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

通过 opengl 库理解图形的消隐和投影等一些列变换。

2. 实验平台

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

3. 实验内容

a) **实现网格模型(*.smf)的导入；**
b) **不调用图形库实现模型的面消隐；**
c) **不调用图形库实现正平行投影；**
d) **用 OpenGL 实现模型的线消隐和真实感绘制；**
e) **用 OpenGL 实现平行投影和透视投影；**

f) **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

4. 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。

1. 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

2. 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。

3. 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。

4. 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。

5. 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。

6. 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：

a) 平移：无
b) 缩放：利用滚轮。调节 glu.gluLookAt
c) 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：

d) 平移：选中平移，鼠标拖动。调节 gl.glTranslated
e) 缩放：利用滚轮。调节 glu.gluPerspective
f) 旋转：给定旋转平面的法向量，然后鼠标拖动。调节 gl.glRotated

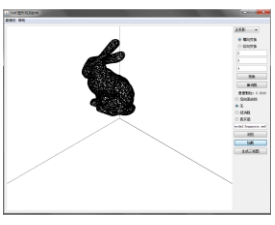
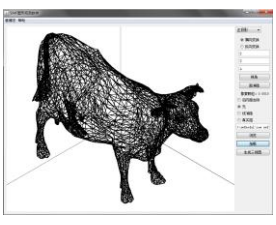
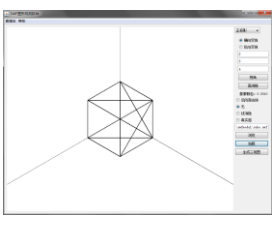
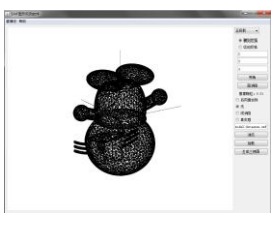
这块比较纠结，因为是三位坐标所以旋转起来的换很麻烦，所以思考了许久如何让用户交互，但最终无果，所以就采用了比较简单的 OpenGl 的库函数。

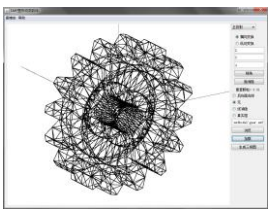
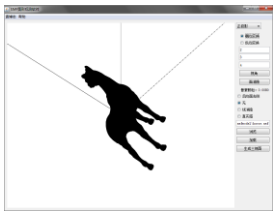
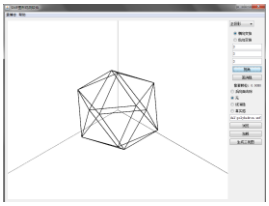
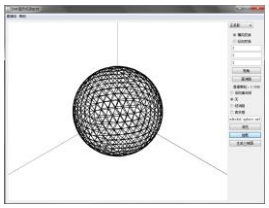
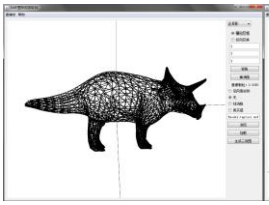
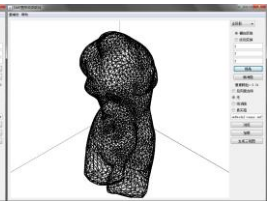
7. 生成三视图

直接利用 OpenGl 的 LookAt 函数来获取三视图。

5. 测试结果分析

1. 导入 SMF

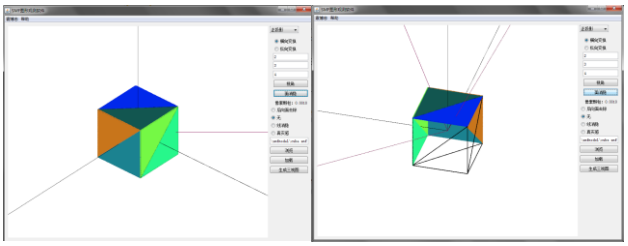







均正确！

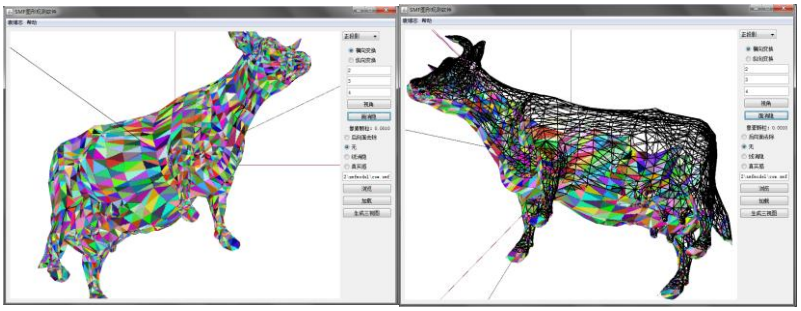
2. 面消隐

此处仅选择比较复杂图形进行不同方位测试

cube.smf

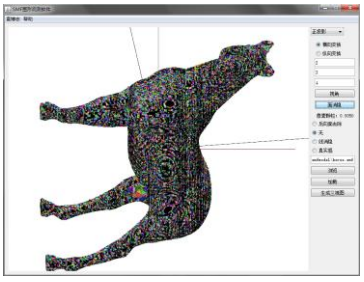
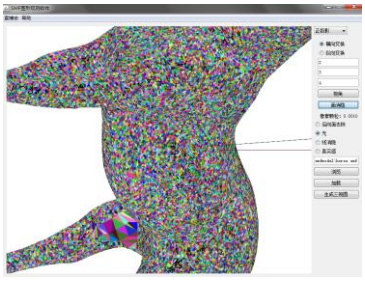


