八叉树 数据结构 三维物体 计算机图形学 27

计算机科学2000Vol.27№.6

99-100

改进的八叉树数据结构

An Improved Data Structure for Octree

TP311.12 TP391.41

王 洵 许胤龙 陈国良

(中国科学技术大学计算机科学技术系 国家高性能计算中心 合肥230027)

Abstract An improved data structure for octrees, which reduces the redundancy of the original octree, is presented. The number of nodes of the improved octree is less than one-eighth of that of the original octree, and the storage requirement is even less. Since the logical structures of both octrees are identical, all the algorithms on the original octree can be adopted on the improved octree and will be faster.

Keywords Data structure, Octree, Computer graphics

1 引言

随着计算机图形学的飞速发展,三维物体的有效表示变得越来越重要,其中,八叉树表示法(octree representation)以其数据结构简单、算法实现方便等特点,成为最广泛使用的三维物体的表示法之一[1,2]。八叉树表示法产生于70年代末、80年代初,然而原有的八叉树数据结构存在着冗余,并且这种冗余已经存在了约20年之久。

本文在介绍原有的八叉树数据结构的基础上,提出了改进的八叉树数据结构,通过删除原有的八叉树的所有叶节点,减少了冗余;并讨论改进的八叉树的节点数目、存储空间、两种八叉树的一致性以及算法在两种八叉树上的运行时间。

2 原有的八叉树数据结构

在文[1,2]中,假设所有需要考虑的物体都存在于一个对象空间(universe)中,对象空间由$2^n\times 2^n\times 2^n$($n>0$)个单位立方体组成。对象空间被划分成8个相同大小的立方体,8个立方体称为8个卦限(octant),并且依次编号为0,1,2,…,7(如图1所示)。

任意卦限与物体之间有三种可能的关系,即卦限与物体完全分离、卦限被物体完全占据或者卦限被物体部分占据。因此,我们给每个卦限一个标号,其值为EMPTY、FULL或者PARTIAL,分别对应于上述三种可能的关系,标号为EMPTY或FULL的卦限不再需要划分,标号为PARTIAL的卦限需要划分为8个子卦限,每个子卦限又有自己的标号,其中,标号为PARTIAL的子卦限还需要进一步划分,直到所有子卦限的标号都为EMPTY或FULL,或者子卦限为单位立方体,无法进一步划分为止。当子卦限为单位立方体时,可以根据它与物体相交的程度,将其标号设置为EMPTY或FULL。图2表示一个物体及其划分过程。

上述物体可以用一棵八叉树来表示。在原有的八叉树数据结构中,每个节点至少包含9个域,即8个子卦限域和1个标号域(如图3所示),八叉树的根节点代表整个对象空间。如果某一卦限的标号为PARTIAL,那么其对应的节点是八叉树的内部节点,child0至child7域分别指向该节点的8个子卦限;如果某一卦限的标号为EMPTY或FULL,那么其对应的节点是八叉树的

参 考 文 献

1 Agrawal R, et al. Concurrency control performance modeling: Alternatives and implications. ACM. Trans. *Database Syst*. 1987, 12(4): 509～654
2 Ramamritham K, Chrysanthis P K. Advances in Concurrency Control and Transaction Processing. IEEE Press, to appear.
3 Carey M J, et al Load Control for Locking. The "Hall and Hall" approach. In the 9th ACM Synposium on the Principles of Database Systems (Nashville, Tenn., May 20－22). ACM, New York. 1990. 72～84
4 卢开澄,卢明华. 图论及其应用(第二版). 清华大学出版社,1995
5 Adler D, et al. A C－based simulation package. [Tech. Rep ECRC－92－27]. ECRC, Munich. 1992
6 Agrawal D, et al. Relative serializability: An approach for relaxing the atomicity of transactions. In, Vianu V, ed. ACM Principles of Database Systems. ACM, New York. 1994. 139～149
7 唐策善,梁维发. 并行图论算法 中国科学技术大学出版社,1991
8 Bernstein P, et al. Concurrency control in a system for distributed database (sdd－1). ACM Trans Darabase Syst. Database Syst., 1980, 5(1): 18～51

