第32卷 第22期 计 算 机 工 程 2006年11月
Vol.32 No.22 Computer Engineering November 2006
·多媒体技术及应用· 文章编号：1000—3428(2006)22—0215—03 文献标识码：A 中图分类号：TP391

基于 XML 的参数化虚拟场景建模技术

陆亚萍，刘厚泉，赵志凯
(中国矿业大学计算机科学与技术学院，徐州 221008)

摘 要：针对 VRML 虚拟场景建模可定制性差和开发效率低的缺陷，提出一种基于 XML 的参数化虚拟场景建模技术，增强了 VRML 场景建模的可定制性，提高了开发效率，并对大场景采用区域划分的方法，提高了大场景的绘制速度，在虚拟城市的构建项目中得到了较好的应用。

关键词：虚拟现实；虚拟现实建模语言；参数化场景建模；XML

Parametric Virtual Scene Modeling Technology Based on XML

LU Yaping, LIU Houquan, ZHAO Zhikai
(School of Computer Science and Technology, China University of Mining Technology, Xuzhou 221008)

【Abstract】 Virtual scene modeling based on VRML has limitation in customizability and developing efficiency. This paper proposes a parametric virtual scene modeling technology based on XML, which improves the customizability and developing efficiency in virtual scene modeling based on VRML. It adopts the method of region partitioning to increase the rendering velocity. The technology is applied well in the project of virtual city.

【Key words】 Virtual reality; Virtual reality modeling language (VRML); Parametric virtual scene modeling; XML

虚拟现实建模语言(VRML)是一种三维模型和渲染的图形描述性语言，凭借其强大的三维功能，已经成为 Internet 上构建交互式虚拟现实场景的标准。

但是，现有的 VRML 场景建模的技术在场景的修改方面不够灵活，可定制性差。其表现为：(1)场景的修改和重复利用方面。开发者用 VRML 设计好一个虚拟场景后，就以文件的形式储存起来，对于很复杂的场景，开发者修改起来困难，如果想根据已建好的场景资源定制一个新的场景也比较困难。比如构建虚拟城市时，需要频繁修改城市的结构和内容，也有可能会利用现有的虚拟城市的场景资源构建另外一座城市。(2)场景的动态构建方面。在分布式虚拟现实中，虚拟场景常常要根据使用者的动作或虚拟环境的改变而动态地添加、替换或删除场景模型的部分内容。结合 VRML 的事件处理机制和 SAI、EAI 接口可以动态地生成场景内容，但是由于场景变化的内容在设计时已经在 Java 程序中固定了，因此无法满足分布式虚拟现实环境的交互性、实时性和异构性的需求。

本文提出了一种基于 XML 的参数化虚拟场景建模技术，它通过 XML 描述需要定制的场景参数，使用 Java 应用程序根据 XML 参数来生成 VRML 格式的场景，并对大场景进行区域划分。当静态或动态修改 XML 文档的内容时，虚拟场景的内容也随之改变。该建模方法主要的优点为：(1)增强了 VRML 场景建模的可定制性，提高了开发效率；(2)使用 XML 文档作为信息交互与场景动态生成的媒介，减少了系统的耦合性，增加了虚拟场景动态生成的灵活性；(3)采用区域划分和动态下载的方法，提高了大场景的绘制速度。

目前该方法在我们开发的虚拟城市的构建项目中得到了较好的应用。

1 参数化场景建模的设计

参数化场景建模就是把 VRML 模型中需要定制的内容作为参数存储在 XML 文档中，比如场景的结构、场景的内容、实体的属性和自动漫游的路径等，由控制中心 Java Applet 读取 XML 参数实现动态场景建模，场景的改变只要通过静态或动态修改 XML 文档来实现。

1.1 采用 XML 参数化的优势

由于 VRML 本身没有参数化的特性，即它的节点的各种属性不能使用变量，在设计时就确定了。为了克服这样的不足，可以利用现有的 VRML 接口的技术将场景的各种需要定制的属性参数化。

参数化可以采用数据库来存储参数，也可以采用 XML 文件存储参数，相比较起来，采取 XML 作为参数的存储格式具有更大的优势：(1)XML 是一种功能强大的标记语言，它具有跨平台性、自描述性和可扩充性等特性，也是 Internet 上分布式应用程序相互交互数据的标准格式；(2)XML 比数据库更轻巧和灵活，访问效率更高，非常适合实时要求高的三维程序。把 XML 以文件形式存放在服务器端可以作为 Java Applet 的参数下载到客户端，直接在客户端执行，速度快，而数据库必须在服务器端，访问速度慢，而且 Java Applet 访问数据库有复杂的安全限制。

