前言

很久就想这篇东西，　可一直懒于动脑且为一些俗事烦恼，　所以才拖到现在才动笔把我上学期间一篇论文整理一下，仅供参考。我分成几段来写，　如果想看懂这篇文章，那首先要了解每段之前的关键词，　不懂的可以在网上搜一下。　然后还有看这篇文章的时候希望你能按自己的需要来看，　有很多是写给自己看的，　你只挑自己有用的看就行了，　省得到时候埋怨我浪费你宝贵的时间，我这人写东西不行，尽量写的有条理，尽量吧。

第 1 章　意义，它是三维造型软件的内核、基础。

关键词：三维造型软件、虚拟现实技术、仿真技术、　OpenGL、几何内核系统、布尔操作

首先，做什么事情都要有意义，因此我先说明一下　Open CASCADE　到底是个什么类型的工具，　使用这个工具是否对你将要完成的目标有帮助。　如果有帮助，　那么你可以继续往下看它的具体用法；　如果没有，　那么你就应该按照你的目标继续寻找实现目标的手段，　不用把时间浪费在这篇文章上；如果你清楚了　Open CASCADE　的这一工具是干什么用的就可以跳过意义直接看下一章。

我先说一下我对这个工具的理解，然后会贴一些网上对这个工具的说明。　Open CASCADE　是一款三维造型软件的内核系统，也就是制作三维造型软件的工具。

那何为三维造型软件呢？其实市面上已经有很多这样的软件了，比如说　PRO/E、UE、Solid Edge、Catia、国内还自主研发的金银花系统（好象叫这个名字）以上这些大多都是用于工业上的，　3DMAX　、VRML　（这里要说明一下　VRML　其实是一款适合网络的三维造型语言，应属语言系列，和　HTML　语言类似，不太附和三维造型软件的标准）也是三维造型软件，当然还有很多很多这种类型的软件，　就不做过多介绍。三维造型软件，　可以在电脑上如实地（尺寸、颜色、材质等）搭建起一个虚拟的模型，这些模型可以用于工业设计、艺术设计、装修设计等等等一系列的虚拟现实技术相关的行业，　也可以用来做一些仿真实验的模型

构建。可能有很多人都已经用过一款或两款三维造型软件，但是你在用的时候想没想过三维造型软件是怎么做出来的呢？它的底层是由什么支持的？它是由两部分支持的，一是硬件支持，如显卡，这个我们不讨论；二是软件，作为软件，我猜想（注意是我猜的，如有疑问请查实）它也是可以分成两个部分，一是硬件驱动的标准体系（请关注 OpenGL），二就是几何内核系统。硬件支持和硬件驱动标准都不是本篇要讨论的内容，这里重点介绍的是几何内核系统。

如果你的目标只是构建一个三维模型用来做仿真实验或做一个虚拟现实环境，那么你可以选用三维造型软件来完成。现在已有的三维造型软件发展的已经很好了（我当年用的第一款是 Solid Edge7.0，看看现在它的最新版本号吧，饿的神啊），不仅品种多，而且相当人性化，你想到了它也想到了，你没想到的它也想到了（注意这句话），恩，很好用。这里我建议，如果你能使用现有的三维造型软件完成你的工作任务，请尽量使用它们，原因有三：第一、站在巨人臂膀上能看的更远；第二、使用几何内核系统开发出来的三维造型软件这一过程是相当艰辛的，且不一定能有什么好的结果；第三、你真的确定三维造型软件不能完成你的工作目标吗？举个例子，Pro/E 里有 N 多模块可供使用，你平常使用的可能只是零件、装配、工程图、板金、管道、电气模块这几个常用的，有些模块你可能从来就没见到过，但是你应该看看帮助或在网上查一下是否能用上其他的模块，如果能用上，那真要恭喜你了；还有，可以用一些编程工具，如 VB、VC 之类的开发 Pro/E 的模块，我没试过，只是听说来的。

什么是几何内核系统？它是制作三维造型软件的工具，三维造型软件都是在几何内核系统的基础上制作完成的，因此一款几何内核系统的好坏直接决定着三维造型软件的优劣。

那么我为什么要选择使用 Open CASCADE？我的专业是机械设计，现在实验室里我们专业都在搞虚拟现实之类的项目，比如说 VRML 与 MATLAB 结合绘制地图、OpenGL 碰撞检测、虚拟装配（我哥们做的，和我基本类似，但由于一些原因他没做完）、虚拟造型（这就是我的活）、机械人仿真（UG 造型、用一个仿真软件做的仿真，名字忘了），说说我的吧，要求是做一个课件（很郁闷，毕业设计竟然是做课件），但还不能说做课件，因为这种东西是毕不了业的，课件的要求是在一个虚拟的环境下，构建各种模型、对这些模型进行必要的布尔操作及显示模型的三视图。问题来了，解决吧。三个要求，一是建模；二是布尔操作；三是三视图的制作。

为了完成课题，我尝试了两种工具，　　VRML　和 Pro/E 建模。

第一，先说　VRML 。导师一开始想让我用　　VRML　来做课题，一开始我也不懂就用了，简单说一下　VRML　吧（从网上的摘了一篇关于　　VRML　的，见附录　1-《什么是　VRML 》），VRML　（Virtual　Reality　Modeling　Language）即虚拟现实建模语言。是一种用于建立真实世界的场景模型或人们虚构的三维世界的场景建模语言，也具有平台无关性。是目前　　Internet 上基于　WWW　的三维互动网站制作的主流语言。见过网页上很炫的　　FLASH 吗?简单的说，VRML　就是三维的　FLASH ，是 FLASH　的升级版（ 3D 版），它就是为了网络的虚拟现实化而生的，与　FLASH　不同的是，它是一种标准化语言，你可以很方便的制作出一个虚拟的世界。

为什么不用　VRML　做课题？

VRML　的优点：简单易用；缺点：不灵活。它就不适合建模，太麻烦，需要依靠其他的三维造型软件来完成。　要它实现布尔操作想都别想，　三视图更别提了，　手画比它要简单些。所以课题不能用　VRML　来完成，不过到用　VRML　做了两个小玩意，一个是虚拟装配（有点意思），二是　VRML　与 MA TLAB　画相贯线（这个写了论文，很扯淡）　。

第二个是　Pro/E 建模。 Pro/E 就不用我多说了，有很多工科专业都会用到这个软件，建模方面很强大。我当时学　　Pro/E 的时候还是瀑布式菜单，一开始挺挺别扭的，现在用的还挺习惯。 现在 Pro/E 野火版中大部分的命令都改成了更为人接受的工具条式命令，算是个不小的改进了，　但还保留了很多当年瀑布式菜单的影子，　　用来做留念还是技术水平达不到就不得而知了。以前给教研室里的老师们上过　　Pro/E 的课，虽然讲课时都把我导师讲的睡着了，但是 Pro/E 我是学的比较精通了，每个命令都仔细研究了下，但实际应用就差很多，毕竟光是上课，又不做什么项目，没得到什么实战锻炼。

为什么不用　Pro/E 做课题？

不选 Pro/E 做课题的最主要原因就是：因为　　Pro/E 太强大了，强大到当时我实在想不出怎么用它来进行我的毕业设计，我总不能答辩的时候打开　　Pro/E　去给那几个老师演示一下 Pro/E 建模吧，　那样我 3007 年也毕不了业了。其实这只是当时的想法，　　现在想起来，有一点我没有想到，那就是　Pro/E 的模块开发，做一个模块来实现我需要的功能就行了，我看到过

关于 Pro/E 模块开发的文章，可惜晚了，我不会有机会再做这个实验了，我也不想有。

题外话 ------

不经意的时候会思考一些问题，读研的时候到底学到了什么，那人生中的三年到底有没有意义，如果有机会再做选择的话还会选择那条路吗？在这里我想说一下，在读研的三年里，学到最重要的东西不是知识，也不是方法，而是意义。意义也可以理解成“方向”、“目标”、“理想”等等等等，虽然这些都是抽象的看上去遥不可及的，但它们就在每个人的心中，它们是每个人的精神支柱，一个人没有了支撑就失去了作为一个人的意义，成了一具行尸走肉。现在我才发现，原来我的精力实在有限，无法实现的理想太多，但我还是会努力去实现每一个对我来说很重要的事情，这样才有意义。每个人都有自己珍惜的意义，去努力珍惜吧。如果没有意义，就不要做这件事。意义是抽象的，但方法就显得具体许多了，但当今社会方法也是海量了，只一句 ----适合自己的方法才是最好的方法，自己能接受、对自己有益的方法才是正确的方法。连方法都是海量的，更何况知识？知识早就爆炸了，用有限的精力调控好意义、方法和知识的关系，这就是我在读研期间最大的收获。

言归正传，VRML 和 Pro/E 这两个工具都是十分优秀的三维造型工具，只是各自的用途不同，一个用于网络虚拟环境建立，另一个是用于专业建模。在我的课题里找不到它们的位置，所以我找到了第三种武器 ---Open CASCADE 。

最后，我说一下为什么选择 Open CASCADE 。第一、Open CASCADE 的建模能力可以达到非常专业的水平，这一点比 VRML 强很多；第二、Open CASCADE 可以轻松完成模型之间的布尔运算；第三、Open CASCADE 提供了很多类用来完成绘制三视图。其实这三点都是废话，说了等于没说。最重要的问题是：它为什么能有如此强大的功能呢，为什么有这么强大的功能但是大部分人却从未听说过它的大名呢？这一切的答案都是因为 --------Open CASCADE 是一款几何内核工具。简单说一下几何内核和三维造型软件的关系，Pro/E、Solid Edge 的几何内核是 Parasolid，这个内核在在市场上是找不到的，属于 Pro/E 公司内部的使用工具；金银花的几何内核的是 ACIS，这个内核是可以买到的。打比方说，Parasolid 是 Pro/E 的父亲，而 Open CASCADE 就应该算是 Parasolid 的兄弟了，Pro/E 碰到 Open CASCADE 是要磕头的。使用 Pro/E 的用户未必知道 Parasolid 是什么东西，他们也用不着知道，只要会用 Pro/E 完成 BOSS 交待的任务就 OK 了，因此作为底层开发工具的 Parasolid、ACIS 和 Open

CASCADE 不为人所知也就不足为怪了。

“所有的三维造型软件都是基于几何内核来开发的。”

既然有三种几何内核工具可选，那么我为什么选 Open CASCADE，是不是因为 Open CASCADE 是最强大的？当然不是，恰恰相反，因为 Open CASCADE 是免费的，在网上可以随便下载， Parasolid 买都买不着， ACIS 要花钱买，谁给我钱？唉，没办法呀，没米下粥只能喝水充饥了。在这里，我建议如果有能力的话还是购买 ACIS 来做你的程序，虽然我没用过 ACIS，但总觉得花钱买的应该更健康些。还有， Open CASCADE 也不是完全免费的，你如果使用它， 遇到不明白的问题无法自己解决的时候， 咨询是要花钱的， 你如果想做一个很完美的程序就一定会有问题要问的，我就碰到了一个棘手的问题，但是我没问， 嘿嘿，其实我是出来混的，后面会写我遇到的问题。

第 2 章 使用 Open CASCADE 的前提条件

2．1、英语

关键词：匈牙利命名法、 CDL

从小到大一直傻乎乎的学英语， 也没人跟我说为什么学、 以后能用上不，读研三年，工作两年，现在我可以负责任的对大家说， 英语很有用，真 TM 有用。且不说从初中到大学要经历无数场英语考试， 光说在学校期间和工作中的用途。 在学校做课题时你就会发现， 先进的技术都 TM 是国外的， 而国外的资料都 TM 是英语的， 当然也偶尔有中文的， 那仅限于比较热门的技术文档， 中等温度或冷门的资料你就看吧，一路鸟语花香的。工作中， 我是做电气的， 对单片机热过一阵，很多数字产品的使用说明都是英语，偶尔也有中文翻译，但你还要小心， 有些中文翻译你还不能全信， 因为有时候它们会翻译错， 我想是因为外行翻译的吧，唉，英语很重要。

Open CASCADE 资料是全英文的，有些是你必须看的，所以英语不好你是要头疼的。我的论文里的有很多内容是把这些资料翻译过来的， 挑出比较精髓的部分整理归类贴进了论文里， 其实我看了很多资料才选出这些， 有很多细节的部分是无法写进论文里的。 举个例子来说，论文里用到的 OCAF 框架，这有一篇资料专门介绍 OCAF 的用法，写的很详细，有理论有实例，这篇文章是必看的。当然还有很多资料需要看，这里就不详细说了。

如果你想熟练使用 Open CASCADE，这里就提两个要求：一是阅读能力，这个不多说，自己看着办； 二是对专业名词的理解， 这个详细说一下， 这里不仅仅是对你的英语能力有要求了，还需要对你的专业水平提出要求，你首先要明白中文的特有名词的清楚明了的认识，了解它的真正含义， 在这套体系中所处的地位， 然后才谈的上英语专业名词的理解。 其实这并不容易， 因为翻译本来就是英语中最有学问的部分， 在很多情况下， 翻译过来的就不是那个味了。 但是你并不用为此担心， 因为大部分的专业名词都是一一对应的。 特殊专业名词的缩写是需要你记住的， 比较好记， 都是单词首字母的大写组合， 比方说 OCAF、MFC、AIS、HLR 等，人家不会拿你开涮，不是重要的不会有简写形式，多看资料自然就记住了。

专业名词缩写： CDL （Component Description Language ）、IGES（模型标准）、XDE（模型标准）、STEP（模型标准）、BREP（模型标准）、HLR（Hidden Line Removal ）、OCAF(Open CASCADE Application Framework)、TCL/TK （Tool Command Language/Tool Kit ）、AIS（Application Interactive Services ）、DF（Data Framework）等

还有一点要说明的是， 既然要涉及到编程了， 就要熟悉匈牙利命名法， 在你熟练使用那些类时对你会有帮助的。在 Open CASCDE 中，利用 CDL （component definition language ）对类、方法、包、可执行文件等进行定义。 CDL 语言的使用方法在帮助文档中有详细介绍。

我当时看资料大概用了一个月？记不清了，反正是以月为单位的，我英语很一般。

2． 2、 VC++

大二的时候开始接触 VB，一开始有很多不明白的问题， 很多繁琐的名词让人不知所措，现在想来 VB 还算是挺简单的，完全模块化的编程工具，微软做好各种基本积木造型，你想造什么房子，搭起来行了，熟练掌握就成了熟练的程序员。 VC++ 也差不多一个道理，不过不一样的是，它的积木造型更细化，你可以设计自己基本积木造型，搭房子的过程更细化，这样你可以加进更多自己的东西，让房子更加合适自己的需要。

学 VC++ 的基础是 C 语言和 C++，其实 C++ 就是 C 语言的升级版，多了继承性、多态性和 **性（忘了）而已，这里就不多说了，想要熟练使用需要多练练。学习 VC++ 强烈建议学习孙鑫老师的教程，由浅入深，很喜欢，一共二十课，每课两个小时，听下来多练练就基本能对 VC++ 有个初步的了解。说实话，我现在的 VC++ 水平已经很低了，由于很长时间也没做程序，生疏了。

