**计算机图形学实验报告**

**内容**：

完整系统

**程序概述**：

制作一个三维图形系统，可以添加、修改、删除球、长方体、圆锥、圆柱、平面等物体，并可以用Z-Buffer，Ray-trace等方式显示，可以保存文件、添加贴图等

**完成功能列表：**

**菜单：**

1. File：
2. New…

新建文件

1. Open…

打开文件

1. Save

保存文件（注：Texture仅保存绝对路径）

二、View：

1. Line Frame

线框图显示

2. Gouroud Shading

以Gouroud Shading方式显示

3. Render…

用Raytrace方式绘制图形

4. Camera Setup…

设置Camera类型以及相关参数

三、Object：

添加、删除、修改图形

四、Light：

添加、删除、修改光源

**鼠标漫游**：

在场景中，按住鼠标左键拖动即可旋转视角或物体，按住鼠标右键拖动为平移视角或物体，滚动滚轮进行视角缩放。

具体如下：（以下坐标系均相对于视角）

1 左键/右键单击物体即选中该物体，在空白处单击取消选择。

2 选中物体的情况下左键拖动，镜头以物体为中心旋转，绕X和Y轴

3 选中物体的情况下右键拖动，镜头平移，在X-Y 平面

4 选中物体的情况下按住Ctrl + 左/右键拖动，变换轴相应变成X和Z

5 选中物体的情况下按住Shift + 左键拖动，则旋转该物体而非镜头（可与Ctrl结合使用，以切换旋转轴）

6 没有选中物体的情况下按左键拖动，旋转镜头，绕镜头位置(Location)旋转，Ctrl键意义不变

7 没有选中物体的情况下按右键拖动，平移镜头，Ctrl键意义不变

8 修改物体：双击物体弹出上述窗口可供修改

9 删除物体：选中物体，按DEL键删除

**选作内容：**

* 1. 软件ZBuffer，位于View -> Gouroud Shading
  2. 纹理贴图，在物体属性窗口中，可以将bmp, png, jpg, gif等图片贴于所有支持的5种形体上
  3. 高级光线跟踪效果，实现了半透明、反射、折射，但没有实现散射

**技术细节的实现：**

内部核心类：

1. World3D:

提供各种方法使得外界可以观察它，例如：

intersect方法取得外界于一条ray相交的最近物体，用于ray-trace和鼠标点选物体中

get\_object\_iterator方法使得外界可以遍历其中的对象

get\_light\_iterator方法可以遍历其中的光源

另外，还提供了各种方法使得外界可以修改里面的内容：

add/remove方法可以用于添加/删除物体或光源

1. Object3D:

提供了各种方法使外界可以观察、修改它（抽象类）。除去各种参数设置（比如位置、方向的set\_position方法）外，主要提供以下两个方法：

to\_triangle\_set方法返回一个TriangleSet，用于scan-convert算法绘制该图形

intersect方法取得交点

1. Light:

提供了一个方法：illuminate，参数为空间给定一点，返回光源对该点的光照强度以及光线的起始位置和方向

1. WorldViewer:

提供了一个draw方法，抽象接口，下有子类RaytraceWorldViewer, LineFrameWorldViewer, GouroudShadingWorldViewer

1. Camera:

提供了get\_normaliztion\_matrix使得世界坐标中的各种元素根据视角不同变化到标准坐标，方便处理和显示

这些接口均有一些子类实现了它们，并提供了各种各样的功能。所有核心算法均在此处实现

交界处：

Graphics2D提供了必要的绘图操作，在这里就是set\_pixel和draw\_line（仅用在LineFrame模式中)，而CDCGraphics2D用CDC实现了它

Texture 提供了纹理贴图所需的get\_color(u, v)，而CImageTexture借助CImage类实现了它

这样，整个核心实现完全与MFC隔离

MFC部分：

MFC部分负责了繁重的GUI以及Serialize工作，用了一系列Adaptor类代理了相应的核心类，并提供了EditObject（跳出窗口编辑自己），Serialize（序列化）的功能

MFC类中的一些列Dialog提供了对Dialog的访问，提供了对于EditObject的支持

CTinyGraphicsDoc存储了单个文档的所有信息，提供了AddObject，AddLight，RemoveObject，RemoveLight等函数，使得View可以轻易索取想要的adaptor（以及相应的核心类），完成了文件保存、Load的所有工作

CTinyGraphicsView负责了所有菜单、鼠标、键盘操作，主要调用其它部分相应的实现即可，虽然代码繁多，却是实现时比较顺利的一关。

其它类：

核心类中还有许多辅助类，比如Position为所有物体和Camera提供了旋转、平移功能，而更底层的Matrix4D和Vector4D等则在这些高端类地下默默工作，在上面相关的代码中却很少见。

**开发中的难点：**

1. 整体架构设计
2. 与MFC的整合
3. Rasterize Triangle（偷懒懒得用对于多边形普遍适用的Active Edge Table，结果没有Clip，使得物体离镜头很近时巨慢的速度成为现在软件的最大不足）
4. Raytrace中取得refract index，考虑了光线从某一物体穿入，再从另一物体穿入，再从第一个物体穿出的情况，由于后两次穿越间物体的材质很难定义，于是目标是即使这么乱穿，出了这两个纠缠不清的物体后，光线的refract\_factor还得正确。想到两种方法，一种是这要求在光线射出，且栈顶对象不是当前相交对象时，一直退栈到现在的交点被包含在栈顶物体内部为止， 本程序采用了这种方法；另一种是在从第一个物体传出时在栈中搜索，将第一个物体从中删除。两者都仅仅解决了部分问题，对于两个物体相切处出去的光线，可能结果仍然会不对，但前者在下次进行折射出时还会修正（还有下次折射出？？），便选择了前者。

**项目总结：**

该程序基本上完成了计算机图形学算法完整系统的基本功能，而Raytrace如此眩目的效果仅能在短短几十行代码中实现也令我大吃一惊。前期对于核形类的编写，使我对于程序框架的设计有了一些新的理解，并且使得对于很多以前只理解了一半的图形学算法有了比较近距离的感觉，比如以前只是知道ray-trace慢，scan-convert快，ray-trace精致，scan-convert粗糙，但书上的那些图片毕竟还是感觉非常遥远。在自己亲自做了之后，还知道了某些“好的算法”和旁边那个看上去差不多的算法相比为什么好等。而后期对于MFC部分的写作更是使得我对MFC的理解大为提高，理解了MFC独特的对象生存周期观，以及MFC奇妙的DYNAMIC\_CREATION功能，对MFC中的Document-View架构有了初步的理解，对使用MFC的恐惧感也减少了不少，在独立完成作业的过程中，增强了现学现卖和独立解决困难的能力。

由于时间仓促，该程序还有许多不足之处，比如对于Texture的保存，以及由于没有做Clip而导致物体离Camera很近时计算速度极慢等。由于对这个项目的喜爱，今后可能会继续完善。

**参考资料**

基于java的计算机图形学 〔日〕青野雅树 著 张文乐 译

ISBN 7-03-012815-X

Computer Graphics Principles and Practice （美）James D.Foley 等著

ISBN 7-111-13026-2

Procedural Elements for Computer Graphics (美) David F.Rogers著

ISBN 7-111-09595-2

深入浅出MFC 2nd ed. 侯俊杰 著

ISBN 7-900614-93-1

Programming Windows with MFC / Jeff Prosise. -- 2nd ed.

ISBN 1-57231-695-0

MSDN Library Copyright 1987-2003 Microsoft Corporation