**图形学大实验实验报告**

计52 徐文昌 2005011261

1. **实验目的**
2. 能渲染一个复杂的场景，掌握基本几何体的作图方法；
3. 理解并学会运用各种基本的光照模型；
4. 能够做出纹理、反射、透明、阴影等基本效果；
5. 了解obj文件的格式及读写方法；
6. 掌握边坍塌的网格简化方法。
7. **实验内容**

**• 光线跟踪部分**

1. 基本部分
2. 基本几何体的绘制，包括立方体、球体等；
3. 基本效果的实现，包括phong光照模型、镜面效果、纹理、反射、透明、阴影等；
4. 没有使用任何OpenGL相关的库。
5. 选做部分
6. 使用mfc实现了bmp图像的直接输出；
7. 实现了光线的折射。

**• 网格简化部分**

1. 基本部分
2. 实现了边坍塌的网格简化方法；
3. 使用了Deep Exploration软件进行网格的显示；
4. 程序可指定输入输出的obj文件以及面数的简化比；
5. 实现了obj文件的读写。
6. 选做部分

使用mfc实现了图形界面。

1. **实验原理**

**• 光线跟踪部分**

1. 数据结构

由于光线追踪过程实现起来比较复杂，因此，程序中定义了一些特定的数据结构，方便算法的实现，同时也提高了程序的可读性。

主要数据结构说明如下：

1. dimension3:

定义了一个表示三维空间的类dimension3，同时定义了其在三维空间的各种运算，包括加、减、求反、数乘、内积、外积、数除、求法向量等。

程序中用到的各种三维变量，如位置、颜色空间等，均用此类型表示。

1. ray:

表示光线的类，包括光源和方向两个dimension3类型的变量。

1. 表示各种物体的类：

表示光照性质的类型materials：定义了表示颜色、漫射、反射、投射、折射等的各种变量；

表示各种物体的基类primitive：包含了光照和纹理等变量以及求法向量等函数；

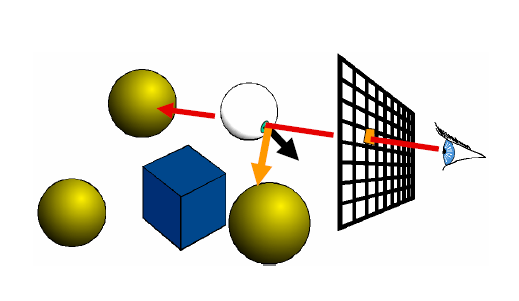
各种物体的类：包括表示球体的类sphere、表示平面的类plane和表示立方体的类cube，各种物体除了继承基类的光照和纹理特征，还包含自己特有的变量，比如球体的圆心、半径，平面的法向量，立方体各面的法向量集合等。特别要注意的是由于程序中光源定义为一个半径很小的球体，因此类sphere中有一个特别的变量m\_light用来区分光源和非光源。

1. drawing：

表示光线追踪过程的类，根据初始化的光源位置、方向已经各个物体的位置，通过循环和递归绘制图像。

1. 光线追踪过程

光线追踪的实现参照了课件上的光线追踪算法，实现过程的图解和描述如下：



光线追踪算法图解

如上图所示，大体上来说，是将光源发出的通过各种光路变化后到达人眼的光线投射到屏幕上。具体来说，首先假想一个视点(即图中眼部的位置)、一个屏幕(即图中的网格，同时也为程序中各像素点组成的图像，程序中设置的大小为800\*600)以及各物体(包括要映射到图像上的实物以及光源)，各物体的位置通过三维坐标来设置。对于图像中的每个像素，从视点向像素点引出一条射线，如果射线未碰到任何物体，那么像素值设为0，即颜色为黑色；如果碰到的第一个物体为光源，那么颜色设为白色；否则，对于碰到第一个物体(非光源)，如果交点与光源之间有非透明的物体隔着，那么该像素点设为阴影，否则使用phong模型计算出光强，然后考虑光在该点反射、折射后的光路，递归追踪该光线，要注意的是每次的光强都有衰减。