VC++ 很难，每次我在看 VC++ 程序看不下去的时候我就想问候一下 BILL GATES，当

然也很佩服他们居然能搞出这么复杂的东东，不过人家搞的有条不紊，确实是很不容易了。

学习它是一条很艰辛的道路，你需要接触并搞清楚无数复杂的晦涩的名词，需要熟悉很多“类”的意义、用途及使用方法，需要了解程序流程是怎么走的，需要明白怎么调试程序。

本人每次听孙老师讲课经常会感觉困意难当，是一个爬下 -起来回放 -爬下 -起来回放的循环过程，每课都要听两遍以上，每次听都感觉在听新的一样，学习 VC++ 大概用了一个月左右吧，毕竟以前有一些学语言的基础，还算比较快了。

VC++ 有个必备的工具，MSDN，是一套帮助系统。

第 3 章 关于 Open CASCADE

3．1 下载

上面说了足够多的废话，下面说正题，刚才说了 Open CASCADE 是免费的，你可以从它的官网上下载（www.opencascade.org），在华军等网站上也提供它的下载程序。下载的时候特别注意，你需要下载两个安装文件，一是 Open CASCADE 工具的安装文件（OpenCASCADE6.1.0.rar-409MB），二是 Open CASCADE 帮助文档（OCCTDocumentation6.1.0.rar--63MB），这个帮助文档也是需要安装的。我当时用的版本号是 Open CASCADE 6.1.0，现在最新的还是这个版本号，看来这是最终版本了。

3．2 安装及示例

安装 Open CASCADE6.1.0 时，你可以选择全安装，这样可以试试每个模块的功能和用法。当时我安装的时候需要花 10-20 分钟（也可能是当时机器落后的缘故），挺慢的，现在配置高的机器几分钟就能搞定。安装完成后，Open CASCADE 不会像其实安装程序一样跳出对话框问你是否运行该程序，因为 Open CASCADE 是一个庞大的类库系统，没有程序可以运行。安装 Open CASCADE6.1.0 完成后，再把帮助文档安上（其实就是把 OCCTDocumentation6.1.0.rar 解压一下）就可以进行下一个环节了。

安装完成后，你可以看看 Open CASCADE 为你提供的示例（如图 1），在我的课题中主要的内容就是在 OCAF 为基础上进行形体的拓扑运算（布尔运算），至于三视图嘛，本来是应该采用 HLR（Hidden Line Removal）消隐线技术来实现的，后来发现需要在二维界面

上进行建模， 需要重新建立一个二维的建模体系， 太麻烦了， 因此就在三维体系上做了一个视角的改动，变向的实现了三视图的生成，这种方法明显不如 HLR 生成三视图功能强大，但还凑合吧。 我的课题就是在 OCAF 的基础上做出来的， 因此首先建议看看 OCAF 的实例。在运行实例程序的时候，会有一个 New Document 对话框，这里面的程序是该实例的核心程序， 而且会随着你的操作而实时更新内容， 很好很强大，好好研究一下吧。然后你可以看一下 AIS （Application Interactive Services ）相关的程序， AIS 很重要，它提供了模型的视图控制功能，比如说选取模型、视角控制、隐藏模型、显示模型等等。其他的实例我没怎么用到，所以这里就不多说了。

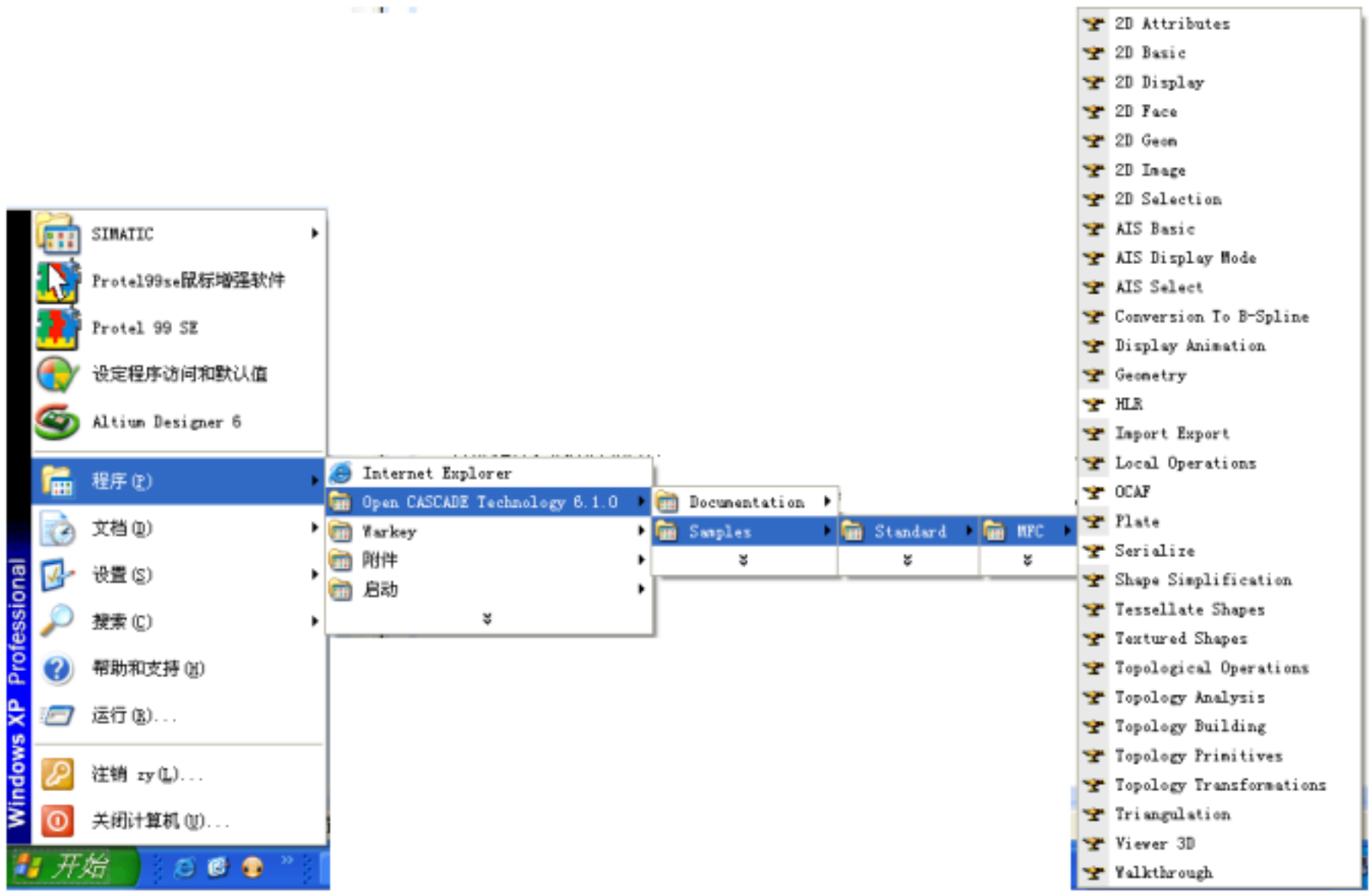

图 3-1

3．3 运行示例时的调试

这个课题已经做过两年多了， 现在我已经转行做电气， 无论做电气改造还是做程序， 调试都是最重要的一个环节。 不管你在运行实例程序， 还是调试自己的程序时， 由于每个人硬件不尽相同、操作系统不同、安装的驱动软件不同等等各种原因，会出现各种各样的错误，而分析错误原因、 解决问题的过程就是调试过程。 我不可能把每个出现的问题说清它的来龙去脉，只能把我在做课题时遇到的一些问题摆出来，讲一讲我是怎么分析、解决它们的。

在安装 Open CASCADE 时，已经把示例程序的 EXE 文件拷贝到“ 安装目录

\OpenCASCADE6.1.0\samples\standard\mfc\release ”下，可以直接运行它们就可以了。

下面是我遇到的两个错误和相关的解决方法：

第一个错误：示例程序无法运行，报错“ DfBrowser.dll 没找到”，解决方法：先从电脑中搜出 DfBrowser.dll ，把该文件放到 windows/system32 下。

第二个错误：在“安装目录 \OpenCASCADE6.1.0\samples\standard\mfc ”里，有 15 个实例的源程序。调试这些程序的时候，报错缺少 mfcsample.lib 头文件。解决方法：见第 5 章第 3 节。

第 4 章 关于我的论文

对于写东西来说，我是很头疼的，没办法，没小就没这个天分。如果要我自己给我的毕业论文打分的话，我只能打 60 分了，我并没有把我使用 Open CASCADE 的经验完全从论文中体现出来， 这也正是我写这篇文章的原因， 不管从哪个角度来说我都应该把这件事做的尽可能完整些。 最后我还会给出我的程序的源码， 供大家参考。 下面对我这篇篇论文再做一些说明。

前三章大家就没有必要看了， 那些是只是又臭又长又无用的陈述。 第 4 章在这看来是篇篇论文里写的最让我满意的一章了，前两节是从资料里摘抄下来的关于 OCAF （Open CASCADE Application Framework ）的功能介绍，是必看的；第三节是基于 OCAF 制作一个应用框架的过程。简单的说， MFC 是 VC++ 提供的一个应用程序框架，而 OCAF 又是 Open CASCADE 在 MFC 基础上的一个应用程序框架，关于 MFC 的工作流程，你可以看一下孙鑫老师的教程，有一课专门讲到了，很详细。那么 OCAF 是怎样的一个工作流程呢？他其实就是在 MFC 的特定类中加入一些代码，过程是：先通过设备类建立一个应用程序类，用来管理文档、视图等；再建立文档类，用来保存各种相关属性参数；然后建立视图类，用来控制怎样显示模型。 刚才又看了一遍第四章， 觉得没什么需要多说的， 如果你具备了一定的 VC++ 编程能力，你应该可以看的懂这一章，当然由于篇幅的原因，有一些可能介绍的简单一点，但是你可以翻看一下帮助文档，里面有一篇是讲 OCAF 的，很详细。结合那篇文章，按照论文中介绍的方法就可以做出一个自己的三维造型框架来。当然，如果你想熟练运用 Open CASCADE 来建模、布尔运算、视图的生成等等，那需要多花点心思把这个应用框架完全搞明白， 了解这个框架能实现什么功能、 通过那些类库来实现、 各种类和类之间的关系

及每个类的具体功能、操作方法等。

第 5 章写的很不太好，可能有很多人看不懂，这也难怪，其实当时我自己都是晕的。但是，这一章的内容却是最重要的，因为课题所要实现的功能都是在这一章完成的，我认为，如果你决定使用 Open CASCADE 来做你自己的程序，那么就要求你必须具备能够熟练运用 OCAF 来进行建模、各种视图操作等的能力。而第 5 章最主要的目的也正是介绍这一部分的内容，由于当时比较懒，也是水平所限，所以介绍的很简单，很惭愧。说一下第 5 章内容吧，前两节不用看，在第三节里，对建模过程得有个比较详细的介绍，可以看一下建模的流程，对你建模能力的提高有帮助，也可以在 OCAF 文档中找到相关的建模实例，多实践一下就 OK 了。布尔操作的介绍相对简单，当时懒了，所以没多写，其实也是没得写，因为那一部分我也是一团雾水的。三视图部分我居然漏写了，有些不应该，我会在后面的部分补上的（见第 6 章第 3 节 5）。其实，写这篇文章最主要的目的就是想把这一章内容充实一下，最近我又写了一个新的程序，我会结合这个程序对建模、视图操作及布尔运算做一个比论文内容更加详细的说明和总结。

第 6 章是泛泛写的，对程序进行一个大体的介绍，然后贴几张图片，秀一下成果，其实这一章没有参考价值，只是不得不写罢了。

对于整篇论文，1、2、3 章和第 6 章都没有参考价值，大家不用去浪费时间看。第 4 章和第 5 章第三节很有参考价值，但是限于论文写作要求、格式、篇幅等原因，没有把这一部分写的很充实，有很多话想写都没有形成文字。而且最最重要的是，论文里又不能粘贴整个程序，而最具有参考价值的是正是程序本身，因此这次我会把程序源码放到网上供有兴趣的参考。写了这么多很大一部分是为了把自己对 Open CASCADE 的理解变成文字写给我自己看的，给大家的只是程序源码而已。

最后，感谢我的父母、哥哥、老师、我爱的人和爱我的人、众多哥们、朋友和每一个认识或不认识的人。我真心的感谢他们！（论文的致谢都是在结尾处写的，习惯性的写个总结性致谢）。

第 5 章 Getting Started

好了，上文中，我对“ Open CASCADE 的功能”、“需要的准备工作”及“我的论文”做了一个简单的介绍。下面就开始说一下帮助文档的分类及使用方法、MS 调试方法和 Open

CASCADE 的结构。

5. 1、帮助文档

有两套帮助文档。

第一套帮助：你安装 Open CASCADE 完成后，单击“开始菜单”里“ introduction ”网页，进入的是 Open CASCAD 的第一个帮助系统， 他的作用是对 Open CASCAD 做一个详细的帮助说明，内容包括他的功能、包含的模块、各模块的作用、扩展工具等。

第二套帮助：你下载的 OCCTDocumentation6.1.0 帮助文档，解压后就可以打开 index.htm 看了，他的作用是对每一个类的属性、 方法、从属于哪个文件等等进行介绍， 举个例子来说，你要用到 BRepAlgoAPI_Cut 类（他是用来对两个模型进行减布尔操作的类）但你不会使用他，怎么从 OCCTDocumentation6.1.0 里找出他来？首先， 你应该知道他是属于哪个模块的，从表中，我们可以看出，他是属于建模运算类的，那么在帮助网页首页上，单击 ModelingAlgorithms 再单击 Class Hierarchy，然后 CTRL+F 搜一下就 OK 了。这两套帮助文档的使用是你做程序时必需的。

5. 2、第一套帮助文档内容及我的补充说明

5. 2. 1 第一套帮助文档

“开始菜单”打开 Open CASCADE 的帮助首页（如图 2），分为三部分，分别是 Open CASCADE 的总体简介、技术概括和一些扩展性工具。

总体简介简单看一下就可以了， 主要内容就是怎么安装、 需要的软硬件支持、 如何开始使用 Open CASCADE 、环境变量的注册等问题，不多说了；最为重要的内容是技术概括，应该对这一部分内容熟悉到无以复加，因为你对这一部分的了解程序直接决定着你运用 Open CASCADE 的熟练程度，下段专门说一下；在这七种扩展性工具中，我使用了 OCAF Browser 和 Wizards/OCAF Wizard 这两样工具，使用方法很简单，在论文里和程序里都能找到相关的应用，就不多说了。

图 5-1

5．2．2 技术概括

很想把这一部分翻译过来，因为这一部分即使是中文也会有很多人看不明白，它是整个 Open CASCADE 的核心部分，怎么熟练的应用这六大类是你做程序是否能顺利进行的关键。

