**Minecraft - 3D打印外挂**

**华南理工大学**

**软件工程卓越班(2015)**

**团队成员 : 练孙鸿**

* **项目简介**

本项目是一个有GUI的PC端软件，主要功能就是加载STL格式的3D模型，提供3D可视化的预览，然后计算模型切层与拓补信息之后，进行光栅化与填充区域计算，然后用模拟按键事件的形式把栅格化后(rasterized)的3D模型“打印”到Minecraft游戏场景里面。

核心模块与算法主要有：

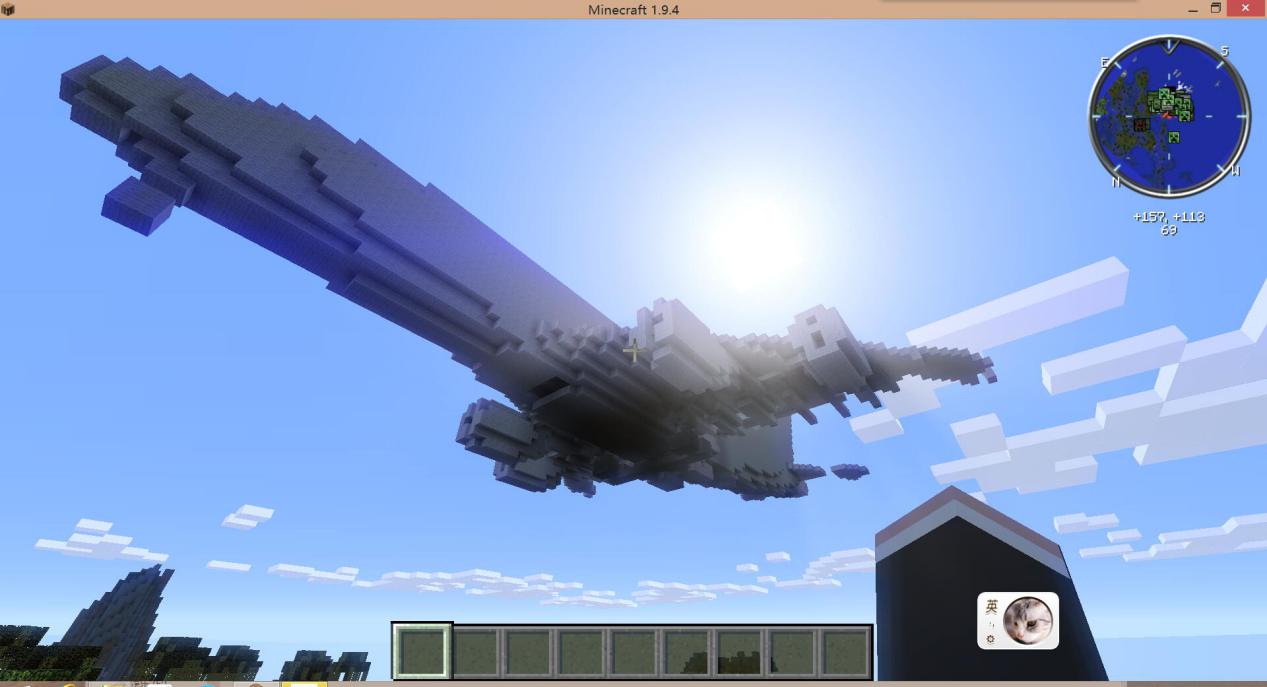
* 3D渲染与可视化(基于自研3D渲染引擎)
* 模型文件加载
* 3D模型切层器(3D打印机也要经过这个步骤，算法由本人设计并实现）
* 拓补信息重建与闭合多边形填充（算法由本人设计并实现）
* 按键模拟
* GUI

项目有一定的亮点，虽然这个软件在这个项目中用在了游戏建模上面，**但是核心算法是可以用在真正的3D打印机上**，颇为实用。而且由于算法是自研的，针对特定3D打印机的优化和适应是绝对可行的。

