计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(12):105−111 [doi: 10.15888/j.cnki.csa.007190]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 OSGEARTH 的国产航天三维仿真软件设计①

施　斌, 王　华, 伍辉华, 姚宇婕, 杨　洋

(中国卫星海上测控部, 江阴 214431)
通讯作者: 施　斌, E-mail: tempsb@126.com

摘　要: 针对目前航天领域缺乏经过验证、能够可靠应用于全国产化平台的三维仿真软件, 本文基于 OSGEARTH 设计和实现了一款国产航天三维仿真软件. 通过地球影像和高程数据的差异化加载、场景树的优化构建、弹道数据筛选等设计策略, 解决了在全国产化平台上进行三维仿真的可靠性和渲染效率问题, 并得到了仿真实验的有效验证.

关键词: OSG; OSGEARTH; 三维仿真; 国产化

引用格式: 施斌,王华,伍辉华,姚宇婕,杨洋.基于 OSGEARTH 的国产航天三维仿真软件设计.计算机系统应用,2019,28(12):105–111. http://www.c-s-a.org.cn/1003-3254/7190.html

Design of 3D Space Simulation Software for Domestic Platform Based on OSGEARTH

SHI Bin, WANG Hua, WU Hui-Hua, YAO Yu-Jie, YANG Yang

(China Satellite Maritime Tracking and Control Department, Jiangyin 214431, China)

Abstract: Now, space field lacks 3D simulation softwares that are verified and can be used on all-domestic platform reliably. We introduce some design strategies, such as differentially loading Earth image and elevation data, optimized build of scene graph, filtering of trajectory data, and so on. The proposed software can solve the reliability and rendering efficiency problems in 3D simulation on all-domestic platform. Some simulation experiments have been conducted to valid reliability and rendering efficiency.

Key words: OSG; OSGEARTH; 3D simulation; domesticization

引言

目前, 航天领域的各类三维仿真软件仍然较依赖于国外成熟商用组件 (如 STK[1,2]), 或者依赖于 X86 架构的硬件平台, 无法完全满足“自主可控”要求. “自主可控”就是要依靠自身研发设计, 全面掌握产品核心技术, 实现信息系统从硬件到软件的自主研发、生产、升级、维护的全程可控. 目前, 国内已经开展了很多基于 OSG 或 OSGEARTH 的跨平台研究和应用, 主要应用于虚拟校园漫游、城市三维场景、空战模拟场景、战场态势等方面[3–9], 在航天三维仿真领域应用[10–13]不多, 在全国产化平台 (国产 CPU 板卡和国产操作系统) 上经过可靠性和性能验证的更是空白.

本文尝试针对全国产化平台进行航天三维仿真软件的设计和实现, 以解决在全国产化平台上进行三维仿真的可靠性和渲染效率问题.

1　相关概念

1.1　国产化

在 CPU 领域, 目前较主流、成熟的国产化 CPU 是飞腾和龙芯. 飞腾 CPU 采用 ARM 指令集, 目前主推的是 FT1500A 芯片, 龙芯 CPU 采用 RISC 指令集, 目前主推的是龙芯 3A3000 芯片, 主频均达到 1.5 GHz.

① 收稿时间: 2019-04-15; 修改时间: 2019-05-16; 采用时间: 2019-06-11; csa 在线出版时间: 2019-12-10

国产化 CPU 芯片与 X86 芯片 (主频大多在 2.8 GHz 以上) 相比, 主频相对落后, 使用体验上存在明显差距, 特别是在软件编译、图形绘制等计算量大的任务中更为明显. 在显卡领域, 目前尚未推出成熟的国产化显卡产品, 暂时仍依赖国外显卡. 在操作系统领域, 目前已出现了中标麒麟、银河麒麟、深度 Linux 等多种国产操作系统, 已经具备较强的替代能力, 但在运行稳定性、性能等方面存在短板.

1.2 OSG 与 OSGEARTH

OSG 是一个开源的场景图形管理开发库, 为图形图像应用程序的开发提供场景管理和图像渲染优化功能[14]. OSG 具备跨平台特性, 可以在大部分 CPU 上编译通过, 可以运行于 Windows、Linux 等大多数操作系统, 具备线程安全性.

