上海海运学院

硕士学位论文

利用虚拟现实技术实现集装箱堆场的可视化

姓名：舒帆

申请学位级别：硕士

专业：机械设计及理论

指导教师：宓为建;徐子奇

20031201

摘 要

集装箱堆场作为进出口集装箱装卸过程的一个重要枢纽，是集装箱作业管理中承载业务较多的场所。它管理程度的好坏将直接影响整个码头的作业速度和效率。论文主要通过对虚拟现实技术的研究，来探讨实现堆场高效运作的技术途径。

虚拟现实作为近年来兴起的一门交叉学科，成为当今计算机界广泛关注的一个热点，是20世纪末发展起来的一种可以创建和体验虚拟世界的可视化计算机系统。它主要的优点在于可以直观地观察三维世界，且能让人与此有充分的人机交互，从而产生身临其境的直接感觉。论文通过创建一个三维数据库实时跟踪动态集装箱生产控制系统，不断地获得堆场状态的实际变化，通过计算机屏幕与虚拟堆场进行交互，从而实现集装箱堆场的可视化管理。

本文第一章主要分析了目前堆场可视化系统的发展现状，指出本研究的现实意义和方法，以及所使用的关键技术。

本文第二章主要介绍了堆场可视化系统所涉及主要技术的发展现状。包括虚拟现实技术、计算机图形学，面向对象的软件开发方法以及可视化开发技术等。这些技术为本文研究提供了理论依据。

本文第三章主要论述了本系统在建模方面所作的工作，包括几何建模和数据库建模两大部份。在几何建模方面，重点介绍了Creator的特点和系统构成，并在三维场景的建模过程中具体说明建模技术的应用。在数据库建模方面，遵循本系统预计实现的功能选择数据库相关表的字段，从而实现三维堆场的驱动。

本文第四章主要对集装箱虚拟堆场驱动仿真的流程进行了分析。介绍了虚拟堆场驱动的实现方法，即依托于Vega软件。在Vega的界面环境下进行应用程序的预定义，通过在LynX中定义各种面板参数，对程序的某些属性进行初始化。并通过编程进行二次开发，由于Vega自身并无面向对象的能力，所以需要借助VC.NET的开发平台，调用Vega函数进行应用系统的开发。论文主要研究了数据库访问、Vega的嵌入、保证实时渲染的方法以及与查询和评判功能的整合等问题。

本文第五章主要介绍了应用系统具体的开发过程。介绍了虚拟堆场驱动时涉及的相关内容，重点研究了应用程序的预定义和二次开发中的若干关键问题。

论文最终完成了一个集装箱虚拟堆场的应用系统。系统可以实现根据数据库的实时信息更新三维场景，而且完全由程序自动跟踪完成；可以按照各种查询条件查询在场箱的信息，以便用户能一目了然地观察；也可以根据用户的发箱顺序渲染某一航次的集装箱，利用颜色的不同代表压箱数，从而预知发箱计划的好坏，便于计划人员及时进行调整；用户可以按键盘相应键实现视点的转移，从而与三维场景进行交互。

本应用系统已经在天津港第二港埠有限公司投入使用。实践证明，它对于堆场的实时监控、计划的安排、计划方案的评价等方面起到了很好的作用。同时，以此系统为基础，可在堆场运动模型的建立、辅助方案决策、实时处理手段等方面进行更深入的研究，这也是数字化港口的一个发展方向。

关键词： 虚拟堆场，虚拟现实，建模，驱动仿真，MFC应用程序开发

Visualization of Container Yard

Based upon Visual Reality Technology

ABSTRACT

As one of the important hinges of container loads and unloads process, Container yard is complex department in container operational management because the operation in it is various. Its managerial degree directly influences operation rate and efficiency of whole quay. The technical approach to efficient operational management of container yard through research on visualization of yard has been discussed in this thesis.

As a cross-subject arisen in recent years, Visual reality (VR) is paid widely attention in computer field, is a visual computer system in which people can establish and experience visual world. The greatest advantage of VR lies in that people can directly observe 3-dimensional (3D) world and carry through man-machine alternation. Through establishment of a 3D database real-time track container production dynamic control system, people can get actual change of yard status continuously and alternate information with visual yard on screen, thereby achieving visual management of container yard.

Chapter one of this thesis has mainly analyzed the development of the current visual system of container yard, and pointed out realistic meaning and important technology of the research.

Chapter two has introduced some technical foundation of container yard visual system. It includes visual reality technology, the computer graphics technology, and software development approach facing the target, developing technology of visualization. These technologies offer the theoretical foundation for the research in this thesis.

Chapter three has discussed modeling works of this system, it includes geometrical modeling and database modeling. In aspect of geometrical modeling, the characteristic and systemic formation of modeling software named Creator has

been introduced in detail, and the application of modeling technology has been concretely showed in the process of modeling of 3D scene. In aspect of database modeling, the fields of tables have been selected according to function of visual yard, in order to drive 3D yard.

Chapter four has mainly analyzed the simulation process of visual yard. It has introduced approach to achieve visual yard through software named Vega. In interface environment named Lynx of Vega, users can define all kinds of preferences, initializing some properties of program. For Vega hasn't object-oriented ability, programmer must recur to VC.NET and call function of Vega to carry through secondary exploitation of system. This thesis has mainly researched calling of database, embedding of Vega, real-time displaying of 3D scene and combining between query function and estimate function and so on.

Chapter five has mainly introduced detailed exploitation process of visual yard system and related content of driving visual yard. This segment emphasize the two aspects that predefining of application and key question of secondary exploitation.

Finally, an application system of container visual yard has been achieved, which can automatically render 3D scene according to real-time database information, can query information of containers in yard according to different conditions, can estimate scheme of loading container of export voyage. So, users can observe and control yard expediently, predict effect of process of loading and modulate scheme in time. At one time, users can transfer angle of view through clicking corresponding key on keyboard, thereby alternating with 3D visual scene.

The application system achieved in this thesis has been applied in the second company of Tientsin port already. It has been proved by practice that this visual yard system has many advantages, and some question with regard to modeling of visual scene and real-time process means have been researched more, which are also development direction of digital port.

Shu Fan (Machinery design and theory)
Directed by Mi WeiJian, Xu ZiQi

KEYWORDS: visual yard, visual reality, modeling technology, driving simulation, MFC application program development

第一章　绪　论

1.1 概述

随着集装箱码头作业吞吐量的快速增长，提高码头堆场利用率与提高码头的作业效率成为主要关注的两大问题，但是由于二者之间的矛盾性导致这一工作相对复杂和困难。堆场的管理可以作为一个入口点来研究。堆场作为进出口集装箱的重要枢纽之一，是集装箱码头作业管理中承载业务较多的场所。它包含各种各样性质的进出口箱，因此与之对应的业务类型也很多，一个堆场就可能同时受理出口箱集港、空箱返场、进口箱进场、中转箱转站、移箱、提重箱、提空箱等多种作业。由于每日进出口的箱量之多、作业种类之繁，堆场的管理存在很大的不便，一个箱位的错位很可能导致一系列连环的错误，所以在堆场的箱位分配和优化放置上特别需要借助一种很直观的手段。

正是基于这种考虑，提出了一种将如此复杂的信息直观表达出来的方法——虚拟现实法，它通过仿真的方式给用户创造一个实时反映实体变化与相互作用的三维世界，提供用户一个可观测并与该虚拟世界交互的三维界面，使用户可直接参与并探索仿真对象在所处环境中的作用与变化。

本课题主要是研究如何将实时的数据库信息通过集装箱堆场的 3D 界面表达出来，从而使画面可以实时的反映堆场的现实情况，使计划者和管理者能够一目了然的查看堆场信息。同时嵌入了智能评判系统，对集装箱发箱方案给予自动评判，分别以图形和界面对话框的形式给出评判结果。由于通过三维图直观的显示，符合人类感知系统的应变能力，相应减少了人要通过大脑想象来进行判断的时间，直接缩短了计划员安排预定箱位的速度和准确性，管理者也可对形势做最为直观的判断和总结，便于其对紧急情况及下一步的工作作出决策。

1.2 堆场可视化系统研究现状

由于虚拟现实技术本身的发展历史较短，所以目前已有的堆场管理系统一般并没有采用专门的与虚拟现实技术相关的软件来编写，而且较多的可视化系统多停留在平面可视的状态，所以要去看堆场时，可能看到的是堆场的各个视图，当需要查看箱区的某个具体位置时，要去看某个剖视图，可以说最终对一个箱位的认识是通过多种信息的综合产生的，人们需要不断的在各种视图间切换，这样的直观性显然不好。

目前的三维可视化系统多由某一独立的软件（如：VC++）单独完成，如果需要用到图形时，则调用 Windows 平台中嵌入的 OpenGL 函数来实现。这样做的好处是在单

一系统中进行开发，重点比较突出，但是对于一个可视化要求程度高，而且需要有很高的交互性的系统，这样的图形处理功能仍不够理想。

虚拟现实技术多用于有预定运动路径的物体的移动，诸如说虚拟一个花园，那么可以让花园以人的视点或一个飞机俯瞰的视点来转移，也就是说，可以有多种运动模型，它最突出的优点在于，所描绘的场景非常逼真，而且充分考虑了人机交互。而一个真正的仿真系统拥有这些仍然不够，诸如一个桥吊驾驶训练系统的仿真，司机并不是看到自己的操作环境就够的，还要根据自己的所见来控制自己的行为，这就需要很多过程判断，预先考虑到种种操作的可能性，然后通过高级语言编程来驱动三维模型，让三维模型中的某些部件按照操作的类型作相应的移动，如移动吊具到相应集装箱的定位孔，达到仿真训练的目的。

然而，本系统最终想达到的目的是希望能够将堆场的实际作业描绘出来，即能够反映堆场上集装箱的增减，这就需要一个数据库，其中记录着箱子与箱位的对应信息，由这些实时的信息来控制场景中物体的显示与隐藏，而且这个过程是由程序自动控制的。操作界面的人可以随意移动键盘的相应按键来改变视点及其它一些属性，这样人们在屏幕上就可以观察到堆场的实际情况。虚拟堆场的意义在于可以帮助码头的操作和管理人员看到堆场的运作情况，而且由于虚拟堆场是真实情况的再现，所以给用户的是最直观和准确的信息，帮助他们了解堆场现有的情况，做出相应的决策。

1.3 堆场可视化的意义与研究思路

集装箱码头年 TEU 数是评价一个集装箱码头的主要经济指标，所有的集装箱码头都是向高吞吐量看齐的，然而有时在一个码头的规划中，有限的场地和机械直接制约着吞吐量的提高。机械的增加比较灵活，也是可以在码头的发展过程中不断增加和翻新的，然而，堆场的变化就没那么方便了，只有依靠各种优化策略来提高它的利用率。堆场的利用好坏取决于很多因素，箱子究竟是集中堆放还是分散堆放，集港的箱子按什么原则来堆放，怎样控制船放和直提箱与进场箱的比例等等。这些都要求计划人员能够充分了解堆场上的现行情况，便于分配计划箱位。现在大多数的计划员是以堆场某一列位的剖视图和堆场的俯视图作为参考的，平面图形的直观性相对较差。正是基于这个原因，希望能够将堆场实时的三维图形呈现出来，便于计划人员进行参考。

同时，这样做也便于对堆场进行实时监控，现在有一个较普遍的现象是码头中控室的监控人员经常要跑出中控室，在窗口看堆场上的信息——仅仅是远距离的观看，以弥补监视头观看范围的有限性，如果能够将码头场景直接在计算机中模拟出来，就不用再跑出中控室，而且可以近距离的观察。这样就可以做到随心所欲的观看堆场中的任何一个位置。

为了描述一个堆场的实际情况，保证实时性非常重要，这里采用的是用一个实时的数据库来驱动三维堆场中集装箱的增减，通过时间控制定时访问数据库，从而保证画面的显示能较快的跟踪堆场实际的变化。

课题最终要得到一个实时虚拟系统，它是用 VC 语言利用 VEGA 提供的函数进行二次开发。课题的整个思路是先对堆场进行三维建模；再建立 SQL SERVER 上相关数据库的数据源，即数据库建模；然后在 VC 语言的环境下通过 ODBC 访问数据库得到箱子的变化信息，接着调用 VEGA 函数来控制三维模型中箱子的变化，从而实现了场景的实时渲染。通过各种类的建立，把数据库的扫描结果通过三维场景呈现出来，从而实现了箱位的实时查询。最后针对某指标，对某一航次的多种配载方案进行综合评判，得到该航次的最优配载方案。整个过程如图 1-1 所示：

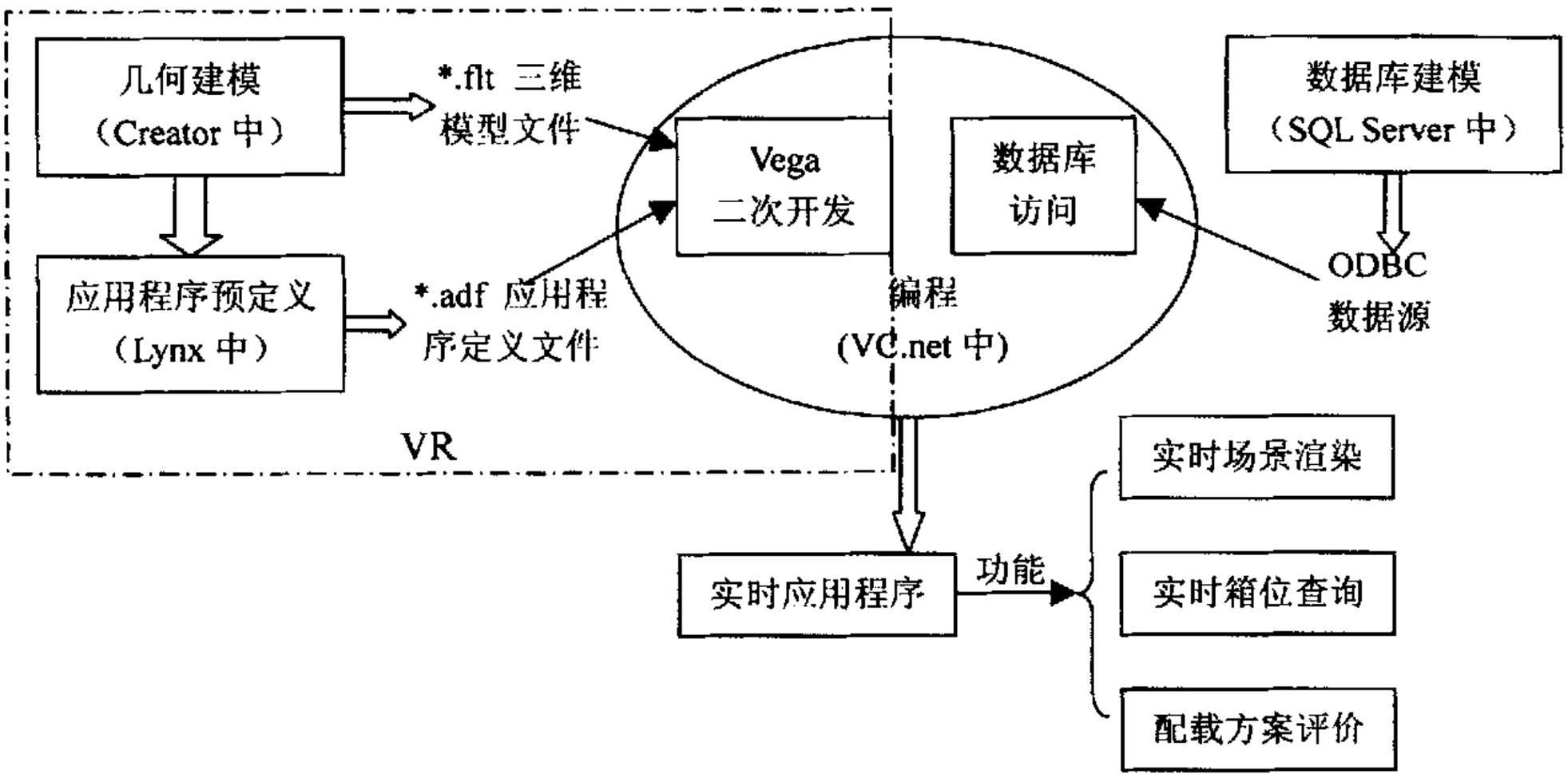


图 1-1　　虚拟现实软件过程分析图

1.4 虚拟堆场构建的技术路线[1 2]

1.4.1 几何建模

考虑集装箱堆场业务和机械的复杂性，在不影响逼真效果的前提下，对模型实体进行一些简化。把集装箱堆场上的装卸和运输机械的三维模型做适当的简化，只在三维场景中表示出来，而不反映它们的运动方式，这也是处于必要性的考虑，因为我们主要是要将堆场上的箱子信息以直观的方式呈现出来，对于机械而言，主要是起装点场景的作用，所以我们把主要的精力放在集装箱的显示上。除此之外，为了逼真的反映堆场周围的环境特点，还会加入一些地形地貌和海洋特征。

建模中我们采用的是 Creator 软件，它拥有逻辑化的层次性景物描述数据结构——OpenFlight 格式。这样图像发生器会根据数据库的结构决定何时绘制、如何绘制、绘制什么，从而生成精确、稳定可靠的三维实时图像。此外，它还可以给几何模型的表面加上材质和纹理，然后设置光照条件就可以生成更加逼真的模型了。整个的几何建模为最终的可视化系统提供了三维效果。

1.4.2 数据库建模

就数据库而言，它是可视化系统实时性的一个重要保证。通过在 PB 上编制码头作业程序，把实时信息源源不断的输入数据库。这就要求对码头作业流程十分熟悉，就卸船作业而言，先由计划员制定卸船计划，输入船图和舱单信息，等到船舶正式靠岸就有了船名、航次信息。当把船图、舱单和航次信息互相综合就能够构成比较完整的对应某一航次的集装箱的箱信息，准确无误的表达出集装箱的当前船上位置和即将流向的场地位置，指引后面的机械作业，当集装箱经过岸吊作业由船卸至集卡，再由场吊作业把集卡上的集装箱按预先计划好的箱位卸至堆场时，场吊司机最终的落位确认就成为箱子最后的准确信息被输入数据库。在建库时，考虑到后面驱动场景工作的需要，主要应包含的数据库信息是箱信息，它应包括船名、航次、箱号、箱型、箱位、尺寸、进出标志、卸货港、目的港、发箱顺序等字段，这样便可以利用箱位与箱号的对应信息来实时渲染三维场景，也能利用这些信息在三维堆场中按船名、航次或目的港来查询和分类显示，并能够通过航次、箱号、发箱顺序之间的关系建立发箱顺序的评判体系。

正是因为基于对一个实时变化的数据库的调用，采用虚拟现实的方法，把它作为驱动三维场景的数据源，不同于以往较多仅仅是用虚拟现实的方法按照一个已知的行为路线和过程驱动场景，而真正做到了实时化的反映，这就是本论题的新意所在。因此这部分的数据资源很宝贵，是实时驱动的基础。

1.4.3 MFC ODBC 访问数据库

现在已经有了三维场景图，以及实时的数据库信息，接下来的工作就是如何将两者有机的结合起来的问题了。利用 VC 强大的库函数调用功能，通过调用 ODBC 类完成与数据库的接合，通过调用 VEGA 类完成与三维场景的接合。

这部分完成的主要工作是与数据库的信息交换，把数据库中的信息变成能够驱动场景图的信息。ODBC 类提供了一组对数据库访问的标准 API。一个基于 ODBC 的应用程序对数据库的操作不依赖于任何 DBMS，所有的数据库操作由对应的 DBMS 的 ODBC 驱动程序完成。ODBC 最大的优点就在于支持对异构数据库的访问，能以统一的方式

处理所有的数据库。利用 ODBC 编制访问数据库的应用程序涉及两方面的数据交换。首先是记录集与数据库之间的数据交换。MFC 提供的 CRrecord 类派生出记录集类，利用 RFX 机制，使记录集中的每一个域数据成员与表中的某一字段相对应。然后是记录集与表单控件之间的数据交换。根据控件的 ID 号，利用 DDX 机制，找出其对应的域数据成员。表单控件实质就是应用程序的组成，由此不难看出记录集是应用程序和数据库之间的一个中转站。

本课题编制程序完成第一部分的数据交换，把数据库中有价值的信息提取出来作为后面驱动场景的依据，实现集装箱的增减变化。第二部分的数据交换将记录集信息在控件中以文字信息的方式显示出来，实现系统中按一定条件查询集装箱信息的要求，同时根据查询结果驱动三维图形相应变化。

1.4.4 实时仿真

上面完成了 VC 对数据库的访问，现在要利用上面的信息完成对生成的三维场景的驱动。首先在三维仿真软件 Lynx 中引入在 Creator 中创建的三维几何模型文件。设置一些窗口参数、系统参数，引入运动对象和场景对象，建立场景。通过加入视点实现多视点的切换；通过加入驱动对象，驱动运动对象或视点按设定的运动模型运动；通过加入干涉对象，当被驱动物体与场景中的其他物体碰撞时，能够检测出物体间的干涉。同时可以设置环境效果和海浪效果，或加入光源，使运动场景更加直观和逼真。进行到这一步，会生成一个应用程序定义文件。

接着的工作是利用 MFC 调用 VEGA 函数库，引入刚才生成的应用程序定义文件，同时利用刚才已经保存在 VC 中的记录集（即数据库信息），通过编程动态的添加和删除集装箱，实现实时的驱动。其中最主要的工作就是建立集装箱类，用来操纵对箱子的查找、增加和删除；以及建立针对 VEGA 的视图类和由其派生出的 MFC 视图类，从而实现所有的信息都能够在三维模型中体现出来。

1.4.5 评价体系的建立

现在系统已经可以显示三维场景中集装箱的变化情况，并且可以按照一定的条件对箱子信息进行查询，到此可以说已经是一个比较完整的系统，可以用来实时监控场景，并可以按照各种条件查询场景中集装箱的属性。为了进一步利用三维显示的优势，这里还将对出口发箱的配载方案按照一定的原则进行评价，评价的结果在三维图形中显示出来，通过不同的颜色信息和统计出的文字信息显示出最优方案。针对一个出口航次的评价因素有很多，如：压箱数、场吊大车的运行时间、场吊同时作业的可能程度等等，但是由于大多数的因素是比较复杂和随机的，同时考虑的难度较大，所以在

这里仅仅对决定方案好坏的主要因素——压箱数进行研究，不同压箱数的集装箱用不同的颜色表示出来，使用户可以一目了然的看出哪个位置的集装箱有压箱现象，压了几个箱等。

小　结：本文所讨论的集装箱堆场的可视化是利用虚拟现实技术，融入数据库访问技术，从而将实时的数据库信息体现在三维模型中，来反映堆场实际的变化情况。本章通过研究大量文献，分析了目前堆场可视化系统的发展现状，指出本研究的现实意义和方法，以及所使用的主要关键技术。

