|  |
| --- |
| 软件81 |
| 计算机图形学大作业2技术文档 |
| SMF图形平台 |

|  |
| --- |
| 袁博志  2008080060 |

[1. 实验目的 1](#_Toc291189202)

[2. 实验平台 1](#_Toc291189203)

[3. 实验内容 1](#_Toc291189204)

[4. 实验具体实现 2](#_Toc291189205)

[全局 2](#_Toc291189206)

[1. SMF文件导入 2](#_Toc291189207)

[2. 面消隐(几乎80%的工作量) 2](#_Toc291189208)

[3. 正平行投影 2](#_Toc291189209)

[4. 用OpenGL实现先消隐和真实感绘制 2](#_Toc291189210)

[5. OpenGL实现平行投影，透视投影 2](#_Toc291189211)

[6. 平移，缩放，旋转 3](#_Toc291189212)

[图形的转换 3](#_Toc291189213)

[视角的转换 3](#_Toc291189214)

[透视投影模式下 3](#_Toc291189215)

[7. 生成三视图 3](#_Toc291189216)

[5．测试结果分析 3](#_Toc291189217)

[1. 导入SMF 3](#_Toc291189218)

[2. 面消隐 4](#_Toc291189219)

[此处仅选择比较复杂图形进行不同方位测试 4](#_Toc291189220)

[3. 线消隐&真实感绘制 5](#_Toc291189221)

[4. 平行投影&透视投影 5](#_Toc291189222)

[5. 正平行投影和三视图 6](#_Toc291189223)

[6. 心得体会 6](#_Toc291189224)

[7. 遗留问题： 6](#_Toc291189225)

# 实验目的

通过opengl库理解图形的消隐和投影等一些列变换。

# 实验平台

Win7 x64，netbeans6.7 jre1.6 jdk6 下的jogl

# 实验内容

* 1. **实现网格模型(\*.smf)的导入；**
  2. **不调用图形库实现模型的面消隐；**
  3. **不调用图形库实现正平行投影；**
  4. **用OpenGL实现模型的线消隐和真实感绘制；**
  5. **用OpenGL实现平行投影和透视投影；**
  6. **通过键盘或鼠标实现模型的缩放、旋转、平移；**

**选做：生成模型的三视图，并通过修改三视图中**

# 实验具体实现

## 全局

由于前期研究一段时间MFC，但是还是不是特清楚所以本次实验室选择采用在java1.6下的jogl实现的。

## SMF文件导入

一行一行的读。采用了自己写的V\_Face & V\_Point类保存实例。并且以便日后显示

## 面消隐(几乎80%的工作量)

本人采用了基于线的Z-buffer算法，具体实现是先把每个面保存在一个Face\_detail的实例中。在这个实例中，保存了几乎所有这个面的信息。然后利用Z-buffer的y从小到大的扫描顺序来扫描的全图片。

由于本次实验室用的三维坐标，然后将其实现在二维平面上，所以我们在进行面消隐的时候比较的Z值是在我们视口为xy平面坐标系中的Z值，所以得把所有的点进行一下坐标转换，然后在这个新的坐标系下比较Z来确定屏幕上的点的颜色，然后在转换回以前的坐标系值来绘制点。

**特别注意：因为透视投影图形会扭曲，所以我们只考虑在平行投影下进行面消隐。**

## 正平行投影

平行投影就是当前视口中的屏幕截图的感觉，但是实现不是那样的，因为电脑里保存的是世界坐标系的坐标变换，我们得像面消隐一样，把坐标系先转换成视口的坐标系，然后就可以忽略这些点的Z值，直接根据x，y绘制。

## 用OpenGL实现先消隐和真实感绘制

这块不用说什么，1姐德ppt上写的很清楚，赞一个。

## OpenGL实现平行投影，透视投影

平行投影就直接用gl.glOrtho搭配着 glu.gluLookAt使用，这个网上好多

透视投影 glu.gluPerspective具体参数刚开始思考了一阵，后来终于思路畅通了。

## 平移，缩放，旋转

### 图形的转换

通过变换矩阵来实现图形的平移旋转

### 视角的转换

#### 平行投影模式下：

* 1. 平移：无
  2. 缩放：利用滚轮。调节glu.gluLookAt
  3. 旋转：选在围绕y轴旋转或围绕x轴旋转，然后利用鼠标。调节glu.gluLookAt

#### 透视投影模式下：

* 1. 平移：选中平移，鼠标拖动。调节gl.glTranslated
  2. 缩放：利用滚轮。调节glu.gluPerspective
  3. 旋转：给定旋转平面的法向量，然后鼠标拖动。调节gl.glRotated