1.2 系统的结构设计

系统的结构基于传统的 C/S 结构。设计如图 1 所示。

在服务器端存储了各种文件，接受用户的 HTTP 请求。HTML 文件中嵌入了场景控制 Java Applet 的 Class 文件和场景动态生成前的初始 VRML 文件 index.vrml，XML 参数文件中设置了动态场景建模的参数，它会引用到 VRML 资源文件集中的 VRML 格式的子场景或实体资源，并作为 Java Applet

作者简介：陆亚萍(1980－)，女，硕士生，主研方向：分布式虚拟现实，网络信息集成，代理技术；刘厚泉，副教授；赵志凯，硕士生
收稿日期：2006-02-22 **E-mail：**yapinglu@126.com

的参数随 HTML 文件一起下载到客户端。在系统中，只要修改 XML 文件的内容，就可以灵活地定制虚拟场景。开发者可以直接修改 XML 文件，也可以创建其它的应用程序接口来修改 XML 文件，从而实现灵活定制场景的功能。比如为了使用户可以动态定制场景，可以通过动态网页交互技术，建立一个网页界面供用户来直接修改 XML 的参数设置。

客户机端把控制中心 Java Applet 和 index.vrml 初始展示文件下载到本地执行，XML 参数化文件作为 Java Applet 的参数也下载到客户机端，接着 Java Applet 开始启动，读取 XML 参数，根据参数的设置动态构建虚拟场景。

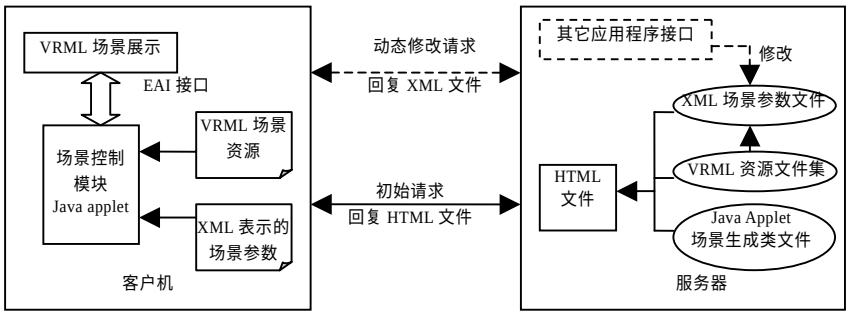

图 1 系统的结构

1.3 建模的步骤

参数化场景建模采用了场景分割、从局部到整体的由下往上的建模思想，这是实现参数化建模的重要基础，也是场景定制和重用的基础。参数化场景建模主要分以下几个步骤：

1.3.1 场景分割

所谓分割技术，就是把整体场景分成若干子场景和实体，子场景可以再分成若干子场景和实体，可以根据实际情况选择分割层次，即整个场景最终是由各种实体组成的，实体是组成虚拟场景最基本的单元，比如一张桌子、一张椅子等在一个房间中就可以看作一个实体，子场景的上一层子场景称为它的父场景，以 3 层为例，分割方法如图 2 所示。

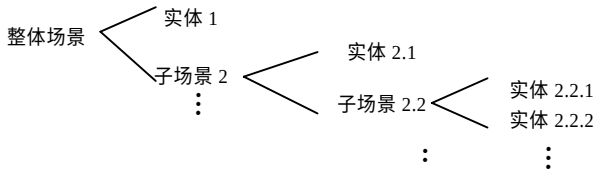

图 2 场景分割示意图

建立一个虚拟城市的街道，场景分割的部分图如图 3 所示。建筑物场景还可以按照需要继续分割。

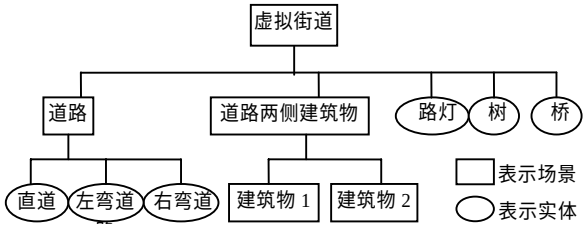

图 3 虚拟城市的场景分割示例

通过场景分割技术，有利于场景模型的模块化，提高场景和实体的可重用性和独立性，子场景之间和实体之间是一种松散耦合的关系，某个子场景或实体的变动不会影响到其它子场景和实体，用户通过改动 XML 参数，可以方便地进行子场景和实体的增加、删除和替换操作，并可以方便地重用实体构造新的虚拟场景。

