独创性声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：贾希　　日期：2010.3.10

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在__年解密后适用本授权书。非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：贾希　　日期：2010.3.10

导师签名：任建华　　日期：2010.3.10

三维物体亮度信息计算软件

摘 要

三维可视化技术正以迅猛速度发展，它应用非常广泛，尤其在图像仿真方向。本文第一章简要介绍了三维物体亮度信息计算软件的研究意义，内容，方法；第二章结合软件功能，介绍背景知识OpenGL(Open Graphic Library)开发库和 MFC(Microsoft Foundation Class Library)开发库的组织结构和使用方法，并且结合软件实现，介绍光反射和光照度学相关光学知识内容；第三章提出了该软件整体需求分析，并分析得出具体软件模块需求，如 MFC 框架，输入输出模块，显示模块，数据计算模块；第四章重点阐述了该软件的整体设计流程，具体模块设计以及实现过程中关键的技术，列举了主要功能模块，比如显示模块的框架组成和关键代码，数据计算模块的数据文件格式，三维图像坐标变换策略，以及 Z-Buffer 消隐算法；第五章通过软件实际运行，得出可视化结果以及相应数值信息，并对该软件提出了进一步要求。

关键词： 开放图像开发库 Z-BUFFER 算法 光反射学 MFC

THE LIGHTNESS INFORMATION CALCULATION SOFTWARE ON THREE-DIMENSIONAL OBJECT

ABSTRACT

Three-dimensional visualization technology is at a rapid pace, It is widely used, especially in the direction of the visualization simulation. This paper the first chapter briefly described on the research significance, research content and research methods on the lightness information calculation software on three-dimensional object. According to the software features, Chapter 2 introduced OpenGL development libraries and MFC development library's organizational structure and the methods of use, It also introduced the related background knowledge of light reflection and the illumination studies; Chapter 3 made the overall demand for software features, and made a analysis of that software specific module needs like: the framework of MFC, input/output module, display module, data calculation module; Chapter 4 importantly focused on the whole process of the software design, a specific module design and implementation process in the key technology, citing the main functional modules, such as framework of the display module and the key code, data file format of the data module, three-dimensional object coordinate transformation strategy, as well as the Z-Buffer hidden surface removal algorithm; In the end of the paper, through the software actual operation, Chapter 5 obtained visualizational results and the corresponding numerical information, and made a further request on the software.

KEY WORDS: OpenGL Z-buffer algorithm Light reflection MFC

目录

第一章 绪论 1
1.1 研究意义 1
1.2 研究内容 1
1.3 研究方法和工具 2
1.4 论文组织结构 2
第二章 相关背景介绍 3
2.1 OpenGL 介绍 3
2.2 MFC 介绍 6
2.3 光反射理论 12
2.4 光照度学理论 13
第三章 软件需求设计 20
3.1 软件需求分析 20
3.2 软件设计流程 21
第四章 软件模块设计 24
4.1 MFC 框架设计 24
4.1.1 控制面板窗体设计 25
4.1.2 显示窗体设计 26
4.2 输入输出模块设计 26
4.2.1 输入模块设计 26
4.2.2 输出模块设计 29
4.3 显示模块设计 30
4.3.1 OpenGL 相关开发库 30
4.3.2 CView 类 OpenGL 实现 35
4.4 数据计算模块设计 40
4.4.1 数据文件读取 40
4.4.2 三维坐标变换 41
4.4.3 消隐算法（Z-Buffer 算法） 43
第五章 结论分析 48
5.1 软件计算结果比较 48
5.2 结论分析 52
参考文献 54
致谢 55
攻读学位期间发表的学术论文目录 56

第一章 绪论

1.1 研究意义

我们生活在一个充满三维物体的三维世界中，为了使计算机能精确地再现这些物体，我们必须能在三维空间描绘这些物体。我们又生活在一个充满信息的世界中，能否尽快地理解并运用这些信息将直接影响事业的成败，所以我们需要用一种最直接的形式来表示这些信息。

最近几年计算机图形学的发展使得三维表现技术得以形成，这些三维表现技术使我们能够再现三维世界中的物体，能够用三维形体来表示复杂的信息，这种技术就是可视化技术。可视化技术使人能够在三维图形世界中直接对具有形体的信息进行操作，和计算机直接交流。这种技术已经把人和机器的力量以一种直觉而自然的方式加以统一，这种革命性的变化无疑将极大地提高人们的工作效率。可视化技术赋予人们一种仿真的、三维的并且具有实时交互的能力，这样人们可以在三维图形世界中用以前不可想象的手段来获取信息或发挥自己创造性的思维。机械工程师可以从二维平面图中得以解放直接进入三维世界，从而很快得到自己设计的三维机械零件模型。医生可以从病人的三维扫描图象分析病人的病灶。军事指挥员可以面对用三维图形技术生成的战场地形，指挥具有真实感的三维飞机、军舰、坦克向目标开进并分析战斗方案的效果。

仿真(emulation)即用一个系统来模仿另一个系统。例如利用程序设计技术和某些特殊的机器特点，使一个计算系统能执行另一个计算系统的程序的方法或过程。我们研究的项目主要集中于用可视化仿真三维物体亮度信息分布以及数值计算；

1.2 研究内容

本课题主要内容是仿真计算三维物体亮度信息分布。研究内容有如下几点：软件需求分析，软件整体设计，软件具体模块设计等。

本论文主要完成的工作有：

（1）充分调研和研究了该软件相关背景知识，如光反射理论，光照度学，OpenGL 三维图形建模知识；

（2）对该软件进行整体需求分析，对具体模块进行需求分析，并设计具体

模块开发策略；

（3）对于 OpenGL 图像开发技术进行深入研究以及实践，并结合使用 VC6.0 开发环境，使用 C++开发语言，开发程序得出结果展示；

（4）对于消隐算法进行深入研究分析，并运用工程软件得出计算结果；

1.3 研究方法和工具

论文工作采用如下研究思路：分析问题，发现问题，寻找解决方案，通过结果对比来检验效果并得出结论。

首先引入 VC6.0 开发工具，以及 OpenGL 开发库进行仿真实现。

然后分析软件整体功能需求，具体模块需求，进行相应软件模块设计，针对不同模块功能，进行模块功能分析，进行功能设计；得到不同场景下结果展示，并得出结果比较结论；

1.4 论文组织结构

全文由 5 章组成，组织结构安排如下：

第一章绪论部分，论述了课题的研究意义、论文内容以及研究方法；

第二章介绍了 OpenGL 相关背景知识，发展历程，以及 OpenGL 具体功能；介绍了 MFC 框架应用背景，MFC 具体框架组成以及不同模块的功能；介绍了光反射理论，对不同反射进行了介绍；介绍了光照度学理论，应用知识，区分了不同的光照度学概念；

第三章给出了该软件整体的功能需求，针对整体功能需求对不同模块提出具体需求，并给出了该软件设计流程和具体模块设计要求；

第四章具体分析了软件模块设计。MFC 框架设计，输入输出模块设计，显示模块设计，数据计算模块所采取的具体设计方法分析以及具体设计策略；

第五章展现了该软件不同光学场景下结果比较，以及得出结论分析，并且提出了该软件待改进的地方；

第二章　相关背景介绍

2.1　OpenGL 介绍

OpenGL[1]是行业领域中最为广泛接纳的 2D/3D 图形 API, 其自诞生至今已催生了各种计算机平台及设备上的数千优秀应用程序。OpenGL 是独立于视窗操作系统或其它操作系统的，亦是网络透明的。在包含 CAD、内容创作、能源、娱乐、游戏开发、制造业、制药业及虚拟现实等行业领域 中，OpenGL 帮助程序员实现在 PC、工作站、超级计算机等硬件设备上的高性能、极具冲击力的高视觉表现力图形处理软件的开发。

OpenGL 的前身是 SGI 公司为其图形工作站开发的 IRIS GL。IRIS GL 是一个工业标准的 3D 图形软件接口，功能虽然强大但是移植性不好，于是 SGI 公司便在 IRIS GL 的基础上开发了 OpenGL。OpenGL 的英文全称是“Open Graphics Library”，顾名思义，OpenGL 便是“开放的图形程序接口”。虽然DirectX在家用市场全面领先，但在专业高端绘图领域，OpenGL 是不能被取代的主角。

OpenGL 是个与硬件无关的软件接口，可以在不同的平台如Windows 95、Windows NT、Unix、Linux、MacOS、OS / 2 之间进行移植。因此，支持 OpenGL 的软件具有很好的移植性，可以获得非常广泛的应用。由于 OpenGL 是图形的底层图形库，没有提供几何实体图元，不能直接用以描述场景。但是，通过一些转换程序，可以很方便地将AutoCAD、3DS/3DSMAX 等 3D 图形设计软件制作的 DXF 和 3DS 模型文件转换成 OpenGL 的顶点数组。

在 OpenGL 的基础上还有 Open Inventor、Cosmo3D、Optimizer 等多种高级图形库，适应不同应用。其中，Open Inventor 应用最为广泛。该软件是基于 OpenGL 面向对象的工具包，提供创建交互式 3D 图形应用程序的对象和方法，提供了预定义的对象和用于交互的事件处理模块，创建和编辑 3D 场景的高级应用程序单元，有打印对象和用其它图形格式交换数据的能力。

OpenGL 的发展一直处于一种较为迟缓的态势，每次版本的提高新增的技术很少，大多只是对其中部分做出修改和完善。1992 年 7 月，SGI 公司发布了 OpenGL 的 1.0 版本，随后又与微软公司共同开发了 Windows NT 版本的 OpenGL，从而使一些原来必须在高档图形工作站上运行的大型 3D 图形处理软件也可以在微机上运用。1995 年 OpenGL 的 1.1 版本面市，该版本比 1.0 的性能有许多提高，并加入了一些新的功能。其中包括改进打印机支持，在增强元文件中包含

OpenGL 的调用，顶点数组的新特性，提高顶点位置、法线、颜色、色彩指数、纹理坐标、多边形边缘标识的传输速度，引入了新的纹理特性等等。OpenGL 1.5 又新增了"OpenGL Shading Language"，该语言是"OpenGL 2.0"的底核，用于着色对象、顶点着色以及片断着色技术的扩展功能。

OpenGL 2.0 标准的主要制订者并非原来的 SGI，而是逐渐在 ARB 中占据主动地位的 3DLabs。2.0 版本首先要做的是与旧版本之间的完整兼容性，同时在顶点 与像素及内存管理上与 DirectX 共同合作以维持均势。OpenGL 2.0 将由 OpenGL 1.3 的现有功能加上与之完全兼容的新功能所组成。借此可以对在 ARB 停滞不前时代各家推出的各种纠缠不清的扩展指令集做一次彻底的精简。此外，硬件可编程能力的实现也提供了一个更好的方法以整合现有的扩展指令。

目前，随着 DirectX 的不断发展和完善，OpenGL 的优势逐渐丧失，至今虽然已有 3Dlabs 提倡开发的 2.0 版本面世，在其中加入了很多类似于 DirectX 中可编程单元的设计，但厂商的用户的认知程度并不高，未来的 OpenGL 发展前景迷茫。

OpenGL 的发展历程：

1992 年 7 月，SGI 公司发布了 OpenGL 的 1.0 版本，随后又与微软公司共同开发了 Windows NT 版本的 OpenGL，从而使一些原来必须在高档图形工作站上运行的大型 3D 图形处理软件也可以在微机上运用。

1995 年 OpenGL 的 1.1 版本面市，该版本较 1.0 性能提高许多，并加入了一些新的功能。包括提高顶点位置、法线、颜色、色彩指数、纹理坐标、多边形边缘标识的传输速度，引入了新的纹理特性等等。

1997 年，Windows 95 下 3D 游戏的大量涌现，游戏开发公司迫切需要一个功能强大、兼容性好的 3D 图形接口，而当时微软公司自己的 3D 图形接口 DirectX 3.0 功能却是很糟糕。因而以制作《雷神之锤》 等经典 3D 射击游戏而著名的 id 公司同其它一些游戏开发公司一同强烈要求微软在 Windows 95 中加入对 OpenGL 的支持。微软公司最终在 Windows 95 的 OSR2 版和后来的 Windows 版本中加入了对 OpenGL 的支持。这样，不但许多支持 OpenGL 的电脑 3D 游戏得到广泛应用，而且许多在 3D 图形设计软件也可以运用支持 OpenGL 标准的3D 加速卡，大大提高其 3D 图形的处理速度。

2003 年的 7 月 28 日，SGI 和 ARB 公布了 OpenGL 1.5。OpenGL 1.5 中包括 OpenGL ARB 的正式扩展规格绘制语言"OpenGL Shading Language"。OpenGL 1.5 的新功包括：顶点 Buffer Object、Shadow 功能、隐蔽查询、非乘方纹理等。

2004 年 8 月，OpenGL2.0 版本发布~OpenGL 2.0 标准的主要制订者并非原来的 SGI，而是逐渐在 ARB 中占据主动地位的 3Dlabs。opengl2.0 支持 OpenGL

Shading Language、新的 shader 扩展特性以及其他多项增强特性。

2008 年 8 月初 Khronos 工作组在 Siggraph 2008 大会上宣布了 OpenGL 3.0 图形接口规范，GLSL1.30 shader 语言和其他新增功能将再次未来开放 3D 接口发展指明方向。

OpenGL 3.0 API 开发代号为 Longs Peak，和以往一样，OpenGL 3.0 仍然作为一个开放性和跨平台的 3D 图形接口标准，在 Shader 语言盛行的今天，OGL3.0 增加了新版本的 shader 语言：GLSL 1.30,可以充分发挥当前可编程图形硬件的潜能。同时，OGL3.0 还引入了一些新的功能，例如顶点矩阵对象，全帧缓存对象功能，32bit 浮点纹理和 渲染缓存，基于阻塞队列的条件渲染，紧凑行半浮点顶点和像素数据，四个新压缩机制等等。

2009 年 3 月又公布了升级版新规范 OpenGL 3.1，也是这套跨平台免费 API 有史以来的第九次更新。OpenGL 3.1 将此前引入的 OpenGL 着色语言“GLSL”从 1.30 版升级到了 1.40 版，通过改进程序增强了对最新可编程图形硬件的访问，还有更高效的顶点 处理、扩展的纹理功能、更弹性的缓冲管理等等。宽泛地讲，OpenGL 3.1 在 3.0 版的基础上对整个 API 模型体系进行了简化，可大幅提高软件开发效率。

2009 年 8 月 Khronos 小组发布了 OpenGL 3.2，这是一年以来 OpenGL 进行的第三次重要升级。该版本仍然延续了 OpenGL 发展的方向让图形程序开发者能在多种操作系统和平台下更好的利用新 的 GPU 功能。OpenGL3.2 版本提升了性能表现、改进了视觉质量、提高了几何图形处理速度，而且使 Direct3D 程序更容易移植为 OpenGL。 除 OpenGL 之外，Khronos 还将其开发的其它标准进行了协调改进，以求可以在更广泛的领域提供强大的图形功能和计算生态系统，这些标准包括用于并 行计算的 OpenCL、用于移动 3D 图形开发的 OpenGL ES 和用于网络 3D 开发的 WebGL。

Khronos 旗下的 OpenGL ARB(Architecture Review Board)工作组推出了 GLSL 1.5OpenGLShading Language(OpenGL 着色语言)的升级版，以及在 OpenGL3.2 框架下推出了两个新功能，可以让开发者在开发新程序时能够在使用流水线内核 特性或兼容性特性之间做出选择，其中兼容性特性会提供与旧版 OpenGL 之间的兼容性。

OpenGL（全称 Open Graphics Library）是个定义了一个跨编程语言、跨平台的编程接口的规格，它用于三维图象（二维的亦可）。OpenGL 是个专业的图形程序接口，是一个功能强大，调用方便的底层图形库。

OpenGL 是一个开放的三维图形软件包，它独立于窗口系统和操作系统，以它为基础开发的应用程序可以十分方便地在各种平台间移植；OpenGL 可以与

Visual C++紧密接口，便于实现机械手的有关计算和图形算法，可保证算法的正确性和可靠性；OpenGL 使用简便，效率高。它具有七大功能[2]：

1.建模：OpenGL 图形库除了提供基本的点、线、多边形的绘制函数外，还提供了复杂的三维物体（球、锥、多面体、茶壶等）以及复杂曲线和曲面绘制函数。

2.变换：OpenGL 图形库的变换包括基本变换和投影变换。基本变换有平移、旋转、变比镜像四种变换，投影变换有平行投影（又称正射投影）和透视投 影两种变换。其变换方法有利于减少算法的运行时间，提高三维图形的显示速度。

