UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Dissertação de Mestrado

Controle de Animações por Computador utilizando Redes de Petri

Autor: **Alessandro de Lima Bicho**
Orientador: **Prof. Dr. Léo Pini Magalhães**

DISSERTAÇÃO DE MESTRADO

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Controle de Animações por Computador utilizando Redes de Petri

Autor: **Alessandro de Lima Bicho**

Orientador: **Prof. Dr. Léo Pini Magalhães**

Banca Examinadora:

Prof. Dr. Léo Pini Magalhães - DCA/FEEC/UNICAMP
Profa. Dra. Carla Maria Dal Sasso Freitas - Inst. de Inf./UFRGS
Prof. Dr. Rafael Santos Mendes - DCA/FEEC/UNICAMP
Profa. Dra. Wu, Shin-Ting - DCA/FEEC/UNICAMP

Dissertação submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos parciais para obtenção do Título de **Mestre em Engenharia Elétrica**. Área de concentração: **Engenharia de Computação**.

10 de setembro de 2001.

RESUMO

As alterações visuais, decorrentes das ações presentes em uma animação por computador, somente são possíveis porque há um mecanismo para selecionar e controlar os atributos da cena necessários para produzir os efeitos visuais desejados. Este trabalho explora o uso de redes de Petri como o mecanismo de controle que possibilita simplificar a coordenação das ações em uma animação ao descrevê-la por meio do tratamento dos eventos presentes. Este mecanismo permite definir modelos hierárquicos de controle com diferentes níveis de abstração, onde o encapsulamento de detalhes facilita a reutilização em animações cujos comportamentos sejam similares e possibilita que mudanças possam ser realizadas sem a necessidade de remodelar toda ou grande parte da animação. Além disso, um modelo de controle baseado em redes de Petri também permite prever e testar o comportamento de uma animação antes mesmo da sua execução. O mecanismo proposto é discutido através do estudo de casos de figuras bípedes.

ABSTRACT

The visual changes, derived from the actions in a computer animation, are only possible because there is a mechanism to select and control the attributes of the scene required to produce the desired visual effects. This work explores the use of Petri nets as the control mechanism that makes possible to simplify the coordination of the actions in an animation by describing it through the treatment of the existing events. This mechanism allows to define control hierarchic models with different levels of abstraction, where the encapsulation of details facilitates the reuse in animations which behaviors are similar and makes possible to carry out changes without remodeling all or great part of the animation. Moreover, a Petri net-based control model also allows to foresee and test the behavior of animation even before its execution. The proposed mechanism is investigated through the case study of biped figures.

AGRADECIMENTOS

Aqui manifesto meus sinceros agradecimentos ao meu orientador e amigo, Prof. Dr. Léo Pini Magalhães, pela oportunidade concedida, confiança e orientação durante o desenvolvimento deste trabalho.

Agradeço à CAPES, pelo apoio financeiro através da bolsa de mestrado, indispensável para a realização desta pesquisa.

Agradeço ao Departamento de Engenharia de Computação e Automação Industrial da Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, pela infra-estrutura disponibilizada.

Agradeço à FUNCAMP, pelo apoio institucional que possibilitou o término deste trabalho.

Agradeço ao Prof. Dr. Norman Badler e seu grupo de pesquisa da Universidade da Pensilvânia, por enviar-nos a biblioteca de programação IKAN e bibliografia relacionada. *Thanks to Prof. Norman Badler and his research group at the University of Pennsylvania, for sending us the IKAN library and related bibliography.*

Nesta oportunidade, gostaria de agradecer em especial aos meus pais Iduarte e Norélia e à minha irmã Carla, pela inesgotável fonte de amor, carinho, incentivo e suporte que propiciaram durante toda a minha jornada acadêmica.

Agradeço à Noêmia, secretária da pós-graduação/FEEC, à Juraci, secretária da diretoria/FEEC e à Carmen, secretária do DCA/FEEC, pelo carinho e atenção com que sempre me atenderam.

Sem citar nomes para não ser injusto, agradeço a todos os amigos do LCA que de alguma forma contribuíram neste trabalho, proporcionando um ambiente agradável e enriquecedor.

Aos meus queridos pais Iduarte e Norélia

e à minha querida irmã Carla.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

A animação por computador é uma área de pesquisa pertencente à computação gráfica que tem um amplo domínio de aplicações como entretenimento, educação, indústria, visualização científica, entre outras.

Na sua definição mais simples, a animação por computador refere-se ao processo no qual é gerada uma série de quadros de um grupo de atores[1], onde cada quadro é uma alteração do quadro anterior. Estes quadros são exibidos seqüencialmente a uma determinada taxa[2], produzindo a ilusão de continuidade no decorrer do tempo [Thalmann, 85]. Embora freqüentemente as pessoas tenham em mente que a animação seja sinônimo de movimento, esta não abrange somente isto, mas quaisquer alterações visuais que a cena apresente, tais como variações na luz, na geometria, na cor, na textura dos atores, etc., decorrentes das ações que ocorrem em uma animação.

Em um sistema de animação, estas ações somente são possíveis porque há um mecanismo que controla os parâmetros da cena necessários para produzir um efeito visual desejado. Em simulações[3] de movimentos dos atores, as técnicas empregadas geralmente utilizam o cálculo diferencial e as leis físicas, tais como as Leis Dinâmicas de Newton e de Euler, para obter um movimento desejado. Desta forma, a natureza complexa dos diversos fenômenos físicos nos leva, muitas vezes, a ter de lidar com sistemas de equações diferenciais para representar corretamente os fenômenos de interesse.

A geração de uma animação pressupõe a existência de um modelo para gerar as ações desejadas. Como será visto ao longo deste trabalho, estes modelos podem estar situados em

[1] O termo "ator" ou "personagem" em animação refere-se aos objetos pertencentes à cena.
[2] taxa corresponde a quadros por segundo.
[3] Note que, em geral, utilizamos indistintamente os termos "animação" e "simulação".

diferentes níveis de abstração, desde aqueles voltados, por exemplo, ao próprio movimento de um ator até aqueles voltados para modelar o comportamento da animação. Neste contexto, define-se controle como sendo a atribuição de valores a parâmetros de algum modelo utilizado para gerar a animação desejada.

Em [Camargo, 95a], foi apresentada uma proposta para a divisão do problema de controle de modo a simplificar o tratamento de animações complexas. As partes envolvidas nesta divisão foram denominadas bloco de controle local e bloco de controle global[4].

Supondo uma determinada animação composta por diversos atores, o bloco de controle local definirá o comportamento de cada ator. Por exemplo, as leis físicas que regem o movimento de um ator estarão embutidas em seu respectivo módulo de controle local. Como há mais de um ator envolvido na cena, poderão ocorrer interações entre os atores e entre estes e o ambiente. Assim sendo, o bloco de controle global terá a finalidade de coordenar o fluxo da animação, sugerindo uma abordagem baseada em lógica para tal tarefa. Por exemplo, caso ocorra um determinado evento, este bloco saberá quais atores serão afetados e como eles deverão responder. Assim, dependendo da seqüência de eventos na animação os atores se comportarão de uma dada forma, podendo ser diferente caso outra seqüência ocorra.

Podemos perceber que a definição de bloco de controle local e bloco de controle global é dependente do contexto e do nível de abstração em que desejamos trabalhar. Em uma animação de figuras bípedes que se assemelham a seres humanos, por exemplo, se estivermos tratando o controle de uma única figura, poderíamos supor que haveria um bloco de controle local para tratar os movimentos de cada membro, enquanto um bloco de controle global coordenaria as interações entre os membros. De outra forma, para mais de uma figura poderíamos supor que haveria blocos de controle local para coordenar os movimentos de cada figura, enquanto o bloco de controle global coordenaria as interações entre as figuras e o ambiente (Figura 1.1).

Através do paradigma de blocos de controle o animador pode, a partir do controle global, coordenar a animação através de diretivas que descrevem o comportamento por

[4] Na verdade, blocos locais e globais de modelagem e controle.

meio de eventos, reações, etc., enquanto no bloco de controle local a ação é produzida utilizando-se técnicas de animação por computador.

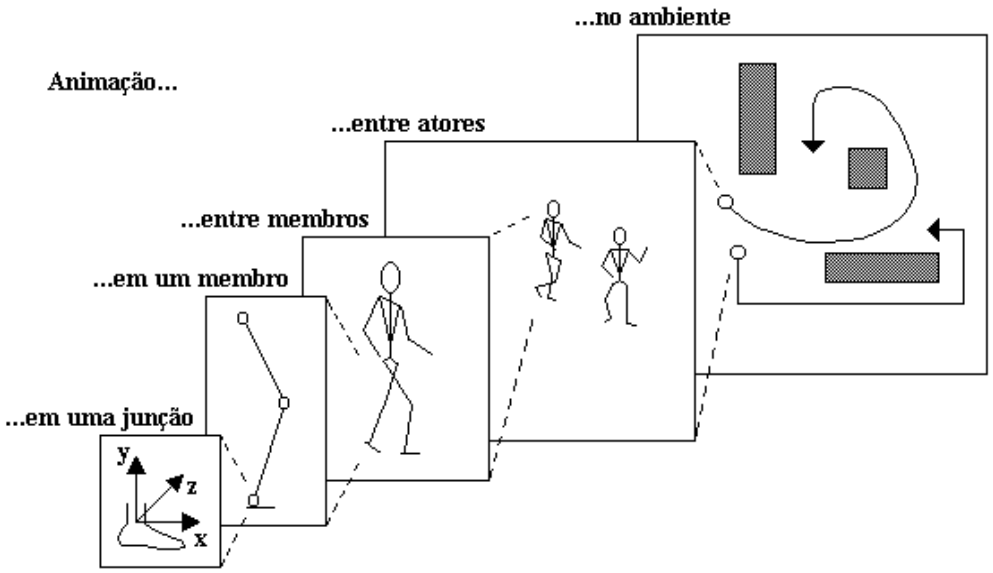


Figura 1.1 - Possíveis níveis de abstração para o controle do movimento da animação de figuras humanas [Bruderlin, 94].

1.1. Objetivo do trabalho

Este trabalho propõe uma abordagem de controle de animações que utiliza o paradigma de blocos de controle local e global, explorando redes de Petri (PNs - *Petri Nets*) no bloco de controle global. Para tanto, verificaremos que animações possuem características análogas a Sistemas Dinâmicos a Eventos Discretos (SDEDs), viabilizando a utilização de redes de Petri em animações por se tratar de uma ferramenta muito empregada em SDEDs. A notação gráfica de uma PN possibilita uma fácil compreensão e encapsulamento de detalhes, oferecendo uma hierarquia de modelagem e controle adequada para animações. Quanto ao controle local, o animador terá a liberdade de escolher a técnica de controle em animação que melhor lhe convenha para produzir a ação desejada.

1.2. Organização do trabalho

No próximo capítulo apresentamos as principais classificações dos sistemas de animação por computador, conforme seus mecanismos de controle, além de alguns trabalhos relacionados à nossa abordagem.

No Capítulo 3 apresentamos a nossa abordagem de controle de animações que utiliza o paradigma de blocos de controle, explorando redes de Petri como o mecanismo de controle global em animações. O capítulo apresenta o algoritmo e a biblioteca de programação provenientes desta abordagem e uma linguagem para descrever a topologia de uma PN.

No Capítulo 4 é apresentado o estudo de casos utilizando PNs em diferentes níveis de abstração para controlar os movimentos e as interações de figuras bípedes que se assemelham a seres humanos.

No Capítulo 5 encontram-se as conclusões e as perspectivas para trabalhos futuros.

No Apêndice A é feita uma apresentação formal sobre redes de Petri.

No Apêndice B descrevemos os principais métodos disponibilizados pela classe SRS da biblioteca de cinemática inversa denominada IKAN (*Inverse Kinematics using ANalytical Methods*), utilizada como ferramenta de controle local no estudo de casos apresentados no Capítulo 4.

No Apêndice C apresentamos as PNs utilizadas no estudo de casos em linguagem de descrição topológica.

Por fim, no Apêndice D encontra-se o artigo publicado relacionado ao conteúdo deste trabalho.

CAPÍTULO 2

SISTEMAS DE ANIMAÇÃO POR COMPUTADOR

Este capítulo aborda as principais classificações dos sistemas de animação propostas no decorrer da história conforme seus mecanismos de controle. Além disso, alguns trabalhos relacionados à nossa proposta de controle de animações por computador também são apresentados.

2.1. Introdução

Em um sistema de animação por computador há, pelo menos, três componentes fundamentais:

1. Um modelador geométrico, para definição e modelagem das características geométricas dos atores envolvidos na animação.
2. Um mecanismo de controle da animação, para definição de um modelo para gerar as ações desejadas e para atribuição de valores a parâmetros deste modelo.
3. Um mecanismo de *rendering* e visualização, para definição dos atributos da imagem, como projeção, dimensão da tela, etc. e do ator, como cor, textura, etc., além da visualização dos quadros gerados durante o processo de animação.

Em relação ao mecanismo de controle, na literatura especializada são encontradas diversas técnicas de controle aplicáveis a determinadas animações, mas nenhuma é genérica a ponto de ser utilizada em qualquer animação. Ou seja, uma solução de controle conveniente para determinado problema pode trazer resultados ruins se aplicada a outro.

De modo a compreendermos melhor estas técnicas de controle e como elas são utilizadas em animações por computador, a seguir apresentaremos as principais classificações dos sistemas de animação, conforme seus mecanismos de controle, de maneira a situarmos o nosso trabalho neste contexto.

2.2. Classificação dos sistemas de animação

Uma das primeiras tentativas de classificar os sistemas de animação foi apresentada em [Zeltzer, 85]. Este trabalho propõe uma classificação consistindo de três níveis: o nível guiado, o nível do animador e o nível de tarefas (em [Tost, 88] foi apresentada uma classificação similar, sendo que o nível do animador foi definido como nível de programa).

Sistemas no nível guiado descrevem o movimento do ator sem nenhum mecanismo de abstração, pois os dados cinemáticos do movimento são obtidos a partir, por exemplo, de rotoscopia ou da técnica de animação *keyframe* (maiores detalhes sobre esta técnica podem ser encontrados na Subseção 3.4.2.1).

Em sistemas no nível do animador, o movimento do ator é especificado algoritmicamente, fornecendo um maior controle e abstração sobre a animação, sendo apropriado para a definição de movimentos complexos. O poder computacional de uma linguagem de programação é disponibilizado ao animador, muito embora esta "facilidade" possa impor problemas associados ao uso desse ferramental computacional caso não seja oferecida uma interface apropriada.

Sistemas no nível de tarefas especificam o movimento do ator em termos de eventos e relacionamentos. Para tanto, são necessárias informações sobre posições, atributos físicos e funcionalidades dos atores no ambiente. Com o controle no nível de tarefas o animador somente esquematiza o movimento em termos gerais, enquanto o sistema de animação fica com a tarefa de resolver os detalhes. Por exemplo, em [Zeltzer, 83] foi esboçada uma abordagem no nível de tarefas onde o comportamento é definido através de uma hierarquia de habilidades comportamentais (*skills*) selecionadas por diretivas que mapeam a ação atual e o contexto para a próxima ação desejada.

Qual destas abordagens é a mais apropriada dependerá da aplicação e, portanto, não há sistemas genéricos. Um animador inexperiente pode ficar satisfeito com os atores e os seus respectivos movimentos predefinidos que um sistema de animação no nível de tarefas provê. Por sua vez, um animador mais experiente talvez deseje um sistema no nível guiado ou do animador, pois estes forneceriam um controle maior sobre os movimentos.

P. M. Isaacs [Isaacs, 87] propôs uma outra classificação dos sistemas de animação, dividindo-os em três classes: sistemas baseados em animação *keyframe*, em animação procedimental e em simulação dinâmica (de forma similar, N. M. Thalmann e D. Thalmann [Thalmann, 87] classificaram os sistemas em duas classes: sistemas baseados em animação *keyframe* e em animação algorítmica, este último correspondente à animação procedimental e à simulação dinâmica).

Embora sistemas baseados em animação *keyframe* forneçam quase que completo controle da cena ao animador, a utilização de interpolações para calcular os quadros intermediários dificulta a criação de seqüências dinamicamente corretas, além de ser muito trabalhosa.

Sistemas de animação procedimental fundamentam-se na capacidade do computador em determinar a cinemática de uma seqüência baseando-se em instruções ao invés de posições explícitas. Em alguns sistemas, como em [Girard, 85], princípios dinâmicos também são utilizados para se obter um certo grau de realismo.

Sistemas de simulação dinâmica requerem, como entrada, as características físicas e dinâmicas dos atores. As características físicas incluem descrições de todas as suas articulações bem como suas conectividades, enquanto as características dinâmicas incluem descrições como momento de inércia, forças e torques aplicados ao ator, etc. A utilização deste tipo de sistema apresenta um maior realismo do movimento gerado e possibilita simular um grande número de fenômenos físicos. Entretanto, apresenta uma maior complexidade e um maior número de variáveis a serem controladas.

Com a evolução das técnicas de controle em animação surgiram sistemas capazes de oferecer um melhor tratamento à interação entre atores, entre ator e o ambiente e entre ator e o animador. Desta forma, estes conceitos de interação exigiram a definição de novos

paradigmas para sistemas de animação, havendo a necessidade de reformular as classificações de modo a que estas se tornassem mais abrangentes.

N. M. Thalmann e D. Thalmann [Thalmann, 91], propuseram uma classificação de sistemas para a geração de cenas animadas por computador envolvendo atores sintéticos em função do Método de Controle do Movimento (MCM) e dos tipos de interação entre os atores e entre estes e o ambiente.

Um Método de Controle do Movimento especifica como um ator é animado. Podemos caracterizar um MCM levando em consideração o tipo de informação relevante no controle de uma animação. Se utilizarmos a técnica de animação *keyframe* para uma figura articulada, as principais informações são os ângulos das articulações. Em um sistema baseado nas leis dinâmicas, a informação privilegiada consiste no conjunto de forças e torques presentes no controle. Neste caso, sabemos que a solução destas equações dinâmicas será justamente os ângulos das articulações necessários para movimentar o ator da forma desejada.

Assim sendo, as principais informações para o controle do movimento de atores dão origem a três categorias de MCMs: geométricos, físicos ou comportamentais.

Os MCMs geométricos usam informações de ordem geométrica como coordenadas e ângulos. Embora os MCMs geométricos concentrem-se principalmente na determinação do movimento de esqueletos, eles também podem ser utilizados no cálculo de deformações de corpos ou faces.

Os MCMs físicos usam leis e características físicas como base para o cálculo do movimento. As informações relevantes para estes MCMs incluem características como massa, momentos de inércia e rigidez. As leis físicas envolvidas são principalmente aquelas da mecânica, particularmente da dinâmica. Estas leis auxiliam o controle do movimento de esqueletos, tendo também uma importante aplicação no cálculo de deformações de corpos e faces, pois a origem destas deformações é muscular e as ações de um músculo são mais apropriadamente caracterizadas em termos mecânicos.

Os MCMs comportamentais especificam o movimento de um ator em termos de seu comportamento, freqüentemente definido como o modo através do qual os seres vivos agem. Estes termos não são necessariamente de fácil tradução em movimentos. Alguns exemplos incluem um ator sorrindo para outro, falando uma certa frase ou levantando-se da cadeira e caminhando até a porta. Os termos que descrevem variações de personalidade de um único ator e entre atores também são informações privilegiadas para MCMs comportamentais.

Além dos MCMs, em [Thalmann, 91] a relação entre o ator e o resto do seu universo, ou seja, a sua interface também foi considerada. Esta interface pode ser descrita por quatro casos básicos:

- Somente o ator: não há interação; ocorre quando o ator se encontra sozinho na cena e não interage com outros atores. Se a cena possuir obstáculos ou outros atores que estejam em movimento, o ator em questão não tem consciência deste ambiente. Somente o animador poderá prever possíveis colisões através do cálculo apropriado das trajetórias.
- Interface ator-ambiente: ocorre no caso em que os atores estão se movendo em um ambiente do qual eles têm conhecimento. Isto significa que o movimento do ator depende em parte do ambiente, evitando obstáculos, segurando objetos, etc.
- Interface ator-ator: ocorre quando as ações realizadas por um ator são conhecidas por outro ator, influenciando no seu comportamento, sem a participação do animador.
- Interface animador-ator: ocorre quando o animador não apenas pode enviar informações ao ator, mas também o ator pode responder, enviando informações ao animador.

Para cada tipo de interface descrita acima são consideradas as três categorias de MCM - geométrico, físico e comportamental - ajustando o melhor controlador para cada caso. A Tabela 2.1 representa um MCM para cada tipo de interface, com o incremento de sua complexidade podendo ser observado de cima para baixo e da esquerda para direita.

	MCMs Geométricos	**MCMs Físicos**	**MCMs Comportamentais**
Único ator	Rotoscopia Animação *keyframe* Restrições cinemáticas	Dinâmica	Modelo muscular Sistema facial
Interface Ator-ambiente	Desvio de obstáculos Detecção interseções	Modelos de colisão Modelos deformáveis	Sensores Desvio de obstáculos baseados na visão
Interface Ator-ator	Colisões geométricas entre atores	Colisões dinâmicas entre atores	Comunicação de emoções entre atores
Interface Animador-ator	Projeto de trajetórias utilizando dispositivos (*mouse,spaceball*, etc.)	Entrada de forças e dispositivos de realimentação	Entrada visual via câmera de vídeo

Tabela 2.1 - Classificação dos sistemas de animação conforme os Métodos de Controle do Movimento (MCM) e da interface do ator [Thalmann, 91].

Na próxima seção apresentaremos alguns trabalhos que propuseram métodos de controle orientados a eventos em animações, já que estão diretamente relacionados à proposta que será apresentada no próximo capítulo.

2.3. Trabalhos relacionados

Algumas abordagens para controle orientado a eventos em animações foram propostas na literatura.

Tomovic e McGhee [Tomovic, 66] sugerem a aplicação da teoria de autômatos de estados finitos para facilitar a análise e síntese de dispositivos de controle em sistemas de bioengenharia. A fim de prover uma conexão entre a teoria de autômatos e a teoria de controle em sistemas contínuos, é proposto o conceito de atuador cibernético (*cybernetic actuator*). Este atuador consiste de um circuito combinatório alimentado por duas entradas binárias assumindo quatro possíveis estados e uma saída determinada a partir dos valores

de entrada, como apresentado na Tabela 2.2 (a numeração dos estados e a codificação binária da entrada são arbitrárias).

Entrada	Estado do atuador	Saída
00	0	Livre
01	1	Decrescente
10	2	Crescente
11	3	Travado

Tabela 2.2 - Definição de um atuador cibernético.

Quando usado como um elemento de um sistema de bioengenharia, é esperado que um atuador cibernético seja capaz de rotacionar uma junta somente entre dois limites fixos. Então, a Tabela 2.2 deve ser interpretada em termos do comportamento do atuador dentro do conjunto de ângulos permitidos (conjunto alcançável). É importante compreender que enquanto a entrada de um atuador cibernético assume somente quatro estados distintos, a saída pode ser dirigida para qualquer ponto no conjunto alcançável. É esta importante propriedade de um atuador que permite associar a teoria de autômato e a teoria de controle em sistemas contínuos.

De maneira a ilustrar a viabilidade deste tipo de controle, em [Tomovic, 66] o comportamento de uma perna na posição de apoio durante o caminhar é descrito através de um modelo de estado finito. Assim, foi projetado um controlador de estado finito para coordenar os movimentos do tornozelo e do joelho em uma prótese ortopédica. Esta metodologia também foi explorada em sistemas de animação por computador.

Zeltzer [Zeltzer, 82] descreve o projeto e a implementação de um sistema de animação que coordena os movimentos de uma estrutura articulada semelhante a um esqueleto humano utilizando um controlador de estados finitos. Como entrada para o sistema, uma

seqüência de tarefas é descrita (*task description*) como, por exemplo, "ir até a porta e abrí-la", "sentar-se" ou "correr com a bola e chutá-la". O gerenciador de tarefas (*task manager*), no maior nível de abstração do mecanismo de controle, recebe as descrições das tarefas e organiza-as em uma fila de movimentos (*movement queue*). Após a organização da fila, o gerenciador de tarefas supervisiona a execução dos movimentos implementados por procedimentos denominados "programas de movimento" (*motor programs*), que são máquinas de estados finitos. Estes programas executam determinadas classes de movimentos, tais como o "caminhar" ou o "correr" (há parâmetros que permitem ajustar estes movimentos). Por sua vez, os estados pertencentes aos "programas de movimento" invocam um conjunto de programas com menor nível de abstração denominados "programas de movimento local" (*local motor programs*), que também são máquinas de estados finitos que controlam as junções. Desta maneira, esta modelagem possibilita a construção de estruturas hierárquicas que descrevem o movimento desejado. Em contrapartida, uma desvantagem desta abordagem é que o animador perde o controle artístico em favor da síntese do movimento.

O trabalho de Fishwick e Porr [Fishwick, 91] apresentou uma abordagem que combina uma ferramenta de modelagem e conntrole a eventos discretos com a técnica de animação *keyframe*. Esta abordagem utilizou PNs para representar a dinâmica do sistema em um maior nível de abstração, possibilitando que o controle de um sistema complexo possa ser representado através de uma hierarquia. O foco deste trabalho foi estudar modelos matemáticos adotados em simulações por computador que podem ser utilizados nas pesquisas em computação gráfica desenvolvidas pela comunidade científica. Os autores apresentaram a animação dos filósofos jantando, um exemplo conhecido sobre sincronização entre processos em sistemas operacionais. A abordagem utilizou a temporização nas transições[1], pois estas representaram as possíveis posições de cada filósofo na animação como, por exemplo, sua mão esquerda apoiada na mesa ou próxima da sua boca.

[1] Maiores informações sobre redes de Petri podem ser encontradas no Apêndice A.

Kalra e Barr [Kalra, 92] apresentaram um tratamento consistente de tempo e eventos para animação por computador. Eles utilizaram formalmente o conceito de evento para criar um objeto denominado unidade de evento (*event unit*). Utilizando esta unidade de evento como uma "primitiva de tempo", o trabalho apresenta uma série de esquemas organizacionais hierárquicos para projetar seqüências de movimentos complexas. A unidade de evento também se mostrou viável para particionar extensas seqüências de animação, dividindo-as em unidades menores e conectando-as por meio de eventos. Os autores apresentam alguns exemplos de animação onde técnicas de controle como cinemática e dinâmica são combinadas com a metodologia proposta no trabalho, mostrando serem produtivas tais associações.

Van de Panne et al. [van de Panne, 94] propuseram um método para projetar modos de locomoção periódicos para figuras articuladas utilizando grafos de controle de pose cíclica (*Cyclic pose control graphs*). Um grafo de controle de pose é uma máquina de estados que tem uma pose particular associada a cada estado. A pose representa uma configuração interna desejada do ator quando o controlador está em um determinado estado no grafo. A pose então especifica a forma desejada do ator, mas não a sua posição ou a sua orientação na cena. A posição e orientação do corpo são determinadas pela interação do ator com o seu ambiente. O controlador então produzirá torques de modo a posicionar o ator conforme esta interação. Estes torques são calculados através de controladores PD (*Proportional-Derivative*) para cada junção. O controlador passará para a próxima pose após ter expirado o tempo indicado pelo arco de transição entre o atual estado e o próximo. Do ponto de vista de animadores, grafos de controle de pose podem ser vistos com uma extensão natural de *keyframe* para animações baseadas em modelos físicos. Os grafos de controle de poses combinados com controladores PD, além do uso de artifícios para descrever o movimento em termos de ciclos limites, bifurcações e movimentos potencialmente caóticos, são mecanismos comuns em animações baseadas em modelos físicos (*physically-based animations*) [Hodgins, 95][van de Panne, 96][Laszlo, 96][Laszlo, 00].

Em [Camargo, 95a] foi apresentado o controle de uma figura bípede que se assemelha a um ser humano conforme apresentado na Figura 2.1. No caso, com o propósito de simplificar o sistema, apenas o controle das pernas é tratado. Mas a idéia pode ser

extensível aos braços, uma vez que ambos membros possuem os mesmos graus de liberdade (DOFs - *degrees of freedom*). Para o controle das interações entre os membros é proposto um esquema orientado a eventos para resolver o problema do controle global, sendo que o tempo não é mais o único fator determinante para o desencadeamento das ações dos atores em uma animação (no caso, cada membro da estrutura foi considerado um "ator"). O sistema simulado foi tratado como um DEDS (*Discrete Event Dynamics Systems*), sendo modelado por ESMs (*Extend State Machines*). Maiores detalhes desta proposta podem também ser encontrados em [Camargo, 94][Camargo, 95b][Camargo, 95c].

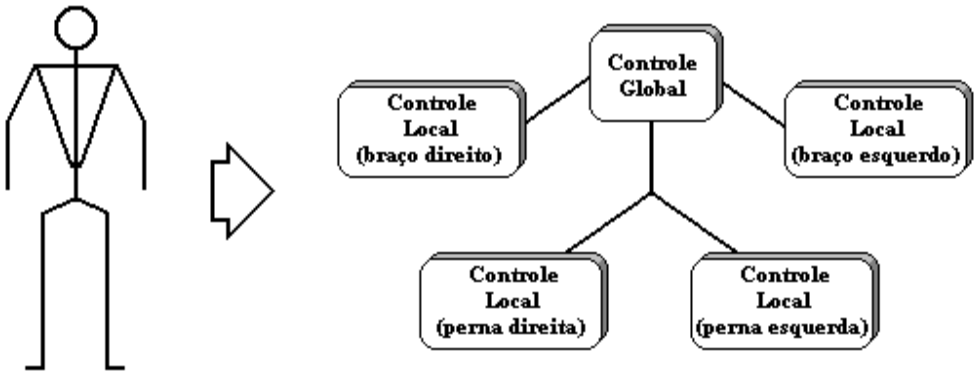


Figura 2.1 - Modelagem de uma figura articulada utilizando o paradigma de blocos de controle.

