VIII SIBGRAPI

SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS

TUTORIAL:

FUNDAMENTOS DA ANIMAÇÃO MODELADA POR COMPUTADOR

José Tarcísio Franco de Camargo
e-mail: tarcisio@dca.fee.unicamp.br
Léo Pini Magalhães
e-mail: leopini@dca.fee.unicamp.br
Alberto Barbosa Raposo
e-mail: alberto@dca.fee.unicamp.br

Outubro 1995

Índice

Lista de Figuras

RESUMO / OBJETIVOS

O campo da "Animação por Computador" é uma rica área de investigação científica, além de possuir um grande potencial de mercado. De fato, a crescente demanda por profissionais nesta área tem exigido uma expansão do número de pessoas que atuam na área de animação por computador.

Este tutorial tem por objetivo apresentar ao público conceitos relativos ao desenvolvimento de animações por computador. Serão abordados desde as ferramentas matemáticas necessárias para a modelagem de animações até os métodos de implementação de animações, incluindo o projeto dos atores, o desenvolvimento de roteiros, a geração dos quadros de uma animação, etc. Adicionalmente, pretendemos ainda no decorrer do tutorial apresentar exemplos práticos de animações, ilustrando os conceitos apresentados.

Para um correto acompanhamento dos temas a serem desenvolvidos, o público deste tutorial deve dispor de conhecimento básico nas áreas de Computação Gráfica e Cálculo, além de muita criatividade para explorar ao máximo os conceitos apresentados.

A meta fundamental deste curso é estimular o interesse pela área de animação por computador, de forma a ampliar o número de interessados, principalmente alunos de graduação e pós-graduação desenvolvendo atividades nesta área.

Capítulo 1

INTRODUÇÃO À ANIMAÇÃO POR COMPUTADOR

A utilização de computadores expandiu-se muito desde a sua criação. Inicialmente esta utilização era restrita à execução de cálculos matemáticos e ao apoio de tarefas administrativas; hoje encontra ampla gama de aplicação na automação e criação, influenciando fortemente a própria linguagem artística. Neste último ponto, interessa-nos o campo da Animação, onde o computador pode atuar como ferramenta de automação do processo produtivo ou mesmo como elemento de expressão da criatividade.

O processo convencional de criação de uma animação constitui um exemplo claro de um conjunto de atividades que, quando automatizadas, proporcionam grande otimização de tempo, de custos, de confiabilidade, etc.; o que deslocou a atenção de cientistas da computação e animadores para uma possível fusão computador-animação. Este é, portanto, o escopo deste curso: estudar a aplicação do computador como ferramenta de apoio ao desenvolvimento de animações. Para tanto, as seções seguintes traçam um paralelo entre a animação convencional e a animação por computador, introduzindo as várias formas de intervenção do computador no processo de animação.

1.1 Animação convencional

Existem diversas definições possíveis para o termo ANIMAÇÃO. Entre elas podemos citar:

1. Arte de "dar vida" a personagens.
2. Ilusão da sensação de movimento através da apresentação de uma sequência de quadros (imagens) sobrepostos ao longo do tempo.

Entretanto, uma animação não precisa, necessariamente, envolver deslocamentos de objetos, podendo também ser caracterizada por:

1. Uma metamorfose: quando um personagem muda sua forma.
2. Uma mudança de cor: quando, por exemplo, um ator se ruboriza de vergonha.

3. Uma variação na intensidade luminosa: por exemplo, a luz ambiente diminui à medida em que o Sol se põe.

O processo de criação de uma animação convencional, ou "desenho animado", pode ser descrito, genericamente, pela seguinte sequência de etapas, segundo [**Thalmann (1985)**]:

1. **A definição da estória:**
   Como em um filme qualquer, o desenho animado também possui um enredo. Para se descrever o enredo três "documentos" são necessários, cada um refinando um pouco mais seu antecessor:

   A Sinópse, que é um resumo da estória em poucas linhas.

   O Enredo propriamente dito, que é um texto detalhado que descreve toda a estória.

   O Storyboard, que é um "rascunho" do desenho animado. Este consiste de um número de ilustrações arranjadas como em uma estória-em-quadrinhos. O número total de ilustrações não é importante; o que realmente importa é que o *storyboard* deve representar os momentos principais da animação. Também é importante notar que a animação é composta por um conjunto de **sequências** que definem ações específicas. Por sua vez, cada sequência é composta por um conjunto de **cenas** que são definidas por uma certa localização e um conjunto de atores que dela participam. Mais ainda, uma cena é composta por um conjunto de **quadros**.

2. **A etapa de desenho:**
   Este passo consiste principalmente do desenho dos personagens a serem animados e das ações a serem executadas. Aqui também são definidas características particulares quanto à forma dos atores, quanto ao relacionamento destes com o ambiente, etc.

3. **A trilha sonora:**
   Na animação convencional a trilha sonora deve preceder o processo de animação, visto que o movimento deve "ser casado" aos diálogos e/ou música.

4. **A animação:**
   Nesta etapa, o processo de animação é levado adiante pelos animadores que desenharão os quadros principais (ou quadros-chave / *keyframes*) da animação.

5. **Interpolação dos quadros-chave:**
   São denominados *in-betweens* os desenhos que interpolam dois quadros-chave. Nesta etapa são, portanto, desenhados os quadros que interpolam os quadros-chave desenhados na etapa anterior.

6. **A etapa de cópia:**
   Até aqui os quadros eram desenhados a lápis. Eles agoram precisam ser transferidos para folhas de acetato. Os contornos devem ser desenhados a tinta manualmente.

7. **Pintura:**
   Visto que os desenhos animados são geralmente coloridos, as células de acetato devem, portanto, passar por um estágio de pintura. Este trabalho requer paciência e precisão.

8. **Verificação:**
   Nesta etapa os animadores devem verificar se as ações em suas respectivas cenas estão corretas, antes de se fotografar os quadros.

9. **Fotografando os quadros:**
   A etapa de fotogravação da animação é agora realizada em um filme colorido.

10. **Edição final:**
    Este último passo é considerado como uma etapa de pós-produção, depois da qual teremos o produto final desejado.

1.2 Animação por computador

As tarefas descritas anteriormente podem, também, dispor de um certo grau de automação. Neste caso, o computador pode auxiliar muito o processo de elaboração de uma animação. Por exemplo, o computador pode auxiliar na execução das seguintes tarefas:

1. **Na criação dos desenhos:**
   i-) Os quadros-chave podem ser digitalizados.
   ii-) Os quadros-chave podem ser criados através de um editor gráfico interativo.
   iii-) Objetos complexos podem ser criados através de algum programa específico.

2. **Na criação do movimento:**
   O computador pode realizar o cálculo dos quadros *in-betweens* ou até mesmo cálcular algum movimento complexo.

3. **Na pintura:**
   Os desenhos podem ser pintados utilizando-se sistemas gráficos interativos.

4. **Na etapa de fotogravação:**
   Uma câmera pode ser controlada por um computador.

5. **Na etapa de pós-produção:**
   Edição e sincronização podem ser controladas por computador.

Para os casos acima descritos (todos 2D), dizemos que a animação é **auxiliada por computador**.

A participação do computador no processo de criação de uma animação pode ainda ser efetivada de outras maneiras. O computador pode, por exemplo, ser responsável por todo o processo de controle da geração de uma animação. Neste caso dizemos que a animação é **modelada por computador**.

Uma animação pode ser considerada **modelada por computador** quando o animador determina os atores, o ambiente e as ações a serem executadas, cabendo ao computador o controle da animação como um todo. Através da animação modelada por computador podemos gerar desde animações até simulações:

1. **animações:**
   Desenhos animados tradicionais (*cartoons*), onde não existe qualquer compromisso com a realidade física.

2. **simulações:**
   Onde simulamos fenômenos físicos tais como o deslocamento do braço de um robô, um lançamento balístico, etc.

Ainda em **[Thalmann (1985)]** os sistemas de animação por computador são classificados em diversos níveis, de acordo com a participação do computador no processo de criação de uma animação:

1. **Nível 1:**
   O computador é utilizado para que, interativamente, sejam realizadas as tarefas de criação, pintura, armazenamento e edição dos desenhos. Tais sistemas são, basicamente, editores gráficos utilizados pelos desenhistas.

2. **Nível 2:**
   O computador é capaz de gerar os quadros de interpolação (*in-betweens*), além de mover um objeto ao longo de uma determinada trajetória. Estes sistemas são utilizados como alternativa aos desenhistas que realizam os quadros de (*in-between*).

3. **Nível 3:**
   O computador torna disponível ao animador um conjunto de operações que podem ser aplicadas aos objetos, agora 3D. Por exemplo, operações de translação ou rotação. Tais sistemas podem, também, incluir facilidades de manipulação de câmeras, tais como *zoom*, *tilt* ou panorâmica.

4. **Nível 4:**
   Aqui o computador é capaz de tornar disponível ao animador um mecanismo de definição dos atores, ou seja, é posível distinguir atores dotados de movimento próprio de outros elementos da cena (tais como a decoração, etc.). O movimento dos atores pode também ser modelado por intermédio de restrições.

5. **Nível 5:**
   Aqui se encontram os sistemas denominados "inteligentes". Neste caso o computador é capaz de modelar atores capazes de aprender à medida em que realizam suas tarefas. Este aprendizado pode consistir, por exemplo, na capacidade do ator alterar seu movimento em função da ocorrência de determinados eventos.

Os sistemas de animação auxiliada por computador geralmente encontram-se nos níveis 1 e 2 acima descritos. Por sua vez, os sistemas de animação modelada por computador geralmente englobam os níveis 3, 4 e 5.

Outra proposta interessante para a classificação de sistemas de animação por computador pode ser encontrada em **[Isaacs (1987)]**. Segundo estes autores um sistema de animação por computador pode ser enquadrado em uma das seguintes categorias:

1. **Animação keyframe**, ou por quadros-chave:
   Derivada diretamente da animação convencional, onde o animador especifica o que deve aparecer em cada quadro. Dessa forma, o animador está explicitamente especificando a "cinemática" ou o movimento dos objetos. Adicionalmente, o computador aumenta a eficiência do processo de criação interpolando quadros de *in-between* entre os quadros-chave (*keyframes*) fornecidos pelo animador. Embora este método de animação permita ao animador controlar quase que totalmente o processo de animação, ele é muito limitado no que diz respeito à criação de sequências "dinamicamente corretas".

2. **Animação procedimental:**
   Métodos de animação procedimentais confiam ao computador a determinação da cinemática do movimento, através da especificação de instruções para o movimento, ao invés da especificação de posições explícitas. Tais métodos são também conhecidos como **animação por roteiros**, onde o animador descreve a animação através de um determinado texto (roteiro), cabendo ao computador a interpretação do mesmo e, consequentemente, a geração do movimento desejado. Através de um sistema de animação procedimental o animador aproxima-se da geração de sequências "dinamicamente corretas".

3. **Simulação dinâmica:**
   Esta classe de sistemas consiste em uma "especialização" da animação procedimental. Neste caso, o objetivo é a geração de uma simulação, ou seja, o desenvolvimento de sequências estritamente "dinamicamente corretas". Portanto, as ferramentas de modelagem de sistemas deste tipo são as leis físicas, tais como as equações de Newton-Euler, etc.

Como podemos notar através de todas as definições anteriormente apresentadas, existe um número muito grande definições que buscam classificar melhor a área da "animação por computador". A proposta de nosso trabalho segue de perto os conceitos relacionados à **animação modelada por computador**. Portanto, a partir deste ponto, nos prenderemos ao estudo de sistemas de animação deste tipo, sendo que na seção seguinte procuraremos detalhar melhor esta abordagem.

1.3 Componentes de um sistema de animação modelada por computador

Durante o processo de criação de uma animação modelada por computador, necessitamos de, pelo menos, três ferramentas fundamentais:

1. **Um modelador geométrico:**
   para que possamos definir e modelar as características geométricas das entidades envolvidas na animação.

2. **Um mecanismo de controle da animação:**
   para que possamos definir e executar o controle das interações entre atores e entre estes e o ambiente.

3. **Um mecanismo de *rendering* e visualização:**
   para que possamos definir os atributos de cor e textura dos objetos, além de visualizar cada quadro gerado durante o processo de animação.

A **modelagem geométrica** dos objetos presentes em uma animação pode ser realizada de diversas maneiras, usualmente são utilizadas a representação por fronteiras (*boundary representation / b-rep*) ou a combinação de formas primitivas por intermédio de operações **CSG** (*Construtive Solid Geometry*) ou, ainda, uma combinação destes.

Em muitos casos o modelador geométrico já se encontra disponível em conjunto com o mecanismo de *rendering*, como é o caso do pacote de geração de imagens por *ray tracing* denominado **POV-Ray** (*Persistence of Vision Ray Tracer*) [**Young (1994)**].

O **mecanismo de rendering e visualização** é responsável pela geração das imagens que comporão a animação. Este mecanismo deve reunir as informações sobre a forma, a cor e a textura dos objetos da animação com as informações de posicionamento fornecidas pelo módulo de controle da animação. Existem diversos pacotes, de domínio público, capazes de realizar de forma satisfatória o *rendering* dos quadros de uma animação. Dentre eles podemos citar o pacote **SIPP** (*SImple Polygon Processor*) [**Yngvesson (1994)**], que é um pacote para síntese de imagens baseado em um algoritmo de *scan-line Z-buffer*, além do **POV-Ray**, que será utilizado para a visualização dos exemplos apresentados neste curso.

Finalmente, temos o **mecanismo de controle da animação** que é, de fato, o responsável pela movimentação dos atores, além da coordenação da interação entre os mesmos e entre estes e o animador. O controle do movimento-interação dos atores é, sem dúvida, o maior desafio dentro da área da animação por computador. Embora existam técnicas padronizadas para a síntese de objetos (b-rep, CSG, etc.) e para a geração de imagens (algoritmos de *scan-line*, *ray tracing*, etc.), não existe um algoritmo genérico para o tratamento da movimentação-interação de objetos em uma animação. De fato, existem apenas casos de estudo onde uma determinada técnica de controle é aplicada a um certo número de modelos.

