YHVRP中的复杂行为建模*

王勇军 李思昆 胡守仁

(国防科技大学计算机系 长沙 410073)

摘 要 提出了一种基于复杂行为的语言模型对虚拟环境尤其是实体行为进行建模，该模型提供较强的定义行为关系及其时间约束的能力。利用自行研制的分布式虚拟环境软件开发平台 YHVRP的建模语言 YHVML实现了该模型，同时 YHVML语言的运行环境实现了面向复杂行为的虚拟环境计算模型，并利用 LLF策略对实体行为进行动态调度，以保障面向复杂行为的虚拟环境应用的实时交互性和真实感。

关键词 虚拟环境，行为建模，复杂行为，真实感

分类号 TP311

Complex Behavior Modeling in YHVRP

Wang Yongjun Li Sikun Hu Shouren

(Department of Computer, NUDT, Changsha, 410073)

Abstract we propose a language model of behavior to define complex behaviors relations and its time constraints. YHVML, the modeling language of distributed VR software development platform YHVRP, implements the model, whose run-time environment implements the VE Computing Model for complex behaviors and adopts LLF method to schedule behaviors dynamically so as to keep the real-time interactivity and fidelity of VR application.

Key words Virtual Environment, behavior modeling, complex behavior, fidelity

行为建模[1]是虚拟环境中的重要组成部分，随着虚拟现实技术的不断发展，为了将现实世界丰富多采的真实特性在虚拟环境中表现出来，有必要把各种复杂的行为嵌入虚拟环境，行为建模及其计算在虚拟环境计算的比例将呈上升趋势。因此，加强对虚拟环境建模语言的行为建模能力也是非常有意义的。虚拟环境软件开发平台对行为建模的语言支持包括两方面：一是语言的表示和描述能力，另一方面是语言实现的运行环境支持。语言的表示和描述能力包括能较灵活方便地定义复杂行为及其之间的控制关系和行为执行的时间约束；而运行环境的支持是指对程序调试和运行性能保障的支持。

为了适应这一需求，本文提出一个对复杂行为进行一致性描述的语言模型及面向复杂行为的虚拟环境计算模型，并设计了相应的语言 YHVML。另外，我们还研制了分布式虚拟环境软件开发平台 YHVRP，该平台基于当前通用的硬件平台 (PC、工作站)、软件平台 (NT, Unix) 和高速以太网 (100Mbps以上) 环境。YHVRP实现了面向复杂行为的虚拟环境计算模型，YHVML运行环境较好地支持了虚拟环境的性能。

1 面向复杂行为的虚拟环境计算

1.1 复杂行为的语言模型

从面向对象的观点来看，虚拟环境是由若干虚拟实体组成，虚拟实体则由若干行为和局部属性组成。我们从语言模型的角度对复杂行为进行形式上的定义。

* 1998年 4月 6日收稿
第一作者：王勇军，男，1971年生，博士生

定义 **1** 虚拟环境 虚拟环境 VE由 N 个虚拟实体组成, 可记为:

$VE = \{Entity_i \mid i = 1, \cdots, N\}$

定义 **2** 虚拟实体 虚拟实体 Entity可以记为三元组:

$Entity ::= < EntityName, AttribSet, BehSet >$

其中: EntityName是 Entity名;

AttribSet是 Entity的属性集合;

BehSet是 Entity的复杂行为集合。

定义 **3** 属性 属性 Attribute是实体属性集合的成员, 可以记为三元组:

$Attribute ::= < EntityName, AttrName, AttrType >$,

其中: EntityName是 Attribute所属的实体名字;

AttrName是 Attribute名字;

AttrType是 Attribute的类型。

定义 **4** 复杂行为 复杂行为 Beh是当某触发条件满足时所激活的处理程序, 可以记为六元组:

$Beh ::= < EntityName, BehName, BehType, TrigCond, BehBody, ConstraintTime >$,

其中: Entity是 Beh所属的实体名字;

BehName是 Beh的名字;

BehType指明 Beh是否可被其它实体调用, 即 $BehType \in$ (External, Internal);

TrigCond是 Beh的触发条件;

BehBody是 Beh的行为体;

