Realization of User-Defined Flying Model in the Visual Simulation System Based on Vega Prime

Chao Chen, Liang Qiao, Jingxiong Zhao

Department of Computer Science, North China Institute of Science and Technology, Beijing, China

Email: chaochen@ncist.edu.cn, compbridge@sina.com, zhaojx@cntmi.com

Abstract: A very important feature, in the visual simulation system which is based on Vega Prime and aims at presenting virtual aircraft, is the control stability of virtual aircraft. However, confined by the way to be realized, the vpMotionFly model in Vega Prime can not operate virtual aircraft stably when the refreshing rate is blow 10Hz, and virtual aircraft will shake and go out of control. This paper provides a new myMotionFly model by expanding the Vega Prime abstract moving model—vpMotion. It has been proved that myMotionFly model can control virtual aircraft stably at any refreshing rate and is more suitable for large-scale scene simulation than vpMotionFly model in Vega Prime.

Keywords: visual simulation; Vega Prime; user-defined flying model

基于 Vega Prime 的视景仿真中自定义飞行模型的实现

陈　超，乔　良，赵竞雄

华北科技学院，北京，中国，101601

Email: chaochen@ncist.edu.cn, compbridge@sina.com, zhaojx@cntmi.com

摘　要：在基于 Vega Prime 并以虚拟飞行器为展示主体的视景仿真系统中，虚拟飞行器的可操纵性是一个非常重要的衡量指标。然而由于 Vega Prime 自身实现方式的限制，其自带的 vpMotionFly 飞行模型在绘制刷新率低于 10 Hz 时将无法继续稳定地操纵虚拟飞行器，虚拟飞行器会剧烈抖动直至失去控制。本文通过对 Vega Prime 抽象运动模型 vpMotion 进行扩展，实现了一个自定义飞行模型 myMotionFly，经验证该模型能在各种绘制刷新率下保持对虚拟飞行器的稳定控制，与 Vega Prime 自带的 vpMotionFly 飞行模型相比更适合应用于大规模场景仿真中。

关键词：视景仿真；Vega Prime；自定义飞行模型

1 引言

视景仿真利用虚拟现实技术来实现可交互的沉浸式虚拟环境，用户置于其中能产生身临其境的感觉。利用视景仿真技术能有效地缩短研制周期、节省研制经费，该技术已经在许多领域得到应用，如军事仿真、城市规划、产品设计、数字博物馆等领域。在众多的视景仿真开发平台中，MultiGen-Paradigm 公司的 Vega Prime（以下简称 VP）以跨平台、开发效率高、扩展模块丰富等特性显示出强大的竞争力，基于 VP 开发视景仿真应用已经成为主流趋势。VP 不但基于 Vega Scene Graph（跨平台实时绘制 API）为用户提供封装良好的 C++开发类库，还为用户提供 Creator 和 Lynx Prime GUI 图形化交互编辑工具，将扩展性与易用性结合在一起，从而使用户可以简单迅速地创建、编辑、展示复杂的仿真应用。VP 默认支持鼠标和键盘交互设备，同时基于 VRCO Trackd 可对其它交互设备进行扩展支持。VP 提供了七种运动模型来操纵三维对象或视点，它们分别为 vpMotionDrive（驾驶模型）、vpMotionFly（飞行模型）、vpMotionUFO（飞碟模型）、vpMotionWalk（行走模型）、vpMotionWrap（卷轴模型）、vpMotionGame（游戏模型）和 vpMotionSpin（球旋转模型）。其中，vpMotionFly 飞行模型常被用来交互操作虚拟飞行器，但由于 VP 自身实现方式的限制，当绘制刷新率低于 10 Hz 时，由 vpMotionFly 运动模型控制的虚拟飞行器会剧烈抖动直至无法操纵，在本文所述的视景仿真系统中即出现了这一现象。

978-1-935068-13-6

为解决这一问题，本文通过对 VP 抽象运动模型 vpMotion 的扩展实现了一个稳定而且简单易用的自定义飞行模型 myMotionFly，同时基于 DirectInput API 实现了专业飞行摇杆对虚拟飞行器的控制。

2. 视景仿真系统框架

本文所述视景仿真系统首先通过 Creator 和 Lynx Prime GUI 工具对三维场景进行配置，包括添加模型、设置纹理、设置碰撞检测方式及添加特殊效果，然后在 Visual Studio 2003 平台上基于 Vega Prime 开发类库、OpenGL、Windows SDK、DirectPlay 实现诸如虚拟飞行器的实时交互、飞行任务的解析执行及多用户间的协同操作等功能。系统框架如图 1 所示：