OSGEARTH 是一个基于 OSG 的开源跨平台类库, 提供了一个地理空间 SDK 和地形引擎, 可以从数据源构建可视化地形模型和影像, 方便快速构建基于三维数字地球的各类应用, 在各领域已有广泛应用.

2 系统设计

2.1 系统架构

航天三维仿真软件的系统架构如图 1 所示, 底层是麒麟国产操作系统, 第二层的 Qt 提供基本界面功能支持, 第三层的 OSG 提供三维仿真基础框架和功能, 第四层的 OSGEARTH 提供三维数字地球相关的应用接口, 顶层是三维仿真软件. 三维仿真软件主要包括了场景资源初始化加载、三维场景管理、关键事件与视点控制、仿真动画控制、粒子系统控制、数据驱动、弹道数据筛选、弹道外推、地球自转及轨道推算、测站自动跟踪、多屏切换、场景配置等主要功能模块.

关键流程如下: 软件初始化时基于 OSGEARTH 加载地球影像和高程数据从而构建三维数字地球场景, 然后通过网络接收弹道、关键事件等各类任务数据, 弹道数据经过筛选、插值处理后驱动飞行器位置变化, 关键事件数据通过事件机制触发相应动画播放, 各模块协同实现飞行器三维仿真.

2.2 地球影像和高程数据的组织

在三维数字地球的构建过程中, 涉及到地球的影像和高程数据的组织. 软件通过 OSGEARTH 自动加载指定位置的影像和高程数据, 叠加渲染在地球表面. 考虑到影像和高程数据量大, 非常消耗软硬件系统资源, 而全国产化平台的性能一般, 因此必须对影像和高程数据的加载进行优化.

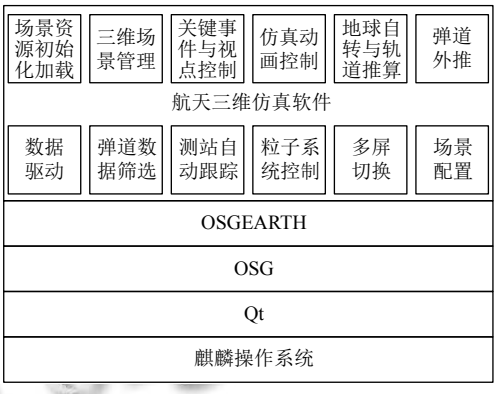

图 1 软件系统架构图

对于影像数据, 地图影像数据量很大, 10 级全球地图影像大约需要 15 GB, 而道路标注影像数据相对较小. 考虑到高精度影像数据不但占用空间, 在进行三维场景渲染时也非常耗费资源, 影响渲染效率. 因此, 软件采用分区域分层方式加载不同精度影像数据, 共分三层: 底层采用中等精度级别 (如 10 级) 的全球地图影像, 中间层采用较高精度级别 (如 16 级) 的全球道路标注影像, 上层采用较高精度级别的各发射场区域地图影像 (如 11–16 级). 当在火箭起飞离地面较近、可视地域较小时调用发射场区域的高精度数据, 当火箭或飞行器远离地面、可视地域较大时调用全球低精度数据, 而道路标注层数据量不大, 采用全球高精度数据.

对于高程数据, 同样存在数据量大的问题, 但简单地通过上述方式加载高程数据无法避免出现较明显的“地形断裂”现象. 因此, 软件仅在中国区域加载了一层中等精度级别 (如 10 级) 的高程数据, 其他区域未使用高程数据. 当火箭起飞至进入大气层前, 恰在中国区域范围内, 此时距离地面较近, 加载中等精度地形, 当火箭进入大气层后, 此时距离地面较远, 高空俯瞰对地形的关注程度降低, 没有加载地形基本不影响整体演示效果.

通过差异化的加载策略, 在不同场景切换时, 影像和地形不会出现明显突变现象, 最大限度地减少了实时渲染的数据量, 有效提升了渲染效率, 保证了流畅的仿真效果.

