

文章编号:1001-9081(2006)02-0500-02

## 基于广义包围盒的交互操作在 Vega 环境中的应用

肖书立,李世其,王峻峰

(华中科技大学 机械学院,湖北 武汉 430074)

(xiaoshuli@163.com)

**摘 要:**介绍了视景仿真软件 Vega 所具备的场景基本操作功能,针对 Vega 在交互操作功能方面的不足,提出了一种广义包围盒。介绍了广义包围盒的实现原理,定义了包围盒的数据结构,给出了获取方法,然后基于 Vega API,通过两种不同方法建立了场景模型广义包围盒。基于广义包围盒,对场景模型直接进行操作,从而实现了 Vega 虚拟场景中的模型拾取、碰撞检测等仿真交互操作。给出了模型拾取、碰撞检测的具体实现算法,并成功应用在一个视景仿真程序中。

**关键词:**广义包围盒;交互操作;模型拾取;碰撞检测

**中图分类号:** TP391.9 **文献标识码:** A

## Application of inter-operation based on generalized bounding box in Vega

XIAO Shu-li, LI Shi-qi, WANG Jun-feng

(School of Mechanical Science & Engineering, Huazhong University of Science & Technology, Wuhan Hubei 430074, China)

**Abstract:** The common scene inter-operations provided by Vega were briefly introduced, with regard to its drawbacks, a generalized bounding box (GBB) was bring forward. The principle of GBB based on AABB and Sphere was given, its data structure was defined and the construct methods were presented. By using Vega API, GBB of the model was established. Based on GBB, several inter-operation methods in Vega virtual scene were put forward, such as model-picking and collision - detecting. The model-picking and collision-detecting algorithms were realized, and were successfully applied in a virtual scene simulation system.

**Key words:** generalized bounding box; inter-operation; mode-picking; collision-detecting

## 0 引言

作为一款主要用于实时视景仿真的优秀仿真软件,Vega 具有强大的场景操作和仿真功能,可以迅速地创建各种实时交互的三维虚拟仿真环境。但是 Vega 由 SGI Performer 移植开发而来,很多功能受到底层软件功能的限制。特别在场景操作方面,尽管 Vega 提供了众多交互操作方法,但在处理碰撞检测等一些复杂场景操作时,就难以胜任。

本文在分析 Vega 场景操作功能的基础上,针对其不足,提出了模型广义包围盒概念。基于广义包围盒,实现了虚拟场景中模型拾取和碰撞检测等交互操作。

## 1 Vega 虚拟场景操作方法简介

在一个典型的 Vega 场景中,必须具备的场景节点有 System、Scene、Object、Observer、Motion、Player 等。对于每种场景节点,Vega 都提供了多种控制方式,因此拥有强大的场景操作功能。Vega 常见场景操作功能有漫游方式控制、碰撞检测、模型拾取等。

但在基于 Vega 的软件二次开发过程中,也发现 Vega 在场景操作方面功能的一些不足。如拾取物体时,Vega 提供的 vgPicker 类只适用于动态坐标系物体;在处理碰撞检测时,其碰撞检测方法则计算量大,不能满足仿真实时性要求。结合开发过程中所遇到的问题和解决方法,本文提出基于模型广义包围盒的场景交互操作方法。

## 2 模型广义包围盒的建立

### 2.1 广义包围盒

通常一个几何对象的包围盒指包含该对象的一个简单几何体,它可以近似代替几何对象进行一些粗略的操作。比较常见的包围盒有沿坐标轴的包围盒 AABB (Axis-Aligned Bounding-Boxes)、包围球 (spheres)、方向包围盒 OBB (Oriented Bounding Boxes)等<sup>[1]</sup>。

