第37卷 第21期
Vol.37 No.21

计 算 机 工 程
Computer Engineering

2011年11月
November 2011

·开发研究与设计技术· 文章编号：1000—3428(2011)21—0270—03 文献标识码：A 中图分类号：TP37

虚拟漫游环境中基于矢量的快速碰撞检测算法

章玉文[1]，戴 青[1]，郭方达[2]

(1. 解放军信息工程大学电子技术学院，郑州 450004；2. 解放军信息工程大学信息工程学院，郑州 450000)

摘 要：从虚拟漫游角度出发，将场景中的碰撞检测转化为视点与三角形的碰撞问题，提出一种基于矢量判别的快速碰撞检测算法。该算法通过位向因子和有向回路的方向来判断视点是否与三角形碰撞，从而避免复杂的余弦函数、移动距离等计算，加快检测速度。实验结果表明，对于大型复杂场景，该算法的检测效率优于其他基于余弦函数的视点与三角形碰撞检测算法。

关键词：虚拟漫游；碰撞检测；位向因子；有向回路；时间复杂度；空间复杂度

Fast Collision Detection Algorithm Based on Vector in Virtual Roaming Environment

ZHANG Yu-wen[1], DAI Qing[1], GUO Fang-da[2]

(1. Institute of Electronic Technology, PLA Information Engineering University, Zhengzhou 450004, China;
2. Institute of Information Engineering, PLA Information Engineering University, Zhengzhou 450000, China)

【Abstract】From the perspective of the Virtual Roaming(VR), this paper translates the collision detection in the virtual environment to the problem between view-point and triangles, and proposes a fast collision detection algorithm based on vector discriminance. This algorithm utilizes the direction of the direction factor and the directed circuit to determine whether collision happens, to avoid the complex computing of cosine function and mobile distance. Experimental results show that, the detection efficiency of the algorithm is superior to others, especially for large complex scenes.

【Key words】Virtual Roaming(VR); collision detection; direction factor; directed circuit; time complexity; space complexity

DOI: 10.3969/j.issn.1000-3428.2011.21.092

1 概述

碰撞检测是构成虚拟现实系统的基本要素，也是实现逼真漫游的关键环节。当人或物体在前进或后退时，若出现进入地面或者穿墙而过的现象，就有违真实世界。因此，一个合理完善的虚拟系统必须能够实时而精确地检测出物体之间是否发生碰撞，并对其进行相应的处理。由于碰撞检测算法对具体碰撞场景非常敏感，到目前为止没有哪一种算法能够适用于所有场景。

对于虚拟装配以及仿真手术等方面的碰撞，检测更注重的是精确度，而对于虚拟漫游系统而言，一般只要近似模拟碰撞并做出反应即可，因此算法更注重的又是实时和快速。近年来，学者们针对漫游系统的碰撞特性提出过不少检测算法。文献[1]将观察者简化成球体，利用球体和三角面的交点位置来判断碰撞是否发生；文献[2]将视点简化成一个点，利用前后视点与三角行的位置情况来进行检测。两者虽然简化了碰撞模式，但在检测频率方面存在缺陷。针对上述问题，文献[3]提出采用包围球来代替化身，通过实施对球与多边形的碰撞检测来计算碰撞点和可移动距离，再利用化身运动的几何连贯性避免了化身运动轨迹的离散化处理，从而避免了步长问题，但是文中所用的碰撞形体子集剔除法存在局限性，若化身球的运动轨迹不是直线，而是毫无规则的扭曲路线，那么它与任何形体子集都有可能发生碰撞，剔除法也就失去实际意义。并且，上述所有算法在检测过程中都存在较复杂的数学计算问题，例如文献[1-2]除了碰撞点的计算与判断外，都是通过余弦函数计算三角形内角之和来判断是否相交，文献[3]需要计算切触点、碰撞点、可移动距离，还要对可移动距离进行求导分析等，这些计算上的复杂度都在一定程度上影响了碰撞检测的速度。

为了更好地解决上述问题，本文提出利用矢量判别法来进行碰撞检测。考虑到虚拟场景中任何物体的表面都可以被剖分成三角行，即程序的输入是顶点数组和由这些顶点构成三角网的索引数组，把视点简化成一个点，把碰撞检测转化为点与三角面是否相交的问题。利用位向因子和有向回路法来判断视点与三角形的矢量关系，并对这个矢量关系进行分析进而判断出视点是否与三角形碰撞，从而避免了余弦函数、移动距离求导等计算。

