ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.6, June 2009, pp.1685−1693
doi: 10.3724/SP.J.1001.2009.03481
© by *Institute of Software, the Chinese Academy of Sciences*. All rights reserved.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于概率计算模型改进的相关性层次遮挡裁剪算法*

梁晓辉+, 任 威, 于 卓, 梁爱民

(北京航空航天大学 计算机学院 虚拟现实技术与系统国家重点实验室,北京 100191)

Improved Coherent Hierarchical Culling Algorithm Based on Probability Computing Model

LIANG Xiao-Hui+, REN Wei, YU Zhuo, LIANG Ai-Min

(State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, BeiHang University, Beijing 100191, China)

+ Corresponding author: E-mail: lxh@vrlab.buaa.edu.cn

Liang XH, Ren W, Yu Z, Liang AM. Improved coherent hierarchical culling algorithm based on probability computing model. *Journal of Software*, 2009,20(6):1685−1693. http://www.jos.org.cn/1000-9825/3481.htm

Abstract: The efficient visibility culling method is one of the important research aspects of real time rendering of complexle dynamic scene. This paper researches the visibility culling problem and improves the coherent hierarchical culling (CHC) algorithm. To handle the redundancy and unnecessary visibility culling of CHC, a probability computing model is presented. The model first calculates the expectation of occlusion query time and rendering time, then compares them to improve the query strategy of CHC algorithm. Experimental results show that with the model high culling efficiency can be achieved in the dynamic scene with highly complex depth and large amount of objects, and also achieve real-time rendering results.

Key words: complexly dynamic scene; real time rendering; visibility culling; coherent hierarchical culling; probability computing model

摘 要: 对复杂动态场景进行高效的可见性裁剪是实时绘制领域研究中的一个重要问题.围绕该问题开展工作,并针对相关性遮挡裁剪算法中的问题进行了改进.针对相关性层次遮挡裁剪算法存在冗余和不必要遮挡查询的问题,给出了一种概率计算模型.通过比较遮挡查询时间开销与绘制时间开销的数学期望,改进了相关性遮挡裁剪算法中遮挡查询的查询策略,从而进一步缩小了查询集合,使遮挡查询更加合理.实验结果表明,该算法对深度复杂度高、面片数量大的复杂动态场景有较好的裁剪效率,能够很好地满足实时绘制的要求.

关键词: 复杂动态场景;实时绘制;可见性裁剪;相关性遮挡裁剪;概率计算模型

中图法分类号: TP391 **文献标识码**: A

随着计算机硬件与三维图形处理技术的不断发展,图形数据获取能力越来越强,需要处理的模型数据量越来越大,场景也越来越复杂.虽然图形绘制已经在很大程度上得到了硬件的支持,但仍不能很好地满足复杂场景

* Supported by the National Natural Science Foundation of China under Grant Nos.60873159, 60533070 (国家自然科学基金); the Trans-Century Training Programme Foundation for the Talents by the Ministry of Education of China (教育部跨世纪优秀人才培养计划); the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z333 (国家高技术研究发展计划(863))
Received 2008-03-24; Revised 2008-08-07; Accepted 2008-10-17

© 中国科学院软件研究所 http://www.jos.org.cn

实时绘制的要求,因此,还需要对各种加速技术进行研究[1−3].在这些技术中,可见性裁剪是一种非常重要的加速方法,它选取并绘制场景中可见的对象,从而有效地降低场景绘制的复杂度.可见性裁剪通常包括背面裁剪、视锥体裁剪、遮挡裁剪、细节裁剪等方面,目前在对可见性裁剪算法的研究中,主要以遮挡裁剪为主.

遮挡裁剪的主要任务是在将场景中对象有效组织的基础上,通过查询和判断策略,确定这些对象之间的遮挡关系,从而快速剔除场景中不可见的对象,使得需要进一步处理的对象尽可能地少.其核心内容就是对象遮挡关系的计算.当场景规模很大时,这个计算的开销会严重影响实时绘制的效率,而将更多的计算转移到硬件上是计算机图形处理领域的重要趋势.近年来,图形硬件生成厂商也提供了遮挡查询的功能以辅助遮挡裁剪的计算.因此,在克服传统场景组织方式局限性的基础上,如何更好地利用硬件提供的遮挡查询,以提高遮挡裁剪与复杂场景实时绘制的效率,就成为本文研究的基本出发点.

