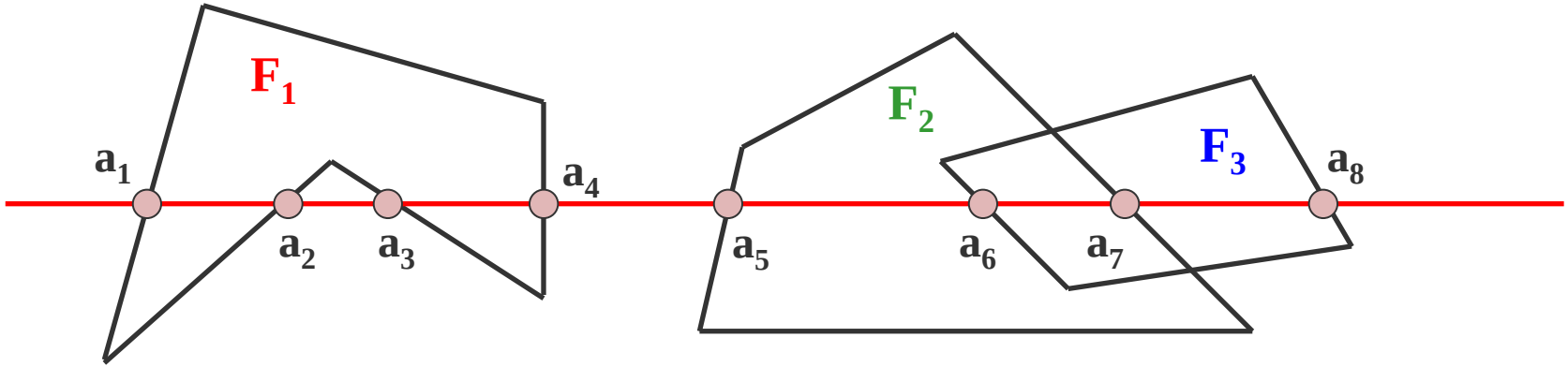
二、区间扫描线算法

前面介绍了经典的z-buffer算法，思想是开一个和帧缓存一样大小的存储空间，利用空间上的牺牲换取算法上的简洁

还介绍了只开一个缓存变量的z-buffer算法，是把问题转化成判别点在多边形内，通过把空间多边形投影到屏幕上，判别该像素是否在多边形内

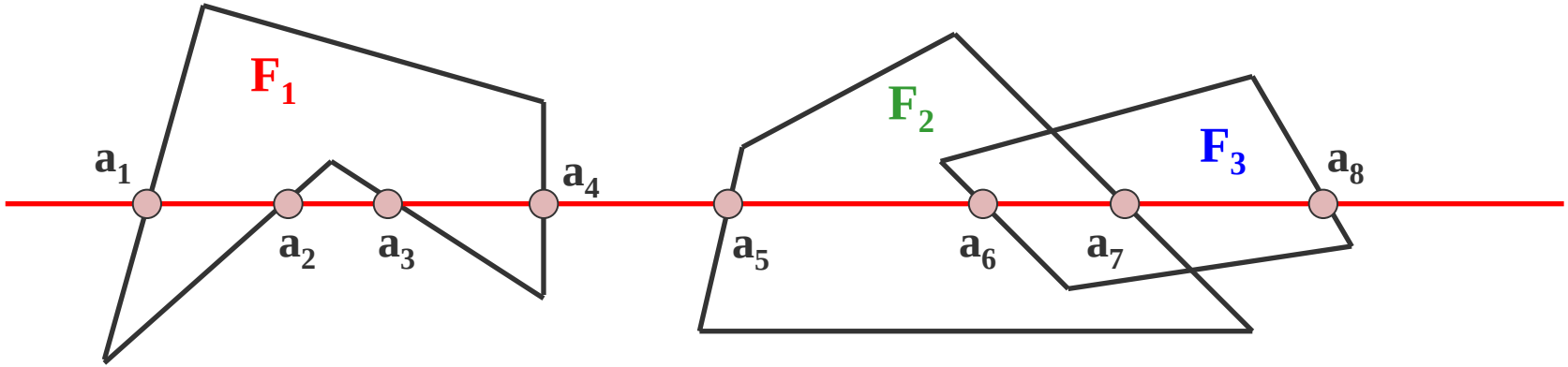
下面介绍区间扫描线算法。该算法放弃了z-buffer的思想，是一个新的算法，这个算法被认为是消隐算法中最快的

