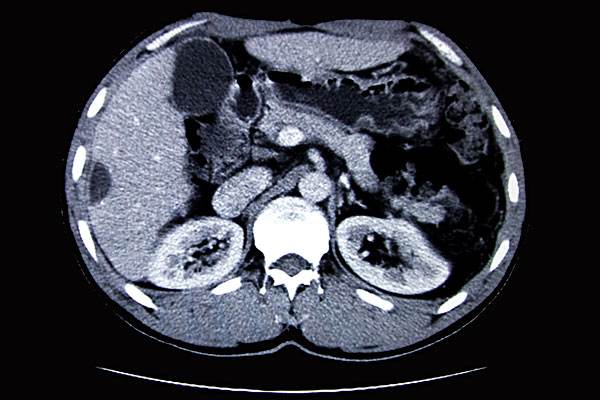
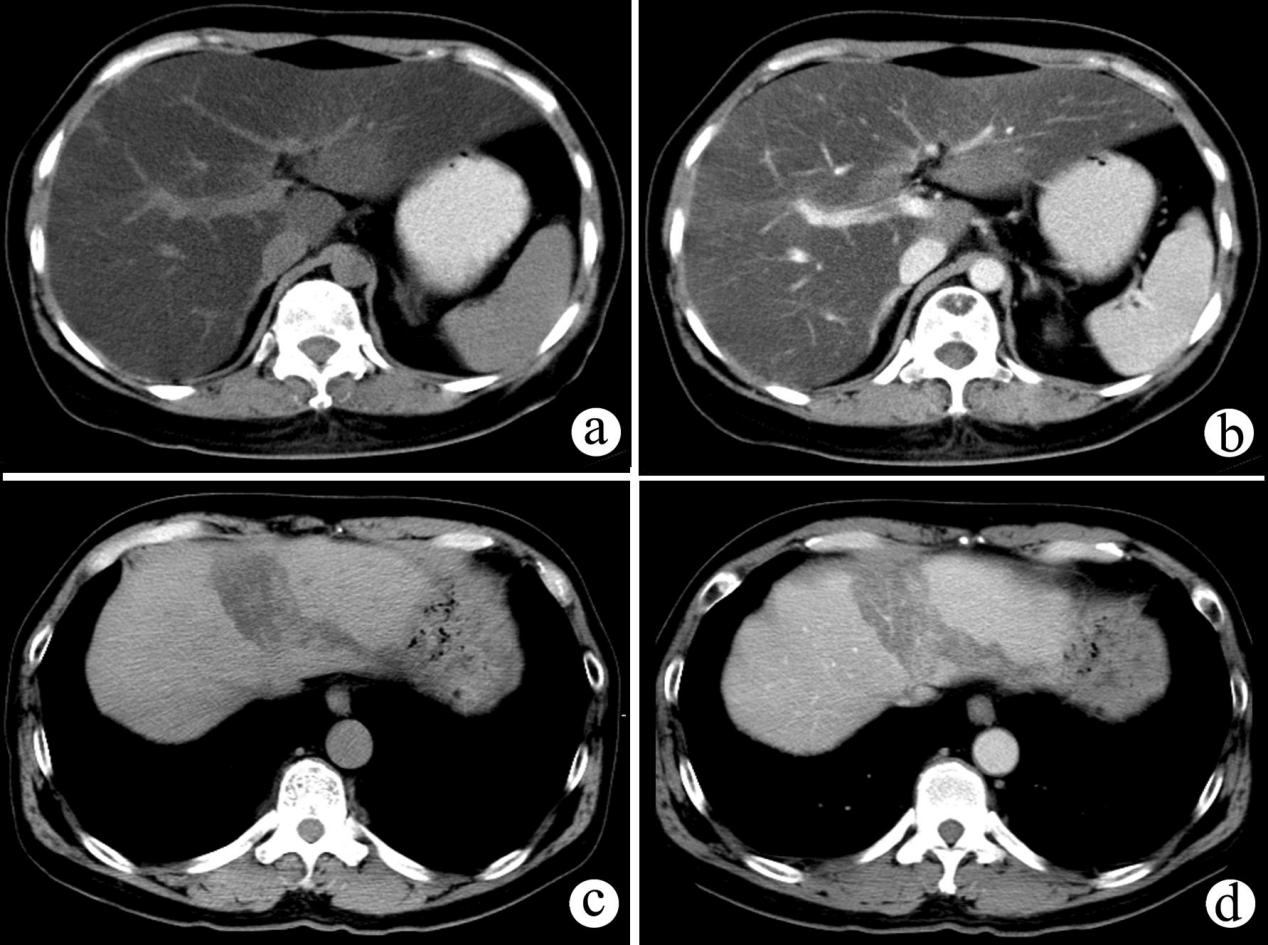
打个擦边球，最近我lead的国家级大学生创(hua)新(shui)创(xian)业(yu)项目快结题了，做的是基于**医学CT切层扫描的肝部三维重建。**最后还要可视化，就是把肝部的三维模型渲染出来，这样子的画我们团队选择的解决方案就是**重建出三角形网格**再基于我的迷你3d引擎用传统方法渲染。

