Animação Comportamental Baseada em Lógica

BRUNO FEIJÓ[1]
MÔNICA M. F. DA COSTA[1]

[1]ICAD - Laboratório de CAD Inteligente
Departamento de Informática, PUC-Rio, SCT/PR
Rua Marquês de São Vicente, 225
22453 Rio de Janeiro, RJ, Brasil
bruno@icad.puc-rio.br

Abstract. This paper presents a conceptual model for the development of behavioral animation systems based on tasks, planning and first-order logic.

Introdução

Este trabalho apresenta um modelo para animação comportamental que se beneficia da simplicidade e robustez da lógica de primeira-ordem. O modelo parte de uma formulação de planejamento desenvolvida por [Genesereth and Nilsson (1987)] e baseada no Método de Green. Os autores também apresentam os conceitos de planejamento ponderado e de eventos que mantêm a consistência da formulação e viabilizam situações básicas de comportamento. O protótipo **MoTion** é desenvolvido para testar o modelo.

Animação Comportamental

A animação comportamental busca o realismo a nível de comportamento dos personagens em cena. Neste tipo de animação, os personagens são "atores sintéticos" dotados de personalidade e habilidades próprias. A atuação de um personagem não é mais oriunda exclusivamente de intervenções diretas do animador, mas sim fruto de sua personalidade, seu humor, suas metas e sua interação com os demais atores. O animador deixa de ser um projetista de quadros-chave e passa a exercer o papel de diretor (ou de *meta-animador*), como idealiza o pioneiro Craig Reynolds [Reynolds (1987)].

Animação comportamental é um conceito que engloba *animação baseada em conhecimento* [Zeltzer (1983)], *atores sintéticos* [Thalmann and Thalmann (1990)] e *animação baseada em tarefas* [Zeltzer (1985)]. A raiz deste conceito mais amplo está no trabalho de Craig Reynolds [Reynolds (1987)] que introduz um modelo comportamental distribuído para simular bandos, manadas e cardumes. Jane Wilhelms [Wilhelms and Skinner (1990)] consolida o termo e apresenta um modelo comportamental baseado em uma rede de sensores, nós e *effectors*. Atualmente os problemas de animação comportamental confundem-se com os da robótica e os da inteligência artificial.

Neste conceito mais amplo de animação, o realismo de movimentos globais [Wilhems (1987)] [Isaacs and Cohen (1987)] [Zeltzer and McKenna (1990)] ou locais como os da face [Guenter (1989)] introduz um novo problema: regularidade. Um ser humano, por exemplo, jamais executará uma mesma ação duas vezes de forma idêntica. A animação comportamental deve, portanto, também considerar aspectos difusos (*fuzzy*) e randômicos do ambiente e dos atores. No caso de expressões faciais, Kalra [Kalra et al. (1991)] tenta solucionar o problema de expressões faciais sincronizando movimentos da face segundo determinadas emoções.

O presente trabalho trata de um caso particular de animação comportamental: a animação orientada a tarefas.

Animação Orientada a Tarefas

Um dos primeiros estágios na direção da animação comportamental foi determinado pelo conceito de animação orientada a tarefas (*task level animation*), apresentado pela primeira vez por Zeltzer [Zeltzer (1985)]. Neste tipo de animação, uma tarefa de alto nível (p.ex. "atenda o telefone") é decomposta em uma seqüência de movimentos elementares.

Segundo [Thalmann and Thalmann (1990)], um sistema orientado a tarefas deve dispor das seguintes informações: **descrição da cena**; **regras que governam o universo da cena**; **comportamento dos personagens**; e **biblioteca de atividades elementares**. Este tipo de sistema orientado a tarefas supõe um **planejador** capaz de gerar uma seqüência de atividades elementares. O ator sintético Igor em

[Strassmann (1989)] é um bom exemplo desta abordagem. A questão fundamental é, portanto, a do planejamento.

Planejamento

Um problema de planejamento consiste em elaborar-se um plano que, quando executado, transforme o estado corrente do universo em um outro estado no qual um conjunto de propriedades pré-estabelecidas estejam satisfeitas. No caso de situações difusas ou randômicas, o planejamento deve ocorrer de forma incremental através de um processo de planejamento do planejamento (i.e. um **meta-planejamento**).

