

学校代号 10532

分 类 号 TB122

学 号 S1802W0236

密 级 公开



硕士学位论文

基于 Qt 和 VTK 的静力学 有限元软件 GUI 开发

学位申请人姓名	汤光泽
培 养 单 位	机械与运载工程学院
导师姓名及职称	崔向阳教授 田永高级工程师
学 科 专 业	车辆工程
研 究 方 向	车辆工程
论文提交日期	2020 年 04 月 20 日

学校代号：10532
学 号：S1802W0236
密 级：公开

湖南大学硕士学位论文

基于 Qt 和 VTK 的静力学有限元 软件 GUI 开发

学位申请人姓名：	汤光泽
导师姓名及职称：	崔向阳教授 田永高级工程师
培 养 单 位：	机械与运载工程学院
专 业 名 称：	车辆工程
论文提交日期：	2020 年 04 月 20 日
论文答辩日期：	2020 年 05 月 29 日
答辩委员会主席：	方棋洪教授

**Development of GUI for static finite element software
based on Qt and VTK**

by

TANG Guangze

B.E.(North University of China)2018

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Engineering

in

Vehicle Engineering

in the

Graduate school

of

Hunan University

Supervisor

Professor Cui Xiangyang

Senior Engineer Tian Yong

May,2020

湖 南 大 学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的
研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或
集体已经发表或撰写的 成果作品。对本文的研究做出重要贡献的个人和集体，
均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：邵光祥 日期：2020 年 4 月 20 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保
留并向国家有 关部门或机构送交论文的复印件和电子版，允许论文被查阅和借
阅。本人授权湖南大学可以将本学位论文的全部或部分内容编入有关数据库进行
检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

1.密□，在____年解密后适用本授权书。

2.不保密☑。

(请在以上相应方框内打“√”)■

作者签名：邵光祥 日期：2020 年 4 月 20 日

导师签名：杨明 日期：2020 年 4 月 20 日

摘 要

随着工业化进程的不断加深, CAE 软件越来越多地被应用到各个领域, 如机械、建筑、水利等关乎国民经济的传统领域, 航空航天、军事国防、爆炸爆破等具有战略地位的核心领域, 以及生物化学、天文学等应用前景较大的新兴领域。本文基于图形用户界面框架 Qt 和可视化工具包 VTK, 以开发静力学有限元软件前后处理系统为目标, 系统地研究了 Qt、VTK 的用法和前后处理系统的集成, 具体内容如下:

首先, 本文系统地深入研究了 Qt 和 VTK 的总体特性和组织结构, 详细介绍了 Qt 的信号与槽通信机制以及 Qt 中常用的功能模块, VTK 中的数据结构、可视化管线及常用的可视化算法等, 并分析了使用 Qt 和 VTK 进行前后处理软件系统开发的优势所在。在本前后处理软件系统中, Qt 主要用于软件界面的开发和各功能控件的集成, VTK 主要负责界面中的视图功能区以及有限元模型的前、后处理显示。

然后, 按照软件研发的技术路线, 本文从软件设计目标、框架设计、功能设计等方面阐述了本前后处理软件系统的研发过程。并给出了系统中部分主要功能和关键技术的实现方法和程序代码, 包括坐标系、颜色条、文本注释等视图中的部件, step、stl、inp、vtk 等各种文件类型的读取及模型的建立, 平移、旋转、缩放、拾取等视图交互样式, 云图、切面图、切割图、等值面(线)图、变形图等后处理可视化方法。

最后, 在本前后处理软件系统开发完成并测试通过的基础上, 介绍了使用悬臂梁算例和汽车白车身算例进行演示的过程, 展示了系统所具备的功能及其使用方法。悬臂梁算例的演示过程验证了本软件具备完整的有限元分析过程以及各功能模块的操作方法, 汽车白车身算例的演示过程说明了本软件具备与 Abaqus 等主流软件进行数据交换的接口, 能够准确处理其产生的模型数据。

关键词: CAE; Qt; VTK; 前后处理; 可视化

Abstract

With the further development of industrialization, CAE software is more and more applied to various fields. The traditional fields such as machinery, construction and water conservancy that relate to the national economy. The core fields such as aerospace, military defense and explosive blasting that have the strategic position. The emerging fields such as biochemistry and astronomy that have the larger application foreground. The paper is based on the graphical user interface framework Qt and visualization toolkit VTK, with the goal of developing a static finite element software pre-processing and post-processing system. The usage of Qt and VTK and the integration of pre-processing and post-processing system are studied systematically. The details are as follows:

Firstly, the paper systematically and intensively studies the overall characteristics and organizational structure of Qt and VTK. It detailedly introduces the signals and slots communication mechanism of Qt, functional modules commonly used in Qt, and the data structures, visualization modules and common visualization algorithms in VTK. And it analyzed the advantages of using Qt and VTK for pre-processing and post-processing software system development. In this pre-processing and post-processing software system, Qt is mainly used for the development of the software interface and the integration of various functional controls. VTK is mainly responsible for the view function area in the interface and the pre-processing and post-processing display of the finite element model.

Secondly, in accordance with the technical route of software development, the paper expounds the development process of the pre-processing and post-processing software system from the design goals, system architecture and functions. In addition, it also gives the implementation methods and program codes of some main functions and key technologies in the system. The components such as coordinate system, scalar bar and text annotation that lie in the view. The file type such as step, stl, inp and vtk that store model information. The interaction style such as pan, rotate, zoom, picker that manipulate the view. The post-processing such as cloud map, cutter map, clip map, contour map and deformation map that display the result of calculation.

Finally, the paper introduces the process of demonstration using cantilever beam examples and car body-in-white examples after the completion and development of this pre-processing and post-processing software system. And it shows the functions and usages of this software system. The demonstration process of the cantilever beam

example verified that the software has a complete finite element analysis process and the operation method of each functional module. The demonstration process of the body-in-white example shows that the software has an interface for data exchange with mainstream software such as Abaqus, and can accurately process the model data generated by it.

Key Words:CAE;Qt;VTK;Pre- and Post-processing;Visualization

目 录

学位论文原创性声明和学位论文版权使用授权书	I
摘 要	II
Abstract	III
目 录	V
第 1 章 绪 论	1
1.1 研究背景及意义	1
1.2 CAE 软件发展现状及趋势	2
1.2.1 国外 CAE 软件发展现状及趋势	2
1.2.2 国内 CAE 软件发展现状及趋势	2
1.2.3 国内外 CAE 软件差距分析	3
1.2.4 国内 CAE 软件发展的可行途径	3
1.3 本文研究工作及内容	4
第 2 章 外部依赖库研究	5
2.1 Qt 库研究	5
2.1.1 信号与槽机制	5
2.1.2 元对象编译器	6
2.1.3 常用 Qt 模块及类库	7
2.1.4 使用 Qt 开发的优势	7
2.2 VTK 库研究	8
2.2.1 可视化管线	8
2.2.2 渲染引擎	13
2.2.3 使用 VTK 开发的优势	17
2.3 Qt 与 VTK 的整合	18
2.4 本章小结	20
第 3 章 软件系统设计与实现	21
3.1 软件系统设计目标	21
3.2 软件系统框架设计	22

3.3 软件系统功能设计	23
3.3.1 文件读取功能设计	23
3.3.2 视图功能设计	24
3.3.3 前处理模块功能设计	24
3.4 软件主要功能实现	26
3.4.1 视图部件	26
3.4.2 文件读取及建模	28
3.4.3 视图交互	33
3.4.4 后处理部分	40
3.5 本章小结	45
第 4 章 软件介绍及应用	46
4.1 主界面介绍	46
4.2 算例分析	46
4.2.1 系统硬件环境	46
4.2.2 悬臂梁算例	47
4.2.3 汽车白车身算例	53
4.3 本章小结	55
总结与展望	56
参考文献	58
致谢	61

第 1 章 绪 论

1.1 研究背景及意义

CAE (Computer Aided Engineering, 计算机辅助工程) 是指借助计算机技术以仿真的方式对工程问题或产品进行计算、分析和优化的过程, 其核心是基于现代计算力学的有限元分析技术^[1]。作为其衍生产品, CAE 软件自上世界 50 年代诞生以来, 由于其能够在产品开发中部分甚至全部代替真实的实验过程, 从而大大降低了开发成本、缩短开发周期、提升产品质量, 在某些特殊领域, 还能够取代具有危险性和破坏性的实验, 也在一定程度上保障了人民的生命财产安全。CAE 软件发展至今, 已被广泛地应用于机械、汽车、航空航天、国防军事、土木、水利、生物化学以及爆炸等各个领域。

当前, 我国正处在实现“工业制造 2025”、“制造强国”等战略目标的关键时期^[2], 需要大量优秀的分析计算软件做支撑。然而, 由于重视程度不高、投入少等原因, 加上我国计算机技术的应用起步较晚, 我国的 CAE 软件市场几乎完全被国外相关产品垄断, 如 Dassault 公司的 Abaqus 软件、ANSYS 公司的 ANSYS 软件、LSTC 公司的 LS-DYNA 软件、MSC.Software 公司的 DYTRAN 软件等等, 这些软件经过长期有效的发展, 均具备较强的通用分析能力和广泛的应用范围。然而, 这些国外商业软件在国内的市场并不十分稳固, 一是 CAE 软件的应用往往关系到国家的战略安全问题, 在某些涉及核心机密的领域可能存在信息泄露的风险, 另一方面, 由于国家之间的博弈与竞争, 国外商业软件在一些敏感模块可能会对国内企业设置使用壁垒, 从而严重影响企业的创新和开发能力。二是价格高, CAE 软件的广大市场在于国内占较大比例的中小企业, 而国外商业软件高昂的费用是很多小企业无法承担的, 这些费用极大地增加了企业的产品研发成本, 并造成一定程度的经济损失。三是在某些专业领域, 国外通用的 CAE 软件存在使用繁琐、计算量大和分析效率低等问题, 不能完全满足甚至根本不能满足企业需求, 需要开发专门的算法才能进行分析计算。除此之外, 近年来随着国家和全社会对关键核心技术的重视程度越来越高, 对 CAE 软件的研发投入也越来越大, 越来越多的 CAE 从业者愿意投身到 CAE 软件的研发事业中, 为发展具有自主知识产权的 CAE 软件贡献力量。

由此可见, 发展国产自主可控 CAE 软件不仅能够助力我国顺利实现“中国制造 2025”和“制造强国”等战略目标, 对提高我国的核心竞争力与自主创新能力、凝聚科技创新人才等都具有重大意义, 也有利于国家经济增长方式的成功转型, 为经济增长提供动力, 更关乎到国家的信息安全和战略安全^[3]。

1.2 CAE 软件发展现状及趋势

1.2.1 国外 CAE 软件发展现状及趋势

国外的 CAE 软件于上世纪六十年代开始应用于实际的工程分析中, 经过三十余年的快速发展, 到上世纪九十年代时, 已经能够基本满足工业分析需求。之后由于有限元理论没有取得重大突破, CAE 软件的求解器技术也没有太大进步, 各企业主要以拓展市场为主。也是在这个时候, 国外的商业软件开始大举进入我国市场, 并迅速占领甚至垄断了我国的 CAE 软件市场^[4]。尤其是最近十几年的时间, 国外的企业开始了大规模的并购进程, 如 Dassault、SIEMENS、ANSYS、ESI 和 MSC 等等。在并购的过程中, 这些企业的 CAE 软件越来越成熟, 但也各具特色, 如 Dassault 公司的 Abaqus 软件在处理非线性问题领域十分出色, 能够分析复杂的固体和结构力学问题, 对于非常庞大的高度非线性问题也能够进行准确地分析求解^[5]; ANSYS 公司的 ANSYS 软件能够集固体、流体、温度场、电场、磁场、声场等分析于一体, 擅长处理多物理场耦合问题^[6]; MSC 公司的 Nastran 软件是美国国家航空航天局为满足航空航天领域的结构分析问题而开发的, 具有极高的软件可靠性和优秀的软件品质^[7]。简单来说, 国外 CAE 软件的发展经历了三个阶段, 即以技术提升为主的发展阶段、以拓展市场为主的壮大阶段和以整合市场为主的成熟阶段。国外的 CAE 软件能够发展到今天的水平, 一是离不开国外工业化进程快、程度高, 其国内企业对 CAE 软件具有极大的应用需求; 二是离不开国外政府的高度重视和大量投入, 给 CAE 软件的发展提供了强有力的支持; 三是离不开科研院所的技术研发、软件厂商的市场营销和相关企业的应用验证相配合, 三者形成良性循环, 进一步促进了 CAE 软件的发展^[8]。

1.2.2 国内 CAE 软件发展现状及趋势

我国对 CAE 技术的研究起步并不晚, 尤其是对有限元理论的研究, 早在上世纪六十年代, 我国数学家冯康就提出了独立于西方理论的基于变分原理的差分方法^[9], 标志着有限元法在我国的诞生。之后, 有限元理论在我国得到进一步发展, 如钟万勰、钱令希、梁国平、顾元宪等人提出的一系列涉及不同领域的有限元方法^[10-17]。自上世纪七十年代开始, 我国就开始了专有数值计算程序的开发, 并在科研院所和一些专有领域得到应用^[18]。进入二十一世纪, 我国出现了一批依托高校和科研院所研发的具有自主知识产权的专用 CAE 软件, 如北京大学研发的用于微机结构分析的 SAP84 软件^[19]、郑州机械研究所研发的用于对机械产品进行结构分析计算的紫瑞 CAE 软件^[20]、华中科技大学研发的用于铸造领域的华铸 CAE 软件^[21]、吉林大学研发的用于覆盖件弹塑性大变形领域的 KMAS 软件^[22]、中国工程物理研究院研发的用于结构冲击响应领域的 DynPack 软件^[23]、中国飞机强度研究所研发的用于航空领域的 HAJIF 软件^[24]、合肥工业大学研发的用于三维

疲劳裂纹扩展领域的 ALOF 软件^[25]、大连理工大学研发的面向 CAE 工程与科学计算集成化的 SiPESC 软件平台^[26]和用于有限元分析与优化设计的 JIFEX 软件^[27]、中国科学院数学与系统科学研究院研发的能够自动生成有限元程序的 FEPG 有限元分析软件平台^[28]、西南交通大学研发的面向对象跨系统 CAE 软件平台 OmtDesk^[29]等等。这些软件在各自领域均具有较大的知名度和影响力，但由于其中绝大多数都用在专业领域和教学科研中，加上国外商业软件的挤压，导致这些国产软件没能顺利朝着商业化和通用化发展。在当前国家大力提升核心竞争力和重视自主知识产权的大背景下，发展国产自主可控 CAE 软件已经成为了全社会的共识，相信在全社会各行各业的共同努力下，国产 CAE 软件定能实现弯道超车。

1.2.3 国内外 CAE 软件差距分析

不可否认，目前国内的 CAE 软件与国外相比还存在较大的差距，无论是软件本身的质量还是市场化程度，均有较大的进步空间。国内外 CAE 软件的差距主要有以下几个方面：

(1) 技术积累。CAE 软件涉及到的知识面广，需要多领域的科研人员共同攻关。然而国内的研发团队分散，大多各自为战，工作重复度高，且研发者多为高校在读学生和科研院所人员，其编写的程序大多面向科研项目，而不是企业和市场，因此难以达到商用标准，可利用价值不高。

(2) 重视程度和投入。CAE 软件的研发周期长、成本高，是一项长期工作，不是一蹴而就的，它需要在技术上不断更新换代、在功能上不断完善扩展，并且在短期内难以看到显著成果，因此需要坚持不懈的努力和长期稳定的投入^[30]。

(3) 声誉度、影响力差距。谈起 Abaqus、ANSYS 等国外商业软件，业内人士几乎无人不知，相比之下，国内即使是比较优秀的 CAE 软件，其声誉度和影响力也相差甚远。

(4) 市场占有率低。国外 CAE 软件自上世纪九十年代开始进入我国市场以来，就一直占据着绝大部分的市场份额，在我国获取了巨大的经济利益，导致国内 CAE 软件的市场占有率极低，没有市场软件就难以发展，发展不好就没有用户，这样就形成了恶性循环。

(5) 参与度。发展自主 CAE 软件不仅需要软件自身的努力和进步，还需要广大用户的全力支持和陪同成长，不断在问题中发展和完善。然而目前在很多高校的有限元分析课程都直接忽略理论知识的教授，直接把学会使用国外某商业软件作为教学目标，这样的做法怎能培养出潜心发展自主 CAE 软件的科研人才。

1.2.4 国内 CAE 软件发展的可行途径

CAE 软件具有重要的经济价值和战略地位，担负着实现“中国制造 2025”和“制造强国”战略目标的重任，其严峻性和紧迫性已形成共识。在当前国内市场几乎被国外商业 CAE 软件垄断的局面下，国产 CAE 软件要想实现弯道超车，需要找准市场切入点，并付出长期艰苦的努力。

（1）构建国家意志。CAE 软件的研发需要提升到国家层面，制定明确、长远的战略规划，不能仅仅把它简单地当作科研项目，由高校和科研院所完成，我们应该像发展高铁和航母一样发展具有自主知识产权的国产 CAE 软件^[8]。

（2）面向市场。CAE 软件想要具备长久的生命力，必须进行商业化开发，让软件研发部门和企业用户形成良性循环，既能提升企业用户的经济效益，又能促进 CAE 软件本身的质量和影响力^[31]。

（3）重视高校教育。开设 CAE 相关专业课程，注重多学科、多领域协作互动，大力培养 CAE 软件研发人员，让有能力、感兴趣的学生有机会参与到国产自主可控 CAE 软件的研发工作中去。

1.3 本文研究工作及内容

本文主要对 CAE 软件前后处理系统的开发工作进行介绍，该系统基于图形用户界面开发框架 Qt 和可视化工具包 VTK 进行开发，具备导入几何模型、建立前处理模型、分析计算以及后处理可视化等功能，本文的主要研究工作及内容如下：

（1）深入研究 Qt 框架的基本组成和总体特性，研究界面控件响应原理和机制，了解常用的 Qt 模块及其作用；深入研究 VTK 工具包的运行机制和组织结构，研究其数据类型和可视化流程；并分析了使用 Qt 和 VTK 开发的优势。

（2）研究 Qt 和 VTK 的整合方法，并基于 Qt 和 VTK 开发前后处理软件系统，前处理模块具备导入模型、网格剖分，以及定义材料、截面属性、分析步、输出变量、载荷、边界条件、单元属性等功能，后处理模块包括云图、切面图、切割图、等值面（线）图和变形图等可视化方法。

（3）利用悬臂梁和汽车白车身两个算例演示本前后处理软件系统的使用方法和操作流程。悬臂梁算例演示了从导入几何模型到后处理的全部流程，汽车白车身算例演示了导入 inp 格式文件的模型在修改约束后进行计算的过程。

第 2 章 外部依赖库研究

为降低开发难度、节省开发时间，本前后处理软件系统的开发主要依赖了图形用户界面框架 Qt 和可视化工具包 VTK 两个外部库，此外还用到了几何引擎 Open CASCADE 库，其中 Qt 库用来设计软件界面窗口及窗口中的控件，VTK 库用来实现软件中的视图功能区域，完成模型的可视化及其他相关操作，Open CASCADE 库用来读取 step/iges 等几何模型，并在 VTK 中进行渲染。由于本文主要对 Qt 和 VTK 进行了较为深入的研究，下面主要对这两个外部依赖库的研究结果进行介绍。

2.1 Qt 库研究

Qt 是一款可用于开发桌面、嵌入式和手机程序的跨平台 C++ 应用程序开发框架。自 1995 年第一个公开版本发布以来，在二十多年的发展过程中，Qt 的功能不断强大，用户群体不断壮大，已成为图形用户界面设计最常用的工具之一。

2.1.1 信号与槽机制

信号与槽机制作为 Qt 的核心功能在 Qt 编程中得到了广泛的应用，这也是 Qt 与其它 GUI 框架最显著的区别之一。在许多常用的 GUI 框架中，通常通过回调函数来响应它们所触发的动作，而这种方式存在类型安全性低、与 GUI 耦合度高等缺点，不利于实现具有复杂关系的动作响应^[32]。然而使用 Qt 中的信号与槽机制既可以实现回调函数所要实现的功能，又具备完全类型安全、独立于 GUI 事件循环等优点。

信号与槽功能用于 Qt 对象之间的通信，常用于 GUI 上控件动作响应，如单击某窗口的“关闭”按钮，关闭窗口，即响应该按钮的关闭动作。但信号与槽功能的使用不仅仅限于此，在实际的编程过程中，会经常用到该功能。信号和槽机制中有三个关键元素：信号（signal）、槽（slot）、连接（connect），其本质均为函数。

当某对象的内部状态发生改变，且该状态的改变会导致其他对象状态的改变时，该对象将发射信号，同时与该信号关联的槽函数将会被立即执行。在 Qt 中，信号的发射使用 emit 关键字，如“emit signalFunction();”，信号的声明使用 signals 关键字，均为 public 类型，因此可以从任何地方发出。与一般的 C++ 成员函数不同，信号函数没有函数的实现部分，而是由元对象编译器（moc）自动产生。Qt 给各控件预先定义了许多信号，以用来响应控件各种内部状态的改变，这些信号可直接使用，同时也支持自定义信号，以实现特殊的事件响应。

