计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(3):228−232 [doi: 10.15888/j.cnki.csa.006248]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于自适应二叉树的场景视锥体裁剪算法①

牛　鹏[1,2], 廉东本[2], 苏　谟[1,2]

[1](中国科学院大学, 北京 100049)
[2](中国科学院 沈阳计算技术研究所, 沈阳 110168)

摘　要: 建立逼真而丰富的三维场景是可视化领域的主要任务. 场景的数据管理和可见性判断对后续渲染的质量和效率起到了至关重要的作用. 为了弥补传统场景组织方式在实际应用中的缺陷, 本文采用自适应二叉树场景组织算法对场景进行管理, 采用层次化裁剪的方式对场景树的节点进行视锥体裁剪, 在裁剪过程中所操作的对象是节点中的包围球和包围盒. 实验表明, 这种基于包围球和包围盒的层次化的视锥体裁剪算法大大的减少了参与裁剪的节点的数量, 提高了裁剪的精确性, 具有较好的裁剪效率和较高的稳定性.

关键词: 场景的数据管理; 可见性判断; 自适应二叉树; 视锥体裁剪; 包围球; 包围盒

引用格式: 牛鹏,廉东本,苏谟.基于自适应二叉树的场景视锥体裁剪算法.计算机系统应用,2018,27(3):228–232. http://www.c-s-a.org.cn/1003-3254/6248.html

View Frustum Culling Algorithm for Scene Based on Adaptive Binary Tree

NIU Peng[1,2], LIAN Dong-Ben[2], SU Mo[1,2]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China)

Abstract: Building realistic and rich 3D scenes is the main task of visualization. The data management and visibility judgments of the scene play a crucial role in the quality and efficiency of subsequent rendering. In order to make up for the shortcomings of traditional scene organization in practical application, this study adopts the adaptive binary tree scene organization algorithm to manage the scene, and uses the hierarchical cutting method to cut the nodes of the scene tree. In the process of cutting, the object of the operation is the bounding sphere and bounding box in the node. Experiments show that this layered algorithm based on the bounding sphere and bounding box greatly reduces the number of nodes involved in cutting and improves the accuracy of cutting. It has better cutting efficiency and high stability.

Key words: scene data management; visibility judgment; adaptive binary tree; view frustum culling; bounding sphere; bounding box

随着计算机硬件与虚拟现实技术的发展, 应用于场景的模型的数据规模呈指数级增长, 场景的结构也更加复杂. 尽管图形渲染已经在硬件方面得到了很好地支持, 却仍不能解决场景在图形实时渲染方面的问题, 还需对相关的裁剪算法进行分析研究. 所以, 在克服传统场景组织方式弊端的基础上, 能够有效地减少绘制对象, 降低模型复杂度, 提升裁剪和场景渲染的效率, 就成为本文研究所关注的内容.

在三维场景中, 模型作为基本的组成单位, 一般是以点线面的存储结构的, 这种点线面的数据结构适用几何建模, 但不能体现场景中的空间分布情况. 比较有效的方法是对场景进行空间分割. 并用树结构进行组织, 这样可以把空间无序的场景模型变成一棵空间有序的层次树. 对层次树的操作也就等同于对整个场景的操作.

本文提出了采用自适应二叉树空间分割算法负责

① 收稿时间: 2017-06-13; 修改时间: 2017-06-30; 采用时间: 2017-07-12; csa 在线出版时间: 2018-02-09

场景的组织管理, 在可见性判断方面, 采用了一种基于包围球和包围盒的层次化的视锥体裁剪算法, 在充分发挥自适应二叉树空间分割算法的优势的基础上, 该视锥体裁剪算法有效地降低了场景模型的复杂度, 提高了裁剪的精确度, 提升了三维场景的绘制效率.

