三维虚拟环境中二维平面动态实时渲染方法

杨 扬 肖 飞 孟 坤 赵晓永
(北京科技大学信息工程学院 北京 100083)

摘 要 为实现三维虚拟世界中二维平面动态实时渲染,提出一种有效的方法并通过实验验证了其可行性。通过在三维虚拟环境中加载该方法,不仅可以实现用户与三维环境中图像交互,而且可以在该系统上集成传统的二维图像业务,弥补了当前三维虚拟世界技术对二维平面业务支持不足的缺点。

关键词 三维虚拟现实,平面绘图,纹理映射,坐标变换,实时渲染
中图法分类号 TP37 **文献标识码** A

Solution to Render the 2D Timely in the 3D Visual Reality

YANG Yang XIAO Fei MENG Kun ZHAO Xiao-yong
(School of Information Engineering, University of Science and Technology Beijing, Beijing 100083, China)

Abstract This paper gave a solution to render the 2D timely in the 3D visual reality, and testified the feasibility of this solution. To load the solutions in the 3D visual reality not only can realize users' interaction with the 3D environment, but also can integrate the excellent 2D applications in 3D visual reality which can cover the defects that current 3D visual reality has little support to the 2D applications.

Keywords 3D visual reality, Dimensional painting, Texture mapping, Coordinate transformation, Real time render

1 引言

20世纪90年代初,第一次提出"虚拟现实"技术,之后便以其独特的魅力迅速成为计算机界最为活跃的研究领域之一[1]。其发展极大地改变着人类的生活方式与娱乐模式,并催生了大批基于该技术的新兴产业。

当前,三维虚拟现实技术的研究主要集中在场景的渲染和模型的精细,对二维纹理动态实时渲染方面的研究还比较匮乏。虽然,目前网络游戏通过采用更新地图或模型所对应的图片来实现三维环境中的渲染效果,但是多数三维虚拟环境仍无法支持视频的实时播放;虚拟教育所用的"黑板"上的板书内容也无法同步地呈现给用户;传统的平面广告、宣传海报等诸多对生活有着无可替代作用的二维元素无法很好地融入三维虚拟世界场景当中等等。目前,上述现象已成为了三维虚拟现实技术最大的缺憾,极大地限制了三维虚拟现实技术的应用范围。

事实上,一方面,由于三维虚拟场景的复杂性和特殊的视觉要求,直接导致在其中对二维图像实施实时渲染需考虑更多的因素,包括位置信息、色彩及视角信息等,增加了动态实时渲染场景的难度[2]。另一方面,传统的解决方式是通过替换场景中相应的模型纹理图片来达到重新渲染的目的,需要先下载图片后重构,然后重新加载图片,其效率和响应速度成为该方法广泛应用的瓶颈。如何突破传统方式,设计实现高效的三维虚拟场景中的二位平面的实时渲染,成为影响虚拟现实技术发展和应用关键问题之一。

随着计算机图形学不断发展,可编程着色技术的出现无疑在三维渲染技术中有着划时代的意义。在2005年的SIGGRAPH会议上,Damien Porquet[3]等人展示了基于可编程着色技术之上的纹理映射的强大特性。对每一顶点或对每一像素的运算可以放到程序片段中来进行,通过使用诸如C++之类的高级语言,或通过GPU的汇编语言,实现用程序来取代之前固定功能管道所做的事情[4]。此技术的采用大大提高了渲染效率,为实现实时渲染打下了基础。针对三维虚拟环境中的二位平面的实时渲染,我们提出了二维与三维坐标间的位置转换映射,以及模型与纹理渲染间的映射。通过采用上述技术,提出并实现了一种在三维虚拟环境中实时渲染二维平面的方法。通过试验分析了该方法的效率和资源占用,并将它应用于实验室自主开发的远程虚拟教育平台之中。

本文第2节给出三维虚拟环境中二维平面动态渲染的理论基础及具体方法;第3节给出该方法的实现,并依托三维虚拟教育平台,通过对比分析,说明该方法的高效性;最后给出总结和未来的研究方向。

2 二维平面动态渲染方法

本文采用的方法是基于纹理映射的。下面将从理论基础及实现两个方面对该方法展开讨论。

三维场景中要想渲染对象,首先要得到该对象在三维环境中的坐标,而最通用的方式便是通过鼠标的点击获得。同

到稿日期:2010-07-08 返修日期:2010-11-11 本文受国家自然科学基金项目(60873192)资助。
杨 扬(1952—),男,教授,博士生导师,主要研究方向为多媒体技术与网络通信,E-mail:yyang@ustb.edu.cn;肖 飞(1985—),男,硕士生,主要研究方向为多媒体通信;孟 坤(1980—),男,博士生,主要研究方向为性能分析;赵晓永(1981—),男,博士生,主要研究方向为云计算与服务理论。