当某信号发出时，与该信号相连的槽函数将会被立即执行。槽函数是普通的 C++ 成员函数，其声明使用 slots 关键字，也分为 public、protected 和 private 三种

访问类型，槽函数的实现部分即该信号功能的实现。调用时，槽函数既可以用来与信号进行关联，也可以像调用一般的 C++ 函数一样被调用。与信号类似，Qt 给各控件预先定义了许多槽，但大多数情况下这些预先定义的槽并不能满足我们的要求，因此需要自定义槽函数以实现特定的功能。

信号与槽使用 `connect(object1,signal,object2,slot)` 函数进行连接，`object1` 参数为发出信号的对象，`object2` 参数为接受信号的对象，`signal` 为信号函数，`slot` 为槽函数。信号与槽的连接并不是一一对应的关系，一个信号可以同时连接多个槽，一个槽也可以同时连接多个信号，如图 2.1 所示。当一个信号同时与多个槽连接时，将按照连接的顺序依次执行各槽函数。

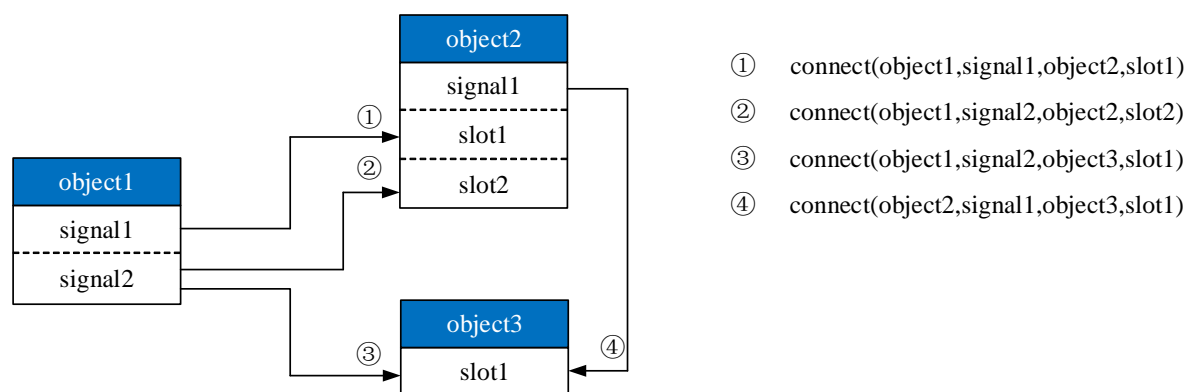


图 2.1 Qt 信号与槽

Qt 中还提供了将信号与槽断开连接的功能，当信号与槽没有继续保持连接的必要时，可以使用 `disconnect()` 函数来断开连接，该函数与 `connect()` 函数类似，需要指定发射信号对象、接收信号对象、信号函数和槽函数四个参数。常见的断开连接有以下四种情况：

- ① `disconnect(object1,signal,object2,slot)`. 将 `object1` 对象的 `signal` 信号与 `object2` 对象的 `slot` 槽断开连接；
- ② `disconnect(object1,signal,0,0)`. 将 `object1` 对象的 `signal` 信号与所有已连接的槽断开连接；
- ③ `disconnect(object1,0,object2,0)`. 将 `object1` 对象与 `object2` 对象的所有已连接信号与槽断开连接；
- ④ `disconnect(object1,0,0,0)`. 将 `object1` 对象的已连接的所有信号断开连接。

2.1.2 元对象编译器

由于 Qt 的信号与槽机制是 Qt 自定义的一种通信机制，它独立于标准的 C/C++ 语言，因此普通的 C/C++ 编译器无法正确处理 Qt 中的信号与槽，必须借助 Qt 中的元对象编译器（moc）将信号与槽进行预处理，转化成普通的 C/C++ 编译器能够处理的代码。元对象编译器在分析 C++ 源文件时，如果发现某个头文件的类声明中包含 `Q_OBJECT` 宏，且该类直接或间接继承自 `QObject` 类，则元对象编译器将

会生成另一个名称带有“moc_”前缀 C++源文件,该文件中包含这些类的元对象代码,将会与原 C++源文件一起参与编译。因此,所有包含信号或槽的类都必须添加 Q_OBJECT 宏,并直接或间接地继承自 QObject 类。

2.1.3 常用 Qt 模块及类库

Qt 由多个模块组成,每个模块都具备不同的功能,本软件系统中涉及到的模块主要有 Qt Core、Qt Gui 和 Qt Widgets 等三个,这三个模块是 Qt 中最重要的模块,几乎所有的 Qt 应用程序都必须使用这些模块。此外,Qt 中还有一些十分重要的模块如 Qt Network、Qt QML、Qt Quick、Qt SQL 等等^[33]。

(1) Qt Core 模块

Qt Core 模块主要包含 Qt 中的核心非图形类,如日期时间类 QDateTime、事件类 QEvent、文件类 QFile、列表类 QList、字符串类 QString、线程类 QThread 等等。该模块是 Qt 中的核心模块,几乎其他所有模块都依赖该模块。

(2) Qt Gui 模块

Qt Gui 模块为 Qt 编写图形用户界面(GUI)程序提供基本支持,主要包括用于窗口系统集成、事件处理、基本图像、字体和文本的类,如颜色类 QColor、字体类 QFont、图标类 QIcon、图像类 QImage、绘图类 QPainter 等等。

(3) Qt Widgets 模块

Qt Widgets 模块为 Qt 提供了各种控件及窗口,如标签类 QLabel、行编辑框类 QLineEdit、按钮类 QPushButton、下拉框类 QComboBox 等基本控件类,以及 QMainWindow、QDialog 和 QWidget 等窗口类。

2.1.4 使用 Qt 开发的优势

Qt 自上世纪 90 年代问世至今,已经被广泛的应用于各类应用程序的 GUI 开发中,经过 20 余年的发展,Qt 的功能在不断扩展的同时,其优点也日益显现。其中最被广泛认同的主要有以下几点:

(1) 开源。Qt 是开源的,因此可以通过学习源代码来更好地理解其接口函数的功能,从而更好地使用 Qt 库。此外,通过源码还能够学习到 Qt 的编程方法、编程思想等,有利于自身编程能力的提高。

(2) 优良的跨平台性。支持 Linux、OS X、Windows、VxWorks、QNX、Android、iOS、BlackBerry 和 Sailfish 等多种平台,这意味着利用 Qt 编写出来的应用程序在几乎不用修改源代码的情况下就可以在以上平台上运行^[34]。

(3) 良好的封装性和丰富的 API。Qt 是由 C++语言采用面向对象的方式编写而成,包含 250 多个 C++类,每个类都提供了大量的函数,可十分方便地设置或获取该类对象的属性,并支持与 VTK 等可视化工具包一起使用^[35]。

(4) 开发方便。Qt 对开发 GUI 常用的控件进行了封装，并提供了自定义控件的接口，能够方便地添加自定义控件，从而大幅减少了开发者的工作，降低了程序的复杂度。此外，Qt Designer 工具能够以拖拽的方式添加控件，使得开发更加直观方便。

(5) 降低开发成本。得益于 C++ 的运行效率优势，Qt 开发的应用程序在低成本的硬件上也可以高效运行，实现更短的启动时间、更快的交互响应以及更高的性能，从而大幅降低硬件成本。

(6) 详细的参考资料。Qt 发展至今已经相当成熟，拥有的专注用户群体十分庞大，其参考资料和开发文档也十分详细，既有利于初学者的快速入门，也有利于使用 Qt 进行复杂开发。

2.2 VTK 库研究

VTK (Visualization Toolkit) 是一个开源的、跨平台的、面向对象的可视化工具包，可用于计算机图形图像处理及科学可视化，支持多种可视化算法及先进的建模技术，如标量可视化、矢量可视化、隐式建模、网格平滑等等。近年来，随着计算机科学可视化技术的高速发展，VTK 也被广泛地应用到医学成像、化学、计算流体力学和有限元分析等各行各业^[36]。VTK 中有两个重要概念：可视化管线 (Visualization Pipeline) 和渲染引擎 (Rendering Engine)，可视化管线主要负责获取外部数据或创建 VTK 数据、处理已获取或创建的数据、将处理完成的数据写入外部文件或传递给渲染引擎，而渲染引擎负责将数据进行可视化。

2.2.1 可视化管线

可视化管线由数据对象 (Data Object)、过程对象 (Process Object) 和数据流三个基本要素组成，数据对象表示 VTK 中各种类型的数据，过程对象用来处理当前数据对象并产生新的数据对象，数据流是指可视化管线中数据的流动方向^[37]。

2.2.1.1 数据对象

在 VTK 中，具有常规结构的数据通常被称为数据集 (Dataset)，数据集通常由组织结构 (Organizing Structure) 和属性数据 (Attribute Data) 组成，数据集类型由组织结构决定，该组织结构又由几何结构 (Geometric Structure) 和拓扑结构 (Topological Structure) 组成，如图 2.2 所示。几何结构取决于点 (point)，当数据集中的点是规则的，该数据集的几何结构也是规则的；拓扑结构取决于单元 (cell)，当数据集中的单元拓扑关系是规则的，该数据集的拓扑结构也是规则的。

常见的 VTK 数据集类型包括图像数据 (Image Data)、直线网格 (Rectilinear Grid)、结构化网格 (Structured Grid)、非结构化网格 (Unstructured Grid) 和多边形数据 (Polygonal Data) 等^[38]。

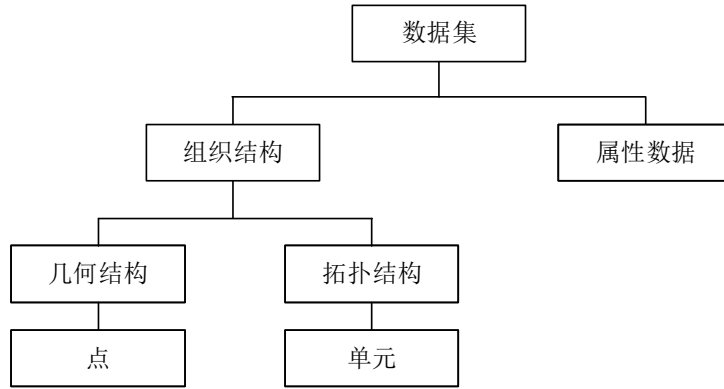


图 2.2 VTK 数据集组成

图像数据如图 2.3 所示，其几何结构和拓扑结构都是规则的，能够隐式定义，可由索引范围、原点和间距三个几何参数确定，每个点的坐标定义为： $坐标 = 原点 + 索引 \times 间距$ ，其中各参数均为长度为 3 的向量。图像数据的单元类型都是相同的，由数据集的维数确定。由于其高度规则性，图像数据相比于其他数据集所需的存储量会更少，因此其效率也更高。

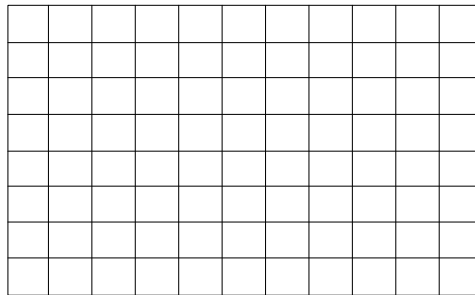


图 2.3 图像数据

直线网格数据集如图 2.4 所示，其拓扑结构是规则的，几何结构是部分规则的，因此能够隐式定义其单元，半隐式定义其点，可由索引范围和三个分别定义 x 、 y 、 z 坐标的数组确定，这种半隐式定义点的方式相比于显式定义每个点的坐标所需的存储量更小。每个点的坐标定义为： $coordinate = (array_x(i), array_y(j), array_z(k))$ ，其中 i 、 j 、 k 分别为 x 、 y 、 z 三个方向上点的索引。直线网格的单元类型也是相同的，由数据集的维数确定。

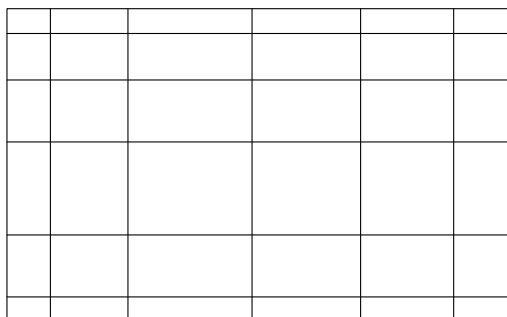


图 2.4 直线网格

结构化网格数据集如图 2.5 所示，其拓扑结构是规则的，几何结构是不规则的，因此仅能够隐式定义单元，而需要显式定义点，可由索引范围和存储每个点坐标的数组确定。每个点的坐标定义为： $coordinate = array(index)$ ，其中 $index$ 为点的索引，结构化网格的单元是四边形（2D）或六面体（3D）。

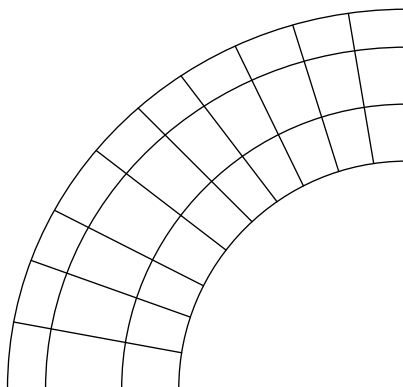


图 2.5 结构化网格

非结构化网格数据集是 VTK 中最通用的数据集，其几何结构和拓扑结构都是不规则的，需要显式存储单元和点，因此该数据集占用的内存空间也是最大的。在 VTK 中，任何类型的数据集都可以用非结构化网格数据集表示，但由于非结构化网格数据集需要更多的内存空间和计算资源，因此优先选用其他类型的数据集，必要时才选用非结构化网格数据集。非结构化网格的单元类型非常丰富，且不同的类型可存在于同一个非结构化网格中，如图 2.6 所示是一些常见的非结构化网格单元类型。

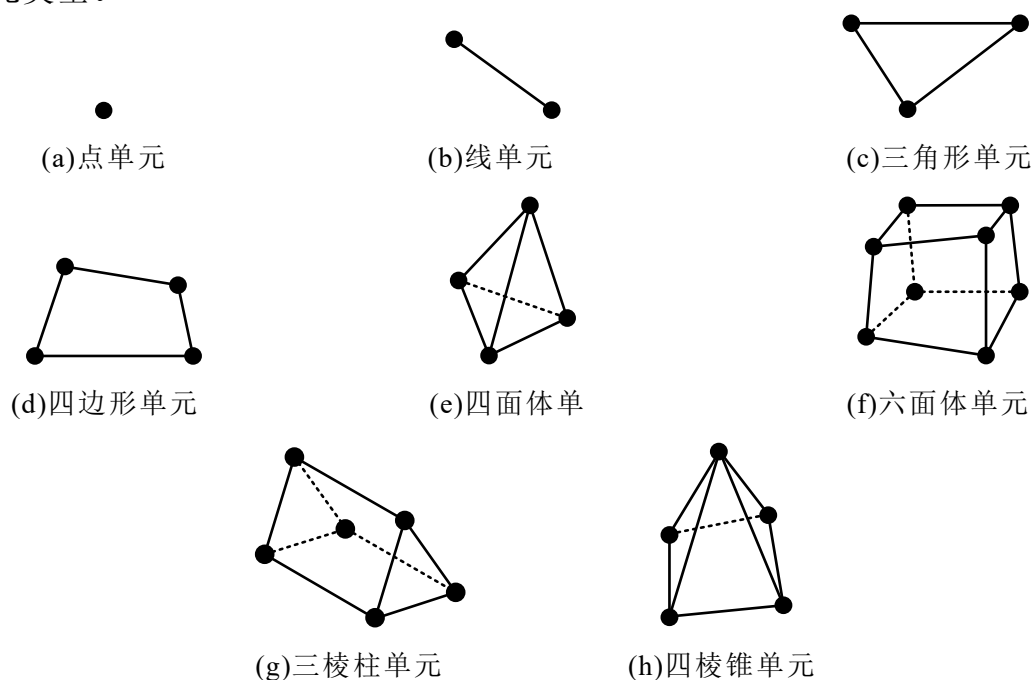


图 2.6 非结构化网格

多边形数据是可视化数据的一种重要形式，因为它是渲染引擎的几何接口，在数据、算法和高速计算机图形间起着纽带作用，几乎除图像数据之外所有的数

据集都必须先转化成多边形数据才能进行渲染^[39]。多边形数据集由顶点、多顶点、线、折线、多边形、三角形带等组成，如图 2.7 所示。

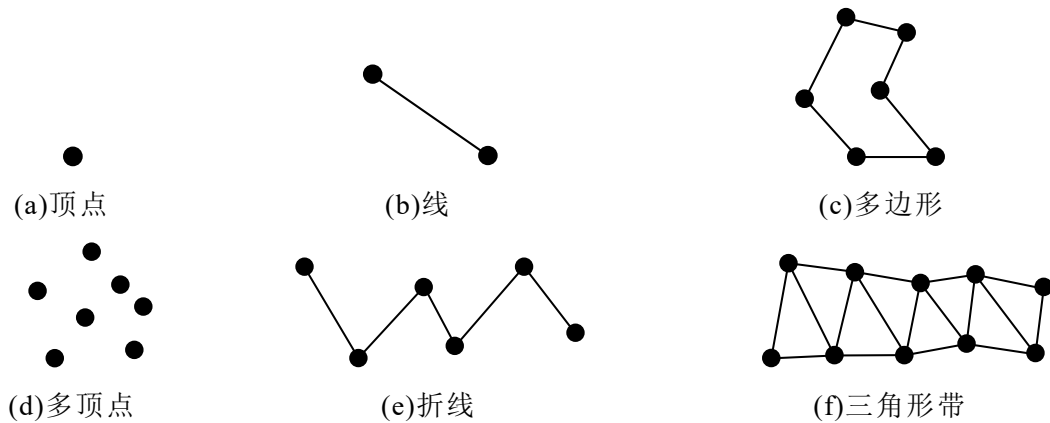


图 2.7 多边形数据

属性数据是与数据集组织结构相关的数据，大多数情况下，属性数据都是与数据集的点或单元关联，如点的温度或位移、单元的质量等属性，但有时候也可以与单元边或单元面关联，如流经单元表面的热通量等属性。常见的属性数据包括标量数据、矢量数据和张量数据等，如图 2.8 所示。标量数据是最简单最常见的属性数据，只有大小没有方向，它只有一个分量，常用来表示温度、压力、密度等属性；矢量数据既有大小也有方向，它有三个分量，常用来表示速度、位移等属性；张量是标量、向量和矩阵的数学概括，第零阶张量为标量，第一阶张量为向量，第二阶张量为矩阵。

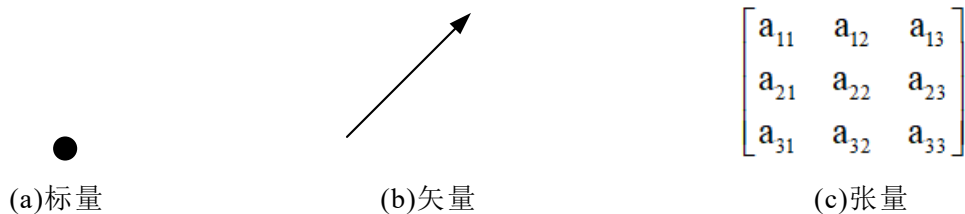


图 2.8 常见的属性数据

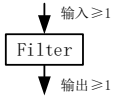

2.2.1.2 过程对象

一般来说，过程对象具有输入端口和输出端口，通过输入端口获取数据进行处理，通过输出端口将处理的数据输出。根据是否有输入端口和输出端口，可将过程对象分成源（Source）、过滤器（Filter）和映射器（Mapper）三种类型^[40]，如表 2.1 所示。

表 2.1 过程对象分类

序号	输入端口	输出端口	名称	图例
1	无	有	源（Source）	<div style="text-align: center;"> <small>无输入</small> <small>输出 ≥ 1</small> </div>

续表

序号	输入端口	输出端口	名称	图例
2	有	有	过滤器 (Filter)	
3	有	无	映射器 (Mapper)	

过程对象实际上是将处理数据的算法封装成的一个类，通过类的成员函数对算法的参数、属性等进行设置，从而控制过程对象对数据对象的处理。下面对一些常用的过程对象进行介绍。

(1) `vtkUnstructuredGridReader` 过程对象是一个源对象，因此其没有数据输入，仅有数据输出，输出的数据对象为非结构化网格数据 `vtkUnstructuredGrid`。该源过程对象是用来读取 `vtk` 格式的非结构化网格数据集文件，其用法如框 2.1 所示。

