ISSN1000-9825 Journal of Software 软件学报 2000,11(6):785~790

一种基于点光源的三维阴影的实时生成算法*

刘列明[1] 吴恩华[1,2]

[1](中国科学院软件研究所计算机科学开放研究实验室 北京 100080)

[2](澳门大学科学技术系 澳门)

E-mail: liulieming@263.net

摘要 阴影的生成在体现虚拟现实环境的真实感程度方面起着重要的作用,但现有生成阴影的算法普遍速度较慢,不能适应实时绘制的要求.该文提出了一种基于点光源的实时生成三维阴影的算法.该算法的基本思想是基于在一个复杂的虚拟环境中,某一时段内只有少数多边形相对于光源发生形状的改变,从而导致相应阴影计算的改变,而大多数多边形相对于光源是静态的.这样就有可能针对静态物体与动态物体分别作不同的处理,使尽可能少的多边形参与阴影计算.方法是,先用 Tiling 方法对静态场景进行预处理,确定相关阴影多边形集合,然后用 SVBSP(shadow volame BSP)树算法计算阴影.文中所有算法均用 C++实现,运行于 Intel Pentium II 400 微机平台,取得了较好的实验结果.

关键词 BSP 树,SVBSP 树,tiling,阴影生成,虚拟现实,真实感图形.

中图法分类号 TP391

阴影的计算在计算机图形学中一直是一项重要而又富有挑战性的课题.在虚拟环境中的阴影能有效地表现出不同物体之间的相对位置关系,从而增加图像的立体感和层次感,提高虚拟环境的真实感程度.阴影分为本影和半影两种,本文所论述的是基于点光源的本影,即有着绝对明暗边界的阴影.目前围绕阴影的生成人们已经做过许多研究,主要的算法大致可分扫描线算法、曲面细节多边形算法(两次消隐算法)、影域多边形(shadow volume)算法、Z-buffer 算法、光线跟踪算法[1,2]、辐射度算法等.这些算法都能较好地完成本影的计算,其中分布式光线跟踪与辐射度方法还能生成半影.但这些算法普遍速度较慢,不能适应虚拟现实环境中实时动态变化的场景的需求.

本文对于复杂动态场景提出了一种新的实时阴影生成算法.在处理具有复杂动态变化场景的虚拟环境时,我们注意到,在同一时段内往往只有少数的多边形相对于光源作形态的变换,从而影响到阴影的计算,而大多数的多边形相对于光源是静止的.这样就可以将某一时段内相对于光源静止与运动的对象分别进行处理,使得在每一时刻参与阴影计算的多边形数目达到最少,从而实现阴影的实时计算.

1 相关工作

1.1 SVBSP

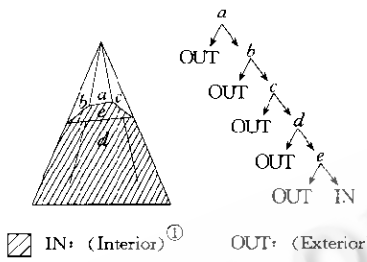
影域 BSP 树(shadow volume BSP tree)算法最早由 Chin 和 Feiner 在文献[3]中提出,是由 Thibault 和 Naylor 的 BSP 树算法[4]演变而来.用一般 BSP 树算法[5]将场景多边形构成了一棵 BSP 树,从而当任意给定一点光源时,可以通过遍历 BSP 树方便地获得相对于该点光源的场景多边形之间的先后次序.而由该次序可以确定场景多边形之间的遮挡关系,即阴影关系,因为相对于光源位于后序的多边形不可能对前序的多边形形成遮挡.于

* 本文研究得到国家自然科学基金(No. 69873044)、国家 863 高科技项目基金(No. 863-306-ZD-10)和 Research Grant of University of Macao(RG009/99-00S/WEH/FST)资助.作者刘列明,1971 年生,博士,主要研究领域为虚拟现实,计算机图形学.吴恩华,1947 年生,博士,研究员,博士生导师,主要研究领域为计算机图形学,虚拟现实,可视化,CAD,CAM.