As PaT-Nets (*Parallel Transition Nets*), propostas por [Becket, 94], são máquinas de estado executadas em paralelo que facilitam a ordenação das ações baseadas no atual estado do ambiente da animação ou do próprio sistema de animação. Estas estruturas caracterizam as tarefas em progresso, condições a serem monitoradas, recursos usados ou qualquer outra finalidade que necessite de sincronização.

Um exemplo de PaT-Net é apresentado na Figura 2.2. Cada descrição da rede é uma classe (estrutura orientada a objetos) contendo um número de nodos conectados através de arcos. A transição de um nodo para outro é feita através da avaliação de uma expressão em Lisp associada a cada arco. O arco selecionado será aquele que primeiro apresentar a condição da expressão verdadeira. Assim como os arcos, os nodos também possuem expressões em Lisp, representando ações a serem executadas. Em relação aos monitores

(por exemplo, o `Monitor 1`, da Figura 2.2), estes executam uma ação se uma condição geral for verdadeira, não levando em consideração em qual nodo da rede o controlador esteja.

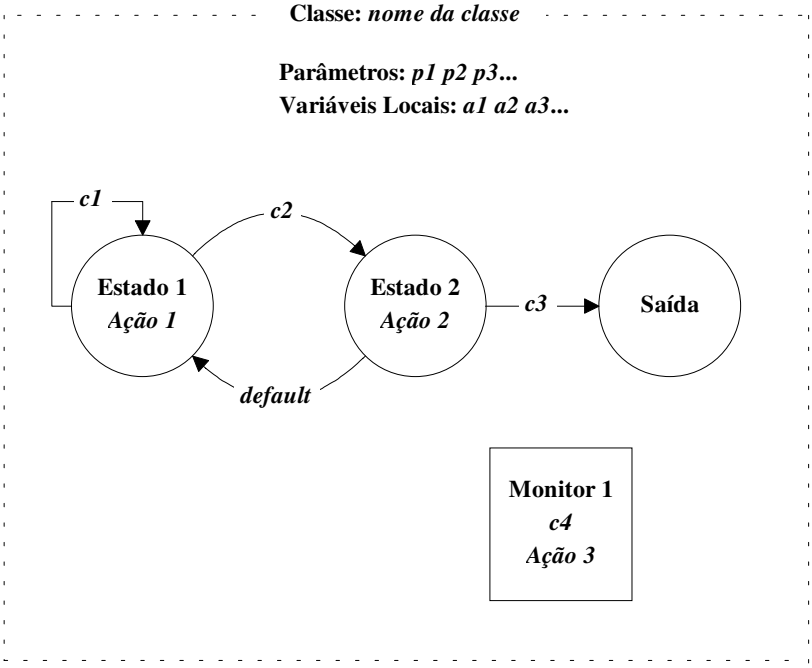


Figura 2.2 - A notação gráfica de uma simples PaT-Net [Becket, 94].

Uma rede é criada através da instanciação da classe PaT-Net. Uma rede pode ter variáveis locais disponíveis para todas as ações e as condições, assim como parâmetros inicializados na sua instanciação. As PaT-Nets são controladas por um sistema operacional que tem condições de interpretar expressões em Lisp, destinando um tempo de processamento (*time-slice*) para cada uma das redes. Este sistema permite expandir novas redes, encerrar a execução de outras, permite a comunicação entre as redes através de semáforos e filas de prioridades ou colocar uma rede no modo de espera (*sleep*) até que uma condição seja satisfeita.

Conforme Badler et al. [Badler, 99], os benefícios de uma PaT-Net derivam não somente da sua organização que permite execuções em paralelo, mas também da sua

estrutura condicional derivada das expressões Lisp. Ferramentas de animação tradicionais usam linhas do tempo (*timelines*) onde as ações são estabelecidas e ordenadas. Uma PaT-Net provê um modelo de controle onde as ações são executadas, modificadas e interrompidas a partir das transições entre os nodos. Este tipo de animação é um passo crucial em direção a comportamentos autômatos, visto que a execução condicional permite que um agente reaja e tome decisões.

2.4. Considerações finais

Neste capítulo apresentamos as principais classificações dos sistemas de animação segundo seus mecanismos de controle e alguns trabalhos relacionados à abordagem que será apresentada no próximo capítulo.

Em relação às classificações apresentadas, a proposta do próximo capítulo é um sistema no nível de tarefas [Zeltzer, 85][Tost, 88] ou um MCM comportamental [Thalmann, 91], não sendo possível, entretanto, encontrar uma classificação correspondente em [Isaacs, 87].

A abordagem proposta, em relação ao trabalho de [Camargo, 95a], utiliza PNs ao invés de ESMs no nível de controle global. Com uma PN é possível prever antecipadamente o comportamento de uma animação em relação ao desencadeamento dos eventos no nível de controle global, possibilitando também definir hierarquias de modelagem e de controle para os movimentos do ator, como proposto em [Zeltzer, 82]. O trabalho de [Fishwick, 91] utilizou PNs temporizadas nas transições, diferentemente da abordagem a apresentada no próximo capítulo que utiliza a temporização nos lugares. Já o trabalho de [Becket, 94] explora o uso de expressões em Lisp para representar condições e/ou ações, diferentemente da estrutura condicional de PNs.

No próximo capítulo, apresentaremos a nossa abordagem utilizando redes de Petri para modelar e controlar animações, além de alguns aspectos de implementação sobre a biblioteca de programação desenvolvida.

CAPÍTULO 3

SISTEMA DINÂMICO A EVENTOS DISCRETOS EM ANIMAÇÃO POR COMPUTADOR

Este capítulo apresenta a abordagem de controle de animações utilizando o paradigma de blocos de controle [Camargo, 95a], empregando a teoria de Sistemas Dinâmicos a Eventos Discretos (SDEDs) para modelar o bloco de controle global. Neste bloco nós exploramos o uso de PNs, uma ferramenta muito empregada em SDEDs, e apresentamos também possíveis técnicas de controle em animação que podem ser aplicadas no bloco de controle local. Além disso, este capítulo apresenta o algoritmo e a biblioteca de programação provenientes desta abordagem e uma linguagem para descrever a topologia de uma PN.

3.1. Introdução

Em um sistema de animação por computador, uma ferramenta de modelagem e de controle tem como finalidade definir os parâmetros adequados para produzir um efeito visual desejado. Como uma animação, em geral, exige uma quantidade de informações muito grande a manipular, torna-se conveniente a disponibilidade de uma ferramenta que possibilite minimizar a definição da seqüência da animação, onde os comandos disponíveis descreveriam o comportamento a partir de eventos, reações, restrições, etc. Neste tipo de ferramenta o principal objetivo não deve ser a definição de como as ações serão executadas, mas o que deverá ser executado na seqüência da animação. Com isso, proveríamos uma abordagem que apresentasse uma hierarquia contendo as facilidades disponibilizadas por uma ferramenta de modelagem e de controle em alto nível de abstração que também englobaria assertivas relacionadas às abstrações de baixo nível.

Por exemplo, para animar um único ator com somente 6 graus de liberdade durante 5 segundos, a uma taxa de 30 quadros por segundo, estão envolvidos cerca de 900 valores numéricos. Animar uma figura humana definida com mais de 200 graus de liberdade é uma tarefa de ordem de grandeza muito maior. Utilizando uma ferramenta que apresentasse uma hierarquia de modelagem e de controle seria possível reduzir a quantidade de dados a manipular, permitindo que o animador especifique apenas as informações relevantes para a seqüência da animação, incumbindo ao computador a tarefa de gerar as demais informações não definidas.

Assim sendo, neste capítulo apresentaremos a nossa abordagem baseada em eventos discretos e no paradigma de blocos de controle proposto em [Camargo, 95a]. Utilizaremos redes de Petri como ferramenta de controle global de animações, enquanto no controle local o animador escolherá a técnica em animação que melhor lhe convenha para produzir de fato a ação.

3.2. Controle orientado pelo tempo *versus* controle orientado a eventos

Em ambientes de animação, o controle é geralmente realizado por métodos que utilizam o domínio do tempo para ordenar as ações. Para este tipo de abordagem, o animador projeta a seqüência da cena tendo como referencial uma linha do tempo (*timeline*) para indicar, por exemplo, o momento em que determinado comportamento de um ator deve ser iniciado e o período em que ele será realizado.

A Figura 3.1 ilustra uma situação exemplo, descrevendo a seguinte seqüência. A esfera A se movimenta de p0 a p1 no período t0 a t1, colidindo com a esfera B. Com a colisão, a esfera B se movimenta de p1 a p2 no período t1 a t2, colidindo com a esfera C. Com a colisão, a esfera C se movimenta de p2 a p3 no período t2 a t3.

Este paradigma encontra-se direcionado às restrições de espaço-tempo [Witkin, 88]. O exemplo apresentado possui um comportamento orientado pelo tempo (ou seja, o passar do tempo determina a evolução da animação), podendo ser descrito por relações algébricas entre as suas variáveis (tempo e posição).

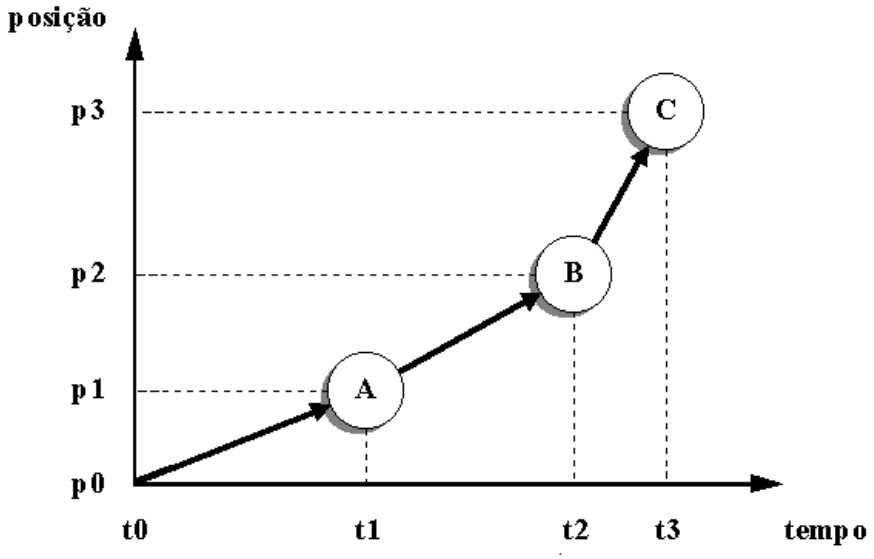


Figura 3.1 - Exemplo de uma animação usando a linha do tempo. O animador especifica as colisões entre as esferas em função de quando (tempo) e onde (posição).

Em contrapartida, também é possível descrever esta mesma seqüência de animação através de um modelo orientado a eventos. Neste caso, a descrição da seqüência da Figura 3.1 resume-se a:

"A **bola A** movimenta-se até colidir com a **bola B.** Por sua vez, a **bola B** movimenta-se até colidir com a **bola C**, que se movimenta até parar".

Os seguintes eventos:

- o início do movimento da **bola A**;
- a colisão da **bola A** com a **bola B**;
- a colisão da **bola B** com a **bola C**;
- o fim do movimento da **bola C**,

são as informações relevantes no desencadeamento da seqüência da animação, uma vez que indicam os comportamentos das esferas na cena. A seguir, apresentaremos alguns conceitos básicos de modo a considerarmos uma animação como um Sistema Dinâmico a Eventos Discretos.

3.3. Conceitos da teoria de Sistemas Dinâmicos a Eventos Discretos [1]

Nesta seção, apresentamos alguns conceitos básicos da teoria de Sistemas Dinâmicos a Eventos Discretos, de maneira a situar a animação por computador neste contexto.

- Sistema

Sistema é um conceito primitivo cuja definição é mais intuitiva do que exata. Mas, cabe salientar que em qualquer definição sobre este assunto alguns aspectos inerentes devem estar presentes como, primeiramente, um sistema é uma agregação e interação de "componentes" e, segundo, que estes "componentes" agem em conjunto para desempenhar uma determinada "função".

- Sistema dinâmico

Um sistema dinâmico é aquele sistema que evolui ao longo do tempo, sendo que, em geral, seu comportamento depende do passado.

- Modelo

Um modelo é definido como um dispositivo teórico que, de alguma maneira, descreve o comportamento de um sistema. Em termos gerais, são definidas as variáveis de entrada e as variáveis de saída, enquanto o modelo estabelece as relações entre estas variáveis.

- Estado

O conceito de estado é fundamental para o estudo de sistemas dinâmicos. O estado de um sistema, de um modo genérico, constitui a informação necessária para se conhecer o valor futuro das variáveis do modelo, desde que se conheçam as entradas. Esta definição, embora qualitativa, é a mais conveniente para o estado. Outras definições de natureza

[1] Baseada em [Cassandras, 93] e [Mendes, 99].

quantitativa mostram-se excessivamente restritivas ou aplicáveis somente a sistemas particulares. Em relação a um sistema a eventos discretos, a informação correspondente ao estado pode ser de natureza variada como, por exemplo, números inteiros, números reais, variáveis booleanas, etc.

- Evento

Assim como para o termo sistema, não há uma definição formal sobre o que é um evento. E, assim como o conceito sobre sistema, o conceito de evento também é primitivo, contendo uma boa base intuitiva. O importante é enfatizar que um evento deve ser pensado como uma ocorrência instantânea (sem duração temporal) que altera o estado do sistema.

Geralmente, uma descontinuidade no sistema está associada a um evento, como no caso da colisão entre dois corpos. Entretanto, um evento nem sempre ocasiona uma descontinuidade. Um evento pode ser identificado como uma ação específica (por exemplo, a ação de pressionar um determinado botão). Um evento pode também ser visto como uma ocorrência espontânea ditada pela natureza ou pode também ser o resultado de várias condições. Um evento pode ser programado por outro evento ou ocorrer aleatoriamente e, em um mesmo sistema, é possível haver vários tipos de eventos. Esta multiplicidade de tipos e causas de eventos é que, em geral, leva um sistema discreto a apresentar grande complexidade.

- Sistemas Dinâmicos a Eventos Discretos

Os Sistemas Dinâmicos a Eventos Discretos (SDEDs) são caracterizados por apresentarem variáveis de estado discretas e serem dirigidos por eventos. A rigor, a primeira condição relacionada acima não caracteriza propriamente um sistema discreto, visto que sua inobservância não impede que um sistema apresente dinâmica discreta. O segundo ponto é talvez o mais importante na caracterização dos sistemas discretos. Sua dinâmica é dirigida por eventos, ou seja, o que determina a evolução do sistema é a ocorrência de eventos e não simplesmente o passar do tempo. Deve ser observado que embora o tempo continue sendo um parâmetro importante na caracterização da dinâmica do sistema, ele não é o fator determinante no sistema.

3.4. Um mecanismo de controle para animações por computador

Para projetarmos um mecanismo de controle para uma animação modelada por computador pensamos em um modelo computacional que descreva o comportamento desta. Se analisarmos uma animação sob a óptica da teoria de SDEDs poderemos verificar que esta possui características implícitas a um sistema, como a combinação de componentes que agem em conjunto (no caso, os atores) e a função a ser desempenhada (no caso, a animação de algo real ou não). Assim, propomos trabalhar em uma animação levando em consideração que esta pode ser vista como um sistema a ser modelado e controlado.

3.4.1. O nível de controle global

O nível de controle global, como já mencionado anteriormente, tem a finalidade de controlar o fluxo da animação, podendo ser sugerida uma abordagem baseada em lógica para tal tarefa. Sendo assim, este mecanismo de controle pode ser fundamentado em um SDED. O controle por meio do paradigma orientado a eventos permite que o animador especifique, com certa facilidade, a seqüência da animação desejada, uma vez que possíveis mudanças podem ser realizadas sem a necessidade de remodelar toda ou grande parte desta seqüência [Kalra, 92]. Além disso, também possibilita que animações de figuras complexas sejam criadas por meio da decomposição em figuras mais simples, definindo uma hierarquia de modelagem e de controle do movimento para estas figuras [Zeltzer, 82].

Como ferramenta de modelagem utilizando a teoria de SDED em ambientes de animação nós exploramos o uso de redes de Petri[2] (PNs - *Petri Nets*) [Petri, 62][Murata, 89]. No trabalho de Magalhães et al. [Magalhães, 98] foi apresentado o uso de PNs como uma ferramenta de modelagem e de análise para ambientes de animação, ilustrando sua utilização em sistemas com concorrência, sincronização e conflito de eventos. No presente trabalho exploramos o uso de PNs na modelagem e no controle de animações.

Para o propósito da modelagem e do controle das ações em uma animação utilizando PNs, dois tipos de lugares foram definidos:

[2] Ver Apêndice A para maiores detalhes sobre redes de Petri.

- lugar de ação (*action place*): representa ações na cena;
- lugar de condição (*condition place*): representa condições de controle da PN.

A execução da ação é iniciada a partir da inserção de um *token* no lugar que representará a ação a ser executada. Um tempo de execução T_i é determinado para cada lugar P_i. O *token* torna-se apto a habilitar as transições de saída do lugar P_i pelo menos T_i unidades de tempo após a sua inserção no lugar P_i. Esta abordagem, em que os lugares temporizados representam as ações, foi escolhida devido às seguintes razões:

- preserva a notação clássica de uma PN, onde as transições representam eventos instantâneos;
- é possível representar e executar mais do que uma ação paralelamente. Se nós tivéssemos representado as ações nas transições, somente uma transição poderia disparar por vez e, consequentemente, seria executada somente uma ação por vez;
- não esconde o estado (situação) do sistema (representado pela marcação da rede) durante o tempo em que a ação está sendo executada.

Um questionamento pode surgir em relação à utilização de redes de Petri ao invés das máquinas de estado, uma estrutura muito utilizada neste nível de abstração de controle de animações por computador.

As máquinas de estado são consideradas subclasses de PNs, ou seja, qualquer modelo definido nesta estrutura pode também ser modelado utilizando uma PN, mas o contrário não é possível. Uma máquina de estado (SM - *State Machine*) pode ser equivalentemente representada como uma PN ordinária (*ordinary PN*), onde todos os seus arcos têm peso igual a 1 e cada transição t_i tem exatamente um lugar de entrada e um lugar de saída [Murata, 89], isto é,

$$|\bullet t_i| = |t_i \bullet| = 1, \text{ para toda } t_i \in T, \text{ tal que } T = \{t_{1, \ldots,} t_n\}$$

Portanto, as máquinas de estado permitem a representação de decisões ($|P_i \bullet| > 1$, $P_i \in P$), mas não a sincronização de ações paralelas ($|t_i \bullet| > 1$, $t_i \in T$). Desta maneira, como em

animações é necessário, em alguns casos, sincronizarmos a execução de determinadas ações, o uso de PNs permite que este comportamento seja modelado adequadamente.

3.4.1.1. Algoritmo para o nível de controle global

Nesta seção, apresentaremos o algoritmo para modelar e controlar uma animação utilizando PNs. Antes de apresentá-lo por meio de um fluxograma, faremos algumas explanações sobre a nossa proposta.

Cada lugar de ação tem associada uma função de animação (AF - *Animation Function*) definida pelo animador que utiliza uma técnica de animação para produzir alterações visuais na cena e um *delay* que determina o tempo de execução desta função.

Para induzir a execução de uma AF é necessária a atribuição de um *token* ao lugar. Porém, em um lugar é possível atribuir mais de um *token* e, por isso, foi necessário definir dois conjuntos de *tokens*:

- *tokens* de espera (*waiting tokens*): são aqueles que ainda não induziram a execução da AF, estando inaptos a habilitar as transições de saída;
- *tokens* de marcação (*marking tokens*): são aqueles que já induziram a execução da AF, estando aptos a habilitar as transições de saída.

O término da execução da AF ocasiona a exclusão de um *token* do conjunto de *tokens* de espera e a sua inserção no conjunto de *tokens* de marcação. Se forem atribuídos *n tokens* ao lugar, a AF será executada sucessivamente por *n* vezes, sendo que o tempo de espera para que o último *token* seja inserido no conjunto de *tokens* de marcação será *n***delay*.

Para termos conhecimento do instante do tempo em que se encontra a execução da AF associada ao lugar, foram definidos três estágios para o *token* que a induziu:

- `ARRIVAL`: a AF começou a ser executada.
- `WAIT`: a AF está sendo executada.
- `NOTHING`: não há AF sendo executada.

O estágio `ARRIVAL` serve apenas para que o animador saiba quando inicializar as variáveis da AF, pois a animação é gerada quadro a quadro e, a cada instante, a AF é

invocada para que novos valores dos atributos da cena sejam gerados. Então, na primeira invocação da AF o estágio do *token* estará `ARRIVAL`, enquanto nas demais invocações o estágio estará `WAIT`. Quando encerrar a execução da AF, o estágio passará para `NOTHING`. Em relação aos dois conjuntos de *tokens* definidos para um lugar, nos dois primeiros estágios o *token* estará contido no conjunto de *tokens* de espera e no último estágio o *token* estará contido no conjunto de *tokens* de marcação.

Em relação aos lugares de condição (*condition places*), o animador poderá optar pela utilização de um *delay*. Neste caso, a atribuição de um *token* ao lugar somente representará uma condição verdadeira após um tempo predeterminado.

Para a transição foi definida uma prioridade de disparo que indicará o quanto ela é importante em relação às demais também habilitadas no mesmo instante. Quanto maior o valor da prioridade, maior é a importância do disparo desta transição. Caso exista mais de uma transição com a mesma prioridade, a escolha de qual deverá disparar será aleatória.

A Figura 3.2, a seguir, apresenta o algoritmo necessário para controlar animações utilizando PNs.

3.4.2. O nível de controle local

O controle de uma animação, no nível local, é baseado na manipulação de parâmetros dos atores da cena ou da imagem de maneira a obter um efeito visual desejado. Nesta seção, apresentamos algumas técnicas e ferramentas para o controle das ações que podem ser aplicáveis neste nível.

3.4.2.1. Keyframe

A técnica da animação *keyframe* é derivada diretamente da animação tradicional, onde o animador desenha a cena em uma determinada situação e depois faz outro desenho, com a cena em outra situação. Estes dois desenhos (quadros-chave ou *keyframes*) são normalmente tratados por outro desenhista que então desenhará os quadros intermediários (chamados *inbetweens*). A técnica da animação *keyframe* é a transposição desta técnica para o computador, tendo sido proposta por Burtnyk e Wein em [Burtnyk, 71].

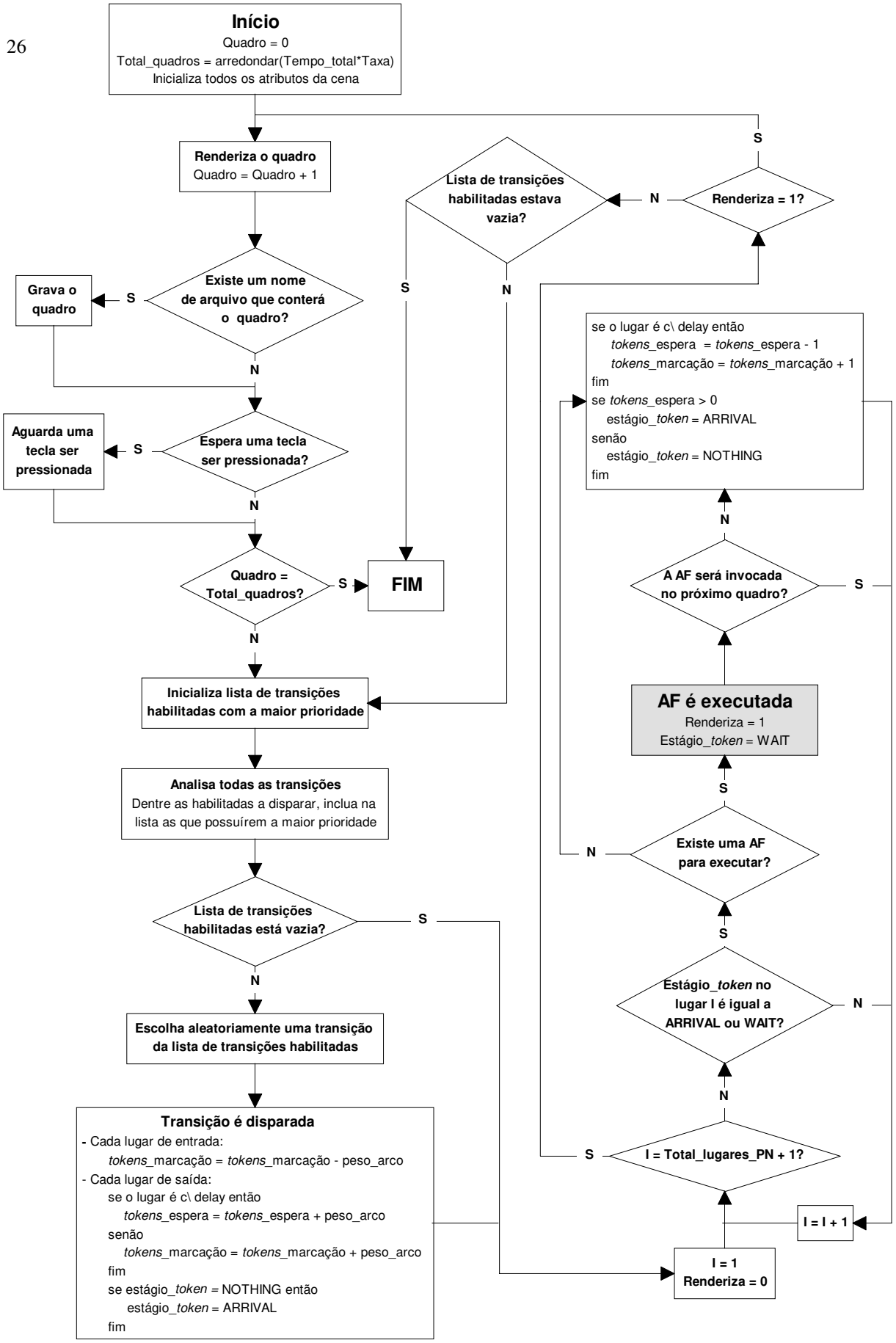


Figura 3.2 - Fluxograma do algoritmo para controlar animações utilizando PNs.

Portanto, o computador é utilizado como ferramenta de apoio na determinação dos quadros intermediários através da interpolação dos quadros-chave determinados pelo animador. A essência da técnica *keyframe* consiste em definir a correspondência entre os quadros-chave, ou seja, determinar quais pontos de um quadro-chave deverão ser mapeados em quais pontos do quadro-chave seguinte. Este problema é resolvido, por exemplo, com a utilização da interpolação paramétrica, que consiste em criar os quadros-chave da animação parametrizando a imagem [Badler, 95]. Além de resolver o mapeamento entre os pontos de quadros-chave diferentes, a interpolação paramétrica impede que haja perda de informações tridimensionais, já que elas estão embutidas nos parâmetros a serem interpolados.

3.4.2.2. Cinemática direta e inversa[1]

Ao invés de determinar os valores-chave de um parâmetro e interpolá-los, o animador pode especificar um valor inicial e uma função do tempo que descreve as modificações deste parâmetro. Por exemplo, a posição de uma bola em uma trajetória pode ser controlada através de uma equação que descreve seu comportamento ao longo do tempo. Esta técnica de controle é chamada cinemática porque os movimentos dos atores são controlados em função de posição, velocidade e aceleração.

Dependendo da complexidade estrutural de um ator, movimentos de certas partes afetarão os movimentos das demais partes que o compõem. Por exemplo, o movimento realizado pela carroceria de um carro afetará os movimentos das rodas, pois estas estão conectadas à carroceria. Entretanto, os movimentos realizados pelas rodas (por exemplo, rotações) não afetarão as demais partes do carro. Assim sendo, as relações de dependência de movimentos entre as partes podem ser definidas através de uma hierarquia (Figura 3.3).

Cada parte do ator é um nodo na hierarquia. Se o movimento realizado por um nodo afeta o movimento de outro, o primeiro é considerado o nodo pai e o segundo é considerado o nodo filho. No nosso exemplo, cada roda é um nodo filho da carroceria do carro e, consequentemente, a carroceria é considerada o nodo pai destas rodas. Os movimentos realizados pelo nodo pai serão propagados aos nodos filho. Entretanto, os movimentos realizados pelo nodo filho não são propagados ao nodo pai. Este é um exemplo da

[1] Baseado em [O'Rourke, 98].

cinemática direta (*forward kinematics*), onde os movimentos do nodo pai são propagados aos seus nodos filho.

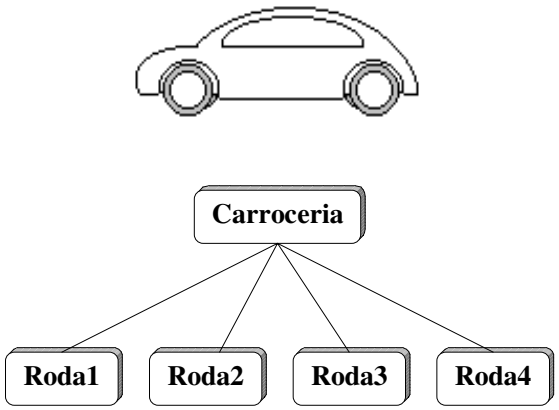


Figura 3.3 - Representação de um ator através de uma hierarquia.

