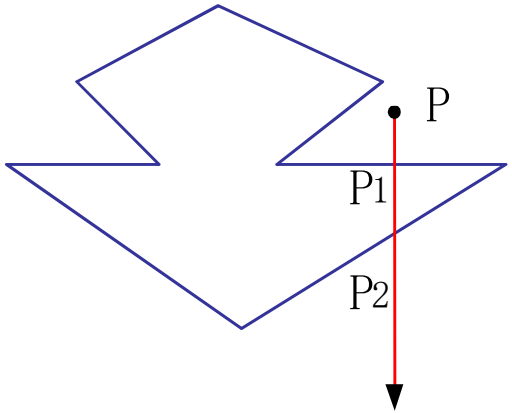
$$ax+by+cz+d=0$$

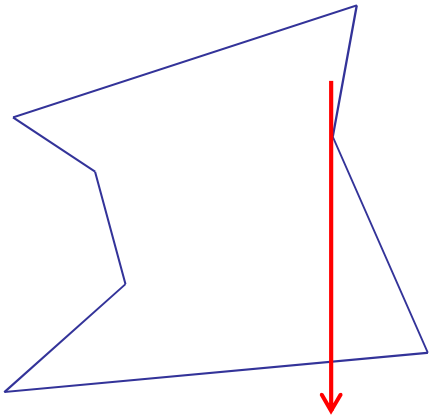
$c=0$?

$$depth = -\frac{ai+bj+d}{c}$$

- 点与多边形的包含性检测
  - 射线法：交点个数为奇数，点在多边形内部；交点个数为偶数，点在多边形外部。
    - 交点在多边形的顶点上的计数原则



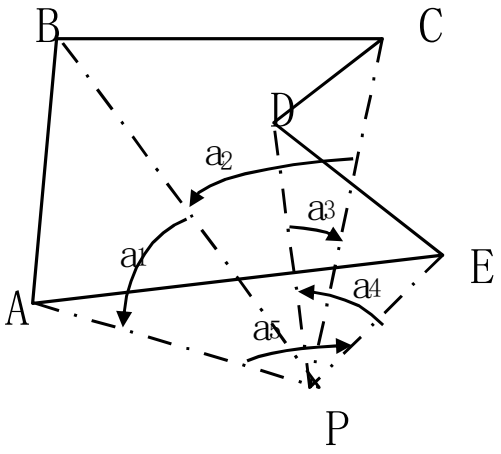


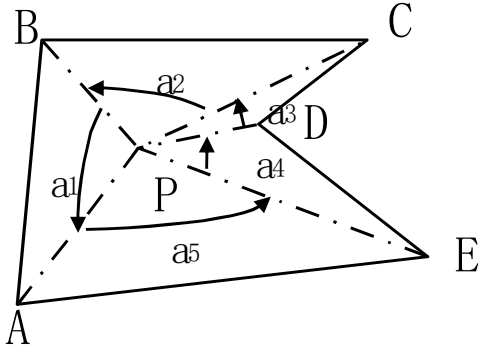


- 无需计算射线与多边形的交点，只要知道有无交点即可，有交点就计数；
- 由点P指向$y=-\infty$的射线与边P_iP_{i+1}的位置关系：
  - $(y<y_i)\&\&(y<y_{i+1})$：P在P_iP_{i+1}下方，无交；
  - $(x>x_i)\ \&\&(x>x_{i+1})$：P在P_iP_{i+1}右方，无交；
  - $(x\leqslant x_i)\ \&\&(x\leqslant x_{i+1})$：$P$在$P_iP_{i+1}$左方，无交；
  - $(y\geqslant y_i)\ \&\&(y\geqslant y_{i+1})\ \&\&((x_i\leqslant x\leqslant x_{i+1})\ \|(x_{i+1}\leqslant x\leqslant x_i))$：$P$在$P_iP_{i+1}$上方，相交；
  - 其它：P在边的包围盒内

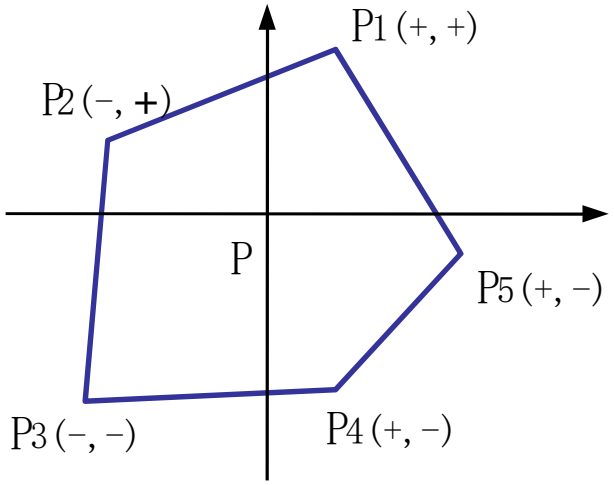
$$y'=\frac{(x-x_i)y_{i+1}+(x_{i+1}-x)y_i}{(x_{i+1}-x_i)}\begin{cases}<y,\text{交点在}P\text{下方，有效}\\>y,\text{交点在}P\text{上方，无效}\\=y,\text{交点即}P,\ P\text{在边界上}\end{cases}$$

- 累计角度法：弧长累计角度为**2π**，点在多边形内部；弧长累计角度为**0**，点在多边形外部。
  - 多边形顶点序列为逆时针方向

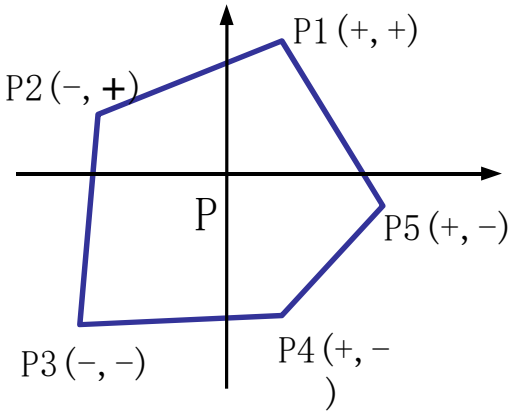




- 以顶点符号为基础的弧长累加方法。
  - 将坐标原点移到被测点***P***。各象限内点的符号对分别为(＋,＋)，(－,＋)，(－,－)，(＋,－)。
  - 算法约定：若顶点P_i的某个坐标为**0**,则其符号为＋。若顶点P_i的***x***、***y***坐标都为**0**，则说明这个顶点为被测点，在这之前予以排除。

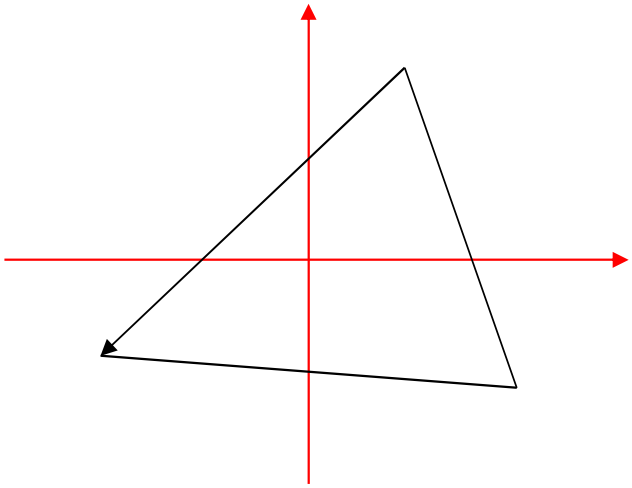


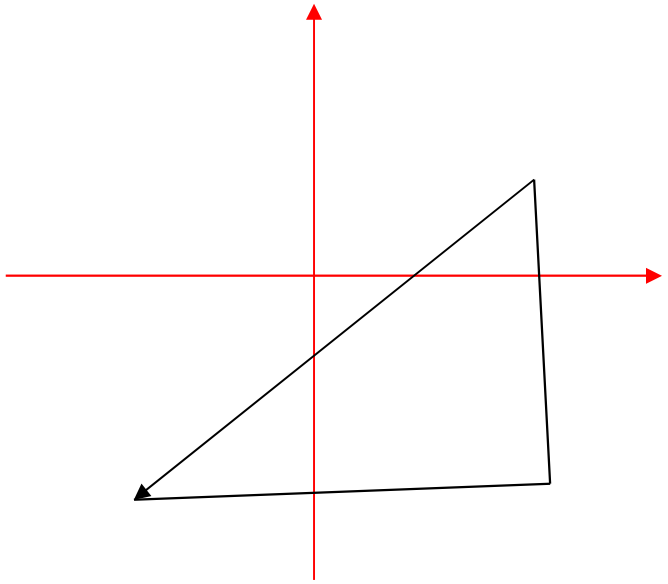
- 弧长变化表：



(sx_i , sy_i)	(sx_{i+1}, sy_{i+1})	弧长变化	象限变化
(＋＋)	(＋＋)	0	I → I
(＋＋)	(－＋)	π/2	I → II
(＋＋)	(－－)	±π	I → III
(＋＋)	(＋－)	-π/2	I → IV
(－＋)	(＋＋)	-π/2	II→ I
(－＋)	(－＋)	0	II→ II
(－＋)	(－－)	π/2	II→ III
(－＋)	(＋－)	±π	II→ IV
…	…	…	…