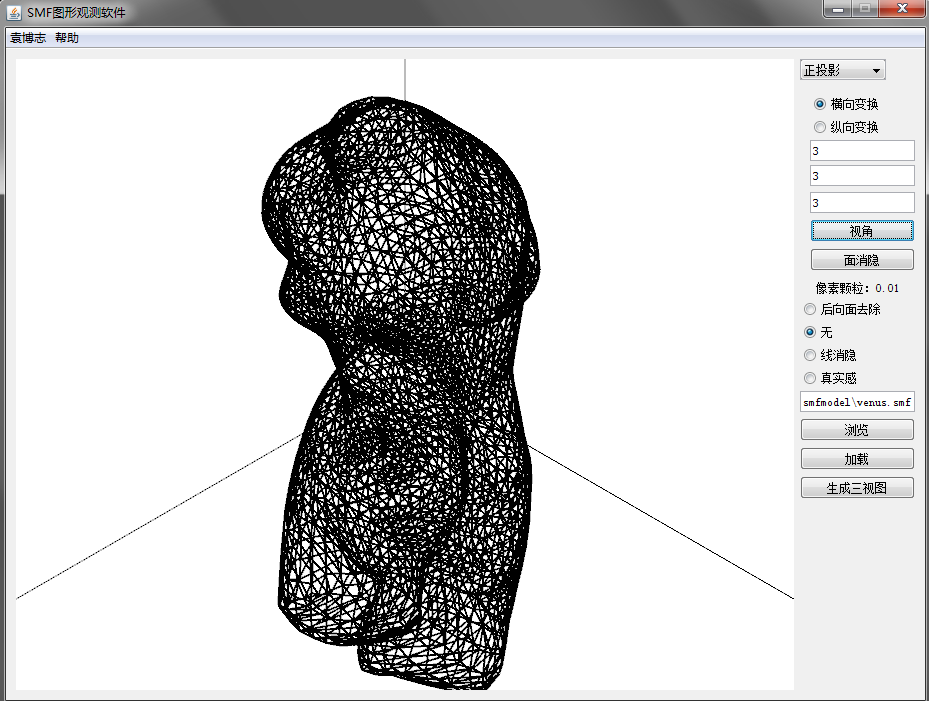
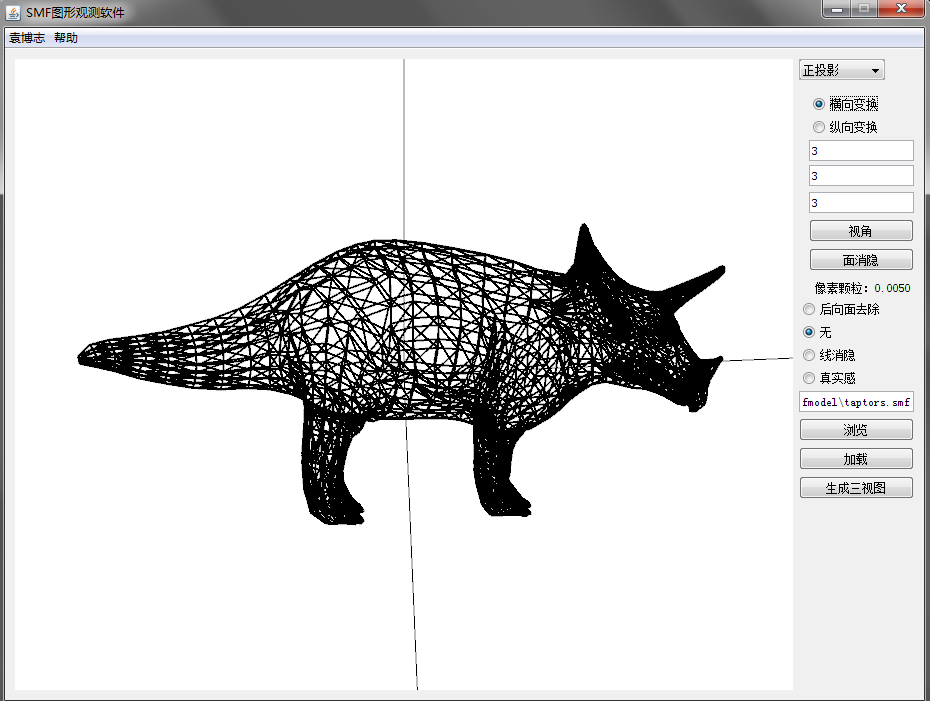
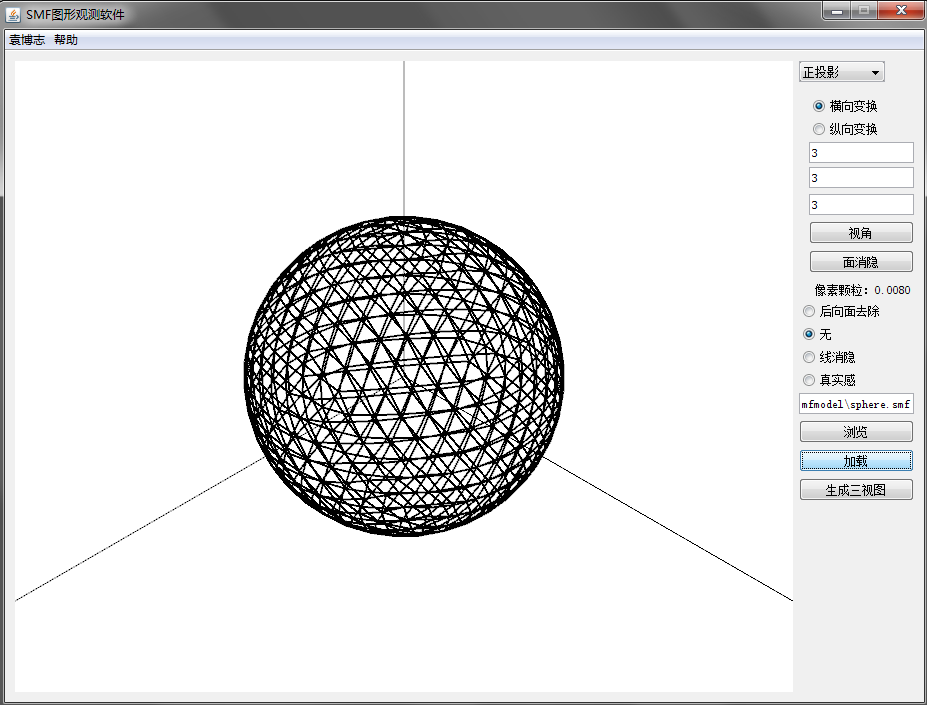
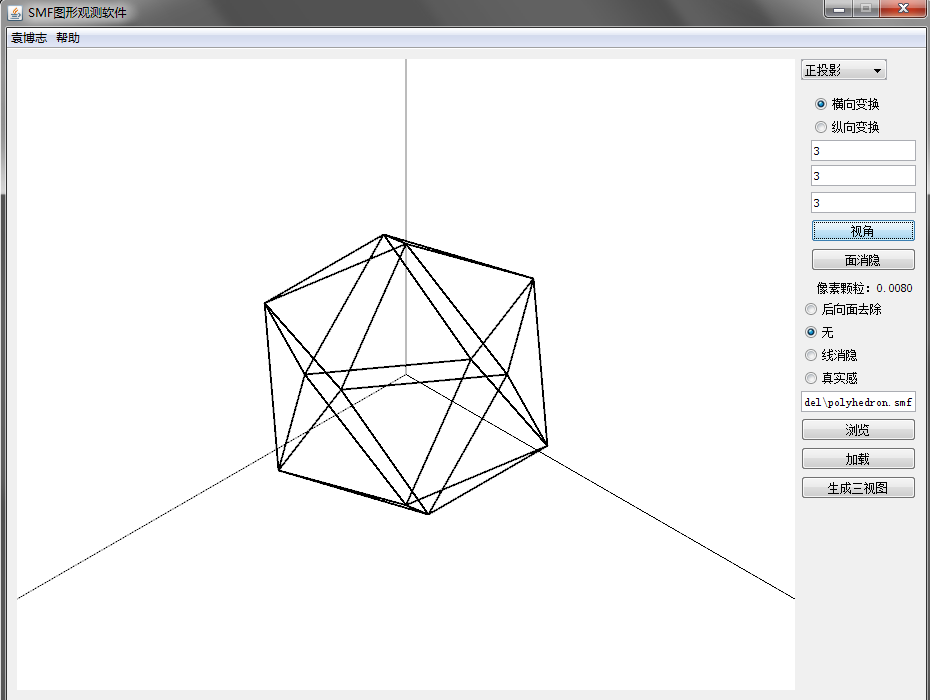
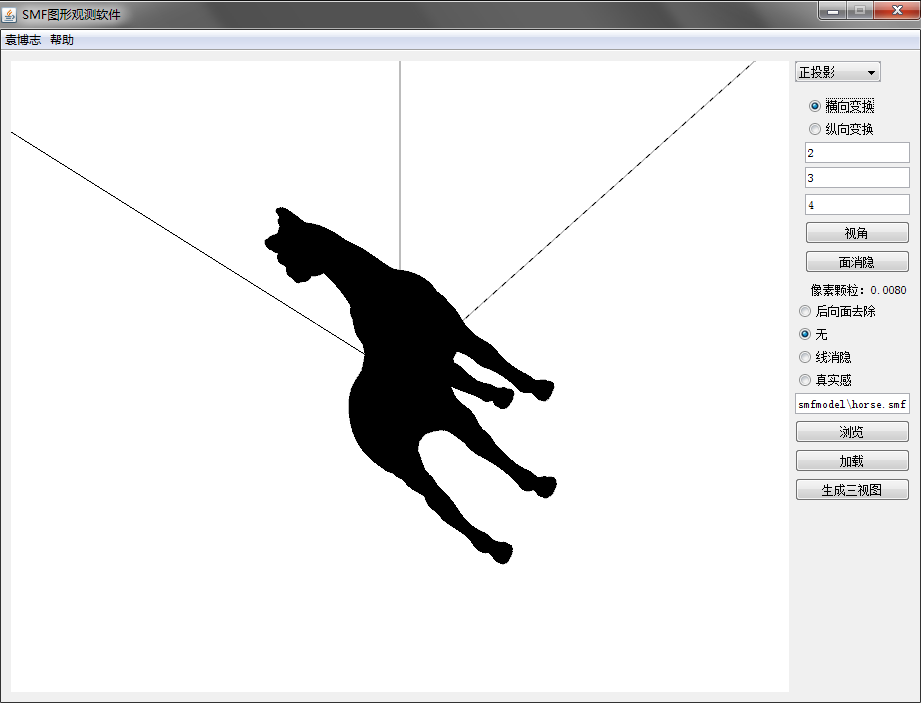
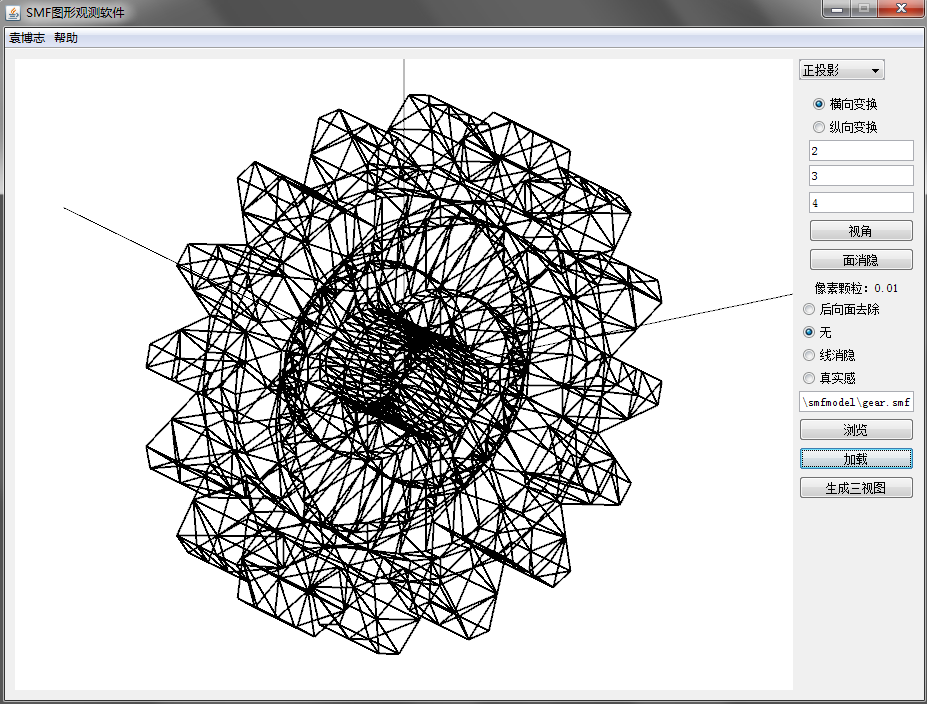