As entradas de um planejador (fig. 1) são as seguintes:

- σ, designador do estado inicial;
- ρ, designador da meta a ser atingida;
- Γ, conjunto de designadores de ações;
- Ω, conjunto de sentenças que descrevem o estado inicial, a meta e as açoes disponíveis no universo.

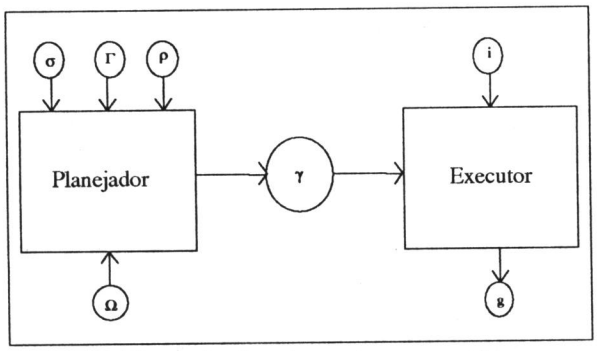

Figura 1: planejamento e execução

O planejador fornece como saída um **plano** γ, que é uma seqüência de designadores de ações. γ, quando executado, é capaz de levar um estado particular ***i*** (entre os muitos permitidos por σ) a um estado ***g*** que satisfaz a descrição geral da meta ρ.

Descrição da cena

Na descrição da cena os objetos são definidos quanto a geometria, topologia, posição, orientação e características físicas. Do ponto de vista do planejamento, a descrição da cena é a definição do estado inicial ***i*** e da meta ρ.

Uma propriedade ***p*** é atribuída a um estado ***s*** pelo predicado **t**, i.e., **t(*p*,*s*)**. Desta maneira, as seguintes sentenças:

t(at(Jotalhão,A),S1)
t(at(pizza,B),S1)
t(hungry(Jotalhão),S1)

descrevem o estado inicial S1 no qual o personagem Jotalhão se encontra na posição A, uma pizza se encontra na posição B e Jotalhão está com fome.

Formalmente, o estado ***s*** é um membro do conjunto de estados descritos por ***p***. Assim, "hungry(Jotalhão)" designa todos os estados nos quais Jotalhão está com fome. O termo ***p*** é chamado de **descritor de estado**.

Define-se a meta ρ como um predicado unário **goal** no domínio dos estados. Por exemplo, a sentença

$\forall$_s goal(_s) $\leftrightarrow$ t(happy(Jotalhão),_s)

afirma que qualquer estado _s em que Jotalhão esteja feliz é um estado que satisfaz a meta.

Regras que governam o universo da cena

As regras do universo governam as mudanças de estado. Estas mudanças são provocadas por instâncias de **operadores** do tipo:

walk(*actor*,*start_point*,*end_point*)
eat(*actor*,*food*)

chamadas de **designadores de ação**.

A função **do(ação,estado)** mapeia uma ação e um estado em um outro estado que é o resultado da execução daquela ação no primeiro estado, p.ex.:

do(walk(Jotalhão,A,C),S1).

A função **do** é usada para definir os operadores. Por exemplo, **walk** é definido por:

t(at(_a,_z),do(walk(_a,_y,_z),_s))$\leftarrow$
t(actor(_a),_s) ^
t(at(_a,_y),_s)

isto é, se um ator _a ocupa a posição _y no estado _s, então este ator ocupará a posição _z em um novo estado que advém da execução da ação walk(_a,_y,_z) no estado _s. Em geral omitem-se os quantificadores "para todo" (p.ex. $\forall$_a$\forall$_y$\forall$_z$\forall$_s) por questões de simplicidade de notação. $\leftarrow$ significa "se" e ^ significa a conjunção "e".

Outras regras do universo são os **axiomas de *frame*** e as **ações condicionais**.

Comportamento dos personagens

Ao designar-se uma mesma tarefa a duas pessoas distintas, dificilmente estes indivíduos executarão a dada tarefa da mesma forma. A animação comportamental leva em consideração os comportamentos diferentes de personagens distintos.