- 注意：当边的终点P_{i+1}在起点P_i的相对象限时，弧长变化可能增加或减少π。





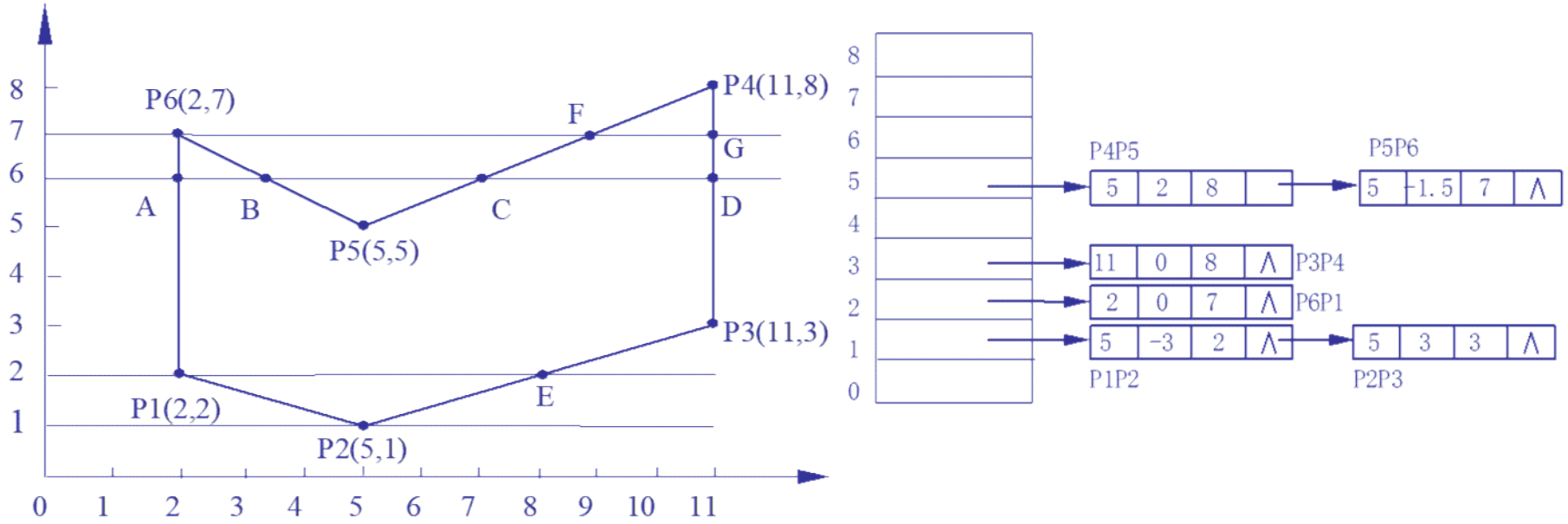
- 设(x_i, y_i)和(x_{i+1}, y_{i+1})分别为边的起点和终点坐标。计算$f = y_{i+1}x_i - x_{i+1}y_i$

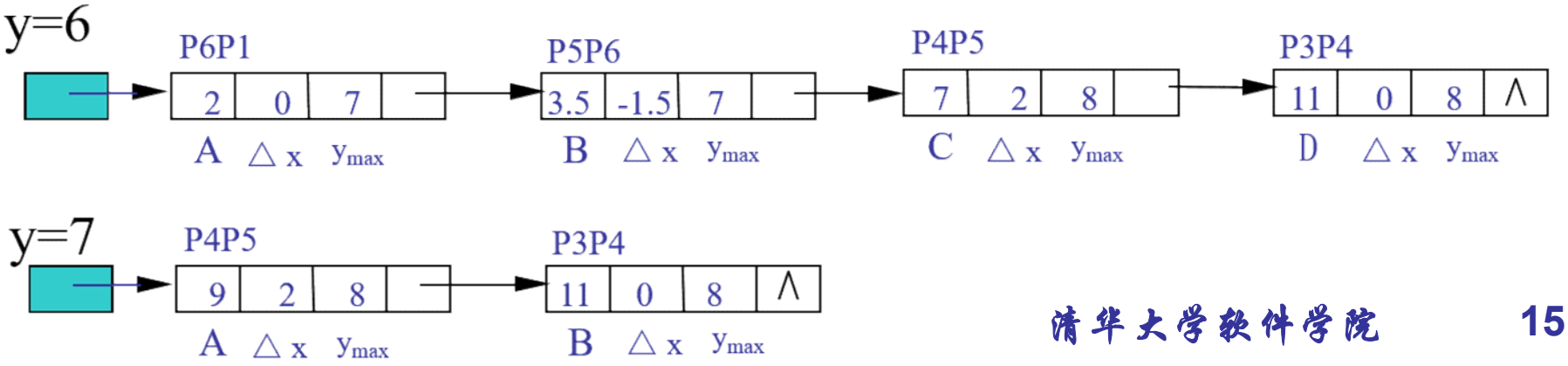
- 若f=0，则边穿过坐标原点。
  - 若f>0，则弧长代数和增加π
  - 若f<0，则弧长代数和减少π

- 点与多边形的包含性检测加速
  - 射线法：只计交点数目，不真正求交点
  - 累计角度法：以顶点符号为基础进行弧长累加，不真正计算角度

- ZB算法的缺点
  - 占用空间大；
  - 对每个多边形覆盖的象素都要计算深度值，没有利用图形的相关性与连续性。

扫描线Z-buffer算法





扫描线Z-buffer算法

- 算法思想
  - 在处理当前扫描线时，开一个一维数组作为当前扫描线的**Z-buffer**。
  - 找出与当前扫描线相关的多边形，以及每个多边形中相关的边对。
  - 对每一个边对之间的小区间上的各象素，采用增量法计算深度，并与**Z-buffer**中的值比较，找出各个象素处可见平面。
  - 计算颜色，写帧缓存。

- 计算x增量

投影多边形的某条边的方程：$px+qy+r=0$

$$x=-(qy+r)/p$$

如果当前扫描线$y=j$与该边的交点为 (x_j, j)，则相邻的下一条扫描线$y=j+1$与该边交点的x坐标：

$$x_{j+1}=-[q(j+1)+r]/p= x_j -q/p$$

$\Delta x= -q/p$：投影边在相邻扫描线间的x增量；

- 计算z增量

多边形所在的平面方程：$ax+by+cz+d=0$

$$z=-(ax+by+d)/c$$

如果当前扫描线$y=j$上的象素(i, j)的深度为z_i，
则该扫描线上相邻的下一个象素$(i+1, j)$的深度

$$z_{i+1}=-[a(i+1)+bj+d]/c= z_i-a/c$$

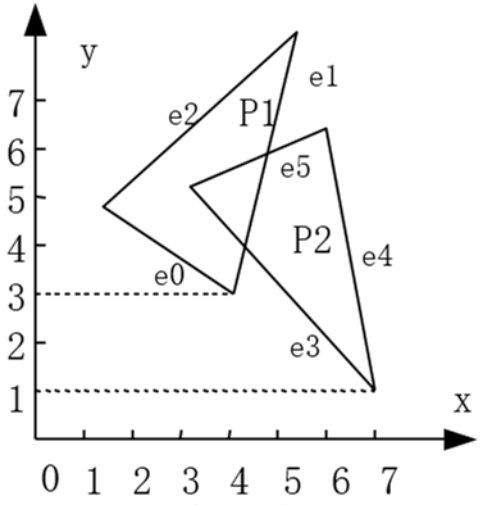
$\Delta z_x= -a/c$：多边形在x方向的深度增量；

- 如果当前扫描线$y=j$与多边形交点象素(x_i, j)的深度为z_i，则下一条扫描线$y=j+1$与多边形交点象素$(x_i+\Delta x, j+1)$深度z_{j+1}

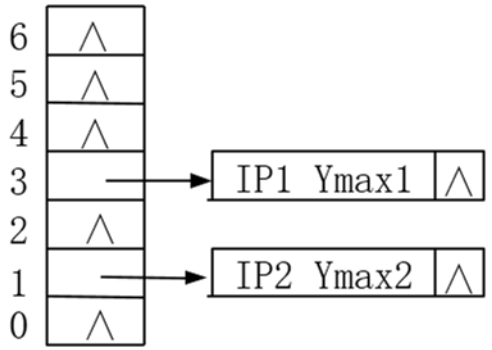
$$z_{j+1}=-[a(x_i+\Delta x)+b(j+1)+d]/c= z_i-a\Delta x/c-b/c$$

$\Delta z_y= -b/c$：多边形在y方向的深度增量；

- 数据结构
  - 多边形Y表：存储所有多边形
    - 根据多边形顶点中最小的*y*坐标，插入多边形Y表中的相应位置。
    - 多边形Y表中只保存多边形的序号和其顶点的最大*y*坐标。根据序号可以从定义多边形的数据结构中取多边形信息。