3.颜色模式设置：OpenGL 颜色模式有两种，即 RGBA 模式和颜色索引（Color Index）。

4.光照和材质设置：OpenGL 光有辐射光（Emitted Light）、环境光（Ambient Light）、漫反射光（Diffuse Light）和镜面光（Specular Light）。材质是用光反射率来表示。场景（Scene）中物体最终反映到人眼的颜色是光的红绿蓝分量与材质红绿蓝分量的反射率相乘后形成的颜色。

5.纹理映射（Texture Mapping）。利用 OpenGL 纹理映射功能可以十分逼真地表达物体表面细节。

6.位图显示和图象增强图象功能除了基本的拷贝和像素读写外，还提供融合（Blending）、反走样（Antialiasing）和雾（fog）的特殊图象效果处理。以上三条可使被仿真物更具真实感，增强图形显示的效果。

7.双缓存动画（Double Buffering）双缓存即前台缓存和后台缓存，简言之，后台缓存计算场景、生成画面，前台缓存显示后台缓存已画好的画面。

此外，利用 OpenGL 还能实现深度暗示（Depth Cue）、运动模糊（Motion Blur）等特殊效果。从而实现了消隐算法。

2.2 MFC 介绍

MFC (Microsoft Foundation Class Library)[3]中的各种类结合起来构成了一个应用程序框架，它的目的就是让程序员在此基础上来建立 Windows 下的应用程序，这是一种相对 SDK 来说更为简单的方法。因为总体上，MFC 框架定义了应用程序的轮廓，并提供了用户接口的标准实现方法，程序员所要做的就是通过预定义的接口把具体应用程序特有 的东西填入这个轮廓。Microsoft Visual C++提供了相应的工具来完成这个工作：AppWizard 可以用来生成初步的框架文件（代码和资源等）；资源编辑器用于帮助直观地设计用户接口；ClassWizard 用来协助添加代码到框架文件；最后，编译，则通过类库实现了应用程序特定的逻辑。

封装，构成 MFC 框架的是 MFC 类库。MFC 类库是 C++类库。这些类或者封装了 Win32 应用程序编程接口，或者封装了应用程序的概念，或者封装了 OLE 特性，或者封装了 ODBC 和 DAO 数据访问的功能，等等，分述如下。

（1）对 Win32 应用程序编程接口的封装

用一个 C++ Object 来包装一个 Windows Object。例如：class CWnd 是一个 C++ window object，它把 Windows window(HWND)和 Windows window 有关的 API 函数封装在 C++ window object 的成员函数内，后者的成员变量 m_hWnd 就是前者的窗口句柄。

（2）对应用程序概念的封装

使用 SDK 编写 Windows 应用程序时，总要定义窗口过程，登记 Windows Class，创建窗口，等等。MFC 把许多类似的处理封装起来，替程序员完成这些工作。另外，MFC 提出了以文档-视图为中心的编程模式，MFC 类库封装了对它的支持。文档是用户操作的数据对象，视图是数据操作的窗口，用户通过它处理、查看数据。

（3）对 COM/OLE 特性的封装

OLE 建立在 COM 模型之上，由于支持 OLE 的应用程序必须实现一系列的接口（Interface），因而相当繁琐。MFC 的 OLE 类封装了 OLE API 大量的复杂工作，这些类提供了实现 OLE 的更高级接口。

（4）对 ODBC 功能的封装

以少量的能提供与 ODBC 之间更高级接口的 C++类，封装了 ODBC API 的大量的复杂的工作，提供了一种数据库编程模式。

继承，首先，MFC 抽象出众多类的共同特性，设计出一些基类作为实现其他类的基础。这些类中，最重要的类是 CObject 和 CCmdTarget。CObject 是 MFC 的根类，绝大多数 MFC 类是其派生的，包括 CCmdTarget。CObject 实现了一些重要的特性，包括动态类信息、动态创建、对象序列化、对程序调试的支持，等等。所有从 CObject 派生的类都将具备或者可以具备 CObject 所拥有的特性。CCmdTarget 通过封装一些属性和方法，提供了消息处理的架构。MFC 中，任何可以处理消息的类都从 CCmdTarget 派生。

针对每种不同的对象，MFC 都设计了一组类对这些对象进行封装，每一组类都有一个基类，从基类派生出众多更具体的类。这些对象包括以下种类：窗口对象，基类是 CWnd；应用程序对象，基类是 CwinThread；文档对象，基类是 Cdocument，等等。

程序员将结合自己的实际，从适当的 MFC 类中派生出自己的类，实现特定的功能，达到自己的编程目的。

虚拟函数和动态约束，MFC 以“C++”为基础，自然支持虚拟函数和动态约束。但是作为一个编程框架，有一个问题必须解决：如果仅仅通过虚拟函数来支持动态约束，必然导致虚拟函数表 过于臃肿，消耗内存，效率低下。例如，CWnd 封装 Windows 窗口对象时，每一条 Windows 消息对应一个成员函数，这些成员函数为派生类所继承。如果这些函数都设计成虚拟函数，由于数量太多，实现 起来不现实。于是，MFC 建立了消息映射机制，以一种富有效率、便于使用的手段解决消息处理函数的动态约束问题。

这样，通过虚拟函数和消息映射，MFC 类提供了丰富的编程接口。程序员继承基类的同时，把自己实现的虚拟函数和消息处理函数嵌入 MFC 的编程框架。MFC 编程框架将在适当的时候、适当的地方来调用程序的代码。本书将充分的展示 MFC 调用虚拟函数和消息处理函数的内幕，让读者对 MFC 的编程接口有清晰的理解。

MFC 的宏观框架体系，如前所述，MFC 实现了对应用程序概念的封装，把类、类的继承、动态约束、类的关系和相互作用等封装起来。这样封装的结果对程序员来说，是一套开发模板（或者说模式）。针对不同的应用和目的，程序员采用不同的模板。例如，SDI 应用程序的模板，MDI 应用程序的模板，规则 DLL 应用程序的模板，扩展 DLL 应 用程序的模板，OLE/ACTIVEX 应用程序的模板，等等。

这些模板都采用了以文档-视为中心的思想，每一个模板都包含一组特定的类。典型的 MDI 应用程序的构成将在下一节具体讨论。

为了支持对应用程序概念的封装，MFC 内部必须作大量的工作。例如，为了实现消息映射机制，MFC 编程框架必须要保证首先得到消息，然后按既定的方法进行处理。又如，为了实现对 DLL 编程的支持和多线程编程的支持，MFC 内部使用了特别的处理方法，使用模块状态、线程状态等来管理一些重要信息。虽然，这些内 部处理对程序员来说是透明的，但是，懂得和理解 MFC 内部机制有助于写出功能灵活而强大的程序。

总之，MFC 封装了 Win32 API，OLE API，ODBC API 等底层函数的功能，并提供更高一层的接口，简化了 Windows 编程。同时，MFC 支持对底层 API 的直接调用。

MFC 提供了一个 Windows 应用程序开发模式，对程序的控制主要是由 MFC 框架完成的，而且 MFC 也完成了大部分的功能，预定义或实现了许多事件和消息处理，等等。框架或者由其本身处理事件，不依赖程序员的代码；或者调用程序员的代码来处理应用程序特定的事件。

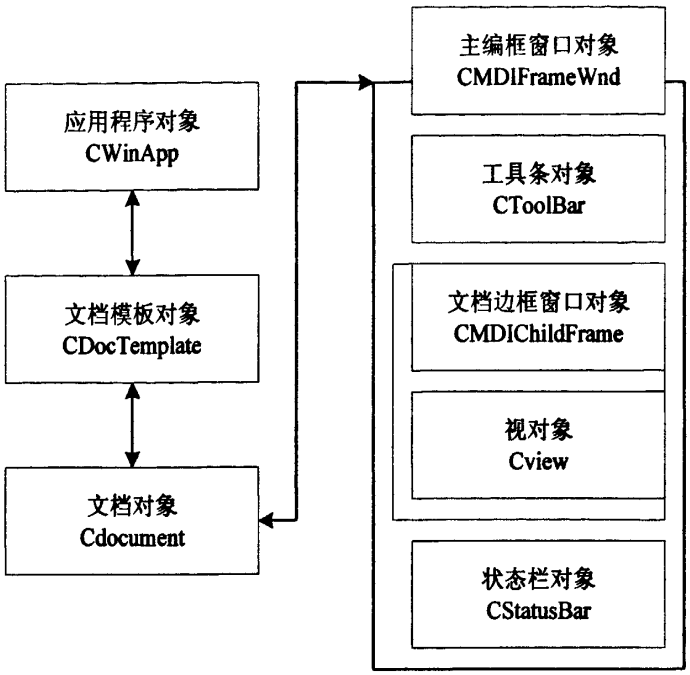

图 2-1 一个应用程序的构成

MFC 是 C++类库，程序员就是通过使用、继承和扩展适当的类来实现特定的目的。例如，继承时，应用程序特定的事件由程序员的派生类来处理，不感兴趣的由基类处理。实现这种功能的基础是 C++对继承的支持，对虚拟函数的支持，以及 MFC 实现的消息映射机制。

本段介绍了 MDI 应用程序的构成。用 AppWizard 产生一个 MDI 工程 t（无 OLE 等支持），AppWizard 创建了一系列文件，构成了一个应用程序框架。这些文件分四类：头文件（.h），实现文件(.cpp)，资源文件(.rc)，模块定义文件(.def)，等。

图 2-1 解释了该应用程序的结构，箭头表示信息流向。从 CWinApp、CDocument、CView、CMDIFrameWnd、CMDIChildWnd 类对应地派生出 CTApp、CTDoc、 CTView、CMainFrame、CChildFrame 五个类，这五个类的实例分别是应用程序对象、文档对象、视对象、主框架窗口对象和文档边框窗 口对象。主框架窗口包含了视窗口、工具条和状态栏。对这些类或者对象解释如下。

（1）应用程序

应用程序类派生于 CWinApp。基于框架的应用程序必须有且只有一个应用程序对象，它负责应用程序的初始化、运行和结束。

（2）边框窗口

如果是 SDI 应用程序，从 CFrameWnd 类派生边框窗口类，边框窗口的客户子窗口(MDIClient)直接包含视窗口；如果是 MDI 应用程序，从 CMDIFrameWnd 类派生边框窗口类，边框窗口的客户子窗口(MDIClient)直接包含文档边框窗口。

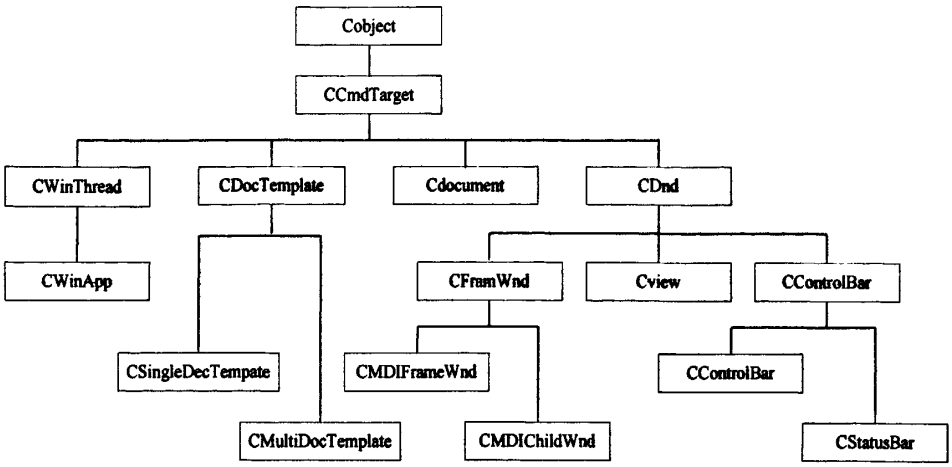

图 2-2　一些 MFC 类的层次

如果要支持工具条、状态栏，则派生的边框窗口类还要添加 CToolBar 和 CStatusBar 类型的成员变量，以及在一个 OnCreate 消息处理函数中初始化这两个控制窗口。

边框窗口用来管理文档边框窗口、视窗口、工具条、菜单、加速键等，协调半模式状态（如上下文的帮助(SHIFT+F1 模式)和打印预览）。

（3）文档边框窗口

文档边框窗口类从 CMDIChildWnd 类派生，MDI 应用程序使用文档边框窗口来包含视窗口。

（4）文档

文档类从 CDocument 类派生，用来管理数据，数据的变化、存取都是通过文档实现的。视窗口通过文档对象来访问和更新数据。

（5）视

视类从 CView 或它的派生类派生。视和文档联系在一起，在文档和用户之间起中介作用，即视在屏幕上显示文档的内容，并把用户输入转换成对文档的操作。

（6）文档模板

文档模板类一般不需要派生。MDI 应用程序使用多文档模板类 CMultiDocTemplate；SDI 应用程序使用单文档模板类 CSingleDocTemplate。

应用程序通过文档模板类对象来管理上述对象（应用程序对象、文档对象、主边框窗口对象、文档边框窗口对象、视对象）的创建。

这里，用图的形式可直观地表示所涉及的 MFC 类的继承或者派生关系，如图 2-2 所示意。

图 1-2 所示的类都是从 CObject 类派生出来的；所有处理消息的类都是从 CCmdTarget 类派生的。如果是多文档应用程序，文档模板使用 CMultiDocTemplae，主框架窗口从 CMdiFarmeWnd 派生，它包含工具条、状态

栏和文档框架窗口。文档框架窗口从 CMdiChildWnd 派生，文档框架窗口包含视，视从 CView 或其派生类派生。

构成应用程序的文件

通过上述分析，可知 AppWizard 产生的 MDI 框架程序的内容，所定义和实现的类。下面，从文件的角度来考察 AppWizard 生成了哪些源码文件，这些文件的作用是什么。表 2-1 列出了 AppWizard 所生成的头文件，表 2-2 列出了了 AppWizard 所 生成的实现文件及其对头文件的包含关系。

表 2-1 AppWizard 所生成的头文件

头文件	用途
stdafx.h	标准 AFX 头文件
resource.h	定义了各种资源 ID
t.h	#include "resource.h" 定义了从 CWinApp 派生的应用程序对象 CTApp
childfrm.h	定义了从 CMDIChildWnd 派生的文档框架窗口对象 CTChildFrame
mainfrm.h	定义了从 CMDIFrameWnd 派生的框架窗口对象 CMainFrame
tdoc.h	定义了从 CDocument 派生的文档对象 CTDoc
tview.h	定义了从 CView 派生的视图对象 CTView

表 2-2 AppWizard 所生成的实现文件

实现文件	所包含的头文件	实现的内容和功能
stdafx.cpp	#include "stdafx.h"	用来产生预编译的类型信息。
t.cpp	# include "stdafx.h" # include "t.h" #include "MainFrm.h" # include "childfrm.h" #include "tdoc.h" #include "tview.h"	定义 CTApp 的实现，并定义 CTApp 类型的全局变量 theApp。
childfrm.cpp	#inlcude "stdafx.h" #include "t.h" #include "childfrm.h"	实现了类 CChildFrame
childfrm.cpp	#inlcude "stdafx.h" #include "t.h" #include "childfrm.h"	实现了类 CMainFrame
tdoc.cpp	# include "stdafx.h"	实现了类 CTDoc

	# include "t.h" # include "tdoc.h"	
tview.cpp	# include "stdafx.h" # include "t.h" # include "tdoc.h" # include "tview.h"	实现了类 CTview

从表 2-2 中的包含关系一栏可以看出：

CTApp 的实现用到所有的用户定义对象，包含了他们的定义；CView 的实现用到 CTdoc；其他对象的实现只涉及自己的定义；

当然，如果增加其他操作，引用其他对象，则要包含相应的类的定义文件。

对预编译头文件说明如下：

所谓头文件预编译，就是把一个工程(Project)中使用的一些 MFC 标准头文件(如 Windows.H、Afxwin.H)预先编译，以后该工程编译时，不再编译这部分头文件，仅仅使用预编译的结果。这样可以加快编译速度，节省时间。

预编译头文件通过编译 stdafx.cpp 生成，以工程名命名，由于预编译的头文件的后缀是“pch”，所以编译结果文件是 projectname.pch。

编译器通过一个头文件 stdafx.h 来使用预编译头文件。stdafx.h 这个头文件名是可以在 project 的编译设置里指定的。编译器认为，所有在指令#include "stdafx.h"前的代码都是预编译的，它跳过#include "stdafx. h"指令，使用 projectname.pch 编译这条指令之后的所有代码。

因此，所有的 CPP 实现文件第一条语句都是：#include "stdafx.h"。

另外，每一个实现文件 CPP 都包含了如下语句：

```
#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif
```

