用VRML和Java实现虚拟小区的交互漫游

冯相忠
(浙江海洋学院 计算机系,浙江 舟山 316000)

摘 要:为了实现虚拟小区的交互漫游,在系统实现过程中采用了几种方式的交互技术,文中对这些交互技术进行了讨论和实现。阐述了在虚拟小区中交互漫游的两种方式,即由用户交互控制视点和由计算机自动控制视点的方式,并对这两种交互漫游方式给出了VRML编码的具体实现;最后采用外部编程接口(EAI)实现了用户界面中的二维平面图和三维场景图的交互。

关键词:VRML;虚拟小区;交互漫游;Java
中图分类号:TP391.9 **文献标识码**:A **文章编号**:1673-629X(2008)09-0194-03

Implementation of Interactive Wandering of Virtual Community with VRML and Java

FENG Xiang-zhong
(Department of Computer, Zhejiang Ocean University, Zhoushan,316000,China)

Abstract: In order to implement interactive of virtual community, several methods of the interactive technology were used for implementation of virtual community system. Discusses and implements these methods of interactive technology. Firstly, two methods of interactive wandering were expounded, interactive change viewpoint by user change viewpoint and interactive change viewpoint by automatically change viewpoint, and two methods of interactive wandering were implemented by VRML and Java. Lastly, interaction of 2D graphics and 3D graphics were implemented by external authoring interface of VRML in user interface.

Key words: VRML; virtual community; interactive wandering; Java

0 引 言

将虚拟小区建筑环境构建在Web上,从而使用户可以随时随地通过互联网访问所构建的虚拟建筑。用户可以进入虚拟的建筑环境中,从不同位置、不同方向观察建筑实体,实现实时漫游[1,2]。在整个虚拟小区系统的实现过程中,采用了许多虚拟现实技术,例如,在虚拟场景的构建中,采用了VRML和3DMAX相结合的方式来构建复杂实体,采用LOD技术对场景进行优化等。而文中主要介绍了在虚拟小区系统实现过程中所采用的交互技术,给出了由用户交互控制视点和由计算机自动控制视点方式的两种交互漫游技术,并用VRML实现了在虚拟小区中的漫游;采用VRML提供的外部编程接口程序实现了虚拟小区用户界面中的二维平面图和三维场景图的交互[3~5]。

收稿日期:2007-12-10
基金项目:浙江省教育科研项目(20020951)
作者简介:冯相忠(1962-),男,副教授,CCF会员,主要研究方向为计算机图形学、虚拟现实技术。

1 虚拟小区系统交互漫游的实现

在虚拟小区中漫游是通过不断地改变视点,来从不同的角度观察虚拟建筑物。在虚拟小区的构建过程中采用了两种交互漫游的方式:由用户交互控制视点和由计算机自动控制视点的交互漫游方式。其中由用户交互控制视点的方式是通过用户的行为(如鼠标的位置)来实时地改变视点,实现实时漫游虚拟小区;由计算机自动控制视点的方式可以在预先设计好的路径上漫游虚拟小区,即在虚拟小区中提供一条浏览路径,中间不需要用户的参与,能自动地改变视点,来浏览小区中路径周围的建筑。

1.1 用户交互控制视点方式漫游

用户交互控制视点方式漫游的原理是当用户在场景漫游时,通过ProximitySensor(邻近传感器)来获取用户的位置,并使用这个信息不断地向视点的position(位置)和orientation(方向)传递,从而更新当前视点的位置,达到交互的实时漫游的目的,如图1所示。可以通过如下的VRML代码来实现:

```
#VRML V2.0 utf8
```

```
DEF vp1 Viewpoint{position 0 1.6 56
Orientation 0 0 1 0}
DEF ps1 ProximitySensor{
  size 1024 1024 1024}
……
DEF p1 Script{
url"p1.class"
eventIn SFVec3f currentPosition
eventIn SFRotation currentOrientation
eventOut SFVec3f setNewPosition
eventOut SFRotation setNewOrientation
}
ROUTE ps1.position_changed TO p1.currentPosition
ROUTE ps1.orientation_changed TO p1. currentOrientation
ROUTE p1.setNewPosition to vp1.set_translation
ROUTE p1.setNewOrientation to vp1.set_orientation
```