cow.smf



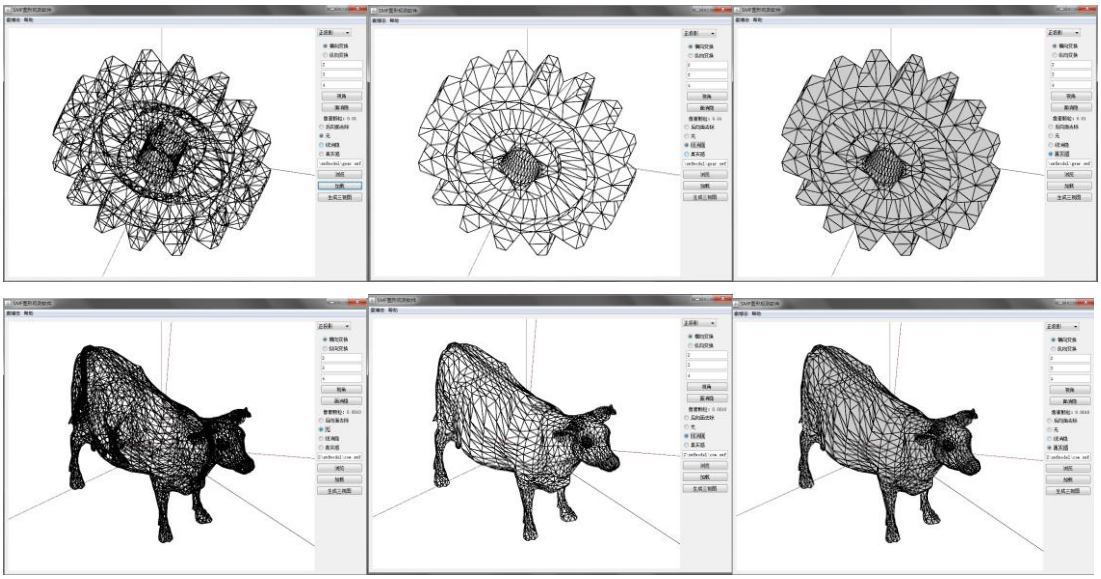
通过面消隐完后的图形，然后拖拽平移，可以看出染色在空间的坐标点。

horse.smf

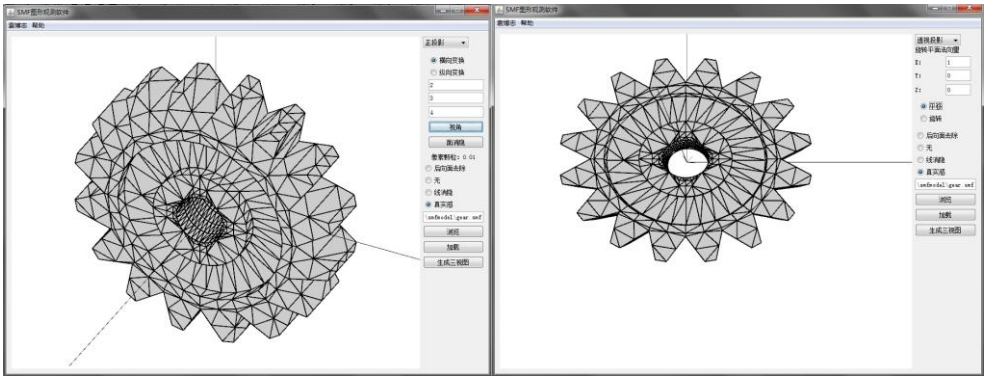

放大后


3. 线消隐&真实感绘制

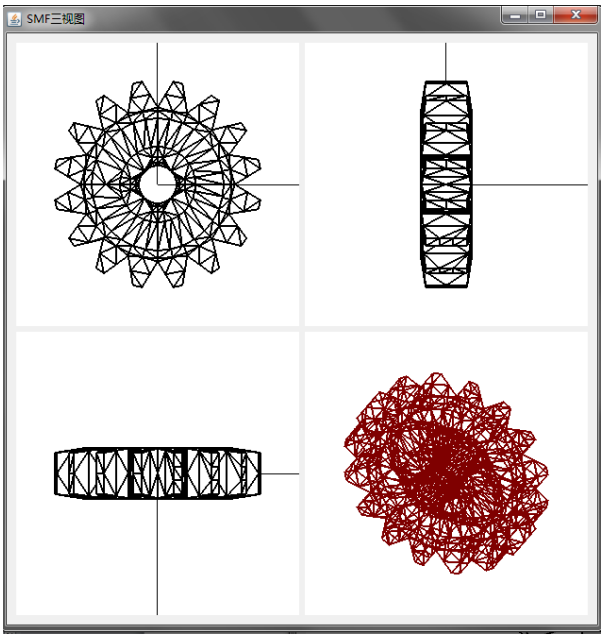
左：无任何操作
中：线消隐
右：面消隐



4. 平行投影&透视投影



5. 正平行投影和三视图



6. 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!

7. 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。