翻译技术概括的首页内容如下：

//////////////////////////*　紫罗兰 为译文，　蓝色 为本人注释

Open CASCADE 对象库是一个面向对象的 C++类库，它是为尖端的某些领域所设计应用的快速制作而设计的。使用 Open CASCADE 开发的典型应用程序能够在通用或具体的 CAD 系统、制造或分析应用、模拟应用或图例工具中处理二维或三维几何建模问题。Open CASCADE 对象库可以帮助你快捷的开发你的应用程序。----Open CASCADE 其实就是一个类库，分成六大模块，每个模块里有 N 多功能类似的类包，每个包里有 N 多功能类似的类。开发的时候，利用编程工具（本文使用 VC++6.0 ）使用这些类来进行程序设计。Open CASCADE 提供了一个 RAD （Rapid Application Development 快速开发工具）框架环境，其实这个 RAD 就是 OCAF （Open CASCADE Application Framework ），用户可以在 OCAF 下快速的开发自己的应用程序。

对象库提供以下功能：

2D 和 3D 几何建模工具包让你可以对任何类型的对象进行建模

创建基本体，如棱柱、圆柱、圆锥和圆环面

布尔操作（加、减和交集）

倒角、倒圆角和草图操作

平移、抽壳、打孔和扫掠建模

计算属性，如表面面积、体积、重心和曲率

进行投影、插补和近似操作（插补和近似是优化设计里的内容）

可视化模块让你可以管理模型的显示和操作视图的显示

例子包括：

3D 旋转

缩放

控制阴影

应用程序框架特征

把非几何应用数据和几何体关联起来

模型的参数化

Java 应用程序平台（ JAD），一个用于创建用户图形接口的框架

CASCADE 模型和三维模型标准化格式之间的数据转换接口

---- 功能很强，越强的东西内容越多，越复杂。

Open CASCADE 对象库是基于 CAS.CADE 技术构建的，由 Open CASCADE 出品。他们是实现了模块化和可扩展性设计。 就其本身而言，这些类库分为三类：定义数据结构 （几何建模，显示和图形选择） 、复杂几何运算、提供应用程序接口。

相关的类被打包进一个包内以防类名冲突； C++ 类名都是以包名为前缀的。例如，所有定义 3D 几何体的类都在 Geompackage 包内。在这个 Geompackage 包内，用于生成 Bezier surfaces（贝赛尔曲面）的类就被命名为 Geom_BezierSurface。这些包又被归档入不同的类库内。最后，这些类库又被分成了六个模块（基类、建模数据类、建模运算类、可视化类、数据转换类和应用框架类） 。

这些类库模块及其内容如下表 5-1，图 5-2

基类	建模数据类	建模运算类	可视化类	数据转换类	应用框架

内核类 数学工具	二维几何体 三维几何体 几何体工具 拓扑	基本体构造 布尔操作 倒角圆角 偏移草图 缝补扫掠 特征 消除隐藏线 几何工具 拓扑工具	二维三维 通用体系 二维可视 化体系 三维可视 化体系	IGES STEP AP203 AP214 扩展数据 exchange (XDE)	数据框架 数据存储 应用界面

表 5-1

*//

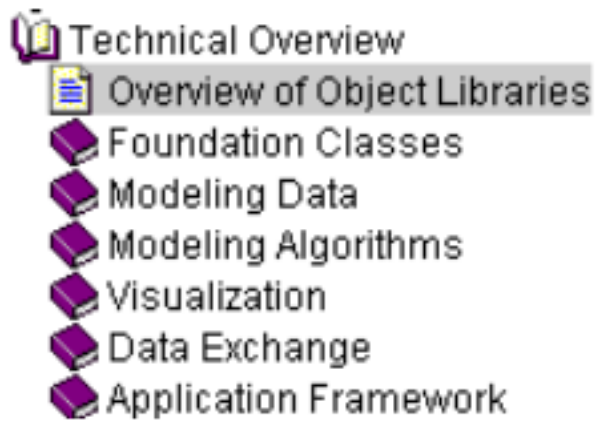

图 5-2

这六大模块我结合我做的程序说一下，一方面简单说一下这些模块的作用，另一方面有例子讲起来应该更生动些。这也有一个缺点，我不可能讲的很全面，能力所限，我理解多少就说多少吧。

第一、 基类

具体的文档内容我就不翻译了，基类，顾名思义，最基本的类，哪些是最基本的类呢？c语言里都如果你想申明一个整数变量，只要写“ int i ；”就行了，而在如果使用 Open CASCADE 做程序，你申明一个整数变量，则应该写成“ Standard_Integer i ”，Standard_Integer 类就是一个基类，明白了吧，其实这种类型的基类的作用就是为了申明 bool、整数、浮点数等基本数据类型。再举个例子，gp类包里的 gp_Pnt类，这个基类是给一个空间点设计了一个数据结构，用于保存该点，看看他的构造函数 gp_Pnt (const Standard_Real Xp, const Standard_Real Yp, const Standard_Real Zp)，他用笛卡尔坐标系（ Xp， Yp， Zp）创建一个点。而这些基类在程序中会经常使用到。

第二、 建模数据类

作用是为 2D、3D、几何工具和拓扑结构定义数据结构。我做的课题中，没有使用 2D

或 3D 的建模数据类，使用的是拓扑结构类，即使用了 TopoDS 类， TopoDS 类是用来构建拓扑的数据结构，我的三维模型都是用 TopoDS_Shape 类保存的。这里为什么不使用 3D 类呢，我的分析：使用 TopoDS_Shape 类主要是为方便 OCAF 框架的使用，在 Modeling Data 的 PDF 文档中， 第 5 章 Topology 的第一句话 “ Open CASCADE Topology allows us to access and manipulate data of objects without dealing with their 2D or 3D representations. ”，没有 3D 的建模数据类，也可以访问和操作模型数据，说白了，即使不用 3D 建模类建立模型也可以使用 TopoDS_Shape 类使用 3D 模型。 而且， 最为重要的一点， TopoDS_Shape 类这种拓扑类提供了图形拓扑功能。拓扑是什么意思，它在图形学里有什么作用？自己网上查。

第三、 建模运算类

本课题布尔操作功能的实现就是利用该模块的布尔运算类完成的， 例如布尔运算里的减运算就是使用 BRepAlgoAPI_Cut 类来完成的，具体编程在下文里讲。这里需要提一下的是三视图的生成方法， 也就是工程图的生成方法， 作为二维图纸的工程图拥有广泛的使用群体，在每个三维造型软件中都会有工程图模块来制作或生成三维模型相对应的工程图。 本课题的生成的三视图并不是二维视图， 只是三维形体的固定视角罢了， 因此并不能算是真正意义的工程图。 真正的工程图是通过二维模块生成的， 而实现工程图最重要的技术是消隐技术， 消隐技术的内容我不做过多介绍， 在 Open CASCADE 中，消隐技术的支持模块就是 Hidden Line Removal 相关类库。在 mfc 的示例中，也有相关的源码，有兴趣的拿相关的示例进行参考。

第四、 可视化类

前面三大模块的作用是解决了定义构建什么样的模型、 怎么构建模型、 模型的数据结构（即怎么存放模型） 这三大问题。 而可视化类解决了另一个大问题： 怎么建立模型的视觉环境（包括二维环境和三维环境两大部分） 。

可视化类解决了视觉方面的所有问题，主要分三个方面，模型的显示和选取、 graphical presentation（数据结构的图形描述？翻译不出来了）和 AIS （Application Interactive Services 应用程序的交互服务，它是联系着数据结构和交互模型之间的关系） 。零碎问题包括视觉角度、观察环境设置、 亮度、模型的材质、 模型的选取、 以哪种方式观察模型 （网格还是实体） 、透明度等等。

第五、 数据转换类

模型的数据接口作用是允许不同 CAD 系统互相利用对方的模型，从以前的 IGES 标准到现今的 STEP 标准，模型的数据接口已经发展了很多年。在课题中，也提供了一个接口来

实现数据转换功能，　就是打开不同标准的三维模型文件，　同时也能保存不同标准的三维模型文件。

第六、应用框架

应用框架为用户提供一个快速开发环境，　让用户可以快速开发自己的应用程序。　论文的第 4 章主要是讲应用框架的，可以参考一下，这里就不多介绍了。

对于应用框架，一定要熟悉　OCAF 的作用、原理和使用方法。　OCAF 的作用和原理是使用方法的理论基础，使用方法是实践手段，理论　+实践才能得到好的效果。论文里简单描述了 OCAF 的作用、原理和建立基于　OCAF 的 MFC 单文档应用程序框架的过程。　关于 OCAF 的作用和原理部分如果有不懂的地方最好看一下帮助文档的　ocaf.pdf 文档，而建立基于 OCAF 的 MFC 单文档应用程序框架的过程，在这里我总结性的说一下流程，具体细节见程序和下文。

建立基于　OCAF 的 MFC 单文档应用程序框架的过程是本论文中最主要的工作之一。　首先，谈一下　MFC 应用程序框架，　孙鑫老师在　VC++ 的教程里讲过　MFC 的运行机制，　大体就是先生成应用程序类（管理整个程序的运行）　，再生成文档类（保存相关参数）　、框架类（总体视图）、视图类，然后把后它们的关系设置一下，形成一个多文档或单文档的应用程序。

VC++ 的用户要做的工作就是修改这个应用程序来实现其想要实现的功能。　这里要强调一下，MFC　绝不是一句话两句话能说明道清的，它是　VC++ 的核心技术之一，如果想要顺利的完成基于　MFC 的开发任务，必须对　MFC 的运行机制、使用方法有一个清楚的了解，这个需要自己多看书多练了。　OCAF 应用程序框架就是基于　MFC 来开发的，　它的运行机制和　MFC 的类似，首先生成一个应用程序类，　再创建一个交互环境和三维视图器，生成文档类、　视图类。这一过程虽然和　MFC 的生成过程类似，但是复杂程度却不可同日而语，因为　OCAF 是在 MFC 基础上建立起来的，　OCAF 只需要在　MFC 应用程序中的特定位置加入特定的代码就可以完成　OCAF 应用程序框架的建立。

在论文里已经写过它的建立过程，在这里只是补充说明几个问题。　1、利用　OCAF 向导生成的　OCAF 框架是　MFC 多文档应用程序框架，　我们可以通过研究该向导生成的程序来得出 OCAF 框架的基本组成和运行原理，　而后再按照此原理在对　MFC 单文档应用程序添加代码就可以得到基于　OCAF 的 MFC 单文档应用程序框架；　2、论文中的　OCAF 应用程序框架是通用的，如果你不使用　VC++ 进行编程而使用其它编程工具，可以按此方法进行框架的建立（未验证）；3、基于　OCAF 应用程序框架是此类程序开发的基础，框架建立起来后，开发人员只需要关心，建模、建模运算和可视化问题，不需要把精力放在底层接口的开发了。

5．3、MS 调试方法

我的程序会放在网上供大家参考，程序名是 MS（当初也不知道怎么会起这么个名字，随手起的），这个程序是用 VC++6.0 和 Open CASCADE 开发的，因此如果想看这个程序需要安装这两个软件。 安装完成这两个软件后， 调试 MS 程序会出现一些错误， 下面我给出相关错误和调试方法。

我刚从一台新机器上安装了 VC++ 和 Open CASCADE ，然后调试 MS 程序，碰到了下述问题，下面我会把出现错误的相关原因和解决方法解释一下：

1、 第一个错误：

```
//////////////////////*  错误摘要
----------Configuration: MS - Win32 Debug----------
Compiling...
MSDoc.cpp
G:\MS6\MSDoc.cpp(17) : fatal error C1083: Cannot open include file: 'DebugBrowser.hxx': No
such file or directory
Error executing cl.exe.
Creating browse info file...
BSCMAKE: error BK1506 : cannot open file '.\Debug\MSDoc.sbr': No such file or directory
Error executing bscmake.exe.

MS.exe - 2 error(s), 0 warning(s)
*////////////
```

这个好解决， hxx 文件是 Open CASCADE 类的头文件，如果运行使用 Open CASCADE 工具制作的程序，就必须用到相关的 hxx， jxx 文件等，解决方法：在 VC++ 中，工具 /选择 /

2、第二个错误

```
//////////////////////*  错误摘要
----------Configuration: MS - Win32 Debug----------
Compiling...
MSDoc.cpp
Linking...
LINK : fatal error LNK1104: cannot open file "TKOffset.lib"
Error executing link.exe.
```

Creating browse info file...

MS.exe - 1 error(s), 0 warning(s)
*////////////

Open CASCADE 的类以 DLL 文件形式提供给用户使用，每个 DLL 文件都有一个与之对应的 lib 头文件，具体的使用方法其实是 VC++ 的内容，仔细看 VC++ 教程里会有一课讲到这部分内容。这里只简单说一下该错误出现的原因和解决方法，原因： VC++ 找不到 TKOffset.lib 文件；解决方法： 1、首先找到 TKOffset.lib 的存放位置 （如图 5.3）；2、在 VC++ 的工具 /选项 /目录里，添加相关目录（如图 5.4）

