中华博士园地

这是本刊特为海内外正在就读和学成立业的博士、博士后青年学者们开辟的一片科普园地. 深学浅著是一门德识、慧学、素质修养的学问. 你们的新知识、新调研、新观察、新目光、新展望, 能够用尽可能深入浅出、通俗流畅的语言, 汇报给祖国人民、家乡父老子弟乡亲们吗? 中华博士园地, 乃耕耘忠孝之地, 科教兴国、民族昌盛之地. 要用慈母听得懂的语言, 写出你们的心声!

中图法分类号: TP391.9 文章编号: 1006-8961(2000)12-1044-05

面向 VR 应用系统的 Java 3D API

淮永建 郝重阳
(西北工业大学电子与信息工程研究所, 西安虚拟现实工程技术研究中心, 西安 710072)

0 引 言

Java 3D API 是用来开发三维图形和开发基于 Web 的 3D 应用程序(applet)的编程接口. 目前用于开发三维图形软件的 3D API(OpenGL、Direct3D)都是基于摄像机模型的思想, 即通过调整摄像机的参数来控制场景中的显示对象, 而 Java 3D 则提出了一种新的基于视平台的视模型和输入设备模型的技术实现方案, 即通过改变视平台的位置、方向来浏览整个虚拟场景. 它不仅提供了建造和操作三维几何物体的高层构造函数, 而且利用这些构造函数还可以建造复杂程度各异的虚拟场景, 这些虚拟场景大到宇宙天体, 小到微观粒子. Java 3D 是 JavaMedia APIs 中的一部分, 可广泛地应用于各种平台, 而且用 Java 3D API 开发的应用程序和基于 Web 的 3D 小应用程序(applet), 还可以访问整个 Java 类, 且可以与 Internet 很好地集成, 即如果在浏览器中安装了 Java 3D 的浏览插件, 在网上也可浏览 Java 3D 所创建的虚拟场景.

Java 3D API 还汲取了已有图形 APIs 的优点, 即 Java 3D 的底层图形构造函数不仅综合了底层 APIs(Direcrt3D、OpenGL)最好的绘制思想, 而且它的高层图形绘制还综合了基于场景图的思想, 同时, 它又引入了一些通用的图形环境所未考虑的新概念(如 3D 立体声), 这样将有助于提高用户在虚拟场景的沉浸感. 本文将着重介绍 Java 3D 针对 VR 应用所提出的基于视模型和输入设备模型的新思想, 在此基础上又讨论了如何利用 Java 3D 来开发 VR 应用程序及其实现方法, 并设计实现了一个应用实例.

淮永建 西北工业大学信号与信息处理学科博士研究生. 感兴趣的研究领域为虚拟现实技术及应用、计算机图形学等.

1　适于 VR 应用开发的 Java 3D API

众所周知, 开发 VR 应用程序是一件很繁琐的工作, 其开发人员必须编写应用程序可能遇到的各种输入和显示设备的接口程序, 或者依赖专为 VR 应用开发而设计的应用程序编程接口(API), 且典型的 VR 应用必须跟踪用户的头部位置和方向, 以生成与头部位置方向相一致的虚拟场景图. 另外, 还需要先跟踪身体的其它部位(手、臂或腿部), 然后通过身体各部位在虚拟场景中的虚拟视点与场景中的对象进行交互, 而应用程序也必须具有能够利用跟踪输入设备在视点内放置物体, 并标明其在生成的三维图象中的位置和方向的功能.

同时, 面向 VR 的应用程序开发接口(API) 必须能支持 3D 图形生成、处理跟踪器的输入, 并能将跟踪信息反馈到图形绘制中. Java 3D API 可自动将头部跟踪器的输入集成到图形生成中, 并具有通过访问其它跟踪器信息来控制其它特征的功能, 但它是通过一种新的视模型(view model) 技术来实现的. 该视模型是将用户真实的物质环境与计算机生成的虚拟环境相互独立, 并建立它们之间的通信桥梁. 该 API 也明确定义了用来探测 Java 3D 物体六自由度(6DOF) 传感器的返回值, 并将其应用于显示场景图中. 总之, 这种新的视模型和输入设备模型可以很方便地将交互式的 3D 图形应用程序转化为 VR 应用程序.