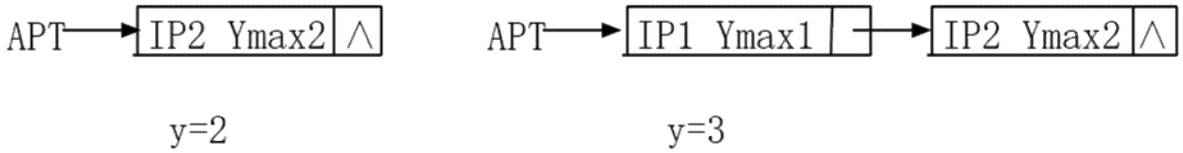


多边形投影

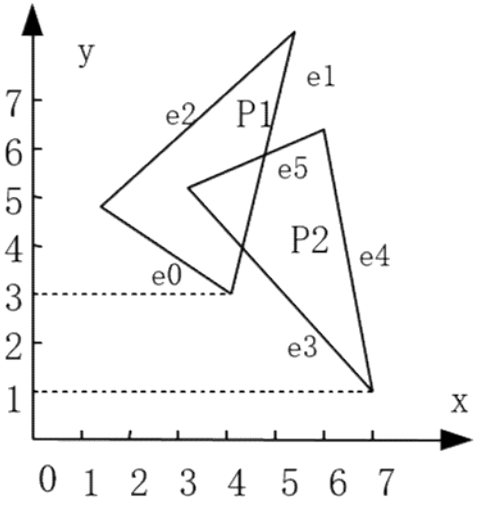


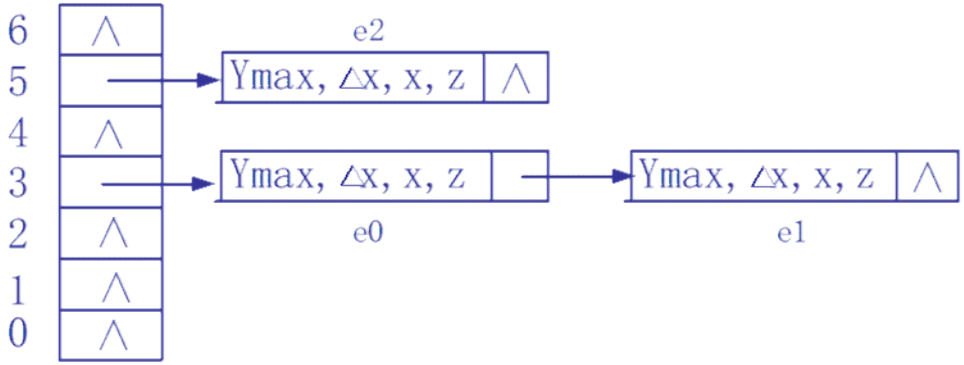
多边形Y表

- 活化多边形表**APT**：与当前扫描线相交的多边形。**APT**是一个动态的链表



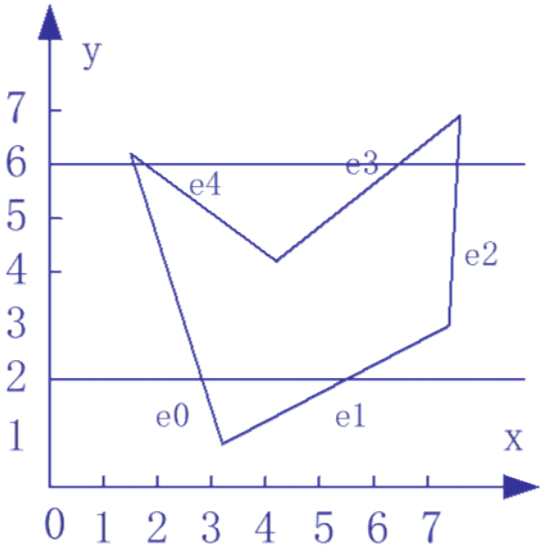
- 边**Y**表**ET**：活化多边形表中的每一个多边形都有一个边表**ET**

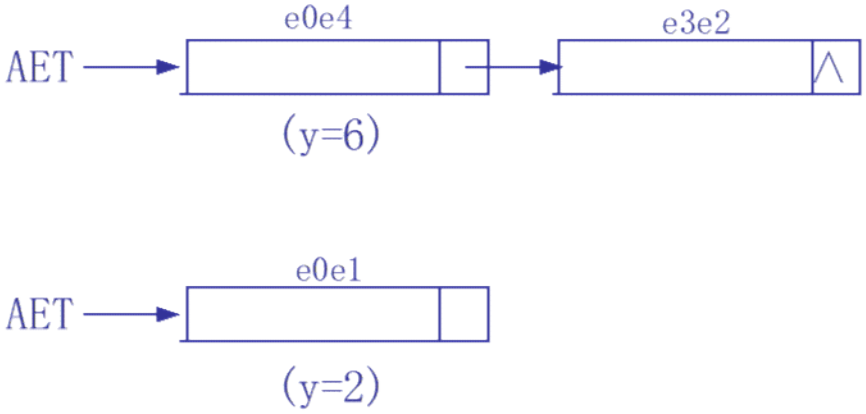




多边形**P1**的边表**ET**

- 活化边对表**AET**
  - 存放当前多边形中与当前扫描线相交的各边对的信息。
  - x_l, Δx_l, y_{lmax}, x_r, Δx_r, y_{rmax}, z_l, IP, Δz_x, Δz_y





扫描线Z-buffer算法()

{ 建多边形Y表；对每一个多边形根据顶点最小的*y*值，将多边形置入多边形Y表。

将活化多边形表APT，活化边对表AET初始化为空。

for(每条扫描线*j*，*j*从小到大)

{ 1. 帧缓存CB中相应单元置为背景色。

2. 深度缓存ZB (一维数组) 置为无穷小。

3. 将对应扫描线*j*的多边形Y表中的多边形加入到活化多边形表APT中。

4. 对新加入的多边形，生成其相应的边Y表。

5. 对APT中每一个多边形，若其边Y表中对应扫描线*j*增加了新的边，将新的边配对，加到活化边对表AET中。

6. 对**AET**中的每一对边：

6.1 对$x_l < x < x_r$ 的每一个象素，按增量公式$z = z+\Delta z_x$算各点深度**depth**。

6.2 与**ZB**中的量比较，**depth > ZB(*j*),** 则令**ZB(*j*) =depth**，并计算颜色值，写帧缓存。

7. 删除**APT**中多边形顶点最大*y*坐标为*j*的多边形，并删除相应的边**Y**表**ET** 。

8. 对**AET**中的每一个边对，作如下处理：

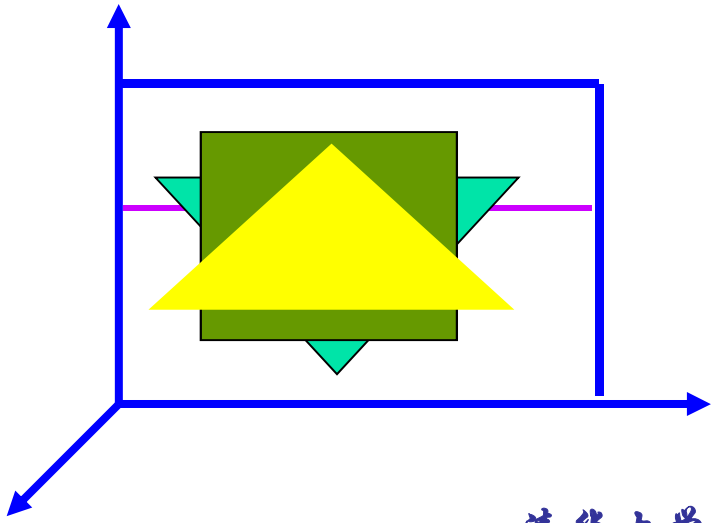
8.1 删除y_{lmax}或y_{lmax}已等于*j*的边。若一边对中只删除了其中一边，需对该多边形的边重新配对。

8.2 用增量公式计算新的x_l、x_r和z_l，更新边对中的信息；

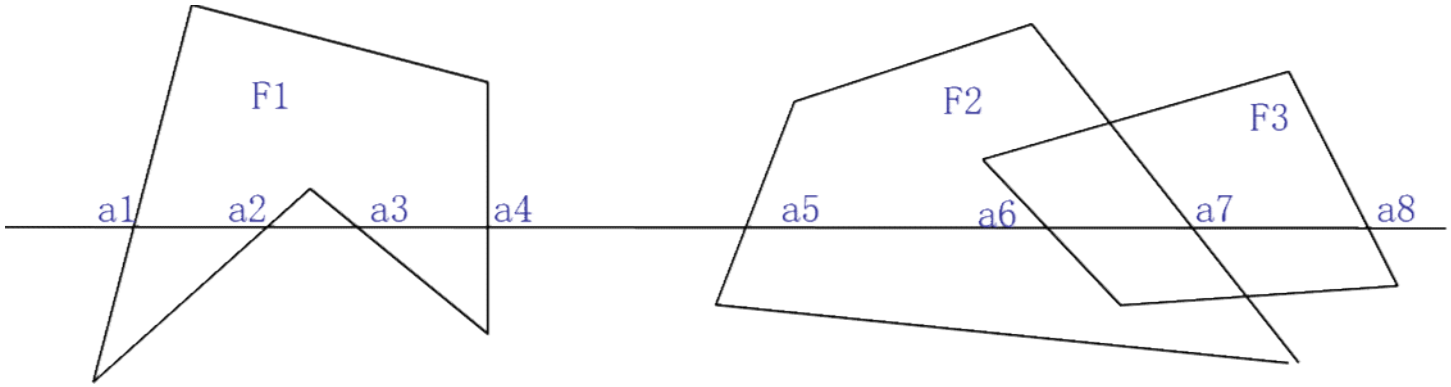
}

}

- 扫描线ZB算法对ZB算法的改进
- 缺点
  - 在每一个被多边形覆盖像素处需要计算深度值；
  - 被多个多边形覆盖的像素需要多次计算深度值；



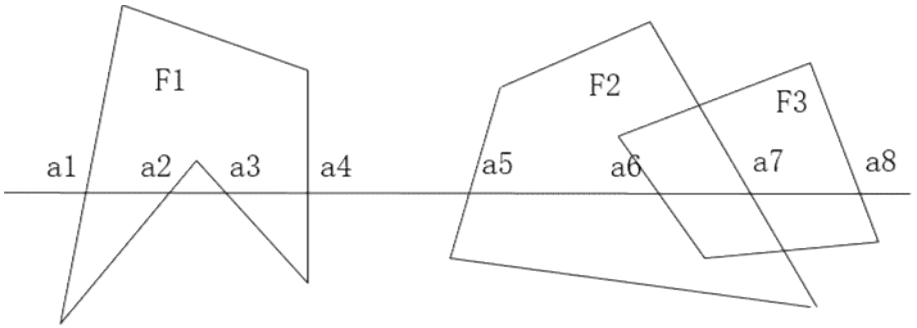
区间扫描线算法



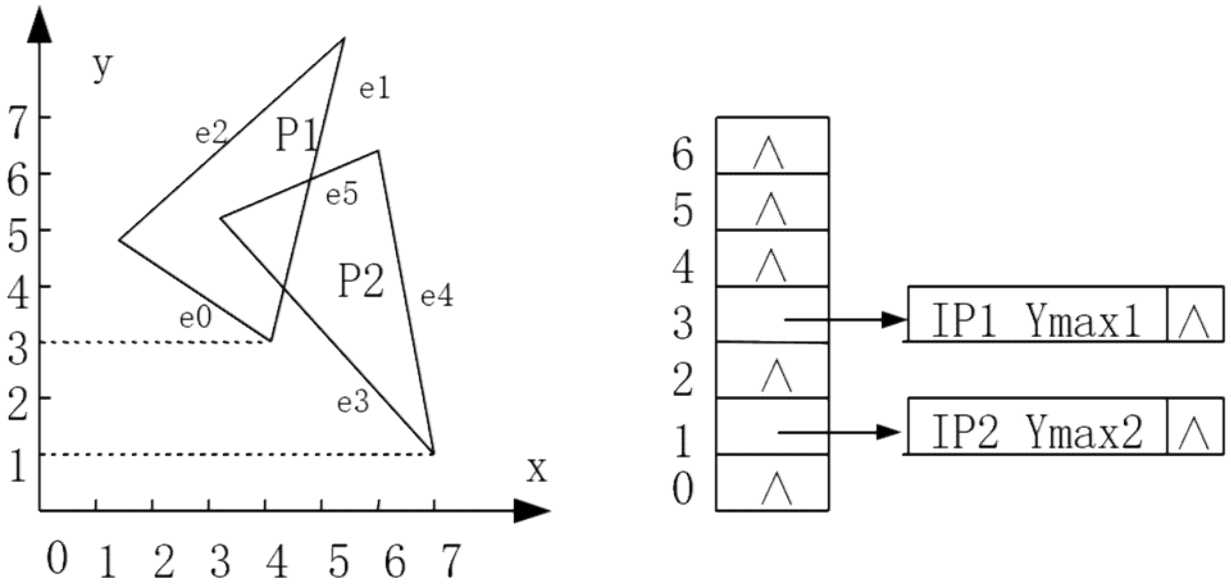
- 基本思想：把当前扫描线与各多边形在投影平面的投影的交点进行排序后，使扫描线分为若干子区间。只要在区间任一点处找出在该处**z**值最大的一个面，这个区间上的每一个象素就用这个面的颜色来显示。