框 2.1 `vtkUnstructuredGridReader` 部分用法的代码

```
vtkSmartPointer <vtkUnstructuredGridReader> reader =  
    vtkSmartPointer <vtkUnstructuredGridReader>::New();  
reader -> SetFileName("test.vtk"); //设置文件名  
reader -> Update(); //更新过程对象，即开始读取文件
```

(2) `vtkGeometryFilter` 过程对象是一个过滤器对象，其输入可为任何类型的数据，输出为多边形数据。该过滤器过程对象可用来从任何类型的数据集中提取几何图形或将数据转化成多边形数据集类型，框 2.2 中的代码演示了将非结构化网格数据转化成多边形数据的方法。

框 2.2 非结构化网格数据转化为多边形数据的代码

```
vtkSmartPointer <vtkUnstructuredGrid> unstructuredGrid =  
    vtkSmartPointer <vtkUnstructuredGrid>::New();  
vtkSmartPointer <vtkGeometryFilter> geometryFilter =  
    vtkSmartPointer <vtkGeometryFilter>::New();  
geometryFilter -> SetInputData(unstructuredGrid); //输入非结构化网格数据  
geometryFilter -> Update(); //更新过程对象  
vtkPolyData* polydata = geometryFilter -> GetOutput(); //输出多边形数据
```

(3) `vtkAppendFilter` 过程对象是一个过滤器对象，其输入可为任何类型的数据，输出为非结构化网格数据。该过滤器过程对象可将多个数据集附加到非结构

化网格数据集中，因此可用其将其他类型的数据集转化成非结构化网格数据集，框 2.3 中的代码演示了将多边形数据集转化成非结构化网格数据集的方法。

框 2.3 多边形数据集转化成非结构化网格数据集的代码

```
vtkSmartPointer<vtkPolyData> polyData =
    vtkSmartPointer<vtkPolyData>::New();
vtkSmartPointer<vtkAppendFilter> appendFilter =
    vtkSmartPointer<vtkAppendFilter>::New();
appendFilter -> AddInputData(polyData); //输入多边形数据
appendFilter -> Update(); //更新过程对象
vtkSmartPointer<vtkUnstructuredGrid> unstructuredGrid =
    vtkSmartPointer<vtkUnstructuredGrid>::New();
unstructuredGrid -> ShallowCopy(appendFilter -> GetOutput()); //输出非结构化网格数据
```

2.2.1.3 数据流

过程对象在读取数据对象时，通常使用 `SetInputData()` 函数或 `SetInputConnection()` 函数作为该过程对象输入数据的端口，使用 `GetOutput()` 函数或 `GetOutputPort()` 函数作为该过程对象输出数据的端口。然而数据输入和输出的过程中，需要对数据对象的类型进行识别，使用与之匹配的输入端口读取数据对象，若该过程对象端口与数据对象类型不匹配，则无法读取，如图 2.9 所示，情况①和②可以成功读取，情况③无法读取。

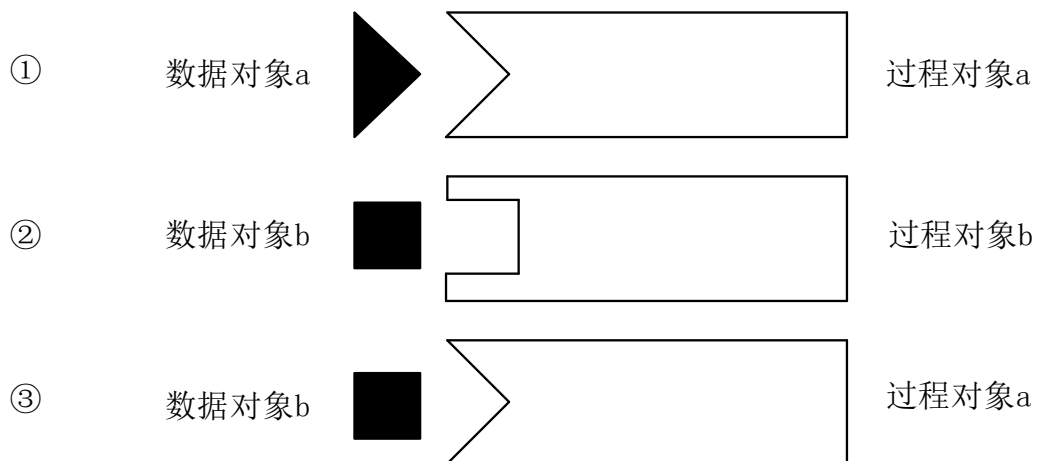


图 2.9 数据匹配

2.2.2 渲染引擎

渲染引擎主要负责将可视化管线中输出的数据在窗口中进行渲染显示，如图 2.10 所示，主要包括映射器、数据属性、数据支撑、相机、灯光、渲染器、渲染窗口和渲染窗口交互器几个部分。

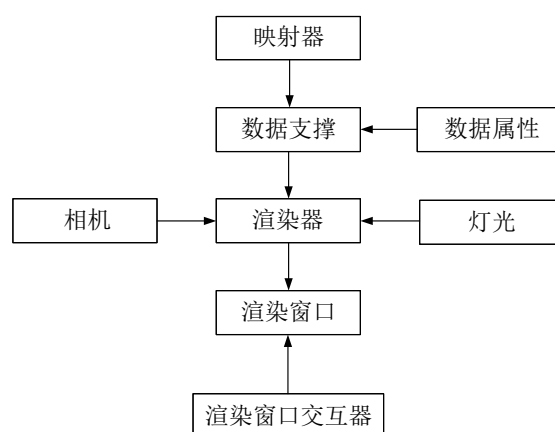


图 2.10 渲染引擎

（1）映射器（vtkMapper）

读取或创建的数据经过过滤器处理后传入到映射器中，映射器负责将数据映射成几何图元，常用的映射器有 `vtkPolyDataMapper` 和 `vtkDataSetMapper` 等。`vtkDataSetMapper` 映射器将数据集映射到几何图元，在映射非多边形数据集时，需要先将其转化成多边形数据集（点、线和多边形），然后才能映射到图形系统。`vtkPolyDataMapper` 是专门针对多边形数据集的映射器，能够将多边形数据直接映射到图形系统。显然，`vtkPolyDataMapper` 的映射速度要高于 `vtkDataSetMapper`，对于多边形数据集，应优先选用 `vtkPolyDataMapper` 映射器。映射器常用的功能如框 2.4 所示。

框 2.4 映射器常用功能的代码

```

vtkSmartPointer <vtkDataSetMapper> mapper =
    vtkSmartPointer <vtkDataSetMapper>::New();
mapper -> SetInputData(); //指定需要映射的数据
mapper -> SetScalarVisibility(); //控制是否将标量数据映射到 vtkActor 上
mapper -> SetScalarRange(); //设置标量值的映射范围，用于云图显示
mapper -> SetScalarMode(); //控制标量数据映射到点上还是单元上
  
```

（2）数据属性（vtkProperty）

数据在窗口中渲染时，可设置其渲染属性，如颜色、透明度等等。当不进行显式设置时，VTK 自动使用默认值。数据属性常用的功能如框 2.5 所示。

框 2.5 数据属性常用功能的代码

```

vtkSmartPointer <vtkProperty> properties =
    vtkSmartPointer <vtkProperty>::New();
properties -> SetColor(); //设置数据对象的颜色
properties -> SetOpacity(); //设置数据对象的透明度
properties -> SetRepresentation(); //设置数据对象的表示形式
  
```

```
properties -> SetEdgeVisibility(); //控制数据对象是否显示边
properties -> SetLineWidth(); //设置线宽
```

(3) 数据支撑 (vtkActor)

数据在场景中渲染时，无法直接将数据添加到渲染场景中，还需要借助一个支撑物，即先将数据映射到该支撑物上，然后添加到渲染场景中，通常使用 `vtkActor` 作为数据支撑，来表示渲染场景中数据对象的几何形状和属性。数据支撑的定义与使用如框 2.6 所示。

框 2.6 数据支撑定义与使用的代码

```
vtkSmartPointer<vtkActor> actor =
    vtkSmartPointer<vtkActor>::New();
actor -> SetMapper(); //设置映射器
actor -> SetProperty(); //设置数据对象属性
```

(4) 相机 (vtkCamera)

相机是渲染场景中必不可少的因素之一，当不进行显式定义时，VTK 会自动创建一个相机对象并使用默认参数，此时可使用 `GetActiveCamera()` 函数获取到场景中的相机。如图 2.11 所示，相机具有位置、焦点、向上方向、视角和前后裁剪平面等参数，从相机位置到相机焦点的方向即为相机的投影方向，前后裁剪平面与投影方向垂直，只有在前后裁剪平面之间的数据对象才能够显示在场景中。相机的定义与使用如框 2.7 所示。

框 2.7 相机定义与使用的代码

```
vtkSmartPointer<vtkCamera> camera =
    vtkSmartPointer<vtkCamera>::New();
camera -> SetPosition(); //设置相机位置
camera -> SetFocalPoint(); //设置相机焦点
camera -> SetViewUp(); //设置相机向上方向
camera -> SetViewAngle(); //设置相机视角
camera -> SetClippingRange(); //设置相机前后裁剪平面
```

(5) 灯光 (vtkLight)

灯光是渲染场景中十分重要的因素，关乎到场景中模型的显示效果，其常用的参数包括开关、位置、焦点、颜色和亮度等，从灯光位置到灯光焦点的方向即为光线的方向。默认情况下，VTK 会定义一个光源位于无限远的平行灯光。灯光的定义与使用如框 2.8 所示。

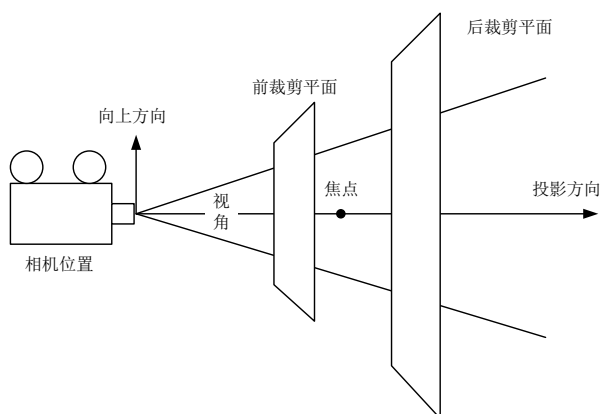


图 2.11 相机示意图

框 2.8 灯光定义与使用的代码

```
vtkSmartPointer<vtkLight> light =
    vtkSmartPointer<vtkLight>::New();
light -> SwitchOn(); //打开灯光
light -> SetPosition(); //设置灯光位置
light -> SetFocalPoint(); //设置灯光焦点
light -> SetColor(); //设置灯光颜色
light -> SetIntensity(); //设置灯光亮度
```

(6) 渲染器 (vtkRenderer) 和渲染窗口 (vtkRenderWindow)

渲染器负责管理场景的渲染过程，将数据对象、相机、灯光等因素转换成图像显示在屏幕上，渲染器需要放在渲染窗口中进行显示，同一个渲染窗口中可放入多个渲染器。渲染器和渲染窗口的定义与使用如框 2.9 所示。

框 2.9 渲染器和渲染窗口定义与使用的代码

```
vtkSmartPointer<vtkRenderer> renderer =
    vtkSmartPointer<vtkRenderer>::New();
renderer -> AddActor(); //添加数据对象
renderer -> SetBackground(); //设置渲染场景背景颜色
vtkSmartPointer<vtkRenderWindow> renWin =
    vtkSmartPointer<vtkRenderWindow>::New();
renWin -> AddRenderer(renderer); //将渲染器放入渲染窗口中
```

(7) 渲染窗口交互器 (vtkRenderWindowInteractor)

渲染窗口交互器为鼠标、键盘和时间等用户事件提供了独立于平台的交互机制。如图 2.12 所示，当某平台发生特定事件时，继承自 vtkRenderWindowInteractor 的平台相关子类（如 Windows32 平台的 vtkWin32RenderWindowInteractor 类）将该消息翻译成独立于平台的 VTK 事件，再将这些 VTK 事件发送给事件观察者

vtkInteractorObserver 中，观察者将事件传递到 vtkInteractorStyle 或其子类的事件响应函数中来执行该事件，从而完成事件的响应。其定义和使用如框 2.10 所示。

框 2.10 渲染窗口交互器定义和使用的代码

```
vtkSmartPointer < vtkRenderWindowInteractor > renWinInteractor =
    vtkSmartPointer < vtkRenderWindowInteractor >::New();
renWinInteractor -> SetRenderWindow(); //将交互器与渲染窗口绑定
```

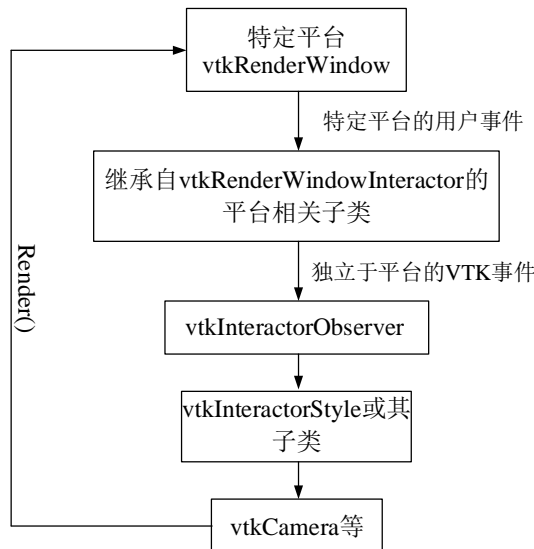


图 2.12 VTK 事件响应过程

2.2.3 使用 VTK 开发的优势

VTK 是以 OpenGL 函数库为基础进行封装的，除了继承 OpenGL 优良的可视化功能外，还衍生出了许多优于 OpenGL 的功能。VTK 所具备的优势主要有以下几点：

(1) 开源。VTK 是一个免费的开源工具包，其所有的代码均可见，因此可直接通过源码来了解某个接口函数的作用与用法，这对于深入学习 VTK 有很大的帮助。

(2) 跨平台。VTK 具有优良的跨平台性，能够在 Windows、Linux、Mac、网页及手机端等平台上运行，极大地提高了代码的复用性和开发效率，降低了开发成本。

(3) 高效。VTK 的核心功能是由 C++ 语言编写而成，能够最大限度地提高运行效率。同时还具备解释包装层，以支持 Java、Tcl 和 Python 等解释型语言，便于快速开发^[41]。

(4) 灵活的可视化管线。VTK 提供了独特的可视化管线机制进行数据可视化，这种机制十分简便，可灵活地对各种类型数据进行筛选过滤，从而得到想要的数据集。

(5) 丰富的单元类型。VTK 提供了多种形状的线性和非线性单元，如常用的三角形、四边形、四面体、六面体、三棱柱、四棱锥等等，极大地方便了有限元网格模型的建立。

(6) 丰富的算法。VTK 提供了大量的可视化算法，如标量值和矢量值的映射、平面切割、等值面提取等等，这些算法极大地提升了 VTK 科学可视化的能力，同时也更加简化了后处理过程。

2.3 Qt 与 VTK 的整合

VTK 程序既可以以控制台窗口的形式独立显示(渲染窗口 `vtkRenderWindow`)，也可以与其他图形用户界面框架整合，以应用程序界面的形式显示，如使用 `QVTKWidget` 类可将 VTK 窗口整合到 Qt 的界面中。下面用一个简单的例子来演示 Qt 与 VTK 的整合，并演示 VTK 的可视化管线及渲染引擎，主要代码如下：

(1) 创建数据源。VTK 中提供了大量的数据源，如圆柱、球体、圆锥、六面体等等，并可设置数据源的各参数。本例创建了一个圆柱数据源 `vtkCylinderSource`，并使用 `SetResolution()` 函数设置其圆的边数，使用 `SetRadius()` 函数设置圆柱的半径，使用 `SetHeight()` 函数设置圆柱的高，主要代码如框 2.11 所示。

框 2.11 创建圆柱数据源的代码

```
vtkSmartPointer<vtkCylinderSource> cylinder =  
    vtkSmartPointer<vtkCylinderSource>::New();  
cylinder -> SetResolution(32);  
cylinder -> SetRadius(1.0);  
cylinder -> SetHeight(2.0);  
cylinder -> Update();
```

(2) 添加过滤器。`vtkFeatureEdges` 过滤器能够提取多边形数据集的各种边界，如边界边、特征边、流形边和非流形边等，现使用 `vtkFeatureEdges` 过滤器提取该圆柱数据源的特征边，主要代码如框 2.12 所示。

框 2.12 提取圆柱特征边的代码

```
vtkSmartPointer<vtkFeatureEdges> featureEdges =  
    vtkSmartPointer<vtkFeatureEdges>::New();  
featureEdges -> SetInputData(cylinder -> GetOutput());  
featureEdges -> FeatureEdgesOn();  
featureEdges -> Update();
```

(3) 渲染引擎。圆柱数据源和经过 `vtkFeatureEdges` 过滤器提取出的圆柱特征边数据，都需要使用 `vtkPolyDataMapper` 映射器来分别将其映射到 `vtkActor` 上，然后添加到渲染器 `vtkRenderer` 中进行渲染并放置到渲染器窗口中，才能显示到 VTK 的控制台窗口中，主要代码如框 2.13 所示，在 VTK 控制台窗口中显示的结果如图 2.13 所示。

框 2.13 渲染过程的主要代码

```
vtkSmartPointer<vtkPolyDataMapper> cylinderMapper =
    vtkSmartPointer<vtkPolyDataMapper>::New();
cylinderMapper -> SetInputData(cylinder -> GetOutput());
vtkSmartPointer<vtkActor> cylinderActor =
    vtkSmartPointer<vtkActor>::New();
cylinderActor -> SetMapper(cylinderMapper);
cylinderActor -> GetProperty() -> SetColor(0.0,1.0,0.0);
vtkSmartPointer<vtkPolyDataMapper> edgeMapper =
    vtkSmartPointer<vtkPolyDataMapper>::New();
edgeMapper -> SetInputConnection(featureEdges -> GetOutputPort());
vtkSmartPointer<vtkActor> edgeActor =
    vtkSmartPointer<vtkActor>::New();
edgeActor -> SetMapper(edgeMapper);
edgeActor -> GetProperty() -> SetColor(1.0,0.0,0.0);
edgeActor -> GetProperty() -> SetLineWidth(3.0);
vtkSmartPointer<vtkRenderer> renderer =
    vtkSmartPointer<vtkRenderer>::New();
renderer -> AddActor(cylinderActor);
renderer -> AddActor(edgeActor);
renderer -> SetBackground(0.3, 0.4, 0.5);
vtkSmartPointer<vtkRenderWindow> renWin =
    vtkSmartPointer<vtkRenderWindow>::New();
renWin -> AddRenderer(renderer);
```

(4) Qt 与 VTK 的整合。`QVTKWidget` 类继承自 Qt 的 `QWidget` 类，因此实际上也是一个窗口类控件，可看做 VTK 与 Qt 的接口，主要用来在 Qt 的窗口中显示 VTK 渲染器窗口。使用 `QVTKWidget` 的 `SetRenderWindow()` 函数可将 VTK 渲染器窗口添加到其中，然后将 `QVTKWidget` 放置到 Qt 的窗口中即可，主要代码如框 2.14 所示，将 VTK 视图窗口添加到 Qt 中的结果如图 2.14 所示。

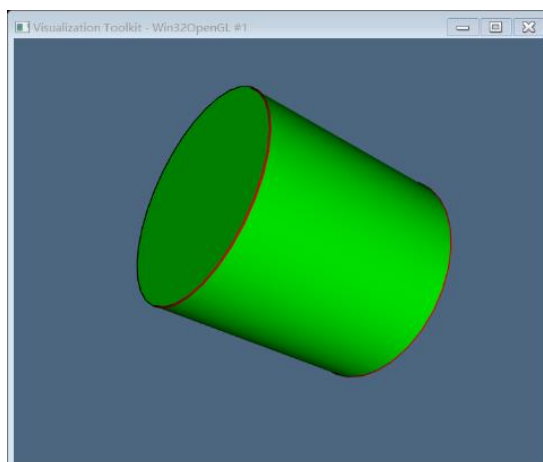


图 2.13 VTK 控制台窗口

框 2.14 Qt 与 VTK 整合的代码

```
QVTKWidget *vtkWidget = new QVTKWidget(this);
vtkWidget -> SetRenderWindow(renWin); //将 VTK 渲染窗口放置于 QVTKWidget 中
vtkWidget -> update(); //更新 VTK 渲染窗口
ui.centralWidgetLayout -> addWidget(vtkWidget); //将 QVTKWidget 放置于 Qt 界面中
```