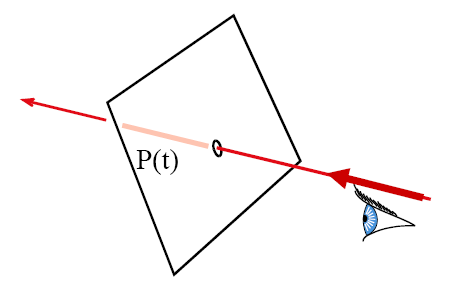
纹理是通过区分物体表面点的坐标值以设置不同的颜色来实现的。

下面简述一下phong模型以及光线与球体、平面、多面体求交的算法：

1. Phong模型

Phong光照模型分为三个累加阶段：漫反射、镜面反射和环境光。每一部分都根据各自的公式进行计算，最后将三部分的结果累加即得到了phong模型的光强。每部分的计算在这里不再详述。

1. 光线与平面求交



光线与平面求交

光射线方程为：

其中，Ro为光源坐标，Rd为光线的方向向量。

平面方程为：

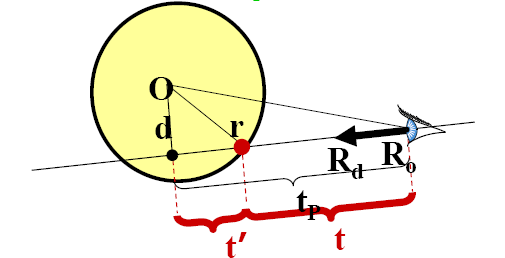
其中，P(t)为平面内点坐标，n为平面法向量。

根据上面两个公式可得，

当t > 0时，相交。

1. 光线与球体求交

采用了课件上介绍的球面求交的Geometric Method。



光线与球面求交

如图所示，如果，则不相交；如果，则相交。

通过图中的几何关系可以计算出t，然后进行判断。

1. 光线与立方体求交

光线与立方体的求交使用了自己改进后的算法，将在下面的“实验改进”中详述。

1. bmp图像的输出

本程序没有使用图像工具，而是自己实现了bmp文件的读写。bmp文件类型的各部分定义在bmpInfo.h中，bmp文件的读写函数在bmpTransfer.cpp中。

程序通过光线追踪可获得图像中各个像素的r、g、b值，通过上面定义的函数即可生成最终的bmp输出文件。

**• 网格简化部分**

1. 数据结构

为了方便编程和提高程序可读性，程序定义数据结构如下：

1. 与网格相关的信息

Point：表示顶点的类，保存了其坐标、关联三角形、二次误差测度矩阵和是否被删除等变量信息；

Edge：表示边的类，保存了关联顶点和next指针等信息；

Triangle：表示面的类，保存了顶点、基本误差二次型矩阵、是否被删除等信息；

1. Heap

实现了最小堆的数据结构，用以存储边，从而提高边插入和删除的效率。

1. mesh

提供了网格的定义和各种操作，重要的变量包括点、边、面的集合，其中边的集合以最小堆的形式存储；重要的操作包括顶点二次误差测度矩阵和面基本误差二次型矩阵的计算、边删除等函数。

1. 算法介绍

程序的实现参考了论文“Surface Simplification Using Quadric Error Metrics”中的算法。

算法的核心就是针对网格中的所有边选择代价最小的边进行简化。假设一条边的两个顶点为v1和v2，那么边的代价定义为：

即在边上找一个点v，使得它到包含边两个顶点的所有面的距离的平方和最小，而这个最小值就作为这条边的代价。

算法中要求每次选择代价最小的边进行简化，并用产生代价最小的点v来代替原来的两个顶点。但是在实际计算中，求出这个点比较困难，因此，程序中所用的方法是取边的两个顶点和中点中的最小值点作为替代的点。程序运行结果证明，这种方法还是可行的。