第二章　堆场可视化研究的若干技术基础

堆场可视化系统涉及的技术主要包括：虚拟现实技术、三维图形学技术、面向对象的软件开发方法以及可视化开发方法等。

2.1 虚拟现实技术[3-11]

虚拟现实技术主要包括两个方面的主要内容：建模与仿真。通过建模技术完成三维场景的绘制，通过仿真技术按照一定的原则来驱动三维场景的变化。

2.1.1 虚拟现实技术简介

虚拟现实（Virtual Reality，缩写为 VR）技术，又称灵境技术，是在信息科学的飞速发展中诞生的。它依托于计算机科学、数学、力学、声学、光学、机械学、生物学乃至美学和社会科学等多种学科，在计算机图形学、图像处理与模式识别、智能接口技术、人工智能技术、传感器技术、语音处理与音响技术、网络技术、并行处理技术和高性能计算机系统等信息技术的基础上迅速发展起来。

从概念上讲，虚拟现实是一种由计算机和电子技术创造的新世界，是一个看似真实的模拟环境，通过多种传感设备，用户可根据自己的感觉，使用人的自然技能对虚拟世界中的物体进行考察和操作，参与其中的事件；同时提供视、听、触等直观而又自然的实时感知，并使参与者“沉浸”于模拟环境中。

VR并不是真实的世界，而是一种可交替更迭的环境，人们可以通过计算机的各种媒体进入该环境，并与之交互；从技术上看，VR与各相关技术有着或多或少的相似之处，但在思维方式上，VR已经有了质的飞跃。由于VR是一门系统性技术，所以它不象某一单项技术那样只从一方面考虑问题，它需要将所有组成部分作为一个整体去追求系统整体性能的最优。

2.1.2 虚拟现实的特征

虚拟现实是一种高度逼真地模拟人在自然或特定环境中视、听、动等行为的人机界面技术，具有沉浸性、交互性及构想性等重要特征，最终将成为人机交互的最高形式。

- **沉浸性（Immersion）**

虚拟现实向使用者提供了全身心地进入机制，对于传统的人机界面，人与机器可以交往，但在交往过程中人十分清楚地感觉到自己身处于界面之外。而对于虚拟

现实，人在与机器交往过程中所感觉到的是自己置身于这一特殊的界面（虚拟环境）之中，与这个环境融为一体。

- **交互性（Interaction）**

虚拟现实向使用者提供了交互的机制，使用者的输入（动作）可使呈现其的界面（虚拟场景）发生相应的变化，这一变化将给使用者的感觉产生新的内容；后者导致使用者又做出新的输入（动作），使界面（虚拟环境）再次发生变化；以上交互过程反复进行，直至用户感到所处的虚拟环境满意为止。

- **构想性（Imagination）**

虚拟现实不仅仅是一个用户与计算机的接口，而且可使用户沉浸于此环境中获取新的知识，提高感性和理性认识，从而产生新的构思。这种构思结果输入到系统中去，系统会将处理后的状态实时显示或由传感器反馈给用户。如此反复，这是一个学习——创造——再学习——再创造的过程。因而可以说，VR是启发人的创造性思维的活动。

2.1.3 虚拟现实技术与多媒体、仿真的区别

媒体（medium）指承载信息的载体，多媒体可以理解为多种媒体的综合应用。而我们一般所说的多媒体仅指视觉和听觉媒体的组合，称为狭义多媒体；把包括视、听、触、嗅、味等媒体的全部组合称为广义多媒体。

仿真（simulation）的核心组成部分仅是一个计算、调度的过程。它并不一定需要表现过程，只要通过对模型的计算最后给出一系列的数据即可，这就是数字仿真，但不直观，不易验证仿真程序的对错。若在仿真过程和结果增加文本提示、图形、图像、动画表现，可使仿真过程更直观，结果更容易理解，并能够验证过程的正确性，这种仿真称为可视化仿真。若在此基础上加入声音，就可得到狭义上的多媒体仿真。若再加入三维界面、交互功能、触、嗅、味知觉等，就成了VR仿真系统。

前面对虚拟现实技术已作了一般的介绍，从软件的角度来看，VR系统的软件主要研究虚拟环境及其中物体的几何、物理及行为仿真，只有很好地解决了这个问题，人与系统的自然交互才有可能。因此，也可以将VR系统看作一种仿真系统，但根据系统目的的不同，可将其分为VR仿真系统和VR应用系统两类。前者研究具有不确定性的某一系统的表现特征，重点关注仿真的过程及结果，根据仿真的过程及结果来辅助决策，如VR作战系统。而后者研究解决某种实际问题，只关心结果，如操纵机器人到危险地带执行任务的VR系统。

从上面的分析可以看出，仿真与多媒体、虚拟现实是目的与表现方法的关系。它们三者的关系可以用图2-1来表示：

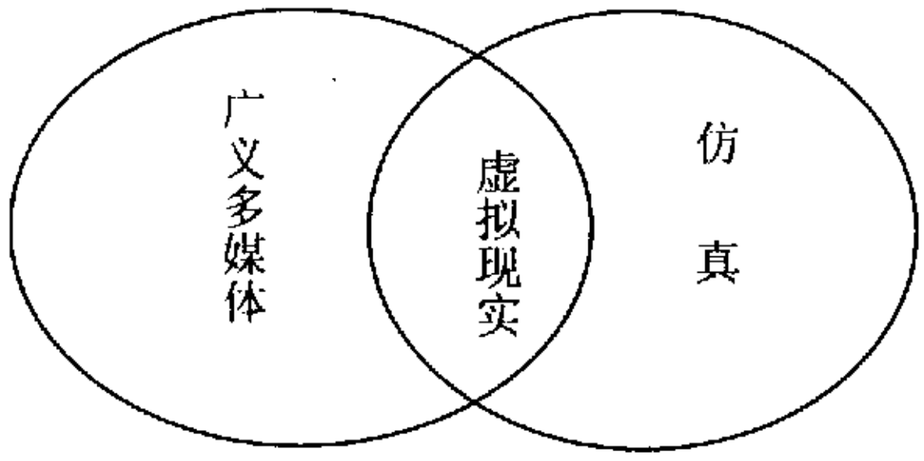


图2-1 三者概念关系图

2.1.4 虚拟现实技术的应用领域

- **军用和民用图形仿真**

采用VR技术进行图形仿真，可以去除传统手工绘图的繁杂劳动，在正式生产之前就可以在系统中进行零部件的组装，可以减少或淘汰传统的物理模型，而且可以模拟产品的工作性能，同时，在显示器上的修改比传统的纸质修改方便得多。如美国的F－18战斗机的重新设计、波音公司的777客机均是在VR系统中完成的设计。

- **科学研究和科学计算可视化**

在虚拟环境中显示和共享数据已成为现实。科学家可以用图形表示复杂的实验数据，通过直观的图形分析数据，可以找到新的信息处理方法。随着VR技术的发展，科学计算可视化系统采用VR接口，促进了科学计算可视化的发展，复杂分子结构和管道布线显示方面就是成功的范例。

- **娱乐业**

利用虚拟现实技术开发旅游业、游戏、虚拟演员等已经取得成绩或正在设想中，通过虚拟现实接口可以将人们带入逼真的虚拟环境中。如英国的W工业公司已研制出虚拟现实游戏设备，配有表示声音和位置的头盔并控制漫游，参与者可以自由控制武器操纵杆。

- **地学**

地球系统的各种现象或过程都可以用VR技术进行模拟。对地球系统结构的模拟，如分子结构、地质构造、大型工程等的模拟；对地球系统的运动现象与过程的模拟，如火山爆发过程、三角洲的演化过程、生态系统的演化过程、城市规划与改造等的模拟；综合开发的虚拟试验。

当然，VR技术在其它行业、学科，如航海、医学、电力工业、电信业、建筑业等的应用也有巨大的潜力和价值。在我国，VR技术的研究相对来说起步较晚，实际应用还不多，市场上VR产品也很少。因此，有必要对此及其基础技术作相关研究以期尽快地将VR技术应用到生产实践中，发挥其优势，为社会生产服务。

2.2 三维图形技术

随着计算机软、硬件突飞猛进的发展，计算机图形学在各个行业的应用也得到了迅速普及和深入。目前计算机图形学已进入三维时代，三维图形技术在走向成熟的过程中，不断地被应用到现实的尖端领域，反过来也不断推动三维图形技术的进一步发展。科学计算可视化、计算机动画、虚拟现实已成为近年来计算机图形学的三大热门话题，而这三大热门话题技术核心均为三维图形。其中涉及的主要技术有：

2.2.1 造型技术

造型（建模）技术是计算机图形学的核心内容，它研究的是如何在计算机中构造出二维、三维物体的模型，并采用合适的数据结构将它用一批数据及相互之间的拓扑关系表示出来。研究如何构造几何模型的理论、方法和技术称为几何造型技术。早期的几何造型系统大都是曲面造型系统，常采用参数曲线（面）、Bézier曲线（面）、B样条曲线（面）等表示形体。近年来，由于非均匀有理B样条方法的成熟，使之被越来越多的几何造型系统所采用。

80年代中期出现的实体造型技术，能较全面地反映物体的信息，如体积、质量等。其中较为活跃的分支是特征造型技术，它将物体表示成一组特征的集合，同一类物体具有相同的特征集，不同的特征值刻画了不同型号的物体。例如，如果用底部半径R和高H作为形状特征来表示圆柱体，那么不同的R和H值则对应不同的圆柱。目前，特征造型技术已在计算机辅助设计系统中获得初步的应用。它通常应用在机械三维建模中。

虚拟现实系统的三维建模通常采用的是多边形建模，因为一个虚拟现实系统最多考虑的是如何实时驱动场景，场景的逼真度是在不影响实时性的前提下做到最优，所以三维场景用多边形代替实体就是处于这样的考虑。

2.2.2 真实感图形绘制技术[12 13]

所谓真实感图形指的是能较逼真地表示真实世界的图形。这种真实感来自于空间中物体的相对位置、相互遮挡关系、由于光线传播产生的明暗过渡的色彩等。真实感图形绘制技术研究的就是如何根据计算机中已构造好的模型，绘制出真实感图形。为了绘制一幅真实感图形，首先要设置光源，模拟光线传播的效果以产生明暗过渡的色彩；还要消除隐藏线、隐藏面，以反映物体间的遮挡关系；投影以产生近大远小的立体效果。

早期的真实感图形绘制技术采用的局部光照模型，仅能反映光线的漫反射和镜面反射现象，而将光线在物体间复杂传播产生的效果笼统地以环境光来表示。20世纪80

年代后，出现了以光线跟踪方法为代表的整体光照模型，它较好地反映了光线在景物表面的反射与折射效果，能够产生真实感程度更高的图形。接着又出现了辐射度方法，它能模拟光能在景物表面间的漫反射传播形式，较好地反映了真实世界中物体之间色彩渗透现象，为生成高度真实感图形提供了可能。除了加光照模型外，可以通过增加物体的纹理和材质属性来增加模型的真实感。

在本虚拟堆场系统中，采用的是多边形建模技术，通过多边形来构建物体，然后通过添加纹理材质属性增加模型的真实感，从而完成虚拟堆场系统的第一步工作——建模工作。

2.3 面向对象的软件开发方法[14-17]

面向对象技术是软件技术的一次革命，在软件开发史上具有里程碑的意义。

随着 OOP（面向对象编程）向 OOD（面向对象设计）和 OOA（面向对象分析）的发展，最终形成面向对象的软件开发方法 OMT(Object Modeling Technique)。这是一种自底向上和自顶向下相结合的方法，而且它以对象建模为基础，从而不仅考虑了输入、输出数据结构，实际上也包含了所有对象的数据结构。不仅如此，面向对象技术在需求分析、可维护性和可靠性这三个软件开发的关键环节和质量指标上有了实质性的突破，彻底地解决了在这些方面存在的严重问题。

- 自底向上的归纳

OMT 的第一步是从问题的陈述入手，构造系统模型。从真实系统导出类的体系，即对象模型包括类的属性，与子类、父类的继承关系，以及类之间的关联。类是具有相似属性和行为的一组具体实例（客观对象）的抽象，父类是若干子类的归纳。因此这是一种自底向上的归纳过程。在自底向上的归纳过程中，为使子类能更合理地继承父类的属性和行为，可能需要自顶向下的修改，从而使整个类体系更加合理。由于这种类体系的构造是从具体到抽象，再从抽象到具体，符合人类的思维规律，因此能更快、更方便地完成任务。在对象模型建立后，很容易在这一基础上再导出动态模型和功能模型。这三个模型一起构成要求解的系统模型。

- 自顶向下的分解

系统模型建立后的工作就是分解。在 OMT 中通常按服务（Service）来分解。服务是具有共同目标的相关功能的集合，如 I/O 处理、图形处理等。这一步的分解通常很明确，而这些子系统的进一步分解因有较具体的系统模型为依据，也相对容易。所以 OMT 能有效地控制模块的复杂性，同时避免了功能分解的困难和不确定性。

- OMT 的基础是对象模型

每个对象类由数据成员（类的属性）和成员函数（类的行为）组成，有关的所有

数据成员（包括输入、输出数据结构）都成了软件开发的依据。在 OMT 中系统边界的改变只是增加或减少一些对象而已，整个系统改动极小。

- 需求分析彻底

需求分析不彻底是软件失败的主要原因之一。即使在目前，这一危险依然存在。传统的软件开发方法不允许在开发过程中用户的需求发生变化，从而导致种种问题。正是出于这一原因，人们提出了原型化方法，推出探索原型、实验原型和进化原型，积极鼓励用户改进需求。在每次改进需求后又形成新的进化原型供用户试用，直到用户基本满意，大大提高了软件的成功率。但是它要求软件开发人员能迅速生成这些原型，这就要求有自动生成代码的工具的支持。

OMT 彻底解决了这一问题。因为需求分析过程已与系统模型的形成过程一致，开发人员与用户的讨论是从用户熟悉的具体实例（实体）开始的。开发人员必须搞清现实系统才能导出系统模型，这就使用户与开发人员之间有了共同的语言，避免了传统需求分析中可能产生的种种问题。

- 可维护性大大改善

在 OMT 之前的软件开发方法都是基于功能分解的。尽管软件工程学在可维护方面做出了极大的努力，使软件的可维护性有较大的改进。但从本质上讲，基于功能分解的软件是不易维护的。因为功能一旦有变化都会使开发的软件系统产生较大的变化，甚至推倒重来。更严重的是，在这种软件系统中，修改是困难的。由于种种原因，即使是微小的修改也可能引入新的错误。所以传统开发方法很可能会引起软件成本增长失控、软件质量得不到保证等一系列严重问题。正是 OMT 才使软件的可维护性有了质的改善。

OMT 的基础是目标系统的对象模型，而不是功能的分解。功能是对象的使用，它依赖于应用的细节，并在开发过程中不断变化。由于对象是客观存在的，因此当需求变化时对象的性质要比对象的使用更为稳定，从而使建立在对象结构上的软件系统也更为稳定。

更重要的是 OMT 彻底解决了软件的可维护性。在面向对象语言中，子类不仅可以继承父类的属性和行为，而且也可以重载父类的某个行为（虚函数）。利用这一特点，我们可以方便地进行功能修改：引入某类的一个子类，对要修改的一些行为（即虚函数或虚方法）进行重载，也就是对它们重新定义。由于不再在原来的程序模块中引入修改，所以彻底解决了软件的可修改性，从而也彻底解决了软件的可维护性。面向对象技术还提高了软件的可靠性和健壮性。

基于 OMT 具有的这些优点，本文所采用的就是面向对象的开发语言，在 VC++的编程环境中建立各种不同的类，如针对集装箱堆场重要的箱子信息建立集装箱类、建

立与 ODBC 数据源连接的记录集类、建立 Vega 视图类等等，从而开发出最后的应用系统。

2.4 可视化开发方法[14-17]

可视化开发是 20 世纪 90 年代软件界最大的两个热点之一。随着图形用户界面的兴起，用户界面在软件系统中所占的比例也越来越大，有的甚至高达 60～70%。产生这一问题的原因是图形界面元素的生成很不方便。为此 Windows 提供了应用程序设计接口 API(Application Programming Interface)，它包含了 600 多个函数，极大地方便了图形用户界面的开发。但是在这批函数中，包含大量的函数参数和数量更多的有关常量，使基于 Windows API 的开发变得相当困难。为此 VC++推出了 Object Windows 编程。它将 API 的各部分用对象类进行封装，提供了大量预定义的类，并为这些类定义了许多成员函数。利用子类对父类的继承性，以及实例对类的函数的引用，应用程序的开发可以省却大量类的定义，省却大量成员函数的定义或只需作少量修改以定义子类。

Object Windows 还提供了许多标准的缺省处理，大大减少了应用程序开发的工作量。但要掌握它们，对非专业人员来说仍是一个沉重的负担。为此人们利用 Windows API 或 VC++的 Object Windows 开发了一批可视开发工具。

可视化开发就是在可视开发工具提供的图形用户界面上，通过操作界面元素，诸如菜单、按钮、对话框、编辑框、单选框、复选框、列表框和滚动条等，由可视开发工具自动生成应用软件。

这类应用软件的工作方式是事件驱动。对每一事件，由系统产生相应的消息，再传递给相应的消息响应函数。这些消息响应函数是由可视开发工具在生成软件时自动装入的。

可视开发工具应提供两大类服务。一类是生成图形用户界面及相关的消息响应函数。通常的方法是先生成基本窗口，并在它的外面以图标形式列出所有其它的界面元素，让开发人员挑选后放入窗口指定位置。在逐一安排界面元素的同时，还可以用鼠标拖动，以使窗口的布局更趋合理。另一类服务是为各种具体的子应用的各个常规执行步骤提供规范窗口，它包括对话框、菜单、列表框、组合框、按钮和编辑框等，以供用户挑选。开发工具还应为所有的选择（事件）提供消息响应函数。

由于要生成与各种应用相关的消息响应函数，因此，可视化开发只能用于相当成熟的应用领域，如目前流行的可视化开发工具基本上用于关系数据库的开发。对一般的应用，目前的可视化开发工具只能提供用户界面的可视化开发。至于消息响应函数（或称脚本），则仍需用通常的高级语言（3GL）编写。只有在数据库领域才提供 4GL，

例如 Microsoft 的 Visual C++开发平台，使消息响应函数的开发大大简化。

可视化开发是软件开发方式上的一场革命，它使软件开发从专业人员的手中解放出来，对缓解上世纪 80 年代中后期爆发的应用软件危机有重大作用。

本课题采用 MFC 编程，可视化方法的使用就贯穿其中，在很多对话框中应用可视控件，通过 VC 提供的消息响应函数来触发这些控件产生一定的行为而实现用户的功能，如本程序中查询类基于的对话框所包含的众多控件就实现了对在场箱查询的功能。

小　结：本章介绍了堆场可视化系统涉及的关键技术及其发展现状。包括虚拟现实技术、计算机图形学，面向对象的软件开发方法以及可视化开发技术等。虚拟现实技术是本课题的灵魂所在，正是借助于虚拟现实的思想才使得各种信息——图形信息、数据库信息能较好的集成在一起。计算机图形学技术提供了建模工作的思路。面向对象和可视化开发方法是本课题重点依托的手段，最后得到的应用系统正是基于这些方法来编写的。

第三章　集装箱虚拟堆场的模型建立

集装箱虚拟堆场的关键思想是虚拟现实方法。但在具体的应用中，还必须找到实现它的强有力的工具。近年来曾出现多种工具和软件，如OpenGL、3Dmax、WTK等。而实时三维建模和仿真软件MultiGen Creator 和Vega，由于其先进的功能，在各个工程领域的视景仿真开发中得以广泛采用。

本章将讨论Creator建模技术及其特点，并介绍虚拟堆场中各种场景模型的创建情况，即几何建模；接着讨论数据库建模。这两大部份的建模工作是后面三维场景驱动的基础。

3.1 MultiGen Creator 建模工具概述[18-20]

3.1.1 MultiGen Creator 建模技术

MultiGen Creator 是 MultiGen Paradigm 公司最新推出的一套高逼真度、最佳优化的实时三维建模工具，它能够满足视景仿真、交互式游戏开发、城市仿真以及其它的应用领域。MultiGen Creator 是唯一将多边形建模、矢量建模和地形生成集成在一个软件包中的手动建模工具，能进行矢量编辑和建模、地形表面生成。

MultiGen Creator 包括一套综合强大的建模工具，具有精简、直观的交互能力。工作在所见即所得、三维、实时的环境中，能够让用户看到在数据库的什么地方发生了什么事情。针对要完成的任务，用户总能找到所需的工具或使用自定义的工具箱。

MultiGen Creator 强大的工具核心为 25 种不同的图像生成器提供自己的建模系统和定制的功能。先进的实时功能如细节等级、多边形删减、逻辑删减、绘制优先级、分离平面等是 OpenFlight 成为最受欢迎的实时三维图像格式的几个原因，这种数据格式已成为视景仿真领域事实上的行业标准。许多重要的 VR 开发环境都与它兼容。

MultiGen Creator 的建模环境提供同时交互的、多重显示和用户定义的三维图形观察器和一个有二维层次的结构图。所有的显示是交互的和充分关联的。这种灵便的组合加速了数据库的组织、模型生成、修改编辑、赋予属性和结构关系的定义。

3.1.2 MultiGen Creator 的构成

MultiGen Creator 软件区别于机械 CAD 等其他建模软件，主要考虑在满足实时性的前提下如何生成面向仿真的，逼真性好的大面积场景。

Creator 是唯一将多边形建模、矢量建模和地形生成集成在一个软件包中的手动

建模工具，它给使用者带来了不可思议的高效和生产力。Creator 不仅能够完美地生成船舶、地面交通工具、建筑物等单一模型，而且也能够生成人们所感兴趣的特征区域，如：码头、港口、城市、工厂等。

➢ 多边形和纹理建模

利用 Creator 交互式、直观的用户界面进行多边形建模和纹理贴图，使用户能够很快生成一个高逼真度的模型，并且它所创建的 3D 模型能够在实时过程中随意进行优化。Creator 提供的转换工具，能够将多种 CAD 或动画软件模型转换成 Creator 所支持的 OpenFlight 格式。包括如下功能：

- 几何多边形的生成与编辑
- 纹理的编辑与应用
- 层次结构的生成与编辑
- 光点生成
- 仪表的生成
- 数据库的自动组织工具
- 支持外部的参考、例子、调色板的继承
- OpenFlight 的数据库的查询、选择和修改功能

➢ 矢量编辑和建模

在 Creator 中能够输入类似地图矢量数据，高效地生成、编辑用户所感兴趣的模拟区域，并且地形表面的纹理、色彩模型都能够自动放置。真正的所见即所得交互式模型编辑环境可以帮助用户了解 3D 模型是如何生成的。通过矢量数据生成区域，再经过 Creator 的多重渲染，就可以生成与现实很相似的场景，从而大大地减少开发的工作量。同时，使用 Creator 的矢量工具，可以将一个个单独的 OpenFlight 文件放置在任何场景中。提供的功能包括：

