DOI:10.13203/j.whugis2004.12.011
第29卷第12期 武汉大学学报·信息科学版 Vol.29 No.12
2004年12月 Geomatics and Information Science of Wuhan University Dec. 2004

文章编号:1671-8860(2004)12-1080-04 文献标识码:A

地形可视化中快速视区裁剪算法研究

许妙忠[1] 李德仁[1]

(1 武汉大学测绘遥感信息工程国家重点实验室,武汉市珞喻路129号,430079)

摘 要:讨论了地形可视化中的视区裁剪技术,分析了多种可能的改进方法,并实现了一种基于层次包围球的快速视区裁剪算法。该算法可以十分快速和准确地判断地形渲染过程中节点和视区的相互位置关系,大大减轻图形系统的负担,有效地提高地形渲染的速度。

关键词:地形可视化;四叉树结构;视区裁剪

中图法分类号:P237.9

1 视区裁剪基本原理

众所周知,在进行视区裁剪之前要进行坐标转换。坐标转换包括两个部分:① 物体坐标系到世界坐标系的转换;② 世界坐标系到视点坐标系的转换。第二个转换取决于视点参数。

在计算机图形学中,不同的投影方式对应不同的视图体(viewing frustum)。在三维显示中,一般采用透视投影和正交投影两种投影模式。

不管是透视投影还是正交投影,其视图体都有6个裁剪面,分别为近裁剪平面、远裁剪平面、左裁剪平面、右裁剪平面、上裁剪平面和下裁剪平面。视区裁剪就是利用这6个裁剪平面判断物体和视图体之间的关系,超出视图体外的物体或物体的一部分将会被裁减掉而不会出现在最终的显示图像中。裁剪的方式有以下两种。

1)点的裁剪。如果已知6个裁剪平面的方程:$A_ix+B_iy+C_iz+D_i=0$($i=1,\cdots,6$)(这里假设每一个裁剪平面的法向量都指向视图体内部),计算点 P 和每一个裁剪平面的关系,$d_i=A_ix_p+B_iy_p+C_iz_p+D_i$。如果对于所有 d_i,只要有一个不大于0,则点 $P(x_p,y_p,z_p)$ 在视图体外,否则,点 P 在视图体内。

2)物体的裁剪。每个物体都是由多边形构成的,假如它有 N 个边界顶点,那么对其裁剪就必须按点的裁剪方式计算 N 个顶点中每一个顶点的6个距离来判断该物体在视图体内,还是在视图体外,还是和视图体相交。这样做很费时,在计算机图形学中,常采用物体包围盒的方式。包围盒是指用一个顶点确定大小足够的多面体来表示该物体,一般用一个长方体或球体来包围该物体。利用长方体作为包围盒,在裁剪时只要判断长方体的8个顶点和视图体的关系。如果它们全在视图体内,则该物体在视图体内;如果这8个顶点都不在视图体里,则该物体在视图体外,否则该物体和视图体相交。利用球体作为包围盒计算球体的中心和6个裁剪平面的距离时,如果这些距离都大于球体的半径,当这些距离都大于0时,则该物体在视图体内,否则,该物体在视图体外。如果这些距离不都大于球体的半径,则该物体和视图体相交。

2 基于四叉树层次结构的视区裁剪

在实时显示过程中,视点参数(视点位置、观察方向、视区中心位置等)都在不断变化中,这就要求视区裁剪也不断地进行,当视点参数变化一次,视区裁剪也要进行一次,所以视区裁剪的效率显得十分重要,它直接影响到整个绘制算法的效率。但是上述讨论裁剪方式的效率太低,其原因是每一个点或物体都必须进行测试,以确定哪些点或物体是否在视图体内。

基于层次结构的区域裁剪能解决上述问题。当采用这样的层次结构表示地形时,一旦高一级

收稿日期:2004-09-28。

的节点不是处于视图体内而被裁剪掉时, 就没有必要再去处理其子节点。这意味着在进行视区裁剪时, 采用层次裁剪算法并不需要对所有三角形进行判断, 这样大大减少了裁剪判断的次数, 加快了视区裁剪的速度。