这是表示，如果生成调试版本，要指示当前文件的名称。__FILE__是一个宏，在编译器编译过程中给它赋值为当前正在编译的文件名称。

2.3 光反射理论

反射有多种不同的类型，如镜面反射、混合反射、完全漫反射以及不规则漫反射等。室内照明几乎全部使用漫反射。材料的反射特性用反射系数 r 来表示。

该系数是材料反射的光通量与接受到的光通量的比值（r＝F/ F0）。

绝大多数情况下，反射系数是针对漫射光（rdif）或准平行光（r）而规定的。在照明技术中，绝大多数使用的是漫射光的反射系数。理论上，反射系数的最大值为 1（100%）。光行进到两种媒质的分界面时，有一部分返回原媒质的现象。如果入射点处分界面的不平整程度比波长小得多，就可看作平滑分界面。

光在平滑分界面上的反射为镜反射，或称单向反射。否则，为漫反射。如果既有镜反射，又有漫反射，则称为混合反射。漫反射时各不同方向的光的亮度如果都相同，则称为均匀漫反射。同一次反射过程中，有些波长的光反射得多些，则称为选择反射。可以利用涂膜技术，改变媒质分界面的光学性质，使反射光中减少或增多某一波长的光的成分，以适应不同光学元件的要求。垂直于或近似垂直于分界面的正入射光所引起的反射称为正反射。单位时间内从界面单位面积上反射光所带走的能量与入射光入射的能量之比，称为反射率。能量之比等于光强之比，故反射率即为反射光强与入射光强之比，也等于反射光与入射光的振幅平方之比。正反射时，反射率可用第二媒质对第一媒质的相对折射率 n21（=n2/n1）表示，为 R=(n21-1)2/(n21+2)2。反射光波振幅与入射光波振幅之比，称为反射比或反射系数。

研究某材料的反射情况，需要测定该材料样品平面在各方向上对各种波长的反射率。对于一般性的研究工作，国际照明委员会（简写 CIE）推荐取 45°±5°方向上照明，90°±10°的法线方向观测；也可以 90°±10°照明而以 45°±5°观测。物质的反射率与厚度有关，因为一部分光要稍进入物体内部，然后再被反射出来。故严格的定义还应限定材料样品的厚度。

反射率有时也称为反射比，反射系数，反射本领或反射因数。我国全国自然科学名词审定委员会于 1988 年公布的物理学名词（基础物理学部分）中，把这些术语统一规范为强度反射率和振幅反射率。

光的反射是生活中常见的现象。沙漠中有时会出现一种称为海市蜃楼的光学幻视现象，这是一种光的反射现象。当大气下层热，上层凉，光线从地面向上传播逐渐弯曲 而至反射，使沙漠中的人们看到远处的景物。这称为下现蜃景。如果冷气流在下层，则可能会发生上部景物的全反射成像，称为上现蜃景。

2.4 光照度学理论

在光照度学中是没有“光强”这样一个概念的。常用的光学量概念有发光强度、光照度、光出射度和光亮度。“光强”只是一个通俗的说法，很难说对应哪一个光度学概念。以上所说的几个概念都是有严格的物理定义的：

发光强度：光源在单位立体角内发出的光通量，单位是坎德拉，即每球面度1流明。

光照度：被照明面单位面积上得到的光通量，单位是勒克斯，即每平方米1流明。

光出射度：光源单位面积上发出的光通量，单位与光照度相同。

光亮度：单位面积上沿法线方向的发光强度，或称单位面积在其法线方向上单位立体角内发出的光通量，单位是尼特，即每平方米每球面度1流明。

由于发光强度、光亮度与方向有关，容易推导出：各个方向上光亮度相同的光源其发光强度是方向的余弦函数，在法线方向上发光强度最大，称为余弦辐射体，也叫朗伯光源。各个方向上发光强度都相等的光源其光亮度就是不等的。发光强度、光出射度和光亮度都是表示光源的发光的发光特性的。考虑太阳到地球距离的平方是将太阳当成点光源，利用地面上的照度计算太阳的发光强度。 而把太阳朝向地球的这一面作为一个面光源，再除以这个面积就是太阳在与地球连线方向的光亮度。当然这与太阳直接发光的发光强度或光亮度相比是有下降的，因为太阳光经过大气还要衰减的。

这些光学量都用到光通量，光通量是与辐射能通量相对应的光学量，因为光是一种电磁辐射。不同波长的电磁波1瓦的辐射能通量所相当的光通量是不一样的，换算到光通量要考虑人眼的光谱灵敏度曲线，即人眼对不同波长同样的辐射能通量所感受到的光是不一样的，如红外光、微波、紫外光等人眼是看不见的，而400nm到760nm波长的可见光是人眼能看得见的。

在物理光学中也提到“光强”，是用麦克斯韦方程组解出光的电矢量，电场强度的平方就是物理光学中的光强，主要用于计算干涉、衍射效应得到的图形。在光学各相关学科中光强度是一个比较含糊的概念，不同的分支有不同的说法，有的等同于发光强度，有的等同于光照度，有的等同于光亮度。而光度学中这几个概念是有严格的物理意义的。

光照度，即通常所说得勒克司度（lux），表示被摄主体表面单位面积上受到的光通量。1勒克司相当于1流明/平方米，即被摄主体每平方米的面积上，受距离一米、发光强度为1烛光的光源，垂直照射的光通量。光照度是衡量拍摄环境的一个重要指标。

计算，室内照明利用系数法计算平均照度：

在平时做照度计算时，如果我们已知利用系数“CU”，则可以方便的利用一个经验公式进行快速计算，求出我们想要的室内工作面的平均照度值。我们通常把这种计算方法称为“利用系数法求平均照度”，也叫流明系数法。

照度计算有粗略地计算和精确地计算2种。例如，假设像住宅那样整体照度

应该在100勒克斯(lx)的情况，而即使是90勒克斯(lx)也不会对生活带来 很大的影响。但是，如果是道路照明的话,情况就不同了。假设路面照度必须在20勒克斯(lx)的情况下，如果是18勒克斯(lx)的话，就有可能造成交通 事故频发。商店也是一样，例如，商店的整体最佳照度是500勒克斯(lx)，由于用600勒克斯(lx)的照度,所以,照明灯具数量和电量就会增加,并在 经济上造成影响。无论是哪一种照度计算都是重要的。虽然只是粗略地估算,也会有20%-30%的误差。所以建议在一般情况下最好采用专业的照明设计软件进 行精确模拟计算，将误差控制在最小范围内。

但有时我们由于情况特殊或场地条件所限，而不能采用照明软件模拟计算时，在计算地板、桌面、作业台面平均照度可以用下列基本公式进行，略估算出灯具照度(勒克斯 lx)=光通量(流明 lm)/面积(平方米 m^2) 即平均1勒克斯(lx)的照度,是1流明(lm)的光通量照射在1平方米(m^2)面积上的亮度。

用这种方法求房间地板面的平均照度时,在整体照明灯具的情况下,可以用下列公式进行计算。

平均照度(Eav)= 单个灯具光通量 Φ×灯具数量(N)×空间利用系数(CU)×维护系数(K)÷地板面积(长×宽)

公式说明：

1、单个灯具光通量Φ，指的是这个灯具内所含光源的裸光源总光通量值。

2、空间利用系数(CU)，是指从照明灯具放射出来的光束有百分之多少到达地板和作业台面,所以与照明灯具的设计、安装高度、房间的大小和反射率的不同 相关,照明率也随之变化。如常用灯盘在3米左右高的空间使用，其利用系数CU可取0.6--0.75之间；而悬挂灯铝罩，空间高度6--10米时，其利用 系数CU取值范围在0.7--0.45；筒灯类灯具在3米左右空间使用，其利用系数CU可取0.4--0.55；而像光带支架类的灯具在4米左右的空间使用时，其利用系数CU可取0.3--0.5。以上数据为经验数值，只能做粗略估算用，如要精确计算具体数值需由公司书面提供，相关参数，在此仅做参考。

3、是指伴随着照明灯具的老化，灯具光的输出能力降低和光源的使用时间的增加，光源发生光衰;或由于房间灰尘的积累，致使空间反射效率降低，致使照度降低而乘上的系数.一般较清洁的场所,如客厅、卧室、办公室、教室、阅读室、医院、高级品牌专卖店、艺术馆、博物馆等维护系数K取0.8；而一般 性的商店、超市、营业厅、影剧院、机械加工车间、车站等场所维护系数K取0.7；而污染指数较大的场所维护系数K则可取到0.6左右。

利用系数法

此方法用于计算平均照度

(光源光通量)(CU)(MF) /照射区域面积

适用于室内，体育照明

利用系数(CU)： 一般室内取 0.4，体育取 0.3

1. 灯具的照度分布

2. 灯具效率

3. 灯具在照射区域的相对位置

4. 被包围区域中的反射光

维护系数 MF=(LLD)X(LDD)一般取 0.7～0.8

举例 1：

室内照明，4×5 米房间，使用 3×36W 隔栅灯 9 套

计算公式:平均照度＝光源总光通×CU×MF/面积＝(2500×3×9)×0.4×0.8÷4÷5＝1080 Lux

结论：平均照度 1000Lux 以上

举例 2：

体育馆照明，20×40 米场地，

使用 POWRSPOT 1000W 金卤灯 60 套

平均照度 ＝光源总光通×CU×MF/面积＝(105000×60)×0.3×0.8÷20÷40＝1890 Lux

结论：平均水平照度 1500Lux 以上

垂直照度 1000Lux 以上（视安装位置）

1967 年法国第十三届国际计量大会规定了以坎德拉、坎德拉/平方米、流明、勒克斯分别作为发光强度、光亮度、光通量和光照度等的单位，为统一工程技术中使用的光学度量单位有重要意义。

烛光、国际烛光、坎德拉（candela）的定义：

在每平方米 101325 牛顿的标准大气压下，面积等于 1/60 平方厘米的绝对“黑体”（即能够吸收全部外来光线而毫无反射的理想物体），在纯铂（Pt） 凝固温度（约 2042K 获 1769℃）时，沿垂直方向的发光强度为 1 坎德拉。并且，烛光、国际烛光、坎德拉三个概念是有区别的，不宜等同。从数量上看，60 坎德拉等于 58.8 国际烛光，亥夫纳灯的 1 烛光等于 0.885 国际烛光或 0.919 坎德拉。

1. 发光强度与光亮度

发光强度简称光强，国际单位是 candela（坎德拉）简写 cd。Lcd 是指光源在指定方向的单位立体角内发出的光通量。光源辐射是均匀时，则光强为 I=F/Ω，

Ω为立体角，单位为球面度(sr),F为光通量，单位是流明，对于点光源由I=F/4 。光亮度是表示发光面明亮程度的，指发光表面在指定方向的发光强度与垂直且指定方向的发光面的面积之比，单位是坎德拉/平方米。对于一个漫散射面，尽管各个方向的光强和光通量不同，但各个方向的亮度都是相等的。电视机的荧光屏就是近似于这样的漫散射面，所以从各个方向上观看图像，都有相同的亮度感。

以下是部分光源的亮度值：单位 cd/m^2

太阳：1.5*10 ；日光灯：（5—10）$*10^3$；月光（满月）：$2.5*10^3$；黑白电视机荧光屏：120左右；彩色电视机荧光屏：80左右。

2. 光通量与流明

光源所发出的光能是向所有方向辐射的，对于在单位时间里通过某一面积的光能，称为通过这一面积的辐射能通量。各色光的频率不同，眼睛对各色光的敏感度也有所不同，即使各色光的辐射能通量相等，在视觉上并不能产生相同的明亮程度，在各色光中，黄、绿色光能激起最大的明亮感觉。如果用绿色光作水准，令它的光 通量等于辐射能通量，则对其它色光来说，激起明亮感觉的本领比绿色光为小，光通量也小于辐射能通量。光通量的单位是流明，是英文lumen的音译，简写为 lm。绝对黑体在铂的凝固温度下，从 $5.305*10^3$ cm^2;面积上辐射出来的光通量为 1lm。为表明光强和光通量 的关系，发光强度为 1 坎德拉的点光源在单位立体角（1 球面度）内发出的光通量为 1 六名。一只 40W 的日光灯输出的光通量大约是2100流明。

3. 光照度与勒克斯

光照度可用照度计直接测量。光照度的单位是勒克斯，是英文lux的音译，也可写为lx。被光均匀照射的物体，在1平方米面积上得到的光通量是1流明时，它的照度是1勒克斯。有时为了充分利用光源，常在光源上附加一个反射装置，使得某些方向能够得到比较多的光通量，以增加这一被照面上的照度。例如汽车前 灯、手电筒、摄影灯等。

以下是各种环境照度值：单位 lux

黑夜：0.001—0.02；月夜：0.02—0.3；阴天室内：5—50；阴天室外：50—500；晴天室内：100—1000；夏季中午太阳光下的照度：约为 10^9；阅读书刊时所需的照度：50—60；家用摄像机标准照度：1400

光亮度是表示发光面明亮程度的，指发光表面在指定方向的发光强度与垂直且指定方向的发光面的面积之比，单位是坎德拉/平方米。对于一个漫散射面，尽管各个方向的光强和光通量不同，但各个方向的亮度都是相等的。电视机的荧光屏就是近似于这样的漫散射面，所以从各个方向上观看图像，都有相同的亮度感。

光亮度(luminance)是指一个表面的明亮程度,以 L 表示， 即从一个表面反射出来的光通量。不同物体对光有不同的反射系数或吸收系数。光的强度可用照在平面上的光的总量来度量，这叫入射光（inci-dent light）或照度（illuminance）。若用从平面反射到眼球中的光量来度量光的强度，这种光称为反射光(reflection light)或亮度(brightness)。 例如,一般白纸大约吸收入射光量的 20%，反射光量为 80%；黑纸只反射入射光量的 3%。所以，白纸和黑纸在亮度上差异很大。

最常用的照度单位是呎烛光 (footcandle)。1 呎烛光是在距离标准烛光一英尺远的一平方英尺平面上接受的光通量。如果按公制单位，则以米为标准，照度就用米烛光 (metrecandle)来表示，即 1 米烛光是距离标准烛光一米远的一平方米面积上的照度。1 米烛光等于 0.0929 呎烛光。

我们不难理解亮度和照度之间的关系,其关系为：

L=R×E

其中 L 为亮度,R 为反射系数,E 为照度。

因此，当我们知道一个物体表面的反射系数及其表面的照度时,便可推算出它的亮度。

亮度也有几种度量单位。亮度的单位是用一种理想化了的标准状态来定义的。以一支标准蜡烛当作光源，放在一个半径为 1 公尺的球体的中心位置。假设这个蜡烛会均匀发散它的全部光线，则落在球体内表面一平方公尺表面积上的所有光量为 1 个流明(lumen)。实际应用中,亮度单位用流明太小了，所以通常取其十倍的单位——毫朗伯(millilambert) 来表示。比毫朗伯稍大的单位是呎朗伯(footlambert)，1 毫朗伯等于 0.929 呎朗伯.英国标准的呎朗伯是用光源的烛光数，从光源到表面积的英尺数和表面的反射率来规定的.在有些国家，普遍使用的是米制单位，是以毫朗伯为基础的[1 毫朗伯(mL)=0.929 呎朗伯(ftL)=3.183 烛光/平方米(c/m^2)=10 阿普熙提(apostilbs)]。光亮度的单位还有：坎德拉/平方米(即尼特,Nit=1cd/m^2)等。

光束经界面折射或者反射后的亮度，忽略散射，吸收损失，有 $d\phi = d\phi' + d\phi''$，其中 $d\phi$ 为入射光通量，$d\phi'$ 为折射光通量，$d\phi''$ 为反射光通量，i 为入射角，i' 为折射角，i'' 为反射角，$d\omega$ 为入射光球面角，$d\omega'$ 为折射光球面角，$d\omega''$ 为反射光球面角，L 为单位面积发光强度；

$$\begin{aligned} &d\phi = LdS\cos i d\omega \\ &d\phi' = L'dS\cos i' d\omega' \\ &d\phi'' = L''dS\cos i'' d\omega'' \end{aligned} \quad \text{且} \quad \begin{aligned} &d\phi = d\phi' + d\phi'' \\ &d\omega = \sin i di d\phi \\ &d\omega' = \sin i' di' d\phi \\ &d\omega'' = \sin i'' di'' d\phi \end{aligned} \qquad \text{式（2-1）}$$

$d\phi$ 按球坐标定义反射时 $i'' = -i$，因此 $d\omega'' = d\omega$

$$\therefore \frac{d\phi''}{d\phi}=\frac{L''}{L}=\rho \text{ 则 } \begin{matrix} L''=\rho L \\ d\phi''=\rho d\phi \end{matrix} \qquad \text{式(2-2)}$$

