基于 VRML 的交互式天体运动场景研究①

毛新光　薛安克　马　里　(杭州电子科技大学 计算机学院 浙江 杭州 310018)

摘　要: VRML 是虚拟现实建模语言(Virtual Reality Model Language)的简称，是一种基于文本的描述三维环境的场景描述语言，是 HTML 的 3D(三维)模拟。本文以天体运动中的“日食、月食”为例，讨论了利用组件化技术构建天体运动的三维场景以及 VRML 虚拟场景与外界交互的手段和方法。针对传统的三维制作软件(如 3D Max)在演示过程中不受用户控制和无法实现实时的交互等缺点，重点研究了利用 VRML 节点库中的内插器节点与传感器节点结合和 Script 节点集成高级语言(如 Javascript)来实现交互式天体运动场景的方法及实现过程，通过该平台可以实现人机交互。由于复杂的交互式三维运动场景在计算机上运行速度不够理想，最后提出了采用编组和内联等方案对运动场景进行了优化。

关键词: VRML; 场景; 交互; 虚拟; 插补器

VRML-Based Interactive Scene of the Moving Celestial Objects

MAO Xin-Guang, XUE An-Ke, MA Li

(College of Computer, Hangzhou Dianzi University, Hangzhou 310018, China)

Abstract: VRML is the Virtual Reality Modeling Language(Virtual Reality Model Language) for short, is a text-based description of three-dimensional environment scene description language, HTML, 3D (three dimensional) simulation. To the moving celestial objects, such as solar eclipse and moon eclipse, this paper discusses the means and methods of the use of component-based technology to build three-dimensional movement of celestial objects as well as VRML virtual scenes interact with the outside world. Because the operation of three-dimensional making software (for example 3D Max) in the tradition can't be controlled by user in the demo and interacted in time, this paper studies to achieve the movement of celestial objects interactive method and implementation process can be achieved by the use of VRML interpolator node, sensor nodes, Script node integrated combination of high-level language (such as Javascript)in the library. The human interacts with computer with the platform. Because speed is not ideal about the complex interactive three-dimensional motion, this article concludes the movement scene is optimized by Group and inline programs.

Keywords: VRML; scene; interactive; VR; interpolator

虚拟现实(Virtual Reality，简称 VR)是继多媒体之后另一个在计算机界引起广泛关注的研究热点，其定义可归纳为：利用计算机生成一种虚拟环境，通过多种传感设备使用户“投入”到环境中去，实现用户与该环境直接进行自然交互的技术[1]。

虚拟现实有两种实现方案：软件实现和硬件实现。所谓虚拟现实的软件实现，主要是通过一些基本的硬件环境，利用软件编程的方法在输出设备上输出逼真的场景系统。VRML (Virtual Reality Model Language,简称 VRML)正是这样一种为了完成虚拟现实的软件实现而开发的虚拟现实语言，它是 3D 模拟，使用 VRML 浏览器能读懂的 ASCII 文本格式来描述世

① 基金项目:基金项目(编号);科研项目(编号)
收稿时间:2010-03-06;收到修改稿时间:2010-05-01

界。通过支持VRML的浏览器，就可以显示所建立的三维场景，并能够在三维中进行实时的移动(进行远近、方位等的改变)和实时操作(物体状态的改变)[1]。

本文以天体运动中的“日食、月食”为例进行研究，相对于传统的 3D Max 和 Flash 等制作软件在演示过程中不受用户控制，无法实现实时的交互，使用户在操作过程中缺乏主动性等缺点，本文提出了以 VRML 为平台，通过用 VRML 的三维建模和实时交互功能，结合 Java 语言，运用两种方法开发了可交互的天体运动模型:一种是利用 VRML 的传感器和内插器，另一种是用 VRML 和高级语言融合的方法。该模型在设计上除了仿真天体运动的场景外，更主要的是实现了人机交互，方便的实现了回放。

1 天体运动三维场景的设计

虚拟的天体运动场景的设计以天体运动中的“日食、月食”为例，利用虚拟现实程序设计语言进行软件的设计和开发，使虚拟的天体运动场景与现实的天体运动场景相符合，从而创建出逼真的三维天体运动场景。虚拟的天体运动场景的设计，是利用先进的渐进式软件开发模式对虚拟现实天体运动场景进行需求分析、设计和编码。设计采用的模块化和组件化的设计思想，开发出设计层次清晰、结构合理的虚拟现实天体运动场景[2]。虚拟现实天体运动场景设计的层次结构如图 1 所示:

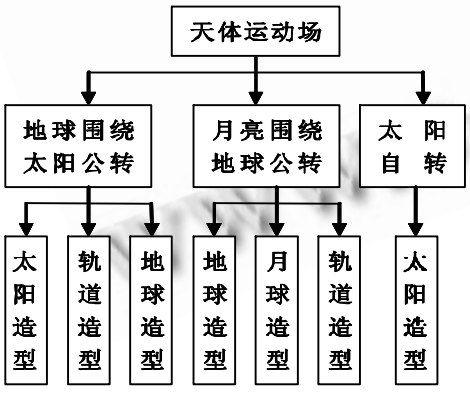

图 1 天体运动场景设计层次结构图

2 VRML产生虚拟动画的原理

传感器是提供交互能力和动态行为的基元，一个具有动态能力的节点需要包含传感器。VRML 共提供了 7 种传感器节点，即 Cylinder Sensor、Plane Sensor、Time Sensor、Visibility Sensor、Proximity Sensor,它们提供了用户与虚拟世界中的物体进行交互的机制。根据时钟或者用户的动作，它们可以产生一个相应的事件，此事件沿着事先设定好的路由传递下去，从而使得虚拟世界对用户做出反应并实现交互。

内插器(Interpolator,或称插补器)是实现动画的重要手段之一。通过一个被启动的时钟节点，不停地向内插器提供时间消息。内插器一旦接受到一个时间消息，就会结合其内的关键时刻列表，从自己的关键值表通过线性插值，得到一个当前时刻的关键值，并经过路由送至 Transform 节点的相应域，从而实现动画。

VRML 的动画效果实质上是依靠一个给定的时间传感器(TimeSensor)和一系列各种各样的插补器节点(如 OrientationInterpolator)实现的关键帧动画[3]。其基本原理是时间传感器给出一个控制动画效果的时钟，这个时钟包含了关键帧动画的开始时间、停止时间、时间间隔和是否循环等动画控制参数。然后通过这个时钟的输出，在虚拟场景中驱动各种插补器节点，VRML 浏览器根据插补器节点设置通过线性插值的方法生成模型的位置数据，并将数据输出至模型节点，使模型运动到一个新位置，从而实现动画过程。[4]

3 交互式场景的复杂动画的实现

为了更好的仿真出天体运动中“日食、月食”的成因，本文研究了使用交互式控制技术。在 VRML 中虚拟世界和用户之间的交互是通过一系列监测器来实现的，通过这些监测器节点，使浏览器感知用户的各种操作，这样用户就可以和 VRML 虚拟世界中的三维对象进行交互。为了体现利用 VRML 在开发交互式场景的优越性，本方案提出了利用 VRML 感应器和 VRML 融合高级语言(如 Javascript、Vrmlscript 或 Java)两种方法来开发交互式三维场景。

3.1 使用传感器和插值器实现交互

利用 VRML 节点库中的节点进行交互式的设计时，在场景中的实体产生动画时，都与事件有关，随着时间的变化而变化，时间的控制可以通过设定节点库中的时间传感器节点(TimeSensor)，并对其它节点发送时间值。在三维场景中，要让实体产生动画效果，按照预定的轨迹进行运动，就要用到 VRML 节点库中的位置插补器节点(PositionInterpolator)、朝向插补器节点(OrientationInterpolator)。PositionIn-

terpolator 节点描述了一系列用于动画的关键帧的位置值，该节点不创建任何造型，适合于对平移进行插值。利用该插值可以控制地球绕太阳公转、月亮绕地球公转时的位置；而 OrientationInterpolator 节点插补器是为场景中的实体在动画中每个指定时刻选择一个旋转的方向，并且在 KeyValue 域中列出这些关键帧的值。通常利用位置插补器和朝向插补器从时间传感器接受 fraction_changed 事件，并且将输出值送到 Transform 的 translation、rotation 域。利用该方法结合触动传感器节点(TouchSensor)、时间传感器节点(TimeSensor)可以使三维场景中的实体实现三维场景中的地球围绕太阳、月亮围绕地球实现公转和太阳实体自转。要实现人机交互，就是利用 VRML 中 TouchSensor，TouchSensor 节点是基于定点输入设备(通常是鼠标)的事件，这些事件表明用户是否在点选某个实体。若用户在指向几何体时按下鼠标键，传感器将要发送一个 touchTime 事件，可以使用这一事件来模拟许多常用的用户接口。在天体运动的三维场景中，将每个球体，即作为一个场景中的一个实体，又作为一个触动的传感器节点，当将鼠标指向这个节点时，触发一个事件，使其仿真天体运动。正是这个方法，使我们可以控制三维场景中的实体模型，产生人机交互的效果，同时也可以利用该方法可以实现循环的回放。这里以月球绕地球公转为例，而地球绕太阳公转、太阳自转与此相似。在实现月球绕地球旋转运动的动画时，需要用到的节点，其编码为：

```
DEF moonorbiter PositionInterpolator
{   key [……]
 #该域值指定了一张浮点时刻关键值列表
 keyValue [……]
 #该域指定了关键位置的列表
}
DEF    time1   TimeSensor
{
   cycleInterval 10
    #指定传感器从 0 到 1 时刻之间的周期间隔
   loop   TRUE
    #该域的域值指定时间传感器是否循环输出
   startTime 1
    #该域制定了时间传感器开始输出事件的决定时间
}
DEF   touch1 TouchSensor {
   #定义触发的传感器节点
  cycleInterval 10 }
```

