3D 游戏中“刀光剑影”特效的实现算法①

张　鹏，陈芝荣，安　峰

(苏州工业园区服务外包职业学院 信息技术系，苏州 215123)

摘　要：“拖影”，就是指 3D 游戏中，冷兵器在舞动时所“拖”出的光影。分析了“拖影”特效的图形变化规律，提出了一种实用的“拖影”特效实现算法。借助该算法，应用程序可以根据兵器的运动轨迹，自动产生“拖影”效果。最后，还结合实际应用的经验，对算法进行了改进。这种算法已成功应用于多款 3D 游戏，并很好地与各种程序引擎相结合。

关键词：拖影；轨迹；3D 游戏；实现算法；采样；自动“回收”

Algorithm of "Sword’s Trail" Effect in 3D Games

ZHANG Peng, CHEN Zhi-Rong, AN Feng

(Information Technology Department, Suzhou Industrial Part Institute of Services Outsourcing, Suzhou 215123, China)

Abstract: “Sword's Trail” means the “lighting effects” when weapons like swords are waving in 3D games. This article analyzes the rules of the image’s changes for “Sword’s Trail’ effect, and hands out an practical algorithm of the effect mentioned above. With this algorithm, the application can automatically produce the “Sword’s Trail” effect of weapons. In combination with real application experience, this article also proposes an ameliorative method of the algorithm. The algorithm can combine with a variety of game engine, and has been applied successfully in a lot of 3D games.

Key words: Sword's Trail; track; 3D game; algorithm; sampling; automatic “recovery”

“拖影”,就是指 3D 游戏中,兵器在舞动时所“拖”出的光影，如图 1 所示。这些“拖影”会增加兵器的魅力，同时也使得打斗场面更加气势磅礴。

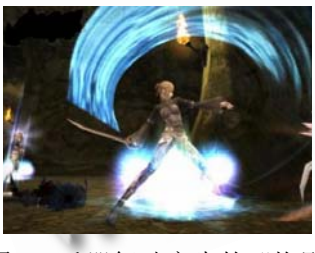

图 1　兵器舞动产生的“拖影”

1　引言

1.1 文章安排

本文第 2 节介绍“拖影”特效的图形变化规律。第 3 节给出“拖影”特效的实现算法。第 4 节根据实际应用经验，提出了算法的改进方案。

1.1.1 基本介绍

制作游戏时，实现“拖影”的方法有两种：一种是由美术员直接制作出与人物动作相匹配的“拖影”模型；另一种则是在游戏运行时，由程序自动计算并显示“拖影”。

显然，第一种方法很不实用。因为游戏中各种人物的动作招式数不胜数，如果招招都需要舞出光影，那会增加大量的美术工作。而且当调整动作招式后，还必须重新制作与新动作相配的“拖影”模型。

绝大多数游戏开发团队都会选择第二种“拖影”特效的制作方法。

2　“拖影”的变化规律

由程序自动计算并显示“拖影”的原理是：首先由程序计算出兵器的舞动轨迹，然后再由兵器的轨迹产生“拖影”模型，最后将事先制作的“拖影”纹理

① 收稿时间:2010-11-05;收到修改稿时间:2010-12-01

平面图“粘贴”到“拖影”模型上。“拖影”的实际变化规律如下：

2.1 “拖影”的产生

当兵器舞动的速度 V 超过 Vmin（产生“拖影”所需的兵器最小运动速度）时，就会拖出光影轨迹，如图 2 所示。

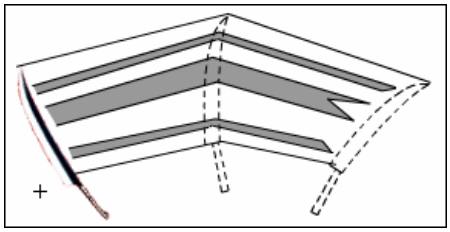

图 2 “拖影”的产生

2.2 “拖影”的跟随

如果兵器继续快速舞动，当实际应该产生的“拖影”长度超过纹理平面图的长度时，可使“拖影”跟随兵器进行移动，此时的“拖影”就像是系在兵器上的绸带，如图 3 所示。

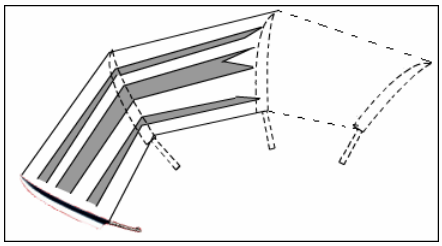

图 3 “拖影”的跟随

2.3 “拖影”的消失

当兵器停止运动时，“拖影”将会自动“回收”，直至消失，如图 4 所示。

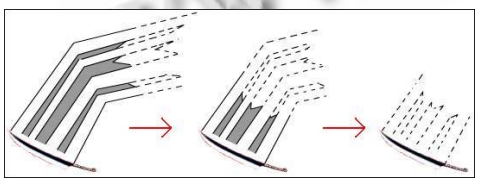

图 4 “拖影”的消失

这里将“拖影”的自动“回收”速度记为 Vmin。如果兵器仍在舞动，但舞动速度很慢，则“拖影”的实际“回收”速度为 Vmin ? V，其中 V 是兵器的运动速度。

3 “拖影”的实现算法

有多种算法可以实现“拖影”特效，实现“拖影”的程序算法很关键，简单算法的模拟效果不够理想，而算法过于复杂又会加重游戏的运行负担。这里介绍一种“拖影”特效的实用算法，该算法能很好地与各种游戏引擎相结合。