Um outro exemplo é a utilização desta técnica em animações de figuras articuladas. Um braço humano poderá ser decomposto em relação às suas juntas que, no caso, serão o ombro, o cotovelo e o pulso, constituindo os nodos da hierarquia (Figura 3.4-a). Desta forma, o movimento realizado pelo ombro se refletirá também no cotovelo e no pulso. Entretanto, caso o animador queira mover o braço de forma que o pulso alcance uma determinada posição no espaço é necessário que sejam feitas várias tentativas de modo a encontrar os ângulos e as orientações adequadas de cada junta para que o objetivo seja alcançado, sendo um processo muito tedioso. Para resolver este tipo de problema pode-se pensar somente no movimento que o pulso fará, enquanto os movimentos do ombro e do cotovelo serão calculados automaticamente. Este é um exemplo da cinemática inversa (*inverse kinematics*), onde as relações de dependência dos movimentos são estipuladas no sentido inverso da cinemática direta, ou seja, do nodo filho para o nodo pai que, neste caso, serão do pulso para o cotovelo e deste para o ombro (Figura 3.4-b).

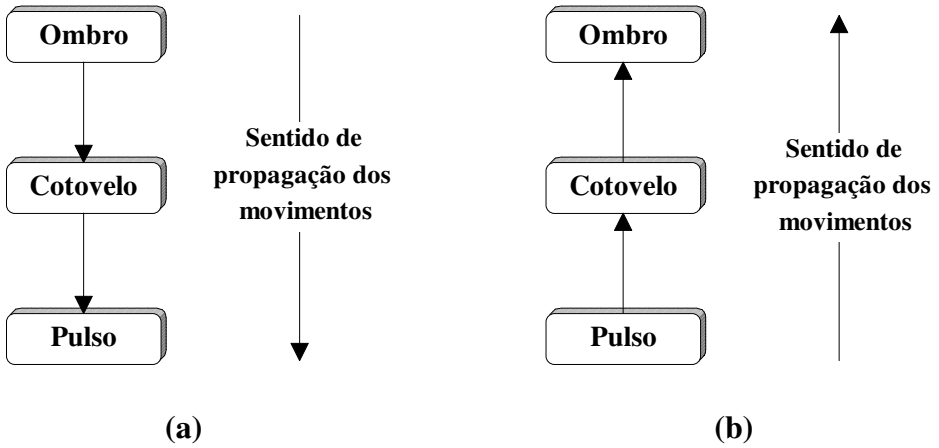


Figura 3.4 - (a) cinemática direta e (b) cinemática inversa.

Normalmente, mais de um resultado é possível para um problema de cinemática inversa, pois várias configurações das juntas podem levar a extremidade à mesma posição e orientação. Por esta razão, restrições internas são necessárias, a fim de se obter um único resultado.

3.4.2.3. Dinâmica direta e inversa[2]

Quando o objetivo de uma animação é a simulação de um processo físico, uma técnica mais sofisticada que a cinemática deve ser utilizada para que o resultado final apresente um aspecto mais realista. No processo de simulação, atores tornam-se massas com forças, torques e momentos de inércia agindo sobre eles e os movimentos são modelados utilizando leis da mecânica clássica. Por isso, o termo simulação é mais utilizado do que animação neste caso.

Há duas diferentes abordagens utilizando a dinâmica:

- A dinâmica direta, onde forças, torques e momentos de inércia são conhecidos, enquanto o movimento é calculado.

[2] Baseado em [Pina, 00]

- A dinâmica inversa, onde o movimento é conhecido, enquanto forças, torques e momentos de inércia são calculados.

Comparada com a animação cinemática, a simulação dinâmica tem a vantagem de apresentar potencialmente um maior realismo a cena. Entretanto, é necessário o conhecimento de todas as variáveis que atuam sobre os atores, o que nem sempre é simples e possível para o animador. Por essa razão, é necessário fazer vários experimentos até a obtenção do resultado desejado na animação. Além disso, o modelo dinâmico apresenta um grau de complexidade maior do que o cinemático, com maior número de variáveis a controlar.

3.4.2.4. Algoritmos genéticos

Algoritmos genéticos são métodos de otimização inspirados na seleção natural, compondo uma parte da chamada computação evolucionária [Holland, 75][Michalewicz, 96].

São baseados em uma população de indivíduos, onde cada indivíduo tem associado a si um cromossomo (genótipo específico) representando uma possível solução para o problema em questão. Os indivíduos são avaliados por uma função de adequação/adaptação (*fitness function*) que os discrimina de acordo com o seu grau de adaptação para o problema. Os indivíduos mais bem adaptados têm mais chance (processo estocástico) de serem selecionados para gerarem novos indivíduos para a próxima geração. Os operadores de reprodução (*crossover*) e mutação ficam responsáveis por gerarem novos genótipos na população.

Uma possível aplicação desta técnica no âmbito deste trabalho seria o de representar os movimentos através dos indivíduos da população, enquanto a função de adaptação avaliaria qual destes indivíduos descreveu melhor o movimento desejado [Gritz, 95][Auslander, 95].

3.4.2.5. Ferramentas orientadas a eventos

A utilização de ferramentas orientadas a eventos (por exemplo, as redes de Petri e as máquinas de estado) no nível de controle local também é possível. Por exemplo, movimentos predefinidos de figuras articuladas representando seres humanos podem ser

modelados com estas ferramentas, como "caminhar", "pular", etc. O animador, em relação ao movimento "caminhar", teria disponível parâmetros de controle como o tamanho do passo, número de passos, velocidade, etc. Para controlar a animação teríamos então uma hierarquia de controle onde as ferramentas orientadas a eventos modelariam as interações entre os atores no maior nível de controle, invocando os movimentos também modelados com estas ferramentas no menor nível de controle. Um exemplo desta abordagem será visto em maiores detalhes no Capítulo 4.

3.5. Aspectos de implementação

Nesta seção são apresentadas a definição formal de uma linguagem para auxiliar no processo de descrição da topologia de uma PN e as abstrações proporcionadas pela biblioteca de programação desenvolvida nesta atividade de mestrado para modelagem e controle de animações por computador utilizando PNs.

3.5.1. Definição formal da linguagem de descrição topológica

A definição formal da sintaxe de uma linguagem é importante tanto para o usuário quanto para o desenvolvedor. Para o usuário ela serve como referência, enquanto para o desenvolvedor ela facilita a manutenção da linguagem.

A BNF (*Backus Normal Form*) é uma notação para escrever "gramáticas" [Gear, 74] e será utilizada para descrever formalmente a sintaxe da linguagem que define a topologia de uma PN para uso na biblioteca desenvolvida neste trabalho. A BNF consiste de sentenças que definem a maneira em que a linguagem deve ser escrita. Ela é chamada de meta-linguagem e usa caracteres diferentes daqueles que são usados na linguagem por ela descrita.

Na BNF, os símbolos não-terminais (isto é, os símbolos da meta-linguagem) são delimitados por `< >` e representam estágios intermediários no processo de descrição da linguagem. O sinal `::=` significa: "é substituído por". Assim, por exemplo, a sentença: `<digit_0> ::= 0` é lida: "o meta-símbolo `<digit_0>` é substituído por `0`". O símbolo `|`

também é usado na BNF, indicando alternativa ("ou"). Assim, para representar um dígito qualquer, tem-se <*digit*> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9.

Para facilitar a descrição da linguagem de descrição topológica de uma PN foram acrescentados outros símbolos que se baseiam na meta-linguagem usada por [Pressman, 92] para a definição de um dicionário de dados. Os símbolos da meta-linguagem a serem usados e seus respectivos significados são descritos na tabela a seguir.

Símbolo	Significado
< >	delimitam meta-símbolo
::=	"é substituído por"
|	"ou"
{ }^{n}	"n repetições de"
[ ]	define opção
⇒	continuação de linha
/* */	delimitam comentários

Tabela 3.1 - Os símbolos da meta-linguagem e seus respectivos significados.

A linguagem de descrição topológica de uma PN é assim descrita (para analisar a linguagem é conveniente começar pelo meta-símbolo <*def_language*>):

<*arc_type*> ::= I | O
/* O tipo do arco é definido a partir do lugar em relação à transição, então o I indica lugar de entrada e O indica lugar de saída de uma transição */

<*character*> ::= a | b | c| ... | z | A | B | C | ... | Z
/* qualquer letra minúscula ou maiúscula */

<*def_language*> ::= <*first_line*> {<*place*>}^{n} {<*transition*>}^{n}
⇒ <*other_lines*>
<*delay*> ::= <*real*>
/* o segundo é a unidade de tempo */

```
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<first_line> ::= <places_number> <transitions_number>
```
/* número de lugares e número de transições que a PN possui */

```
<inhibitor_arc> ::= IB <place_name> <transition_name>
<integer> :: <digit> | {<digit>}^n
<in_out_arc> ::= <arc_type> <place_name> <transition_name>
                 ⇒ <weight>
<other_lines> ::= [{<inhibitor_arc>}^n] [{<in_out_arc>}^n]
<place> ::= P <place_name> <delay> <tokens>
<place_name> ::= <string>
<places_number> ::= <integer>
<priority> ::= <integer>
<real> ::= <integer> | <integer>. | .<integer> |
          ⇒ <integer>.<integer>
<string> ::= <character> | <string> <character> | <string>
             ⇒ <integer>
<transition> ::= T <transition_name> <priority>
<transition_name> ::= <string>
<transitions_number> ::= <integer>
<tokens> ::= <integer>
<weight> ::= <integer>
```

Como restrição, as definições dos componentes contidas em uma linguagem de descrição topológica devem seguir uma seqüência específica, a saber, primeiro as definições dos lugares, depois das transições e, por último, a dos arcos.

3.5.1.1. Exemplo de uma PN descrita com a linguagem definida

Utilizando a linguagem definida, a seguinte seqüência descreve a topologia da PN apresentada na Figura 3.5 (exemplo do Apêndice A):

```
4 2
P P1 0 1
P P2 0 1
P P3 0 1
P P4 0 0
T t1 1
T t2 1
I P1 t1 2
I P2 t2 1
I P3 t2 1
O P4 t1 1
O P4 t2 1
```

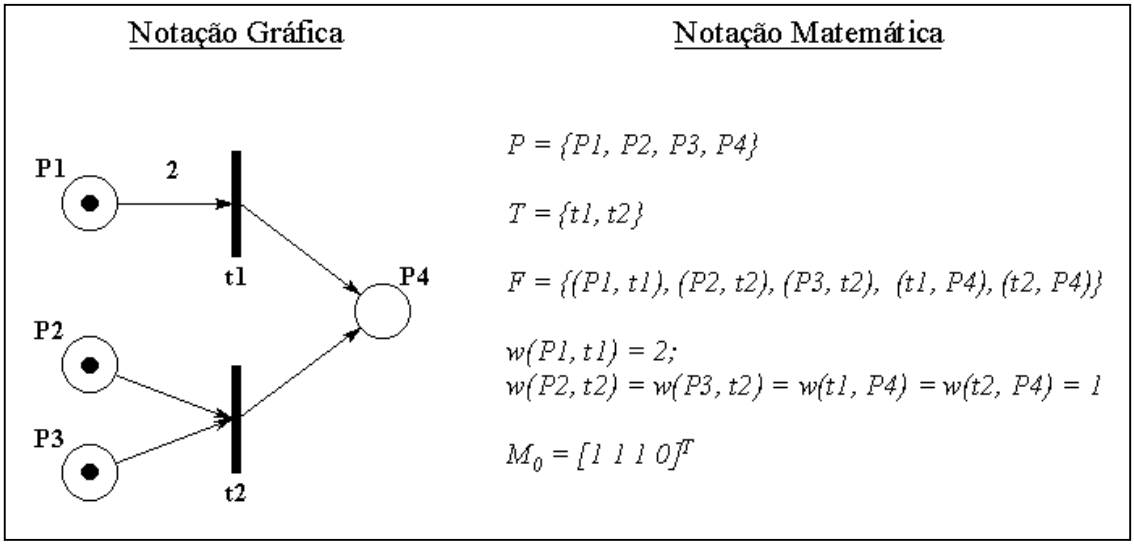


Figura 3.5 - Exemplo de uma PN.

3.5.2. Biblioteca desenvolvida

A biblioteca desenvolvida neste trabalho foi denominada TPNA (*Timed Petri Net for Animations control*). Ela foi implementada em linguagem C++ na plataforma UNIX, sendo executável em SUN (Solaris) e PC (Linux). Para visualização das animações foram usadas as funções da biblioteca Mesa [Mesa] (clone *freeware* da interface gráfica OpenGL [Neider, 96]).

A biblioteca TPNA possibilita que o animador defina uma PN que modelará e controlará a seqüência da animação desejada. A definição poderá ser realizada através de uma linguagem de descrição topológica ou diretamente através de métodos que possibilitem, por exemplo, a inserção de lugares, transições ou arcos à rede. Definida uma PN, o animador deverá disponibilizar as AFs que serão encapsuladas nos lugares de ação. Esta característica de encapsulamento e abstração flexibiliza a utilização da TPNA, tornando-a aplicável a diferentes ambientes de animação.

O restante desta seção contém uma descrição detalhada de todos os métodos da biblioteca TPNA. O animador terá duas classes a sua disposição: a `TPNA` e a `Place`.

Para a classe `TPNA` os métodos disponíveis para o animador são:

- `TPNA(int placesnumber, int transnumber, float time, int rate);`

  Método construtor que inicializa o objeto `TPNA`. É utilizado quando o animador não utiliza a linguagem de descrição topológica para descrever uma PN.

  Parâmetros:

`placesnumber`	número de lugares da PN.
`transnumber`	número de transições da PN.
`time`	tempo total da animação (em segundos).
`rate`	número de quadros por segundo.

- `TPNA(float time, int rate);`

  Método construtor que inicializa o objeto `TPNA`. É utilizado quando o animador utiliza a linguagem de descrição topológica para descrever uma PN.

  Parâmetros:

`time`	tempo total da animação (em segundos).
`rate`	número de quadros por segundo.

- `void insertplace(char *name, float delay, int tokens);`

  Método que insere um lugar na PN.

Parâmetros:

`name` nome do lugar.

`delay` tempo mínimo de espera de um *token* antes dele estar apto a habilitar as transições de saída.

`tokens` número de *tokens* do lugar. Se o *delay* > 0, os *tokens* serão incluídos no conjunto de *tokens* de espera, senão serão incluídos no conjunto de *tokens* de marcação.

- `void inserttrans(char *name, char priority);`

  Método que insere uma transição na PN.

  Parâmetros:

  `name` nome da transição.

  `priority` prioridade de disparo da transição.

- `void insertarch(char *type, char *placename, char *transname, int weight);`

  Método que insere um arco na PN.

  Parâmetros:

  `type` tipo do arco a ser inserido na PN (o tipo é definido a partir do lugar em relação à transição, podendo ser de entrada '`I`' , de saída '`O`' ou inibidor '`IB`').

  `placename` nome do lugar onde o arco será conectado.

  `transname` nome da transição onde o arco será conectado.

  `weight` peso do arco (se for inibidor este valor é desconsiderado).

- `void settime(float time);`

  Método que altera o tempo total da animação.

  Parâmetro:

  `time` tempo total da animação (em segundos).

- `float gettime(void);`

  Método que fornece o tempo total da animação.

- `void setrate(int rate);`

  Método que altera a taxa de reprodução dos quadros na animação.

  Parâmetro:

  `rate` número de quadros por segundo.

- `float getrate(void);`

  Método que fornece a taxa de reprodução dos quadros na animação.

- `void read_PN(char *filename);`

  Método que lê um arquivo contendo uma PN descrita com a linguagem topológica.

  Parâmetro:

  `filename` arquivo que contém uma PN descrita com a linguagem topológica.

- `void run(int wait, char *filename, int W, int H, void (*renderfunc)(void), int (**placesfunc)(Place place));`

  Método que gera os quadros da animação. Os quadros gerados serão imagens no formato `TIF`[3] (Tagged Image File).

  Parâmetro:

  `wait` recurso utilizado caso o animador deseje acompanhar a animação quadro a quadro. Os possíveis valores são: `1`, a função aguardará alguma tecla ser pressionada para renderizar o próximo quadro e `0`, caso contrário.

  `filename` define o prefixo do nome do arquivo que conterá um quadro da animação. Por exemplo, se o `filename` for `"quadros"`, os quadros da animação serão gerados como `quadros1.tif`, `quadros2.tif` e assim por diante. Caso ele seja nulo, os quadros não serão gravados.

[3] Código fonte disponível em [Tif].

`W` largura da janela onde será renderizada a animação.

`H` altura da janela onde será renderizada a animação.

`renderfunc` endereço da função responsável pela renderização da animação.

`placesfunc` lista de endereços das AFs que serão invocadas nos lugares. O primeiro endereço de uma função contido na lista será associado ao primeiro lugar inserido na rede, o segundo endereço será associado ao segundo lugar inserido e assim por diante. O total de endereços na lista corresponde ao total de lugares existentes na rede. Se existirem lugares que não invocarão funções, o animador deverá colocar um valor nulo (`NULL`) nas respectivas posições na lista.

Pode-se perceber que uma AF (*Animation Function*) possui uma estrutura predefinida: deverá retornar um valor do tipo `int` e deverá possuir apenas um parâmetro do tipo `Place`, possibilitando o acesso aos atributos do lugar que a invocou. Assim, é possível saber, por exemplo, quando é necessário inicializar as variáveis desta função e qual é o seu tempo de execução. O retorno da AF (`int`) é um *flag* que indicará o término da sua execução. Dessa forma, a função `run` percebe o momento em que deve alterar o estágio do *token* de `WAIT` para `NOTHING`.

Para a classe `Place` os métodos disponíveis para o animador são:

- `char* getname(void);`

  Método que fornece o nome do lugar.

- `int getstage(void);`

  Método que fornece o estágio do *token* que induziu a execução da AF associada ao lugar. Os possíveis valores são `NOTHING` (`0`), `ARRIVAL` (`1`) e `WAIT` (`2`).

- `float getdelay(void);`

  Método que fornece o tempo mínimo de espera de um *token* antes dele estar apto a habilitar as transições de saída. Em relação à AF este *delay* corresponde ao seu tempo de execução.

- `int marking(void);`

  Método que fornece ao animador o número de *tokens* contido no conjunto de *tokens* de marcação.

- `int getwaiting(void);`

  Método que fornece o número de *tokens* contido no conjunto de *tokens* de espera.

O trecho de programa a seguir apresenta a definição de uma PN que modela e controla os movimentos de uma figura bípede caminhando (Figura 4.4, Seção 4), utilizando a biblioteca de programação TPNA.

```
const W = 400;
const H = 400;
const RATE = 30;
const ANIMATION_TIME = 10;

void mount_run_net(void) {
int (*placesfunc[])(Place) =
{NULL,
 halfforwardlarm,halfbackwardrarm,halfstancelleg,halfswingrleg,
 NULL,
 backwardlarm,forwardrarm,swinglleg,stancerleg,
 NULL,
 forwardlarm,backwardrarm,stancelleg,swingrleg,
 halfforwardlarm,halfbackwardrarm,halfstancelleg,halfswingrleg,
 halfbackwardlarm,halfforwardrarm,halfswinglleg,halfstancerleg,
 NULL};

// endereços das AFs - funções que executam determinados
// movimentos.

TPNA net(ANIMATION_TIME,RATE);
   net.read_PN("walk.pn");
   net.run(0,"walk",W,H,display,placesfunc);
   // display é o endereço da função de renderização.
}
```

3.6. Considerações finais

Embora seja necessário ter uma técnica ou ferramenta no controle local, uma vantagem da abordagem proposta é encapsulá-la através dos lugares de ação no nível de controle global. Isto permite que o animador escolha a melhor metodologia para realizar uma determinada ação, sem que isso influencie no restante da animação.

Para descrever uma seqüência da animação no nível de controle global é primordial que o animador identifique os eventos que a regerão. Algumas perguntas podem ser colocadas como "Quando um evento ocorrer, como um ator deverá responder?" ou "Quais atores serão afetados por estes eventos?". Em síntese, é necessário identificar as diretivas comportamentais da animação. Uma vez que estas foram identificadas, é necessário construir as ações que serão executadas no nível de controle local. Então, estas ações serão organizadas no nível de controle global considerando os eventos que causarão as transições entre elas. Esta abordagem provê uma habilidade em particionar a modelagem e o controle da seqüência em problemas menores, disponibilizando facilidades como o encapsulamento e a reutilização das ações.

A utilização da linguagem de descrição topológica auxilia o processo de modelagem, possibilitando a reutilização de PNs em seqüências de animação com comportamentos similares e, consequentemente, simplificando o trabalho do animador. Para animadores que se adaptam melhor em interfaces gráficas para compor animações, a biblioteca de programação TPNA disponibiliza métodos que poderão ser utilizados junto a uma biblioteca de programação específica para a criação de tais interfaces, de modo a produzir aplicativos que apresentem uma modelagem mais compreensiva e menos técnica (do ponto de vista da necessidade do prévio conhecimento sobre a linguagem de descrição topológica).

No próximo capítulo apresentaremos exemplos que ilustram a modelagem e o controle de movimentos de figuras articuladas bípedes utilizando PNs em diferentes níveis de abstração.

CAPÍTULO 4

ESTUDO DE CASOS

Este capítulo apresenta exemplos utilizando PNs em diferentes níveis de abstração para modelar e controlar os movimentos e as interações de figuras bípedes que se assemelham a seres humanos. São apresentados movimentos muito estudados no campo da animação de figuras bípedes, como o "caminhar" e o "pular". É estudado também a modelagem e o controle necessário para animar duas figuras bípedes jogando bola [Bicho, 01].

4.1. Introdução

Uma figura bípede pode ser modelada através de uma estrutura articulada. De fato, são necessárias duas estruturas articuladas, idênticas no caso, para compor a figura bípede. Analogamente, para um modelo que se assemelha a um ser humano devemos considerar mais duas estruturas articuladas para os seus braços [Camargo, 95a]. Em nossa abordagem, ambos membros superiores e inferiores das figuras bípedes possuem o mesmo número de graus de liberdade. Em todos os exemplos, o animador define as PNs que irão modelar e controlar as interações entre os atores[1] da cena (controle global) e define as AFs (*Animation Functions*) invocadas no menor nível de abstração de controle (controle local) que realizarão o movimento de fato.

Para o controle local destas estruturas nós adotamos uma biblioteca de cinemática inversa denominada IKAN[2] (*Inverse Kinematics using ANalytical Methods*) [Tolani, 00]. Com esta técnica é necessário especificar as posições e as orientações da parte extrema de cada membro (*end effector*), sendo o sistema responsável pelo cálculo dos ângulos e das

[1] Dependendo do nível de abstração trabalhado, o ator pode ser uma figura bípede inteira ou apenas um membro desta.
[2] Disponível em [IKAN].

orientações das junções intermediárias de maneira a posicionar o membro adequadamente conforme a trajetória do *end effector*, produzindo o movimento.

Se tivéssemos utilizado a técnica da cinemática direta, o movimento do *end effector* seria calculado indiretamente como o resultado dos movimentos das junções intermediárias. Nós também não usamos a técnica da dinâmica direta porque seria necessário obter todas as variáveis físicas, tais como forças e torques, que produziriam os movimentos adequadamente. Por exemplo, não é simples obter as leis físicas que predizem como a perna se moverá na animação de uma figura bípede caminhando. Uma alternativa para este problema seria a utilização da técnica da dinâmica inversa para analisar o movimento de maneira a identificar as forças e torques requeridos, mas este método também não é fácil de trabalhar. Consequentemente, nós decidimos utilizar a técnica da cinemática inversa para este nível de controle, possibilitando o uso da biblioteca de programação IKAN.

Nosso objetivo não foi produzir movimentos realistas, mas demonstrar que o uso de PNs é uma ferramenta eficaz para a modelagem e o controle de animações em um maior nível de abstração. Dessa maneira, para simplificar, no controle local nós trabalhamos com equações de reta para produzir trajetórias para o movimento de um membro como, por exemplo, "membro para frente", "membro para trás" e assim por diante (Figura 4.1).

A biblioteca de programação IKAN disponibiliza uma função que analisa se o objetivo definido para o *end effector* se encontra no espaço de alcance do membro. Caso esteja fora deste espaço de alcançe, a função escala o objetivo em relação ao tamanho do membro de maneira a coincidir com o *end effector*. Então, quando trabalhamos com equações de reta, a trajetória extrema realizada pelo *end effector* será o perímetro de uma circunferência cujo raio será igual ao tamanho do membro. Outro detalhe a acrescentar é que o membro percorrerá a trajetória definida em velocidade constante, independentemente do movimento estipulado.

No Apêndice B há uma descrição dos principais métodos disponibilizados pela biblioteca IKAN, enquanto no Apêndice C estão disponíveis as PNs definidas através da linguagem de descrição topológica, utilizadas nos exemplos deste capítulo.

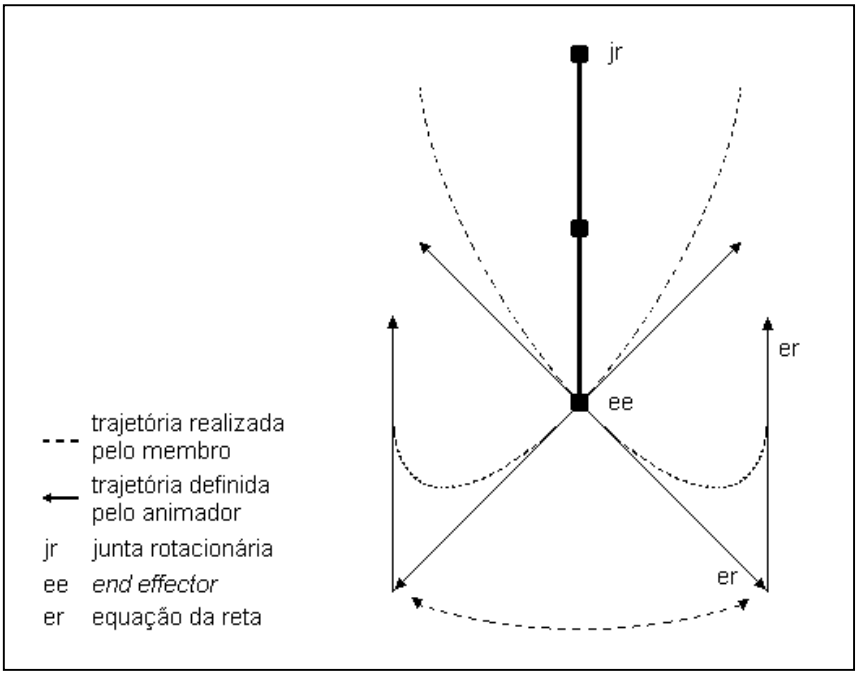


Figura 4.1 - Exemplo de trajetória utilizando equações da reta para mover um membro.

4.2. Modelo a ser animado

Antes de apresentarmos os exemplos, é necessário primeiramente definirmos o modelo corporal no qual desejamos aplicar o método de controle a fim de obtermos o movimento desejado. Segundo Nedel [Nedel, 98], os modelos corporais tridimensionais podem ser classificados em quatro categorias:

- modelos de esqueleto (*stick figures models*): consistem de um conjunto hierárquico de segmentos rígidos conectados por junções;
- modelos de superfície (*surface models*): consistem de um esqueleto envolvido por superfícies planares ou curvas;
- modelos de volume (*volume models*): aproximam a estrutura e a forma do corpo a partir de um conjunto de primitivas volumétricas, como cilindros, elipses ou esferas;

- modelos multi-camadas (*multi-layered models*): são estruturas mais elaboradas, projetadas para representar figuras humanas. Nestes modelos, normalmente um esqueleto é usado como apoio das camadas que simulam o volume do corpo, da camada representando a pele e, em alguns casos, da camada representando a roupa. O animador então especifica as várias relações restritivas entre as camadas, de modo a controlar o movimento global do ator.

Em seu trabalho, Nedel abordou a modelagem do corpo humano, mas estes modelos também podem ser utilizados em outras estruturas corporais.

Nós focaremos o trabalho em animações de modelos de esqueleto. A complexidade destes modelos é proporcional ao número de segmentos e junções envolvidos na estrutura. A sua principal vantagem em relação aos demais modelos é a facilidade para especificar o movimento, uma vez que é necessário apenas controlar os ângulos e as orientações referentes às junções. Em contrapartida, este tipo de representação produz animações pouco realistas. A falta de volume dificulta a percepção de profundidade e pode ocasionar uma ambigüidade caso exista mais de um ator na cena. O ponto relevante aqui é que qualquer modelo corporal pode ser animado a partir do movimento da sua estrutura base, descrita através de um modelo de esqueleto. Mas, certamente, a escolha do modelo corporal é determinante na efetividade da percepção dos movimentos, como estudado em [Hodgins, 98].

As animações apresentadas nos exemplos a seguir foram geradas a uma taxa de 24 quadros/segundo.

4.3. O caminhar de uma figura bípede

No campo da animação por computador, o caminhar humano tem sido estudado de forma exaustiva e, pelo menos conceitualmente, é bem compreendido. O caminhar é uma seqüência cíclica onde as pernas se movimentam para frente e para trás, provendo suporte alternado ao corpo. Esta seqüência é obtida impondo-se algumas diretivas comportamentais, de maneira que o movimento pareça real:

- ambas as pernas não podem estar no ar ao mesmo tempo;
- a mesma perna não pode ser levantada mais de uma vez consecutivamente.

