图形学上机实验三

实验目的

利用ray tracing渲染模型，并与OpenGL渲染模型做对比。

实验环境

- Ubuntu 系统
- Eigen库

代码组成

本实验代码参考GitHub项目amrictor/ray-tracer。

头文件

- Camera.h；定义camera类，实现相机，定义相机位置、朝向等参数。
- Face.h；定义Face类。mesh是渲染的基本单位，包括顶点数组、索引数组、纹理和OpenGL所需各缓冲对象等成员变量；成员函数setupMesh()初始化缓冲对象，Draw()调用着色器渲染图形。
- Light.h；定义Light类
- Material.h；定义Material类
- Model.h；定义Model类
- Ray.h；定义Ray类

源文件

- Camera.cc

  实现Camera类，设置相机各项参数，实现光线追踪

- Face.cc

  实现Face类，记录顶点，计算法向量

- Light.cc

  实现光线类

- Material.cc

实现Material类，设置材质各项参数

- Model.cc

  实现Model类，读取模型，设置模型材质；接受参数构造Model矩阵

- Ray.cc

  实现Ray类

- raytracer.cc

  主函数，读取driver文件，执行模型加载和初始化，并调用函数渲染结果

资源文件

- driver.txt

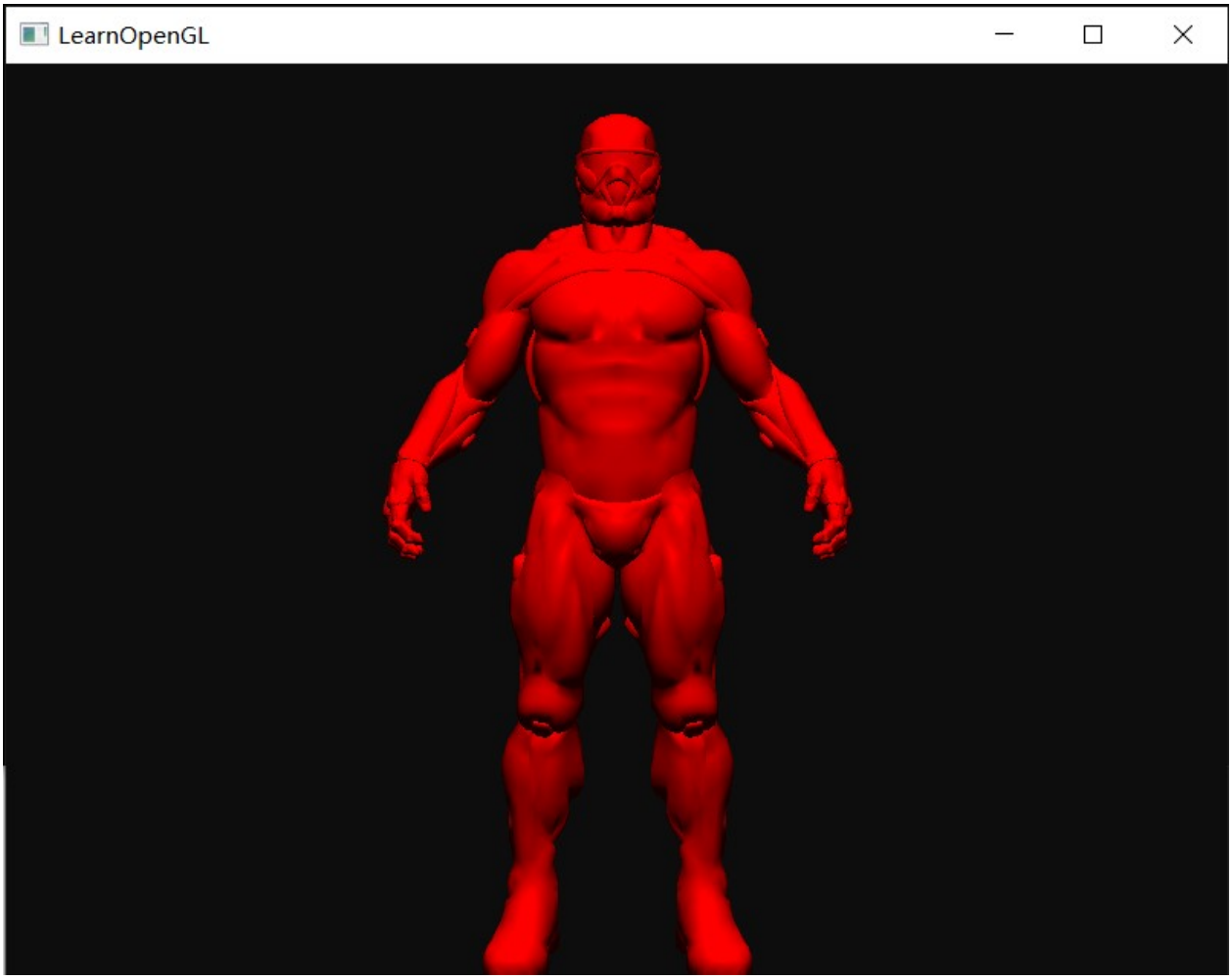
  参数设置，设置光源，模型，相机等参数。

算法描述

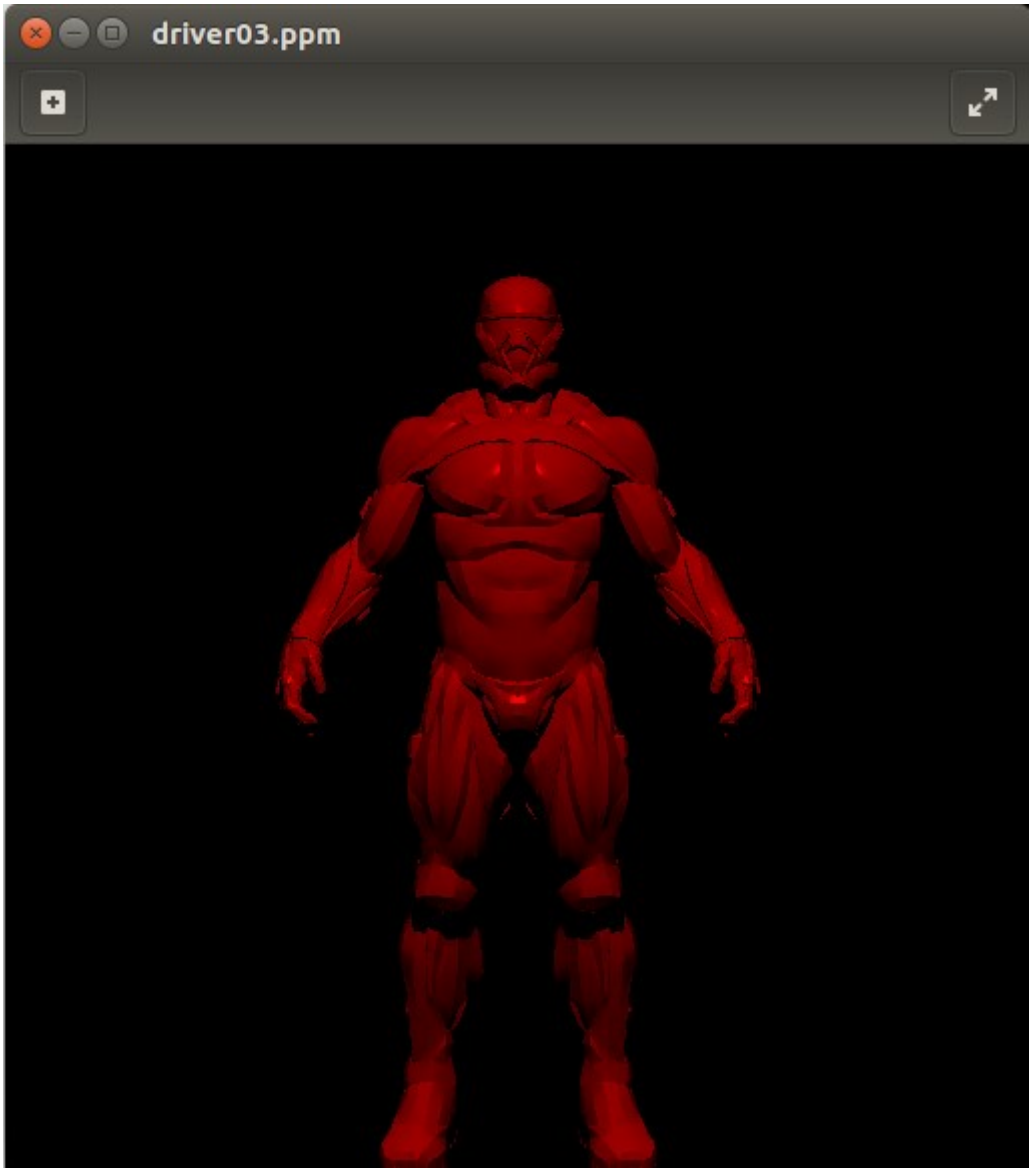
读取driver文件内的参数，初始化相机、光线和模型。遍历输入文件中的像素，依次计算光线与模型的交点，依据材质信息计算颜色，再判断反射光线是否与模型相交，如果相交，计算反射光线给当前像素带来的颜色分量；如果不相交或达到最大递归深度，返回该像素颜色。将像素颜色写入ppm图片。