Este é o escopo deste curso, um estudo detalhado sobre as técnicas de controle de movimento-interação mais comuns em animação modelada por computador. Especificamente, serão abordadas técnicas relacionadas à animação baseada em procedimentos, ou **procedimental**, pois estas constituem excelentes ferramentas de desenvolvimento para um grande número de casos de estudo.

1.4 Abordagem das estratégias de controle em animação

A filosofia que orientou o desenvolvimento de grande parte dos sistemas de animação modelada por computador desenvolvidos até hoje orientou-se pela utilização de conceitos cinemáticos ou dinâmicos para a descrição do movimento. Isto ocorreu devido ao fato de que, durante o período de desenvolvimento destes sistemas, a maior parte dos "cientistas

da animação" havia abordado o problema de controle do movimento de objetos através da utilização do cálculo diferencial e das leis físicas, tais como as Leis Dinâmicas de Newton e de Euler. De fato, sem a aplicação das leis físicas que regem os movimentos dos objetos dificilmente seríamos capazes de simular os movimentos de qualquer objeto.

Assim a pesquisa por ferramentas de controle para animações (convencionais ou simulações) baseadas em técnicas cinemáticas e dinâmicas constitui uma vasta área de investigação científica neste ramo da ciência da computação. Dessa forma, dedicaremos grande parte deste trabalho à apresentação de ferramentas de controle para animações, conforme os capítulos 2 a 7.

Entretanto, a complexa natureza de diversos sistemas físicos nos leva a lidar com sistemas de equações diferenciais cada vez mais complexos. Mais ainda, não existe uma solução global que possa ser aplicada à animação de qualquer objeto. Ou seja, uma solução conveniente para determinado sistema pode não trazer bons resultados quando aplicada a um outro sistema.

Como alternativa para a solução do problema de controle do fluxo de uma animação apresentamos a divisão do modelo que controla os movimentos numa animação em diversas partes, denominadas **Blocos de Controle Local** e **Bloco de Controle Global**, [**Camargo (1994)**] e [**Camargo (1995)**]. Imaginemos um sistema a ser simulado composto por diversos "atores". Podemos atribuir a cada ator um bloco de controle local, que conterá as "regras de comportamento" de cada um. Por exemplo, as leis físicas que regem o movimento de um único ator estão embutidas em seu respectivo módulo de controle local. A natureza deste tipo de bloco sugere uma ferramenta matemática (tal como cálculo diferencial) para que seja definido o modelo de um ator e seu comportamento próprio.

Uma vez que o sistema sugerido é composto por muitos atores, podem ocorrer interações entre atores e o animador. O bloco de controle global sugere uma abordagem lógica para a determinação de um modelo para estas interações. Por exemplo, caso ocorra um determinado evento, quais atores serão afetados ? Como um determinado ator deve responder ? Dessa forma, dependendo daquilo que está ocorrendo no sistema, os atores se comportam de uma dada forma, que poderia ser diferente caso outro evento houvesse ocorrido. Isto é tratado no capítulo 8.

O capítulo 9 dedica-se à apresentação de três sistemas de animação para ilustrar os conceitos estudados ao longo deste tutorial.

Capítulo 2

ANIMAÇÃO CINEMÁTICA - MODELO PARA UM PÊNDULO SIMPLES

Para introduzirmos uma metodologia para o desenvolvimento de animações modeladas por computador recorreremos, inicialmente, a um modelo cinemático relativamente simples: o modelo de movimento de um pêndulo simples. Trata-se de um modelo cuja implementação é relativamente fácil mas que, contudo, é muito didático e possui um resultado visual bastante agradável.

Consideremos a Figura 2.1.

Para a Figura 2.1 temos que:

θ é o ângulo que o pêndulo descreve em relação à normal.

l é o comprimento do fio que sustenta a esfera do pêndulo. Supõe-se que este fio seja rígido, inextensível e de massa desprezível.

m é a massa da esfera.

$\vec{g}$ é a aceleração da gravidade local, que possui módulo "g".

Para este exemplo, a equação que descreve o movimento é dada por:

$$\frac{d^2\theta}{dt^2} = -\frac{g}{l} \cdot sen(\theta) \tag{2.1}$$

Devemos notar que a equação 2.1 é não-linear e, portanto, adotamos:

$$sen(\theta) \approx \theta$$

para pequenas oscilações.

Neste caso, a solução para a equação (2.1) é:

$$\theta = k \cdot cos(\omega \cdot t + \beta) \tag{2.2}$$

onde:

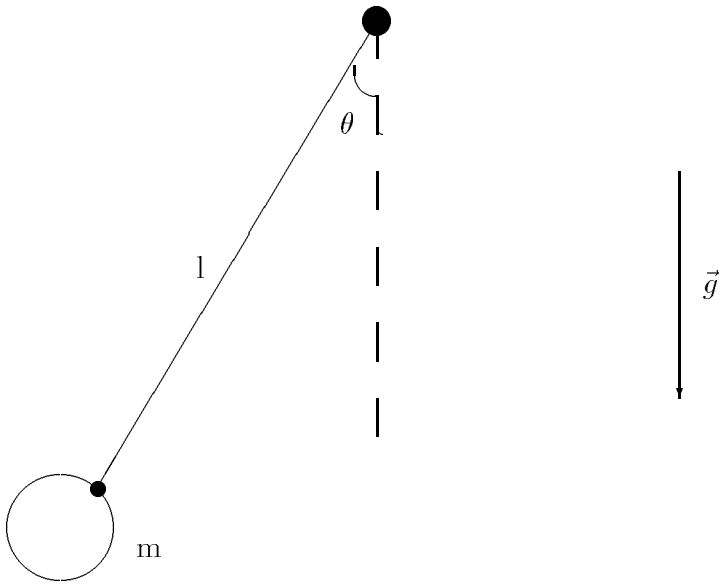

Figura 2.1: Esquematização de um pêndulo simples.

$\omega = (g/l)^{1/2}$ é a frequência angular do pêndulo (constante).

"k" é a amplitude inicial do movimento (constante).

"β" é a "fase" do movimento (constante).

2.1 Gerando a animação do pêndulo

Uma vez que o modelo de movimento do pêndulo tenha sido definido, a próxima etapa consiste na geração da animação propriamente dita. Para tanto, serão necessárias as seguintes ferramentas:

- Um programa para a geração de uma trajetória para o pêndulo, com base no modelo acima apresentado. Tal programa deverá gerar um conjunto de arquivos que serão interpretados pela ferramenta de *rendering*, sendo que cada arquivo conterá a descrição de um ponto da trajetória do pêndulo.

- Um arquivo que contenha a descrição (modelo) da animação, que possa ser interpretado pela ferramenta de *rendering*. Este arquivo deve conter a descrição de cada objeto que participa da cena (geometria, textura, etc.).

- Uma ferramenta (programa) para o *rendering* de cada quadro da animação. Esta ferramenta deverá ser capaz de interpretar o arquivo que contém o modelo da animação, além dos arquivos que descrevem a trajetória do pêndulo.
- Um arquivo de processamento em lote, para a automação do processo de síntese dos quadros.

De acordo com esta filosofia, este exemplo foi implementado em um microcomputador padrão PC-486. Foram implementados/utilizados os seguintes programas/arquivos:

- **PENDULO.C / PENDULO.EXE:**
  Este programa realiza o cálculo da trajetória do pêndulo (θ) em função de seu comprimento (l), da gravidade local ($\vec{g}$), da amplitude inicial do movimento (k) e de sua fase (β). Além disso, este programa gera um conjunto de arquivos com o nome **FRAME*.H** onde se encontram as informações relativas à trajetória descrita pelo pêndulo.
- **FRAME0.H / FRAME1.H / FRAME2.H / ... :**
  Armazenam, respectivamente, a posição do pêndulo para o primeiro quadro, para o segundo quadro, ...
- **FRAME.H:**
  Contém a posição do pêndulo para o quadro que está sendo **atualmente** processado.
- **BASICFRM.POV:**
  Contém, de acordo com a linguagem de descrição de uma cena do pacote **POV-Ray**[1], os elementos que farão parte da cena: "peças fixas" (hastes e chão) e "peças móveis" (esfera e fio).
- **GERANIM.C / GERANIM.EXE:**
  Gera um arquivo denominado ANIMACAO.BAT que descreve a sequência de quadros a serem sintetizados, além de conter informações sobre a qualidade das imagens a serem geradas.
- **ANIMACAO.BAT:**
  Arquivo para processamento em lote que executa a síntese dos quadros da animação.

A Figura 2.2 esquematiza o relacionamento existente entre os arquivos que participam do processo de animação.

Para melhor ilustrar este exemplo são apresentados a seguir alguns dos arquivos utilizados para a animação do pêndulo:

O ARQUIVO ANIMACAO.BAT (para n+1 quadros):

[1]O pacote POV-Ray é um *renderer* (*free-ware*) capaz de sintetizar uma cena através de um algoritmo de *ray tracing*.

```
copy frame0.h frame.h
povray nd320.def -ibasicfrm.pov -oframe000.tga
del frame.h

copy frame1.h frame.h
povray nd320.def -ibasicfrm.pov -oframe001.tga
del frame.h

     .
     .
     .

copy framen.h frame.h
povray nd320.def -ibasicfrm.pov -oframe003.tga
del frame.h
```

O ARQUIVO FRAME.H (define a posição atual do fio e da esfera):

```
#declare l= 3.000
#declare ex= 1.368
#declare ey= 1.241
#declare ez= 0.000
#declare fix= 0.000
#declare fiy= 5.000
#declare fiz= 0.000
#declare ffx= 1.026
#declare ffy= 2.181
#declare ffz= 0.000
```

O ARQUIVO BASICFRM.POV:

```
#include "colors.inc"      // CONTEM AS CORES DO POVRAY
#include "textures.inc"    // CONTES AS TEXTURAS DO POVRAY
#include "shapes.inc"      // CONTEM AS FORMAS GEOMETRICAS DO POVRAY

#include "frame.h"        // CONTEM AS POSICOES DA ESFERA E DO FIO DE
                          // SUSTENTACAO NO QUADRO EM PROCESSAMENTO

camera {
        location <2,4,-8> // LOCALIZACAO DA CAMERA
        look_at <0,3,0>   // PONTO DE OBSERVACAO
        }

light_source {<4,7,-1> color White}    // FONTE DE LUZ
```

```
light_source {<-4,7,-1> color White}   // FONTE DE LUZ

background {color SkyBlue}             // COR DE FUNDO

plane {                                // PLANO DE SUSTENTACAO DO PENDULO
        <0,1,0>,0
        pigment{checker
                        color  Red
                        color  Yellow
                }
        }

// CONSTRUCAO DA ESFERA

sphere {                  // ESFERA DO PENDULO
        <ex, ey, ez>, 1   // INFORMACAO CONTIDA EM FRAME.H
        texture{Glass}
        }

// CONSTRUCAO DO "FIO"

cylinder {                                          // FIO DO PENDULO
          <fix, fiy, fiz>, <ffx, ffy, ffz>, .05  // INFORMACAO CONTIDA EM
                                                    // FRAME.H
          pigment{color Silver}
          finish{Metal}
          }

// CONSTRUCAO DA BASE
// TODOS OS ELEMENTOS A SEGUIR SAO FIXOS

cylinder {
          <0, l+2, -2>, <-4, 0, -2>, .2
          pigment{color Silver}
          finish{Metal}
          }

cylinder {
          <0, l+2., -2>, <4, 0, -2>, .2
          pigment{color Silver}
          finish{Metal}
         }

sphere {
        <0, l+2., -2>, .2
```

```
        pigment{color Silver}
        finish{Metal}
        }

cylinder {
          <0, 1+2, 2>, <-4, 0, 2>, .2
          pigment{color Silver}
          finish{Metal}
          }

cylinder {
          <0, 1+2., 2>, <4, 0, 2>, .2
          pigment{color Silver}
          finish{Metal}
         }

sphere {
        <0, 1+2., 2>, .2
        pigment{color Silver}
        finish{Metal}
        }

cylinder {
          <0, 1+2., -2>, <0, 1+2., 2>, .2
          pigment{color Silver}
          finish{Metal}
         }
```

Finalmente, a Figura 2.3 apresenta um dos quadros gerados nesta animação.

Como podemos notar, através deste exemplo, a implementação de uma animação deste tipo baseada em um modelo cinemático constitui uma tarefa simples, visto que apenas um pequeno número de variáveis, tais como posição, orientação, velocidade e aceleração deve ser controlado. Entretanto, os resultados obtidos são, na maioria das vezes, completamente satisfatórios.

Por outro lado, quando o objetivo de uma animação é a **simulação de um processo físico**, então um modelo mais sofisticado deve ser utilizado para que o resultado final apresente um aspecto "mais realista". Dessa forma, no próximo capítulo será apresentado um modelo dinâmico para o movimento de objetos rígidos, de maneira a obter-se "animações realistas", ou seja, simulações.

Mais adiante abordaremos novamente o tema da animação cinemática, porém direcionado ao estudo do movimento de objetos articulados.