- 不必通过多边形建模即可交互地生成文化物体
- 支持的标准数据格式有 NIMA DFAD(Digital Feature Analysis Data)、USGS DLG(Digital Line Graph)
- 可以通过标准的 Creator 编辑系统对数据进行编辑修改
- 矢量数据库的查询、选择和修改功能
- 可以在限定的范围内随机放置某些物体
- 通过自动的光点生成模拟城市、乡村、街道的灯光
- 快速准确地放置 3D 物体在地形表面

➢ 地形表面生成

所有的模型都需要放置在一个特定的地形表面。Creator 提供了一套完整的工具，能够依据一些标准数据源快速、准确地生成地形。它所支持的标准数据格式包括：USGS、NIMA 和图像转换生成的数据。Creator 所生成的地形已经包含了 LOD 分层及截取（Culling）组的层次结构，这样就可以让您针对不同的硬件平台建立不同的地形分辨率。

- 采用标准的 DTED(Digital Terrain Elevation Data)、USGS DEM(Digital Elevation Data)数据
- 多重 LOD 的自动生成
- 交互式 Delaunay 算法三角化
- 截取（Culling）组自动生成
- 多边形网格和 Delaunay 算法
- 纹理和色彩都能够生成
- 用户自定义截取和 LOD 块的大小
- 准确地探测山脊、谷底和海岸线
- 用户自定义地形表面的误差和多边形数目
- 支持平面投影和固体地球表面算法

3.2 MultiGen Creator 中的一些关键技术

MultiGen Creator 作为一个业界领先的三维建模软件，包含很多先进的技术：层次结构视图、纹理映射技术、实例化技术、层次细节技术等。层次结构视图使数据库模型非常直观、逻辑层次分明，方便了建模工作；纹理映射技术不仅可以大大减少渲染的多边形数目，而且在某种程度上可增强模型的真实性；实例化技术可以减少模型占用的磁盘空间，从而提高运算速度；层次细节技术可以减少渲染的多边形数目，以提高场景的渲染速度。下面详细介绍这些技术：

3.2.1 层次结构视图（Hierarchy View）

三维建模是视景、灵境系统设计的核心问题之一，而三维数据库又是整个建模的基础。一个场景中对象模型的组织结构(在 Creator 中称为场景数据库)对视景系统的运行质量有极大的影响。适当的组织结构(数据库结构)是用户创建满足自身需要模型的关键技术。目前，树状层次结构被广泛用作模型的组织结构。在描绘一个基于真实地形数据的大型虚拟环境的同时，往往会涉及到虚拟场景中各实体的具体结构和详细状态。因此，在建立这个庞大的场景之前，应该根据虚拟场景中每个实体的几何空间

位置，以及模型间的结构关系，确定虚拟场景中所有实体模型的树状层次结构。这种分层结构可以使用自顶向下的方法将一个模型对象分解，也可以使用自底向上的构造方法重构一个模型对象。

建模的原则为：(1)确立这个模型的最终目的(即要达到的程度，需要用到的技术)；(2)优化目标实时模型系统(如限制软硬件平台、颜色、多边形数目、材质、光源和纹理等参数)；(3)模型系统的背景要简单、真实；(4)提高模型系统中重要部分的精度.在建模过程中，即使是最简单的模型，也应该调整层次结构视图，而达到优化的目的。

OpenFlight 为 MultiGen 的层次结构视图的数据格式，该数据格式是 MultiGen 的根基。OpenFlight 数据库的树状结构以二进制文件方式存储，文件由一系列线性记录组成。记录的前两个区域是解析信息，定义了记录的操作码和长度，每个记录的最大长度为 64kb。文件包含了 3 种类型的记录： 节点原始记录、辅助记录和控制记录。记录的长度随着记录项的增加而增加，而且所有的记录大小必须是 4 的倍数个字节。树的层次结构由控制记录来标识，层次结构中的每个节点都由一个主节点记录和 0 个或多个辅助记录来表示。主节点记录标识一个节点的类型，并包含大部分该节点属性数据。余下的节点属性，例如注释、ID、平移变换等都存储在辅助记录中。辅助记录紧跟在主节点记录之后，控制记录之前。

OpenFlight 是逻辑化的有层次的视景描述数据库，用来通知图像生成器何时以及如何渲染实时三维场景，非常精确可靠。开放的连接及简易交互式的操作与细节等级、多边形筛选、逻辑筛选、绘制优先级、二进制分离面等先进的实时功能一起使得它成为实时三维视景仿真领域的工业标准。

它是一个分层的数据结构，它用层次和属性来描述三维物体。它采用层次结构对物体进行描述，可以保证对物体顶点、面的控制，允许对集合层的数据进行直接操作，使建模过程快捷方便。它包含五个层次，分别为：

1）**OpenFlight 数据库头层次（Header level）**： OpenFlight 使用几何体，层次结构和属性来描述三维物体。它可以： 配置控制系统和数据库建造历史，包括外部引用的路径名，模型各部分和几何造型定义、位置和大小。

2）**集合层次（Group Level）**：以逻辑组的形式组织和定义模型的各组件，用于总体模型的建造、动画和实时渲染 。其扩展的功能有：

● 外部引用：分布在不同文件中的所有模型，通过数据库关系进行快速的视景装配。允许用户直接在其它数据库中进行重新定位。

● 等级细节（LOD）：为减少实时渲染的开销，按距离范围切换模型的不同版本，达到减少多边形的目的。它用于管理实时系统的显示负载。在远距离观察时，物体表示非常简单，但随着眼睛移近物体，就会不断增加物体的复杂性。

● 自由度（DOF）：为实时动画序列定义部件的绞链关节和运动范围，如模型的移动和旋转。

● 切换：为动态场景的变化而设计的功能，如特殊效果和运输工具的损坏。

3）**物体层次（Object Level）**：提供更好的结构细节、组织个别部件的删减、模型面的实时渲染。其扩展的功能有：

● 文本：把 3D/2D 和静态/动态的文本放在仪表显示中

● 光源：定义光源的类型、位置和方向

● 声音：定义和附加声音（Waveforms）到动态三维的物体

4）**表面层次（Face Level）**：

● 为渲染提供更细致的控制而定义和组织的表面和属性

● 公告板选择标记

● 定义表面渲染的选项：色彩和纹理

● 定义明暗模式，包括平坦、光滑和顶点色彩

● 定义材质：放射的、光谱的、周边的、扩散的、有光泽的和带有透明度的

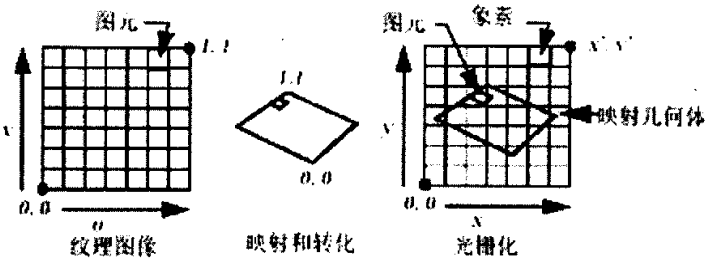
5）**顶点层次（Vertex Level）**：

● 组织和定义数据库中几何造型最细的一级，提供对顶点的位置、色彩、纹理映射和光线矢量和绝对控制。

3.2.2 纹理映射技术

纹理映射技术在三维图形中用得很广。比如绘制一面砖墙，就可以用一幅真实的砖墙图案或照片作为纹理贴在一个矩形上，一面逼真的砖墙就画好了。如果不使用纹理映射的方法，则墙上的每一块砖都必须作为一个独立的多边形来画。因此，使用纹理映射技术可以避免对场景的每个细节都使用多边形来表示，进而可以大大减少场景模型的多边形数目，提高图形显示速度。由于透视变换，纹理提供了良好的三维线索，因此，纹理还可以增加景物的真实感。

纹理是一个 2D 图像，它被数字化成颜色数据元素矩阵，并映射到 3D 形状上，最后投射到屏幕上。它是一种将二维图像映射到一个几何图形上并产生特殊效果或真实感的一种技术，它并不是实际的几何模型。纹理图像被定义成 u 、v 坐标平面，这个平面被映射到一个几何模型的 x 、 y 、 z 坐标，当 3D 模型转换和投影到屏幕时，映射后的纹理也被旋转和改变大小，并在屏幕上绘制出来，就好似位于模型的表面上。



Each texel is mapped to a geometric location on a transformed polygon. The underlying screen pixels are colored to match a blend of the overlaid texel(s) and the face color.

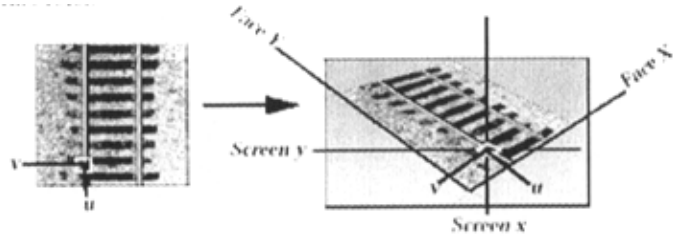


图 3-1　纹理映射过程

纹理映射过程见图 3-1：(1)选择或确定当前纹理；(2)将映射纹理 u 、v 坐标变换为几何坐标；(3)调整面上图像的颜色数据和阴影数据；(4)应用过滤器消除由象素到图元间不正常效果。每个图元被映射到变换后的多边形上一个几何位置，位于下面的屏幕像素是通过表面的图元颜色和面颜色的混合颜色来着色的。

与纹理映射相关的是应时刻注意纹理消耗的内存空间大小。例如：Creator 可以支持大于 4096×4096 图元的纹理，但是多数图像产生器都限制纹理大小和所占的内存。在创建模型纹理库之前，要考虑目标系统的纹理限制是很重要的，计算纹理需求内存容量的一个典型公式是：

$$内存容量=x\times y\times c \qquad (3\text{-}1)$$

式（3-1）中： x 和 y 分别是纹理图像在 X 轴和 Y 轴方向上的图元数，C 是颜色的位数。为了更充分地利用纹理内存，并且阻止来自某些图形产生器的副效果，禁止使用单维数来“保存”纹理空间，纹理维数应该是 2 的 n 次方数。

3.2.3 实例化技术

虚拟场景模型对象的表示一般分为两种：(1)具体地表示对象中基元的轮廓和形状，这可以节省生成时的计算时间，但存储和访问所需要的时间比较多，空间比较大；

(2)抽象地表示，它有利于存储，但使用时需要重新计算。采用哪一种方法表示取决于对存储和计算开销的综合考虑，一般情况下都采用具体的表示方法，但是有时抽象表示会大大提高系统的性能。Creator 中的实例化技术便是抽象表示对象的一个典型应用。

当三维复杂模型中具有多个几何形状相同但是位置不同的物体时，可采用实例化技术。例如，一个动态地形地貌场景有很多结构、形状、纹理相同的树木，树木之间的差别仅在于其位置、大小、方向的不同，如果把每棵树都放入内存，将造成极大的浪费；若采用内存实例的方法，将相同的树木只在内存中存放一份实例，再将这棵树平移、旋转、缩放之后即可得到所有结构相同的树木，从而大大地节约了内存空间。

实例化技术的处理方法为矩阵变换，它牺牲了时间，换得了内存空间。三维空间中物体的几何变换矩阵可用 T_{3D} 表示：

$$T_{3D}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{3-2}$$

式（3-2）中，T_{3D} 在变换功能上被分为 4 个子矩阵，其中：$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ 产生比例、旋转等几何变换；$\begin{bmatrix} a_{41} & a_{42} & a_{43} \end{bmatrix}$ 产生平移变换；$\begin{bmatrix} a_{14} \\ a_{24} \\ a_{34} \end{bmatrix}$ 产生投影变换；$\begin{bmatrix} a_{44} \end{bmatrix}$ 产生整体比例变换。由此很容易推导出每种变化矩阵。

（1）**平移变换**：若对象的位置为（x，y，z），则变换矩阵为

$$\begin{bmatrix} x^* & y^* & z^* & 1 \end{bmatrix}=\begin{bmatrix} x & y & z & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}=\begin{bmatrix} x+T_x & y+T_y & z+T_z & 1 \end{bmatrix} \tag{3-3}$$

式（3-3）中：T_x，T_y，T_z 分别是目标在 3 个轴方向的平移量

（2）**比例变换**：若比例变换的参考点为(x_f，y_f，z_f)，其变换矩阵为

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_f & -y_f & -z_f & 1 \end{bmatrix}\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_f & y_f & z_f & 1 \end{bmatrix}=\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ (1-s_x)x_f & (1-s_y)y_f & (1-s_z)z_f & 1 \end{bmatrix} \quad (3\text{-}4)$$

式（3-4）中：s_x，s_y，s_z分别为目标在3个轴方向上的缩放比例因数。

（3）**旋转变换**：在右手坐标系下相对坐标系原点绕坐标轴旋转θ角的变换公式是

（a）绕x轴旋转

$$[x^* \quad y^* \quad z^* \quad 1]=[x \quad y \quad z \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3\text{-}5)$$

（b）绕y轴旋转

$$[x^* \quad y^* \quad z^* \quad 1]=[x \quad y \quad z \quad 1]\begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3\text{-}6)$$

（c）绕z轴旋转

$$[x^* \quad y^* \quad z^* \quad 1]=[x \quad y \quad z \quad 1]\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3\text{-}7)$$

在Creator中，实例化的这种变换矩阵分别被存储在模型层次结构的实例化节点的辅助记录中。当显示这些实例化模型时，只需将原对象的各个信息乘以各自的变换矩阵及其逆矩阵即可。因此，当实例化对象增多时，系统的运算量明显增大，但是它节省了大量的内存，这种技术在分布式仿真中使用会大大减小数据的传输量。

3.2.4 层次细节（LOD）技术[21-22]

LOD是细节层次模型(Levels of Detail)的英文简称，其主要思想是为虚拟环境

中的地形和建筑物构造多层次的细节模型。 对于同一块地形或建筑物，根据其在虚拟场景中的距离、视觉特性等规则，构造一组可显示不同多面体数量的三维模型。在模型驱动时根据所制定的规则，自动选择相应的显示层次，从而达到实时简化模型，而又不影响视觉效果的目的。如对于一个具有二层 LOD 的模型，在实时仿真系统中，如果视点远离物体，它将显示为一个低细节层次、具有较少多面体的模型；当视点逐渐移近物体时，实时仿真系统将显示为一个高细节层次的模型。由于实时仿真系统所能显示的最大多面体数目是有限的，它与图形卡性能以及内存有关，利用 LOD 可以让你在近距离看到非常细致的三维模型图，而在远距离只能看到轮廓图形。因而，用 LOD 技术构建的层次细节模型结构，可以让我们针对不同的硬件平台和视觉特性建立不同的地形或建筑物分辨率。

LOD 简化多边形的目的，不是为了从初始模型中移去粗糙的部分，而是为了保留重要的视觉特征来生成简化的模型，其理想的结果应是一个初始模型序列的简化，这样简化模型才可以应用于不同的实时加速。为了实现连续帧的绘制，LOD 简化方法必须均衡考虑模型的细节度和绘制速率。

LOD 简化算法需要某种启发式准则，以选择相关的近似误差估计来控制简化，下面讨论简化中应考虑的各种条件准则。

- **LOD 连续性**

利用 LOD 进行实时绘制时，不同 LOD 模型转换时所引起的图像跳跃，是简化中急需解决的问题，因此在实时绘制时，需要从 LOD 简化序列中选择合适的 LOD 模型。如果连续的 LOD 模型之间相差仅一个或两个多边形，则转化时可保持模型的视觉连续性。任意网格多分辨率简化算法和递增网格法都提出了连续 LOD 模型生成的数据结构，它们虽使得不同 LOD 模型转化时能光滑改变，但由于实际应用中，要考虑存储和计算需求，仅生成有限层次细节度的 LOD 模型，且不同层次细节间的多边形数差异较大，转化时会引起图像的跳跃，因此层次细节光滑技术在简化中显得至关重要。

- **形状保留**

由于简化算法必须尽可能保留模型的形状和表面特征，因此算法必须首先找出模型的特征信息(如平面曲率、尖点和特征边)，然后才能通过融合平坦的区域和线性变化的特征边来简化模型。如今，大多数算法是用特征边折叠来简化模型或采用合并曲率较小的相邻面来简化模型，并通过门限值来控制简化。这种合并相邻面的方法，门限值可定义为相邻平面法向量的角度值，即超过门限值的平面不予合并，且门限值越大，简化程度越高。

➢ **近似误差估计**

为了控制简化过程，每一步简化均应评价模型局部的近似误差，如有的算法使用局部的误差估计或者距离准则（顶点到平面的距离）来评判简化误差；而有的算法则采用几何体来限制简化程度。

LOD 简化算法中，目前需要达到的目的有：(1)LOD 模型的自动生成；(2)LOD 模型转换时的图像跳跃；(3)同一场景不同的层次细节区域间如何保持网格的连续性，而不出现裂缝；(4)简化中，如何保留模型的表面特征信息。这些功能在 MultiGen Creator 中均已得到很好实现。

3.3 虚拟堆场的几何建模 [23 24 25]

几何模型的描述与建立是计算机图形学中重要的研究领域。首先，在计算机中建立起三维几何模型。在给定观察点和观察方向后，使用计算机的硬件功能，实现消隐、光照以及投影这一成像的全过程，从而产生几何模型的图像。几何对象的几何模型描述了虚拟对象的形状和它们的外观(纹理、颜色、表面反射系数等)。几何模型具有两个信息，一个包含点的位置信息，另一个是它的拓扑结构信息，用来说明这些点之间的连接。

3.3.1 堆场可视化系统模型数据库的建立

在虚拟堆场的场景建模中，由于堆场本身的场景不大，而且为了直观细致的观察箱子的变化情况，观察者一般距离场景很近，因为这个原因，这里的建模一般不考虑 LOD（细节等级）的问题。本虚拟堆场最终要达到的目的是能够跟踪箱子的变化情况、并能够查询箱子的信息以及根据发箱顺序来决定配载方案的优劣。所以建模工作相对简单，重点放在箱子的自动生成上，其他的场景建模仅仅作为辅助的信息来帮助场景更加逼真和有可观测性。根据这种实际的需要，虚拟堆场包括下列模型：

1)静态模型：这一般是指在场景中静止的模型，主要起到增加模型真实感的功能。诸如：桥吊、集装箱船、集装箱卡车、岸上的楼房等。

2)自然景观：这部分模型也是静止的，诸如：地面、海面、公路、绿化设施以及周边的环境设施等。

3)动态模型：这里的动态模型是指集装箱。动态和静态的区别在于在堆场上有无增减的变化。堆场上随时有集装箱的增减，这是由码头的具体作业决定的，由数据库信息来控制的，所以把集装箱当作动态模型。当人的观察角度有所改变时，整个模型全部动起来，整个场景在这种情况下就变成动态的了。

3.3.2 堆场可视化系统建模流程

在堆场可视化系统建模之前，必须搜集各方面的相关资料，接着应该宏观观测整个场景，弄清如何布局整个场景。建模过程大致包括：建模资料准备、纹理图片处理、三维模型建立、场景数据库整合与优化。

1）建模资料准备

建模资料准备主要是要获取堆场的布局情况和其中各种三维模型的尺寸，各个港口的堆场形式各有不同，本虚拟堆场主要是按照天津港第二港埠有限公司集装箱码头来模拟的，它的堆场平面图如图 3-2 所示：

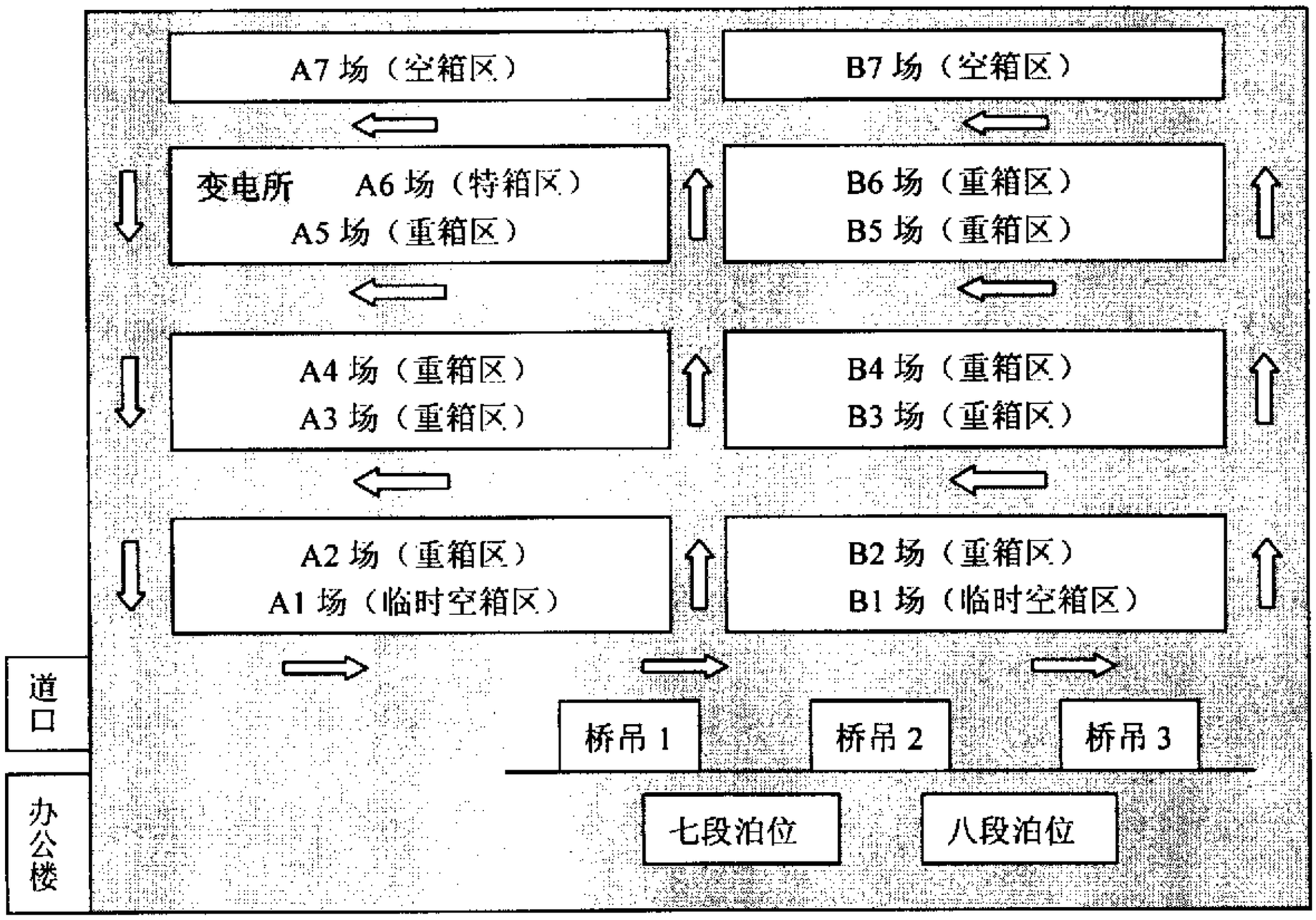


图 3-2　集装箱泊位堆场平面图

通过这张平面图就可以大致掌握堆场建模时的方位安排。在这个阶段还要注意搜集各种建筑物的三视图以及三视角正向照片等，对于较复杂的结构，如桥吊、集卡则要有比较细致的尺寸数据。