* **CT断层扫描**



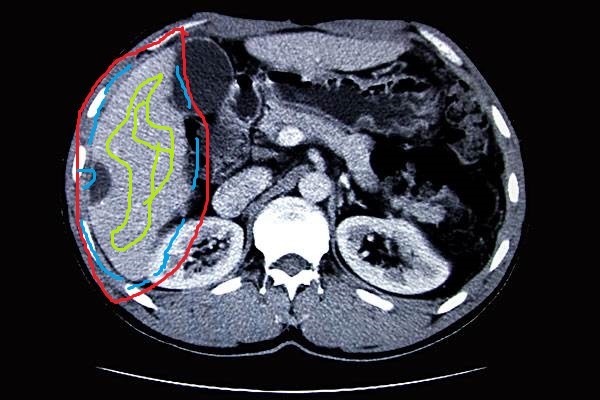
**（引用自百度百科\_CT）**CT(Computed Tomography)，即电子计算机断层扫描，它是利用精确准直的X线束、γ射线、超声波等，与灵敏度极高的探测器一同围绕人体的某一部位作一个接一个的断面扫描，具有扫描时间快，图像清晰等特点，可用于多种疾病的检查；根据所采用的射线不同可分为：[X射线](http://baike.baidu.com/item/X%E5%B0%84%E7%BA%BF" \t "http://baike.baidu.com/_blank)CT（X-CT）、超声CT（UCT）以及[γ射线](http://baike.baidu.com/item/%CE%B3%E5%B0%84%E7%BA%BF" \t "http://baike.baidu.com/_blank)CT（γ-CT)等。

反正要做一次CT扫描，就整个人躺在那，从那个洞里经过，全身过一遍。然后就能得到全身几十或者几百张CT扫描断层。（想象把一个人切成几百片吧，但是CT扫描只是像一系列采样而已，而且数据是灰度图）



一般的CT图就长上面的样子，上面的图大概就是在腰或者胸附近切的几刀（= =我也不知是哪，专业人士轻喷）虽然真的CT图会有更多信息，但其实就把它当**灰度图**看就好了，CT值和器官密度什么的在这个项目就不管了。

* **肝部识别**



这个不是本回答的重点，只说两句。师兄说要像上图那样标出目标器官大致区域和前后景区域，然后用openCV里的grabCut来把目标区域挖出来，这样精度会相对高一点。把目标区域挖出来以后就对图像**二值化**再保存成文件，**属于肝的像素是1，背景是0**。

