技术应用

基于卫星仿真的三通道 3 维视景系统

黄海峰
（中国科学院空间科学与应用研究中心，中国科学院研究生院，北京 100080）

杨 新 顾 明
（中国科学院光电研究院，北京 100080）

随着计算机软硬件以及网络技术的发展，特别是高性能图像生成系统、高性能计算处理系统和高速网络技术的飞速发展，使三通道分布式视景仿真成为可能。所谓三通道分布式视景仿真就是，先将一幅动态的画面分成 3 个部分或者两个部分，然后由各通道对应的投影设备将它们投影到大型圆柱型屏幕上，再通过三通道边缘融合与非线性失真校正技术在画面的边界处实现无缝连接，重新组成一幅流畅、完整的画面，这样做的目的是为了形成 135°的水平视场角，以便给用户获高度沉浸其中的感受。

基于卫星仿真的三通道 3 维视景系统是应用 3 维可视化技术，为分布式卫星姿态、轨道控制仿真系统开发的一个视景系统。它能实时地将卫星在空间的飞行姿态和轨道控制过程立体、动态地展示给用户。因此，视景系统开发的意义有以下几点：

（1）可作为卫星仿真系统的重要辅助功能之一

将数字、图表化的卫星仿真过程通过 3 维模型动态地展示出来，是整个系统设计自动化的重要组成部分，它不仅可有效地提高卫星总体设计水平，而且它是整个卫星仿真系统不可分割的的重要组成部分；

（2）可作为卫星姿轨控仿真模型的一种验证方法

卫星的位置和姿态以及有效载荷的工作状况，都可由视景系统以 3 维效果实时地演示给用户，使用户可以很直观地在视景系统中观察和判断仿真状况；

（3）仿真视景系统可以直观地显示卫星运行的动态过程，让仿真人员可以迅速地对飞行器姿轨控的整个过程和事件有清晰的了解和认知；

（4）可以作为一种重要的宣传手段和窗口。

1 开发工具简介

该视景系统以 Windows 2000 为操作系统平台，使用 MultiGen_Creator 创建 3 维模型，同时应用专业场景驱动软件 MultiGen_Vega 为渲染引擎，并通过能提供应用程序定义文件（high level architecture，HLA）服务的软件 pRTI 从其他仿真计算机接收数据。

1.1 MultiGen_Vega

Vega 包括图形环境界面（lynx）、C 语言应用程序接口 API、相关实用库函数和一批可选的特定功能模块，其能够满足视景仿真的要求。

其中图形界面 Lynx，是一种点击式图形环境，包括应用开发所需的全部功能。由于用 Lynx 可以快速地初始化 Vega 程序需要的所有要素，简单迅速地创建、编辑、运行复杂的仿真应用，其不仅可大幅度减少源代码的编写，并且可以预览初始化的效果，还可使软件的进一步维护和实时性能的优化变得更容易，从而大大提高了开发效率。

1.2 MultiGen_Creator

MultiGen_Creator 是高效的、交互式的软件包，它整合了大量的工具来建立"所见即所得的"层级可视数据库环境，其产生的数据库遵循 OpenFlight 数据格式；数据库里的每个节点类型都有数据属性和函数。主要的节点类型有：数据库头（database Header）、组（group）、对象（object）、多边形（polygon）、顶点（vertex）。其他还有光源、光点、道路、路径、文本、声音、细节层次、切换等节点，而熟悉这些节点的特性就能更加高效地建立满足自己需要的 3 维模型，例如本软件中推力器的火焰、星上相机的摆动、数传信号的纹理更新等就用到了有关节点的属性以及相关 API。

1.3 HLA

HLA 是一个通用的仿真技术框架，它定义了构成分布交互仿真各节点的功能和相互关系。

其主要目标是用于解决仿真系统的集成问题和为构造大规模仿真应用提供一种应用集成方法。HLA 将实现某种特定仿真目的的仿真系统称为联邦（Federation）。联邦由联邦对象模型、若干联邦成员和运行时间支撑系统（run-time infrastructure，RTI）构成（如图 1 所示），成员之间的交互通过 RTI 提供的服务来实现。在这种结构中，RTI 从某种程度上来说，可以看成是一种"软总线"，联邦成员可以在联邦运行