目前,硬件提供遮挡查询的通常过程是采用包围体代替复杂的几何体,并将其发送给 GPU,包围体被光栅化后,将其片元与 Z-Buffer 中的值进行比较,并由 GPU 返回可见像素的数目,如果比较后没有像素小于 Z-Buffer 的值,则对象不可见,不需要进行渲染.硬件遮挡查询简化了场景中物体之间的遮挡关系的确定,但它也存在一定的时间开销,即调用遮挡查询本身的开销(因为每次查询都增加了一次另外的绘制调用)和等待查询结果引起的开销.

在遮挡裁剪的研究中,Hillesaland等人[4]首先提出了一种基于NVIDIA硬件遮挡查询的遮挡裁剪算法,将场景划分成多个单元,并按照垂直视线的方向从前至后进行遍历,每次执行同一平面上所有单元的遮挡查询.MeiBner等人[5]采用一种层次组织方式来进行遮挡查询.Bittner等人[6]提出了较为经典的相关性层次优化的遮挡裁剪算法(coherent hierarchical culling,简称CHC),利用层次相关性的方式,优化了遮挡查询的使用,用于解决前面所述的遮挡查询调用中的两个问题.Kovalcik等人[7]通过统计前几帧的查询结果,对可能被遮挡的部分不进行查询,从而减少了调用硬件的总遮挡查询次数.这些方法基本上是从通过场景组织或前后帧间的相关性筛选出需要进行遮挡查询的对象,从而提高裁剪效率的角度而开展工作的.但对于如何能够进一步减少冗余及不必要的遮挡查询这一问题,并未深入进行探讨和分析.

本文重点针对如何减少冗余及不必要的遮挡查询这一问题进行了研究,并且在 CHC 算法的基础上提出了基于概率计算模型的相关性层次遮挡裁剪算法.为了更好地解决遮挡裁剪中的问题,提高裁剪算法的效率,本文首先对传统场景组织结构进行了研究,采用一种基于松散八叉树严格空间划分的复杂场景组织方法,在保留空间分割优点的同时,避免了传统八叉树场景组织方式的局限性,减少了场景结构更新的开销.在此基础上,提出了基于概率计算模型的方法,通过比较遮挡查询时间开销与绘制时间开销的数学期望,改进了 CHC 算法中遮挡裁剪中的查询策略,从而使得遮挡查询和裁剪更加合理,在减少遮挡查询数量的同时提高了场景的绘制效率.

1 基于概率计算模型改进的遮挡裁剪算法

本文给出的遮挡裁剪算法主要包括数据重组、优化计算等阶段,首先将复杂场景的原始数据进行松散八叉树场景组织,在此基础上,为了降低场景绘制的复杂度,通过优化计算裁剪不可见的对象,将可见对象发送到图形管线,如图 1 所示.

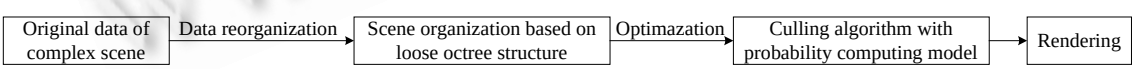


Fig.1 Main process of the visibility culling algorithm

图 1 本文算法的主要过程

图 1 中关于松散八叉树场景组织采用文献[8]中提出的方法.松散八叉树通过对传统八叉树节点进行松散处理,可以减少穿过分割平面的对象数量,能够使较小的对象处于树形结构的更深层,可以更为充分地利用空间数据结构的优势,并更好地支持对动态对象的更新操作,从而可以提供更精确的遮挡查询集合.本文第 2 节的实验 1 将给出松散八叉树场景组织的实验结果.下面重点介绍本文的核心算法,即基于概率计算模型的遮挡裁剪

© 中国科学院软件研究所 http://www.jos.org.cn

算法.

1.1 基于概率计算模型的遮挡裁剪算法

为了在利用硬件加速的同时更好地完成遮挡裁剪,Bittner基于相关性的思想,借助硬件遮挡查询的方式,提出了比较经典的CHC算法[6].该算法利用相关性的思想,假设前几帧可见的节点下一帧仍然是可见的,从而将遮挡查询的集合由场景树中的所有节点缩小至Terminate节点集合,如图 2 所示.在CHC算法中,Terminate节点集合中的可见性状态有两种:可见叶子节点与不可见内部节点.可见叶子节点其内部的对象均可见,不可见的内部节点内部的对象均不可见.CHC算法仅对Terminate集合内可见的叶子节点和不可见的内部节点进行遮挡查询.