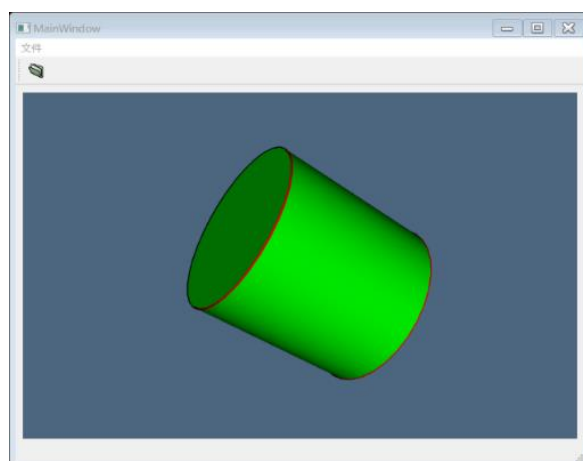


图 2.14 VTK 与 Qt 整合

2.4 本章小结

本章主要对软件系统开发中用到的 Qt 和 VTK 库进行了深入研究，为下一章介绍软件系统的开发过程奠定基础。首先，研究了 Qt 库的信号与槽机制、元对象编译器和常用的 Qt 模块等内容，并分析了使用 Qt 进行软件系统开发的优势所在；然后，研究了 VTK 库中两个重要的内容——可视化管线和渲染引擎，同时对 VTK 工具包的优势进行了简单的分析；最后，用一个简单的实例演示了 Qt 与 VTK 的整合方法。

第3章 软件系统设计与实现

在深入研究了开发过程中用到的 Qt 和 VTK 库之后,本章主要介绍基于这两个外部库开发的软件系统的设计与实现过程,包括软件设计目标、软件框架设计、软件功能设计以及部分主要功能的实现等内容。

3.1 软件系统设计目标

前后处理系统的设计直接影响到整个 CAE 软件的使用性能,不仅关乎到用户对软件的体验感,甚至还影响着软件对潜在用户的吸引力。因此,设计高性能的前后处理系统至关重要,通常,一个完整的 CAE 软件前后处理系统应该实现以下几点功能:

(1) 具有简洁美观、交互友好的界面。界面的设计应该遵循交互一致性、面向多层次用户、隐藏复杂功能、提供反馈和出错恢复机制等重要设计原则,能够使用户提高学习速度和使用效率,降低操作失误率,减轻用户的记忆负担。同时,应确保软件不会因用户的操作失误或其他原因而出现崩溃情况,造成数据遗失。

(2) 支持多种类型文件模型的导入导出。一般来说,在整个 CAE 分析过程中,模型有多种类型,如几何模型、网格模型、前处理模型以及后处理模型等,这些模型出现在分析过程中的不同阶段,其各自也具有不同的属性,导入模型的类型不同,整个分析流程也略有不同,因此 CAE 软件应具备导入各种类型模型的功能。此外,为提高工程分析效率,CAE 软件应具备与其他行业主流软件进行信息交互的接口,能够兼容这些行业主流软件的文件格式。

(3) 具有强大的交互功能。CAE 软件前后处理系统的交互功能主要包括视图操作交互和模型拾取交互,视图操作交互除基本的平移、旋转、缩放等功能外,还应具备一些其他常用功能,如框选放大、适应窗口、平面旋转等。模型拾取交互关乎到模型约束的定义,要求具有多种拾取模式和拾取类型,如单选、多选和框选等拾取模式,几何面、几何边、几何顶点、网格单元、网格面、网格边、网格节点等拾取类型,并能够根据视图中当前模型的类型自动对拾取类型进行筛选和约束。

(4) 具备完善的前处理功能。从工程分析的角度看,一个完整的分析过程应包括模型导入、网格生成、材料定义、截面属性定义、分析步定义、输出变量定义、载荷定义、边界条件定义和单元属性定义等内容,这些内容的定义直接影响到最终的计算结果,也是整个 CAE 软件开发过程中最为复杂的部分之一。对于不同的 CAE 软件,前处理内容可能略有不同,但都包含导入几何模型、生成网格、模型约束定义等模块。

(5)具备必要的后处理功能。后处理过程是对计算得到的大量数据进行处理，然后以各种图或表的形式进行显示，从而使分析人员对计算结果有一个更直观的认识，如常用的云图、切面图、切割图、等值面（线）图、变形图、动画、二维曲线图等方式。当然，还应具备查询或提取原始结果数据的功能，以便对部分计算结果进行更为精确的分析。

3.2 软件系统框架设计

CAE 软件功能复杂，模块众多，数据量大，因此搭建一个具有可重用性、高安全性、易扩展性的软件框架不仅使得开发工作得心应手，也能够让后续的功能扩展及软件维护更加容易。本软件系统采取“一总多分”的框架模式，将整个系统分成多个模块，以主界面 **MainWindow** 模块为主，其他模块依赖于主模块，若有需要，各子模块之间也会相互依赖，如图 3.1 所示。当要对系统进行功能扩展时，可添加一个依赖于主模块的模块用来实现扩展功能，这样就避免了对其他模块的修改。本框架的现有模块中，**MainWindow** 模块聚合了主界面类及各控制面板类，是整个软件的信息中心，负责软件界面信息的收集与分发；**File** 模块聚合了软件中读、写文件的类，负责各种格式文件的读写操作，如网格文件、计算文件、结果文件等等；**Data** 模块聚合了软件中的各种数据类，负责大量数据的存储与管理，如网格数据、材料数据、模型约束数据等等；**View** 模块聚合了界面中的视图类，负责视图场景中各部件、模型，以及视图交互的管理；**Model** 模块聚合了各种模型类，负责各种模型的显示与属性定义，如几何模型、网格模型、高亮模型等等；**InteractorStyle** 模块聚合了各种视图交互样式类，包括视图操作样式和模型拾取样式，负责各样式的定义工作；**Mesh** 模块和 **Solver** 模块分别负责网格生成和计算求解，不属于界面开发中的内容。

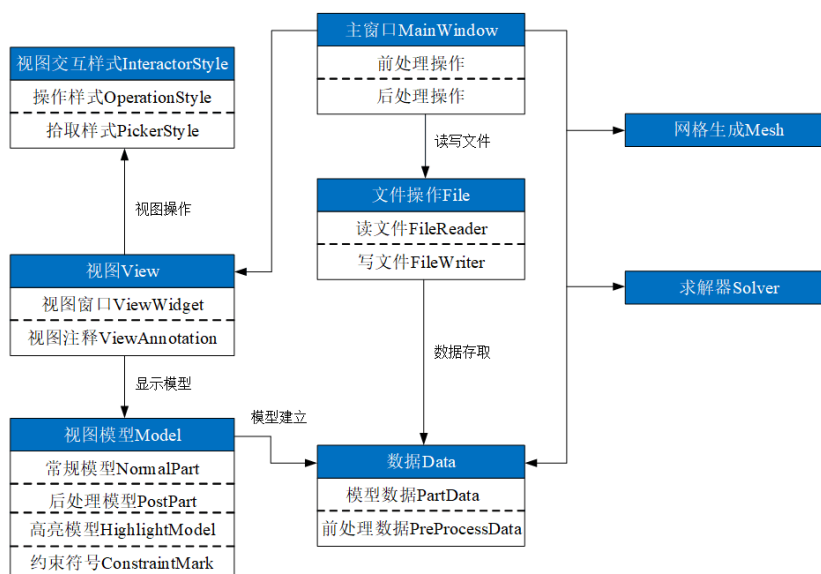


图 3.1 软件系统框架图

3.3 软件系统功能设计

3.3.1 文件读取功能设计

根据模型文件中包含的数据类型不同,可将模型文件大致分为几何模型文件、网格模型文件、前处理模型文件和后处理模型文件四类。

(1) 几何模型文件

几何模型文件仅包含模型的几何信息,常见的有 `step/iges` 等文件类型。几何模型导入进来后要经过网格剖分、定义材料和约束等过程,然后才能进行计算和进入后处理模块,如图 3.2 所示。

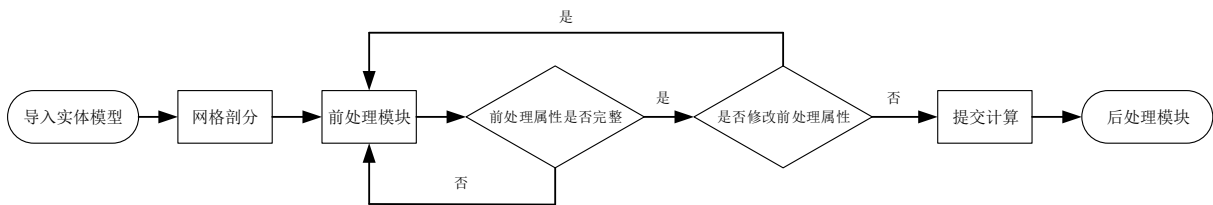


图 3.2 读取实体模型文件处理流程

(2) 网格模型文件

网格模型文件仅包含模型的网格信息,即节点和单元信息。该类模型导入进来后不需要进行网格剖分,但需要定义材料和约束后,才能进行计算和进入后处理模块,如图 3.3 所示。

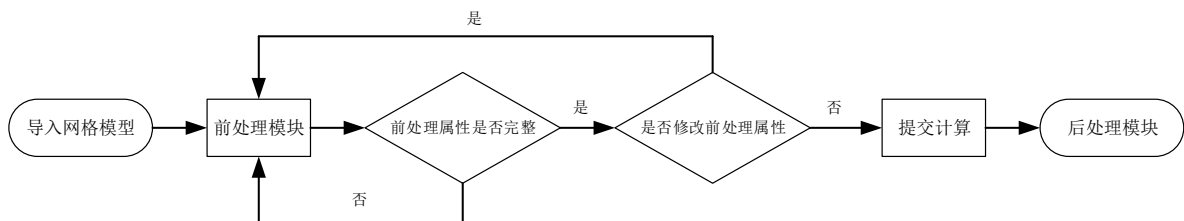


图 3.3 读取网格模型文件处理流程

(3) 前处理模型文件

前处理模型文件不仅包含模型的网格信息,还包含材料和约束等仿真参数信息,如 Abaqus 的 `inp` 文件。若文件中仿真参数定义完整,则可以直接进行计算,否则还需要在界面中完善缺少的仿真参数性,如图 3.4 所示。

(4) 后处理模型文件

后处理模型文件包含模型的网格信息和计算完成的结果数据,常见的有 `plt`、`vtk` 等。该类文件是计算完成后的结果文件,可直接进行云图等后处理可视化操作。

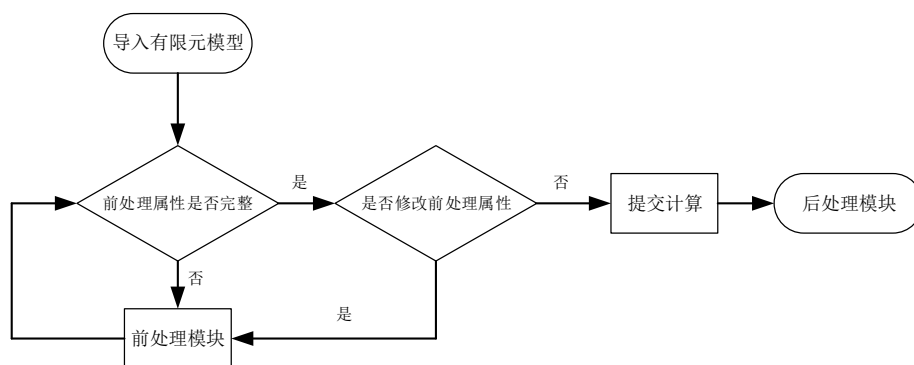


图 3.4 读取前处理模型文件处理流程

3.3.2 视图功能设计

视图部分是整个 CAE 软件界面的核心，其功能主要包括显示窗口部件及模型、视图操作和模型拾取三大部分。视图除了要能够显示各种类型的模型，如实体模型、网格模型和后处理模型等，以及模型的约束信息，如载荷和边界条件等，还要求具备显示文字注释等功能，以辅助说明视图场景中的相关信息，如当前缩放比例、动画帧数及时间等。除了显示功能外，视图的交互功能也极其重要，其交互包含两个部分，一是视图的常规操作交互，二是对视图中模型的拾取交互。前者应包含视图投影、平移、旋转、缩放、框选放大、适应窗口和平面旋转等交互功能，使分析人员能够以不同的视角和比例查看视图中的模型信息；后者关系到前处理过程中边界条件、载荷等约束的定义，要求能够提供单选、多选、框选等多种拾取模式，并针对不同类型的模型拾取不同的模型特征，如对于几何模型而言，应能够拾取几何面、几何边和几何顶点等几何特征，对于网格模型而言，应能够拾取网格单元、网格面、网格边和网格节点等网格特征。

3.3.3 前处理模块功能设计

前处理模块是建立有限元模型的过程，该过程将几何模型进行网格剖分生成网格模型，并定义约束用来计算，主要包括网格生成、材料定义、截面属性定义、分析步定义、输出变量定义、载荷定义、边界条件定义和单元定义等内容。

(1) 网格生成

网格自动生成是一个很复杂的过程，也是限制当前国产 CAE 软件发展的一个重要因素。网格的密度和质量影响着计算的精度和效率，是整个 CAE 分析过程中至关重要的一环。本前后处理软件系统使用外部网格生成器进行网格剖分，能够生成三角形网格和四面体网格，并能够设置最大半径-边缘比约束、最小二面角约束、网格尺寸和最大单元体积等参数，以控制生成网格的密度。

(2) 定义材料

CAE 分析中的材料类型十分丰富，不同领域的问题，材料类型也不尽相同。由于本前后处理软件系统是针对静力学问题开发的，因此目前仅提供了线弹性材

料、弹塑性材料和脆性材料等材料类型，通过定义材料名、选择材料类型和设置材料值三个参数来定义模型的材料属性。

（3）定义截面属性

定义好的材料不能直接赋予到网格或几何实体上，而是先在截面属性中定义添加材料特性，然后再为零件模型赋予相应的截面属性，这样做的好处是当重新生成网格或进行网格编辑时，不必重新定义材料参数。本前后处理软件系统通过定义截面属性名、截面属性种类及其值、截面属性赋予的零件或集合等参数来定义截面属性并将其赋予到模型上。

（4）定义分析步

通常，一个问题中的工况是变化的，因此需要将整个分析过程细化成多个分析步，每个分析步包含不同的分析过程和状态，如变化的载荷和边界条件、部件间的相互作用等。此外，还可以定义每个分析步的输出变量，以得到中间过程的计算结果。本前后处理软件系统通过定义分析步名、分析步类型、分析步顺序及各类型分析步的具体值等参数来定义分析步属性。

（5）定义输出变量

同一个问题经过计算后能得到许多不同的属性结果，如位移、应力、应变及其各自分量等，计算的属性越多，所需的计算资源也就越多，计算时间也就越长。因此，可通过自定义输出变量和分析步来灵活地控制变量的输出方式，即以什么样的频率输出哪些变量，从而节约计算资源和时间。本前后处理软件系统通过定义输出变量名、输出变量所属分析步、输出变量作用域、变量输出的频率以及要输出的变量等参数来定义输出变量属性。

（6）定义载荷和边界条件

载荷和边界条件的定义与分析步有关，需指定哪些载荷或边界条件在哪些分析步中起作用。常见的载荷类型有节点载荷、体力载荷等，常见的边界条件类型有位移/转角、速度/角速度和加速度/角加速度等。本前后处理软件系统通过定义载荷或边界条件名、施加位置、所属分析步、类型及具体值等参数来定义载荷或边界条件属性。

（7）定义单元

单元是进行有限元分析计算的主要因素之一，其类型十分丰富，每种类型的单元都有其优缺点和适用条件，不同类型单元的计算精度和计算效率也有较大差异，因此要根据当前问题的分析条件，选择最合适的单元类型进行计算，用尽量短的计算时间得到尽量精确的计算结果。本前后处理软件系统通过定义单元所属截面、单元阶次、单元族以及单元的具体值和单元算法等参数来定义网格模型的单元属性。

3.4 软件主要功能实现

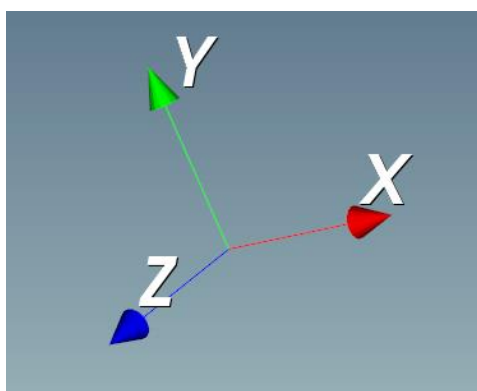
3.4.1 视图部件

(1) 坐标系

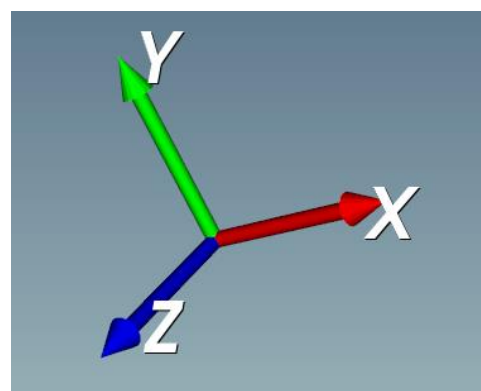
在渲染视图窗口中，存在一个全局坐标系，以辅助显示当前模型在视图中的方位。VTK 中，可使用 `vtkAxesActor` 表示坐标系，坐标系的轴有线型和圆柱型两种样式，分别使用 `SetShaftTypeToLine()` 和 `SetShaftTypeToCylinder()` 函数进行设置，如图 3.5 所示。为使坐标系能够在视图中某一固定位置显示，不因视图中的旋转、平移等操作而产生位置上的移动，通常需将其放置于方向标记窗口 `vtkOrientationMarkerWidget` 中，并通过控制该窗口位置、大小、是否交互、是否启用等属性来控制视图中的坐标系。此外，通过指定当前渲染窗口的交互器 `vtkRenderWindowInteractor` 可将该标记窗口添加到视图中，其主要代码如框 3.1 所示。

框 3.1 定义坐标系的主要代码

```
vtkSmartPointer<vtkAxesActor> axesActor =
    vtkSmartPointer<vtkAxesActor>::New();
axesActor->SetShaftTypeToLine();
vtkSmartPointer< vtkOrientationMarkerWidget > axesWidget =
    vtkSmartPointer<vtkOrientationMarkerWidget>::New();
axesWidget -> SetViewport(0, 0, 0.1, 0.1);
axesWidget -> SetOrientationMarker(axesActor);
axesWidget -> SetInteractor(this->GetRenderWindow()->GetInteractor());
axesWidget -> On();
axesWidget -> InteractiveOff();
```



(a)线型轴坐标系



(b)圆柱型轴坐标系

图 3.5 不同样式的坐标系

（2）文本注释

视图中除了显示模型及其相关云图外，有时还需要显示一些起辅助作用的文本注释，如视图当前缩放比例、动画总帧数和当前帧数等等。在 VTK 中，可使用 `vtkTextActor` 类表示文本注释，它通常与文本窗口 `vtkTextWidget` 和文本表达 `vtkTextRepresentation` 结合使用，通过 `vtkTextActor` 类设置文本内容、颜色、大小等属性，通过 `vtkTextRepresentation` 类设置文本位置、大小、边框显隐等属性，然后通过 `vtkTextWidget` 类将二者结合起来，并放置于视图中。文本注释的主要代码如框 3.2 所示。

框 3.2 文本注释的主要代码

```
vtkSmartPointer<vtkTextActor> textActor =
    vtkSmartPointer<vtkTextActor>::New();
textActor->SetInput(); //设置文本内容
textActor->GetTextProperty()->SetColor(); //设置文本颜色
vtkSmartPointer<vtkTextRepresentation> textRepresentation =
    vtkSmartPointer<vtkTextRepresentation>::New();
textRepresentation->GetPositionCoordinate()->SetValue(); //设置文本位置 1
textRepresentation->GetPosition2Coordinate()->SetValue(); //设置文本位置 2
vtkSmartPointer<vtkTextWidget> textWidget =
    vtkSmartPointer<vtkTextWidget>::New();
textWidget->SetInteractor(); //设置文本窗口交互器
textWidget->SetTextActor(textActor); //设置文本对象
textWidget->SetRepresentation( textRepresentation ); //设置文本表示
```

（3）颜色条

颜色条是在显示云图时用来指示颜色与标量数据之间对应关系的标尺，如图 3.6 所示，VTK 中使用 `vtkScalarBarActor` 类表示颜色条，与文本注释类似，它通常与颜色条窗口 `vtkScalarBarWidget` 和颜色条表示 `vtkScalarBarRepresentation` 结合使用，通过 `vtkScalarBarActor` 类可设置颜色条本身属性，如标题、标签数、标签格式、方向、透明度、文字属性等等，通过 `vtkScalarBarRepresentation` 类可设置颜色条位置、大小、边框显隐等属性，然后通过 `vtkScalarBarWidget` 类将二者结合起来，并放置于视图中。使用时，必须将颜色查找表 `vtkLookupTable` 与颜色条 `vtkScalarBarActor` 关联才可以进行颜色映射。定义颜色条的主要代码如框 3.3 所示。