为了使 VRML 产生真正的动画，则需要设定路由和事件。事件则决定了节点接受外界信息和向外界发送信息的能力。节点通过事件入口(eventin)接受事件，通过事件出口(eventout)发送事件。通过路由(route)联系起来的节点形成事件体系，通过事件体系，事件可以蔓延传播，从而引起其他节点的变化。当一个节点接受到一个事件时，该节点将根据其特征，产生不同的效果。事件一旦产生，就按时间顺序由路由向目标节点发送，并被目标节点处理，这种处理可能改变节点状态，产生其它事件，从而为 VRML 场景实体产生动画的效果。在 VRML 中，通过 ROUTE 语句来创建一个信息通道来连通事件出口和事件入口[5]。这里以月球围绕地球公转为例，为了实现人机交互，这里把月球设计成即作为三维场景中的实体，又作为人机交互时的触发传感器按钮。当人触发月球实体按钮，使月球绕地球旋转的动画事件传递流程图如图 2 所示，而地球围绕太阳公转、太阳自转的动画事件与此类似。而三维场景中实体运动的动画事件，其主要的编码为：

```
ROUTE touch1.touchTime TO time1.startTime
ROUTE   time1.fraction_changed   TO   moonor
biter.set_fraction
ROUTE   moonorbiter.value_changed   TO moon.
translation
ROUTE touch2.touchTime TO time2.startTime
ROUTE   time2.fraction_changed   TO   earthor
biter.set_fraction
ROUTE   earthorbiter.value_changed   TO earth.
translation
ROUTE touch3.touchTime TO time3.startTime
ROUTE   time3.fraction_changed   TO   star.set_
fraction
ROUTE star.value_changed TO sun.rotation
```

通过人机交互点击触发传感器节点引起时间传感器时间的变化，moonorbiter 节点、earthorbiter 节点、star 节点 的 set_fraction 事件接受到值时，相应的值就会发生变化，moonorbiter 节点、earthor