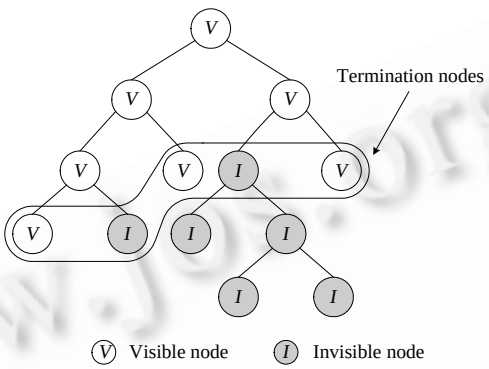


Fig.2 Occlusion queries set of CHC algorithm

图 2 CHC 算法遮挡查询集合示意图

但是,这种查询方式存在冗余及不必要的遮挡查询两个问题,具体情况如下:

1) Terminate 集合内所有可见节点仍然要进行冗余的遮挡查询判断,因此有很多遮挡查询被浪费了,它并没有减少需要绘制对象的数目,反而增加了绘制的开销.

2) CHC 算法并没有分析每个节点的具体情况,如果绘制开销本身就小于遮挡查询的开销,则无论当前节点是否可见,均不需要提出遮挡查询.

针对上述问题,本文提出了一个概率计算模型,通过比较遮挡查询时间开销与绘制时间开销的数学期望,改进了 CHC 算法中遮挡查询的查询策略,减少了遮挡查询的数量,使遮挡查询更加合理,从而提高了场景的绘制效率.

1.1.1 概率计算模型

实际上,对绘制时间开销小于遮挡查询时间开销的节点进行遮挡查询是没有意义的,因此在进行遮挡查询之前,如果判断出当前节点的绘制时间开销比较小,则无论可见与否都可以直接进行绘制,从而提高效率.这一思路的核心在于,需要对节点的绘制时间与查询时间的开销进行估计.

我们可以用节点内部对象的状态来代表节点的状态,而对于场景中的对象,也具有可见或者不可见两种状态.设场景中对象用O_i表示,其状态函数$S(O_i)$的值为可见(用V表示)和不可见(用$\overline{V}$表示),即

$$S(O_i)=\{V,\overline{V}\} \tag{1}$$

设对象O_i可见的概率为$P_i(V)$,不可见的概率为 $P_i(\overline{V})$ (或 $1-P_i(V)$).另外,假设对象O_i直接绘制的时间开销为C_r,对象O_i进行遮挡查询的时间开销为C_q,相应的对象绘制方法也可以有直接的绘制方式(A_1)和先进行遮挡查询,然后根据查询的结果确定是否需要对对象绘制方式(A_2).对于A_1方式,其开销为C_r;对于A_2方式,当对象可见时,整个时间开销为C_q+C_r,当对象不可见时,由于不进行绘制,整个时间为开销C_q.对象在不同状态下采用不同方式的时间开销,见表 1.

© 中国科学院软件研究所 http://www.jos.org.cn

Table 1 Time consuming

表 1 时间开销

Object state	A_1: Render directly	A_2: Render after queries
Visibility	C_r	C_q+C_r
Invisibility	C_r	C_q

对象不同状态下时间开销的数学期望为

$$E(A_1)=P_i(V)\times C_r+P_i(\overline{V})\times C_r=C_r \tag{2}$$

$$E(A_2)=P_i(V)\times(C_r+C_q)+P_i(\overline{V})\times C_q=P_i(V)\times C_r+C_q \tag{3}$$

如在进行遮挡查询之前对不同状态的开销进行计算,即在Terminate集合中仅选择$E(A_1)>E(A_2)$的节点进行遮挡查询,可以进一步减少遮挡查询的数量,从而使遮挡查询更为合理.

但是开销时间的计算不能过于复杂,否则会对绘制造成很大的影响,使大部分绘制时间用于开销的计算上.因此,需要进一步寻找一种快速的方法来确定不同事件的时间开销,即需要快速地确定$E(A_1)$和$E(A_2)$中的 3 个参数绘制时间开销C_r、遮挡查询时间开销C_q和可见概率$P_i(V)$.为了减少这 3 个参数的计算复杂度,下面进一步给出了对这 3 个参数进行估计的方法(不失一般性,后文中对于开销的估计仍然用上述 3 个符号表示).

1.1.2 绘制时间开销估计

当前的图形绘制管线大致可以分为 3 个阶段:顶点处理阶段、三角形化阶段、像素处理阶段.时间开销主要以顶点处理阶段与像素处理阶段为主.根据图形管线结构的特点,对象绘制的时间开销是由图形管线的瓶颈,即图形管线中最慢的阶段决定的.如果能够确定这一瓶颈位置,同时知道所有数据通过该位置需要花费的时间,就可以计算出绘制速度[8].因此有,

$$t_r(O_i)=\max(t_r^v(O_i),t_r^s(O_i),t_r^f(O_i)) \tag{4}$$

其中,$t_r(O_i)$为绘制时间;$t_r^v(O_i)$是顶点处理阶段的时间开销,主要与对象的多边形数量$N(O_i)$相关;$t_r^s(O_i)$为三角化阶段的时间开销,通常可以忽略不计;$t_r^f(O_i)$为像素处理阶段的时间开销,主要与处理的像素片元的数量$f(O_i)$相关.假设T_{tri}为处理每个三角形的时间,T_{pix}为处理每个片元的时间,由公式(4)可知,绘制时间开销C_r的估计为

$$C_r\approx\max(t_r^v(O_i),t_r^f(O_i))=\max(N(O_i)\times T_{tri},f(O_i)\times T_{pix}) \tag{5}$$