2 基本概念

2.1 位向因子

设 M 为 R^3 空间中的标准模型物体，其中任一模型 M_i 都有 $M_i \in M$。设 M_i 由 N 个三角形 $T_{i1}, T_{i2}, \cdots, T_{in}$ 构成，记为：$M_i = \bigcup_{k=1}^{N} T_{ik}$。其中，$T_{ik} = \{V_{ik1}, V_{ik2}, V_{ik3}\}$，三角形 3 个顶点连线构成 3 条有向边，用向量表示分别计为：$\boldsymbol{E}_{ik1} = \boldsymbol{V}_{ik1}\boldsymbol{V}_{ik2}$，$\boldsymbol{E}_{ik2} = \boldsymbol{V}_{ik2}\boldsymbol{V}_{ik3}$，$\boldsymbol{E}_{ik3} = \boldsymbol{V}_{ik3}\boldsymbol{V}_{ik1}$。

三角形平面在顶点 V_{ik1} 处的单位法矢为：

$$\boldsymbol{n}_{ik} = (\boldsymbol{E}_{ik1} \times \boldsymbol{E}_{ik2}) / (\boldsymbol{E}_{ik1} \times \boldsymbol{E}_{ik3}) \quad (1)$$

由于三角形在 3 个顶点处的法矢是相同的，因此只需计算其中一个即可，文中采用 $\boldsymbol{n}_{ik}$ 作为三角行平面的法矢，其指

作者简介：章玉文(1984－)，女，硕士研究生，主研方向：多媒体技术；戴 青，副教授；郭方达，硕士研究生

收稿日期：2011-04-20 **E-mail：**octopuszyw@yahoo.cn

向为平面的外侧。

设 V 是 R^3 空间中的任一点，对于属于模型 M 的任一个三角形 T_{ik}，判断点与三角形 T_{ik} 的位置关系可以用 T_{ik} 的任一顶点，不妨取点 V_{ik1}，来判断 $\boldsymbol{V}_{ik1}\boldsymbol{V}$ 与 $\boldsymbol{n}_{ik}$ 的夹角，则可以确定点在三角形平面的外侧、内侧，还是共面。定义位向因子为：

$$D(V,T_{ik})=\boldsymbol{V}_{ik1}\boldsymbol{V}\cdot\boldsymbol{n}_{ik} \tag{2}$$

则 $D(V,T_{ik})>0$，V 点在三角形外侧；$D(V,T_{ik})=0$，V 点与三角形共面；$D(V,T_{ik})<0$，V 点在三角形内侧。

取 T_{ik} 的法向矢量(非单位化的) $\boldsymbol{N}_{ik}=\boldsymbol{E}_{ik1}\times\boldsymbol{E}_{ik2}$，则 $\boldsymbol{N}_{ik}$ 与 $\boldsymbol{n}_{ik}$ 方向相同。因此，为计算简便，可以用 $\boldsymbol{N}_{ik}$ 代替 $\boldsymbol{n}_{ik}$，则式(2)可以改写为：$D(V,T_{ik})=\boldsymbol{V}_{ik1}\boldsymbol{V}\cdot\boldsymbol{N}_{ik}$。

为了提高算法速度，可以在建模或初始化时将 $\boldsymbol{N}_{ik}$ 计算出，以便需要时直接调用。

2.2　有向回路法

本文定义 2 点如下：

定义 1 三角形的方向，三角形的方向即顶点的排列顺序，规定逆时针方向为正，顺时针方向为负。

定义 2 有向回路的内侧和外侧，三角形所构成的有向回路的内侧和外侧规定如下：对于方向为正的三角形，内侧为左，外侧为右；对于方向为负的三角形，内侧为右，外侧为左。

对于三角形(最简单的凸多边形)，有如下引理：

引理 1 将平面上的三角形 T_{ik} 看作一个有向边的序列 $\boldsymbol{V}_{ik1}\boldsymbol{V}_{ik2}$、$\boldsymbol{V}_{ik2}\boldsymbol{V}_{ik3}$、$\boldsymbol{V}_{ik3}\boldsymbol{V}_{ik1}$ 所构成的有向回路。对于平面上的任意一点 Q，如果 Q 在其中某 2 条有向边的异侧，那么 Q 为 P 的外点；如果 Q 在 3 条有向边的同一侧(即该三角形回路的内侧)，那么 Q 为 P 的内点；如果 Q 在其中某一条有向边上，那么 Q 为 T_{ik} 的边界点。