1.3.2 区域划分

场景分割是从场景的组成角度来划分场景的，但由于三维场景一般都是比较大型的，一次性创建下载会严重影响速度，因此有必要根据用户视野角度对场景进行区域划分。主要思想是当用户在一个大型的虚拟场景中漫游的时候，由于视野的大小限制，用户每个时刻看到的场景是有限的，因此可以将场景分成多个区域，当用户在虚拟场景中漫游的时候，随着用户视点的改变，只将用户周围区域的场景展现出来。这样每次只显示一部分场景，大大减少了场景中多边形和纹理的数量，场景的渲染所需时间减少，浏览速度提高。

比如，虚拟城市中每个建筑物都可以作为一个区域，在区域外只需要看到建筑物的简单外观，内部的复杂构造可以不显示，即不下载到浏览器，只有用户进入建筑物内部才逐层显示；比如，还可以根据道路的延伸来划分区域，把一段道路和道路的建筑物作为一个区域，当进入这个区域时，道路和建筑物等子场景才从服务器下载。

1.3.3 XML 文件设计

场景分割和区域划分完后，就完成了虚拟场景的总体设计，这是 XML 参数化的基础。由于 XML 是可扩展的自描述性标记语言，XML 文件一般由两部分组成，一部分是规定文档约束的 DTD 或 XML Schema 文件，另一部分是根据文档约束文件规定的元素来设计参数内容，即 XML 文件，XML 文件中的所有元素的标记、属性和组成必须遵循文档约束文件中的规定。

以下是构建虚拟城市中 XML 参数化的 DTD 设计，主要有场景(scene)、子场景(sub-scene)、实体(entity)、区域(region)和内容(content)等元素，区域的动态下载采用了基于包围盒 ProximitySensor 的临近检测法，因此在 region 元素中还定义了 ProximitySensor 和 PContent 元素，分别用来定制包围盒和进入包围盒后显示的子场景内容。

```
<!ELEMENT scene(entity*,sub-scene*,region*,content?)>
<!ATTLIST scene
translation CDATA #IMPLIED
rotation CDATA #IMPLIED
fileName CDATA #IMPLIED
defName CDATA #REQUIRED>
<!ELEMENT content(#PCDATA)>
<!ELEMENT entity>
<!ATTLIST entity(content?)
translation CDATA #IMPLIED
rotation CDATA #IMPLIED
fileName CDATA #IMPLIED
defName CDATA #REQUIRED
number CDATA #IMPLIED
method X+|X-|Y+|Y-|Z+|Z- #IMPLIED
value CDATA #IMPLIED>
<!ELEMENT sub-scene (entity*,sub-scene*, region*,content?)>
<!ATTLIST sub-scene
translation CDATA #IMPLIED
rotation CDATA #IMPLIED
fileName CDATA #IMPLIED
defName CDATA #REQUIRED>
<!ELEMENT region(ProximitySensor,PContent)>
```

```
<!ELEMENT ProximitySensor>
<!ATTLIST ProximitySensor
PName CDATA #REQUIRED
PCenter CDATA #REQUIRED
Plwh CDATA #REQUIRED>
<!ELEMENT PContent(#PCDATA)>
<!ATTLIST PContent filename CDATA #IMPLIED
defName CDATA #REQUIRED>
```

1.3.4 实体模型创建

实体是组成场景的基本单位，对于简单的实体可以直接用 VRML 来创建，对于复杂的实体可以通过 3DMAX 等软件来制作，保存为 VRML 格式的文件。创建好总体设计中的所有实体后，就可以通过 XML 参数文件把这些实体组合起来，构建各种复杂的场景。下面是虚拟城市中初始场景的 XML 参数文件，它定义了一个子场景和一个区域，子场景中包含的实体有路、树和两个建筑物，区域中定义了一个包围盒和定制该区域中场景的 XML 文件：

```
<scene defName="whole">
<sub-scene defName="s1">
<entity translation="-20,0,0" fileName="tree.vrml" defName="leftTree" number=20 method="z-" value="5"/>
<entity translation="20,0,0" fileName="tree.vrml" defName="rightTree" number=20 method="z-" value="5"/>
<entity translation="0,0,0" fileName="road.vrml" defName="road" />
<entity translation="-40,0,0" fileName="building1.vrml" defName="building1"/>
<entity translation="40,0,0" fileName="building2.vrml" defName="building2"/>
</sub-scene>
<region>
<ProximitySensor PName="PS1" PCenter="0,0,-200" Plwh="200,200,200">
<PContent filename="ps1.xml" defName= PName="PC1">
</PContent>
</region>
</scene>
```