Além disso, é possível também sincronizar os movimentos dos braços com os movimentos das pernas. Então, os movimentos do braço esquerdo podem ser sincronizados com os movimentos da perna direita e os movimentos do braço direito podem ser sincronizados com os movimentos da perna esquerda, caracterizando o clássico comportamento do caminhar.

Quando consideramos cada um dos membros individualmente, estamos diante do controle local da figura bípede. Ou seja, devemos solucionar o problema de posicionamento de cada um dos membros ao longo do tempo de forma isolada. A solução proposta, já mencionada anteriormente, foi utilizar a biblioteca de cinemática inversa IKAN. Mas não basta controlarmos o movimento de cada perna isoladamente, visto que, em um bípede, deve haver coordenação entre ambas para que a translação da figura seja efetivamente realizada. Dessa forma, quando estamos lidando com a interação entre os membros, estamos diante de um exemplo de problema que pode ser tratado sob a óptica de controle global do movimento.

Os possíveis movimentos de cada perna serão aqui definidos usando uma abordagem parcialmente baseada no modelo apresentado por Girard e Maciejewski [Girard, 85]. Alguns parâmetros importantes descritos neste trabalho auxiliam a descrição do caminhar de uma figura bípede, como veremos a seguir.

4.3.1. Modelo de locomoção cinemático

O modelo de locomoção discutido em [Girard, 85] define alguns parâmetros para descrever o deslocamento de um ator dotado de pernas.

Um padrão de locomoção descreve uma seqüência de elevações e descidas do pé. O padrão se repete à medida que a figura se move: cada repetição de uma seqüência é chamada de ciclo de locomoção. O tempo (ou número de quadros) necessário para se completar um único ciclo equivale ao período *P* do ciclo de locomoção. Além disso, o tempo em que um pé permanece no solo é chamado de *duração de suporte*, enquanto o

tempo em que uma perna permanece no ar é chamado de *duração de transferência*. Portanto, nós temos:

$$P = DuraçãoSuporte + DuraçãoTransferência \quad (1)$$

Durante o período do ciclo, uma perna permanecerá uma porcentagem do tempo no chão. Este fator é chamado *fator de suporte* da perna e é dado por:

$$FatorSuporte = \frac{DuraçãoSuporte}{P} \quad (2)$$

O caminhar requer que o *fator de suporte* de cada perna seja, no mínimo 0,5, já que por definição ambos os pés devem estar no chão simultaneamente por uma percentagem do período do ciclo. Um *fator de suporte* menor que 0,5 resultaria em um bípede correndo, pois todo o corpo sairia do chão por alguns instantes. Por último, nós podemos definir o *deslocamento* como a distância percorrida pela figura durante a fase de suporte do pé. Para o nosso exemplo, os valores para o *deslocamento* e para a *duração de suporte* serão, respectivamente, denotados por *x e n.*

Notemos que, de fato, não existe nenhum controle interagindo com as duas pernas simultaneamente. Dessa forma, existem apenas regras cinemáticas que não garantem a coordenação entre elas. A seguir, estruturaremos um controle do sistema para coordenar os movimentos dos membros através de um modelo a eventos discretos.

4.3.2. Modelo de locomoção a eventos discretos

Quando recorremos a um modelo de sistema dinâmico a eventos discretos para o exemplo em questão desejamos que, efetivamente, exista um mecanismo de controle interagindo entre os membros da figura. As diretivas comportamentais e os parâmetros especificados anteriormente são úteis para construirmos um mecanismo de controle utilizando uma PN. Então, a Figura 4.2 apresenta a PN definida para este propósito, enquanto a Tabela 4.1 apresenta os movimentos que devem ser realizados em cada lugar de ação e a Tabela 4.2 apresenta as condições de controle da rede.

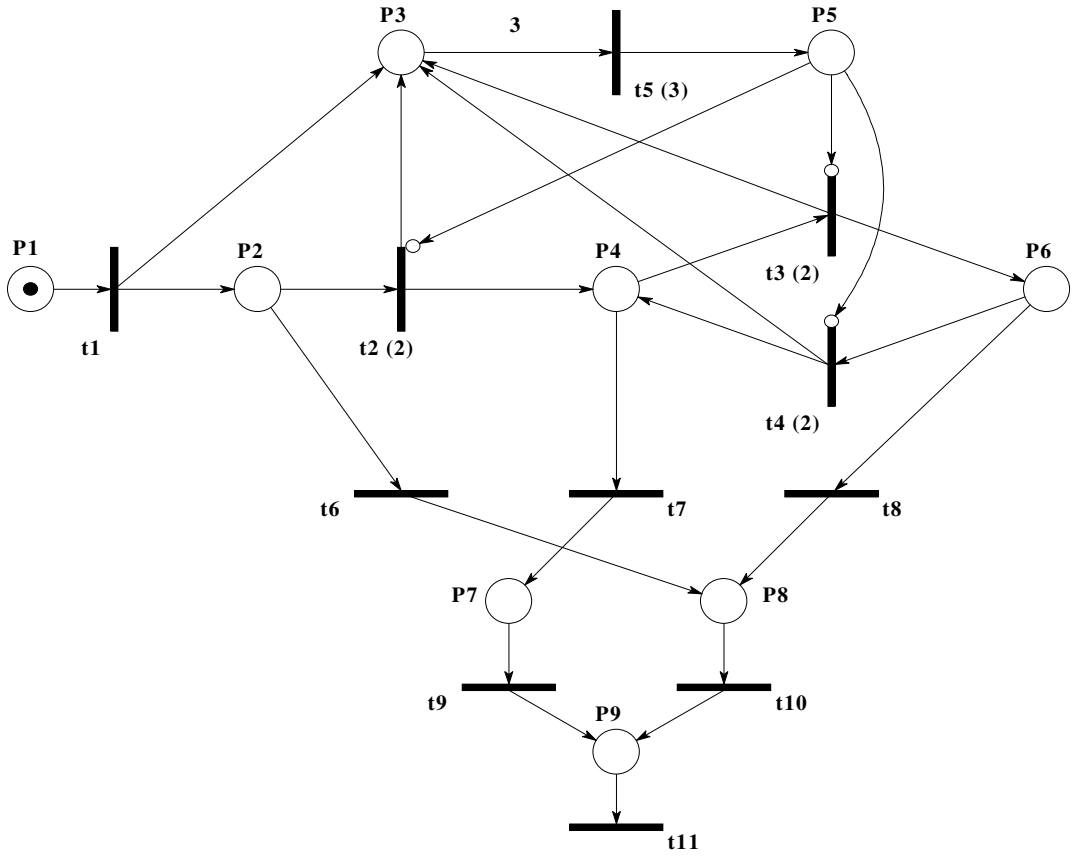


Figura 4.2 - Modelo de PN: o caminhar de uma figura bípede.

Lugares de ação	Movimentos	Valores dos parâmetros
P2, P7	perna esquerda apoiada, deslocando o corpo; perna direita para frente; braço esquerdo para frente; braço direito para trás	tempo = n/2 deslocamento = x/2
P8	perna esquerda para frente; perna direita apoiada, deslocando o corpo; braço esquerdo para trás; braço direito para frente	
P4	perna esquerda para frente; perna direita apoiada, deslocando o corpo; braço esquerdo para trás; braço direito para frente	tempo = n deslocamento = x
P6	perna esquerda apoiada, deslocando o corpo; perna direita para frente; braço esquerdo para frente; braço direito para trás	

Tabela 4.1 - Lugares que executam os movimentos no caminhar de uma figura bípede.

Lugar de condição	Função
P1	início do caminhar
P3	contador de passos
P5	contador de passos igual ao número de passos definido
P9	término do caminhar

Tabela 4.2 - Lugares que estipulam as condições de controle no caminhar de uma figura bípede.

A seqüência da animação de uma figura bípede caminhando em uma reta comporta-se da seguinte maneira. A figura está inicialmente na posição de descanso, tal que os braços e as pernas estão em paralelo ao comprimento do corpo. O número de passos que a figura deverá caminhar é definido pelo peso do arco compreendido entre o lugar P3 e a transição t5, que no nosso exemplo é igual a 3. Como a figura deverá retornar à posição de descanso ao final da caminhada, o primeiro e o último passo terão a metade do deslocamento e do tempo de um passo normal. Cabe salientar que estamos considerando na contagem total dos passos somente aqueles que executam o movimento de colocar uma perna à frente do corpo. Se tivéssemos associado um passo ao deslocamento do corpo, então neste exemplo a figura daria 4 passos e não 3. Nós consideramos que o último passo é apenas o movimento necessário para retornar a figura à posição de descanso. Assim sendo, o início da execução de um passo em um lugar de ação adiciona um *token* em P3 (exceto o último passo), enquanto o seu final habilita duas transições de saída do lugar em questão. Enquanto não houver *tokens* em P5, a transição com maior prioridade dentre as habilitadas disparará, indicando que o caminhar ainda não terminou. De outra forma, o arco inibidor desabilitará esta transição e consequentemente somente a transição para retornar a figura à posição de descanso estará habilitada a disparar.

Na PN da Figura 4.2 (resultado na Figura 4.3), a transição t1 dispara, adicionando um *token* em P2 e em P3. O *token* em P2 induz a execução do primeiro passo (quadros 1 a 12), habilitando as transições t2 e t6. A transição t2 dispara, adicionando um *token* em P3 e em P4. O *token* em P4 induz a execução do segundo passo (quadros 12 a 36), habilitando as

transições t3 e t7. A transição t3 dispara, adicionando um *token* em P3 e em P6. Neste instante, P3 tem três *tokens* e consequentemente a transição t5 dispara, adicionando um *token* em P5. O *token* em P6 induz a execução do terceiro passo (quadros 36 a 60), habilitando apenas a transição t8, pois o *token* em P5 desabilita a transição t4. A transição t8 dispara, adicionando um *token* em P8. O *token* em P8 induz a execução do último passo, responsável apenas por retornar a figura à posição de descanso (quadros 60 a 72), habilitando a transição t10. A transição t10 dispara, adicionando um *token* em P9, indicando o término do caminhar. Perceba que, neste exemplo, o lugar P7 não é executado, pois foi definido um número ímpar de passos. O lugar P7 é executado caso seja definido um número par de passos, deixando de ser executado o lugar P8.

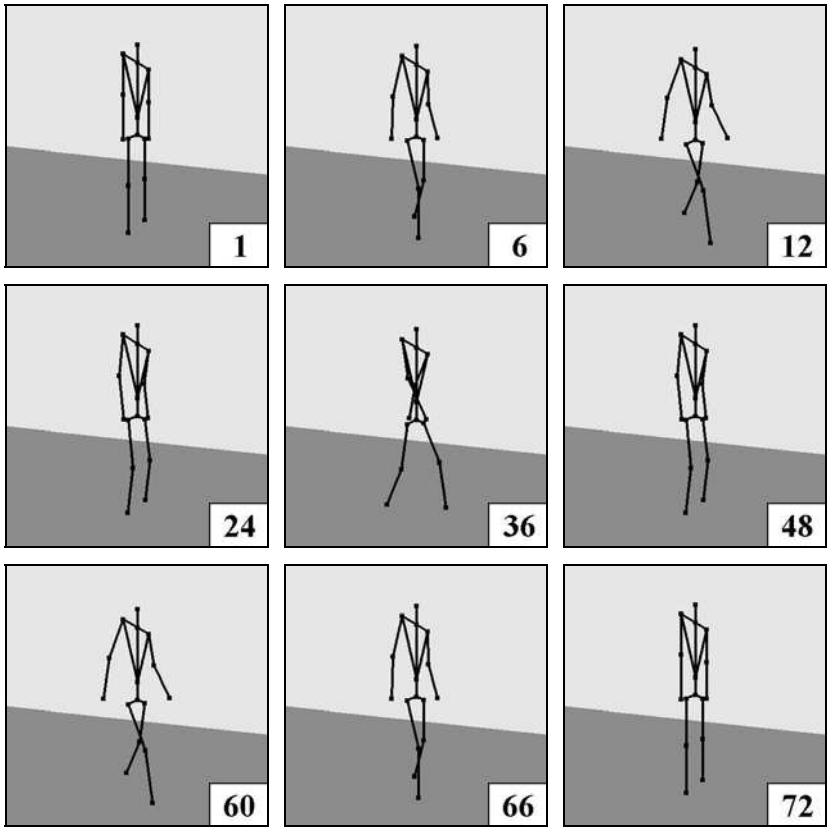


Figura 4.3 - O caminhar de uma figura bípede.

Na Figura 4.2, cada lugar de ação é responsável por executar os movimentos de todos os membros. Na Figura 4.4 apresentamos uma outra PN onde cada lugar de ação executa o

movimento de um único membro. Então, ao invés de um há quatro lugares executando paralelamente para cada passo. A Tabela 4.3 apresenta os lugares que executam os movimentos dos braços e a Tabela 4.4 apresenta os lugares que executam os movimentos das pernas para esta modelagem. Este particionamento facilita a alteração do movimento de um certo membro e favorece a reutilização desta PN em outras situações, removendo, quando necessário, determinados lugares de ação (consequentemente, os movimentos de certos membros). Na Seção 4.5 apresentamos um exemplo de reutilização de PNs.

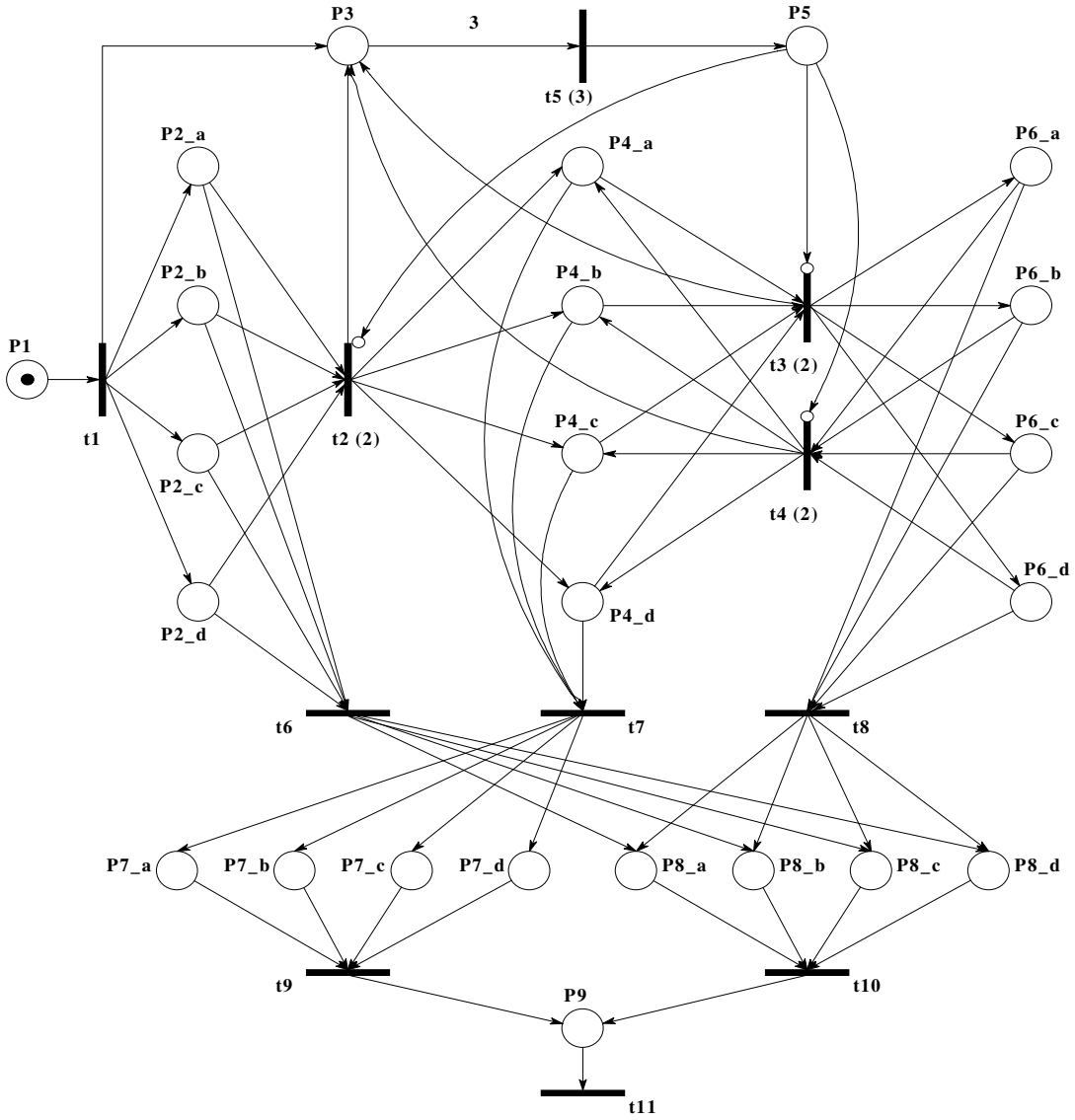


Figura 4.4 - Modelo de PN: o caminhar de uma figura bípede em um menor nível de abstração.

Lugares de ação		Movimento	Valores dos parâmetros
Braço esquerdo	**Braço direito**		
P2_a, P7_a	**P8_b**	para frente	tempo = n/2 deslocamento = x/2
P8_a	**P2_b, P7_b**	para trás	
P6_a	**P4_b**	para frente	tempo = n deslocamento = x
P4_a	**P6_b**	para trás	

Tabela 4.3 - Lugares que executam os movimentos dos braços no caminhar de uma figura bípede.

Lugares de ação		Movimento	Valores dos parâmetros
Perna esquerda	**Perna direita**		
P8_c	**P2_d, P7_d**	para frente	tempo = n/2 deslocamento = x/2
P2_c, P7_c	**P8_d**	apoiada, deslocando o corpo	
P4_c	**P6_d**	para frente	tempo = n deslocamento = x
P6_c	**P4_d**	apoiada, deslocando o corpo	

Tabela 4.4 - Lugares que executam os movimentos das pernas no caminhar de uma figura bípede.

4.4. O pular de uma figura bípede

O pular de uma figura bípede é um movimento que não apresenta um comportamento predefinido. Um ser humano pode realizá-lo de diferentes maneiras, dificultando a definição de um modelo que descreva coerentemente a seqüência deste movimento.

Assim sendo, uma alternativa encontrada foi analisar os princípios básicos do pular que nortearam a definição de um modelo. No trabalho de Witkin e Popovi'c [Witkin, 99], foi apresentado um método para reduzir a complexidade de seqüências de animação de figuras utilizando uma formulação baseada em equações dinâmicas de restrições de espaço-tempo, preservando as propriedades físicas essenciais do movimento. Neste trabalho, foram modelados o correr e o pular de uma figura bípede, sendo que este último em quatro diferentes modos: o pulo girando (*twist jump*), o pulo diagonal (*diagonal jump*), o pulo de um obstáculo (*obstacle jump*) e o pulo desequilibrado (*unbalanced jump*).

Analisando as animações dos quatro modos de pular descritos em [Witkin, 99], percebemos que estes possuíam a seguinte seqüência de movimentos:

- agachar;
- impulsionar;
- saltar;
- pousar;
- agachar;
- levantar.

Uma vez definida esta seqüência, era necessário definir qual, dentre os quatro possíveis pulos, seria utilizado como uma aproximação do pular que pretendíamos modelar. Nós então descrevemos uma seqüência cujos movimentos dos braços, pernas e torso fossem similares à seqüência do pular de um obstáculo, sem o obstáculo a superar, mas mantendo as demais características do movimento. A Figura 4.5 apresenta a PN que modela este movimento e a Tabela 4.5 apresenta os movimentos que devem ser realizados em cada lugar de ação. Para o movimento "erguer ambos os braços", executado pelo lugar de ação P2_c da Figura 4.5, foi utilizada uma PN que modela a associação dos movimentos "membro para frente" e "membro para frente e para cima" (Figura 4.6). A intenção foi apresentar um exemplo de reutilização no nível de controle local, pois já havíamos definido o movimento "membro para frente" (ver Seção 4.3), bastando definir o movimento "membro para frente e para cima" (Tabela 4.6).

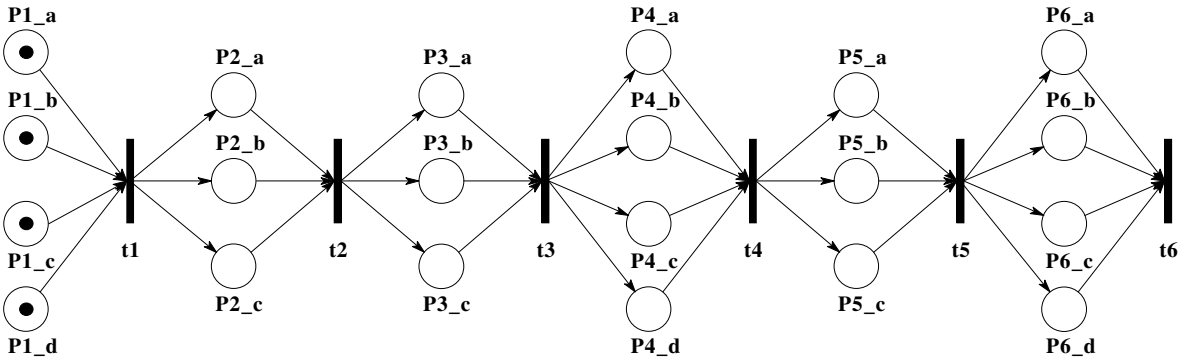


Figura 4.5 - Modelo de PN: o pular de uma figura bípede.

Movimento		**Lugar**
Pernas	flexionar	**P1_a, P5_a**
	estender	**P2_a, P6_a**
Inclinar o torso	para frente	**P1_b, P4_b, P5_b**
	para trás	**P2_b, P3_b, P6_b**
Deslocar o corpo	para cima	**P3_a**
	para baixo	**P4_a**
Braço esquerdo	para frente	**P4_c**
	para trás	**P1_c, P3_c, P6_c**
	para baixo	**P5_c**
Braço direito	para trás	**P1_d, P6_d**
	para baixo	**P4_d**
Erguer ambos os braços		**P2_c**

Tabela 4.5 - Lugares que executam os movimentos no pular de uma figura bípede.

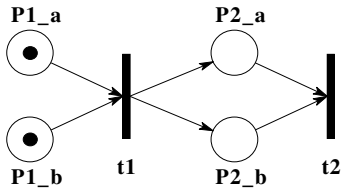


Figura 4.6 - Modelo de PN: o erguer os braços no pular de uma figura bípede.

Movimento		**Lugar**
Braço esquerdo	para frente	**P1_a**
	para frente e para cima	**P2_a**
Braço direito	para frente	**P1_b**
	para frente e para cima	**P2_b**

Tabela 4.6 - Lugares que executam os movimentos para erguer os braços no pular de uma figura bípede.

A seqüência da animação do pular de uma figura bípede, descrita pela PN da Figura 4.5 (resultado na Figura 4.7), comporta-se da seguinte maneira. A figura está inicialmente na posição de descanso, tal que os braços e as pernas estão em paralelo ao comprimento do corpo. O primeiro movimento a ser executado será o de se agachar para preparar a impulsão (quadros 1 a 36), modelado nos lugares P1_a a P1_d. A seguir, a figura inicia o movimento de impulsionar (quadros 36 a 60), modelado nos lugares P2_a a P2_c. Com a impulsão, a figura desloca-se para cima, executando o movimento de saltar (quadros 60 a 84), modelado nos lugares P3_a a P3_c. Após uma certa altura, a figura prepara-se para pousar (quadros 84 a 108), modelado nos lugares P4_a a P4_d. Quando a figura chega ao chão, as pernas flexionam-se para absorver o impacto (quadros 108 a 126), modelado nos lugares P5_a a P5_c. Por fim, a figura retorna à posição inicial (quadro 126 a 180), modelado nos lugares P6_a a P6_d.

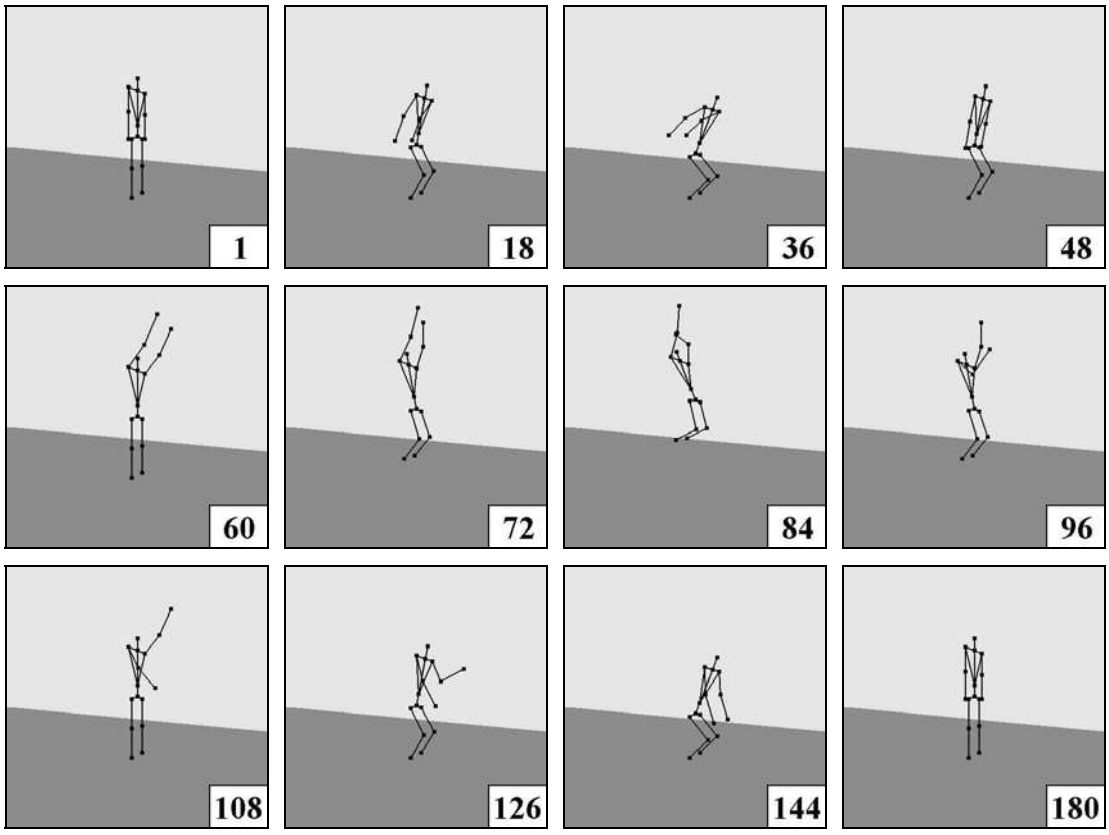


Figura 4.7 - O pular de uma figura bípede.

Este movimento poderia ter sido modelado como uma máquina de estados, onde um único lugar representaria cada movimento do corpo (agachar, saltar, etc.). Entretanto, a modelagem onde há lugares executando paralelamente os movimentos de determinados membros facilita a reutilização e a produção de variações do pular.

4.5. As figuras bípedes jogando bola

Este exemplo apresenta o controle dos movimentos de figuras bípedes em um maior nível de abstração. Assim sendo, a intenção é demonstrar que nossa abordagem utilizando PNs é adequada para diferentes níveis de abstração.

A animação consiste de duas figuras bípedes caminhando em linha reta e jogando uma bola. Para que isso ocorra, de tempos em tempos os atores param e aquele que está com a bola a arremessa para o outro ator; a caminhada continua, sempre alternando a posse da bola. Neste exemplo, é importante observar que os atores não necessariamente caminham a mesma distância ou o mesmo número de passos antes de pararem. Então, as interações entre os atores podem ser definidas através de diretivas comportamentais que são utilizadas para controlar a animação:

- o ator com a bola, depois de ter caminhado, somente continuará a caminhada após arremessar a bola para o outro ator;
- o ator sem a bola, depois de ter caminhado, somente continuará a caminhar após receber a bola do outro ator;
- o ator com a bola somente a arremessará se o outro ator estiver pronto para recebê-la.

Nós modelamos a PN para o controle global de acordo com estas diretivas comportamentais (Figura 4.8). A Tabela 4.7 apresenta os movimentos dos atores que devem ser executados nos lugares de ação da rede. Os lugares de ação P10 e P12 modelam a trajetória da bola que, neste exemplo, será uma reta entre os atores. Os lugares de condição P9 e P11 indicam que o ator está pronto para receber a bola. Para este modelo, estes lugares são temporizados (eles possuem o mesmo *delay* que os lugares de ação P14 e

P6, respectivamente). Dessa maneira, o ator terá tempo para posicionar o corpo antes de receber a bola.