关于如何判断平面上一点 Q 在有向线段的哪一侧，有如下引理：

引理 2 对于平面上的有向线段 $\boldsymbol{V}_{ik1}\boldsymbol{V}_{ik2}$ 和平面上任意一点 V，如果 $(\boldsymbol{V}_{ik1}\boldsymbol{V}_{ik2}\times\boldsymbol{V}_{ik1}\boldsymbol{V})\cdot\boldsymbol{e}_z>0$，那么 Q 在有向线段 $\boldsymbol{V}_{ik1}\boldsymbol{V}_{ik2}$ 的左侧；如果 $(\boldsymbol{V}_{ik1}\boldsymbol{V}_{ik2}\times\boldsymbol{V}_{ik1}\boldsymbol{V})\cdot\boldsymbol{e}_z<0$，那么 Q 在有向线段 $\boldsymbol{V}_{ik1}\boldsymbol{V}_{ik2}$ 的右侧；如果 $(\boldsymbol{V}_{ik1}\boldsymbol{V}_{ik2}\times\boldsymbol{V}_{ik1}\boldsymbol{V})\cdot\boldsymbol{e}_z=0$，那么 Q 在有向线段 $(\boldsymbol{V}_{ik1}\boldsymbol{V}_{ik2}\times\boldsymbol{V}_{ik1}\boldsymbol{V})\cdot\boldsymbol{e}_z=0$ 所在的直线上(证明略)。

同时利用引理 1 和引理 2 就能够判断平面上点 Q 是否为三角形的内点，为方便分析，这里记为：

$$A(Q,T_{ik})=(\boldsymbol{V}_{ik1}\boldsymbol{V}_{ik2}\times\boldsymbol{V}_{ik1}\boldsymbol{V})\cdot\boldsymbol{e}_z\cap(\boldsymbol{V}_{ik2}\boldsymbol{V}_{ik3}\times\boldsymbol{V}_{ik2}\boldsymbol{V})\cdot\boldsymbol{e}_z\cap(\boldsymbol{V}_{ik3}\boldsymbol{V}_{ik1}\times\boldsymbol{V}_{ik3}\boldsymbol{V})\cdot\boldsymbol{e}_z$$

则有：$A(Q,T_{ik})>0$，点 Q 在三角形内侧；$A(Q,T_{ik})<0$，点 Q 在三角形外侧；$A(Q,T_{ik})=0$，点 Q 在三角形的一条边上。

3　碰撞检测算法

3.1　算法分析

从虚拟漫游的角度出发，把摄像机(文中称之为视点)看做一个点 V，把场景中物体表面剖分成三角形，记为 T_{ik}，碰撞检测就转化为判断点与三角形的位置关系。算法的过程为首先记录当前帧和上一帧视点的位置，然后计算出相关的位向因子 $D(V,T_{ik})$。当 $D(V,T_{ik})>0$ 时，点在三角形外侧说明碰撞没有发生。当 $D(V,T_{ik})=0$，点 V 与三角形共面，这不能说明碰撞发生，因为平面是无边界的，因此还要利用有向回路法判断点 V 是否在三角形内，若是，则碰撞发生，反之就没有发生。当 $D(V,T_{ik})<0$ 时，计算相对多了一步。连接前后帧视点做一直线，使之与三角形平面相交与点 Q，再同上利用有向回路法判断点 Q 是否在三角形内，以确定碰撞是否发生。具体算法步骤如下：

Step1 记录视点在上一帧的位置 prepoint 和当前帧的位置 nowpoint。

Step2 遍历场景中的每一个三角形，确定当前三角形各个顶点的位置，并计算出其位向因子 $D(V,T_{ik})$。

Step3 若 $D(V,T_{ik})>0$，则说明碰撞没有发生，使下一个三角形成为当前三角形跳转到 Step 2。否则跳转到 Step 4。

Step4 若 $D(V,T_{ik})=0$，点 V 就是点 Q，求出 $A(Q,T_{ik})$，跳转到 Step 6。否则跳转到 Step 5。

Step5 当 $D(V,T_{ik})<0$ 时，点在三角行的内侧，此时过点 prepoint 和点 nowpoint 做直线，使其相交与三角形平面，并得到交点 Q。求出 $A(Q,T_{ik})$，跳转到 Step 6。