1.3.5 从局部到整体建模

从局部到整体，就是先从最基本的实体开始，把每个实体的三维模型创建出来，保存成单个文件，并设计好场景构建的 XML 参数文件，在此基础上，通过 Java Applet 程序动态地创建出各层的子场景和最终的整体场景，实体文件之间的连接用 VRML 中的 Inline 命令，各个实体和子场景需要动态改变的数据也可以设计在 XML 参数中，只需对以上设计的 DTD 作扩展，具体可以按照需要由设计者来扩展或重新设计新的元素。

2 VRML 场景生成算法

参数化场景建模是在初始的三维虚拟场景的基础上，利用 Java Applet 程序读取 XML 场景参数，并根据参数的设置动态地在初始场景中添加、删除或替换场景的内容。

初始的三维虚拟场景命名为 index.vrml，其中必须创建一个空的名称为 original 的组节点 Group，称为原始父节点，还可以有选择地在其中构造整个场景的背景、地面、初始视点、灯光效果等内容。index.vrml 下载到浏览器端就由 Java Applet 根据 XML 场景参数的设置向原始父节点的 children 域中添加子场景、实体和区域的包围盒(即 ProximitySensor 节点)等内容，具体是什么内容可以通过改变 XML 的参数设置动态改变。

根据 XML 参数设置动态生成场景首先要解析 XML 元素，实现中采用了效率较高的 SAX 解析方法，针对不同的元素作出相应的处理，具体的场景生成算法如下：

[算法开始]

(1)初始化：

1) 定义数组 scenes，存放父场景的 defName 值，scenes[0]= ”original”；

2)定义数组 proximitySensors,存放所有包围盒的 PName 值；

3)定义数组 Pcontents,存放所有包围盒中动态下载的节点；

4)初始化以上各数组的索引变量 i,j,m=0；

(2)启动 SAX 解析器，捕捉 StartElement 事件，在事件处理中判断元素标记，执行如下操作：

1)如果是 scene 和 sub_scene 元素

①构造 childern 域为空的 Transform 组节点，它的 translation 和 rotation 值和属性值对应；

②调用 scenes[i]标识的父场景的 addChildren 的事件，把构造的组节点加到该父场景中；

③把 i 递增 1,获取元素的 defName 属性值，赋给 scenes[i]；

2)如果是 entity 元素

①构造 children 域为该元素声明实体的 Transform 组节点，它的 translation 和 rotation 域值和元素的属性值对应；

②调用 scenes[i]标识的父场景的 addChildren 的事件，把构造的组节点加到该父场景中；

3)如果是 ProximitySensor 元素

①按照元素的属性构造 ProximitySensor 节点，加到 scenes[i]标识的父场景中；

②把元素的 PName 属性值和当前场景标识 scenes[i]赋给 proximitySensors[j],把 j 递增 1；

4)如果是 PContent 元素

①判断元素的 fileName 是否为空，如果不是空，把 fileName 的属性值赋给 Pcontents[m],把 m 递增 1；

②如果是空的，读取元素的内容，把内容作为字符串值赋给 Pcontents[m],把 m 递增 1；

5)如果是 content 元素，直接读取元素的内容创建 VRML 节点，调用 scenes[i]标识的父场景的 addChildren 的事件，把构造的组节点加到该父场景中；

(3)捕捉 EndElement 事件，判断元素标记，如果是 scene 和 sub_scene 元素，把 i 递减 1。

[算法结束]

其中场景的分区域下载采用了临近检测法，具体算法如下：

[算法开始]

循环检测以上 proximitySensor 数组的值，对每个值执行如下操作：

(1)根据 PName 值获取场景中传感器节点；

(2)检测场景中 ProximitySensor 节点的 isActive 属性；

(3)获取当前父场景节点；

(4)若 isActive=true,读取数组 Pcontents 中的相应值，构造子场景，并调用当前父场景节点的 addChildren 事件，把子场景动态下载添加到父场景中；

(5)若 isActive=false,检测该包围盒的子场景是否在场景中，如果存在，就调用父场景节点的 removeChildren 事件，把子场景删除。

[算法结束]

3 应用举例

参数化虚拟场景建模技术在虚拟城市的构建项目中得到了应用，并充分体现了可定制性、可重用性和开发效率的优

(下转第 217 页)