

vBSP 的算法设计与应用

段 富, 李 强

(太原理工大学 计算机与软件学院, 山西 太原 030024)

摘 要: BSP 数据结构已经在大规模的游戏引擎产品中取得了突破性的成功, 但是在现实应用中还有许多需要实际改进后才能应用。为此, 改进 BSP 开发出一种简单、灵活的解决方案, 即虚拟 BSP (vBSP) 来解决实际应用。通过研究并改进原始的 BSP 树, 意在使 BSP 树简单化、实用化, 并可以减少算法的复杂度, 降低其它改进算法的负面影响。基于“空间换时间”的思想, 通过开辟引用存储空间而不是直接分割 BSP 树的方法, 最终达到提高算法的实用性。

关键词: 三维引擎; 虚拟二叉空间分割技术; 二叉空间分割技术; Z 缓冲

中图分类号: TP391.9

文献标识码: A

文章编号: 1673-629X(2007)12-0119-03

vBSP Algorithm and Application

DUAN Fu, LI Qiang

(School of Computer & Software, Taiyuan University of Technology, Taiyuan 030024, China)

Abstract: BSP datastruct is successfully applied in great 3D game products, but this method must be improved to be applied virtually. This article will give a simple and flexible solution with an improved BSP method, that is virtual BSP which will resolve the application in practice. This paper will predigest algorithm complexity and make it more practicable and reduce the negative effect of improved solution by research on original BSP tree. The vBSP will use index buffer but dividing BSP tree which based on the idea of "Time can be changed into Space". Finally, it improved the algorithm performance.

Key words: 3D engine; vBSP; BSP; Z buffer

0 引 言

作为大规模的场景管理方案, BSP 以及它的扩展算法已经在国际著名游戏中取得了巨大的成功, 越来越受到国内外图形引擎设计人士的普遍关注。由于它具有很多良好的灵活性和可扩展性, 通过空间换时间的思想, 能够取得时间和空间的比较完美的平衡, 并得到运行时的资源和时间的性价比等优势, BSP 已经成为 3D 图形引擎设计的首选方案。另外随着 VR, 3D 游戏, 三维仿真系统的需要, BSP 已经成为三维图形引擎的主流核心, 更使得 BSP 算法以及改进算法成为大型图形系统的重要手段^[1,2]。

传统的 BSP 树生成会带来很多其他问题, 如 BSP 树的平衡、多边型的分割等等, 需要子算法来解决。作为三维图形引擎核心的传统的 BSP 具有以下特点: (1) 管理大量的多边形; (2) 迅速决定在任何位置哪些多边形该显示; (3) 只绘制显示的多边形; (4) 每个像素只画一次等。但是直接建立 BSP 树会影响建模文件,

并且造成了顶点的破坏以及在其上的纹理信息。针对这些问题, 为了不破坏模型文件及顶点和纹理信息, 并对场景实现由到远的渲染, 避免了 Z Buffer 算法对系统的巨大开销, 笔者提出了一新的虚拟 BSP 来解决该实际应用问题, 也就是用 vBSP 来创建 3D 引擎的核心系统。

1 vBSP 解决方案思想

1.1 方案的基本构想

原始 BSP 思想^[3]是“空间换时间的思想”, 通过静态预处理程序建立二叉数据结构, 存储面片信息且通过二叉树的二分关系反映面片的前后的制约关系, 达到在运行时的面片关系快速反映。vBSP 的思想受网络域概念的启发, “只对网络进行逻辑划分, 而避免了对物理上的直接划分”, 建立 vBSP 二叉树只对面片编号进行索引, 从而避免了对面片的直接物理切割划分, 只进行逻辑上的划分。

1.2 方案的整体步骤

首先, 把模型文件的信息导入内存中包括对象的面片、顶点的索引、顶点、纹理和纹理坐标, 然后根据原

收稿日期: 2007-02-29

作者简介: 段 富 (1958-), 男, 山西怀仁人, 硕士, 教授, 研究方向为图形图像。

始 BSP 的思想对其面片信息进行建立缓存引用,然后构建 vBSP 树,对面片进行管理。

1.3 方案组件

该方案包含以下组件:

模型管理器:将模型文件中的信息按一定的格式导入到内存中,以便调用。这些信息一般包括面片、面片顶点、顶点索引、顶点法线、纹理以及纹理坐标等等。该数据一般由 3D 建模软件导出,如 3Dmax, Maya 等等,且有专门的导出插件和工具,如 Paradox 插件可以导出 DX 文件,以被 DirectX 对象识别。将建立 CMeshManager 类实现导入导出模型文件,并对信息管理。

