网络环境下改善虚拟场景实时显示的方法

Method to Improve Real-time Displaying virtual Scene in Network Environment

林冬梅（佛山科学技术学院信息与教育技术中心 528000）
王 东（佛山科学技术学院工学院计算机系 528000）

摘 要：本文针对基于网络环境的大型虚拟场景的实时显示问题，从减少可视多边形的个数及动态控制场景的生成两个方面，提出了相应的技术解决方案并给出了相关的算法。最后给出了在VRML中影响场景生成速度的其他因素。

关键词：虚拟现实 虚拟场景 实时显示 纹理映射

1 前言

虚拟现实技术是一种高度逼真地模拟人在自然环境中视、听、动等行为的人机交互技术，这意味着计算机必须对操作者的各种行为做出实时响应。因此，图形的实时显示成为虚拟现实技术的一个重要方面。所谓实时显示，是指当用户的视点变化时，图形显示速度必须跟上视点的改变速度，否则就会产生迟钝现象。研究表明，计算机在绘制图像时，当帧与帧之间的时间间隔大于16ms时，人眼能明显感觉到画面的不连续。这种停顿感会严重影响到所产生的虚拟环境的真实感。上述问题可以通过硬件性能的提高得到改善，然而，目前的硬件水平还不能完全解决这个问题。因此，图形绘制的帧频率成为提高虚拟环境真实感的一个瓶颈。为此，人们通常采用的方法是尽可能减少每一帧需要绘制的多边形的个数，从而提高显示绘制的速度。

对于用VRML实现的网络虚拟现实而言，由于VRML的访问方式是基于浏览器/服务器模式的。其中，服务器提供VRML文件及支持资源（图像、视频、声音等），通过网络下载到客户端，并通过本地平台上的VRML浏览器交互式地访问该文件描述的虚拟环境。因此网络虚拟场景的实时显示还与另外一个因素有关，即网络带宽。由于较低网络带宽，使得下载用VRML建造的大规模场景速度较慢，等待时间较长，并且渲染速度也较低。这样在建造大型的虚拟环境，例如虚拟校园（图1）的最大挑战就是针对场景中包含的大量的多边形及纹理，还要求有较高的显示速度。因此，对场景进行优化，提高显示速度，要解决的主要问题就是如何提高下载速度，以及如何实时生成及显示网络虚拟世界。

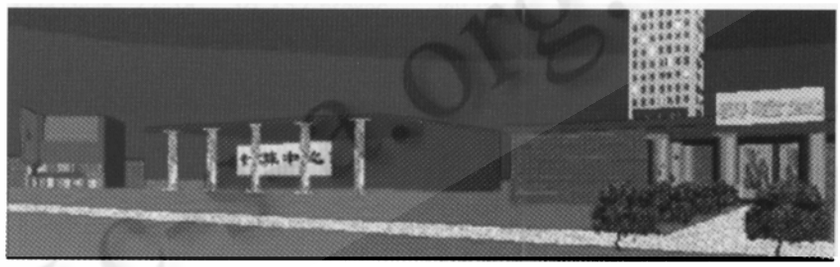

图1 虚拟校园

由此可知，对于网络虚拟现实而言，改善场景的实时显示主要从两方面来考虑，一是减少可视多边形的个数，二是控制文件的下载顺序，减少冗余文件的下载，动态控制场景的生成，从而提高场景的显示速度。

2 减少可视多边形的个数，提高场景生成速度

对于大型虚拟场景优化，必须提高它的显示速度，又得保证其逼真性。提高帧显示速度的方法是：在显示每帧时，在既保证显示的质量不会下降得太多又保证不影响用户视觉效果的前提下，尽量减少场景中包含的多边形个数。在虚拟环境中，减少可视多边形的常用方法有：LOD技术、mipmapping技术等，对这些技术不再详细介绍。下面主要针对在网络环境下，利用虚拟现实模型语言VRML所建造的大型虚拟场景，所采用的其他方法及有关算法。

2.1 限制化身的运动范围

这种方法的基本思想是：当化身在虚拟世界中漫游时，限制化身的自由，使得禁止到达的

区域中的物体是不可见的，因此，此区域中的物体不被绘制。

例如，在虚拟校园中，有很多的建筑物，这些建筑物内有很多相应的物体，事先规定这些建筑物的窗户都是半透明的，用户通过窗子不能看清楚里面的物体，因此用户在打开门之前，这些物体可以不被绘制。从而可以减少绘制的多边形的数目。这里VRML 2.0的碰撞检测机制很有用，使得在禁止到达的区域，化身不可能穿墙而过。