由于现在的显卡都配置了光栅化引擎,对于一个复杂的对象而言,像素处理部分的时间与顶点处理部分相比一般较小,且遮挡查询可以在光栅化阶段裁剪大部分不可见的像素片元.为了进行快速估计,绘制时间开销可以进一步近似为

$$C_r\approx N(O_i)\times T_{tri} \tag{6}$$

1.1.3 遮挡查询开销估计

遮挡查询是将对象的包围盒送入图形管线,因此在顶点处理阶段,只需要对包围盒的 8 个顶点进行处理即可,时间开销比较小.因此,图形管线的绘制时间以像素处理的时间为主.

为了估计对象的遮挡查询时间开销,需要计算包围体在屏幕上的投影像素数以及每个像素的处理时间和管线的延迟时间.

因为包围体投影面积之比等于视锥体棱长之比的平方(如图 3 所示),即

$$\frac{a}{a'}=\frac{l}{l'},\frac{b}{b'}=\frac{l}{l'} \tag{7}$$

$$\frac{A_{bb}(O_i)/6}{S}=\left(\frac{Dis}{h}\times 2\tan\left(\frac{\theta}{2}\right)\right)^2 \tag{8}$$

所以

$$S=(h^2\times A_{bb}(O_i)/6)\Big/\left(2\times Dis\times\tan\left(\frac{\theta}{2}\right)\right)^2 \tag{9}$$

可以得到对象遮挡查询的开销估计C_q为

$$C_q=T+N_q\times T_q=T+S\times\rho\times T_q \tag{10}$$

© 中国科学院软件研究所 http://www.jos.org.cn

其中,T为遮挡查询的时间延迟,N_q为对象包围盒投影到屏幕上的像素数,T_q为对每个像素进行遮挡判断所需要的时间,Dis为对象中心到视点的距离,θ为视场角,h为视口垂直长度,ρ为单位面积的像素个数,$A_{bb}(O_i)$为包围盒的表面积.

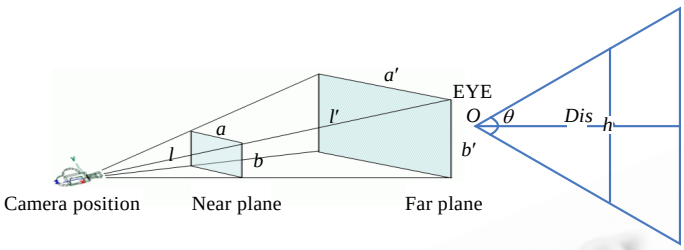


Fig.3 Illustration of view frustum and its projection

图 3 视锥体与其平面投影示意图

1.1.4 可见概率估计

在实时绘制系统中,相关性是客观存在的,对象可见性的状态是与特定帧相关的,尤其是在动态的场景中.当视点进行连续移动时,帧与帧之间的可见性状态变化是很小的,对于前一帧可见的对象,其在后一帧可见的概率是很大的,Bittner仅仅简单地利用了相关性的思想,认为上一帧可见的对象在下一帧一定是可见的,从而将遮挡查询集合限制在Terminate节点集合中,但他并没有对可见性状态概率进行深入研究.本文在利用相关性的基础上,将相关性限制在两帧之间,利用上一帧对象的可见性概率来估计下一帧可见性概率,利用相关性来估计参数$P_i(V)$.因为场景内的对象分为可见对象集合与不可见对象集合,在两帧之间,会有一小部分不可见的对象变为可见,有一小部分可见的对象变为不可见.假设可见对象从第$t-1$ 帧到第t帧保持可见状态的概率为w_1,不可见对象保持不可见状态的概率为w_2,则两帧之间的状态转移矩阵为

$$P=\begin{pmatrix} w_1 & 1-w_1 \\ 1-w_2 & w_2 \end{pmatrix},$$

则可以将对象在第 t 帧可见的概率计算为

$$P_i^t(V)=w_1P_i^{t-1}(V)+(1-w_2)P_i^{t-1}(\overline{V}) \tag{11}$$

1.1.5 估计错误分析

如前说述,为了比较绘制与查询开销,需要确定C_r,C_q和$P_i(V)$.当上述 3 个参数已知时,在对节点进行遮挡查询之前,先判断绘制开销与查询开销的期望时间.在Terminate节点集合中,仅对绘制时间开销大于遮挡查询时间开销的节点进行遮挡查询,从而进一步缩小Terminate节点集合,减少提出遮挡查询的数量,以提高绘制效率.为了不影响绘制速度,本文采用了公式(6)、公式(10)和公式(11)的近似方法来快速地确定,因此,很可能因为估计误差导致出现下面两种错误情况,但在这两种情况下均不会产生错误的绘制结果:

1) 错误地认为绘制开销大于查询开销.当这种结果发生时,算法的行为是先进行遮挡查询,根据查询结果进行绘制或者裁剪.与 CHC 算法相比,只是引入了估计的开销,并不会导致绘制结果错误.

2) 错误地认为绘制开销小于查询开销.当这种结果发生时,则直接进行绘制.与 CHC 算法相比,当节点可见时,节省了遮挡查询的开销;当节点不可见时,仅是多绘制了一些几何面片数简单的对象,绘制结果也不会发生错误.

1.1.6 算法流程

改进算法与 CHC 算法基本相同,使用了遍历和查询两个队列,遍历队列存放遍历场景树时等待处理的节点,而遮挡查询队列则存放进行遮挡查询的节点,以实现交替进行遮挡查询和可见节点的绘制,从而减少因为遮挡查询延迟造成的等待时间.

在绘制时,按照自顶向下的顺序遍历场景树中的节点,首先执行视锥体裁剪算法,以裁剪不在视锥内部的节

© 中国科学院软件研究所 http://www.jos.org.cn

点,并将视锥内部节点划分为 Terminate 节点集合与非 Terminate 节点集合,针对 Terminate 集合中的节点进行概率计算来估计绘制开销与遮挡查询的时间开销,如果符合遮挡查询的提出策略,则执行遮挡查询操作,并将该节点插入到查询队列中;否则,直接对该节点执行绘制操作.

节点绘制过程分为两种情况:

(1) 如果是可见叶子节点,则直接送入图形管线进行绘制.

(2) 如果是内部节点,则将其子节点依据视点到节点的中心距离从前到后的顺序插入到遍历队列中,从而保证距离视点近的节点较先被处理,以提高遮挡查询的效率.

改进算法的流程图如图 4 所示.

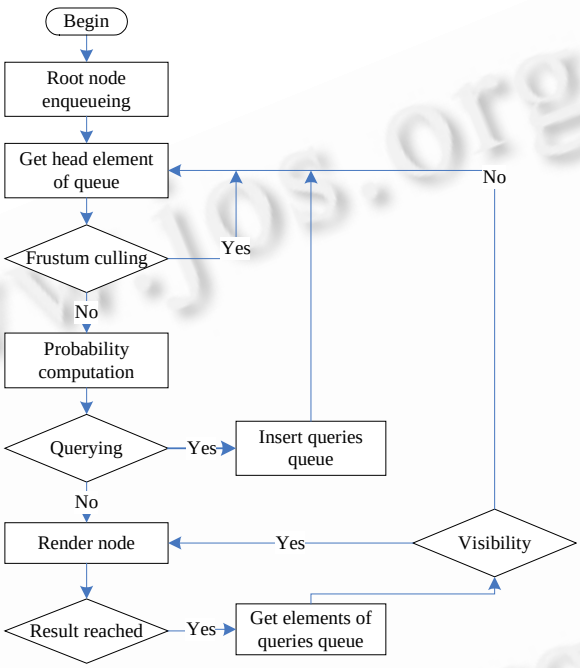


Fig.4 Flow chart of the rendering algorithm

图 4 绘制算法流程图

2 实验结果及分析

为了对本文提出的方法进行验证,我们分别对松散八叉树场景和概率计算模型进行了实验.其中,实验的环境 CPU 为 PIV3.0GHz,内存 1GB,显卡 GeForce6800,显示分辨率为 1024×768,操作系统为 Windows XP,编程语言为 Visual C++6.0 和 OpenGL.

实验 1. 测试在松散八叉树场景组织结构下的对移动对象更新操作的性能.程序设定了一个 3D 正方形虚拟场景,每一边有 1 000 个单位,并生成一些球体填充该场景.每个球体有一个随机的位置并且进行随机的运动(沿着自己的局部坐标系中的 X 轴进行移动).如图 5 所示,场景中总的球体数目为 5 000 个,每个球体的面片数目为 576.

