三角剖分

在计算平面上可见多边形的整个算法流程当中，三角剖分起到了一个非常重要的地位。它为最终线性集合算法生成可见多边形提供了重要的边的拓扑排序信息。为了实现线上时间复杂度为O(n)的平面上可见多边形的生成算法，三角剖分部分的算法被分成了预处理阶段（Pre-process Phase）和执行阶段（Process Phase）。其中预处理阶段使用时间复杂度为O(nlogn)的受约束Delaunay三角剖分生成算法，而处理阶段使用时间复杂度为O(n)的三角网格更新算法。

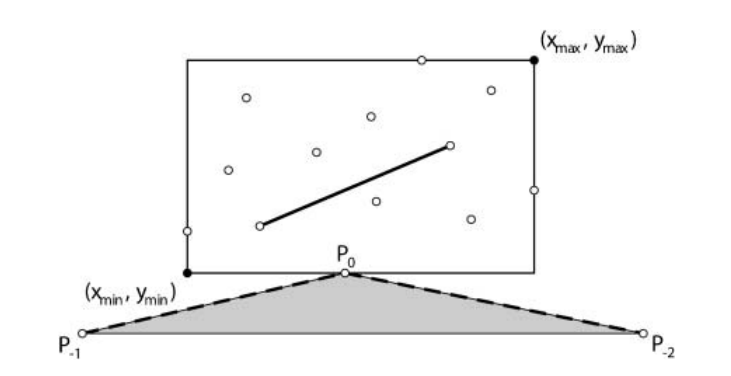
预处理阶段：

预处理阶段的受约束Delaunay算法需要完成的事情就是在O(nlogn)的时间复杂度之内，对平面上的多边形进行三角剖分，要求最终生成的三角剖分包含所有多边形上的每一条边，并且最终生成的三角剖分形成一个原有多边形的凸包（之所以需要生成凸包是为了方便后面执行阶段的时候向上方射出一条射线更新三角剖分时的操作）。

在程序的实现当中，这一块使用了开源的多边形受约束Delaunay三角剖分库poly2tri来完成。poly2tri的算法思路来源于V. Domiter 和B. Zalik在2008年3月19日在IJGIS上发表的论文Sweep-line algorithm for constrained Delaunay triangulation。这是一种基于扫描线的三角剖分算法，具体的算法如下所示：

初始化：

将所有点按Y坐标升序排列，如果Y坐标相同，则按X坐标升序排列。插入两个人工点P-1，P-2，使得他们位于所有点的下方，并且对于X坐标来说，P-1位于所有点的左侧，P-2位于所有点的右侧。这样我们就得到了构造的第一个三角形，P0P-1P-2。该三角形包含两条前沿边（Advancing Front），P-1P0和P0P-2。在最后一步最终确定的时候，所有与人工点相连的三角形将会被删去。这一步结束后，示意图如下所示：



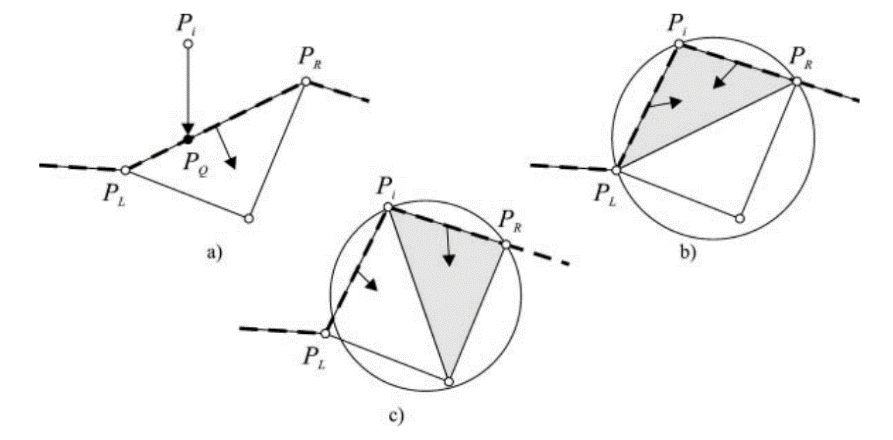
扫描：

对各个点按Y轴正方向进行扫描，扫描时分为这两种情况：

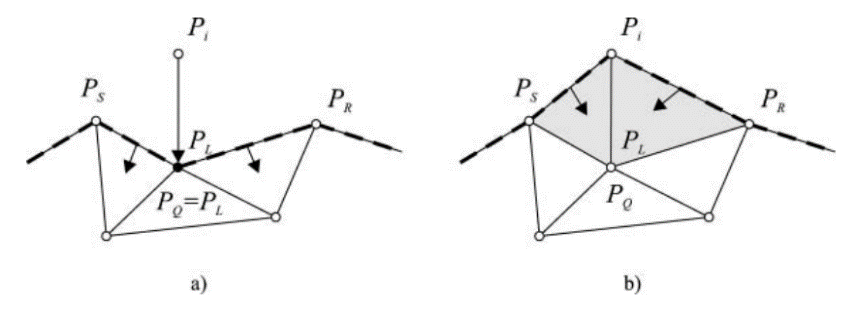
1. 某个顶点是孤立的点，或者是一个下边缘点时，顶点事件（Point Event）被触发。

设该点为Pi，该点到前沿边线段上的竖直方向投影记为PQ，前沿边线段的左右断点记录为PL,PR，PL在前沿边上的左边的临近线段断点PS。此时包含两种情况：

* 1. 点PQ在PLPR线段上，并不与端点重合。此时新的三角形PRPiPL将会被构造出来，新的前沿边将会被更新为PLPi和PiPR。如下图所示：

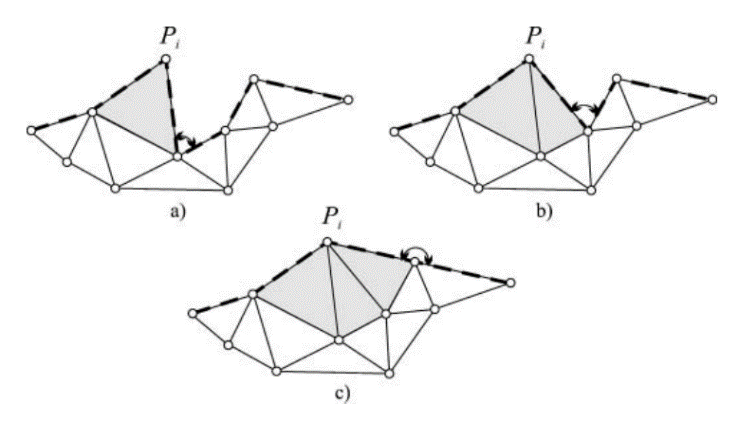


* 1. 点PQ与PL端点发生重合。此时将构成两个新三角形，PLPiPS和PLPRPi。如图所示：

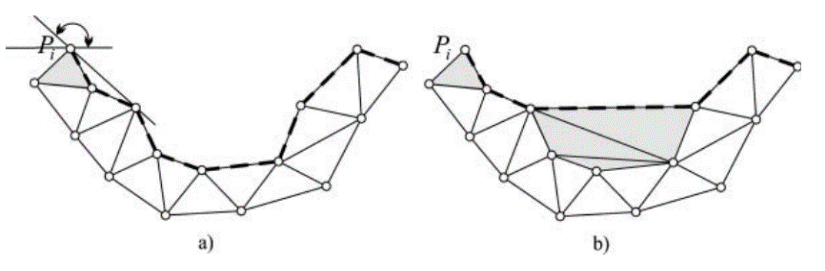


将前沿边上其他的对于Pi来说可见的点与Pi相连以构成新的三角形。在构成新的三角形的时候，需要进行必要的角度检查：

1. 新的三角形的边与前沿边线段之间的夹角必须小于90度，否则创建三角形行为将停止。如图所示：



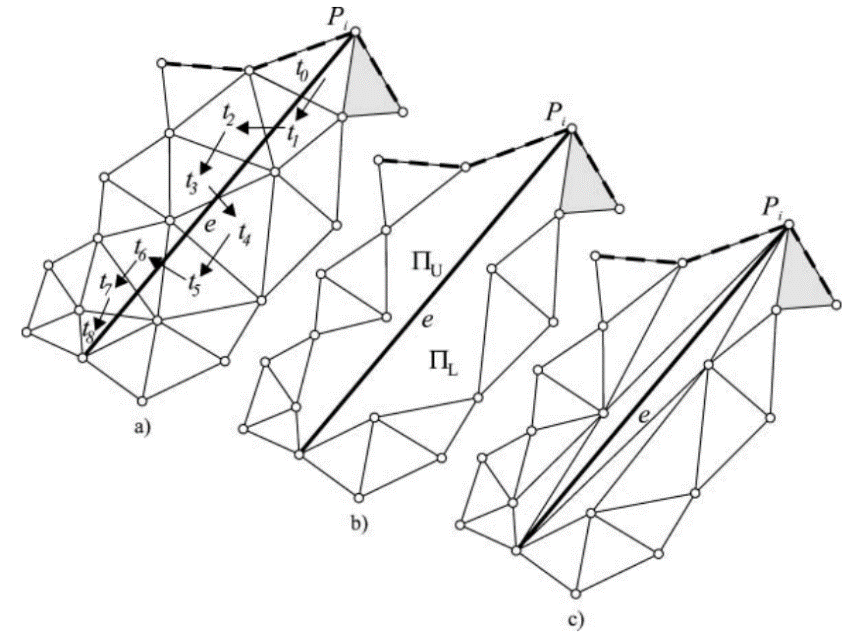
1. 前沿边上线段之间的夹角如果大于135度，说明存在盆状结构（Basins）。该盆状结构需要被更多新的三角形填充。如下图所示：



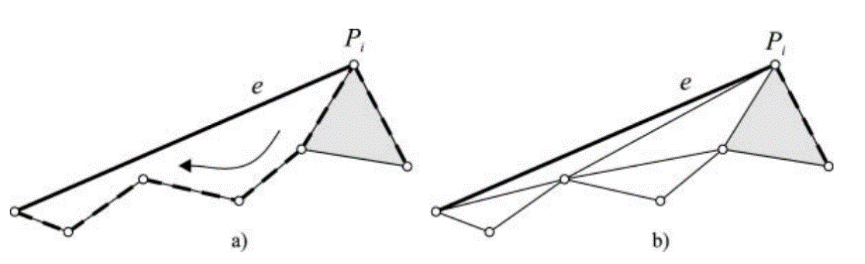
1. 某个顶点是一个上边缘点时，边事件（Edge Event）被触发。

当扫描遇到一个Y坐标较大的上边缘点时，一条受约束边e（Constrained Edge）必须被插入到三角剖分的结果当中。在插入这条边的时候，将会遇到该条边与目前结果当中的三角形相交的问题。对于这种情况，该算法使用这样的三个步骤进行处理：

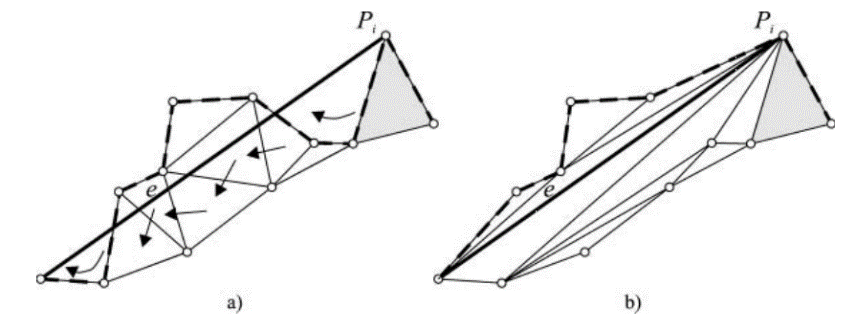
* 1. 确定第一个与e相交的三角形，并按照特定的遍历方案，将该条边e所串联的所有三角形标记为需要删除
  2. 删除所有被e相交的串联三角形
  3. 将被删除的空白区域重新进行三角剖分



这里将会面临三种情况。第一种情况就是e在前沿边线段的下方，这种情况照上述处理办法进行处理即可。第二种情况是e在前沿线段的上方。这是最简单的处理情况。由于没有任何相交的三角形，故不需要进行第一步遍历所有串联的三角形并标记删除的任务，只需要将e与上沿边构成的空白区域进行三角剖分即可。如下图所示：

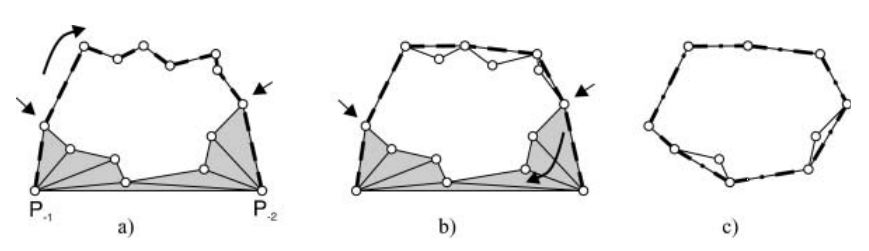


最复杂的是第三种情况，e与前沿边有一个或多个交点时。此时需要将遍历方法进行一些转换。当前沿边线段上的点在e的上方时，遍历方案将依据被串联的相交三角形进行遍历；当前沿边线段上的点在e的下方时，遍历方案将依据前沿边线段上的顶点进行遍历。



当所有顶点和约束边都被扫描后，进入最终确定环节：

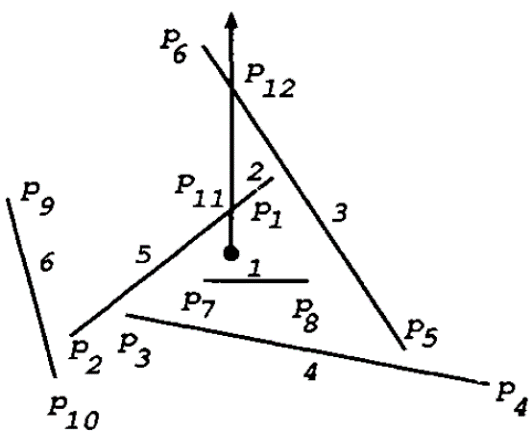
删除所有的与人工点相连的三角形，添加构成模型凸包的边框三角形到最终的结果当中。这样便完成了约束Delaunay三角化的扫描线算法。如下图所示：



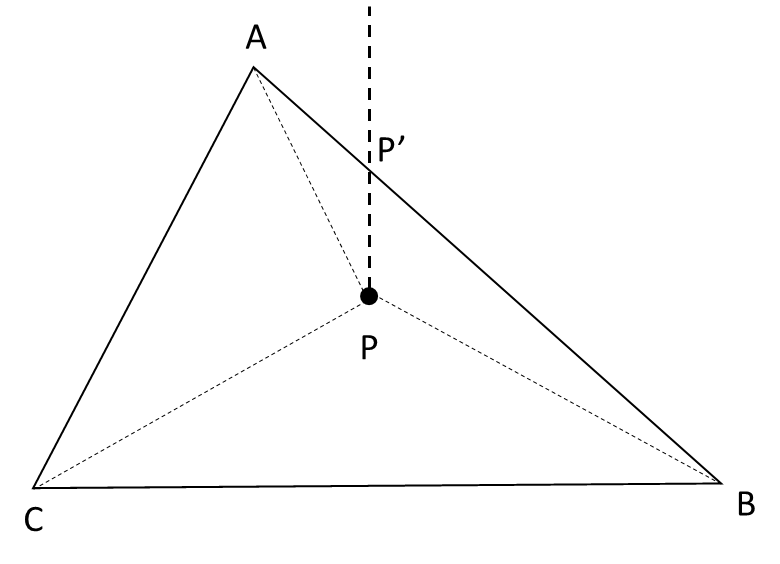
执行阶段：

预处理阶段结束以后，我们得到了一个多边形所在凸包的受约束Delaunay三角剖分。前面提到过，之所以使用多边形所在的凸包进行三角剖分的预处理，是为了执行阶段的时候射出一条射线对三角网格进行更新更加方便。在介绍执行阶段的三角网格更新之前，先对发射一条射线切分三角网格的目的加以解释。

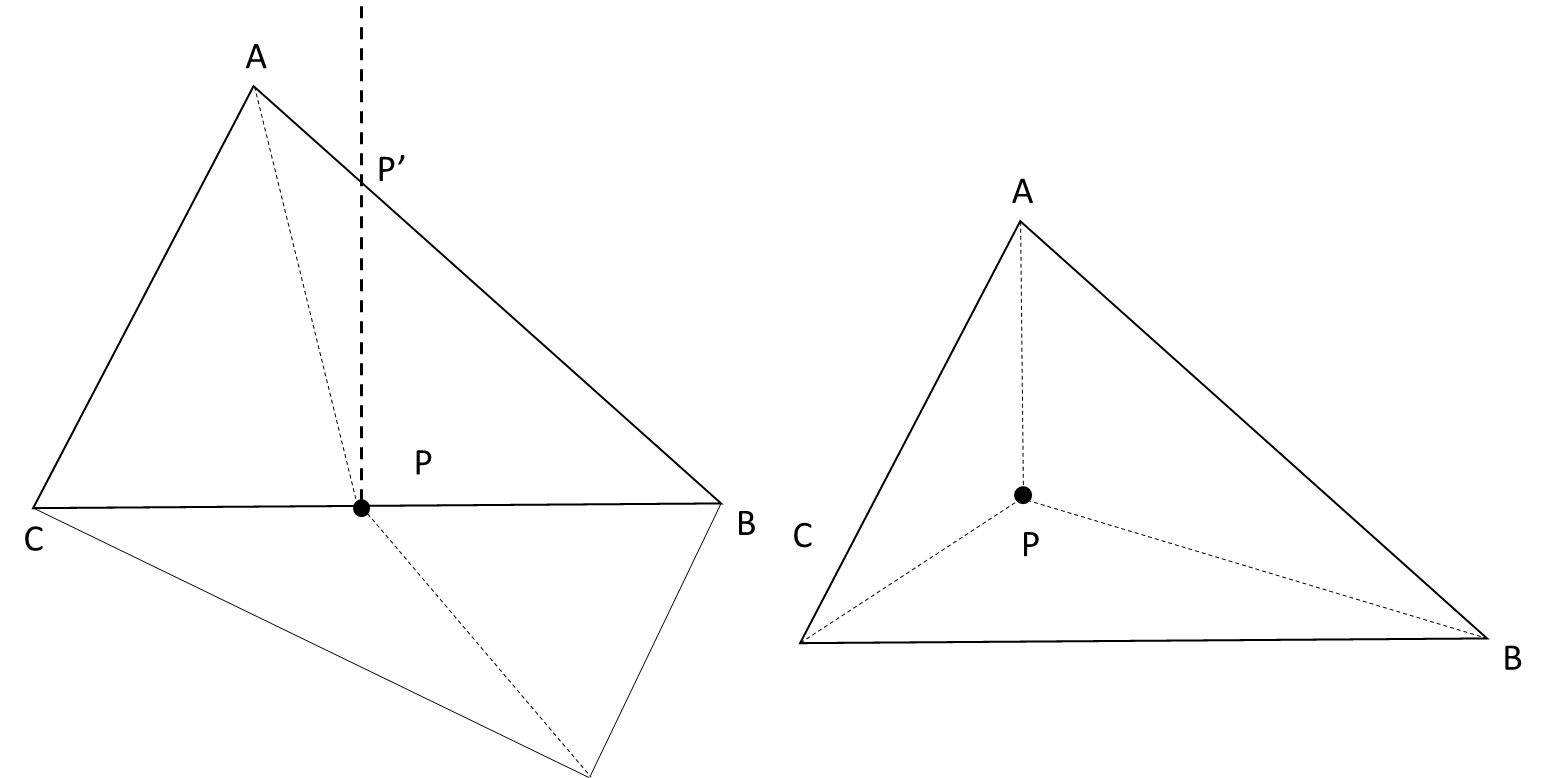
执行期间的三角剖分更新是为了给最后的线性集合算法提供多边形线段的拓扑顺序。而线性集合算法需要一个角度范围作为横坐标的输入，所以我们要把连续的360度空间给划分成一段从0度到360度的空间，为此，我们就需要射出一条直线作为0度和360度的划分线，并且把这条直线相交的所有线段分成两段。如下图所示：



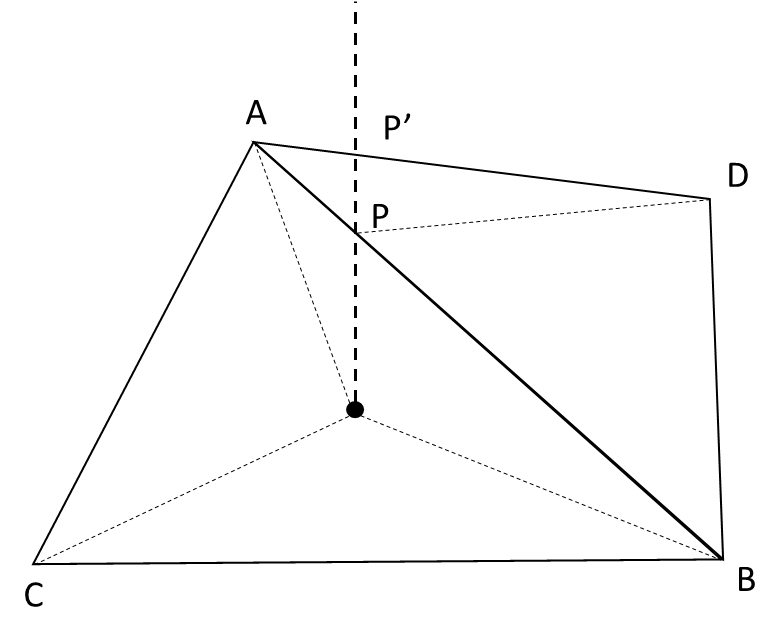
三角剖分更新的过程比较简单。首先查找到当前点P所在的三角形，设为ABC，而P向上垂直发射一条射线，交三角形ABC的AB边上，交点设为P’。这个时候，把三角形分为四个三角形，分别为APC，PBC，AP’P和P’BP。如下图所示：



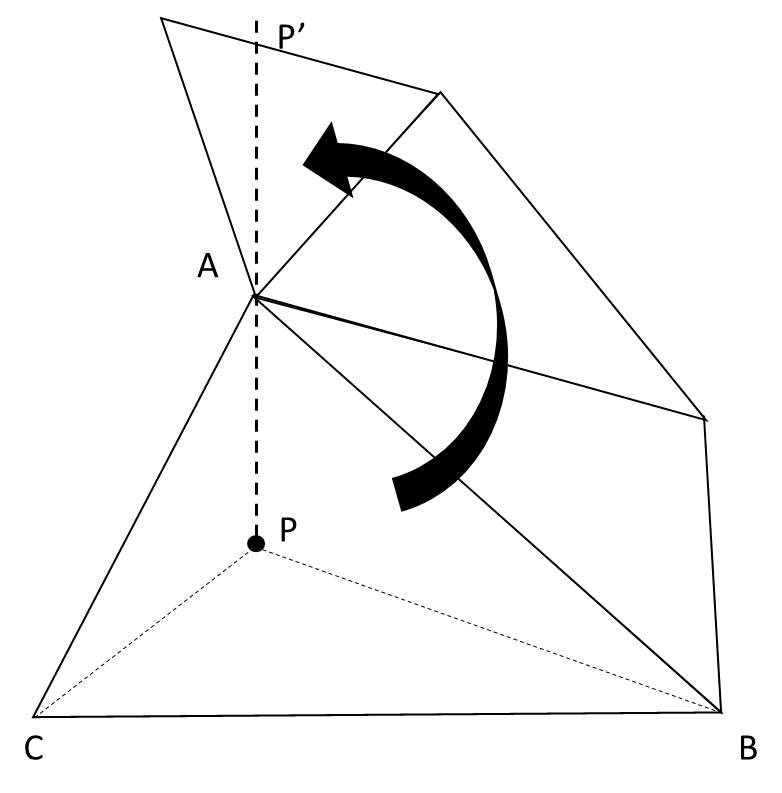
如果出现了P’与A重合的情况，则三角形AP’P退化为一条边；如果出现了P落在边BC上的情况，则三角形PBC退化成两条边，并且切分另一个与三角形ABC共边BC的三角形。如下图所示：



接下来只要循环处理，将P替换成P’，将三角形ABC替换成三角形ABC共边AB的三角形，然后继续按照上面的对三角形进行细分的方式对下一个三角形继续进行细分。如下图所示：

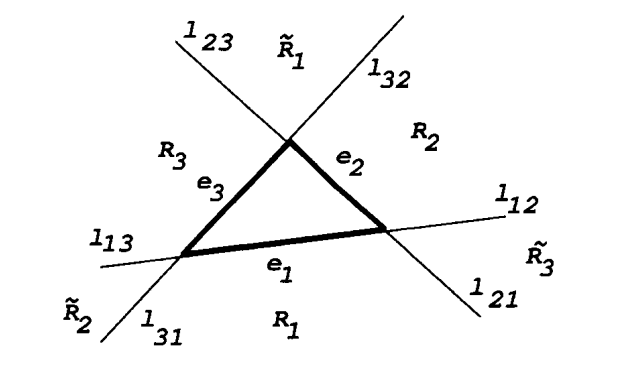


需要注意的是，当这条射线射到某个顶点上时，需要跳过很多个三角形知道射线射到最上的三角形的边界上，这种退化的情况如下图所示：



该循环将一直进行到射线射到一条不与任何三角形公共的边上后终止。所以对于多边形可能存在凹陷（Reflex）的情况，所以初始三角剖分要对该多边形的凸包进行，从而保证射线射到多边形的边界的时候，循环就可以终止。

执行阶段更新完了所有的三角形以后，将每一个三角形的每条边作为节点，建立一个图的偏序关系。建立偏序关系的方法如下图所示，首先每个三角网格当中的三角形都可以将空间划分成几个部分，如图中的R1，R2，R3，，，，l13，l31，l12，l21，l23，l32等



如果此时待求可见多边形的点位于R1的内部，那么就认为e2>e1,，e3>e1，R2，R3的情况类似；如果此时待求可见多边形的点位于的内部，那么就认为e2<e1,，e3<e1，，的情况类似；如果此时点位于l12上，那么就认为e2>e3。根据这种偏序关系建立一个有向无环图，最后对该图进行拓扑排序，得到最后的结果就是多边形所有边的拓扑顺序，为下一步的线性集合算法提供y坐标上的排列信息。