2　Java 3D 视模型

2.1　新的视模型概念(view model)

基于摄像机的视模型是模仿虚拟环境中的摄像机, 而不是虚拟环境中人的"替身", 而且它是通过控制摄像机与视点的相关参数来控制所显示的场景, 但这种方法, 在用户物质环境确定某些视参数的系统中是不合理的, 例如在头盔显示器(HMD) 系统中, HMD 的光学性能就直接确定了应用程序所显示的视域. 由于不同的 HMD 有不同的光学特性, 因此如果允许终端用户随意改变光学参数显然是不合理的. 这里视参数的值将随终端用户物质环境的不同而不同, 而影响视参数的主要因素有显示器大小、显示器的位置(戴在头上, 还是放在桌子上)、三维空间中用户的头部位置、头盔显示器的实际显示视域、每英寸的显示象素等. 由于 Java 3D 的视模型直接提供了头部跟踪的功能, 因而使用户产生了真实存在于虚拟环境中的错觉.

Java 3D 不仅提出了新的基于视平台的视模型概念, 同时将其推广到包括显示设备和 6DOF 外围输入设备(如头部跟踪器等) 的接口支持中, 而且新的视模型继承了 Java 的"write once, view everywhere"本质. 这意味着由 Java 3D 视模型开发的应用程序或 applet 可广泛地应用于各种显示环境. 这种显示环境可以是标准的计算机显示屏 、多元显示空间, 也可以是头盔显示器.

Java 3D 视模型是通过将虚拟环境和物质环境完全独立的方式来实现上述功能的, 且该视模型可将虚拟环境中视平台的位置、方向和大小, 与 Java 3D 绘制的与视平台位置、方向相一致的虚拟场景相区分. 一般应用程序控制视平台的位置和方向, 而绘制着色系统则依据终端用户的物质环境以及用户在物质环境中的位置和方向来确定显示场景.

2.2　视模型的组成

Java 3D 视模型由虚拟环境和物质环境两部分组成, 其中, 虚拟环境由 ViewPlatform 对象来表示, 它是虚拟对象存在的空间; 而物质环境则由 View 对象以及和它相关的对象来表示. 这里, View 对象和它的相关对象就描述了用户所处的显示和操纵输入设备环境. 虽然视模型将虚拟环境和物质环境相互独立, 但可通过一一对应关系来建立两种世界之间相互通信的桥梁, 这样将使得终端用户的行为会影响虚拟环境中的对象, 同时虚拟环境中的对象行为也会影响终端用户的视点.

Java 3D 可通过几个对象来定义视模型参数. 这些对象包括 View 对象及其相关对象、PhysicalBody 对象、Canvas3D 对象、PhysicalEnvironment 对象、Screen3D 对象. 视模型相关的对象(如图 1 所示) 其作用如下:

ViewPlatform 用来标志场景图中视点位置的节点. 其父节点则指明了视平台在虚拟环境中的位置、方向和大小.

View 用于指定需要处理场景图的信息.

Canvas3D 定义了 Java 3D 绘制图象的窗口, 它提供了 Canvas3D 在 Screen3D 对象中的大小、形状和位置信息.

Screen3D 用于描述显示屏幕的物理属性.

PhysicalBody 用于封装那些与物质体相关的参

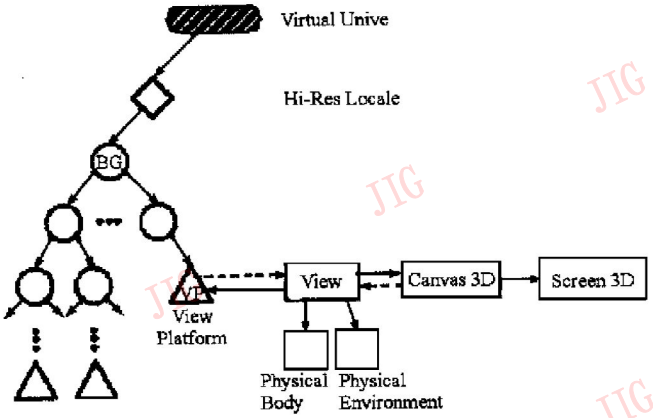

图 1 视模型的组成及其相互关系

数(如左、右眼的位置等).

