

# 基于三维地形的虚拟仪表仿真技术研究是实现

陈雷, 赵刚, 李荣江

(四川大学电子信息学院, 成都 610064)

**摘要:** 虚拟仪表仿真技术研究一直是航空仿真领域中热门研究课题。文章研究了某型飞机虚拟仪表在 Creator/Vega 开发环境下利用 DOF 技术的快捷实现方法, 讨论仪表的数据驱动技术, 应用多线程编程技术实现仪表的实时仿真功能。仿真结果表明该系统具有开销小、延迟低特点, 平均帧率达到 30 fps 以上, 满足实时交互需求。

**关键词:** 虚拟仪表; 视景仿真; 自由度

**中图分类号:** TP391.9

**文献标识码:** A

## 引言

虚拟仪表技术是指利用计算机图形生成技术和人机交互技术, 在通信和数据分析的基础上, 仿照传统真实仪器仪表的信息采集、加工、处理和显示的技术<sup>[1]</sup>。特别是在进入 20 世纪 90 年代以来, 随着计算机软硬件的发展以及复杂系统仿真应用需求的提高, 仿真飞行中的虚拟仪表动态可视化技术已成为当前的研究热点。

虚拟仪表建模无论是使用 OpenGL 底层语言生成<sup>[2]</sup>, 还是使用诸如 3Dmax SolidWorks 建模工具, 或是用脚本语言描述逻辑, 均存在工作量大、效率低、难以维护等问题<sup>[3]</sup>。目前通常使用的是 GL Studio 等专业仪表仿真平台, 虽然通过专用平台可以创建三维的、照片级的互动图形界面, 但是作为专用的仪表建模平台, 不可避免的存在价格昂贵等问题, 而且仪表模型在移植到驱动平台时需单独设计数据接口来实现仪表驱动仿真<sup>[4]</sup>。基于上述问题, 本文利用 Creator/Vega 开发环境<sup>[5]</sup>, 研究实现了快捷的仪表建模、实时驱动的解决方案。

## 1 基于三维地形漫游的虚拟仪表系统

整个软件系统分为编控建模系统和播出仿真系统两个独立的子系统。系统组成如图 1 所示。编控建模系统的主要工作是设计建立满足需要的虚拟仪表模型和系统模型; 播出仿真系统实现实时仿真功能, 建立和

编控建模系统之间的连接关系, 通过数据接口不断接收外部输入的仪表参数, 并对参数进行解释, 不断刷新虚拟仪表的显示。

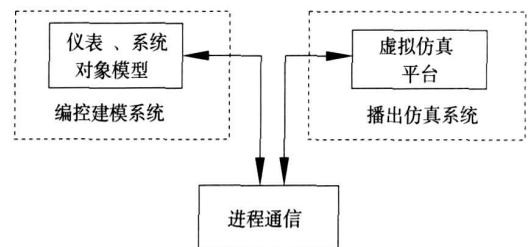


图 1 系统组成框图

仿真系统在计算机屏幕上实时绘制三维地形, 通过数据接口接收需要处理显示的仪表数据, 实时刷新显示数据。该系统具有三个主要特点: 一是具有方便的交互性, 可以通过鼠标控制飞行器来改变各项参数; 二是实现了数据传输和绘制的实时性, 可以在不同的飞行状态下实时更新虚拟仪表的显示, 达到沉浸感仿真的目的; 三是开发快捷、简便, 代码量小, 不需设计模型和驱动间的数据和坐标转换接口, 使用内部消息函数实现数据驱动<sup>[8]</sup>。系统开发流程如图 2。



图 2 系统开发流程

### 2 基于树形结构的仪表对象模型

虚拟平显仪表由多个单元组成, 不同的单元提供不同的信息, 一起协同反馈人能辨识的信息。面向对象进行建模, 系统中仪表单元称为仪表元素, 仪表元素可以包含子仪表元素或者作为父元素的子元素, 从而形成倒置的树形结构, 在 Creator中编辑界面如图 3示。