实验结果

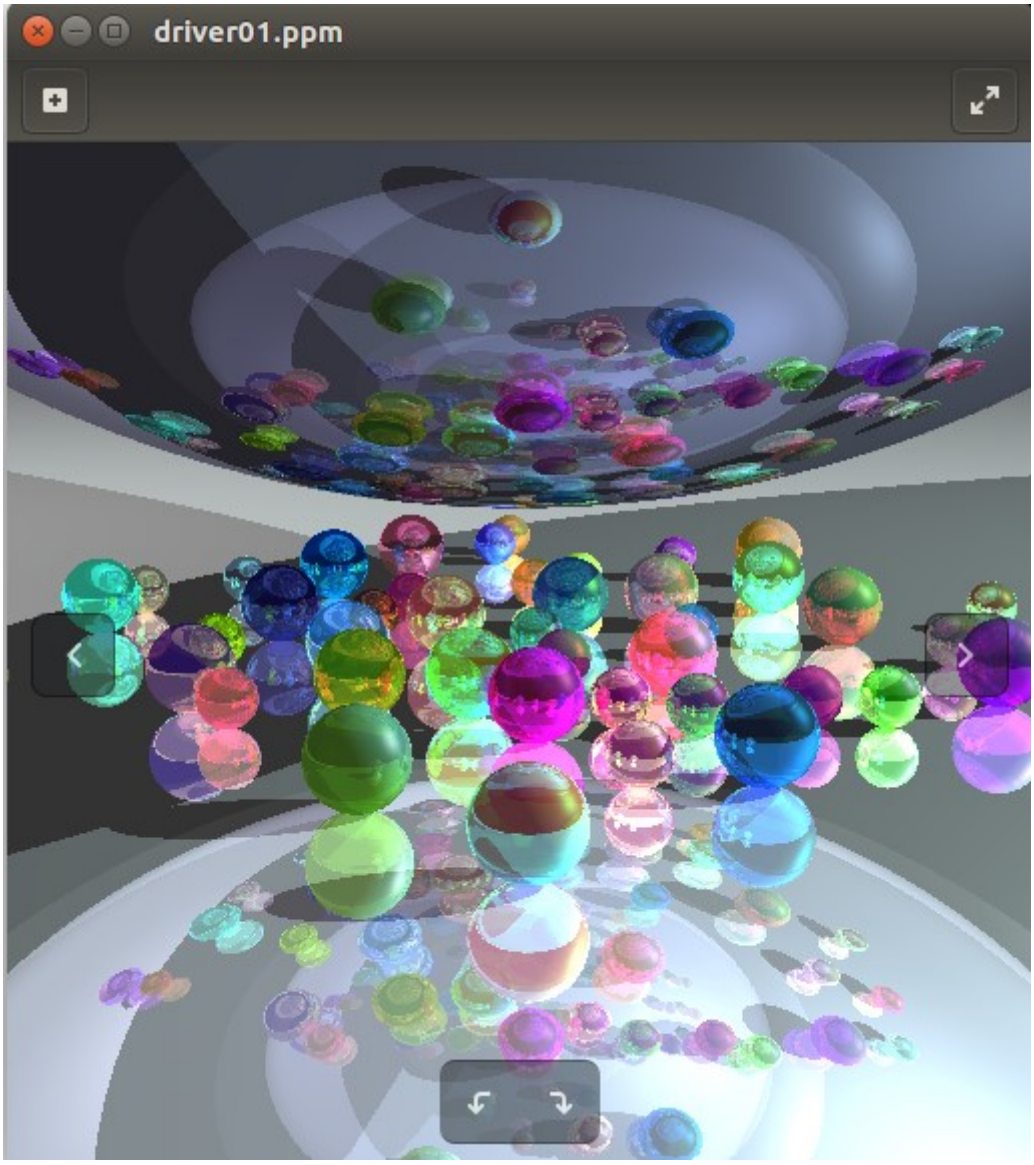
- 设置光源为红色，取消纹理贴图，光源位置为(0,5,5),环境光系数为0.1，漫反射和镜面光系数为1.0，OpenGL渲染结果如下：



光线追踪渲染结果如下：



- 使用光线追踪实现的多物体渲染：


driver01.ppm