这里 ρ 是反射率。当入射角不大时 $\rho=\left(\frac{n'-n}{n'+n}\right)^2$ 式(2-3)

此二式相乘，可导出 $\frac{d\omega'}{d\omega}=\frac{\sin i'di'}{\sin idi}=\frac{n^2\cos i}{n'^2\cos i'}$ 得 $L'=(1-\rho)L\left(\frac{n'}{n}\right)^2$ 式(2-4)

且 $d\phi'=d\phi-d\phi''=d\phi(1-\rho)$ 式(2-5)

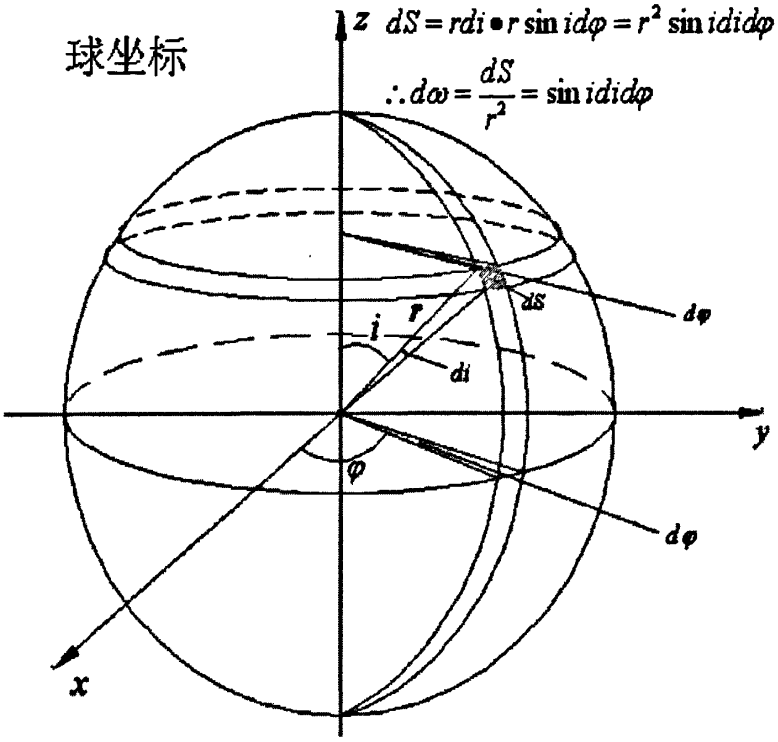

图 2-3　球坐标空间角

第三章 软件需求设计

三维物体亮度信息计算软件是基于 WINDOWS 操作系统，使用 VC6.0 开发环境，利用 MFC，OPENGL 类库[4]开发，用于仿真不同入射角度以及不同观察角度，三维物体的亮度信息显示以及相关辐射功率计算的软件。

该软件主要功能：

- 导入相关三维模型数据，以及相关材料反射系数文件，输出图像文件序列
- 设置不同入射光角度，以及不同观察者角度；
- 设置仿真场景，计算并正确显示物体亮度信息以及物体发光功率；
- 操纵显示图像，能够看到三维物体完整亮度信息情况；

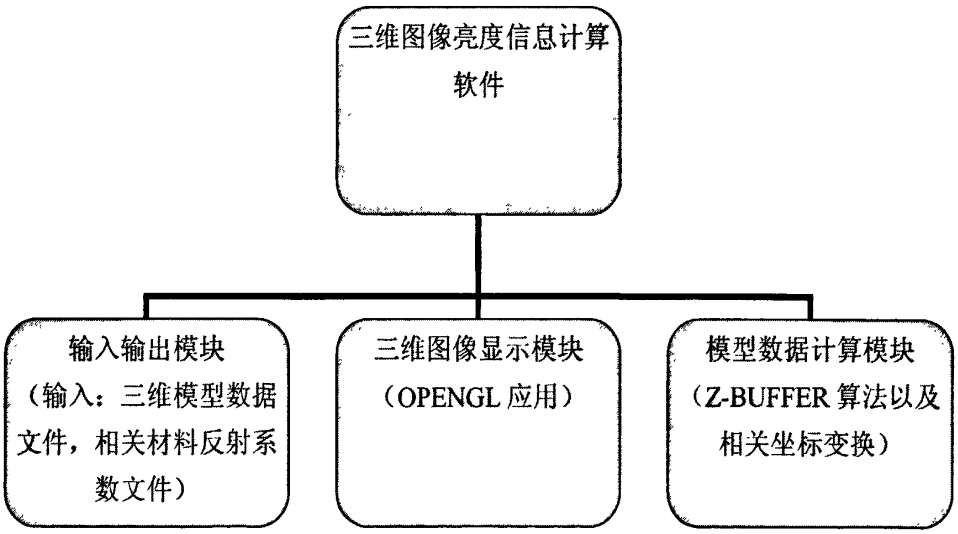

图3-1 软件主要模块图

3.1 软件需求分析

软件的假定前提条件为：

- 假定太阳光为平行光（不考虑太阳为点光源）；入射光照射物体，物体反射的光线都为平行光，不考虑漫反射以及二次反射情况；
- 该软件数据文件以及反射系数文件都是第三方提供，计算物体辐射功率公式也为第三方计算得出，不在本论文推导以及展示，该计算准确性与提供公式相关，不对预期数据结果进行分析探讨；

该软件主要需求为用于一定三维数据模型的亮度信息计算，能够显示计算基本物体形状各部位的亮度信息以及显示该物体辐射功率。

具体各模块需求分析：

- 软件输入部分能够添加删除采用特定格式的数据模型文件和特定的材质反射系数文件，能够正确加载模型文件，清除相应数据；
- 软件显示部分应该能够正确显示数据模型，坐标系，能够旋转，放大，缩小物体以及分析计算后模型的亮度信息分布情况；
- 软件设置部分应该能够设置入射光方位俯仰角，观察方位俯仰角，以及旋动变换，视场角度，观察距离，归一化系数；
- 软件输出部分应该能够正确显示当前计算的条件以及计算的最终结果；

该软件大致处理流程，如图 3-2：

1. 输入模块：添加数据模型文件，选择相应材质，加载数据进入内存；
2. 显示模块：显示模块读取内存数据，正确显示数据模型；
3. 计算模块：分析计算内存数据，得出正确结果；
4. 显示输出模块：显示模块正确显示相应模型亮度分布信息，输出模块输出输入条件以及计算结果；

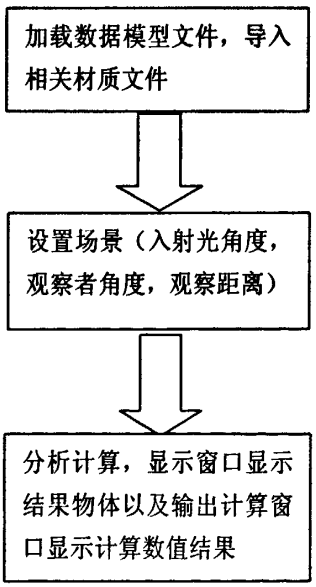

图 3-2　软件处理流程图

3.2　软件设计流程

该软件基于 VC 下 MFC 窗体[5]框架构成，利用 OPENGL 类库开发图像界面接口，采用 z-buffer 算法进行三维图形消隐算法。我们需要设计四大模块，四大模块主要为：窗体框架模块，输入输出模块，显示模块，数据计算模块；

该软件设计遵照如下图 3-3 流程：

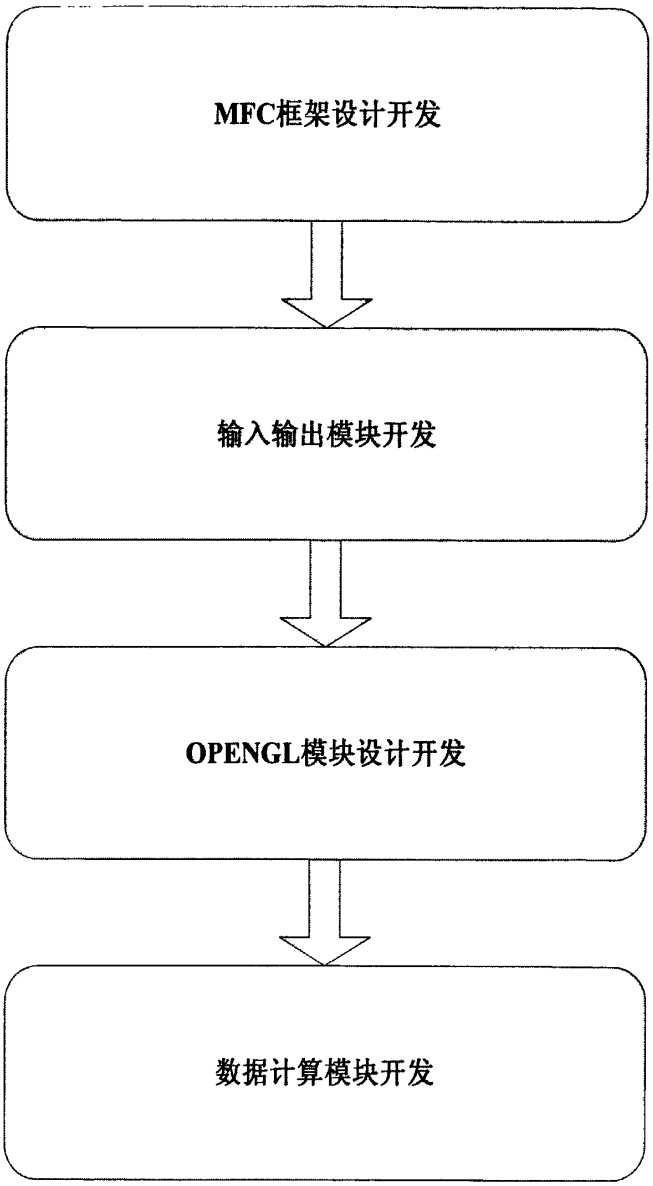

图 3-3 设计开发流程图

MFC 框架设计开发：MFC 框架设计采用 CView,CSplitter，ListBox,ControlBox 等类开发，主要开发控制台界面和显示界面；

输入输出模块开发：采用 File 类或者 CFile 类对模型数据文件进行读写操作，文件读写操作需要区分通配符，输出模块采用 ListBox 开发；

OPENGL 模块开发：OPENGL 主要建立在开发 CView::OnDraw()函数上，同时关注旋转，放大，缩小功能函数的开发；

数据计算模块开发：数据计算主要采用 z-buffer 算法，主要集中开发三维物体旋转函数以及遮挡算法函数；

第四章　软件模块设计

4.1　MFC 框架设计

VC6.0 下 MFC 框架图如下：

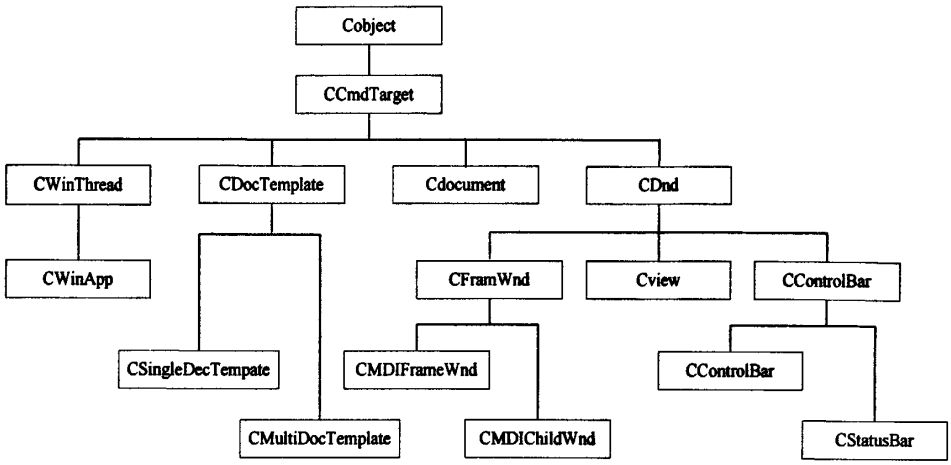

图 4-1　MFC 类框架图

该窗体设计分为控制面板和显示窗口，使用 CSplitter 类使窗体分割成两部分，左半部分：显示部分；右半部分：控制面板；

我们主要使用的 MFC 类有 Cview,CWnd,CstatusBar,Cdocument 等类。

划分主窗口为显示和控制两部分的类为 CSplitter，利用该类的方法在 CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs,CCreateContext* pContext)函数中划分窗口，关键部分代码：

```
int cx = GetSystemMetrics(SM_CXSCREEN);
int cy = GetSystemMetrics(SM_CYSCREEN);
//将主窗口分为一行两列
if(m_Sp1.CreateStatic(this,1,2)==NULL)
 return FALSE;
 m_Sp1.SetColumnInfo(0,cx*4/7,100);
m_Sp1.CreateView(0,0,RUNTIME_CLASS(CTest1View),CSize(800,800),
pContext);
m_Sp1.CreateView(0,1,RUNTIME_CLASS(CControlWnd),CSize(186,400),p
Context);
```

4.1.1 控制面板窗体设计

控制面板主要包括：输入输出部分，设置场景以及相应角度，数据计算部分，图像显示控制部分。

输入输出部分：输入部分应该能够正确显示输入文件名，以及该数据输入文件使用对应的材质反射系数文件，对文件能够进行加载以及添加删除数据文件功能，还附加能够浏览材质反射系数文件功能，对比不同角度下的反射系数；

设置部分：能够设置入射光角度以及观察者角度：0-360 度，入射光以及观察者都应包括方位俯仰角，场景设置应该包括物体与观察者之间的距离，视场角度以及视场归一化亮度系数；

数据计算部分：能够恢复默认角度设置和场景设置，能够清空驻留在内存的数据，能够分析计算和退出计算系统；

图像显示控制部分：设定滚轴，使显示图像能够绕指定轴旋转，能够设置是否光照，对物体特定区域能够放大缩小拾取，能够回复到初始图像状态；

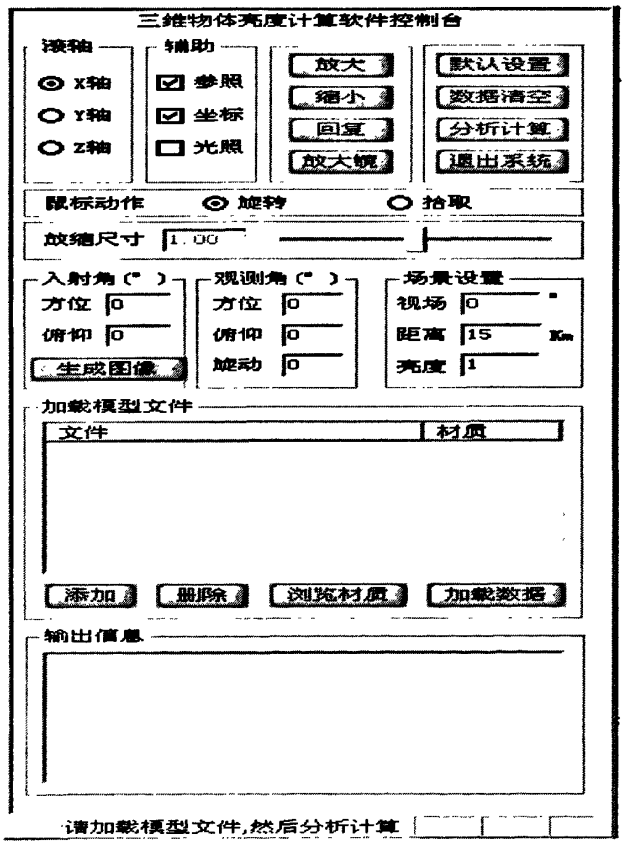

图 4-2　控制台界面图

4.1.2 显示窗体设计

显示窗体设计：显示窗体背景设计为黑色，包含 x,y,z 轴以及原点，建立笛卡尔坐标系；默认坐标系设置：y 轴向右，z 轴向下，x 轴由屏幕从外到里；

实际上计算机上的三维坐标[6]默认为，x 轴向右，y 轴向上，z 轴指向由屏幕里到屏幕外；该显示坐标设置是为了适应 OPENGL 默认光照方向，由屏幕外指向屏幕里，这样建立默认坐标系，能够让默认光照方向处于入射方位角度为 0 度，俯仰角度为 0 度的初始状态；结果显示见图 4-3；