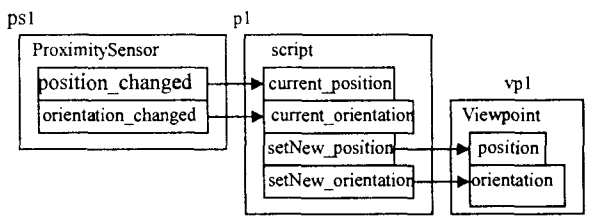

图 1　用户交互控制视点方式

当 ProximitySensor(邻近传感器)感知用户在 size 设置的区域内移动坐标和改变方向时,向 Script 节点 p1 的入事件 currentPosition 和 currentOrientation 传送值,并调用 Java 语言程序 p1.class 进行计算,返回给出事件 setNewPosition 和 setNewOrientation,再通过路由方式改变视点 vp1 的位置和方向。

1.2　计算机自动控制视点方式漫游

自动控制视点方式漫游相当于一组动画的播放过程,在播放动画之前最关键的是如何制作每一帧动画。动画制作的主要任务是采集动画演示过程中的全部预定视点,这些预定的视点需要视点动画制作者在场景中交互漫游,当到达符合要求的观察角度时,把当前的视点记录下来,作为一个预定视点,然后继续寻找下一个合适的视点。在采集视点过程结束后,得到了一组预定视点并把它们存储在视点轨迹表中。采集预定视点时,不必采集用户经过的所有视点,只需采集关键的视点即可。关键视点之间的其它视点可以通过前后关键视点的线性插值计算而得到。

在动画演示过程中,视点轨迹表中的视点值会依序赋予用户的当前视点,使得用户的观察角度和视点动画制作者在采集视点时的观察角度是相同的。在 VRML 中没有视点表这一个节点,可以通过位置节点和方向节点共同组成这一项,这两个节点都是用插值器进行组织的,如图 2 所示。

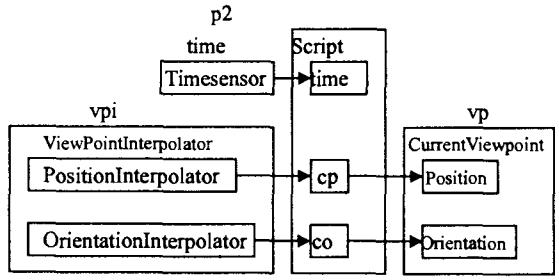

图 2　自动控制视点方式

可以通过如下的 VRML 代码来实现:

```
#VRML V2.0 utf8
DEF vp Viewpoint{
  position 0 0 1
  Orientation 0 0 1 0 }
DEF Time TimeSensor{
    cycleInteral 4.0
    }
DEF pi PositionInterPolator{ key[……]
      keyValue[……]}
DEF oi OrientationInterPolator{key[……]
KeyValue[……] }
……
DEF p2 Script{
url"p2.class"
eventIn SFTime time
field SFNode cp USE pi
field SFNode co USE oi
}
ROUTE time.fraction_changed TO cp.set_fraction
ROUTE time.fraction_changed TO co. set_fraction
ROUTE cp.value_changed TO vp.set_position
ROUTE co.value_changed TO vp.set_orientation
```

可以通过访问 VRML 虚拟场景,采集路径上的所有关键视点值。并把采集到的关键值存放在 ViewpointInterpolator(视点表插值器)vpi 中,它由 PositionInterpolator(位置插值器)和 OrientationInterpolator(方向插值器)组成,如图 2 所示。

在场景中点击按钮,使 TimeSensor(时间感知器)time 被触发,这时执行 Java 程序 p2.class,把存储在视点表插值器 vpi 中的一组视点关键值读出,并赋给事件域 cp 和 co。接着通过路由的方式不断地把事件域 cp 和 co 的值传递给 CurrentViewPoint(当前视点)vp。在当前视点接收到新的视点值后,用户的观察角度将发生变化。通过时间传感器的连续推动,用户的当前视点值也连续地变化,从而实现了自动漫游的目的。

2　用户界面二维平面图和三维场景图交互

在虚拟小区中,为了使访问者快速地被带到所要

漫游的位置,使在虚拟小区中进行有目的地漫游,为此在虚拟小区具体实现中,采用了如图 3 所示界面方式,即用 EAI 实现在同一 HTML 页面中二维平面图的显示(由 Applet 实现)和三维场景图(由 VRML 实现)。当在右面小区的二维平面图中用鼠标点击某个位置时,左边的三维场景图就显示该场景的三维实体,同样在三维场景中漫游时,在二维平面图中用标记来标示出所处位置。