在编写程序之前，需要先由美工制作一张漂亮的“拖影”纹理平面图，然后通过下面介绍的算法来实现“拖影”特效。

3.1 设置“拖影”生成点

首先，美术员在制作兵器模型的同时，还要设定两个“拖影”的生成点，如图 5 所示。兵器舞动时，只有两个生成点之间的部分才可以拖出光影。

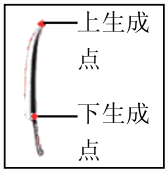

图 5 轨迹生成点

3.2 获取兵器运动轨迹

尽管各种三维游戏引擎的内部原理不同，但可以肯定的是，在游戏运行时，都能够通过引擎程序获取兵器“拖影”生成点的实际位置。

游戏运行时，每隔一定时间对“拖影”生成点的位置进行采样，就能获得一系列的兵器运动轨迹点。采样的时间间隔一般以 50 毫秒为佳，间隔时间过短会增加系统计算负担，间隔时间过长则无法获得正确的兵器运动轨迹。

3.3 产生“拖影”

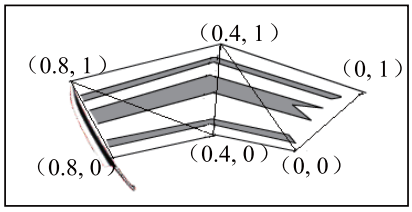

图 6 轨迹的三角形带

获得兵器运动的轨迹顶点后，可将这些顶点连成带状的轨迹平面。然后在系统刷新每帧图像之前，计算轨迹面上各顶点的纹理坐标，再将事先制作的纹理平面图“贴”到轨迹面上，就可以产生“拖影”，如图

6 所示。

轨迹面中，纹理坐标的 S（横坐标）值为 0 的一对顶点是最初的“拖影”生成点，这里记作“拖影原点”。同时将位于当前兵器上的顶点记作“兵器原点”。

通常情况下，轨迹面中每对顶点的纹理坐标应该分别为：

(L / LEN，0)、(L / LEN，1)

其中 LEN 是纹理平面图的长度，而 L 则是当前顶点到“拖影原点”的轨迹长度。如果将“拖影原点”定义为第 0 对轨迹顶点，则顺次排列的第 N 对轨迹顶点到“拖影原点”的轨迹长度（沿轨迹面的长度）为：$L = L_N + L_{N-1} + \cdots\cdots + L_1$ 。公式中，L_N 是第 N 对轨迹点到第 N-1 对轨迹点的轨迹长度，如图 7 所示。

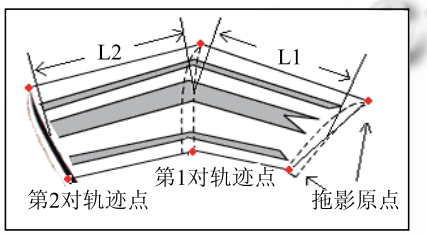

图 7　轨迹的拖出

3.4 “拖影”的跟随

当 L 大于 LEN 时，即实际“拖影”长度已经超过了纹理平面图的长度，为了使纹理正常显示，可令整个“拖影”随着兵器向前移动。此时，每对轨迹顶点的纹理坐标应分别为：

(1- L_r / LEN，0)、(1- L_r / LEN，1)

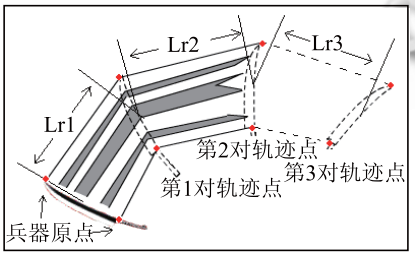

图 8　轨迹的跟随

其中 Lr 是当前顶点到“兵器原点”的轨迹长度。此情况下，各顶点的排列序号与前面情况刚好相反，第 N 对顶点到“兵器原点”的轨迹长度为：$L = Lr_N + Lr_{N-1} + \cdots\cdots + Lr_1$。公式中，$Lr_N$ 是第 N 对顶点到第 N-1 对顶点的轨迹长度，如图 8 所示。

3.5 “拖影”的自动“回收”

在刷新每帧图像之前，除了进行产生“拖影”的计算外，还要对已产生的“拖影”进行“回收”处理。具体的“回收”方法是：将当前轨迹面中各顶点的纹理 S 坐标（横坐标）减去 L_{BACK}，L_{BACK} 是“拖影”的“回收”长度，且：