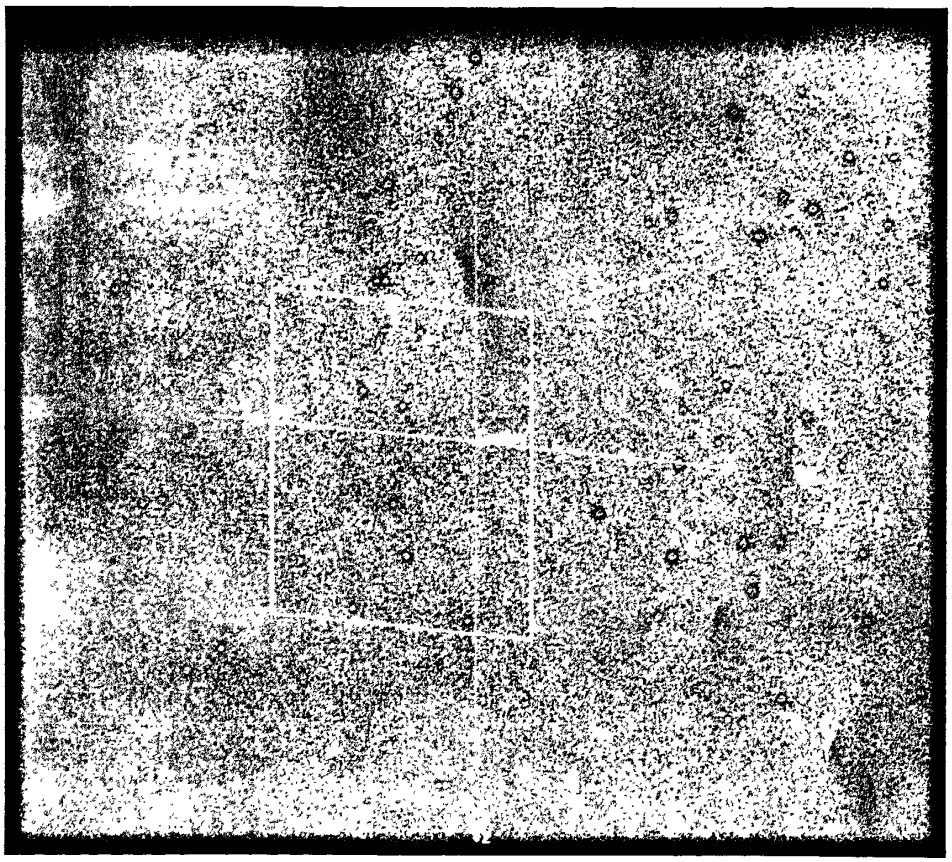

图 4-3 显示窗体设计图

4.2 输入输出模块设计

输入输出模块是整个软件赖以生存的接口模块，输入模块主要包括：数据模型文件输入以及材料反射系数文件输入；输出模块主要包括：计算结果显示；

4.2.1 输入模块设计

输入模块采用 MFC 中的 CFILE 类对我们需要添加的文件进行操作，并且读

取文件数据存储到内存；

CFile 操作文件的读写(基本功能)介绍：

文件读写的最普通的方法是直接使用 CFile 进行，如文件的读写可以使用下面的方法：

```
//对文件进行读操作
char sRead[2];
CFile mFile(_T("user.txt"),CFile::modeRead);
if(mFile.GetLength()<2)
return;
mFile.Read(sRead,2);
mFile.Close();
//对文件进行写操作
CFile mFile(_T("user.txt "), CFile::modeWrite|CFile::modeCreate);
mFile.Write(sRead,2);
mFile.Flush();
mFile.Close();
```

虽然这种方法最为基本，但是它的使用繁琐，而且功能非常简单。我们还可以使用 CArchive，它的使用方法简单且功能十分强大。首先还是用 CFile 声明一个对象，然后用这个对象的指针做参数声明一个 CArchive 对象，你就可以非常方便地存储各种复杂的数据类型了。

```
//对文件进行写操作
CString strTemp;
CFile mFile;
mFile.Open("d:\\dd\\try.TRY",CFile::modeCreate|CFile::modeNoTruncate|CFile
::modeWrite);
CArchive ar(&mFile,CArchive::store);
ar<<strTemp;
ar.Close();
mFile.Close();
//对文件进行读操作
CFile mFile;
if(mFile.Open("d:\\dd\\try.TRY",CFile::modeRead)==0)
return;
CArchive ar(&mFile,CArchive::load);
```

```
ar>>strTemp;
ar.Close();
mFile.Close();
```

CArchive 的 << 和>> 操作符用于简单数据类型的读写，对于 CObject 派生类的对象的存取要使用 ReadObject()和 WriteObject()。使用 CArchive 的 ReadClass()和 WriteClass()还可以进行类的读写，如：

```
//存储 CAboutDlg 类
ar.WriteClass(RUNTIME_CLASS(CAboutDlg));
//读取 CAboutDlg 类
CRuntimeClass* mRunClass=ar.ReadClass();
//使用 CAboutDlg 类
CObject* pObject=mRunClass->CreateObject();
((CDialog* )pObject)->DoModal();
```

让用户选择文件进行打开和存储操作时，就要用到文件打开/保存对话框。MFC 的类 CFileDialog 用于实现 这种功能。使用 CFileDialog 声明一个对象时，第一个 BOOL 型参数用于指定文件的打开或保存，当为 TRUE 时将构造一个文件打开对话框，为 FALSE 时构造一 个文件保存对话框。

在构造 CFileDialog 对象时，如果在参数中指定了 OFN_ALLOWMULTISELECT 风格，则在此对话框中 可以进行多选操作。此时要重点注意为此 CFileDialog 对象的 m_ofn.lpstrFile 分配一块内存，用于存储多选操作所返回的所有文件路径 名，如果不进行分配或分配的内存过小就会导致操作失败。下面这段代码演示了文件打开对话框的使用方法。

```
CFileDialog    mFileDlg(TRUE,NULL,NULL,
OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT|OFN_ALLOWMULTISEL
ECT,"All Files (*.*)|*.*||",AfxGetMainWnd());
CString str(" ",10000);
mFileDlg.m_ofn.lpstrFile=str.GetBuffer(10000);
str.ReleaseBuffer();
POSITION mPos=mFileDlg.GetStartPosition();
CString pathName(" ",128);
CFileStatus status;
while(mPos!=NULL)
{
  pathName=mFileDlg.GetNextPathName(mPos);
```

```
    CFile::GetStatus( pathName, status );
}
```

采用 CFile 和 CFileDialog 类，我们能够添加文件，读取文件内容，但是我们数据文件格式有一定规范，我们根据特定的数据文件格式，采用通配符的方法读入数据：

通过使用 strcmp（）函数，我们能够比较读取数据，发现标志位，然后再将数据读取至内存，如下例：

```
//读到第一个标志位"$   GRID Data"
while(strcmp(s,"GRID")!=0)
{
    fscanf(stream,"%s",s);
};
fscanf(stream,"%s",s);
//读到第二个标志位"GRID 1"
while(strcmp(s,"GRID")!=0)
{
    fscanf(stream,"%s",s);
};
//读到第一个标志位"$   CTRIA3 Data"
while(strcmp(s,"CTRIA3")!=0)
{
    fscanf(stream,"%s",s);
};
fscanf(stream,"%s",s); //丢弃其他数据
//读到第二个标志位"CTRIA3"
while(strcmp(s,"CTRIA3")!=0)
{
    fscanf(stream,"%s",s);
};
```

4.2.2 输出模块设计

输出模块是利用 MFC 中的 Listbox 设计，通过分析计算，使结果呈现在 Listbox 中，输出的信息应该为：入射方位角，入射俯仰角，观察方位角，观察俯仰角，辐射功率等信息；

用 CListBox::AddString int AddString(LPCTSTR lpszItem);这个函数来添加结果数据，并且将结果数据显示在 listbox 中。

例如：CString str1,str2;

str2="***** 三维图像软件计算输出 *****";

this->m_pFr->m_pCtrlWnd->m_List.AddString(str2,RGB(255, 0, 0));

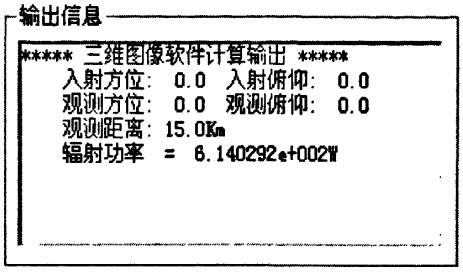

图 4-4　输出模块设计图

4.3　显示模块设计

显示模块设计主要采用 OPENGL 类库开发，使用类库相关函数正确显示图形以及进行相关图形计算。我们将综合相关 OPENGL 开发库进行设计：

4.3.1　OpenGL 相关开发库

开发基于 OpenGL 的应用程序，必须先了解 OpenGL 的库函数。它采用 C 语言风格，提供大量的函数来进行图形的处理和显示。OpenGL 库函数的命名方式非常有规律。所有 OpenGL 函数采用了以下格式：

<库前缀><根命令><可选的参数个数><可选的参数类型>

库前缀有 gl、glu、aux、glut、wgl、glx、agl 等等，分别表示该函数属于 OpenGL 那个开发库等，从函数名后面中还可以看出需要多少个 参数以及参数的类型。I 代表 int 型，f 代表 float 型，d 代表 double 型，u 代表无符号整型。例如 glVertex3fv() 表示了该函数属于 gl 库，参数是三个 float 型参数指针。我们用 glVertex*()来表示这一类函数。

OpenGL 函数库相关的 API 有核心库(gl)、实用库(glu)、辅助库(aux)、实用工具库(glut)、窗口库(glx、agl、wgl)和扩展函数库等。gl 是核心，glu 是对 gl 的部分封装。glx、agl、wgl 是针对不同窗口系统的函数。glut 是为跨平台的 OpenGL 程序的工具包，比 aux 功能强大。扩展函数库是硬件厂商为实现硬件更新利用 OpenGL 的扩 展机制开发的函数。下面逐一对这些库进行详细介绍。

1.　OpenGL 核心库

核心库包含有 115 个函数，函数名的前缀为 gl。

这部分函数用于常规的、核心的图形处理。此函数由 gl.dll 来负责解释执行。由于许多函数可以接收不同数以下几类。据类型的参数，因此派生出来的函数原形多达 300 多个。核心库中的函数主要可以分为以下几类函数。

绘制基本几何图元的函数。如绘制图元的函数 glBegain()、glEnd()、glNormal*()、glVertex*()。

矩阵操作、几何变换和投影变换的函数。如矩阵入栈函数 glPushMatrix()、矩阵出栈 函数 glPopMatrix()、装载矩阵函数 glLoadMatrix()、矩阵相乘函数 glMultMatrix()，当前矩阵函数 glMatrixMode()和矩阵标准化函数 glLoadIdentity()，几何变换函数 glTranslate*()、glRotate*()和 glScale*()，投影变换函数 glOrtho()、glFrustum()和视口变换函数 glViewport()等等。

颜色、光照和材质的函数。如设置颜色模式函数 glColor*()、glIndex*()，设置光照效果的函数 glLight*() 、glLightModel*()和设置材质效果函数 glMaterial()等等。

显示列表函数、主要有创建、结束、生成、删除和调用显示列表的函数 glNewList()、 glEndList()、glGenLists()、glCallList()和 glDeleteLists()。

纹理映射函数，主要有一维纹理函数 glTexImage1D()、二维纹理函数 glTexImage2D()、设置纹理参数、纹理环境和纹理坐标的函数 glTexParameter*()、glTexEnv*()和 glTetCoord*()等。

特殊效果函数。融合函数 glBlendFunc()、反走样函数 glHint()和雾化效果 glFog*()。

光栅化、象素操作函数。如象素位置 glRasterPos*()、线型宽度 glLineWidth()、多边形绘制模式 glPolygonMode()，读取象素 glReadPixel()、复制象素 glCopyPixel()等。

选择与反馈函数。主要有渲染模式 glRenderMode()、选择缓冲区 glSelectBuffer()和反馈缓冲区 glFeedbackBuffer()等。

曲线与曲面的绘制函数。生成曲线或曲面的函数 glMap*()、glMapGrid*()，求值器的函数 glEvalCoord*() glEvalMesh*()。

状态设置与查询函数。主要有 glGet*()、glEnable()、glGetError()等。

2. OpenGL 实用库 The OpenGL Utility Library (GLU)包含有 43 个函数，函数名的前缀为 glu。

OpenGL 提供了强大的但是为数不多的绘图命令，所有较复杂的绘图都必须从点。线、面开始。Glu 为了减轻繁重的编程工作，封装了 OpenGL 函数，Glu 函数通过调用核心库的函数，为开发者提供相对简单的用法，实现一些较为复杂的操作。此函数由 glu.dll 来负责解释执行。OpenGL 中的核心库和实用库可以在

所有的 OpenGL 平台上运行。主要包括了以下几种。

辅助纹理贴图函数，有 gluScaleImage() 、gluBuild1Dmipmaps()、gluBuild2Dmipmaps()。

坐标转换和投影变换函数，定义投影方式函数 gluPerspective()、gluOrtho2D() 、gluLookAt()，拾取投影视景体函数 gluPickMatrix()，投影矩阵计算 gluProject()和 gluUnProject()等等。

多边形镶嵌工具，有 gluNewTess()、 gluDeleteTess()、gluTessCallback()、gluBeginPolygon() gluTessVertex()、gluNextContour()、gluEndPolygon()等等。

二次曲面绘制工具，主要有绘制球面、锥面、柱面、圆环面 gluNewQuadric()、gluSphere()、gluCylinder()、gluDisk()、gluPartialDisk()、gluDeleteQuadric()等等。

非 均匀有理 B 样条绘制工具，主要用来定义和绘制 Nurbs 曲线和曲面，包括 gluNewNurbsRenderer()、gluNurbsCurve()、gluBeginSurface()、gluEndSurface()、gluBeginCurve()、gluNurbsProperty()等函数。

3. OpenGL 辅助库包含有 31 个函数，函数名前缀为 aux。

这部分函数提供窗口管理、输 入输出处理以及绘制一些简单三维物体。此函数由 glaux.dll 来负责解释执行。创建 aux 库是为了学习和编写 OpenGL 程序，它更像是一个用于测试 创意的预备基础接管。Aux 库在 windows 实现有很多错误，因此很容易导致频繁的崩溃。在跨平台的编程实例和演示中，aux 很大程度上已经被 glut 库取代。OpenGL 中的辅助库不能在所有的 OpenGL 平台上运行。

辅助库函数主要包括以下几类。

窗口初始化和退出函数，auxInitDisplayMode()和 auxInitPosition()。

窗口处理和时间输入函数， auxReshapeFunc() 、 auxKeyFunc() 和 auxMouseFunc()。

颜色索引装入函数，auxSetOneColor()。

三维物体绘制函数。包括了两种形式网状体和实心体，如绘制立方体 auxWireCube()和 auxSolidCube()。这里以网状体为例，长方体 auxWireBox()、环形圆纹面 auxWireTorus() 、圆柱 auxWireCylinder() 、二十面体 auxWireIcosahedron() 、八面体 auxWireOctahedron() 、四面体 auxWireTetrahedron()、十二面体 auxWireDodecahedron()、圆锥体 auxWireCone() 和茶壶 auxWireTeapot()。

背景过程管理函数 auxIdleFunc()。

程序运行函数 auxMainLoop()。

4. OpenGL 工具库 OpenGL Utility Toolkit 包含大约 30 多个函数，函数名前缀为 glut。

glut 是不依赖于窗口平台的 OpenGL 工具包，由 Mark KLilgrad 在 SGI 编写（现在在 Nvidia），目的是隐藏不同窗口平台 API 的复杂度。 函数以 glut 开头，它们作为 aux 库功能更强的替代品，提供更为复杂的绘制功能，此函数由 glut.dll 来负责解释执行。由于 glut 中的窗口管理函 数是不依赖于运行环境的，因此 OpenGL 中的工具库可以在 X-Window, Windows NT, OS/2 等系统下运行，特别适合于开发不需要复杂界面的 OpenGL 示例程序。对于有经验的程序员来说，一般先用 glut 理顺 3D 图形代码，然后再集成为完整的应用程序。

这部分函数主要包括

窗口操作函数，窗口初始化、窗口大小、窗口位置等函数 glutInit() glutInitDisplayMode() glutInitWindowSize() glutInitWindowPosition()等。

回 调函数。响应刷新消息、键盘消息、鼠标消息、定时器函数等，GlutDisplayFunc() glutPostRedisplay() glutReshapeFunc() glutTimerFunc() glutKeyboardFunc() glutMouseFunc()。