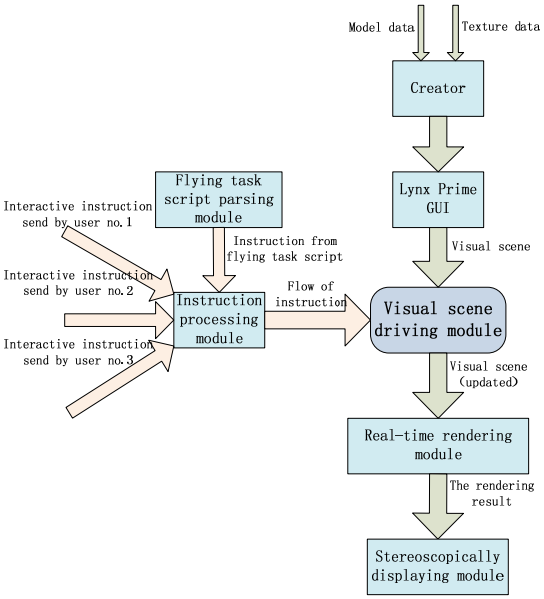

Figure 1. The framework of visual simulation system

图 1. 视景仿真系统框架

模型数据和纹理数据在经过 3dsmax、Maya 等建模软件处理后导入到 Creator 工具中进行编辑和优化，然后由 Lynx Prime GUI 工具搭建三维场景树，指定各三维物体间的层级关系和碰撞检测方式，并设置天空、海洋等环境参数，添加雨、雪、粒子等特效；运行时由飞行任务解析模块解析由 XML 语言描述的自定义飞行任务，将飞行指令和用户交互操作指令处理后合并为指令流传递给视景驱动模块；在视景驱动模块中根据指令流对视景中各对象的形态参数和外观参数进行更新；最后由基于 Vega Scene Graph 及 OpenGL 的实时绘制模块进行渲染，并将最终结果发送给立体显示模块进行展示。

3. 自定义飞行模型的设计及实现

Vega Prime 基于面向对象技术封装了 vpMotion 基类来满足用户开发自定义运动模型的需求，基于 vpMotiom 所开发的自定义运动模型可无缝集成于 Vega Prime 系统中。下面将从 vpMotion 基类的派生，输入数据的采集，缩放向量、旋转向量、位置向量的更新及操纵灵敏度的调整四个方面进行展开讨论。

3.1 派生 vpMotion 基类

运动模型就是一种位置姿态更新策略，它使输入设备的交互操作能对三维物体进行动态定位。Vega Prime 用 vpMotion 定义出抽象的运动模型，vpMotionDrive、vpMotionFly 等运动模型都是从 vpMotion 这个基类派生而来的。

vpMotion 可以接受三种类型的输入数据，分别为 SourceBoolean、SourceFloat 和 SourceInteger，一般由 SourceBoolean 类型的数据表示状态的开或关，如 vpMotionDrive 中的加速或减速，由 SourceFloat 和 SourceInteger 类型的数据来定量地描述运动参量，如 vpMotionDrive 中的行驶速度。派生 vpMotion 基类的一个重要工作就是准确地从交互设备采集输入数据，并按数据特征相应地转化为 SourceBoolean、SourceFloat 和 SourceInteger 类型。

派生 vpMotion 基类的另一个重要工作为实现派生类的 compute 方法，该方法在三维物体的动态定位过程中被自动调用，其原型为：

virtual const vpCoordConverter *compute

(vpPositionable::StateVector *sv, const vpCoord-Converter *conv, double dt)

其中的 vpPositionable::StateVector* sv 参数使用缩放、旋转、位置三个向量来描述三维物体当前的状态，实现 compute 方法的主要任务就是根据交互设备的输入数据更新 sv 参数的当前值。

3.2 采集输入数据

本文所述视景仿真系统除支持鼠标及键盘交互设备对虚拟飞行器进行操纵之外，还支持专业飞行摇杆设备对虚拟飞行器的操纵。系统所使用的飞行摇杆设备及其输入数据如图 2 所示：