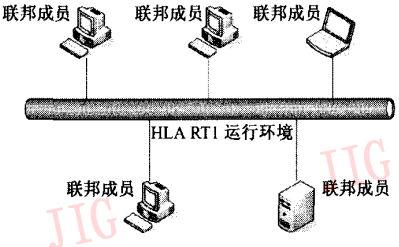


图 1 HLA RTI 运行环境

过程中随时“插入”。

2　视景仿真软件的实现

2.1　设计流程

首先使用 MultiGen_Creator 建模软件建立卫星、地球、月球、太阳、星空、地形、数传信号、卫星推力器的喷射火焰、与卫星固连的坐标系等模型,以组成所需要的场景;然后在 VC 下,用 MFC 方式编程创建一个 Vega 线程,并在该线程中利用 Vega 的 API 函数来驱动和渲染整个场景;接着再创建一个 HLA 的线程,该线程负责创建和加入联邦,用于实现与其他联邦成员的仿真同步,并以视景系统为一个联邦成员从其他负责仿真模型计算的联邦成员那里接收相关的位置、姿态、指令等数据,并将它们传给 Vega 线程。

2.2　具体实现

2.2.1　3 维建模

建立的 3 维模型包括:卫星、地球、太阳、星空、数传信号、相机光束、卫星推力器火焰等。

为了增加 3 维模型的真实感和动态感,同时又要简单易行,不增加额外的工作量和费用,在建模过程中可都尽量使用 Creator 软件包中自带的基本功能或工具,而不采用编程的方式。

例如,为了使用 Creator 自带的动画功能实现助推器喷射火焰的动画效果,可先将火焰模型简化成由两个面交叉组成的模型,尽管这样整个火焰模型设计有 9 个对象节点,但是只有一个对象节点显示,它们都是火焰模型组节点的子节点,每个对象节点包含 2 个多边形节点,而这 2 个多边形节点也就是前面所说的两个交叉面在数据库中的形式,各个多边形节点的纹理根据效果的需要都不相同;然后火焰模型的组节点被设置成动画属性,这样火焰模型的组节点就能让 9 个对象节点依次循环地显示,并给人以喷射的效果,因此能够满足应用的要求。

另外,为了使用户能观察到星上相机在地面上的拍照范围,还为相机建立了一个类似探照灯光束一样的模型,并将它和卫星做成一个整体。由于光束的方向会随着星上相机的镜头方向而不断改变,因此在它与卫星之间增加了一个自由度节点,这样就可以在程序中通过调用 OpenFlight API 来对它进行操作,以达到光束随相机镜头指令而摆动的效果,其实这样的应用实例还有很多,如数传信号等。

2.2.2　程序参数初始化

使用 Lynx 设置程序中一些基本的对象来为程序制作调用高体系结构层(application definition files,ADF)文件,是减少编程量的有效手段。因此笔者创建了一个 ADF 文件,并在文件中初始化了 GraphicsStates、Windows、Channels、Observers、Scenes、Object、Players、Environments、Environment Effects 等程序所需要的基本对象的初始值,以供 Vega 线程调用。同时还可以根据程序需要,调用相关的 API 函数来重新设置和调用这些对象的属性值,以满足演示的需要。

2.2.3　程序实现

各对象属性初始化完成以后,为了实现太阳、月球、地球、卫星等模型的运动,系统还建立了相应的场景运动体,并使它们与对象体联系起来,以便通过调用 pRTI 提供的服务从各个仿真节点计算机上获得各场景运动体的位置、方位角以及指令数据来驱动它们。为了能使整个仿真过程同步起来,还利用 pRTI 中的时间管理机制在每一个帧循环里使用接收到的位置数据来更新场景运动体的位置,以便真正作到“所见即所仿真”。下面就程序实现过程中的几个重点事件的实现方法做简要介绍:

(1)在 MFC 方式下创建 Vega 线程

对于 Windows NT 平台上的 Vega 应用,主要有以下 3 种类型:控制台程序、传统的 Windows 应用程序和基于 MFC 的应用。但无论是哪一种应用,建立 Vega 应用都基本需要进行以下 3 个步骤:

①初始化:初始化 Vega 系统,并建立共享内存以及信号量;

②定义:通过 ADF 应用定义文件初始化程序的要素以及模型;

③配置:通过调用配置函数来完成 Vega 的配置。

接下来是 Vega 应用的主循环,其作用是对 3 维视景进行渲染驱动。分为以下两步:

①对于给定的帧速进行同步;

②对当前的显示帧进行必要的处理。

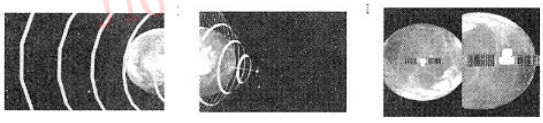
以下就是程序中基于 MFC 应用的 Vega 线程主体部分,它保证了画面能在 Windows 窗口中显示、渲染。

```
vgInitDv( argc, argv, NULL) //初始化主从设备
vgInitWinSys( ); //初始化
//获得窗口句柄来初始化 Vega 的显示窗口
vgDefineSys( "myapp. adf" ); //定义
vgConfigSys( ); //配置
while ( pOwner - > getContinueRunning( ) )
{
pOwner - > key_input( pOwner - > m_windows);
vgSyncFrame( );
vgFrame( );
pOwner - > Circulate( );
//application specific code
}
```