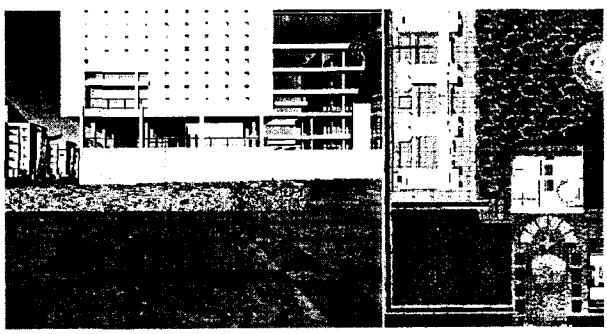

图 3 虚拟小区示意图

实现上述功能的 Applet 程序 p3.java 如下:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import vrml.external.Browser;
import vrml.external.Node;
import vrml.external.field.*;
public class p3 extends Applet implements Runnable
{ private Browser browser; //创建 Browser 类实例
  Node pro1;
  Node pro2;
  EventInSFVec3f newpoint;
  EventOutSFVec3f pos;
  float pp[]=new float[3];
  int ppp[]=new int[3];
  Image image;
  Thread t;
public void init()
{ image=getImage(getDocumentBase(),"map.jpg");
  browser=Browser.getBrowser(this); // 获取对象
  pro1=browser.getNode("p1"); //获得节点 p1
  pro2=browser.getNode("viewpoint1"); //获得节点 viewpoint1
  pos=(EventOutSFVec3f) pro1.getEventOut("position_changed");//获得 eventOut 事件
  }
public void paint(Graphics g)
{ g.drawImage(image,0,0,240,500,this); //将地图 map.jpg 应用到 Applet 中 g.setColor(Color.red);
  pp=pos.getValue();
  ppp[0]=(int)pp[0]+120;
  ppp[2]=(int)pp[2]+250;
  g.fillArc(ppp[0],ppp[2],4,4,0,360); //画点
  }
public boolean mouseDown(Event event,int x,int y)
{float[] val=new float[3];
val[0]=x-120;
val[1]=1.6f;
val[2]=y-250;
newpoint.setValue(val);
return true;
}
public void start()
{
newpoint=(EventInSFVec3f) pro2.getEventIn("set_position");
t=new Thread(this);
t.start();
}
public void run()
{while(true)
  { try{Thread.currentThread().sleep(1000);} //休眠间隔 1000 毫秒
    catch(InterruptedException e){}
    repaint();
  }
}
}
```

在程序中,pro1 = browser.getNode("p1");pro2 = browser.getNode("viewpoint1")分别获取三维场景中的邻近传感器节点 p1 和视点 viewpoint1;pos = (EventOutSFVec3f) pro1.getEventOut ("position_changed")使 EAI 获得了三维场景中的当前位置;paint()方法中的 pos.getValue()获取 x、y、z 坐标值,赋值给数组 pp[0]、pp[1]、pp[2];并转换成二维坐标 ppp[0]和 ppp[2](即平面图中的 x,y 坐标),使用方法 g.fillArc()在坐标为(ppp[0],ppp[2])的位置画一个红色的小圆点;为了使代表观察者的红色圆点跟随三维世界的视点变化而移动,在 Applet 中使用 Thread.currentThread().sleep(1000)语句使 Applet 每隔 1 秒钟就刷新一次,重新获取三维视点所在位置,在平面图上的相应位置上显示更新的红色圆点。

同样,当用户在 Applet 上按下鼠标,就触发了 mouseDown 事件,并且返回一个坐标值 (x,y),将其转化为三维场景图中的坐标,经过一定的处理后,用 newpoint.setValue(val)将值赋给 newpoint;通过 newpoint=(EventInSFVec3f)pro2.getEventIn("set_posi-

(下转第 201 页)

重整中表现出较好的节能特性,但在静态网络中节能效果较差,如图 10 所示(二个节点各为 4 台笔记本的网络,以每十分钟二次整合与退出)。

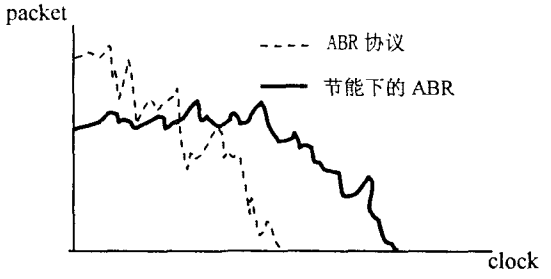

图 9　ABR 协议节能与无节能下网络寿命仿真

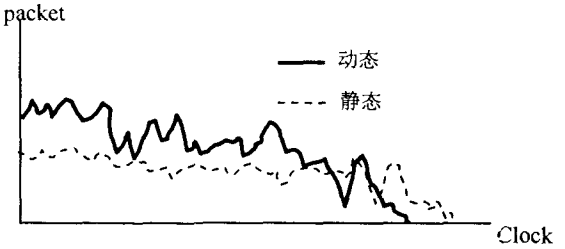

图 10　动态与静态网络 GA 节能寿命仿真

4　结束语

基于遗传算法的抛撒 WSN 寿命预知分析,给出了网络寿命预知的 p_x 分布模型,认为:

(1) 在节点密度由邻居数决定,抛撒的 WSN 的寿命预知模型是与抛撒角度无关,且清晰地建立了 WSN 的寿命预期算法;