为操纵简便直观起见，在系统中由飞行摇杆主杆

绕 x 轴的旋转控制虚拟飞行器的俯仰角，数据类型为 SourceFloat；由飞行摇杆主杆绕 y 轴的旋转控制虚拟飞行器的滚动角，数据类型为 SourceFloat；由飞行摇杆主杆绕 z 轴的旋转控制虚拟飞行器的方位角，数据类型为 SourceFloat；由飞行摇杆副杆的旋转控制虚拟飞行器的飞行速度，数据类型为 SourceFloat；由飞行摇杆主杆的按钮 1 控制虚拟飞行器发动机的开或关，数据类型为 SourceBoolean；由飞行摇杆主杆的按钮 2 控制虚拟飞行器起落架的抬起或放下，数据类型为 SourceBoolean。由于 Vega Prime 平台并没有实现对飞行摇杆设备的支持，因此无法通过直接访问摇杆设备来采集输入数据，本系统以 DirectInput API 为中间层实现飞行摇杆设备输入数据的间接访问，并将访问结果转化为相应的 vpMotion 可接受的数据类型。

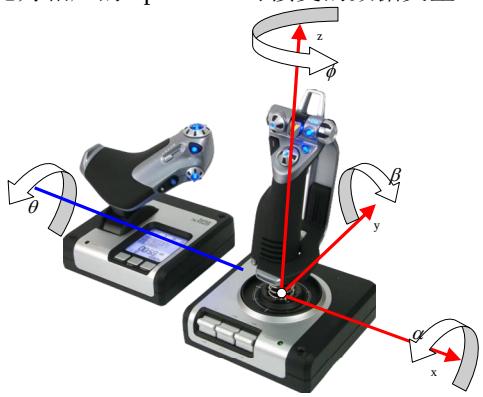

Figure 2. The flight joystick used in this system

图 2. 系统应用的飞行摇杆设备

3.3 更新缩放向量、旋转向量、位置向量

上文提到，派生 vpMotion 基类的主要任务为给 vpPositionable::StateVector* sv 参数赋当前值，即对采集到的输入数据进行处理，更新描述三维物体当前状态的缩放向量、旋转向量及位置向量：

- 在飞行过程中，虚拟飞行器一般不会产生缩放变形，因此缩放向量不用进行更新。
- 旋转向量由飞行摇杆输入数据中的绕 x 轴旋转角 α、绕 y 轴旋转角 β、绕 z 轴旋转角 ϕ 进行更新：

设原旋转仿射变换矩阵为 T_{R0}，绕 x 轴旋转 α 的变换矩阵为 T_{RX}：

$$T_{RX}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha & \sin\alpha & 0\\ 0 & -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

绕 y 轴旋转 β 的变换矩阵为 T_{RY}：

$$T_{RY}=\begin{bmatrix}\cos\beta & 0 & -\sin\beta & 0\\ 0 & 1 & 0 & 0\\ \sin\beta & 0 & \cos\beta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

绕 z 轴旋转 ϕ 的变换矩阵为 T_{RZ}：

$$T_{RZ}=\begin{bmatrix}\cos\phi & \sin\phi & 0 & 0\\ -\sin\phi & \cos\phi & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

则更新后的旋转仿射变换矩阵 T_{R1} 为：

$$T_{R1}=T_{RZ}T_{RY}T_{RX}T_{R0}$$

由 T_{R1} 可解算出更新后的旋转向量 $\vec{R}$。

- 位置向量由虚拟飞行器的上一位置 $\vec{P}_0$，虚拟飞行器头部所指方向 $\vec{m}$、更新间隔时间 dt 及虚拟飞行器当前速度 v 共同决定，其中 v 由飞行摇杆副杆的旋转进行控制，更新后的位置向量 $\vec{P}_1$ 为：

$$\vec{P}_1=\vec{P}_0+dt\cdot v\cdot\vec{m}$$

其中 $\vec{m}$ 可由 T_{R1} 对单位向量进 $[0\quad 1\quad 0\quad 1]$ 行变换得到。

3.4 调整操纵灵敏度

不同品牌规格的飞行摇杆的灵敏度是不同的，用户更换飞行摇杆后常会有操纵过于灵活或过于迟钝的感觉。为解决这一问题，系统给每个 SourceFloat 类型的输入数据都设定了变换比率参数，并将参数记录在 XML 格式的配置文档中，当更换飞行摇杆时，只要切换不同的配置文档，用户就能获取相同的操纵体验。

3.5 实现 myMotionFly 中 compute 方法

自定义飞行模型 myMotionFly 中 compute 方法的关键实现代码如下：

```
vuVec3<double> translate, rotate, scale, tmp;
vuMatrixAffine<double> mat1, mat2, mat3, mat4;
// 取得原旋转仿射变换矩阵 T_R0
mat1.setRotate(sv->m_rot[0], sv->m_rot[1],
```