2）纹理图片处理

在港口实地用数码相机拍摄各种建筑物的三视图及各式照片，这些图片必须经过处理才能作为三维物体的纹理使用。比如必须将图片的长、宽像素变成 2 的次方。一般是运用 PhotoShop 或 Creator 中自带的图片处理软件进行修正。

3）三维模型建立

在场景数据库中建立各种模型，包括地面、海面、桥吊、集卡、办公楼、绿化以及集装箱等。

4）场景数据库整合与优化

各个场景模型可作为单独的文件被创建，但是出于对后面驱动方便性的考虑，将各种模型进行适当的组合。组合的原则就是依据前面所说的模型的静、动类型。把所有的静态模型组合在一个文件中，而把动态模型即集装箱作为单一的文件存储，如果做到集装箱纹理与到场的实际箱完全一样，那么就要创建所有可能的纹理类型，这样做的工作量很大，考虑到堆场集装箱样式的这种多样性，仅仅建立一个模型文件可视效果是不佳的，但是建立拥有所有纹理的模型又不实际，所以这里采取一种折中的方法，建立三个集装箱模型文件，它们之间的区别仅仅在于纹理的不同。所谓建模为驱动的方便性考虑，主要也是针对所创建模型文件的流向问题考虑的。模型文件通过两种关系加到最后的应用系统中，大致的关系如图 3-3 所示：

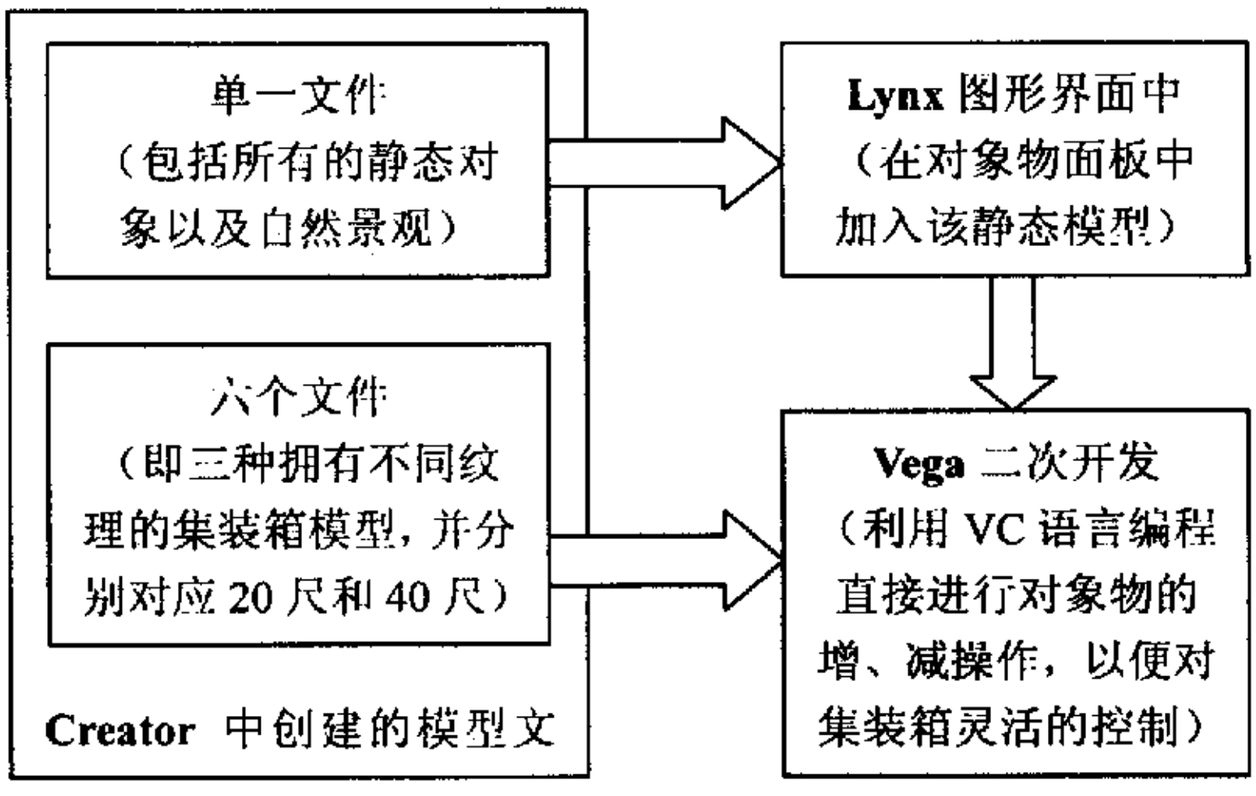


图 3-3　　模型文件的流向关系

建模之后，模型到具体的应用程序中可能发生变形，或者绘制顺序出错，或者运行比较缓慢，这时要对几何面和结构节点进行优化调整，选用正确的数据库结构或者重新组合各个节点、删除一些不可见面等。

3.4 Creator 在虚拟堆场建模中的运用[26 27 28]

整个模型数据库的建立都是在 Creator 软件中完成的，该软件最大的优点在于它能够充分考虑实时性的需要，用最简单的信息表达复杂的结构，而且它能够有效的借助光源、材质、纹理等加强场景的逼真度而不影响实时的渲染速度。

3.4.1 总体布局

数据库的总体布局是首要的任务，主要是要确定好场景数据库的原点以及三个坐标的方向。单独的七个模型文件（一个静态模型的总合文件和六个集装箱文件）有自己的局部坐标，当最后集成在一起时，就要通过应用系统的绝对坐标来统一。

Creator 中默认的坐标方向是 x 轴向右，y 轴向里，z 轴向上。在六个模型文件中，以集装箱的左下角作为原点（如图 3-4）。因此，为了方向上的统一，也约定沿集装箱的长度方向作为整个静态模型文件的 y 轴，根据右手准则也就确定出了 x 轴（如图 3-5）。

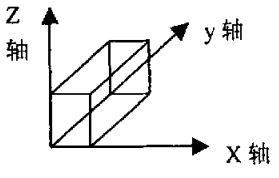


图 3-4　集装箱局部坐标

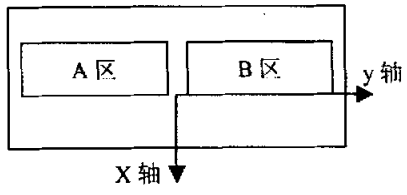


图 3-5　静态场区局部坐标

最终，应用程序的绝对坐标原点可以取在任何位置，只要调整好局部坐标与绝对坐标的相对位置就可以保证场景准确的渲染，为了今后编程算法上的简便，这里就把应用程序的绝对坐标原点取在静态场区局部坐标的原点上。

3.4.2 静止场景建模

这里所谓的静止是指在场景中无增、减变化的三维模型，能够集中有效地反映一个港口的建筑特色，主要起到增加模型真实感的功能。诸如：桥吊、集装箱船、集装箱卡车、岸上的楼房等。

➢ 岸上楼房建模

岸上的楼房主要是指集装箱作业部的办公楼、变电站以及集装箱进离堆场时的必经关口——检查桥。Creator 中采用的方法是多边形建模，因此即使是一个实体的外形，它实际上也是由若干个面组成的，这样做的好处是占用内存较少，利于应用程序高速渲染的要求，这也是实时虚拟与三维动画的主要不同之一。所以，对于一个外形简单的房屋，如类似于一个长方体，那么只要建立 6 个面即可，甚至可以把底面删除，保留可见的五个面，而真实感是通过添加纹理来实现的。Creator 中提供了很多建模方法，可以先画出一个长方形，然后通过拉伸生成出长方体（如图 3-6）。

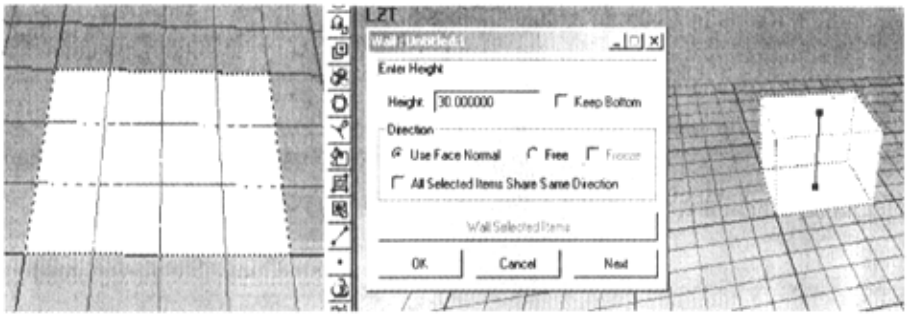


图 3-6　多边形建模

生成的长方体现在是不具有实际意义的，所以最后模型需要加上纹理，以显示模型的真实感。Creator 中提供很多纹理粘贴技术，这里利用三点放置法来放置各面的纹理，使长方体看上去是房子的外观（如图 3-7）。

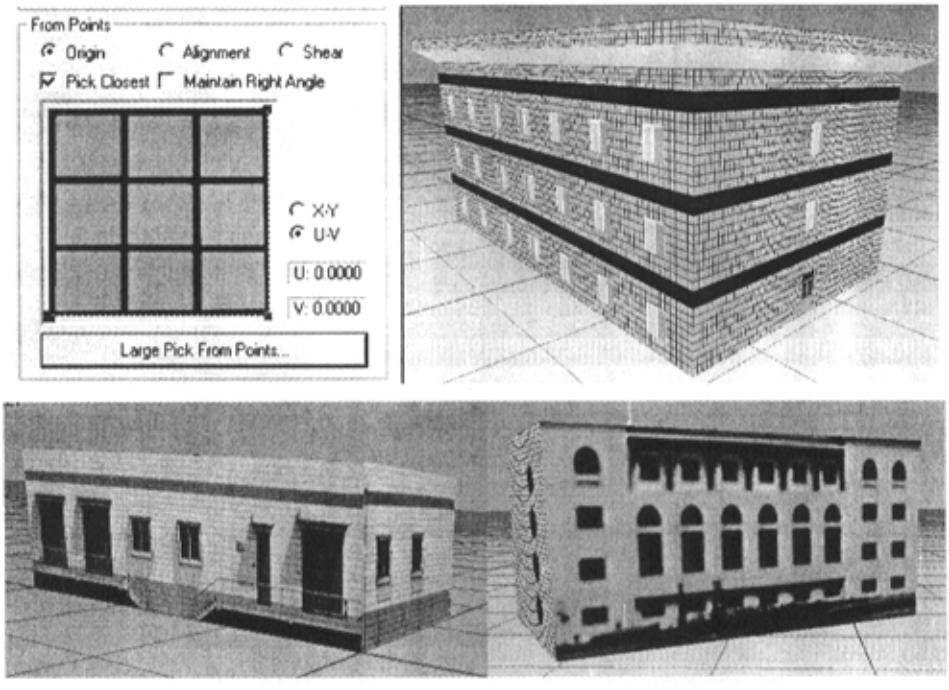


图 3-7　真实感房屋模型

用 Creator 建模要遵循几条基本的原则，其一应建立凸多边形，凹多边形会影响实时显示的效果，出现凹多边形时应把它分解成若干个凸多边形；其二不要使两个面产生覆盖现象，因为当两个面的深度值一样，场景渲染时就会无法确定覆盖部分的显示，出现扭曲等现象，当两个面确实需要重叠时，应该采用一定的处理手段，可以用

子面的方法，也就是说把一个面作为另一个面的子面，即两个面在数据库中不是同级关系，而是上下关系，或者就把相互覆盖的两个面作为几个面来建模以消除覆盖现象。例如图 3-7 中第一个房屋，它的某个面的纹理不是用一张图片来获取的，墙壁、门、窗有各自的纹理，这里采用的就是生成子面的技术，先构造整个墙壁面，贴上墙壁的纹理，然后以墙壁为基面，画出与门、窗等大小的面，贴上门窗的纹理。

对于检查桥，它的建模也比较简单。利用 Creator 的层次结构视图，可以建立新的组节点（group）和对象物节点（object），然后重新组合各个面，以便归属于相应的节点。最后的结果是把检查桥的模型分成：顶、立柱、房屋三部分。立柱、检查桥顶部和房屋都采用拉伸建模的方法，要注意的就是在建立基础多边形的时候，防止出现凹多边形。因为房屋形式的单一性，所以这里采用复制再移动的方法，建立多个结构相同的房屋，Creator 提供许多修改几何体的工具，可以方便的移动、旋转和缩放几何体。为了节省内存资源，还提供了转换矩阵的方法，这是实例化技术的具体应用。图 3-8 是检查桥的模型以及与之对应的数据库结构。

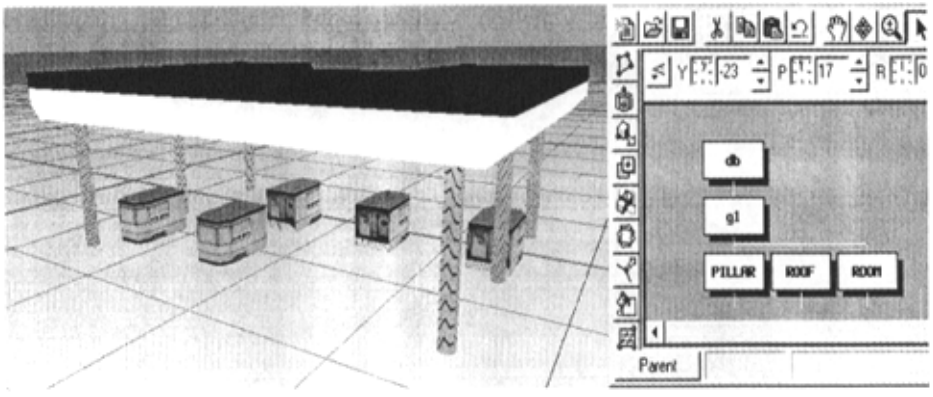


图 3-8　检查桥模型和数据库结构

➢ 集卡建模

为了使集卡的结构能够清晰的呈现出来，将各个部件复制再移动到不同的地方，以显示出部件与整体的效果（如图 3-9）。集卡的建模分四步进行，即集卡头部（司机室）、车架、底盘和轮子。就建模而言，多数是长方体建模，所以主要涉及的是多边形拉伸技术。在集卡司机室的建模过程中，拉伸技术会使模型显得比较呆板，因为司机室是有些棱角的，所以可以通过找特征面，然后用放样技术加适当的纹理来实现。放样技术将在后面船身的建模中重点演示。

建模时各个部件的归属关系不一定很清楚，但可以方便的利用层次结构视图来修改。建模过程中，任何部件的建立都应该基于一个特定的节点，如果建模时忽略了这

一点，在建模之后也应该调整数据库的结构，使有相同特征的面尽量在一起，Creator 提供了方便的数据库修改功能，这也为数据库的进一步优化提供了手段。

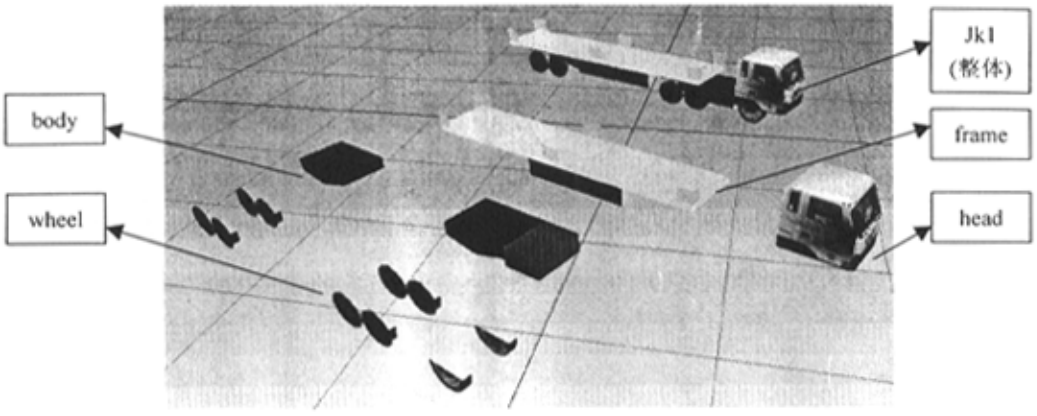


图 3-9　集卡的整体视图与部件视图

➢ 桥吊建模

桥吊的建模是项很复杂的工作，虽然它所涉及的建模技术并不难，但是建模的工作量很大，最后产生的面很多。

建模之前必须搜集桥吊的尺寸参数，通过三视图得到各种部件的尺寸和依托关系，并且考虑清楚大致的建模流程和步骤。如把整个桥吊分成梁、拉杆、小车行走机构、大车行走机构、主体框架结构等来建模。模型可以适当的简化，因为它不是最终应用系统主要观察的对象，所以它的建模不强调逼真性，只要形似即可，图 3-10 为桥吊的模型。

➢ 集装箱船建模[29]

集装箱船建模的主要工作虽没有桥吊那么繁琐，但是很讲求技巧的应用，这些技巧分别是：船身建模时用到的放样技术以及船上的集装箱生成时用到的纹理粘贴技术和实例化技术。它么都很好地反映了 Creator 建模的特点和关键技术。

船身建模采取的方法与前面提到过的集卡司机室的建模方法一样，采用的是放样（loft）技术。放样技术主要用于有特征面，但是特征面之间的过渡相对复杂的情况，这时就要画出几个特征面，然后通过放样生成最后的模型，模型之间的过渡完全由计算机来完成。因此，决定模型逼真度的主要因素是特征面的选取要有代表性。图 3-11 是为了构建船身而生成的 4 个特征面，图 3-12 是通过放样后生成的船身模型，基本反映了船身的外形，这样的一个船身就包括了 200 多个面，是无论如何无法用手工一

一画出的。

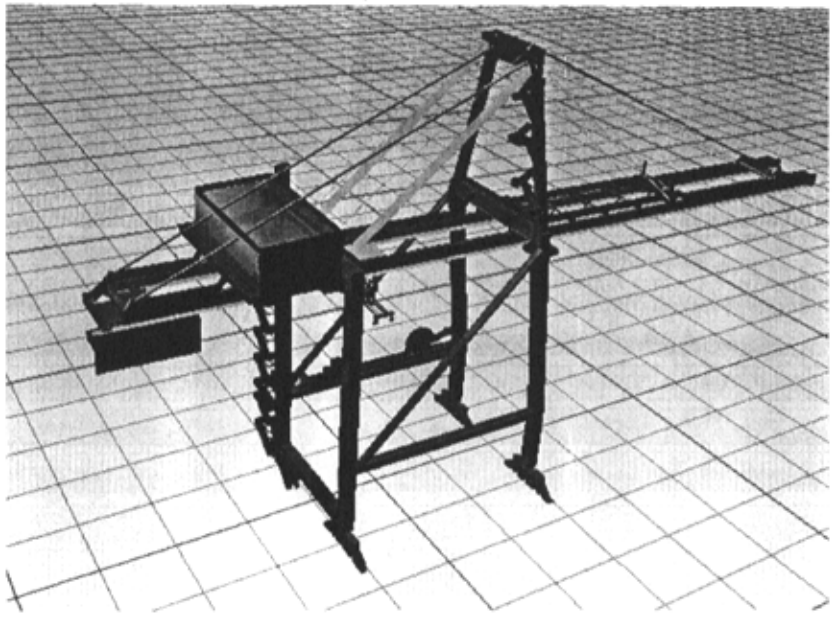


图 3-10　桥吊模型

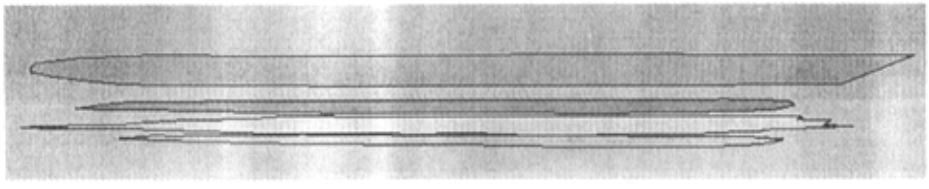


图 3-11　船身的特征面

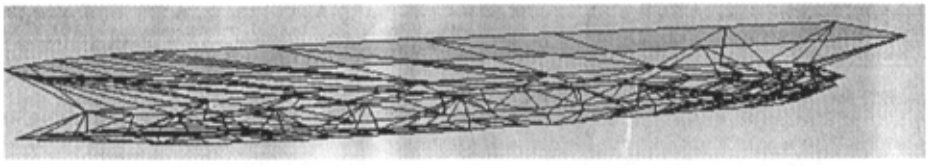


图 3-12　船身模型

在集装箱船的建模中，还涉及纹理粘贴技术，主要用在船上集装箱的建模上，按照通常的思路，一个箱子对应一个纹理，直接存在的问题就是所生成的面过多，最终还是要通过合并面的技术（Combine Faces）来使得多个面合并成一个面，因此就思考，不如直接就构造一个长方体，但是各个面的纹理已经包含了若干个集装箱。如图3-13，就是将贴于一个长方体的上、正、侧面的纹理图。这样，一个长方体可以按照比例画出多个集装箱。这就体现出构造纹理的讲究，如何使建模简单，如何使模型最

节省内存资源成为构造纹理的主要依据。

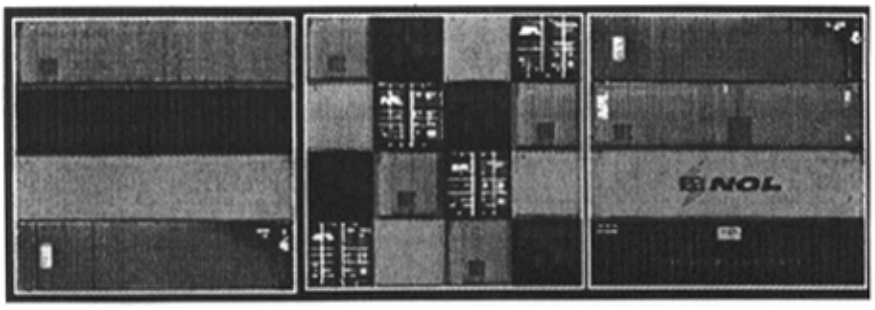


图 3-13　集装箱群的上、正、侧面纹理图

一个这样的长方体建模完成后，就可以利用实例化技术生成若干个长方体，具体的说，就是利用 Creator 中“插入转换矩阵”的方法构造出一样的长方体，这种处理方法不仅方便了建模，而且也大大节省了内存资源。