1. obj文件读写

根据obj文件的格式，利用文件流进行读写即可。obj文件在程序中保存在mesh类中。

obj文件的文本格式如下：

第1行为：# NV NF

其中#即ASCII码的’ #’，NV、NF都是整数，分别表示顶点和三角面的个数。

第2~NV+1行分别记录第1~NV个顶点的坐标，每行的格式为：

v x y z

其中v即ASCII码的’ v’， x y z为三个浮点数，表示该顶点的坐标。

第NV+2~NV+NF+1行记录NF个三角面的信息，每行的格式为：

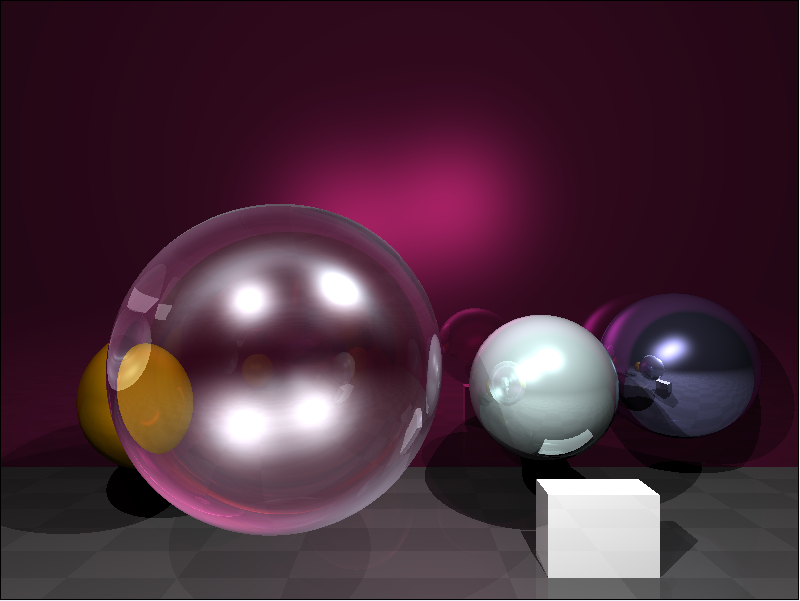
f v0 v1 v2

其中f即ASCII码的’ f’， v0 v1 v2为三个整数(1≤v0,v1,v2≤NV，注意点的编号是从1开始的)分别对应三个顶点的编号。

1. **实验效果**

**• 光线跟踪部分**

光线跟踪的效果图如下：



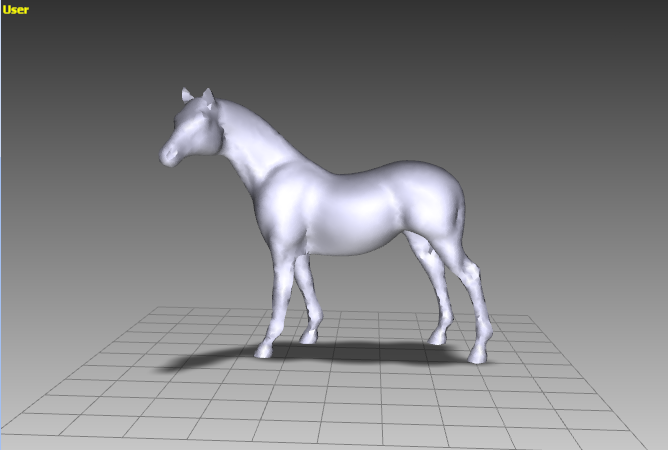
光线跟踪的效果图

**• 网格简化部分**

网格简化的效果图如下：

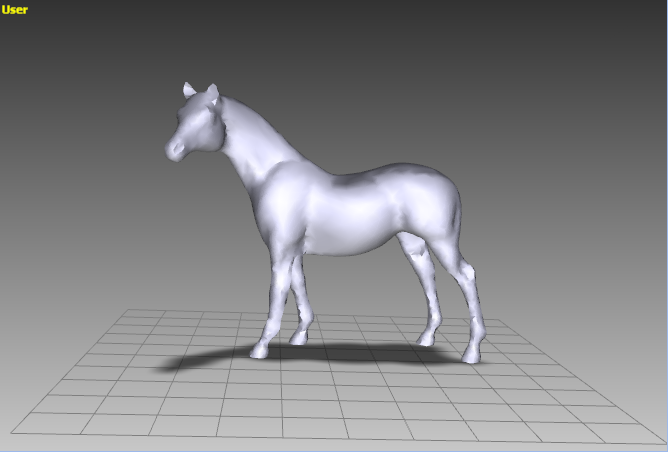
**horse.obj:**

原图：



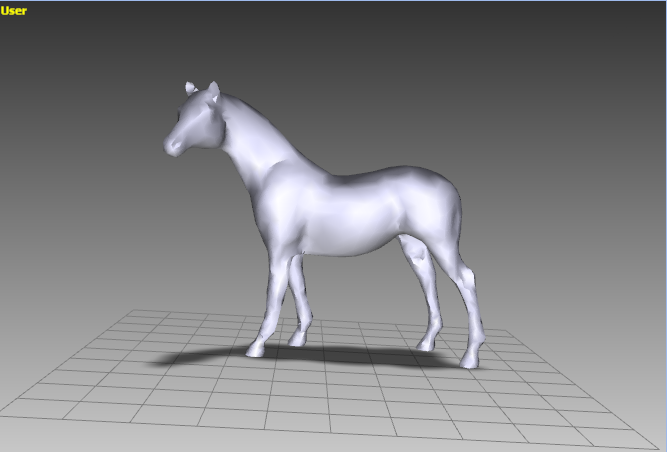
horse.obj

面数的简化比为0.6：