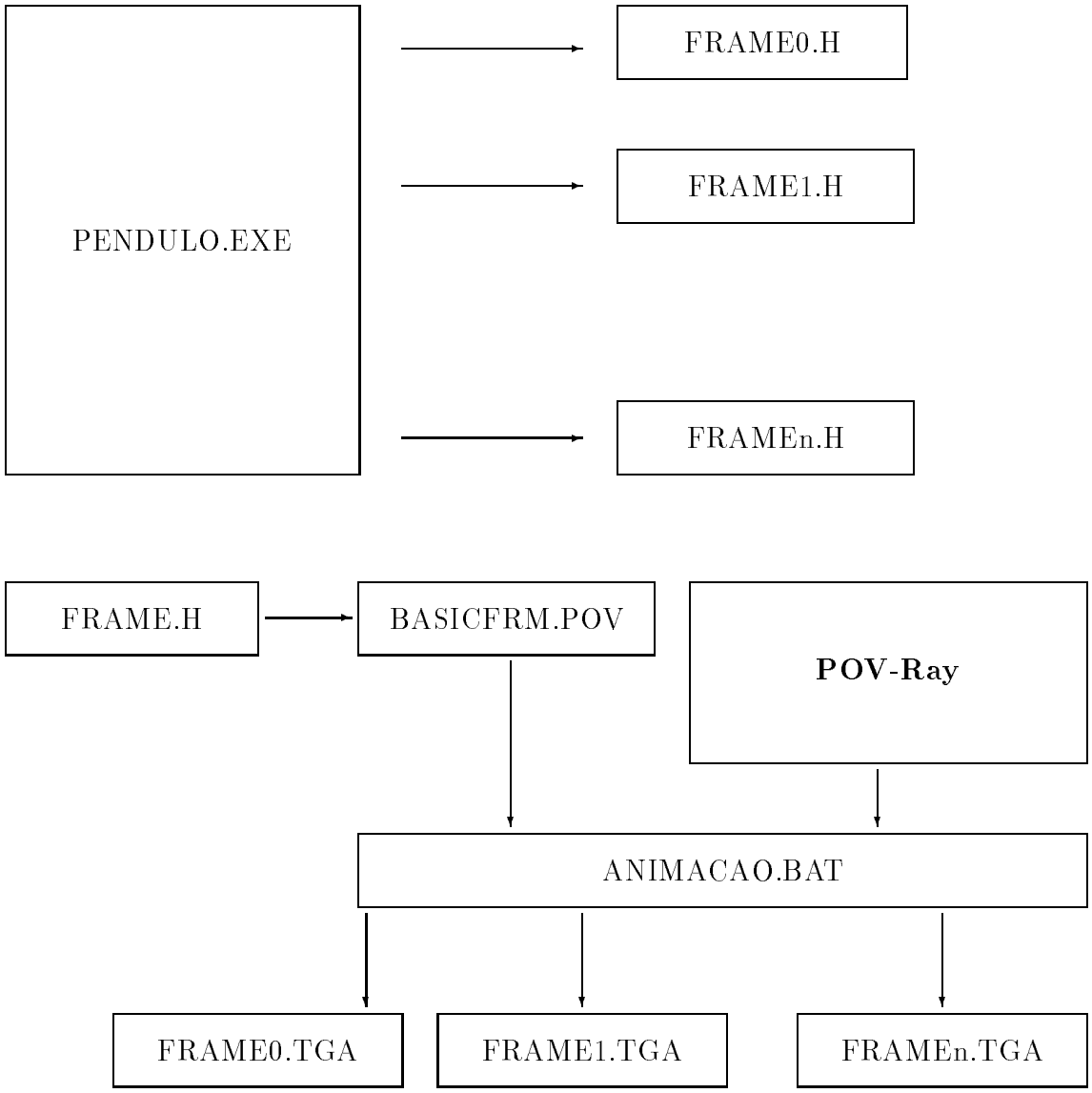

Figura 2.2: Relacionamento entre os arquivos utilizados para a síntese da animação do pêndulo.

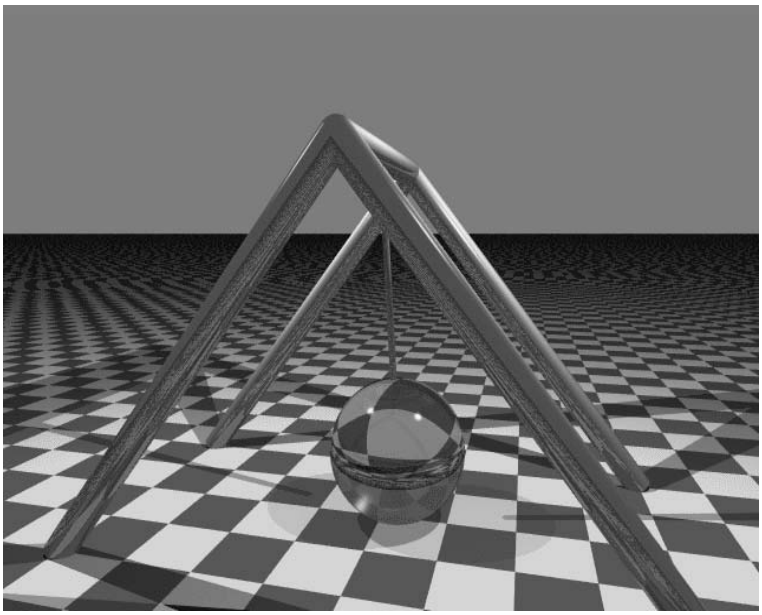

Figura 2.3: Imagem gerada durante o processo de animação do pêndulo.

Capítulo 3

ANIMAÇÃO DINÂMICA - MODELO PARA OBJETOS RÍGIDOS

O modelo dinâmico para a animação de objetos rígidos é complementar ao modelo cinemático apresentado anteriormente. Isto significa que o modelo dinâmico apresenta variáveis que haviam sido desprezadas pelo modelo cinemático. Tais variáveis são, por exemplo, as forças e torques aplicados ao objeto, o momento de inércia do objeto, etc.

A utilização de um modelo mais completo (dinâmico, por exemplo) apresenta algumas vantagens, tais como um maior "grau de realismo" do movimento gerado e, também, a possibilidade de simulação de um grande número de fenômenos físicos. Por outro lado, tais modelos também apresentam desvantagens tais como maior grau de complexidade, maior número de variáveis a serem controladas, etc. Além disso, certamente o animador deve dominar alguns conhecimentos de mecânica e, mais ainda, o resultado visual nem sempre compensará o alto custo computacional destes modelos. Portanto, a escolha entre a utilização de um modelo cinemático ou dinâmico deve ser muito criteriosa.

Dentro desse contexto, para que seja traçado um paralelo entre modelos cinemáticos e dinâmicos apresentaremos na subseção seguinte um exemplo de implementação de um modelo dinâmico relativamente simples.

3.1 Desenvolvimento de um modelo dinâmico para um amortecedor

Consideremos o esquema de um amortecedor apresentado na Figura 3.1.

Trata-se de um modelo relativamente simples onde só é possível o deslocamento do êmbolo do pistão ao longo de seu eixo de simetria. Neste sistema temos um componente capaz de armazenar energia (mola), além de um componente dissipador de energia (atrito viscoso entre o êmbolo e o corpo do amortecedor), que são capazes de gerar um movimento amortecido para o êmbolo do objeto. Dessa forma, visto que não existe movimento de rotação neste sistema, seu modelo dinâmico pode ser dado por:

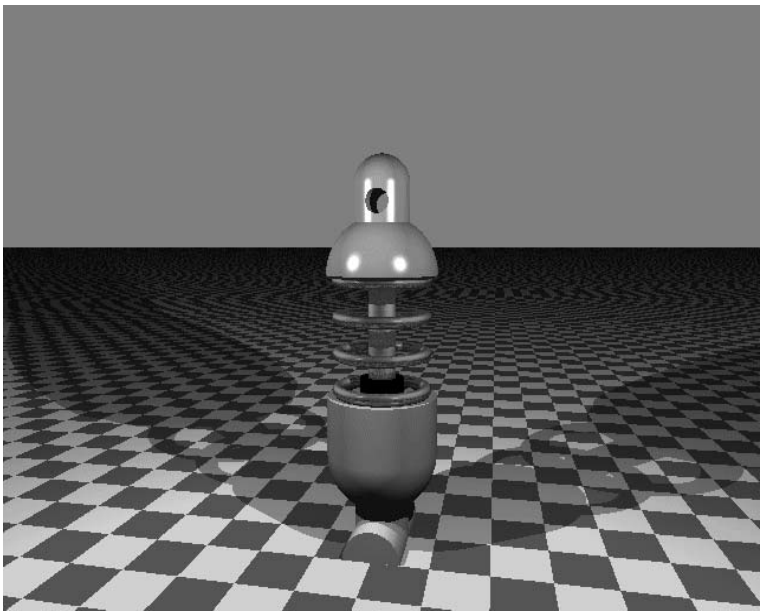

Figura 3.1: Amortecedor.

$$F_{resultante} = F_{externa} + F_{mola} + F_{atrito\_viscoso} \tag{3.1}$$

ou seja, a força resultante sobre o objeto é igual à soma das forças que atuam no mesmo.

Como forças atuantes sobre o objeto temos a força externa que deve provocar o deslocamento do êmbolo ($F_{externa}$), a força de restauração da mola (F_{mola}) e a força decorrente do atrito viscoso entre o êmbolo e o corpo do amortecedor ($F_{atrito\_viscoso}$).

Dessa forma, o modelo para o amortecedor pode ser re-escrito como sendo:

$$m \cdot \frac{d^2x}{dt^2} = F_{externa} - k \cdot x - b \cdot \frac{dx}{dt} \tag{3.2}$$

onde:

x é a posição do objeto em deslocamento.

m é a massa em movimento.

k é a constante de elasticidade da mola.

b é o coeficiente de atrito viscoso.

Discretizando o modelo consideraremos:

$$\frac{dx}{dt} = \dot{x}_i = \frac{x_i - x_{i-1}}{\Delta t} \tag{3.3}$$

$$\frac{d^2x}{dt^2} = \frac{\dot{x}_i - \dot{x}_{i-1}}{\Delta t} = \frac{x_i - 2 \cdot x_{i-1} + x_{i-2}}{(\Delta t)^2} \tag{3.4}$$

onde Δt equivale ao intervalo de amostragem.
Substituindo na equação (3.2):

$$m \cdot \frac{x_i - 2 \cdot x_{i-1} + x_{i-2}}{(\Delta t)^2} = F_{externa} - k \cdot x_i - b \cdot \frac{x_i - x_{i-1}}{\Delta t} \tag{3.5}$$

Finalmente:

$$x_i = \frac{(\Delta t)^2 \cdot F_{externa} + (b \cdot \Delta t + 2 \cdot m) \cdot x_{i-1} - m \cdot x_{i-2}}{m + k \cdot (\Delta t)^2 + b \cdot \Delta t} \tag{3.6}$$

Dessa forma, de acordo o modelo representado na equação (3.6) podemos avaliar a posição do êmbolo do amortecedor em função das forças que atuam no corpo. Especificamente, para uma determinada força externa aplicada ao objeto, podemos avaliar seu respectivo posicionamento ao longo do tempo. A partir deste ponto, uma vez que o modelo para o movimento do objeto tenha sido desenvolvido, o mesmo procedimento utilizado para a geração final da animação do exemplo cinemático pode ser utilizado aqui.

Um ponto importante a ser novamente considerado ainda neste exemplo é o fato de que o mesmo não dispunha de movimento de rotação. Tal fato contribuiu em muito para a obtenção deste modelo relativamente simples. Para sistemas onde é possível a ocorrência de movimentos de rotação, deve ser utilizada uma formulação mais complexa, tal como as equações dinâmicas de Newton-Euler, que é descrita na seção seguinte.

3.2 Modelo para objetos sujeitos a rotações

Quando dispomos de modelo onde é permitida a ocorrência de movimentos de rotação devemos considerar, além das forças que atuam no sistema, os torques associados à rotação do objeto. Para tanto, podemos considerar o **Modelo Dinâmico de Newton-Euler**, descrito através do seguinte conjunto de equações:

$$F = m \cdot [a + \dot{\omega} \times P_c + \omega \times (\omega \times P_c)] \tag{3.7}$$

$$N = I \cdot \dot{\omega} + (m \cdot P_c \times a) + \omega \times (I \cdot \omega) \tag{3.8}$$

onde:

m é a massa do objeto.

P_c é um vetor que localiza o centro de massa do objeto em relação ao sistema de coordenadas utilizado.

F representa a soma das forças que atuam no centro de massa do objeto.

N representa a soma dos torques que atuam no centro de massa do objeto.

I é a matriz **tensor de inércia** do objeto, relativa ao seu centro de massa.

a é a aceleração linear do objeto.

ω é a velocidade angular do objeto.

$\dot{\omega}$ é a aceleração angular do objeto.

Embora o modelo acima seja suficiente para a descrição do movimento de translação e rotação de um objeto rígido, voltaremos a estudar os modelos dinâmicos mais adiante, quando direcionarmos nossa atenção para uma outra classe de objetos rígidos, especificamente aqueles que são dotados de articulações. Um grande número de objetos interessantes são dotados de articulações, entre eles diversas partes de nosso corpo tais como mãos, braços, pernas, etc.

Entretanto, antes de introduzirmos os conceitos relacionados à animação de objetos articulados, é interessante desenvolver um conjunto de convenções de modo a facilitar a modelagem destes objetos. Portanto, o capìtulo seguinte introduz um conjunto de convenções úteis à modelagem de objetos articulados.

Capítulo 4

REPRESENTAÇÃO DE OBJETOS NO ESPAÇO TRIDIMENSIONAL

Antes de partirmos para o desenvolvimento das técnicas de controle de movimento de estruturas articuladas é interessante definir uma convenção para a representação dos objetos de nosso interesse. As definições encontradas neste capítulo são derivadas das convenções propostas em **[Craig (1989)]**, inicialmente voltadas à área de Robótica. Consideremos a Figura 4.1:

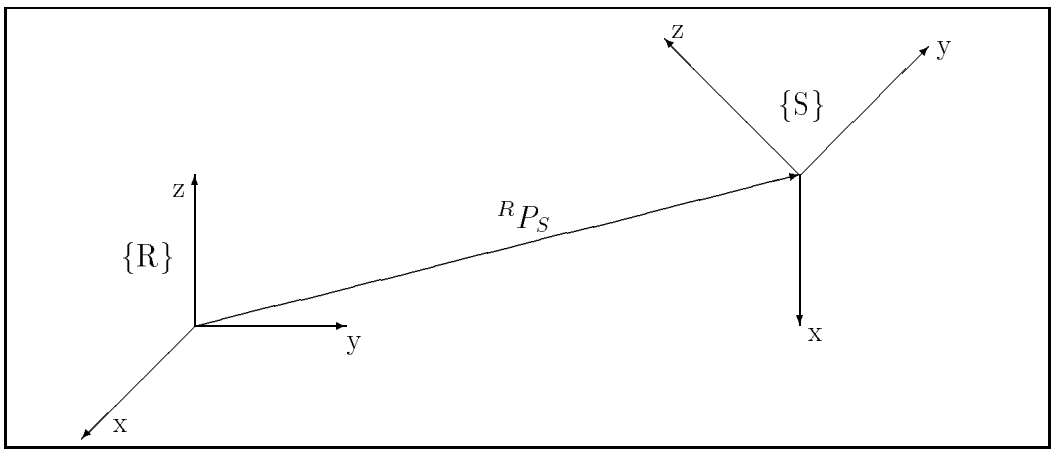

Figura 4.1: Esquema de representação de sistemas de coordenadas no espaço 3-D.