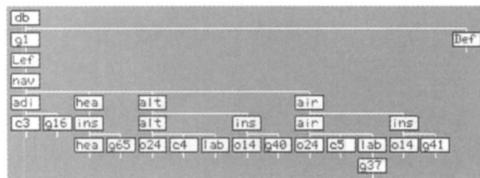


图 3 树形结构的仪表对象模型设计

虚拟仪表的动态更新显示: 旋转、滑动、字体显示等, 这些是仪表能够在实时仿真中更新显示的关键。本文中仪表的可视化更新显示是通过自由度 (Degrees of freedom) 技术实现的。DOF 技术可以使模型对象具有活动的能力, 控制仪表指针按照设置的自由度范围内进行移动或者旋转。本文实现了虚拟仪表的 6-DOF 实时动态显示: 中央的地平仪 (用于显示飞行器的俯仰自由度、偏转自由度), 左侧为飞行速度显示自由度, 右侧为高度显示自由度, 顶部是航向显示。在该模型 (图 4) 中引入的动态元素, 可通过在 Vega环境中调用相关的动态元素的驱动函数接口实现座舱仪表的动态变化与实时仿真。

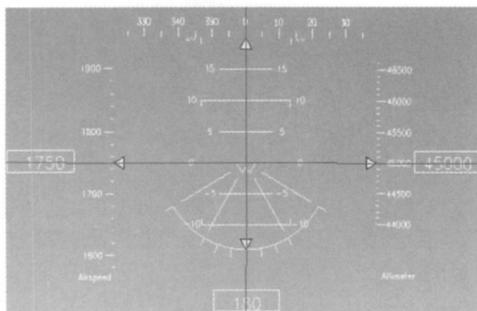


图 4 虚拟平显模型

### 3 虚拟仪表的数据驱动

仪表的外观和仪表类型设计好后, 需要绑定虚拟仪表和仪表的驱动数据, 从而建立仪表和控制该仪表的信号数据之间的联系。当数据信号被仪表接收到时, 仪表可以根据预先定义的参数转换方法, 计算出显示参数并进行数据的刷新显示。

仪表控制系统是虚拟飞机控制系统的显示核心, 在设计虚拟仪表控制系统时, 首先通过 Vega中的仪表控制面板将已经在 Creator中建好的仪表模型加载进去, 并在仪表控制面板中进行适当的参数设置, 然后在 VC

+ + 6 0中编程绑定仪表的驱动数据, 通过程序控制模型中的仪表 DOF 节点实现对仪表的交互驱动。

#### 3.1 多线程编程技术

为了提高系统软件运行速度, 采用了多线程编程技术。多线程是指操作系统支持一个进程多个线程的能力。在实际的软件编写中, 运用多线程技术, 整个仿真应用中含有完成不同功能的多个线程, 如数据采集、数据处理、实时数据显示和用户界面线程。这样, 多个线程同时执行, 在一个时间内完成更多的任务, 加快了系统的反映速度, 提高执行效率。本仿真系统通过 Vega提供的主线程函数在线程池中分别创建了三个线程: 主线程、数据采集分析线程、图形刷新线程。主线程完成数据初始化和控制、数据采集分析线程完成模型和驱动间的数据接收和存储、图形刷新线程完成虚拟仪表的动态显示, 通过使用该技术, 大大提高了系统的反应速度, 满足仿真实时性的要求。

#### 3.2 仪表控制系统核心部分代码

Vega仪表控制模块的初始化在 Vega系统初始化之后进行, 首先调用函数 vgIn itSym 初始化 Vega仪表控制模块, 然后调用函 vgConfigSys配置 Vega仪表控制模块, 这时候, 仪表对象就被加载到内存中, 通过调用 Vega API接口函数可以对仪表的动态显示进行实时的控制。

为了真实模拟场景中飞行器的六自由度 (俯仰、横滚、侧滑), 结合在建模中设计的六个 DOF 节点 (即六个自由度节点), 每一个节点控制一个自由度动作。在 VC6. 0+ + 中通过编写代码来控制这些 DOF 节点, 并实时反馈到显示界面当中, 部分关键代码如下:

```

.....
/* 在视景数据库的仪表模型中查找各个可动节点
* /
pitchDOF= ( vgSymDOF * ) vgF indSymNode( hud
" dof pitch", NULL );
rollDOF= ( vgSymDOF * ) vgF indSymNode( hud "
dof roll", NULL );
speedDOF= ( vgSymDOF * ) vgF indSymNode( hud
" dof airspeed", NULL ); .....
/* 在视景数据库的仪表模型中查找需动态更新的
文本 * /
headingText= ( vgSymStr * ) vgF indSymNode( hud
" t40", NULL );
altText= ( vgSymStr * ) vgF indSymNode( hud " alt r
titude inset", NULL );
speedText= ( vgSymStr * ) vgF indSymNode( hud "
airspeed inset", NULL );
/* 限制翻滚的最大和最小值 * /
vgSymDOEM in( ro llDOF, VGSYMDOF_HL - 180 );