最后生成的集装箱船模型如图 3-14 所示。

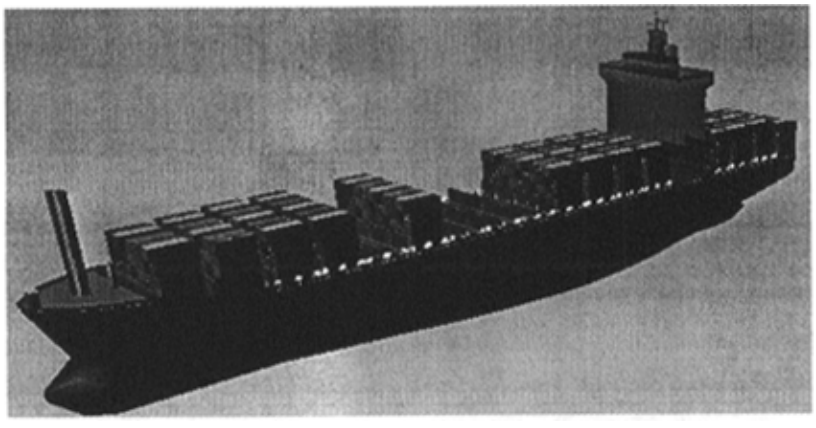


图 3-14　集装箱船模型

➢ 自然环境建模

上述这些模型都反映了港口的人文景观，再加入一些自然景观将会使模型更加逼真。处于必要性的考虑，这里对地面、海面、公路的建模都采用面上粘贴纹理的方法，而没有采用软件中的地形生成工具，原因是地形数据不易获得，而且有高低的地面也将增加整个模型的多边形数目。除此之外，还增加堆场的场区标志以及场区的划线，大大增加了整个堆场的逼真度。

这部分的建模，建模技术本身的应用比较简单，重点在纹理图片的处理上。通常

所拍的照片不能直接使用，必须进行一定的处理，图片的长宽必须是二维的，即应该是 2 的幂次，纹理的颜色和亮度有时也要做适当的处理，更为重要的是要学会制作透明纹理，尤其在背景图片以及花草树木的图片处理上，要让树完全的置于自然景观中，就必须消除主体(如树木本身)周围（如天空）的颜色。

上述所有的建模完成后，把它们组织到一个文件中，这就是所谓的静态模型文件（如图 3-15），它将直接加到 lynx 的对象物面板中。

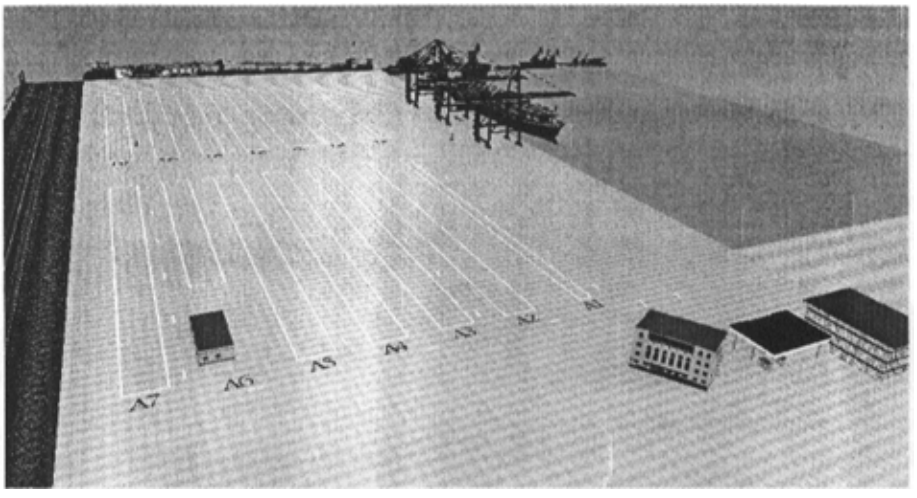


图 3-15　堆场整体模型

3.4.3 运动场景建模

运动场景是指在堆场上有增减变化的模型，这里仅仅指的是集装箱。前面已针对需要建立的集装箱模型数作过讨论。建全所有纹理式样的模型不仅浪费内存资源且不合实际，只建立一个又会使堆场看上去单调不堪，所以最终采用的方法是：建立 3 个拥有不同纹理的集装箱模型，并分别对应 20 尺和 40 尺（如图 3-16），因此就一共对应了 6 个模型文件。这样，在后面的模型驱动中，通过随机抽取的方式来添加集装箱，使得场景看上去具有真实感且比较美观。

图 3-16　三个不同纹理的集装箱模型（20 尺）

3.5 数据库建模

整个应用系统的建模分两大部份，上面已经提到了几何场景的建模，这一部分将主要讨论数据库建模。数据库是由若干表组成的集合，本应用系统将直接引入一个实时的数据库，并且利用其中两个表的信息来实现实时的驱动，也就是说最后场景所生成的画面就是对数据库文字信息的三维呈现，数据库的信息是随时变化的，所以三维场景也在不停的变化，这样场景的内容就生动的呈现在大家面前，给人一种直观的印象，更主要的是这个信息是三维场景的真实再现，便于对生产情况的了解。

在数据库中，我们用到了一个名为 container 和 bayshunxu 的表，这两个表中一共拥有几十个字段，在后面编程处理的过程中，不需要一一读入，因为不是每一个字段都有用处，而且每多一个变量，也就加大了系统资源的开销，所以这里数据库建模的主要工作就是弄清哪些字段对编程有用，并且把所有有用的字段按照功能来归类，这样便于编程时思路的理清。

3.5.1 数据库字段介绍

进行实时驱动时需要用到两个表，其中 container 表对应在场箱的信息，一个在场箱对应一条记录，bayshunxu 表的功能在于与 container 表中的信息结合，共同决定某个集装箱的发箱顺序。

- container 表

在 container 表中，一条记录代表一个集装箱的完整信息，根据应用系统需要实现的功能确定所需的字段有：集装箱箱号(ctnno)、箱位(rpos)、尺寸(size)、船名(vesscd)、航次(voyage)、进出口标志(iocd)、发箱顺序(fx_shunxu)、目标船箱位(cellno)。表的样式如图 3-17，各字段介绍如下：

1) 集装箱箱号(ctnno)：箱号多由 11 位字符组成，它在世界范围内统一编码，一般出 4 位字母+7 位数字构成。是具有唯一标识性的字段。
2) 集装箱箱位(rpos)：集装箱箱位由 6 位字符组成，它用来确定集装箱在堆场中的位置。前两位代表箱区，三四位代表列位，第五位代表行位，第六位代表层高。
3) 尺寸(size)：可取 20、40、45 三个值。但是在场景渲染时不区别 40 尺与 45 尺，所以仅仅区分为 20 尺、40 尺两种。
4) 船名(vesscd)：指出该集装箱属于的船名。
5) 航次(voyage)：指出该集装箱属于的航次。船名和航次的组合具有唯一性。
6) 进出口标志(iocd)：可取‘I’或者‘0’。‘I’表示进口航次，‘0’表示出口航次。针对出口航次要进行发箱顺序的判断，而进口航次无须此操作，这时就必须读取该字段以进行进出口的判断。

	ctnno	rpos	size	vesscd	voyage	iocd	cellno	celldh	fx_shunxu
	ACCU2003779	<NULL>	20	LKMHC	115	0	038410	D	3
	FWLU3001828	<NULL>	20	LKMHC	115	0	030803	H	11
	ACCU2003845	<NULL>	20	LKMHC	115	0	010802	H	8
	ACCU2003790	<NULL>	20	LKMHC	115	0	038210	D	2
	FWLU3001509	A26361	20	LKMHC	115	0	010201	H	1
	ACCU2011408	<NULL>	20	LKMHC	115	0	030404	H	6
	ACCU2011347	A30111	20	LKMHC	115	0	030801	H	12
	ACCU2011476	A30161	20	LKMHC	115	0	031206	H	20
	FWLU3001324	A16511	20	LKMHC	115	0	031205	H	19
	GVTU2008847	<NULL>	20	LKMHC	115	0	098610	D	3
	GVTU2008831	<NULL>	20	LKMHC	115	0	118210	D	1
	ACCU2011373	<NULL>	20	LKMHC	115	0	030603	H	7
	ACCU2011384	<NULL>	20	LKMHC	115	0	030604	H	10
	ACCU2011429	<NULL>	20	LKMHC	115	0	030602	H	9
	FWLU3001480	A26311	20	LKMHC	115	0	010402	H	4
	ACCU2003927	<NULL>	20	LKMHC	115	0	038208	D	1
	ACCU2011481	<NULL>	20	LKMHC	115	0	010601	H	5
	CBHU1307267	<NULL>	40	LKMHC	115	0	021001	H	1
	GVTU2008600	<NULL>	20	LKMHC	115	0	158410	D	2
	FWLU3001556	<NULL>	20	LKMHC	115	0	031003	H	16
	ACCU2011368	<NULL>	20	LKMHC	115	0	031004	H	17
	GSTU6722500	B23232	40	SALZACH	336E	0	<NULL>	<NULL>	<NULL>
	SKLU2208530	A24122	20	SALZACH	336E	0	<NULL>	<NULL>	<NULL>
	SKLU4536101	B22662	40	SALZACH	336E	0	<NULL>	<NULL>	<NULL>

图 3-17　container 表

7) 目标船箱位(cellno)：指一个出口箱放到船上的位置，它由 6 个字符组成，前两位表示船的倍位，中间两位表示层，后两位表示列位。“倍”是指船长度方向上的截面，一个倍就是船的一个截面。这里真正用到的是前两位的倍位信息。

8) 目标船层位信息（celldh)：该字段取值“D”或“H”,D 代表甲板，H 代表舱内。该字段与 cellno 组合构成一个发箱单元，如 cellno 为 031003，celldh 为 D,则 03D 就构成了一个发箱单元，同时这两个字段的组合也作为与 bayshunxu 表的连接字段。

9) 发箱顺序(fx_shunxu)：表示出口航次发箱时，针对船上目的箱位的倍位而言的发箱顺序，即针对一个发箱单元的发箱顺序，并不是最终的发箱顺序，最终的发箱顺序是由倍的发箱顺序和该倍（发箱单元）的发箱顺序共同决定的。

➢ bayshunxu 表

该表的读取是为了获取倍的发箱顺序，表的样式如图 3-18 所示。下面介绍一下它各字段的内容，从而引出与 container 表的关系。

1)船名（vesscd)、航次（voyage）:代表船名和航次。

2)航次标识（voyaio）:它是航次信息的又一种体现，它具有唯一性。

3)桥机号（jihao）：它代表针对一个航次作业的桥机号，也称作业线。

4)倍发箱序号（xuhao）：它就是读取本表最希望获得的倍发箱序号。

5)倍位（bayno）：它代表某个发箱单元。

	vesscd	voyage	voyaio	jihao	xuhao	bayno
	YUHE	0338	E0000274	Q02	10	18D
	YUHE	0338	E0000274	Q03	1	15H
	CHAPE	0338	E0000312	Q03	1	01H
	CHAPE	0338	E0000312	Q03	2	03H
	YUHE	0338	E0000274	Q03	2	13H
	YUHE	0338	E0000274	Q03	3	11H
	CHAPE	0338	E0000312	Q03	3	03D
	CHAPE	0338	E0000312	Q03	4	05H
	YUHE	0338	E0000274	Q03	4	09H
	YUHE	0338	E0000274	Q03	5	03H
	CHAPE	0338	E0000312	Q03	5	05D
	CHAPE	0338	E0000312	Q03	6	07H
	YUHE	0338	E0000274	Q03	6	01H
	YUHE	0338	E0000274	Q03	7	14D
	CHAPE	0338	E0000312	Q03	7	09H
	CHAPE	0338	E0000312	Q03	8	08D
	YUHE	0338	E0000274	Q03	8	05D
	YUHE	0338	E0000274	Q03	9	06D
	CHAPE	0338	E0000312	Q03	9	11H

图 3-18　bayshunxu 表

➢ 两个表之间的连接

集装箱的大多数基本信息已经在 container 表中给出，但是在做出口航次的发箱评判时，需要先知道倍之间的发箱顺序，而这个信息必须在 bayshunxu 表中获取，所以最终要读取两个表。图 3-19 说明了两表之间的连接关系。

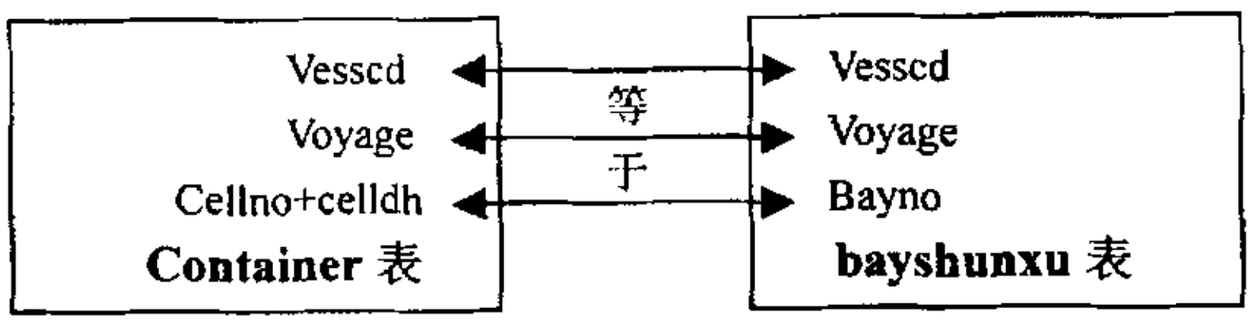


图 3-19　两表连接字段关系图

通过三个条件的对应相等，两个表的数据可以做到一一对应。相当于为 container 表每条记录后面加上了 jihao 和 xuhao 字段。

3.5.2 数据库字段的功能组织

数据库表中的信息与编程是直接相关的，也是为编程服务的，我们已经从庞大的数据库中选择了两个表，又从两个表众多的字段中选出了我们需要用到的字段，现在需要弄清每个字段的作用，也就是它们在程序中发挥的作用。

应用系统要完成的三大功能是：实时渲染数据库，箱信息查询以及发箱顺序评价。在编程之前应该建立与数据库连接的数据源。应用系统通过数据源与数据库连接，建立两个分别对应两个表的数据集，这样这些字段就被读入程序中，各字段的功能各有不同，在程序中各自起到了不同的作用。

初始渲染场景时，渲染的原则就是根据集装箱的箱位和尺寸在场景中增加集装箱，数据库在实时变化，三维场景也必须跟着变化，依据前后两次读取数据库的不同决定是否增减集装箱。

查询是按照箱位、箱号、船名、航次四者之一为条件进行的，如果按照船名、航次查询，返回的是若干个记录，如果按照箱位、箱号查询则返回一个记录。

发箱顺序的评价是嵌入在查询功能中的，这时就需要判断航次的进出口标志，如果是出口航次才需进行发箱顺序的评价，此处需要用到两个表的连接，在 bayshunxu 表中获取某倍的发箱顺序，在 container 表中获取某倍中箱子的发箱顺序，通过这两者的结合得出某个箱子相对于整个航次的发箱顺序。在具体的作业过程中，一个航次通常是几条作业线并行作业，所以还应该分作业线来分析，最终得到的信息就是箱子相对于该航次某作业线的发箱顺序。由于各作业线之间的箱子分布无法做到完全的独立，即不同作业线之间的箱子存在压箱现象，所以在评价时就涉及一个最理想的情况和最不理想的情况，实际的情况就应该在这两个临界值之间。

小　结：虚拟堆场的仿真包括建模与驱动两大工作，它们是相辅相成的关系。本章主要介绍了虚拟堆场建模时涉及的相关内容，包括几何建模和数据库建模。几何建模采用Creator软件，介绍了Creator建模技术的特点、系统构成以及Creator中使用的关键技术，包括OpenFlight层次结构视图、纹理映射技术、实例化技术以及层次细节技术等。在具体建模时讲解怎样将这些技术贯穿到整个三维场景的建模过程中，从而说明建模应该注意的问题和技巧。数据库建模为应用系统提供了最原始的数据，是场景三维呈现的文本依据，这里介绍了在程序中需要用到的数据字段及各字段对应的功能。

第四章 集装箱虚拟堆场驱动仿真流程

堆场的三维模型建立之后，它们只是一个个独立的模型文件，而且体现的仅仅只是堆场的静态信息，就如同摄像机拍摄出的一幅画一样。如果想将数据库的实时信息通过模型体现出来，就必须通过编程实现实时信息对场景的驱动，只有这样，三维模型才能活起来，达到模拟堆场的作用。应该说，所有虚拟系统的开发都要包括这两大主要的工作：建模与驱动。驱动是建模之后的工作，让场景能通过视点的转换动起来，并实现与用户的交互以及其它一些更高级的功能，帮助用户观察场景并能更好的控制场景。也正是因为虚拟的共同目的，所以虚拟系统的多边形建模与普通机械的实体建模有所不同，而驱动形成的应用文件也与动画不同。

4.1 虚拟堆场驱动的实现方法[30 31]

虚拟堆场的驱动和建模一样，也需要找到最适合的软件，在建模时采用的是 Creator 软件，模型采用的是 OpenFlight 层次结构，所以在驱动的时候也采用能最好支持这种模型结构的驱动软件，也就是 Multigen 公司出品的 Vega 软件。

Vega 是 MultiGen 公司最主要的工业软件环境，用于实时视觉和听觉仿真、虚拟现实和通用的视觉应用。它把先进的仿真功能和易用的工具结合到一起，创建了一种使用最简单，但最具创造力的体系结构，来创建、编辑和运行高性能的实时应用。Vega 能显著地提高工作效率，同时大幅度减少源代码开发时间。

Vega 对于程序员和非程序员都是称心如意的。它的开发大致包括两大步骤，先是在 LynX 图形界面下配置各种模板中的参数，不用任何的编程工作，如果通过它可以达到理想的效果，应用系统的开发就完成了。但是如果通过这样的界面配置无法达到用户的要求，就要用编程语言对 Vega 进行二次开发，Vega 提供了完整的 C 语言应用程序接口，通过在程序中调用 Vega 中的函数，可以自如的控制模型，按用户的需要实现一些高级的功能。

LynX，一种基于 X/Motif 技术的点击式图形环境，使用 LynX 可以快速、容易、显著地改变应用性能、视频通道、多 CPU 分配、视点、观察者、特殊效果、一天中不同的时间、系统配置、运动模型、数据库及其它，而不用编写源代码。

LynX 还可以扩展成包括新的、用户定义的面板和功能，快速地满足用户的特殊要求。事实上，LynX 是强有力的和通用的，能在极短时间内开发出完整的实时应用。用 LynX 的动态预览功能，用户可以立刻看到更改任何一个面板中参数的变化结果。LynX 的界面包括用户应用开发所需的全部功能。

Vega 还包括完整的 C 语言应用程序接口，为软件开发人员提供最大限度的软件控制和灵活性。实时应用软件开发人员喜欢 Vega，因为 Vega 提供了稳定、兼容、易用的界面，使他们的开发、支持和维护工作更快和高效。Vega 可以使用户集中精力解决特殊领域的问题，而减少在图形编程上花费的时间。

4.2 应用系统开发平台的选用

由于无法在 LynX 中完全得到预计的效果，所以必须通过编程来实现一些较复杂的功能。Vega 自身提供了很方便于调用的函数库，但是它自身并没有开发平台，没有面向对象的能力，所以必须借助其它的开发平台，通过调用 Vega 函数实现实时仿真的目的。VC++.NET 是基于 VC6.0 的思想，重新开发的一套功能更强大，使用更方便的软件，它其中的 MFC 包含了强大的基于 Windows 的应用框架，提供了丰富的窗口和事件管理函数，因此我们就把它作为开发平台进行虚拟堆场的开发。

4.3 应用系统的开发目标

每一个应用系统的开发都必须针对具体的目的进行，虚拟堆场虚拟的目的在于能够让人们坐在办公室中就知道堆场中各箱区箱子的变化情况，所以如何使箱子能够按照实际的情况显示出来就成为了首要的任务。在这个基础上，增加查询功能，使用户可以按照船名、航次、箱位、箱号进行查询，这样便于计划人员观察堆场的分布情况，帮助了解箱子的信息，从而对箱位的分配给出参考意见；增加方案评价功能，主要测评的是出口航次集装箱发箱顺序的优劣，如果存在太多的压箱现象，可通过三维场景中渲染颜色的变化告知用户及时的更改发箱顺序，起到了预知实际发箱过程的作用，从而避免实际作业效率的下降。同时，由于整个应用系统的开发是基于“虚拟现实”的思想，所以应用系统与用户有很好的交互性，能够让用户有置身于堆场中的感觉，而且可以方便自如的到达堆场的任何地方。应用系统的开发就是围绕这三大目的进行的。

4.4 应用系统开发主要技术分析[32-45]

整个软件的设计围绕它将实现的三个功能来展开，应用系统之所以能够显示三维场景，并能够按照用户输入的条件进行适当的变化，都有赖于 VC 对 Vega 的调用，所以在 VC 中必须能够调用 Vega 的窗口，即要建立一个适合 Vega 运行的视图类，在这个视图类中，要符合 Vega 的运行规则。这种让场景随用户的输入而变化的情况正好体现了三维模型良好的交互性，也正是虚拟现实系统的优势所在。

堆场的变化最终归于集装箱信息的变化，所以集装箱就成为程序控制的单元，因

此必须建立一个集装箱类，对该类的操作包括增加、移去集装箱，变换颜色和其他属性等。

三维图形的变化依赖于数据库信息的改变，所以必须建立一个记录集类，使得应用系统可以与某一个数据源连接，从而访问数据库的信息。数据库的信息在实时变化着，所以必须在程序中建立适当的函数来控制对数据库的访问，及时得到数据库的变化，从而在三维模型中反映出来。

为了能够对场景中的集装箱信息进行查询，必须建立相应的对话框类来提供一个可视化的界面供用户输入各种查询条件，从而控制相应属性的集装箱外观的改变，这是作为人为输入的一个方面；用户的输入也可以是键盘操作，通过按相应的键来控制场景发生变化，比如程序中的场景移动就是靠按键盘来实现的。

4.4.1 Vega 应用程序的特点[46 47]

无论是哪种类型的 Vega 应用程序都要分三个步骤：

1)初始化

调用 vgInitSys 函数初始化系统并创建共享的内存区和信号区。

2)定义

通过创建需要的事件和需要的类来定义系统。可用两种不同的方法来完成。第一种方法是利用一个应用程序定义文件（ADF）的名称调用 vgDefineSys 函数；第二种方法是调用外部函数创建 Vega 类事件。本系统采用的是第一种方法，这个应用程序定义文件就是在前面提到过的在 LynX 中定义过的文件，在这个文件中已经对一些类相应的参数值赋值。