Seja:

${}^{R}V_{3\times1}$ um vetor qualquer, representado no sistema de coordenadas {R}.
${}^{S}V_{3\times1}$ um vetor qualquer, representado no sistema de coordenadas {S}.
${}^{R}_{S}T_{4\times4}$ uma transformação de coordenadas, que leva um vetor representado no sistema de coordenadas {S} para o sistema de coordenadas {R}.

Para estabelecermos uma relação entre vetores e transformações de coordenadas, e melhor descrever as translações, vamos considerar, quando realizarmos o produto entre uma matriz de transformação e um vetor, o vetor expresso na forma homogênea:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

sendo "x", "y" e "z" as coordenadas do vetor em seu devido sistema de coordenadas[1]. Dessa forma:

$$^{R}V = {}^{R}_{S}T \cdot {}^{S}V \tag{4.1}$$

A matriz de transformação ${}^{R}_{S}T$ entre os SC pode ser expressa na forma:

$${}^{R}_{S}T_{(4\times4)} = \begin{bmatrix} {}^{R}_{S}R_{(3\times3)} & {}^{R}P_{S(3\times1)} \\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{4.2}$$

onde:

${}^{R}P_{S(3\times1)}$ é um vetor, representado no SC {R}, que vai da origem do SC {R} à origem do SC {S}.

${}^{R}_{S}R_{(3\times3)}$ é uma matriz que representa a orientação do SC {S} em relação ao SC {R}.

Note que ${}^{R}_{S}R$ é formada por três "vetores-coluna", ortonormais entre si, representados no SC {R}, que constituem uma base para o SC {S}. Dessa forma, podemos escrever que:

$${}^{S}_{R}R = \left[{}^{R}_{S}R\right]^{-1} = \left[{}^{R}_{S}R\right]^{T} \tag{4.3}$$

Temos ainda que:

$$^{R}V = {}^{R}_{S}R \cdot {}^{S}V + {}^{R}P_{S} \tag{4.4}$$

A transformação ${}^{R}_{S}T$ é inversível e pode ser representada na forma:

$${}^{S}_{R}T = \left[{}^{R}_{S}T\right]^{-1} = \begin{bmatrix} \left[{}^{R}_{S}R\right]^{T} & -\left[{}^{R}_{S}T\right]^{T} \cdot {}^{R}P_{S} \\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{4.5}$$

Assim sendo:

$$^{S}V = {}^{S}_{R}T \cdot {}^{R}V \tag{4.6}$$

$$^{S}V = \left[{}^{R}_{S}R\right]^{T} \cdot {}^{R}V - {}^{R}_{S}R \cdot {}^{R}P_{S} \tag{4.7}$$

Podemos concatenar diversas transformações como produto entre matrizes:

$${}^{A}_{D}T = {}^{A}_{B}T \cdot {}^{B}_{C}T \cdot {}^{C}_{D}T \tag{4.8}$$

[1] É comum, em robótica, a substituição do termo "Sistema de Coordenadas" por outros, tais como "frame" ou, simplesmente, SC.

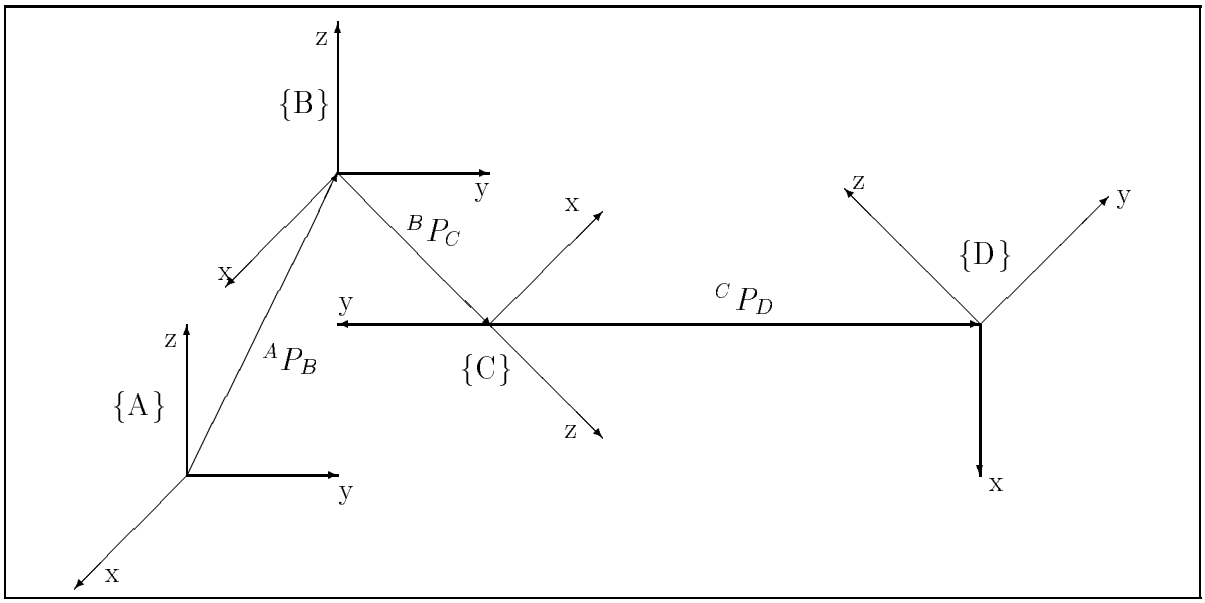

Figura 4.2: Concatenando diversos sistemas de coordenadas.

É importante lembrar que o vetor ^{S}V denota tanto a sua forma homogênea (4x1) como a sua forma convencional (3x1). No decorrer do texto, a forma homogênea será utilizada no cálculo das transformações entre sistemas de coordenadas, conforme a equação (4.1), enquanto a forma convencional, no cálculo da orientação do vetor, conforme a equação (4.4).

4.1 Obtenção de uma matriz de orientação entre sistemas de coordenadas

Consideremos dois SC {A} e {B}, inicialmente coincidentes, sendo {A} o SC referencial e {B} o SC que deve ser rotacionado. É possível a representação de uma matriz de orientação através da "notação de rotação de Euler", onde a rotação será representada em relação ao próprio SC que está girando. Dessa forma, consideremos a seguinte sequência de passos:

- Primeiro, gire o SC {B} em torno de seu próprio eixo "x" de um ângulo "α",
- em segundo lugar, gire o SC {B} em torno de seu próprio eixo "y" de um ângulo "β" e,
- finalmente, gire o SC {B} em torno de seu próprio eixo "z" de um ângulo "γ".

Formalmente temos:

$$ {}^{A}_{B}R_{xyz}(\alpha,\beta,\gamma) = R_x(\alpha)\cdot R_y(\beta)\cdot R_z(\gamma) = \tag{4.9} $$

$$ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sen(\alpha) \\ 0 & sen(\alpha) & cos(\alpha) \end{bmatrix} \cdot \begin{bmatrix} cos(\beta) & 0 & sen(\beta) \\ 0 & 1 & 0 \\ -sen(\beta) & 0 & cos(\beta) \end{bmatrix} \cdot \begin{bmatrix} cos(\gamma) & -sen(\gamma) & 0 \\ sen(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.10} $$

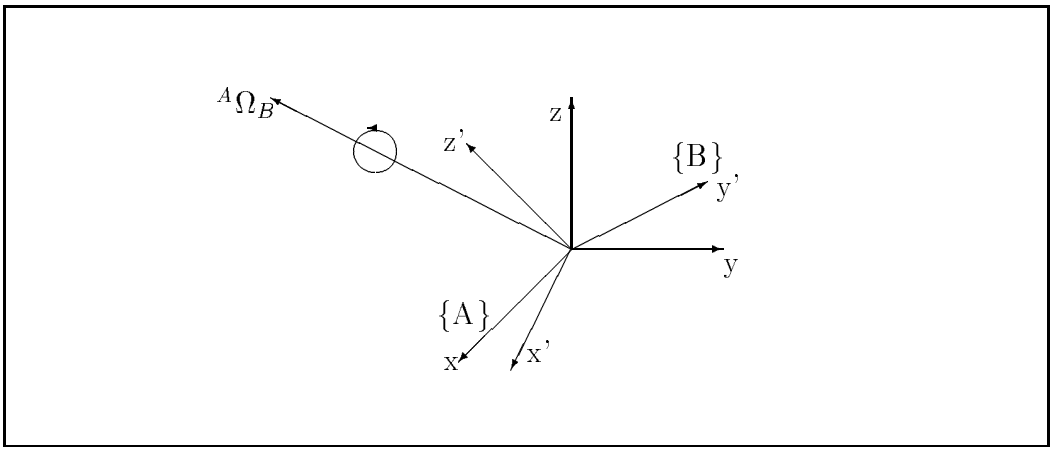

Figura 4.3: Rotação entre sistemas de coordenadas.

Podemos também representar a orientação entre dois SC através da rotação de um SC em relação a um vetor, por intermédio dos "parâmetros de Euler", conforme mostra a Figura 4.3. De acordo com esta figura, o SC {B} pode ser rotacionado de um ângulo qualquer, digamos θ, ao redor do vetor Ω, de acordo com a "regra da mão direita", na forma:

$$\begin{aligned} \varepsilon_1 &= \Omega_x \cdot sen(\theta/2) \\ \varepsilon_2 &= \Omega_y \cdot sen(\theta/2) \\ \varepsilon_3 &= \Omega_z \cdot sen(\theta/2) \\ \varepsilon_4 &= cos(\theta/2) \end{aligned} \tag{4.11}$$

Sendo que Ω_x, Ω_y e Ω_z são as componentes do vetor $^A\Omega_B$, a matriz de orientação será dada por:

$${}^A_B R = \begin{bmatrix} (1-2\cdot\varepsilon_2^2-2\cdot\varepsilon_3^2) & 2\cdot(\varepsilon_1\cdot\varepsilon_2-\varepsilon_3\cdot\varepsilon_4) & 2\cdot(\varepsilon_1\cdot\varepsilon_3-\varepsilon_2\cdot\varepsilon_4) \\ 2\cdot(\varepsilon_1\cdot\varepsilon_2-\varepsilon_3\cdot\varepsilon_4) & (1-2\cdot\varepsilon_1^2-2\cdot\varepsilon_3^2) & 2\cdot(\varepsilon_2\cdot\varepsilon_3-\varepsilon_1\cdot\varepsilon_4) \\ 2\cdot(\varepsilon_1\cdot\varepsilon_3-\varepsilon_2\cdot\varepsilon_4) & 2\cdot(\varepsilon_2\cdot\varepsilon_3-\varepsilon_1\cdot\varepsilon_4) & (1-2\cdot\varepsilon_1^2-2\cdot\varepsilon_2^2) \end{bmatrix} \tag{4.12}$$

4.2 Notação para sistemas de coordenadas variantes no tempo

Consideremos a Figura 4.4, onde são apresentados dois SC {A} e {B}. Consideremos também o ponto Q, representado no SC {B} (^BQ).

Caso o ponto ^BQ esteja em movimento, seu vetor velocidade será dado por:

$$^BV_Q = \frac{d(\,^BQ)}{dt} \tag{4.13}$$

Caso desejássemos expressar a velocidade do ponto "Q" em termos do SC {A}, deveríamos fazer:

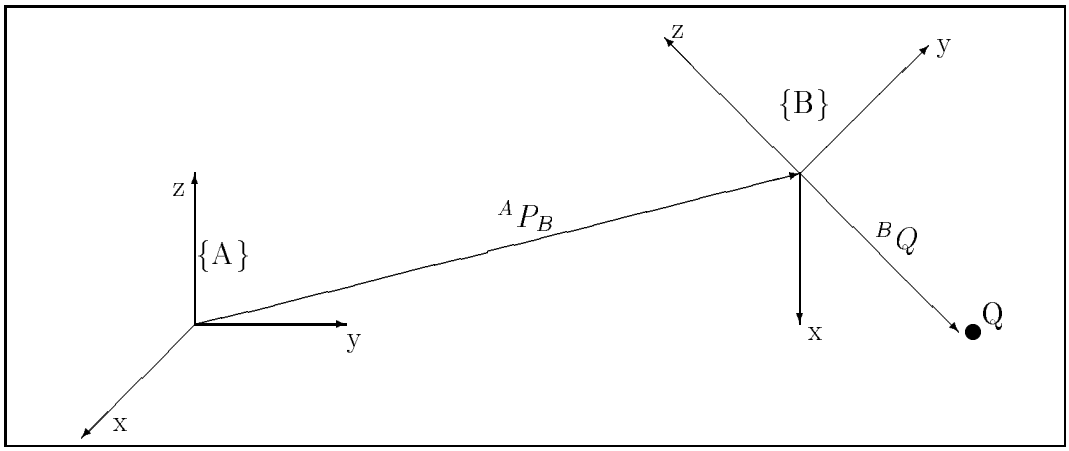

Figura 4.4: Representação para S.C. variantes no tempo.

$$ {}^{A}\left[{}^{B}V_Q\right] = {}^{A}_{B}R \cdot {}^{B}V_Q \tag{4.14} $$

Por outro lado, se existe variação na distância entre as origens dos SC {A} e {B} (os SC estão em movimento), então devemos considerar um novo termo:

$$ {}^{A}\left[{}^{B}V_Q\right] = {}^{A}V_{BORG} + {}^{A}_{B}R \cdot {}^{B}V_Q \tag{4.15} $$

sendo ${}^{A}V_{BORG}$ a velocidade entre as origens dos SC {A} e {B}, medida no SC {A}.