虚拟 BSP 管理器:将建立 CVBSPManager 类,实现虚拟 BSP 的管理和主要算法的集成。它负责根据模型管理器的信息,建立 vBSP 树,并对其管理。

2 方案的实现

要使这个系统能够实际运行起来,需要实现模型管理器和虚拟 BSP 管理器两部分,下面将说明如何分别实现它们。

2.1 模型管理器的实现

模型管理器的实现比较简单,主要依赖于模型文件的格式,一般需要多层递归解析,将其对象的组成面片,以及顶点、顶点索引和法线、纹理和纹理坐标导入到内存当中去。只要程序中启动导入内存命令,并选择文件路径即可。

```
HRESULT CMeshManager::LoadMesh (LPSTR strName)
```

```
HRESULT CMaterial::LoadMaterial (LPDIRECTXFILEDATA pxFileData)
```

第一条语句以参数指定的文件名开始装载模型文件的信息,导入到内存中去。第二条语句则是当 CMeshManager 的对象中设定装载复杂信息时,即 BOOL Additional = TRUE 时,开启 CMeshManager 的成员对象 CMaterial,负责装载法线、纹理和纹理坐标等其他材质信息到内存中去。否则也就是 BOOL Additional = FALSE 时,仅仅导入对象的面片和顶点信息。

2.2 虚拟 BSP 管理器的实现

首先, CVBSPManager::CreateVBSP() 的调用,根据已经导入内存中的面片间的关系,递归建立二叉树进行索引。

这里存在子树的根的选择问题^[1],一般主要使用三种选择方式:(1)随机选择;(2)最大或最小相交面选择;(3)中合相交选择。这三种选择方式可作成类的方

法,根据用户设定,在建立子树时调用,由 SelectSubRoot() 方法实现。

生成 vBSP 树主要由面片间的关系确定,而面片间的关系,主要靠法线和顶点关系确定:顶点可以确定相交关系,法线可以确定方向关系。如果其它面与子树根面法线乘积成正值,说明该面在根面前,方法 NormalDirection() 实现该判断;反之负值,该面在根面的背后。若其它面存在两个顶点到根面的距离乘积成负值,则该面被根面穿过,由方法 Traversed() 实现。

接着就等待树的建立的完成,获得返回的 vBSP 树根的指针。如果建立 vBSP 树成功,就可以调用 CVBSPManager::Render() 方法,渲染某个视角下的场景,并及时返回渲染时间。

2.3 系统的运行过程

系统首先提供导入模型文件的界面供用户选择路径和模型文件名,并提示用户导入信息的对象和面片,还有顶点的个数,提供计算量的初始依据,确定后由模型管理器进行模型文件的内存导入工作。并可打开 45 度角静态摄像机,看到全景,不过此时需要开启 Z Buffer 功能。然后,用户可以启动 vBSP 建立按钮,系统通过虚拟 BSP 管理器,开始生成 vBSP 树信息,可能需要相当长的时间,依赖于场景的复杂度,并有运行状态条提示。此时,用户再打开摄像机,此时关闭 Z Buffer 功能,可以进行观测。系统会自动给出在两种渲染全景的条件下,时间以及渲染速度的加速比。

3 实际应用举例

图 1 是一个简单的三维场景的平面投影, A, B, C, D, E, F 分别表示 6 个平面。接下来描述 vBSP 树的建立过程,在每次迭代计算后,显示每个树的状态。以图 1 为初始多边形列表。根的选择采用自然的随机顺序。

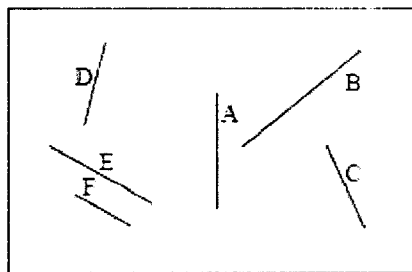


图 1 6 平面投影

3.1 vBSP 树的建立过程

图 2、图 3 显示了迭代的主要步骤^[1,4]:

(1) A 被选择做为场景的根,多边形 B 和 C 的编号被引用放入前列表中,且 D, E 和 F 的编号被引用到后列表中。

(2) 前列子表分解: B 被引用做子树的根且 C 被引用到前列表中, 这个 vBSP 完成它的前半部分。

(3) 后列子表的分解: D 被引用成根。面片 E 被 D 穿越成两部分, 所以 E 被引用两次到 D 的前后子表中去。 F 仅仅在 D 的一侧, 所以仅被引用在后表中。