另外，还可以利用“导游”漫游方式，通过ViewPoint将用户绑定到指定的视点，如果将一个事件发送给初始ViewPoint节点的position域和orientation域，则用户的视野发生变化。通过将插值器连接到这些域中，便可以创建一条有导游的游览路线。化身被限制在指定的路线行走，因此，只需要绘制化身所到达的场景，对于那些化身暂时不能到达的场景可以不必绘制。

2.2 限制物体可见性

在VRML中，VisibilitySensor节点能够用来检测化身当前时刻能看到场景中特定的区域。利用该节点的特性，能够检测化身的可视距离，所有可视距离以外的物体可以不被绘制。当这些物体进入化身的可视距离时，如果不进行相应的处理，物体会突然出现在化身面前，会令用户感到不真实。为了改善这种不理想效果，利用VRML的Color节点和Material节点进行特殊编程，使得远处物体具有较低的色素值和简单的纹理，当接近时再变得明亮些。这样就会自然些。

具体实现算法如下：

[算法开始]

（1）初始化

① 定义合适的VisibilitySensor和ProximitySensor感应器，使得ProximitySensor的中心区域比VisibilitySensor的中心域距离用户要近，ProximitySensor感应器的个数可以根据具体情况而定为一个或多个。

② 定义物体最初可见的较低色素值颜色和简单纹理。

（2）定义Script节点

其中包括多个用来改变物体颜色和纹理的函数，激活不同的感应器，将调用不同的函数。各函数用到的主要数据结构如下：

eventIn SFTime touchTime ;感应输入事件

eventOut SFColor Color ;改变对象颜色的输出事件

eventOut MFString clock ;改变对象纹理的输出事件

eventOut SFFloat transparency ;改变对象透明度的输出事件

field SFNode node USE 已定义过的某一物体节点；访问场景中的节点上面的各数据项都可能为多个。

（3）建立路由

① 根据用户在场景中的位置，调用相应的函数。

② 利用ROUTE…TO…将函数中所定义的颜色及纹理传送给相应的对象。

[算法结束]

另外，对于某些被遮挡的物体，只有用户在适当的位置才能看到，这种情况也可以利用上述方法，在没有到达合适的位置时，使得这些物体不可见。只有在适当的视觉范围之内时，才使该物体可见。

2.3 用二维纹理代替三维模型

对于复杂场景中具有非常复杂细节的物体，例如花草、树木等，若用三维模型表示，将需要大量的多边形，对场景的渲染速度有很大的影响。为了提高复杂场景的渲染速度，利用VRML提供的BillBoard节点的功能，制作一个最简单的四边形。然后，将物体的图象作为纹理粘贴在四边形上，将这个四边形作为BillBoard的children域即可。因为，BillBoard节点能使帖有纹理的二维图形无论化身怎样走动都始终面向化身，其效果比用大量的多边形来制作好得多。在物体离视点很远时，也没有必要用多边形去构造它。这时，用物体的图像粘贴在设计好的相应的平面上，并放置在场景中。

以上方法实质就是用适当的二维纹理代替三维模型，它具有很强的真实性，节约生成视景所耗用的时间，并且可以增加实时效果。

3 优化场景文件，提高文件下载及场景生成、显示速度

VRML文件.wrl在网络上是以压缩格式wrz进行传输的，文件的文本部分，被压缩了20倍甚至更高。而文件中的纹理部分已经是以压缩文件形式存在，不会再被压缩太多。因此，纹理图象文件占据了整个传输文件的绝大部分。但还能够通过以下技术使VRML文件中的文本和纹理图象部分变得更短。

3.1 场景重用

在建造虚拟世界的过程中，虚拟世界中可能会存在很多相同的对象或许多对象用相同的材质作为纹理。如果对这些相同的对象反复进行建模，将明显地增加场景文件的尺寸，因此应该对它们进行单独定义，然后对这些对象重用。在VRML中可以有三种途径。

（1）EF和USE。建造虚拟景象时，其中有很多物体是相同的，这时只需建立一个模型，并用DEF对它命名，在其他需要此模型的地方只需用USE对它重用即可。这样，程序的源代码量明显减少。

（2）内联。Inline功能提供另一种打包类型并且具有收集节点和路由的能力，可以从万维网的某个位置读取指定的数据。这样在多个虚拟场景中可以共享同一个部分场景。

（3）原型构造。VRML中的原型构造为创作人员提供了收集节点和路由的能力，并且提供一个标准接口。通过原型构造，可以在VRML中创建扩展节点，原型节点一旦定义，就可以同内部节点类型一样在场景中实例化。

