* **NoiseFileManager：**

1. (2015.7.22)支持纯粹加载文件到vector<char>；导出文件(vector<char>)
2. (2015.7.22加载stl文件，要求输入vertex\index\normal\info 的buffer指针。
3. (2015.8.2)新增导入/导出 NOISELAYER切层文件，自己瞎编的一个格式，详情参阅下一部分的 2015.7.30 NoiseUtSlicer
4. 在读取二进制文件的时候不要用析取器">>"，这个会自动解析数据并转码，跟标准流是一样的。但在读取ASCII文本的时候析取器会很方便。
5. (2015.2.13)把各个不同的格式的文件操作分在不同的cpp里面实现，比较模块化(/手动微笑)。

**————2015.7.22————**

**加载STL**

(2015.7.22)唔，想不到在VB里面实现很容易的东西在c++还有那么多手尾。加载主要还是用ifstream，然而是一个一个char的读。之后reinterpret的时候又出问题了，把4个char合成一个int或者float的时候，本来想着是用位移再合并。然后在类型转换的时候还是出了问题，static\_cast之类的会做处理，不是单纯的复制二进制，所以4char合1float的时候还是用到了memcpy。然后还有一个坑爹的地方，float或者int那种东西的储存大部分机器都是低位在前，跟人类思维有点不同...坑死了.....结果移位是把最后一个char移到最前。

int tmpInt = static\_cast<int>(c1)|static\_cast<int>(c2)<<8|static\_casr<int>(c3)<<16|static\_cast<int>(c4)<<24;

二进制用fstream.read读取char\*的数据串，然后再强制转换为需要的数据。

char\* fileBlock;

#define REINTERPRET\_READ(var)\

fileIn.read(fileBlock,sizeof(var));

var = \*(decltype(var)\*)fileBlock;

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)

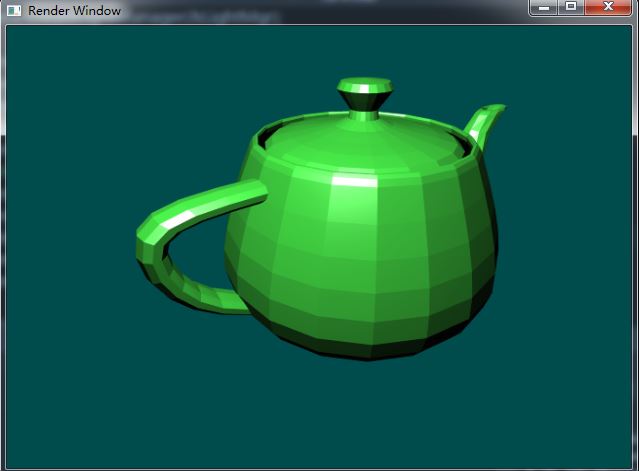
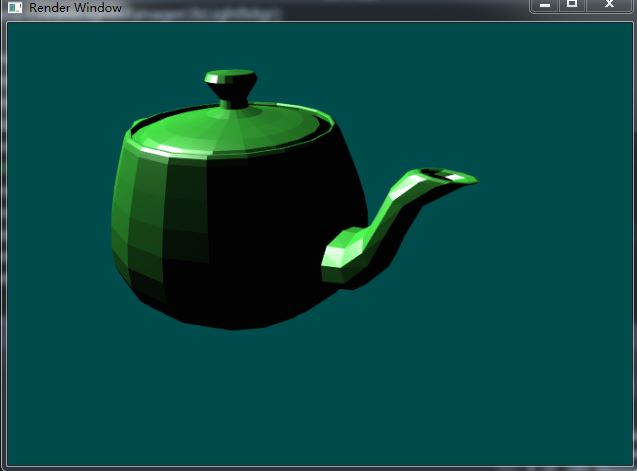
3个4字节浮点数(2顶点的坐标)