时，通过对所渲染模型纹理上的相应坐标点绘制像素来达到总体渲染的目的。基于此，提出了“虚拟画布”的概念。所谓“虚拟画布”，是一段开辟在内存中的一段数据结构，主要用于建立和材质纹理之间的映射关系。按下面的渲染关系（如图1所示）实现了动态渲染，通过将绘图平面数据转化到虚拟画布中，使虚拟画布中的颜色数据与材质纹理中相应位置数据相对应，来实现总体的动态渲染。

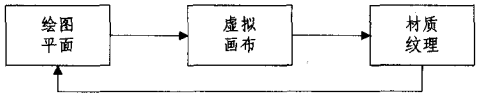

图1　渲染关系图

2.1　理论基础

将纹理空间中的点像素映射到三维空间中的点像素的过程称为纹理映射。纹理空间由许多的像素点组成，其中每一个像素点都对应一个纹理坐标。若这些纹理坐标沿着UV坐标被映射到[0,1]之中，则称之为UV纹理映射。

图像的渲染离不开两个重要数据：位置坐标和该坐标上相应的颜色数据。三维场景的渲染也是如此。我们所实现的方法主要定位于这两种数据之间的关系。

假设纹理处于直角坐标系中，其对应色点的坐标为(u,v)，即它的颜色属性可以表示为$f(u,v)$。三维模型表面纹理的空间坐标为(x,y,z)，其颜色属性可以表示为$g(x,y,z)$。屏幕上点坐标为(x_v,y_v)，其颜色属性可以表示为$r(x_v,y_v)$，于是问题可简化为从坐标点(x,y,z)与坐标点(x_v,y_v)之间的映射。

从纹理到模型表面渲染点之间的映射：

$$r(x_v,y_v)=\theta(g(x,y,z))$$

从纹理到屏幕点之间的映射：

$$r(x_v,y_v)=\sigma(f(u,v))$$

若要在三维虚拟环境中实现平面纹理动态交互渲染，只要找到一种映射使以下关系成立即可：

$$\theta(g(x,y,z))=\sigma(f(u,v))$$

2.2　屏幕捕捉点的三维坐标与二维坐标转换

当在三维虚拟场景中的某平面上动态绘制鼠标划过的图像时，首先要解决的问题是如何建立鼠标所捕获的屏幕坐标点(X,Y)与该平面所处三维空间坐标点(x,y,z)之间的关系。在该问题上主要采用通过包围体的碰撞检测算法[5,6]获得所操作的模型，并通过光线检测的碰撞检测算法[7]来判定鼠标点击三维空间坐标模型上相应的点。实现通过相应算法将两种数据进行转化，将所转换过的数据存入“虚拟画布”。这里，采用右手坐标系，如图2所示。

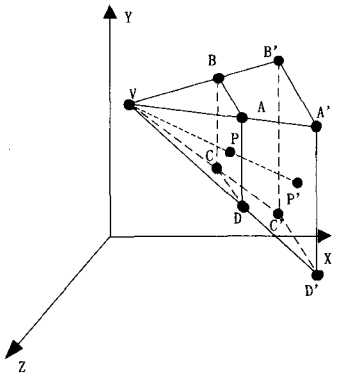

图2　三维空间模型与其坐标系

V点为当前屏幕上某点（鼠标在屏幕上的坐标点），平面$ABCD$与平面$A'B'C'D'$分别为空间中的两个不相交的平面，两平面均平行于XOY平面。在三维虚拟场景中分别表示两个平面模型，直线VP'与平面$ABCD$相交于P点，与平面$A'B'C'D'$相交于P'点，VP'垂直于屏幕。现在需要通过鼠标在空间平面$ABCD$上进行动态绘图。

鼠标点击屏幕上的V点，此时创建一条虚拟光线R垂直于屏幕并沿Z轴负方向伸长。该直线将与平面$ABCD$相交于P点，与平面$A'B'C'D'$相交于P'点，即该线与直线VP'重合。当直线R与平面相交时，根据光线检测算法可反馈P点在空间的坐标，同时将该模型添加入迭代器中。