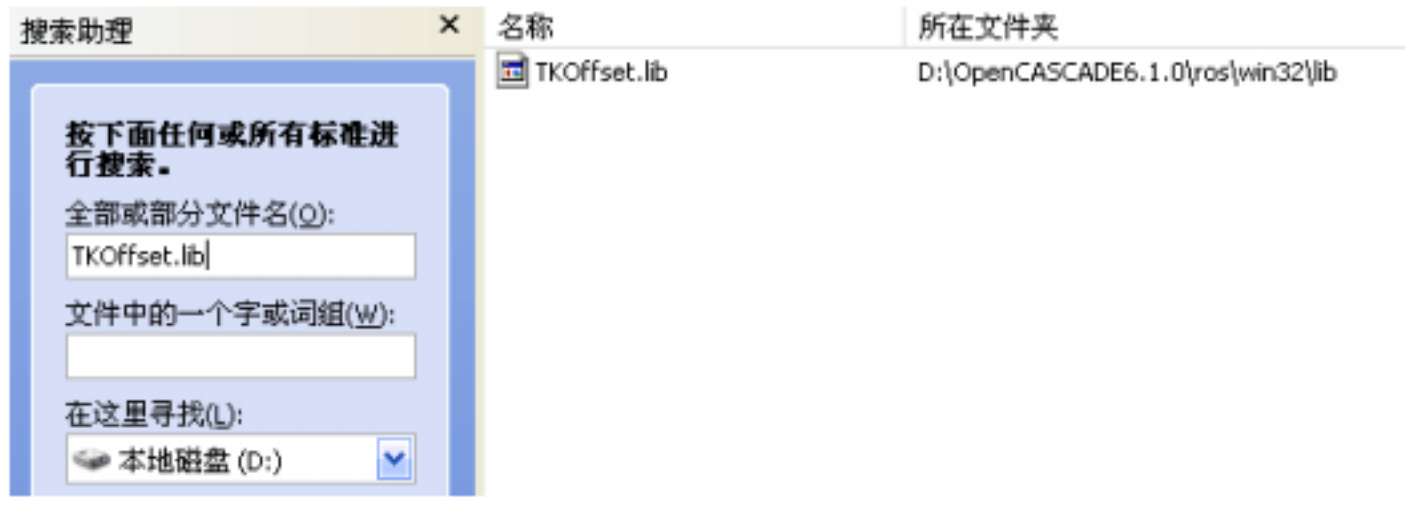

图 5.3

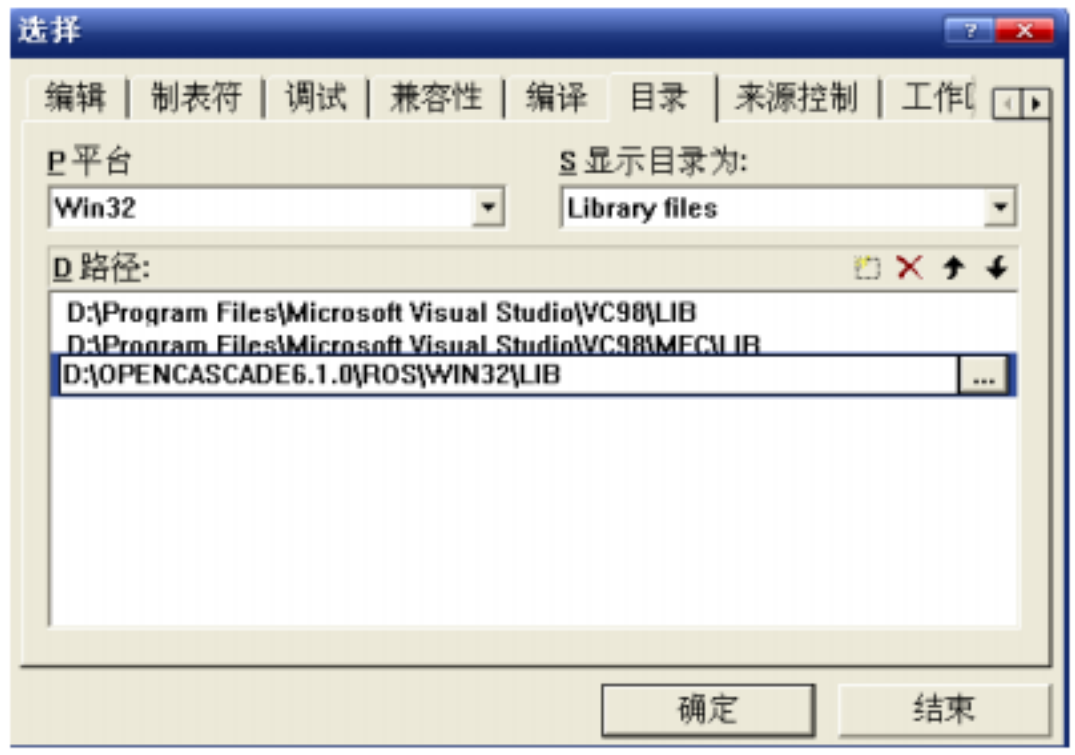

图 5.4

3、 第三个错误

//////////////////////////* 错误摘要
----------Configuration: MS - Win32 Debug----------

Linking...
LINK : fatal error LNK1104: cannot open file "DFBrowser.lib"
Error executing link.exe.

MS.exe - 1 error(s), 0 warning(s)
*////////////

这个错误和　2 错误的原因和解决方法是一样的，不多说。那为什么它会单独出现呢？

DFBrowser 是 Open CASCADE 提供的一个扩展工具，我在课题用到了，功能是观察　　OCAF 框架中的总体参数结构，用法很简单，后面再详说。

4、 第四个错误

////////////////////////*　错误摘要
----------Configuration: MS - Win32 Debug----------
Linking...
LINK : fatal error LNK1104: cannot open file "mfcsample.lib"
Error executing link.exe.

MS.exe - 1 error(s), 0 warning(s)
*////////////

这个错误和　2 错误的原因和解决方法是一样的，不多说。

mfcsample 是个什么东西呢？从文件名就可以看出，　　它是为　MFC 示例提供的一个类库。我一开始做程序的时候，　是先从示例程序开始，　即对示例程序做一点小改动，　看能实现不？后来了解多了，　就开始独立做自己的程序了，　但是有几部分还是带着示例的影子，　所以在程序中我的程序需要　mfcsample。

还有一问题需要说明的是，我曾经碰到过一个问题，在整个电脑中都找不到 mfcsample.lib 文件。如果你也找不到，那就不用找了，找不到的。我告诉你怎么处理这个问题，在“安装目录\OpenCASCADE6.1.0\samples\standard\mfc\mfcsample　”下打开 mfcsample.dsw 文件，然后调试程序，出现如下错误，找到对应的源码，屏敝掉（如果你能把这个错误解决也行，我是没办法通过正当途径解决，只能粗暴对待了）　　，再调试，这样就能调试成功，在　debug 文件夹下可以找到　mfcsample.lib 文件了，下面的事情就容易了，自己看着解决吧。

//////////////////////*　错误摘要

d:\opencascade6.1.0\samples\standard\mfc\common\isession2d\isession2d_objectowner.h(52) : error C2679: binary '<<' : no operator defined which takes a right-hand operand of type 'class TCollection_AsciiString' (or there is no acceptable conversion)

ISession2D_ObjectOwner.cpp

d:\opencascade6.1.0\samples\standard\mfc\common\isession2d\isession2d_objectowner.h(52) : error C2679: binary '<<' : no operator defined which takes a right-hand operand of type 'class TCollection_AsciiString' (or there is no acceptable conversion)

ISession2D_Shape.cpp

*//////////////////////

5、第五个错误，如图　5.5

图 5.5

这种类型的错误相当经典，　相信大家碰到过吧？有时候，　我安装完游戏软件后，　运行的时候偶尔会碰到类似的错误。其实错误原因就是系统要使用　dll 文件中的类，结果系统找不到 dll 了。这种问题的解决方法：　1、找到相关的　dll 文件（安游戏时找不到可以从网上搜，网上有专门提供　dll 文件的网站）　2、放到 C:\WINDOWS\system32　文件夹下即可，在这里我不建议把　DFBrowser.dll　放到　C:\WINDOWS\system32　文件夹，最好把它放到“安装目录\OpenCASCADE6.1.0\ros\win32\bin　”下，在注册环境变量的时候，　系统已经把上述文件夹设为 dll 文件的存放路径了。

6、 第六个错误

这段错误是通过　OCAF　向导生成多文档程序后调试过程中出现的，如果出现类似错误解决方法都是一样的。

//////////////*　错误摘要

--------------------Configuration: 1 - Win32 Debug--------------------

Compiling resources...

Compiling...

StdAfx.cpp

Compiling...

OcafApplication.cpp

ChildFrm.cpp

MainFrm.cpp

1.cpp

1Doc.cpp

1View.cpp

Generating Code...

Linking...

Creating library Debug/1.lib and object Debug/1.exp

LINK : warning LNK4098: defaultlib "MSVCRTD" conflicts with use of other libs; use /NODEFAULTLIB:library

OcafApplication.obj : error LNK2001: unresolved external symbol "public: virtual void __thiscall TDocStd_Application::InitDocument(class Handle_TDocStd_Document const &)const " (?InitDocument@TDocStd_Application@@UBEXABVHandle_TDocStd_Document@@@Z)

OcafApplication.obj : error LNK2001: unresolved external symbol "public: virtual void __thiscall TDocStd_Application::NewDocument(class TCollection_ExtendedString const &,class Handle_TDocStd_Document &)" (?NewDocument@TDocStd_Application@@UAEXABVTC ollection_ExtendedString@@AA VHandle_TDocStd_Document@@@Z)

OcafApplication.obj : error LNK2001: unresolved external symbol "public: virtual class Handle_Resource_Manager __thiscall TDocStd_Application::Resources(void)" (?Resources@TDocStd_Application@@UAE?A VHandle_Resource_Manager@@XZ)

OcafApplication.obj : error LNK2001: unresolved external symbol "public: virtual __thiscall TDocStd_Application::~TDocStd_Application(void)" (??1TDocStd_Application@@UAE@XZ)

OcafApplication.obj : error LNK2001: unresolved external symbol "class Handle_Standard_Type & __cdecl TDocStd_Application_Type_(void)" (?TDocStd_Application_Type_@@Y AAA VHandle_Standard_Type@@XZ)

OcafApplication.obj : error LNK2001: unresolved external symbol "public: virtual unsigned int __thiscall TDocStd_Application::IsKind(class Handle_Standard_Type const &)const " (?IsKind@TDocStd_Application@@UBEIABVHandle_Standard_Type@@@Z)

OcafApplication.obj : error LNK2001: unresolved external symbol "public: __thiscall Handle_TDocStd_Application::~Handle_TDocStd_Application(void)" (??1Handle_TDocStd_Application@@QAE@XZ)

OcafApplication.obj : error LNK2001: unresolved external symbol "protected: __thiscall TDocStd_Application::TDocStd_Application(void)" (??0TDocStd_Application@@IAE@XZ)

1.obj : error LNK2001: unresolved external symbol "public: __thiscall TDF_Transaction::TDF_Transaction(class TCollection_AsciiString const &)" (??0TDF_Transaction@@QAE@ABVTCollection_AsciiString@@@Z)

1Doc.obj : error LNK2001: unresolved external symbol "public: __thiscall TDF_Transaction::TDF_Transaction(class TCollection_AsciiString const &)" (??0TDF_Transaction@@QAE@ABVTCollection_AsciiString@@@Z)

1View.obj : error LNK2001: unresolved external symbol "public: __thiscall

TDF_Transaction::TDF_Transaction(class TCollection_AsciiString const &)" (??0TDF_Transaction@@QAE@ABVTCollection_AsciiString@@@Z)
1Doc.obj : error LNK2001: unresolved external symbol "public: __thiscall Handle_TDocStd_Document::~Handle_TDocStd_Document(void)" (??1Handle_TDocStd_Document@@QAE@XZ)
1View.obj : error LNK2001: unresolved external symbol "public: __thiscall Handle_TDocStd_Document::~Handle_TDocStd_Document(void)" (??1Handle_TDocStd_Document@@QAE@XZ)
1Doc.obj : error LNK2001: unresolved external symbol "public: void __thiscall TDocStd_Application::Close(class Handle_TDocStd_Document const &)" (?Close@TDocStd_Application@@QAEXABVHandle_TDocStd_Document@@@Z)
1Doc.obj : error LNK2001: unresolved external symbol "public: enum CDF_StoreStatus __thiscall TDocStd_Application::SaveAs(class Handle_TDocStd_Document const &,class TCollection_ExtendedString const &)" (?SaveAs@TDocStd_Application@@QAE?AW4CDF_Store Status@@ABVHandle_TDocStd_Document@@ABVTCollection_ExtendedString@@@Z)
1Doc.obj : error LNK2001: unresolved external symbol "public: enum CDF_RetrievableStatus __thiscall TDocStd_Application::Open(class TCollection_ExtendedString const &,class Handle_TDocStd_Document &)" (?Open@TDocStd_Application@@QAE?AW4CDF_Retrievab leStatus@@ABVTCollection_ExtendedString@@AA VHandle_TDocStd_Document@@@Z)
1View.obj : error LNK2001: unresolved external symbol "public: unsigned int __thiscall TDF_Label::FindAttribute(class Standard_GUID const &,class Handle_TDF_Attribute &)const " (?FindAttribute@TDF_Label@@QBEIABVStandard_GUID@@AA VHandle_TDF_Attribute @@@Z)
1View.obj : error LNK2001: unresolved external symbol "public: __thiscall Handle_TDF_Data::~Handle_TDF_Data(void)" (??1Handle_TDF_Data@@QAE@XZ)
1View.obj : error LNK2001: unresolved external symbol "public: class Handle_TDF_Data __thiscall TDocStd_Document::GetData(void)const " (?GetData@TDocStd_Document@@QBE?A VHandle_TDF_Data@@XZ)
1View.obj : error LNK2001: unresolved external symbol "public: __thiscall TDF_Label::TDF_Label(void)" (??0TDF_Label@@QAE@XZ)
1View.obj : error LNK2001: unresolved external symbol "private: __thiscall TDF_Label::TDF_Label(class TDF_LabelNode * const &)" (??0TDF_Label@@AAE@ABQA VTDF_LabelNode@@@Z)
Debug/1.exe : fatal error LNK1120: 18 unresolved externals
Error executing link.exe.

1.exe - 22 error(s), 1 warning(s)

*//////////////////////

第一句是警告： LINK ：warning LNK4098: defaultlib "MSVCRTD" conflicts with use of other libs; use /NODEFAULTLIB:library ，原因：和其他类库冲突。解决方法：如图 5-6

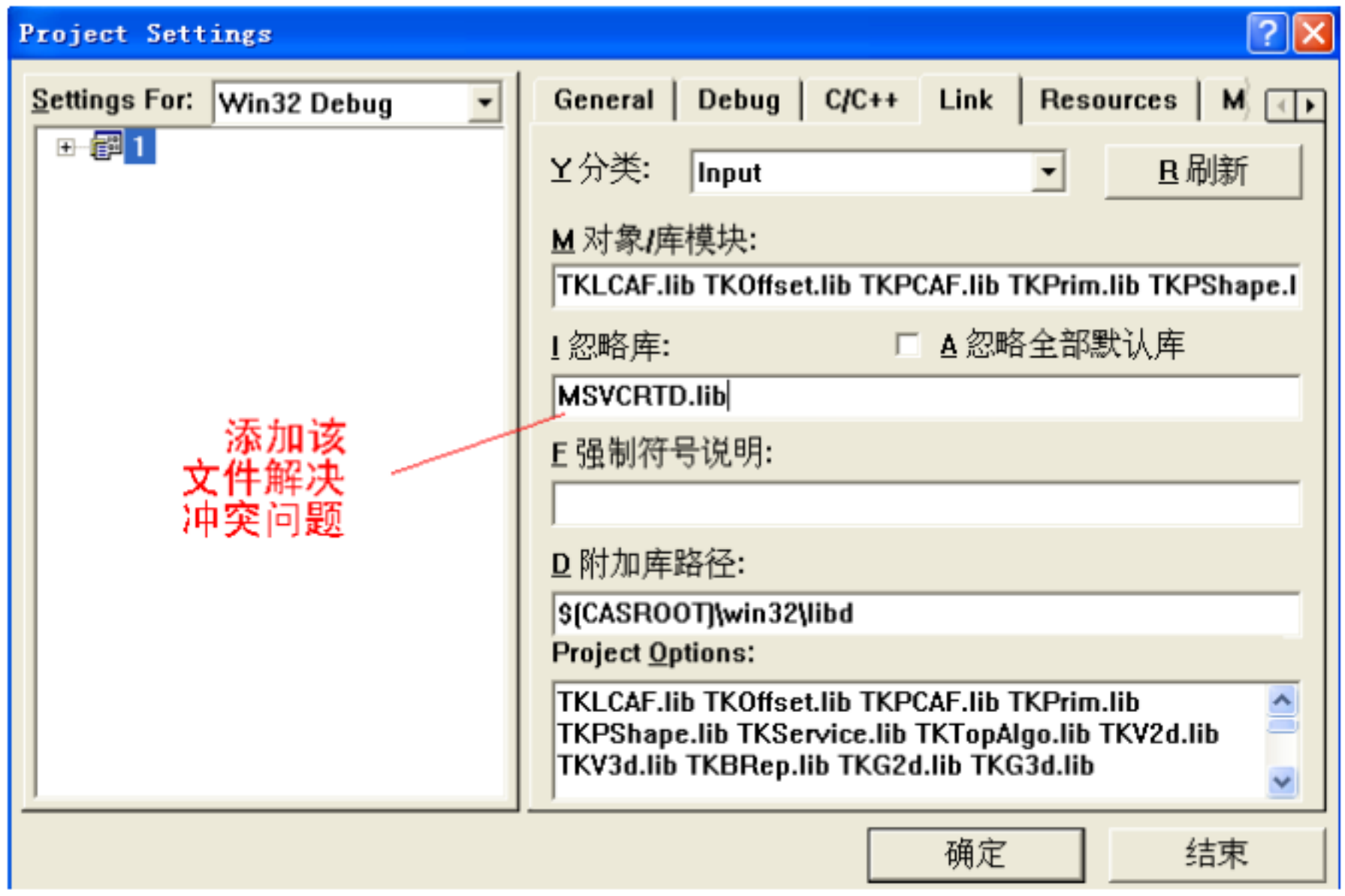

图 5-6

第二句： OcafApplication.obj : error LNK2001: unresolved external symbol "public: virtual void __thiscall TDocStd_Application::InitDocument(class Handle_TDocStd_Document const &)const "
(?InitDocument@TDocStd_Application@@UBEXABVHandle_TDocStd_Document@@@Z)