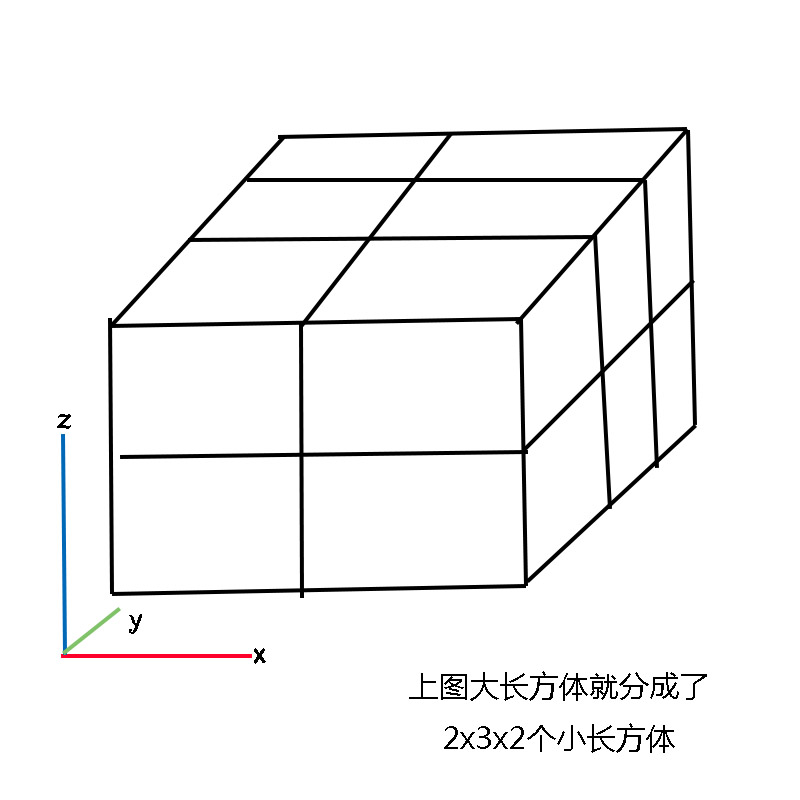
！！！！！！！！这个才是重点！！！！！！！！！！

* **基于MarchingCube算法的三维重建**

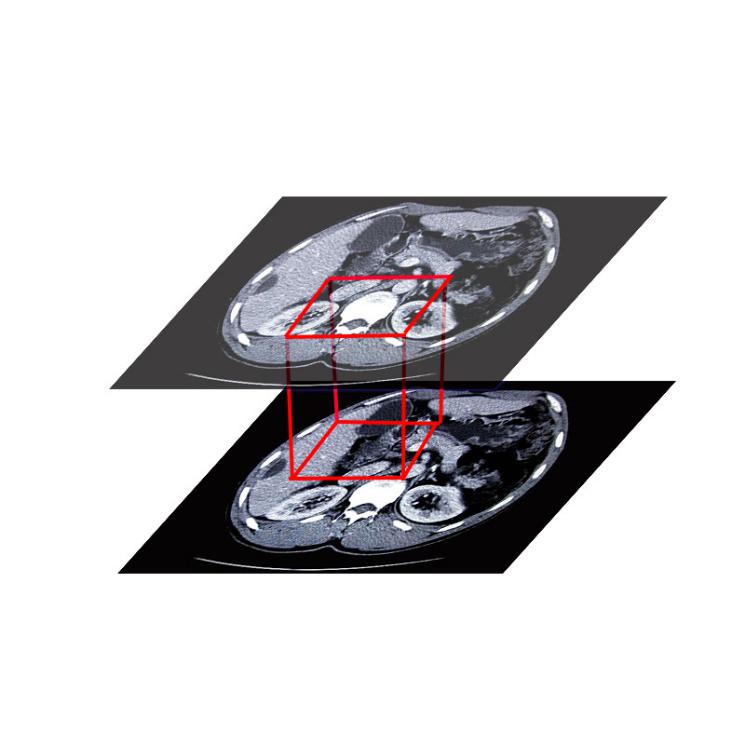
这个方法最早在1987年就在论文《Marching Cubes : A High Resolution 3D Surface Construction Algorithm》中被Willian E. Lorensen 和Harvey E. Cline共同提出[1]，论文里还特别地提到了一下医学影像的3d表面重建的基本流程。（ps：这论文到现在已经1.2万引用了,害怕）因为我看论文并没有非常仔细，所以有些细节可能跟这篇经典论文有出入，但是大致还是没问题的，毕竟MarchingCubes算法**我觉得是一个算法框架，可以有非常多的变体算法来适应自己的项目**。

首先，Marching Cube有个迷之中文翻译，叫“**移动立方体算法**”。所以从名字就能大概能看出了，这个三维重建的算法是一个**分治**的算法（divide and conquer）。（虽然“移动”这个词是有点迷）。