- 优点：将逐象素计算改为逐段计算。

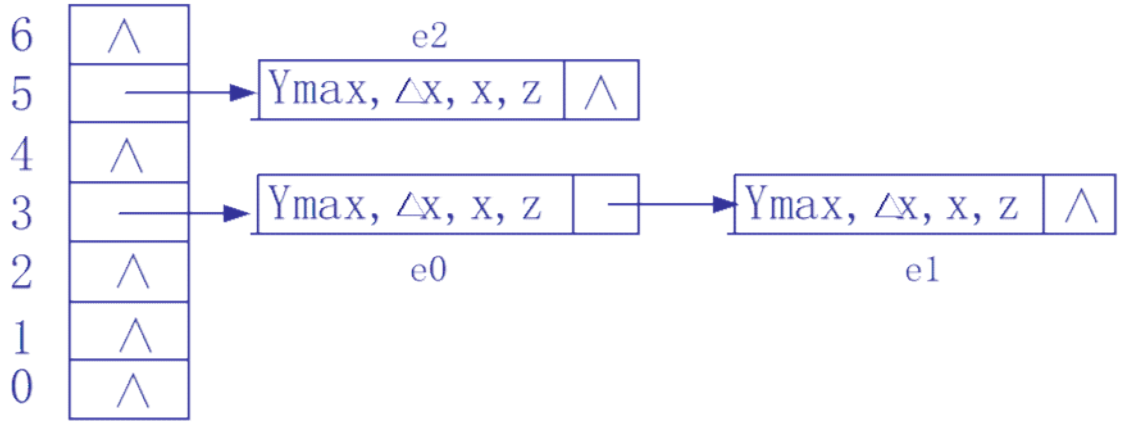
- 确定小区间的颜色：
  - 小区间上没有任何多边形，该小区间用背景色显示。
  - 小区间上只有一个多边形，可用对应多边形在该处的颜色显示。
  - 小区间上存在两个或两个以上的多边形，必须通过深度测试判断哪个多边形可见。



- 确定某区间内哪一多边形可见：
  - 可在区间内任取一采样点 (如区间中点)，分析该点处哪个多边形离视点最近，该多边形即是在该区间内可见的多边形。
- 具体实现：
  - 多边形**Y**表：存所有多边形。
    - 根据多边形顶点中最小的*y* 坐标，插入多边形**Y**表中的相应位置。


y
x
e0
e1
e2
e3
e4
e5
P1
P2
0 1 2 3 4 5 6 7
6
5
4
3
2
1
0
IP1 Ymax1
IP2 Ymax2

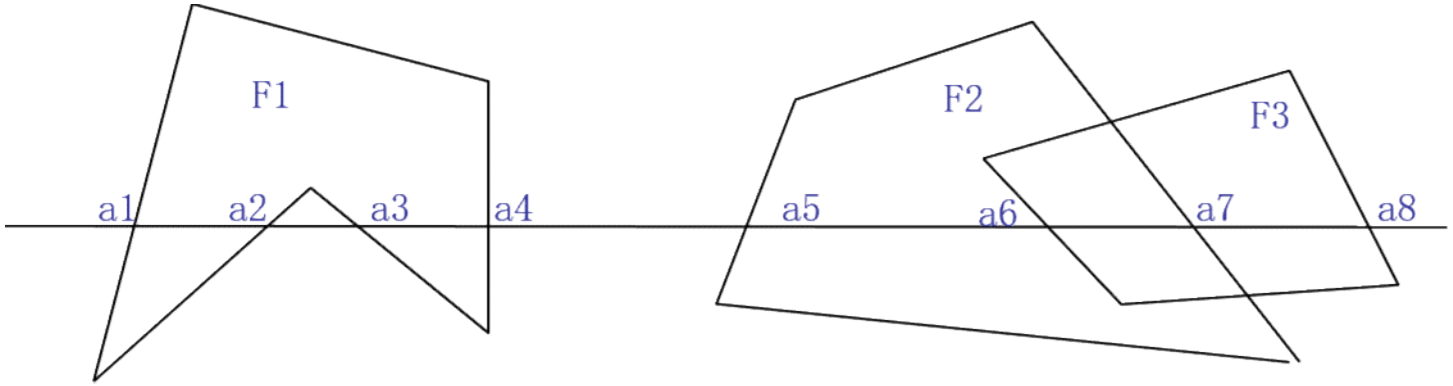
- 活化多边形表**APT**：与当前扫描线相交的多边形。
- 边**Y**表**ET**：活化多边形表中的每一个多边形都有一个边表**ET**。



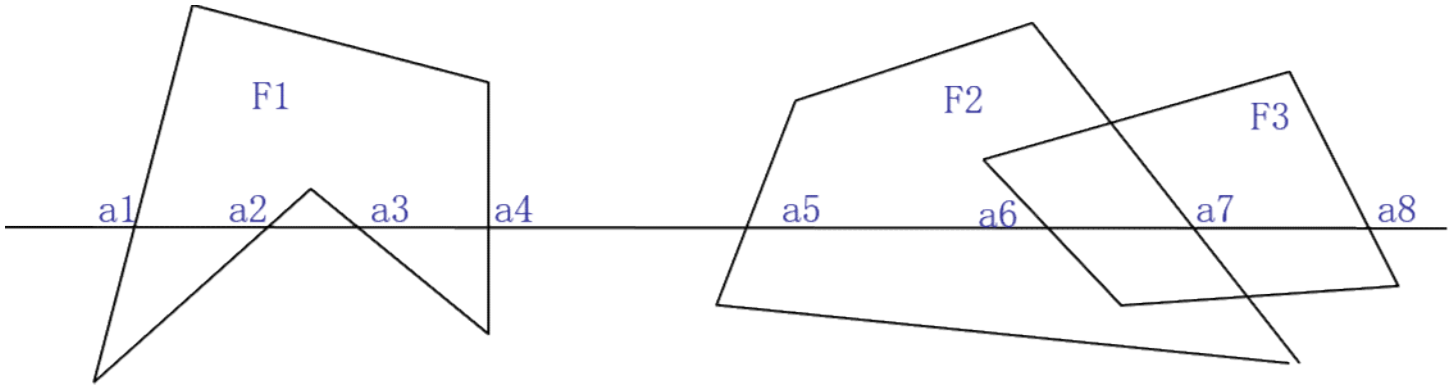
- 活化边表**AET**：投影与扫描线有交的多边形的边按交点的x坐标值由小到大排列；

$x, \Delta x, y_{max}$, IP, z

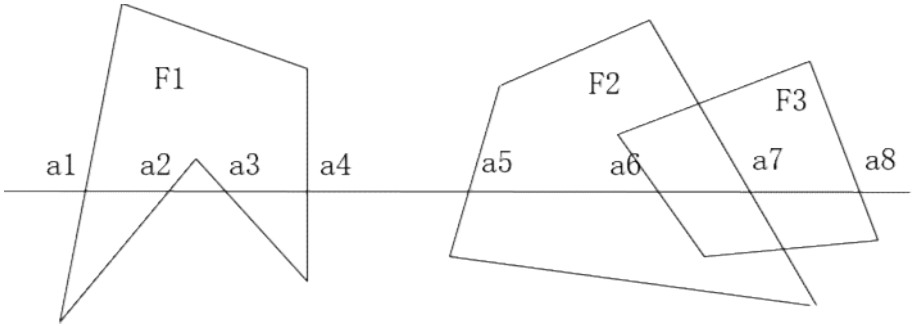
- 区间扫描线算法与扫描线**Z-buffer**算法数据结构的区别：活化边表中的结点是边，而非边对。

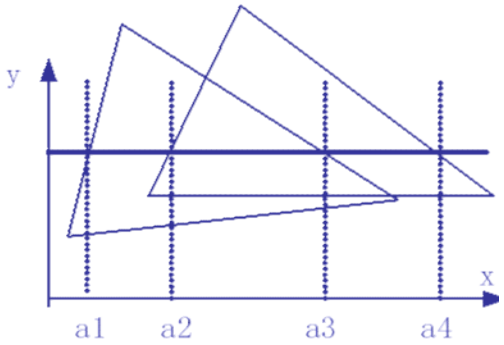


- 关键：如何知道每一个区间中，有几个相关的多边形？是哪几个？
- 解决方案：活化多边形表中增加一个标志，**$flag_i$=0**；每遇到该多边形的边，**$flag_i$**取反。

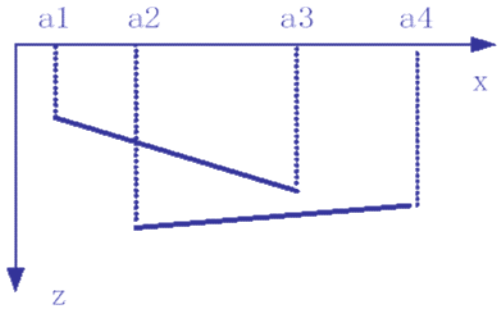


	$flag_1$	$flag_2$	$flag_3$	flag
[-∞, a1)	0	0	0	0
[a1, a2)	1	0	0	1
[a2, a3)	0	0	0	0
[a3, a4)	1	0	0	1
[a4, a5)	0	0	0	0
[a5, a6)	0	1	0	1
[a6, a7)	0	1	1	2
[a7, a8)	0	0	1	1
[a8, ∞)	0	0	0	0