PhysicalEnvironment 用于封装那些与物质体环境相关的参数(如,用于头状物体或头盔式跟踪器的校验信息).

2.3 虚拟环境中的视平台(ViewPlatform)

鉴于视平台定义了一坐标系统,于是虚拟环境中的原始点和参考点就有了一参考坐标系. 这里视平台代表与视对象相关的一个点,并充当确定绘制图象的基础. 图 2 显示了包括视平台节点场景图的一部分. 由图 2 可见,视平台的父节点确定了视平台在虚拟环境中的位置和方向. 若通过修改与 TransformGroup 节点相关的 Transform3D 对象,就可以在虚拟场景中随意移动视平台. 虽然虚拟环境中可以有许多不同的视平台,但特定的视对象只能与一个视平台相关联,于是在 Canvas3D 对象中所绘制的场景均来自于一个视平台的视点. 这样应用程序就可通过修改视平台的 TransformGroup 节点,在虚拟环境中漫游.

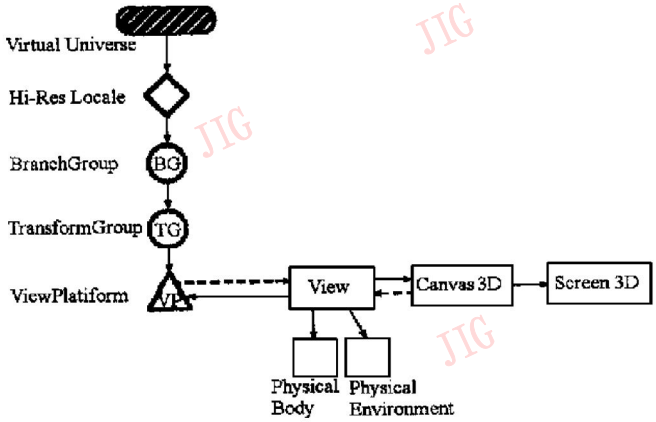

图 2 包括视平台的局部场景图

3 Java 3D 的输入设备模型

Java 3D 除了支持通用的键盘、鼠标输入外,还能给各种不间断的输入设备,如 6DOF 跟踪设备和操纵杆提供支持. 由于不同的跟踪输入设备其工作原理不同,因而计算机与其交互的方式也不同. 为了给不同的 6DOF 输入设备提供支持,Java 3D 还提供了一个输入设备接口,而且该输入设备接口还定义了一个抽象的输入设备,虽然用其可以实现对一特定设备的驱动,但输入设备接口的实现必须实现接口所定义的所有方法(如设备开、关、读取操作、状态设置及查询等). Java 3D 的输入设备列表就用这些方法同特定的设备进行交互. 一般 Java 3D 环境中,可能包括许多输入设备,而且这些输入设备不一定是实际的物理设备,也可能是虚拟设备,例如通过软件的方法将鼠标的运动参数转化为 6DOF 虚拟跟踪球的参数,来模拟虚拟跟踪球的输入.

由于所有的输入设备都由许多传感器对象组成,因此每一种输入设备都与一定数量的传感器对象相关,且每一种传感器都与其传感器设备 6DOF 数据的一个数据源相关. 当输入设备驱动的数据改变时,传感器对象的数据也会相应改变,而且传感器对象由读取传感器对象组成. 由于缓冲区中记录了各传感器最近 N 个读取传感器对象的值,因此可以对传感器数据进行平均,以及对传感器输入值的趋势进行预测等处理,但应用程序的开发并不直接使用输入设备. Java 3D 是通过一个传感器数组将输入设备抽象化,传感器对象数组是物质环境对象的一个子类,该数组是由与输入设备相关的对象指针组成. Java 程序可以直接从传感器数组中获取传感器的值,并将其用到场景图中,或按任意方式对其进行处理.

4 用 Java 3D 开发 VR 应用程序

利用 Java 3D 开发的 VR 应用程序或者 applets 程序,可建造一个虚拟场景,并能将一个或多个场景图插入到虚拟场景中,虚拟场景由超结构对象集组成,对象集则包括一个世界对象(Universe object)、一个或多个场所对象(Locale objects)和按树状结构排列的由节点物体组成的一个或多个场景图(Scene graphs). 该场景图又称为分枝图(Branch graph),它

包括绘制对象节点、光照节点、行为节点和声音节点等, 其中, 包含内容节点的分枝图称为内容分枝, 包含视平台对象的分枝图称为视分枝, 视平台对象用来确定用户的位置和方向. 图 3 表示了具有多分枝图的 Java 3D 场景.