图 1 是一个大小为 8× 8 的三维地形块在二维平面上的投影图, 共有 64 个地形网格。观察者在地形块左侧, 可视范围用黑色块表示。在这种情况下, 最后绘制出来的地形是虚线表示的地方,

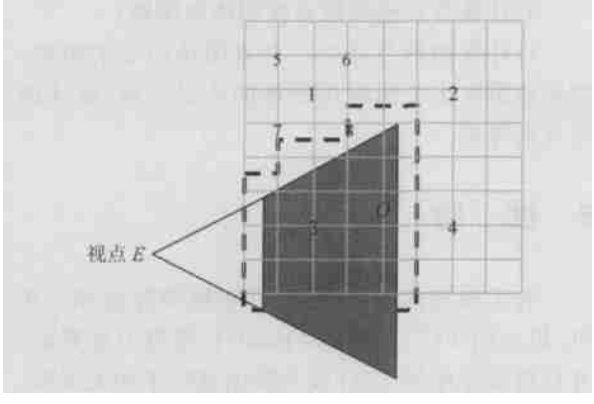

图 1　四叉树裁剪

Fig. 1　Quad-tree Frustum Culling

一共是 26 个网格, 即超过一半的网格被裁剪掉了, 但这对最后绘制的结果没有影响。如果采用一般的裁剪方式, 必须对所有 64 个网格进行投影变换和裁剪测试, 效率很低。

采用四叉树层次结构的方法如下。从根节点 O 开始, 先测试它是否可见, 如果可见, 则把节点 O 分成 4 个子节点 1、2、3、4; 再从节点 1 开始, 顺序测试节点 2、3 和 4; 一直递归, 直至结束。以节点 1 为例, 因为它可视, 则把它分成节点 5、6、7 和 8; 因为节点 5 和节点 6 不可视, 不再对它们的子节点进行测试。而节点 7 和节点 8 可视, 则必须进一步分解。

3　算法的改进和优化

在地形可视化中, 地形节点裁剪都是实时进行的, 即当视点数发生变化时, 都要首先提取当前的视图参数, 然后再进行视区裁剪。为了加快视区裁剪的速度, 增加帧速, 笔者认为原则上可以采用如下一些措施和方法来优化视区裁剪算法。

1)优先估算屏幕半径, 删除一部分地形节点。在对一个四叉树地形节点进行裁剪测试之前, 应先对它进行屏幕半径的估算。如果该地形块的屏幕半径小于一个像素, 可以认为该块对最后地形显示没有贡献, 即认为该地形块在视图体之外。当然, 这种屏幕半径的估算方法只对离视点较远的地形块有效。

屏幕半径的估算可用下式计算:

$$r = \frac{wR}{2d\tan(\theta/2)}$$

式中, r 为地形块的屏幕半径(以像素为单位); w 为屏幕宽度; R 为地形块包围盒的对角线长度; d 为视点到地形块中心的距离; θ 为视图体的水平方向上的视场角。

2)优先测试视点位置。如果视点处于所要进行裁剪测试的包围盒里, 不对它进行裁剪测试, 因为在这种情况下, 该节点肯定和视图体相交。

3)改进判断的方法。在四叉树的情况下, 裁剪测试从根节点开始, 首先测试它是否可见, 如果是, 则进一步测试其子节点, 否则结束测试。这是一个对每一个节点包围盒进行的递归过程。一旦测试结束, 就对所有可见的节点进行渲染。在整个测试过程中, 如果测试到一个节点可见时, 进一步判断该节点是完全可视还是部分可视, 如果是部分可视, 则进一步测试其子节点, 否则结束测试, 因为其所有的子节点肯定也是完全可视的。

4)减少视图体的裁剪平面。这种方式是放弃对视图体的严格判断, 不考虑视图体全部裁剪面, 比如不对近裁剪平面和上下裁剪平面进行测试。因为地形显示时, 在视点比较低的情况下, 大部分节点都在视图体上下裁剪平面的包含范围内, 而如果视点增高, 动态生成的网格复杂度本身就会降低, 所以此时即使不对上下裁剪平面进行判断, 要显示总的三角形数目也不会太多, 而视图体离视点很近, 也可以不考虑。利用这种方式进行裁剪计算, 对于每一个地形节点, 可以减少一半的计算量。