项目模块化实践得非常好，所有模块都经过了单元测试之后再进行系统集成，稳定性好。软件开发过程采用了软件工程中的迭代式开发，项目开发风险能更好地控制。

有一些最终效果图如下（游戏截图）：



****

****

更多的截图可以到”\演示视频与图片文件”文件夹里查看。

* **开发工具、平台、框架、语言**

**Visual Studio 2015(IDE)**

**C++(语言，用于写主逻辑与算法)**

**C#(语言，用于写GUI)**

**DirectX SDK（微软的一个多媒体底层开发框架）**

* **成员分工**

练孙鸿：项目构想、算法设计、系统架构设计、算法实现、测试用3D模型的建模。

* **各功能模块介绍与技术细节**
* **3D渲染与可视化(基于自研3D渲染引擎)**

此模块的开发是基于我的自研的C++ 3D渲染引擎Noise3D，目前在github上有10个star，代码规模约20000行。

* **模型文件加载**

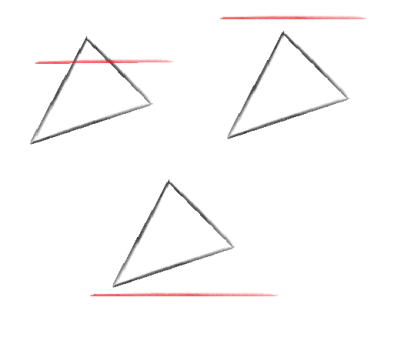
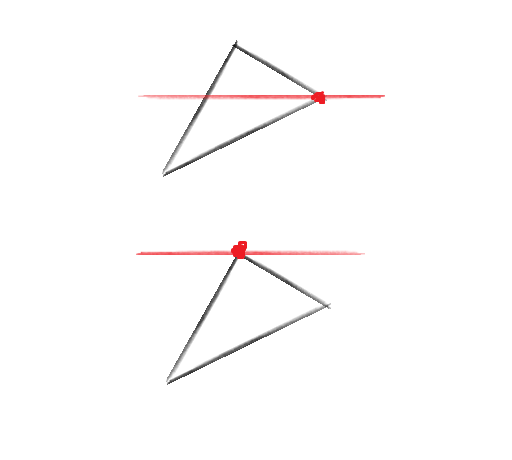
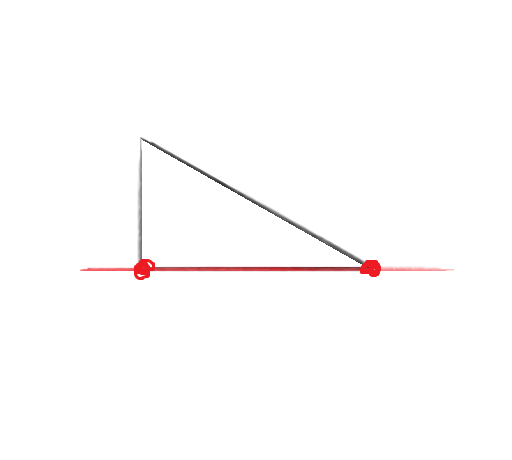
.STL是储存着三维网格模型的文件格式，因为没有太丰富的信息，所以STL格式比较 友好。STL文件分成两种储存形式，一个是用ASCII文本表示的可读性更好的**文本**，这 种格式占用空间更大。为了更高的效率，这个项目采用了**二进制**的STL储存形式。

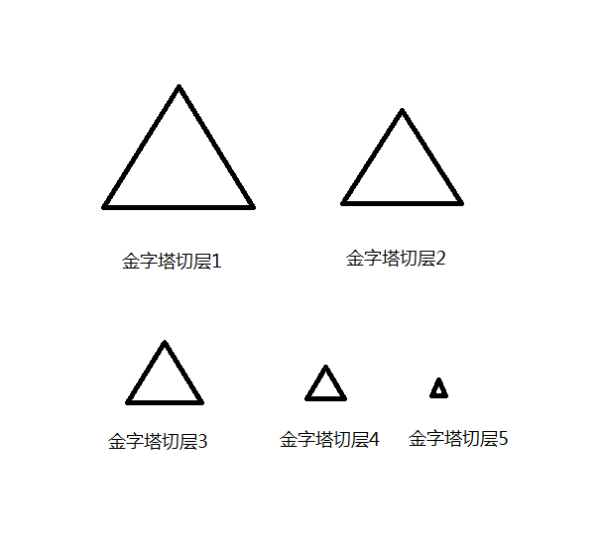
* + 二进制STL的构造：（百度百科）  
    二进制STL文件用固定的字节数来给出三角面片的几何信息  
    文件起始的80个字节是文件头，用于存贮文件名；  
    紧接着用 4 个字节的整数来描述模型的三角面片个数，  
    后面逐个给出每个三角面片的几何信息。每个三角面片占用固定的50个字节，依次是 :
  + 3个4字节浮点数(角面片的法矢量)  
    3个4字节浮点数(1顶点的坐标)  
    3个4字节浮点数(2顶点的坐标)  
      
