国 防 科 技 大 学 学 报
第 23 卷 第 4 期 JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY Vol.23 No.4 2001

文章编号：1001-2486（2001）04-0067-06

基于行为复合机制的结构化图形动画技术的研究与实践*

汪诗林，吴泉源，张晨曦

（国防科技大学计算机学院，湖南 长沙 410073）

摘 要：在各类计算机动画中，角色动画可满足实时性及交互性的需要，其核心技术在于角色的建模技术。该文主要介绍了作者所研制的角色动画制作系统——“网动王 98”（简称 Net_Anim 系统）中所采用的角色建模机制及其相关技术，内容包括：角色的对象模型及其运行机制；角色复合行为的实现算法及相关的关键问题；应用示例：Net_Anim 系统的特色和有待进一步研究和解决的相关问题。

关键词：角色；行为复合；结构化图形；计算机动画

中图分类号：TP317.4 文献标识码：A

The Study and Practice about Vector Graphics Animation Based on Compound Behavior

WANG Shi-lin，WU Quan-yuan，ZHANG Chen-xi

（College of Computer，National Univ. of Defense Technology，Changsha 410073，China）

Abstract：According to the designing methods，computer animations can be divided into three kinds：frame-by-frame animation，keyframe animation and actor animation. Actor animations may meet the demand for real-time and interactivity. The model of actor is the key for actor animation. This paper mainly presents the modeling method of the actor in Net_Anim system developed by the author，which includes：the object-oriented model of the actor and the control scheme based on the model；the algorithms for synthesizing compound behaviors and some key problems to be considered；an example to demonstrate the compound actor with compound behaviors；the characteristics of Net_Anim and some problems to be solved in the future.

Key words：actor；compound behavior；vector graphics；computer animation

近年来计算机动画技术已广泛地应用于教育、娱乐、影视制作以及计算机图形用户接口等众多领域，是深受人们喜爱的媒体表现形式之一。按动画的设计方法进行分类，计算机动画可划分为逐帧动画、关键帧动画和角色动画[1]。逐帧动画要求逐帧设计出每一幅画面，因此动画产出率很低，而关键帧动画则只需要设计出关键画面，中间过渡画面由计算机自动插值完成，因而动画设计效率大大提高。传统帧动画的设计则主要借助于逐帧动画技术和关键帧动画技术。有别于逐帧动画和关键帧动画，角色动画是一种过程动画，动画的主体是各种各样的角色，动画的设计过程就等价于设计和描述各种角色的过程，而动画的播放过程则是各种角色按照自身的行为进行表现的过程，因此角色动画属于实时动画。不过，在角色的设计和表现过程中也需要频繁使用逐帧动画技术和关键帧动画技术。由于在角色动画中可以实现充分的交互性，因而其应用领域大大拓宽，如游戏、仿真和虚拟现实等。角色动画技术的核心是角色的建模，它主要包含两个方面：几何建模和行为建模。考察目前比较流行的二维角色动画制作软件，如 Flash、Director 等，可以发现这些软件在软件易用性及角色图形绘制效果方面比较突出，但不支持构造具有复杂结构及复杂行为模式的角色（需要借助于帧动画的手法来表现复杂的角色行为），因而动画设计的效率不高。近年来，为了配合 CAI 及远程教育的需要，我们研制完成了一套基于 Internet 环境和结构化图形技术、支持具有复合结构及复合行为描述的角色动画制作软件——Net_Anim 系统，本文将主要介绍在 Net_Anim 系统中所采用的角色建模机制及其相关技术，包括：（1）角色的对象模型及其运行机制；（2）复合行为的实现算法及相关问题；（3）应用示例；（4）Net_Anim 系统的特色及有待进一步研究和解决的相关问题。

* 收稿日期：2001-02-14
作者简介：汪诗林（1968-），男，博士生。

1 角色建模及其运行机制

1.1 角色的对象模型

在基于角色机制的结构化动画设计过程中，角色的建模至关重要，它不仅影响到动画的设计方法和设计效率，而且影响到动画的表现方式和表现能力。在 Net_Anim 系统中我们对角色（尤其是具有动画行为的角色）的建模机制进行了深入研究，全面采用了面向对象技术来进行角色建模[2,3]。在该系统中涉及动画角色建模的核心类主要有 3 种：Actor 类、GraphObject 类和 Animation 类。以下我们以 C++ 类描述的形式进行简要介绍。