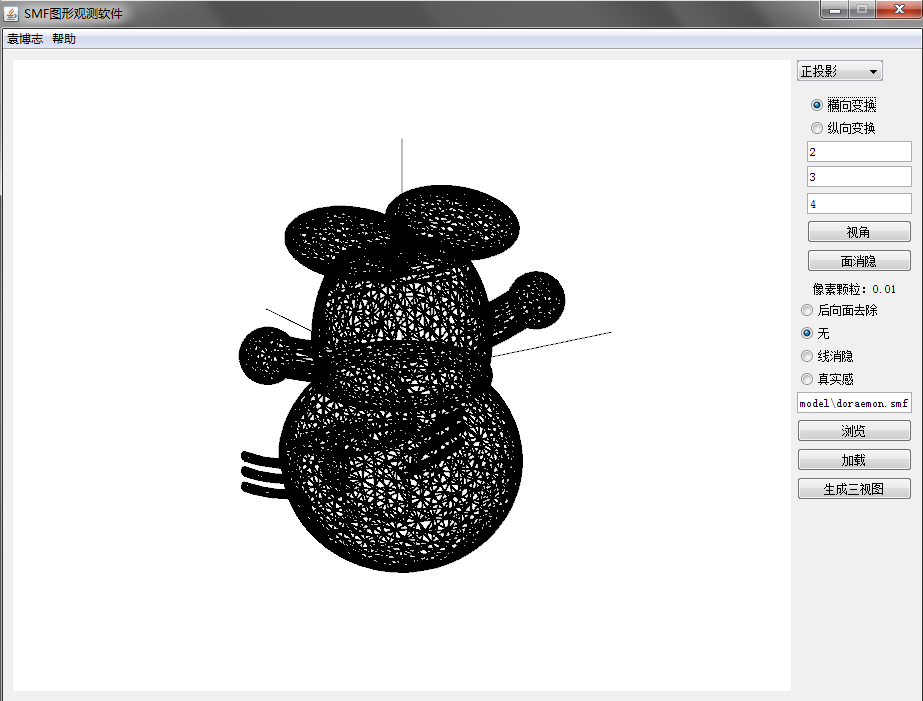
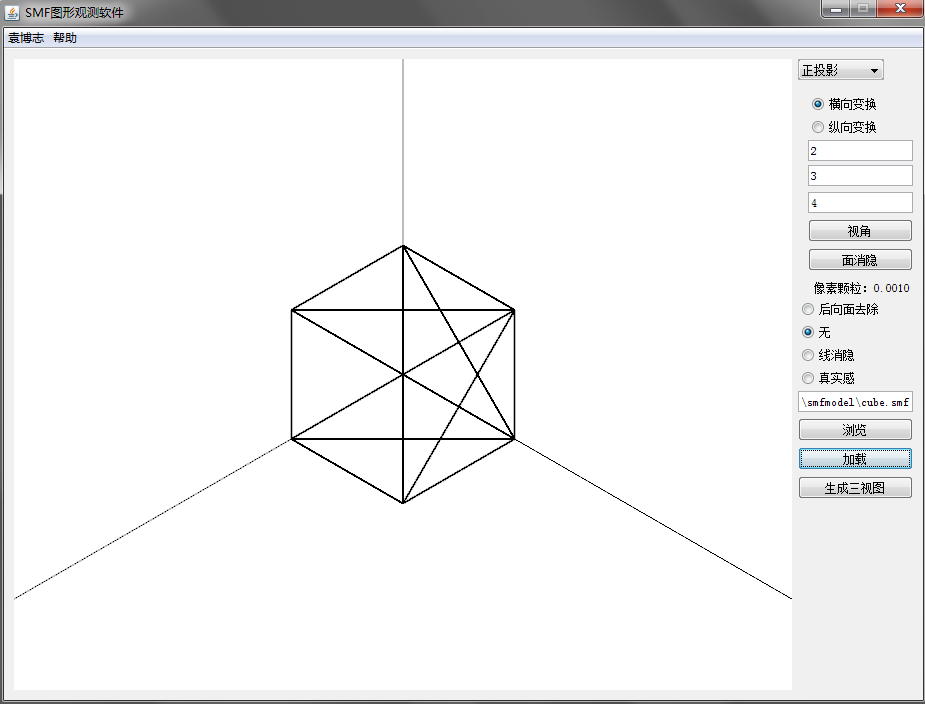
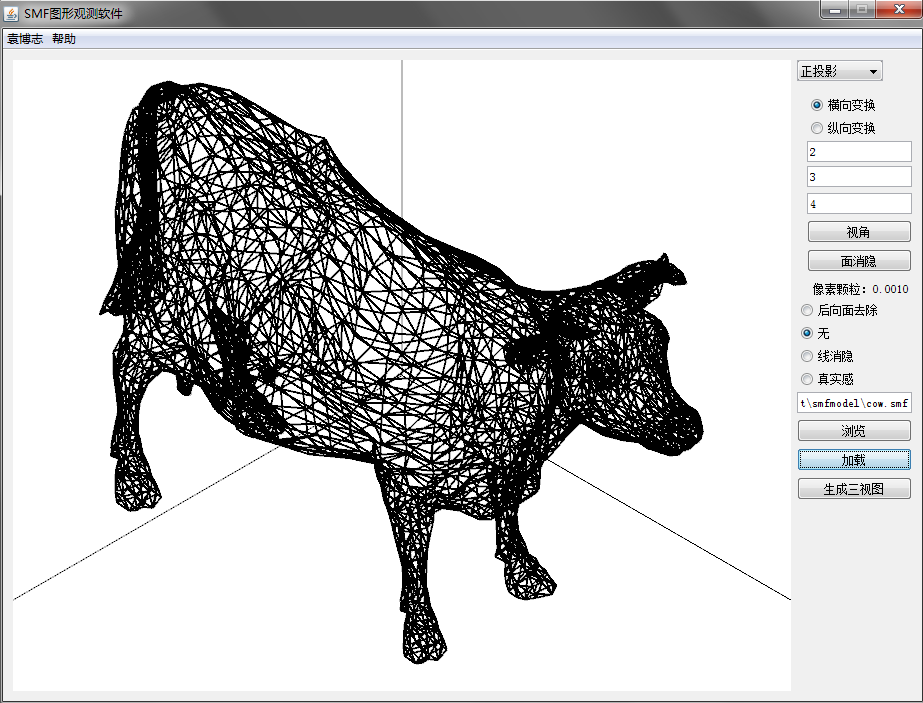
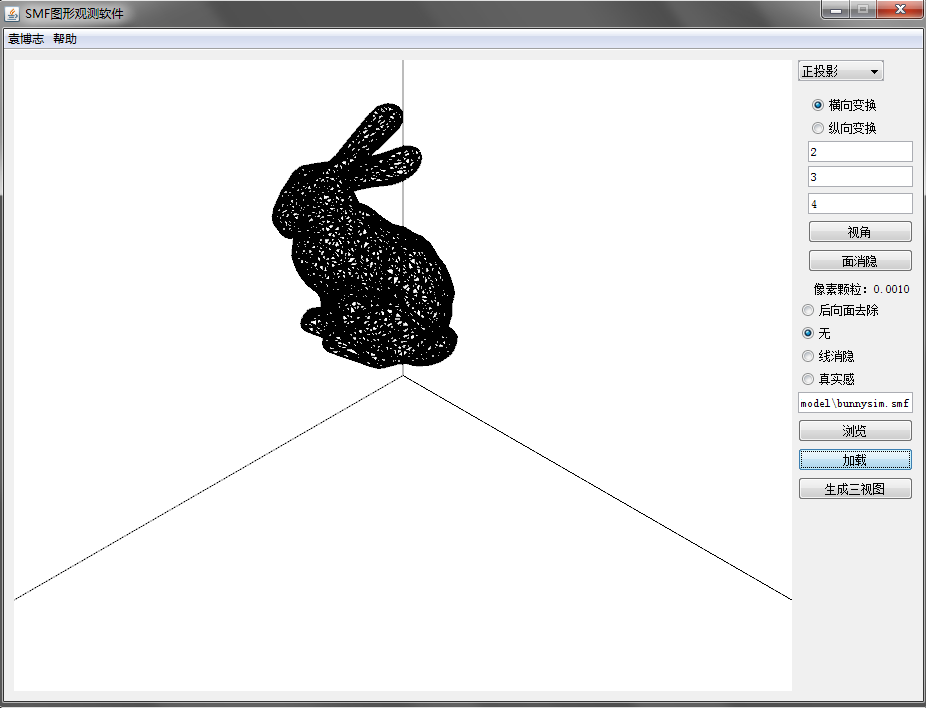
这块比较纠结，因为是三位坐标所以旋转起来的换很麻烦，所以思考了许久如何让用户交互，但最终无果，所以就采用了比较简单的OpenGl的库函数。