ConstrainTime是显式声明的时间约束。

为了方便起见, 在下文中, 如果不是特别需要, 我们仍将复杂行为称为行为。

从行为的性质来看, 实体的行为可以分为两类: 一类是一般行为 (GeneralBeh), 所谓一般行为是对实体属性进行具体操作的行为; 另一类行为是冲突消解行为 (ResolutionBeh), 行为冲突通常指因以实体属性为单位的冲突而引起的实体内多个行为不协调, 或因以实体为单位的冲突而引起的实体间多个行为的不协调。而冲突消解行为相当于多个行为冲突时的仲裁处理程序, 它也是描述实体间交互的基本支持。用户可显式提供消解冲突行为的方法, 主要包括对行为状态进行直接设置或调整其运行次序或结构。冲突消解行为是高于一般行为之上的高级行为。这二者靠行为的触发条件来区分。

定义 **5** 触发条件 触发条件 TrigCond包括一般事件 (GeneralEvent) 和行为冲突事件 (BehColEvent), 可记为:

$TrigCond ::= GeneralEvent \mid BehColEvent$

定义 **6** 一般事件 一般事件 GeneralEvent是一般行为的触发条件, 包括系统定义事件 (SystemDefinedEvent)、自定义事件 (UserDefined Event)。所谓系统定义事件包括时钟事件 (Clock Event)、外设产生的消息事件 (I/O Event)、碰撞事件 (Collision Event) 和行为调用事件 (BehCall Event) 等。而自定义事件主要指用户定义的条件设置程序 (CondSetProc), 即一个条件判断语句的值为真时产生的事件。

定义 **7** 行为冲突事件 行为冲突事件 BehColEvent是冲突消解行为的触发条件, 可用一个行为集合来表示, 若集合中行为都处于活跃状态, 则冲突事件发生。集合中的行为可是同一个实体的行为, 也可以是不同实体的行为。

定义 **8** 行为体 行为体 BehBody体现了行为执行体层次表示结构, 可以记为四元组:

$BehBody ::= < InputAttrSet, OutputAttrSet, ParameterSet, ProcessingProc >$

其中, 处理过程 ProcessingProc是行为计算程序;

输入属性集合 InputAttrSet是行为计算所需的实体属性;

输出属性集合 InputAttrSet是行为计算进行修改的实体 (包括其它实体) 属性;

参数集合 ParameterSet是行为计算所需的参数。

定义 **9** 行为计算程序 行为计算程序是用类 C 语言编写的处理过程, 除了通常的类 C 语法外, 还有对行为状态的设置函数 (SetBehState( )), 以及并行行为调用函数 (PBehCall( )), 设置锁函数等。

定义 **10** 并行行为调用函数 并行行为调用函数用于对一个或一系列行为的调用。当对一系列行为进行调用时, 指同时激活这些行为。只有当这些行为都执行完后, 当前行为才继续执行。该函数的调用形式为: PBehCall (BehSet), 其中, BehSet是一系列行为的集合。

并行行为调用函数的引入增强了对行为之间控制关系的描述能力。

从行为体的语法描述上, 复杂行为可分为两类, 一种是原子行为 (AtomBeh), 即不调用其它行为的行为, 对应于层次表示结构中的简单行为执行体。一种是复合行为 (CombinedBeh), 即是调用了其它行为的行为, 对应于层次表示结构中的复合行为执行体。我们称被调用的行为为子行为 (SubBeh), 调用子行为的行为称为父行为 (CalleeBeh)。复杂行为对行为间控制关系及时序关系的描述能力很强, 对用户而言, 还有利于行为代码的共享。

定义 **11** 调用子行为集合 复杂行为 Beh调用的子行为集合记为 SubBehSet (Beh), 该集合包括行为 Beh自身。

根据所操作属性的功能, 一般行为又可划分为三种行为: 一是表现属性行为 (PresentationBeh), 即仅对实体表现属性进行操作的行为, 如动画行为就是一种表现属性行为, 所谓表现属性是指与视觉、听觉等感觉通道效果输出的有关物理属性, 如三维几何、速度、位置等; 二是非表现属性行为 (NonPresentationBeh), 即仅对实体非表现属性进行操作的行为, 如涉及到智能属性的行为; 第三种行为既对实体表现属性进行操作, 又对实体非表现属性进行操作, 我们称之为混合属性行为 (SynBeh)。