实验并没有加入任何裁剪算法,仅仅对不同数量对象移动时的更新开销进行了测试,并对时间进行了统计,与传统八叉树场景组织节点插入和删除的更新操作进行了对比分析,从而检测出松散结构的场景组织下对移动对象更新操作的性能.

从图 6 可以看出,在松散八叉树的场景组织结构下,当场景中的对象数量不断增加时,更新的开销并没有显著地增加,而对于传统的八叉树结构,因为场景中对象移动所引起的插入与删除操作随着对象数量的增加而显

© 中国科学院软件研究所 http://www.jos.org.cn

著增大,且远大于松散八叉树中的更新开销.因为传统八叉树更新操作包含节点的删除、对象当前所归属节点的确定、插入节点的开销、包围体的更新等.而松散八叉树中只需要一次 $O(1)$操作的计算即可确定移动对象所属的节点.因此,在更新操作上,也就是对动态对象的支持方面,松散八叉树要比传统八叉树有更好的性能.

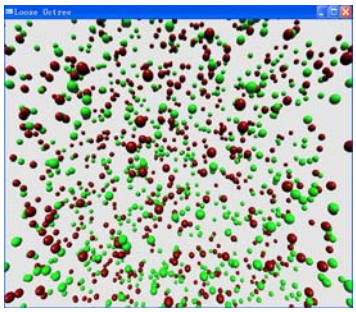


Fig.5 Test of dynamic Sphere scene based on loose Octree

图 5 基于松散八叉树的动态 Sphere 场景测试

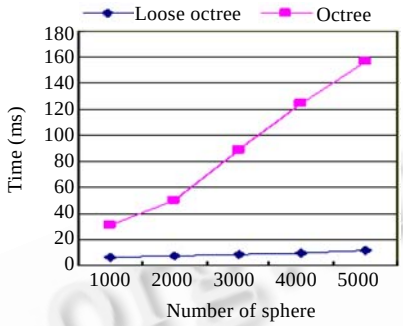


Fig.6 Update time comparison of different number of dynamic objects

图 6 不同数量动态对象更新时间对比图

实验 2. 测试遮挡裁剪算法的性能.本文选取了两个复杂场景,主要测试算法对静态场景与动态场景的支持情况,实验中定义场景树深度为 4.场景 1 由 6 000 个 Teapot 组成,并随机进行了位置与姿态的放置,总体排列是以纵向为主.这个场景不仅提供了复杂的遮挡关系,而且还具有很多自身的特点:第一,Teapot 模型是由许多小的三角面片组成,因此,众多可见的小三角面片的整体融合才能产生遮挡的效果;第二,Teapot 之间可能包含很多的小孔,通过它们可以看到很多其他的 Teapot;第三,以列为主体的位置放置,当视点位于 Teapot 场景的一侧时,可以看到一多半的对象,此时可见对象的变化程度比较大,如图 7 所示.

图 8 是 North Carolina 大学的 Powerplant 场景.该场景由 1 185 个对象组成,共包含上千万的三角面片数.该场景不仅数据量大,而且深度复杂度比较高,在场景中存在很复杂的遮挡关系,可见性与深度复杂度在模型的不同部分之间的差别很大.因此,这个测试场景也是国内外研究裁剪算法应用比较广的场景之一,其主要是为了测试算法在数据量大、深度复杂度高的情况下的执行性能.

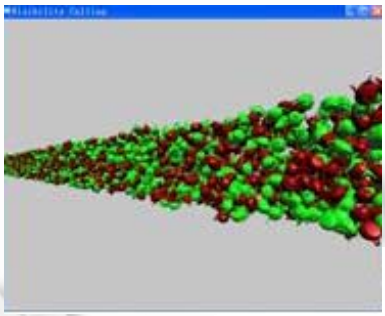


Fig.7 Teapot scene

图 7 Teapot 场景

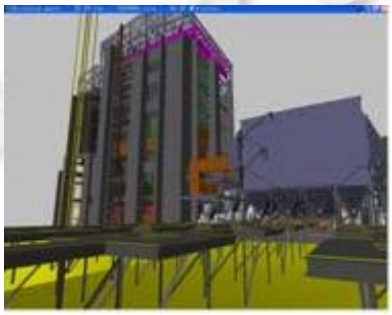


Fig.8 Powerplant scene

图 8 Powerplant 场景

为了进行算法的比较,本文采用了几种不同的算法对上述实验场景进行绘制,包括视锥体裁剪算法(view frustum culling,简称 VFC)、层次等停算法(hierarchical stop and wait,简称 HSW)、CHC 算法与本文算法.视锥体裁剪算法不需要进行遮挡查询,HSW 算法就是对场景的每个节点在绘制前首先进行遮挡查询,然后根据返回的查询结果决定是否对节点进行绘制.各种算法对上述两个场景进行绘制的性能如图 9、图 10 所示.