3)系统配置

进行系统配置，使 ADF 中的定义与函数调用结合起来，调用 vgConfigSys 函数完成步骤。

运行系统对数据库每一帧的绘制分为三个阶段：Application(应用),Cull（截取）,Draw（绘制)。运行系统在 Application 阶段将数据库载入后，必须在另两个阶段绘制可见场景。在 Cull 阶段，运行系统遍历整个数据库结构来发现可见数据。在 Draw 阶段，运行系统绘制可见数据。

Vega 应用程序包括 vgSyncFrame 和 vgFrame 函数的调用，通常由每个主循环或者每次需要一个新的显示时调用这些函数。

VgSyncFrame 函数把请求线程同步到一个给定的帧率上，其作用是检查退出标记，如果设置了退出标记则试图退出，执行用户已定义的任何 postsync 系统回调，更新激活的运动系统、场景运动体、观察者或纹理，协调输入输出设备的同步。

VgFrame 函数在当前帧下进行 Vega 执行而引起的所有的内部处理工作，其作用是执行用户指定的任何 preframe 系统回调，启动选择和绘制线程，选择某一通道并绘制已定义的回调。

即使是最小的 Vega 程序也要使用一个应用程序定义文件和上述 5 种函数，一个简单的 Vega 应用程序代码如下：

```
vgInitSys()
vgDefineSys(adf文件)
vgConfigSys()
while(1)
{
  VgSyncFrame()
  VgFrame()
}
```

值得一提的是，这只是最简单的模式，在 VC++编程中还需要处理很多复杂的操作，所以必须在这个基础上添加很多函数，比如各种函数的后回调函数，如 postFrame 中（post 是所有后回调函数的关键字，表示“之后”之意），可以定义键盘操作，使视点能够在场景中移动。用户自定义的函数也可以加入其中，这主要看用户实际的需要。

在具体的编程中，建立基于 vega 的视图类，在其中包含上述 5 类基本函数。

4.4.2 数据库相关操作

要用数据库信息驱动三维场景的变化，就必须使得应用系统能够读取数据库相关表的字段，而且还必须设置一个机制来跟踪数据库的变化，从而保证三维场景也能够相应变化。

➢ 数据库的访问

在建立与数据库的连接方面，考虑到数据库管理系统的多变性，所以采用 ODBC 方式进行数据访问，在 ODBC 层之上的应用程序看来，各个异构的关系数据库只是相当于几个不同的数据源，而数据源组织的不同对程序员来说是透明的，所以可以编写独立于数据库的访问程序。

在编写程序之前，利用 WIN2000 管理工具中的数据源图标进入新建数据源窗口，建立数据源，让它指向相应的数据库，在编程时与数据库的连接就转化为与数据源的连接了，数据源其实就是数据库与应用系统之间的桥梁。

在 VC 中建立记录集类（CRecordSet）的派生类，运用 RFX 机制，从数据源中提

取出记录集，使得记录集的数据成员可以与数据源中某个表的某个字段对应起来。这样，数据库的信息就通过数据源存储在记录集的数据成员中。

➢ 数据库信息的跟踪

建立与数据库的连接后，必须解决的问题是如何实时的读取数据库的变化，可以采用两种方法来实现这种跟踪。一种是主动的，在 DBMS 中编写触发器程序，一旦有某些字段的信息发生变化，就通知应用程序作相应的处理，变化是由 DBMS 发给应用程序的，它的实时性就很强；另一种是被动的，在程序设计时由应用程序定时的访问数据库，根据前后两次数据库信息的不同推断出各种变化情况，当数据库发生变化时，应用程序是不知道的，而只有在下一个访问时间开始时，才能够读取到在这两次访问时间之间的变化，也就是只有在又一次访问数据库后场景才会重新渲染。如果访问的间隔时间设置的过长，那么实时性就不够好，就这一点而言它不如第一种方法。但是由于起落一个集装箱的时间并不短，所以只要合理的设置访问数据库的时间间隔，还是可以保证实时性能的，而且计算机的处理速度绝对可以胜任这项工作；再者这种方法涉及的只有 VC 编程，避免了某一 DBMS 与 VC 程序的连接问题，便于最后的统一和发布。所以本系统就采取第二种方法实现数据库的实时跟踪。

接下来的问题就是如何比较数据库前后的变化，每当访问数据库时，记录集永远反映的是数据库的当前信息，所以必须设计一种结构来存储上一次访问的信息，这样才能通过比较看出不同。考虑到比较的灵活性，所以选用链表结构来存储前一次的字段内容，它总是在访问数据库后被赋值，当下一次访问发生时就与记录集比较，出现不同时就触发场景变化，同时也改变自己以便和最新的数据库一致。

4.4.3 建立链表结构

进行数据库跟踪时，必须寻找一种结构来存储前一次访问数据库的信息，通过比较对场景发出相关的操作要求，然后更新链表与当前记录集的内容一致。所以在程序中需要新建一个类来存放这种链表结构，链表类的定义如下：

```
class Container
    {
public:   //数据成员
     CString ctnno;        //箱号
     CString rpos;         //箱位
     Container* nextcon; //下一箱子
public:   //成员函数
```

```
    Container(void);              //构造函数
    Container(CVPortSet*);        //构造函数
    ~Container(void);             //析构函数
};
```

这个 Container 类定义链表中的每一个结点，结点的数据成员包括集装箱箱号、箱位以及下一个结点的指针。成员函数包括构造函数和析构函数，提供了类构造时的初始化以及类销毁时的操作。此外，为了配合链表的操作，也建立一个相应的操作链表的类，它的数据成员为链表的头指针，成员函数除了构造和析构函数外，还有用户自定义的几个函数，起插入、删除、统计和清空结点的作用。

```
class Opercon
{
public:      //数据成员
   Container* fircon;
public:      //成员函数
      Opercon(void);
      ~Opercon(void);
      void inscon(Container*);    //插入结点
      void delcon(Container*);    //删除结点
      int  lengcon();             //统计结长度
      void zerocon();             //清空所有结点
};
```

4.4.4 场景对象的访问与处理

这里的场景对象主要是指需要通过程序来控制的对象物，也就是集装箱。数据库的信息、链表的信息都属于文字信息，它们的存在都是为场景变化服务的，所以就涉及如何添加删除场景对象物的编程问题。

这部分的编程效果等价于在 LynX 中一个个的添加集装箱对象物，但是之所以没有在图形界面下这样操作，是因为集装箱的放置位置必须要由数据库的箱位信息来控制，而且随着实际情况的变化，还要做增、减处理，所以就采用编程来实现集装箱的控制。和在 LynX 界面下操作对象物的思路一样，要先将对象物对应的模型文件载入，然后作为一个对象物添加到场景中，并且指定其位置。

Vega 中用数据集 vgDataSet 函数来对文件进行装载，它可以把文件信息转换成一种运行状态下可用的结构。接着就要通过 vgObjDS 函数来把装载的模型文件添加到

指定的对象物中，利用 vgPos 函数放置对象物。然后通过 vgAddScene 函数将对象物添加到场景中。可见，编程思路就是怎样通过编程来完成在 LynX 界面中完成的工作。具体的程序模块如下：

```
ds=vgNewDS();
vgName(ds,集装箱模型文件);
vgLoadDS(ds);
obj=vgNewObj();
vgName(obj,集装箱对象物的命名);
vgProp(obj,VGOBJ_CS,VGOBJ_DYNAMIC);
pos=vgNewPos();
vgPosVec(pos,相应的六个坐标值);
vgPos(obj,pos);
vgObjDS(obj,ds);
vgMakeObj(obj,VGOBJ_COPY);
vgAddSceneObj(vgGetScene(0),obj);
```

以上分析了编程中需要解决的关键问题，在程序中必须依托 Vega 函数来描绘场景，把程序和三维模型连接起来就是依赖于对 Vega 函数的调用。此外，还分析了控制整个过程的数据库信息、链表以及场景对象物，模型的渲染主要是依靠它们三者之间的相互作用来实现的，这三者之间的关系如图 4-1，数据库信息是主导信息，它在渲染开始时初始化链表和场景对象物，然后定期更新数据库，通过与保存了上一次数据库信息的链表作比较，有不同时就更新场景，同时也更新链表，使得链表与本次数据库信息一致。

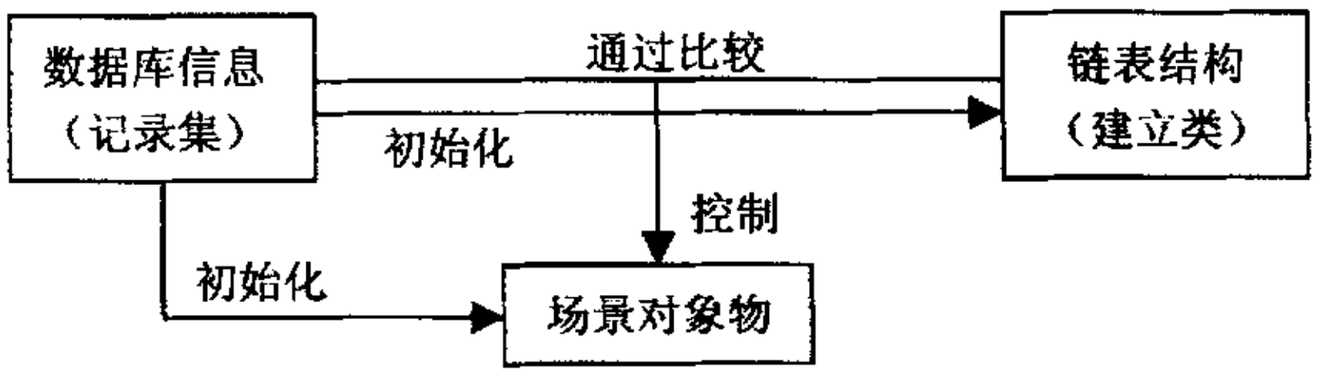


图 4-1　记录集、链表、场景对象物三者之间的关系

4.4.5 箱位的确定

集装箱箱位的确定直接保证三维堆场显示的准确性。在三维堆场实际的观测中，人们需要借助很多辅助手段，或是用望远镜、或是亲临现场实地观看，或是手握大量

数据，然而这种种手段都无法建立堆场整体的效果，人们对堆场的认识仅仅局限于某一个区域，即使用摄像头也是如此，摄像头的观测角度制约了它的观测范围。所以这个可视化程序在这方面就可以帮助人们以最直观和便捷的方式观察场景。它可以总览整个堆场，也可以任意变化观测方向，犹如在堆场中行走一样，也可以局部观察某个箱区，可以以不同的速度切换视点等，用它来观察时方便、直观，最主要是信息比较完整，整个堆场的分布包括每个箱区的堆放情况都能一目了然。

但是，这一切的前提是信息要准确，也就是要确保集装箱放置位置的准确性。集装箱堆场有自己的编码规则，所以编程之前应该将它们转换成场景数据库中具体的点的位置。

在数据库中，箱位字段占 6 位，诸如 A10111，前两位表示集装箱所在的箱区，即 A1，可在 A1 至 A7，B1 至 B7 之间取值；中间两位代表它所处的列，即 01 列，可在 01 到 64 之间取值；第五位表示它所处的行，即第一行，可在 1 到 6 之间取值；最后一位代表层，可在 1 到 4 之间取值。根据全局坐标的标准，集装箱的 X、Y 轴和堆场的整体布局关系如图三所示。

在三维模型中放置对象物依靠的是坐标点，所以就存在一个将字段信息转化成坐标点值的问题。这里主要是要把数据库箱位字段的每位分解出来，分别对应到 X、Y、Z 三个坐标轴上。如图 4-2，列位和箱区的 A、B 之别决定了 Y 方向的位置；行和第几箱区决定了 X 方向的位置，而层决定了 Z 方向的位置。

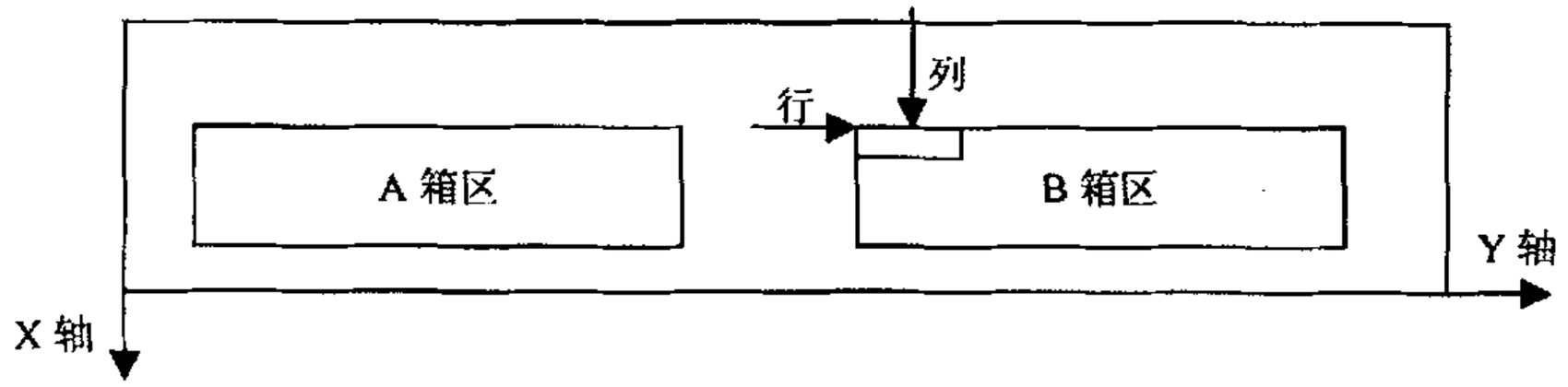


图 4-2　堆场的坐标信息

小　结：本章主要对集装箱虚拟堆场驱动仿真的流程进行了分析。介绍了虚拟堆场驱动的实现方法，即依托于Vega软件。首先要在Vega的界面环境下进行应用程序的预定义，通过在LynX中定义各种面板参数，可以对程序的某些属性，如窗口状态、通道属性、观察者等进行初始化。因为程序的功能无法在LynX界面中得以实现，所以必须通过编程进行二次开发，由于Vega自身并无面向对象的能力，所以需要借助VC.NET的开发平台，调用Vega函数进行应用程序的开发，其中应主要注意的问题是数据库访问、Vega的嵌入、保证实时渲染的方法以及与查询和评判功能的整合等问题。

第五章　集装箱虚拟堆场仿真应用系统

在上一章中已经介绍了虚拟堆场仿真过程的步骤以及开发过程应解决的关键问题，这一章将遵循这个步骤，并结合虚拟堆场预期达到的目标进行应用系统的开发，具体说明实现这些关键问题的方法。

5.1 应用程序预定义

应用系统开发采用的是 Vega 软件，利用它开发应用系统包括两步，首先要在 LynX 界面中定义各个面板的参数（如图 5-1），进行动态预览，找出最佳效果的参数值。然后再根据实际情况，看看是否需要对 Vega 进行二次开发。

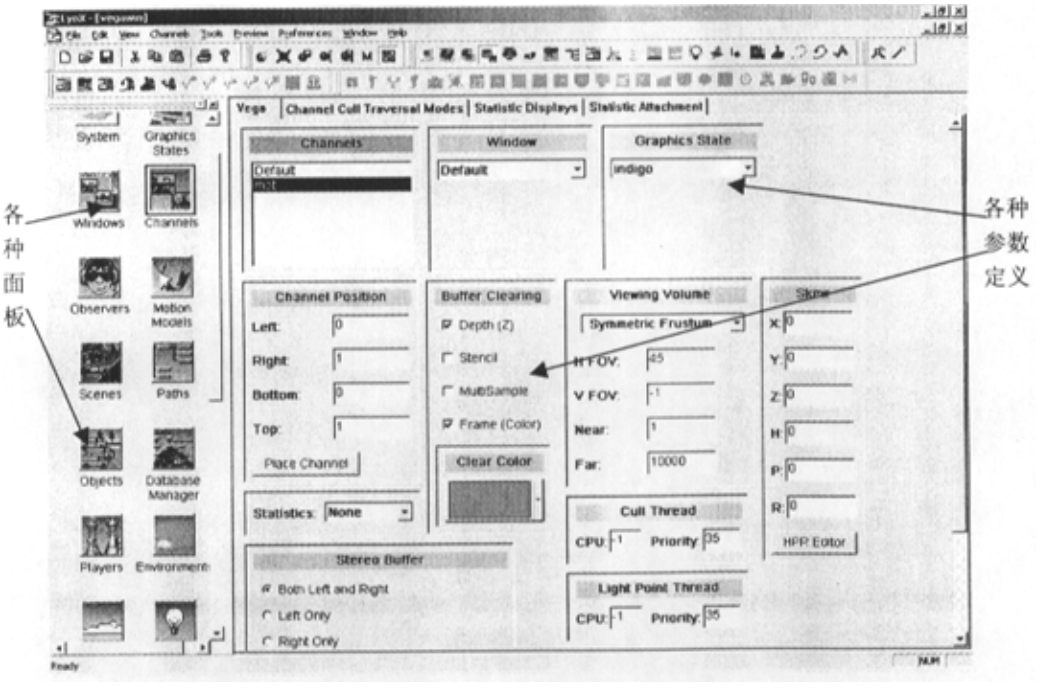


图 5-1　Lynx 操作界面

虚拟堆场应用系统因为涉及实时渲染、查询和方案评判等功能，所以必须用编程语言进行二次开发。在 LynX 界面中定义好各种参数后，可以生成一个应用程序定义文件，对 Vega 进行编程开发就是在这个文件的基础之上进行的，所以把这个过程称为应用程序的预定义。

所有在 LynX 中对各种面板参数的定义都可以在编程中直接进行，但是通过书写编程语言没有这样来得简单和直观，这里定义的参数可以立即通过动态预览看到效果，以便修改参数，直到满意为止，这时可以把各种参数的定义保存在一个文件（*.adf）中，这个文件称为应用程序定义文件，也就是后面编程开发时要调用的文件，调用这个文件也就相当于为应用程序定义好了各种参数。

根据虚拟堆场实际的需要，在 LynX 中配置以下面板中的参数。

➢ 对象物（Objects）面板

对象物面板用来为场景添加对象物，也就是在建模阶段建立的模型文件，然后定义该对象物在场景中的位置，这个位置的定义可以保证局部坐标与整体坐标的统一。

在虚拟堆场的建模中，把模型分为静、动两种类型，也讨论过将模型加到应用程序方法的不同，一种通过编程实现，一种就是在这里加入。静态模型是除了集装箱以外所有可见场景的总和，它相对于整个应用程序的位置始终不变，所以可以在这个面板中直接加入，但是动态集装箱的位置始终随着实际的情况发生着变化，它的位置没法通过这里的坐标定义来确定，所以只能通过编程来加入。在建模时，为了使对象物的位置关系较为简单，已经将静态模型的原点取在了整个应用程序坐标的原点上，所以这里对象物的 X,Y,Z 坐标均取 0 值。

在确定一个对象物位置时，除了有 X,Y,Z 值外，还有 H,P,R 三个值，分别代表对象物相对于 Z,X,Y 三轴旋转的角度，一个拥有 6 个自由度物体的位置就由它们完全确定出来。对静态模型而言，它应该始终位于 XY 平面上，所以只涉及模型相对于 Z 轴旋转的角度，这个角度值应该与观察者的位置相协调，因为用户希望的观察角度即受对象物坐标的影响，也受观察者坐标的影响。

➢ 场景(Scenes)面板

场景面板中所定义的就是用户将实际看到的场景，任何添加到对象物面板中的对象物若想可见，都必须再加到场景中，只有这样才能呈现出来，也就是说真正进入人视觉中的是场景，而不是对象物。定义的对象物是否添加到场景中就决定了它的可视与否。

这里应该把定义好的静态模型对象物添加到场景中。

➢ 图形状态（Graphics states）面板

图形状态面板用来定义模型载入时是否有纹理、光照、雾化效果、深度缓存、面的正反显示、透明度以及是否为线框方式显示等。这些参数的确定就决定了模型最初的显示样式。

这里取消对光照的选择，因为通过比较，没有光照的场景显得比较清晰。

➢ 窗口（windows）面板与通道（channels）面板

窗口面板用来定义窗口标题名称、窗口的样式、窗口的位置、窗口模型显示之前的纹理、位面属性等。通道面板用来定义通道所连接的窗口和图形状态、通道的位置、视椎体的方式等。这里都用缺省设置。

➢ 环境（Environments）面板

环境面板中定义环境的名称、光源、雾效果、天空的颜色、一天中时间所决定的场景的明暗等。

这里应该适当选取一天中的时间设置，使得场景观测起来明暗适中。

➢ 环境效果（Environment Effects）面板

环境效果面板主要用来模拟某一种天气类型，如可以定义云的类型、雾的效果、风暴、太阳月亮等。这样可以使得场景更加逼真而具有更高的观赏性。定义的环境效果必须加入到某一个具体的环境才能起作用。这里将环境效果定义为云效果。

➢ 观察者(Observers)面板

这是最后一个要定义参数的重要面板，它的定义也是基于前面诸多面板的参数值。如前面定义的场景、通道、环境、图形状态等，所有这些定义都必须进入人的视觉才能起作用，即要与某个观察者联系起来。

观察者面板主要定义视点的范围和变化方式。视点的范围就是指视点将看到什么，实质就是定义视点将与哪个场景挂钩。视点的变化方式是指视点将按照哪种运动模型来运动，LynX 中提供了很多运动模型，可以让视点犹如在飞机，汽车，或人身上那样，达到自如的模拟各种视点变化的情况。

在这里，将刚才定义的静态模型场景添加到视点范围中，并把视点的运动方式定义为手动(manual)方式，这是因为最后的应用系统要能够看到堆场中的任何一个地方，选用如飞行、旋转、驾驶等类型时，它们的速度太快，对于堆场这相对较小的场景没有必要，而且这样并不利于细致的观察某个位置的集装箱，而如果选用手动控制，即意味着它不能运动，仅仅有个初始值决定视点的最初位置，视点的移动是通过编程不断赋予视点新的位置来实现的，这种方式对视点移动的控制非常灵活。

在定义每个参数时，都可以利用动态预览功能来观察效果，效果理想后，可以保存该文件，保存的文件就是后面程序中即将调用的文件，用来初始化整个场景。

5.2 建立 ODBC 数据源

建立应用系统之前，要先加载数据库，在较普遍的客户机/服务器体系结构中，通常数据库只存于服务器上，所以无须在本机导入数据库，但为了适于应用系统开发的需要，这里在 SQL SERVER 中导入数据库。接着建立数据源，在控制面板的“管理工具”中建立数据源与相应数据库的连接。最后开发的应用系统需要与实际的数据库连接，所以必须改变数据库，正是因为选用的是 ODBC 编程，所以这种修改发生在程序之外，只需修改相应数据源的定义就可以了，增加了应用系统的适应性。