Os comportamentos dos personagens são definidos como **restrições de estado**, ou seja, os comportamentos são propriedades que independem das ações realizadas no universo em questão. Exemplos simples são:

$\forall$_s t(likes(Jotalhão,pizza),_s)
$\forall$_s t(likes(Manezinho,sandwich),_s)

Plano

Chama-se formalmente de **plano** uma seqüência de designadores de ação γ tal que:
1. $\gamma \in \Gamma$;
2. $\Omega \Rightarrow \rho(do(\gamma,\sigma))$

Por exemplo, a seqüência de ações [walk(Jotalhão,A,C),eat(Jotalhão,pizza)] é um plano desde que:
goal(do([walk(Jotalhão,A,C),eat(Jotalhão,pizza)],S1))
possa ser obtido a partir de Ω.

Método de Green

O método de Green, originalmente proposto por [Green (1969)], resume-se na prova de existência de um plano, i.e.:

$$\exists\gamma\ \rho(do(\gamma,\sigma))$$

A prova é realizada por Resolução [Robison (1965)]. Como conseqüência da força da lógica de primeira-ordem, o método de Green para planejamento é consistente no sentido de que somente produz planos corretos. Além do mais, o método é **completo** no sentido de que está garantida a descoberta de um plano se existe um.

Biblioteca de atividades elementares

Durante a execução de um plano pelo **mecanismo executor**, as ações disparam funções que realizam os seus movimentos básicos. Neste trabalho, estas funções são chamadas de **atividades elementares** e estão armazenadas em uma biblioteca de atividades.

Para obter-se o máximo de expressividade, torna-se necessário dar ao animador vários níveis de acesso à hierarquia de controle de movimento. Neste caso, o animador pode modificar uma seqüência gerada automaticamente pelo sistema ou criar novas atividades elementares.

O grau de realismo depende do grau de detalhes implementados nas atividades elementares. A atividade elementar *walk* pode apenas executar translações do centro de massa de um ator ou disparar um movimento coordenado dos membros do personagem.

Planos ponderados

Para permitir a definição de preferências de comportamento (p.ex. Jotalhão gosta mais de pizzas do que de sanduíches), os autores introduziram o conceito de planos poderados.

Os planos gerados pelo planejador são associados a pesos que traduzem o benefício proporcionado por suas possíveis execuções. Desta maneira, inicia-se o planejamento com um plano que é uma lista vazia de ações e que tem peso 0 (zero). No final, compara-se o peso de todos os planos gerados e seleciona-se o de menor peso.

O peso de um plano é a soma dos pesos associados às ações de sua lista. Por sua vez, o peso de uma ação é determinado pelo comportamento do personagem e pelas regras do universo que definem as ações. Por exemplo, a preferência de Jotalhão por pizzas é determinada pelos comportamentos:

$\forall$_s t(likes(Jotalhão,pizza,10),_s)
$\forall$_s t(likes(Jotalhão,sandwich,7),_s)

e pelos operadores **walk** e **eat**. A nova definição de **walk** pode ser:

t(at(_a,_z),do(walk(_a,_y,_z,_w),_s)) $\leftarrow$
t(actor(_a),_s) ^
t(at(_a,_y),_s) ^
_w=-1*dist(_y,_z)

Neste exemplo, a contribuição da ação de andar de um ponto _y para um ponto _z é negativa e tem módulo igual à distância entre _y e _z. Assim, quanto menor for a distância a ser percorrida, maior será a contribuição desta ação para um plano da qual ela faça parte.

Eventos

Um efeito importante em animação comportamental é a mudança abrupta de atitude de um personagem diante da percepção de algo inesperado no universo da cena. Na formulação tradicional de planejamento a previsibilidade do universo impede a simulação deste efeito. Os autores, neste trabalho, apresentam o conceito de **eventos** para contornar esta dificuldade.