框 3.3 定义颜色条的主要代码

```

vtkSmartPointer <vtkScalarBarActor> barActor =
    vtkSmartPointer <vtkScalarBarActor>::New();
barActor -> SetLookupTable(); //设置颜色查找表
barActor -> SetTitle(); //设置颜色条标题
barActor -> SetNumberOfLabels(); //设置颜色条标签个数
vtkSmartPointer <vtkScalarBarRepresentation> barRepresentation =
    vtkSmartPointer <vtkScalarBarRepresentation>::New();
barRepresentation -> SetPosition(); //设置颜色条左下角位置
barRepresentation -> SetPosition2(); //设置颜色条右上角相对于左下角位置
barRepresentation -> SetShowBorder(); //设置颜色条边框显隐
vtkSmartPointer <vtkScalarBarWidget> barWidget =
    vtkSmartPointer <vtkScalarBarWidget>::New();
barWidget -> SetInteractor(); //设置交互器
barWidget -> SetScalarBarActor(barActor); //设置颜色条
barWidget -> SetRepresentation(barRepresentation); //设置颜色条表示
    
```

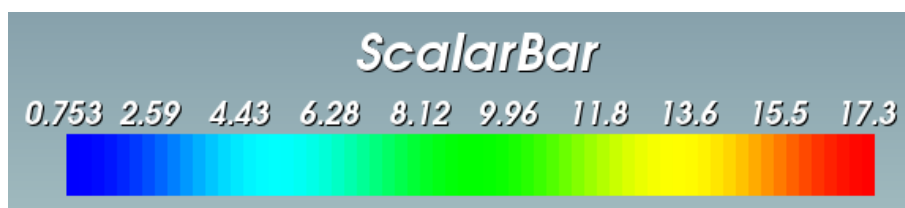


图 3.6 水平颜色条

3.4.2 文件读取及建模

(1) 读取 step/iges 等几何文件

由于 VTK 擅长处理离散数据，无法直接渲染几何模型，因此需要借助几何引擎 Open CASCADE 来导入 step 和 iges 等几何模型文件，并对模型的几何特征进行离散后才能在 VTK 中显示。下面通过一个简单的连杆几何模型来演示具体的离散及显示过程，主要代码如框 3.4 所示，导入的几何模型结果如图 3.7 所示。

框 3.4 读取 step 文件的代码

```

TopoDS_Shape stepShape;
STEPControl_Reader stepReader;
IFSelect_ReturnStatus status;
status = stepReader.ReadFile( ); //读取 step 几何模型文件
if (status == IFSelect_RetDone)
    
```

```

{
    stepReader.TransferRoots();
    stepShape = stepReader.OneShape();
}
BRepTools::Clean(stepShape); //清除 stepShape 上的三角形面
BRepMesh_IncrementalMesh(stepShape, this -> deflection, false, 0.5, true, false);
TopExp_Explorer expFace;
for (expFace.Init(this->shape, TopAbs_FACE); expFace.More(); expFace.Next())
{
    //遍历每个面并进行三角剖分
    TopoDS_Face face = TopoDS::Face(expFace.Current()); //获取 stepShape 的当前面
    TopLoc_Location location;
    Handle(Poly_Triangulation) triangulation = BRep_Tool::Triangulation(face, location);
    if (triangulation.IsNull())
        continue;
    const gp_Trsf& transformation = location.Transformation();
    int nTriangle = triangulation->NbTriangles(); //获取该拓扑面上的三角形面片个数
    int nNode = triangulation->NbNodes(); //获取该拓扑面上的点个数
    if (nTriangle < 1 || nNode < 1)
        continue;
    //将三角面片转化成 VTK 中的三角形单元
    int n1, n2, n3;
    vtkSmartPointer<vtkPolyData> polyData = vtkSmartPointer<vtkPolyData>::New();
    vtkSmartPointer<vtkTriangle> vtkTri = vtkSmartPointer<vtkTriangle>::New();
    polyData -> Allocate(nTriangle, nNode);
    const Poly_Array1OfTriangle& triangles = triangulation->Triangles(); //获取该拓扑面
    上的所有三角形面片
    const TColgp_Array1OfPnt& nodes = triangulation->Nodes(); //获取该拓扑上的所有
    点
    for (int i = triangles.Lower(); i <= triangles.Upper(); i++)
    {
        triangles(i).Get(n1, n2, n3); //获取该三角形面片三个点的索引
        if (face.Orientation() != TopAbs_FORWARD)

```

```

{
    int tmp = n3;
    n3 = n2;
    n2 = tmp;
}
vtkTri->GetPointIds()->SetId(0, n1 - nodes.Lower());
vtkTri->GetPointIds()->SetId(1, n2 - nodes.Lower());
vtkTri->GetPointIds()->SetId(2, n3 - nodes.Lower());
polyData->InsertNextCell(vtkTri->GetCellType(), vtkTri->GetPointIds());
}
//将离散得到的点转化成 vtkPoints
vtkSmartPointer <vtkPoints> points = vtkSmartPointer <vtkPoints>::New();
for (int i = nodes.Lower(); i <= nodes.Upper(); i++)
{
    gp_XYZ XYZ = nodes(i).Coord();
    transformation.Transforms(XYZ);
    points -> InsertPoint(i - nodes.Lower(), XYZ.X(), XYZ.Y(), XYZ.Z());
}
polyData -> SetPoints(points);
//可视化管线渲染离散后的几何面
.....
}

```

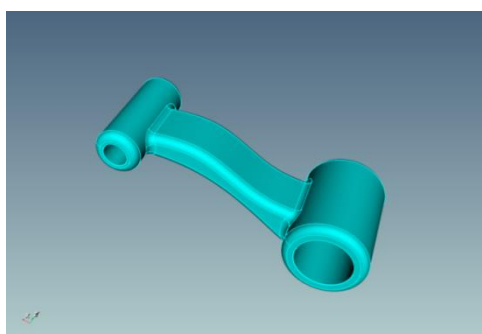


图 3.7 导入连杆几何模型

(2) 读取 STL 等表面网格文件

STL 格式文件用来描述物体表面几何形状，仅由一层表面三角形网格构成。在 VTK 中，可使用 `vtkSTLReader` 类读取 STL 格式的文件，`vtkSTLReader` 是一个

源过程对象，其输出的数据对象经过 `vtkPolyDataMapper` 映射器进入 VTK 渲染引擎即可显示，如图 3.8 所示为使用 `vtkSTLReader` 读取的一个模型。

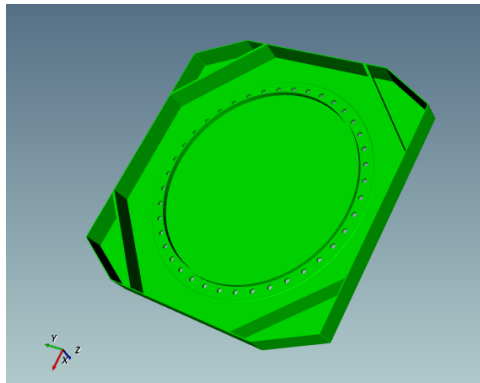


图 3.8 stl 模型

(3) 读取 inp 等前处理文件

Abaqus 的 `inp` 格式文件包含了对整个模型的完整描述，可直接使用该文件进行求解计算。VTK 中没有封装读取该文件的类，因此需要自己编写代码读取，具体读取过程此处省略，仅介绍读取后模型建立的过程。假设读取的节点 ID 和坐标存储在二维数组 `nodeArray` 中，每个单元包含的节点 ID 存储在二维数组 `elementArray` 中，节点总数为 `nodeAmount`，单元总数为 `elementAmount`，单元类型为四面体。首先需要将节点 ID 和坐标添加到 `vtkPoints` 中，即建立 VTK 网格节点，主要代码如框 3.5 所示。

框 3.5 建立 VTK 网格节点的代码

```
vtkSmartPointer<vtkPoints> points =
    vtkSmartPointer<vtkPoints>::New();
for(int i = 0 ; i++ ; i < nodeAmount)
{
    points -> InsertPoint(nodeArray[i][0], nodeArray [i][1],
                          nodeArray [i][2], nodeArray [i][3]);
}
```

然后将每个单元包含的节点 ID 添加到 `vtkUnstructuredGrid` 中，即建立 VTK 非结构化网格数据集类型的网格单元，并将上一步建立的 VTK 网格节点数据添加到网格中，主要代码如框 3.6 所示。

框 3.6 建立 VTK 网格单元的代码

```
vtkSmartPointer<vtkUnstructuredGrid> elements =
    vtkSmartPointer<vtkUnstructuredGrid>::New();
elements -> Allocate(elementAmount);
```



```

for(int i = 0 ; i < elementAmount ; i++)
{
    elements -> InsertNextCell(VTK_TETRA,4,elementArray[i]);
}
elements -> SetPoints(points);

```

至此，非结构化网格数据集的组织结构已定义完成，最后经过 `vtkDataSetMapper` 映射器进入 VTK 渲染引擎即可完成显示，如图 3.9 所示为读取的 `inp` 格式文件模型。

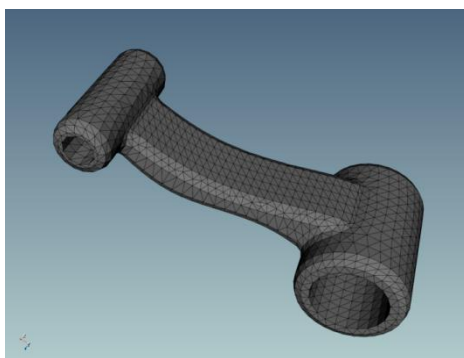


图 3.9 导入 `inp` 网格模型

(4) 读取 `vtk` 等后处理文件

VTK 的 `vtk` 格式文件存储了模型的网格信息和属性信息，可视为后处理文件。VTK 中封装了大量读取 `vtk` 格式文件的类，根据文件中存储数据集类型的不同，使用的类也不同，如读取直线网格数据集可用 `vtkRectilinearGridReader` 类、读取结构化网格数据集可用 `vtkStructuredGridReader` 类、读取非结构化网格数据集可用 `vtkUnstructuredGridReader` 类等等。读取完成后，可通过 `vtkPolyDataMapper` 或 `vtkDataSetMapper` 映射器进入渲染引擎进行显示，关于利用 `vtk` 文件进行各种后处理操作的具体方法，将在 3.4.4 节中详细介绍。此外，还可通过 `GetOutput()` 函数输出对应类型的数据集，进一步获取网格模型的更多信息，如节点、单元数等，常用功能的代码如框 3.7 所示。

框 3.7 `vtkUnstructuredGridReader` 常用功能的代码

```

vtkSmartPointer<vtkUnstructuredGridReader> reader =
    vtkSmartPointer<vtkUnstructuredGridReader>::New();
reader -> SetFileName(); //设置要读取的文件路径及文件名
reader -> Update(); //更新管线，即开始读取文件
reader -> GetOutput() -> GetBounds(); //获取模型边界
reader -> GetOutput() -> GetScalarRange(); //获取标量值范围

```

```
reader -> GetOutput() -> GetNumberOfCells(); //获取单元数
reader -> GetOutput() -> GetNumberOfPoints(); //获取节点数
```

3.4.3 视图交互

3.4.3.1 投影视图

通过控制视图窗口中的相机 `vtkCamera` 可控制视图的投影方向，主要设置相机的位置、焦点和向上方向三个参数，分别使用 `SetPosition()`、`SetFocalPoint()`和 `SetViewUp()`函数实现，最后使用 `SetActiveCamera()`函数将相机添加到渲染器中，主要代码如框 3.8 所示。

框 3.8 投影视图的主要代码

```
vtkSmartPointer<vtkCamera> camera =
    vtkSmartPointer<vtkCamera>::New();
camera -> SetPosition(0,0,0);
camera -> SetFocalPoint(1,0,0); //前视图
camera -> SetViewUp(0,0,1);
camera -> SetFocalPoint(-1,0,0); //后视图
camera -> SetViewUp(0,0,1);
camera -> SetFocalPoint(0,-1,0); //左视图
camera -> SetViewUp(0,0,1);
camera -> SetFocalPoint(0,1,0); //右视图
camera -> SetViewUp(0,0,1);
camera -> SetFocalPoint(0,0,1); //仰视图
camera -> SetViewUp(-1,0,0);
camera -> SetFocalPoint(0,0,-1); //俯视图
camera -> SetViewUp(-1,0,0);
camera -> SetFocalPoint(1,1,1); //轴测图
camera -> SetViewUp(-1,0,0);
renderer -> SetActiveCamera(camera);
```

3.4.3.2 视图操作样式

常用的视图操作样式有平移、旋转、缩放等等，定义继承自 `vtkInteractorStyleTrackballCamera` 交互样式的类 `OperateStyle`，使用基类的 `StartPan()`和 `EndPan()`，`StartRotate()`和 `EndRotate()`，`StartDolly()`和 `EndDolly()`函数可实现自定义的视图操

作样式。首先在 `OperateStyle` 类中定义一个操作样式枚举，以用来表示当前的操作样式，便于实现对应的动作，主要代码如框 3.9 所示。

框 3.9 定义操作样式枚举的代码

```
enum OperateStyleMode
{
    PanMode,          //平移
    RotateMode,       //旋转
    DollyMode,        //缩放
};
```

当鼠标左键按下时，触发 `OnLeftButtonDown()` 函数，渲染窗口交互器根据当前操作样式，开始平移（或旋转、或缩放）视图，主要代码如框 3.10 所示。当鼠标左键释放时，触发 `OnLeftButtonUp()` 函数，交互器结束当前操作样式。

框 3.10 鼠标左键按下的代码

```
void OperateStyle::OnLeftButtonDown()
{
    switch(operateStyleMode)
    {
        case OperateStyle::PanMode:
            StartPan();
            break;
        case OperateStyle::RotateMode:
            StartRotate();
            break;
        case OperateStyle::DollyMode:
            StartDolly();
            break;
        default:
            break;
    }
}
```

以上为操作样式的定义过程，定义完成后，还需要将该操作样式添加到当前渲染窗口中去，以使视图中的鼠标事件能够被操作样式捕获并响应，主要代码如框 3.11 所示。

框 3.11 将操作样式添加到渲染窗口的代码

```

vtkSmartPointer <OperateStyle> operateStyle =
    vtkSmartPointer <OperateStyle>::New();
renderer -> GetRenderWindow() -> GetInteractor() -> SetInteractorStyle(operateStyle);

```

除以上三种基本操作样式外,还有框选放大、适应窗口两种常用的操作样式,这两种操作样式的定义方式与前三种略有区别。框选放大可使用 VTK 中封装的 `vtkInteractorStyleRubberBandZoom` 交互样式,直接将该交互样式的对象添加到当前渲染窗口中即可,主要代码如框 3.12 所示。

框 3.12 实现框选放大功能的代码

```

vtkSmartPointer <vtkInteractorStyleRubberBandZoom> boxZoom =
    vtkSmartPointer <vtkInteractorStyleRubberBandZoom>::New();
renderer -> GetRenderWindow() -> GetInteractor() -> SetInteractorStyle(boxZoom);

```

适应窗口不需要改变交互样式,它是根据渲染窗口中当前模型的边界尺寸,重置渲染窗口中的相机,使当前模型以适当的比例显示,代码如框 3.13 所示。

框 3.13 实现适应窗口功能的代码

```

renderer -> ResetCamera(actor -> GetBounds());

```

3.4.3.3 拾取

交互式拾取是该软件系统中极其重要的一部分,因为其涉及到前处理部分中边界条件及载荷等模块的定义。根据需求,分为几何模型的交互式拾取和网格模型的交互式拾取,几何模型的交互式拾取包括拾取几何面、几何边和几何顶点;网格模型的交互式拾取包括拾取网格单元、网格面、网格边、网格节点等,其拾取方式包括单选、多选、鼠标框选、输入坐标框选等多种方式。本文仅对拾取几何面、单选拾取网格单元、框选拾取网格节点三种拾取功能的实现方法加以介绍,其他拾取功能类似。

在 VTK 中,只要是通过鼠标操作进行的拾取,首先都需要获取鼠标事件发生的位置,可使用 `GetEventPosition()` 函数实现,然后根据拾取类型的不同,选择对应类型的拾取器进行拾取,主要代码如框 3.14 所示。

框 3.14 获取鼠标事件发生位置的代码

```

int xpos = this -> GetInteractor() -> GetEventPosition()[0];
int ypos = this -> GetInteractor() -> GetEventPosition()[1];
vtkSmartPointer <vtkPointPicker> pointPicker =

```

```

vtkSmartPointer <vtkPointPicker>::New();
pointPicker -> Pick(xpos, ypos, 0, this -> GetCurrentRenderer());
vtkIdType pickedId = this -> pPointPicker -> GetPointId();
    
```

VTK 中的拾取器类型及其关系如图 3.10 所示。其中，`vtkCellPicker` 是拾取网格单元的拾取器，`vtkPointPicker` 是拾取网格节点的拾取器，`vtkPropPicker` 是拾取 `vtkProp` 实例对象的拾取器。

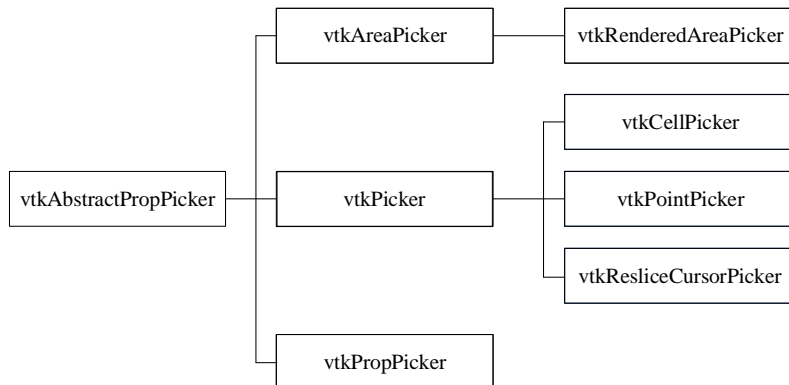


图 3.10 VTK 中的拾取器类型

(1) 拾取几何面

在 3.4.2 节第一部分介绍到，`step` 和 `iges` 等几何模型经几何引擎 `Open CASCADE` 读取并离散后在 VTK 中显示，原几何模型的拓扑结构（面、边、顶点）已经离散成 VTK 中 `vtkActor`，因此要拾取到原几何模型的拓扑结构，使用 `vtkPropPicker` 拾取器即可进行拾取，并将拾取到的几何特征进行高亮显示，如图 3.11 所示为拾取到的几何面。主要代码如框 3.15 所示。

框 3.15 拾取几何面的代码

```

vtkSmartPointer <vtkPropPicker> propPicker =
    vtkSmartPointer <vtkPropPicker>::New();
propPicker->Pick(xpos, ypos, 0, this->GetDefaultRenderer());
vtkSmartPointer <vtkActor> pickedActor = propPicker->GetActor();
pickedActor ->GetProperty()->SetColor(1, 0, 0);
pickedActor ->GetProperty()->SetLineWidth(5.0);
pickedActor ->GetProperty()->SetPointSize(5.0);
    
```

(2) 单选拾取网格单元

由于单选拾取网格单元功能需要鼠标进行操作，因此自定义的拾取类需继承 `vtkInteractorStyleTrackballCamera` 类来监测鼠标事件，然后通过上面的方法即可获取拾取到的网格单元 ID。

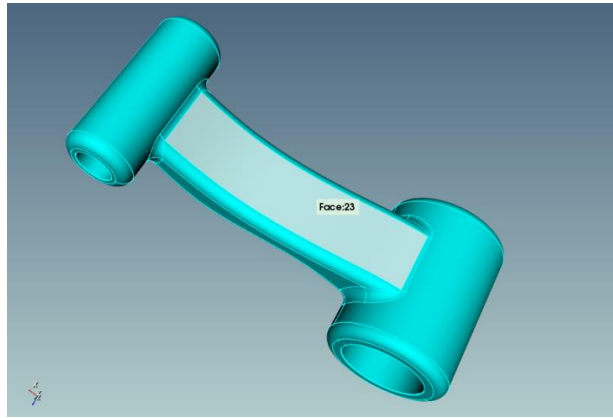


图 3.11 拾取几何面

通常，拾取后还要进行高亮显示，其主要流程如图 3.12 所示。首先将拾取到的 ID 存储到 `vtkIdTypeArray` 中，然后通过 `vtkSelectionNode`、`vtkSelection` 和 `vtkExtractSelection` 等类将其从整个数据集中提取出来，即可进行高亮渲染，主要代码如框 3.16 所示。如图 3.13 所示为使用该方法拾取到的一个网格单元。