Devemos notar que, para que a equação (4.15) seja válida, não é permitida a ocorrência de alterações na orientação entre os SC {A} e {B}. Ou seja, ${}^{A}_{B}R = constante$.

Consideremos agora que os SC {A} e {B} possuem origens coincidentes e que esta condição não se altera com o passar do tempo. Dessa forma: ${}^{A}P_B = 0 = constante$, como mostra a Figura 4.3.

Devemos também notar que, embora as origens permaneçam coincidentes, existe movimento entre os SC {A} e {B}, visto que existe uma velocidade angular ${}^{A}\Omega_B$ entre os mesmos, dada em termos do SC {A}.

A velocidade angular descreve a variação da orientação do SC {B} relativa ao SC {A}. A direção do vetor velocidade angular indica o eixo de rotação instantâneo de {B} relativo a {A}; a magnitude do vetor velocidade angular indica a velocidade da rotação.

Consideremos novamente a Figura 4.4. Caso exista velocidade linear (${}^{A}V_{BORG}$) e velocidade angular (${}^{A}\Omega_B$) entre os SC {A} e {B} a velocidade do ponto "Q" será dada por:

$$ {}^{A}V_B = {}^{A}V_{BORG} + {}^{A}_{B}R \cdot {}^{B}V_Q + \left[{}^{A}\Omega_B \times {}^{A}_{B}R \cdot {}^{B}Q\right] \tag{4.16} $$

sendo:

${}^{A}\Omega_B \times {}^{A}_{B}R \cdot {}^{B}Q$ a componente de rotação; caso o SC {B} gire em relação ao SC {A}, representado no SC {A};

${}^{A}_{B}R \cdot {}^{B}V_Q$ a velocidade do ponto "Q" em relação ao S.C. {B}, representada no SC {A} e

${}^{A}V_{BORG}$ a velocidade da origem do SC {B} em relação ao SC {A}.

Vamos chamar o Sistema de Coordenadas Inercial de {U}. Quando desejarmos representar a velocidade angular de um SC {A} qualquer em relação a {U}, utilizaremos a notação ω_A.

Caso dois SC estejam se movendo, um em relação ao outro, é possível que exista contínua alteração na matriz de orientação que relaciona os dois sistemas. Para analisarmos esta situação, consideremos o SC {A} em movimento, em relação ao SC inercial, {U}.

Neste caso temos:

$$\frac{d\left[{}^{U}_{A}R\right]}{dt} = [\omega_A]^* \cdot \left[{}^{U}_{A}R\right] \tag{4.17}$$

onde:

$$[\omega_A]^* = \begin{bmatrix} 0 & \omega_z & -\omega_y \\ -\omega_z & 0 & \omega_x \\ \omega_y & -\omega_x & 0 \end{bmatrix} \tag{4.18}$$

é denominada a matriz DUAL do vetor:

$$\omega_A = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

Uma vez que tenhamos definido uma convenção para a notação a ser utilizada na modelagem de estruturas articuladas, podemos proceder à modelagem de objetos desta classe. Nos capítulos seguintes definiremos metodologias para a representação e simulação de objetos articulados no espaço tridimensional. Especificamente, o capìtulo seguinte apresenta um modelo cinemático para objetos articulados.

Capítulo 5

MODELO CINEMÁTICO PARA ESTRUTURAS ARTICULADAS

"Manipulador" é um termo comum em robótica, utilizado para designar estruturas rígidas dotadas de articulações como, por exemplo, o braço de um robô. Em **[Korein (1982)]** temos apresentada uma primeira aplicação do modelo cinemático de um manipulador em Computação Gráfica.

Um problema bastante comum em robótica consiste em se determinar as posições angulares das juntas de um robô de tal forma que sua extremidade (*end-effector*) seja posicionada em um determinado ponto de interesse do espaço. Ou seja, dada uma trajetória para o *end-effector* do robô quais as posições angulares das juntas que satisfazem esta condição ? Este é o mesmo problema encontrado quando se deseja realizar a animação de um objeto articulado qualquer, para o qual devem ser consideradas as seguintes questões:

1. Por quais pontos do espaço 3-D deve se deslocar a extremidade do objeto articulado (definição da trajetória) ?

2. Qual a orientação a ser seguida pelo *end-effector* do objeto ?

3. Para a trajetória desejada, quais as posições e orientações dos demais segmentos articulados do robô ?

4. Qual deve ser o deslocamento angular de cada junta para que o movimento final desejado seja efetivamente realizado ?

Para solucionarmos estas questões podemos desenvolver, por exemplo, um modelo cinemático para o objeto articulado. Dessa forma, a seção seguinte introduz uma metodologia para a representação geométrica de uma estrutura articulada qualquer.

5.1 Parâmetros de um objeto articulado segundo Denavit-Hartenberg

Conforme apresentado em **[Girard (1985)]**, considere os seguintes parâmetros de uma estrutura articulada apresentados na Figura 5.1:

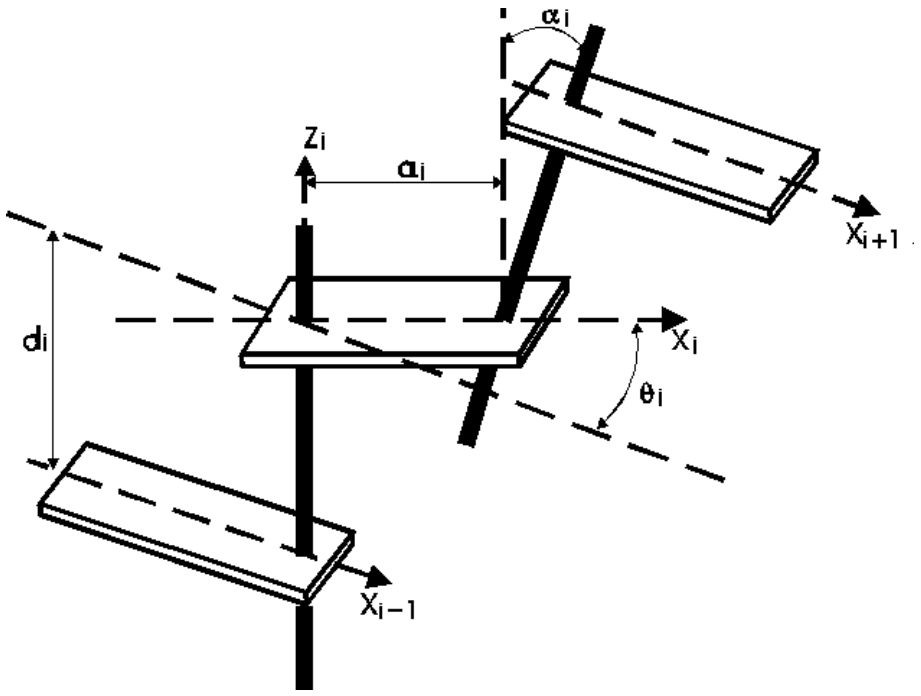

Figura 5.1: Parâmetros de Denavit-Hartenberg para uma estrutura articulada.

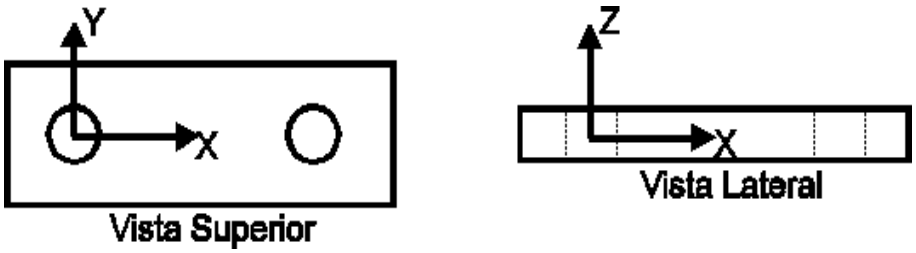

Figura 5.2: Sistema de Coordenadas de um segmento.

a_i é o comprimento do segmento "i" (medido ao longo do eixo "x" do SC "preso" ao segmento "i").

α_i é o ângulo entre os eixos "z" dos SC dos segmentos "i" e "i+1".

θ_i é o ângulo entre os eixos "x" dos SC dos segmentos "i-1" e "i" (também conhecido como **variável de junta**).

d_i é a distância entre os segmentos "i-1" e "i".

Como podemos notar na Figura 5.1, para melhor representar as características de cada segmento, associamos a cada segmento do manipulador um SC próprio (e fixo) a este, conforme mostra a Figura 5.2:

Se associarmos a cada segmento do objeto articulado um SC, então poderemos definir também transformações que relacionam tais SC.

Logo, seja a transformação entre dois segmentos adjacentes, "i-1" e "i":

$$ {}^{i-1}_{i}T = \begin{bmatrix} \vec{n}_i & \vec{o}_i & \vec{a}_i & \vec{p}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1} $$

onde:

$\vec{n}_i$, $\vec{o}_i$, $\vec{a}_i$, são vetores-coluna que definem a orientação do segmento,

$\vec{p}_i$ equivale ao vetor-coluna de posição, que vai da origem do SC {i-1} até a origem do SC {i}.

e ainda:

$$ \begin{array}{cccc} n_{ix} = cos(\theta_i) & o_{ix} = -cos(\alpha_i) \cdot sen(\theta_i) & a_{ix} = sen(\alpha_i) \cdot sen(\theta_i) & p_{ix} = a_i \cdot cos(\theta_i) \\ n_{iy} = sen(\theta_i) & o_{iy} = cos(\alpha_i) \cdot cos(\theta_i) & a_{iy} = -sen(\alpha_i) \cdot cos(\theta_i) & p_{iy} = a_i \cdot sen(\theta_i) \\ n_{iz} = 0 & o_{iz} = sen(\alpha_i) & a_{iz} = cos(\alpha_i) & p_{iz} = d_i \end{array} \tag{5.2} $$

Concatenando as transformações entre quaisquer segmentos teremos:

$$ {}^{j}_{i}T = {}^{j}_{j+1}T \cdot {}^{j+1}_{j+2}T \cdot ... \cdot {}^{i-1}_{i}T \tag{5.3} $$

Notemos que, devido à sua estrutura não deformável, quando o objeto sofrer algum deslocamento, apenas o parâmetro "θ" sofrerá variação.

5.2 Geração de trajetórias para uma estrutura articulada

Consideremos a estrutura articulada representada na Figura 5.3.

Para a Figura 5.3, consideremos completamente conhecida sua configuração inicial; ou seja, todos os parâmetros do objeto são conhecidos ($\alpha_i = 0$ e $d_i = 0$), inclusive as **variáveis de junta**: θ_1 , θ_2 , θ_3 e θ_4.

Com relação à configuração final do objeto, ou seja, o estado do objeto após sua movimentação, consideremos conhecidas a **posição** e a **orientação** do último SC (fixo ao último segmento) em relação ao SC inercial {0}.

Dessa forma, sendo conhecidas a **posição** e a **orientação final** do último segmento (associado ao SC {4}) em relação ao SC inercial, {0}, podemos definir a transformação final que relaciona os respectivos SC.

$$ {}^{0}_{4}T = \begin{bmatrix} \vec{n}^f & \vec{o}^f & \vec{a}^f & \vec{p}^f \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4} $$

Analisando cuidadosamente a equação (5.4), podemos concluir que, embora a matriz de transformação se encontre numericamente definida, cada um de seus elementos está relacionado a uma ou mais variáveis de junta.

Através das informações acima, podemos propor o seguinte problema cinemático inverso:

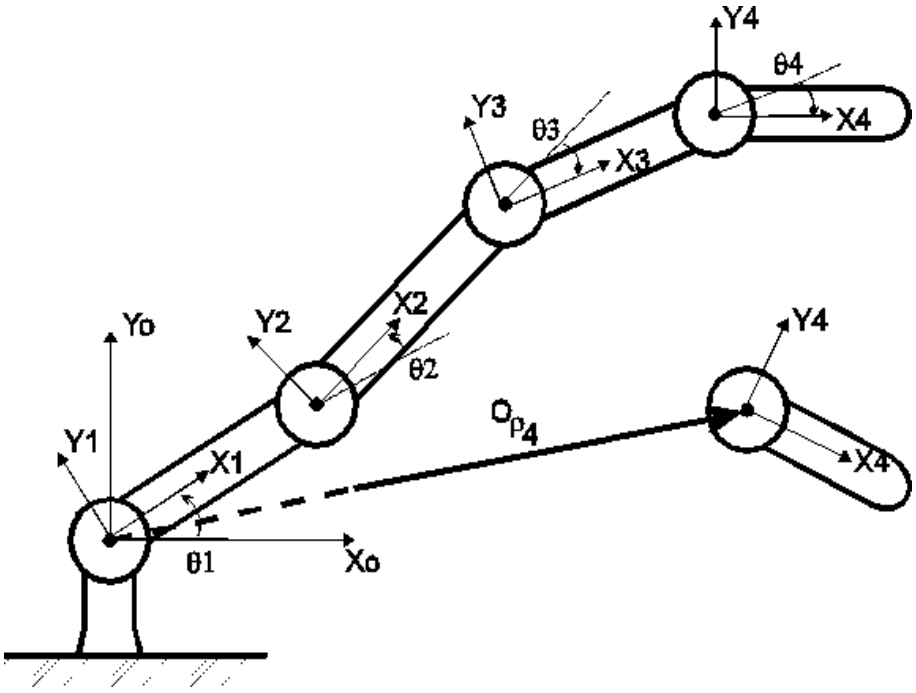

Figura 5.3: Posições inicial e final de uma estrutura articulada.

Como determinar os valores para as variáveis de junta (θ_1 , θ_2 , θ_3 e θ_4) que definem a configuração final do objeto ?

Para a solução deste tipo de problema propomos um procedimento iterativo, onde os valores das variáveis de junta são constantemente atualizados até que se chegue à solução do problema.