Step6 若 $A(Q,T_{ik})<0$，则碰撞没有发生，改下一三角形为当前三角形，跳转到 Step2。否则跳转到 Step 7。

Step7 当 $A(Q,T_{ik})\geqslant 0$，即说明碰撞发生，修正当前帧的位置，使视点沿着三角形平面平行的方向前进。

当遍历完场景中所有三角形后，碰撞检测就全部结束。图 1 为上述算法的流程。

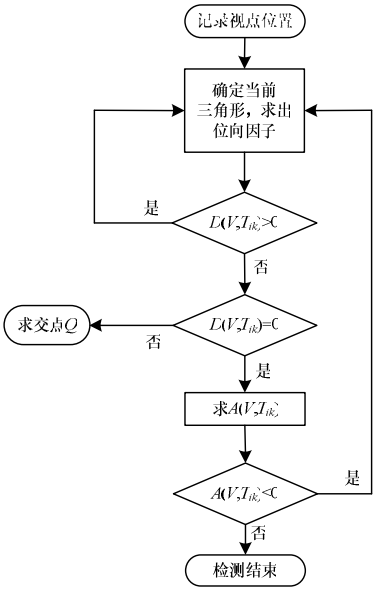


图 1　矢量判别算法流程

3.2　算法优化

对于简单空旷的虚拟场景而言，需要逐个检测的物体数量不多，上述算法能够满足虚拟漫游快速实时检测碰撞情况的要求，但在物体不仅数量多而且形状复杂的大型环境中，算法所要遍历的三角形数目就非常巨大，致使检测速度大大降低，影响漫游效果。然而事实上，再复杂的场景，视点在前进的过程中也不可能与所有物体都发生碰撞，最为可能碰撞的就是视点运动轨迹上的物体。因此需要把其他一切根本不可能发生碰撞的物体排除在检测范围之外，这就要对被测物体进行有效的过滤。本文采用分割过滤法[4]，将虚拟空间分成规则网格，借此还可以把场景中物体分成更小的组群，每当视点移动就计算出当前所在的网格，只对网格内的物体进行碰撞检测，从而大大减少了检测次数。另外，视点每移动一次距离的长短也对检测效率有很大的影响。空旷的地段

上基本不会发生碰撞，此时视点移动距离可以增大，而拥塞的地段上时刻都有可能发生碰撞，移动距离应该适时减小，这实际上就是一个检测频率的问题，本文采用可变时间片长的方法，把性能和真实感[5]2 个指标做综合考虑，使上述算法得到进一步的优化。

3.3 算法复杂度分析

设虚拟环境中进行分割过滤后所需遍历的三角形个数为 n，则整个算法过程中时间复杂度分析为如下：(1)矢量判断法实现简单，尤其对 V 在三角形外侧时效率较高，最优情况下只需做一次卷积计算和一次比较运算即可，时间复杂度为 $O_1(n)$。(2)当 V 与三角形共面时，除情况(1)外还要利用有向回路法做进一步判断。当点 Q 在三角形外时，最优情况下做一次叉积和一次比较运算，最差情况下做 3 次叉积和 3 次比较运算。时间复杂度为 $O_1(n)+O_2(n)$。(3)当点 V 在三角形内侧时，整个判断过程就比情况(2)多了一步计算，即求前后帧视点连线与三角形平面交点 Q 的步骤，记这个时间复杂度为 $O_3(n)$，则总体复杂度为 $O_1(n)+O_2(n)+O_3(n)$。

由于算法在整个检测过程中都只需调用三角形的索引数组及顶点数组，无需额外的系统开销，因此空间复杂度为 $O(n)$，但是在检测过程中并不是每个三角形都需要被检测，采用上述优化算法即网格过滤法之后完全不可能发生碰撞的物体就不对其进行三角剖分，故算法的空间复杂度可以提高到 $O(\lg n)$。

从以上的分析中可以看出，矢量判别法对没有碰撞发生的场景检测效率极高，在最优的情况下只需要做一次卷积运算和一次比较运算。即使在最坏情况下，需要进行 3 个步骤的计算，回路判断法与文献[1-2]中利用余弦函数计算三角形内角之和的方法相比较，叉积运算更加简单快速，尤其是对点 Q 在三角形外时，检测效率就更高。当大型复杂场景中被测物体数量特别巨大时，这种节约所积累出的时间就很明显，检测速度明显提高。

4 实验比较与分析

实验所用的 PC 配置为 CPU PIV 内存 1 GB，显卡 ATI HD 4 500，显存 512 MB，在 VC++.NET 编译环境下使用 Direct 3D 渲染某大规模场景。将本文所提的矢量判别法与文献[6]中基于余弦函数的视点与三角形碰撞检测算法进行比较，实验内容为在同一环境下 2 种不同算法所得到的点与三角形碰撞每帧所用的平均时间，限于篇幅，仅罗列部分实验数据，表 1 为 5 组实验下的场景数据及实验结果。