    3个4字节浮点数(3顶点的坐标)  
    三角面片的最后2个字节用来描述三角面片的属性信息。  
    一个完整二进制STL文件的大小为（三角形面片数乘以 50）再加上84个字节。
* **3D模型切层器(3D打印机也要经过这个步骤，算法由本人设计并实现）**

既然3d打印的原理是“层层叠”，那就肯定需要横着一刀刀切成薄片，**并找出这些切片里属于模型内部的部分**。假设模型在 mc里的高度是50个方块，那么我们就去切50刀。从模型的最低水平面均匀切到模型的最高水平面。换句话说，就是用**50个不同Y坐标的、平行于XZ水平面的平面去和模型求交**。（是的，这样子的话，如果在面和面之间有曲率变化很大的地方，可能会被忽略掉，不过这种采样方式的问题我觉得还是先忽略掉吧=。=）。

由于模型本身是由一个个三角形构成的，所以问题可以简化为 **平面Y=y与三角形的求交问题**。

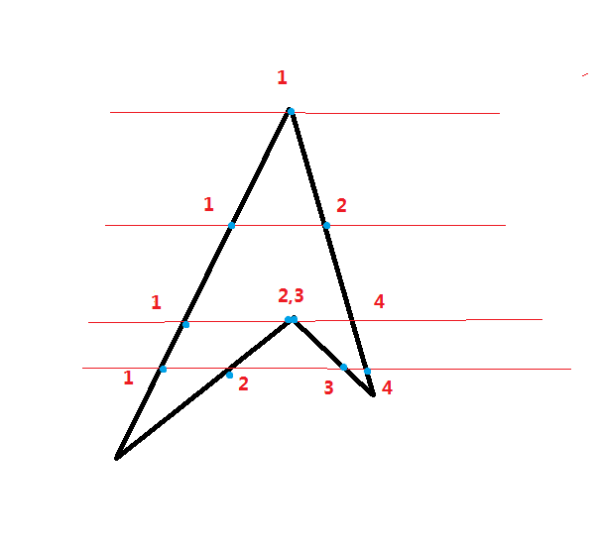
这个切割面和三角形的求交，还是要分类讨论一下，我按如下方式讨论：

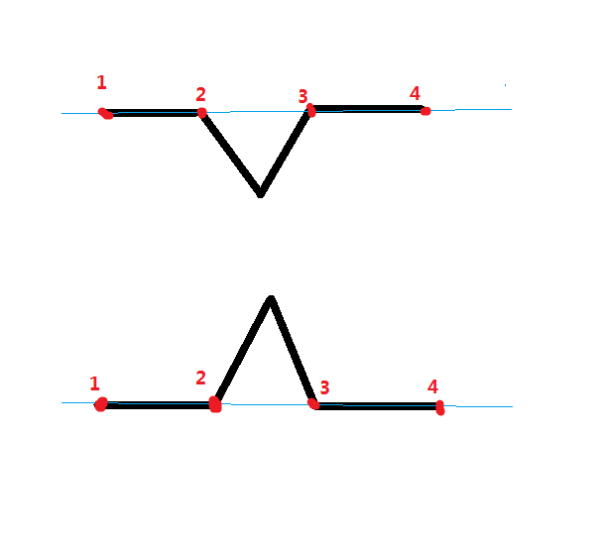
1. 切割面没有落在三角形任何一个顶点上，那就把问题转化为**两条线段分别和切割面求交**，求交得两个端点，这两个端点组成的线段添加到这一层的切割结果里。
2. 切割面落在一个三角形顶点上，那就只需要把**那个顶点对面的线段和切割面求交**，一个顶点一个端点组成的线段添加到这一层的切割结果里。
3. 切割面切到了两个顶点，直接把两个顶点组成的线段添加到这一层的切割结果里。
4. 切割面切到了三个顶点，即三角形就在切割面上，那就抛弃掉，爸爸不要你了，不要问我为什么（不解释太多，但是总的来说就是上面三种情况对于“求切割面切割出的闭合多边形”已经足够，再把这种情况考虑进来可能会产生重复而且多余的线段）