2.3 三维场景的组织与构建

OSG 中存在场景树的概念, 这棵树是一棵由 Node

(节点) 组成的树, 反映了场景的空间结构和可绘制对象的状态. 场景树结构的顶部是一个根节点, 从根节点向下延伸, 各级树枝上的 Group 节点 (组节点) 包含了几何信息和用于控制其外观的渲染状态信息, 底层的 Geode 节点 (叶节点) 包含了场景中物体的实际几何信息[15]. 通过建立场景树, OSG 可以高效地渲染对应的三维场景.

根据场景树的定义, 构建出整个三维仿真场景对应的场景树结构, 如图 2 所示.

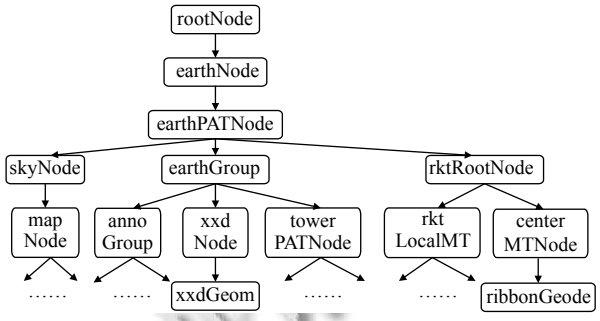

图 2 三维场景组织图

顶层的 rootNode 是整个场景的根节点, 第二层的 earthNode 表示整个地球节点, 所有具体的场景节点都隶属于该节点控制, 第三层的 earthPATNode 是地球惯性系旋转节点, 可以控制地球自转, 其下属所有节点均随着该节点的旋转变化而进行相应的相对变化, 第三层有 3 个分支: 左侧的 skyNode 及其下各级节点是通过 OSGEARTH 加载外部的影像和高程数据后自动创建的, 描述了天空和地球地形; 中间的 earthGroup 是包含了各种附属于地表模型的组节点, 第四层的 3 个节点分别代表测站、星下点轨迹和发射塔架, 第五层及以下均是各类测站、星下点轨迹和发射塔架的具体组成节点; 右侧的 rktRootNode 是火箭的根节点, 第四层的 rktLocalMT 是火箭局部位移节点, 用于完成火箭整体模型加载后从模型坐标系到真实火箭坐标系的转换, 其下各级节点是火箭的各级部件和载荷的分层节点, centerMTNode 是火箭中心位置节点, 用于在火箭部件分离后火箭中心位置的调整, 其下的节点是火箭轨迹线叶节点.

某级节点的旋转平移就可以改变对应仿真对象的位置姿态, 增加和删除某个节点就可以显示和隐藏相应仿真对象. 在运行时, OSG 将会按照以上的场景树结构逐层裁剪、渲染三维场景, 通过合理、灵活地组织、调整场景树, 即可高效地完成场景的渲染, 有效提升在全国产化平台上的渲染效果.

2.4 弹道数据筛选方法

通过网络接收以驱动飞行器模型的弹道数据可能存在非正常数据 (主要是相对飞行时出现跳动), 因此在后端弹道数据插值处理前应先进行相应的筛选, 避免不必要的计算处理, 有效提升飞行器仿真流畅程度.

整体流程只要满足网收启动的条件就循环执行. 内部基本流程如下: 首先, 如果没有新数据到达或为非弹道类数据, 则结束本轮处理, 否则继续根据弹道数据和起飞时间计算当前弹道飞行时间 *rt*, 获取本地飞行时间 *lft*(每秒自动计算当前本地时间与起飞时间之间的差值), 并获取上一帧弹道数据的飞行时间 *rt*0; 然后, 判断 *rt* 和 *rt*0 的大小, 如果 *rt* 小于 *rt*0, 则数据非法, 结束本轮处理, 否则继续判断 *rt* 和 *lft* 的绝对误差, 如果误差超过阈值 *Threshold*, 则数据非法, 结束本轮处理, 否则数据合法, 继续完成弹道数据插值处理并更新数据引擎驱动模型的位置变化, 转入下一轮处理. 整体流程如图 3 所示.

2.5 弹道外推自动控制

在飞行器飞行过程中存在滑行段的过程, 即一段时间没有弹道数据更新, 或者突发的链路故障导致没有弹道数据更新, 但飞行器模型的位置却不能中断更新, 因此需要设计自动启停的弹道外推控制机制, 根据当前速度和位置矢量预测飞行器的位置和速度, 保证飞行器位置连续更新.