Partindo-se de uma estimativa inicial qualquer, podemos definir para o objeto em questão a transformação relativa à "k-ésima" iteração que relaciona o último SC, 4, com o SC inercial, $\{0\}$, como sendo:

$$ {}^{0}_{4}T^{k} = \begin{bmatrix} \vec{n}^k & \vec{o}^k & \vec{a}^k & \vec{p}^k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.5} $$

Sendo o **vetor de variáveis de junta** dado por:

$$ q^k = (\theta_1^k, \theta_2^k, \theta_3^k, \theta_4^k)^T \tag{5.6} $$

Podemos ainda definir uma relação entre as transformações ${}^{0}_{4}T^{final}$ e ${}^{0}_{4}T^{k}$ na forma:

$$ \begin{array}{rcl} r_1 & = & \vec{n}^k \cdot (\vec{p}^f - \vec{p}^k) \\ r_2 & = & \vec{o}^k \cdot (\vec{p}^f - \vec{p}^k) \\ r_3 & = & \vec{a}^k \cdot (\vec{p}^f - \vec{p}^k) \\ r_4 & = & (\vec{a}^k \cdot \vec{o}^f - \vec{a}^f \cdot \vec{o}^k)/2 \\ r_4 & = & (\vec{n}^k \cdot \vec{a}^f - \vec{n}^f \cdot \vec{a}^k)/2 \\ r_4 & = & (\vec{o}^k \cdot \vec{n}^f - \vec{o}^f \cdot \vec{n}^k)/2 \end{array} \tag{5.7} $$

Sendo:

$$r(q) = (r_1, r_2, r_3, r_4, r_5, r_6) \tag{5.8}$$

conhecido como sendo o **vetor residual** da "k-ésima" iteração.

Quando $r(q) = 0$, o manipulador estará na posição e orientação final desejadas.

Segundo **[Goldenberg (1985)]**, para chegarmos a $r(q) = 0$, o "vetor q" pode ser atualizado da seguinte forma:

$$q^{(k+1)} = q^{(k)} + \delta^{(k)} \tag{5.9}$$

sendo $\delta^{(k)} = [\delta_1, \delta_2, ..., \delta_n]^T$ a solução para o sistema linear:

$$r_j[q^{(k)}] + \sum_{i=1}^{n} J_{ji}^{(k)} \cdot \delta^{(k)} = 0, \qquad j = 1, ..., 6 \tag{5.10}$$

e J_{ji} são os elementos da matriz Jacobiano, "J", para um objeto com "n" **graus de liberdade**, que é dada por:

$$J_{ji}^{(k)} = \frac{\delta r_j^{(k)}}{\delta \theta_i^{(k)}} \tag{5.11}$$

Notemos que, ao invés de associar a uma junta um grau de liberdade, é associado um SC a cada grau de liberdade.

A solução deste problema geralmente depende da inversão da matriz Jacobiano. Entretanto, nem sempre o Jacobiano será uma matriz quadrada, o que o tornará não inversível. Neste caso, para que ocorra convergência no processo iterativo de cálculo do vetor de variáveis de estado, deve ser cálculada a pseudo-inversa do Jacobiano, "J^+", ao invés de sua inversa. Para uma matriz Jacobiano de dimensão "$m \times n$" e posto (*rank*) "p", a sua pseudo-inversa será:

$$J^+ = \begin{array}{ll} (J^T \cdot J)^{-1} \cdot J, & se : m > n = p \\ J^T \cdot (J \cdot J^T)^{-1}, & se : p = m < n \end{array} \tag{5.12}$$

Uma alternativa para a resolução da equação (5.9) é a utilização do "Método de Newton-Raphson Modificado", como segue:

$$q^{(k+1)} = q^{(k)} + \lambda^{(k)} \cdot \delta^{(k)} \tag{5.13}$$

onde $\lambda^{(k)}$ é um número positivo escolhido de tal forma que a seguinte inequação seja satisfeita:

$$G[q^{(k+1)}] < G[q^{(k)}] \tag{5.14}$$

onde:

$$G(q) = \sum_{j=1} 6[r_j(q)]^2 \tag{5.15}$$

Conforme mostrado na Figura 5.4, a partir da definição de um ponto da trajetória desejada, aplicamos o "algoritmo" acima descrito (equação (5.9) ou (5.13)) para que sejam calculadas as variáveis de junta do objeto neste ponto.

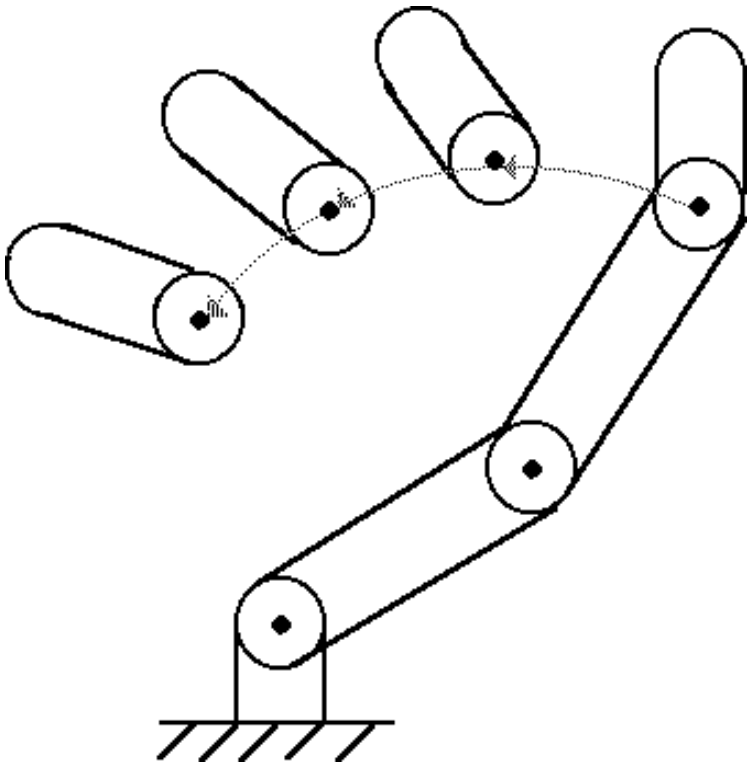

Figura 5.4: Marcação da trajetória do objeto.

Assim, definimos para cada ponto da trajetória do objeto a posição e a orientação do extremo livre[1] do objeto articulado.

5.3 Exemplo de animação de um robô modelo "Yasukawa Motoman L-3"

Consideremos o robô apresentado na Figura 5.5.

Neste exemplo, o robô constitui uma estrutura articulada com cinco graus de liberdade. Matematicamente, a relação existente entre cada um dos graus de liberdade pode ser expressa através do seguinte conjunto de matrizes de transformação:

$$ {}^{0}_{1}T = \begin{bmatrix} cos(\theta_1) & -sen(\theta_1) & 0 & 0 \\ sen(\theta_1) & cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.16} $$

$$ {}^{1}_{2}T = \begin{bmatrix} cos(\theta_2) & -sen(\theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -sen(\theta_2) & -cos(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.17} $$

[1]Consideremos o extremo livre de um objeto articulado como sendo o segmento mais extremo do mesmo, capaz de realizar uma determinada tarefa.

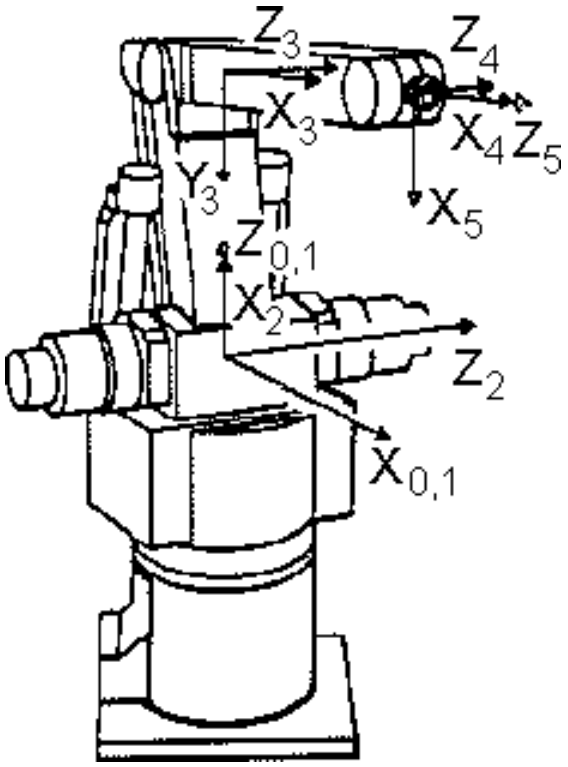

Figura 5.5: Esquema de um robô modelo Yasukawa Motoman L-3.

$$ {}^{2}_{3}T = \begin{bmatrix} cos(\theta_3) & -sen(\theta_3) & 0 & l_1 \\ sen(\theta_3) & cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.18} $$

$$ {}^{3}_{4}T = \begin{bmatrix} cos(\theta_4) & -sen(\theta_4) & 0 & l_2 \\ sen(\theta_4) & cos(\theta_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.19} $$

$$ {}^{4}_{5}T = \begin{bmatrix} cos(\theta_5) & -sen(\theta_5) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ sen(\theta_5) & cos(\theta_5) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.20} $$

onde:

θ_1 , θ_2 , θ_3 , θ_4 e θ_5 representam os graus de liberdade do objeto e

l_1 e l_2 representam os comprimentos do primeiro e segundo segmentos, respectivamente.

A partir destas informações podemos desenvolver a animação do objeto para uma determinada trilha a ser seguida pelo extremo livre do mesmo. A Figura 5.6 apresenta uma possível visualização para o objeto a ser animado.

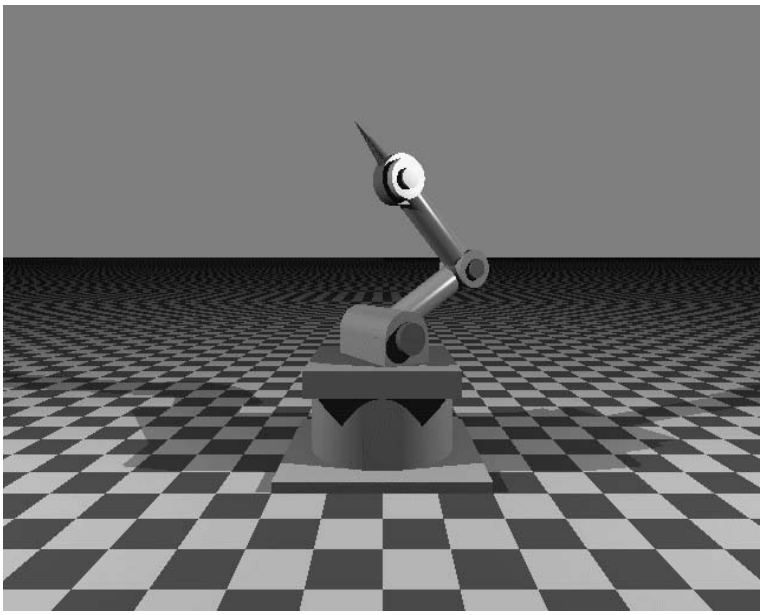

Figura 5.6: Representação visual do robô modelo Yasukawa Motoman L-3.

5.4 Exemplo de animação de uma luminária "Luxo"

Alterando ligeiramente o modelo do robô Yasukawa Motoman L-3 podemos construir um outro objeto articulado interessante, a luminária "Luxo". O modelo matemático para este objeto pode ser descrito através das seguintes transformações:

$$ {}^{0}_{1}T = \begin{bmatrix} cos(\theta_1) & -sen(\theta_1) & 0 & p_x \\ sen(\theta_1) & cos(\theta_1) & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.21} $$

$$ {}^{1}_{2}T = \begin{bmatrix} cos(\theta_2) & -sen(\theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -sen(\theta_2) & -cos(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.22} $$

$$ {}^{2}_{3}T = \begin{bmatrix} cos(\theta_3) & -sen(\theta_3) & 0 & l_1 \\ sen(\theta_3) & cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.23} $$

$$ {}^{3}_{4}T = \begin{bmatrix} cos(\theta_4) & -sen(\theta_4) & 0 & l_2 \\ sen(\theta_4) & cos(\theta_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.24} $$

onde:

θ_1 , θ_2 , θ_3 e θ_4 representam os graus de liberdade rotacionais do objeto.

p_x , p_y e p_z representam os graus de liberdade translacionais do objeto.

l_1 e l_2 representam os comprimentos do primeiro e segundo segmentos, respectivamente.

Novamente, com este modelo podemos desenvolver uma animação para uma determinada trilha planejada. A Figura 5.7 apresenta uma visualização para a luminária "Luxo".

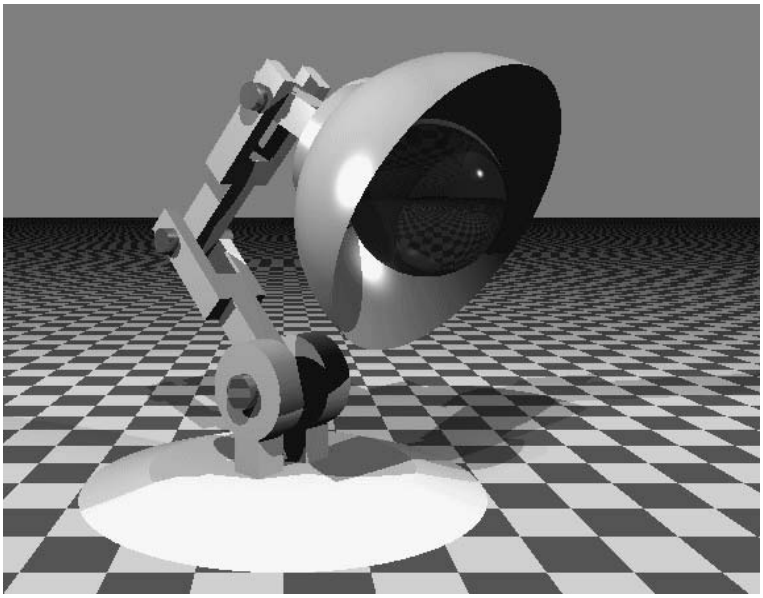

Figura 5.7: Representação visual para a luminária Luxo.