叶节点，child0至child7域都为NULL。图4是图2所示物体对应的八叉树表示。

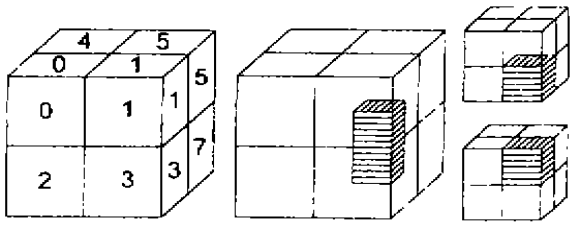

图1 8个卦限及其编号方式(卦限6不可见)

图2 物体及其划分过程

3 改进的八叉树数据结构

图4所示八叉树一共有25个节点，其中，22个节点是叶节点，3个节点是内部节点。我们注意到：所有叶节点的标号都为EMPTY或FULL，并且它们的child0至child7域都为NULL。也就是说，所有叶节点，除了标号可能不同(非EMPTY即FULL)之外，其余部分完全相同。因此，我们不需要保存这些叶节点，只需要在其父节点对应的child域中作标志即可。如果叶节点的标号为EMPTY(或FULL)，那么其父节点对应的child域也标志为EMPTY(或FULL)。这样一来，原有的八叉树的所有叶节点都可以被删除，只留下内部节点构成改进的八叉树。由于原有的八叉树的所有内部节点的标号都为PARTIAL，因此改进的八叉树的节点中就不再需要标号域。图5是图2所示物体的改进的八叉树表示，现在我们只需要3个节点即可。

图3 八叉树的节点

图4 图2所示物体对应的八叉树表示

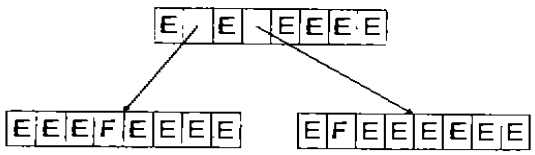

图5 图2所示物体的改进的八叉树表示

4 讨论

由于我们删除了原有的八叉树的所有叶节点，只留下内部节点构成改进的八叉树，即改进的八叉树的节点数目与原有的八叉树的内部节点数目相同，因此改进的八叉树的节点数目与原有的八叉树相比大为减少。

假设原有的八叉树的内部节点数目和叶节点数目分别为m和n。因为只有根节点的入度为0，其余节点的入度都为1，所以八叉树的入度之和为m+n-1。因为内部节点的出度为8，叶节点的出度为0，所以八叉树的出度之和为8m。显然，任何有向图的入度之和与出度之和应该相等，因此有等式m+n-1=8m，即m+n=8m+1。改进的八叉树的节点数目和原有的八叉树的节点数目分别为m和m+n，其比值m/(m+n)=m/(8m+1)<1/8。也就是说，改进的八叉树的节点数目不到原有的八叉树的节点数目的八分之一，考虑到改进的八叉树的节点中不再需要标号域，所以改进的八叉树需要的存储空间更加少于八分之一。

因为我们通过删除原有的八叉树的所有叶节点构成改进的八叉树，因此两种八叉树的逻辑结构完全相同，只不过改进的八叉树的层次比原有的八叉树少了一层。所以我们知道，运行于原有八叉树上的所有算法，如集合运算(并、交、差)、几何变换(平移、缩放、旋转)[1,2]、寻找邻居[3]、八叉树的光线跟踪[4]等，都可以稍加修改后继续运行于改进的八叉树上，并且运行速度会更快。这是因为改进的八叉树的节点数目和层次都比原有的八叉树要少。

参考文献

1 Meagher D. Geometric Modeling Using Octree Encoding. Computer Graphics and Image Processing, 1982, 19: 129~147

2 Jackins C L, Tanimoto S L. Oct-trees and Their Use in Representing Three-Dimensional Objects. Computer Graphics and Image Processing 1980, 14: 249~270

3 Samet H. Neighbor Finding in Images Represented by Octrees. Computer Vision, Graphics, and Image Processing, 1989, 46: 367~386

4 Samet H. Implementing Ray Tracing With Octrees and Neighbor Finding. Computers & Graphics, 1989, 13(4)