数据库通过数据源与应用程序连接起来，而在应用程序中，利用 MFC 的 RFX 机制将数据源表中的字段信息转化成记录集类的数据成员。关系图如 5-2 所示。

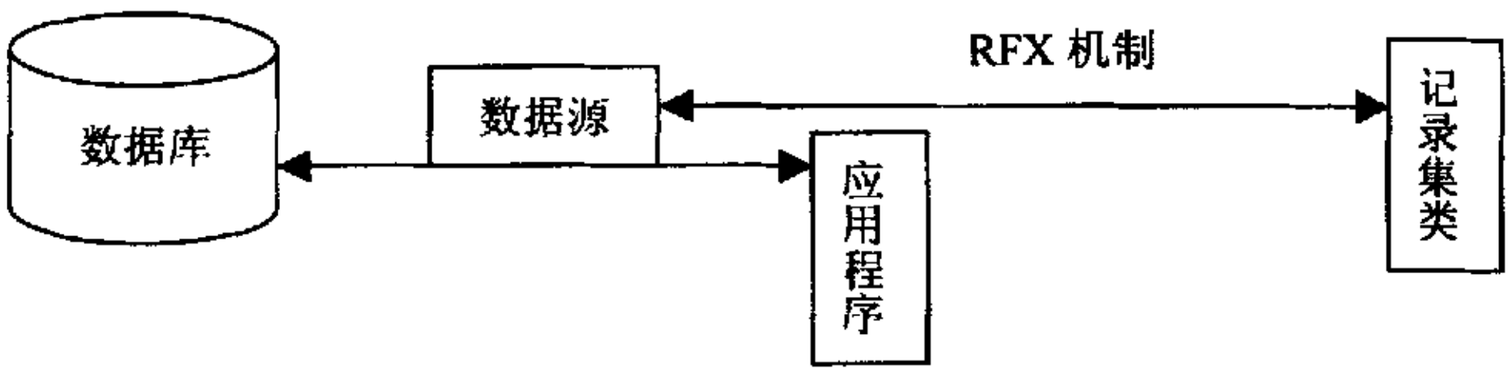


图 5-2　数据库信息的传递方式

通过上图可知，最终数据库的信息是由记录集类的数据成员体现出来的，对数据库的访问是通过不断的读取记录集类来实现的。

5.3 MFC 应用程序向导

前面已经说明，本程序选用的是 VC.NET 平台，并且能够进行数据库操作，所以在用应用程序向导添加应用程序时，要选择不支持文件的数据库视图。这样就可以建立应用程序的基本框架，即四大基本类。本应用系统名为 VPort，因为涉及数据库编程，所以建立以下五大类为：

1)CVPortApp:它是应用程序的“应用程序类”，负责初始化和运行应用程序。

2)CMainFrame:它是应用程序的“框架窗口类”，负责显示和搜寻用户命令。

3)CVPortDoc:它是应用程序的“文档类”，负责装载和维护文档，文档包括应用程序的工作成果或者环境设置数据等，或是程序需要保存的任何内容。本程序涉及很多参数的设置，在后面程序的讲解中会具体谈到在文档类中应该设置的数据。

4)CVPortView：它是应用程序的“视图类”，负责为文档提供一个或几个视图，视图的作用是为修改、查询文档等任务提供人机交互的界面。在本程序中，因为不涉及这个视图类的编程，所以在程序的加载过程中不加载这个类，取而代之的是自定义的与 Vega 相关的视图类，这个类必须能够调用 Vega 函数，它用来显示三维的场景。

5) CVPortSet：它是应用程序的“记录集类”，代表从数据源选择的一组记录集。通过在它之中内嵌的 Cdatabase 类完成与数据库的连接工作。主导整个程序数据流的就是该记录集。

5.4 建立基于 Vega 的视图类

5.4.1 建立视图基类 CVegaView

程序已有的视图类不予加载，这时就需要建立新的视图类，基于应用程序的基本目的，一定要使该类能够显示三维场景，所以整个视图类也要围绕 Vega 的核心来进行。

它的定义文件中包括很多成员函数，它们的共同作用使得三维场景能够成功的渲染出来，并能够不断的交互。其中比较重要的函数如下：

```
BOOL   create( const RECT& rect, CWnd* parent );
void   runVega( void );
void   stopVega( void );
void   pauseVega( void );
void   unPauseVega( void );
HWND   getSafeHwnd( void );
BOOL   getContinueRunning( void ) const { return continueRunning; }
Void   toggleGfx( int what );
virtual    const char*   getAdfName( void );
virtual    void   postInit( void );
virtual    void   postDefine( void );
virtual    void   postConfig( void );
virtual    void   postSync( void );
virtual    void   postFrame( void );
```

Create 函数用来创建窗口的样式，runVega 函数开启一个线程来运行 Vega，它包括对以下线程的调用。其中通过调用 vgInitWinSys 函数来把 Vega 窗口加到应用程序的 View 窗口中。这个线程中包括了 Vega 运行的五个基本函数，

```
UINT runVegaApp( LPVOID pParam )
{
    CVegaView* pOwner = (CVegaView*)pParam;
```

```
vgInitWinSys( AfxGetInstanceHandle(),  pOwner->GetSafeHwnd());
pOwner->setVegaInitted( TRUE );
pOwner->postInit();
 vgDefineSys( pOwner->getAdfName() );
pOwner->setVegaDefined( TRUE );
pOwner->postDefine();
 vgConfigSys();
pOwner->setVegaConfiged( TRUE );
pOwner->postConfig();
 while ( pOwner->getContinueRunning() ) {
    vgSyncFrame ();
    pOwner->postSync();
    vgFrame ();
    pOwner->postFrame();}
   pOwner->setVegaInitted( FALSE );
   return 0;}
```

5.4.2 建立视图派生类 CMFCView

上述基类几乎可以用于任何需要显示 Vega 窗口的应用程序中，但是根据每个程序的不同应建立自己的视图类，所以以刚才定义的类为基类，建立本应用程序的视图类 CMFCView 类。派生类可以调用基类的函数，也可以重载基类函数以及添加自己需要的函数，这里将根据功能介绍相关函数。

在 OnInitialUpdate()函数中，调用文档的记录集类及链表操作类，打开记录集，并把记录集传递给 CmainFrame 类，因为当做查询操作时，信息交换是在 CmainFrame 类中完成的，并执行 runvega()函数来开启三维场景的绘制线程。

为了将场景能够根据数据库的信息初始化，所以重载 postConfig()函数，在这里完成数据库信息对链表和场景对象物的初始化。

在 postFrame()函数中，利用 vgGetWinKey()函数获得用户的键盘输入，这时就可以定义一些按键来操纵场景，分别完成图形状态的改变，视点的定位转移等功能。在与用户的交互过程中，需要不断的改变视点的方位值，这是通过定义 OnKeyDown、OnKeyUp、OnChar 三个函数来实现的。

集装箱的模型文件有三个，如果利用程序计数来逐一添加进场景中，很可能出现比较单一呆板的视觉效果，所以就采用由系统产生随机数的方法，这样通过随机数找

到要加入数据集的文件名，实验后效果明显提高。这是一个提取文件名的函数，所以取名该函数为 getConName(CString size)，它带有一个箱子的尺寸参数，根据 20 尺与 40 尺的不同分别处理。

堆场的实时变化是依靠不断访问数据库，并且与链表进行比较而产生相关的操作来保证的，所以建立系统消息映射函数 OnTimer(),通过设定的时间参数来定时的访问数据库，将取得的新数据库信息与老信息（链表）进行比较，更新场景对象物，实现场景的变化；并且更新链表与当前记录集一致，以便和下一次的数据库信息进行比较。这里包括对链表结构的调用，它的函数在下一小节介绍。

此外，这里还包括三个位置放置的函数 getConX(CString rpos) 、getConY(CString rpos)、 getConZ(CString rpos)，它们把记录集箱位字段的信息转换为 X、Y、Z 三个坐标轴的数值。

5.5 建立链表结点类 container 和链表操作类 Opercon

在 VC.NET 环境下建立一般 C++类，取名为 container，该类主要代表一个链表的结点，每个结点反映一个集装箱的基本信息，所以包括箱位、箱号信息，以及指向下一个结点的指针。在该类的构造函数中，可以通过记录集构造一个结点的数据。

此外建立了一个 Opercon 类，它是对结点类的操纵者，作用是插入、删除结点，统计链表的长度以及清空链表所有结点。它主要解决了指针分配的问题。与数据库比较时，新的结点插入到哪里，删除结点后指针的相应变化等都在这里处理。

这里需要加以说明的是目前这几个类之间的作用关系，文档类是存储基本数据的地方，所以在文档类中定义属于 CVPortSet 类的数据成员 m_VportSet 及属于 Opercon 类的数据成员 m_Opercon;在 CMFCView 中的 OnInitialUpdate()函数中，直接调用文档类的数据成员，任何对数据集以及链表操作类的调用都源于文档类。在 CMFCView 类的 OnTimer()函数中，依靠记录集类与链表结点类之间的比较，得到数据库改变的信息，通过链表操作类来操作链表，将新的结点加入链表中，或是删除已存在的结点。同时，对场景进行更新，加入或删除对象物。通过这几个类的协作关系基本实现了场景的实时渲染功能。

5.6 建立查询类 CQuery

查询类是应程序功能的需要建立的，它是基于对话框生成的类。但是该类继承的不是对话框基类，而是对话栏基类，这两者之间的不同之处在于对话栏可以停靠在窗口的某一个位置而对话框不行。当执行某项操作时可以弹出对话框，这时只是一个浮动的界面；而对话栏具有工具栏和无模式对话框两者的特性，在进行查询的同时用户

依然要能够操纵三维场景，所以一定要用无模式对话框，对话栏正是这样，同时因为它具有工具栏的性质，所以它可以像工具栏那样位于窗口指定的地方，使程序的界面很具有条理性，而且可以安排一些操作让对话栏消失。

查询类的主要作用包括：可以进行箱区选择，这样可以观看堆场中不同箱区的集装箱，还可以根据行、列、层进一步缩小渲染范围，比如当要查询的箱子被挡住时，就可以利用这种过滤来显示；可以按照箱位、箱号、船名、航次等条件来查询集装箱，这种按不同条件查询的方式增加了用户查询的灵活性；在整个对话栏的下部，安排了几个编辑控件，这些控件用文字信息显示出该箱号或该箱位的集装箱属性信息，它适于按箱位和箱号查询的情况，因为这时查询的结果唯一，而按船名、航次进行查询时，查询结果包含若干条记录，因此对于结果中的所有集装箱信息的显示就不能依靠这种控件来处理，所以必须再建立一个对话框，该对话框的相关内容见下一小节。

在该对话栏中，所有的控件都需要建立控件事件以执行相应的操作，所以必须编写事件处理程序，处于数据共享方面的考虑，将事件处理程序编写在 CMainFrame 类中。查询主要涉及对记录集的访问，正如前面视图类调用了文档类中的记录集成员，在视图类中再调用 CMainFrame 类的 setPset(CVPortSet* pSet)函数来将记录集传递给框架类。

5.7 建立列表类 VPListDlg

这里所谓的列表之意是指它的功能类似于一个列表，它主要用来显示按船名、航次进行查询的结果。它其实仍然是基于对话框的类，列表是该对话框中的一个控件。在该类的 OnInitDialog()函数中应该做一些初始化的工作，比如应该划分列表的标题栏，并给每个标题栏一个名字。在该类中还建立一个 additem(CString str[11])，这个函数将利用一个数组来在列表控件中插入一条记录。

因为查询的事件处理程序是在 CMainFrame 中编写的，所以应该在该类中调用列表类的 additem(CString str[11])函数，并且调用之前还应该先将查询的信息转化成数组。

该列表类中包含一个静态文本，用来显示压箱情况，在这里给出某个出口航次的压箱情况综述，指定“压箱率”指标，用户可以通过该指标获知压箱的大概情况。

5.8 建立参数设置类 COption

这个类的建立主要是因为一些参数设置的必要性。在进行数据库跟踪访问时，可以由用户指定间隔时间，当用户不希望场景变化时，还可以指定场景不进行渲染。当进行查询时，符合条件的集装箱会变颜色，这个颜色可以由用户指定，同时按航次查

询时可以根据压箱数的不同显示出不同的颜色，这些颜色的指定也由用户来进行。

它仍然是基于对话框建立的，里面有很多控件，控件的事件处理程序在该类自己中编写。

每次程序运行后，用户可能已经找到了比较理想的参数设置，当再次进入该程序时用户希望所有的参数是自己在上一次进入后已经修改过的，所以程序就涉及一个如何保存用户设置的问题。问题的解决依赖于文档类的操作，将所有参数都保存在文档类中，需要时调用，修改时又能及时反馈给文档类，这样就解决了数据保存的问题。在文档类中应该建立各数据成员与各控件的初始值对应，并且应该建立保存和装载数据的函数，还必须建立一个文件来储存这些变量值。在关闭 CMainFrame 的函数中就可以调用文档类的保存函数，将参数保存到文件中；再次进入应用程序时装载数据，将各参数值初始化。

5.9 工具栏操作

工具栏的操作仍然需要程序控制，对工具栏的操作响应与普通控件一样，也是通过编写事件处理程序来进行的。图 5-3 是应用系统界面的工具栏，工具栏的事件响应程序在CMainFrame中进行，系统会自动在工具栏的ID号和函数间建立消息映射关系。

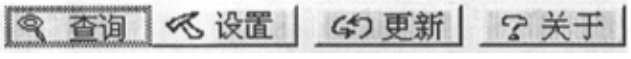


图 5-3　工具栏

当按下“查询”时，会自动显示或隐藏 CQuery 类所在的对话框。按下“设置”时，会显示 COption 类所在的对话框，并将文档的数据传递给控件变量。按下“更新”时，意味着场景要更新以显示最新的数据库信息，这主要用于当在参数设置对话框中取消自动更新功能而由人为更新的情况。不管何种更新，其实调用的都是 CMFCView 类中的 OnTimer()函数。

5.10 数据流向说明

这里的数据流向主要是说明在文档类中定义的数据成员是如何被调用的，它们的关系如图 5-4 所示，简单说明一下它们的含义和相互之间的关系。

在文档类中，一共定义了 14 个数据成员，它们大致分为以下几组：

1）m_color1;m_color2;m_color3;m_color4;m_color5;m_auto;m_time, file1: 它们分别代表查询的颜色，不压箱的颜色，压 1 箱的颜色，压 2 箱的颜色，压 3 箱的颜色，是否进行自动更新，刷新场景的时间和指向保存数据文件的指针。它们都是与

设置类 COption 相关的，也就是它们的值可以通过用户的设置来改变。这里存在两种数据流向问题：

➢ 文件与文档类数据成员

用户根据自己的需要来改变这些参数，也希望这些参数能保存下来，所以在文档类中就定义了保存和加载数据的函数 SaveData()和 LoadData()，实现数据成员与文件之间的数据传输。当用户结束应用程序时，在框架窗口的关闭函数 OnClose()中调用 SaveData()，就将各种参数保存到文件中；当新的应用程序开始时，在文档类的构造函数 CVPortDoc()中调用 LoadData()，将文件中的参数值加载到数据成员中。

➢ 文档类数据成员与其它类数据成员

这是文档类数据成员最普通的用途，那就是在其他类中调用它的数据成员，比如说这里有了这些参数后，当按工具条上的“设置”时，即响应了框架类中的 OnOptionBox()函数，在这个函数中就调用这些参数，把它们加载给设置类的数据成员，保存的数据在这时才真正发挥了作用。

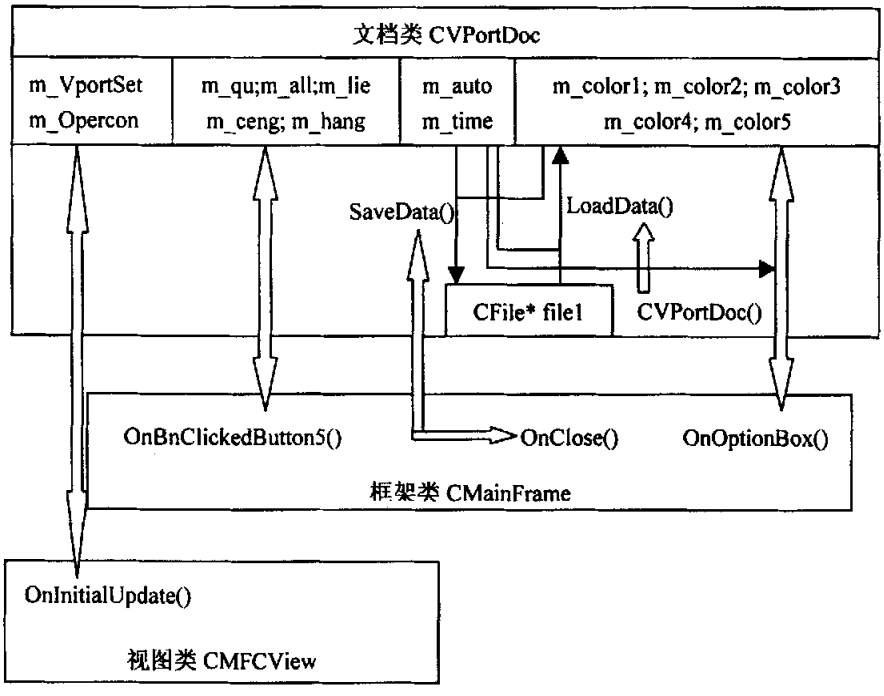


图 5-4　文档类中数据成员的调用

2)m_all;m_qu;m_lie;m_hang;m_ceng:它们分别代表是否显示全部，显示的区号，显示的列号，显示的行号和显示的层号。在文档类的 SetFilter()函数中用后面四个

数据成员的组合构成了数据库的过滤条件，并且执行对数据库的查询。当用户在查询对话栏上选择有关于范围选择的控件时，都会将查询类的数据成员传递给文档类的数据成员，这样也相当于修改了数据库的过滤条件，而且这些控件处理函数都会调用SetFilter()，数据库的过滤条件发生了变化，那么场景也跟着相应的改变。

3）m_VportSet;m_Opercon：它们分别代表记录集类和链表操纵类，在视图类CMFCView的初始化函数OnInitialUpdate()中，将调用文档类这两个数据成员。当查询类改变渲染范围时，改变的实际是上述2）中的数据成员的值，但是由于四者的组合决定了数据库的过滤条件，所以直接决定了数据库的查询结果，当视图类中调用了文档类的记录集时，才使得数据库访问的变化呈现在三维场景中。

5.11 集装箱虚拟堆场应用

集装箱虚拟堆场系统在天津港第二港埠公司集装箱生产过程中实现了初步的应用。运行该系统可进入以下界面（如图5-5）：

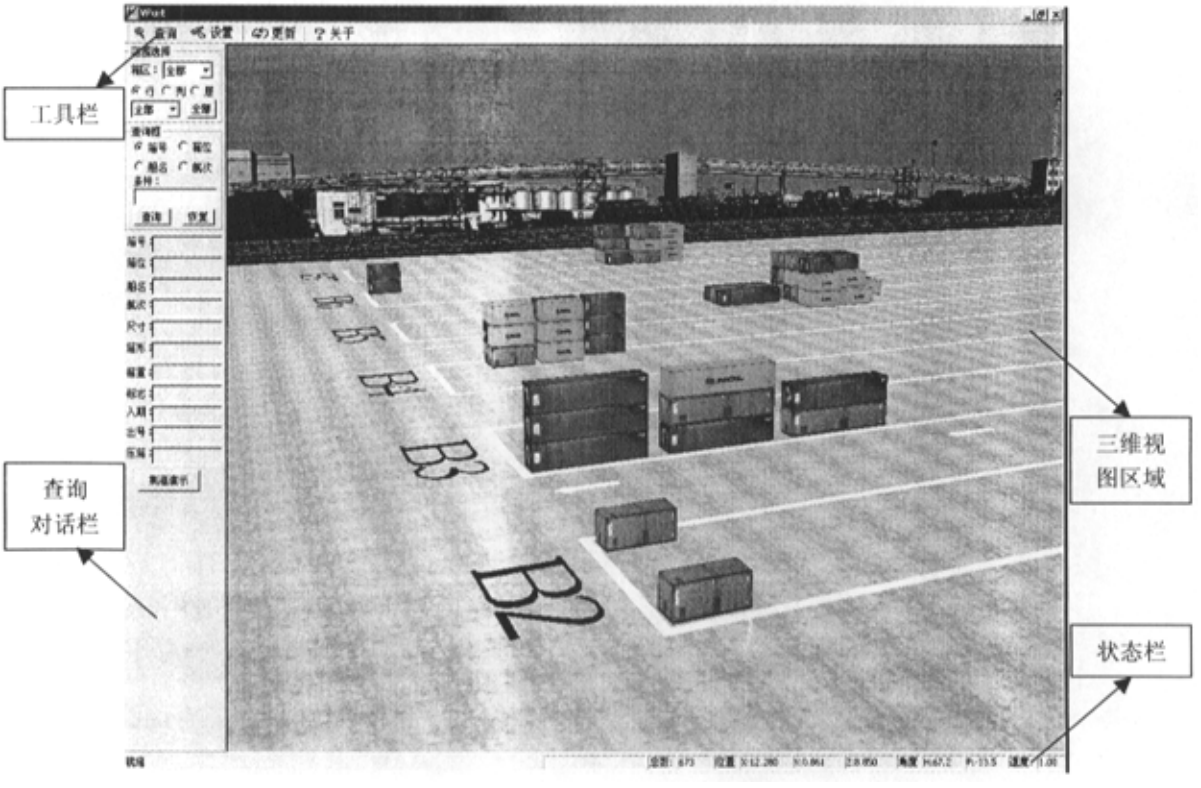


图5-5　程序初始化图

应用系统的初始化界面包括 4 大部分，工具栏，对话栏、三维视图区域以及状态栏。

- 工具栏用于调出相应的对话框，它们各自的功能正如每个工具栏的名称一样。
- 查询对话栏主要便于用户查询箱信息以及设置场景的渲染范围。
- 三维视图区域主要用于完成用户与场景的交互，例如视点的移动，场景图形状态的改变，更为主要的是能够将数据库的变化信息及时的呈现出来，以及将查询结果呈现出来。
- 状态栏的信息是用于帮助用户了解当前的状态，它用来显示在场箱的个数、视点的位置及移动的速度，当用户按键盘相应的键移动场景时，它的数字也不断的发生变化。当按船名或航次查询集装箱时，还会显示符合条件的箱子的数量。

按工具栏上的“查询”可以控制对话栏的显示，当希望三维视图的显示区域大些时，可以按该工具栏，以使查询对话栏消失。同样方法可以让对话栏呈现出来，并停靠在屏幕的左边，这就是对话栏优于对话框的地方。对话栏显示出来后就可以对场景中的集装箱进行查询，查询对话栏如图 5-6 所示，由于位置关系，所以将它一分为二。在“范围选择”中用户可以依照实际情况的需要来渲染场景，在“查询框”中可以选择查询依据（4 个单选按钮），然后输入查询条件，按查询按钮或直接按回车键即可，在对话栏下部区域会显示查询结果，对应该集装箱的箱信息就显示出来，同时三维场景发生相应的变化，符合查询条件的箱子被渲染成红色，如图 5-7 所示。