针对虚拟场景,基于常见包围盒,这里定义一种包围盒,该包围盒同时包含 AABB 和包围球,不妨称为广义包围盒 (Generalized Bounding Boxes, GBB)。通常一个对象的 AABB 被定义为包含该对象且各边平行于坐标轴的最小的六面体,由组成对象的基本几何元素中各元素顶点的 x、y、z 坐标的最大值和最小值决定。包围球则由几何对象的形心和广义半径 (对象表面上距离形心最远的点)决定。在得到几何对象的 AABB 和包围球后,就可以建立其广义包围盒。

### 2.2 GBB 数据结构定义

根据定义,设定 GBB 的数据结构如下:

```
typedef struct gbox{
    Vector3Df Min;
    Vector3Df Max;
    Vector3Df Center;
    float Radius;
} GBOX;
```

收稿日期:2005-08-07;修订日期:2005-10-27 基金项目:国家 863 计划项目(2004AA804022)

作者简介:肖书立(1980-),男,湖南邵阳人,硕士研究生,主要研究方向:虚拟现实技术;李世其(1965-),男,江西人,教授,博士生导师,主要研究方向:计算机辅助工程、虚拟样机技术;王峻峰(1970-),男,湖北人,讲师,博士,主要研究方向:并行工程、协同设计。

其中  $\text{Vector3Df}$  为空间三维向量,  $\text{Min}$ ,  $\text{Max}$  分别为 GBB 中 AABB 的最小、最大坐标点,  $\text{Center}$ ,  $\text{Radius}$  分别为 GBB 中包围球的球心位置、球半径。

### 2.3 模型 GBB 的建立

由 GBB 的定义可知,建立 GBB,需先获取模型的 AABB 和包围球。由于 Vega 基于 Performer 开发,支持 Performer 大部分操作功能,因此 GBB 的实现有两种途径,一种是使用底层的 Performer API,另外一种是采用 Vega API 实现。考虑到操作的通用性,这里采用 Vega API。GBB 的建立流程如下:先创建一个空体盒对象,再分别设置体盒的类型为球体和立方体,获取模型包围球和 AABB,从而建立模型 GBB。主要相关代码如下:

```
GBBOX CSimLaser::GetGBBox (vgObject *obj)
{
    GBox gbox;                                //包围盒
    vgBox vbox;                                //AABB 对象
    vgSphere sphere;                            //包围球,
    vgVolume *vol1, *vol2; vol1 = vgNewVol();    //创建一个新体对象

    vol2 = vgNewVol();
    //包围球的获取
    vgProp(vol1, VGVOL_TYPE, VOL_SPHERE);
    vgUpdate(vol1);                            //更新包围盒
    vol1 = vgGetObjVol(obj);                    //获取对象体盒
    vgGetVolSphere(vol1, &sphere);              //获取包围球
    gbox.Center = sphere.center;                //球心
    gbox.Radius = sphere.radius;                //球半径
    //AABB 的获取
    vgProp(vol2, VGVOL_TYPE, VGVOL_BOX);
    vgUpdate(vol2);
    vol2 = vgGetObjVol(obj);
    vgGetVolBox(vol2, &vbox);                  //获取包围盒
    gbox.Max = vbox.max;                        //最大坐标点
    gbox.Min = vbox.min;                        //最小坐标点
    return gbox;                                //返回 GBB
}
```

## 3 基于 GBB 的虚拟场景交互操作应用

模型广义包围盒建立后,在虚拟场景中就可以通过包围盒间接对模型进行场景相关交互操作。这里主要介绍该方法在模型拾取和碰撞检测等交互操作方面的应用。

### 3.1 模型拾取

在拾取场景物体时,Vega 提供了  $\text{vgPicker}$  类进行操作。其使用流程一般为:先创建一个  $\text{vgPicker}$  对象,再调用该对象拾取模型。通过设置  $\text{vgPicker}$  对象相关属性可以拾取不同类型的物体,它支持的选择对象主要有  $\text{object}$ 、 $\text{part}$ 、 $\text{geode}$ 、 $\text{deset}$ 、 $\text{player}$  等几种类型。