978-1-935068-13-6 © 2010 SciRes.

```
sv->m_rot[2]);
        // 根据绕 x 轴旋转角 α 进行变换
        tmp[0] = 1.0;
        tmp[1] = 0.0;
        tmp[2] = 0.0;
        mat2.setRotate(fRX, tmp);
        mat1.preMultiply(mat2);
        // 根据绕 y 轴旋转角 β 进行变换
        tmp[0] = 0.0;
        tmp[1] = 1.0;
        tmp[2] = 0.0;
        mat3.setRotate(fRY, tmp);
        mat1.preMultiply(mat3);
        // 根据绕 z 轴旋转角 φ 进行变换
        tmp[0] = 0.0;
        tmp[1] = 0.0;
        tmp[2] = 1.0;
        mat4.setRotate(fRZ, tmp);
        mat1.preMultiply(mat4);
        tmp[0] = 0.0;
        tmp[1] = 1.0;
        tmp[2] = 0.0;
        // 解算更新后的旋转向量 R
        mat1.decompose(&translate, &rotate, &scale);
        sv->m_rot[0] = rotate[0];
        sv->m_rot[1] = rotate[1];
        sv->m_rot[2] = rotate[2];
        // 生成向量 m
        mat1.transformVector(&tmp);
        // 计算更新后的位置向量 P1
        sv->m_pos[0] += tmp[0]* m_dSpeed * dt;
        sv->m_pos[1] += tmp[1]* m_dSpeed * dt;
        sv->m_pos[2] += tmp[2]* m_dSpeed * dt;
```

4 myMotionFly 在大规模场景仿真中的应用

为检验系统中所实现自定义飞行模型 myMotionFly 在低绘制刷新率下的表现，本文设计出一个大规模场景仿真验证实例，当虚拟飞行器朝地面飞行时，当前可见面片数达 200 万以上，平均绘制刷新率为 6 Hz，操作者可在 Vega Prime 自带的 vpMotionFly 运动模型与自定义飞行模型 myMotionFly 间动态切换。实验证明，当操作者使用 vpMotionFly 模型操纵虚拟飞行器时，会发现虚拟飞行器做剧烈抖动直至失去控制，而使用 myMotionFly 模型操作虚拟飞行器时，对虚拟飞行器的控制十分稳定。

5. 结论

本文所实现的自定义飞行模型 myMotionFly 能在低绘制刷新率时保证对虚拟飞行器的稳定控制，可以在大规模场景的仿真应用中替代 Vega Prime 自带的 vpMotionFly 运动模型。同时，由于 myMotionFly 在设计时以使操作直观简便为目标，因此没有引入 vpMotionFly 运动模型中的突进角、衰减率等专业性参数，今后将在增强自定义飞行模型的专业性方向上做进一步探索。

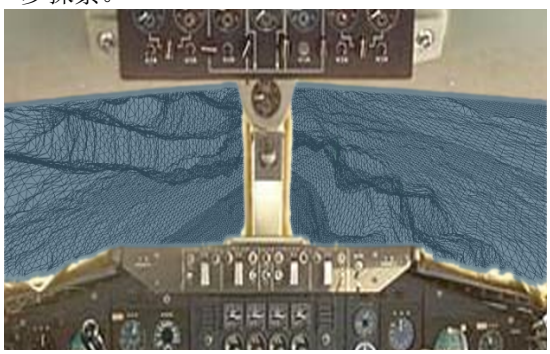

Figure 3. The application of myMotionFly in large-scale scene simulation

图 3. myMotionFly 在大规模场景仿真中的应用

致　谢

本研究受华北科技学院 2010 年度科研基金资助。

References (参考文献)

[1] Chen Gang, Gan ZhiChun, Sheng JianJun, Lu Xu, Equipment simulation training system based on virtual reality [C], Proceedings of the 2008 International Conference on Computer and Electrical Engineering, p 563-567, 2008.

[2] Yongjun Qiao, Xiaoming Bai, Xiaofang Xie, Yongsheng Li, Visual simulation of aircraft carrier assistant landing system[C], Proceedings - 2008 Pacific-Asia Workshop on Computational Intelligence and Industrial Application, v 2, p 714-717, 2008.

[3] Zhang Yue, Wang Jun, Lv Huifang, Jiang Fan, Chen Gang, Research on 3D equipment operation simulation training system [C], Proceedings - 2009 International Conference on Information Management and Engineering, p 564-568, 2009.

Yang Si, Li Xiaomin, Xie Hui, UAV servicing and training system based on VR and HLA[C], 2007 8th International Conference on Electronic Measurement and Instruments, p 2341-2345, 2007.