碰撞检测模块反馈回欲渲染平面$ABCD$上的点P的空间三维坐标(X_p,Y_p,Z_p)。通过二维坐标转换模块将P点的空间三维坐标转换为渲染纹理的二维坐标，即构建映射关系：$r(x_v,y_v)=\theta(g(x,y,z))$。

具体算法如下：

假设平面$ABCD$大小为$N*M$，其法向量沿Z轴正方向并垂直于XOY平面，单位与三维环境所用标量相等。其中心点O处于空间坐标(X_o,Y_o,Z_o)，平面材质为$T*T$像素。

$T=2^n$，n取大于0的整数。

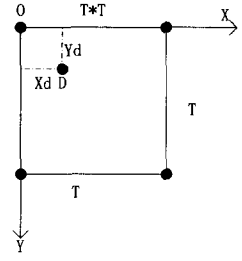

图3　虚拟画布

虚拟画布大小为$T*T$像素。通过将平面上P点的数据转换到虚拟画布中的$P_d(X_d,Y_d)$点，进而实现对该点的渲染，虚拟画布如图3所示。根据几何学中平面坐标平移的方法可以得到如下公式：

$$\begin{cases} X_d=\dfrac{(X_p-X_o)+\dfrac{N}{2}}{N}\times T \\ Y_d=\dfrac{-(Y_p-Y_o)+\dfrac{M}{2}}{M}\times T \end{cases}$$

如果该平面法向量沿Z轴负方向，则有：

$$\begin{cases} X_d=\dfrac{-(X_p-X_o)+\dfrac{N}{2}}{N}\times T \\ Y_d=\dfrac{-(Y_p-Y_o)+\dfrac{M}{2}}{M}\times T \end{cases}$$

对于平面法向量沿X轴正方向时，则有：

$$\begin{cases} X_d=\dfrac{-(Z_p-Z_o)+\dfrac{N}{2}}{N}\times T \\ Y_d=\dfrac{-(Y_p-Y_o)+\dfrac{M}{2}}{M}\times T \end{cases}$$

同理，平面法向量沿X轴负方向时有：

$$\begin{cases} X_d=\dfrac{(Z_p-Z_o)+\dfrac{N}{2}}{N}\times T \\ Y_d=\dfrac{-(Y_p-Y_o)+\dfrac{M}{2}}{M}\times T \end{cases}$$

式中，X_d 为虚拟画布上渲染的点的 X 坐标，Y_d 为虚拟画布上渲染的点的 Y 坐标。

最后将转化得到的二维坐标储存到渲染列表中，等待渲染模块的渲染。

2.3 基于纹理着色的动态渲染

假设待渲染平面模型处于虚拟三维环境的 (X_o, Y_o, Z_o) 点。该模型表面上附着空白的 $T*T$ 像素的纹理材质。同时将该纹理管道与虚拟画布进行衔接。

对于视频流数据以及网络的广告数据，可以根据数据视频或图像大小动态构建虚拟画布。

通过纹理映射技术将内存中的图像数据映射为屏幕显示：

$$r(x_v, y_v)=\sigma(f(u,v))$$

具体算法描述如下：

在内存中开辟一段大小为 $4*T*T$ 字节的缓存，用于及时储存当前帧中材质纹理所要渲染的数据。纹理中每一像素都对应缓存中 4 个字节的空间，这 4 个字节的空间中分别储存着该像素点上的红、绿、蓝颜色分量的值 RGB 以及透明度值 A。当有数据接收时，将判断是来自本地的绘制数据还是来自网络的图像或视频流数据。对于本地的绘图数据，则遍历渲染点来将各点数据渲染到虚拟画布纹理中。对来自网络的视频数据流或图像数据包进行解析，解析出每帧视频流所对应的图像信息并存放入虚拟画布缓存中，等待每帧进行渲染。

设 T_0 表示输入数据像素位置：

$$T_0=[x_0 \quad y_0 \quad 1] \tag{1}$$

T_1 表示渲染纹理像素位置：

$$T_1=[x_1 \quad y_1 \quad 1] \tag{2}$$

F 表示输入数据与纹理数据映射：

$$F=[f_x \quad f_y \quad 1] \tag{3}$$

于是有如下表示：

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}=\begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{4}$$

式中，

$$f_x=\begin{cases} \dfrac{Height}{T}, & \dfrac{Height}{T}\in Z \\ \left[\dfrac{Height}{T}\right]+1, & \dfrac{Height}{T}\notin Z \end{cases} \tag{5}$$

$$f_y=\begin{cases} \dfrac{Width}{T}, & \dfrac{Width}{T}\in Z \\ \left[\dfrac{Width}{T}\right]+1, & \dfrac{Width}{T}\notin Z \end{cases} \tag{6}$$