因为不管是哪一种z-buffer算法，都是在像素级上处理问题，要进行消隐，每个像素都要进行计算判别，甚至一个像素要进行多次（一个像素可能会被多个多边形覆盖）



扫描线的交点把这条扫描线分成了若干个区间，每个区间上必然是同样一种颜色

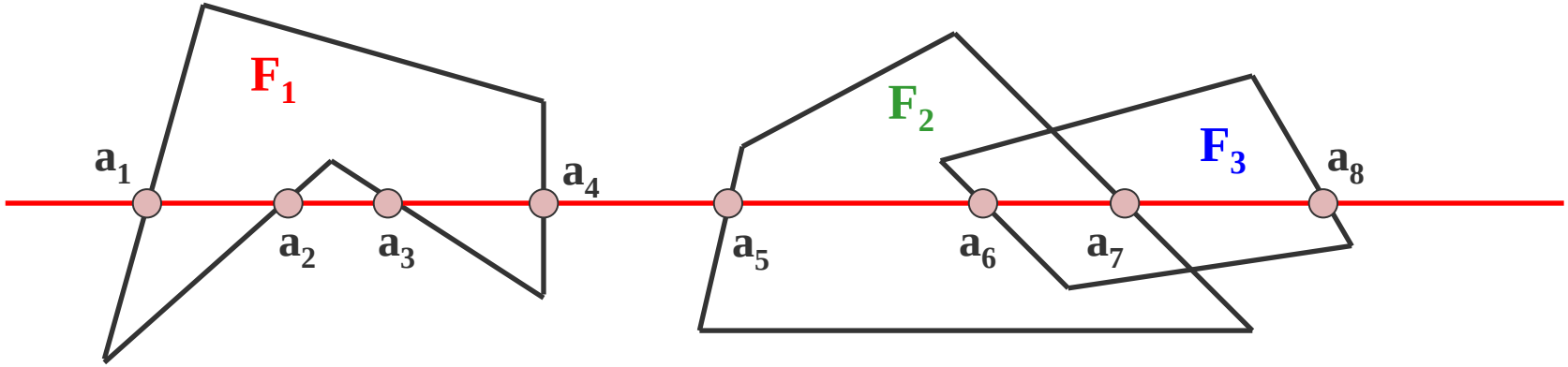
对于有重合的区间，如a6a7这个区间，要么显示F2的颜色，要么显示F3的颜色，不会出现颜色的跳跃