定义 **12** 行为时间约束 行为时间约束 ConstraintTime给出了虚拟环境中实体行为执行的相关时间参数, 它可以记为二元组:

ConstrainTime::= < BeginTime, DeadLine>

其中, 起始时间点 BeginTime指行为被激活的时间点;

死限时间 DeadLine指行为计算结束的时间点 (以 ms为单位)。

我们给出一个简化了的 YHVML 语言编写的例子 (如下), 其中, 复杂行为 A 拥有一个调用行为集合 {A, B, C, D}, 只有行为 C、D 是原子行为。

```
Behavior A
Begin
    处理程序 A1;
    PBehCall (B);
End
Behavior B
Begin
    处理程序 B1;
    PBehCall (C, D);
    处理程序 B2;
End
Behavior C
Begin
    处理程序 C1;
End
Behavior D
Begin
    处理程序 D1;
End
```

1.2 基于复杂行为的虚拟环境计算模型

通过建立行为的语文模型, 我们可以将虚拟环境的计算看作由虚拟环境中的虚拟实体的行为计算组成。首先, 虚拟环境必然存在着两种特殊的虚拟实体: 用户实体 (UserEntity) 和图形渲染实体 (RenderEntity)。它们具有特定的行为。用户实体 I/O 行为对外部输入感兴趣, 以获得用户与虚拟环境的交互信息, 而图形渲染实体的绘制行为则是根据用户当前视点, 将虚拟环境中实体行为的表现属性和虚拟场景通过三维图形显示的方式表现出来。

考虑单处理机的情况, 虚拟环境的计算可以看作是由一个个仿真步组成的, 每个仿真步一般又可以看作由 3段操作组成: 第一段, 用户实体进行 I/O 行为计算; 第二段, 计算当前步处于活跃状态的除特殊实体外的一般实体行为, 修改相应实体属性和公共属性; 第三段, 图形渲染实体进行绘制行为计算。在具有复杂行为的虚拟环境中, 实体的行为计算将成为重点, 因此, 第二段的算法将变得复杂得多。由于一般行为之间存在的相互调用关系, 使得复杂行为不可能在一个仿真步内完成, 而是要在一系列被调用的行为并行或串行的执行完成之后, 即要在若干仿真步才能真正完成。我们将重点分析第二段的一般行为计算的执行算法。

对于每个虚拟实体 Ent, 都维护一个活跃行为表 ActiveBe-

hTab
(Ent); 一般行为的执行算法如下:

当 Exec (BehBody (Beh)) 时, 每个 Beh都维护一个执行结构: (CallerBeh, ProgramPointer, State);
其中, CallerBeh是调用该行为的父行为, ProgramPointer是程序执行的指针, State是行为的状态。YHVML中的行为状态有活跃、挂起和中止三种。

```
Exec (BehBody (Beh))
Begin
    从 ProgramPointer处开始执行, 直到 BehBody (Beh) 结束或遇到 PBehCall语句。
    IF BehBody (Beh) 执行结束
    THEN 将 Beh的状态置为中止, 将 CallerBeh的状态置为活跃, 将其放入活跃行为表
    IF 遇到 PBehCall语句
    THEN 将 ProgramPointer指向 PBehCall语句之后的语句。
         将 Beh的状态置为挂起;
         将 PBehCall参数中的所有行为状态都置为活跃, 将其放入活跃行为表;
    END
```

以前面所举的复杂行为 A 为例, 基于复杂行为的虚拟环境计算算法的运行时间图如图 1所示。

由该算法可以看出, YHVML除了提供给用户灵活一致的复杂行为定义功能外, 它还充分地支持实体间行为的并行和实体内的行为并行。

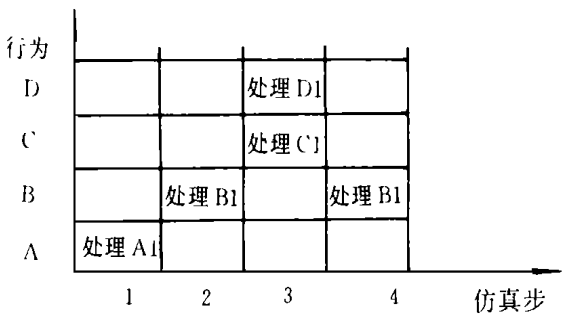

图 1 复杂行为 A 的执行图

2 YHVML运行环境对虚拟环境性能的支持

在 YHVML中, 我们提供给用户描述行为执行时间的支持, 它体现了虚拟环境中行为真实感的需求。在本节中, 我们将阐述 YHVML运行环境对虚拟环境性能的支持, 尤其是对时间真实感的支持。