(2)三通道视景显示的实现

由于视景系统设计了 3 个观察者,每个观察者对应一幅画面,因此用户将看到 3 幅观察方式和角度都迥然不同的画面,其中第 1 幅画面占用了两个通道,剩下的两幅画面共用一个通道,其画面效果如图 2 所示。

其中对应第 1 幅画面的观察者在太空中的位置可以自由调节,这样就可以让人们对卫星姿态、轨道控制(后面简称



(a) (b) (c)

图 2 三通道视景显示效果图

如前所述要实现 3 条通道必须有 3 台工作站分别对应 3 条通道，每条通道数据由一台工作站负责渲染，其中如图 2 所示的对应中间通道的工作站定义为主设备，另外两台为从设备。

接下来需要确定三通道的数据通信方式，因为当主设备作为联邦成员从其他仿真计算机获得位置、姿态、指令数据后，它还需要将这些数据传给从设备来分享，以保证它们也能同步渲染自己对应的通道，这样才能形成完整的视觉效果。其实在 Vega 内部就有主从设备位置、姿态数据的通信解决方式——数据插播；另外一种通信解决方式就是采用 WinSock 方式自己编写网络接口来传输数据。本文采用了后者，原因是 Vega 在数据插播时，只能传位置、姿态数据，而无法将模型的纹理更换属性传给两个从设备，这样就无法从设备上看到数传信号、火焰等模型的动画效果，而且考虑到该视景平台的重复利用，需保证以后不在 HLA 环境下也能运行，所以主设备采用 Vega 内部输入插播的方式，而数据通信采用 WinSock 编程来实现，这样就能保证达到预期的三通道演示效果，而且能更方便地将主设备上的键盘输入利用 Vega 的输入插播方式传给从设备，而不用另外编程。

(3)创建视景联邦成员线程

创建视景联邦成员线程是为了让主设备能从联邦中的其他成员计算机上接收视景所需要的数据。在视景软件启动之前，管理者联邦成员就已经在分布式平台上建立起了一个空间飞行器仿真联邦，这样当视景联邦成员线程被启动后，视景系统便作为视景联邦成员加入到了该联邦，并等待所有联邦成员的加入和数据的发送。

这一线程主要采用了 pRTI 提供的服务接口函数来完成相应的功能，同时它要将接收到的数据传给 Vega 线程，供 Vega 作为场景的驱动数据。线程建立的部分代码实现如下：

```
Visual _ Federate:: runFederate ( CMFCVegaView * pCurrentView )
{.....
  federateThread = AfxBeginThread ( runFederateApp, this );
  .....}

CMainFrame::CMainFrame( )
{.....
VisualFederateii = new Visual_Federate ( );
VisualFederate - > runFederate ( m_pCurrentView );
.....}
```

当程序创立该联邦成员线程后，要经过 3 次同步，然后才进入联邦仿真主循环，在主循环内接收数据，并传给 Vega 线程，其主要代码实现如下：

```
while ( VisualFederate - > getContinueRunning( ) )
    { //将其他联邦成员的值传给视类
    VisualFederate - > ToVega( );
    //申请时间推进
    VisualFederate - > requestTimeAdvance ( VisualFederate - > m_currTime + VisualFederate - > m_timeStep); //等待时间推进
    while ( ! VisualFederate - > bl _ VisualIsTimeGranted)
    {
    VisualFederate - > _rtiAmbassador. tick ( 0. 01, 1. 0);}
    VisualFederate - > bl_VisualIsTimeGranted = RTI:: RTI_FALSE;
    }
```

3 结 语

本文介绍了如何利用 Multigen_Creator、Multigen_Vega 和 pRTI 来建立三通道立体分布式实时仿真系统视景部分的全过程，并在 VC 环境下实现了系统的功能。

系统的开发过程表明，利用面向对象的程序设计技术、采用先进高效的虚拟仿真开发工具和综合运用多学科技术是开发视景仿真类系统的有效方法。实践证明，在 Multigen_Creator 中建立 3 维模型以及基于 MFC 的应用，在 Vega 和 HLA 开发环境中实现分布式视景仿真系统，不仅开发周期短、效率高，而且能取得很好的仿真效果。