## 生成三视图

直接利用OpenGl的LookAt函数来获取三视图。

# 5．测试结果分析

## 导入SMF

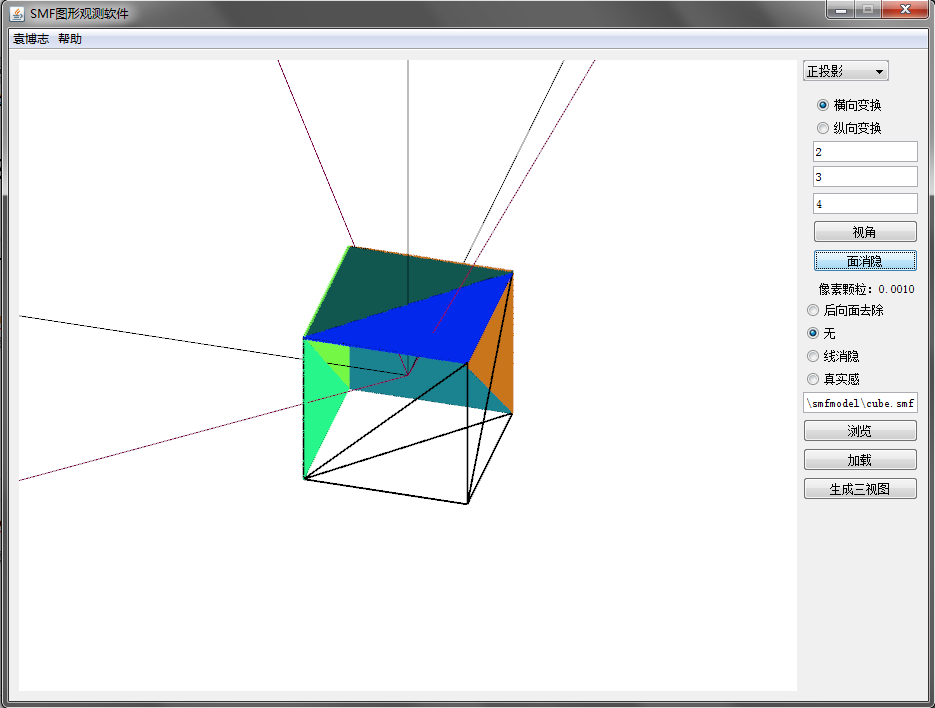
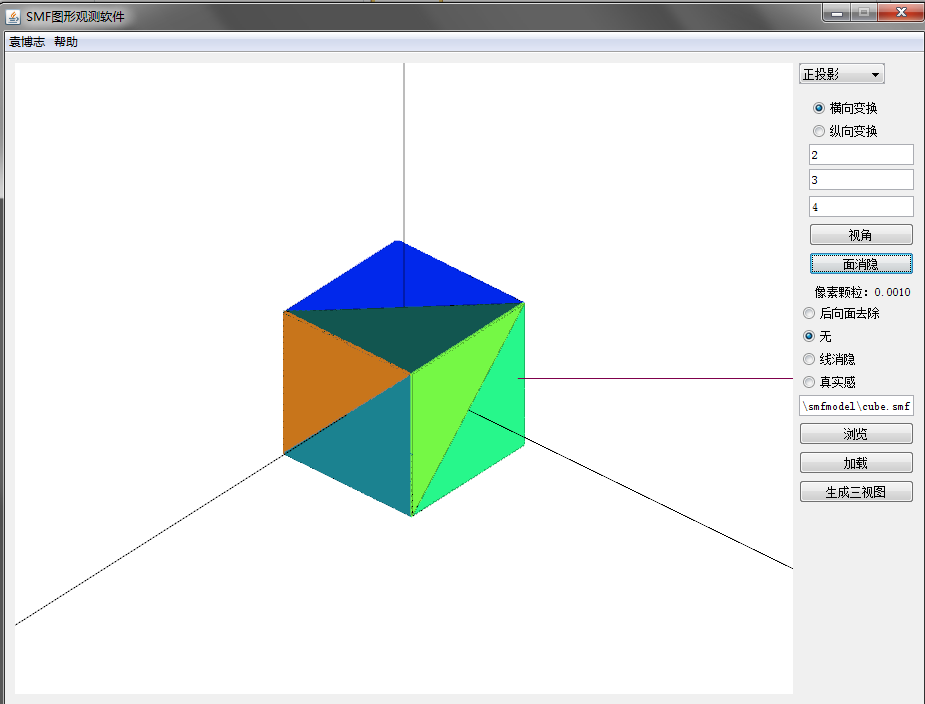