2.1 虚拟环境的性能需求

虚拟环境运行的主要性能需求是, 实时交互性和真实感。实时交互性是指虚拟环境计算要足够快, 使得用户的输入能实时地在输出上反映出来, 如视点的变化引起的场景变化。文献 [2]将用户执行一个动作和在屏幕上显示出来之间的时间差称为端端延迟 (EndEndLag)。

我们有如下相关定义:

定义 **13** 仿真步延迟 虚拟环境计算在仿真步 st的仿真步延迟 SimuStepLag可记为:

$$SimuStepLag(st) = InputLag(st) + SimuLag(st) + RenderLag(st)$$

其中, InpuLag (st) 是用户实体 IO 行为计算时间延迟;
SimuLag (st) 是一般实体的行为计算时间延迟;
RenderLag (st) 是图形渲染实体绘制行为计算时间延迟。

定义 **14** 端端延迟 虚拟环境计算在仿真步 st的端端延迟 EndEndLag可记为:

$$EndEndLag = \sum_i SimuStepLog_i, \quad i = 1, \cdots, M$$

其中, M 是用户实体的 IO 行为引起相应虚拟环境状态的变化在图形渲染实体的渲染行为中表现处理的仿真步数.

定义 **15** 虚拟环境的实时交互阀值 对于不同的应用, 满足实时交互性的需求不同, 我们记虚拟环境的实时交互阀值为W, 如果虚拟环境的端端延迟 EndEndLag< W, 则称虚拟环境满足实时交互性.

2.2 复杂行为的真实感

虚拟环境中实体行为的真实感体现在两方面: (1) 稳定帧速; (2) 实体的行为具有时间真实感.

2.2.1 稳定帧速

由于虚拟环境具有高度动态性, 在虚拟环境计算的仿真步中的一般实体的行为计算和图形渲染实体的绘制计算所花费的时间是动态变化的, 它带来了帧速的不恒定. 文献 [3] 指出, 变化的帧速使人们无法预测虚拟环境的运行速度, 因此无法与虚拟环境进行有效的交互, 保持帧速的恒定对于人与虚拟环境交互的真实感是非常重要的. 另外, 保持恒定的帧速将有利于用户编程时, 对时间的把握.

在面向复杂行为的虚拟环境中, 由于引进高复杂计算量的行为和大量的实体, 一般实体的行为计算负载变化较大, 很难限定最坏执行情况. 另外, 随着图形硬件的不断发展, 渲染行为的计算量将大幅度降低, 而会使得一般实体的行为计算成为新的瓶颈, 因此, 为了恒定帧速, 我们只有依靠动态的行为计算调度来限定行为计算时间, 使一般行为的计算时间与渲染行为计算时间之和不会超过恒定帧速所要求的时间, 并利用第二种方法, 通过控制帧缓冲交换时间的方法来达到稳定帧速的目的.

定义 **16** 图形绘制实体的绘制行为执行时间延迟上限 我们记图形绘制实体的绘制行为执行时间上限为 T_r, 即对于任意仿真步 st, 有 RenderLag (st) < T_r.

定义 **17** 恒定帧速的仿真步时间延迟上限 我们记使帧速恒定的仿真步时间延迟上限为 γ.

即对于虚拟环境计算, 关于每一仿真步的时间约束如下:

约束 **1** 对于任意仿真步 st, 有 SimuStepLag (st) < γ.

约束 **2** 端端延迟所需的仿真步数 $M < \delta\gamma$.

约束 **3** 对于任意仿真步 st, 有 SimuLag (st) < $\gamma - T_r$. 我们忽略了用户实体的 IO 行为计算时间延迟 InputLag (st).

2.2.2 实体行为的时间真实感

实体的每个行为都有自己的时间约束. 通常, 复杂行为连续地在多个仿真步进行计算, 直到完成.