3.2 减少纹理图象的大小

建造逼真的VRML世界，必然要用到纹理，而不只是一些单调平滑的轮廓模型，因

此，设计者必须尽可能地减小GIF或JPEG文件的大小，以减少其传输及下载速度。

对于GIF文件格式来讲，通过降低颜色深度到4位（16色）或者更低来实现。低细节的纹理，清晰的轮廓和平缓的颜色变化都能提高GIF的压缩能力。

对于JPEG格式，在保真率相同的情况下，最好使用平滑的图像，因为平滑的图象比颜色反差较大的图象压缩率要高。另外，只有几个像素点大小的纹理对文件的大小影响不是很大。

3.3 控制场景文件下载的顺序

由于基于网络的虚拟环境的生成是基于浏览器/服务器模式的，默认的情况下，用户在客户端浏览一个虚拟场景，需要下载构成该场景的各种文件，包括用于定义物体的VRML文本文件，以及这些文件中用到的相关图象、声音、视频等文件。无论用户浏览场景时是否会用到它们。显然下载这些文件需要很长时间，因此，应当解决如何根据用户浏览的具体情况，动态下载生成场景所需的文件，控制场景的生成。

（1）控制场景生成的算法。由于本文所研究的基于网络环境的虚拟场景的构造，主要是基于建筑物的，为了实现动态下载场文件，控制场景的生成，必须将场景按照空间可见性分成若干区域。由于遮挡关系，在用户没有到达某些区域时，此区域的场景就不必生成，只有接近此区域时，或者对于封闭的空间，只有打开门时，再对这部分的场景进行生成。对于这些场景中的特殊区域（例如门、窗），将之称为port，在上述前提下，控制场景生成的算法如下：

[算法开始]

① 构造分场景

* 按场景划分结构，将各部分场景独立构造，以独立文件存放

* 建造各独立场景时，在适当port区初始化检查化身位置或行为的相应的感应器

② 生成主场景

* 按照初始视点，即化身的初始位置，生成初始场景

* 跟踪化身的位置变化，检查化身是否到达某一port

若已到达，则 判断与此port相关的部分场景已被生成否。

若已生成，则以后不再检查此区域。

否则，下载相应的场景文件，生成该场景。

[算法结束]

（2）算法实现的主要相关技术

① 用JavaScript和EAI实现动态场景调用

在上述算法实现过程中，除了场景构造之外，最关键的是如何控制某一部分场景的生成。控制场景生成的关键在于如何动态地调用某一场景文件。如果静态地调用场景文件，则需通过执行VRML中Inline节点所设置的URL域。但要动态调用场景文件，就必须借助VRML与Java之间的通信才能实现。有两种方式，第一种方式将JavaScript作为Scripting语言用于VRML的Script节点中；第二种方式，就是通过EAI接口技术，从外部控制VRML世界。用上述两种方法中的任意一种都可以实现对场景的动态调用。

② 局部与全局的坐标转换

在网络虚拟现实中，环境共享是非常重要的，要确保环境中各位置和视点的图形在不同系统中显示的正确性和一致性，必须将不同系统的局部坐标转换到整个环境中的全局坐标。在上述算法中，对于分场景中的各感应器保证它在主场景能够准确感应到化身的位置，也必须在调进主场景时，进行坐标变换。这对于场景随着化身的移动而正确的动态生成相应的场景是至关重要的。

4 在虚拟环境中影响场景实时显示的其他因素

用VRML生成虚拟场景，还要考虑以下几个对实时显示有影响的因素：

（1）描述环绕Group节点的立方体中心和大小的bboxCenter和bboxSize域，要精确计算，因为它们将影响Group节点所包含物体的渲染速度。

（2）对顶灯和碰撞检测的动态处理，有时也是很好的办法，只有在必要的时候，才将它们打开。

（3）NavigationInfo节点的各个域对帧速度也有很大的影响。

5 改善的效果

将上述这些方法用于虚拟校园中，实时显示的改善效果从两个方面考证，一方面是场景文件的下载时间也就是初始场景的渲染时间，另一方面是漫游虚拟场景时实显示的速度。例如在PIII 800的计算机上，原始的虚拟校园场景的渲染时间是4.2秒，优化后场景的渲染时间是1.9秒；未优化前漫游虚拟场景的帧速度是6.57fps，优化后漫游虚拟场景的帧速度是12.39fps，由此可见，采用优化策略后，具有较高复杂度大场景的初始生成时间及绘制速度几乎提高了一倍，而对于较低复杂度的场景，也有显著提高。

综上所述，在网络环境用VRML生成虚拟场景，网络传输和场景实时显示是应当首要考虑的问题。利用以上所提出的优化技术，这两方面的问题都在一定程度上得到改善。

参考文献

1 Chris Marrin,Bruce Campbell 21 天学通VRML 2.0，人民邮电出版社。

2 王晓丹，虚拟环境中的实时性改善途径，计算机应用研究,1999.6。

3 汪成为等，灵境（虚拟现实）技术的理论、实现及应用[M],清华大学出版社，1996。