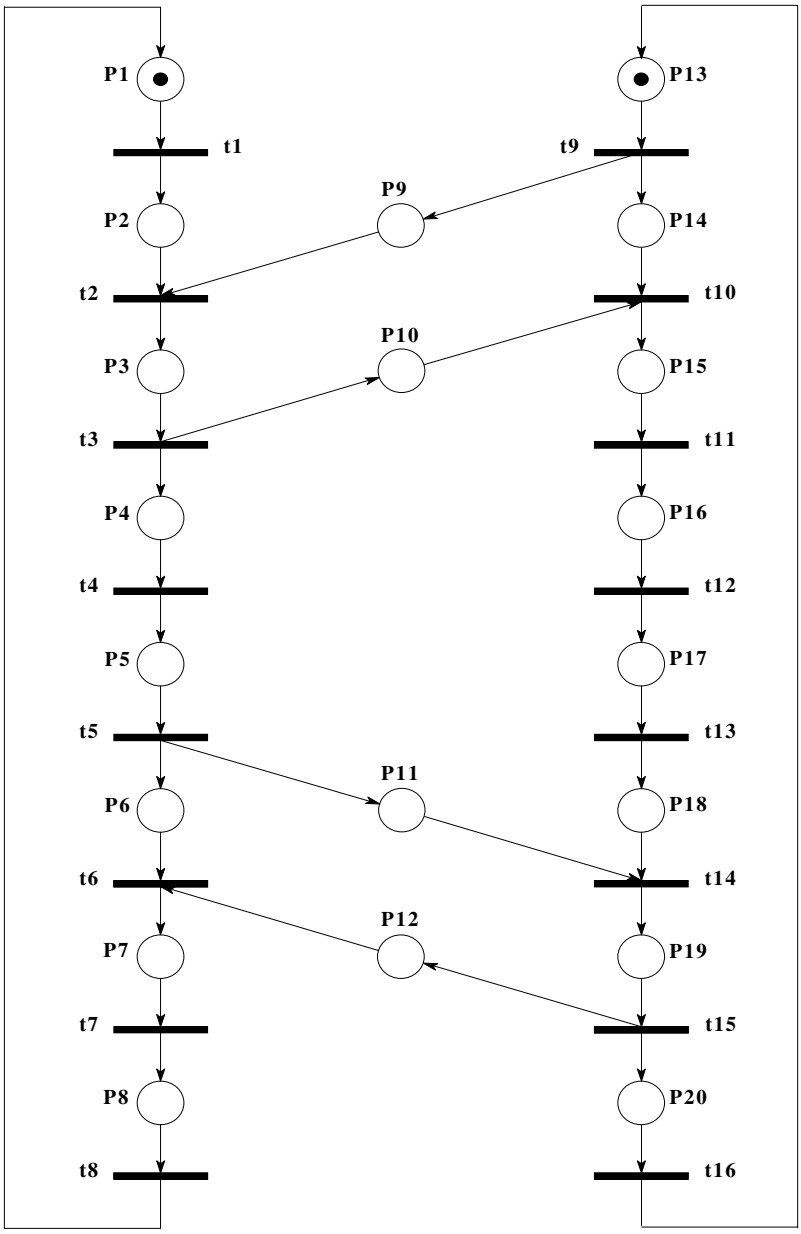


Figura 4.8 - Modelo de PN: as figuras bípedes jogando bola.

Ator A	Ator B	Movimento
P1	**P17**	Caminhar com a bola
P2	**P18**	Posicionar para arremessar a bola
P3	**P19**	Arremessar a bola
P4	**P20**	Posicionar para caminhar sem a bola
P5	**P13**	Caminhar sem a bola
P6	**P14**	Posicionar para receber a bola
P7	**P15**	Receber a bola
P8	**P16**	Posicionar para caminhar com a bola

Tabela 4.7 - Lugares que executam os movimentos das figuras bípedes jogando bola.

Neste exemplo, há um conjunto de movimentos predefinidos para os atores, apresentados na Tabela 4.7, que fazem parte da seqüência da animação. Estes movimentos também foram modelados utilizando-se PNs, ilustrando a utilização desta ferramenta para modelos hierárquicos. A seguir apresentaremos os modelos definidos para cada um destes movimentos, exceto o modelo do caminhar, pois este já foi exposto (ver Seção 4.3). Por simplicidade e sem perda de generalidade na modelagem da rede, nós descrevemos os movimentos em relação ao ator B. Para que os modelos sejam aplicados ao ator A, será preciso apenas mudar o sentido de rotação do movimento "girar o corpo" (se o sentido for horário no ator B, no ator A será anti-horário e vice-versa) e a perna definida para um certo movimento (se for a perna esquerda no ator B, no ator A será a perna direita e vice-versa). Ao final, será apresentada a seqüência da animação das figuras bípedes jogando bola.

♦ O posicionar de uma figura bípede para arremessar a bola

As diretivas comportamentais que descrevem este movimento são:

- Movimentar a perna direita para trás, preparando-se para girar;
- Girar o corpo no sentido horário, reposicionando a perna direita.

A Figura 4.9 apresenta a PN que modela este movimento, enquanto a Tabela 4.8 apresenta os movimentos que devem ser realizados em cada lugar de ação e a Figura 4.10 apresenta a animação resultante. Esta figura apresenta a PN encapsulada nos lugares P2 e P18 da Figura 4.8.

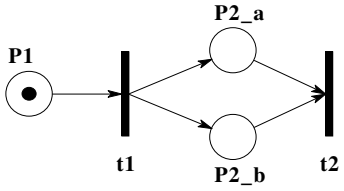


Figura 4.9 - Modelo de PN: o posicionar de uma figura bípede para arremessar a bola.

Lugar	Movimento
P1	perna direita para trás
P2_a	girar o corpo no sentido horário
P2_b	perna direita para frente

Tabela 4.8 - Lugares que executam os movimentos do posicionar de uma figura bípede para arremessar a bola.

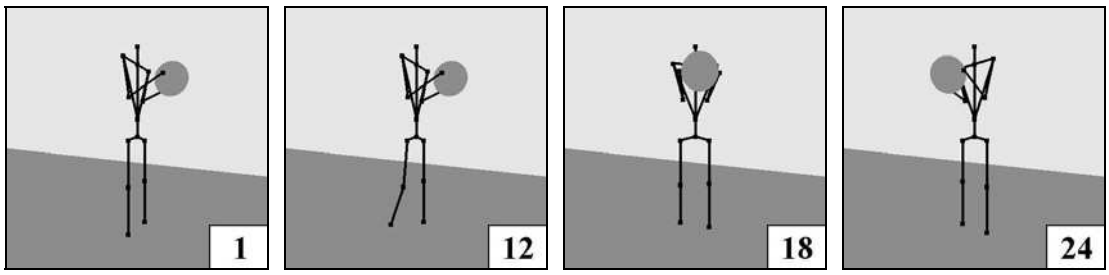


Figura 4.10 - O posicionar de uma figura bípede para arremessar a bola.

♦ O movimento de uma figura bípede para arremessar a bola

Como é um movimento de impulsionar a bola para frente, as diretivas comportamentais que descrevem este movimento são:

- Estender os braços e curvar o corpo para frente, flexionando as pernas.

A Figura 4.11 apresenta a PN que modela este movimento, enquanto a Tabela 4.9 apresenta os movimentos que devem ser realizados em cada lugar de ação e a Figura 4.12 apresenta a animação resultante. Esta figura apresenta a PN encapsulada nos lugares P3 e P19 da Figura 4.8.

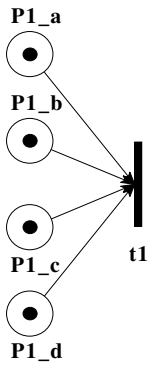


Figura 4.11 - Modelo de PN: o movimento de uma figura bípede para arremessar a bola.

Lugar	Movimento
P1_a	flexionar as pernas
P1_b	torso para frente
P1_c	braço esquerdo para frente
P1_d	braço direito para frente

Tabela 4.9 - Lugares que executam o movimento de uma figura bípede para arremessar a bola.

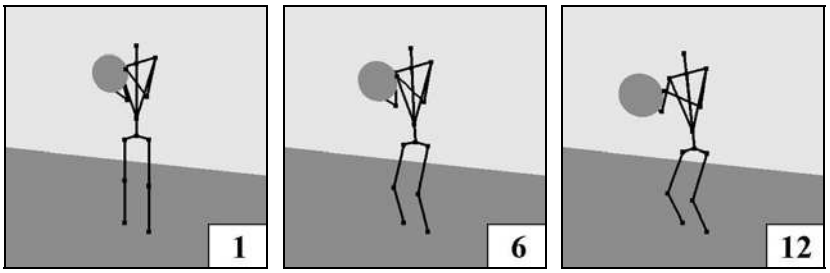


Figura 4.12 - O movimento de uma figura bípede para arremessar a bola.

♦ O posicionar de uma figura bípede para caminhar sem a bola

As seguintes diretivas comportamentais descrevem este movimento:

- Estender as pernas e curvar o corpo para trás, baixando os braços;
- Movimentar a perna esquerda para trás, preparando-se para girar;
- Girar o corpo no sentido anti-horário, reposicionando a perna esquerda.

A Figura 4.13 apresenta a PN que modela este movimento, enquanto a Tabela 4.10 apresenta os movimentos que devem ser realizados em cada lugar de ação e a Figura 4.14 apresenta a animação resultante. Esta figura apresenta a PN encapsulada nos lugares P4 e P20 da Figura 4.8.

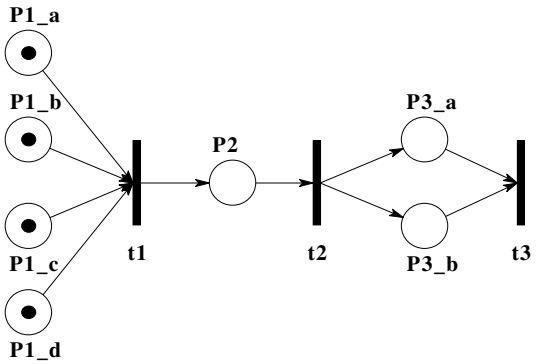


Figura 4.13 - Modelo de PN: o posicionar de uma figura bípede para caminhar sem a bola.

Lugar	Movimento
P1_a	estender as pernas
P1_b	torso para trás
P1_c	braço esquerdo para baixo
P1_d	braço direito para baixo
P2	perna esquerda para trás
P3_a	girar o corpo no sentido anti-horário
P3_b	perna esquerda para frente

Tabela 4.10 - Lugares que executam os movimentos do posicionar de uma figura bípede para caminhar sem a bola.

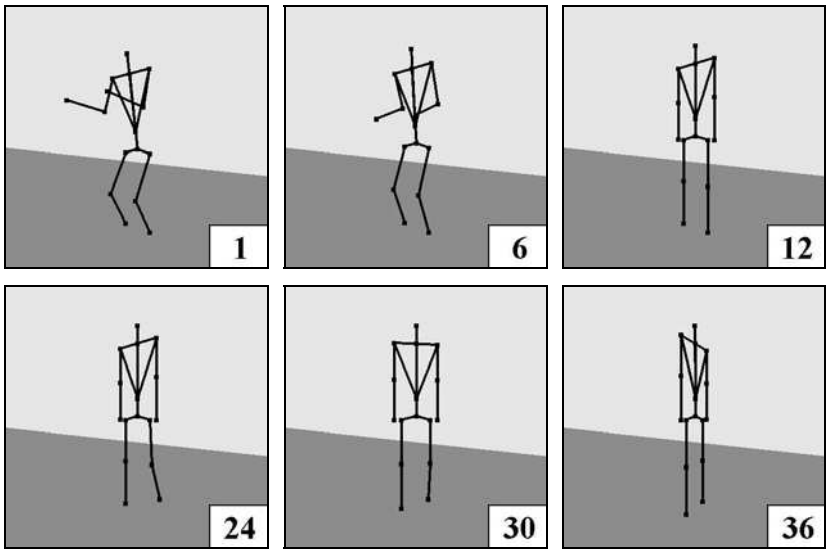


Figura 4.14 - O posicionar de uma figura bípede para caminhar sem a bola.

- **O posicionar de uma figura bípede para receber a bola**

As diretivas comportamentais que descrevem este movimento são:

- Movimentar a perna direita para trás, preparando-se para girar;
- Girar o corpo no sentido horário, reposicionando a perna direita;

- Estender os braços e curvar o corpo para frente, flexionando as pernas.

A Figura 4.15 apresenta a PN que modela este movimento, enquanto a Tabela 4.11 apresenta os movimentos que devem ser realizados em cada lugar de ação e a Figura 4.16 apresenta a animação resultante. Esta figura apresenta a PN encapsulada nos lugares P6 e P14 da Figura 4.8.

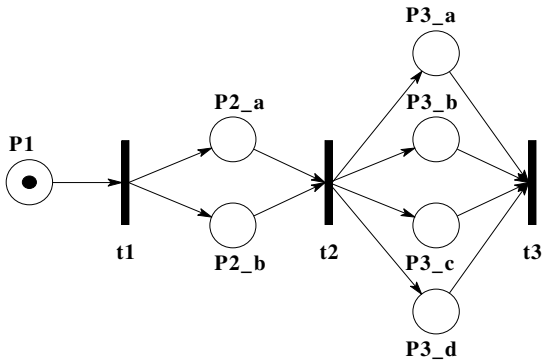


Figura 4.15 - Modelo de PN: o posicionar de uma figura bípede para receber a bola.

Lugar	Movimento
P1	perna direita para trás
P2_a	girar o corpo no sentido horário
P2_b	perna direita para frente
P3_a	flexionar as pernas
P3_b	torso para frente
P3_c	braço esquerdo para frente
P3_d	braço direito para frente

Tabela 4.11 - Lugares que executam os movimentos do posicionar de uma figura bípede para receber a bola.

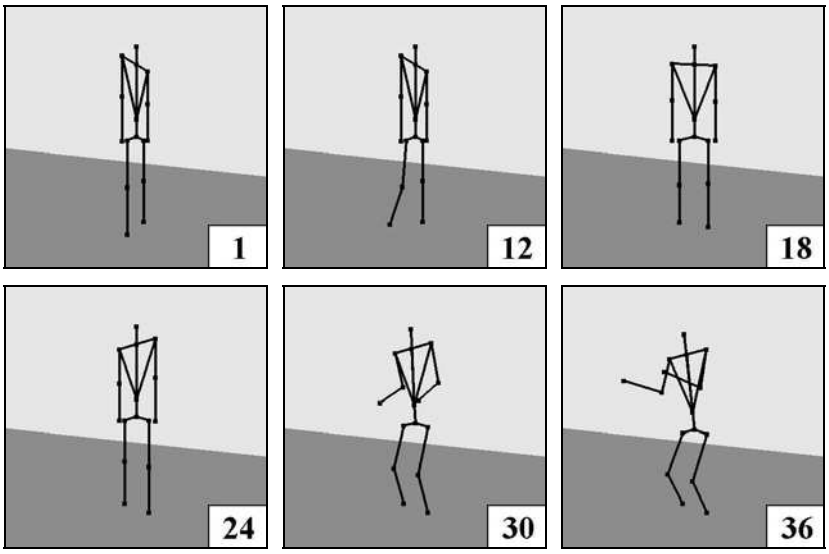


Figura 4.16 - O posicionar de uma figura bípede para receber a bola.

♦ O movimento de uma figura bípede para receber a bola

Como é um movimento de receber a bola, as seguintes diretivas comportamentais descrevem este movimento:

- Flexionar os braços para receber a bola;
- Estender as pernas e curvar o corpo para trás.

A Figura 4.17 apresenta a PN que modela este movimento, enquanto a Tabela 4.12 apresenta os movimentos que devem ser realizados em cada lugar de ação e a Figura 4.18 apresenta a animação resultante. Esta figura apresenta a PN encapsulada nos lugares P7 e P15 da Figura 4.8.

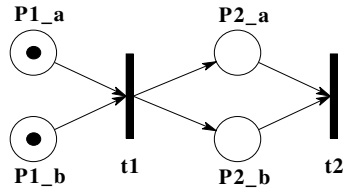


Figura 4.17 - Modelo de PN: o movimento de uma figura bípede para receber a bola.

Lugar	Movimento
P1_a	braço esquerdo para trás
P1_b	braço direito para trás
P2_a	estender as pernas
P2_b	torso para trás

Tabela 4.12 - Lugares que executam o movimento de uma figura bípede para receber a bola.

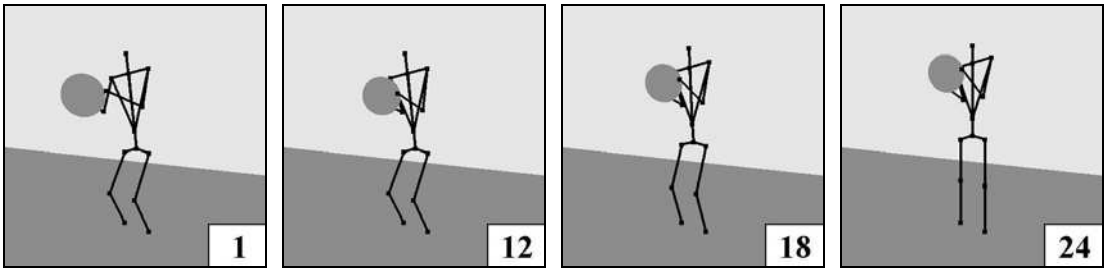


Figura 4.18 - O movimento de uma figura bípede para receber a bola.

♦ O posicionar de uma figura bípede para caminhar com a bola

As seguintes diretivas comportamentais descrevem este movimento:

- Movimentar a perna esquerda para trás, preparando-se para girar;
- Girar o corpo no sentido anti-horário, reposicionando a perna esquerda.

A Figura 4.19 apresenta a PN que modela este movimento, enquanto a Tabela 4.13 apresenta os movimentos que devem ser realizados em cada lugar de ação e a Figura 4.20 apresenta a animação resultante. Esta figura apresenta a PN encapsulada nos lugares P8 e P16 da Figura 4.8.

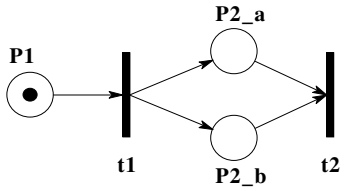


Figura 4.19 - Modelo de PN: o posicionar de uma figura bípede para caminhar com a bola.

Lugar	Movimento
P1	perna esquerda para trás
P2_a	girar o corpo no sentido anti-horário
P2_b	perna esquerda para frente

Tabela 4.13 - Lugares que executam os movimentos do posicionar de uma figura bípede para caminhar com a bola.

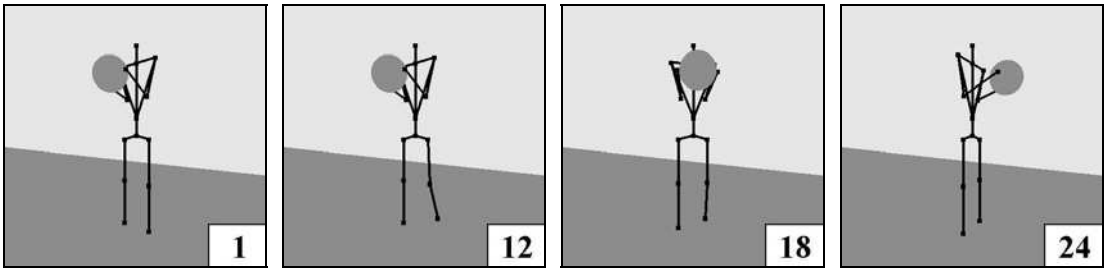


Figura 4.20 - O posicionar de uma figura bípede para caminhar com a bola.

No exemplo das figuras bípedes jogando bola os atores poderiam caminhar indefinidamente (ver Figura 4.8), mas isto não ocorre devido ao tempo total da animação definido pelo animador . Em relação ao caminhar do ator, o tamanho do passo de ambos é o mesmo, sendo que aquele que estiver com a bola dará 2 passos, enquanto o outro dará 3 passos. A Figura 4.21 apresenta a animação resultante deste exemplo, onde o ator A está no lado esquerdo e o ator B está no lado direito de cada quadro.

A seqüência da animação comporta-se da seguinte maneira (ver Figura 4.8). Ambos atores estão caminhando (quadros 1 a 48, lugares P1 e P13) até o momento em que o ator A pára, posicionando-se para arremessar a bola para o ator B (quadros 48 a 72, lugar P2). O ator B logo após também pára, posicionando-se para receber a bola (quadros 72 a 108, lugar P14). O ator A arremessa a bola para o ator B (quadros 108 a 120, lugar P3), pois este está preparado para recebê-la (lugar P9). O ator B recebe a bola, enquanto o ator A posiciona-se e começa a caminhar (quadro 120 a 168, lugares P15, P4 e P5). O ator B então se posiciona para começar a caminhar (quadros 168 a 192, lugar P16). Enquanto o ator B está

caminhando, o ator A pára (quadros 192 a 228, lugar P17), posicionando-se para receber a bola (quadros 228 a 264, lugar P6). Neste mesmo instante, o ator B também pára, posicionando-se para arremessar a bola (lugar P18). O ator B arremessa a bola para o ator A (quadros 264 a 276, lugar P19), pois este está preparado para recebê-la (lugar P11). O ator A recebe a bola, enquanto o ator B posiciona-se e começa a caminhar (quadro 276 a 324, lugares P7, P20 e P13). O ator A então se posiciona para começar a caminhar (quadros 324 a 348, lugar P8).

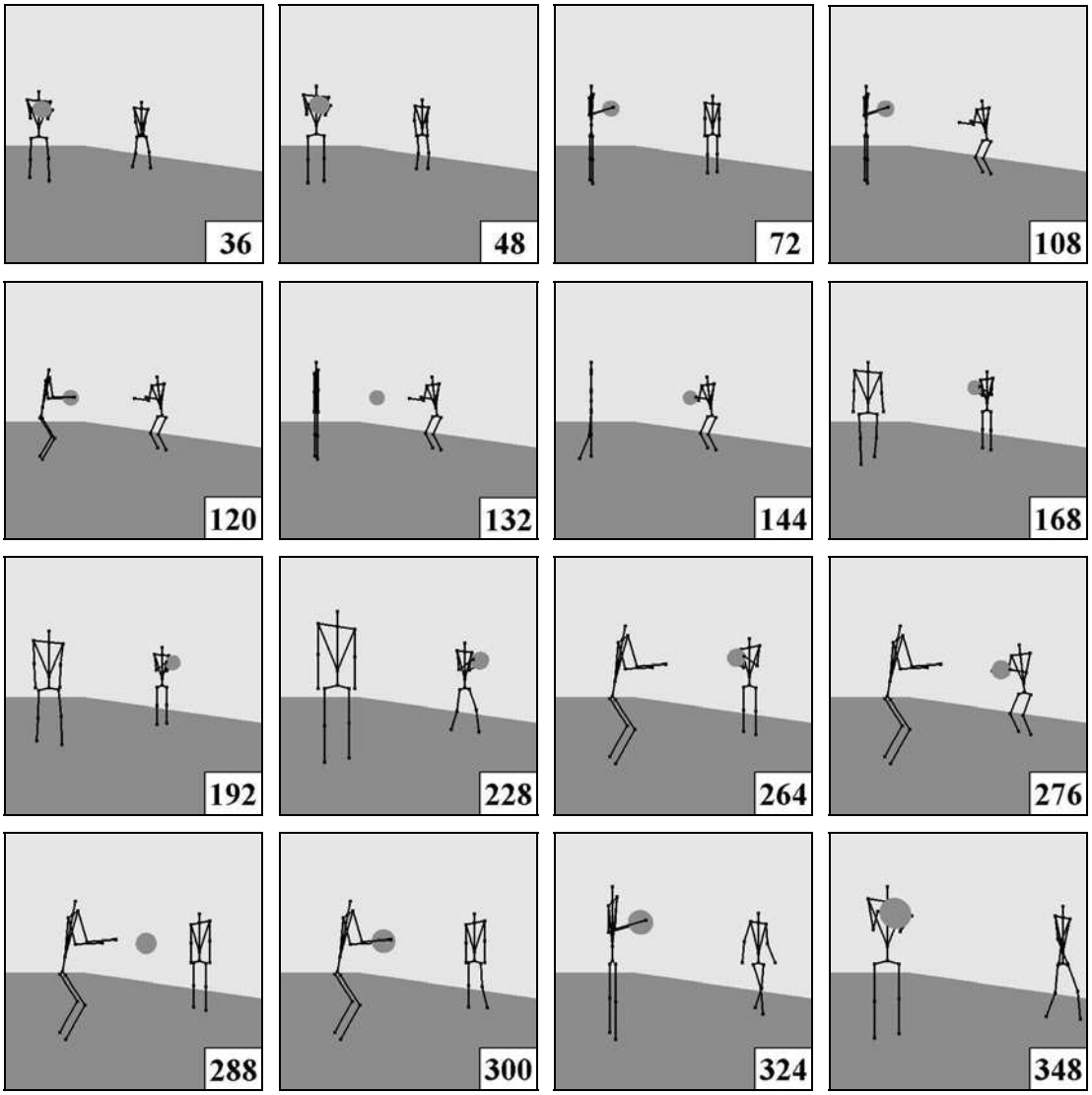


Figura 4.21 - As figuras bípedes jogando bola.

Este exemplo demonstra não somente a reutilização de modelos básicos de PNs (o movimento do caminhar, por exemplo, é modelado pela PN apresentada na Seção 4.3), mas também a capacidade de encapsulamento oferecida por uma PN. O grafo da Figura 4.8 oculta os detalhes dos movimentos, tal como o caminhar ou o posicionar, reforçando um modelo de descrição hierárquico.

4.6. Considerações finais

Neste capítulo apresentamos exemplos utilizando PNs em diferentes níveis de abstração para modelar e controlar os movimentos e as interações de figuras bípedes que se assemelham a seres humanos.

Nós utilizamos o modelo de esqueleto para a representação estrutural das figuras bípedes devido à maior facilidade para especificar os movimentos, uma vez que é necessário controlar apenas os ângulos e as orientações referentes às junções. Com a estrutura definida, nós descrevemos os modelos comportamentais dos exemplos através de PNs no nível de controle global. No nível de controle local nós definimos as AFs invocadas nos lugares de ação das PNs utilizando, para os exemplos em questão, a biblioteca de cinemática inversa IKAN.

A modelagem utilizando PNs demonstrou que o particionamento propiciado por esta ferramenta facilita a alteração de movimentos sem comprometer o restante da seqüência da animação. Além disso, a notação de uma PN favorece a reutilização, o encapsulamento e, consequentemente, a descrição utilizando hierarquias de modelagem e de controle.

No próximo capítulo apresentaremos as conclusões obtidas com este trabalho e as sugestões de trabalhos futuros.

CAPÍTULO 5

CONCLUSÃO

O objetivo deste trabalho foi propor uma abordagem de controle de animações utilizando o paradigma de blocos de controle [Camargo, 95a], explorando o uso de redes de Petri no bloco de controle global. Para isso, foi necessário compreendermos alguns conceitos básicos da teoria de Sistemas Dinâmicos a Eventos Discretos, de modo a viabilizar o uso de redes de Petri como mecanismo de controle em animações por computador.

O Capítulo 2 apresentou as principais classificações dos sistemas de animação segundo seus mecanismos de controle, situando a abordagem proposta no presente contexto. Além disso, alguns trabalhos que propuseram métodos de controle orientados a eventos em animação por computador também foram apresentados.

O Capítulo 3 apresentou a proposta de controle de animações utilizando o paradigma de blocos de controle, explorando o uso de redes de Petri no bloco de controle global. Alguns conceitos básicos pertencentes à teoria de Sistemas Dinâmicos a Eventos Discretos foram apresentados, de forma a tratar animações sob a óptica desta teoria. Também foram apresentadas técnicas de controle em animação por computador aplicáveis no bloco de controle local. Ao final, o capítulo apresentou o algoritmo e o conjunto de funções da biblioteca de programação denominada TPNA (*Timed Petri Net for Animations control*), desenvolvida para modelar e controlar animações por computador utilizando PNs, e uma linguagem para descrever a topologia de uma PN.

O Capítulo 4, aplicando a proposta apresentada no Capítulo 3, ilustrou a utilização de PNs em diferentes níveis de abstração para modelar e controlar os movimentos e as

interações de figuras bípedes que se assemelham a seres humanos. Foram apresentados exemplos como o "caminhar" e o "pular" de uma figura bípede e figuras bípedes jogando bola.

5.1. Contribuições

Para organizar as ações em uma seqüência de animação utilizando PNs no nível de controle global é primordial que o animador identifique os eventos que a regerão, ou seja, é necessário identificar as diretivas comportamentais da animação. Após a identificação destas diretivas, é necessário construir as ações, através de técnicas de controle em animação, que serão executadas no nível de controle local. Então, estas ações serão organizadas no nível de controle global considerando os eventos que causarão as transições entre elas. Esta abordagem provê uma habilidade em particionar a modelagem e o controle da seqüência da animação em problemas menores, disponibilizando facilidades como o encapsulamento e a reutilização das ações.

A utilização de PNs como mecanismo de controle é vantajoso porque permite que a técnica de controle empregada no nível de controle local seja encapsulada em um lugar de ação no nível de controle global. Este encapsulamento possibilita escolher a melhor técnica para realizar cada ação na cena, reutilizar AFs e/ou PNs em animações cujos comportamentos sejam similares e realizar alterações sem a necessidade de remodelar toda ou grande parte da animação. Além disso, as PNs possibilitam definir hierarquias de modelagem e de controle com vários níveis de abstração, desde um modelo em baixo nível (como o controle das interações entre os membros no exemplo do caminhar de uma figura bípede) a um modelo em alto nível (como o controle das interações entre figuras bípedes no exemplo do jogo de bola).