(4) 最终, D 的后子表被分解, 并且 F 也放在了 D 和 E 的最后方。

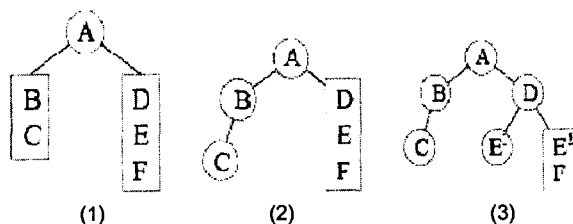


图 2 步骤(3)后的 vBSP 树

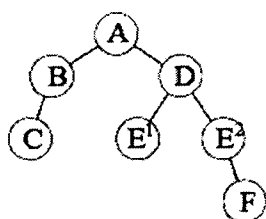


图 3 带有 7 个碎片的 vBSP 树(步骤(4)后)

以下是 vBSP 的构造代码:

```

vBSP_tree vBSP_Make(List_of_Polygons pList)
{
    Polygon root, frontPart, backPart, p;
    List_of_Polygons frontList, backList;
    if (EMPTY(pList)) return NULL;
    else
    {
        root = select_and_remove_poly(&pList); //可以根据用户需要
        设定子树根的三种选择方式
        for (each remaining polygon, p, in pList)
        {
            if (NormalDirection())
                vBSP_add_to_list(p, &frontList);
            else
                vBSP_add_to_list(p, &backList);
            if (Traversed())
            {
                vBSP_split_polygon(p, root,
                    &frontPart, &backPart);
                //由于根面穿过了该面,所以该面被前后引用两次
                vBSP_add_to_list(frontPart, &frontList);
                vBSP_add_to_list(backPart, &backList);
            }
        }
        return vBSP_combine_tree(vBSP_Make(frontList), root, vB-
            SP_Make(backList));
    }
}
    
```

3.2 vBSP 树的遍历

基于 vBSP 引擎的渲染也就是 vBSP 树遍历的过程。给出一个观测点, 就可以得到面片的从后往前的渲染优先级, 即“画家算法”, 描述了画家怎样由远逐近的描述^[5]。

用 vBSP 渲染同 BSP 渲染过程, 采用“修正顺序”, 遍历且检测每个节点。这时开始注意根, 如果那个节点的位置在前面(后面), 就首先检测树的后面(前面), 然后渲染该节点, 再检测树的前面(后面)。当到达一个叶节点时, 就靠该算法迭代的内在本质回溯。当回溯时, 面片开始被渲染。以下是 vBSP 算法描述:

```

CVBSPManager::Render(vBSP_tree vtree)
{
    if (!EMPTY(vtree))
    {
        if (Observer is located on front of root)
        {
            Render(vtree->backChild);
            DWORD FacetIndex = GetIndex(vtree->root); //得到面片索引号
            DisplayFace(FacetIndex); //根据指定的面片索引渲染
            Render(vtree->frontChild);
        }
        else
        {
            Render(vtree->frontChild);
            DWORD FacetIndex = GetIndex(vtree->root); //得到面片索引号
            DisplayFace(FacetIndex); //根据指定的面片索引渲染
            Render(vtree->backChild);
        }
    }
}
    
```

渲染结果演示如图 4 所示。



图 4 渲染结果例子

3.3 效率问题

由于 vBSP 是 BSP 的一种改进实现, 所以可以完全替代 Z Buffer 具有巨大开销的测试环节, 彻底取消

(下转第 164 页)

差不多。服务器端程序应具有对客户端授权以及对数据服务器整个注水管网数据管理的功能,即具有编辑全部注水管网图或者选择编辑某个矿注水管网图的权限。服务器端程序也是借助 WinSock 机制接受和发送图文信息,用 DCOM 机制接受和发送注水系统数据库数据。图 5 是某油田整个注水系统在服务器端显示。

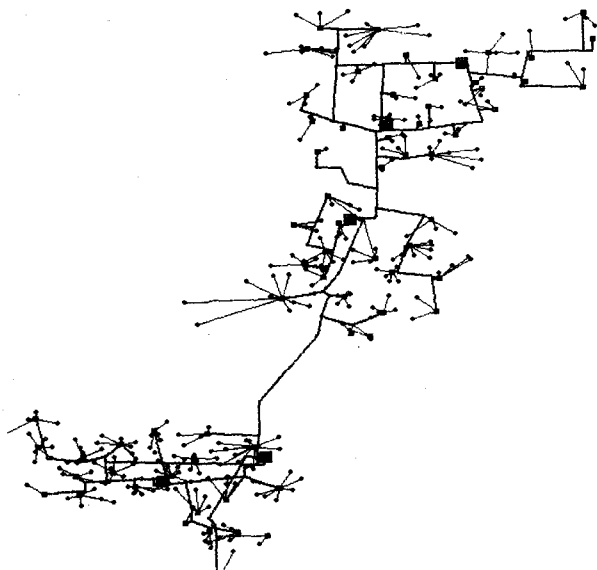


图 5 某油田注水系统

(上接第 121 页)