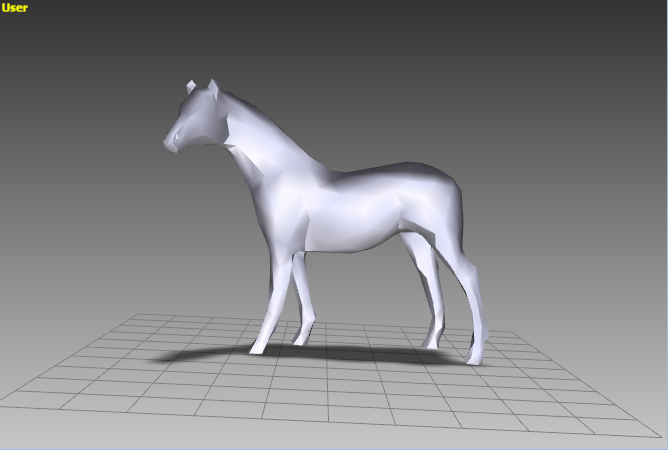
0.600000\_horse.obj

面数的简化比为0.3：



0.300000\_horse.obj

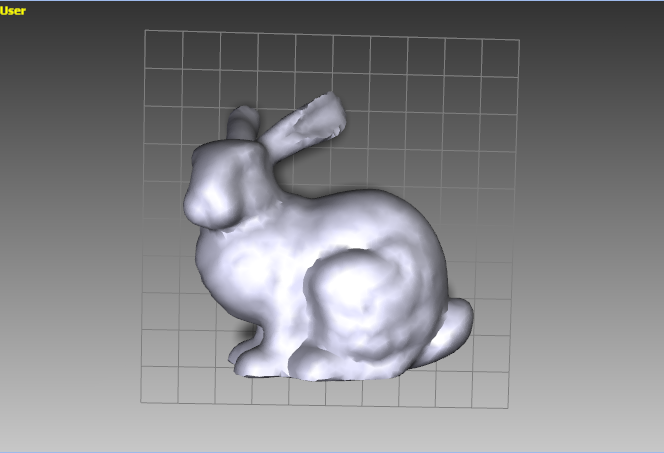
面数的简化比为0.1：



0.100000\_horse.obj

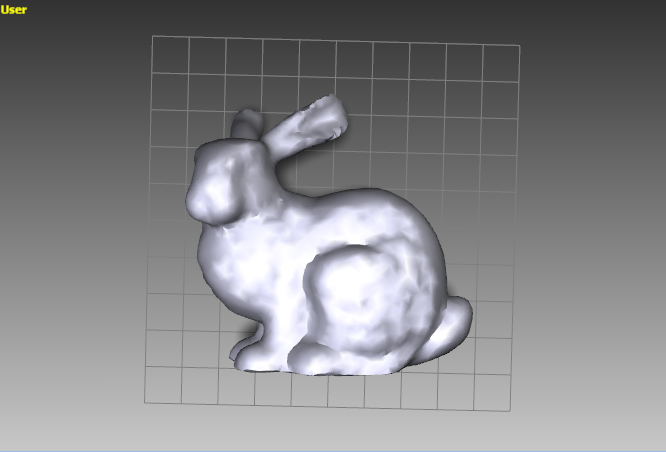
**bunny.obj:**

原图：



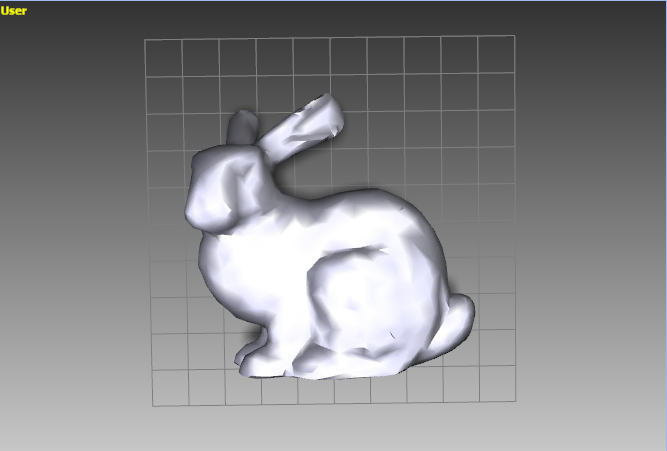
bunny.obj

面数的简化比为0.6：



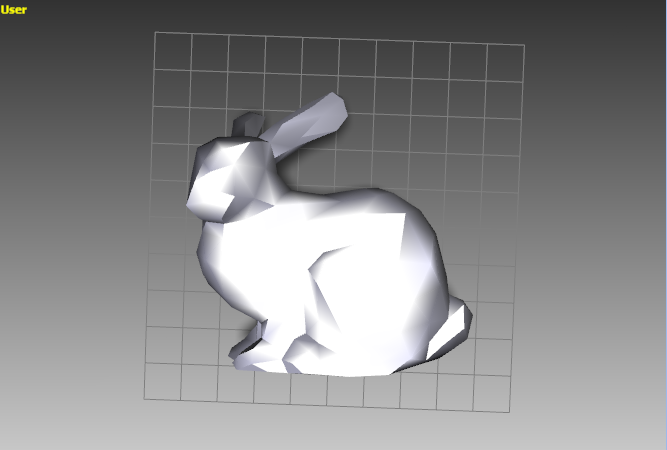
0.600000\_bunny.obj

面数的简化比为0.3：



0.300000\_bunny.obj

面数的简化比为0.1：



0.100000\_bunny.obj

1. **实验改进**
2. 对于光线与立方体求交算法的一点改进

光线与长方体求交算法的伪代码如下：

for ( i = x, y, z )

if ( Vi = 0 )

光线与长方体无交点，返回；

else

if ( Vi > 0 )

, ;

else

, ;

end if

if ()

光线与长方体无交点，返回；

end if

;

if ( )

if ()

光线与长方体无交点，返回；

end if