(1) Actor 类

系统中所有的动画角色都是从类 Actor 派生而来的，类 Actor 的定义如下：

```
class Actor {
属性：类型、包围矩形、中心坐标、屏幕深度、出现时刻、消亡时刻、出现方式、消亡方式等；
        GraphObjectList * Shape；   //角色的几何造型
        ActorList * Child；         //角色中所包含的子角色列表
        AnimationList * Action；    //角色的行为描述列表
        char * UserDefAction；      //用户使用脚本语言自定义的角色行为描述
方法：角色按出现方式出现；角色按消亡方式消失；角色自主更新（以下称 UpdateMyself 方法）；
        角色绘制；角色隐藏；执行用户自定义的动作；角色存储；角色读取等 }；
```

在上述类定义中，形式为 ..List 的变量类型均为链表数据类型，用以存储和管理相应类型的数据对象，如链表 ActorList 用于存储和管理 Actor 对象，链表 GraphObjectList 用于存储和管理 GraphObject 对象，链表 AnimationList 则用来存储和管理 Animation 对象。

(2) GraphObject 类

角色的几何造型是由各种不同类型的图形对象组成的，所有的图形对象都是从 GraphObject 类派生而来的，如点、直线、折线、多边形、曲线、图像、三维平面体图形等基本图元以及由基本图元构成的、具有特殊结构的各种图形对象等。GraphObject 类的定义如下：

```
class GraphObject {
属性：图形对象的类型、包围矩形、中心点坐标、屏幕深度、画笔颜色、填充颜色、笔型、笔
        粗、填充模式、显示精度等
方法：改变比例；平移；旋转；水平或垂直翻转；自我绘制；隐藏；存储、读取等 }；
```

(3) Animation 类

角色的行为由各种行为对象来描述，在 Net_Anim 系统中我们抽象出如下角色行为：沿轨迹线做插值运动、沿关键点移动、随机移动、改变比例、旋转、变形、变深度、变色、闪烁、规律隐现、动态生长、外部模型控制等。所有的行为对象都是由类 Animation 派生而来的，Animation 类的定义如下：

```
class Animation {
属性：行为类型、循环运动标志、运动起始时刻、运动结束时刻、循环结束时刻等
方法：行为存储；行为读取；virtual void Play ()；//按预定方式执行动画行为 }；
```

1.2 角色的运行机制

Net_Anim 系统中的角色完全基于时空控制模式[2~6]来运行，其核心机制如下。

(1) 角色的自主更新过程

系统按一定的时钟周期不断向前推进，在每个时钟周期，场景中的各个角色分别按自身的行为规律不断进行更新，由此产生动画。角色更新的方式有两种，一是由系统来负责进行角色的更新，这必将增加系统的负担和复杂程度，二是由角色自己来负责更新自己，这样不仅可以降低系统的复杂程度，而且可以大大提高系统的灵活性和可扩充性。上述角色对象模型可以很好地满足角色自主更新的需要，其过程如下：

① 系统在每个时钟周期依次调用各个角色的 UpdateMyself 方法，启动角色的自主更新过程；

② 角色依一定次序启动其行为列表 Action 中每个行为对象的 Play（）方法。每个行为对象的 Play（）方法负责对角色的状态进行更新并记录状态的更新量（方法见后），或仅记录状态的更新量。角色状态的更新包含两个方面，一是对角色属性的更新，二是对 Shape 中各图形对象的更新，方法是通过启动 Shape 中各个图形对象的相应方法来实现，如，要进行旋转，则调用图形对象的 Rotate（）方法即可；

③ 角色解释执行用户在 UserDefAction 中定义的脚本语言，其间或直接进行角色的状态更新并记录状态的更新量，或仅记录状态的更新量；

④ 若前两步中仅记录了状态的更新量，则根据所记录的状态更新量对角色的状态进行更新。

⑤ 将所记录的状态更新结果叠加在 Child 中各子角色上，作为各子角色的初始更新状态，并依次启动每个子角色的 UpdateMyself 方法，进行子角色的自主更新（方法同上）。