图 9 是 Teapot 场景的算法对比图,虽然 HSW 遮挡裁剪方法通过遮挡查询减小了绘制的三角形面片数量,但是因为硬件遮挡查询固有的延迟时间,在某些情况下,如在 Frame 600~Frame 1 200 之间,视点位于 Teapot 场景

© 中国科学院软件研究所 http://www.jos.org.cn

的侧面,使得大部分的 Teapot 均可见.此时,HSW 算法和 CHC 算法的帧用时有可能低于 VFC 无遮挡裁剪算法.

图 10 是不同算法绘制 PowerPlant 场景的算法对比图,并给出了理想情况下的绘制时间.该理想情况首先由 HSW 算法进行遮挡查询,而不进行裁剪工作,将可见的节点标记出来,再进行绘制,并取得了实验结果.这种情况对于具体给定的场景实际上是绘制效率最高的.由实验结果可见,在 Frame 200~Frame 300 之间,PowerPlant 模型的大部分都是可见的,HSW 算法的效率要低于 VFC 算法.

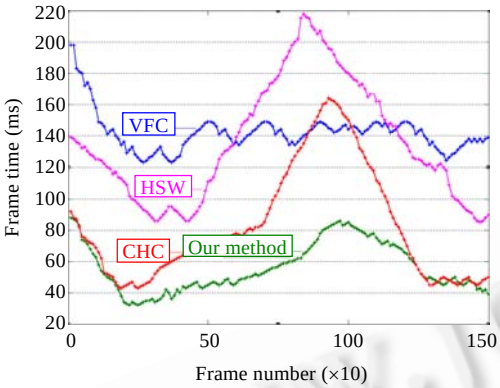


Fig.9 Frame times of different algorithms for the Teapot scene

图 9 Teapot 场景中不同算法帧用时图

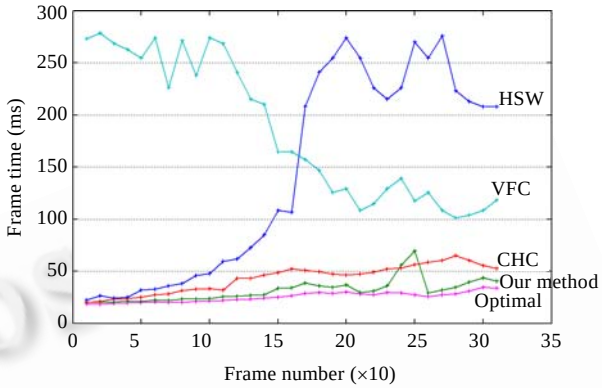


Fig.10 Frame times of different algorithms for the Powerplant scene

图 10 Powerplant 场景中不同算法帧用时图

由表 2 可以看出,本文算法在提出遮挡查询之前,通过比较不同事件的数学期望,选择一种更优的绘制方法,使遮挡查询的数量小于 CHC 算法提出的遮挡查询的数量,从而减少了绘制调用的开销与处理器空等待所带来的延迟,提高了绘制的帧率.另一方面,本文算法在 Terminate 节点集合的基础上,对于绘制开销小于遮挡查询开销的节点不进行遮挡查询.也就是说,当节点不可见时,本文方法为了减少遮挡查询的数量,对于开销小的、不可见节点采用直接绘制,会增加绘制面片数量;而对于可见节点,则会减少 CHC 算法中遮挡查询数量.

Table 2 Experimental results of different algorithms for the Powerplant scene

表 2 Powerplant 场景中不同算法的实验数据

Algorithm	Number of occlusion queries	Number of rendered faces	Flame rate (f/s)
VFC	—	2 608 510	6
HSW	832	436 988	10
CHC	654	542 563	18
Our method	618	656 378	23

3 结论及下一步工作方向

本文主要是采用将复杂场景组织成基于严格空间划分方式的松散八叉树场景组织结构,并以此为基础对遮挡裁剪算法进行研究,给出了一种基于概率计算模型的改进的相关性层次遮挡裁剪算法.

松散的场景组织可以使对象尽量保持在树形结构的更深层次,减少了剖分边界对象的开销,从而克服了传统八叉树场景组织的很多缺点.在此基础上,为了进一步减少冗余和不必要的遮挡查询,给出了一种概率计算模型,通过比较对象直接绘制时间及遮挡查询再绘制时间的数学期望,达到提高遮挡查询效率的目的.文中还给出了相关的实验结果,对算法的有效性进行了验证.