我们先想象一下，用一个超大的长方体**包住目标物体**（就是目标器官必须完全地位于大长方体内部）。再把这个大长方体，分成 A x B x C个一模一样的小长方体。



其中，每个**平行于水平面的长方体截面就对应着一个CT断层图片（的一部分）**。

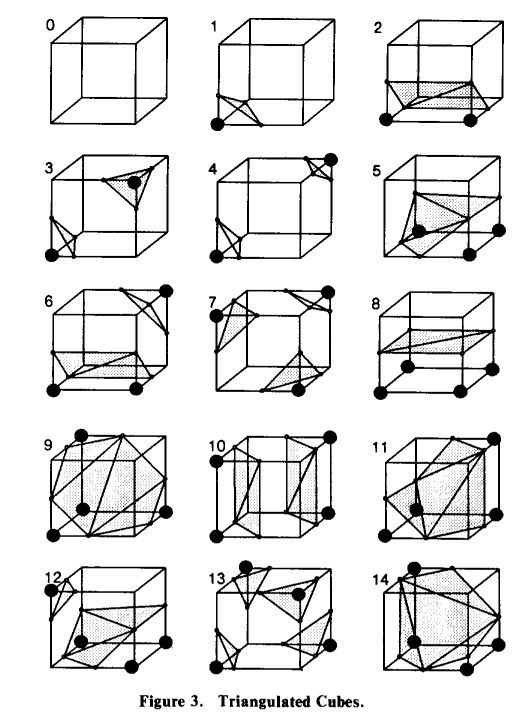


上图是我随手p的一张示意图，示意着某个小长方体与和它相邻两个CT截面之间的关系。

所以，这个“分而治之”的**基本单元就是一个小立方体**，这么做其实相当于是三维空间上的重采样，所以你要分成多少个长方体其实你可以自己定的。下一个思想也是比较关键的，就是**判断小立方体的8个顶点分别是否在目标器官的内部**。如果某个顶点在物体内部，那么给这个顶点标上一个0；如果这顶点在物体外部，则给它标上一个1。好了那么怎么判断某个顶点在不在目标器官内部呢？这就用上了**上一个步骤得到的二值化图**，直接看图的对应位置是不是1就好了（刚才不是说了图像处理步骤得到的结果，目标器官的区域的像素就标为1）。

之后我们根据图片判断出8个顶点的“0”与“1”。排列组合就有2^8=256种情况。每一种情况，我们可以在小立方体内生成一些**等值面**，或者理解成生成**0个或多个位于立方体内部的三角形**。对，一共有256种局部三角形的组成的情况。