1 场景组织及视锥体裁剪算法

针对传统二叉树 (BSP)[1]存在的问题, 本文采用自适应二叉树算法 (Adaptive Binary Tree, ABT) 进行场景的组织管理, 其较强的自适应性, 适合剖分任何类型的场景. 与传统的八叉树[2]相比, ABT 提供了“多态”的优点, 在某种意义上是几何模型和空间划分的抽象, 而不限于固定的形状. 它主要用在复杂 3D 场景下的视锥体裁剪. 而八叉树由于其严格的空间结构, 增加了可见性计算的开销. 这可以通过使用松散的八叉树来缓解, 但是仍然有很多限制, 特别是八叉树并不能保证网格数据的唯一性 (这在现代 3D 硬件上是非常重要的), 此外, 八叉树不如自适应二叉树那样平滑地适应于几何体的分割. 基本上, 为了中等或复杂场景的通用渲染, ABT 比八叉树更有效率. 在视锥体裁剪方面, 对场景树中的节点的采用层次化裁剪的方式来剔除模型中冗余的数据, 降低三维模型的复杂度, 提高后续的渲染效率.

1.1 场景组织

1.1.1 自适应二叉树的建立

ABT 算法基于场景的几何体对空间进行分割. 分割平面由一个考虑了多种不同参数 (分割面的数量和位置等) 的评分系统决定, 使得到的分割平面更适合特定类型的场景的模型划分. ABT 建立的具体算法的执行流程如图 1 所示.

在创建过程中, 首先, 将与坐标轴对齐且覆盖整个场景的包围盒作为根节点 (在创建节点的过程中, 每个节点中都要存放一个包围盒和包含该包围盒的一个包围球, 用来进行后续的裁剪和碰撞检测). 然后, 开始执行一个递归剖分函数, 该递归剖分函数的功能就是将当前的包围盒用一个垂直于三个主坐标轴之一的分割面分开, 作为当前节点的两个子节点. 然后分别对两个子节点继续进行细分, 直到节点中的面的数量达到预定阈值或树的深度上限.

1.1.2 最佳分割面的获取准则

求解最佳分割平面是 ABT 创建过程中最重要的部分, 其目的是对当前包围盒中的几何体进行有效地分割, 使得建立的场景树更有利于后续的裁剪和渲染. 在文献[3]描述了求解有效分割面的标准.

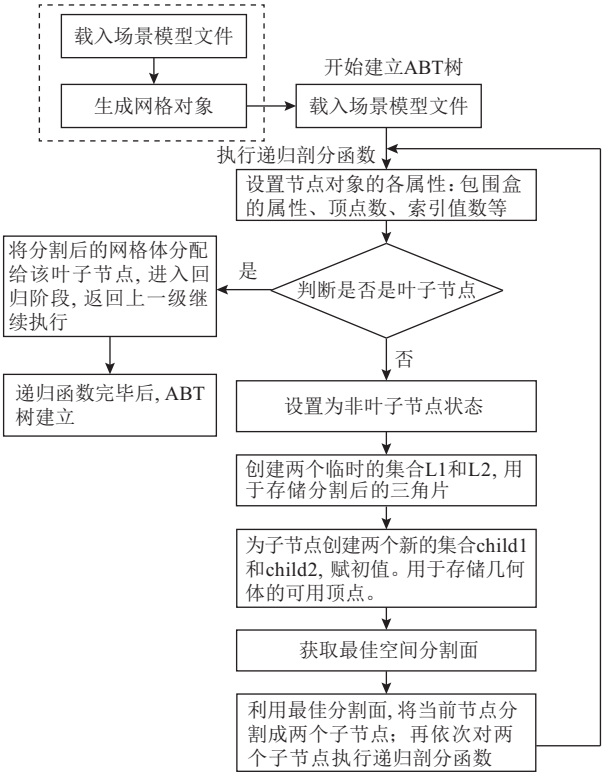

图 1 自适应二叉树的构建流程图

标准 1. 最优空间定位

该标准侧重于分割当前节点包围盒最长坐标轴的分割平面.

$f_1(pos)=\max(split\_x\_axis, split\_y\_axis, split\_z\_axis)$

标准 2. 子节点体积

该标准侧重于将当前节点分割成对等体积子节点的分割平面.