软件中设计了弹道计数监视器 DDMonitor 和弹道外推驱动器 PredictDriver, 弹道计数监视器用于监视接收的弹道数据的计数并产生启动和停止信号, 弹道外推驱动器用于实际控制弹道外推的启动和停止, 这两个类都是线程类, 主体函数 Run 函数在各自的线程中执行. 当 DDMonitor 每隔固定时间进行轮询, 如果判定进入滑行段则向 PredictDriver 发送 DDStop 信号, PredictDriver 启动弹道外推, 如果判定退出滑行段则向 PredictDriver 发送 DDRecover 信号, PredictDriver 停止弹道外推.

弹道计数监视器 DDMonitor 的轮询控制流程如图 4 所示, 其中 *tSinceLast* 表示距离上次收到有效弹道后轮询的次数, *isHXD* 表示滑行段标志 (true 为滑行段, false 为非滑行段), *isStart* 表示线程运行标志, *lastCount* 表示上次收到的有效弹道计数值, *maxSpan* 表示判定进入滑行段的最大轮询次数.

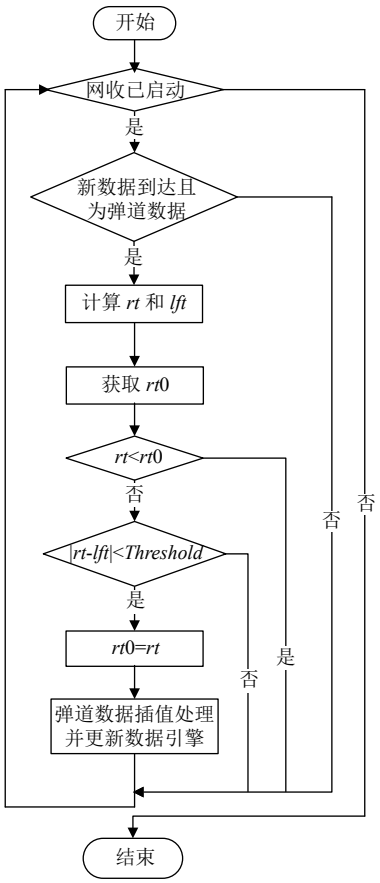

图 3 弹道筛选流程图

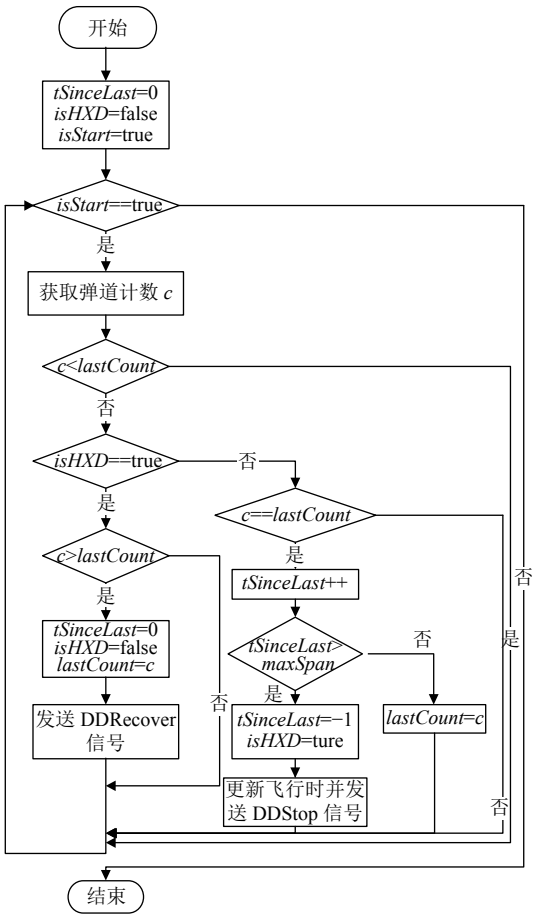

图 4 弹道计数监视器轮询控制流程图

2.6 测站自动跟踪

在飞行器仿真过程中, 需要显示各测站到目标飞行器的跟踪状态. 当目标飞行器相对于测站出地平, 即仰角大于等于 0 度, 测站开始跟踪, 显示跟踪线, 当目标飞行器相对于测站入地平, 即仰角小于 0 度, 测站停止跟踪, 隐藏跟踪线.