Eventos são estados que se caracterizam pela ocorrência de algo inesperado. Uma vez ocorrido um evento se faz necessário um novo planejamento. Para formalizar esta idéia define-se o predicado **event**. Este predicado tem associado a ele uma regra que estabelece as propriedades que devem ter os estados que são eventos. Um estado que satisfaça a este predicado é, portanto, um evento. A detecção de eventos é feita através de um **teste de eventos**. Desta maneira, a sentença:

$\forall$_s event(_s) $\leftarrow$ t(near(Jotalhão,Toy),_s)

é usada para a situação na qual Jotalhão, no seu caminho para comer a pizza, percebe, com pavor, a presença do cãozinho Toy. Neste caso, um evento é gerado e Jotalhão, firme em seu propósito de ser feliz, vai certamente planejar novamente as suas ações (indo,

por exemplo, comer o sanduíche).

O procedimento proposto é submeter o plano ao teste de eventos, ou seja, verifica-se se cada estado resultante de uma ação do plano satisfaz o predicado **event**. Se este predicado é satisfeito, o estado em questão é passado ao planejador como um novo estado inicial. A seqüência de ações anteriores à ocorrência do evento será mantida no plano final. O novo plano gerado é, por sua vez, submetido ao teste de eventos e assim procede-se sucessivamente.

Arquitetura de um protótipo

As figuras 2, 3 e 4 resumem a arquitetura do protótipo **MoTion**.

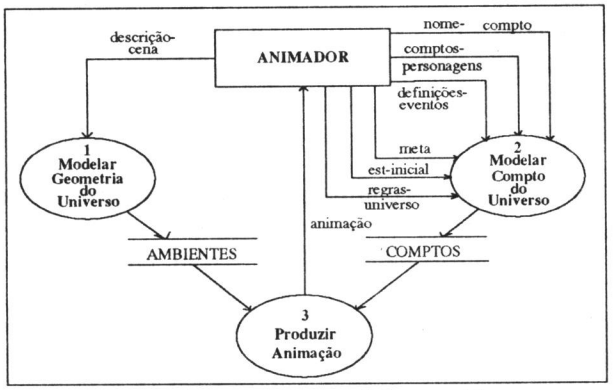

Figura 2: estrutura geral do protótipo

O módulo **Planejar** (fig. 4) é implementado pelo Método de Green. Os módulos **Planejar** e **Testar Eventos** (fig. 4) são executados repetidamente até que não haja mais eventos provocados por novos planos gerados. O módulo **Planejar Animação** (fig. 3) e seus descendentes na hierarquia de módulos da figura 4 foram implementados na linguagem **prolog**.

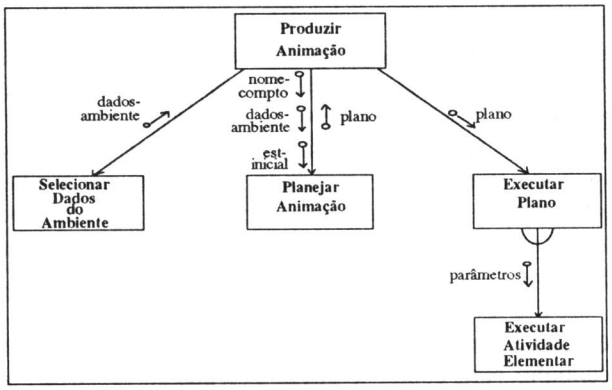

Figura 3: estrutura do módulo que produz a animação

O módulo **Executar Plano** (fig. 3) invoca, para cada ação de um plano, a atividade elementar correspondente. A execução seqüencial destas atividades produz a saída gráfica do **MoTion**.