通过  $\text{vgPicker}$  可以实现普通的拾取功能,但通常只适用于动态坐标系模型。在 Vega 中,模型坐标系有动态坐标系和静态坐标系之分,并且一旦设置好,之后在程序中就不能更改。而 Vega 提供的许多操作功能,如物体拾取等,都是针对动态坐标系模型有效。对于静态物体,使用  $\text{vgPicker}$  只能间接获取模型所在地形数据库  $\text{vgDataSet}$  节点等,不能获取对象本身  $\text{vgObject}$  节点。而在 Vega 中,对物体的操作往往是通过物体本身所在场景节点进行的,因此使用  $\text{vgPicker}$  不能操作静态模型,这时可以模型广义包围盒来完成。在虚拟场景中,基于广义包围盒 GBB 的模型拾取算法如下:

- 1) 遍历场景模型,建立场景模型 GBB 链表数据结构;
- 2) 获取场景世界中鼠标指针位置;
- 3) 依次判断鼠标位置与包围盒链表中各个包围盒的关系,即点是否在包围盒内,若在其中,则已选中模型,转 5);否则未选中,转 4);
- 4) 继续移动鼠标,作新的判断,转 2);
- 5) 当选中模型时,高亮显示该模型,记录当前选中对象,以进行其他操作。

在 2) 中,获取鼠标在场景世界坐标系中的位置有两种方法。一是先获取鼠标的屏幕坐标,再利用  $\text{vgGetChanScreenToWorld}()$  等 Vega 函数转换为 Vega 场景世界坐标系;二是使用  $\text{vgPicker}$  提供的功能实时获取鼠标位置。由于第一种方法精确度不高,因此本文采用了第二种方法。

### 3.2 碰撞检测

在 Vega 中,提供了 XYZPR、LOS、Hat、Bump、Tripod、Volume 等多种碰撞检测方式,都是基于地形掩码匹配原理实现的。即当模型对象的掩码值与地形或者其他对象物的掩码值的交集不为空时,即认为发生碰撞。各种碰撞检测方式适用于不同场合,可以满足一般场景漫游仿真需要。

然而这些方法计算量大,消耗资源厉害,严重影响仿真效果,不能满足实时性要求,因此本文使用基于模型广义包围盒的碰撞检测方法。在虚拟场景中,基于模型广义包围盒体的碰撞检测算法可以描述如下:

- 1) 遍历场景模型,建立场景模型 GBB 链表数据结构;
- 2) 获取当前运动模型 GBB;
- 3) 将当前活动模型 GBB 与链表(自身除外)中的模型 GBB 依次进行比较,判断彼此间的位置关系,确定是否发生碰撞;
- 4) 若碰撞,则记录当前碰撞模型,改变模型颜色,前进终止,转 5);否则继续前进,转 2);
- 5) 改变运动方向,在新方向运动,转 2);

对于 3),判断两个模型广义包围盒间的位置关系,有好几种情形。考虑到检测速度和简易程度,首先利用 GBB 中的包围球来判断。计算两个模型广义包围盒形心之间的距离,再与包围盒半径之和相比较,若前者小于后者,则碰撞,否则不发生碰撞。但是这种方法对于形状不规则物体,通常误差较大,因此使用最多的是利用对象 GBB 中的 AABB。

基于 GBB 的 AABB 方法相交检测原理如下:假设两个模型的 AABB 如图 1 所示,利用模型 AABB 在 X、Y、Z 轴上投影的重叠区域来判别它们是否相交。只要两个包围盒在某个方向上的投影不重叠,就可以判定它们不相交。如果它们在所有方向上的投影都重叠,则认为它们相交,因此最多需要六次比较运算,效率较高。