基于目标跟踪的原理[16], 通过 OSG 中节点更新回调机制实现测站自动跟踪. 某个测站到目标飞行器的跟踪线通过跟踪线场景节点对象 *lineNode* 定义, *lineNode* 节点注册一个节点回调对象 TTCNodeCallback. 测站自动跟踪的流程是这样的: 当渲染整个三维场景的场景图且遍历至 *lineNode* 节点时, 会自动调用回调函数并传入该测站的地理位置信息和飞行器目标节点对象 *targetNode*, 根据 *targetNode* 获取当前时刻目标的地理位置信息, 从而计算出该测站到目标节点的仰角 *E*. 如果 *E*<0, 则清理 *lineNode* 的显示缓存, 隐藏跟踪线, 否则用测站和目标的位置信息更新 *lineNode* 的顶点信息, 刷新 *lineNode* 的显示缓存, 从而显示跟踪线.

2.7 基于地惯坐标系的地球自转设计

在航天器分离后, 需要根据轨道根数推算航天器飞行轨迹并展示. 基于地惯坐标系的轨迹能形成趋于闭合的大椭圆轨迹, 符合航天器在轨态势显示的需要.

OSGEARTH 中的世界坐标系是按照地固坐标系定义的, 原点在地球中心, *x* 轴向右 (指向本初子午面), *y* 轴指向屏幕里, *z* 轴 (地轴) 向上. 在该坐标系下地球是静止的, 而地惯坐标系下地球是自转的. 因此需要实现坐标系的转换和地球的自转.

首先, 进行坐标系转换. 采用瞬时真赤道地心系 (一种地惯坐标系) 定义场景的坐标系统, 而地固坐标系与它的差别即地球自转角——格林尼治恒星时 S0[17]. 因此, 在系统初始化时根据当时时刻计算 S0, 让地球绕 *z* 轴旋转 S0 角度, 完成地球初始位置的设定.

然后, 实现地球的自转. 在场景树中, 地球是其中的一级节点, 利用 OSG 中节点更新回调机制控制地球节点进行连续角度的旋转变化, 从而实现地球的自转

效果. 地球节点注册一个节点回调对象 EarthRotate Callback, 当渲染整个三维场景的场景图且遍历至地球节点时, 会自动调用回调函数并传入地球节点对象, 回调函数的流程如图 5 所示, 其中 $t0$ 表示当前时刻值, *lastT* 表示上一次回调时刻值, *dt* 表示本次回调与上次回调的时间间隔, Span 表示更新地球旋转的时间间隔 (例如, 按照 1 秒 24 帧的刷新频率, Span 就设置为 1/24 秒), EarthRotSpeed 表示地球自转角速度常量 (7.292e–5 rad/s), *deltaZaw* 表示 *dt* 时间间隔对应的地球自转角度.

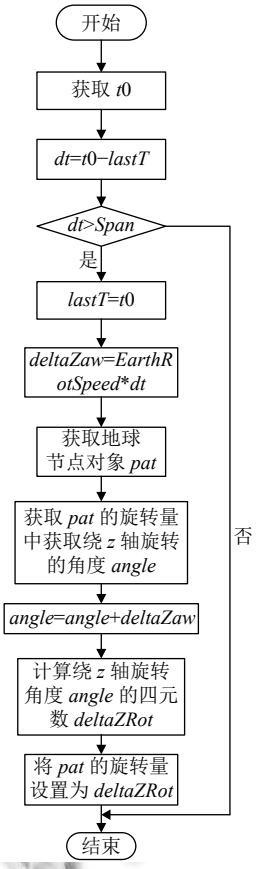

图 5 地球自转回调函数流程图

2.8 多屏输出与切换控制

实际应用中, 三维仿真窗口常需投影到外部设备 (如超长超宽屏幕) 进行全屏演示, 这种需求往往通过分别部署于两个平台的控制端和显示端软件进行通信来实现. 软件利用支持多屏输出显示的显卡和 Qt 中 QDesktopWidget 多屏显示[18]技术实现了同一软件在同一平台上进行多屏扩展显示与切换控制的效果, 降低了软件结构的复杂性, 避免了对网络通信的依赖, 有效提升了在全国产化平台环境下切换控制的可靠性.