Em relação à implementação, a biblioteca de programação TPNA mostrou ser adequada para o propósito em questão. Para utilizar esta biblioteca, conforme o paradigma de blocos de controle, há a necessidade de serem definidos dois níveis de abstração de controle: o de maior nível de abstração (global) contendo uma PN e o de menor nível (local) contendo as AFs executadas nos lugares de ação. Uma vez definidos estes níveis, os

quadros da animação serão gerados automaticamente. O término da animação é indicado pelo número total de quadros definido pelo animador ou quando não existirem mais transições habilitadas a disparar e nem AFs a serem executadas, sendo utilizado aquele que primeiro ocorrer. A linguagem de descrição topológica, por sua vez, auxilia o processo de modelagem, possibilitando a reutilização de PNs em seqüências cujos comportamentos sejam similares.

Como pontos fracos, a modelagem utilizando PNs pode apresentar dificuldades na definição das diretivas comportamentais, dependendo da animação. Além disso, a modelagem da PN pode também apresentar uma quantidade expressiva de lugares e transições, tornando inviável a utilização da linguagem de descrição topológica.

5.2. Trabalhos futuros

Como uma possível atividade futura, sugerimos a criação de uma interface gráfica para modelar e controlar as animações utilizando PNs, de maneira que o animador não necessite ter um prévio conhecimento sobre a linguagem de descrição topológica. Os métodos disponibilizados pela biblioteca de programação TPNA poderiam ser utilizados junto a uma biblioteca de programação específica para a criação desta interface, de modo a produzir aplicativos que facilitassem a descrição de PNs. Outra possibilidade é utilizar aplicativos já disponíveis que produzissem uma saída (uma PN) que fosse traduzida para a linguagem de descrição topológica, de modo a utilizá-la pela biblioteca TPNA. A utilização/criação de uma interface gráfica também é uma boa solução para modelos hierárquicos que apresentem vários níveis de abstração, pois simplifica o trabalho do animador para a definição destes modelos.

Por também ser uma ferramenta de análise, uma PN possibilita antecipar o comportamento da animação, visto que há várias técnicas e ferramentas disponíveis para tal finalidade. Assim, esta interface também poderia incorporar ferramentas de análise para o modelo. Cabe salientar que a análise do comportamento considera apenas o desencadeamento dos eventos no nível de controle global e não o comportamento das técnicas de controle empregadas no nível de controle local. Há três possíveis tipos de

análise: verificação, validação e desempenho [van der Aalst, 98]. A análise de verificação, baseada em propriedades das PNs (*reachability*, *liveness*, *reversibilidade*, etc.) [Murata, 89], pode ser realizada para descobrir se a rede apresenta *deadlocks*, se atinge algum estado não permitido, se há transições mortas, etc. A análise de validação pode ser feita para garantir que a rede esteja corretamente definida e corresponda com exatidão ao sistema modelado (os testes são feitos por meio de simulação iterativa de situações para verificar se a rede funciona como o esperado). A análise de desempenho pode ser realizada para avaliar a capacidade do sistema em atingir certos requisitos, tais como tempo médio de espera, número médio de casos pendentes, etc.

Uma outra possibilidade muito interessante seria prover um mecanismo que, a partir das diretivas comportamentais definidas pelo animador, gerasse automaticamente a PN que modelaria e controlaria as interações dos atores na cena, permitindo assim ao animador trabalhar em um nível de abstração mais próximo da sua realidade.

REFERÊNCIAS BIBLIOGRÁFICAS

[Auslander, 95] J. Auslander et al. *Further experience with controller-based automatic motion synthesis for articulated figures*. ACM Transactions on Graphics, 14(4): 311-336. 1995.

[Badler, 95] N. I. Badler. *Computer Animation Techniques*. In: Introduction to Computer Graphics, M. Bailey et al., Course Notes for SIGGRAPH'95. 1995.

[Badler, 99] N. I. Badler, M. S. Palmer and R. Bindiganavale. *Animation Control for Real-Time Virtual Humans*. Communications of the ACM, 42(8): 64-73. August, 1999.

[Bause, 96] F. Bause and P. S. Kritzinger. *Stochastic Petri Nets: An Introduction to the Theory*. Advanced Studies in Computer Science, Verlag Vieweg. 1996.

[Becket, 94] W. M. Becket. *The Jack Lisp API*. Technical Report MS-CIS-94-01, University of Pennsylvania. 1994.

[Bicho, 01] A de L. Bicho, A. B. Raposo e L. P. Magalhães, *Control of Articulated Figures Animations Using Petri Nets*. Proc. of the SIBGRAPI'2001 - XIV Brazilian Symposium on Computer Graphics and Image Processing. Florianópolis, SC, Brazil, October, 2001. SBC - Sociedade Brasileira de Computação, IEEE Press.

[Bruderlin, 94] A. Bruderlin, C. G. Teo and T. Calvert. *Procedural movement for articulated figure animation*. Computer & Graphics, 18(4): 453-461. 1994.

[Burtnyk, 71] N. Burtnyk and M. Wein. *Computer-generated key-frame animation*. J. of SMPTE, 80: 149-153. 1971.

[Camargo, 94] J. T. F. Camargo, L. P. Magalhães and A. B. Raposo. *Modeling motion simulation with DEDS*. Proc. of 13$^{th.}$ IFIP Congress - International Federation of Information Processing, pp. 162-167. 1994.

[Camargo, 95a] J. T. F. Camargo. *Animação modelada por computador - técnicas de controle de movimento em animação*. Tese de Doutorado, DCA-FEEC-UNICAMP. 1995.

[Camargo, 95b] J. T. F. Camargo, L. P. Magalhães e A. B. Raposo. *Local and global control in computer animation*. Proc. do SIBGRAPI'95 (VIII Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens), pp. 151-157. 1995.

[Camargo, 95c] J. T. F. Camargo, L. P. Magalhães e A. B. Raposo. *Fundamentos da animação modelada por computador*. Tutorial apresentado no SIBGRAPI'95 (VIII Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens). 1995.

[Cassandras, 93] C. G. Cassandras. *Discrete event systems: modeling and performance analysis*. Aksen Associates Incorporated. 1993.

[Coolahan, 83] J. E. Coolahan, Jr. and N. Roussopoulos. *Timing Requirements for Time-Driven Systems Using Augmented Petri Nets.* IEEE Transactions on Software Engineering SE-9(5): 603-616. 1983.

[Fishwick, 91] P. A. Fishwick and H. A. Porr. *Using Discrete Event Modeling for Effective Computer Animation Control*. Proc. of Winter Simulation Conference, pp. 1156-1164. December, 1991.

[Gear, 74] C. W. Gear. *Computer Organization and Programming*, 2nd. Edition. McGraw-Hill Book Co. 1974.

[Girard, 85] M. Girard and A. A. Maciejewski. *Computational modelling for the computer animation of legged figures*. ACM Computer Graphics (SIGGRAPH Proc. '85), 19(3): 263-270. July, 1985.

[Gritz, 95] L. Gritz and K. Hahn. *Genetic programming for articulated figure motion*. The Journal of Visualisation and Computer Animation, 6(3): 129-142. 1995.

[Hodgins, 95] J. K. Hodgins et al. *Animating Human Athletics*. Proc. of SIGGRAPH'95, pp. 71-78. August, 1995.

[Hodgins, 98] J. K. Hodgins, J.F. O'Brien and J. Tumblin. *Perception of human motion with different geometric models*. IEEE Transactions on Visualization and Computer Graphics, 4(4): 307-316. October-december, 1998.

[Holland, 75] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press. 1975.

[IKAN] IKAN - *Inverse Kinematics using ANalytical Methods*. http://hms.upenn.edu/software/ik/

[Isaacs, 87] P. M. Isaacs and M. F. Cohen. *Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics*. ACM Computer Graphics (SIGGRAPH Proc. '87), 21(4): 215-224. July, 1987.

[Kalra, 92] D. Kalra and A. H. Barr. *Modeling with Time and Events in Computer Animation.* Computer Graphics forum (*Proc. of EUROGRAPHICS '92*), 11(3): 45-58. 1992.

[Laszlo, 96] J. Laszlo, M. van de Panne and E. Fiume. *Limit Cycle Control And Its Application To The Animation Of Balancing And Walking*. Proc. of SIGGRAPH'96, pp. 155-162. August, 1996.

[Laszlo, 00] J. Laszlo, M. van de Panne and E. Fiume. *Interactive Control For Physically-Based Animation*. Proc. of SIGGRAPH'2000, pp. 201-208. July, 2000.

[Magalhães, 98] L. P. Magalhães, A. B. Raposo and I. L. M. Ricarte. *Animation modeling with Petri nets.* Computers & Graphics, 22(6): 735-743. 1998. Pergamon Press.

[Mendes, 99] R. S. Mendes. Notas do curso *Tópicos em Controle de Sistemas a Eventos Discretos.* DCA-FEEE-UNICAMP. 1999.

[Mesa] *Mesa*. http://mesa3d.sourceforge.net/

[Michalewicz, 96] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag. 1996.

[Murata, 89] T. Murata. *Petri Nets: Properties, Analysis and Applications*. Proceedings of the IEEE, 77(4):541-580. April, 1989.

[Nedel, 98] L. P. Nedel. *Simulating Virtual Humans.* Tutorial apresentado no SIBGRAPI'98 (XI Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens). 1998.

[Neider, 96] J. Neider, T. Davis and M. Woo. *OpenGL Programming Guide*. Addison-Wesley Publishing Company. New York, USA, 1996.

[O'Rourke, 98] M. O'Rourke. *3D Computer Animation Workshop*. SIGGRAPH'98 Course #34 Notes. 1998.

[Petri, 62] C. A. Petri. *Kommunikation mit Automaten*, Schriften des IIM Nr. 3, Bonn: Institute für Instrumentelle Mathematik. 1962.

[Pina, 00] A. Pina, E. Cerezo and F. J. Serón. *Computer animation: from avatars to unrestricted autonomous actors (A survey on replication and modelling mechanisms).* Computer & Graphics, 24: 297-311. 2000.

[Pressman, 92] R. S. Pressman. *Software Engineering - A Practitioner's Approach*, 3rd. Edition. McGraw-Hill, Inc. 1992.

[Raposo, 00] A. B. Raposo. *Coordenação em Ambientes Colaborativos Usando Redes de Petri*. Tese de Doutorado, DCA-FEEC-UNICAMP. 2000.

[Thalmann, 85] N. M. Thalmann and D. Thalmann. *Computer Animation: Theory and Pratice*. Tokyo: Springer-Verlag. 1985.

[Thalmann, 87] N. M. Thalmann and D. Thalmann. *The Direction of Synthetic Actors in the Film Rendez-vous à Montréal*. IEEE Computer Graphics & Applications, pp. 9-19. December, 1987.

[Thalmann, 91] N. M. Thalmann and D. Thalmann. *Complex models for animating synthetic actors*. IEEE Computer Graphics & Applications, pp. 32-44. September, 1991.

[Tif] *How to save an OpenGL-rendered image to file*. http://www.seas.gwu.edu/~graphics/cs206/opengl_save_howto.html

[Tolani, 00] D. Tolani, A. Goswami and N. I. Badler. *Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs*. Graphical Models 62(5):353-388. 2000.

[Tomovic, 66] R. Tomovic and R. B. McGhee. *A Finite State Approach to the Synthesis of Bioengineering Control Systems*. IEEE Transactions on Human Factors in Electronics, HFE-7(2): 65-69. 1966.

[Tost, 88] D. Tost and X. Pueyo. *Human body animation: a survey*. The Visual Computer, 3: 254-264. 1988.

[van der Aalst, 98] W. M. P. van der Aalst. *The Application of Petri Nets to Workflow Management*. The Journal of Circuits, Systems and Computers, 8(1): 21-66. 1998.

[van de Panne, 94] M. van de Panne, R. Kim and E. Fiume. *Virtual Wind-up Toys for Animation*. Proc. of Graphics Interface '94, pp. 208-215. 1994.

[van de Panne, 96] M. van de Panne. *Parameterized Gait Synthesis*. IEEE Computer Graphics and Applications, pp. 40-49. March, 1996.

[Witkin, 88] A. Witkin and M. Kass. *Spacetime Constraints*. Computer Graphics, 22(4): 159-168. 1988.

[Witkin, 99] A. Witkin and Z. Popovi'c. *Physically Based Motion Transformation*. Proc. of SIGGRAPH'99, pp. 11-20. August, 1999.

[Zeltzer, 82] D. Zeltzer. *Motor Control Techniques for Figure Animation*. IEEE Computer Graphics and Applications, 2(9): 53-59. November, 1982.

[Zeltzer, 83] D. Zeltzer. *Knowledge-based Animation*. Proc. ACM SIGGRAPH/SIGART Workshop on Motion, pp. 187-192. 1983.

[Zeltzer, 85] D. Zeltzer. *Towards an integrated view of 3-D computer animation*. The Visual Computer, 1(4): 249-259. 1985.

APÊNDICE A

REDES DE PETRI[1]

Rede de Petri (PN - *Petri net*) [Petri, 62][Murata, 89] é uma ferramenta de modelagem aplicável a uma série de sistemas, especialmente aqueles com concorrência, sincronização e conflito de eventos. Formalmente, uma PN é definida como uma quíntupla (P, T, F, w, M_0), onde:

$P = \{P_1, ..., P_m\}$ é um conjunto finito de lugares (*places*).

$T = \{t_1, ...,t_n\}$ é um conjunto finito de transições.

$F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos.

$w: F \rightarrow \{1, 2, ...\}$ é uma função que atribui peso aos arcos.

$M_0: P \rightarrow \{0, 1, 2, ...\}$ é a marcação inicial da rede (número de *tokens* em cada lugar), com $(P \cap T) = \varnothing$ and $(P \cup T) \neq \varnothing$.

No modelo de PN, os estados estão associados aos lugares e suas marcações, enquanto os eventos às transições. O comportamento de um sistema modelado por PN é descrito em termos de seus estados e suas mudanças [Murata, 89]. Os estados são representados por lugares e *tokens*, que definem o estado atual do sistema. As transições (regras de disparo) modelam o comportamento dinâmico do sistema. Os arcos indicam as seqüências de possíveis transições entre os estados.

O conjunto de lugares de entrada da transição t é representado por $\bullet t$ e o conjunto de lugares de saída de t é representado por $t\bullet$. Similarmente, $\bullet P$ e $P\bullet$ representam, respectivamente, os conjuntos de transições de entrada e saída do lugar P. Uma transição t está habilitada se cada um de seus lugares de entrada $P_i \in \bullet t$ possuir pelo menos $w(P_i , t)$

[1] Baseado em [Raposo, 00].

tokens, onde $w(P_i , t)$ é o peso do arco ligando P_i a t. Estando habilitada, uma transição pode ser disparada quando o evento associado a ela ocorrer. O disparo de t remove $w(P_i , t)$ *tokens* de cada um de seus lugares de entrada P_i e adiciona $w(t, P_o)$ *tokens* a cada lugar de saída $P_o \in t\bullet$.

A notação gráfica de PNs é também muito usada (Figura A.1). Nesta notação, os lugares são representados por círculos, as transições por barras ou retângulos, os *tokens* por pontos e os arcos por setas com os pesos escritos em cima; por definição, um arco não marcado tem peso 1.

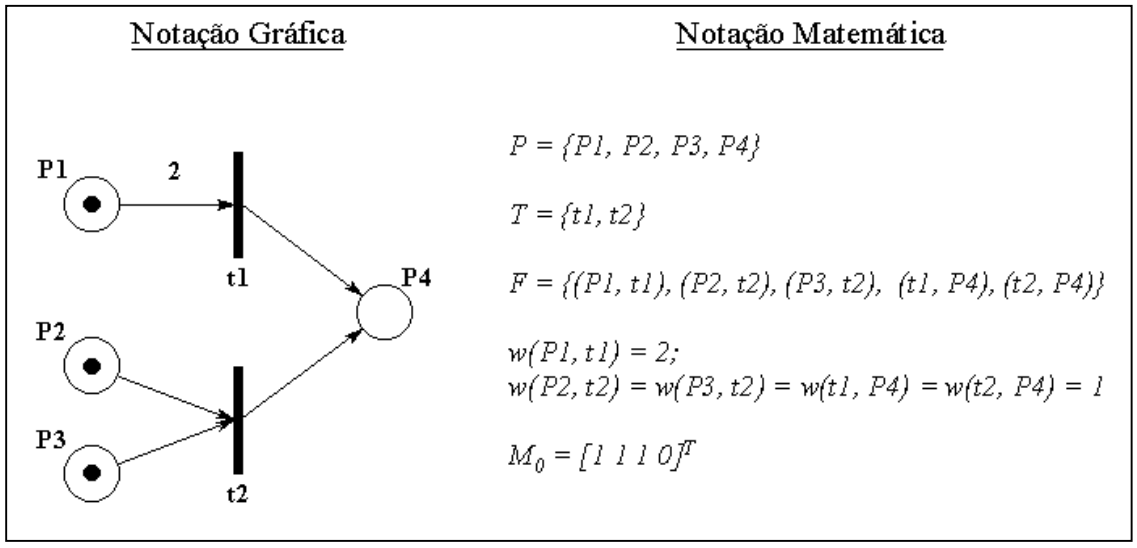


Figura A.1 - A notação gráfica e a notação matemática de PNs.

Na PN dada como exemplo na Figura A.1, apenas a transição t_2 está habilitada; t_1 não está habilitada porque seriam necessários dois *tokens* em P_1 para dispará-la, já que $w(P_1, t_1) = 2$. Quando t_2 for disparada, os *tokens* em P_2 e P_3 são retirados e P_4 recebe um *token*. Notemos que o número de *tokens* não é necessariamente conservado.

Além das notações apresentadas, existe também uma notação matricial para indicar as possíveis mudanças de estado em uma PN. O estado seguinte ao disparo da transição t_j é dado por $M_{k+1} = M_k + C \times e_j$, onde e_j é um vetor coluna com *1* na posição *j* e *0* nas demais posições e $M_k = [q_1 \; q_2 \; ... \; q_m]^T$, onde q_a indica a quantidade de *tokens* no lugar P_a (estado

atual da rede). A matriz *C* representa a topologia da rede, tem dimensões $m \times n$ (*m* é o número de lugares e *n* o é o número de transições) e o elemento c_{ij} indica quantos *tokens* o lugar P_i vai receber (valor positivo) ou perder (valor negativo) quando a transição t_j disparar. Para o exemplo da Figura A.1, a matriz *C* é:

$$C = \begin{bmatrix} -2 & 0 \\ 0 & -1 \\ 0 & -1 \\ 1 & 1 \end{bmatrix} \tag{1}$$

Na matriz acima, o elemento $c_{11} = -2$, por exemplo, indica que *2 tokens* serão retirados de P_1 quando t_1 for disparada. O elemento $c_{42} = 1$ indica que *1 token* será adicionado a P_4 quando t_2 for disparada. A determinação do próximo estado, disparando t_2 a partir do estado inicial (M_0) é feita de acordo com a equação:

$$M_1 = M_0 + c \times e_j = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -2 & 0 \\ 0 & -1 \\ 0 & -1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2}$$

Além do modelo básico, várias extensões de PN existem na literatura [Murata, 89]. Neste trabalho nós utilizamos algumas extensões, a saber: arco inibidor, redes temporizadas e prioridade de disparo das transições.

O arco inibidor liga um lugar *P* a uma transição *t* e funciona de maneira oposta aos arcos comuns. Ele habilita a transição *t* se *P* possuir um número de *tokens* menor que *w(P, t)*. Na notação gráfica, arcos inibidores são representados com um círculo na extremidade (na Figura A.2, o arco ligando P_1 a t_1 é um arco inibidor; nesse caso a transição está habilitada porque não há *tokens* em P_1, enquanto $w(P_1, t_1) = 1$).

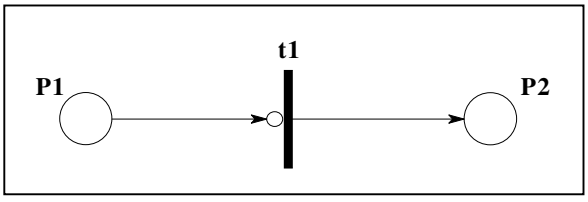


Figura A.2 - Arco inibidor.

A noção de tempo em PNs é importante em algumas situações como na avaliação de desempenho dos sistemas modelados. O modelo básico de PN, no entanto, não faz nenhum tipo de consideração quanto ao tempo de disparo das transições, ou seja, a partir do momento que estão habilitadas, as transições podem disparar. Uma maneira de incluir a noção de tempo em uma PN é estabelecer um tempo de espera para o *token* em um lugar, antes dele habilitar as transições de saída [Coolahan, 83]. Também é possível estabelecer funções de probabilidade para o tempo de disparo de uma transição [Bause, 96]. Neste tipo de PN, chamada PN estocástica, a transição dispara algum tempo depois de habilitada, tempo este determinado pela função de probabilidade associada à transição. O tempo também pode estar associado à "execução" do disparo das transições. Neste caso, os *tokens* não ficam nos lugares de entrada esperando o disparo da transição, mas são retirados deles e algum tempo depois (tempo de disparo) são entregues aos lugares de saída. Este tipo de disparo não-instântaneo é também chamado de disparo com reserva de *tokens*.

A fim de dar maior consistência à dinâmica das PNs, nós definimos um conjunto de prioridades de disparo associado às transições. No caso de ter mais de uma transição habilitada no mesmo instante, somente aquela com a maior prioridade dispará. Se há mais de uma transição habilitada com a mesma prioridade, a escolha de qual deverá disparar será aleatória. Na notação gráfica, a prioridade da transição é indicada através de um número entre parênteses ao lado do nome da transição; por definição, quando este número não é indicado a transição tem prioridade 1.

APÊNDICE B

IKAN
INVERSE KINEMATICS USING ANALYTICAL METHODS

IKAN [Tolani, 00][IKAN] é uma biblioteca de programação que possui um conjunto de algoritmos de cinemática inversa adequado para modelar o movimento de um braço ou de uma perna antropomórfico. Esta biblioteca utiliza uma combinação de métodos analíticos e numéricos para resolver problemas de cinemática inversa como posição, orientação e restrição de objetivo. A combinação destes métodos resulta em algoritmos que apresentam soluções mais rápidas e confiáveis do que os algoritmos convencionais como Jacobiana inversa e técnicas baseadas em otimização. Além disso, diferentemente dos algoritmos numéricos, a biblioteca IKAN permite que o usuário explore interativamente todas as possíveis soluções utilizando um conjunto de parâmetros que define a redundância do sistema.

Este apêndice contém uma descrição detalhada de uma classe disponibilizada pela biblioteca IKAN, a SRS, que foi utilizada neste trabalho. Maiores informações acerca das demais classes desta biblioteca podem ser encontradas em IKAN - *Programmer's Manual*, disponível em [IKAN].

Classe SRS

Dada as matrizes *G, S* e *T* para resolver a equação:

$$G = R_2 \times S \times R_y \times T \times R_1$$

onde:

R_1 e R_2 representam matrizes de rotação

R_y representa a matriz de rotação em relação ao eixo *y* local

e

G é a matriz objetivo desejada

S, T são matrizes constantes

No caso de um braço:

R_2: articulação do pulso

S: transformada do pulso para o cotovelo

R_y: articulação do cotovelo

T: transformada do cotovelo para o ombro

R_1: articulação do ombro

A classe SRS configura e resolve a equação $G = R_2 \times S \times R_y \times T \times R_1$.

Métodos:

- `SRS();`

  Método construtor.

- `SRS(const Matrix &T, const Matrix &S, const float a[3], const float p[3]);`

  Método construtor. A Figura B.1 apresenta um braço indicando os parâmetros. Nota: é importante assumir que o cotovelo (no caso de um braço) e o joelho (no caso de uma perna) possuem 1 DOF e rotacionam em relação ao eixo *y* (local).

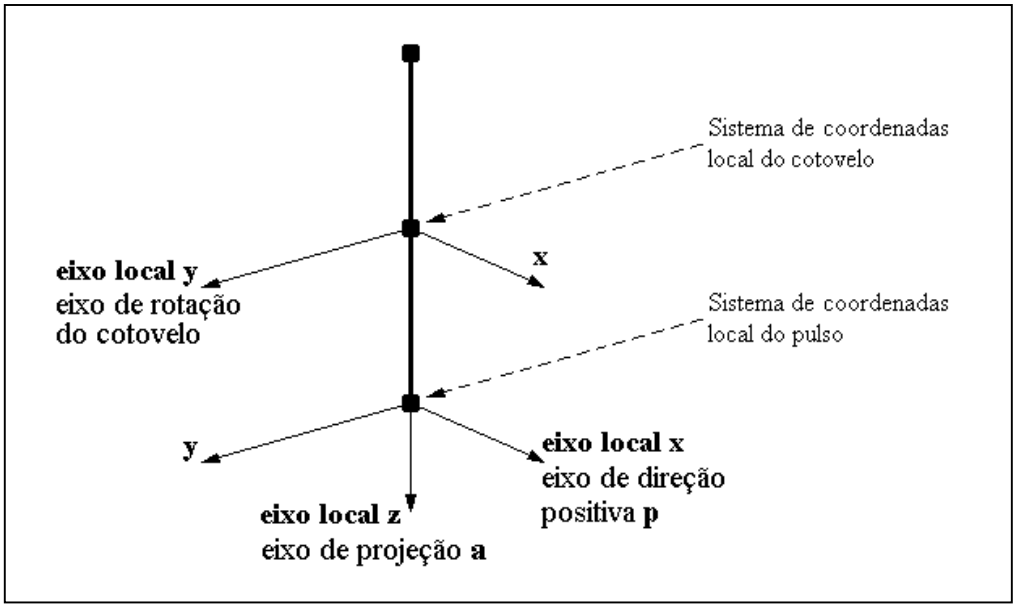


Figura B.1 - Exemplo de um braço mostrando o eixo de projeção `a` e o eixo de direção positiva `p`.

Parâmetros:

`T` Matriz constante. No caso de um braço é a transformada do cotovelo para o ombro.

`S` Matriz constante. No caso de um braço é a transformada do pulso para o cotovelo.

`a` Eixo de projeção usado para determinar o vetor `u` (um dos eixos do sistema de coordenadas local que define o plano contendo a circunferência descrita pelo movimento do cotovelo - Figura B.2).

`p` Eixo de direção positiva que aponta para fora do corpo, usado para determinar a direção positiva do ângulo ϕ (Figura B.2).

- `void init(const Matrix &T, const Matrix &S, const float a[3], const p[3]);`

Rotina de inicialização. Este método somente é utilizado com o método construtor sem parâmetros.

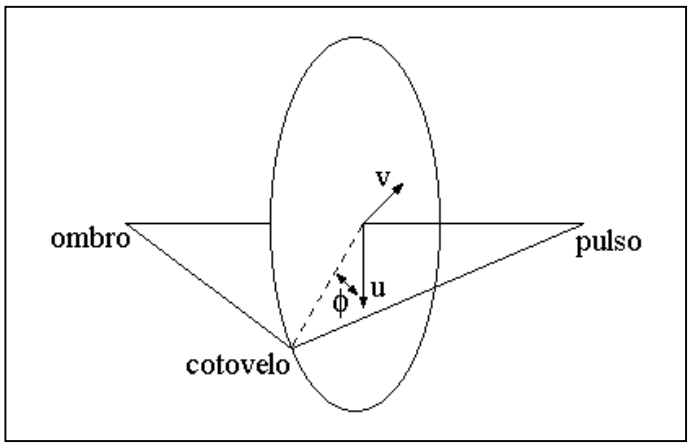


Figura B.2 - Para um dado objetivo, o cotovelo é livre para mover em uma circunferência.

Parâmetros:

`T` Matriz constante. No caso de um braço é a transformada do cotovelo para o ombro.

`S` Matriz constante. No caso de um braço é a transformada do pulso para o cotovelo.

`a` Eixo de projeção usado para determinar o vetor `u` (um dos eixos do sistema de coordenadas local que define o plano contendo a circunferência descrita pelo movimento do cotovelo - Figura B.2).

`p` Eixo de direção positiva que aponta para fora do corpo, usado para determinar a direção positiva do ângulo ϕ (Figura B.2).

- `void ProjectOn();`

  Se o objetivo está além do espaço alcançável, este método projeta o objetivo dentro deste espaço.

- `void ProjectOff();`

  Não projeta o objetivo no espaço alcançável.

- `int SetGoal(const Matrix &G, float &rangle);`

  Dada a matriz objetivo, o ângulo ϕ da articulação *R* (cotovelo ou joelho) é calculado. Este método retorna `1` se o objetivo é possível. Durante este processo, o eixo de projeção previamente inicializado é utilizado para descobrir a posição do *end effector* e para calcular a equação da circunferência que define como a articulação *R* pode rotacionar. A matriz $S \times R_y \times T$ também é calculada e salva para futuro uso. Este método deve ser invocado no início de cada animação.

  Parâmetros:

  `G` Matriz objetivo.

  `rangle` Ângulo da articulação *R*.

- `int SetGoalPos(const float g[3], const Matrix &EE, float &rangle);`

  Dado o vetor posição objetivo, o ângulo ϕ da articulação *R* (cotovelo ou joelho) é calculado. Este método retorna `1` se o objetivo é possível. Durante este processo, o eixo de projeção previamente inicializado é utilizado para descobrir a posição do *end effector* e para calcular a equação da circunferência que define como a articulação *R* pode rotacionar. *EE* é a transformada relativa da posição do *end effector* em relação à posição da articulação do pulso/tornozelo, isto é, relaciona a posição do *R* à posição do *end effector* de acordo com o produto $E \times S$. Subsequentes invocações do método `SolveR1` resolverão a equação de posição $g = [0,0,0,1] \times E \times S \times R_y \times T \times R_1$.