本文通讯联系人:刘列明,北京 100080,中国科学院软件研究所计算机科学开放研究实验室

本文 1999-03-17 收到原稿,1999-06-08 收到修改稿

是可以按照(相对于光源)从前到后的顺序依次处理各个多边形,同时构造一棵辅助计算阴影的 SVBSP 树.其节点是由点光源与面向光源的多边形的边形成的阴影平面.对于每一个多边形而言,它的每一边与点光源确定的阴影平面所构成的 BSP 树,完整地表示出了该多边形的影域空间(shadow volume).其叶节点 IN 与 OUT 分别表示影域空间的内、外两部分,如图 1 所示.



①区域内部,②区域外部.

Fig. 1　Representation of an open region with BSP tree
图1　含开集的空间区域的BSP树表示

1.2　表面网格算法(Tiling)

Tiling 方法在图形学的许多领域中都有应用,最早可以追溯到 Encarnacao 的用于隐藏面计算的扫描网格算法[7],在辐射度方法中用于计算形状因子的半立方体结构也是类似的思想.考虑单一的点光源,构造一个包围该点光源所在位置的立方体.将该立方体的表面划分为正方形的网格,每一网格可称为一个网元(tile).以光源为投影参考点,可以将景物多边形投影到包围平面上,根据其二维投影与网元之间的关系可以确定多边形之间的三维位置关系.本文将采用 Tiling 方法来计算运动物体与静态场景之间的阴影关系.具体算法将在下文中述及.

2　本文的算法

动态对象在运动过程中,与静态场景之间以及与其自身所包含的多边形之间的遮挡关系会发生改变,因而场景中的阴影也会随之改变.本算法的目的就是首先精确确定与运动对象在某一时刻实际形成遮挡关系(遮挡与被遮挡)的静态多边形集合,然后由该多边形集合构成一棵 BSP 树,并与原运动对象的 BSP 树合并在一起,共同计算出阴影.这样就使得在每一时刻都只有尽可能少的多边形参与阴影计算,从而极大地提高了运算速度.

2.1　关键步骤

(1) 遮挡多边形集合的确定

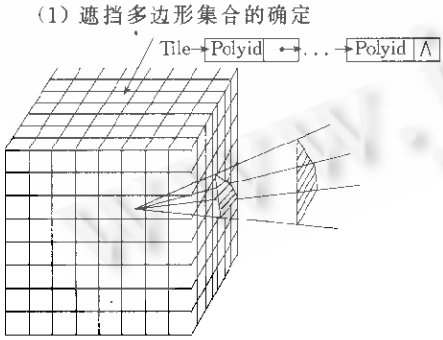


Fig. 2　Tiling algorithm
图2　Tiling算法

本算法采用 Tiling 方法来确定与运动对象相关的遮挡多边形集合[8].如图 2 所示,在预处理阶段,对点光源如上文所述形成六面的 Tiling 结构.然后,以点光源所在位置为投影参考点,将整个静态场景中的所有多边形投影到 Tiling 表面上.每个网元中记录下投影到该网元的多边形标识,用链表的形式组织在一起.

在实际阴影计算时,只需对运动对象构造一个六面的包围盒.然后将这 6 个面相对于光源作一次投影,取出所投网元上的静态多边形的记录,便可得到非常精确的遮挡多边形集合.

(2) BSP 树的合并操作

结构实体几何法(constructive solid geometry)是一种常用的建模方法,它是用集合操作来组合复杂的三维物体.该方法的缺点是只能处理闭集,并且需要复杂的数据结构和算法来进行各种运算操作.Naylor 和 Thibault[4,9]提出了一种简单、高效且适用于开集的空间区域表示法——BSP 树方法(如图 1 所示),分别用"IN"和"OUT"表示区域的内部与外部.