```

```

vgSymDOFM ax( rollDOF, VGSYMDOF_H, + 180 );
/* 获得飞行器的空间坐标 */
vgGePos( plane, pos);
vgGePosVec( pos & x & y & z & heading & pitch
& roll);
/* 更新俯仰标尺显示 */
vgSymDOFCur( pitchDOF, VGSYMDOF_Y, pitch*
0.1 );
/* 更新翻滚标尺显示 */
vgSymDOFCur( rollDOF, VGSYMDOF_H, roll);
/* 更新速度标尺显示 */
vgProp( plane, VGPOS_CALC_VEL_MAG, VG_
TRUE);
speed= vgGePosCalc( plane, VGPOS_VEL_MAG);
vgSymDOFCur( speedDOF, VGSYMDOF_Y, 17.5 -
speed* 0.01 );
/* 更新高度标尺显示 */
vgSymDOFCur( altDOF, VGSYMDOF_Y, 45- ( z/
1000 ) );
/* 更新速度文本显示 */
sprintf( tmpstring "% d", ( int) speed);
vgSymStrStr( speedText tmpstring );
/* 更新高度文本显示 */
sprintf( tmpstring "% d", ( int) ( z);
vgSymStrStr( altText tmpstring );
/* 更新朝向标尺显示 */
sprintf( tmpstring "% d", ( int) ( roll);
vgSymStrStr( headingText tmpstring );

```

系统运行截图如图 5所示。

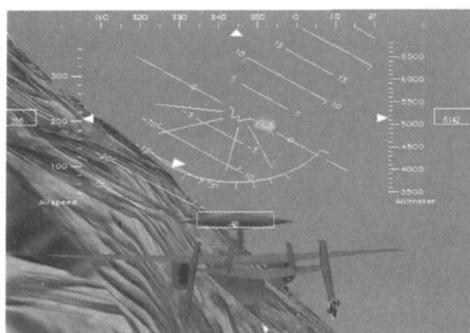


图 5 系统运行图

#### 4 实验仿真数据分析

本文采用的测试平台为: 处理器 Intel(R) Celeron (R) CPU, 1.6Ghz 1G内存; 显卡 Geforce 7300. 为测试仪表模型对系统帧率的影响, 测试分为两个阶段, 分别为不载入仪表模型、载入仪表模型后在不同 LOD(细节层次)情况下的帧率记录, 如表 1。

表 1 不同 LOD下刷新帧率

测试项目	未载入仪表	载入仪表
LOD1(4500- 6000米)	42.0 fps	42.0 fps
LOD2(2000- 4500米)	39.0 fps	37.5 fps
LOD3(500- 2000米)	33 fps	30.2 fps

通过系统联调结果表明: 当飞行器在 4500米飞行时, 由于用于显示三维实景地形的三角面片少, 系统资源富余, 仪表模型载入、仿真对系统的开销几乎没有影响; 当飞行器降低到 4000米或 2000米以下时, 系统资源用于处理三维地形的面片消耗较大, 但是仪表模型的载入、仿真对系统开销的影响较小, 系统延迟在 1- 2 帧 秒左右, 且平均帧率达到实时仿真要求的 30fps 以上, 满足交互仿真的要求。

#### 5 结束语

本文探讨了基于三维地形漫游的虚拟仪表建模和实时驱动问题, 利用 C reator/V ega开发环境实现快捷的动态仪表实时仿真的解决方案。仿真结果表明, 该系统不但满足常用仪表仿真的参数要求, 且对整体系统的资源开销小, 延迟低, 且平均帧率达到 30fps 以上, 使仿真过程直观, 仿真结果容易理解, 并且在仿真过程中可实现实时响应交互操作, 已用在实际项目开发中, 且该系统在写入接口程序后, 能够接收外界传来的数据, 实时显示飞行器各种飞行姿态, 从而更好地实现可控飞行。