$Height$ 为输入数据的图像高度，$Width$ 为输入数据的图像宽度。根据以上表达可以将输入的 $Width*Height$ 大小的图像转化到 $T*T$ 的渲染纹理缓存中。新的渲染区中的空位用双线性插值算法[8]进行填充。

另 $0<x<1, 0<y<1$，于是 $g(x,y)$ 表示二维平面纹理空间中相邻两点像素之间的空位插值。

$$g(x,y)=g(0,0)+x[g(1,0)-g(0,0)] \tag{7}$$

$$g(x,1)=g(0,1)+x[g(1,1)-g(0,1)] \tag{8}$$

$$g(x,y)=g(x,0)+y[g(x,1)-g(x,0)] \tag{9}$$

将式(7)，式(8)两式代入式(9)中，会得出两输入点像素之间的插值。通过该值来渲染原相邻像素在新的渲染纹理之中的空位。

渲染算法

输入：渲染数据列表。

Step1 根据不同渲染数据来源与类型创建与之相应的虚拟画布并将接收到的数据按其类型放入渲染列表中。

Step2 获得渲染纹理的数据缓存区，并锁定该区域所在的内存。

Step3 创建一段 $4*T*T$ 字节的缓存区，用来转换图像数据，并将该缓存拷贝给渲染纹理的数据缓存区。

Step4 循环遍历虚拟画布中的数据，将画布中 (x,y) 像素点的数据拷贝给数据缓存区。转换格式依次存放顺序为：虚拟画布的蓝色管道 B、绿色管道 G、红色管道 R 和透明度管道 A。

Step5 每转换该点数据后，宽度标示符 $x+1$，当宽度标示符 x 大于或等于 T 时，高度标示符 y 加 1，然后开始读取画布 $(x,y+1)$ 像素点的数据。

Step6 当 y 大于或等于 T 时，结束遍历并释放掉该缓存区域。

输出：经过纹理映射后待渲染的纹理通道。

}

通过构建虚拟画布作为衔接三维材质渲染和二维图像纹理处理的桥梁。在我们所构建的系统中采用 $T*T$ 像素的位图虚拟画布，数据结构如下所示：

```
VirtualMap
{
  INT x;//图像横坐标
  INT y;//图像纵坐标
  Color getPixel(INT x,INT y);//x,y 点颜色
  VOID setPixel(INT x,INT y,Color col);
  //填充颜色
  HBITMAP mBit;//封装位图格式
  VirtualMap * fromBITMAPINFO(
  const BITMAPINFO * gdiBitmapInfo,
  VOID * gdiBitmapData);
  //封装图像头文件
  Bitmap * fromHBITMAP(HBITMAP hbm,
  HPALETTE hpal);
  //封装图像格式
  BOOL intrudePixel(Pass * pass,MaterialPtr mPtr);
  //衔接虚拟画布与渲染纹理
}
```

3 系统实现

该系统构架主要由 3 个模块组成：纹理渲染模块、输入设备捕获模块、数据转换模块，如图 4 所示。

输入设备捕获模块对输入设备进行监听并捕获相应输入数据，包括网络传输过来的数据以及鼠标键盘或其他外接设备所输入的数据。

数据转换模块由碰撞检测模块和坐标转换模块组成。碰

撞检测模块主要响应鼠标事件并返回点击点的三维空间坐标点。坐标转换模块主要将所收到的三维坐标点通过特定算法转化为虚拟画布上的二维坐标点，并将该点储存到渲染列表中，交给渲染模块进行渲染。

纹理渲染模块由 GDI 渲染模块和纹理映射模块组成。其中，GDI 渲染模块是对接收到的数据进行处理，并根据所渲染的方式在内存中建立虚拟画布，将欲渲染的数据绘制在虚拟画布上。纹理映射模块主要将虚拟画布以特定方式与所要渲染的模型或区域上的纹理进行映射，将虚拟画布数据动态渲染到该纹理中并显示。