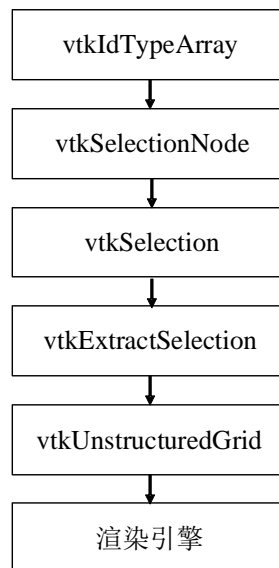


图 3.12 高亮显示流程

框 3.16 高亮拾取到的网格单元的代码

```

vtkSmartPointer <vtkIdTypeArray> pickedIdArray =
    vtkSmartPointer <vtkIdTypeArray>::New();
pickedIdArray -> SetNumberOfValue(1);
pickedIdArray -> InsertNextValue(pickedId);
vtkSmartPointer <vtkSelectionNode> selectionNode =
    vtkSmartPointer <vtkSelectionNode>::New();
selectionNode -> SetFieldType(vtkSelectionNode::CELL);
selectionNode -> SetContentType(vtkSelectionNode::INDICES);
selectionNode -> SetSelectionList(pickedIdArray);
  
```

```

vtkSmartPointer<vtkSelection> selection =
    vtkSmartPointer<vtkSelection>::New();
selection -> AddNode(selectionNode);
vtkSmartPointer<vtkExtractSelection> extractSelection =
    vtkSmartPointer<vtkExtractSelection>::New();
extractSelection -> SetInputData(0, cellPicker -> GetDataSet());
extractSelection -> SetInputData(1, selection);
extractSelection -> Update();
vtkSmartPointer<vtkUnstructuredGrid> unstructuredGrid =
    vtkSmartPointer<vtkUnstructuredGrid>::New();
unstructuredGrid -> ShallowCopy(extractSelection -> GetOutput());

```

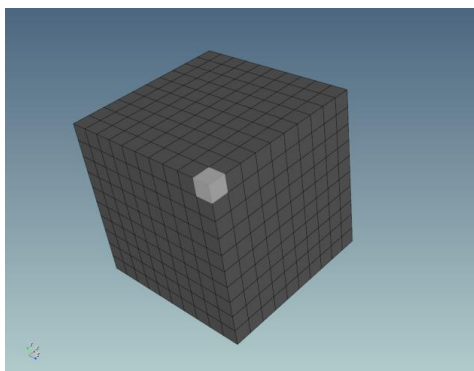


图 3.13 单选拾取网格单元

(3) 框选拾取网格节点

在 VTK 中, 可使用 `vtkAreaPicker` 类实现框选, 该类是通过将二维屏幕上绘制的矩形框转化成三维场景中的视锥, 然后通过 `vtkExtractSelectedFrustum` 将场景中的数据与视锥相交, 位于视锥内的网格节点即为被拾取, 如图 3.14 所示, 主要代码如框 3.17 所示。

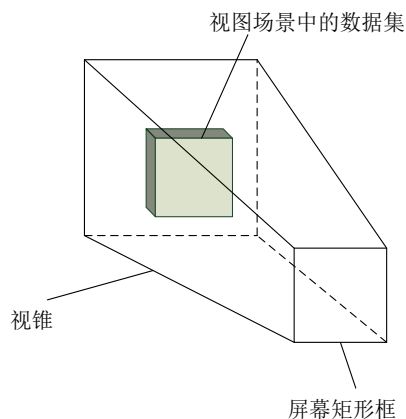


图 3.14 框选产生的视锥

框 3.17 实现框选功能的代码

```

vtkSmartPointer <vtkAreaPicker> areaPicker =
    vtkSmartPointer <vtkAreaPicker>::New();
renWin -> GetInteractor() -> SetPicker(areaPicker);
vtkSmartPointer <vtkExtractSelectedFrustum> frustum =
    vtkSmartPointer <vtkExtractSelectedFrustum>::New();
frustum -> SetInputData(areaPicker->GetDataSet()); //设置数据集
frustum -> SetFrustum(areaPicker->GetFrustum()); //设置视锥
frustum -> Update();
vtkUnstructuredGrid *uGrid =static_cast<vtkUnstructuredGrid*>
    (frustum -> GetOutput()); //输出为非结构化网格数据集
vtkIdTypeArray *ids = static_cast<vtkIdTypeArray*>
    (uGrid -> GetPointData() -> GetArray("pickedIds")); //获取拾取到的节点 ID

```

此外，在此之前还需使用 `vtkIdFilter` 过滤器为原始数据集中的节点和单元 ID 数组赋予名称，便于拾取后能够根据数组名获取拾取到的节点或单元 ID，主要代码如框 3.18 所示。

框 3.18 为节点和单元赋名的代码

```

vtkSmartPointer <vtkIdFilter> idFilter = vtkSmartPointer <vtkIdFilter>::New();
idFilter -> SetInputConnection();
idFilter -> SetIdsArrayName("pickedIds");
idFilter -> Update();

```

与单选拾取类似，自定义的框选拾取类 `BoxPicker` 需继承 VTK 中的 `vtkInteractorStyleRubberBandPick` 类来监测视图中的鼠标事件，并将其添加到视图窗口交互器中去，主要代码如框 3.19 所示。

框 3.19 将框选拾取添加到视图窗口中的代码

```

vtkSmartPointer <BoxPicker> boxPick =
    vtkSmartPointer <BoxPicker>::New();
renWin -> GetInteractor() -> SetInteractorStyle(boxPick);
boxPick -> SetCurrentRenderer(renderer);
boxPick -> StartSelect();

```

经过以上步骤即可实现框选拾取功能，如图 3.15 所示为使用该方法拾取到的网格节点。

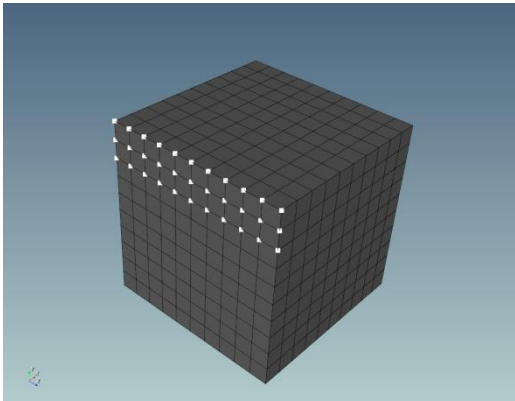


图 3.15 框选拾取网格节点

3.4.4 后处理部分

有限元后处理的形式非常丰富，本文仅介绍云图、切面图、切割图、等值面图（等值线图）及变形图等。以下介绍的几种可视化方法中，都是使用后缀为.vtk的非结构化网格数据文件，并使用 `vtkUnstructuredGridReader` 类进行读取，该类能够将文件中所有的属性数据（包括标量数据和矢量数据）读入并存储，因此所展示的代码中没有额外赋予属性数据过程。

（1）云图

通常，云图是通过标量值映射实现的，即某点或单元上的标量值通过颜色查找表映射成一个颜色，渲染时将该颜色赋予该点或单元。颜色查找过程如下：如图 3.16 所示，已定义的颜色查找表中有 n 个颜色，可通过 `SetNumberOfColors()` 函数设置，通过映射器的 `SetScalarRange()` 函数可设置要映射的标量值的范围，小于最小限制值的标量值将被映射成最小标量值颜色，大于最大限制值的标量值将被映射成最大标量值颜色。

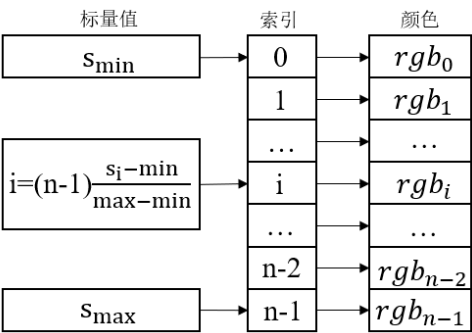


图 3.16 颜色映射原理

在 VTK 中，建立颜色查找表可使用 `vtkLookupTable` 类，其经常使用的功能主要有设置颜色查找表中颜色的个数、设置色调范围等，主要代码如框 3.20 所示。

框 3.20 定义颜色查找表的代码

```

vtkSmartPointer <vtkLookupTable> lookupTable =
    vtkSmartPointer <vtkLookupTable>::New();
lookupTable -> SetNumberOfColors(); //设置颜色查找表中颜色的个数
lookupTable -> SetHueRange(); //设置色调范围
lookupTable -> Build(); //建立颜色查找表

```

建立好颜色查找表后，还需要将其添加到映射器中，并设置要映射的标量值范围，其主要代码如框 3.21 所示。

框 3.21 将颜色查找表添加到映射器中的代码

```

vtkSmartPointer <vtkDataSetMapper> mapper =
    vtkSmartPointer <vtkDataSetMapper>::New();
mapper -> SetLookupTable(lookupTable); //添加颜色查找表
mapper -> SetScalarRange(); //设置标量值范围

```

如图 3.17 所示为使用该方法得到的云图示例：

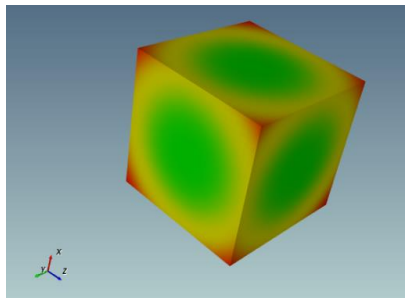


图 3.17 云图示例

(2) 切面图

在后处理过程中，有时候需要观察三维模型内部的某个切面云图。在 VTK 中，可使用 `vtkCutter` 类用来生成切面数据集，根据需求可选择不同形状的切面，常用的有平面（`vtkPlane`）、球面（`vtkSphere`）和立方体面（`vtkCube`）等，方法如框 3.22 所示。如图 3.18(a)所示为平面切面图，图 3.18(b)所示为球面切面图。

框 3.22 实现切面图的主要代码

```

vtkSmartPointer <vtkPlane> plane = vtkSmartPointer <vtkPlane>::New();
plane -> SetOrigin (); //设置切面平面原点
plane -> SetNormal(); //设置切面平面法线
vtkSmartPointer <vtkCutter> cutter = vtkSmartPointer <vtkCutter>::New();
cutter -> SetInputConnection(); //设置切面数据集
cutter -> SetCutFunction(plane); //设置切面函数

```

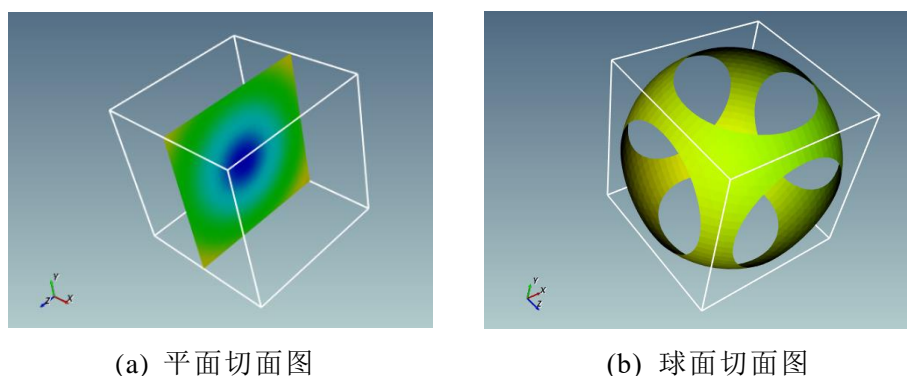


图 3.18 切面图示例

(3) 切割图

与切面图不同，切割图是将原数据集切除一部分，对剩余的部分进行观察，而切面图仅仅保留一个切面。在 VTK 中，可使用 `vtkClipDataSet` 和 `vtkBoxClipDataSet` 类来处理非结构化网格数据生成切割图。`vtkClipDataSet` 类的使用方法与 `vtkCutter` 类相似，需要设置切割函数，一般为平面切割。此外，可以通过 `SetValue()` 函数设置切割平面的偏移，方法如框 3.23 所示。如图 3.19 所示分别为偏移值等于 0、0.4 和 -0.4 切割后的效果。

框 3.23 实现切割图的主要代码

```

vtkSmartPointer<vtkPlane> plane = vtkSmartPointer<vtkPlane>::New();
plane -> SetOrigin(); //设置切割平面原点
plane -> SetNormal(); //设置切割平面法线
vtkSmartPointer<vtkClipDataSet> clipper = vtkSmartPointer<vtkClipDataSet>::New();
clipper -> SetInputData(); //设置待切割数据集
clipper -> SetClipFunction(plane); //设置切割函数
clipper -> SetValue(value); //切割平面偏移量
clipper -> Update();
    
```

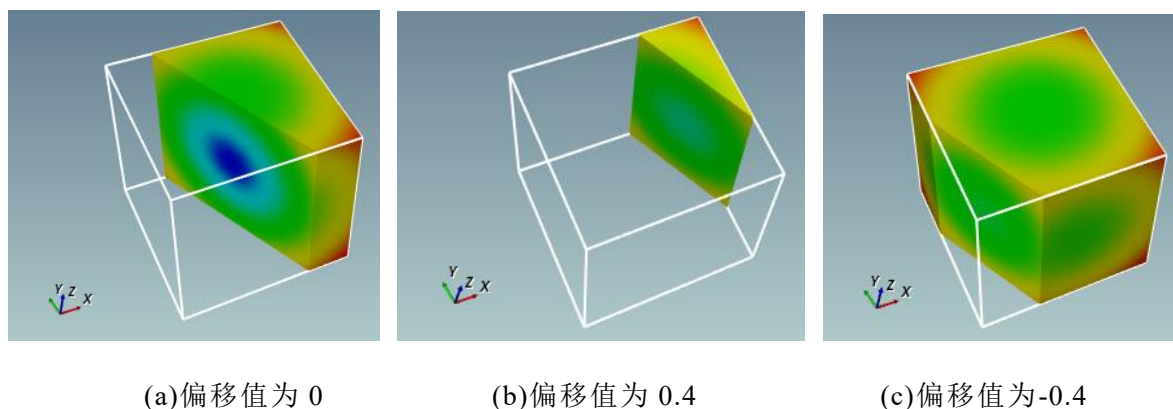


图 3.19 平面切割图示例

使用 `vtkBoxClipDataSet` 进行切割仅可以得到由六个面围成的方盒形状, 该形状由两个对角点及六个面的法线确定, 通过 `SetBoxClip()` 函数设置, 主要代码如框 3.24 所示, 如图 3.20 所示为使用该方法对立方体进行切割后所得的一部分。

框 3.24 实现盒型切割的主要代码

```
vtkSmartPointer<vtkBoxClipDataSet> boxClip =
    vtkSmartPointer<vtkBoxClipDataSet>::New();
boxClip->SetInputConnection(); //设置待切割数据集
boxClip->SetBoxClip(); //设置切割范围
```

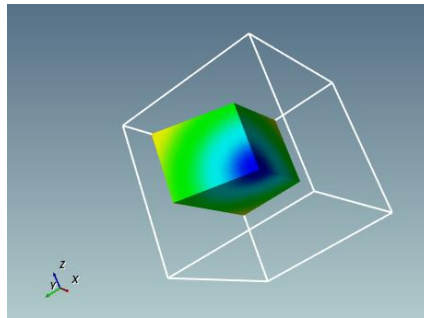


图 3.20 盒型切割图示例

(4) 等值线图 and 等值面图

等值线图或等值面图是将数据集中标量数据相等的位置进行连接, 形成线或面数据集, 该方法可用来精确地表示出数据集上标量值相等的点。当数据集中所有位置的标量值均已确定时, 可使用 VTK 中的边缘追踪法 (edge-tracking) 和分治法 (divide and conquer) 将值相等的点连接成轮廓。如图所示为一个简单的图像数据集, 其节点旁的数字为该节点上的标量值, VTK 会自动进行线性插值计算出两个节点之间的标量值, 如图 3.21 所示使用的边缘追踪法连接标量值为 5 的等值点。

在 VTK 中, 可以使用 `vtkContourFilter` 过滤器生成等值线图或等值面图, 若输入到过滤器中的数据为三维数据, 则输出为等值面; 若输入到过滤器中的数据为二维数据, 则输出为等值线, 其主要代码如框 3.25 所示。

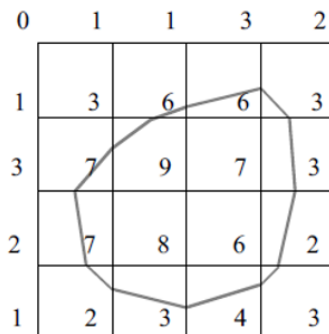


图 3.21 等值线形成原理

框 3.25 实现等值面图的主要代码

```
vtkSmartPointer<vtkContourFilter> contourFilter =
    vtkSmartPointer<vtkContourFilter>::New();
contourFilter->SetInputData(); //输入数据
contourFilter->GenerateValues(); //设置等值面（线）的个数及起始值
```

如图 3.22 所示为采用该方法得到的标量值在 10 到 15 之间的三个等值面图示例：

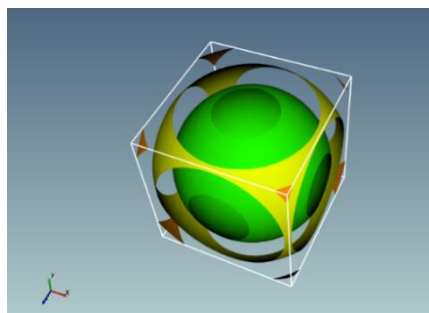
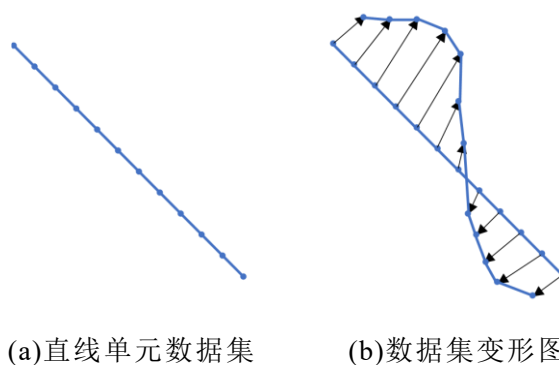


图 3.22 等值面图示例

（5）变形图

变形图的实现与矢量数据有关，如图 3.23(a)所示，一个仅包含直线单元的数据集，以该数据集的节点为起点，每个节点上的矢量数据为终点，将所有终点用直线单元进行连接，即可得到该数据集的变形图，如图 3.23(b)所示。



(a)直线单元数据集 (b)数据集变形图

图 3.23 变形图原理

在 VTK 中，可使用 `vtkWarpVector` 类实现矢量数据的变形，通过该类的 `SetScaleFactor()` 函数可设置形变因子，即可得到沿位移矢量乘以形变因子后点的坐标，其主要代码如框 3.26 所示。

框 3.26 实现变形图的主要代码

```
vtkSmartPointer<vtkWarpVector> warp =
    vtkSmartPointer<vtkWarpVector>::New();
warp->SetInputConnection();
warp->SetScaleFactor();
```

如图 3.24 所示为采用该方法得到的变形图及未变形轮廓图示例：

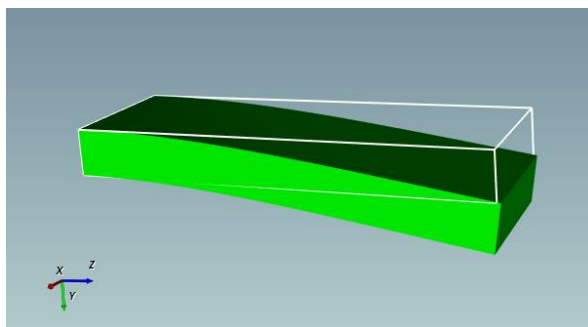


图 3.24 变形图示例

3.5 本章小结

本章主要介绍了基于 Qt 和 VTK 的前后处理软件系统的设计与实现过程，包括软件设计目标、软件框架设计、功能模块设计等，最后介绍了本前后处理系统界面上视图部分的一些重点功能的实现，包括视图部件、文件读取及建模、视图交互和后处理技术等。

第 4 章 软件介绍及应用

本章在该前后处理软件系统已开发完成并通过测试的基础上，对软件界面及用法进行介绍，并使用悬臂梁算例和汽车白车身算例演示整个分析流程。

4.1 主界面介绍

如图 4.1 所示，主界面共分为菜单栏、工具栏、状态栏、左侧模型树和网格零件信息栏、右侧前处理属性定义栏以及中央视图区等几个部分，其中工具栏包含一些常用的功能，如文件操作、视图操作、右侧停靠窗口的切换等；左侧模型树展示当前视图中的零件、材料等前处理信息；左侧网格零件信息栏显示当前网格零件的节点数、单元数、尺寸等信息；右侧前处理属性定义栏用来建立有限元模型，包括生成网格、定义材料、定义截面属性、定义分析步、定义输出变量、定义载荷和边界条件以及定义单元等模块。