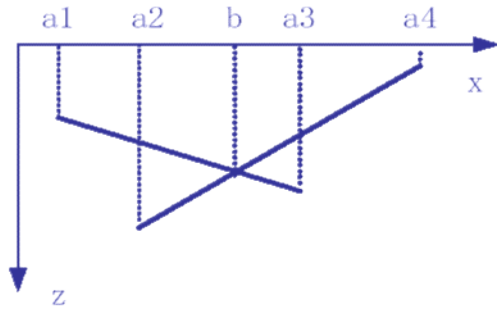




两多边形在屏幕上的投影



无贯穿的情形

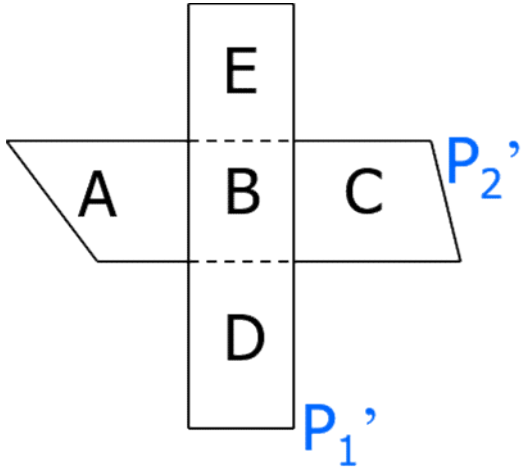


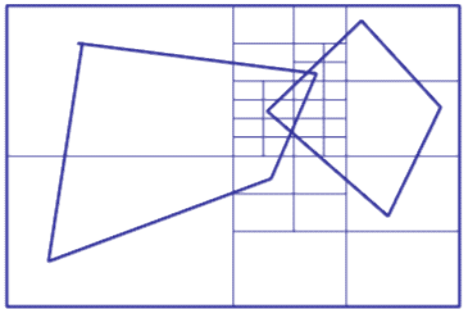
相互贯穿的情形

- 若允许物体表面相互贯穿时，还必须求出它们在扫描平面(**ZX**平面)的交点。用这些交点把该小区间分成更小的子区间(称为间隔)，在这些间隔上决定哪个多边形可见。如将**[a2,a3)**区间分成**[a2,b)[b,a3)**两个子区间。

区域子分割算法 (Warnack算法)

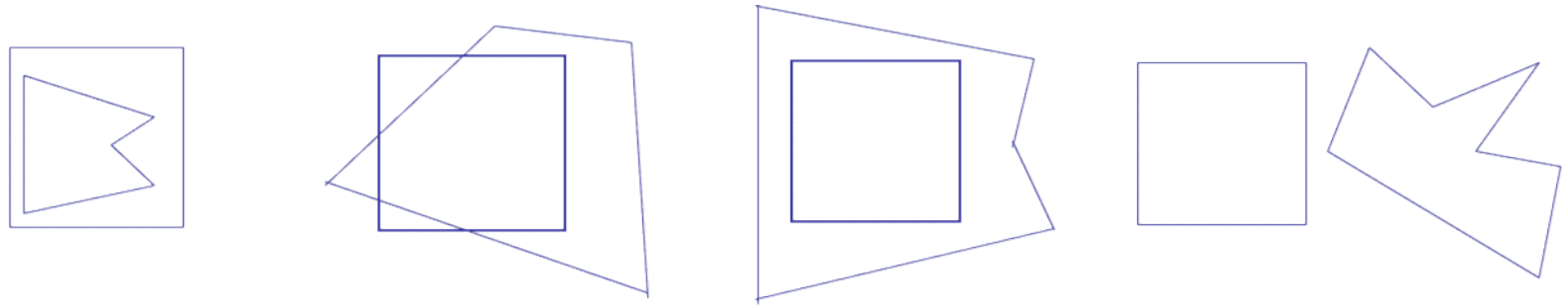
- 基本思想
  - 把物体投影到屏幕区域上，然后递归分割区域，直到区域内目标足够简单，可以显示，或者已经分割到象素为止。





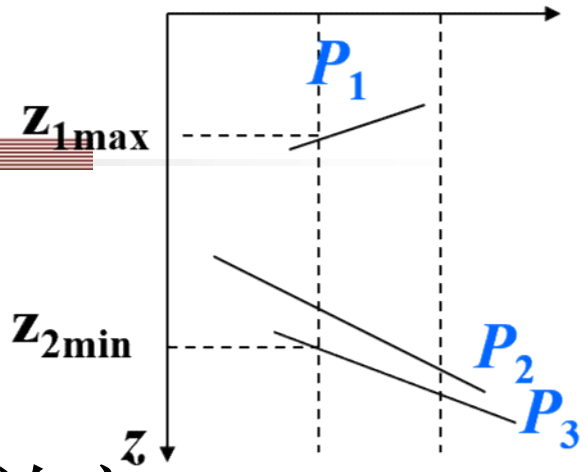
区域子分的过程

- 区域与多边形的简单覆盖关系有四种：内含、相交、包围和分离。



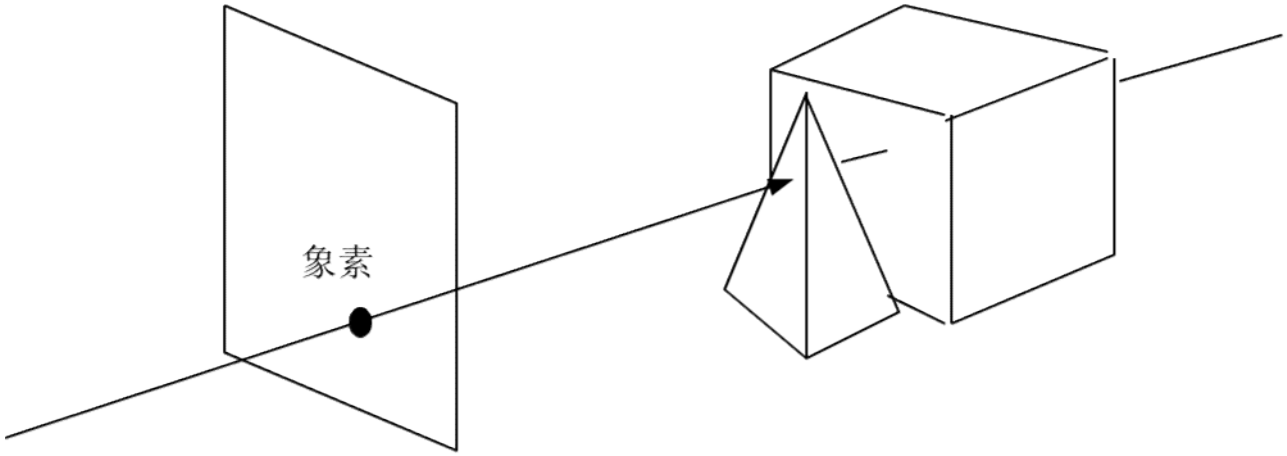
- 算法步骤：
  - 判断多边形和区域的位置关系是否属于：
    - 所有多边形均与区域分离。该区域置背景色
    - 只有一个多边形与区域相交，或该多边形完全在区域内，则先整个区域置背景色，再对多边形在区域内的部分用扫描线算法绘制。
    - 有一个多边形包围了区域，或区域与多个多边形相交，但离视点最近的一个多边形包围区域，把整个区域填上离视点近的那个多边形的颜色。
  - 将不属于上述三种情况的区域一分为四，重复上述步骤，直到区域的边长为一个象素，取该区域的颜色为最靠近视点的多边形颜色或与该窗口相交的多边形颜色的平均。

- **Divide-Conquer**



- 处理细节问题，提高算法效率：
  - 空间分割；
  - 对所有多边形按其顶点的**z**坐标最大值排序，组成多边形列表，随着区域的分割，不断修改多边形列表，从多边形列表中找与视点最近的多边形；
  - 先尽量用包围盒法判断区域与多边形的位置关系，无法判断的，再用求交法判断；

光线投射算法

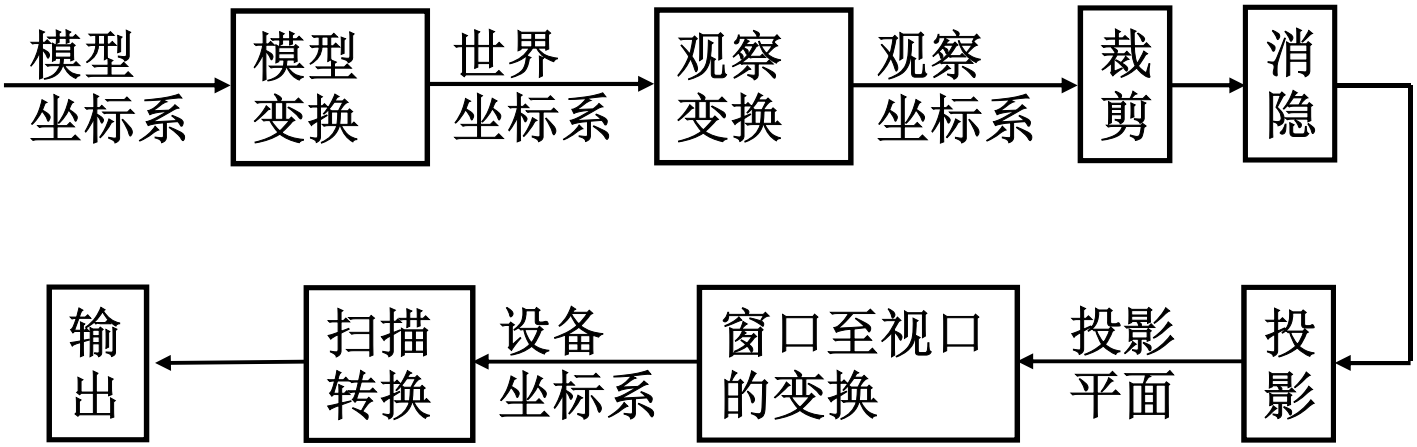


- 基本思想：
  - 考察由视点出发穿过观察屏幕的一象素而射入场景的一条射线，则可确定出场景中与该射线相交的物体。
  - 在计算出光线与物体表面的交点之后，离象素最近的交点的所在面片的颜色为该象素的颜色；
  - 如果没有交点，说明没有多边形的投影覆盖此象素，用背景色显示它即可。

```
for(屏幕上的每一象素)
{   形成通过该屏幕象素(u,v)的射线;
    for (场景中的每个物体)
        将射线与该物体求交;
    if (存在交点)
        以最近的交点所属的颜色显示象素(u,v)
    else
        以背景色显示象素(u,v)
}
```