4　基于层次包围球的快速视区裁剪算法

在地形显示过程中, 由于要进行裁剪处理的顶点和三角形很多, 这对图形处理系统来说是一个很大的负担。采用层次包围盒裁剪技术, 虽然能省去很多不必要的计算, 但依然有较多重复计算, 一个大的包围盒通常要多次细分才能得到比较精确的裁剪结果。

事实上, 对于当前的视图体, 一个顶点必须同这个视图体的所有 6 个裁剪平面进行测试。这样, 对于地形四叉树中的每一个节点, 有 8 个顶点

必须进行测试，很显然，当每一个视图体中包含大量顶点时，测试效率就低下来了。

很明显，采用包围球取代包围盒进行裁剪计算能减少测试的次数。利用包围盒时，在裁剪时，要判断四棱柱的 8 个顶点和视图体的关系。而利用包围球时，只要计算球体的中心和 6 个裁剪平面的距离。

再进一步考虑，如果把当前视图体也建立其包围球，在这种情况下，视区裁剪测试就更加简单有效，因为要判断两个球体是否相交只要进行一次测试(计算两个球体中心之间的距离，如果这个距离小于两个球体的半径之和，则它们相交，否则不相交)。

4.1 地形节点包围球的建立

地形节点包围球是在地形数据的预处理阶段建立的。对于待显示的整个地形块，从底到顶地建立每一个节点包围球。

包围球的中心坐标为：

$$x_c=\frac{\sum_{i=1}^{4}x_i}{4},\quad y_c=\frac{\sum_{i=1}^{4}y_i}{4},\quad z_c=\frac{\sum_{i=1}^{4}z_i}{8}$$

式中，x_i、y_i、z_i ($i=1, 2, 3, 4$)是地形节点 4 个角点的坐标值。包围球的半径取地形节点四棱柱对角线距离的最大值的 1/2。

这里要注意的是，和节点的误差传递一样，节点包围球的半径也必须传递，父节点的半径要取它本身的半径和其子节点半径的最大值。

4.2 视图体包围球的建立

在图 2 中，$P_1P_2Q_1Q_2$ 为视图体在 XZ 平面上的投影，P_1P_2 为近裁剪平面，Q_1Q_2 为远裁剪平面。以 C 为中心、CQ_1 为半径作圆即是视图体的包围球。在以视点 E 为原点、视点方向为 Z 方向的视点坐标系中，视图体包围球的中心坐标为(0, 0, (F_1+F_2)/2)；Q_1 的坐标为(F_2 *tan(Fov.2) *r, 2 *tan(Fov.2), F_2)（r 为视区的宽高比)，则视图体包围球的半径可以用C和Q_1两点的坐标计算出来。

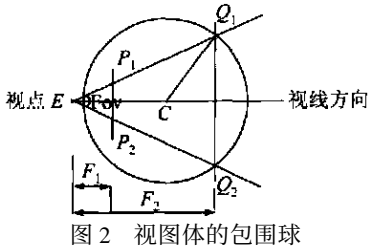

图 2 视图体的包围球

Fig. 2 Frustum Bounding Sphere3

4.3 算法步骤

当视点和视线方向改变后，首先确定视图体包围球，计算它的中心点的坐标和包围球的半径。

对一个地形节点，其算法步骤如下。

1)首先判断视点中心是否处于该节点，如果是，则该点可视；

2)获取当前视图体的参数；

3)计算节点包围球和视图体包围球；

4)计算地形节点中心和视图体中心的距离，如果该距离小于两者包围球的半径之和，则该地形节点可视。

5 试 验

为了检测本算法的实用性和实际效果，在 PC 机上用 VC^{++} 6.0 和 OpenGL 实现上述算法，并利用实际地形 RSG 模型数据进行了相关试验。试验所用硬件设备的主要配置为：P4, 1.7G, 256M 内存，GeForce 2 显卡，操作系统为 Windows 2000。屏幕分辨率为 1 024×768。