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。

**最后，上图**



(2016.2.1)二进制和ASCII的STL文件是可以区分的，这个网上有人是看文件第一个字节是不是's'，因为只有ASCII的STL是以"solid"开头，二进制的不可以以's'开头。

**————2016.2.1————**

**加载OBJ**

 (2016.2.1) OBJ文件是Alias|Wavefront公司为它的一套基于工作站的3D建模和动画软件"Advanced Visualizer"开发的一种标准3D模型文件格式。OBJ是用ASCII文本书写的文件。是这种文件支持很多种几何元素，例如点、直线、三角形、多边形、曲线。但是对于Noise3D来说，OBJ一般只会用到三角形图元。若只考虑基本的顶点信息，那么OBJ就用考虑几个关键字。

v ----顶点，接三个float

vn----顶点法线，接三个float

vf----纹理坐标，接两个float

f ----面的各顶点的顶点信息索引，接多组整数(组数视图元而定）

例如f 1/1/1 2/2/1 3/3/1，每一组数记录的是：

顶点位置坐标索引/纹理坐标索引/顶点法线索引

每一个关键字后面都接一堆数字，就是对应的信息了。所以其实是很容易读的。文件会给出三堆东西，分别记录着v,vn,vf的信息，然后f记录的索引就在这三堆东西里面选择。因为引擎最后需要的只是完整顶点和顶点索引，所以f给出的顶点信息索引组合要进行的一定的整合和去重。

其实Obj也可以加载材质的，关键字mtllib和usemtl就可以做到，但是问题就是获取材质信息要链接一个.mtl，而这个.mtl里面的信息好像也不怎么好用...纹理文件居然用绝对路径，给跪。

**————2016.2.13————**

**加载3ds**

 (2016.2.13)这格式才是正儿八经可以带相对路径下的材质的，那也是不错的。3ds有个小优点就是信息很丰富，而且文件构造决定了自己可以选择加载哪些信息。3ds文件总体来说比较像树状的结构，由一个个chunk（块）组成。每个chunk有三个部分组成：

1. 2 byte(uint16\_t)的identifier，这个能读到预定义的数字，分别代表了不同的块
2. 4 byte(uint32\_t)的length，这个是当前块的字节长度（包括head的6字节）。
3. 块的内容（可能全是子块，也可能是当前块的data+子块）

下面是部分chunk的id和层级结构：

MAIN CHUNK 0x4D4D

3D EDITOR CHUNK 0x3D3D

OBJECT BLOCK 0x4000

TRIANGULAR MESH 0x4100

VERTICES LIST 0x4110

FACES DESCRIPTION 0x4120

FACES MATERIAL 0x4130

MAPPING COORDINATES LIST 0x4140

SMOOTHING GROUP LIST 0x4150

LOCAL COORDINATES SYSTEM 0x4160

LIGHT 0x4600

SPOTLIGHT 0x4610

CAMERA 0x4700

MATERIAL BLOCK 0xAFFF

MATERIAL NAME 0xA000

AMBIENT COLOR 0xA010

DIFFUSE COLOR 0xA020

SPECULAR COLOR 0xA030

TEXTURE MAP 1 0xA200

BUMP MAP 0xA230

REFLECTION MAP 0xA220

[SUB CHUNKS FOR EACH MAP]