O modelo cinemático apresentado neste capítulo pode ser complementado através de um modelo dinâmico para estruturas articuladas, o qual será apresentado no capítulo seguinte.

Capítulo 6

MODELO DINÂMICO PARA ESTRUTURAS ARTICULADAS RAMIFICADAS

No capítulo anterior foram tratadas estruturas articuladas que são caracterizadas por uma **cadeia de segmentos**. Entretanto, uma grande variedade de objetos articulados admite **cadeias ramificadas**, caracterizando estruturas tipo **árvore**. Um exemplo bastante comum seria o esqueleto humano. [**Armstrong (1985)**] foi um dos pioneiros a tratar este tipo de problema aplicado à Computação Gráfica. Ainda na segunda metade da década de 80, foram propostas soluções alternativas de forma mais genérica em [**Isaacs (1987)**] e [**Isaacs (1988)**]. O modelo dinâmico apresentado nesta seção foi desenvolvido por [**Wilhems (1988)**] de modo a ser aplicável a estruturas articuladas ramificadas ou não. Este tipo de abordagem para a solução do problema de controle de movimento de uma estrutura tipo árvore nos leva a um algoritmo iterativo que pode ser implementado em computador de maneira relativamente fácil.

Consideremos a estrutura articulada em árvore apresentada na Figura 6.1.

Uma estrutura tipo árvore consiste de um objeto articulado cujas juntas podem conectar mais de dois segmentos. Da mesma forma que o manipulador estudado no capítulo anterior, associamos a cada segmento um SC que ficará fixo no seu respectivo segmento.

A **raiz** da estrutura será escolhida arbitrariamente e será considerada uma referência para o objeto. As **folhas** da árvore são os segmentos extremos da estrutura.

Temos que, para dois segmentos adjacentes, será chamado de **pai** aquele que estiver mais próximo da raiz, sendo o segundo chamado de **filho**. Deve ficar claro desde já que um segmento não pode ter dois pais.

Para a análise das estruturas tipo árvore podemos utilizar a formulação dinâmica de Newton-Euler, conforme apresentado em [**Wilhelms (1988)**] e [**Craig (1989)**].

6.1 Formulação dinâmica para estrutura em árvore

Consideremos o objeto rígido da Figura 6.2. As equações vetoriais de Newton-Euler que descrevem o movimento deste objeto no espaço tridimensional são [**Wilhelms (1988)**]:

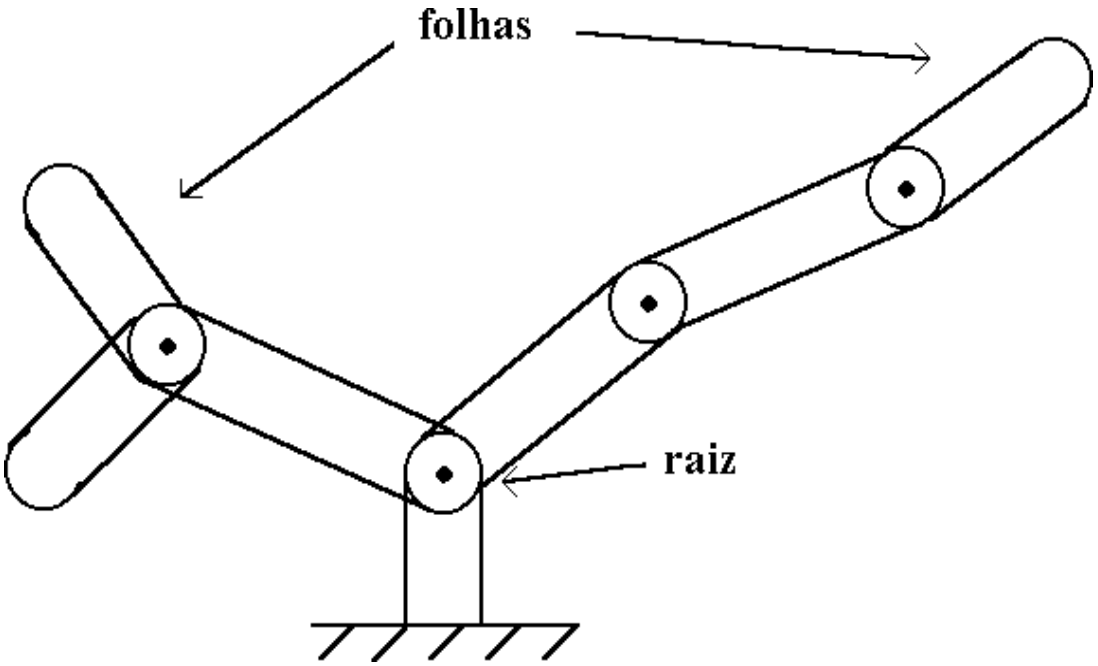

Figura 6.1: Exemplo de estrutura tipo árvore.

$$F = m \cdot [a + \dot{\omega} \times P_c + \omega \times (\omega \times P_c)] \tag{6.1}$$

$$N = I \cdot \dot{\omega} + m \cdot P_c \times a + \omega \times (I \cdot \omega) \tag{6.2}$$

onde:

m representa a massa do objeto.

P_c é um vetor que vai da origem do SC fixo no objeto até o centro de massa do mesmo.

F representa a soma das forças que atuam no centro de massa do objeto.

N representa a soma dos torques que atuam no centro de massa do objeto.

I é a matriz tensor de inércia (3×3) do objeto relativa ao seu centro de massa.

a é o vetor aceleração linear do corpo.

ω é o vetor velocidade angular do corpo.

$\dot{\omega}$ é o vetor aceleração angular do objeto.

Sendo que, por conveniência, os valores acima devem ser medidos em relação a um sistema de coordenadas fixo no objeto.

Em um objeto articulado, para que dois segmentos adjacentes (pai e filho) sejam mantidos unidos através de uma junta, devemos considerar um conjunto de **equações de restrição** além das equações de Newton-Euler para cada conjunto pai-filho:

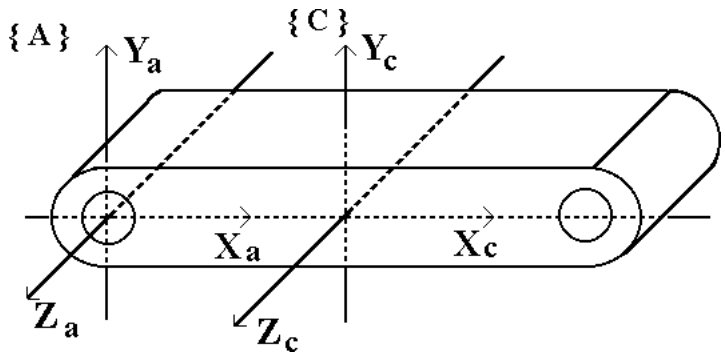

Figura 6.2: Um segmento de uma estrutura tipo árvore.

$$f_p = F_p + {}^p_f R \cdot f_f \tag{6.3}$$

$$n_p = N_p + {}^p_f R \cdot n_f + {}^p P_f \times ({}^p_f R \cdot f_f) \tag{6.4}$$

onde:

f_p é a força aplicada na origem do SC preso ao segmento "pai".

f_f é a força aplicada na origem do SC preso ao segmento "filho".

n_p é o torque aplicado na origem do SC preso ao segmento "pai".

n_f é o torque aplicado na origem do SC preso ao segmento "filho".

${}^p_f R$ é a matriz de orientação existente entre os segmentos "pai " e "filho ".

P_c é o vetor que vai da origem do SC preso ao segmento "pai" até a origem do SC preso ao segmento "filho".

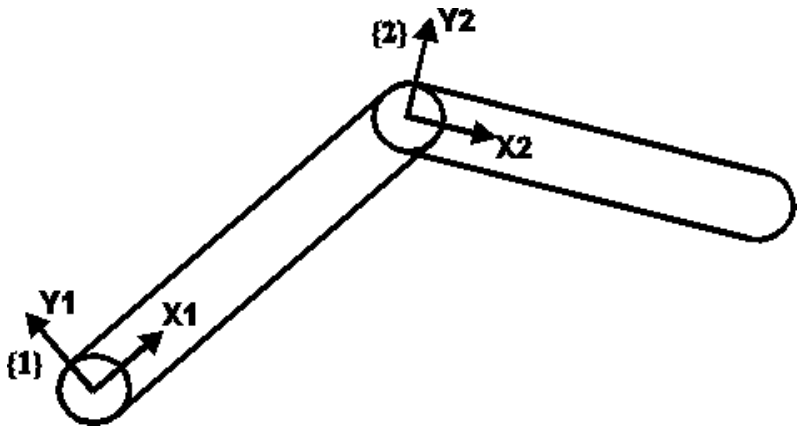

Figura 6.3: Dois segmentos unidos através de uma junta rotacional.

Dessa forma, para o objeto da Figura 6.3 temos:

$$\begin{array}{rcl} F_1 & = & m_1 \cdot [a_1 + \dot{\omega_1} \times P_{c1} + \omega_1 \times (\omega_1 \times P_{c1})] \\ N_1 & = & I_1 \cdot \dot{\omega_1} + m_1 \cdot P_{c1} \times a_1 + \omega_1 \times (I_1 \cdot \omega_1) \\ F_2 & = & m_2 \cdot [a_2 + \dot{\omega_2} \times P_{c2} + \omega_2 \times (\omega_2 \times P_{c2})] \\ N_2 & = & I_2 \cdot \dot{\omega_2} + m_2 \cdot P_{c2} \times a_2 + \omega_2 \times (I_2 \cdot \omega_2) \\ f_1 & = & F_1 + {}^1_2R \cdot f_2 \\ n_1 & = & N_1 + {}^1_2R \cdot n_2 + {}^1P_2 \times ({}^1_2R \cdot f_2) \end{array} \tag{6.5}$$

Supondo que as forças e torques externos são aplicados nas origens dos SC presos aos segmentos 1 e 2, e considerando a existência de apenas dois segmentos, podemos ainda escrever que:

$$\begin{array}{rcl} F_1 & = & f_1 - {}^1_2R \cdot f_2 \\ N_1 & = & n_1 - {}^1_2R \cdot n_2 - {}^1P_2 \times {}^1_2R \cdot f_2 \\ F_2 & = & f_2 \\ N_2 & = & n_2 \end{array} \tag{6.6}$$

Os conjuntos de equações anteriores podem ser reescritos na seguinte forma matricial:

$$\begin{bmatrix} a_1 \\ \dot{\omega_1} \\ a_2 \\ \dot{\omega_2} \end{bmatrix} = \begin{bmatrix} m_1 & -P_{c1}\times & 0 & 0 \\ m_1 \cdot P_{c1}\times & I_1 & 0 & 0 \\ 0 & 0 & m_2 & -P_{c2}\times \\ 0 & 0 & m_2 \cdot P_{c2}\times & I_2 \end{bmatrix}^{-1} \cdot$$

$$\begin{bmatrix} f_1 - {}^1_2R \cdot f_2 - \omega_1 \times (\omega_1 \times P_{c1}) \\ n_1 - {}^1_2R \cdot n_2 - {}^1P_2 \times {}^1_2R \cdot f_2 - \omega_1 \times (I_1 \cdot \omega_1) \\ f_2 - \omega_2 \times (\omega_2 \times P_{c2}) \\ n_2 - \omega_2 \times (I_2 \cdot \omega_2) \end{bmatrix} \tag{6.7}$$

Através da simulação da expressão matricial acima podemos determinar por completo o comportamento dinâmico do objeto articulado ao longo do tempo. Para tanto, devem ser conhecidas, num determinado instante de tempo, a posição, orientação e a velocidade (linear e angular) do objeto. Para determinarmos a posição, a orientação e velocidade do objeto no "próximo instante de tempo" devemos calcular a aceleração atual (linear e angular) e, através de integração numérica, determinar os próximos valores para as variáveis de posição, orientação e velocidade.

Logo, através do método de integração numérica de Euler, temos:

$$\begin{array}{rcl} \omega_i^{(k+1)} & = & \omega_i^{(k)} + \dot{\omega}_i^{(k)} \cdot \Delta t \\ v_i^{(k+1)} & = & v_i^{(k)} + a_i^{(k)} \cdot \Delta t \\ S_i^{(k+1)} & = & S_i^{(k)} + v_i^{(k)} \cdot \Delta t + 0,5 \cdot a_i^{(k)} \cdot (\Delta t)^2 \end{array} \tag{6.8}$$

sendo Δt o intervalo de amostragem.

Outro método de integração (mais eficiente) que pode ser utilizado é o método de Runge-Kutta [**Boyce (1985)**].

Dessa forma, a primeira matriz da equação (6.7) representa as "incógnitas" (acelerações) do objeto a ser simulado. Já a segunda matriz, que é constante e pode ser previamente calculada, representa a configuração do objeto sendo comumente chamada de **matriz de**

massa. Finalmente, a terceira matriz é composta pelas forças e torques que estão atuando no objeto.

Como um novo exemplo, consideremos Figura 6.4, que apresenta uma estrutura em árvore.

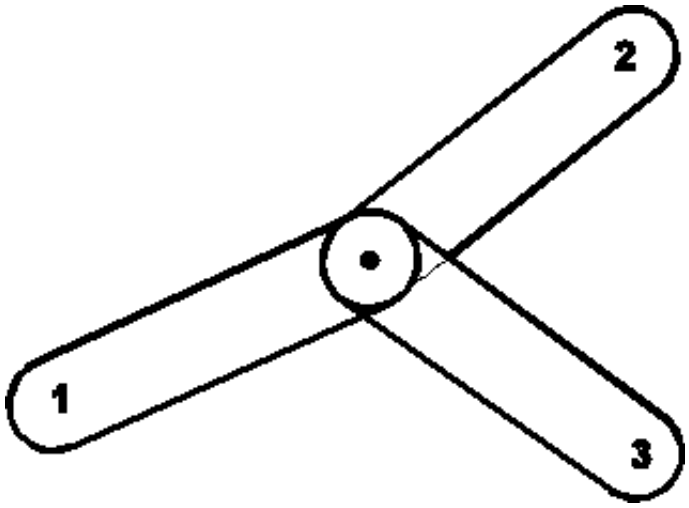

Figura 6.4: Três segmentos unidos por intermédio de uma junta rotacional.