表 1 不同地形区域裁剪前后节点数目和三角形数目比较

Tab. 1 Numbers of Nodes and Triangles with & Without Frustum Culling

地形大小	视场角	裁剪前		裁剪后	
		节点数	三角形数	节点数	三角形数
1 025×1 025	60	4 576	34 331	2 814	21 176
	90	4 576	34 331	3 256	24 490
	120	4 576	34 331	3 543	26 657
2 049×2 049	60	10 510	78 915	5 522	40 781
	90	10 510	78 915	6 435	48 352
	120	10 510	78 915	7 549	56 700
4 097×4 097	60	14 970	141 789	8 210	63 640
	90	14 970	141 789	9 300	75 234
	120	14 970	141 789	9 708	77 490

测试过程中，采用透视投影方式，视点位置为 0, 3 000, 6 000；视角为 90，视区中心为每一个地形的中心；视图体的近平面离视点的位置为 100，远平面为 1 000 000。

图 3 是视区裁剪效果图。表 1 给出了 3 种地形数据在不同的视场角下最后绘制四叉树节点数目和三角形数目的大小。表 2 给出了对 3 种地形数据进行实时渲染帧速的比较。

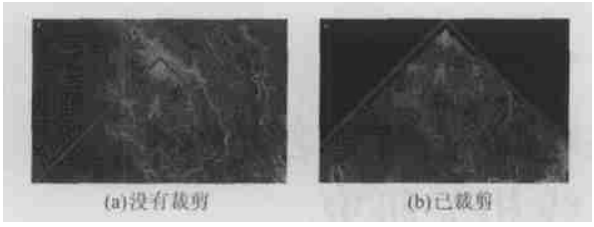

图 3　地形可视化中的视区裁剪效果图

Fig. 3　Result of Frustum Culling

表 **2**　不同地形区域裁剪前后帧速比较

Tab. 2　FPS with &Without Frustum Culling

地形大小	视场角	FPT(裁剪前)	FPT(裁剪后)
	60	37	51
1 025× 1 025	90	37	46
	120	37	46
	60	18	33
2 049× 2 049	90	18	26
	120	18	25
	60	01	17
4 097× 4 097	90	01	14
	120	01	11

从试验结果可以看出, 经视区裁剪后, 最后所绘制的四叉树节点数目和三角形数目为只有原先数目的 70%～50%, 帧速比原先提高了 1/4～2 倍, 而且随着地形区域的增大, 效果越来越明显。

试验表明, 使用本文所提出的基于层次包围球的视区裁剪算法, 可以准确判断各节点与视图体的相互位置关系, 避免了大量无谓的节点被分解合并与误差计算, 可以明显地提高实时渲染的效率。

参　考　文　献

1　彭群生, 鲍虎军, 金小刚. 计算机真实感图形的算法基础. 北京: 科学出版社, 1999

2　王毅刚. 基于可见性预处理和细节简化的虚拟环境快速漫游算法. 计算机学报, 1998, 21(9): 787～792

3　Sudarsky O, Gotsman G. Dynamic Scene Occlusion Culling. IEEE Transaction on Visualization and Computer Graphics, 1999, 5(1): 13～29

4　Danie G, Lastra A A. Automatic Image Placement to Provide a Guaranteed Frame Rate. ACM SIGGRAPH' 99, Los Angeles, California, 1999

第一作者简介: 许妙忠, 副教授, 博士, 主要从事摄影测量方面的教学与研究工作。

E-mail: miaox@mail.lies mars.wtusm.edu.cn

A Fast Culling Algorithm for Visualization of Terrain

XU Miaozhong[1]　*LI Deren*[1]

(1　State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, 129 Luoyu Road, Wuhan 430079, China)

Abstract: This paper discusses a clipping method of using quadtree in terrain visualization and proposes some improvement techniques. The approach is based on the bounding sphere to culls large chunks of triangles. The algorithm is easy to implement and fast enough to achieve real-time culling in terrain visualization. Some examples illustrates the power of the approach.

Key words: terrain visualization; quadtree; view-frustum culling

About the first author: XU Miaozhong, associate professor, Ph. D, majors in photogrammetric positioning.

E-mail: miaox@mail.liesmars wtusm.edu.cn

(责任编辑:　晓晨)