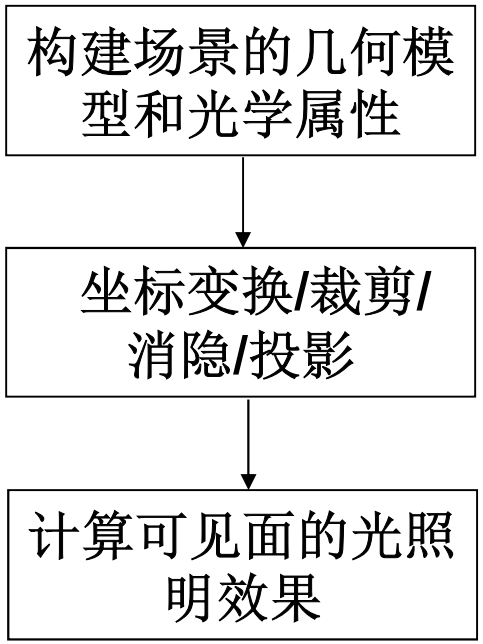
射线与三维形体求交

光栅图形学


模型
坐标系
模型
变换
世界
坐标系
观察
变换
观察
坐标系
裁
剪
消
隐
输
出
扫描
转换
设备
坐标系
窗口至视口
的变换
投影
平面
投
影

真实感图形学

- 颜色视觉
- 简单光照明模型
- 局部光照明模型
- 光透射模型
- 纹理和阴影
- 整体光照明模型



第八章　颜色视觉

清华大学软件学院　张慧　2019-10-9

基本概念

- 颜色是外来的光刺激作用于人的视觉器官而产生的主观感觉；
- 影响的因素有：
  - 物体本身
  - 光源
  - 周围环境
  - 观察者的视觉系统

颜色的特性

- 颜色的三个视觉特性(心理学度量)
  - 色调(**Hue**)：一种颜色区别于其他颜色的因素，如：红、绿、蓝
  - 饱和度(**Saturation**) ：颜色的纯度
  - 亮度(**Lightness**)：光的强度，光给人的刺激的强度

- 对应的颜色物理特性
  - 主波长(**Dominant Wavelength**)

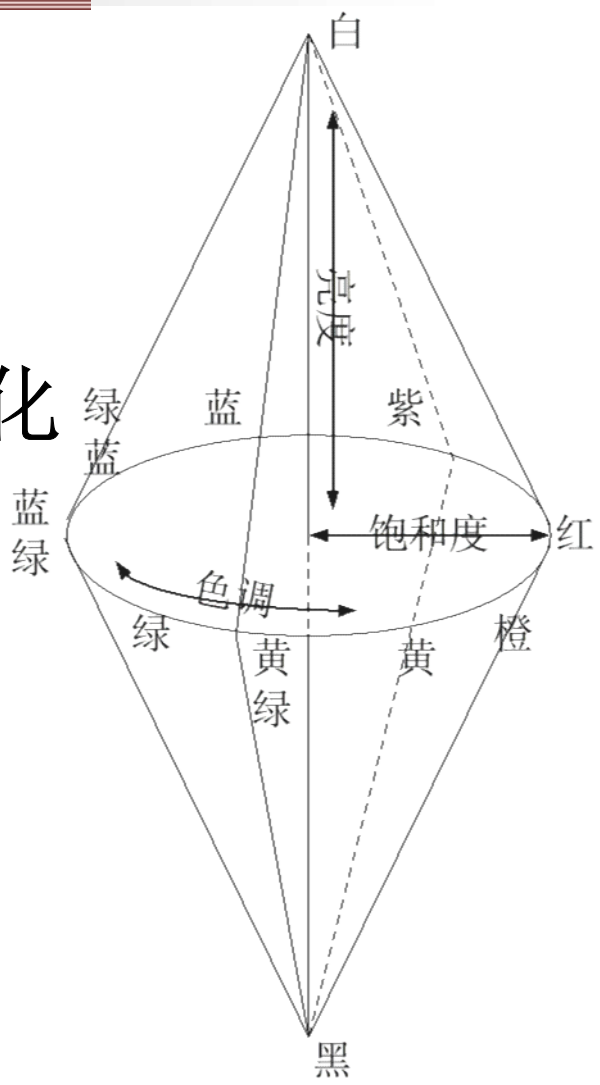
    产生颜色光的波长，对应于视觉感知的色调
  - 纯度(**Purity**)

    对应于饱和度
  - 明度(**Luminance**)

    对应于光的亮度

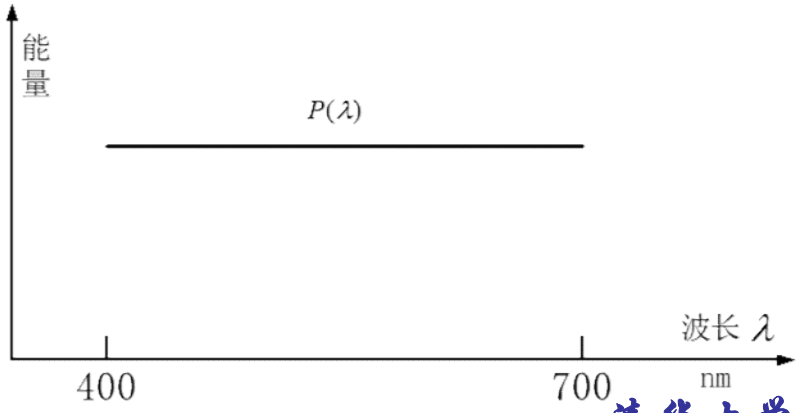
颜色纺锤体

- 颜色三特性的空间表示
- 垂直轴线表示白黑亮度变化
- 水平圆周上的不同角度点代表了不同色调的颜色
- 从圆心向圆周过渡表示同一色调下饱和度的提高

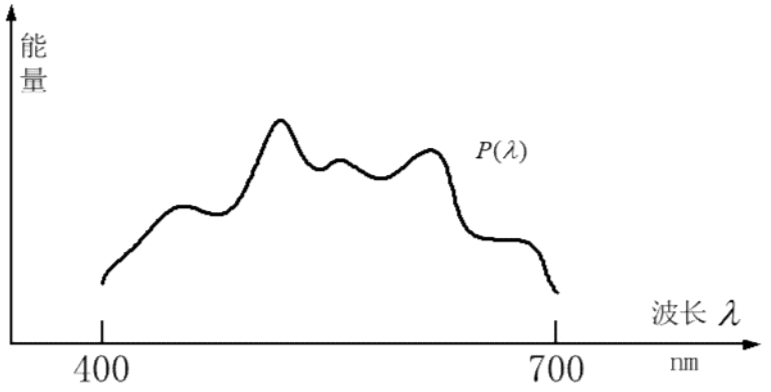


光的物理知识

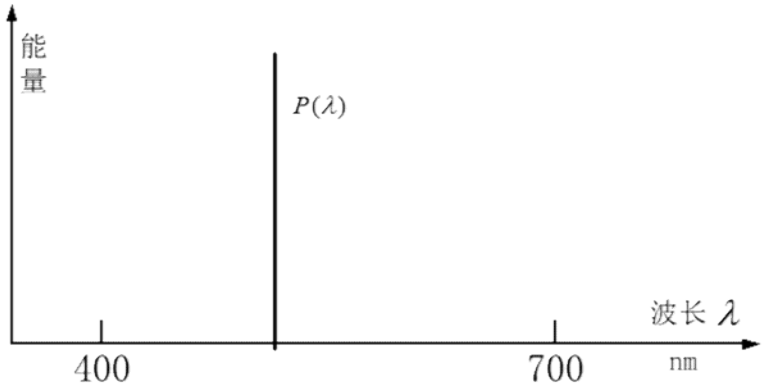
- 光是人的视觉系统能够感知到的电磁波
  - 波长在**400nm**到**700nm**之间**(1nm=10^{-9}m)**
- 光可以由它的光谱能量分布来表示
  - 光的各种波长的能量大致相等，为白光



- 光的各波长的能量分布不均匀，为彩色光



- 光只包含一种波长的能量，其他波长都为零，是单色光



- 光谱能量分布定义颜色十分麻烦
- 光谱与颜色的对应关系是多对一
  - 两种光的光谱能量分布不同而颜色相同的现象称为“异谱同色”
- 必须采用其他的定义颜色的方法，使光本身与颜色一一对应

三色学说

- 颜色具有恒常性，颜色之间的对比效应能使人区分不同颜色
- 颜色具有混合性，牛顿在十七世纪后期用棱镜把太阳光分散成光谱上的颜色光带，证明白光由很多颜色光混合而成

- 十九世纪初，**Yaung**提出某种波长的光可以通过三种不同波长的光混合而复现出来的假说
  - 红**(R)**、绿**(G)**、蓝**(B)**三原色
  - 把三种原色按照不同的比例混合就能准确地复现其他任何波长的光
  - 三原色等量混合产生白光
- **Maxwell**用旋转圆盘所做的颜色混合实验证实了**Yaung** 假设。

- **1862**年，**Helmhotz**在上面的基础上提出颜色视觉机制学说，即三色学说，也称为三刺激理论。
- 近代三色学说
  - 视网膜中存在三种椎状视觉细胞，对光刺激的兴奋程度不同，分别感受红、绿、蓝光。
  - 作用与颜色混合相同：黄光刺激眼睛
- 三色学说是真实感图形学中**RGB**颜色模型的理论基础