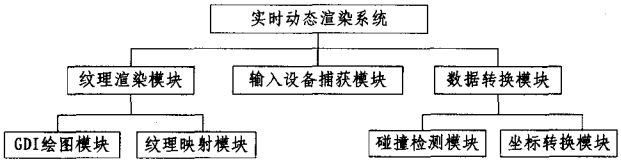

图 4　系统组成模块

对于普通的三维虚拟世界而言，每秒渲染 30 帧，则可得到很好的用户体验。流媒体的播放，每秒达到 24 帧左右就可以流畅观赏。

通过在三维虚拟世界中运用本系统，便可在三维虚拟世界中实现远程视频播放，以及通过网络在虚拟世界中发布公告，张贴广告，实现网络白板等。传统的图像业务也可以应用在三维虚拟世界当中，而不需要重新构建环境或加载模型。系统速度快，渲染图像清晰流畅。

该方法易于集成，应用范围广，可应用于多种当前主流的三维虚拟世界或 3D 网络游戏之中。它已经成功集成到北京科技大学网络与多媒体通信实验室自主研发的三维远程教育平台之中，运行效果好，渲染帧数快。通过该方法可以实现虚拟教学、虚拟远程会议等诸多强大的功能。

图 5 为运用本系统在三维环境中播放视频的效果。通过网络数据对平面进行实时渲染，从而播放视频。图 6 为三维虚拟世界中平面纹理的动态绘制。通过鼠标在白板上进行动态绘图，实现三维空间中网络白板的功能。处于三维世界中的虚拟人物从不同的角度观看可以获得不同的播放效果，真实地模拟了现实中的场景。

图 5　在三维虚拟世界中的视频流播放

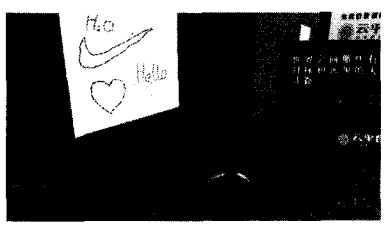

图 6　虚拟世界中平面纹理的动态绘制

图 7 为我们在不同配置的电脑上进行系统实现的数据。在三维虚拟环境中播放一段标准 wmv 格式视频，视频参数如下：帧宽度，728；帧长度，480；帧速率，29FPS。

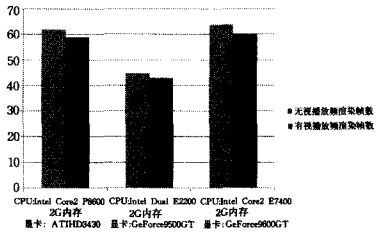

图 7　运行效果比较

可以看到，对于当前标准配置的机器可以跑到每秒 40 帧以上。当在该环境中用本方法实现视频流的实时播放后，总体平均帧数仅下跌 2～4 帧。经过实验验证，在三维虚拟世界中应用该方法可以实现在三维虚拟环境中实时渲染，包括视频流在内的二维图像业务，且内存占用率低。对于二维图像的动态渲染响应速度快，图像清晰。

结束语　提出了一种将传统的二维图像业务集成入三维虚拟世界中的方法，并设计系统实现了在三维虚拟环境中动态实时展现二维元素。通过构建合适的“虚拟画布”，并结合坐标转换技术和纹理映射技术，使二维元素在三维环境中得以展现，并能满足多数实时性的要求。但是，对于处在复杂网络环境下的多用户而言，各用户 PC 机配置不同，导致其渲染帧数也不同。对于实时性要求很高的网络流媒体业务，其保证在三维环境中的同步性，有待于进一步研究。

参 考 文 献

[1] Novak-Marcincin J. Application of the Virtual Reality Modeling Language for Design of Automated Workplaces [J]. World Academy of Science, Engineering and Technology, 2007, 31: 160-163

[2] Yan Han-bing, Hu Shi-min, Martin R. 3 D Morphing Using Strain Field Interpolation [J]. Journal of Computer Science & Technology, 2007, 22(1): 156-160

[3] Porquet D, Dischler J-M, Ghazanfarpour D. Real-time high-quality View-Dependent Texture Mapping Using Per-pixel Visibility [C]// SIGGRAPH. 2005: 213-222

[4] Junker G. Pro OGRE 3D Programming [M]. America: Apress

[5] Kamat V V. A survey of techniques for simulation of dynamic collision detection [J]. Computer & Graphics, 1993, 17(4): 379-385

[6] Klosowski J T, Martin H, Joseph S B, et al. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs [J]. IEEE Transactions on Visualization and Computer Graphics, 1998, 4(1): 21-36

[7] 候丽平，刘越，王涌天. 基于光线跟踪的碰撞检测技术[J]. 系统仿真学报，2006，18(1 增)：84-87

[8] 何东健. 数字图像处理(第 2 版)[M]. 西安：西安电子科技大学出版社，2008