**均正确！**

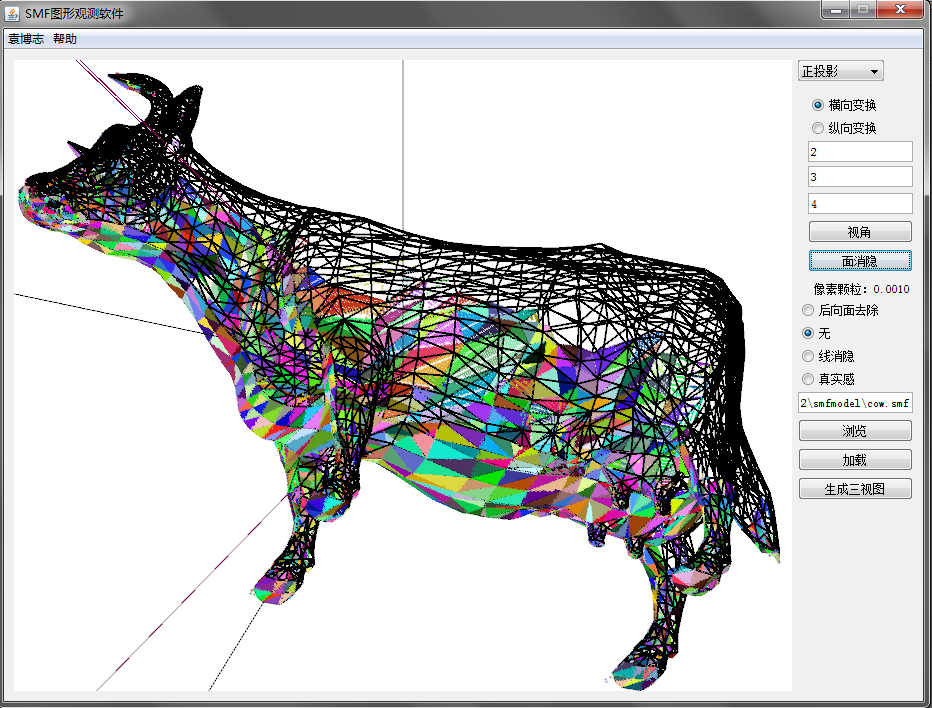
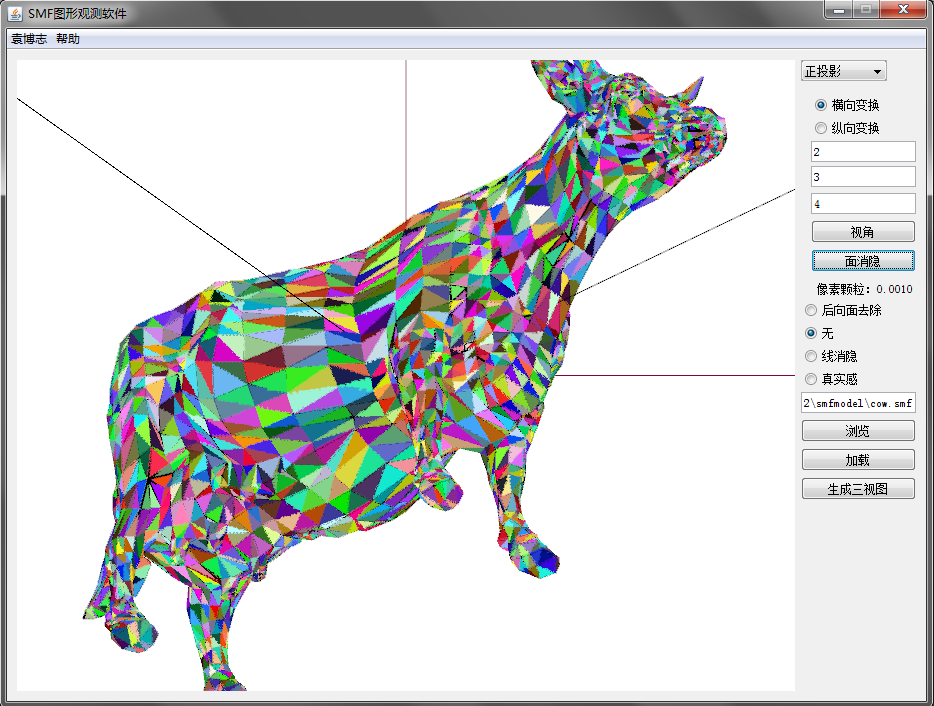
## 面消隐