CIE色度图

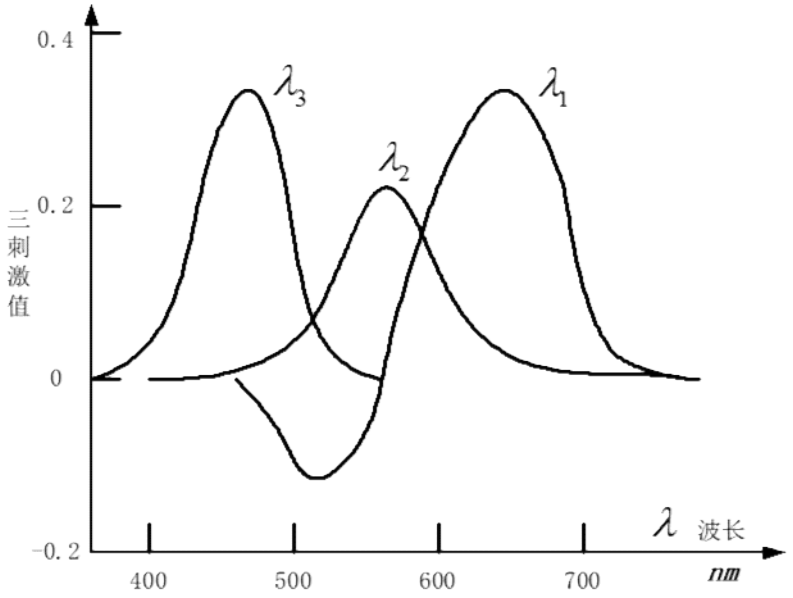
- 三色学说说明任何一种颜色可以用红、绿、蓝三原色按照不同比例混合得到。
- 如何使三原色按某唯一比例混合复现给定颜色的问题？
- 颜色匹配：混合光与给定光的颜色相同
- **CIE**－国际照明委员会
  - 选取标准红、绿、蓝三种光 **(700nm, 546nm, 435.8nm)**

CIE-RGB系统

- 光的颜色匹配式子：**c=rR+gG+bB**
  - 权值**r**、**g**、**b**为颜色匹配中所需要的**R**、**G**、**B**三色光的相对量
- **RGB**三原色匹配

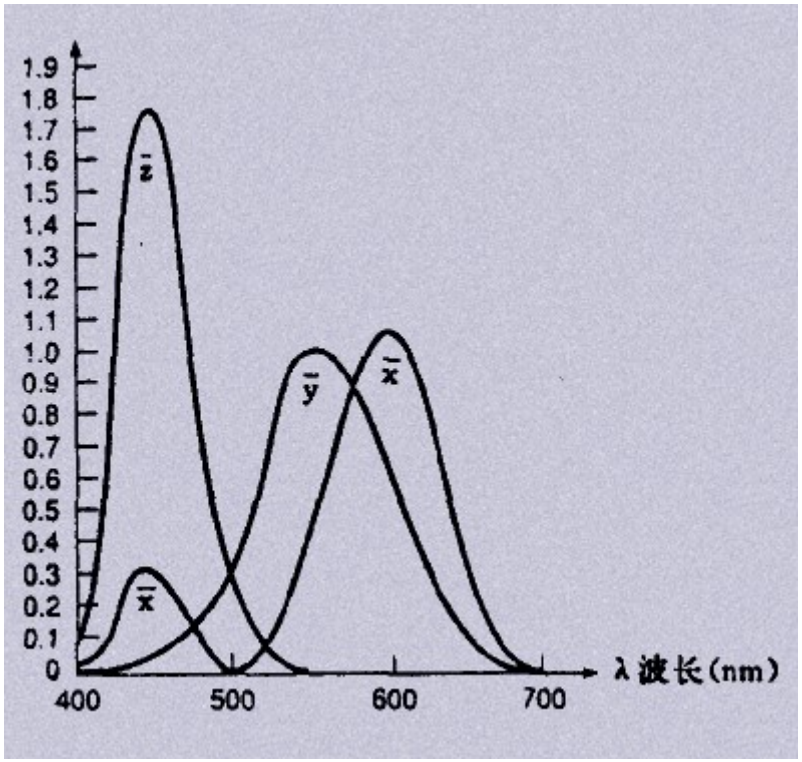
  任意颜色的光谱

  三刺激值线
- 曲线一部分三刺激值是负数



CIE-XYZ系统

- **1931年CIE，利用三种假想的标准原色X、Y、Z，使颜色匹配三刺激值都是正值：**
  c=xX+yY+zZ
- 任何颜色都能由标准三原色混合匹配

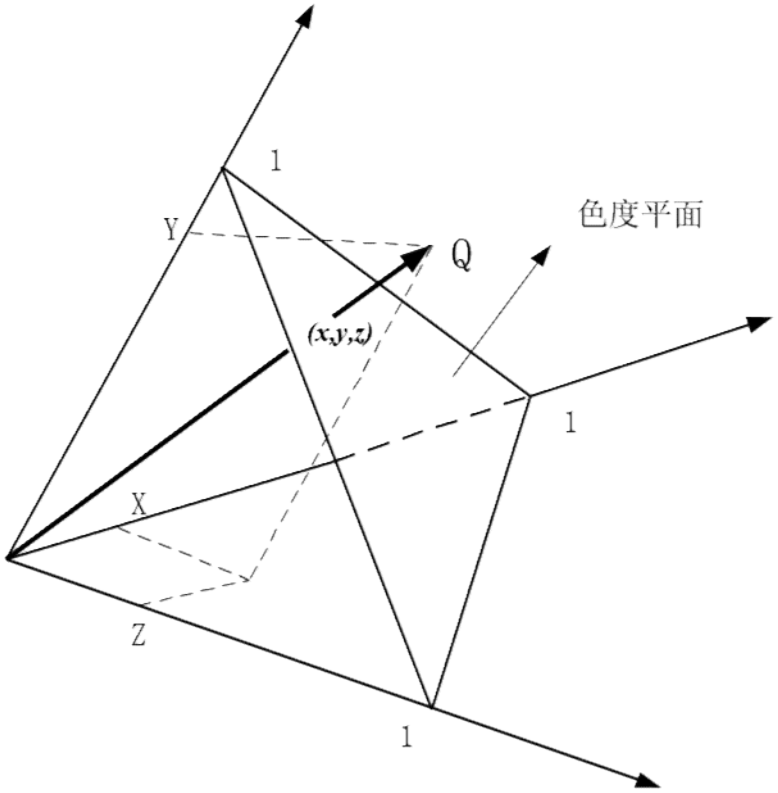


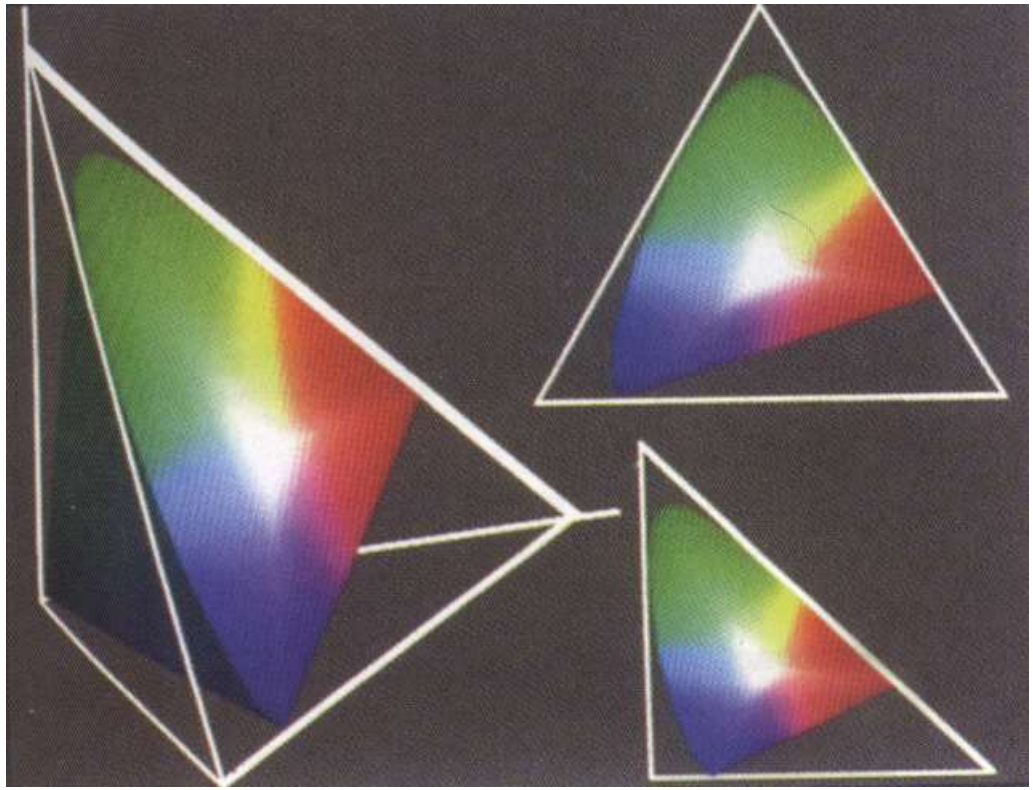
色度图

- 复现颜色的三原色比例值是否唯一？
- 三刺激空间：用三原色的单位向量定义三维颜色空间
- 颜色刺激表示为三刺激空间中以原点为起点的向量，向量的方向代表颜色
- 色度平面
- 色度图
  - 唯一地表示三刺激空间中所有颜色值

- 色度图上每一个点代表不同的颜色，对于三刺激空间中坐标为**X**、**Y**、**Z**的颜色刺激向量**Q**，它与色度平面交点的坐标**(*x*, *y*, *z*)**即三刺激值也被称为色度值