图 5-6　查询对话栏

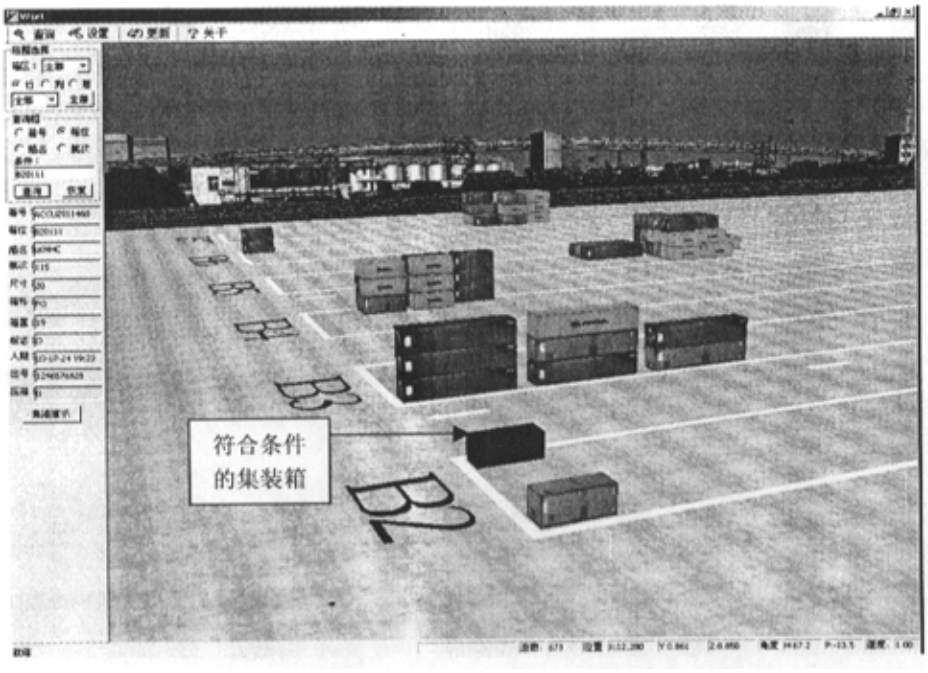


图 5-7　场景的渲染效果

按工具栏上的“设置”时，会弹出如图 5-8 所示的对话框，在该对话框中，可以设置 5 类不同的颜色。查询的颜色是指按箱位、箱号、船名查询时箱子的渲染颜色；后四类颜色是针对按航次查询的情况。场景查询的可视化就是依靠颜色的特殊性，所以颜色的选择要注意，能够使场景越一目了然越好。还可以设置程序是否自动更新，如果允许更新则可以选择更新的时间间隔，这样场景的更新就不受用户的控制，由计算机自动完成，否则需要用户按工具栏上的“更新”才能再次渲染场景。

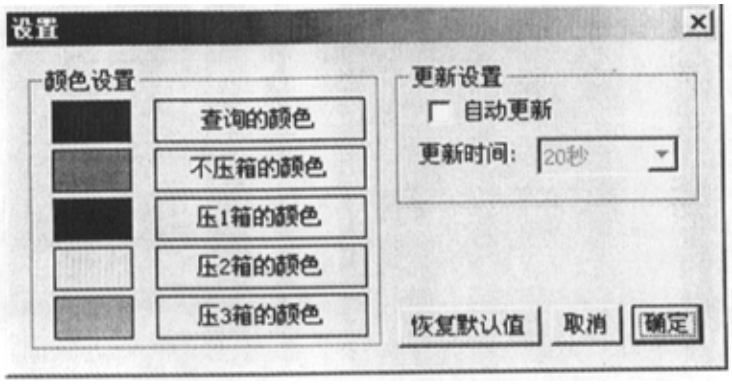


图 5-8　参数设置对话框

不管是否自动更新，总之当场景中新增或删除集装箱时，要起变化的集装箱都会闪动若干次，以便告诉用户该箱位有变化发生，如图 5-9，即为刚加入新的集装箱后场景的变化，由于这里只是一幅静态的画面，所以箱子闪烁的动态过程没有呈现出来，但是当实际使用时，用户恰恰正是利用闪动的效果来得知该箱位有操作发生的。

当用户觉得视点不够理想时，可以按键盘的相应键实现视点的游走，对视点的控制包括可以让视点向前、后、左、右、上、下移动，并且可以让它抬头或低头。这些操作大大方便了用户对整个场景的观看，人们通过一个计算机屏幕就可以了解到整个堆场的信息，甚至能够比实际观看场景更直观，更灵活，这也可以说是本应用系统最大的开发价值。

图 5-9　新增箱位后场景变化

下面介绍一下本程序提供的评判功能。目前为止，本程序的评判功能主要是针对出口箱的发箱顺序给予评定，它的操作已经嵌入到对某一航次集装箱查询的功能中，用户仍然只需输入航次的查询条件。查询的结果分两部分，一部分是场景中集装箱颜色的变化（如图 5-10），另一部分是列表框（如图 5-11），会列出若干条符合条件的记录，同时增加发箱顺序号和压箱数，并在下方的静态文本框中给出综述信息，分别统计了各种压箱数箱子的个数和压箱率。

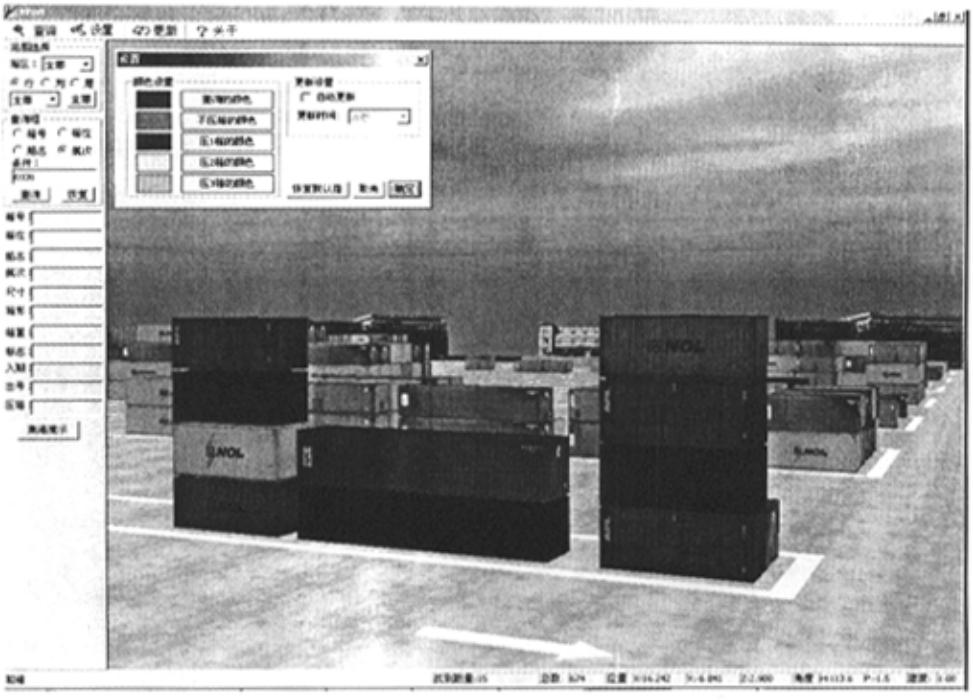


图 5-10　按航次查询时场景的变化

查询结果

箱号	箱位	船名	航次	尺寸	箱形	箱重	标志	入期	出号	压箱
GSTU0900011	A41151	CHAPE	0338	20	FO	16	O	2003-10-31 13:22	13	0
GSTU0900002	A41141	CHAPE	0338	20	FO	10	O	2003-10-31 13:18	12	0
GSTU0900009	A41031	CHAPE	0338	40	FO	26	O	2003-10-31 13:21	11	0
GSTU0900007	A41021	CHAPE	0338	40	FO	20	O	2003-10-31 13:20	14	0
NCSU0380070	A40611	CHAPE	0338	40	FO	52	O	2003-08-13 15:27	15	0
APLU3321001	A10714	CHAPE	0338	20	FO	21	O	2003-10-31 17:47	10	0
XLMO3321002	A10713	CHAPE	0338	20	FO	22	O	2003-10-31 17:47	9	1
GSTU0900001	A10712	CHAPE	0338	20	FO	10	O	2003-10-31 13:18	8	2
GSTU0900005	A10711	CHAPE	0338	20	FO	12	O	2003-10-31 13:19	7	3
GSTU0900012	A10412	CHAPE	0338	40	FO	27	O	2003-10-31 13:23	6	0
GSTU0900008	A10411	CHAPE	0338	40	FO	23	O	2003-10-31 13:20	5	1
GSTU0900004	A10114	CHAPE	0338	20	FO	10	O	2003-10-31 13:19	1	0
GSTU0900003	A10113	CHAPE	0338	20	FO	10	O	2003-10-31 13:18	3	0
GSTU0900006	A10112	CHAPE	0338	20	FO	12	O	2003-10-31 13:19	2	1
GSTU0900010	A10111	CHAPE	0338	20	FO	16	O	2003-10-31 13:21	4	0

找到数量:15；其中不压箱数:10；压1箱数:3；压2箱数:1；压3箱数:1；压箱率:53.00%

返回

图 5-11　列表框中的查询结果

具体分析一下，当用户输入某一航次后会看见场景中某些箱子的颜色发生变化，所有变了颜色的箱子说明都是属于该航次的。在图 5-10 中，明显有 4 个箱子呈绿色、2 个箱子呈蓝色，1 个箱子呈黄色，1 个箱子呈紫红色。对照参数设置对话框，知道绿色代表不压箱，蓝色代表压 1 箱，黄色代表压 2 箱，紫红色代表压 3 箱。由此可见，压箱情况可以由颜色的不同一目了然的呈现出来。如图 5-11 的列表框中已经列出了集装箱的出号和压箱数，与场景中的数据吻合，便于查询某箱子的信息。在列表对话框的下端，给出文字的总结信息和评价指标，使用户对压箱情况有一个定量的了解。

通过渲染，如果场景中大多数箱子的颜色不是绿色，则说明存在太多的压箱，应考虑及时更改发箱计划。这个功能的好处是可以预知实际发箱作业的情况，使用户能够及时更改发箱方案，避免实际作业时大量的翻箱发生。

小　结：本章主要介绍了应用系统具体的开发过程，遵循上一章流程分析中的步骤，结合实际的开发情况进行讲解。介绍了虚拟堆场驱动时涉及的相关内容，主要包括应用程序的预定义和编程二次开发中从构思到实现的种种问题，与上一章共同构成本论文的重点。应用程序的预定义可以对各种面板参数设置初始值，并保存所有设置在应用程序定义文件（*.adf）中，编程时，通过调用该文件实现应用程序的初始化。程序的二次开发主要说明了程序中所使用的类的大致组成情况和各类之间的联系。最后，演示应用系统的效果，进一步分析开发系统所实现的功能和本应用系统开发的意义。

第六章　结论与展望

6.1 总结

本课题基于虚拟现实的思想实现集装箱堆场的可视化，虚拟现实是本课题依托的手段，也为整个的工作流程提供了理论依据。所以沿着这个思路，进行这项课题的研究开发主要包括两大部份的工作：虚拟堆场建模与虚拟堆场的驱动仿真，具体使用的软件也是在仿真行业比较公认的优秀软件，出自同一家公司的 Multigen Creator 建模软件和 Vega 仿真软件。

在具体的工作开始之前，首先学习虚拟现实系统开发的基础知识，了解建模与驱动仿真工作的内容和相互关系，同时从课题出发，宏观的分析用该方法实现堆场可视化的可行性，最后确定利用该方法可以很好的完成三维场景的再现，而且对三维模型的调用和操纵能力都比较强，比单纯用某些程序语言直接开发要方便和简单。

在学习 Multigen Creator 建模软件的过程中，可以进一步了解到虚拟现实系统开发的特点和须注意的问题，学会很多建模技术，而且也进一步学习到每一种建模技术的适用情况以及为何要采用这样的建模方法。在 Multigen Creator 中，基本采用多边形建模方法，可以说这正是为实时性考虑的一个表现，当然为了保证渲染的连贯性，该软件中也提供了很多简化模型的方法。在建模过程中应时刻注意不要使最后的场景模型过于复杂，一切为了实时处理的考虑。建模过程本身就是熟悉虚拟系统建模方法的好机会，通过建模，切身体会到了一些优化操作的必要性，比如会在建模过程中或是建模之后有意识的删除一些根本不可见的面，会在建模的时候考虑哪些地方是可以用纹理取代建模等。

仿真软件 Vega 的学习分两步进行。首先学习 Vega 的界面系统 LynX，在这个系统中，了解仿真其实就是在操纵一些类的变化，诸如窗口、通道、对象物、场景、系统、观察者、运动模型、环境、路径等，通过对这些类面板的操作，可以改变这些类中的相关参数以满足不同的需要，但是由于这个界面系统不能实现较复杂的功能，所以必须对它进行二次开发。因为 Vega 包括完整的 C 语言应用程序接口，所以采用功能比较强大的 VC.NET 作为开发平台，编写程序实现虚拟堆场的功能。编程需处理好的两大问题:其一是 VC 与 Vega 的连接；其二是 VC 与数据库的连接。与 Vega 的连接可以保证三维场景进入到 windows 的视窗中来，而与数据库的连接则是驱动场景的源动力，应用程序就是通过读取的数据库信息来控制三维场景的变化。

本课题的最终结果是要得到一个应用系统，运行该应用系统可以让三维场景实时

的跟踪数据库的变化，同时可以依据用户指定的条件对堆场上的集装箱进行查询，查询的结果在三维场景中是通过改变集装箱的颜色体现出来的，通过控件和对话框也可以将查询的结果以文字的方式体现出来，两种信息的综合可以让用户更全面直观的了解在场箱的信息。本系统也涉足于方案评判的研究，可以说是一些初步的尝试，主要针对出口箱发箱顺序给予一定的评价，它可以预先模拟出按照某发箱顺序发箱时的压箱情况，对于压箱数不同的集装箱用不同的颜色来渲染，使用户能一目了然地看出在什么地方的发箱顺序安排上存在问题，便于在实际的作业之前进行调整，以便保证发箱过程的最优化。

6.2 展望

在本课题的研究中，仍然有很多需要进一步改进的问题，这样才能使这个实时跟踪系统具有更好的实用性。

1、对数据库管理系统进行更深入的研究，因为采用触发器传递消息给应用程序是最实时的办法，关键是要解决如何让它与 VC 之间的连接做到方便可靠。

2、学习码头管理方面的知识，给程序找到最佳的切入口，也就是说要让程序最能为用户所想，满足他们最切实的需求。

3、增加程序的智能化，不仅对出口发箱给予评判，还要针对需要在多方面给予功能评价，就像预知维修一样能够预知方案的好坏。如果可能，让它自动参与到计划的安排中就更为理想，不过这需考虑的因素很多，必须经过大量的实践研究才可能实现。

4、增加程序的可视化效果，驱动各种三维模型能够协调运动起来。尽管应用系统已能够让集卡按要求运动，但还需让集卡和场吊随着箱子的变化一起协调运动起来等。

致　谢

本课题的研究开发过程已经告一段落，其间凝聚了很多人的心血，正是有赖于大家共同对我无微不至的关怀和帮助，使得我的论文工作得以顺利完成。

首先要感谢我的导师宓为建教授和徐子奇老师。两位老师以严谨的治学态度和广博扎实的专业知识为我的求学之路不断引航指点。在学习阶段，他们认真教授我各种理论和基础知识，使得我为论文阶段打下了较好的基础；进入课题研究阶段，为了配合课题的开发，从硬件平台的搭建、建模仿真软件的建立到实际码头的考察实践等各个环节，他们都给予了我大力的支持，为我营造了十分理想的开发环境。在论文的后期阶段，他们认真阅读我的论文，给我提出了许多宝贵意见和建议。在此，我对两位老师表示最崇高的敬意和感激！

同时，我要感谢在整个研究生阶段授予我知识的各位老师，特别是梁刚老师在论文的起步阶段给予我很多的启迪和帮助；也要感谢我的许多同学，能够在课题的研究过程中毫无保留的与我分享他们自身的知识、经验和资料；还要感谢我的父母，正是因为他们对我无私的关心和照料，使我能够专心于课题的研究。

最后，让我再次向各位老师、同学以及我的父母表示最诚挚的谢意！

参考文献

[1] 杨波. 虚拟现实技术——MultiGen，Vega 应用研究：[学位论文]. 电子科技大学，2001

[2] 刘晓波，张琴舜，张和林. 一个基于 MultiGen-Vega 的虚拟场景漫游系统. 计算机应用，2002（12）

[3] 张茂军. 虚拟现实系统. 北京：科学出版社，2001

[4] 杨宝民，朱一宁. 分布式虚拟现实技术及其应用. 北京：科学出版社，2000

[5] 吴重光. 仿真技术. 北京：化学工业出版社，2000

[6] 刘国香. 虚拟环境技术. 北京:中国铁道出版社, 1996

[7] 肖曙红. 基于虚拟现实的原型分析技术探究. 机械工艺师, 2000(8)

[8] 陈正鸣. 虚拟现实在 CAD/CAM 中的应用. 计算机工程, 1999(7)

[9] 朱恒. 虚拟制造系统建模与仿真. 中国机械工程, 1996(7)

[10] 周江华. 基于 OpenGL 的制造系统虚拟仿真环境研究. 测控技术, 2000, 19(7)

[11] 孙连胜. 虚拟制造中生产线可视化设计. 北京理工大学学报, 2002(2)

[12] 吴振宇. 虚拟场景中模型真实感表现方法的研究与实践：[学位论文]. 武汉：武汉理工大学，2001

[13] 任鸿翔，王科伦，金一丞. 光照模型与 Creator 中的明暗处理. 大连海事大学学报，2003（5）

[14] 陈维兴, 林小茶. C++面向对象程序设计教程. 北京:清华大学出版社, 2000

[15] 马安鹏. Visual C++ 6 程序设计导学. 北京：清华大学出版社, 2002

[16] David J. Kruglinski 著. Visual C++6.0 技术内幕. 修订 5 版. 北京：清华大学出版社，2001

[17] 杨竞锐，张连卫，王贵新. Visual C++.NET 深入编程. 北京：北京希望电子出版社，2002

[18] 《The MultiGen Creator Desktop Tutor》. MultiGen-Paradigm Inc. 2001

[19] 《Creating Models for Simulation》. MultiGen-Paradigm Inc. 2001

[20] 《The MultiGen Creator HelpSummary》. MultiGen-Paradigm Inc. 2001

[21] Cohen J, Wright W. Simplification envelopes [A]. In: Proceedings of SIGGRAPH'96[C], Held in New Orleans, Louisiana, August 1996: 119～128.

[22] Gueziec A. Surface simplification inside a tolerance volume[R]. Technical Report RC 20440, IBM Research Division, T. J. Watson Research Center.

[23] 冯德俊. 基于虚拟现实技术的城市三维建模：[学位论文]. 四川：西南交通大学，2001

[24] 关沫. 交互式三维场景生成技术的研究与实现：[学位论文]. 沈阳：沈阳工业大学，2001

[25] 关克平. 航海模拟器视景建模技术研究及应用：[学位论文]. 上海海运学院，2002

[26] 李瑞. 基于 MultiGen CreatorPro 建立的虚拟环境中的几何模型. 计算机工程, 2002(8)

[27] 任庆东. 场景模型的建立及快速显示. 大庆石油学院学报, 2002(6)

[28] 宋志明. 水下航行器视景仿真系统的研究. 系统仿真学报, 2002(6)

[29] 魏刚，宋裕农. 基于 MultiGen 构建舰船三维模型的探讨. 青岛大学学报，2002（12）

[30] 龚卓蓉. LynX 图形界面. 北京:国防工业出版社, 2002

[31] 龚卓蓉. Vega 程序设计. 北京:国防工业出版社, 2002

[32] 王华清. 基于分布式虚拟现实技术的飞行仿真系统研究：[学位论文]. 南京：南京航空航天大学，2002

[33] 徐东平. 实时动态交互视景仿真：[学位论文]. 武汉：武汉理工大学，2001

[34] 徐轶超. 基于虚拟现实技术的机舱漫游系统：[学位论文]. 武汉理工大学，2003

[35] 张新艳. 基于虚拟现实的港口集装箱码头装卸系统仿真建模技术. 武汉理工大学学报, 2001(12)

[36] 张煜. 仿真技术在港口集装箱装卸作业中的应用. 武汉交通科技大学学报. 2000(12)

[37] 李青. 虚拟现实技术及矿井工业广场的可控可视化研究. 计算机仿真, 2002(3)

[38] 李青. 矿井工业广场的可控可视化与虚拟环境开发. 中国矿业大学学报, 2002(1)

[39] 寿建敏, 吴志雄. 港口运输系统全面应用计算机仿真设计探析. 水运管理, 1997

[40] 徐东平. 港口集装箱业务可视化管理信息系统. 计算机辅助工程, 1998

[41] 真虹. 集装箱码头生产过程动态图形仿真优化的研究. 中国图像图形学报, 1999

[42] 秦永宏, 陈强努. 港口集装箱装卸工艺系统的计算机仿真. 计算机辅助工程, 2000

[43] 申闫春. 虚拟现实技术在露天矿生态重建仿真中的应用. 中国矿业大学学报, 2002(1)

[44] 申闫春. 露天矿生态重建虚拟现实仿真系统. IM&P 化工矿物与加工, 2001(11)

[45] 李丽荣. 在SGI 图形工作站上实现低空突防三维视景及跨平台实时仿真. 南京航空航天大学学报, 2002(4)

[46] 李瑞. Vega 程序设计在 MFC 中的应用. 计算机工程与设计. 2002(8)

[47] 凌江春，胡庆夕，杨小兵. 基于 Vega 的虚拟装配研究. 制造业信息化，2003（8）

论文独创性声明

本论文是我个人在导师指导下进行的研究工作及取得的研究成果。论文中除了特别加以标注和致谢的地方外，不包含其他人或其他机构已经发表或撰写过的研究成果。其他同志对本研究的启发和所做的贡献均已在论文中作了明确的声明并表示了谢意。

作者签名：舒帆　　日期：04.01.10

论文使用授权声明

本人同意上海海运学院有关保留、使用学位论文的规定，即：学校有权保留送交论文复印件，允许论文被查阅和借阅；学校可以上网公布论文的全部或部分内容，可以采用影印、缩印或者其它复制手段保存论文。保密的论文在解密后遵守此规定。

作者签名：舒帆　导师签名：[illegible]　日期：04.01.10