### 此处仅选择比较复杂图形进行不同方位测试

cube.smf

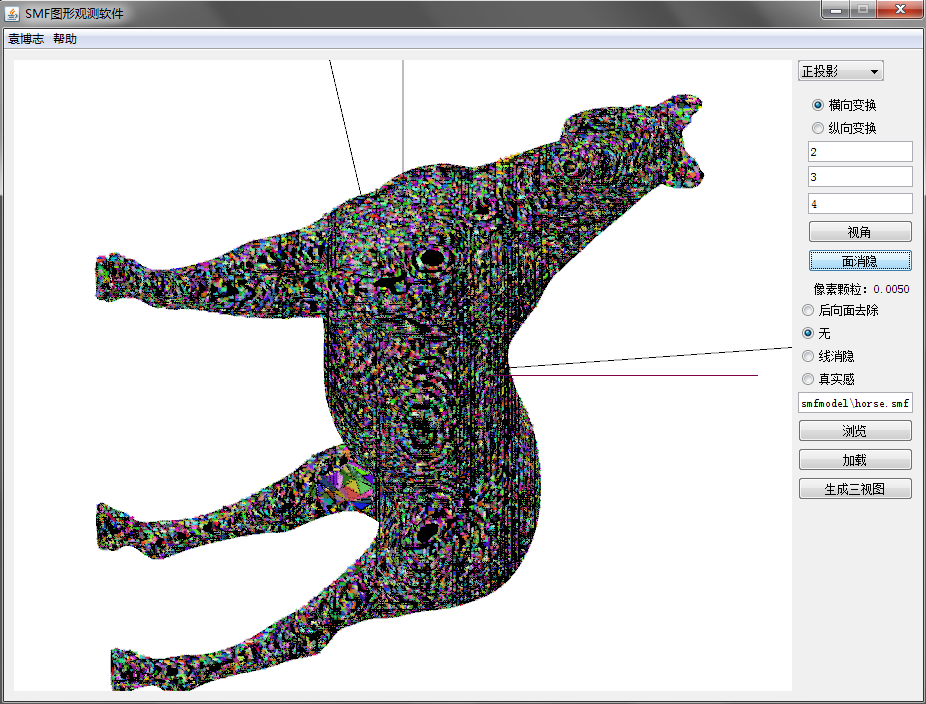
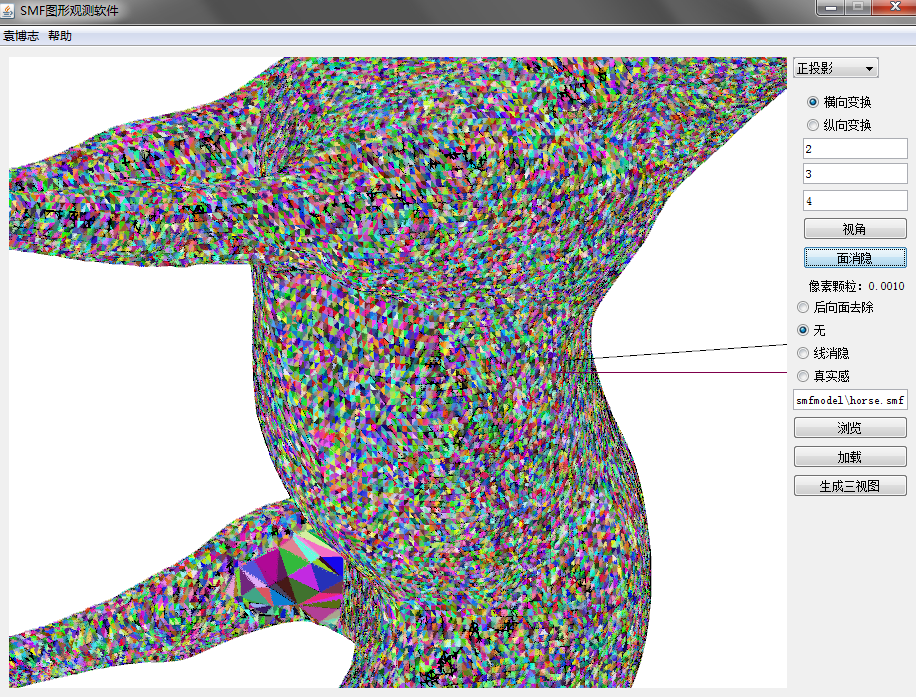


cow.smf



通过面消隐完后的图形，然后拖拽平移，可以看出染色在空间的坐标点。

horse.smf

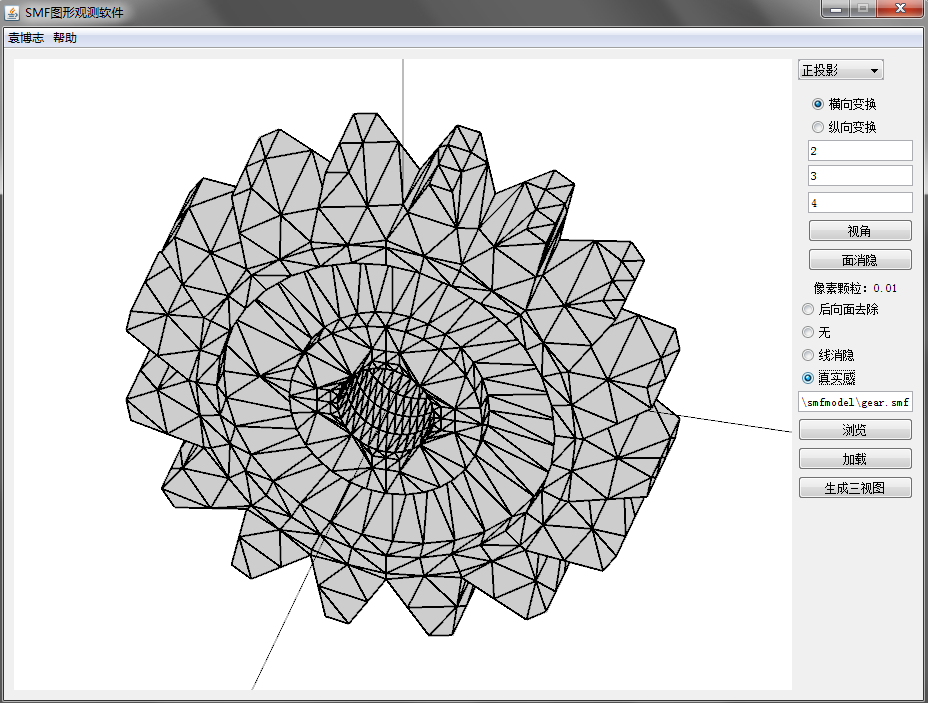
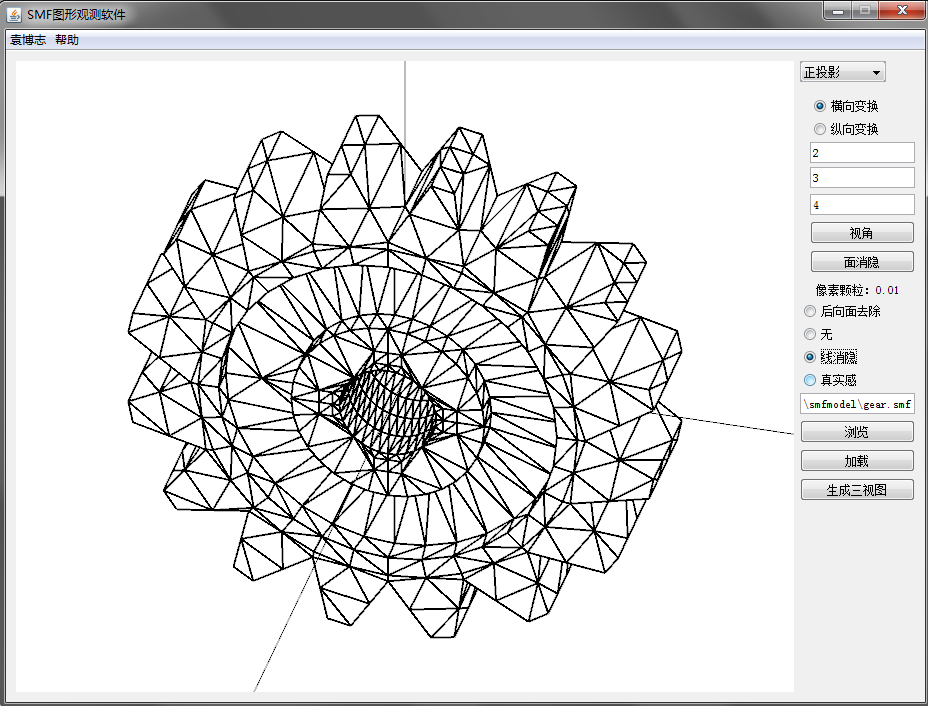
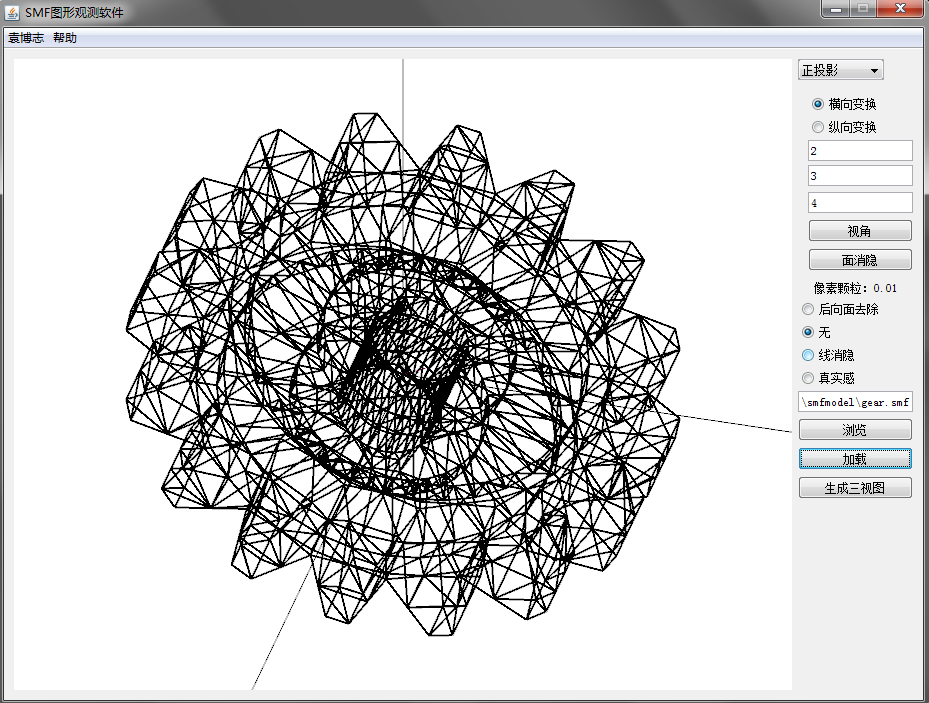
 放大后 