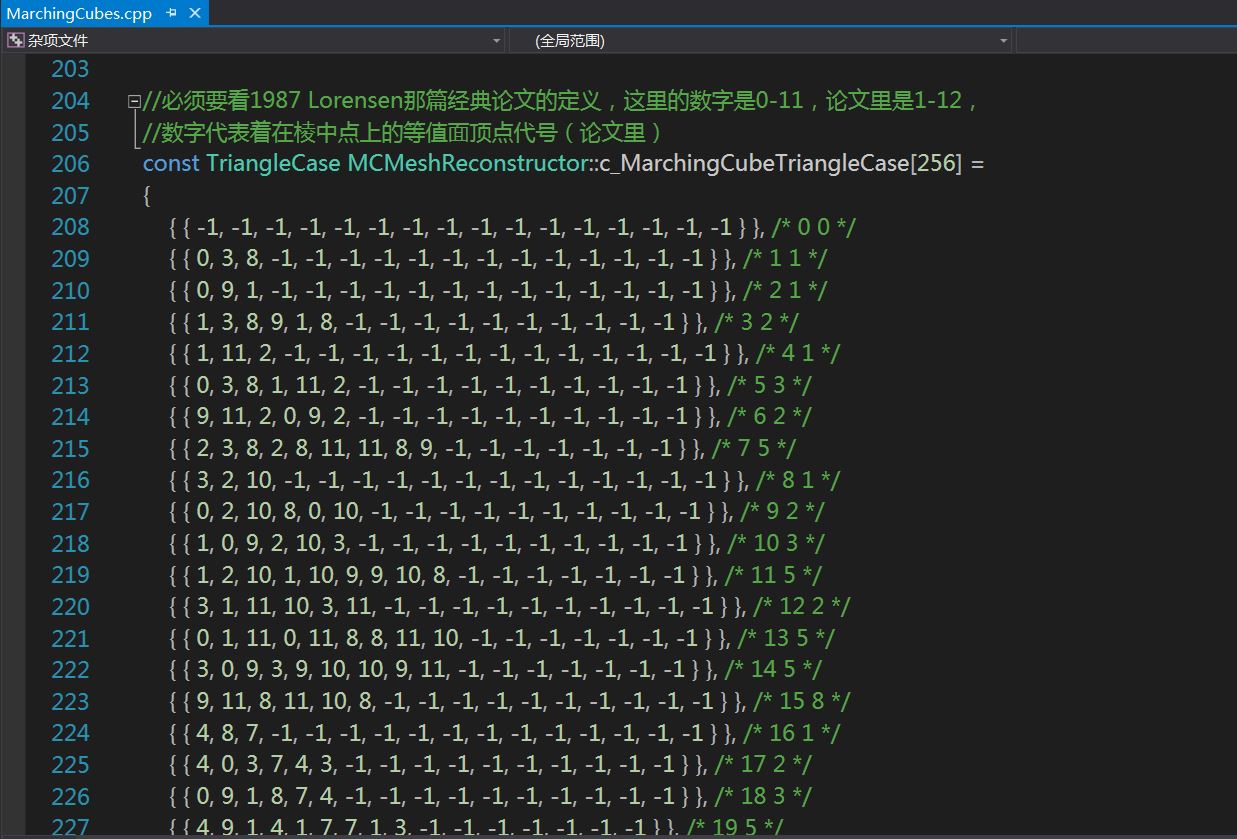
为了针对某个小立方体去生成局部三角形，MarchingCubes非常丧心病狂地做了一件事：**穷举了256种情况能生成的三角形**。（虽然，这256种情况可以**被概括成15个基本情况**，基本情况经过旋转、镜像等操作可以生成所有的256种情况）。



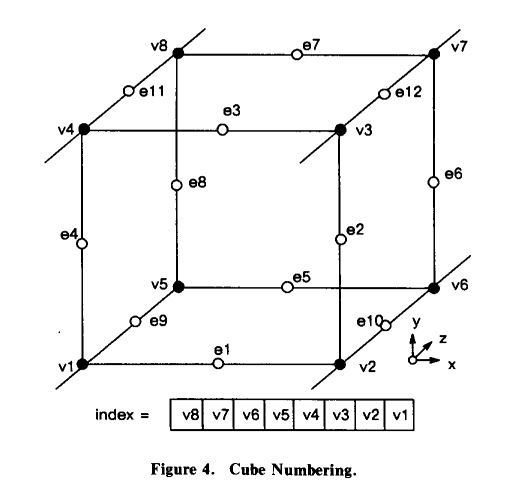
上图就是引用自1987年论文[1]里面的图，图里描述了15种基本情况。上图基本说说明了，什么情况就可以在哪生成三角形。其中**三角形的每个顶点都是在小立方体的棱上**的。于是每**个小立方体都这么干，所有小立方体生成的三角形拼在一起就是目标器官的表面了**。

（ps : 这种穷举法的描述里面是会有**二义性(Ambiguity)**的情况存在，在这里不展开讲）

论文这么说是很舒服的，代码写起来一点都不舒服好吧。于是就稍微参考了下VTK (Visualization Toolkit)的MC模块的源码，可以上github找去，这库还是很强的。然而看别人代码，而且是没什么注释的代码真的难受，所以还是自己写了。**只不过VTK这个模块的代码有段很值钱的代码，对，就是那个穷举256种情况的数组。**