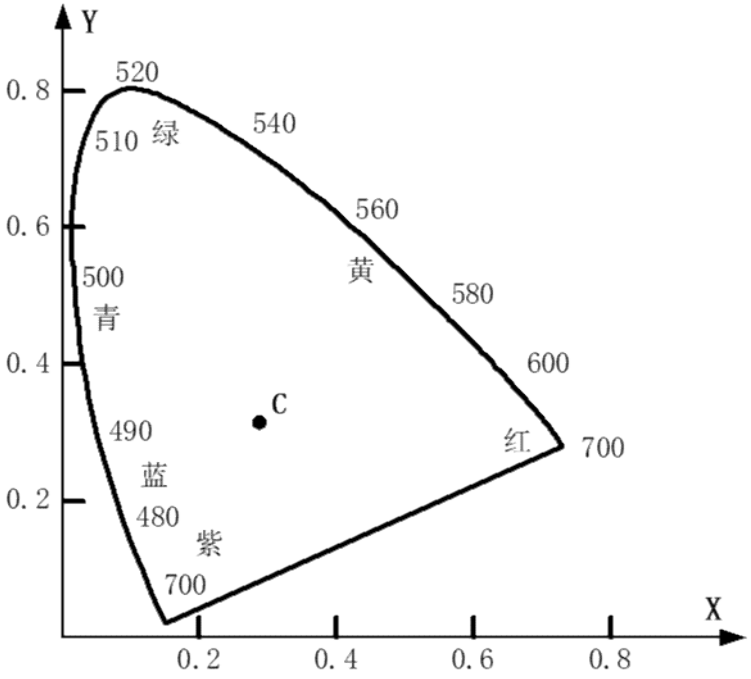
$$x=\frac{X}{X+Y+Z},y=\frac{Y}{X+Y+Z},z=\frac{Z}{X+Y+Z}$$


1
Y
色度平面
Q
(x,y,z)
1
X
Z
1



CIE色度图

- **CIE色度图**—色度图投影到**XY**平面上
- 马蹄形区域的边界和内部代表了所有可见光的色度值
- 边界弯曲部分代表了光谱的某种纯度为百分之百的色光

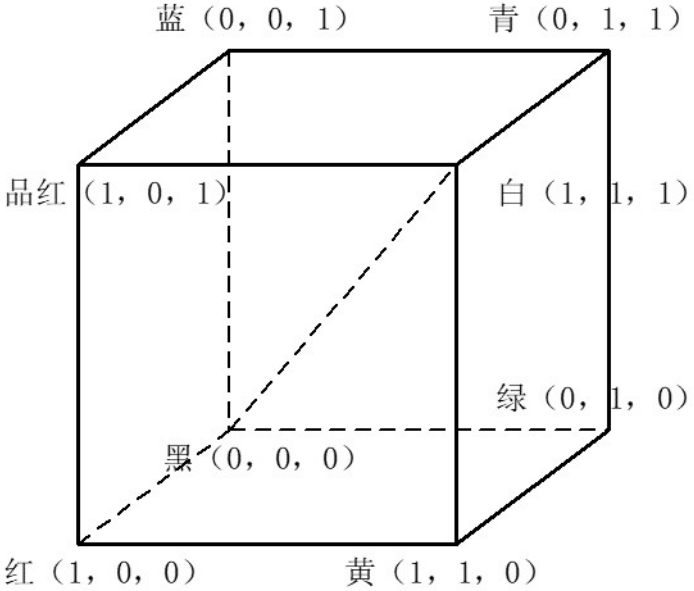


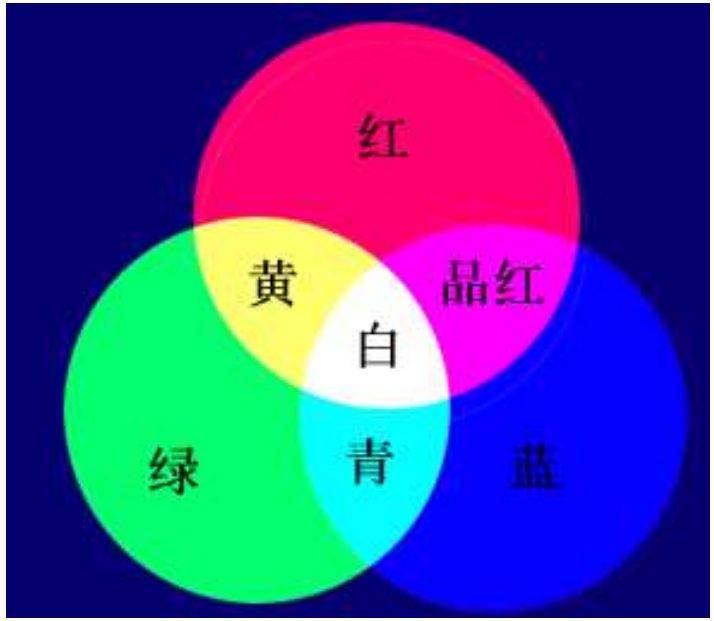
常用颜色模型

- 颜色模型是指某个三维颜色空间中的一个可见光子集，包含某个颜色域的所有颜色
- 颜色模型的用途是在某个颜色域内方便地指定颜色
- **RGB**颜色模型
- **CMY**颜色模型
- **HSV**颜色模型

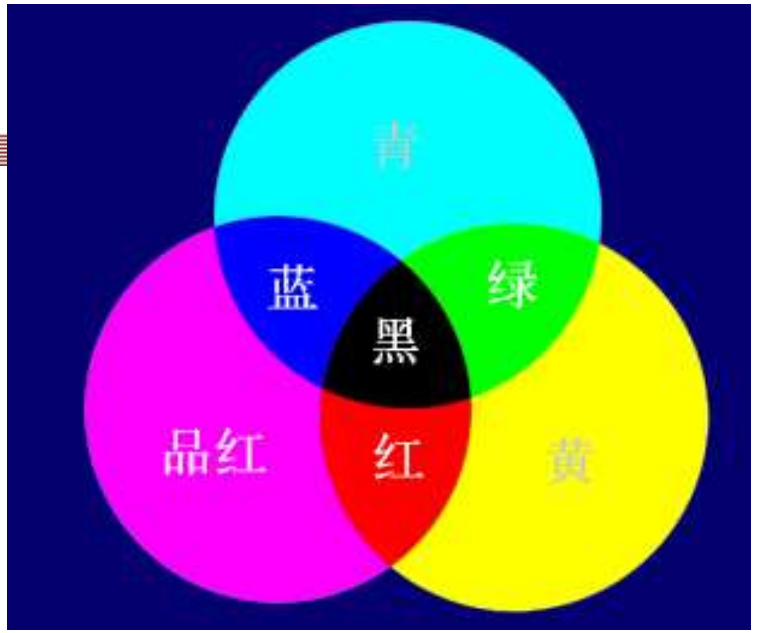
RGB颜色模型

- 通常用于彩色光栅图形显示设备中
- 真实感图形学中的主要的颜色模型
- 采用三维直角坐标系
- **RGB**立方体
- 三原色混合可产生复合色





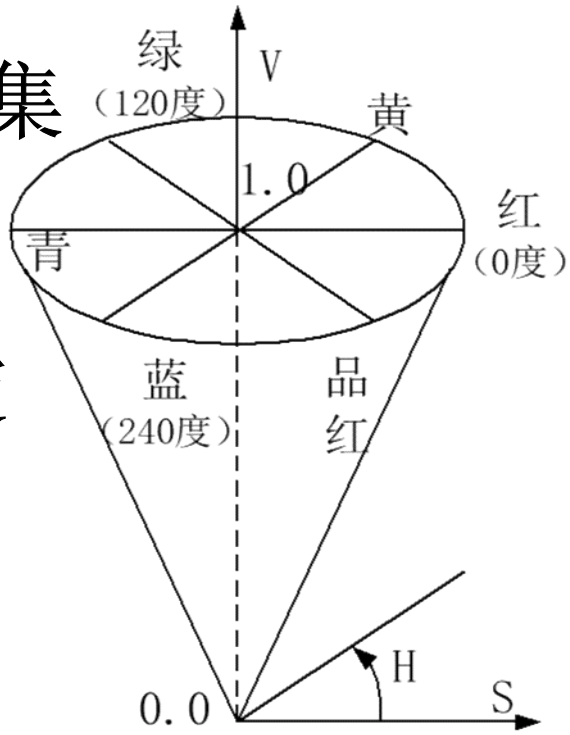
CMY颜色模型



- 以红、绿、蓝的补色Cyan(青)、Magenta(品红)、Yellow(黄)为原色构成的颜色模型
- 常用于从白光中滤去某种颜色，又称为减性原色系统，用在白色中减去某种颜色来定义一种颜色；
- 用于印刷行业硬拷贝设备的颜色处理。

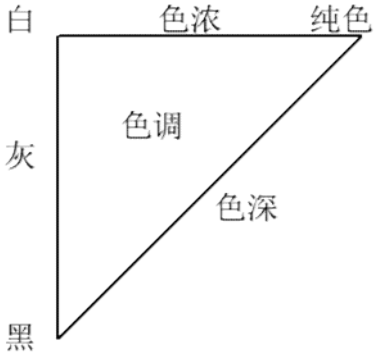
HSV颜色模型

- **HSV**(Hue, Saturation, Value)颜色模型是面向用户的
- 对应圆柱坐标系的圆锥形子集
- 圆锥的顶面对应于**V=1**
- 色调**H**由绕**V**轴的旋转角给定
- 饱和度**S**取值从**0**到**1**，由圆心向圆周过渡

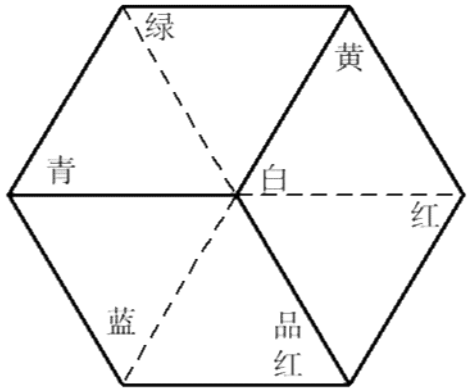


画家配色方法

- **HSV**模型对应画家的配色的方法
- 在一种纯色中加入白色以改变色浓，加入黑色以改变色深。同时加入不同比例的白色、黑色，即得到不同色调的颜色



RGB模型与HSV模型



- **RGB**立方体从白色顶点沿着主对角线向原点方向投影，可以得到一个正六边形，该六边形是**HSV**圆锥顶面的一个真子集
- **RGB**空间的主对角线，对应于**HSV**空间的**V**轴