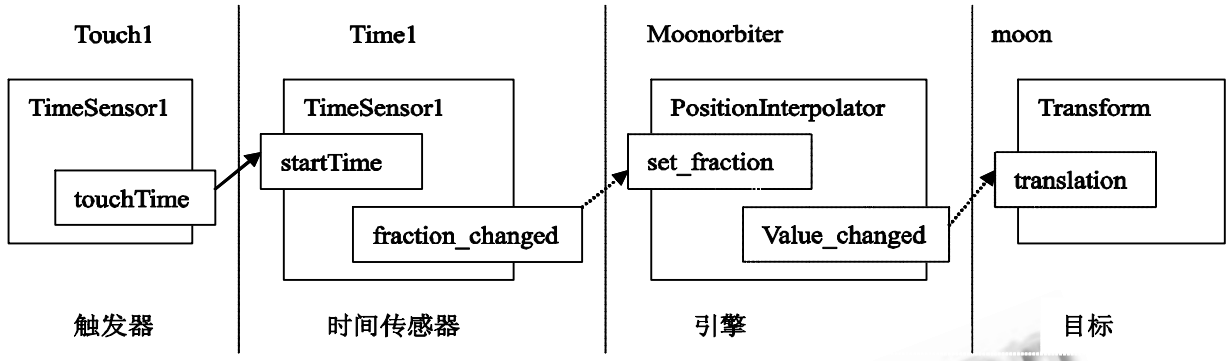

图 2 动画事件的传递流程图

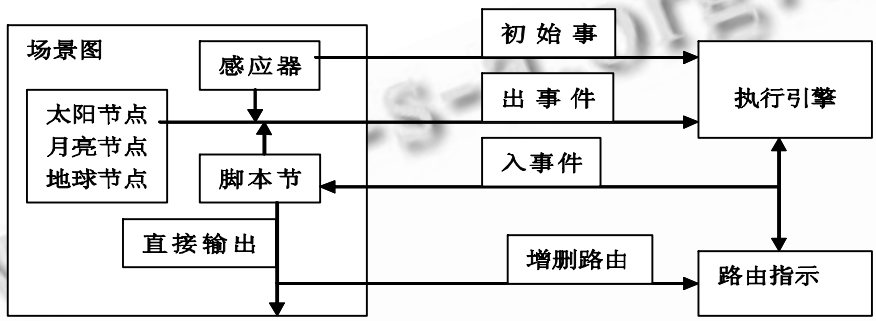

图 3 动画事件的传递流程图

biter 节点新的值将用相应事件 value_changed 输出，引起地球、月亮的位置发生变化，使地球绕太阳公转、月球绕地球公转；而 star 节点新的值将用相应的事件 value_changed 输出，使太阳自转，从而产生动画效果。

3.2 VRML 与高级语言融合实现交互

依靠传感器所能感应事件的交互行为毕竟有限，为了支持更多事件的处理，VRML 定义了脚本节点(Script)。Script 节点可以包含一组利用高级描述语言(如 Javascript、Vrmlscript 和 java)编写的事件处理函数。Script 节点接受到事件后执行相应函数，函数能通过常规的路由机制发送事件或直接向 Script 节点指向的节点发送事件，Script 脚本节点提供了一种强大的机制，能完成更为复杂的交互功能。对于上述的天体运动中“日食、月食”三维交互场景，流程图设计如图 3 所示。

VRML 的脚本节点(script node)主要有四部分组成，：第一部分为一定数量的 eventIn,当接收到事件时引发脚本的执行，第二部分为接收到事件时所执行到的函数，第三部分为在脚本执行过程中，各个函数发行一定数量的 eventout。Script 好像一个智能的插值器，通过 java、javascript 编程语言，可以创建包含任何逻辑判断等复杂的函数，大大提高了 VRML 动画的性能[5]。这里以地球绕太阳公转为例，而月球绕地球公转编程代码与此相似。这里利用 Javascript 脚本语言编写函数，实现人机交互，其主要的编码是：

```
DEF earth Transform {
children [
DEF touch TouchSensor {}
#定义触发传感器节点
DEF time1 TimeSensor {
#定义时间传感器节点
cycleInterval 1
loop TRUE
enabled FALSE}
DEF turn1 PositionInterpolator
#定义位置插值器
{key [ ……]
#该域值指定了一张浮点时刻关键值列表
keyValue [ ……]
#该域指定了一个关键时刻位置的列表 }
DEF go Script {
```

```
#定义一个 Script 节点
eventIn  SFBool  begin
#定义输入事件的变量
eventOut  SFBool  output
#定义输出事件的变量
eventIn  SFTime toucht
eventOut  SFTime  turntime1
eventOut  SFTime  turntime2
url  "Javascript:
function  begin(value)
#嵌入的 Javascript 事件处理函数
{ if(value) { output=value;} }
function  toucht(value)
#嵌入的 Javascript 事件处理函数
{ turntime1=value; }"}
ROUTE  touch.isActive  TO  go.begin
ROUTE  go.output  TO  time1.enabled
ROUTE  go.turntime1  TO time1.startTime
ROUTE  time1.fraction_changed  TO  turn1.set_fraction
ROUTE turn1.value_changed TO earth.translation
```

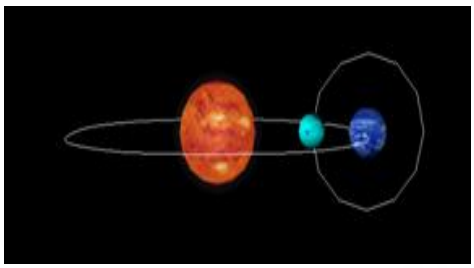

图 4　日食时的场景

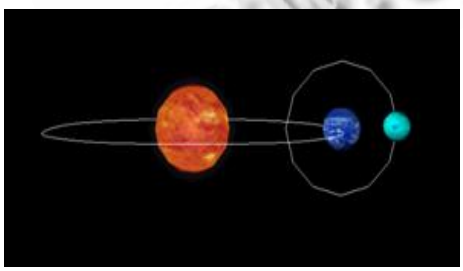

图 5　月食时的场景

通过鼠标单击触发传感器节点，使触发传感器节点处入激活状态，传入输入事件 begin 值，经过 Javascript 脚本函数 begin()的处理，输出经过脚本函数处理后的输出事件的值给 output。经过路由，将输出值传给 time1 TimeSensor 节点的 enabled,使 enabled 值由原来的 False 变成 True,就是在触发之后，使地球始终绕着太阳公转。随着时间的变化，引起 Script 节点中的 turn1 位置插值器中在三维场景中的位置坐标的发生变化，从而使节点的位置发生变化，使场景中实体产生动画效果。