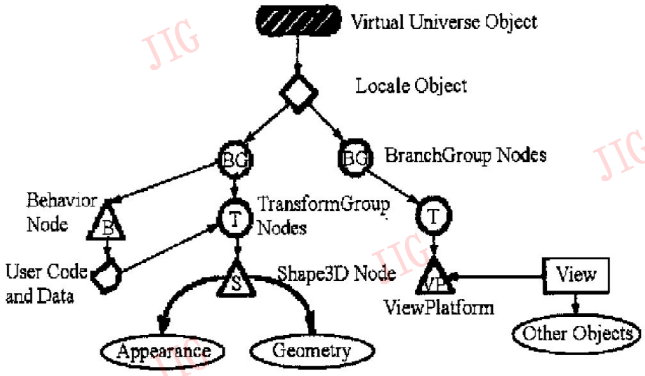

图 3　应用程序场景图

由于这种分枝图只描述了场景所要绘制的对象, 并不确定对象的绘制次序, 因此图中节点的次序和位置与对象的绘制次序无关, 而图中的父节点和子节点的直线路径就唯一确定了子节点的图形范围. 由于绘制次序的不确定性, 因而使得 Java 3D 能横越场景图的任何次序, 且它能从左到右, 从顶部到底部穿过场景图; 或者从右到左, 甚至并行遍历整个场景图. Java 3D 的分支图为树状结构, 且图中的每一个节点只有一个父节点. 这样通过辅助的场景图机制就可以实现通用场景图的共享, 而且具有连接属性的叶节点可以连接到共享子图. 分枝图中的节点分为群节点(group node) 和叶节点(leaf node) 两类, 其中, 群节点按照粘贴的原理来组织场景图单元, 其群节点的层次机构如图 4所示.

SceneGraphObject
- Node
  - Group
    - BranchGroup
    - OrderGroup
      - DecalGroup
    - ShareGroup
    - Switch
    - TransformGroup

图 4　群节点的层次结构图

一般群节点包括: BranchGroup、TransformGroup、Switch、OrderGroup、DecalGroup 和 ShareGroup 节点, 其中, BranchGroup 是分枝图的根节点; 而 TransformGroup 用来指明所有子节点的位置和方向; Switch 则用于实现一个或多个子图的转换; OrderGroup 用于使它的子节点按照特定的次序绘制; DecalGroup 是 OrderGroup 的一个子集; ShareGroup 跟 BranchGroup 一样, 是一个场景图的根节点. 虽然共享图作为 Java 3D 场景图的一部分从不直接出现, 但是连接节点可以引用. 另外群节点还可以包含各种子节点以及所包含对象的群节点或叶节点. 这些子节点用一个关联索引属性来允许对特定的子节点进行操作. 如果没有指明特定的顺序群节点, Java 3D 还可以按照任意指定的顺序来绘制群节点的子节点.

虽然叶节点是场景图的抽象类, 它没有子节点, 但叶节点包括了 Java 3D 的各种信息. 叶节点由 Shape3D、ViewPlatform、Sound、Light 以及用户定义的行为节点等组成, Shape3D 和 ViewPlatform 节点在 Java 3D 的视模型和输入模型中扮演着重要的角色, 因为它描述了图形系统的两个重要方面, 其中, Shape3D 描述了场景中对象的几何形状, 而 ViewPlatform 则标定了用户或其视点在虚拟环境中的方向或位置. 另外, 应用程序还可像操纵分枝图中的任意对象一样, 来操纵 ViewPlatform, 而且应用程序还可平移、旋转和缩放 ViewPlatform, 即通过改变 ViewPlatform 的位置和方向, ViewPlatform 将随同用户的视点一起移动, 来浏览整个虚拟环境. 虽然 ViewPlatform 是按照事先规定的路线浏览场景, 但不会限制用户视点的移动和向不同方向浏览场景. 图 5 显示了叶节点的层次结构图.