$f_2(pos)=abs(volume\_child1-volume\_child2)$

标准 3. 面的数量

该标准侧重于将当前几何体表面均匀地分布到所有叶子节点上的分割平面.

$f_3(pos)=abs(numfaces\_child1-numfaces\_child)$

标准 4. 被分割的面的数量

计算几何体被分割平面分割的面的数量. 被分割平面分割的面越多, 该分割平面越不适合作为分割面.

$f_4(pos)=total\_number\_of\_split\_faces$

上面所有函数必须最小化, 才能找到最优的剖分平面. 另外, 每个函数都有一个权重因子, 根据场景类型的特点, 灵活的设置权重因子, 权重因子分布取决于场景策略的选择, 从而保证算法更可控、有效. 最终可获取可行平面时必须最小化的方程:

$f(pos)=f_1(pos)\cdot w1+f_2(pos)\cdot w2+f_3(pos)\cdot w3+f_4(pos)\cdot w4$

1.1.3 获取最佳空间分割面的步骤

步骤 1. 采用随机采样方法, 确定用于分割的候选面数据集.

步骤 2. 采用评分函数对候选面数据集中的每个候选面进行评分, 分数最高者则为最佳分割面.

步骤 3. 采用最佳分割面对节点进行分割.

1.2 基于自适应二叉树的视锥体裁剪算法

三维场景规模越来越大, 能够有效地减少绘制对象, 降低模型复杂度, 是三维渲染引擎中实现复杂场景快速绘制的关键所在. 可见性判断是一种很有效的方法, 可见性裁剪算法[4,5]常常分为 3 类: 视锥体裁剪、背面裁剪和遮挡裁剪. 本文主要研究在自适应二叉树的场景组织的基础上采用基于包围球[6]和包围盒[7]的层次化的裁剪方式进行视域剔除. 在既定的场景中, 采用 ABT 算法来组织场景元素, 面片在在叶子节点中分布的更加均匀, 树的深度达到最佳, 形成了一棵高效平衡的二叉树, 基于此优势, 在该数据结构上进行视锥体裁剪, 会大大的提高裁剪的效率.

1.2.1 算法描述

该算法的原理是首先利用 ABT 算法对场景中的对象进行组织, 构建起自适应二叉树, 自适应二叉树中的每个节点都包含了一个包围球和包围盒 (轴对齐包围盒), 然后在视锥裁剪方面, 采用递归的方式对树中的节点进行裁剪处理, 首先, 判断节点的包围球是否在视锥体内, 如果包围球位于视锥体内, 那么就说明该节点分支下的所有子节点均是可见的, 所以可直接送入渲染管道进行绘制. 如果包围球位于视锥体之外, 则说明该节点及分支下的子节点均不可见, 则可以直接停止对该分支上的节点的可见性判断, 从而节省了 CPU 的周期开销. 当包围球与视锥体相交时, 就开始判断视锥体与包围球中的轴对齐包围盒的位置关系, 如果包围盒位于视锥体内, 同样将该节点及其分支节点直接送入渲染管道进行绘制. 如果包围盒位于视锥体外, 则停止对该节点及其分支子节点的可见性判断. 如果包围盒与视锥体相交, 则继续遍历其子节点, 然后对其子节点同样进行基于包围球和包围盒的可见性判断. 该算法的执行流程图如图 2 所示.