根据以上技术，模拟出的三维动画运动场景如图 4、图 5，当月亮出现在太阳与地球之间的连线上时，月球挡住太阳的光线而形成日食；当太阳、地球、月球三者恰好或几乎在同一条直线上时(地球在太阳和月球之间)，太阳到月球的光线便会部分或完全地被地球掩盖，产生月食。

3.3 模型构建的一些优化技术

对于简单模型，无论是数据点描述还是语义描述，数据传输和再现速度差距不大，但对于具有动画的复杂模型，必须考虑数据容量和重复性设置，由于运行时要占用大量的内存空间，使运行的速度缓慢。采用编组(Group)、内联节点(Inline)的基于语义的模块化设计，充分利用模型和坐标变换(Transform),可以极大优化程序设计，易于修改参数和增删实体。[6]对于天体运动的三维运动场景，通过对场景中的对象、纹理、观察点等进行加工、优化并建立 InLine 节点提高浏览器的性能，加快运行速度。在三维天体运动场景中，多次采用 InLine 节点引用其 VRML 文件来加快运行速度。

3.4 基于 VRML 应用实例的分析

本文是以“月食、日食”为例，创建了基于 VRML 的交互式天体运动场景。如模拟整个太阳系的交互式演示系统，由于其复杂的的三维场景，需要将复杂的三维场景分割成一些独立天体模块，各个模块分别建立每个天体的*.wrl 文件，然后用内联节点可以组成一个大的 VRMl 三维场景[7]，如建立地球造型的 earth.wrl 文件代码：

```
Transform {
rotation
children [
Shape {  #定义地球天体的纹理贴图
appearance  Appearance {
texture ImageTexture{
url  "…… "}}
```

```
geometry    Sphere { # 定义地球的大小
radius ……}}]}
```

在创立太阳系演示系统的 VRML 文件用内联节点中引进 earth.wrl 的文件，其代码为：

```
children [ #引进地球天体到演示系统中，并定义名称
DEF  earth Transform {
children    Inline { url "earth.wrl"}}}]}
```

在太阳系演示系统中，利用内联节点模块化设计，极大的优化了程序设计。由于创建的三维场景数据量之大，运行的速度较慢，可以利用 DEF 和 USE 联合使用。用 DEF 对一个天体进行定义，然后利用 USE 直接引用，避免了代码的重复，加快太阳系演示系统的运行速度，起到优化系统的作用。

4 结束语

由于 VRML 为虚拟场景的建立提供了良好的规范，它具有高效的三维建模、基于事件的交互等功能[8]。因此，本文提出了基于 VRML 感应器和 Script 节点与高级语言融合构建虚拟的交互场景的方案。实际应用表明,在交互式场景中可以利用 VRML 节点库中的各种插补器、传感器、监测器实现交互，同时它还可以与高级语言融合，更能够发挥出 VRML 的强大交互作用。

虽然 VRML 在许多领域得到广泛的应用，但大多数应用是基于三维的渲染、精度要求较低的场景中再现，但在精确设计等方面还无法实现，有待于我们作进一步的研究。

参考文献

1 高钦和,成曙.虚拟现实语言 VRML 与仿真结果可视化.计算机工程与应用, 2001,27(1):97－107.

2 兰翼,杨自栋.基于 VRML 的交互式农田三维虚拟场景的设计与实现.农机化研究, 2009,2(2):77－79.

3 叶琳.VRML 网络 3D 仿真系统中的机构运动仿真.计算机辅助工程, 2002,(2):59－24.

4 吴金来,胡青泥,刘喜平.基于 VRML 的火炮虚拟训练系统的研究.第一届中国图学大会.烟台. 2007:205－207.

5 冯开平,左宗义.VRML 外部程序的研究.工程图学学报, 2003,3(3):49－51.

6 董艇舰,王大勇,李宏伟,汪文津.基于VRML的虚拟模型的构建和交互.机床与液压, 2005(5):148－150.

7 吴小华. VRML 从入门到精通.北京:国防工业出版社, 2002:125－128.

8 张金创,张金锐,张金镝,杨昊诚.编程实训教程.北京:清华大学出版社,北京交通大学出版社, 2008:78－86.