可以方便地通过BSP树的合并来实现几个多面体之间的集合操作,如并、交、差等.不同集合操作的合并算法其递归过程都是一样的,见算法1.不同的结果由算法中Cell与Tree的不同的具体运算方式(TreeOpCell)决定.本算法所采用的运算与集合运算不同,为了便于区分,我们定义本算法所采用的合并操作为整合运算,用符号"+"表示.该运算可以表述成如下形式:

给定同一三维空间的两个BSP树划分:Partition1与Partition2,要求Partition3,使得:Partition3=Partition1+Partition2,即

$$c_3 \in \text{Partition3 and } c_3 \in Cell$$
$$\Leftrightarrow \exists\ c_1 \in \text{Partition1}, c_2 \in \text{Partition2},$$

使得$c_3=c_1 \cap c_2, c_3 \neq \varnothing$,如图3所示.

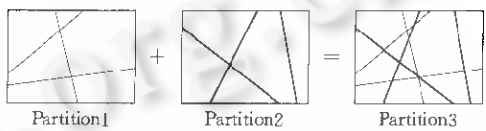


Fig. 3 Plus of two BSP trees
图3 BSP树的整合

算法1. BSP树的合并

```
BSPTree merge (BSPTree t1, BSPTree t2)
{
    /* Cell 与 Tree 的运算 */
    if (IsCell(t1) or IsCell(t2))
        return TreeOpCell (t1,t2);
    endif
    /* 用 t1 根节点的平面将 t2 划分为 t2+ 与 t2- */
    {t2+,t2-}=PartitionTree(t2,root(t1));
    t1.negNode=merge(t1.negNode,t2-);
    t1.posNode=merge(t1.posNode,t2+);
    return t1;
}
```

两棵BSP树T_1与T_2的合并操作可以看作是将T_2插入T_1的过程:从T_1的根节点开始,用位于T_1根节点的平面将T_2分成两棵子树T_2^+与T_2^-(具体算法参见文献[9]),再将T_2^+与T_2^-分别递归插入T_1根节点的左右子树T_1^+与T_1^-,直至到达叶节点(cell).

叶节点与子树的合并决定了实际运算结果.Naylor与Thibault在文献[4,9]中给出了集合运算的列表,本文增加运算符"+",一起列表,见表1.

Table 1 Operation in merging
表1 扩充的合并运算

op	Cell	Tree	Cell(op) Tree
∩	IN	T	T
	OUT	T	OUT
∪	IN	T	IN
	OUT	T	T
−	IN	T	$\sim T$
	OUT	T	OUT
+	IN	T	T
	OUT	T	T

整合操作"+"表示将两棵BSP树完整地合并为一棵BSP树(如图3所示).在两棵BSP树的整合过程中,最基本的操作是两个多边形之间的比较与分割,故可将其作为基本的运算量$O(1)$.设T_1有m个节点,T_2有n个节点,为了方便,我们不妨假设T_1与T_2接近平衡(接近BSP树"好"的标准).这样,将T_2插入T_1时,T_1的每一个节点的多边形都要遍历T_2中的相应子树,故复杂度为$n\log(\min(m,n))$.另外,已知n个初始多边形构造一棵BSP树的空间与时间复杂度的上界为$O(n^2)$[10,11],因此,总的运算量可表示为$Kn^2+n\log m$.这表明,应尽量取T_2为节点较少的树,可减少运算量.

同时,若将组成T_2的多边形直接依次插入,其运算复杂度为$m\times n$,同上比较,当$m\times n>kn^2+n\log m$,即$m>kn+\log m$时,本算法的效率会更高.一般情况下,若取T_1为运动物体且较复杂($m\gg n$)时,便能满足上式.

2.2 算法流程

算法可以分为预处理与实时处理两个阶段.

(1) 预处理阶段

〈1〉将整个虚拟环境分为动态场景与静态场景,确定点光源的位置 lightposition $i(i=1..n)$.

〈2〉对每一个光源 light,计算它的用作 Tiling 投影的包围盒.

〈3〉计算静态场景的 BSP 树 Bsp1.

〈4〉由静态场景的 BSP 树生成阴影树(SVBSP),确定静态场景本身的阴影关系.

〈5〉计算静态场景相对于 light 的 Tiling 投影,在 Tile 中记录为链表结构.

〈6〉计算动态场景的 BSP 树 Bsp2.

〈7〉计算动态场景的包围盒.

(2) 实时处理阶段

〈1〉动态场景实现坐标变换(可设为左乘 4×4 的矩阵 M).