MAPPING FILENAME 0xA300

MAPPING PARAMETERS 0xA351

KEYFRAMER CHUNK 0xB000

MESH INFORMATION BLOCK 0xB002

SPOT LIGHT INFORMATION BLOCK 0xB007

FRAMES (START AND END) 0xB008

OBJECT NAME 0xB010

OBJECT PIVOT POINT 0xB013

POSITION TRACK 0xB020

ROTATION TRACK 0xB021

SCALE TRACK 0xB022

HIERARCHY POSITION 0xB030

**参考http://anony3721.blog.163.com/blog/static/5119742011525103920153/**

下面是部分chunk的内容及构造介绍（貌似有点错误的）：

0x4D4D：根chunk，每一个3ds文件都起自它，它的长度也就是文件的长度。它包含了两个chunk：编辑器，和关键帧。

父chunk：无

子chunk：0x3D3D、0xB000

长度：头长度+子chunk长度

内容：无

0x3D3D：编辑器主chunk，它包含有：网格信息、灯光信息、摄象机信息和材质信息。

父chunk：0x4D4D

子chunk：0x4000、0xafff

长度：头长度+子chunk长度

内容：无

0x4000：网格主chunk，它包含了所有的网格。

父chunk：0x3D3D

子chunk：0x4100

长度：头长度+子chunk长度+内容长度

内容：

名称（以空字节结尾的字符串）

0x4100：网格信息，包含网格名称、顶点、面、纹理坐标等。

父chunk：0x4000

子chunk：0x4110、0x4120、0x4140、0x4160

长度：头长度+子chunk长度

内容：无

0x4110：顶点信息。

父chunk：0x4100

子chunk：无

长度：头长度+内容长度

内容：

顶点个数（一个字）

顶点坐标（三个浮点数一个坐标x、y、z，个数\*3\*浮点数）

0x4120：面信息。

父chunk：0x4100

子chunk：0x4130

长度：头长度+子chunk长度+内容长度

内容：

面个数（一个字）

顶点索引（三个字一个索引1、2、3，个数\*3\*字）

0x4130：与网格相关的材质信息。

父chunk：0x4120

子chunk：无

长度：头长度+内容长度

内容：

名称（以空字节结尾的字符串）

与材质相连的面的个数（一个字）

与材质相连的面的索引（个数\*字）

0x4140：纹理坐标。

父chunk：0x4100

子chunk：无

长度：头长度+内容长度

内容：

坐标个数（一个字）

坐标（两个浮点数一个坐标u、v，个数\*2\*浮点数）

0x4160：转换矩阵。

父chunk：0x4100

子chunk：无

长度：头长度+内容长度

内容：

x轴的向量（三个浮点数u、v、n）

y轴的向量（三个浮点数u、v、n）

z轴的向量（三个浮点数u、v、n）

源点坐标（三个浮点数x、y、z）

0xafff：材质信息。

父chunk：0x4D4D

子chunk：0xa000、0xa020、0xa200

长度：头长度+子chunk长度

内容：无

0xa000：材质名称。

父chunk：0xafff

子chunk：无

长度：头长度+内容长度

内容：

名称（以空字节结尾的字符串）

0xa020：Diffuse。

父chunk：0xafff

子chunk：0x0011、0x0012

长度：头长度+子chunk长度

内容：无

0xa200：纹理帖图。

父chunk：0xafff

子chunk：0xa300

长度：头长度+子chunk长度

内容：无

0xa300：贴图名称。

父chunk：0xa200

子chunk：无

长度：头长度+内容长度

内容：

名称（以空字节结尾的字符串）

其实呢主要还是要参考一个叫Assimp的项目里面的3ds loader，这项目支持四十几种3d模型格式import/export，牛逼炸了，那还是挺完整的解决方案。

(2016.3.12)所以最后还是选择了比较线性的解决方案，而不是树状或者递归加载。3ds可以包括很多个物体，但是现时的接口是让Mesh去load3DS，所以多个网格会被合并为一个网格。就为了这个合并，我是在3ds的实现cpp里面定义了partialMeshObj来加载各个子网格，然后最后再合并（不然如果要计算offset加载索引上面的话，很容易乱，这个地方我是吃过屎的）。因为material什么的都要用string的名字来identify啦，所以就比较方便加载文件，只需要再output一个名字String到路径string的hash map(std::unordered\_map)给用户就好了。然后这里又有个坑，就是3ds文件的路径可能不只是一个文件名，而是前面有一小段相对路径的目录，那就稍微需要处理一下。