⑥ 当角色中所包含的各级子孙角色均已完成自主更新时，角色的自主更新过程结束。

上述的角色自主更新过程非常模式化，因此适合于程序实现。事实上，角色的绘制、存储、读取等过程也都与角色的自主更新过程相类似。

(2) 系统的控制模型

由于角色能够进行自主更新，因此相应的系统控制模型就得以大大简化，如图 1 所示。

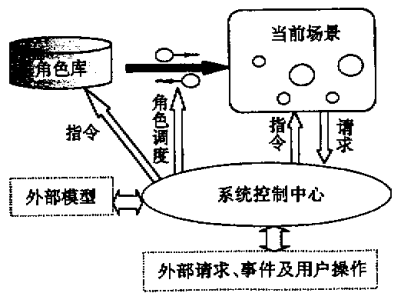

图 1 系统控制模型

Fig.1 The system control model

在图 1 的系统控制模型中，系统控制中心是系统核心所在，它的主要任务如下：

· 维护一个系统时钟，用于控制动画的播放；

· 负责进行角色调度，将完成使用的角色从场景中取出并放回角色库，同时按时序关系从角色库中选取合适的对象进入场景；

· 负责处理外部请求、事件和用户操作，并实时地将某些请求、事件或操作转发给相应的角色；

· 负责外部模型与受控制角色之间的通信及接口控制；

· 负责定时启动角色的各种行为（如自主更新行为、自主绘制行为等），同时接收并处理角色的请求。

2 角色行为的复合

2.1 行为复合算法

从上述角色对象模型中我们可以看到，每个角色对象都可以有自己的行为描述，每个角色中所包含的子角色也同样有自己的行为描述，因此，在角色进行自主更新时，在复合角色的内部不可避免地需要进行行为的复合。为简洁起见，在以下的讨论中，我们把角色的行为主要划分为 4 类：移动、变比例、旋转、变形和属性修改，其中属性修改是指除角色位置、比例和旋转角度以外的属性的修改。

由于变形和属性修改行为一般不宜进行多级复合，因此在以下的行为复合实现算法中我们仅考虑移动、变比例和旋转行为的任意复合。

为了便于描述由移动、变比例和旋转行为所导致的角色姿态的变化，我们首先定义一个数据包含以下成员的结构类型 ACTSTATE：

```
BOOL Move；                //移动标志
float MoveX，MoveY；       //x 和 y 方向的移动量
BOOL Extend；              //变比例标志
float ExtendX，ExtendY；//x 和 y 方向的变比例量
BOOL Rotate；              //旋转标志
float RotateAngle；        //旋转角度
```

根据上述结构，我们可以用一个 3 元组（state，t_1，t_2）来描述角色的姿态变化（以下简称行为状态），其中 state 为 ACTSTATE 类型的变量，其值表示当系统时钟从 t_1 推进到 t_2 时角色姿态所发生的变化。为了实现角色的自主更新，我们必须能够快速地推算出当系统时钟向前推进时，角色的状态变化量（不管该角色是顶层角色还是子角色）。

考察角色 $Actor_i$，假定系统时钟从 t 推进到 $t+\Delta t$，并假定在 $Actor_i$ 进行自主更新之前，复合在该角色上的行为状态为（state，t，$t+\Delta t$），现在，我们需要确定，当 $Actor_i$ 完成自主更新时的行为状态（state '，t，$t+\Delta t$），为此，我们进一步假定在（t，$t+\Delta t$）时间范围内，该角色将会连续执行以下行为：$Animation_1$，$Animation_2$，…，$Animation_n$。我们依次对 $Animation_1 \sim Animation_n$ 执行以下判断和操作：

(1) $Animation_i$ 涉及平移交换（不涉及比例变换和旋转变换）

假定在（t，$t+\Delta t$）时间内由 $Animation_i$ 所导致的平移变换量为（Dx，Dy），则执行以下操作：

- 若 state.Move 为 FALSE，则取 state.Move = TRUE，state.MoveX = Dx，state.MoveY = Dy；
- 若 state.Move 为 TRUE，则取 state.MoveX + = Dx，state.MoveY + = Dy；

(2) $Animation_i$ 涉及比例变换

假定在（t，$t+\Delta t$）时间内由 $Animation_i$ 所导致的变比比率为（Fx，Fy），并假定变比例中心为（Sx，Sy），则执行以下操作：

- 假定角色的当前中心点为（Cx，Cy）。若（Cx，Cy）与变比中心（Sx，Sy）不同，则首先计算因变比例而产生的角色位置的变化：

$$dx = (1 - Fx)(Sx - Cx)；dy = (1 - Fy)(Sy - Cy)；$$

  若 state.Move 为 FALSE，则取 state.Move = TRUE，state.MoveX = dx，state.MoveY = dy；否则取 state.MoveX + = dx，state.MoveY + = dy；
- 若 state.Extend 为 FALSE，则取 state.Extend = TRUE，state.ExtendX = Fx，state.ExtendY = Fy；否则取 state.ExtendX * = Fx，state.ExtendY * = Fy；