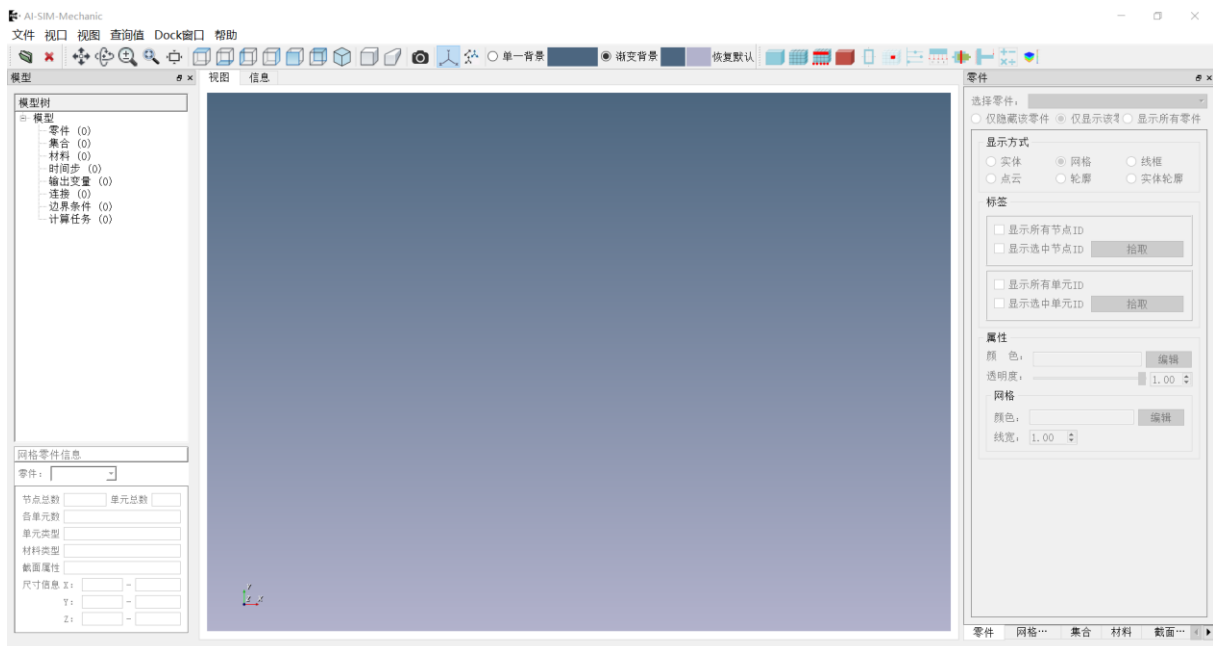


图 4.1 主界面

4.2 算例分析

4.2.1 系统硬件环境

以下两个算例的分析均是在开发前后处理软件系统所使用的本机中进行的，本机操作系统为 64 位 Windows 10 系统，处理器主频为 2.80GHz，RAM 内存为 16.0GB，显卡为 6GB 独立显卡。

4.2.2 悬臂梁算例

悬臂梁算例演示了导入 stl 格式文件模型后建立有限元模型并进行计算的过程，该过程主要包括生成网格、定义材料、定义截面属性、定义分析步、定义输出变量、定义载荷和边界条件以及定义单元等内容，最后提交计算并进行后处理。

(1) 问题描述

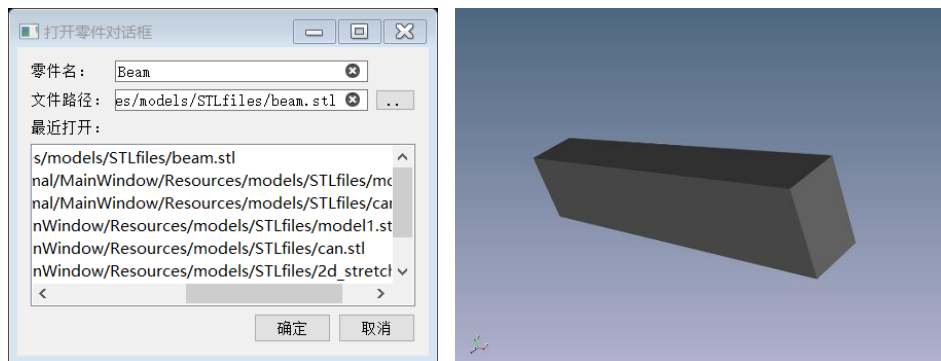
如图 4.2 所示，悬臂梁长 1m，横截面为矩形，长 0.26m，宽 0.18m。一端固定，另一端受到垂直于梁的力 F ，大小为 300N。悬臂梁材料的弹性模量 $E=2.1\times 10^7\text{ N/mm}^2$ ，泊松比 $\mu=0.3$ 。



图 4.2 悬臂梁示意图

(2) 导入 stl 模型

如图 4.3(a)所示为打开零件对话框，在该对话框中，可设置零件名和文件路径，也可以直接从最近打开的文件路径列表选取，如图 4.3(b)所示为导入的悬臂梁模型。



(a) 打开零件对话框

(b) 悬臂梁模型

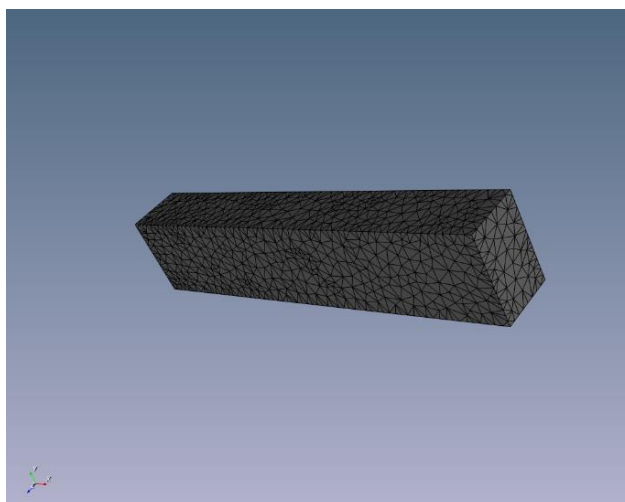
图 4.3 打开悬臂梁模型文件

(3) 生成网格

如图 4.4(a)所示为网格生成定义窗口，首先需选择待生成网格零件，然后选择网格形状并设置相应控制参数，如四面体网格可设置最大半径-边缘比约束、最小二面角约束、网格尺寸、最大单元体积等控制参数。点击【生成网格】按钮即可自动进行网格剖分，并实时显示网格剖分进度，及生成的节点总数、单元总数和所用时间等信息。如图 4.4(b)所示为生成的悬臂梁网格模型。



(a) 网格生成定义窗口



(b) 生成的网格模型

图 4.4 网格生成

(4) 定义材料

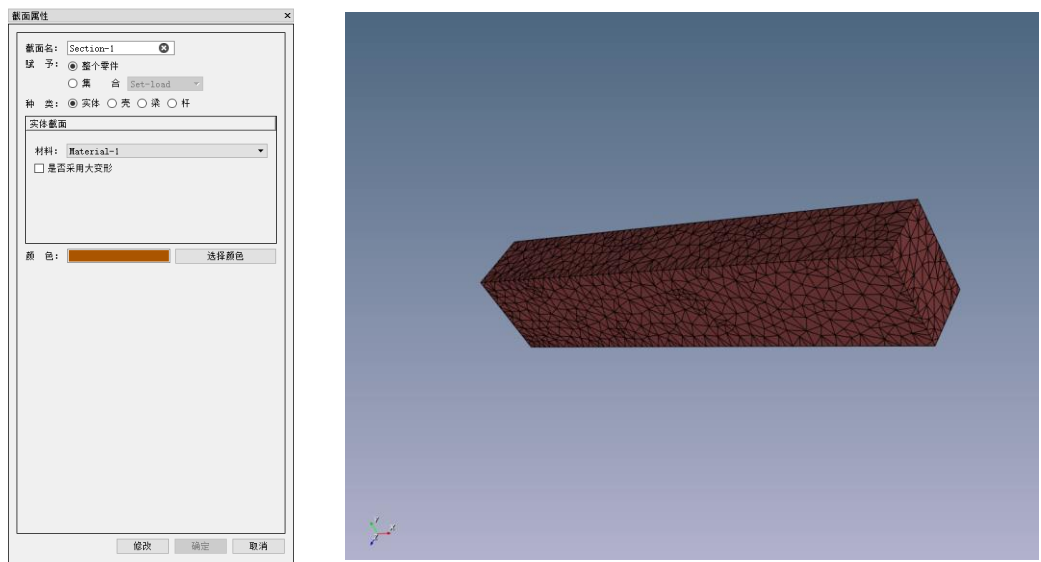
如图 4.5 所示为材料属性定义窗口，可设置材料名、材料种类、具体材料类型及其参数等内容。本算例选择线弹性材料类型，并设置密度为 $\rho = 7900 \text{ kg/m}^3$ ，弹性模量为 $E = 2.1 \times 10^7 \text{ N/mm}^2$ ，泊松比为 $\mu = 0.3$ ，热膨胀系数为 $\alpha = 1$ 。



图 4.5 材料属性定义窗口

(5) 定义截面属性

如图 4.6(a)所示为截面属性定义窗口，可设置截面名、截面所赋予的零件或某个集合、截面种类及该种类截面的具体参数、截面颜色等内容。截面属性既可以赋予到整个网格零件，也可以通过集合赋予到部分网格单元上，从而实现对同一个零件定义不同的截面属性。本算例对整个零件赋予相同截面属性，其类型为实体截面，材料为第 (3) 步新建的材料 Material-1。如图 4.6(b)所示为赋予截面属性后网格模型。



(a)截面属性定义窗口 (b)定义的截面属性

图 4.6 定义截面属性

(6) 定义分析步

如图 4.7 所示为分析步定义窗口，可设置分析步名、分析步顺序、分析步类型及该类型分析步的具体参数等内容。本算例新建的分析步为线性静态分析步，其时间步默认为 1，使用 Pardiso 方程求解器和列压缩矩阵类型。



图 4.7 分析步定义窗口

(7) 定义输出变量

如图 4.8 所示为输出变量定义窗口，可设置输出变量名、输出变量分析步、输出变量域、输出频率以及要输出的变量等内容。本算例输出变量的分析步为第 (5) 步新建的分析步 analysisStep-1，作用域为整个零件，输出频率为所有时间步，输出平动位移、转动位移、应力分量和应变分量四个属性变量。

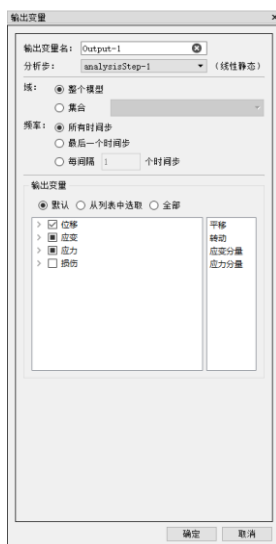
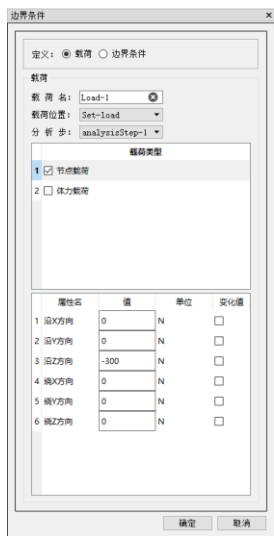


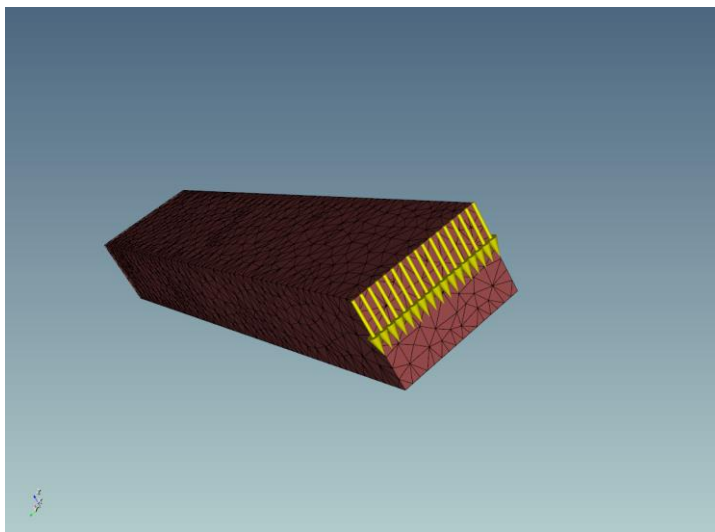
图 4.8 输出变量定义窗口

(8) 定义载荷

如图 4.9(a)所示为载荷定义窗口，可设置载荷名、载荷位置、载荷分析步、载荷类型及载荷值等内容。本算例载荷位置为集合 Set-load，即图 4.2 所示悬臂梁右端面的一条边，载荷所在分析步为 analysisStep-1，载荷类型为节点载荷，载荷沿 -Z 方向大小为 300N，其余各方向均为 0。如图 4.9(b)所示为定义的节点载荷。



(a)载荷定义窗口

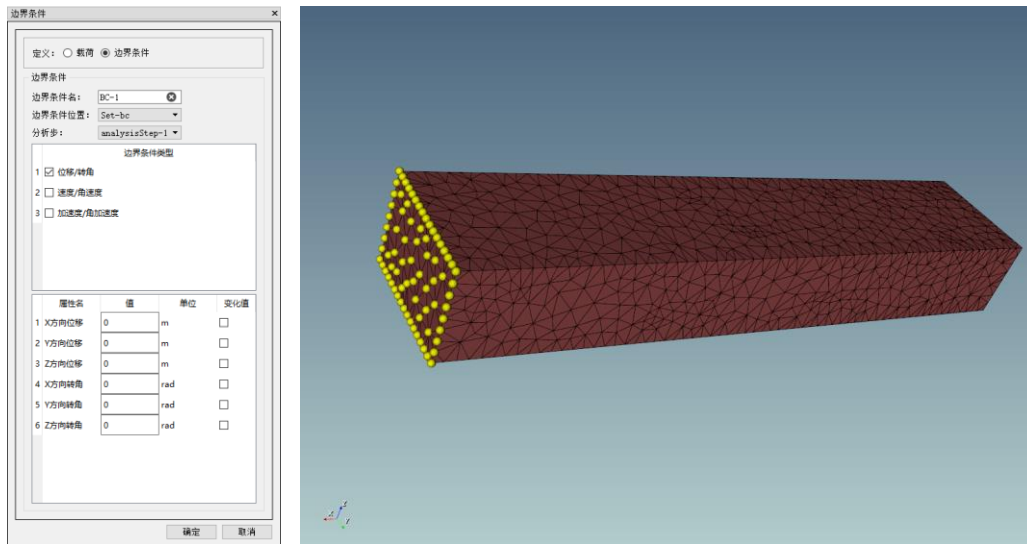


(b)定义的载荷

图 4.9 定义载荷

(9) 定义边界条件

如图 4.10(a)所示为边界条件定义窗口，可设置边界条件名、边界条件位置、边界条件分析步、边界条件类型及边界条件值等内容。本算例边界条件位置为集合 Set-bc，即图 4.2 所示悬臂梁的左端面，边界条件类型为位移/转角边界条件，各方向值均为 0。如图 4.10(b)所示为定义的位移/转角边界条件。



(a)边界条件定义窗口

(b)定义的边界条件

图 4.10 定义边界条件

(10) 定义单元

如图 4.11 所示为单元定义窗口，可设置单元所属截面、单元阶次、单元族及具体的参数值等内容。本算例单元所属截面为 Section-1，单元阶次为线性单元，单元族为三维结构应力，积分点数目为 1，单元算法为有限元法（FEM），最终定义的单元名称为三维结构有限元四面体单元。



图 4.11 单元定义窗口

(11) 提交计算

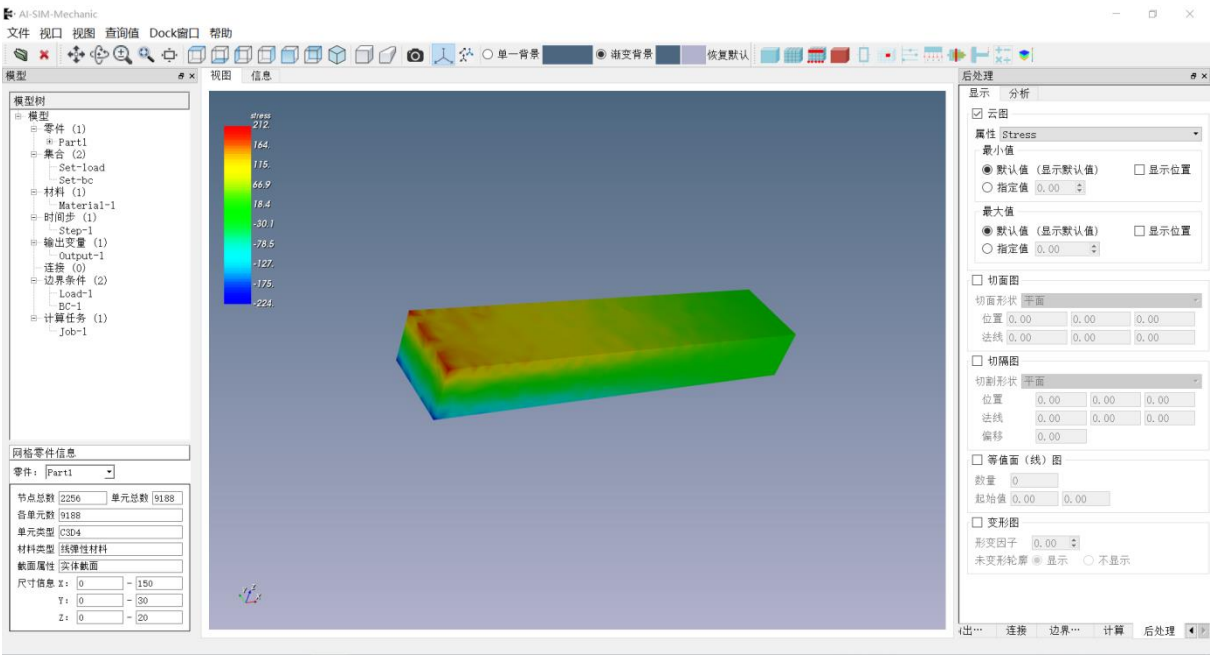
如图 4.12 所示为提交计算窗口，可设置当前计算任务名和要进行计算的零件，零件的选取有三种方式，一是选取当前视图中的所有零件，二是选取当前视图中的某一个零件，三是通过计算文件的形式指定外部零件。开始计算后，可实时监控计算进度，包括时间步、增量、迭代次数、时间步时间和总时间等信息。



图 4.12 提交计算窗口

(12) 后处理

计算完成后即可进入后处理模块，本软件系统能够定义云图、切面图、切割图、等值面（线）图和变形图等可视化形式。如图 4.13 所示，(a)为应力云图，(b)为切面图，(c)为切割图，(d)为等值面图，(e)变形图。



(a)悬臂梁应力云图

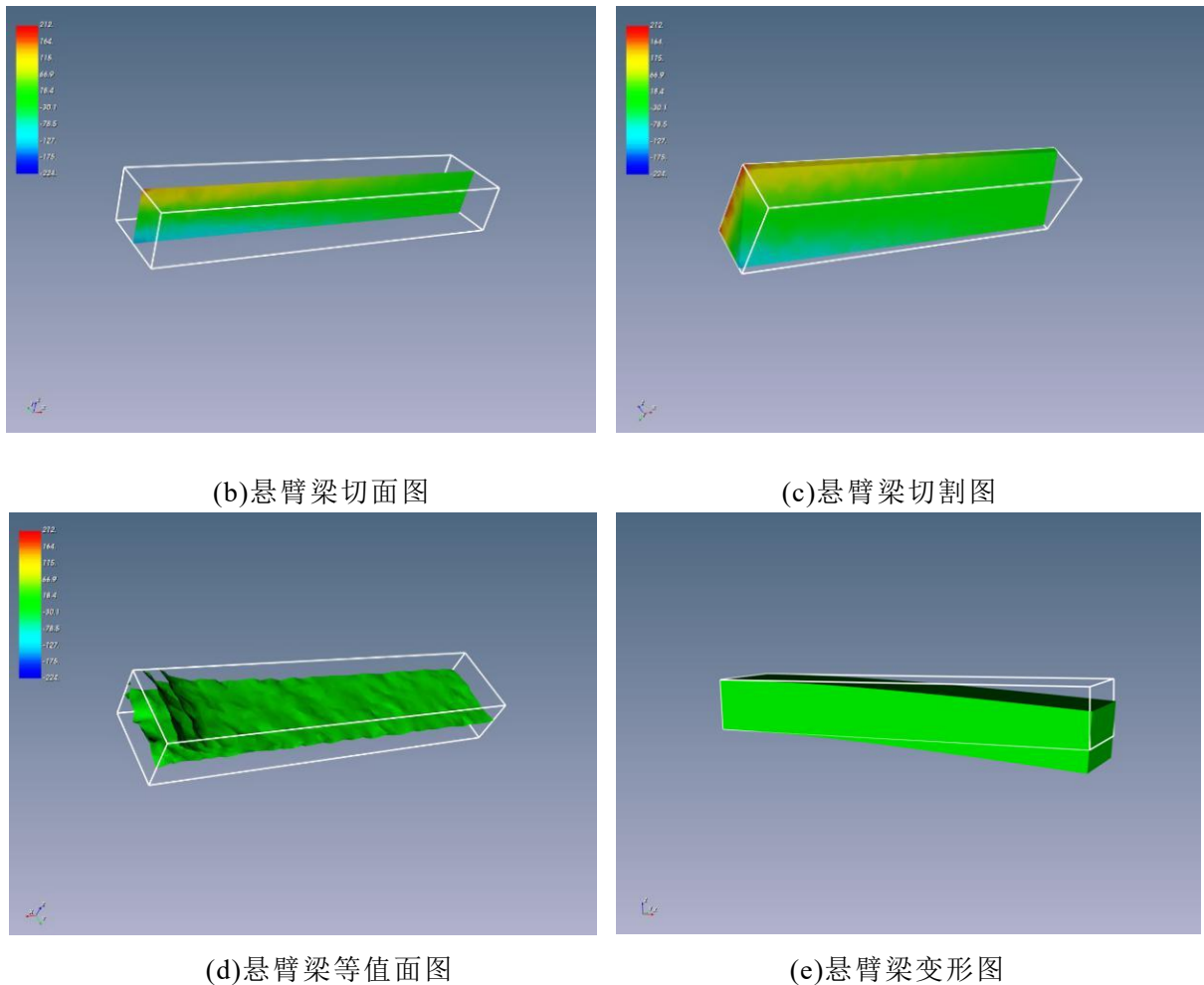


图 4.13 悬臂梁后处理显示

4.2.3 汽车白车身算例

汽车白车身算例演示导入 `inp` 文件后修改原有的前处理属性并进行计算的过程，`inp` 文件为 Abaqus 的输入文件，不仅包含完整的网格模型信息，还可以包括前处理属性信息。因此，若导入的 `inp` 文件中前处理属性定义完整，则可以直接提交计算，也可以在界面上修改属性再计算。本算例仅演示在界面上修改载荷和边界条件两部分属性，其他属性保持不变。