#### 参考文献:

- [1] 丁霖, 方卫宁, 田生采, 等. 车载柔性虚拟仪表的显示与生成 [J]. 北京交通大学学报, 2009 33(1): 32-36
- [2] 金晓明, 茅坪. 用 VC++ 与 OpenGL 开发虚拟仪表控件 [J]. 编程语言, 2008 (9): 23-26
- [3] 许颖慧, 杨峰. GL Studio 在仪表仿真开发中的关键技术研究 [J]. 自动化技术与应用, 2008 27(10): 76-79
- [4] 谢勇, 李治庆. GL Studio 在飞机虚拟座舱实现中的应用 [J]. 计算机时代, 2007 (3): 43-44
- [5] MultiGen-Paradigm, Inc Vega Lynx User's Guide Version 3.7 for Windows NT and Windows 98 [Z]. MultiGen-Paradigm, Inc
- [6] George A Geri, Marc D Winterbottom. Effect of display resolution and antialiasing on the discrimination of simulated aircraft orientation [J]. 2005 26 159-169
- [7] Ashim Garg, Adrian Rusu. Area-efficient planar straight-line drawings of outerplanar graphs [J]. Discrete Applied Mathematics 2006
- [8] 刘东鑫, 欧阳中辉, 王东. 虚拟座舱中虚拟仪表的设计与实现 [J]. 计算机与现代化, 2009 (6): 44-47.
- [9] 杨鹏, 赵刚, 张翀. 无缝 LOD 算法在大规模地形绘制中的实现 [J]. 四川理工学院报: 自然科学版, 2008 21 (2): 64-66

(下转第 184 页)

中增加一个短整型的字段即可,数值小的优先级高,数值大的优先级低。工单类型的优先级可以根据实际情况随时调整。

但是,如果业务工单的优先等级是固定不变的话,会出现另外一个问题。那就是,如果某个业务工单的优先级比较低,后面产生的业务工单均比其优先级高,且数量较多的话,那么就可能导致该业务工单很长时间都得不到处理,造成该工单超时甚至成为死单,永远都得不到处理。为了避免这个问题,需要设置动态的优先级,即具体业务工单的优先级是随着时间变化的。业务工单产生之初,其优先级设置为工单类型的优先级,然后在工单等待处理的过程中,每间隔一定时间(比如说半小时)其优先级提高一级。这样一来,虽然某个工单开始的时候优先级较低,但随着时间的推移其优先级会逐步变高,当其优先级升高到一定程度,该工单就会被处理。

### 3 结束语

电子工单系统是电信企业运营信息化的重要 IT 支

撑系统,能有效提高服务质量,缩短障碍处理和业务开通时间,改善企业的服务效率。本文设计的电子工单系统已经在南充电信公司投入使用,实现了业务处理和障碍处理的流程化、自动化以及运维监控部门日常工作的电子化、无纸化,取得了良好的社会和经济效益。

### 参 考 文 献:

- [1] 徐联华,王加阳,汤丹. 基于 MVC 模式的电子工单系统设计与实现[J]. 计算机应用, 2006 25(10): 14-16
- [2] 龙启明,刘斌,程捷. Delphi 7 高级编程范例[M]. 北京:清华大学出版社, 2004
- [3] 江山. 电子工单系统的设计与实现[J]. 福建电脑, 2006 (3): 153-155
- [4] 叶晓彤,何海东. 基于 XML 的网络监控系统指令跨平台和安全传输研究[J]. 四川理工学院学报:自然科学版, 2008 21(5): 39-42
- [5] 范春晓,许慕鸿,刘杰. 一种本地交换网管系统数据处理模型的建立[J]. 北京邮电大学学报, 2002 25(1): 83-86

## Design and Realization of Electronic Work's Form System

PAN Wei

(Computer School of China West Normal University, Nanchong 637009, China)

**Abstract** Electronic work's form system is very important in communication enterprises which can help communication enterprises to improve their serving standard and efficiency. The electronic work's form system is composed of tripartite business treatment, malfunction treatment and day-to-day affairs treatment in the text. The paper introduces the design and realization of business treatment subsystem and that of malfunction. Then the paper thoroughly analyses some important problems about the realization and then gives some solutions.

**Key words** electronic work's form; handling malfunction; handling business instruction; system

(上接第 180 页)

## Research of Virtual Instrument Simulation Technology Based on 3D Terrain

CHEN Lei, ZHAO Gang, LI Rong-jiang

(School of Electronics and Information Engineering, Sichuan University, Chengdu 610064, China)

**Abstract** Virtual instrument dynamic simulation technology has been a hot field of aviation simulation research. This paper studied the quick carried out methods of the virtual instrument in a certain types aircraft based on Creator/Vega, discussed the virtual instrument data-driven technique and applied multi-threading technology to realize the function of instrument dynamic simulation. Simulation results show that the system has the characteristics of small overhead, low latency, and the average frame reached 30 fps or more and meets the needs of real-time interaction.

**Key words** virtual instrument; visual simulation; DOF