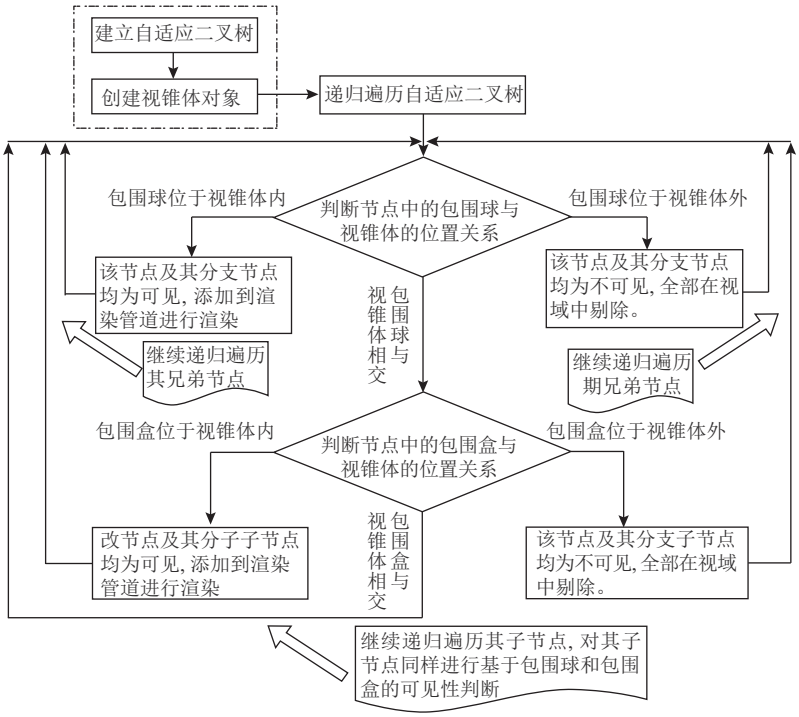

图 2 基于包围球和包围盒的视锥体算法执行流程图

1.2.2 确定包围球、包围盒与视锥体的位置关系

确定包围球是位于视锥体内、视锥体外还是与视锥体相交实际上是一个非常简单的过程, 需要做的是计算球体中心到视锥体的 6 个平面的距离. 如果距离的绝对值小于球体的半径, 那么球体与平面相交. 在相交不成立的情况下, 如果距离大于 0, 那么球体位于平面的前侧 (也可能在视锥体内). 如果小于 0, 那么球体位于视锥体背面, 绝对在视锥体之外. 判断包围球与视

锥体位置关系的流程图如图 3 所示.

确定视锥体与包围盒的位置状态, 用以下方式来完成该操作, 首先, 用三个最小值和三个最大值来定义包围盒, 然后将该包围盒的 8 个顶点与视锥体进行比较. 如果所有的点在视锥体内, 那么说明这个包围盒位于视锥体内, 如果包围盒至少有一个顶点 (但不是全部) 不在视锥体内, 则说明包围盒与视锥体相交. 如果包围盒所有的点都在特定的视锥体平面的背面. 那么这个包围盒位于视锥体的外面, 是不可见的. 考虑到, 视锥体完全包含在包围盒内的情况. 在这种情况下, 这些点都不在视锥体内, 但包围盒仍然被视为可见的. 判定算法的执行流程图如图 4 所示.