如果把扫描线和多边形的这些交点都求出来，对每个区间，只要判断一个像素的要画什么颜色，那么整个区间的颜色都解决了，这就是区间扫描线算法的主要思想

算法的优点：将象素计算改为逐段计算，效率大大提高！

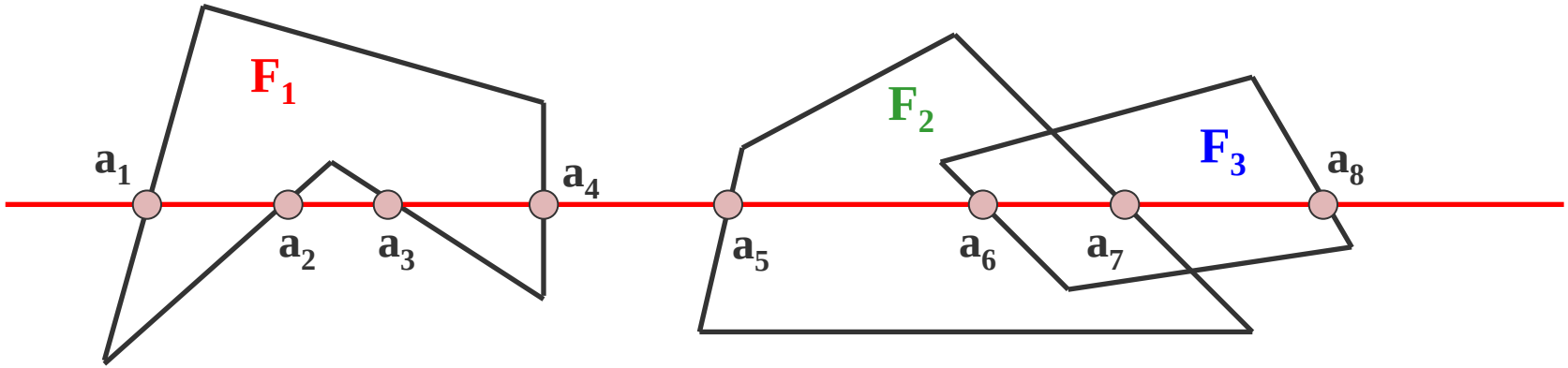
如何实现这个算法？



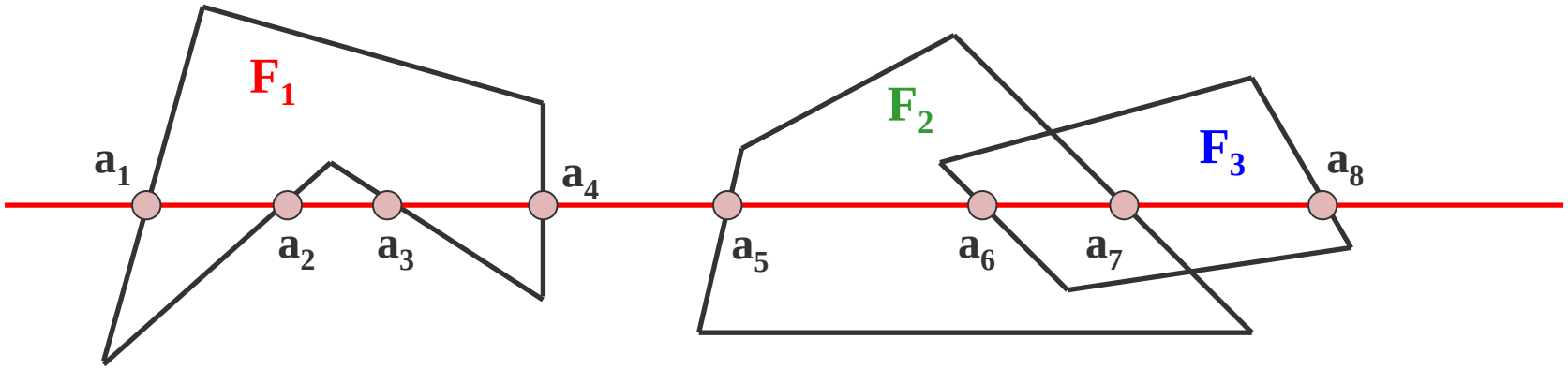
首先要有投影多边形，然后求交点，然后交点进行排序排序

排序的结果就分成了一个个区间，然后在每个区间找当中的一个像素（i，j），在（i，j）处计算每个相关面的z值，对相关深度值z进行比较，其中最大的一个就表示是可见的。整个这段区间就画这个z值最大面的颜色

如何确定小区间的颜色?



（1）小区间上没有任何多边形，如[a4, a5]，用背景色显示



（2）小区间只有一个多边形，如[a1, a2]，显示该多边形的颜色

（3）小区间上存在两个或两个以上的多边，比如[a6, a7]， 必须通过深度测试判断哪个多边形可见

这个算法存在几个问题：

1、真的去求交点吗？ 利用增量算法简化求交！

2、每段区间上要求z值最大的面，这就存在一个问题。如何知道在这个区间上有哪些多边形是和这个区间相关的？