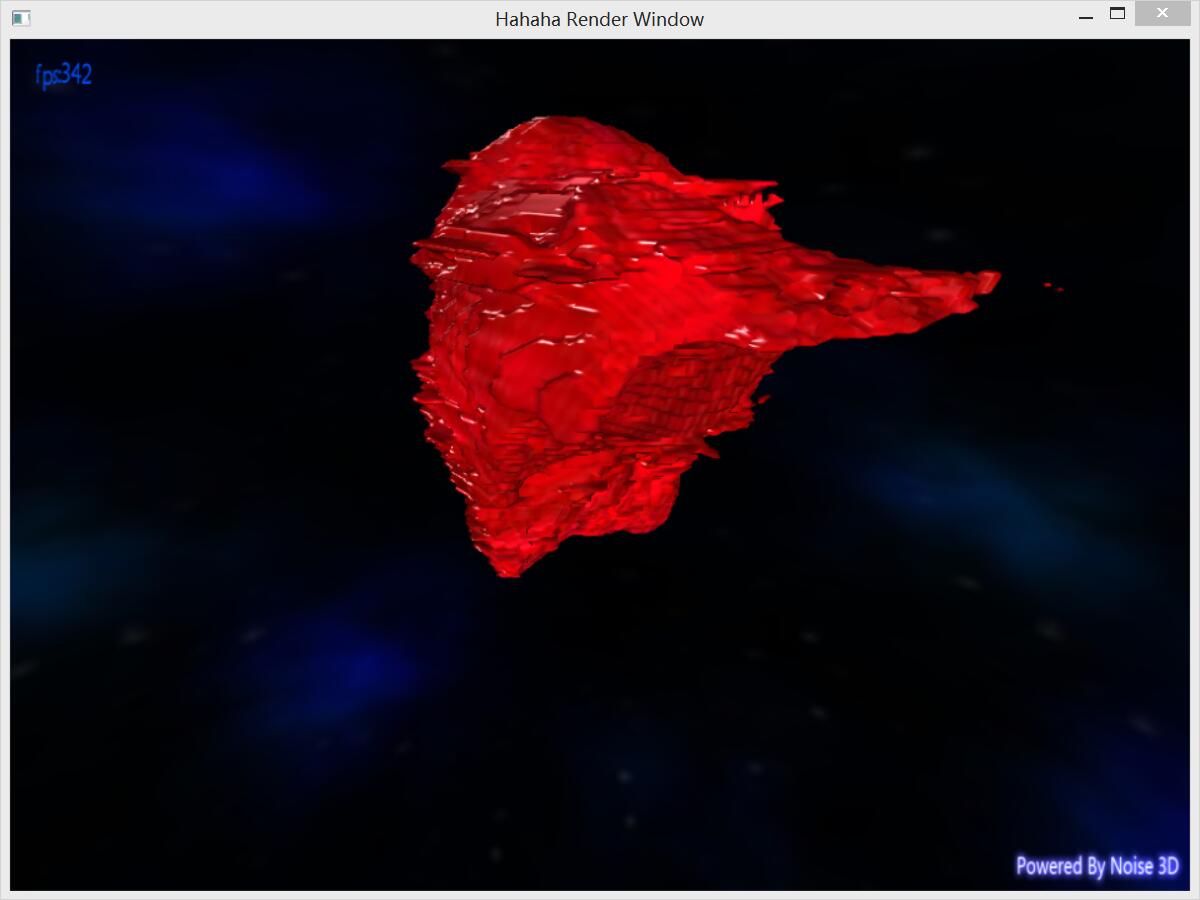
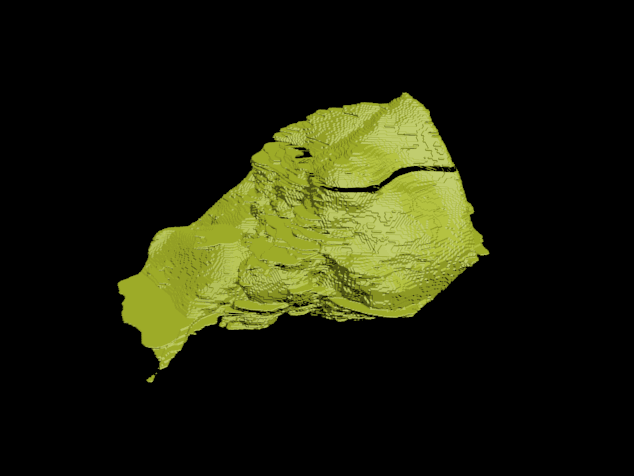