$L_{BACK} = (T \times Vmin)/LEN$

其中 T 是前后两帧的间隔时间，Vmin 是“拖影”的自动“回收”速度。

3.6 删除无用的轨迹点

经过“产生”与“回收”两次运算后，轨迹面上会产生一些无用三角形，这些三角形的所有顶点的纹理坐标都小于 0 或大于 1，所以该三角形面不会产生任何“贴”图。此时可在轨迹面上删除这些顶点，以节省系统资源。

4　实现算法的改进

在理想情况下，兵器舞动出的“拖影”轨迹应该平滑连续，如图 9 所示。

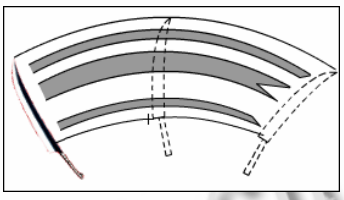

图 9　平滑的“拖影”

理论上，按照本文前面所讲述的方法，只要采样时间间隔足够短，就可以达到图 9 所示的效果。

然而在实际中，轨迹采样时间间隔往往不能太短，否则会影响系统的正常运行。这种情况下，对前面的算法稍加改进，仍然可以实现很理想的效果。具体的改进方法如下：

1) 将临近的轨迹顶点相连，形成若干段带状轨迹面。对每段轨迹面的上下两边进行 n 等分（图 9 中，n=4），产生 n-1 对等分点。

2) 将上下对应的等分点连线，以轨迹面中较短的边（图 9 中的下边）上的等分点为基础，在等分点连线的方向上调整另外一个等分点的位置，使一对等分点的距离等于兵器的长度。

3) 将调整后的等分点插入轨迹顶点序列，新轨迹

(下转第 243 页)

Karlsruhe: University of Karlsruhe, 2004.
2 张子振,王存刚.本体进化关键技术研究.科技信息(学术版),2008,(8):202–204.
3 Sure Y, Tempich C. State of the art in ontology engineering methodologies. Karlsruhe: Semantic Knowledge Technologies (SEKT), 2004.
4 鲍爱华,姚莉,刘芳,张维明.本体变化管理技术研究综述.计算机科学,2007,34(9):151–155.
5 Stojanovic L, Maedche A, Stojanovic N, Motik B. User-driven ontology evolution management. Benjamins R V, eds. Proc. of the 13th European Conference on Knowledge Engineering and Knowledge Management. Madrid, Spain: Springer-Verlag, 2002. 285–300.
6 Castano S, Ferrara A, Hess NG. Discovery-driven ontology evolution. Pisa: Scuola Normale Superiore, 2006.
7 Plessers P, Troyer O D. Ontology change detection using a versioning log. Yolanda Gi, eds. Proc. of the 4th International Semantic Web Conference. Ireland: Springer-Verlag, 2005. 578–592.
8 Javed M, Yalemisew M, Abgaz, Pahl C. A pattern-based framework of change operators for ontology evolution. In: Shvaiko P, eds. The 4th International Workshop on Ontology Content. Algarve: OTM Workshops, 2009. 544–553.
9 Musen AM, Chugh A, Liu W, Noy FN. A framework for ontology evolution in collaborative environments. In: Isabel F, eds. Proc. of the 5th International Semantic Web Conference. Heidelberg: Springer-Verlag, 2006.544–558.
10 鲍爱华,姚莉,张维明.基于变化生成图的 OWL 本体协同进化方法研究.计算机科学,2007,34(3):186–191.
11 刘晨,韩燕波,陈旺虎,王建武.MINI——一种可减小变更影响范围的本体演化算法.计算机学报,2008,31(5):711–720.
12 宋奂寰,张志祥.基于概念代数的本体演化.计算机工程,2009,35(13):23–25.
13 鲍爱华,张维明,袁金平,姚莉.一种基于规则的 OWL_S 本体语法一致性维护方法.国防科技大学学报,2009,31(3):97–103.
14 何扬帆,何克清.一种支持可靠语义互操作的本体演化管理框架.计算机工程,2007,33(18):26–30.
15 常春.大型 Ontology 构建工具 KAON 的使用和评价.现代图书情报技术,2004,20(8):14–17.

(上接第 194 页)

顶点序列所构成的轨迹平面将变得更加平滑，如图 10 所示。

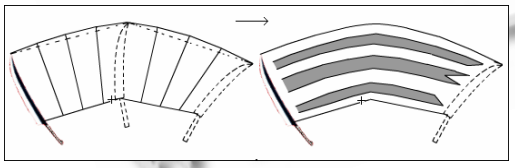

图 10 改进算法后的效果

以上便是“拖影”特效的全部算法，这种算法的计算量不大，但实际的显示效果却十分理想。

参考文献

1 Lake A. Game Programming Gems 8. America: Course Technology PT, 2010: 140–203.
2 Jacobs S. Game Programming Gems 7. America: Charles River Media, 2008: 255–336.
3 Walsh P. Advanced 3D Game Programming with DirectX 9.0. Texas: Wordware Publishing, 2003: 413–481.
4 黄蓝枭.游戏引擎中特效的实时渲染技术研究[硕士学位论文].成都:电子科技大学,2008.
5 赵明.基于 DirectX 的三维游戏特效技术的研究与实现[硕士学位论文].哈尔滨:哈尔滨工程大学,2009.