以扩展 3 屏显示为例说明. 在多屏显示环境的系统中, 应用程序所在屏幕默认为主屏幕 (screen 0), 其他屏幕都是扩展屏幕 (screen 1, screen 2, …), 主屏幕的左下角屏幕坐标为 (*p.x*, *p.y*). 整个扩展显示区域横跨了 3 个扩展屏幕, 其左下角坐标为 (*p.x*+*p.width*, *p.y*), 宽度 *e*1.*width*+*e*2.*width*+*e*3.*width*, 高度为 *p.height*, 其中 *p*、*e*1、*e*2、*e*3 分别表示主屏幕和 3 个扩展屏幕, *width* 表示屏幕的宽度, *height* 表示屏幕的高度, 如图 6 所示.

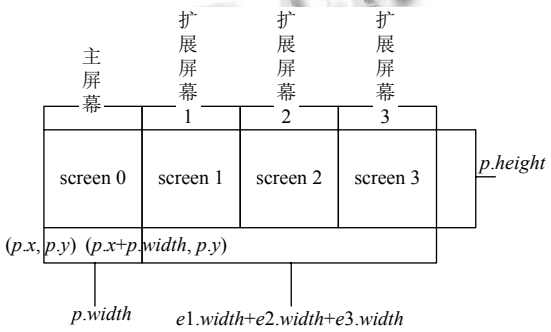

图 6 多屏显示区域的定义

在切换扩展显示时, 首先按照上述定义扩展显示区域对象 extWinGeometry, 设置其宽度和高度; 然后, 将软件界面中的仿真演示窗口部件对象 viewerWidget 设置为扩展窗口 extendWindow 的中央窗口部件, 将 extendWindow 的显示区域设置为 extWinGeometry, 设置 extendWindow 为无边框显示模式; 最后, 更新显示 extendWindow 即可实现扩展 3 屏效果. 这样, 主屏幕显示软件的控制界面, 扩展的 3 屏显示仿真演示窗口.

3 国产平台下的仿真验证

在研发机房部署仿真验证环境, 包括两台工作站, 一台是国产化工作站 (飞腾 1500A、16 GB 内存、1 TB 硬盘、AMD RX570 独显、银河麒麟操作系统 4.0.2), 部署航天三维仿真软件, 一台部署数据仿真重演软件及数据处理软件. 仿真重演软件加载仿真数据发送到数据处理软件经处理后, 模拟任务网络环境向三维仿真软件发送数据, 验证三维仿真的可靠性和渲染效率.

仿真重演软件以 100 帧/s 的帧频发送数据, 三维仿真软件接收、仿真渲染均正常. 飞行器三维仿真全过程中, 仿真环境逼真、飞行器位置和姿态调整流畅, 实时监测结果如图 7 所示, 实时采集的三维渲染性能统计结果如表 1 所示.

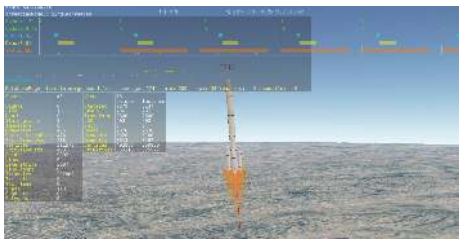

图 7 三维仿真渲染效率实时监测结果

表 1 实时采集的三维渲染性能统计

统计指标	含义	值
帧速率	每秒钟渲染帧数 (fps)	28.78
事件处理时间	每帧事件处理时间 (ms)	1.72
更新时间	对场景树遍历更新时间 (ms)	0.43
拣选时间	对场景树遍历拣选时间 (ms)	1.32
绘制时间	对场景树遍历绘制时间 (ms)	6.81
GPU 渲染时间	显卡 GPU 渲染时间 (ms)	28.65
几何体数	当前渲染的几何体数量 (个)	496
三角形数	当前渲染的三角形数量 (个)	207 862
顶点数	当前渲染的顶点数量 (个)	311 578

根据实时渲染性能统计, 在绘制几何体和三角形数量较大的情况下, 平均渲染帧率不低于 24 fps, 能够较好满足三维仿真的需求.