上图我把VTK里那个珍贵的数组复制了过来=。=不知道有没有侵权，嘘



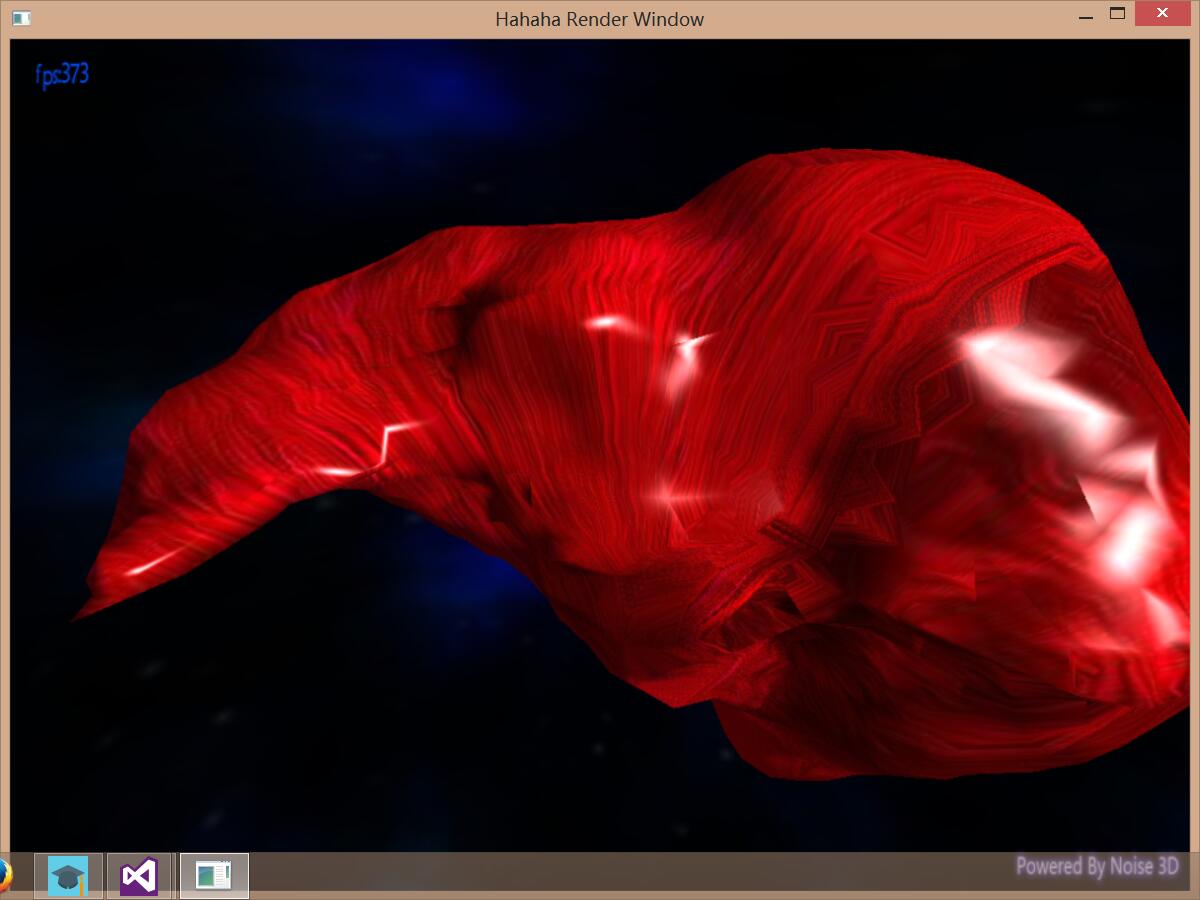
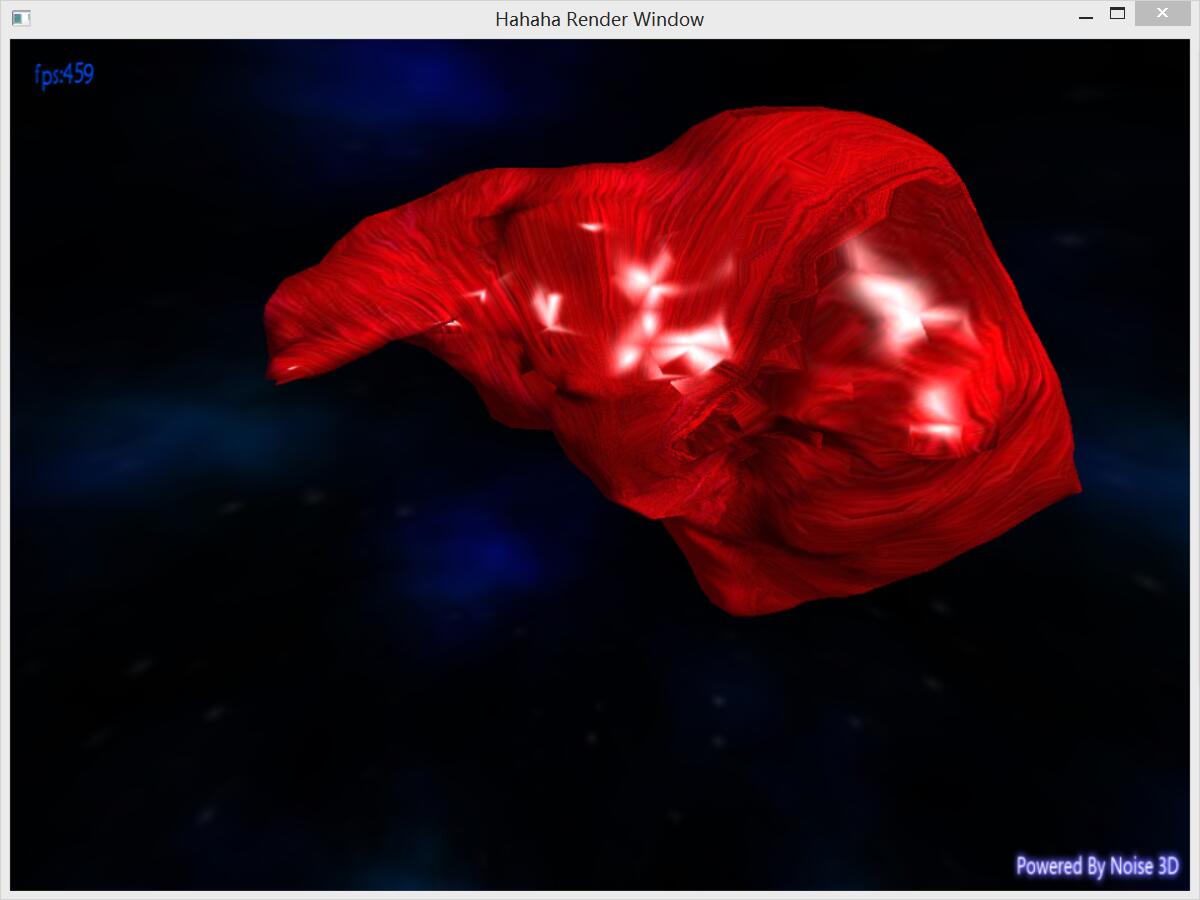
仔细把那个数组和上面论文里的图的v1-v8, e1-e12 对比了一下，发现是没问题的，VTK就是按论文的定义去实现的，注意下标的起始数字就行了。

* **三维重建效果图**



我的天，丑的吓人！但是这个丑讲道理是不能甩锅给MarchingCube的，要怪就怪图像识别做的不怎么样。（但在这项目里改过的MarchingCube算法生成的模型还是比较多的明显锯齿）。

所以最后折衷一下做个网格简化吧！



好多了好多了！

就讲到这！

我写的MarchingCube算法代码也在github上：

<https://github.com/CHINA-JIGE/MarchingCube>

--------------------------------------------------------

[1]Lorensen W E, Cline H E. Marching cubes: A high resolution 3D surface construction algorithm[J]. Acm Siggraph Computer Graphics, 1987, 21(4):163--169.