SceneGraphObject
- Node
- Leaf
  - Background
  - Behavior
    - Predefined behaviors
  - Boundingleaf
  - Clip
  - Fog
  - Light
  - Link
  - Morph
  - Shape3D
  - Sound
  - Soundscape
  - ViewPlatform

图 5　叶节点的层次结构图

5 应用实例

本文利用 Java 3D 建造了包含一立方体的简单虚拟场景,并利用 Java 3D 提供的输入设备模型接口,设计了一虚拟输入设备控制台,来对虚拟场景中的立方体进行考察和操作. 该虚拟控制台可通过 Java 3D 的传感器对象,来模拟各种 VR 输入跟踪设备,其中在 Java 平台下浏览的虚拟场景及虚拟输入设备控制台如图 6、图 7 所示.

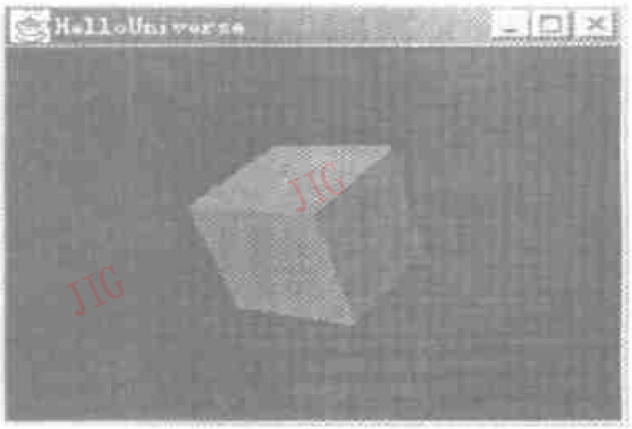

图 6 虚拟场景

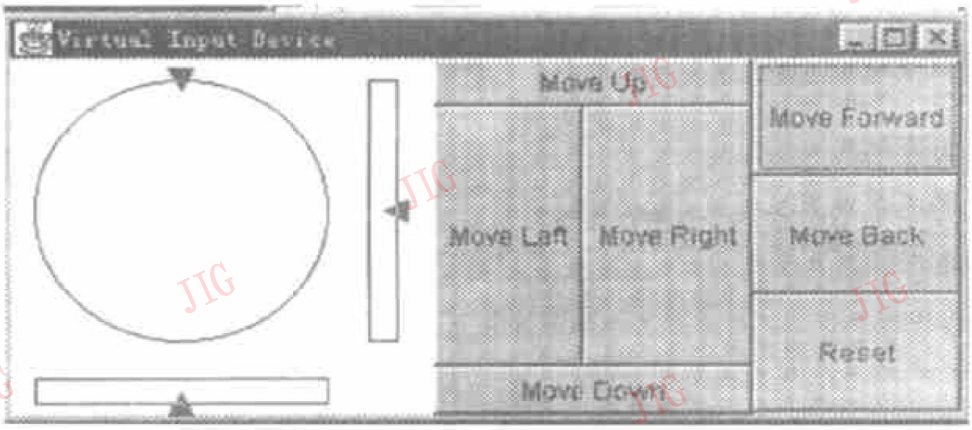

图 7 虚拟输入设备控制台

6 结语与展望

Java 3D API 的出现为 Java 在 VR 和 Internet 的应用和开发注入了新的活力,因为 Java 3D 所提出的全新的基于视平台的视模型和输入设备模型技术,不仅可实现对头盔显示和多输入 6DOF 跟踪设备的支持,而且可针对不同的应用方便灵活地开发自己的 VR 应用系统,由于 Java 3D 是基于 Internet 的软件开发平台,同时它能将图形功能与 Internet 很好地集成在一起,因此,利用它来开发基于网络的 VR 系统将会有巨大的潜力. 可以预见,不久的将来,各种面向网络的 VR 应用(如电子商场、远程教学和远程医疗等)将会把人们带入复杂逼真的虚拟场景中.

参 考 文 献

1 Java 3D Tutorial. (http://java.sun.com/product/java-media/java 3D).

2 Sowizral H A, Rushforth K C, Deering M F. The Java 3D API Specification. Addison Wesley. Peeding. Mass, USA: 1998.

3 The Java 3D White Paper. (http://java.sun.com/product/java3D).