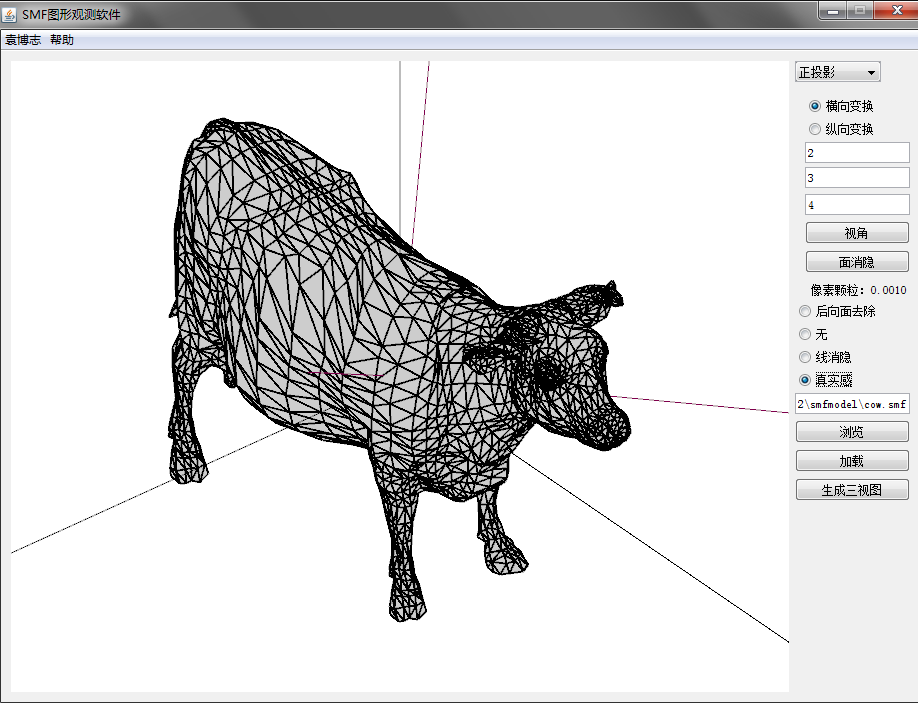
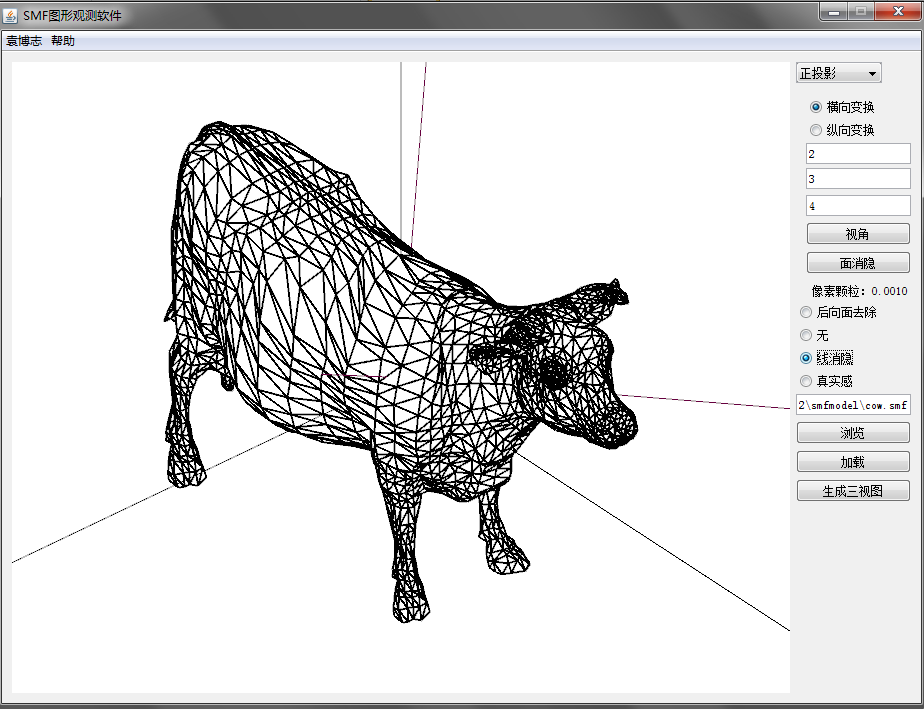
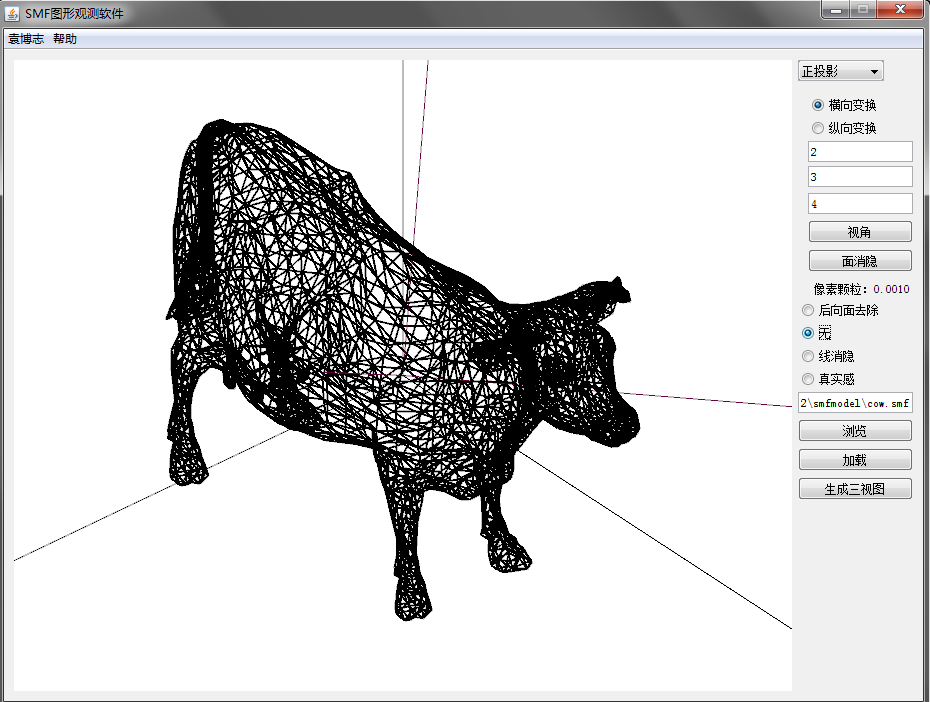
## 线消隐&真实感绘制