(2) 在预知的 p_x 分布模型中,遗传算法比较好地解决了动态传感器网络的能量控制与休眠造成的网络 QoS 下降问题,并延长 WSN 的寿命;

(3) 在高密度的传感器网布局中,休眠系数 β 只有在 $n_{上限} \geqslant p_x \geqslant n_0$ 时,才能使网络寿命最大化,过高的密度与过低的密度,反而降低网络寿命。

参考文献:

[1] Narayanaswamy S, Kawadia V. Power control in Ad - hoc networks: theory, architecture , algorithm and implementation of the COMPOW protocols[C]//Proceedings of the European Wireless Conference - Next Generation Wireless Networks: Technologies, Protocols, Services and Applications. Florence, Italy: [s. n.], 2002: 156 - 162.

[2] van Dam T, Langendoen K. An Adaptive Energy - efficient MAC Protocol for Wireless Sensor Networks[C]//Proceedings of 1st International Conference on Embedded Networked Sensor Systems. New York, NY, USA: ACM Press, 2003: 171 - 180.

[3] Schurgers C, Tsiatsis V, Ganeriwal S, et al. Topology Management for Sensor Networks : Exploiting Latency and Density[C]//Proc 3rd ACM Int'l Symp on Mobile Ad Hoc Networking & Computering. Lausanne, Switzerland: [s. n.], 2002: 135 - 145.

[4] Cerpa A, Estrin D. Ascent: Adaptive self - configuring sensor networks topologies[C]//In: Proc 11th Joint Conf on IEEE Computer and Communications Societies (INFOCOM). New York, NY: [s. n.], 2002.

[5] Li Qun, Aslam J A, Rus D. Distributed Energy, Conserving Routing Protocols[C]//HICSS - Proceedings of the 36th Annual Hawaii International Conference. [s. l.]: [s. n.], 2003.

[6] 王雪飞. 节点密度对自组织传感网络寿命的定量分析[J]. 传感器世界, 2005(11): 29 - 35.

[7] 王雪飞. 自组织传感器网的节点节能与网络节能策略[J]. 计算机应用, 2006(z1): 143 - 146.

[8] Szewczyk R, Ferencz A. Energy implications of network sensor designs[EB/OL]. 2003. http://www. cs. Berkeley. edu/~szewczyk/cs252/paper. pdf.

[9] Hedetniemi S, Liestman A. A survey of gossiping and broadcasting in communication networks[J]. Networks, 1988, 18 (4): 319 - 349.

(上接第 196 页)

tion")将这个三维坐标传给 VRML 中的节点 viewpoint1,将新的值赋给该视点的 position 域,从而达到改变当前视点的效果。

3　结束语

在虚拟场景中交互漫游是实用虚拟现实系统所必需的主要功能之一。由用户交互控制视点的方式主要用于交互地实时地改变视点的漫游;由计算机自动控制视点的方式主要应用于无用户干预的预先设定好的路径上漫游。同样,用 EAI 实现二维平面图和三维场景图的交互漫游也是一个虚拟现实系统的重要组成部分。文中对虚拟小区系统实现中用到的几种交互方式进行了详细的讨论并给出了具体的实现,这对其它虚拟现实系统漫游的实现有一定的参考价值。

参考文献:

[1] 赛博科技工作室. VRML 与 Java 编程技术[M]. 北京:人民邮电出版社, 2002.

[2] 冯相忠, 高　禹, 王　萍. 用 VRML 构建基于 Internet 的虚拟小区建筑环境[J]. 计算机应用, 2005. 25(s): 274 - 275.

[3] 王汝传, 姚旭敏, 王海艳, 等. 基于 java 和 VRML 虚拟场景通讯方式的研究[J]. 系统仿真学报, 2003, 15(7): 986 - 990.

[4] 张茂军. 虚拟现实系统[M]. 北京:科学出版社, 2003.

[5] 李海庆, 殷国富, 胡瑞飞. 基于 X3D 的虚拟场景交互架构的实现方法研究[J]. 系统仿真学报, 2006, 18(s2): 383 - 393.