创建复杂的三维物体。这些和 aux 库的函数功能相同。创建网状体和实心体。如 glutSolidSphere()、glutWireSphere()等。

菜单函数。创建添加菜单的函数 GlutCreateMenu()、glutSetMenu()、glutAddMenuEntry()、glutAddSubMenu() 和 glutAttachMenu()。

程序运行函数，glutMainLoop()。

5. Windows 专用库针对 windows 平台的扩展。包含有 16 个函数，函数名前缀为 wgl。

这部分函数主要用于连接 OpenGL 和 Windows ，以弥补 OpenGL 在文本方面的不足。 Windows 专用库只能用于 Windows 环境中。

这类函数主要包括以下几类

绘图上下文相关函数 wglCreateContext(),wglDeleteContext() ，wglGetCurrentContent()，wglGetCurrentDC()，wglDeleteContent()等；

文字和文本处理函数 wglUseFontBitmaps()、wglUseFontOutlines()。

覆盖层、地层和主平面层处理函数： wglCopyContext() 、wglCreateLayerPlane()、wglDescribeLayerPlane()、wglReakizeLayerPlatte()等

其他函数：wglShareLists()、wglGetProcAddress()等。

6. Win32 API 函数库包含有 6 个函数，函数名无专用前缀，是 win32 扩展函数。这部分函数主要用于处理像素存储格式和双帧缓存。这 6 个函数将替换 Windows GDI 中原有的同样的函数。Win32API 函数库只能用于 Windows 95/98/NT 环境中。

7. X 窗口专用库是针对 Unix 和 Linux 的扩展函数。

包括渲染上下文、绘制图元、显示列表、纹理贴图、等等

初始化 glXQueryExtension() 渲染上下文函数，glXCreateContext() glXDestroyContext() glXCopyContext() glXMakeCurrent() glXCreateGLXPixmap()

执行 glXWaitGL()、glXWaitX() 缓冲区和字体 glXSwapBuffers()、glXUseXFont（）

8. 其他扩展库

这些函数可能是新的 OpenGL 函数，并没有在标准 OpenGL 库中实现，或者它们是用来扩展已存在的 OpenGL 函数的功能。和 glu、glx 和 wgl 一 样，这些 OpenGL 扩展是由硬件厂商和厂商组织开发的。OpenGL 扩展(OpenGL Extention) 包含了大量的扩展 API 函数。

随着硬件的更新，硬件厂商首先向 SGI 申请登记新的扩展，编写规格说明书（specification）。然后按照说明书进行开发扩展程序。不同的 OpenGL 实现（OpenGL Implementation）支持的扩展可能不一样，只有随着某一扩展的推广与应用以及硬件技术的提高该扩展才会在所有的 OpenGL 实现中被给予支 持，从而最终成为 OpenGL 标准库的一部分。扩展由 SGI 维护，在 SGI 网站上列出了目前公开的已注册的扩展及其官方说明书。

扩展源由扩展函数的后缀来指明(或使用扩展常量后缀)。例如，后缀 WIN 表明一个符合 Windows 规范的扩展，EXT 或 ARB 后缀表明该扩展由多个卖主定义。

下面给出 OpenGL 官方规定的命名规则：

ARB – OpenGL Architecture Review Board 正式核准的扩展，往往由厂商开发的扩展发展而来，如果同时存在厂商开发的扩展和 ARB 扩展，应该优先使用 ARB 扩展

EXT – 多家 OpenGL 厂商同意支持的扩展

HP – Hewlett-Packard 惠普

IBM – International Business Machines

KTX – Kinetix, maker of 3D Studio Max

INTEL – Intel 公司

NV – NVIDIA 公司

MESA – Brian Paul's freeware portable OpenGL implementation

SGI – Silicon Graphics 公司开发的扩展

SGIX – Silicon Graphics (experimental) 公司开发的实验性扩展

SUN – Sun Microsystems

WIN – Microsoft

由于 OpenGL 扩展在针对不同平台和不同驱动，OpenGL 不可能把所有的接口程序全部放到 gl.h、glx.h、wgl.h 中，而是将这些函数头放在了 glext.h、glxext.h 和 wglext.h 中。这些扩展被看作时 OpenGL 核心库规范的增加和修改。

OpenGL 扩展也不是没有缺点，正因为各个硬件厂商都可以开发自己的扩展，所以扩展的数目比较大，而且有点混乱，有些扩展实现的相同的功能，可因为是不同厂商开发 的，接口却不一样，所以程序中为了实现这个功能，往往要为不同的显卡写不同的程序。这个问题在 OpenGL 2.0 出来后可能会得到解决，OpenGL 2.0 的一个目标就是统一扩展，减少扩展数目。

4.3.2 CView 类 OpenGL 实现

0penGL 作图非常方便，故日益流行。OpenGL 在 VC 环境下的编程步骤：

建立基于 OpenGL 的应用程序框架：

创建项目：在 file -> New 中建立项目，基于单文档，View 类基于 CView

添加库：在 project->Setting 中指定库

初始化：选择 View->Class Wizard,打开 MFC 对话框，添加相应的定义

添加类成员说明

基于 OpenGL 的程序框架已经构造好，以后用户只需要在对应的函数中添加程序代码即可。

下面介绍如何在 VC++ 上进行 OpenGL 编程。 OpenGL 绘图的一般过程可以看作这样的,先用 OpenGL 语句在 OpenGL 的绘图环境 RenderContext (RC)中画好图, 然后再通过一个 Swap buffer 的过程把图传给操作系统的绘图环境 DeviceContext (DC)中,实实在在地画出到屏幕上.

产生程序框架 Test.dsw

New Project | MFC Application Wizard (EXE) | "Test" | OK

设置编译环境：

首先在 TestView.h 内各加上：

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glaux.h>
```

然后在集成环境中

Project | Settings | Link | Object/library module | "opengl32.lib glu32.lib glaux.lib" | OK

设置 OpenGL 工作环境：(下面各个操作，均针对 TestView.cpp)

1. 处理 PreCreateWindow(): 设置 OpenGL 绘图窗口的风格

```
cs.style |= WS_CLIPSIBLINGS | WS_CLIPCHILDREN | CS_OWNDC;
```

2. 处理 OnCreate():创建 OpenGL 的绘图设备。

OpenGL 绘图的机制是: 先用 OpenGL 的绘图上下文 Rendering Context (简称为 RC)把图画好，再把所绘结果通过 SwapBuffer() 函数传给 Window 的绘图上下文 Device Context (简记为 DC).要注意的是，程序运行过程中，可以有多个 DC，但只能有一个 RC。因此当一个 DC 画完图后，要立即释放 RC，以便其它的 DC 也使用。在后面的代码中，将有详细注释。

OnCreate()函数主要是初始化 OpenGL，构造窗体函数：

```
int CTestView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
if (CView::OnCreate(lpCreateStruct) == -1)
return -1;
myInitOpenGL();
return 0;
}
```

InitOpenGL()主要是用来初始化 OpenGL 相关资源 DC 与 RC：

```
void CTestView::myInitOpenGL()
{
//创建 DC
m_pDC = new CClientDC(this);
ASSERT(m_pDC != NULL);
//设定绘图的位图格式，函数下面列出
if (!mySetupPixelFormat())
return;
//创建 RC
m_hRC = wglCreateContext(m_pDC->m_hDC);
//RC 与当前 DC 相关联
wglMakeCurrent(m_pDC->m_hDC, m_hRC);
} //CClient * m_pDC; HGLRC m_hRC; 是 CTestView 的成员变量
```

SetUpPixelFormat()主要是用来设置 OPENGL 像素格式：

```
BOOL CTestView::mySetupPixelFormat()
{
```

```
static PIXELFORMATDESCRIPTOR pfd =
{
sizeof(PIXELFORMATDESCRIPTOR), // size of this pfd
1, // version number
PFD_DRAW_TO_WINDOW | // support window
PFD_SUPPORT_OPENGL | // support OpenGL
PFD_DOUBLEBUFFER, // double buffered
PFD_TYPE_RGBA, // RGBA type
24, // 24-bit color depth
0, 0, 0, 0, 0, 0, // color bits ignored
0, // no alpha buffer
0, // shift bit ignored
0, // no accumulation buffer
0, 0, 0, 0, // accum bits ignored
32, // 32-bit z-buffer
0, // no stencil buffer
0, // no auxiliary buffer
PFD_MAIN_PLANE, // main layer
0, // reserved
0, 0, 0 // layer masks ignored
};
int pixelformat;
if ( (pixelformat = ChoosePixelFormat(m_pDC->m_hDC, &pfd)) == 0 )
{
MessageBox("ChoosePixelFormat failed");
return FALSE;
}
if (SetPixelFormat(m_pDC->m_hDC, pixelformat, &pfd) == FALSE)
{
MessageBox("SetPixelFormat failed");
return FALSE;
}
return TRUE;
}
```

3. 处理 OnDestroy()，主要是释放 OPENGL 占用资源的函数，我们在这个函数里面主要负责释放删除 RC，删除相关 DC;

```
void CTestView::OnDestroy()
{
//释放与 m_hDC 对应的 RC
wglMakeCurrent(m_pDC->m_hDC,NULL);
//删除 RC
wglDeleteContext(m_hRC);
if (m_pDC)
//删除当前 View 拥有的 DC
delete m_pDC;
CView::OnDestroy();
}
```

4. 处理 OnEraseBkgnd() ,这个函数主要处理显示背景的刷新；

```
BOOL CTestView::OnEraseBkgnd(CDC* pDC)
{
// TODO: Add your message handler code here and/or call default
// return CView::OnEraseBkgnd(pDC);
//把这句话注释掉，若不然，Window
//会用白色北景来刷新，导致画面闪烁
return TRUE;//只要空返回即可。
}
```

5. 处理 OnDraw()，该函数主要用来画图，即绘制我们的三维图像函数；

```
void CTestView::OnDraw(CDC* pDC)
{
//使 RC 与当前 DC 相关联
wglMakeCurrent(m_pDC->m_hDC,m_hRC);
//具体的绘图函数，在 RC 中绘制
myDrawScene( );
//把 RC 中所绘传到当前的 DC 上，从而在屏幕上显示
SwapBuffers(m_pDC->m_hDC);
```

```
//释放 RC，以便其它 DC 进行绘图
wglMakeCurrent(m_pDC->m_hDC,NULL);
}

DrawScene()是主要的绘图函数，我们的绘图在里面进行：
void CTestView::myDrawScene( ) //主要的画图函数
{
glClearColor(0.0f,0.0f,0.0f,1.0f);//设置背景颜色为黑色
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glPushMatrix();
glTranslated(0.0f,0.0f,-3.0f);//把物体沿(0,0,-1)方向平移
//以便投影时可见。因为缺省的视点在(0,0,0),只有移开物体才能可见
//下面主要是画图函数
//选择框画正方体
 if(this->m_pFr->m_pCtrlWnd->m_bShowConsult)
//Draw3DRect 画正方体子程序
 this->Draw3DRect();
//画数据
 if(m_bGetFileSuccess)
{
    this->PutGraph();//数据输出函数
}
//坐标系和文字
if (this->m_pFr->m_pCtrlWnd->m_bShowCoordinate)
{
    this->DrawLines();
    this->DrawText();
}
//强制绘图完成
glFinish();
//交换缓冲区数据
SwapBuffers(wglGetCurrentDC());
//具体的函数见附录
 glPopMatrix();
```

```
//结束 RC 绘图
glFlush();
return;
}
```

6. 处理 OnSize()，当我们对图像大小进行操作，主要重写这个函数；

```
void CTestView::OnSize(UINT nType, int cx, int cy)
{
CView::OnSize(nType, cx, cy);
//确认 RC 与当前 DC 关联
VERIFY(wglMakeCurrent(m_pDC->m_hDC,m_hRC));
w=cx;
h=cy;
//确认 DC 释放 RC
VERIFY(wglMakeCurrent(NULL,NULL));
}
```

通过以上步骤，对 OpenGL 相关函数进行书写，基本能够达到显示模块要求，更具体的功能需要书写其他函数。

4.4 数据计算模块设计

数据计算模块主要功能：

- 读取相应格式的数据文件，解析数据文件并进行数据存储；
- 利用内存数据进行相关三维坐标变换计算；
- 消隐算法计算遮挡关系；

4.4.1 数据文件读取

我们对三维物体[7]进行细小的划分，将每个三维物体划分成细小的三角形组成，一个物体就是成千上万的小三角形，我们的数据文件包含的就是三角形每个点的坐标，数据文件按照一定文件格式生成，我们将数据文件内容分为两部分：

第一部分：点的坐标，我们包括内容为点的表示符 GRID，点的序列号，每个点包含的 X,Y,Z 三个坐标值；

例如：GRID 1 -3400.0 -50.0 250.0

该点是序列号为 1 的点，该点坐标为（-3400，-50，250）；

第二部分：三角形组成，内容为三角形标识符 CTRIA3,三角形序列号，组成三角形三个点的序列号；

例如：CTRIA3　1　523　92　67

该三角形为序列号为 1 的三角形，该三角形是由序列号为 523，92，67 的三角形组成；

我们对以上数据文件通过使用函数 getFileData()获取；

该函数内部包含 CFile 基本操作，将文件数据读入相应的临时缓存中，Databuf[No][0]代表数据点 No 的 X 轴坐标,Databuf[No][1] 代表数据点 No 的 Y 轴坐标,Databuf[No][2] 代表数据点 No 的 Z 轴坐标：

```
fscanf(stream,"%8f",&Databuf[No][0]);// 读数据点的 x 值
fscanf(stream,"%8f",&Databuf[No][1]);// 读数据点的 y 值
fscanf(stream,"%8f",&Databuf[No][2]);// 读数据点的 z 值
```

4.4.2 三维坐标变换

选择合适的观察坐标系不但可以更好地描述物体，而且可以大大简化和降低消隐算法的运算。因此，利用物体空间法进行消隐的第一步往往是将物体所处的坐标系转换为适当的观察坐标系。这需要对物体进行三维旋转和平移变换。

这里，我们软件数据模型是围绕坐标原点建模，所以我们对三维物体坐标系不考虑平移转换，我们只需要对物体进行旋转变换，让物体变换到适当的观察坐标系。

首先我们知道，我们使用的物体模型是三维模型，三维模型有 x,y,z 三维坐标，我们想要从不同角度照射物体观察物体就必须转动物体，但最终我们是通过计算机屏幕这个不动的平面观察结果，所以我们必须对不同的角度进行三维坐标变换，实际上变换后变动的是坐标轴，相当于我们对物体进行旋转，观察平面不变，那么坐标轴必须变动我们才能看到我们想要看到的结果，总而言之，我们对物体的变换最后是基于坐标轴变换得出的；

对于三维物体坐标变换，我们需要了解基本知识如下：

方位角：从某点的指北方向线起，顺时针方向至目标方向线的水平夹角。

在该软件坐标系里面的定义是：从 x 轴正方向向 y 轴正方向逆时针旋转的角度；

俯仰角：在同一铅垂面内，当视线在水平线上方时，视线与水平线的夹角叫仰角。在同一铅垂面内，当视线在水平线下方时，视线与水平线的夹角叫俯角。

在该软件坐标系里面的定义：与 xoy 平面成正数度数的角是仰角，与 xoy 平面成负数度数的角是俯角；

因为我们涉及不同方位俯仰角，我们在计算的过程中需要将数据根据设置的

角度值进行相关坐标转换，屏幕看到物体的角度应该是观察者角度下的俯仰为0度，方位角度为0度角度，所以我们应该基于屏幕这个基本参考角度不变情况下对物体入射光设置以及观察设置进行相应变换，默认 OPENGL 光照方向是从屏幕里沿法线指向外，即光线方向是从我们坐标系定义的 z 轴正方向指向负方向，即该软件方位角为0度，俯仰角为+90 度的方向；

在讨论旋转的时候理解参照系是重要的。从一个观点，你可以保持坐标轴固定旋转向量。从另一个观点，你可以保持向量固定旋转坐标系。

在第一种观点看来，坐标或向量关于原点的逆时针旋转；或者从第二种观点看来，平面或轴关于原点的顺时针旋转。这里的 (x,y) 被旋转了 θ 并希望知道旋转后的坐标 (x',y')：

$$x = x\cos\theta - y\sin\theta$$
$$y = x\sin\theta + y\cos\theta \qquad \text{公式（4-1）}$$

我们的策略是：固定入射方向（即光线永远都是从屏幕外照向屏幕内），相应的入射角为旋转坐标轴得出，然后利用三维坐标变换公式进行计算；显示计算时候我们会最终对物体进行旋转（相对不变的是观察方向，也是从屏幕外到内），这时我们看到的物体即为观察角度设置后的物体旋转后的图形；

具体做法：先对入射光进行变换，首先变换方位角，即对计算机实际三维坐标的（x，z）进行变换；然后进行俯仰变换，即对计算机实际三维坐标的（y，z）进行变换；

对观察角度进行变换，采取和入射光变换一样的策略；

例如：

//根据入射光线的方位和俯仰进行坐标变换

判断入射光线方位角是否为0

如果方位角为 0，直接将 Databuf[No][0],Databuf[No][2]的值赋给临时变量 DataDot[No][0],DataDot[No][2]；

不为0，进行旋转变换，将方位变换到0度观察角，相当于旋转坐标轴：

temp1=Databuf[i][0]*CosAngle-Databuf[i][2]*SinAngle;

temp2=Databuf[i][2]*CosAngle+Databuf[i][0]*SinAngle;

DataDot[i][0]=temp1;

DataDot[i][2]=temp2;

判断入射光线俯仰角是否为0；

如果俯仰角为 0，直接将 Databuf[No][1],Databuf[No][2]的值赋给临时变量 DataDot[No][1],DataDot[No][2]；

不为0，进行旋转变换，将俯仰变换到0度观察角，相当于旋转坐标轴：