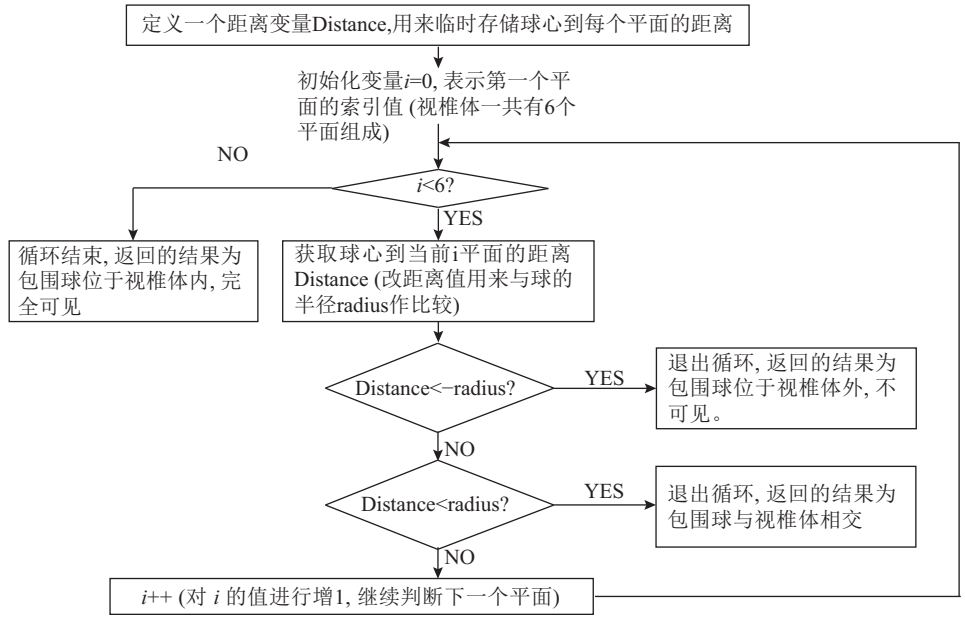

图 3 包围球与视锥体位置关系的判断的流程图

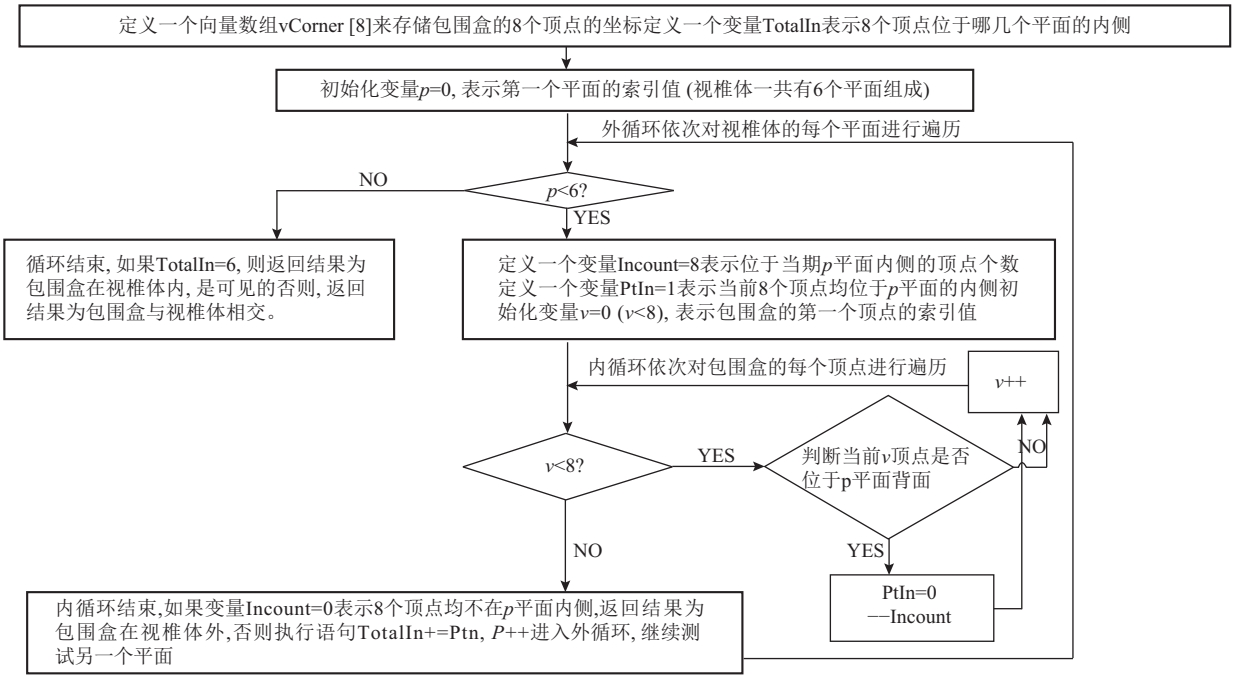

图 4 包围盒与视锥体位置关系的判断的流程图

2 实验结果及分析

本实验的硬件配置: CPU Intel Core(TM)i3-4160 3.6GHz, 四核处理器, 内存为 4 GB, 显卡类型为 NVIDIA GeForce GT610, 显存大小为 1 GB. 软件环境: Windows 7 旗舰版 64 位操作系统, OpenGL4.3 版本, VS2015, NVIDIA 驱动版本 R320, OSG 版本号 3.4. 设置可视化场景生成时屏幕的分辨率为 800 * 600.