这种错误出现的原因： 使用类找不到相关的类库， 简单说就是， 你在程序里使用了一个类库里的类， 但是没有正确给出类库的路径， 所以出现了该错误。 这种错误和前几个错误类似，都是找不到类库的路径，但是解决方法是不一样的。解决方法：使用一个类库中的类，必须在 VC++ 中申明相关类库的头文件（ lib 文件）才能使用它，方法如图 5-7。

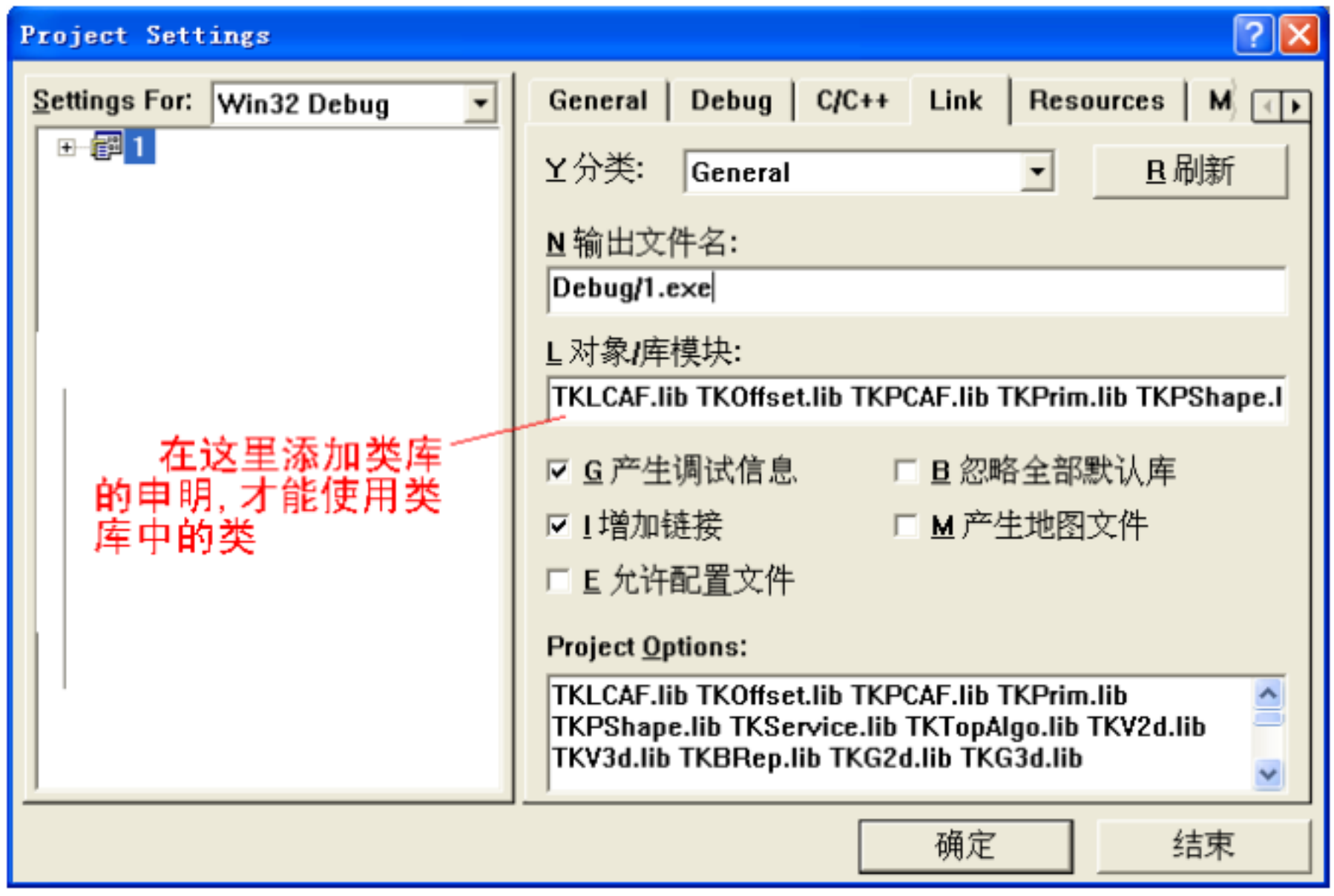

图 5-7

在第二句错误中， 找不到 TDocStd_Application 类，那么怎么知道 Open CASCADE 的哪个类库里有这个类呢？在这里我们就需要使用第二套帮助文档。 第一步，应该确定 TDocStd_Application 类是属于哪个模块的。猜一下就知道了，是属于 ApplicationFramework 模块的，如果猜不到，就在每个模块的 Class Hierarchy 中搜一下看有没有该项类。在 ApplicationFramework 模块的 Class Hierarchy 中能找到该类。第二步，找到 TDocStd_Application 类是属于哪个类库。 ApplicationFramework 模块包含了 15 个类库，TDocStd_Application 类放在哪个类库呢？在 Class Hierarchy 中能看出，TDocStd_Application 类的最高级父类是 CDM_Application ，在 Packages 里能找到 CDM 类包，打开 CDM 类包在标题位置写明了 CDM 类包是属于 TKCDF 类库的。因此，在上图位置添加 TKLCAF.lib 就可以解决该类错误。 上述橙色内容过于罗嗦， 说了半天你也不一定能看明白到底怎么找哪个类属于哪个类库，我来介绍一下简单的方法。把 Open CASCADE 的类库名都在上图位置加上，就不会出现这种类型的错误了。我添加的类库头文件如下： TKOffset.lib TKPCAF.lib TKPrim.lib TKPShape.lib TKService.lib TKTopAlgo.lib TKV2d.lib TKV3d.lib TKBRep.lib TKG2d.lib TKG3d.lib TKGeomBase.lib FWOSPlugin.lib PTKernel.lib TKBool.lib TKCAF.lib TKCDF.lib TKDraw.lib TKernel.lib TKFillet.lib TKGeomAlgo.lib TKHLR.lib TKMath.lib TKLCAF.lib TKBO.lib DFBrowser.lib TKXSBase.lib TKShapeSchema.lib TKIGES.lib TKShHealing.lib TKFeat.lib mfcsample.lib 。

总结一下：在调试示例程序和 MS 相关程序的时候会遇到的错误大部分不是程序本身的代码错误，而是 VC++ 的设置问题，常见就是在使用类时没有设置好相关的路径或是做类的申明，因此，需要大家在调试别人的程序遇到错误时，不要先看程序源码，先从 VC++ 设置里找问题。

好了，在调试 MS 程序时就出现了上述问题，这其实也是使用 VC++ 做程序时经常会碰到的基础问题，大家应该学会怎么调试自己的程序，多练多问吧。当然，这只是个基础，在做程序的时候还会出现这样或那样的问题， 我的 VC++ 水平现在已经不行了， 其实就没行过，如果遇到 VC++ 的相关问题，大家可以试着加入相关的 QQ 群，那里有很多热心人。

5．4、Open CASCADE 的结构

安装 Open CASCADE 完成后，会注册一些环境变量，以方便编程工具使用。下面介绍一下 Open CASCADE 每个文件夹的用处（我记得我在帮助文档里见过相关的介绍，结果找不着了，先介绍下吧，以后找到再说） ，如图 5-8

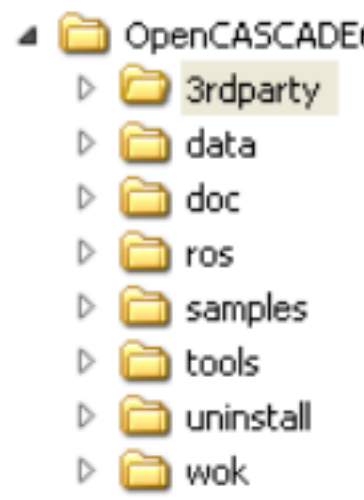

图 5-8

1、 3rdparty

第三方支持文件，我只用了 vs 的 mfc42.dll 文件，这是 VC++6.0 的 dll 文件

2、 data

三维模型的格式有很多标准，现在比较认可的是 step 标准，有兴趣的可以研究一下。这个文件夹里保存了各种标准格式的模型数据，在做文件标准格式转换相关程序中可以用到。

3、 doc

这个文件夹里装的是帮助文档， 网页的就是你所见到的第一套帮助文档， 重要的是 PDF 格式的帮助文件， ocaf.pdf、visu.pdf 等都是必看的。

4、 ros

这个文件夹装的是 Open CASCADE 系统文件，开发软件需要使用到它们。 Open CASCADE 定义的上万个类就包含在其中， 使用时需要： 1、添加相关的 DLL 、LIB 文件； 2、申明相关的 *.hxx 文件。在设置 VC 的时需要添加这个文件夹下的相关路径，添加方法上一小节已经提到了。

5、 samples

示例文件夹，我主要看的是 mfc 示例，每个示例都有相关的源码，我正是从这些源码开始学习使用 Open CASCADE 的，有很好的启蒙作用。

6、 tools

扩展性工具， 这几个工具我都用过了， 很简单。 OcafAppWizard 就是生成 OCAF 框架的工具，在论文里介绍使用方法。 Viewer3dAppWizard 工具不太好用，我试过了一下就没在课题中使用它， 你可以试试。 DFBrowser 这个类库我在程序里使用了， 是用资源管理器形式来表示 OCAF 各个属性参数之间的关系，只能看不能改，具体的 OCAF 内部的属性参数之间的关系还要编程来实现，这里不多说。

7、 uninstall

这个嘛。。。不用说了

8、 wok

这个没看懂，没用过。

第 6 章 程序说明

前几章对 Open CASCADE 做了简单的介绍，下面对我做的程序做一个说明。在网

6．1、框架前 OCAF 向导生成的应用程序和单文档应用程序框架

程序名：框架前程序 .rar

先使用 OCAF 向导生成应用程序，过程在论文里有介绍，这里再介绍一下。把 Open CASCADE 根目录下 Tools 文件夹中的 OcafAppWizard.awx 文件添加到 VC++\Common\MSDev98\Template 目录里， 然后再 VC++ 新建工程里 （如图 6-1）生成 Open

CASCADE Technology Ocaf AppWizard 工程，通过该向导生成的工程是一个多文档程序。调试该程序，我在初次调试的时候碰到了错误提示，请查阅第 5 章 MS 调试方法一节中的第六个错误。