```
temp1=Databuf[i][1]*CosAngle-Databuf[i][2]*SinAngle;
temp2=Databuf[i][2]*CosAngle+Databuf[i][1]*SinAngle;
DataDot[i][1]=temp1;
DataDot[i][2]=temp2;
```

4.4.3 消隐算法（Z-Buffer 算法）

常用的物体空间消隐算法[8]包括平面公式法、背面消除法、径向预排序法、径向排序法、隔离平面法、深度排序法、光线跟踪法和分解法。其中前四种算法最常用，它们的基础都是背面消隐原理。所谓背面消隐原理[9]，即是相对观察点来说朝向后面的物体表面是不可见的，应被隐藏。

1. 平面公式法

根据解析几何原理，通过标准的平面方程可以判断给定点是在平面的正面还是背面。平面公式法利用此原理来判断观察点位于物体表面的哪一面，如位于背面一侧，则表面不可见，应被消隐；反之则可见。

对物体得任意表面，可将其划分为若干个平面，在根据平面上任意三点的坐标可以求 得其平面方程。标准的平面方程为：

Ax+By+Cz+D = 0;

其 中 A、B、C、D 为决定平面得常数。 如 果（x1, y1, z1）、（x2, y2, z2）、（x3, y3, z3）为平面上已知得三点坐标， 则可求得 A、B、C、D 如下：

$$A = y1(x2-x3)+y2(z3-z1)+y3(z1-z2);$$
$$B = z1(x2-x3)+z2(x3-x1)+z3(x1-x2);$$
$$C = x1(y2-y3)+x2(y3-y1)+x3(y1-y2);$$
$$D = -x1(y2z3-y3z2)-x2(y3z1-y1z3)-x3(y1z2-y2z1); \quad \text{式（4-2）}$$

设观察点坐标为（x, y, z），如果

Ax+By+Cz+D = 0，则观察点（x, y, z）位于平面上；

Ax+By+Cz+D > 0，则观察点（x, y, z）位于平面背面一侧，平面不可见，应被隐藏；

Ax+By+Cz+D < 0， 则观察点（x, y, z）位于平面正面一侧，平面是可见面，应被画出。

通过对物体进行适当旋转和平移后，可将物体变换到以观察点为原点得观察坐标系中，如果在观察坐标系中求得了平面得方程 Ax+By+Cz+D = 0，将观察点坐标（0，0，0）代入上面得判断准则，则可得出如下的简单判据：

D > 0， 则平面不可见，应被隐藏；

D < 0， 则平面是可见面，应被画出。

平面公式法算法简便，是在实际中使用最频繁的消隐算法。但它只能用于凸面体的消隐，而不适用于凹面体消隐。

2. 背面消除法

背面消除法是直接运用背面消隐原理的消隐算法。在数学上，物体表面的法向量即是表面的朝向，因此，法向量方向背向观察点的物体表面都应被消隐。表面的法向量是否背向观察点可以通过表面法向量与视向量的点积来决定。设经坐标变换后，坐标系的原点 O 即是观察点，空间中任意平面 ABC 的法向量为，法向量为与平面的交点为 P，则从向量 OP 即是平面 ABC 的视向量。

如果>0，则物体表面是可见的朝向观察点的面；如果，则物体表面是不可见的背向观察点的面，应被消隐。

设 θ 为向量和之间的夹角，视向量的长度为线段 OP 的长度|OP|，则根据向量点积的定义可知=|OP|cosθ。如果点积大于 0，则 cosθ>0（即>θ>0）；反之，如果点积小于 0，则 cosθ<0（即 θ<0）。

因此，背面消除法的判据简化为：

cosθ<0，则物体表面不可见，应被消隐；

cosθ>0，则物体表面可见，应被画出。

根据平面法向量的定义可知，在平面上按逆时针方向选取 P1（x1,y1,z1）、P2（x2,y2,z2）、P3（x3,y3,z3）三点，则

经过投影变化后，视向量与 Z 轴是平行的，因此向量和之间的夹角 θ 即为 Z 轴与向量的夹角，所以

由于绝对值大于 0，所以 cosθ 的正负取决于 C，因此背面消除法的判据转化化为：

C<0，则物体表面不可见，应被消隐；

C>0，则物体表面可见，应被画出。

3. 径向预排序法

径向预排序法根据物体在三维坐标系 XY 平面中的角位置来判断哪些物体挡住了其它物体，物体的哪些表面挡住了其它表面。对具有相同角位置的物体或表面，与观察点较近的将挡住较远的。

径向预排序法消隐的要点是先对物体及物体的表面进行出远及近的排序，对具有相同角位置的物体或表面，先画较远的，后画较近的，这样如果较近的物体或表面挡住了较远的物体或表面，则被遮挡的部分被覆盖而实现消隐。但对具有不同角位置的物体或表面，先画哪一个可根据需要来决定。如果存在凹面物体的消隐，一般应先画物体中心部分，再画物体的两侧，以正确地表现互相重叠的凹

面模型。

径向预排序法可以对任意形状的物体进行消隐处理。但需要预先知道观察角度，并根据角位置对物体的画图顺序预先排序。而且构造模型的编码受到这种排序的限制，模型不能进行旋转变换。

4. 径向排序法

径向排序法是对径向预排序法的改进算法，使得构造模型的编码能根据观察角度的变化，来自动调整物体或表面的远近顺序即画图顺序，以实现对模型的旋转变换，以便能从不同的角度来观察物体。算法需要检测旋转变换的角度，并随角度的变化而调整物体或表面的远近顺序。

5. 隔离平面法

隔离平面法主要用于多个物体之间的消隐处理，其基础是平面公式法。其基本原理是，在需要进行消隐处理的两个物体之间建立一个虚拟平面，并根据平面公式法判断出两个物体分别位于该平面的哪一侧，以及该平面的哪一侧朝向观察点，则可以推论得到位于平面朝向观察点一侧的物体离观察点较近，将遮挡位于平面背向观察点一侧的物体。即位于平面背向观察点一侧的物体应被首先画出，且应进行消隐。

6. 深度排序法

深度排序法也是主要用于分析多个物体之间是否存在表面遮挡的消隐算法。其原理是比较不同物体或表面的表示远近的 z 坐标，在观察点位于原点的观察坐标系中，$|z|$值越大的物体或表面离观察点越远，被消隐的可能性越大，应先画出；$|z|$值越小的物体或表面离观察点越近，将可能遮挡较远的物体或表面，应后画出。

深度排序法需要用到深度信息和绘图顺序，通常用于模型数据中包含深度信息和绘图顺序的物体造型。

7. 光线跟踪法

光线跟踪法的基本原理是，人能看见物体是因为物体能反射光，因此，跟踪从光源发出的光线，光线投射到物体上，再从物体反射到观察点，在光线轨迹中离观察点最近的物体表面将遮挡其它物体表面。

光线跟踪法需要分析物体表面的每一点的光反射状态，因此需要的内存空间较大，运算速度也较慢。但这种方法可同时生成物体的光照模型，产生的消隐效果和真实感都很好。

8. 分解法

分解法是对 CSG 模型的一种消隐算法，首先将复杂物体分解为一系列的立方体，离观察点近的立方体将遮挡远的立方体，从而实现消隐。分解法算法复杂，需要的内存空间大，速度也满，近仅用于一些特殊的场合。

图象空间法

图象空间法基于物体三维模型的二维显示图形来确定物体或表面上的每一点与观察点的远近关系，从而判断哪些表面遮挡了其它表面。为了获得三维物体的二维显示图形，在对物体进行旋转和平移变化后，还需对物体进行透视投影变换。

三维空间中点 P(x,y,z)由透视点 E 沿 Z 轴透视投影变换到 XOY 二维平面中的点 P*(x*,y*,z*)。设 E 点距原点 O 的距离为 r，则透视投影变换公式为：

（x*,y*,z*,1）=（x,y,z,1）MPeMPr

式中 MPe、MPr 分别为透视变换矩阵和投影变换矩阵；

通过上面消隐算法比较，我们采取 Z 缓冲区法[10]：

Z 缓冲区[11]法首先建立一个大的缓冲区，用来存储三维物体沿 Z 轴透视投影而得到的二维图形的所有象素的值，因此叫做 Z 缓冲区。Z 缓冲区的单元个数与屏幕上象素点的个数相同，也和帧缓冲区的单元个数相同，而且它们之间是一一对应的。Z缓冲区每个单元的大小取决于图形在观察坐标系中Z方向的变化范围。Z 缓冲区的每个单元的值是对应象素点所对应的物体表面点的 Z 坐标值。

利用 Z 缓冲区法进行消隐和造型的过程就是对屏幕中每一点进行判断并给帧缓冲区和 Z 缓冲区中相应单元进行赋值的过程。现用形式化语言描述该算法如下：

Z 缓冲区消隐算法：