左：无任何操作

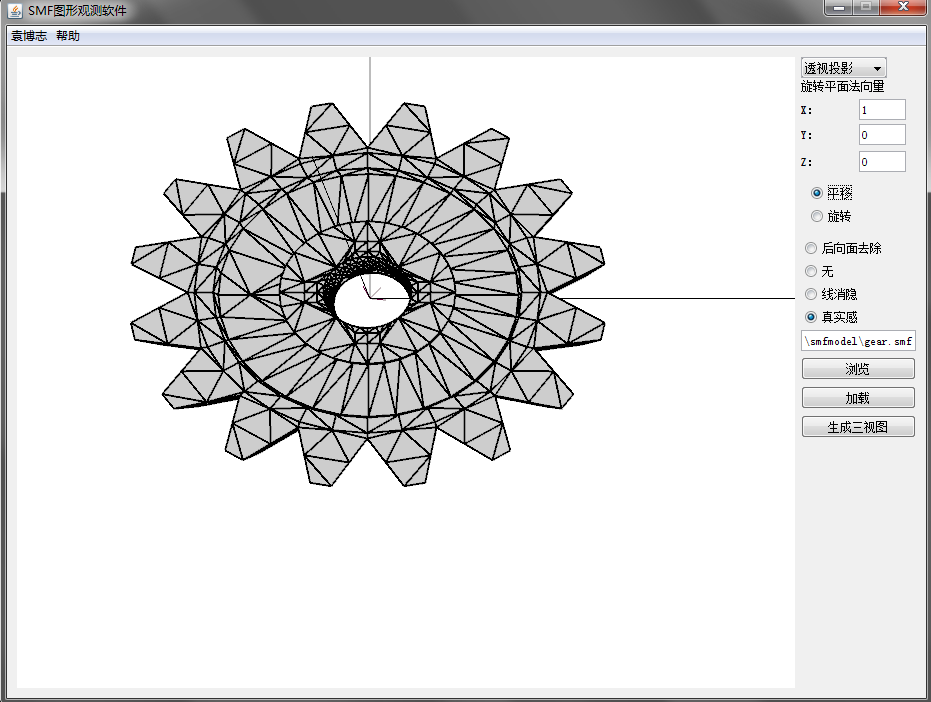
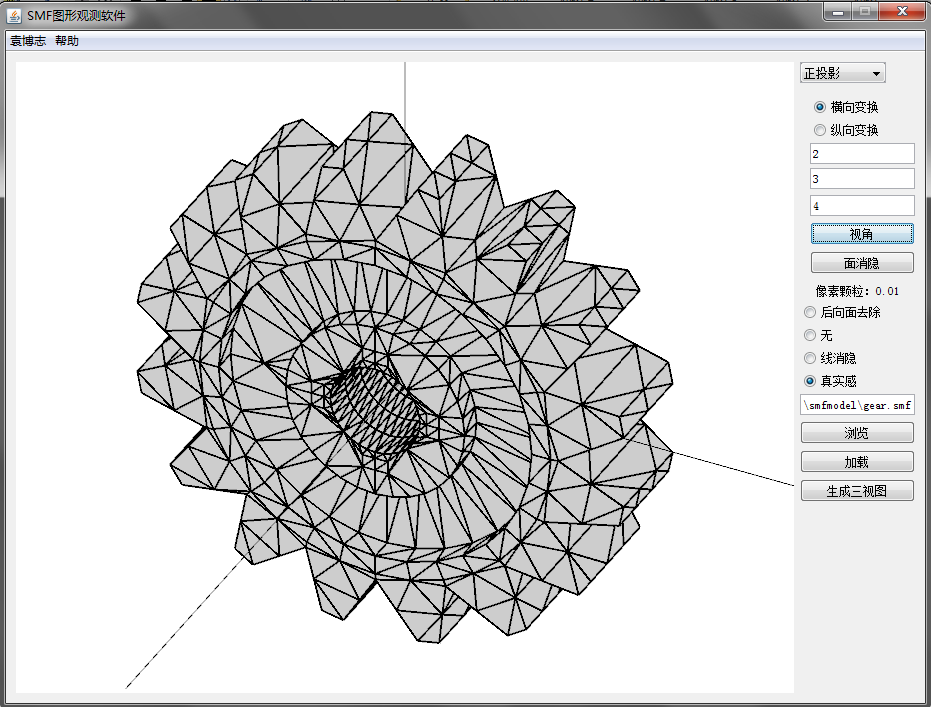
中：线消隐

右：面消隐

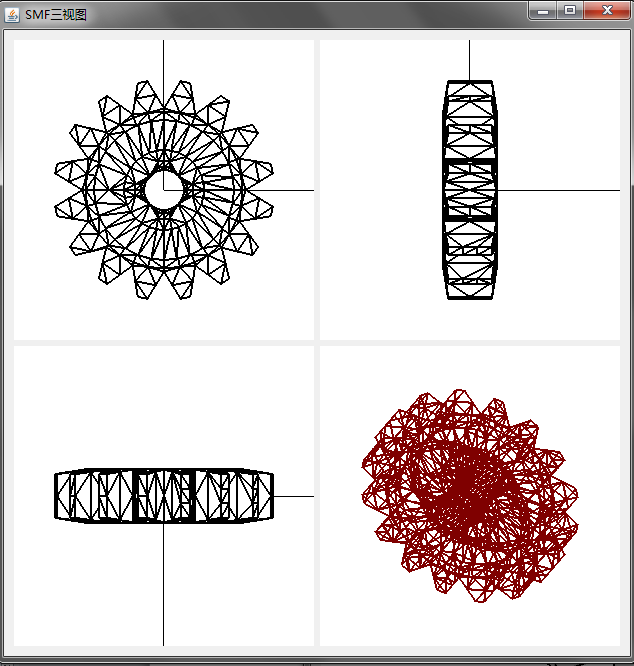




## 平行投影&透视投影



## 正平行投影和三视图



# 心得体会

在这次大作业中，使我深刻认识了坐标变换，和见识到了浮点小数带来的麻烦，因为坐标变换或很容易出现不规则小数，必须做处理。

面消隐几乎运用到了所有的坐标转换，线Z-buffer的确是个很好的办法，重要的是控制好数据结构，否则小路很低，而且很容易堆栈溢出。

空间的旋转很是麻烦。

综上，所以在今后的生活中我提高了对发明3d效果的大神们的敬仰！真心的。

本次作业绝对100%独立完成！因为加上封皮和目录还有截图，可能页数微微有点超过老师说的3页请谅解。

有事与我联系~谢谢助教或老师的批改！

# 遗留问题：

在空间坐标系中如何用鼠标确定空间任意点，且误差尽量小。

透视投影中的面消隐。有待思考。

MFC的复杂度不适合我们没学C++的人直接用来做大作业。希望今后可以越来越好。