表 1 检测点与三角形碰撞每帧所用的平均时间 ms

三角面片数	基于余弦	基于矢量
10 173	2.96	2.68
17 462	3.42	3.04
28 165	4.51	3.97
45 568	6.87	5.41
78 361	10.83	7.18

由此可以看出，随着三角片数的增加，矢量判别法在进行碰撞检测的时间上明显优于普通基于余弦函数的视点与三角形碰撞检测算法。

5 结束语

本文利用位向因子和有向回路相结合的方法来判断碰撞是否发生，避免了普通射线与三角面相交测试法中余弦函数等计算，能降低算法的时间复杂度，从而提高碰撞检测效率，在一定程度上提升了大型复杂虚拟系统的整体功能。

参考文献

[1] 熊 伟, 毛善君, 马蔼乃. 基于观察者的碰撞检测技术在虚拟环境漫游中的应用[J]. 计算机应用, 2002, 22(11): 6-10.

[2] 高丽娜, 马尧海. 虚拟漫游中的碰撞检测问题的解决方法[J]. 计算机仿真, 2006, 23(2): 189-191.

[3] 常 明, 罗自荣, 李 丹, 等. 面向虚拟环境漫游的快速碰撞检测算法[J]. 华中科技大学学报, 2006, 36(11): 7-10.

[4] 涂 超, 颜辉武. 碰撞检测技术研究[J]. 计算机工程与应用, 2001, 37(19): 142-144.

[5] 陈 焱, 周莹莹, 杨柏林. 适用于移动设备的实时碰撞检测算法[J]. 计算机工程, 2010, 36(6): 273-275.

[6] 孔勇平, 莫思泉, 朱 娟. 基于视点的碰撞检测算法的应用研究[J]. 微计算机信息, 2009, (30): 140-142.

编辑 任吉慧

(上接第 269 页)

出一个互斥向量 $\boldsymbol{E}=(e_1,e_2,\cdots,e_m)$，$e_i=1$ 表示第 i 项用途使用互斥操作，$e_i=0$ 表示共享操作。根据 $\boldsymbol{E}$，可以把对相关操作互斥的资源分为 2 个资源来进行计算。比如 $\boldsymbol{R}_1=(1,1,1,1,0)$、$\boldsymbol{E}_1=(0,1,0,1,0)$，在进行计算时可把 $\boldsymbol{R}_1$ 拆分为 $\boldsymbol{R}_{1a}=(1,1,1,0,0)$ 和 $\boldsymbol{R}_{1b}=(0,0,0,1,0)$。在系统设计中，互斥在软件逻辑中必然也有所体现。另外对综合化系统的健康管理框架，可以为系统设置多 Agent 结构，依次有 Mission Agent，Task Agent，Procedure Agent, Function Agent 和 Resource Agent。第 1 种 Agent(前 4 个)用来管理需求，第 2 种 Agent 用来测试报告资源有效性。这样便于收集资源的测试结果，得到当前度量的输入；也便于进行协同健康管理，目的是通过 Agent 之间的相互协作形成一个自治的综合化健康管理系统。

6 结束语

现代航空电子系统正沿着系统综合化的道路发展。健康管理是综合化系统研究中的一个重要方向，一个健康的综合化系统是良好的资源有效性(物理健康)和系统管理(逻辑健康)有机结合的整体。综合化系统的 PHM 有预测与管理 2 个方面。健康度量是预测与管理的基础。从用户接口的应用层面来看，能够满足一定的任务需求是关键。因而评价一个综合化系统的健康程度必然要以任务需求为健康基线。本文的健康度量把当前系统的任务需求与资源健康结合起来考虑，为未来的健康管理优化工作做了一个铺垫。

参考文献

[1] 赵宁社, 翟正军, 王国庆. 新一代航空电子综合化及预测与健康管理技术[J]. 测控技术, 2011, 30(1): 1-5.

[2] 褚文奎, 张凤鸣, 樊晓光. 综合模块化航空电子系统软件体系结构综述[J]. 航空学报, 2009, 30(10): 1912-1917.

[3] Zhang Shunong, Kang Rui, He Xiaofei, et al. China's Efforts in Prognostics and Health Management[J]. IEEE Trans. on Components and Packaging Technologies, 2008, 31(2): 509-518.

[4] 张 莉, 袁海文, 王秋生. ATML 在综合运载器健康管理系统中的应用[J]. 计算机工程, 2008, 34(12): 215-217.

[5] 李宁波, 汪 溢, 俞 慎, 等. 嵌入式 RTOS 健康监控技术研究[J]. 计算机工程, 2009, 35(3): 260-262.

编辑 顾姣健