As equações dinâmicas que modelam este objeto são:

$$\begin{aligned}
F_1 &= m_1 \cdot [a_1 + \dot{\omega_1} \times P_{c1} + \omega_1 \times (\omega_1 \times P_{c1})] \\
N_1 &= I_1 \cdot \dot{\omega_1} + m_1 \cdot P_{c1} \times a_1 + \omega_1 \times (I_1 \cdot \omega_1) \\
F_2 &= m_2 \cdot [a_2 + \dot{\omega_2} \times P_{c2} + \omega_2 \times (\omega_2 \times P_{c2})] \\
N_2 &= I_2 \cdot \dot{\omega_2} + m_2 \cdot P_{c2} \times a_2 + \omega_2 \times (I_2 \cdot \omega_2) \\
F_3 &= m_3 \cdot [a_3 + \dot{\omega_3} \times P_{c3} + \omega_3 \times (\omega_3 \times P_{c3})] \\
N_3 &= I_3 \cdot \dot{\omega_3} + m_3 \cdot P_{c3} \times a_3 + \omega_3 \times (I_3 \cdot \omega_3) \\
f_1 &= F_1 + {}^1_2R \cdot f_2 + {}^1_3R \cdot f_3 \\
n_1 &= N_1 + {}^1_2R \cdot n_2 + {}^1P_2 \times ({}^1_2R \cdot f_2) + {}^1_3R \cdot n_3 + {}^1P_3 \times ({}^1_3R \cdot f_3)
\end{aligned} \tag{6.9}$$

Mais uma vez, de posse do modelo que descreve o movimento do objeto, o processo de geração final da animação pode ser repetido como descrito nas seções anteriores.

Notemos que, para os algoritmos apresentados até o momento, com exceção deste último, a trilha a ser seguida pelo objeto deve ser completamente definida pelo animador. Isto pode, em certos casos, constituir uma tarefa enfadonha que, a princípio, poderia ser automatizada. Por exemplo, para um determinado objeto, dados os pontos inicial e final de seu movimento, o sistema de controle de movimento poderia automaticamente calcular a trilha a ser percorrida. Esta tarefa pode ser efetuada em diversos níveis, desde a simples consideração de uma reta como trajetória até o cálculo de caminhos otimizados e livres e livres de colisão.

Dentro desse contexto, o próximo capítulo explora um algoritmo para o planejamento e cálculo de trilhas. O algoritmo a ser apresentado é capaz de gerar caminhos livres de colisão para o deslocamento de um objeto.

Capítulo 7

PLANEJAMENTO DE TRILHAS / DESVIO DE OBSTÁCULOS

Modelos cinemáticos e dinâmicos são amplamente utilizados para o cálculo do movimento de qualquer objeto em animação por computador. Dessa forma, de acordo com grande parte das técnicas de controle em animação, o animador deve especificar "o quê" um ator deve fazer, por exemplo "mova-se daqui para lá" (restrição espacial), e "como" a ação deve ser realizada, por exemplo "mova-se o mais rápido possível mas não desperdice energia" (restrição temporal-energética). Mais ainda, o animador deve especificar as outras restrições presentes no sistema a ser modelado, tais como as leis físicas que regem o movimento, etc. A seguir, de acordo com as restrições impostas, a trilha para o movimento é gerada por alguma técnica de otimização.

Isto significa que o animador tem consciência da tarefa atribuída ao ator, mas a trilha a ser propriamente seguida pelo mesmo pode ser uma incógnita.

Analisando as restrições impostas ao movimento é possível extrair todas as informações sobre a trilha gerada, mas isto nem sempre é uma tarefa trivial. Mais ainda, a determinação de caminhos livres de colisões pode exigir um conjunto muito complexo de restrições em alguns casos.

Portanto, o paradigma acima descrito refere-se muito mais a "como se especificar as ações desejadas" do que a "como se planejar o movimento".

Para se solucionar o problema do planejamento do movimento de um único ator sugere-se, então, um mecanismo de planejamento prévio da trilha a ser seguida. Dessa forma, a avaliação da trilha a ser seguida pelo ator deve ser realizada antes da aplicação de algum modelo cinemático ou dinâmico ao movimento. Uma vez que uma trilha conveniente tenha sido calculada de acordo com algum conjunto de restrições, tais como "não se aproxime muito deste objeto" ou "encontre o caminho mínimo entre estes dois pontos", algum modelo cinemático ou dinâmico pode ser aplicado ao sistema para se gerar o movimento final do ator.

Esta estratégia fornece ao animador um maior controle da animação como um todo, visto que ele não apenas sabe o que deve ser feito, mas também é consciente de tudo aquilo que ocorre durante o movimento. Mais ainda, dividindo o modelo inicial em duas partes, ou seja, um planejamento prévio da trilha a ser seguida e posterior aplicação de um modelo fisicamente válido, tornamos o controle da animação como um todo mais fácil de ser

manipulado.

Existem desenvolvidos, atualmente, diversos métodos para a determinação de trilhas (*path planning*) de objetos rígidos evitando-se colisões (*collision avoidance*). E. G. Gilbert apresentou um método em [**Gilbert (1985)**] que utiliza funções de distância para o cálculo de uma trilha ótima; a idéia principal consiste em expressar o impedimento de colisões com outros obstáculos em termos das distâncias entre partes que apresentam grande probabilidade de colisão. O. Takahashi, por sua vez, apresentou um método para se evitar colisões entre figuras planas (polígonos) no espaço 2-D utilizando Diagramas de Voronoi [**Takahashi (1989)**]. L. P. Gewali considerou o problema de planejamento de trilhas no espaço 3-D na presença de obstáculos verticais [**Gewali (1990)**].

Um método muito interessante para o problema do impedimento de colisões ou planejamento de trilhas é apresentado por Y. K. Hwang em [**Hwang (1992)**]. Neste método o planejamento de trilhas no espaço 3-D é obtido através da representação de obstáculos por intermédio de campos potenciais.

Por outro lado, M. Moore em [**Moore (1988)**] preocupou-se em solucionar o problema da deteção em resposta a colisões em animação por computador.

O algoritmo para planejamento de trilhas que será apresentado a seguir é parcialmente baseado na estratégia de atribuição de "potenciais" aos obstáculos presentes numa região do espaço. Para se encontrar um conjunto de possíveis trilhas, de acordo com algum conjunto de restrições impostas ao ambiente onde se passa a animação, sugerimos a atribuição de "funções potenciais" aos obstáculos presentes no ambiente. Uma vez que o potencial para cada ponto tenha sido calculado, podemos obter um conjunto de trilhas possíveis para o movimento desejado. A seguir, dados os pontos inicial e final para o movimento, através de uma "função de otimização" podemos encontrar uma trilha ótima para o movimento.

7.1 Um caso estudado

O método aqui apresentado será discutido através de um exemplo relativamente simples. Para o exemplo em questão será apenas aplicado o algoritmo de planejamento de trilhas, visto que já foram apresentadas nos capítulos anteriores diversas técnicas cinemáticas e dinâmicas que podem ser aplicadas posteriormente ao cálculo da trilha do movimento.

7.1.1 Definição do modelo de campos de potenciais

O método a seguir pode ser aplicado para regiões no espaço 3-D ou 2-D. Consideremos, por exemplo, a região planar apresentada na Figura 7.1.

Como primeiro passo, a região planar deve ser decomposta em uma rede de pontos. A rede resultante será, portanto, composta por pontos dentro de "obstáculos" ou no "espaço livre". A seguir, a cada um dos pontos da rede devemos atribuir um determinado potencial. Cabe ressaltar que o cálculo de potenciais que se fará para cada ponto tem um pressuposto particular, a analogia com potenciais de campos elétricos.

Iniciaremos este procedimento atribuindo um potencial "P_i" a cada ponto no interior ou na borda de um obstáculo "i". Notemos que, atribuindo potenciais diferentes a cada

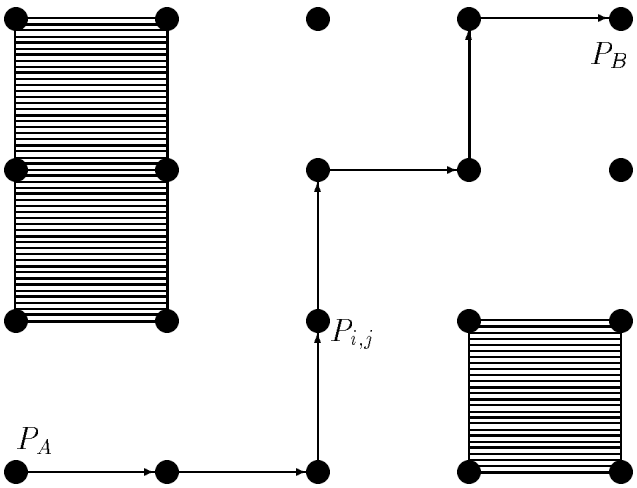

Figura 7.1: Uma região planar a ser estudada.

obstáculo podemos obter comportamentos diferentes para o movimento do ator, isto é, um potencial elevado fará com que o ator se distancie do obstáculo.

Podemos também considerar a fronteira que delimita a região planar como um obstáculo, atribuindo um potencial apropriado à mesma. Para os pontos no espaço livre inicialmente atribuiremos um potencial zero.

Visto que existem regiões com potenciais superiores em relação a outras, um campo de potenciais surgirá no espaço livre. Isto significa que deve-se atualizar os valores inicialmente atribuidos aos pontos no espaço livre.

Isto pode ser feito através do processo iterativo apresentado abaixo.

De acordo com a **Equação de Laplace**, em sua forma discretizada, apresentada em **[Hayt (1981)]**, é possível avaliar o potencial de um ponto "$P_{i,j}$" na região planar como sendo dado pela média dos potenciais de seus pontos vizinhos. Logo, para o exemplo apresentado na figura 7.1, temos:

$$P_{i,j} = \frac{P_{i-1,j} + P_{i,j-1} + P_{i,j+1} + P_{i+1,j}}{4} \tag{7.1}$$

O algoritmo para a avaliação do campo de potencial de uma região é dado por:

1. Divida a região em uma rede de pontos.

2. Para cada ponto nesta região:

   Se o ponto encontrar-se no interior ou na borda de um obstáculo, atribua um valor para seu potencial.

   Se o ponto encontrar-se no espaço livre, atribua a ele um potencial inicial igual a zero.

3. Repita, até que o potencial de cada ponto no espaço livre apresente convergência para algum valor:

   Para cada ponto no espaço livre, atualize seu potencial de acordo com a equação (7.1).

Até o momento, dispomos da topologia da região descrita através de um campo potencial. Isto significa que o problema inicial, isto é, "como evitar uma colisão", encontra-se representado pelo campo potencial presente no espaço livre. Portanto, sabemos que o ator estará se aproximando de um obstáculo quando o potencial dos pontos percorridos estiver aumentando. Ou seja, a probabilidade de colisão diminui à medida em que o potencial dos pontos percorridos também diminui.

Dessa forma, a trilha composta pelos pontos de menores potenciais certamente constitui a trilha com menor probabilidade de colisão. Entretanto, isto não implica que esta trilha é a mais curta entre dois pontos. Isto significa que necessitamos de algum critério/algoritmo de otimização de forma a descobrirmos "uma boa trilha" entre dois pontos de uma região.

A subseção seguinte discute esta etapa do problema, ou seja, "como encontrar uma boa trilha entre dois pontos mediante algum critério de otimização".

7.1.2 Definição de uma trilha "otimizada"

Para se avaliar uma trilha "ótima" entre dois pontos, vamos considerar uma regra básica em programação dinâmica. De acordo com a Figura 7.1, se a trilha desenhada entre os pontos A e B é ótima, então a "sub-trilha" entre os pontos $P_{i,j}$ e B também é.

Isto significa que, de acordo com o exemplo da Figura 7.1, se $P_{i,j}$ se encontra na trilha ótima, e é necessário passar por algum vizinho de $P_{i,j}$ para se chegar a este ponto, então este ponto vizinho também se encontra na trilha ótima.

Existem diversos algoritmos para se calcular uma trilha ótima entre dois pontos. Algoritmos de programação dinâmica, como aqueles apresentados em **[Gill (1981)]** e **[Gondran (1986)]**, são adequados ao método aqui desenvolvido porque já dispomos da região de nosso interesse representada através de uma rede.

Podemos considerar esta região como um grafo onde os "nós" do grafo são representados pelos pontos discretizados da região. Os "arcos" do grafo são representados pela conectividade da região e os "pesos de transição" entre "nós" são dados por algum critério de otimização.

O critério de otimização aqui sugerido é a soma ponderada do potencial do ponto com sua distância euclidiana até o ponto inicial da trilha. A função que representa o "custo" do ponto (i,j), $C_{i,j}$, pode ser expressa por:

$$C_{i,j} = \alpha \cdot V_{i,j} + \beta \cdot D_{i,j} \tag{7.2}$$

onde:

α é o "peso" atribuido ao potencial do ponto (i,j).

$V_{i,j}$ é o potencial do ponto (i,j).

β é o "peso" atribuido à distância.

$D_{i,j}$ é a distância euclidiana entre os pontos (i,j) e o **ponto inicial** da trilha.

Para aplicarmos o algoritmo de otimização a ser apresentado a seguir, associamos, a cada ponto (i,j) no espaço livre, os seguintes atributos:

- **Point_Number**: Número associado ao ponto.
- **Father_Number**: Número do "pai" deste ponto.
- **X:** Coordenada "x" do ponto.
- **Y:** Coordenada "y" do ponto.
- **Z:** Coordenada "z" do ponto.
- **C:** "Custo" deste ponto.
- **Mark:** Indica se este ponto foi marcado.

O algoritmo desenvolvido para a otimização de uma trilha é representado por:

1. