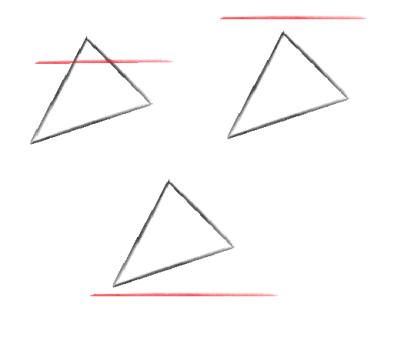
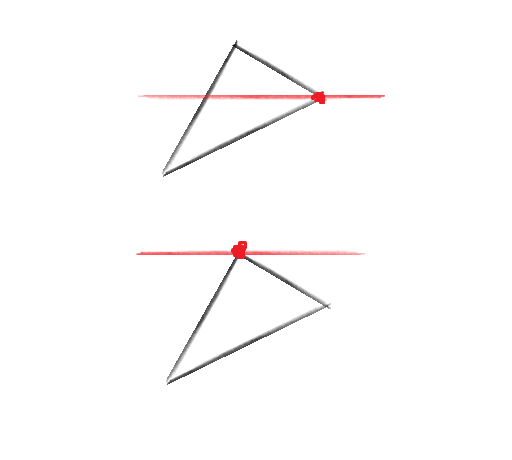
**\*\*\*\*\*\*\*\*\*————2015.7.30————\*\*\*\*\*\*\*\*\*\*\***

**NoiseUtSlicer**

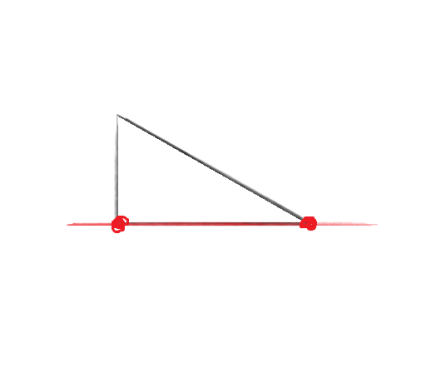
**3D打印机切层器**

1. (2015.7.30)加载STL
2. (2015.7.30)开始切层，每个水平面(y=??的平面)和每个三角形进行切割，一开始先按有多少个三角形顶点在当前切割面上进行分类讨论（因为生成线段的方式有点不一样）。一共有4类，就是有0、1、2、3个顶点在切割面上。反正经过判断切得顶点的个数之后，切得两个顶点的，就能生成一条单独的线段，加入Line\_Segment的buffer之后再处理。最特殊的，三角形和切割面共面，此时切得点有3个，这个三角形直接加入轮廓Buffer（想办法判断哪一边是实心的）（用LineStrip /轮廓 表示的区域）。
3. 

0个顶点在切层上，只能尝试用边去切割

1. 

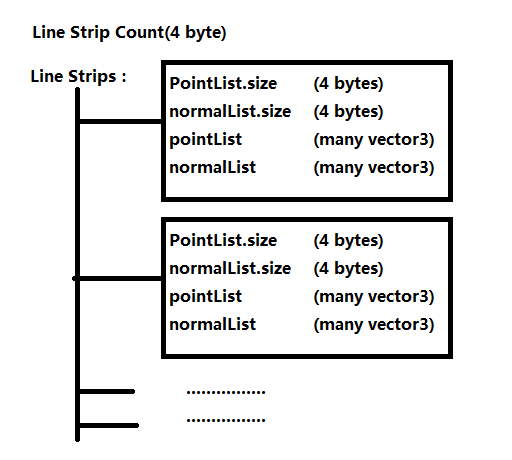
1个顶点在切层上，看看可不可能切出第二个顶点

1. 

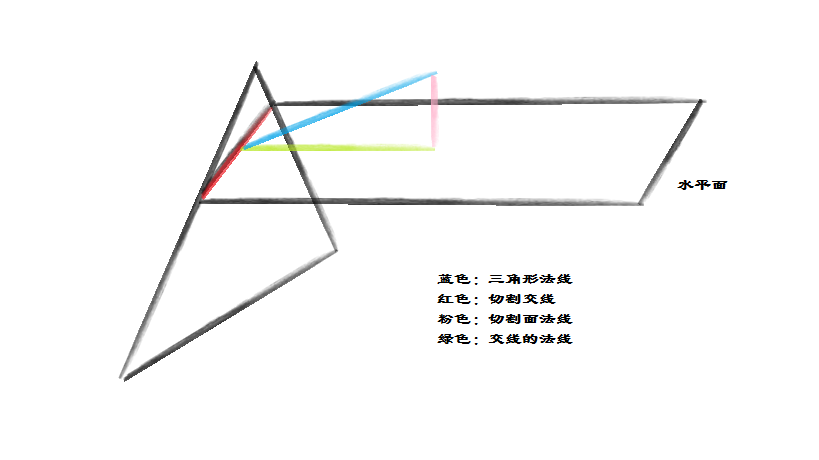
两个顶点在切割面上，直接就是想要的线段。

1. (2015.7.30)把Line\_Segment连起来，连成LineStrip。有个问题，可能到某个顶点时会有多个分支，可能要用到树（？）。再说再说
2. (2015.8.1)不如设定每条Line Strip要首尾相等（共点）才是有效闭合折线
3. 
4. (2015.8.2)好吧，如果STL文件的网格没有缺陷（完全封闭的话），那个Vertex On Layer = 3 的情况都简直可以去掉了，因为肯定有Line Segment会和那个三角形的边重合，这样搞下去的line strip会有重合的部分，麻烦死。而且，Vertex On Layer =3 意味着三角形法线竖直向上，这样的话，边法线也不好确定。所以Intersection的Case 3去掉。（尼玛我在讲什么= =）
5. (2015.8.2)因为LineStrip的顶点是一连串的，所以现在规定线段NormalList的元素个数要比PointList少一个，第几个normal对应第几条线段。
6. (2015.8.3)导出NOISELAYER文件：自定义切层格式。全文件第一个四字节是记录Line Strip的个数4 byte的"magic number"——把'k''A''s''T'塞到一个UINT32里面，然后4byte Version ID 暂时是0xffffff01，然后4 byte Line Strip个数。

然后之后是一段一段的Line Strip数据。对于每一段line strip，4 byte的int的层ID (layer ID)，然后4 byte int是PointList的point个数，再一个4 byte int是normalList的normal个数，是比point少一个，因为n个连续点有n-1条线段，也就是n-1条线段法线。之后就是PointList的12 byte 的vector3；之后再normalList的12 byte 的Vector3。（vector3是三个float,xyz分量）



（这个只是示意图，文件的区块逻辑大概是这样，但是新版本加了magicNumer和版本号和layerID）

1. 

(2015.8.2)推论：交线段法线是三角形法线在切割面上的投影。

证明：

∵蓝线⊥三角形

∴蓝线⊥红线

∵粉线⊥水平面

又∵红线水平面

∴ 粉线⊥红线

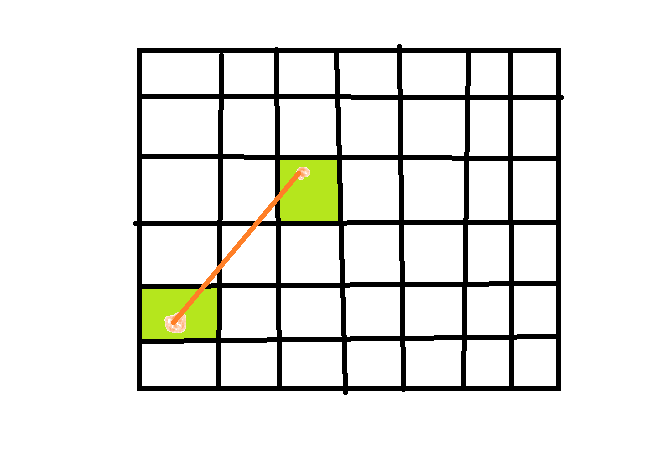
∵

∴红线⊥面<蓝绿粉>

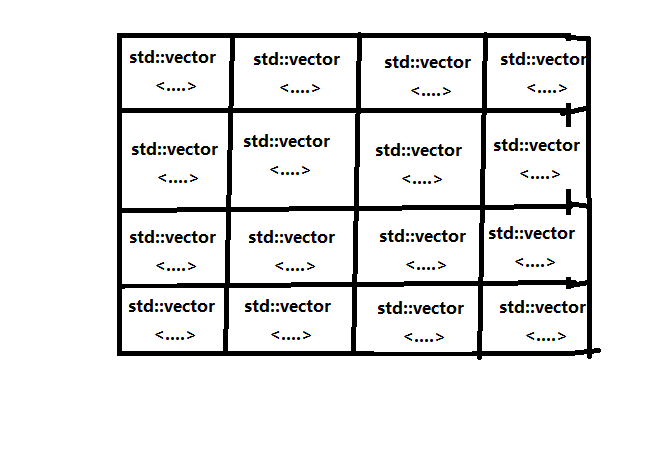
∴红线⊥绿线

绿线即线段法线，所以三角形法线在切割面上的投影就是线段法线

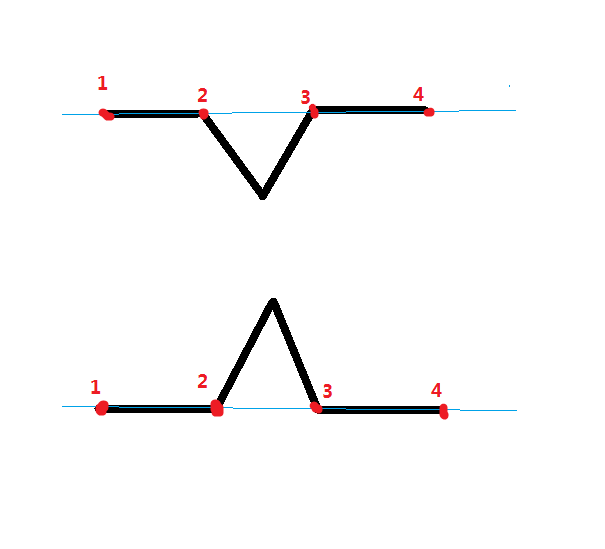
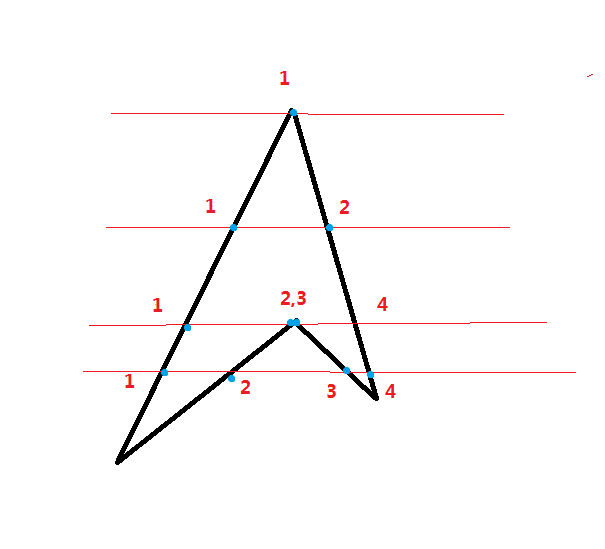
1. (2015.9.30)连接线段算法：对于每一条正在处理的line strip，查找所有dirty标记为false的线段，看看v1或者v2能不能焊接上line strip的尾部，如果可以就在linestrip的pointlist和normalist上增加东西，然后更新当前的tail point。不过这样非常慢，因为对于每一个line strip的tail point要查找能焊接的顶点基本都要遍历所有line segment,时间复杂度直逼O(n^2)，而且n一般非常大，这是不能接受的。所以现在做一定的预处理改进：新增Layer数据结构，layer被分成很多个tile（或者说是块），每个tile都有std::vector来装line segment的ID（相当于先预先计算好每条线段所属layer的所属tile）。只要有至少一个顶点在当前tile，这条线段就属于这个tile。这么分块是因为，有可能用于焊接到line strip末尾的顶点，必定在附近，所以遍历line segment只需要在tail point所在layer tile搜寻即可。记得焊接好顶点后，把新顶点所属line segment的信息从layer tile的列表中删去(或者点亮line segment的dirty标记），从此这条line segment不再属于这个tile。



如图：红色线段同时属于绿色的layer tile，如果line segment已经加入了line strip，那么它的信息要从两个绿色块中抹除。



1. (2017.1.17)现在做了个mineCraft 3d打印的项目发现，只有 没有邻接信息的Line Segment就想生成填充好的多边形不太实际，因为用扫描线填充算法时，可能一条线段和扫描线求交得两个顶点，这一行的填充就会对图形是凹形还是凸形产生了巨大的依赖，需要这个迷之的顶点的更多邻接信息才能做出求交结果的决策。



如上图，扫描线填充算法可能会对填充区域判断错误。