(1) 问题描述

如图 4.14 所示为某型轿车的白车身模型，现对其扭转刚度特性进行分析。假定汽车的悬架类型为独立悬架，当汽车在不平路面上行驶时，其右前轮向上抬起，其余三个车轮均保持在同一水平面上。白车身材料的弹性模量为 $E=3.0\times 10^7\text{ N/mm}^2$ ，泊松比为 $\mu=0.3$ ，密度为 $\rho=7900\text{ kg/m}^3$ 。

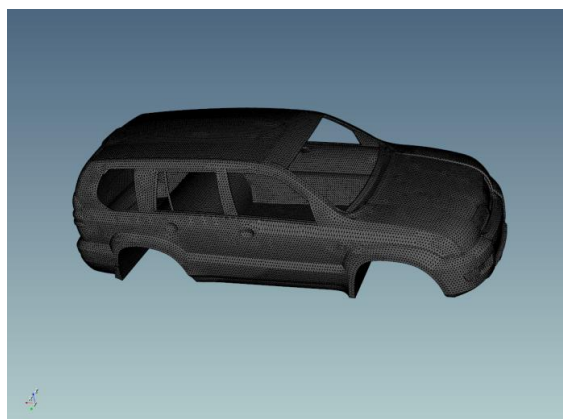


图 4.14 汽车白车身模型

(2) 定义载荷

根据实际工况，汽车右前轮受到地面一垂直向上的力，该力经右前悬架传递到白车身上与悬架相连的点。因此，直接对白车身上与右前悬架相连的节点施加一垂直向上的力 F ，大小为 $F=400\text{N}$ ，如图 4.15 所示。

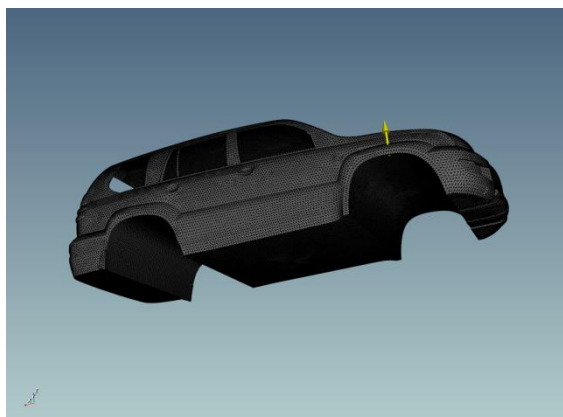


图 4.15 汽车白车身载荷

(3) 定义边界条件

根据实际工况，汽车左前轮和两个后轮在垂直方向上没有位移，只能沿着水平方向移动。因此，对白车身上与左前悬架和两个后悬架相连的节点施加平动自由度约束，如图 4.16 所示。

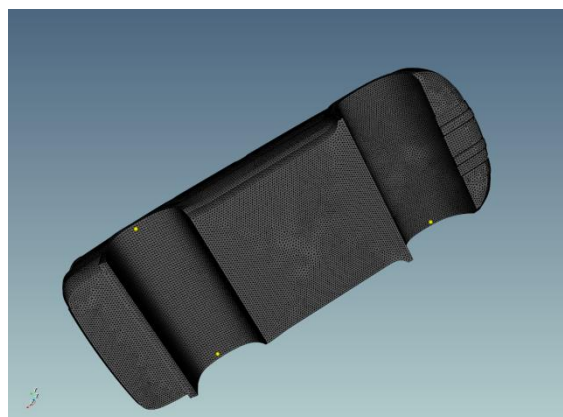


图 4.16 汽车白车身边界条件

(4) 后处理

由于本算例考察的是汽车白车身的扭转刚度，因此仅计算了白车身在载荷作用下的位移结果，如图 4.17 所示为汽车白车身在垂直方向上的位移云图。

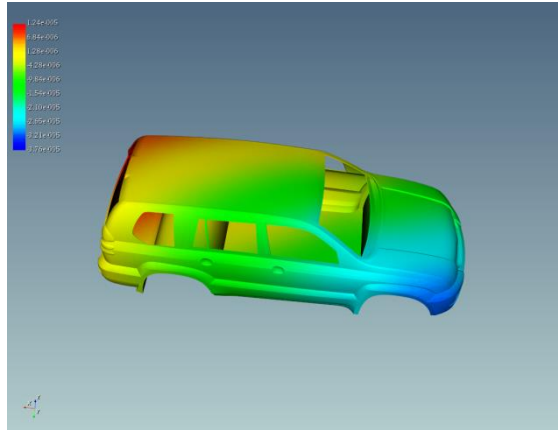


图 4.17 汽车白车身云图

4.3 本章小结

本章主要介绍了前后处理软件系统的界面和使用，并用悬臂梁算例和汽车白车身算例演示了软件使用的整个流程。悬臂梁算例演示了从网格剖分到后处理的全部过程，展示了本软件系统所具备的功能。汽车白车身算例演示了导入 Abaqus 的 inp 格式文件模型，并修改原有的载荷和边界条件属性后进行计算的过程。

总结与展望

CAE 软件应用于工业产品的设计、分析和优化等研发过程，是工业发展所必需的核心技术，具有重要的经济价值和战略地位，对国家凝聚科技创新人才、提升核心竞争力和自主创新能力也具有重要意义，因此发展具有自主知识产权的国产 CAE 软件迫在眉睫。本文在详细了解 CAE 软件研发背景及意义、发展现状及趋势，深入研究图形用户界面框架 Qt 和可视化工具包 VTK 的前提下，基于 Qt 和 VTK 开发了一款用于处理静力学问题的 CAE 前后处理软件系统，该系统具备导入外部模型、建立有限元模型、分析计算及后处理等功能。主要研究内容可总结为以下几点：

(1) 基于 Qt 实现了软件界面部分的开发。本文深入研究了图形用户界面框架 Qt 的基本组成和总体特性，研究其独有的信号与槽通信机制、元对象编译器及常用的 Qt 模块，并基于 Qt 设计了前后处理系统的界面。本软件系统的界面部分由主界面和前后处理各个模块的子界面构成，在设计过程中，严格遵循交互一致性、提供反馈机制、减少失误、提供出错恢复机制、隐藏复杂内容和尽量减轻用户记忆负担等设计原则，开发出了交互友好的图形用户界面。

(2) 基于 VTK 实现了软件系统的视图功能区开发。视图功能区是整个软件界面的核心，模型的显示以及后处理的各种可视化方式都依赖于视图。本文在原有的平移、旋转和缩放等基本操作样式的基础上，添加了框选放大、适应窗口、正交投影和透视投影等操作样式。并根据需要实现了丰富的拾取样式，能够拾取几何模型和网格模型，包括拾取几何面、几何边、几何顶点以及网格单元、网格面、网格边和网格点等模型特征，以及单选、多选和框选三种拾取模式。

(3) 实现了在 VTK 视图中渲染几何模型的功能。由于 VTK 只能处理离散化数据，因此无法直接渲染 step/iges 等格式文件中的几何模型，极大地限制了软件的功能。为解决此问题，本文借助了 Open CASCADE 几何引擎来读取 step/iges 等格式文件中模型的几何信息，然后将模型的几何边和几何面按照一定的精度进行离散，最后在 VTK 中进行渲染，渲染时，将离散的数据按照原几何结构分别渲染成 vtkActor，这样在拾取时就能够拾取到完整的几何边或几何面，从而间接实现了在 VTK 中渲染并拾取几何模型的功能。

(4) 较为全面地阐述了整个前后处理软件系统的设计和实现过程，并给出了系统中的部分主要和关键功能的代码实现。本文按照软件工程的基本要求，阐述了软件系统的设计目标和架构设计，并从文件处理、视图功能和前处理模块等方面对软件的具体功能进行设计，最后详细介绍了视图中的部件、模型的建立过程、视图的交互以及后处理模块等功能的实现方法和关键代码。基于以上流程，最终开发出了一个简单的前后处理软件系统，本软件系统的前处理过程包括网格剖分、

定义材料、定义截面属性、定义分析步、定义输出变量、定义载荷、定义边界条件以及定义单元等部分，后处理模块具备云图、切面图、切割图、等值面（线）图和变形图等功能。

开发出功能复杂、安全可靠的 CAE 软件需要大量的时间投入、专业的知识积累和丰富的经验总结，需要长期艰苦的努力。受研究时间、知识水平、开发经验的限制，本文研究的内容总体来说尚不够全面深入，开发的软件功能较为简单，无法达到商用标准，但也提供了一定的技术和经验积累，熟悉了 CAE 软件的开发流程，为后续的开发工作奠定基础。本次开发的前后处理软件系统还有许多进步的空间，今后仍需继续努力，主要有以下几点：

（1）渲染 step/iges 等几何模型效率较低。在将几何模型离散的过程中，为保证原模型的几何特征不失真，需要设置较高的离散精度，然而离散精度越高离散效率就越低，使得在读取文件并渲染模型的过程中存在效率低的问题。

（2）界面不够美观。本软件系统的界面主题单一，无法满足软件在不同外部环境下的主题切换功能以及不同用户对界面主题的需求。此外，界面控件的大小和布局也不够合理，有些子界面存在大块空白区域，有些子界面又过于拥挤繁杂，从而影响整体美观。

（3）功能较为简单。前处理模块的各个部分功能较简单，仅包含一些常用的属性设置，对于较为复杂的情况难以满足，如材料、载荷和边界条件等属性的定义。另外，后处理模块的可视化功能也不够丰富，还需要研究更多的可视化形式，如动画、XY 图表等。

（4）没有预留足够的接口。由于前期的考虑不周，在设计阶段无法涵盖所有功能，导致在开发过程中，接口设计不足，因此给后续功能扩展带来了一定的困难。

对于以上几点以及尚未发现的不足之处，在今后的研究工作中仍需要继续努力、坚决克服，积极总结开发经验，努力提升编程方法，争取在团队协作中开发出一款优秀的、达到商业标准的国产 CAE 软件。

参考文献

- [1] 金建海.船舶 CAE 前后处理系统研制[D].江南大学,2012.
- [2] 国务院.国务院关于印发《中国制造 2025》的通知[EB/OL].
http://www.gov.cn/zhengce/content/2015-05/19/content_9784.htm,2015.
- [3] 发展 CAE 软件产业的战略对策——第 339 次香山科学会议召开[J].计算机辅助工程,2009,18(01):98-99.
- [4] CAE 软件的差距及自主路[C].西南汽车信息(2018 年 12 期 总第 393 期).重庆汽车工程学会,2018:24-29.
- [5] ABAQUS/Standard, User's guide and theoretical manual, Version6.8, Hibbit, Karlsson & Serensen, Inc(2008).
- [6] Tadeusz Stolarski, Y. Nakasone, S. Yoshimoto. Engineering Analysis with ANSYS Software[M]. Elsevier Science, 2018.
- [7] S.Kodiyalam,G.N.Vanderplaats,H.Miura. Structural shape optimization with MSC /NASTRAN[J]. Computers & Structures,1991,40(4):821-829.
- [8] 钟万勰,陆仲绩.紧密结合工程 实现自主 CAE 软件产业跨越式发展[J].工程研究-跨科学视野中的工程,2009,1(01):34-38.
- [9] Feng Kang. Difference Schemes Based On Variational Principle[J].Journal of Applied and Computational Mathematics,1965,2(4):238-262.
- [10] 钟万勰,朱建平.薄壁杆件有限元中的广义位移方法[J].大连工学院报,1985(04):25-30.
- [11] 钟万勰,纪峥.理性有限元[J].计算结构力学及其应用,1996(01):1-8.
- [12] 张洪武,钟万勰,钱令希.土体固结弹塑性分析的参数二次规划理论及有限元解[J].岩土力学,1995(01):35-45.
- [13] 梁国平,何江衡.广义有限元方法——一类新的逼近空间[J].力学进展,1995(04):562-565.
- [14] 梁国平.变网格的有限元法[J].计算数学,1985(04):377-384.
- [15] 梁国平,傅子智.混合杂交罚函数有限元方法及其应用[J].应用数学和力学,1984(03):377-390.
- [16] 李华,程耿东,顾元宪.一种新的全四边形网格快速生成方法——模板法[J].计算结构力学及其应用,1996(01):25-33.
- [17] 王海文,顾元宪.结构-声学耦合系统的有限元分析方法[C].大连理工大学工程力学系.计算力学研究与进展——中国力学学会青年工作委员会第三届学术年会论文集.大连理工大学工程力学系:中国力学学会,1999:119-123.

- [18] 钟万勰.发展自主 CAE 软件产业的战略探讨[J].计算机辅助工程,2008,17(04):110-115.
- [19] 袁明武,陈璞,郑东,张会杰,石艳华,孙树立,黄吉锋,杨罗宾.微机结构分析通用程序 SAP84 (版本 4. 0) [J].计算结构力学及其应用,1995(03):298-300.
- [20] 王锡山.紫瑞 CAE “傻瓜” 版软件介绍[J].航空工程与维修,2000(02):49-50.
- [21] 刘瑞祥,周建兴,魏华胜,林汉同.开发中国自主知识产权的铸造 CAE 实用软件[J].铸造,2001(11):692-695.
- [22] 胡平,卫教善.冲压成形模具分析软件——KMAS[J].模具制造,2004(06):9-11.
- [23] 陈成军,柳阳,张元章,白小勇,何颖波.基于 PANDA 的并行显式有限元程序开发[J].计算力学学报,2011,28(S1):204-207+214.
- [24] 孙侠生,段世慧,陈焕星.坚持自主创新 实现航空 CAE 软件的产业化发展[J].计算机辅助工程,2010,19(01):1-6.
- [25] 吴圣川,吴玉程.ALOF——新一代三维疲劳裂纹扩展分析软件[J].计算机辅助工程,2011,20(01):136-140.
- [26] 张洪武,陈飙松,李云鹏,张盛,彭海军.面向集成化 CAE 软件开发的 SiPESC 研发工作进展[J].计算机辅助工程,2011,20(02):39-49.
- [27] 张洪武,顾元宪,关振群,亢战,李云鹏,赵国忠.用于有限元分析与优化设计的 JIFEX 软件[J].计算机集成制造系统-CIMS,2003(S1):160-166.
- [28] 梁国平,唐菊珍.有限元分析软件平台 FEFG[J].计算机辅助工程,2011,20(03):92-96.
- [29] 冯志强. 自主 CAE 平台 OmtDesk 及计算软件研发新进展[C]. 中国力学学会产学研工作委员会、中国机械工程学会机械工业自动化分会、中国计算机学会高性能计算专业委员会、陕西省国防科技工业信息化协会.第十届中国 CAE 工程分析技术年会会议论文集.中国力学学会产学研工作委员会、中国机械工程学会机械工业自动化分会、中国计算机学会高性能计算专业委员会、陕西省国防科技工业信息化协会:中国力学学会产学研工作委员会,2014:203-205+227.
- [30] 张芸,秦俊荣,程自力.我国 CAE 产业发展综述[J].中国科技信息,2010(15):115-116.
- [31] 陆仲绩.发展自主 CAE 软件产业的建议与对策[J].计算机辅助工程,2009,18(01):1-4+13.
- [32] 许德新,谈振藩,高延滨.基于 Qt 组件库应用程序的生成及其跨平台实现[J].东北农业大学学报,2006(03):373-376.
- [33] 王帅,蒋林,刘镇弢,郭佳乐.Qt 建模仿真用户界面设计[J].实验室研究与探索,2016,35(07):70-74.

- [34] 袁媛,王延红,江凌,蒋阳.C++类库 Qt 在数值模拟软件开发中的应用[J].现代电子技术,2010,33(02):80-83.
- [35] 陈航. 基于 Qt 和 VTK 的铸造有限元后处理可视化系统研发[D].华中科技大学,2018.
- [36] Marcus D.Hanwell,Kenneth M.Martin,Aashish Chaudhary,Lisa S.Avila. The Visualization Toolkit (VTK): Rewriting the rendering code for modern graphics cards[J]. SoftwareX,2015,1-2:9-12.
- [37] 张晓东,罗火灵.VTK 图形图像开发进阶[M].北京:机械工业出版社,2015.
- [38] Lisa Avila, Katie Osterdahl, Sandy McKenzie,Steve Jordan. The ParaView Guide: Community Edition[M]. Kitware Inc,2019.
- [39] Will Schroeder, Ken Martin, Bill Lorensen. The Visualization Toolkit:An Object-Oriented Approach To 3D Graphics(Edition 4.1)[M]. Kitware Inc,2018.
- [40] Lisa S. Avila, Utkarsh Ayachit, Jeff Baumes,et al. The VTK User's Guide:11th Edition[M]. Kitware Inc,2010.
- [41] W.J. Schroeder,L.S. Avila,W. Hoffman. Visualizing with VTK: a tutorial[J]. IEEE Computer Graphics and Applications,2000,20(5):20-27.

致谢

麓山巍巍，湘水泱泱，宏开学府，济济沧沧。两年前，我来到了这岳麓山下、湘江水旁的湖南大学，怀着对学术的敬畏和对知识的渴望有幸加入到崔向阳教授团队，开启了我的硕士研究生生涯。两年时间一晃而过，如今我即将毕业，踏入新的征途。在这两年的时间里，我获得的不仅仅是知识，更有良师和益友，他们带给我的是开阔的视野、协作的精神以及为人处世的道理，这笔宝贵的财富将伴随我一生。借此机会，在这里对所有帮助我的人表示感谢，文字虽短，但谢意颇深。

感谢我的导师崔向阳教授。崔老师学识渊博，细致耐心，治学严谨，从论文选题到最终成型，给予了我大量的指导意见，倾注了大量心血。崔老师不仅在学术上为我答疑解惑、指点迷津，更为我人生的道路指明方向。在此，特向恩师崔向阳教授致以崇高的敬意和谢意！

感谢柳礼泉、刘陶文、刘继常、谈琦、李蓉、何智成等所有传授过我知识的老师们，他们在教学中表现出的诲人不倦、有教无类、鞠躬尽瘁的优秀品质深深地感染了我。

感谢李射、刘鹏伟、李子超、裴泳杰、杨宏、曹兴刚、田励、田浩、崔振强、洪克城、唐炳娴等师兄师姐们，他们在学术上给予了我非常多的帮助与指导，对本论文的完成提出了大量宝贵意见。感谢张良、刘文杰、王海东、石晟、张洪铭、孙文韬、袁恒君、刘源、孔德华、孙霄等同门和师弟，他们在学习和生活中给予我的帮助。

感谢父母这二十余年的养育之恩，在我背后默默的付出与奉献，让我能够安心完成学业。感谢爷爷、奶奶、妹妹等亲人们的在我困难时给予我的帮助与鼓励，让我有勇气重新战斗。

今年是特殊的一年，一场突如其来的疫情将我们阻隔在家三月有余。作为一个湖北人，我要特别感谢所有奋战在一线的医护人员，感谢所有帮助过湖北的人，感谢学校对湖北籍学子的关怀，感谢党和国家的坚强领导。正是由于他们的努力与付出，我才能够顺利毕业。

无论今后从事什么工作，我都将带着一颗感恩之心，带着实事求是、敢为人先的校训，尽自己之所能回馈社会，报效国家！

汤光泽

2020年5月于湖南大学