  Parâmetros:

  `g` Vetor posição objetivo.

  `EE` Matriz constante. Transformada relativa da posição do *end effector* em relação à posição da articulação do pulso/tornozelo.

  `rangle` Ângulo da articulação *R*.

- `float PosToAngle(const float p[3]);`

  Dada a posição da articulação *R* (`p`), calcula o ângulo ϕ correspondente. Antes de utilizar este método, deve ser invocado o método `SetGoal` ou `SetGoalPos`.

  Parâmetros:

  `p` Vetor posição da articulação *R*.

- `void AngleToPos(float psi, float p[3]);`

  Dado o ângulo ϕ (`psi`), calcula a posição da articulação *R*. Antes de utilizar este método, deve ser invocado o método `SetGoal` ou `SetGoalPos`.

  Parâmetros:

  `psi` Ângulo ϕ.

  `p` Vetor posição da articulação *R*.

- `void SolveR1R2(const float pos[3], Matrix &R1, Matrix &R2);`

  Calcula R_1 e R_2, dado o vetor posição da articulação *R*.

  Parâmetros:

  `pos` Vetor posição do *end effector*.

  `R1` Matriz de rotação que satisfaz $G = R_2 \times S \times R_y \times T \times R_1$.

  `R2` Matriz de rotação que satisfaz $G = R_2 \times S \times R_y \times T \times R_1$.

- `void SolveR1R2(const float angle, Matrix &R1, Matrix &R2);`

  Calcula R_1 e R_2, dado o ângulo da articulação *R*.

  Parâmetros:

  `angle` Ângulo da articulação *R*.

  `R1` Matriz de rotação que satisfaz $G = R_2 \times S \times R_y \times T \times R_1$.

  `R2` Matriz de rotação que satisfaz $G = R_2 \times S \times R_y \times T \times R_1$.

- `int R1Psi(Matrix alpha, Matrix beta, Matrix gamma);`

  Retorna as equações psi da matriz R_1, isto é, *alpha[i][j]*cos(phi) + beta[i][j]*sin(phi) + gamma[i][j] = R1[i][j]*. É necessário invocar primeiro o método `SetGoal`.

  Parâmetros:

  `alpha` Valor α na equação acima.

  `beta` Valor β na equação acima.

  `gamma` Valor γ na equação acima.

- `int R1R2Psi(Matrix alpha, Matrix beta, Matrix gamma, Matrix alpha2, Matrix beta2, Matrix gamma2);`

  Retorna as equações psi das matrizes de rotação R_1 e R_2, de forma análoga ao método `R1Psi`. É necessário invocar primeiro o método `SetGoal`.

  Parâmetros:

  `alpha` Valor α na equação acima para R_1.

  `beta` Valor β na equação acima para R_1.

  `gamma` Valor γ na equação acima para R_1.

  `alpha2` Valor α na equação acima para R_2.

  `beta2` Valor β na equação acima para R_2.

  `gamma2` Valor γ na equação acima para R_2.

- `void SolveR1(const float pos[3], Matrix &R1);`

  Calcula somente R_1 utilizando o vetor posição, depois de invocar o método `SetGoalPos`.

  Parâmetros:

  `pos` Vetor posição.

  `R1` Matriz de rotação.

- `void SolveR1(float angle, Matrix &R1);`

  Calcula somente R_1 utilizando o ângulo de rotação, depois de invocar o método `SetGoalPos`.

  Parâmetros:

  `angle` Ângulo de rotação.

  `R1` Matriz de rotação.

- `void SetAimGoal(const float goal[3], const float axis[3], float flex_angle);`

  Seta o objetivo para um problema alvo.

  Parâmetros:

  `goal` O ponto que nós queremos apontar.

  `axis` O eixo indicando a mão.

  `flex_angle` A quantidade de flexão no cotovelo.

- `void SetAim(float psi_angle, Matrix &R1);`

  Resolve o problema alvo para um dado ângulo da circunferência da mão. É necessário invocar primeiro o método `SetAimGoal`.

  Parâmetros:

  `psi_angle` O ângulo ψ para a equação.

  `R1` A matriz de rotação para a equação.

APÊNDICE C

PNs DO ESTUDO DE CASOS EM LINGUAGEM DE DESCRIÇÃO TOPOLÓGICA

C.1. O caminhar de uma figura bípede

A seguinte linguagem descreve a topologia da PN apresentada na Figura 5.4:

```
24 11

P P1 0 1
P P2a 0.5 0
P P2b 0.5 0
P P2c 0.5 0
P P2d 0.5 0
P P3 0 0
P P4a 1 0
P P4b 1 0
P P4c 1 0
P P4d 1 0
P P5 0 0
P P6a 1 0
P P6b 1 0
P P6c 1 0
P P6d 1 0
P P7a 0.5 0
P P7b 0.5 0
P P7c 0.5 0
P P7d 0.5 0
P P8a 0.5 0
P P8b 0.5 0
P P8c 0.5 0
P P8d 0.5 0
P P9 0 0

T T1 1
T T2 2
T T3 2
T T4 2
T T5 3
```

```
T T6 1
T T7 1
T T8 1
T T9 1
T T10 1
T T11 1

I P1 T1 1
O P2a T1 1
O P2b T1 1
O P2c T1 1
O P2d T1 1
O P3 T1 1

I P2a T2 1
I P2b T2 1
I P2c T2 1
I P2d T2 1
IB P5 T2
O P4a T2 1
O P4b T2 1
O P4c T2 1
O P4d T2 1
O P3 T2 1

I P4a T3 1
I P4b T3 1
I P4c T3 1
I P4d T3 1
IB P5 T3
O P6a T3 1
O P6b T3 1
O P6c T3 1
O P6d T3 1
O P3 T3 1

I P6a T4 1
I P6b T4 1
I P6c T4 1
I P6d T4 1
IB P5 T4
O P4a T4 1
O P4b T4 1
O P4c T4 1
O P4d T4 1
O P3 T4 1

I P3 T5 3
O P5 T5 1

I P2a T6 1
I P2b T6 1
I P2c T6 1
I P2d T6 1
O P8a T6 1
```

```
O P8b T6 1
O P8c T6 1
O P8d T6 1

I P4a T7 1
I P4b T7 1
I P4c T7 1
I P4d T7 1
O P7a T7 1
O P7b T7 1
O P7c T7 1
O P7d T7 1

I P6a T8 1
I P6b T8 1
I P6c T8 1
I P6d T8 1
O P8a T8 1
O P8b T8 1
O P8c T8 1
O P8d T8 1

I P7a T9 1
I P7b T9 1
I P7c T9 1
I P7d T9 1
O P9 T9 1

I P8a T10 1
I P8b T10 1
I P8c T10 1
I P8d T10 1
O P9 T10 1

I P9 T11 1
```

C.2. O pular de uma figura bípede

A seguinte linguagem descreve a topologia da PN apresentada na Figura 5.5:

```
24 7

P P1a 1.5 1
P P1b 1.5 1
P P1c 1.5 1
P P1d 1.5 1
P P2a 1 0
P P2b 1 0
P P2c_P1a 0.5 0
P P2c_P1b 0.5 0
P P2c_P2a 0.5 0
P P2c_P2b 0.5 0
P P3a 1 0
P P3b 1 0
P P3c 1 0
P P4a 1 0
P P4b 1 0
P P4c 1 0
P P4d 1 0
P P5a 1.5 0
P P5b 1.5 0
P P5c 1.5 0
P P6a 1.5 0
P P6b 1.5 0
P P6c 1.5 0
P P6d 1.5 0

T T1 1
T P2c_T1 1
T T2 1
T T3 1
T T4 1
T T5 1
T T6 1

I P1a T1 1
I P1b T1 1
I P1c T1 1
I P1d T1 1
O P2a T1 1
O P2b T1 1
O P2c_P1a T1 1
O P2c_P1b T1 1

I P2c_P1a P2c_T1 1
I P2c_P1b P2c_T1 1
O P2c_P2a P2c_T1 1
```

```
O P2c_P2b P2c_T1 1

I P2a T2 1
I P2b T2 1
I P2c_P2a T2 1
I P2c_P2b T2 1
O P3a T2 1
O P3b T2 1
O P3c T2 1

I P3a T3 1
I P3b T3 1
I P3c T3 1
O P4a T3 1
O P4b T3 1
O P4c T3 1
O P4d T3 1

I P4a T4 1
I P4b T4 1
I P4c T4 1
I P4d T4 1
O P5a T4 1
O P5b T4 1
O P5c T4 1

I P5a T5 1
I P5b T5 1
I P5c T5 1
O P6a T5 1
O P6b T5 1
O P6c T5 1
O P6d T5 1

I P6a T6 1
I P6b T6 1
I P6c T6 1
I P6d T6 1
```

C.3. As figuras bípede jogando bola

A seguinte linguagem descreve a topologia da PN apresentada na Figura 5.8:

```
136 70

P P1_P1 0 1
P P1_P2a 0.5 0
P P1_P2b 0.5 0
P P1_P3 0 0
P P1_P4a 1 0
P P1_P4b 1 0
P P1_P5 0 0
P P1_P6a 1 0
P P1_P6b 1 0
P P1_P7a 0.5 0
P P1_P7b 0.5 0
P P1_P8a 0.5 0
P P1_P8b 0.5 0
P P1_P9 0 0
P P2_P1 0.5 0
P P2_P2a 0.5 0
P P2_P2b 0.5 0
P P3_P1a 0.5 0
P P3_P1b 0.5 0
P P3_P1c 0.5 0
P P3_P1d 0.5 0
P P4_P1a 0.5 0
P P4_P1b 0.5 0
P P4_P1c 0.5 0
P P4_P1d 0.5 0
P P4_P2 0.5 0
P P4_P3a 0.5 0
P P4_P3b 0.5 0
P P5_P1 0 0
P P5_P2a 0.5 0
P P5_P2b 0.5 0
P P5_P2c 0.5 0
P P5_P2d 0.5 0
P P5_P3 0 0
P P5_P4a 1 0
P P5_P4b 1 0
P P5_P4c 1 0
P P5_P4d 1 0
P P5_P5 0 0
P P5_P6a 1 0
P P5_P6b 1 0
P P5_P6c 1 0
P P5_P6d 1 0
P P5_P7a 0.5 0
P P5_P7b 0.5 0
```

```
P P5_P7c 0.5 0
P P5_P7d 0.5 0
P P5_P8a 0.5 0
P P5_P8b 0.5 0
P P5_P8c 0.5 0
P P5_P8d 0.5 0
P P5_P9 0 0
P P6_P1 0.5 0
P P6_P2a 0.5 0
P P6_P2b 0.5 0
P P6_P3a 0.5 0
P P6_P3b 0.5 0
P P6_P3c 0.5 0
P P6_P3d 0.5 0
P P7_P1a 0.5 0
P P7_P1b 0.5 0
P P7_P2a 0.5 0
P P7_P2b 0.5 0
P P8_P1 0.5 0
P P8_P2a 0.5 0
P P8_P2b 0.5 0
P P9 1.5 0
P P10 1 0
P P11 1.5 0
P P12 1 0
P P13_P1 0 1
P P13_P2a 0.5 0
P P13_P2b 0.5 0
P P13_P2c 0.5 0
P P13_P2d 0.5 0
P P13_P3 0 0
P P13_P4a 1 0
P P13_P4b 1 0
P P13_P4c 1 0
P P13_P4d 1 0
P P13_P5 0 0
P P13_P6a 1 0
P P13_P6b 1 0
P P13_P6c 1 0
P P13_P6d 1 0
P P13_P7a 0.5 0
P P13_P7b 0.5 0
P P13_P7c 0.5 0
P P13_P7d 0.5 0
P P13_P8a 0.5 0
P P13_P8b 0.5 0
P P13_P8c 0.5 0
P P13_P8d 0.5 0
P P13_P9 0 0
P P14_P1 0.5 0
P P14_P2a 0.5 0
P P14_P2b 0.5 0
P P14_P3a 0.5 0
P P14_P3b 0.5 0
P P14_P3c 0.5 0
```

```
P P14_P3d 0.5 0
P P15_P1a 0.5 0
P P15_P1b 0.5 0
P P15_P2a 0.5 0
P P15_P2b 0.5 0
P P16_P1 0.5 0
P P16_P2a 0.5 0
P P16_P2b 0.5 0
P P17_P1 0 0
P P17_P2a 0.5 0
P P17_P2b 0.5 0
P P17_P3 0 0
P P17_P4a 1 0
P P17_P4b 1 0
P P17_P5 0 0
P P17_P6a 1 0
P P17_P6b 1 0
P P17_P7a 0.5 0
P P17_P7b 0.5 0
P P17_P8a 0.5 0
P P17_P8b 0.5 0
P P17_P9 0 0
P P18_P1 0.5 0
P P18_P2a 0.5 0
P P18_P2b 0.5 0
P P19_P1a 0.5 0
P P19_P1b 0.5 0
P P19_P1c 0.5 0
P P19_P1d 0.5 0
P P20_P1a 0.5 0
P P20_P1b 0.5 0
P P20_P1c 0.5 0
P P20_P1d 0.5 0
P P20_P2 0.5 0
P P20_P3a 0.5 0
P P20_P3b 0.5 0

T P1_T1 1
T P1_T2 2
T P1_T3 2
T P1_T4 2
T P1_T5 3
T P1_T6 1
T P1_T7 1
T P1_T8 1
T P1_T9 1
T P1_T10 1
T T1 1
T P2_T1 1
T T2 1
T T3 1
T P4_T1 1
T P4_T2 1
T T4 1
T P5_T1 1
```

```
T P5_T2 2
T P5_T3 2
T P5_T4 2
T P5_T5 3
T P5_T6 1
T P5_T7 1
T P5_T8 1
T P5_T9 1
T P5_T10 1
T T5 1
T P6_T1 1
T P6_T2 1
T T6 1
T P7_T1 1
T T7 1
T P8_T1 1
T T8 1
T P13_T1 1
T P13_T2 2
T P13_T3 2
T P13_T4 2
T P13_T5 3
T P13_T6 1
T P13_T7 1
T P13_T8 1
T P13_T9 1
T P13_T10 1
T T9 1
T P14_T1 1
T P14_T2 1
T T10 1
T P15_T1 1
T T11 1
T P16_T1 1
T T12 1
T P17_T1 1
T P17_T2 2
T P17_T3 2
T P17_T4 2
T P17_T5 3
T P17_T6 1
T P17_T7 1
T P17_T8 1
T P17_T9 1
T P17_T10 1
T T13 1
T P18_T1 1
T T14 1
T T15 1
T P20_T1 1
T P20_T2 1
T T16 1

I P1_P1 P1_T1 1
O P1_P2a P1_T1 1
```

```
O P1_P2b P1_T1 1
O P1_P3 P1_T1 1

I P1_P2a P1_T2 1
I P1_P2b P1_T2 1
IB P1_P5 P1_T2
O P1_P4a P1_T2 1
O P1_P4b P1_T2 1
O P1_P3 P1_T2 1

I P1_P4a P1_T3 1
I P1_P4b P1_T3 1
IB P1_P5 P1_T3
O P1_P6a P1_T3 1
O P1_P6b P1_T3 1
O P1_P3 P1_T3 1

I P1_P6a P1_T4 1
I P1_P6b P1_T4 1
IB P1_P5 P1_T4
O P1_P4a P1_T4 1
O P1_P4b P1_T4 1
O P1_P3 P1_T4 1

I P1_P3 P1_T5 2
O P1_P5 P1_T5 1

I P1_P2a P1_T6 1
I P1_P2b P1_T6 1
O P1_P8a P1_T6 1
O P1_P8b P1_T6 1

I P1_P4a P1_T7 1
I P1_P4b P1_T7 1
O P1_P7a P1_T7 1
O P1_P7b P1_T7 1

I P1_P6a P1_T8 1
I P1_P6b P1_T8 1
O P1_P8a P1_T8 1
O P1_P8b P1_T8 1

I P1_P7a P1_T9 1
I P1_P7b P1_T9 1
O P1_P9 P1_T9 1

I P1_P8a P1_T10 1
I P1_P8b P1_T10 1
O P1_P9 P1_T10 1

I P1_P5 T1 1
I P1_P9 T1 1
O P2_P1 T1 1

I P2_P1 P2_T1 1
```

```
O P2_P2a P2_T1 1
O P2_P2b P2_T1 1

I P2_P2a T2 1
I P2_P2b T2 1
I P9 T2 1
O P3_P1a T2 1
O P3_P1b T2 1
O P3_P1c T2 1
O P3_P1d T2 1

I P3_P1a T3 1
I P3_P1b T3 1
I P3_P1c T3 1
I P3_P1d T3 1
O P4_P1a T3 1
O P4_P1b T3 1
O P4_P1c T3 1
O P4_P1d T3 1
O P10 T3 1

I P4_P1a P4_T1 1
I P4_P1b P4_T1 1
I P4_P1c P4_T1 1
I P4_P1d P4_T1 1
O P4_P2 P4_T1 1

I P4_P2 P4_T2 1
O P4_P3a P4_T2 1
O P4_P3b P4_T2 1

I P4_P3a T4 1
I P4_P3b T4 1
O P5_P1 T4 1

I P5_P1 P5_T1 1
O P5_P2a P5_T1 1
O P5_P2b P5_T1 1
O P5_P2c P5_T1 1
O P5_P2d P5_T1 1
O P5_P3 P5_T1 1

I P5_P2a P5_T2 1
I P5_P2b P5_T2 1
I P5_P2c P5_T2 1
I P5_P2d P5_T2 1
IB P5_P5 P5_T2
O P5_P4a P5_T2 1
O P5_P4b P5_T2 1
O P5_P4c P5_T2 1
O P5_P4d P5_T2 1
O P5_P3 P5_T2 1

I P5_P4a P5_T3 1
I P5_P4b P5_T3 1
```

```
I P5_P4c P5_T3 1
I P5_P4d P5_T3 1
IB P5_P5 P5_T3
O P5_P6a P5_T3 1
O P5_P6b P5_T3 1
O P5_P6c P5_T3 1
O P5_P6d P5_T3 1
O P5_P3 P5_T3 1

I P5_P6a P5_T4 1
I P5_P6b P5_T4 1
I P5_P6c P5_T4 1
I P5_P6d P5_T4 1
IB P5_P5 P5_T4
O P5_P4a P5_T4 1
O P5_P4b P5_T4 1
O P5_P4c P5_T4 1
O P5_P4d P5_T4 1
O P5_P3 P5_T4 1

I P5_P3 P5_T5 3
O P5_P5 P5_T5 1

I P5_P2a P5_T6 1
I P5_P2b P5_T6 1
I P5_P2c P5_T6 1
I P5_P2d P5_T6 1
O P5_P8a P5_T6 1
O P5_P8b P5_T6 1
O P5_P8c P5_T6 1
O P5_P8d P5_T6 1

I P5_P4a P5_T7 1
I P5_P4b P5_T7 1
I P5_P4c P5_T7 1
I P5_P4d P5_T7 1
O P5_P7a P5_T7 1
O P5_P7b P5_T7 1
O P5_P7c P5_T7 1
O P5_P7d P5_T7 1

I P5_P6a P5_T8 1
I P5_P6b P5_T8 1
I P5_P6c P5_T8 1
I P5_P6d P5_T8 1
O P5_P8a P5_T8 1
O P5_P8b P5_T8 1
O P5_P8c P5_T8 1
O P5_P8d P5_T8 1

I P5_P7a P5_T9 1
I P5_P7b P5_T9 1
I P5_P7c P5_T9 1
I P5_P7d P5_T9 1
O P5_P9 P5_T9 1
```

```
I P5_P8a P5_T10 1
I P5_P8b P5_T10 1
I P5_P8c P5_T10 1
I P5_P8d P5_T10 1
O P5_P9 P5_T10 1

I P5_P5 T5 1
I P5_P9 T5 1
O P6_P1 T5 1
O P11 T5 1

I P6_P1 P6_T1 1
O P6_P2a P6_T1 1
O P6_P2b P6_T1 1

I P6_P2a P6_T2 1
I P6_P2b P6_T2 1
O P6_P3a P6_T2 1
O P6_P3b P6_T2 1
O P6_P3c P6_T2 1
O P6_P3d P6_T2 1

I P6_P3a T6 1
I P6_P3b T6 1
I P6_P3c T6 1
I P6_P3d T6 1
I P12 T6 1
O P7_P1a T6 1
O P7_P1b T6 1

I P7_P1a P7_T1 1
I P7_P1b P7_T1 1
O P7_P2a P7_T1 1
O P7_P2b P7_T1 1

I P7_P2a T7 1
I P7_P2b T7 1
O P8_P1 T7 1

I P8_P1 P8_T1 1
O P8_P2a P8_T1 1
O P8_P2b P8_T1 1

I P8_P2a T8 1
I P8_P2b T8 1
O P1_P1 T8 1

I P13_P1 P13_T1 1
O P13_P2a P13_T1 1
O P13_P2b P13_T1 1
O P13_P2c P13_T1 1
O P13_P2d P13_T1 1
O P13_P3 P13_T1 1
```

```
I P13_P2a P13_T2 1
I P13_P2b P13_T2 1
I P13_P2c P13_T2 1
I P13_P2d P13_T2 1
IB P13_P5 P13_T2
O P13_P4a P13_T2 1
O P13_P4b P13_T2 1
O P13_P4c P13_T2 1
O P13_P4d P13_T2 1
O P13_P3 P13_T2 1

I P13_P4a P13_T3 1
I P13_P4b P13_T3 1
I P13_P4c P13_T3 1
I P13_P4d P13_T3 1
IB P13_P5 P13_T3
O P13_P6a P13_T3 1
O P13_P6b P13_T3 1
O P13_P6c P13_T3 1
O P13_P6d P13_T3 1
O P13_P3 P13_T3 1

I P13_P6a P13_T4 1
I P13_P6b P13_T4 1
I P13_P6c P13_T4 1
I P13_P6d P13_T4 1
IB P13_P5 P13_T4
O P13_P4a P13_T4 1
O P13_P4b P13_T4 1
O P13_P4c P13_T4 1
O P13_P4d P13_T4 1
O P13_P3 P13_T4 1

I P13_P3 P13_T5 3
O P13_P5 P13_T5 1

I P13_P2a P13_T6 1
I P13_P2b P13_T6 1
I P13_P2c P13_T6 1
I P13_P2d P13_T6 1
O P13_P8a P13_T6 1
O P13_P8b P13_T6 1
O P13_P8c P13_T6 1
O P13_P8d P13_T6 1

I P13_P4a P13_T7 1
I P13_P4b P13_T7 1
I P13_P4c P13_T7 1
I P13_P4d P13_T7 1
O P13_P7a P13_T7 1
O P13_P7b P13_T7 1
O P13_P7c P13_T7 1
O P13_P7d P13_T7 1

I P13_P6a P13_T8 1
```

```
I P13_P6b P13_T8 1
I P13_P6c P13_T8 1
I P13_P6d P13_T8 1
O P13_P8a P13_T8 1
O P13_P8b P13_T8 1
O P13_P8c P13_T8 1
O P13_P8d P13_T8 1

I P13_P7a P13_T9 1
I P13_P7b P13_T9 1
I P13_P7c P13_T9 1
I P13_P7d P13_T9 1
O P13_P9 P13_T9 1

I P13_P8a P13_T10 1
I P13_P8b P13_T10 1
I P13_P8c P13_T10 1
I P13_P8d P13_T10 1
O P13_P9 P13_T10 1

I P13_P5 T9 1
I P13_P9 T9 1
O P14_P1 T9 1
O P9 T9 1

I P14_P1 P14_T1 1
O P14_P2a P14_T1 1
O P14_P2b P14_T1 1

I P14_P2a P14_T2 1
I P14_P2b P14_T2 1
O P14_P3a P14_T2 1
O P14_P3b P14_T2 1
O P14_P3c P14_T2 1
O P14_P3d P14_T2 1

I P14_P3a T10 1
I P14_P3b T10 1
I P14_P3c T10 1
I P14_P3d T10 1
I P10 T10 1
O P15_P1a T10 1
O P15_P1b T10 1

I P15_P1a P15_T1 1
I P15_P1b P15_T1 1
O P15_P2a P15_T1 1
O P15_P2b P15_T1 1

I P15_P2a T11 1
I P15_P2b T11 1
O P16_P1 T11 1

I P16_P1 P16_T1 1
O P16_P2a P16_T1 1
```

```
O P16_P2b P16_T1 1

I P16_P2a T12 1
I P16_P2b T12 1
O P17_P1 T12 1

I P17_P1 P17_T1 1
O P17_P2a P17_T1 1
O P17_P2b P17_T1 1
O P17_P3 P17_T1 1

I P17_P2a P17_T2 1
I P17_P2b P17_T2 1
IB P17_P5 P17_T2
O P17_P4a P17_T2 1
O P17_P4b P17_T2 1
O P17_P3 P17_T2 1

I P17_P4a P17_T3 1
I P17_P4b P17_T3 1
IB P17_P5 P17_T3
O P17_P6a P17_T3 1
O P17_P6b P17_T3 1
O P17_P3 P17_T3 1

I P17_P6a P17_T4 1
I P17_P6b P17_T4 1
IB P17_P5 P17_T4
O P17_P4a P17_T4 1
O P17_P4b P17_T4 1
O P17_P3 P17_T4 1

I P17_P3 P17_T5 2
O P17_P5 P17_T5 1

I P17_P2a P17_T6 1
I P17_P2b P17_T6 1
O P17_P8a P17_T6 1
O P17_P8b P17_T6 1

I P17_P4a P17_T7 1
I P17_P4b P17_T7 1
O P17_P7a P17_T7 1
O P17_P7b P17_T7 1

I P17_P6a P17_T8 1
I P17_P6b P17_T8 1
O P17_P8a P17_T8 1
O P17_P8b P17_T8 1

I P17_P7a P17_T9 1
I P17_P7b P17_T9 1
O P17_P9 P17_T9 1

I P17_P8a P17_T10 1
```

```
I P17_P8b P17_T10 1
O P17_P9 P17_T10 1

I P17_P5 T13 1
I P17_P9 T13 1
O P18_P1 T13 1

I P18_P1 P18_T1 1
O P18_P2a P18_T1 1
O P18_P2b P18_T1 1

I P18_P2a T14 1
I P18_P2b T14 1
I P11 T14 1
O P19_P1a T14 1
O P19_P1b T14 1
O P19_P1c T14 1
O P19_P1d T14 1

I P19_P1a T15 1
I P19_P1b T15 1
I P19_P1c T15 1
I P19_P1d T15 1
O P20_P1a T15 1
O P20_P1b T15 1
O P20_P1c T15 1
O P20_P1d T15 1
O P12 T15 1

I P20_P1a P20_T1 1
I P20_P1b P20_T1 1
I P20_P1c P20_T1 1
I P20_P1d P20_T1 1
O P20_P2 P20_T1 1

I P20_P2 P20_T2 1
O P20_P3a P20_T2 1
O P20_P3b P20_T2 1

I P20_P3a T16 1
I P20_P3b T16 1
O P13_P1 T16 1
```

APÊNDICE D

ARTIGO PUBLICADO

Neste apêndice é apresentado o artigo publicado pelo autor e que está diretamente relacionado ao conteúdo apresentado neste trabalho.

- [Bicho, 01] A de L. Bicho, A. B. Raposo e L. P. Magalhães, *Control of Articulated Figures Animations Using Petri Nets*. Proc. of the SIBGRAPI'2001 - XIV Brazilian Symposium on Computer Graphics and Image Processing. Florianópolis, SC, Brazil, October, 2001. SBC - Sociedade Brasileira de Computação, IEEE Press.

Control of Articulated Figures Animations Using Petri Nets

ALESSANDRO DE LIMA BICHO
ALBERTO BARBOSA RAPOSO
LÉO PINI MAGALHÃES
Department of Computer Engineering and Industrial Automation - DCA
School of Electrical and Computer Engineering - FEEC
State University of Campinas - UNICAMP
Caixa Postal 6101, 13083-970 Campinas, SP, Brasil
{bicho, alberto, leopini}@dca.fee.unicamp.br

Abstract. In this paper we explore the use of Petri Nets as a tool to control the movements of articulated figures in computer animations. This approach permits us to describe the animation sequence by means of the treatment of events present in its execution. An advantage of this method is that the control may be abstracted in different levels, spanning from the definition of the relation among limbs for a single movement to behavioral directives. In addition, our treatment of events hides the mathematical model that describes the movement in fact, allowing the animators to choose the better technique for their applications. In this paper we use an inverse kinematics tool for this purpose. The use of Petri Nets also allows previewing the behavior of the animation before starting any shot.

1 Introduction

This paper investigates one of the facets of computer animation, which is the control of articulated figures movements. Movement control strategies can be thought as driven by two categories of algorithms, one directed to specific aspects of animating characters' parameters–e.g., a movement equation–and another to high level control aspects, where animation intentions are expressed as, for example, goal directed control strategies. Figure 1 illustrates this idea [1].

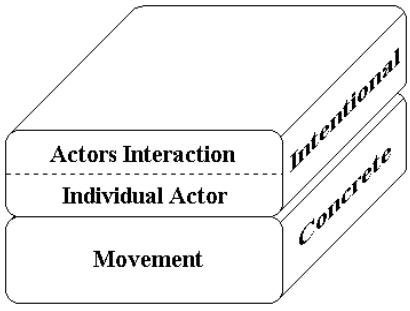


Figure 1 Movement modeling techniques framework.