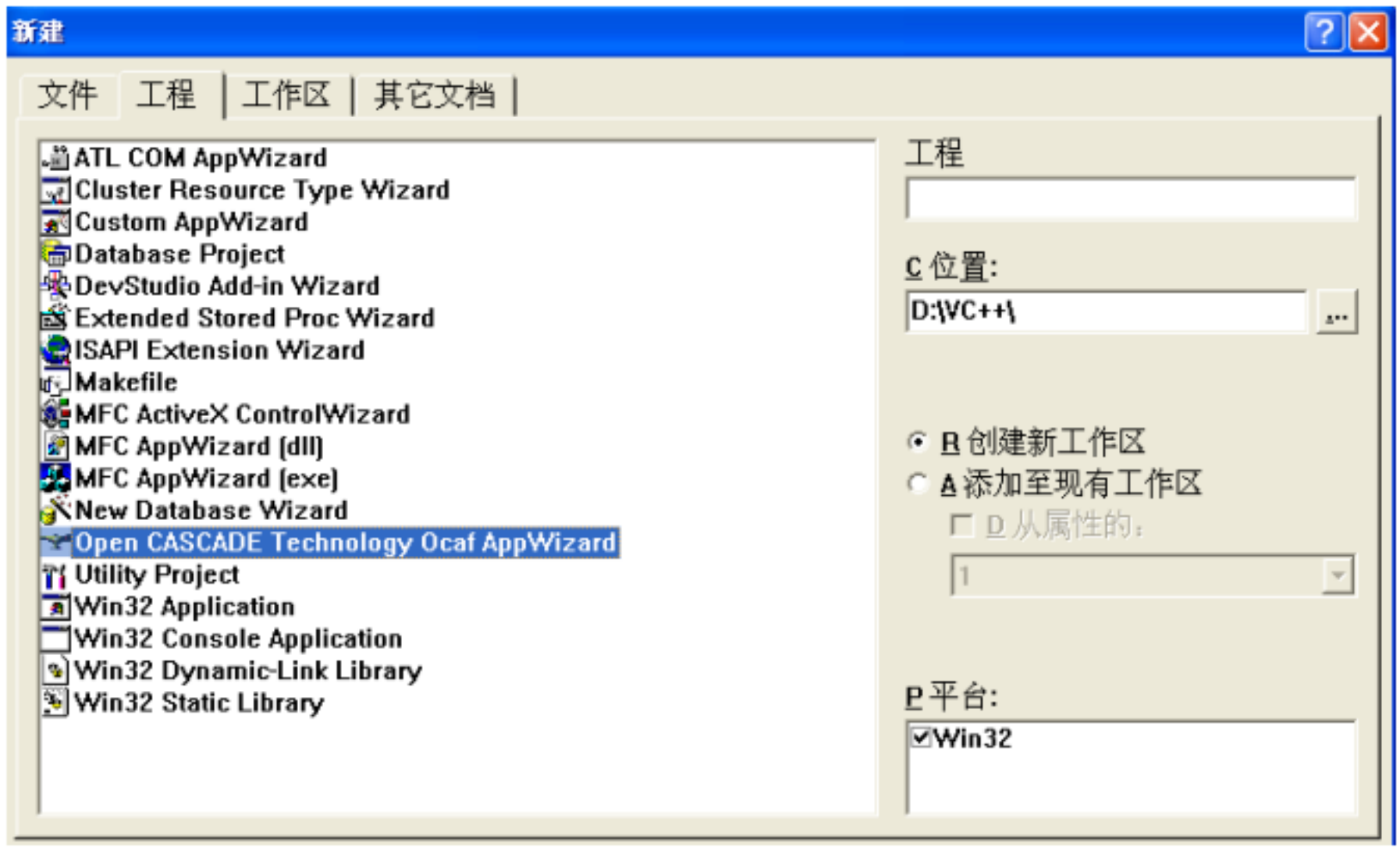

图 6-1

调试完成后，程序界面如图 6-2：

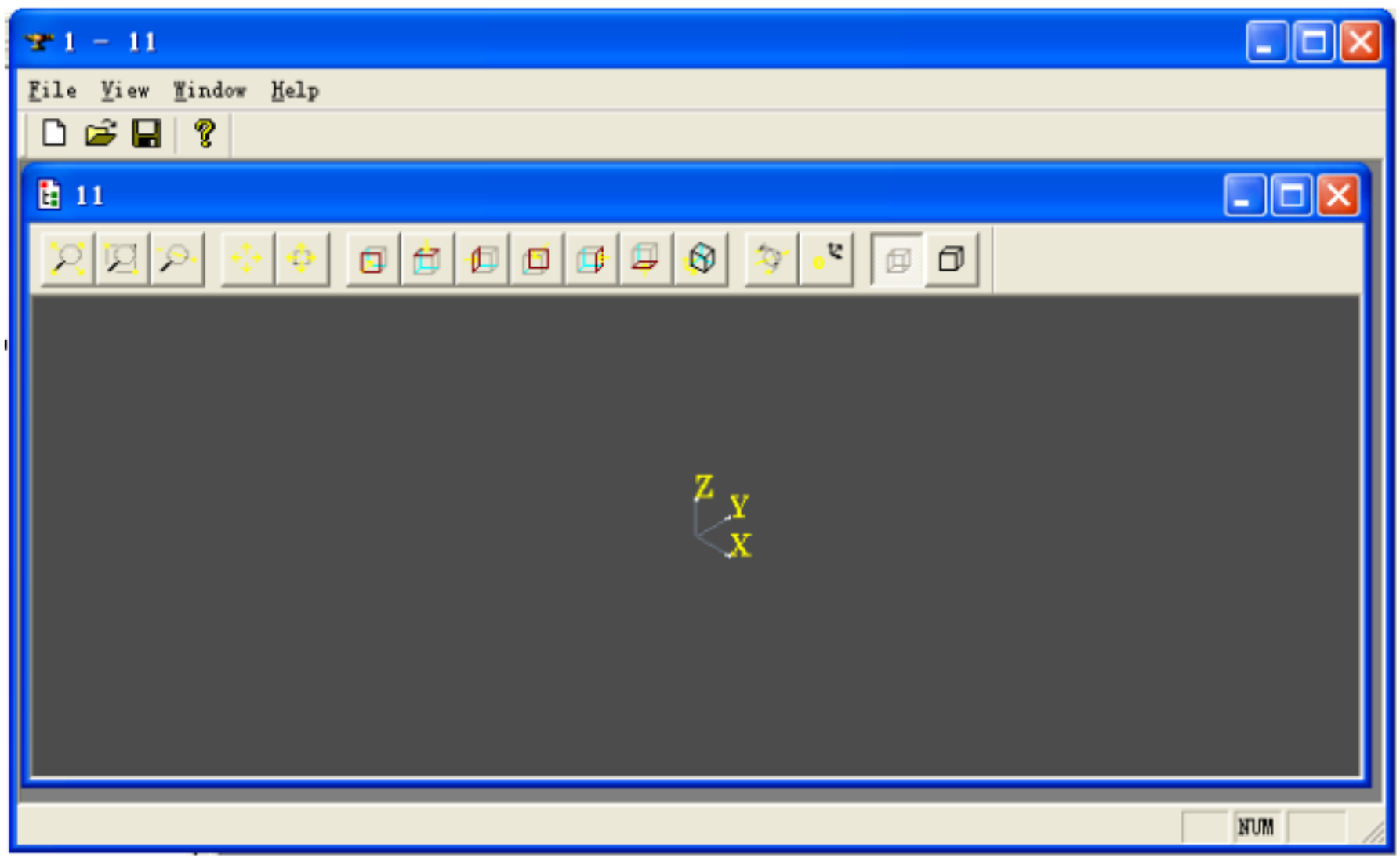

图 6-2

在论文里，我说明了 OCAF 框架的生成流程，每一步应该在哪个位置加哪些代码都做

了介绍，大家可以对对照着论文，通过研究这个程序的源码来看一下这个流程，如图 6-3。

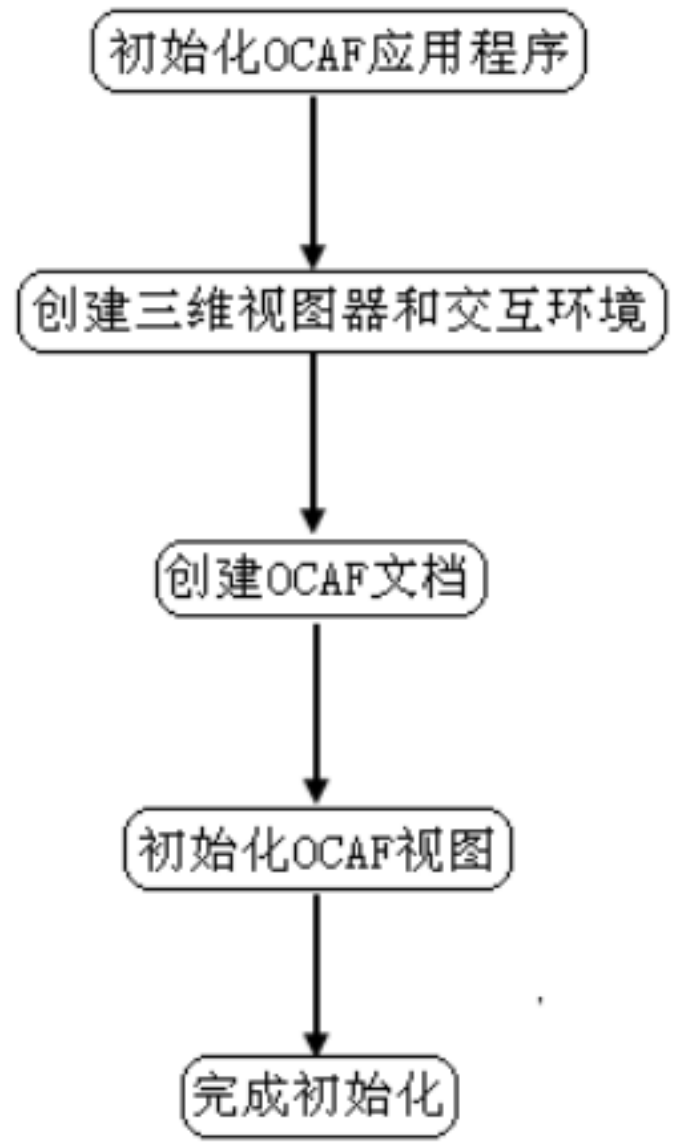

图 6-3

在哪里加载哪些代码，再说一遍。

6．2、OCAF 基于 MFC 单文档应用程序框架

程序名： MS.rar。

基于 MFC 单文档应用程序框架，使用工程向导生成一个 MFC 单文档应用程序，取名叫 MS。在了解了 OCAF 框架的生成流程后， 就可以自己制作一个 OCAF 应用程序框架。 当然，用户也可以用不同的编程工具来制作不同的应用程序框架，流程都是一样的。

6．3、MS 程序 1

程序名： MS 程序 1.rar

这个程序就是用来毕业答辩的程序之一， 申明一下， 这个程序中有很多不尽如人意的地方，没有时间也没有心情去修改，对于其中的毛病就不评论也不改了， 只对建模、 布尔操作功能是怎样实现的做一个介绍。如图 6-4

要介绍的几个部分： 1、几个类的功能； 2、框架类的制作； 3、对话框； 4、CControlWnd

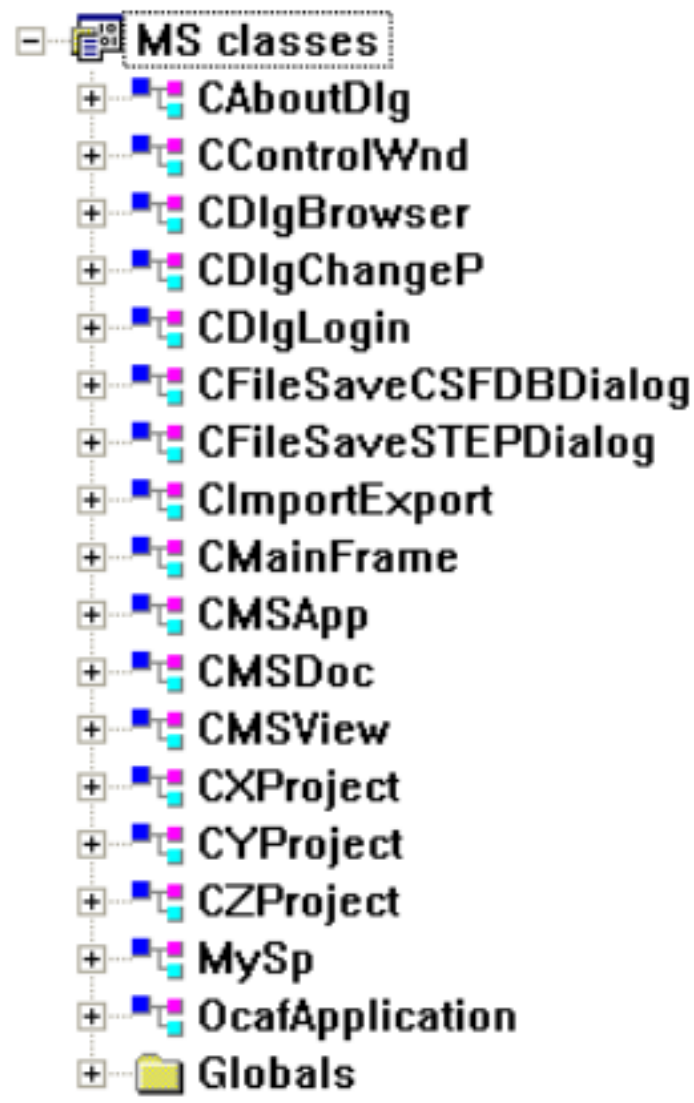

图 6-4

6. 3. 1、类的简介

其实写 VC++ 程序就是在编写各种类的属性、方法等，把他们组织协调好以实现相应的功能。 在编程的时候，代码的编写位置并不是固定不变的，比如说：视角控制的代码并不一定非要写在视图类（ CMSView ）下，也可以写在文档类（ CMSDoc ）下，只要在文档类中申明一下就可以使用 V3d_View 类的相关方法来控制视角，在示例程序中这种情况出现过。

前几个小节已经描述了基于 MFC 应用程序框架的构建过程，因此在这里构建框架的相关代码就不做过多描述，重点说明一下 CControlWnd 类的构建过程，其次说明一下对话框类、分区框架类和数据转换类的使用过程。

对 MS 程序 1 类的简介如下：

1、 CControlWnd 类里来实现建模、布尔操作功能。

2 、 CDlgBrsowser 、 CDlgChangeP 、 CDlgLogin 、 CFileSaveCSFDBDialog 、 CFileSaveSTEPDialog ： 对 话 框 类 ， 前 三 个 不 说 了 ， CFileSaveCSFDBDialog 、 CFileSaveSTEPDialog 这两个对话框是示例中的，我照搬过来使用了。

3、 CImportExport ：数据格式转换类，使用了示例中的数据格式转换类库。

4、 CMainFrame、 CMSApp 、 CMSDoc、 CMSView 、 OcafApplication 是 MFC 运行机制

的基础，在第一个程序里已经做了相关介绍，请查阅。

5、 MySp 和 CMainFrame ：框架类，介绍了把一个框架分成五个部分，一个主显示区、三个视图区和一个建模控制区的方法。

6、 CXProject 、 CYProject 、 CZProject 、 CMSView ：视图类，这几个类都是 CView 的子类，与视图相关的操作都可以在这些类里编写，在这些类中完成了 OCAF 视图类的创建。

6．3．2、CControlWnd 类

CControlWnd 类是从 CFormView 类继承过来的，该类在程序中的作用有两个： 1、建模控制的人机界面，控制显示区（四个显示区）的模型生成流程，位于框架的右侧部分； 2、提供建模功能、布尔操作功能。

6．3．2．1、人机界面

界面的设计做的比较难看，艺术细胞不够，将就了。界面的设计是 VC++ 的内容，不涉及 Open CASCADE 内容，因此简单说一下，有不懂的可以查 MSDN 。

使用 CFormView 类来做界面类， 首先在 resourses的 Dialog 中做一个界面对话框， 如图6-5。

图 6-5

然后，把这个界面对话框与 CControlWnd 类做一个关联，即在申明类的时候写如下代码。

```
class CControlWnd : public CFormView
{
public:
	//{{AFX_DATA(CControlWnd)
	enum { IDD = IDD_CONTROLDLG };
	int		m_SaveFormat;
	//}}AFX_DATA
}
```

最后，把这个界面放置在框架的最右侧框内。在 CMainFrame::OnCreateClient 下面添加如下代码，可查程序文件；

```
m_Sp2.CreateView(0,0,RUNTIME_CLASS(CControlWnd),CSize(400,400),pContext);
```

6．3．2．2、CControlWnd 类的主要功能

一、使用相关类的方法

提供建模和布尔操作的功能，在这里介绍一下，使用 Open CASCADE 的相关方法。

使用 Open CASCADE 实际上就是在使用其提供的一套类库系统，这套类库帮助用户来开发具有三维建模功能的软件。这些类被封装在 DLL 文件和 LIB 文件（申明头文件）里，因此用户在使用 Open CASCADE 的某一个类时，首先，需要告诉编程工具这个类所在的 DLL 和 LIB 文件的位置，这样才能使用这个类，在调试程序一节（第 5 章第 3 节）中已经提到了使用 Open CASCADE 所需要对 VC++ 进行相关路径的设置；然后，还需要在使用这个类之前做类的申明，即在使用该类的头文件里添加相就的 *.hxx 头文件。例如，你想在 CControlWnd ：：CreateCone 下使用 BRepPrimAPI_MakeCone 类生成圆锥体，那么必须在 CControlWnd 头文件中（ ControlWnd.h ）添加 #include "BRepPrimAPI_MakeCone.hxx" 语句。如果不加会报错，告诉你相关类未申明，报错列表如下。 *.hxx 头文件都放在“ 安装目录 \OpenCASCADE6.1.0\ros\inc ”下。