本文将自适应二叉树场景组织算法和基于包围球和包围盒的层次化视锥体裁剪算法应用到三维渲染引擎上, 通过实验, 对本文提出的裁剪算法与文献[8]中提出的基于几何投影降维的裁剪算法 (该算法是在二维投影的平面上采用传统的 Cyrus-Beck 算法完成对视锥体的裁剪) 在裁剪的性能上作了测试, 测试结果如表 1 和表 2 所示.

表 1 基于包围球和包围盒的裁剪算法的性能

实验次数	模型多边形总数(个)	划分成节点总数(个)	裁剪所用时间(秒)
第一次	200000	60000	0.0041
第二次	200000	60000	0.0034
第三次	200000	60000	0.0052
第四次	200000	60000	0.0041
第五次	200000	60000	0.0034
第六次	200000	60000	0.0041

注：平均裁剪所用时间为: 0.0041 s.

表 2 基于几何及投影降维的裁剪算法的性能

实验次数	模型多边形总数(个)	划分成节点总数(个)	裁剪所用时间(秒)
第一次	200000	60000	0.0081
第二次	200000	60000	0.0076
第三次	200000	60000	0.0083
第四次	200000	60000	0.0074
第五次	200000	60000	0.0081
第六次	200000	60000	0.0076

注：平均裁剪所用时间为: 0.0079 s.

通过所得数据发现, 两种算法在场景组织方面因为均采用自适应二叉树算法, 所以在模型的多边形总数一致的情况下, 所生成的节点总数是相同的, 对于裁剪, 在时间消耗上, 基于包围球和包围盒的裁剪算法所花费的时间要少于基于几何投影降维的裁剪算法, 特别是在场景复杂的情况下本文提出的裁剪算法的优势会更加突出.

3 结语

本文采用自适应二叉树的场景组织结构来对场景进行管理, 其高效的分割平面的选择机制, 使其具有很强的自适应性, 可以用来划分各种类型的场景. 然后在此场景结构的基础上对可见性裁剪算法进行了研究, 在视锥体裁剪方面, 采用了基于几何包围球和包围盒的视锥体裁剪算法, 该算法充分利用了自适应二叉树的场景结构的优势, 在一定程度上减少了参与裁剪计算的节点的数量, 提高了裁剪的效率, 特别适合于复杂场景中各个模型的裁剪. 接下来将进一步研究如何将本文采用的场景组织算法和裁剪算法高效地运用到三维动态场景中去.

参考文献

1 黄晓康. 基于 BSP 和四叉树的场景管理研究[硕士学位论文]. 南京: 南京理工大学, 2008.

2 张宇. 一种基于八叉树的动态场景管理方式. 电脑知识与技术, 2015, 11(14): 54–57.

3 Dickheiser M. Game programming gems 6. 孟宪武, 译. 北京: 人民邮电出版社, 2007. 339–341.

4 Yagel R, Ray W. Visibility computation for efficient walkthrough of complex environments. Presence: Teleoperators and Virtual Environments, 1996, 5(1): 45–60. [doi: 10.1162/pres.1996.5.1.45]

5 Zhang HS, Manocha D, Hudson T, *et al*. Visibility culling using hierarchical occlusion maps. Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques. New York, NY, USA. 1997. 77–88.

6 王倩, 高保禄, 高锐军, 等. 基于四叉树包围球和屏幕误差的 LOD 算法. 微电子学与计算机, 2016, 33(5): 127–132.

7 彭艳莹, 陆国栋, 李基拓, 等. 基于包围盒编码的三维线段裁剪新算法. 计算机辅助设计与图形学学报, 2003, 15(11): 1369–1374. [doi: 10.3321/j.issn:1003-9775.2003.11.008]

8 余沛文. 视锥体裁剪几何算法与测试方法研究[硕士学位论文]. 上海: 东华大学, 2016.