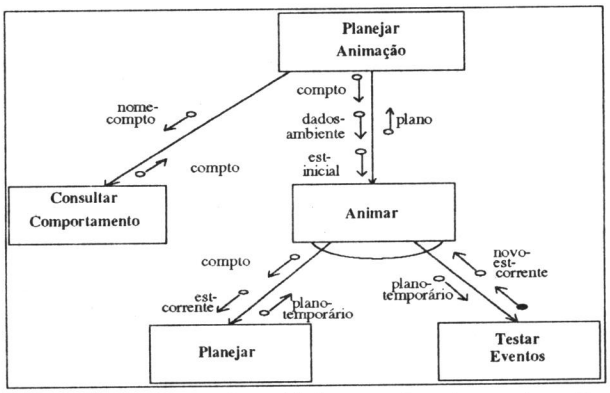

Figura 4: estrutura do módulo que planeja a animação

O protótipo **MoTion** foi desenvolvido em uma SparcStation 1+, com a interface gráfica **IntGraf** e o modelador de sólidos **GeneSys**. A programação foi híbrida com **C** e **BIM-Prolog**. As atividades elementares foram implementadas em C e fazem uso de funções da biblioteca **GeneSys**. A visualização final é feita no ambiente tri-dimensional do sistema GeneSys, sem a preocupação de apresentar um *rendering* de melhor qualidade.

As figuras 5, 6, e 7 apresentam três momentos de uma animação com Jotalhão, o cãozinho Toy, uma pizza e um sanduíche. Nesta animação, Jotalhão assusta-se com o cãozinho, desvia-se da pizza e prefere comer o sanduíche.

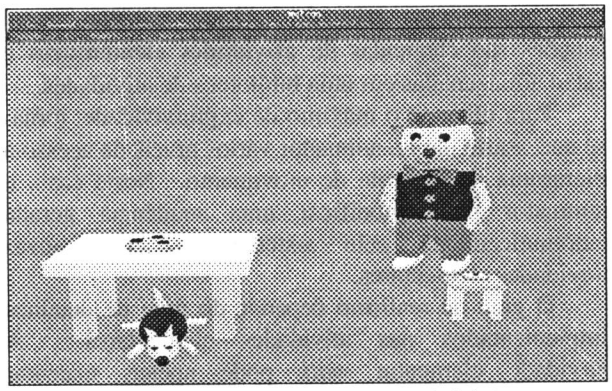

Figura 5: Jotalhão no estado inicial

O arquivo utilizado para modelar geometricamente o universo tem aproximadamente 3MB. O tempo de *rendering* de um quadro (*frame*) é de aproximadamente 5 segundos utilizando-se um algoritmo de *hidden-lines* e 14 segundos utilizando-se um algoritmo de *constant-shading*. O tempo de geração do plano adequado é de aproximadamente 7

segundos.

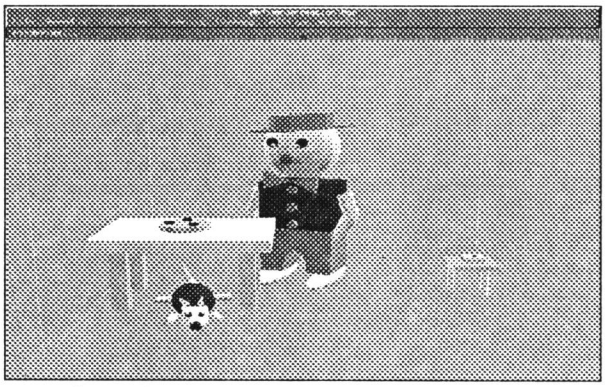

Figura 6: Jotalhão se encaminha para a pizza

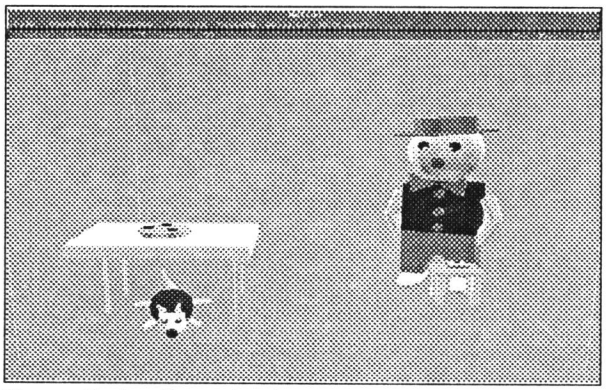

Figura 7: Jotalhão desvia de Toy e se aproxima do sanduíche

Conclusões

O modelo apresentado pelos autores estabelece os princípios da animação comportamental baseada em tarefas, planejamento e lógica, de uma maneira formalmente elegante, simples e robusta.