使用 Open CASCADE 其实和 OpenGL 的方法是一样的， OpenGL 的类库少一些。

```
//////////////////* 报错摘要
G:\Me （ 研） \MS 备 份 \ControlWnd.cpp(169) ： error C2065: 'BRepPrimAPI_MakeCone' ：
```

undeclared identifier
G:\Me （研） \MS 备份 \ControlWnd.cpp(169) : error C2146: syntax error : missing ';' before identifier 'mkCone'
G:\Me （研） \MS 备份 \ControlWnd.cpp(169) : error C2065: 'mkCone' : undeclared identifier
G:\Me （ 研 ） \MS 备 份 \ControlWnd.cpp(170) : error C2228: left of '.Shape' must have class/struct/union type
G:\Me （研） \MS 备份 \ControlWnd.cpp(299) : error C2065: 'BRepPrimAPI_MakeSphere' : undeclared identifier
G:\Me （研） \MS 备份 \ControlWnd.cpp(299) : error C2146: syntax error : missing ';' before identifier 'mkSphere'
G:\Me （研） \MS 备份 \ControlWnd.cpp(299) : error C2065: 'mkSphere' : undeclared identifier
G:\Me （ 研 ） \MS 备 份 \ControlWnd.cpp(300) : error C2228: left of '.Shape' must have class/struct/union type
*//////////////////////////

二、建一个圆锥体

在论文里写了圆柱体和棱柱的建模程序, 这里再介绍一下构建圆锥体的过程, 不以论文的模式写，这次解释一下具体编程语句。

CControlWnd 类的大部分的方法是自己编写的, CreateCone 是其中的方法, 作用是用半径、高度、位置和角度参数来建一个圆锥体。

Handle(TDocStd_Document) D = myDoc->GetOCAFDoc();

在 OCAF 的 MFC 单文档应用程序框架建立起来后，就生成了一个文档类（ TDocStd_Document ）, 该文档类可以通过“ Handle(TDocStd_Document) D = myDoc->GetOCAFDoc(); ”语句获得其句柄，然后在程序里使用。在 Open CASCADE 中, 模型都是以文档类下属的参数（ refenerence KEY）形式保存起来的，因此，在建模的过程就是如何在文档类中创建相关的参数的过程。 粉红部分 就是这个过程。

说明几点：

1、 D->NewCommand(); 能够提供恢复功能，恢复次数在“ CMSDoc::OnNewDocument() 的 OCAFDoc->SetUndoLimit(10); ”设置；

2、“ D->Main() ”指向的是文档类的根标签，这个标签不可变的。“ L_Body1= D->Main().FindChild(1); ”是把第一个形体设置为根标签的第一个子标签。

3、“ TDataStd_Real::Set(L_Cone.FindChild(1), r1); ”是把 r1 参数设置为 L_Cone 的第一子标签。

4、设置完标签后, 标签结构如下图 6-6。建立基于 MFC 的应用程序框架的过程中, root 标签 (代表应用程序的标签)和文档 1 根标签 (代表建立的第一个文档类)就已经被创建完成

了，开发人员所需要的工作就是给模型（圆锥体）和其参数分配标签，使用 TDF_Label .FindChild(n) 分配标签。可以利用 DF 窗口来观察标签分配情况。如图 6-7。

5、程序的青色部分是建模过程，建模过程有一套命名系统，可以追逆模型的前世今生。

6、程序的浅蓝部分主要实现模型的可视化功能。给圆锥体标签 L_Cone 配置一个 TPrsStd_AISPresentation 类，该类是把一个 AIS_ InteractiveObject 与标签联系起来，用来控制在 AIS 中的显示。AIS_ InteractiveObject 是控制模型的显示和选取。TPrsStd_AISPresentation 是属于应用框架模块，AIS_ InteractiveObject 是属于可视化模块。这两个类的使用方法可以查阅第二套帮助文档的相关资料。

7、程序里面的 if 语句是为满足编程需要的，没有参考价值。

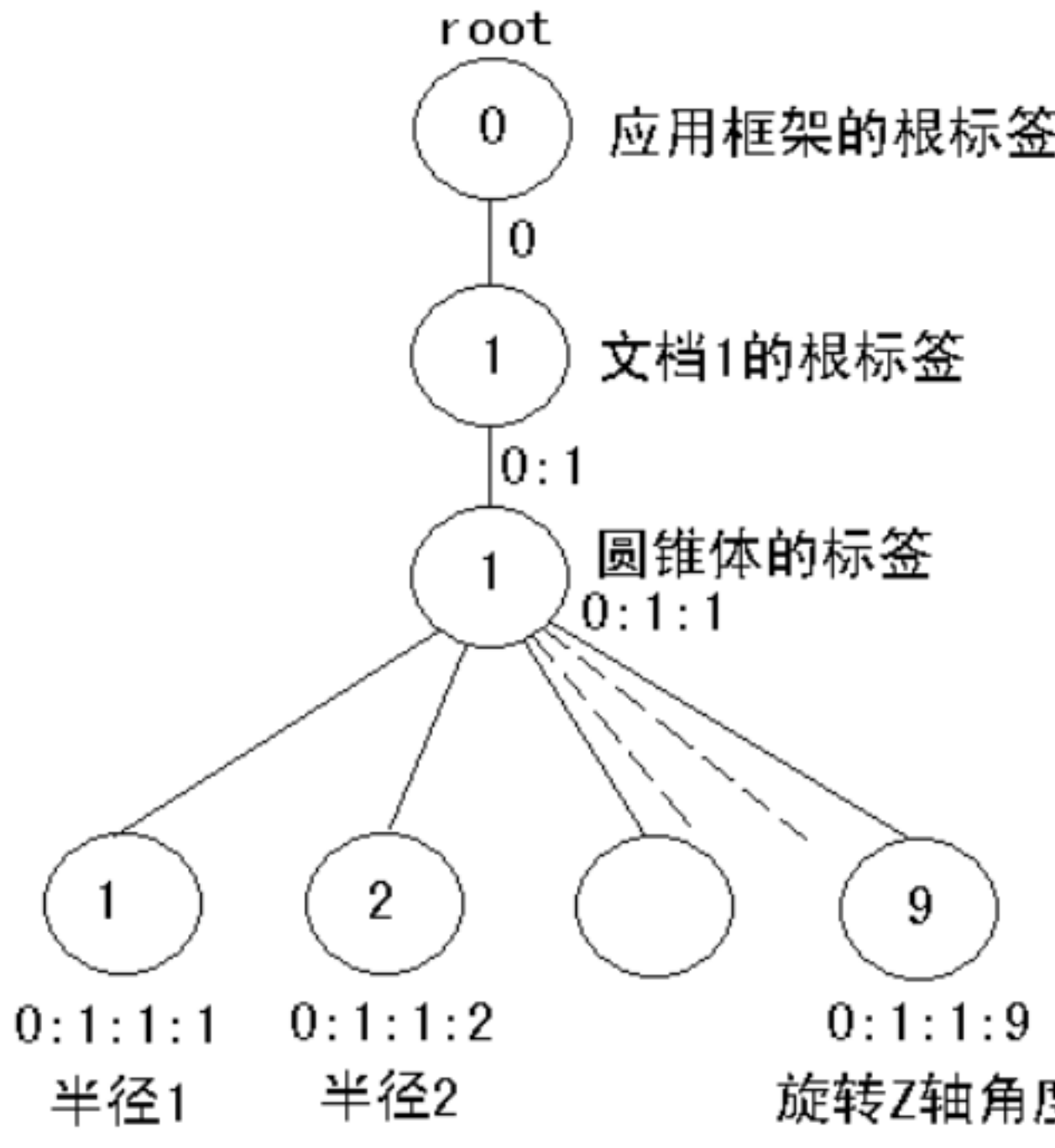

图 6-6

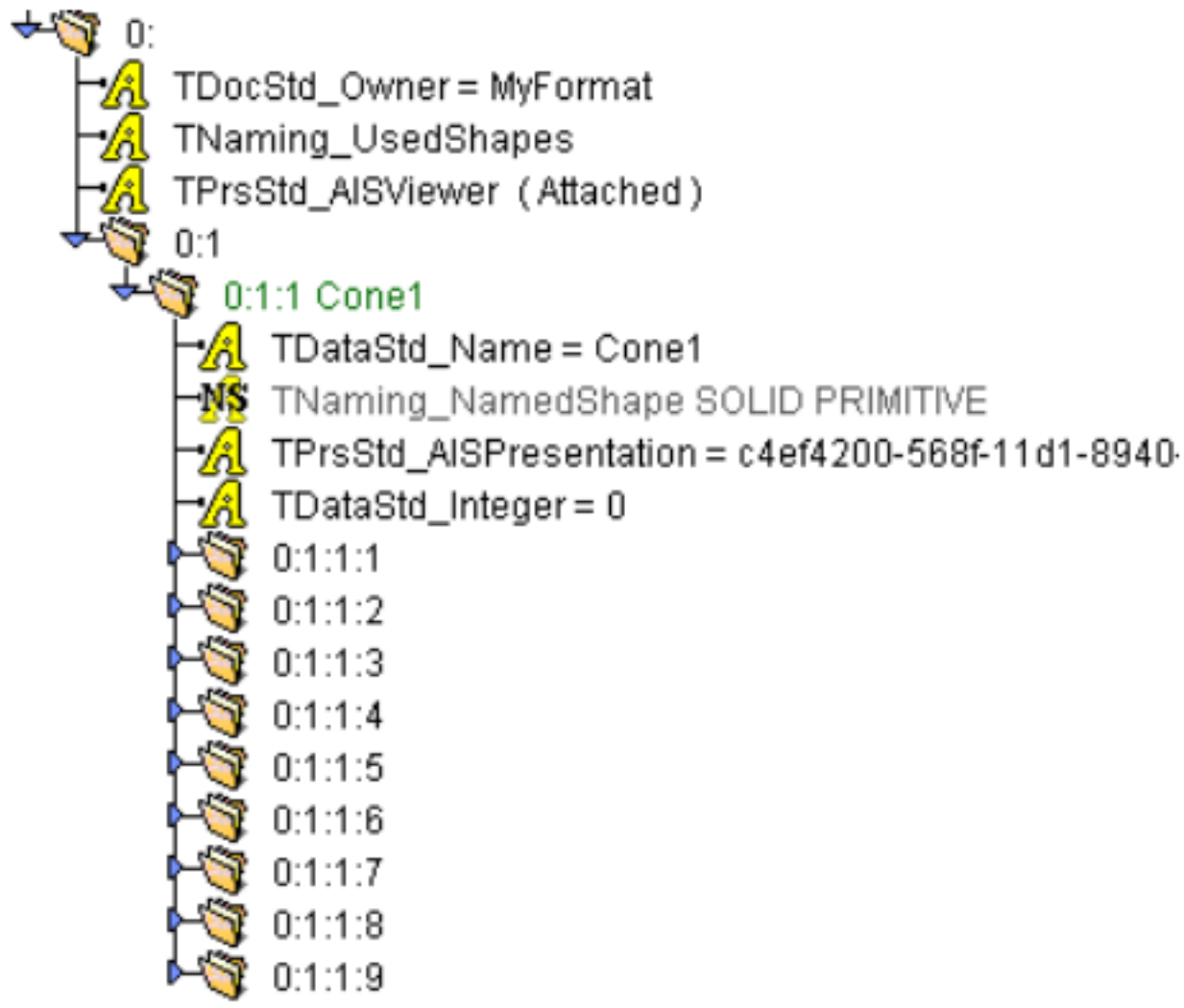

图 6-7

```
void CControlWnd::CreateCone(Standard_Real R1,Standard_Real H,
                                        Standard_Real X_D,Standard_Real Y_D,
                                        Standard_Real X,Standard_Real Y,Standard_Real Z)
{
        Handle(TDocStd_Document) D = myDoc->GetOCAFDoc();
        // Open a new command (for undo)
        D->NewCommand();
        TDF_Label L_Cone;
        if(m_Object_Count==1)
        {
                L_Body1= D->Main().FindChild(1);
                L_Cone=L_Body1;
        }
        else if(m_Object_Count==2)
        {
                L_Body2 =     D->Main().FindChild(2);
                L_Cone=L_Body2;
        }
        CString str="Cone";
        str.Format("%s%d",str,m_Object_Count);
        TCollection_AsciiString Name((Standard_CString)(LPCTSTR)str);
        // Create a new box using the CNewCylDlg Dialog parameters as attributes
        // Create a new label in the data structure for the cylinder
        // Create the data structure : Set the dimensions, position and name attributes
        Standard_Real r1=R1;
        Standard_Real r2=0.1;
```

```
Standard_Real h=H;
Standard_Real x=X;
Standard_Real y=Y;
Standard_Real z=Z;
Standard_Real x_d=X_D;
Standard_Real y_d=Y_D;
Standard_Real z_d=1;
gp_Ax2 Axes(gp_Pnt(x,y,z),gp_Dir(x_d,y_d,z_d));
TDataStd_Real::Set(L_Cone.FindChild(1), r1);
TDataStd_Real::Set(L_Cone.FindChild(2), r2);
TDataStd_Real::Set(L_Cone.FindChild(3), h);
TDataStd_Real::Set(L_Cone.FindChild(4), x);
TDataStd_Real::Set(L_Cone.FindChild(5), y);
TDataStd_Real::Set(L_Cone.FindChild(6), z);
TDataStd_Real::Set(L_Cone.FindChild(7), x_d);
TDataStd_Real::Set(L_Cone.FindChild(8), y_d);
TDataStd_Real::Set(L_Cone.FindChild(9), z_d);
TDataStd_Name::Set(L_Cone, Name);// 给圆锥体标签起个名字“      Cone1”
BRepPrimAPI_MakeCone mkCone(Axes,r1,r2,h);
ResultShape_L_Cone = mkCone.Shape();
TNaming_Builder B(L_Cone);
B.Generated(ResultShape_L_Cone);
// Get the TPrsStd_AISPresentation of the new cylinder TNaming_NamedShape
if(m_Object_Count==1)
{
        prs1= TPrsStd_AISPresentation::Set(L_Cone,TNaming_NamedShape::GetID());
// Display it
        prs1->Display(1);
        hAIS_Object1=prs1->GetAIS();
        L_Body1=L_Cone;
}
else if(m_Object_Count==2)
{
        prs2= TPrsStd_AISPresentation::Set(L_Cone,TNaming_NamedShape::GetID());
// Display it
        prs2->Display(1);
        hAIS_Object2=prs2->GetAIS();
        L_Body2=L_Cone;
}
// Attach an integer attribute to L to memorize it's displayed
TDataStd_Integer::Set(L_Cone, 0);
myDoc->Fit3DViews();
myDoc->myAISContext->UpdateCurrentViewer();
((CMy20Doc*)GetDocument())->myAISContext->SetDisplayMode(((CMy20Doc*)GetDoc
```

ument())->myAISContext->Current(),1);
// Close the command (for undo)
D->CommitCommand();
}