上图是金字塔（四面体）切5层的示意图。每一层都可能会有几个闭合多边形。

* **拓补信息重建与闭合多边形填充**

切层模块并不足以把模型量化/栅格化，接下来我们要把线段连成闭合多边形，然后找出这些闭合多边形的**内部区域**。

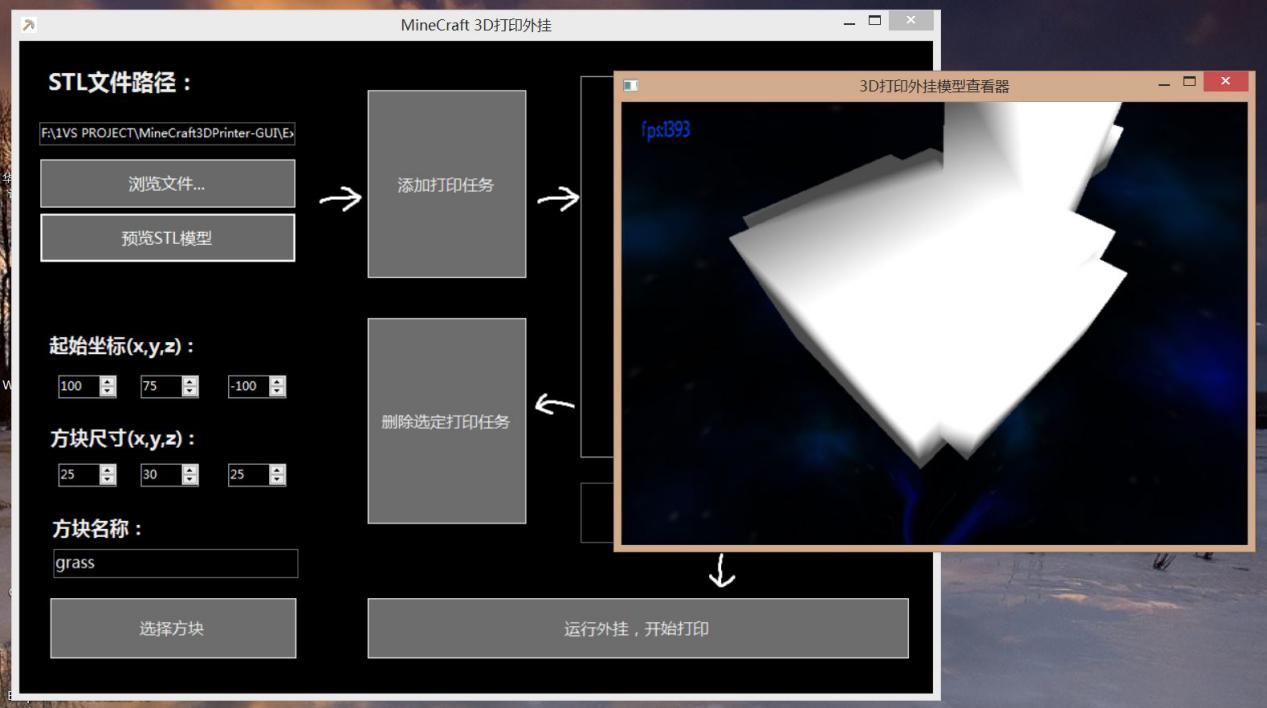
 对于一个切层上的所有线段，一般情况下，就用很多条均匀间隔的扫描线扫描下来，线线求交，然后就填充1和2之间，3和4之间....2k-1和2k之间的区域。

看起来很好很简单很强大。但是有一点我的程序为了简单化和运行效率没有考虑到，就是如果一个线段两个端点正好落在扫描线上的话，填充区域可能会判断失误。上图虽然都标了1，2，3，4，但是填充区域是严重依赖于2，3之间的角是突出还是凹下，就是凹凸性在这就非常关键了。我大致搜了一下，解决方案还是需要生成线段的邻接信息，于是碰到1，2这种棘手的线段时，就分别根据1和2的邻接线段做进一步的讨论。

* **按键模拟**

类似于按键精灵的方法，把复制到剪切板的MC控制台命令在游戏控制台中执行（特指/fill命令），从而在游戏场景里生成方块。

* **GUI**

****

用C# Winform做的GUI。