但是这种基于严格的空间划分方式必然要求八叉树的层次不能过深,否则会产生过多的节点,增加了相交测试的开销,今后需要提高相交测试算法的效率来解决这一问题.另外,本文的遮挡裁剪算法对于场景的遮挡情况有一定的要求,主要面向于深度复杂度高的场景,这也是遮挡裁剪算法的共性问题.这个问题对于算法的实际应用具有一定的影响,需要作进一步的研究来解决该问题.

© 中国科学院软件研究所 http://www.jos.org.cn

References:

[1] Wei F, Wang WC, Wu EH. Fast selecting visible surfaces in high precision. Journal of Software, 2005,17(10):2199−2210 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/2199.htm

[2] Pu JT, Zha HB. Research on visibility for large-scale and complex scene. Journal of Computer Research and Development, 2005, 42(2):236−246 (in Chinese with English abstract).

[3] Wald I, Ize T, Kensler A, Knoll A, Parker SG. Ray tracing animated scenes using coherent grid traversal. ACM Trans. on Graphics(TOG), 2006,25(3). http://www.sci.utah.edu/~wald/Publications/2006///Grid/download//grid.pdf

[4] Hillesland K, Salomon B, Lastra A, Manocha D. Fast and simple occlusion culling using hardware-based depth queries. Chapel Hill: University of North Carolina, 2002.

[5] Meißner M, Bartz D, Hüttner1 T, Müller G, Einighammer J. Generation of subdivision hierarchies for efficient occlusion culling of large polygonal models. Technical Report, WSL-99-13, Tubingen: University of Tubingen, 1999.

[6] Bittner J, Wimmer M, Piringer H, Purgathofer W. Coherent hierarchical culling: Hardware occlusion queries made useful. Computer Graphics Forum, 2004,23(3):615−624.

[7] Kovalcik V, Sochor J. Occlusion culling with statistically optimized occlusion queries. In: Proc. of the 13th Int'l Conf. in Central Europe on Computer Graphics, Visualization, and Computer Vision. 2005. 109−112. http://wscg.zcu.cz/WSCG2005/Papers_2005/Short/G71-full.pdf

[8] Liang AM, Liang XH, Yu Z. Visibility culling for complex scene on a loose octree structure. Journal of Computer-Aided Design & Computer Graphics, 2007,19(12):1593−1598 (in Chinese with English abstract).

[9] Akenine-Moller T. Realtime Rendering. A K Peters, Ltd., 2002. 202−210.

[10] DeLoura M. Game Programming Gems. Charles River Media Inc., 2000.

[11] Hsu CK, Tai WK, Chiang CC, Yang MT. Exploiting hardware-accelerated occlusion queries for visibility culling. IEICE Trans. on Fundamentals, 2005,E88-A(7):2007−2014.

[12] Foley J, van Dam A, Feiner S, Hughes J. Computer Graphics Principles and Practice. Addison-Wesley Publishing Company, 1990. 23−52.

[13] Assarsson U, Moller T. Optimized view frustum culling algorithms for bounding boxes. Journal of Graphics Tools, 2000,5(1):9−22.

[14] Coorg S, Teller S. Real-Time occlusion culling for models with large occluders. In: Proc. of the Symp. on Interactive 3D Graphics. 1997. 83−90. http://jesper.kalliope.org/blog/library/vfcullbox.pdf

[15] Greene N, Kass M. Hierarchical Z-buffer visibility. In: Proc. of the SIGGRAPH'93. New York: ACM Press, 1993. 231−240. http://www.cs.princeton.edu/courses/archive/spring01/cs598b/papers/greene93.pdf

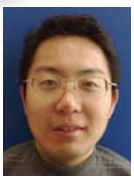
附中文参考文献:

[1] 魏峰,王文成,吴恩华.快速高精度的可见面选择.软件学报,2005,17(10):2199−2210. http://www.jos.org.cn/1000-9825/17/2199.htm

[2] 普建涛,查红彬.大规模复杂场景的可见性问题研究.计算机研究与发展,2005,42(2):236−246.

[8] 梁爱民,梁晓辉,于卓.一种基于松散八叉树的复杂场景可见性裁剪算法.计算机辅助设计与图形学报,2007,19(12):1593−1598.

梁晓辉(1970－),男,山西临汾人,博士,副教授,CCF 高级会员,主要研究领域为虚拟现实,计算机图形学.



于卓(1980－),男,博士生,主要研究领域为计算机图形学.

任威(1984－),男,硕士生,主要研究领域为计算机图形学.

梁爱民(1980－),男,硕士,主要研究领域为计算机图形学.

© 中国科学院软件研究所 http://www.jos.org.cn