{

1）将帧缓冲区各单元的值置为背景色值;

2）将 Z 缓冲区各单元的值置为 Z 坐标可能出现的最大值;

3）循环：对每一物体

{

循环：对物体每一面的每一点（x,y,z）

{

i)对（x,y,z）做透视投影变换，得到变换后的 X、Y 坐标（x*,y*）；

ii)如果 Z 缓冲区中（x*,y*）对应单元的值小于 z，则

{

a)将 Z 缓冲区中（x*,y*）对应单元的值置为 z;

b)将帧缓冲区中（x*,y*）对应单元的值置为点（x,y,z）的属性值（通常是亮度、颜色值或颜色查找表的索引值）；

}

iii)如果 Z 缓冲区中（x*,y*）对应单元的值大于 z，则

{

a)说明目前帧缓冲区中（x*,y*）对应单元的所表示的物体上点比点（x,y,z）更接近观察点，即点（x,y,z）应被消隐；

b)将 Z 缓冲区和帧缓冲区中（x*,y*）对应单元的值均保持不变；

}

}

}

4）循环：对屏幕上每一点（x*,y*）

根据帧缓冲区中（x*,y*）对应单元的值画出象素点。

}

Z 缓冲区消隐算法简单、可靠，而且消隐和表现效果很好。但需要的内存容量大，运算复杂，费时。

第五章　结论分析

本软件结果前提条件：假定太阳光为平行光，照射到物体后计算一次反射后的辐射功率并且用图形界面显示图形被照射的结果；对比不同入射角度和观察角度下物体辐射功率以及阴影显示，能够记录不同位置的结果；

5.1　软件计算结果比较

首先打开应用程序：默认观察物体距离为 15KM，入射方位俯仰角都为 0 度，观察方位俯仰角都为 0 度，默认光照方向为屏幕外指向屏幕里；

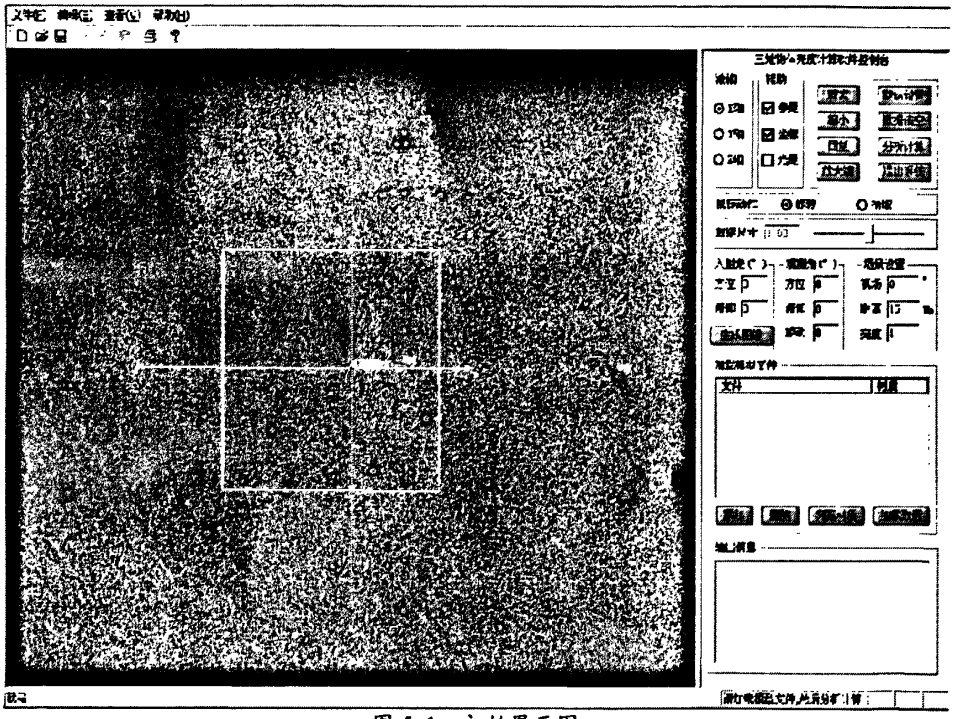

图 5-1　初始界面图

加载模型文件：

首先添加数据，选择材质文件→板材，文件加载圆柱体三维空间数据，圆柱体三维图像会加载到显示界面；

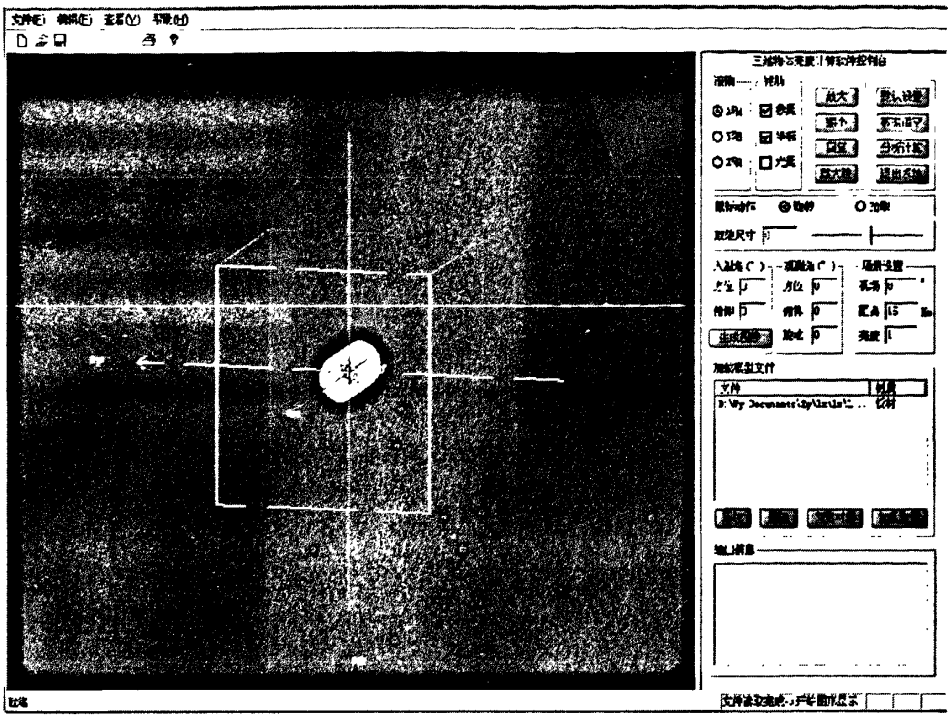

图 5-2　加载文件后显示物体

1.分析计算入射方位俯仰角度都为 0 度，观察方位俯仰角度都为 0 度的情况如下：

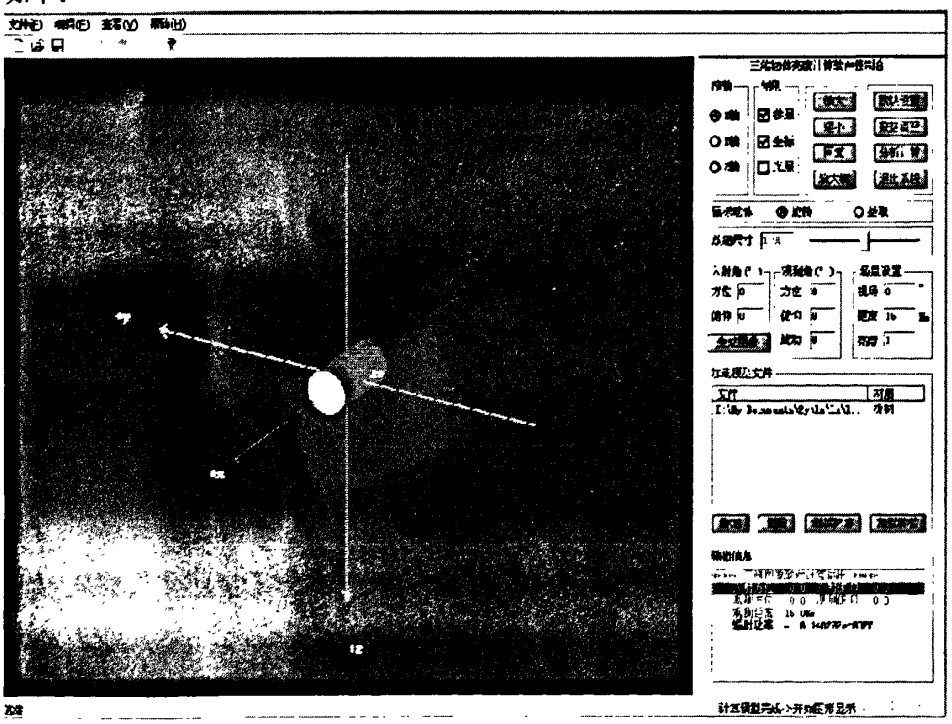

图 5-3　分析计算图 1

结论：可以看到入射光方位俯仰角度都为 0 度，观察方位俯仰角都为 0 度时，物体光亮度信息分布情况如图 5-3，光辐射功率为 6.14e+002W。

2.分析计算入射方位俯仰角度都为 0 度，观察方位俯仰角度都为 30 的情况：

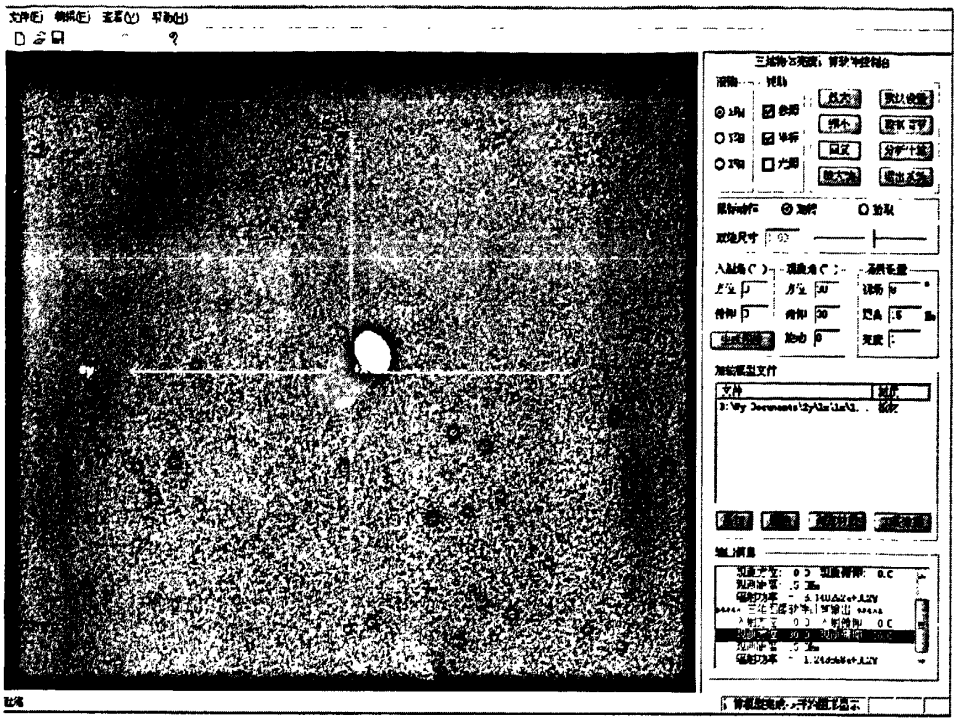

图 5-4　分析计算图 2

结论：可以看到入射光方位俯仰角度都为 0 度，观察方位都为 30 度时，物体光亮度信息分布情况如图 5-3，物体绕坐标轴旋转，我们看到图 5-4 物体即为我们观察到的物体姿态，光辐射功率为 1.24e+002W。

3.分析计算入射方位角度为 0 度俯仰角度为 90 度，观察方位俯仰角度都为 0 度的情况：

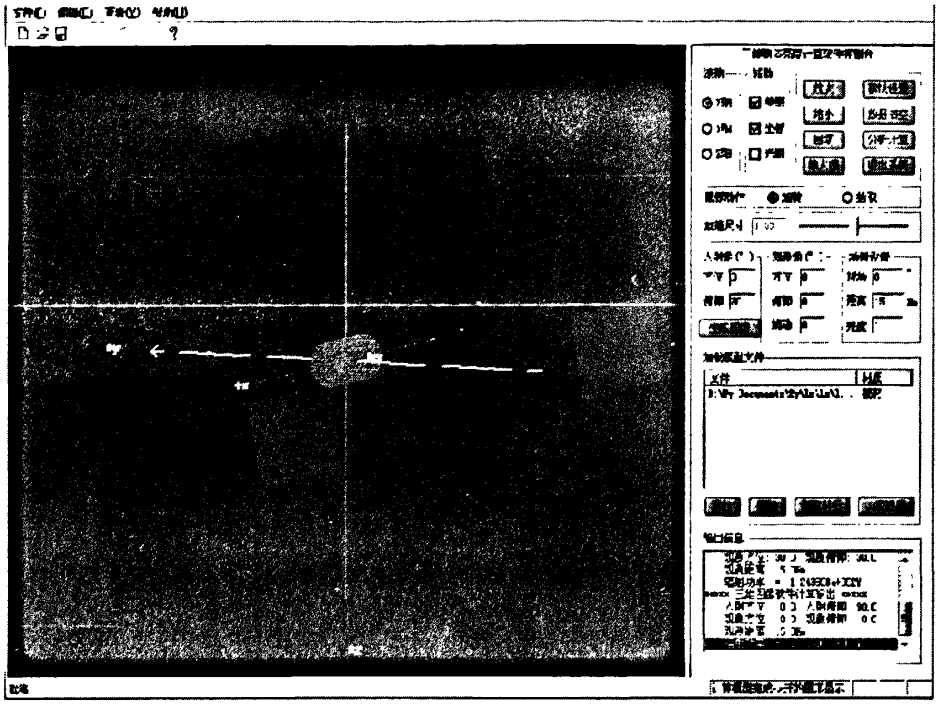

图 5-5　分析计算图 3

结论：可以看到入射光方位角为 0 度，俯仰角度为 90 度，观察方位俯仰角都为 0 度时，物体光亮度信息分布情况如图 5-5，光辐射功率为 0W，即我们看不到任何光照信息。

4.分析计算入射方位角度为 30 度，入射俯仰角度为 0 度，观察方位角度为 30 度，观察俯仰角度为 0 度的情况：

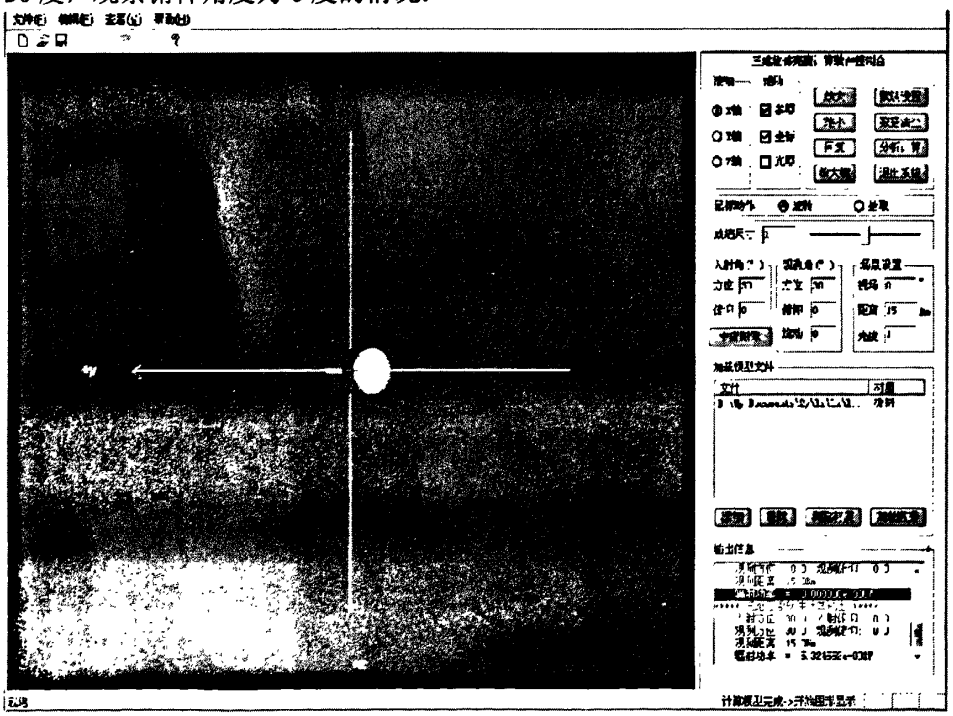

图 5-6　分析计算图 4

结论：可以看到入射光方位角为 30 度，俯仰角度为 0 度，观察方位角度为 30 度，俯仰角度为 0 度时，物体光亮度信息分布情况如图 5-6，光辐射功率为 5.32e+001W。

5. 分析计算入射方位角度为 0 度，入射俯仰角度为 30 度，观察方位角度为 0 度，观察俯仰角度为 30 度的情况：

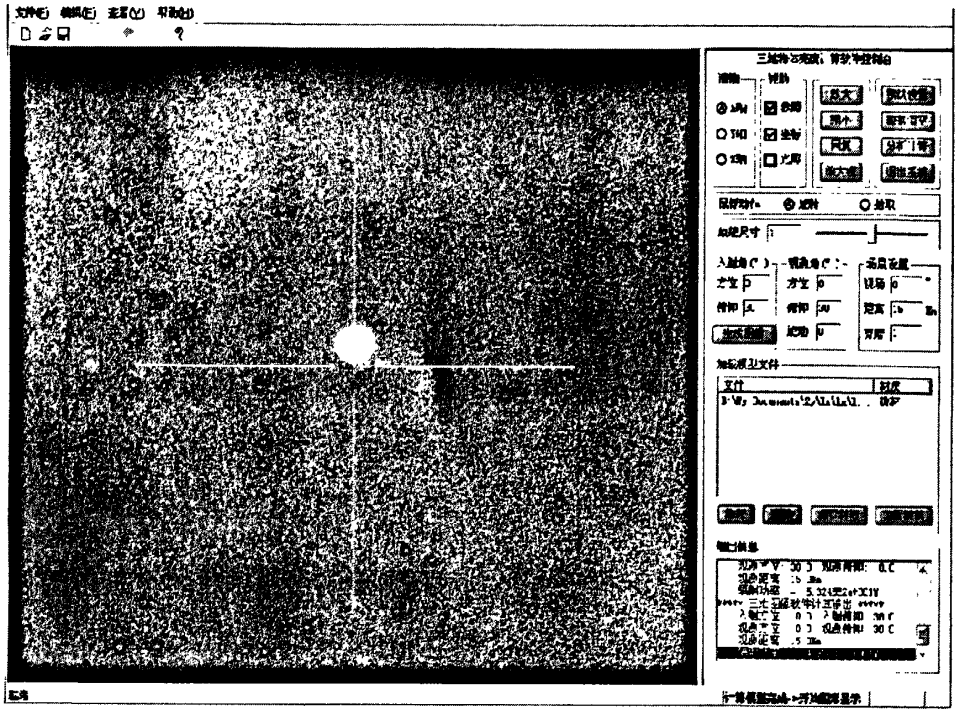

图 5-7　分析计算图 5

结论：可以看到入射光方位角为 0 度，俯仰角度为 30 度，观察方位角度为 0 度，俯仰角度为 30 度时，物体光亮度信息分布情况如图 5-6，光辐射功率为 4.95e+001W。

5.2　结论分析

综合上面结果比对：我们可以看出，当入射角度和观察角度一致的时候，光反射后的辐射功率最强，当入射角度和观察角度偏转比较大，反射后的辐射功率会逐渐减弱，当成正交，即入射角度和观察角度成 90 度时，基本没有反射后的辐射功率；当物体入射角度和观察角度偏转一样时，比如俯仰角差 30 度或者方位角差 30 度，由于物体旋转后，反射光面积不同，辐射功率也不同，面积越大，辐射功率越大；

该软件使用 VC6.0 开发环境，使用 MFC 开发窗体程序，结合 OpenGL 可视化接口，能够满足不同光学场景设置，正确得出光学亮度信息在物体上的分布，给光学计算带来可视化输出以及数据计算。

该软件待改进的地方：

1. 能够处理点光源情况；
2. 能够处理复杂物体，尤其凹凸平面物体，考虑二次反射以及环境混合反射情况；
3. 实时计算速度更快；

参考文献

[1] 徐俊波，李晓红 基于MFC的OpenGL三维图形类的创建 科学技术与工程 17期 2005年 1310-1313

[2] 许四平 基于MFC的OpenGL图形开发 硅谷 23期 2009年 87+71

[3] 严栋铭，钟艳如 基于VC++和OpenGL的STL文件读取显示 计算机系统应用 03期 2009年 172-175

[4] 张华 基于OpenGL的Phong明暗处理软件实现 计算机工程与设计 04期 2009年 1003-1004+1007

[5] 蒋万秋， 赵云峰，袁水平 海面背景红外图像建模与OpenGL仿真电光与控制 11期 2009年 19-21+49

[6] 林培炎，冯开平 采用VC++与OpenGL的三维场景编辑系统的研究与设计 工程图学学报 05期 2009年 58-62

[7] 周莉，苏鸿根 通用3D模型文件格式和算法的研究及其OpenGL实现 计算机工程与设计 02期 2009年 433-436+439

[8] 靳海亮，高井祥 图形消隐算法综述 计算机与数字工程 09期 2006年 27-31

[9] 夏小玲 三维消隐算法研究 东华大学学报(自然科学版) 02期 2002年 137-142;

[10] 简学东，陆玲，莫桂花 Z缓冲消隐算法的改进 计算机应用与软件 09期 2007年 149-150

[11] 生滨，李东，徐学敢 Z缓冲区消隐算法的改进 计算机工程与应用 23期 2001年 114-116

致谢

在本论文完成之际，我要特别感谢我的导师任建华教授。是任老师给了我研究生学习的机会，以及接触研究项目的机会。任老师两年来对我的学习和生活都给予了悉心的指导。每次和您谈话，都能感受到他对科研的严谨治学态度，对学生的认真负责和关心。他忘我的工作精神也激励我不断克服困难和进取。

感谢赵同刚副教授。赵同刚副教授两年来对我的学习和生活都给予了悉心指导，帮助。每次和您交谈，都能感受到您对我的科研水平进一步发展的期望，您的教导是我不断进取的动力。

感谢于淼博士，张志强博士，苏宇博士，在我的研究生求学期间，他们给予了我最多的科研指导，带着我一步步深入自己的研究项目，并帮助我排除困难，完成了本论文的研究成果。

感谢实验室所有的博士，硕士同学们。和他们在一起的学习和生活，让我感觉到愉快，每一次交流都有所收获。他们是李枭，邱洁，齐立荣，周勇发，喻超，李乐坚，徐爱梅等。

感谢一起生活的室友和在北邮结识的新朋友，他们是肖巍、庄浩、苏天醒、郭峰、吴瑜等。

感谢陈铮同学多年来对我学习生活上的帮助与指导。

最后要感谢我的父母，家人，我的好朋友们。你们是我生活中最坚固的后盾，是我不断前进，不断追求的最强大动力。没有你们的鼓励和支持，我无法顺利完成我的研究生学业。

攻读学位期间发表的学术论文目录

[1] 罗希 RLE-8 格式的 BMP 图像解码 中国科技论文在线 200912-435 2009 年 12 月 14 日