测站跟踪、模型动画、地惯坐标系下轨迹效果如图 8 至 10 所示.

图 8 测站跟踪

图 9 火箭助推器分离动画

4 结论

本文基于 OSGEARTH 设计和实现了一款国产航天三维仿真软件, 通过地球影像和高程数据的差异化加载、场景树的优化构建、弹道数据筛选等设计策略, 解决了在全国产化平台上进行三维仿真的可靠性和渲染效率问题, 并通过仿真实验进行了验证.

图 10 空间目标在地惯坐标系下的轨迹

下一步的研究方向:

(1) 进一步研究大规模地形组织和渲染效率的优化问题, 使得软件加载更大范围、更高精度的地形成为可能;

(2) 在现有软件基础上, 进一步实现可视化任务完成情况快速评估功能, 并拓展至空间态势感知领域.

参考文献

1 张云燕, 张科, 李言俊, 等. 基于 STK 的月球任务设计与仿真. 火力与指挥控制, 2008, 33(11): 13–16. [doi: 10.3969/j.issn.1002-0640.2008.11.004]

2 范纪松, 任辉, 史红艳. STK 三维战场态势研究与实现. 火力与指挥控制, 2017, 42(6): 132–135. [doi: 10.3969/j.issn.1002-0640.2017.06.030]

3 吴晓雪, 王魏, 李响, 等. 基于 osgEarth 虚拟校园漫游的研究. 现代电子技术, 2017, 40(20): 18–21.

4 王雷, 丁华. 基于 OSGEarth 的大型三维空战场景的搭建. 软件, 2016, 37(1): 114–116, 131. [doi: 10.3969/j.issn.1003-6970.2016.01.025]

5 于艳超, 许捍卫, 吴小东. 基于 OSGEarth 的城市三维地物模型组织与调度研究. 测绘与空间地理信息, 2014, 37(11): 63–67. [doi: 10.3969/j.issn.1672-5867.2014.11.018]

6 韩哲, 刘玉明, 管文艳, 等. osgEarth 在三维 GIS 开发中的研究与应用. 现代防御技术, 2017, 45(2): 14–21. [doi: 10.3969/j.issn.1009-086x.2017.02.003]

7 陈波, 任清华, 杨化斌. 基于 OSGEARTH 的三维数字地球平台设计与实现. 电子科技, 2015, 28(10): 65–68. [doi: 10.3969/j.issn.1009-6108.2015.10.034]

8 吴小东, 许捍卫. 基于 OSGEarth 的城市三维场景构建. 地理空间信息, 2013, 11(2): 107–110. [doi: 10.11709/j.issn.1672-4623.2013.02.034]

9 余航. 基于 OSG 的战场态势三维可视化关键技术研究[硕

士学位论文]. 西安: 西安电子科技大学, 2017.
10 赵小丹. 基于 OSG 的航天任务三维仿真系统研究与实现[硕士学位论文]. 重庆: 重庆大学, 2015.
11 连云霞. 基于 OSG 的运载器与航天器飞行仿真系统研究[硕士学位论文]. 太原: 中北大学, 2018.
12 闫大洲. 基于 osgEarth 的卫星轨道外推数据可视化仿真系统研究[硕士学位论文]. 太原: 中北大学, 2018.
13 杨亮. 空间任务视景仿真系统的设计与实现[硕士学位论文]. 北京: 中国科学院研究生院 (空间科学与应用研究中心), 2007.
14 肖鹏, 刘更代, 徐明亮. OpenSceneGraph 三维渲染引擎编程指南. 北京: 清华大学出版社, 2010. 41–42.
15 Wang R, Qian XL. OpenSceneGraph 3.0: Beginner's Guide. Birmingham: Packt Publishing Ltd, 2010. 93–94.
16 简仕龙. 航天测量船海上测控技术概论. 北京: 国防工业出版社, 2009. 254–255.
17 刘林. 航天器轨道理论. 北京: 国防工业出版社, 2000. 17–21.
18 金大臣尔. Qt5 开发实战. 张红艳, 译. 北京: 人民邮电出版社, 2015. 77–79.