〈2〉用 M 变换动态场景的 BSP 树.

〈3〉用 M 变换动态场景的包围盒.

〈4〉将该包围盒相对于光源求 Tiling,求出所有相对于该光源与该包围盒有遮挡关系的静态场景面片(即可能遮挡它或被它遮挡的所有静态场景中的多边形).

〈5〉将求出的这些多边形生成一棵新的 BSP 树 Bsp3.

〈6〉合并 Bsp2 与 Bsp3,得到一棵新树 Bsp4.

〈7〉显示静态场景的 SVBSP 树.

〈8〉对 Bsp4 求其 SVBSP 树并完成显示.

3 实验结果

本文中的所有算法均用 C++实现,运行于 Intel Pentium II 平台,主频为 400.场景中含有单个点光源.静态场景取恒定的多边形个数.所得实验结果见表 2.

Table 2 Experimental results

表 2 实验结果

ISPN①	IMPN②	MBSP③	SBSP④	SSPN⑤	DCSP⑥	Total⑦	Tiling⑧(s)	Merge⑨(s)	Shadow⑩(s)	TR⑪(s)
255	6	6	477	5 491	75	5 566	0.00	0.06	0.16	0.22
255	48	60	477	5 491	659	6 150	0.00	0.06	0.90	0.84
255	48	60	477	5 491	801	6 292	0.00	0.05	0.98	0.96
345	48	60	477	5 491	1 691	7 182	0.00	0.06	2.96	2.41

①原始静态多边形数目,②原始动态多边形数目,③动态多边形经过 BSP 树划分后所得多边形数目,
④静态多边形经过 BSP 树划分后所得多边形数目,⑤对静态多边形进行阴影计算后所得总多边形数目,
⑥动态多边形与它的遮挡多边形计算阴影所生成的多边形数目,⑦每一次绘制的总多边形个数,
⑧Tiling 所需的时间,⑨BSP 树整合所需时间,⑩动态阴影生成时间,⑪每一帧的总绘制时间.

在表 2 中,Total 项为绘制时实际参与运算的总多边形个数.由以上结果可以看出,运算所花费的时间主要由表 2 中的 DCSP 项,即需实时计算的多边形数目所决定,随 DCSP 项的增大而耗时增加,而与静态场景中的绝对多边形数目关系不大.因此,当 DCSP 项数目合适时,可以满足实时的要求(表 2 中第 1 项约为 5 帧/s).同时,在场景中数目占大多数的静态多边形的阴影计算在预处理阶段完成,这一方面节约了运算时间,另一方面又极大地增强了场景的真实感.图 4 为实验场景,中间的坐标轴本身在空间作旋转运动,与场景中的转椅等物体交互形成动态阴影.其余的静态物体如书架、花盆等的阴影在预处理阶段获得.图 5 为物体旋转后所得场景.图 6 与图 7 为线框图.

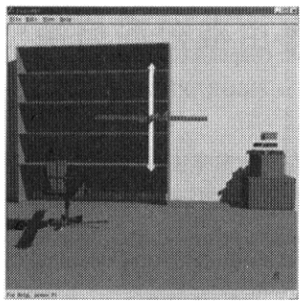


Fig. 4 Dynamic scene
(The coordinate axis in the middle is the moving object)
图4 运动场景(中间坐标轴为运动物体)