三、布尔操作

布尔操作的前提工作，需要两个模型，

1、 粉红部分设置布尔操作的结果模型

2、 深红色的 部分就是布尔操作的编程部分，这一部分始终没有搞明白， TDF_Reference 、TDataStd_Name、 TopoDS_Shape 和 TNaming_NamedShape 这几个类的用法不甚明了，也没有太多时间、精力去深入研究，所以就不做解释了。

3、 青色 部分和建模的青色部分一样，区别在于，布尔操作是修改模型而不是创建模型。

4、 浅蓝色部分 是可视化功能的实现。

总结：

简单总结一下交集布尔操作的流程

在交集布尔操作中， 首先要创建布尔操作生成模型的标签放置在文档标签下 （程序中设置标签为 L_Bool ）；然后要在布尔模型标签下设置两个基本模型的标签（ Lody1 、Lody2 ）；通过 Lody1 和 Lody2 得到两个基本模型的拓扑形体（ TopoDS_Shape ）；BRepAlgoAPI_Common 可对拓扑形体进行交集的布尔操作，通过 TNaming_Builder 来对此过程进行记录；最后为布尔模型配置一个 TPrsStd_AISPresentation 类与布尔模型的形体 TopoDS_Shape 相关联，执行相关的显示操作。

```
void CControlWnd::OnBtnCommon()
{
    // TODO: Add your control notification handler code here
    Handle(TDocStd_Document) D = myDoc->GetOCAFDoc();
    // Open a new command (for undo)
    D->NewCommand();
    TCollection_AsciiString Name((Standard_CString)(LPCTSTR)"Common");
    // Create a new box using the CNewCylDlg Dialog parameters as attributes
    // Create a new label in the data structure for the cylinder
    L_Bool=D->Main().FindChild(11);;
    TDataStd_Name::Set(L_Bool, Name);
    // Create the data structure : Set a reference attribute
    //on the Original and the Tool objects, set the name attribute
    TDF_Reference::Set(L_Bool.FindChild(1), L_Body1);
    TDF_Reference::Set(L_Bool.FindChild(2), L_Body2);
    Handle(TDataStd_Name) ObjectName;
    L_Body1.FindAttribute(TDataStd_Name::GetID(),ObjectName);
    Handle(TDF_Reference)     OriginalRef, ToolRef;
    L_Bool.FindChild(1).FindAttribute(TDF_Reference::GetID(), OriginalRef );
    TDF_Label OriginalLab = OriginalRef->Get();
    L_Bool.FindChild(2).FindAttribute(TDF_Reference::GetID(), ToolRef);
    TDF_Label ToolLab = ToolRef->Get();
    // Get the TNaming_NamedShape attributes of these labels
```

```
        Handle(TNaming_NamedShape) OriginalNShape, ToolNShape;
        OriginalLab.FindAttribute(TNaming_NamedShape::GetID(),OriginalNShape);
        ToolLab.FindAttribute(TNaming_NamedShape::GetID(),ToolNShape);
        // Now, let's get the TopoDS_Shape of these TNaming_NamedShape:
        TopoDS_Shape OriginalShape    = OriginalNShape->Get();
        TopoDS_Shape ToolShape = ToolNShape->Get();
// STEP 2:
        // Let's call for algorithm computing a cut operation:
        BRepAlgoAPI_Common mkCommon(OriginalShape, ToolShape);
        if (!mkCommon.IsDone())
        {
                ::AfxMessageBox("Cut not done");
        }
        TopoDS_Shape ResultShape = mkCommon.Shape();
        // Build a TNaming_NamedShape using built cut
        TNaming_Builder B(L_Bool);
        B.Modify(OriginalShape, ResultShape);
        prs_Bool= TPrsStd_AISPresentation::Set(L_Bool, TNaming_NamedShape::GetID());
        TDataStd_Integer::Set(L_Bool, 1);
        prs_Bool->Display(1);
        myDoc->Fit3DViews();
        TDataStd_Integer::Set(L_Body1, 0); ////   去掉此句一样效果
        prs1->Erase(0);
        TDataStd_Integer::Set(L_Body2, 0); ////   去掉此句一样效果
        prs2->Erase(0);
        // Attach an integer attribute to L to memorize it's displayed
        myDoc->myAISContext->UpdateCurrentViewer();
        // Close the command (for undo)
        D->CommitCommand();
        ((CButton*)GetDlgItem(IDC_BTN_CONFORM2))->EnableWindow(true);
        ((CButton*)GetDlgItem(IDC_BTN_REFRESH2))->EnableWindow(true);

        ((CButton*)GetDlgItem(IDC_BTN_FUSE))->EnableWindow(false);
        ((CButton*)GetDlgItem(IDC_BTN_INTERSECTION))->EnableWindow(false);
        ((CButton*)GetDlgItem(IDC_BTN_CUT))->EnableWindow(false);
        ((CButton*)GetDlgItem(IDC_BTN_COMMON))->EnableWindow(false);
        ((CComboBox*)GetDlgItem(IDC_COMBO_MODEL))->EnableWindow(false);
}
```

6．3．3、对话框

对话框是用户和程序交互的重要手段，除去系统自带的，该程序用了五个对话框类，

CFileSaveCSFDBDialog 和 CFileSaveSTEPDialog 类是用来保存 CSFDB 和 STEP 格式的对话

框，其他三个瞎弄的。

6．3．4、CImportExport

该类的作用是把生成的模型保存为现今流行的三维模型标准格式，有 Step、BREP、IGES、CSFDB 等格式。在保存之前需要用户先在显示区内选取模型，然后保存会弹出对话框让用户选择保存的标准文件格式名，选择完成后，使用 CImportExport 类的相关静态方法来保存。如果想做这方面的程序，可以把 CImportExport 类的相关文件拷入自己的程序里使用即可。

```
void CControlWnd::OnBtnSave()
{
	// TODO: Add your control notification handler code here
	UpdateData();
	switch(m_SaveFormat)
	{
	case 0:
		CImportExport::SaveBREP(myDoc->myAISContext);
		break;
	case 1:
		CImportExport::SaveSTEP(myDoc->myAISContext);
		break;
	case 2:
		CImportExport::SaveSTL(myDoc->myAISContext);
		break;
	case 3:
		CImportExport::SaveIGES(myDoc->myAISContext);
		break;
	case 4:
		CImportExport::SaveCSFDB(myDoc->myAISContext);
		break;
	default:
		break;
	}
}
```

6．3．5、MsSp、框架类和视图类

6．3．5．1 MsSp 、框架类

框架的分割使用 CSplitterWnd 来完成（这个类的具体用法请参照 MSDN），先做一个类 MySp，然后 CMainFrame::OnCreateClient 下编写分割程序，把各个视图类分配在每个显示区域内，详情查阅程序。这一部分不涉及 Open CASCADE，是 VC++ 内容，我记得当时是

照着别人的程序葫芦画瓢画出来的，所以就不细说了。

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext*
pContext)
{
    // TODO: Add your specialized code here and/or call the base class
    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*2/3,100);
    //将左边的分为两行一列
    if(m_Sp3.CreateStatic(&m_Sp1,2,1,WS_CHILD|WS_VISIBLE,
m_Sp1.IdFromRowCol(0, 0))==NULL)
        return FALSE;
    m_Sp3.SetRowInfo(0,100,100);
    //最大的窗口
    m_Sp3.CreateView(0,0,RUNTIME_CLASS(CMSView),CSize(cx*3/4,cy-324),
pContext);
    //将第二行分为一行三列
    if(m_Sp4.CreateStatic(&m_Sp3,1,3,WS_CHILD|WS_VISIBLE,
m_Sp3.IdFromRowCol(1, 0))==NULL)
        return FALSE;
    m_Sp4.SetRowInfo(0,224,200);
    m_Sp4.CreateView(0,0,RUNTIME_CLASS(CXProject),CSize(224,224),pContext);
    m_Sp4.CreateView(0,1,RUNTIME_CLASS(CYProject),CSize(224,224),pContext);
    m_Sp4.CreateView(0,2,RUNTIME_CLASS(CZProject),CSize(224,224),pContext);
    //将右边的分为两行一列
    if(m_Sp2.CreateStatic(&m_Sp1,1,1,WS_CHILD|WS_VISIBLE,
m_Sp1.IdFromRowCol(0, 1))==NULL)
        return FALSE;
    m_Sp2.CreateView(0,0,RUNTIME_CLASS(CControlWnd),CSize(400,400),pContext);
    return true;
}
```

6．3．5．2 视图类

CMSView、CXProject、CYProject 和 CZProject 为模型的显示区，CMSView 是主显示区，CXProject、CYProject 和 CZProject 是三视图的显示区。

CMSView 类虽然方法很多，但是都是比较简单的操作，在生成该视图类之后，在构造函数、CMSView::OnInitialUpdate()、CMSView::OnSize(UINT nType, int cx, int cy) 和 CMSView::OnDraw(CDC* pDC) 下做一些基本的配置就可以使用视角控制的相关操作。

CXProject、CYProject 和 CZProject 显示三视图并实时的刷新模型生成变化，这里的三

视图并不是真正意义的三视图，　　真正的三视图应该是二维的，　　但是程序中的只是把三维模型的视角固定在某一个平面上。

6．3．6 其他问题

这里要说一下卡机和程序崩溃的问题。

做三视图的程序时候，出现一种情况，在移动三维模型的时候会出现　　LAG（延时），也就是卡机关于　LAG　问题，可能原因一：有可能是当年用的机器比较烂　　512M 内存、 1GHZ 主频，移动模型时（移动模型的过程就是生成新位置模型来不停地取代旧位置模型的过程）由于计算量比较大，　所以会卡机；　原因二：　内存整理速度跟不上，　导致堆栈溢出报错。　Open CASCADE　的基类模块中有内存整理的相关类，　也可以使用　C++ 的相关内存类来编写相关程序。

在移动模型的时候，　经常会出现程序崩溃的情况。　这可能也是与内存有关，　没有及时进行内存的相关整理，导致堆栈溢出，程序崩溃。

现在的机器没有出现过卡机的情况，　　2GHZ 主频 P4，1G 内存。

关于这两个错误，我并没有经验和能力去处理，出错就出错，就这么着吧。

6．4、MS 程序 2

这是最近做的一个程序，这个程序和　　MS1 程序很类似，只比上一个程序稍微完善了一下，顺便又多出个功能来。

程序名：　MS 程序 2.rar

6．4．1 数据结构

在使用　Open CASCADE　开发工具时，模型数据以标签形式　　(TDF_Label) 存放在文档类中，　MS 程序 2 的数据结构如图　6-8，

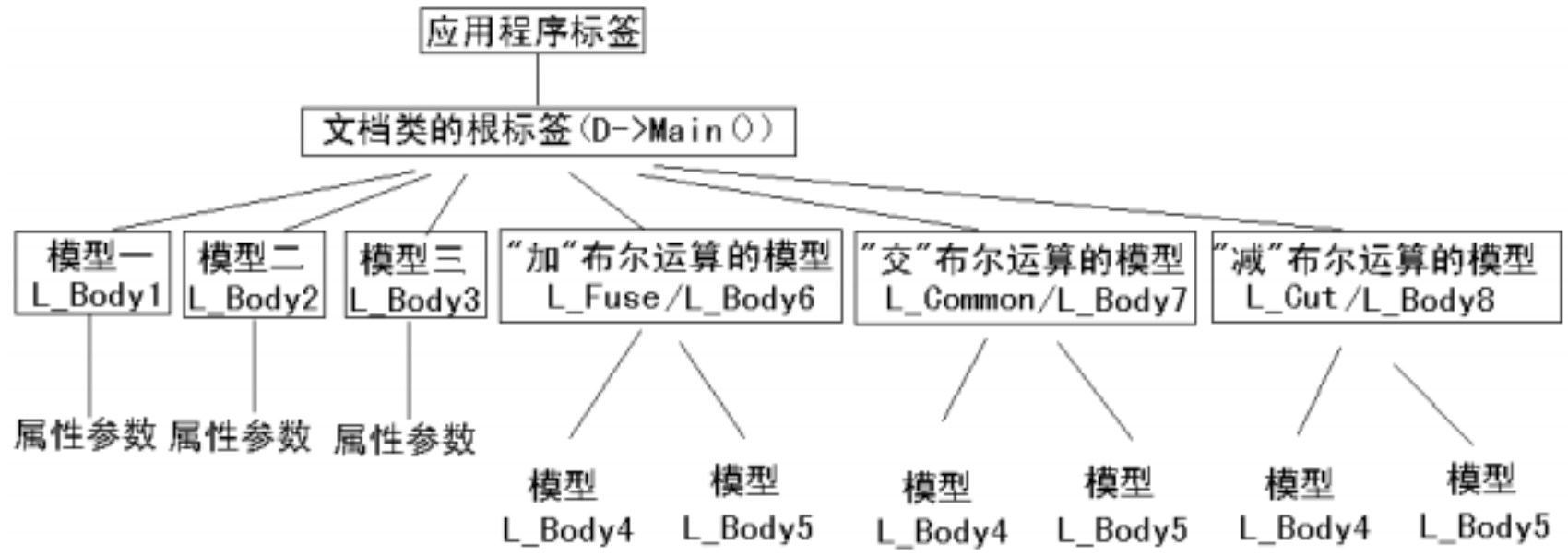

图 6-8

6．4．2 选择模式

使用 AIS_InteractiveContext 类可完成模型选取功能，即如下程序段：

```
for
(myAISContext->InitSelected();myAISContext->MoreSelected();myAISContext->NextSelected())
    ○○○○○○○○
}// 检测选取模型的个数
    if((myAISContext->IsSelected(hAIS_Object2))&&(myAISContext->IsSelected(hAIS_Objec
t3)))
    {
        L_Body4=L_Body2;// 为布尔运算提供标签
        L_Body5=L_Body3;
        prs2->Erase(0);//把参与布尔运算的两个模型隐藏
        prs3->Erase(0);
    }
    else
if((myAISContext->IsSelected(hAIS_Object1))&&(myAISContext->IsSelected(hAIS_Object3)))
    {
        L_Body4=L_Body1;
        L_Body5=L_Body3;
        prs1->Erase(0);
        prs3->Erase(0);
}
```

第 7 章 总结

Open CASCADE 工具使得开发人员可以从底层实现二维或三维模型的建模，实现各种

曲线建模、 各种曲面建模、 布尔操作、 模型标准化、 模型可视化等等功能。 优点是功能强大，使用免费，实现灵活，它提供的 OCAF 框架能够帮助开发人员快速的进行开发；缺点是开发难度比较大，需要熟练运用开发工具（即编程工具，如 VC， Dephin 等），并能够清楚了解 Open CASCADE 的系统结构。

我对 Open CASCADE 的使用