(3) $Animation_i$ 涉及旋转变换

假定在（t，$t+\Delta t$）时间内由 $Animation_i$ 所导致的旋转角度改变量为 dAngle，并假定旋转中心为（Rx，Ry）。则执行以下操作：

- 假定角色的当前中心点为（Cx，Cy）。若（Cx，Cy）与旋转中心（Rx，Ry）不同，则首先计算因旋转而产生的角色位置的变化：

$$dx = (Cx - Rx) * (\cos(dAngle) - 1) - (Cy - Ry) * \sin(dAngle)；$$
$$dy = (Cx - Rx) * \sin(dAngle) + (Cy - Ry) * (\cos(dAngle) - 1)；$$

  若 state.Move 为 FALSE，则取 state.Move = TRUE，state.MoveX = dx，state.MoveY = dy；否则取 state.MoveX + = dx，state.MoveY + = dy；
- 若 state.Roate 为 FALSE，则取 state.Rotate = TRUE，state.RotateAngle = dAngle；否则，取 state.RotateAngle + = dAngle。

(4) $Animation_i$ 为其它变换

若 $Animation_i$ 不会导致角色发生平移、变比例和旋转，则 state 不发生变化，否则参照（1）～（3）修改 state。

当按照上述算法依次执行 $Animation_1 \sim Animation_n$ 之后，我们即可得到 $Actor_i$ 新的行为状态（state '，t，$t+\Delta t$），可依据该状态对角色实施状态更新，同时，该状态又继续叠加在 $Actor_i$ 的各个子角色上，作为各子角色进行自主更新时的初始行为状态。

2.2 相关问题

上述角色自主更新算法看似简单，但在具体实现过程中还存在不少困难，有许多细节问题需要慎重考虑和解决，否则角色实际表现出的行为将难以预料。这些问题有：

（1）中心点坐标的表示

在上述算法中所涉及的中心点坐标有三种：角色中心点坐标、变比例中心点坐标和旋转中心点坐标。中心点坐标的表示方式可以有两种：相对坐标（相对于某参考点）和绝对坐标（相对于屏幕原点）。比如角色中心点可以采用相对于父角色（若存在）中心点的相对坐标，变比例中心点和旋转中心点也可以采用相对于角色中心点的相对坐标。不同的坐标表示方式将直接影响到上述算法的执行方式和执行效果。在 Net_Anim 系统中，我们采用了基于相对坐标的行为实现机制。

（2）行为执行的次序

基于（1）中不同的中心点坐标表示方法，$Animation_1 \sim Animation_n$ 的不同执行次序将会影响到状态（state '，t，$t+\Delta t$）的实际取值。为了使角色的行为能按预定的方式进行，即状态（state '，t，$t+\Delta t$）的取值唯一，我们必须合理确定 $Animation_1 \sim Animation_n$ 的相对执行次序。一般可采用如下几种方式来确定执行次序：a）由用户直接指定各 Animation 之间的执行次序，这种方式实现起来较为简单、灵活，但增加了用户的负担；b）按各 Animation 的起始运动时刻 StartTime 的大小进行排序；c）按 Animation 的类型进行排列，类型相同时再按照起始运动时刻 StartTime 进行排序。在 Net_Anim 系统中我们采用了 c）中的行为排序方式。

（3）实施角色状态更新的时机

在上述算法中我们需要按序执行每个 Animation。当执行 $Animation_i$ 时，由 $Animation_i$ 所导致的角色状态的变化是立即施加到角色上，还是仅累加到状态（state，t，$t+\Delta t$）中，待所有 Animation 均被执行完毕时再根据所得到的状态（state '，t，$t+\Delta t$）来更新角色，这个问题将同样影响到行为复合的效果。因为，若每次执行 $Animation_i$ 时立即进行角色状态更新，那么后续 Animation 的处理结果将受到 $Animation_i$ 的影响，而如果仅进行状态累加，那么后续 Animation 的处理结果将不会受到 $Animation_i$ 的影响。在 Net_Anim 系统中，我们采用的是状态累加方式。