At the concrete level, movements are considered strictly according to their mathematical modeling. Paradigms like key-frames interpolation, kinematics and dynamic models and biological techniques are examples of this class of solutions. The intentional level, on the other hand, is built on top of the concrete level and uses techniques that support more abstract directives, allowing to define events, restrictions, reactions, etc. Representative techniques in this context are behavior-based models, reactive systems, etc. The main objective at this level is not to define how movements will be performed, but the sequence of events that causes or affects a movement.

In parallel to these two levels, we also proposed the division of the animation control problem into two parts, local control and global control [2]. The local control suggests a mathematical tool, such as inverse kinematics, for modeling an individual character (e.g., a limb of an articulated figure). The global control, on the other hand, suggests a logical approach to manage the interaction among the characters (e.g., arms and legs of a biped figure). The definition of what is local or global control, however, is dependent of the context, since there can be different abstraction levels for control. For instance, both the relation between two actors and the relation between the leg and the foot of a biped figure can be considered global control, depending on the abstraction level we intend to work.

In this paper, we explore the use of Petri Nets (PNs) as a global control tool, used on top of an inverse kinematics tool (IKAN–Inverse Kinematics using Analytical Methods) [3], which implements the local control of articulated figures in the concrete level. In the following section we review some techniques related to our work. In Section 3 we discuss PNs fundamentals and

present the use of this tool for the control of articulated figures animations. Section 4 presents some examples of the PN-based control in different abstraction levels to biped figures. Finally, Section 5 contains the conclusions of this work.

2 Related Work

Many approaches for the control of articulated figures movements have been proposed in the literature. The work of Tmovic and McGhee suggested the application of automata theory of finite-state to the analysis and synthesis of bioengineering systems [4]. Using this approach, the behavior of a natural leg during a steady walk may be modeled by means of a finite-state model. Therefore, it is possible to design a finite-state controller to coordinate ankle and knee motion in prosthesis. This theory has been applied to animation systems.

Zeltzer created a hierarchical model in which the movement is obtained by "motor control programs", which are finite-state machines for executing a particular class of movements, such as "walk" or "run" (there are parameters that allow for different performances) [5]. These programs represent the highest abstraction level of the model. Their states invoke a fixed set of lower abstraction level programs called "local motor programs" (LMPs) to manipulate the joints. LMPs are also finite-state machines that access the joints by changing parameter values. A drawback of this approach is that the animator looses artistic control in favor of automatic motion synthesis.

The work of Fishwick and Porr [6] presented a method for combining discrete event modeling methods with key-frame computer animation. This approach uses PNs to represent the system dynamics at a fairly high abstraction level, and therefore complex systems can be represented as networks or hierarchies of discrete event and continuous models. The focus was to study existing methods in computer simulation, such as PNs, that may be used to aid the graphics community. The authors presented a common example in the literature about synchronization among processes in the operating systems theory (the dining philosophers' scenario) and produced an animation by key-framing method which was not very realistic.

Kalra and Barr [7] introduced a representation of time in which simulation can be neatly partitioned into sub-behaviors connected through events. They formalized the concepts of events and created time primitives called *event units* that may be hierarchically organized to construct motion sequences. This approach provides a partitioning for the problem of motion design, namely, a hierarchical scheme to compose motion behaviors from time primitives and a programming model for organizing animation.

Camargo et al. [2] divided the computer simulation control problem into two parts, namely, a local control problem and a global control problem, as previously mentioned. They proposed an event-oriented scheme to solve the global control problem using concepts related to DEDS (Discrete Event Dynamic System) and ESM (Extended State Machines). This approach used an extension of the "Space-Time Constraints" paradigm of computer modeled animation, named "Space-Time-Event Constraints" paradigm. In the presented example, the walking model for an articulated biped figure, a complex tree figure is split into several smaller blocks, simplifying the initial problem.

Our approach uses a formal framework based on PNs as a global control tool that is better than the ESM because it allows an animator to preview the behavior of an animation even before starting any shot. With PNs it is possible to define a hierarchical model of the actor movements, like that proposed by Zeltzer [5], and it is also suited to event-based systems, like that of Kalra and Barr [7]. Fishwick and Porr [6] also used PNs in their approach. They used the key-framing method rather than inverse kinematics as the local control tool and their transitions, rather than the places, were temporized. Moreover, the abstraction level used in that work does not concern the interaction among limbs of an articulated figure, differently from our approach.

3 Control Using Discrete Events

In animation environments, the movements of articulated figures may be controlled using the time as a frame clock, whose ticks indicate the moments when the behavior of the system must be changed. In spite of that, behaviors in an animation system may also be controlled using event-based tools in a more abstract level. An event means an important point in an animation or physical simulation. The control by means of events enables the animator to easily specify a desired animation sequence, because the treatment using the events domain facilitates changes in parts of an animation without having to remodel all or a large part of the sequence [7]. It also provides a nice composition methodology so that complex figures can be created by combining simpler figures, defining a hierarchical model that facilitates the movement control.

In the work of Magalhães et al. [8] the use of PNs as a modeling and analysis tool for animation environments was presented, showing to be suited for systems with concurrency, synchronization and event conflicts. In the present paper we apply this methodology to control articulated figures, since behavior of the animation of

these structures have such characteristics. In the following we discuss PNs fundamentals and present the approach for the control using this tool.

3.1 Petri Nets Fundamentals

Petri Nets [9, 10] are a modeling tool applicable to a variety of fields and systems, specially suited for systems with concurrency, synchronization and event conflicts. Formally, a PN can be defined as a 5-tuple (P, T, F, w, M_0), where: $P = \{P_1, ..., P_m\}$ is a finite set of places; $T = \{t_1, ..., t_n\}$ is a finite set of transitions; $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs; $w: F \rightarrow \{1, 2, ...\}$ is a weight function; $M_0: P \rightarrow \{0, 1, 2, ...\}$ is the initial marking; with $(P \cap T) = \varnothing$ and $(P \cup T) \neq \varnothing$.

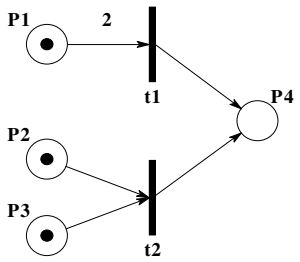
In a PN model, states are associated to places and marks (also called tokens), and events to transitions. A transition t is said to be enabled if each input place $P_i \in {}^\bullet t$ is marked with at least $w(P_i, t)$, which is the weight of the arc between P_i and t. Once enabled, a transition will fire when its associated event occurs. Firing transition t, $w(P_i, t)$ tokens are removed from each input place P_i and $w(t, P_o)$ tokens are added to each output place $P_o \in t^\bullet$. Here, ${}^\bullet t$ and $t^\bullet$ means, respectively, the set of input and output places of transition t.

A very useful notation for PNs is the graphical notation (Figure 2) which is used in the examples throughout this paper. In this notation, circles represent places, rectangles represent transitions, dots represent tokens and arrows represent the arcs, with weights above. By definition, an unlabeled arc has weight 1.

In the PN of Figure 2, only transition *t2* is enabled; *t1* is not enabled because it would require two tokens in *P1* to fire, since *w(P1, t1) = 2*. When *t2* is fired, the tokens in *P2* and *P3* are removed and *P4* receives one token. Note that the number of marks in a PN is not necessarily conserved.

In addition to the basic PN model, several extensions appear in the literature [10]. In this paper we use some extensions, namely, inhibitor arcs, fire priorities to transitions and timed nets. An inhibitor arc connects a place P with a transition t and enables t only if P has no tokens. In the graphical notation, inhibitor arcs are represented with a circle on the edge. The basic PN model does not consider the notion of time. One way to include this notion is to establish a waiting time for the tokens in a place before they enable the output transitions [11]. In order to give more consistency to the dynamics of the PNs we have defined a set of firing priorities associated with the transitions. In the case of having more than one transition enabled, only that with higher priority will fire. If there is more than one transition enabled with the same priority, it is necessary to use a random function to define which one will fire. In the graphical notation, the priority of the transition is indicated by means of a number in parenthesis at the side of the transition label. By definition, when this number is not indicated, the transition has priority equal to 1.

Graphical Notation



Mathematical Notation

$P = \{P1, P2, P3, P4\}$

$T = \{t1, t2\}$

$F = \{(P1, t1), (P2, t2), (P3, t2), (t1, P4), (t2, P4)\}$

$w(P1, t1) = 2;\ w(P2, t2) = w(P3, t2) = w(t1, P4) = w(t2, P4) = 1$

$M_0 = [1\ 1\ 1\ 0]^T$

Figure 2 PN graphical and mathematical notations.

3.2 The Control Method

Before presenting our method, we must define the body model that will be controlled. In this paper, the stick figure model, which consists of a hierarchical set of rigid segments (limbs) connected at spherical joints, will be used. The main advantage of the stick figure model is that the motion specification is easy because it is only necessary to give, for each joint, the values to its three degrees of freedom (DOF). An important point here is that any body model may be animated by moving an underlying skeletal approximation, which does not have to bear any resemblance to the final rendered appearance of the figure. Thus, the motion control problem for figures reduces to that of controlling the movement of an abstract articulated skeleton that, in this case, is a stick figure model.

3.2.1 Global Control

In the global control level we use the PN theory described previously. Two types of places are defined in our model, *condition place* and *action place*. *Condition places* are used to represent conditions of the net control and do not model any movement. *Action places* are used to represent movements that will be executed by an actor. In a lower abstraction level, action places represent the movement of a limb or a set of limbs of the actor. In a higher abstraction level action places may represent, for instance, activities such as "walk" or "jump".

The execution of the movement is modeled by a transition representing its instantaneous starting, with a directed arc to a place representing the movement being executed. An execution time T_i is assigned to each place P_i. A token becomes ready to aid in enabling an output transition of place P_i only T_i time units after P_i received the token. This approach, in which the timed places represent a movement or a net condition, has been chosen for three reasons:

- It preserves the classic PN notion of transitions as instantaneous events.
- It is possible to represent and execute more than one movement concurrently. If we had represented the movements as transitions, only one could fire at an instant, therefore executing only one movement.
- It does not obscure the state of the system (represented by the net marking) during the time that a movement is executing.

3.2.2 Local Control

The local control level represents the effective movements of a stick figure model. For this purpose, we adopted the inverse kinematics method. This method was chosen because it is only necessary to specify discrete positions and motions for end parts. The system then computes the necessary joint angles and orientations for other parts of the body to put the specified parts in the desired position, executing the necessary motions. Trying to animate an articulated figure using, for example, the forward kinematics method is completely intuitive but tedious to do in practice. The motions of the end parts are determined indirectly as the accumulation of all transformations that lead to those end parts. We also do not use the forward dynamics method because it is necessary to know forces and physical laws *a priori* to realistically simulate determined movements. For example, it is not easy to find physical laws and joint torque patterns to predict how the leg will move in a walking biped figure. An alternative to this problem could be to use the inverse dynamics method to analyze the torque and forces required for the given motion, but this method is also very difficult to use. Consequently, we decided to use the inverse kinematics method in the local control level in this work.

We used a tool named IKAN that provides an inverse kinematics algorithm to compute the desired limb posture [3]. This tool uses a combination of analytical and numerical methods to solve generalized inverse kinematics problems including position, orientation, and aiming constraints. The combination of analytical and numerical methods results in faster and more reliable algorithms than conventional inverse Jacobean and optimization-based techniques. This method also allows for the user to interactively explore all possible solutions using an intuitive set of parameters that defines the redundancy of the system.

3.2.3 Global Control over Local Control

Although it is necessary to have a local control method, an advantage of our approach is to abstract the movement technique used by means of action places in the global control level. This allows that the animator chooses the best technique to simulate the desired movement without having to remodel the sequence of the animation, because it was defined in the event domain. Many simulations of the articulated figures movements have implicit discrete behaviors. For example, in a biped figure walk there are events (behavioral rules) that must be used so that the animation looks realistic [8]:

- Both legs may not be out of the ground at the same time.
- The same leg may not be raised more than one time sequentially.

Furthermore, we may also synchronize the arms motions to those of the legs. Thus, the left arm motions may be synchronized with the right leg motions and the right arm motions may be synchronized with that of the left leg, characterizing the classic behavior of this skill.

In this case, using a modeling tool to describe this cyclic sequence of movements facilitates the work of the animator. It is only necessary to construct, in the time domain, a set of movement functions that will be used in the local control level, such as forward limb, backward limb, and so on. It is used the time domain in this level because the events could not change the behavior of these movements. Thus, these movements are organized in the global control level considering the events that cause transitions among them. This provides an ability to partition the motion design problem into smaller problems of determining behavioral rules and of determining events that connect the movements that compose these rules.

Moreover, if the animator wants to change parameters of the movements, e.g., the size and/or velocity of a step, it is only necessary to alter the respective functions in the local control level. In another situation, if the animator wants to remodel the walk such that the arms do not move anymore, it is necessary to remove the action places in the global control level that invoke the functions responsible by the arms movements, without changing the rest of the animation control model.

4 Experimental Results

To illustrate the feasibility of our method, this section presents two examples of movement control in different abstraction levels to biped figures that resemble human beings. Thus, both upper and lower limbs of these figures have the same characteristics, i.e., the same DOF.

In the first example, we model the control among limbs to simulate the walk of a figure. In the second one, in a higher abstraction level, we model the control between two figures to simulate a ball game.

4.1 Biped Figure Walking

In the computer animation field the human walk has been exhaustively studied and, at least conceptually, it is well understood. The walk is a cyclic sequence in which the legs swing forward and backward, providing alternatively support to the body. This sequence is achieved by imposing some "behavioral rules" mentioned in the previous section, which we used in this example.

In order to solve this motion control problem, we define that the global control level is responsible for the coordination among the legs and arms. Thus, the possible movements of each limb are defined using an approach partially based on the model present by Girard and Maciejewski [12]. We use in this work a number of parameters presented in that paper as necessary for describing the gait of a biped figure.

A *gait pattern* describes the sequence of lifting and placing of the feet. The pattern repeats itself as the figure moves; each repetition of the sequence is called the *gait cycle*. The time taken to complete a single gait cycle is the period *P* of the cycle. Moreover, the duration of the leg phases is called *support duration* and *transfer duration*. Hence, we have:

$$P = SupportDuration + TransferDuration \quad (1)$$

During each gait cycle period any given leg will spend a percentage of that time on the ground. This fraction is called the *duty factor* of leg and it is given by:

$$DutyFactor = \frac{SupportDuration}{P} \quad (2)$$

The walk requires that the duty factor of the each leg exceeds 0.5 since, by definition, both feet must be on the ground simultaneously for a percentage of the gait cycle period. Duty factors less than 0.5 would result in running skill, because the entire body would leave the ground for some duration. In addition, we may define the *stroke* as the distance traveled by the body during leg support duration. For our example, the values for stroke and duration will be, respectively, equal to *x* and *n*. Consequently, the velocity of the step will be *x/n*.

The movement rules and parameters, and also the feet and hands trajectories are defined in the local control level and encapsulated in action places in the PN that models the global control level.

We modeled the PN for the global control level (Figure 3) according to previously defined "behavioral rules". Table 1 presents the function defined to each condition place and Table 2 presents the movements that should be executed for each action place. For simplicity and without losing generality in the modeling of the net, it is defined that the duty factor is equal to 0.5. In addition, the arms motions are synchronized with the legs motions. Thus, it is necessary a single action place to execute these motions and consequently a single transition to represent their instantaneous beginnings.

The animation sequence of a biped figure walking in a straight line behaves in the following way. The figure is initially at rest position, such that the arms and legs are parallel to the length of the body. After the figure have walked the number of steps defined in the weight of the arc between place P3 and transition t5, which in our example is equal to 3, it will be again at rest position. Thus, the first and last steps will have half the size and duration of a normal one. In the PN of Figure 3 (result in Figure 4), the first step is modeled by P2 (frames 1 to 10), the last one by P7 or P8 (frames 50 to 60), whereas the entire step is modeled by P4 or P6 (frames 10 to 50). The start of a step, except the last one, adds one token to P3, and its finish enables two output transitions of the place at issue. While there is not a token in P5, the transition enabled with higher priority will fire, indicating that the walk is not finished. Otherwise, the inhibitor arc will disable this transition and only the transition enabled for the last step fires.

An action place in Figure 3 is responsible for a set of limbs movements. However, in a lower control level, we could assign for each action place a single limb movement. Thus, there would be four action places executing concurrently for each step.

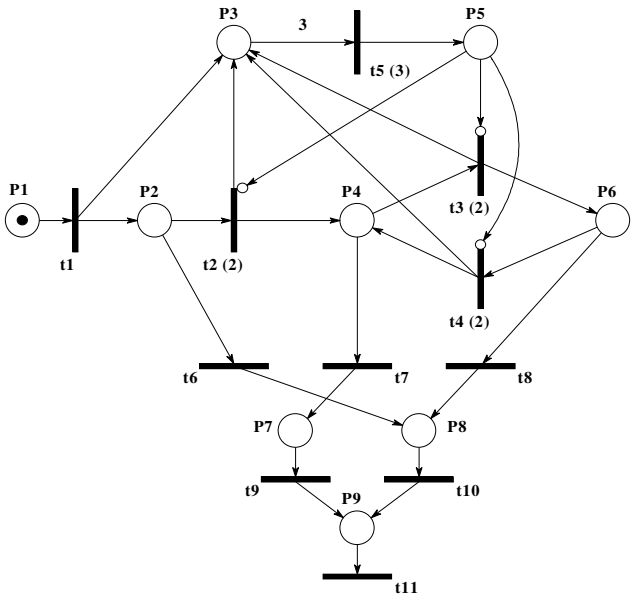


Figure 3 PN model for the global control of the walk of a biped figure.

Condition places	Function
P1	start walk
P3	steps counter
P5	steps counter equals to the number of steps defined
P9	finish walk

Table 1 Condition places.

Action places	Movements	Parameters values
P2, P7	left leg supports the body; right leg transfers forward; left arm swings forward; right arm swings backward	time = n/2 stroke = x/2
P8	left leg transfers forward; right leg supports the body; left arm swings backward; right arm swings forward	
P4	left leg transfers forward; right leg supports the body; left arm swings backward; right arm swings forward	time = n stroke = x
P6	left leg supports the body; right leg transfers forward; left arm swings forward; right arm swings backward	

Table 2 Action places for the movements.

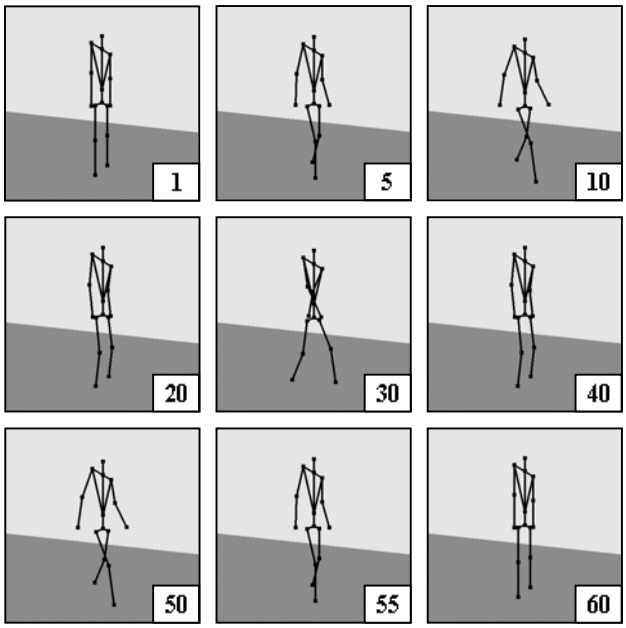


Figure 4 The biped figure walking.

The partitioning could change the movement of a determined limb more easily and also facilitate the reuse of this PN in another situation, removing unnecessary places.

4.2 Biped Figures Playing Ball

This example illustrates the control of articulated figures movements in a higher control level. Hence, the intention is to demonstrate that our approach using PNs is suitable for different abstraction levels. The animation consists of two biped figures walking in a road and playing a ball. Thus, from time to time the actors stop and that with the ball throws it for the other. The walk continues, always alternating the possession of the ball. It is important to observe that the actors do not necessarily walk the same distance or the same number of steps before stopping. Therefore, the interactions among these actors may be described by means of events (behavioral rules) that must be used in the animation control:

- The actor with the ball, after executing his walk, continues only after throwing the ball for the other.
- The actor without the ball, after executing his walk, continues only after grasping the ball.
- The actor with the ball may only throw it if the other one is ready to grasp the ball.

We modeled the PN for the global control level (Figure 5) according to these behavioral rules. Table 3 presents the movements of the actors that should be executed for action places of the net.

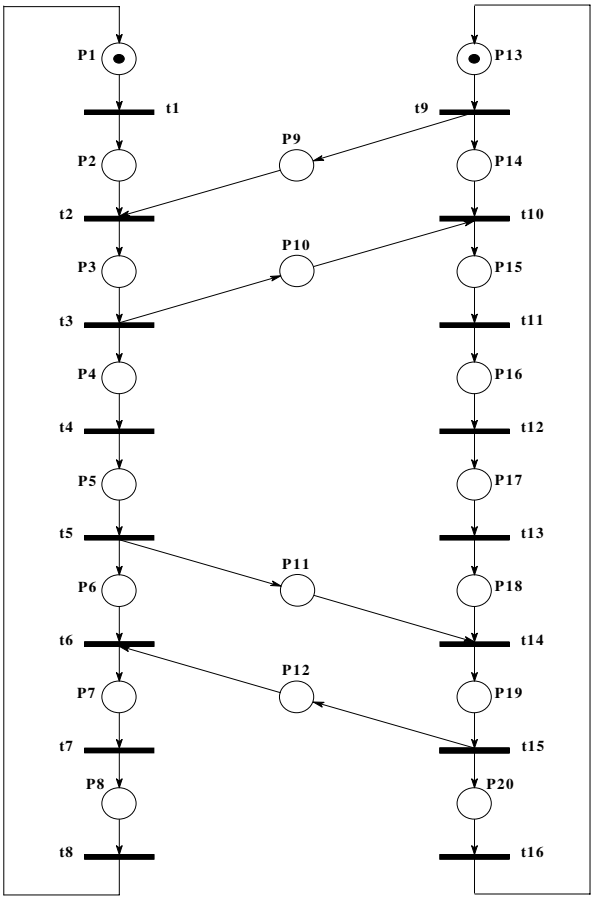


Figure 5 PN model for the global control of the figures playing ball.

Movement	Action place	
	Actor A	**Actor B**
Walk with the ball	P1	P17
Position to throw the ball	P2	P18
Throw the ball	P3	P19
Position to walk without the ball	P4	P20
Walk without the ball	P5	P13
Position to get the ball	P6	P14
Get the ball	P7	P15
Position to walk with the ball	P8	P16

Table 3 Action places of the actors movements.

The action places P10 and P12 model the ball's trajectory. The condition places P9 and P11 indicate that the actor is ready to grasp the ball. For this model, these places are temporized (they have the same time that the action places P14 and P6, respectively). Thus, the actor will have time to position the body to grasp the ball. In this example the actors could walk infinitely, but it does not occur because we defined a limit of frames for the animation. Figure 6 presents the resulting animation, with actor A on the left side and actor B on the right side of each frame.

This example illustrates not only the reuse of basic PN models (the movement of walking, for example, is modeled by the PN presented in the previous example), but also the capacity of encapsulation offered by PNs. The graph of Figure 5 hides the details of the movements, such as walk or position, enforcing a hierarchical description model.

5 Conclusion

In this paper we used the notions of global and local control to implement a PN-based method for the global control of articulated figures movements that works over an inverse kinematics library (local control). Taking into account the division of movement control strategies into concrete and intentional, we consider our approach as belonging to the concrete level, although it has a higher abstraction level than the direct use of inverse kinematics. The event-based model and the use of PNs, although yet mathematical tools, hide the low level "operations" of the inverse kinematics. For this reason, our approach is a concrete level control method that steps towards the intentional level.

The use of PNs as movement control tool is justified because they are amenable both to simulation and formal verification, since there are numerous tools and techniques available. Moreover, PNs may accommodate models at different abstraction levels, ranging from models closely related to the local control (such as the control of each limb of an articulated figure, as shown in the walking biped example) to higher level models (such as the ball game example, that deals with the interaction among actors).

It is our belief that in order to make use of the full potential of computer animation, it is necessary to provide means for application-oriented users like artists to use physically based control methods. On the other hand, for users like scientists interested in complex simulations, it is important to avoid the trial and error method to compose their animated visualizations. Our PN-based control approach is a step towards both directions, in the sense that it provides a higher abstraction level tool and also a powerful analysis and simulation tool.

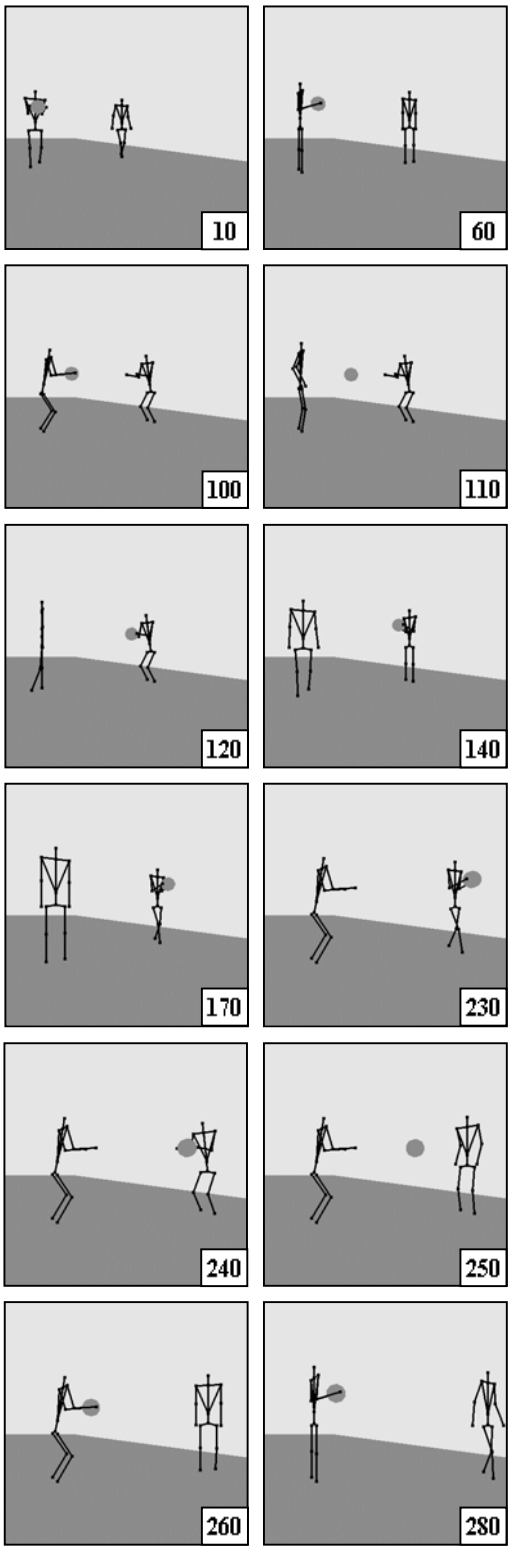


Figure 6 The biped figures playing ball.

Acknowledgements

The first author is sponsored by CAPES. The second author is sponsored by FAPESP, grant number 00/10247-3. Special thanks to Prof. Norman Badler and his research group at the Computer and Information Science Department, University of Pennsylvania, for sending us the IKAN library and related bibliography.

References

[1] L. P. Magalhães, "Modeling and Analysing Computer Animations", *Proc. of XI SIBGRAPI* (1998), 18–19.

[2] J. T. F. de Camargo, L. P. Magalhães and A. B. Raposo, "Local and Global Control in Computer Animation", *Proc. of VIII SIBGRAPI* (1995), 151–157.

[3] D. Tolani, A. Goswami and N. I. Badler, "Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs", *Graphical Models* 62(5) (2000), 353–388.

[4] R. Tomovic and R. B. McGhee, "A Finite State Approach to the Synthesis of Bioengineering Control Systems", *IEEE Transactions on Human Factors in Electronics* HFE-7(2) (1966), 65–69.

[5] D. Zeltzer, "Motor Control Techniques for Figure Animation", *IEEE Computer Graphics and Applications* 2(9) (1982), 53–59.

[6] P. A. Fishwick and H. A. Porr, "Using Discrete Event Modeling for Effective Computer Animation Control", *Proc. of Winter Simulation Conference* (1991), 1156–1164.

[7] D. Kalra and A. H. Barr, "Modeling with Time and Events in Computer Animation", *Computer Graphics forum* 11(3) (1992), 45–58, (*Proc. of EUROGRAPHICS '92*).

[8] L. P. Magalhães, A. B. Raposo and I. L. M. Ricarte, "Animation Modeling with Petri Nets", *Computers & Graphics* 22(6) (1998), 735–743.

[9] C. A. Petri, "Kommunikation mit Automaten", Schriften des IIM Nr. 3, Bonn: Institute für Instrumentelle Mathematik, 1962.

[10] T. Murata, "Petri Nets: Properties, Analysis and Applications", *Proc. of the IEEE* 77(4) (1989), 541–580.

[11] J. E. Coolahan, Jr. and N. Roussopoulos, "Timing Requirements for Time-Driven Systems Using Augmented Petri Nets", *IEEE Transactions on Software Engineering* SE-9(5) (1983), 603–616.

[12] M. Girard and A. A. Maciejewski, "Computational Modeling for the Computer Animation of Legged Figures", *Computer Graphics* 19(3) (1985), 263–270, (*Proc. of SIGGRAPH '85*).