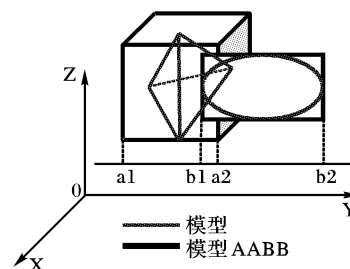


图1 模型 AABB 相交测试

设长方体A、B的边长分别为 $\{a, b, c\}$ 、 $\{d, e, f\}$ ,形心  
(下转第504页)

适合程度的优度值  $T_i$ , 其中  $T_i = \sum_{k=1}^m w_k \cdot s_i^k$

然后按照  $T_i$  的大小排序, 即如果有  $T_{i_1} \geq T_{i_2} \geq \dots \geq T_{i_n}$ , 则机器人对角色承担适合度大小排序  $robot_{i_1} \geq robot_{i_2} \geq \dots \geq robot_{i_n}$ , 从而可以确定各个机器人对某一角色的适合度, 选择最适合承担此角色的机器人, 并将这一角色分配给  $robot_{i_1}$ , 完成对角色集合中某一个具体角色的分配。

#### 4 仿真实例

以 FIRA(5VS5) 机器人足球比赛作为实验。现假设我方处于左半场, 处于进攻态势, 某一周期时选择队形为进攻队形, 角色集合  $ROLE = \{\text{守门员, 主前锋, 协前锋, 中卫, 后卫}\}$ , 机器人集合  $ROBOT = \{robot_0, robot_1, robot_2, robot_3, robot_4\}$ 。从场地传来环境信息, 包括  $ROBOT_i$  的位置  $(X_i, Y_i)$  与角度  $(A_i)$ , 表示成  $robot_i = \{(X_i, Y_i, A_i), i = 0, 1, 2, 3, 4\}^T$ 。Ball 的位置  $Ball(X_b, Y_b)$ 。环境信息表示成  $E = \{ROBOT_0, ROBOT_1, ROBOT_2, ROBOT_3, ROBOT_4, Ball\}^T$ 。

从实验场地获得以下信息:

$ROBOT_0$ : (10.661302, 42.274818, 54.025784)

$ROBOT_1$ : (33.339508, 53.563919, 0.642543)

$ROBOT_2$ : (27.546658, 28.819582, 11.145736)

$ROBOT_3$ : (56.887867, 64.385658, 9.513366)

$ROBOT_4$ : (51.028740, 35.811844, 38.507683)

$BALL$ : (66.203316, 59.916004)

选择四个评价因子:

$O_1$ : 机器人的位置;

$O_2$ : 机器人距离球的距离;

$O_3$ : 机器人的姿态 (即机器人的方向与机器人和球的连线的夹角);

$O_4$ : 机器人与球之间是否有障碍物;

依据本文提出的角色分配系统建立 FIRA 的角色分配系统, 将环境信息预处理成能够由评价因子比较的信息, 经过角

色分配系统的综合决策, 对机器人球队进行如下角色分配:  $robot_0 \rightarrow$  守门员,  $robot_1 \rightarrow$  协前锋,  $robot_2 \rightarrow$  后卫,  $robot_3 \rightarrow$  主前锋,  $robot_4 \rightarrow$  中卫, 可以看到仿真结果与预想的结果一致。在 FIRA 仿真系统中运用了本角色分配系统的队, 成绩有明显提高, 机器人跑位灵活, 抢球准确、及时, 出现了不少配合进球。

#### 5 结语

本文通过引进模糊一致关系, 建立基于模糊一致关系的机器人角色分配系统, 分配时综合考虑了比赛现场各方面的因素, 得到的分配结果与预想结果的符合度很高, 且角色分配算法随队形的变化是动态的, 提高了机器人球队的灵活性。本文中模型的建立依赖于手工确定, 不能根据比赛自主学习, 在实际设计角色分配系统时决策因素的选择也依据经验, 下一步要增加球队自主学习的能力。