;

end if

if ()

;

end if

if ( )

if ()

光线与长方体无交点，返回；

end if

;

end if

end if

end for

2、在网格简化所使用的算法中，要求每次选择代价最小的边进行简化，并用产生代价最小的点v来代替原来的两个顶点。但是在实际计算中，求出这个点比较困难，即使采用某种算法可以求出来，其复杂度也会相当高，影响程序的运行速度。在本程序中，简单的取边的两个顶点和中点中的最小值点作为替代的点。这种简化的方法使寻找最优点的速度得到明显改善，由于所删除的边是确定的，那么其上各点的代价应该相差不大，因此这种简化的取点方式是可以接受的。程序运行结果证明，这种方法还是可行的。

1. **实验总结及收获**

1、对于光线追踪部分，个人感觉最关键的部分是各种求交的判断以及光线的递归追踪算法。这部分在实验原理光线追踪的第二部分已经有了详细的介绍。在程序实现过程中，碰到的一个问题是移动了图像中最左侧的黄球后，本来透明的最大的球看不到它后面的黄球。仔细检查后，发现是已经将黄球移出了光线的范围。稍稍改变了位置，又重新产生了透明效果。

2、网格简化部分的核心就是论文中给出的算法，设计合理的数据结构和实现方式，就可以比较容易的实现该算法。此部分的完成还是比较顺利的。

3、通过本次实验，掌握了光线追踪的基本思想与方法，加深了对图形学这一计算机科学分支的理解和兴趣，为今后相关的学习和研究打下了一定的基础。