了 Z Buffer 存在的精度问题^[6,7],大大提高了渲染速度,可以快速正确地展示观测者面前的 3D 图像^[8]。但是,由于尚未实现基于 vBSP 树的隐藏面剔除技术,所以还不能对场景更进一步地加速显示。还需要进一步的工作来实现 vBSP 上的加速和场景的局部划分。

4 小结以及未来工作

文中主要的目的是使用改进的 BSP 技术来开发一种新的简单、实用的 3D 引擎的核心以及这种改进的方法。由于这种方案不破坏模型文件的信息,故它适合基于顶点以及纹理的引用。现在相当多的商业应用 3D 核心是用 BSP 的,我们的方法可以避免复杂的实际应用,如对面的切割等,同时替代了具有巨大开销的 Z Buffer 测试功能。对于建立复杂的建筑体的三维引擎就是一个很好的应用例子,因此它可以直接应用在建筑体的三维引擎中,把从模型文件中的面片信息转换为一个个对顶点以及纹理的应用。

基于这种方法的实现和应用还有很多改进的空间,存在的问题是,在场景中,没有实现对隐藏面的剔除。以后还有很多方面的工作要做:如建立基于 vBSP 的 vPFD 可以根据观测者的位置,渲染最少的面片和

6 结 论

OWFSCM 既要考虑普通网络数据传输,又要考虑 Multi-Tier 结构数据库文件传输,因此,在结构设计上采用 WinSock 和 COM/DCOM 组合。利用 COM/DCOM 机制实现基于网络的三层数据库设计,利用 WinSock 实现实时图文传输。这样能够发挥各自在网络传输和数据安全方面的优点,实现优化配置,从而提高协同建模效率和质量。从实际测试来看,比单纯的 WinSock 连接更容易保证系统的稳定,也比单纯的 DCOM 结构更加灵活。

参考文献:

- [1] 朱时银,马承志,杨 飞,等. C++ Builder 5 编程实例与技巧[M]. 北京:机械工业出版社,2001:577-638.
- [2] 徐新华. C++ Builder 5 高级编程技术—COM、CORBA 与 Internet 编程[M]. 北京:人民邮电出版社,2000:150-169.
- [3] 陈周造,陈灿煌. 精通 C++ Builder 5 程序设计高级教程[M]. 北京:中国青年出版社,2001:471-587.
- [4] 朱保国,常玉连. 油田注水生产过程仿真分布式协同建模系统设计[J]. 系统仿真学报,2004,16(6):1288-1291.
- [5] 李柏林,卢茂华. 协同设计系统中基于角色的访问控制[J]. 西安交通大学学报,2005,40(3):330-333.

对隐藏面的剔除;建立 vMRBSP,可以实现多细节层次的地形方案等等。

参考文献:

- [1] Baciu G, Wingo S K, Apr W. Image-Based Techniques in a Hybrid Collision Detector[J]. IEEE Transactions on Visualization and Computer Graphics, 2003(4):67-69.
- [2] Premoze S, Hansen C. A Model for Volume Lighting and Modeling[J]. IEEE Transactions on Visualization and Computer Graphics, 2003(3):45-49.
- [3] James A. Binary Space Partitioning for Accelerated Hidden Surface Removal and Rendering of Static Environments[J]. ACM SIGGRAPH Computer Graphics, 1999,28(6):1-36.
- [4] 张继开. 三维图形引擎技术的研究[D]. 北京:北方工业大学,2004.
- [5] 左鲁梅. 三维图形引擎中的关键技术研究[D]. 北京:北方工业大学,2004.
- [6] Watt A, Policarpo F. 3D GAMES Animation and Advanced Real-time Rendering[M]. 北京:电子工业出版社,2003.
- [7] Engel W F. Direct3D Game Programming[M]. 北京:电子工业出版社,2004.
- [8] 张勇刚. 基于图形建模的城市道路虚拟视景快速生成研究及应用[D]. 昆明:昆明理工大学,2004.