#### 参考文献:

- [1] 洪炳熔. 机器人足球比赛——发展人工智能的里程碑[J]. 电子世界, 2000, 247(4): 4-5.
- [2] 吴丽娟, 张春晖, 徐心和. 足球机器人决策系统推理模型[J]. 东北大学学报(自然科学版), 2001, 22(6): 597-599.
- [3] 徐心和. 足球机器人六步推理模型研究[A]. 机器人足球研讨班论文集[C]. 东北大学, 沈阳: 东北大学出版社, 1998. 42-46.
- [4] ASADA M, KITANO H. The Robocup challenge[J]. Robotics and Autonomous System, 1999, 29(1): 3-12.
- [5] 吴丽娟, 翟玉人, 徐心和. 足球机器人系统中角色分配策略的设计[J]. 基础自动化, 2000, 7(1): 4-6.
- [6] 姚敏. 计算机模糊信息处理技术[M]. 上海: 上海科学技术文献出版社, 1996.
- [7] 柳长安, 刘春阳, 李国栋. 基于模糊综合决策的足球机器人策略子系统[J]. 哈尔滨工业大学学报, 2004, 36(7): 857, 858, 873.
- [8] 符海东, 雷大江. 基于 Vague 集的机器人足球比赛策略[J]. 控制理论与应用, 2004, 21(S): 24-28.

(上接第 501 页)

坐标分别为  $(x1, y1, z1)$ 、 $(x2, y2, z2)$ 。如果 A、B 间满足以下关系式:

$$\begin{aligned} &(|x1 - x2| < (a + d)/2) \&\& (|y1 - y2| < (b + e)/2) \\ &\&\& (|z1 - z2| < (c + f)/2) \end{aligned} \quad (1)$$

则这两个长方体必然发生碰撞。

在本算法中, 由于已获取当前物体和目标物体的广义包围盒, 不妨设当前物体的最大最小坐标点分别为 SrcMax、SrcMin, 目标物体的最大最小坐标值分别为 DestMax、DestMin, 则当前对象的边长为  $\{SrcMax.x - SrcMin.x, SrcMax.y - SrcMin.y, SrcMax.z - SrcMin.z\}$ , 目标对象的边长为  $\{DestMax.x - DestMin.x, DestMax.y - DestMin.y, DestMax.z - DestMin.z\}$ , 再把这些值代入 (1) 式则可以判断当前物体与目标物体之间的位置关系。

#### 3.3 交互操作运行实例

在已开发的一个仿真程序中, 应用该方法进行物体拾取和模型碰撞检测, 操作判断准确。图 2 是仿真程序运行时仿真界面的一个局部截图, 红色外框表示管道物体已经被选中, 红色和绿色表示小球通过管道物体时二者相碰撞。

#### 4 结语

本文针对 Vega 在场景操作功能方面的局限性, 定义了一种广义包围盒, 提出基于广义包围盒的场景交互操作方法, 并将该方法应用在场景物体拾取和碰撞检测等场合, 解决了实际问题。该算法可以推广应用在虚拟场景操作其他场合。

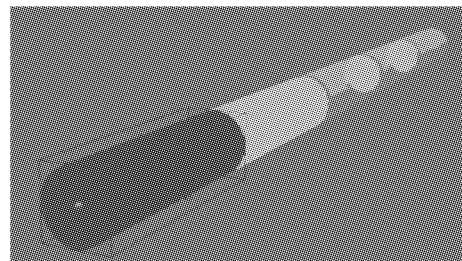


图2 程序运行截图

#### 参考文献:

- [1] 章勤, 黄琨, 李光明. 一种基于 OBB 的碰撞检测算法的改进[J]. 华中科技大学学报(自然科学版), 2003, 31(1): 46-48.
- [2] 魏迎梅, 吴泉源, 石教英. 虚拟环境中的碰撞检测方法[J]. 计算机工程与科学, 2001, 23(2): 44-47.
- [3] Vega Programmer's Guide. Version 3.7 for Windows [Z]. U. S. A: MultiGen-Paradigm Inc, March 2001.