定义 **18** 行为实际执行时间 行为 Beh 的实际执行时间 ActualExecTime 可以记为:

ActualExecTime (Beh) = BehCompLen (Beh) × γ

其中, BehCompLen 是 Beh 实际的计算仿真步.

为了满足行为的时间真实感, 我们有约束 4.

约束 **4** 对于行为 Beh, 有 DeadLine (Beh) > ActualExecTime (Beh).

2.3 面向复杂行为的实体行为动态调度策略

2.3.1 基本思想

当虚拟环境中实体数量很大, 且行为的计算量也很大时, 维护每个行为的时间约束将可能带来无法满足每一仿真步实体行为计算的总时间约束. 带来帧速的下降和实时交互性能的下降, 破坏了虚拟环境的真实感. 因此, 就要调整某些实体行为的执行时间, 即将某些行为延迟到后续的仿真步运行, 从而使得当前仿真步计算满足约束 3.

由于虚拟环境计算具有高度动态性, 只能通过动态调度才可能满足虚拟环境计算的需要. 算法首先通过行为计算预测的方法来判断当前仿真步的一般实体的行为计算 SimuLag 是否超过了上限, 如果是, 就根据以下原则选择优先级高的行为推迟到下一仿真步执行, 直到当前仿真步计算满足约束 3.

(1) 由于表现属性行为和混合属性行为对同步的要求较高, 因此, 非表现属性行为的优先级比这两类行为的优先级高;

(2) 在同类行为中, 遵循最小松弛度优先 (Least Laxity First, 简称 LLF)[4] 算法的思想, 即松弛度大的行为优先级高.

2.3.2 松弛度定义

定义 **19** 复杂行为计算的已执行量 复杂行为 Beh在仿真步 st的已执行量 BehFinished-CompLen (Beh, st) 定义为该行为已被执行的仿真步数。

定义 **20** 复杂行为计算的松弛度 复杂行为 Beh在仿真步 st的松弛度 Relaxity (Beh, st) 是指除去行为执行所必须的仿真步数后，离行为的死限时间约束的距离，即

Relaxity (Beh, st) = DeadLine (Beh) - γ× (BehCompLen (Beh) - BehFinishedCompLen (Beh, st))

2.3.3 支持行为动态调度算法的虚拟环境计算

我们将动态实体行为调度单元嵌入面向复杂行为的虚拟环境计算仿真步的第二步。一般实体的行为计算，如图 2所示。

动态实体行为的调度算法以对当前仿真步的行为计算时间预测为调度依据，因此，一般实体的行为计算模型将实体行为计算割裂为两部分，在进行调度前，首先对所有实体的活跃行为进行判断，在根据性能需求进行重新调度之后，再确定执行实体的行为。

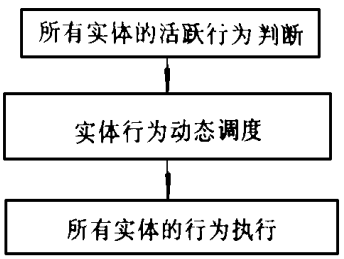

图 2 一般实体的行为计算

3 小结

本文通过分析复杂行为建模的需求，提出一种支持复杂行为的语言模型，该模型具有很强的行为和时间描述能力。同时，我们提出了相应的面向复杂行为的虚拟环境计算模型，并对其性能进行了分析。我们自行研制的分布式虚拟环境软件开发平台 YHVRP的建模语言 YHVML实现了语言模型，而 YHVRP提供的 YHVML运行环境则利用 LLF策略进行动态的行为调度，较为合理地保证了虚拟环境系统的实时交互性和真实感。

参考文献

1 Roehl B. Some Thoughts on Behavior in VR Systems. Http://sunee.uwaterloo.ca/~broehl/behave.html, 1995

2 Wloka M M. Lag in Multiprocessor Virtual Reality. Presence: Teleoperators and Virtual Environments, 1995, 4 (1): 50~63

3 Hawkes R, et al. Update Rates and Fidelity in Virtual Environments. VR, 1995, 1 (2): 99~108

4 Cheng S. Schduling Algorithms for Hard Real-Time Systems-A Brief Survey. Hard Real-Time Systems (eds: John A. Stankovic and Krithi Ramamritham). ISBN 0-8186-0819-06, 1988