（4）Animation 的处理机制

每个 Animation 如何根据自己的运动规律以及运动起始时刻、运动结束时刻、是否循环等属性来快速、准确地确定在（t，$t+\Delta t$）区间内对角色状态的更新结果，是上述算法在具体实现时的关键技术之一。

3 应用

下面我们用一个具体实例来演示行为复合的效果。如图 2 所示的一辆简易自行车，我们可把它定义成一个复合角色，该角色由 4 个子角色构成，分别是：①—前轮；②—后轮；③—脚蹬；④—车架。我们分别为①、②和③定义了旋转行为，旋转中心分别为各自的中心点，每个时钟周期顺时针旋转 1 度，一直循环到角色消失为止。④没有行为定义。至此，一个较为逼真的自行车角色模型已经构造完毕。接下来，我们可以为自行车角色定义各种行为，如沿各种轨迹移动（产生骑车效果）、改变大小（产生远近效果）和旋转等，也可以同时定义多种运动。图 3 给出了该自行车角色沿一个“滑坑”滑动的复合行为。

4 结束语

基于以上思想研制完成的 Net_Anim 系统具有如下显著特点：（1）具有所见即所得的编辑风格，动

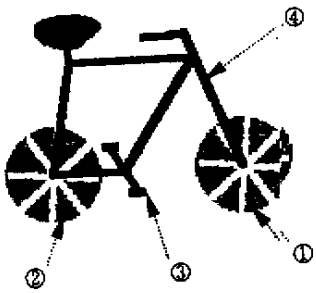

图2 自行车角色结构

Fig.2 The structure of bicycle actor

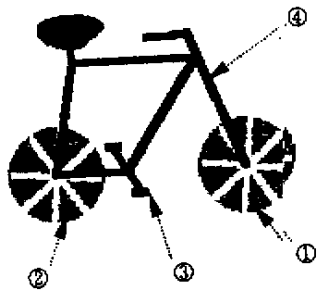

图3 自行车角色的应用

Fig.3 The application of bicycle actor

画设计的过程等价于角色绘制和行为描述的过程，直观、自然，易学易用，动画产出率高；(2) 与其它同类系统（如 Flash 等）相比，角色的行为更丰富、表现能力更强。(3) 由于基于结构化信息描述，因此所产生的动画存储量很小（在未压缩的情况下，存储量仅相当于同类 Flash 动画的 20% ~ 50%；(4) 由于提供了脚本语言及外部控制模型的支持，因此可以满足许多特殊应用场合的需要。由于以上的特点，目前 Net_Anim 已经在 CAI 及远程教育领域得到了大量、广泛的应用，如国讯网校及其各分校中的动画 CAI 课件均由 Net_Anim 系统制作完成。但 Net_Anim 系统也存在如下缺陷，有待进一步研究和改进：(1) 虽然角色的行为模式多种多样且易于描述，但行为一旦定义以后即机械执行，难以随机应变，缺乏智能的自主控制。(2) 行为定义的手段和方法还不够丰富，不支持行为的动态捕获；(3) 角色的建模能力有限，用于逼真地仿真现实生活中的许多真实角色时还存在一定困难；(4) Net_Anim 系统不支持三维对象建模（目前系统中仅支持等厚度、平面体三维对象，如三维文字等），因而在三维应用不断普及的今天，其应用范围会受到一定限制。

参考文献：

[1] Nadia M T, Daniel T. Computer Animation: Theory and Practice [M]. Springer-Verlang, Second Edition. 1990.

[2] Wang shilin, Zhang chenxi, Zhang chunyuan, et al. The Design And Implementation of IPS-Maker: An Object-oriented Authorware for Generation Information Playing Software [A]. Int Conf On Computers In Education, 1995. 10.

[3] Zhang chenxi, Wang shilin, et al. Net_Anim: a new software for creating animations for web pages [A]. Asia Pacific Web Conference, 1998.

[4] Michael F. Cohen. Interactive Spacetime control for Animation [J]. Computer Graphics, 1992, 26 (2): 293 - 302.

[5] Zicheng Liu, Steven J G, Michael F C. Hierarchical Spacetime control [A]. In Proceedings of SIGGRAPH '94, pages 35 - 42, 1994. In Computer Graphics proceedings, Annual Conference Series.

[6] Thomas J N, Joe M. Spacetime Comstraints Revisited [A]. In Proceedings of SIGGRAPH '93, pages 343 - 350, 1993. In Computer Graphics proceedings, Annual Conference Series.