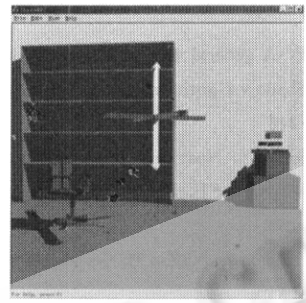


Fig. 5 Dynamic scene after rotation of the axis
图5 坐标轴旋转后的场景

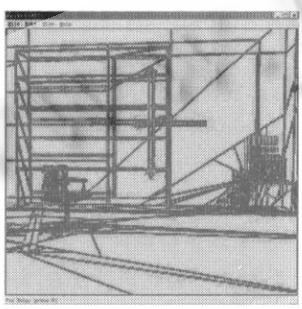


Fig. 6 Wire frame 1
图6 线框图1

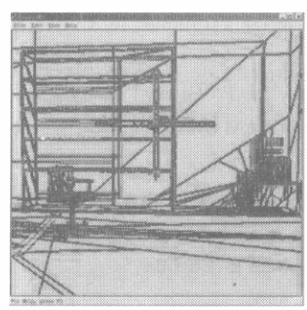


Fig. 7 Wire frame 2
图7 线框图2

4 与相关工作的比较

表3是 Chin 和 Feiner 在文献[3]中给出的用 BSP 树生成阴影的实验结果,程序用 C 编写,运行于 HP 9000 350 TurboSRX 图形工作站.由此可见,单纯使用 BSP 方法生成阴影,在场景多边形较多时很难满足实时要求.

Table 3 Results for comparison

表3 比较结果

Input polygons①	Lights②	Fragments③	Time for shadow Producing④(s)
27	1	49	0.23
258	1	723	4.15
1 026	1	3 345	25.44

①输入多边形个数,②点光源个数,③分割多边形个数,④阴影生成时间.

5 结束语

本文针对动态变化的复杂场景提出了一种基于点光源的三维阴影生成算法,并在微机平台上实现,取得了较好的效果.随着分布式虚拟现实技术的广泛应用,真实感场景的实时生成已成为必需.希望本文的工作能对这方面的研究有所助益.

参考文献

1 Bergeron P. A general version of crow's shadow volumes. IEEE Computer Graphics & Applications, 1986,6(9):17～28

2 Crow F. Shadow algorithms for computer graphics. In: Proceedings of the SIGGRAPH'77. Computer Graphics, 1977,11(3):242～248

3 Chin N, Feiner S. Near real-time shadow generation using BSP trees. ACM Computer Graphics, 1989,23(3):99～106

4 Thibault W, Naylor B. Set operations on polyhedra using binary space partition trees. ACM Computer Graphics, 1987,21(4):153～162

5 Fuchs H, Kedem Z, Naylor B. On visible surface generation by a priori tree structures. ACM Computer Graphics, 1980,14(3):124～133

6 Haines E A, Greenberg D P. The light buffer: a shadow testing accelerator. IEEE Computer Graphics & Applications, 1986,6(9):6～16

7 Naylor B, Amandatides J, Thibault W. Merging BSP trees yields polyhedral set operations. ACM Computer Graphics, 1990,24(4):115～124

8 Paterson M, Yao F. Binary partitions with applications to hidden surface removal and solid modeling. In: Mehlhorn K ed. Proceedings of the 5th Annual ACM Symposium on Computational Geometry (SCG'89). Saarbrücken, FRG: ACM Press, 1989. 23～32

9 Paterson M, Yao F. Optimal binary space partitions for orthogonal objects. In: Johnson D ed. Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'90). San Francisco, CA: ACM Press, 1990. 100～106

A Real-Time Algorithm of 3-D Shadow Generation for Point Light Sources

LIU Lie-ming[1] WU En-hua[1,2]

[1](*Laboratory of Computer Science Institute of Software The Chinese Academy of Sciences Beijing* 100080)

[2](*Faculty of Science and Technology University of Macao Macao*)

Abstract Shadow generation plays an important role in realistic image synthesis, but with the old algorithms, generation of shadows is quite time-consuming and can't be used in a real-time interactive environment. The authors present a new method to produce the real-time shadows for the point light sources in this paper. The algorithm takes advantage of the fact that in many cases, only a small number of polygons in the complex virtual environment change their shape relative to the lights resulting in a corresponding change to the shadow calculation while most of the polygons remain static relative to the lights. So it is possible to process the static and dynamic objects separately and thus make the number of polygons to be calculated as less as possible. The main idea is to prepocess the static scene with "Tiling" method and decide the precise shadow-relative sets, and then use SVBSP method to calculate the dynamic shadows. All the algorithms in this paper are realized in C++, and run on the microcomputer platform of Intel Pentium II 400. The experimental results are satisfactory.

Key words BSP tree, shadow volume BSP tree, tiling, shadow generation, virtual reality, realistic graphics.