O modelo proposto no presente trabalho, entretanto, não discute estratégias de aumento de desempenho, tais como poda por inviabilidade (*unachievability pruning*), alinhamento de estados (*state alignment*), supressão de axiomas de *frame*, regressão de meta e diferenças de estado.

O protótipo desenvolvido, apesar de demonstrar a viabilidade do modelo, está ainda bastante limitado. Tendo em vista o uso direto da linguagem **prolog**, não se dispõe de mecanismos de fatoração e não há negação clássica. A ausência de fatoração impede a geração de planos condicionais. A falta de negação clássica pode inviabilizar a definição de algumas restrições de estado usadas na especificação do comportamento do personagem, bem como pode impedir a definição de alguns axiomas de *frame* e de algumas ações usadas na especificação das regras do universo da cena.

O símbolo de igualdade disponível no protótipo é uma igualdade dita relaxada, isto é, ela é uma relação extra-lógica cujo procedimento associado só é correto quando todas as variáveis estão instanciadas. Entretanto, a consideração de uma lógica, mais completa, com igualdade não é recomendada por questões de simplicidade e eficiência.

Uma outra limitação do protótipo desenvolvido é a ausência de uma interface de alto nível que permita o processo interativo de modelagem comportamental do universo. Na situação atual o animador é obrigado a especificar regras e comportamentos em linguagem **prolog**.

Melhorias mais imediatas do protótipo referem-se a atividades elementares com realismo dinâmico e mecanismos de resposta (*feed-back*) do ambiente com detecção de obstáculos, tratamento de colisões e deformações locais.

Agradecimentos

Os autores agradecem ao CNPq, FAPERJ, The British Council e TeCGraf pelo suporte financeiro, bem como todos os participantes dos projetos KAD e BA em cooperação com o ESL do Imperial College e CMEST do Instituto Superior Técnico de Lisboa.

Referências

M. R. Genesereth, N. J. Nilsson, Logical Foundations of Artificial Intelligence, Morgan Kaufmann (1987).

C. W. Reynolds, Flocks, Herds, and Schools: A Distributed Behavioral Model, ACM SIGGRAPH'87 21(4) (1987) 25-34.

D. Zeltzer, Knowledge-based Animation, ACM SIGGRAPH/SIGART Workshop on Motion (1983) 187-192.

N. Magnenat-Thalmann, D. Thalmann, Computer Animation: Theory and Practice, Springer, Tokyo (1990).

D. Zeltzer, Towards an integrated view of 3D computer animation, The Visual Computer 1(4) (1985) 249-259.

J. Wilhelms, R. Skinner, A "Notion" for Interactive Behavioral Animation Control, IEEE Computer Graphics & Applications 10(3) (1990) 14-22.

J. Wilhelms, Using Dynamic Analysis for Realistic Animation of Articulated Bodies, IEEE Computer Graphics & Applications 7(6) (1987) 12-27.

P. M. Isaacs, M. F. Cohen, Controlling Dynamic Simulation with Kinematic Constraints, Behavior Functions and Inverse Dynamics, ACM SIGGRAPH'87 21(4) (1987) 215-224.

D. Zeltzer, M. McKenna, Dynamic Simulation of Autonomous Legged Locomotion, ACM SIGGRAPH'90, 24(4) (1990) 29-38.

B. Guenter, A system for simulating human facial expression, Magnenat-Thalmann N., Thalmann D. (eds) State-of-the-art in computer animation, Springer, Tokyo (1989) 191-202.

P. Kalra et al., Smile: A Multilayered Facial Animation System, IFIP Conf. on Modeling in Computer Graphics, Springer, Tokyo, (1991) 189-198.

S. Strassmann, D. Zeltzer, An Architecture for Task-level Animation, unpublished working paper, M.I.T. Media Lab Computer Graphics and Animation Group (1989).

C. Green, Applications of theorem proving to problem solving, IJCAI-69 (1969) 219-239.

J. A. Robinson, A machine-oriented logic based on resolution principle, J. ACM 12(1) (1965) 23-41.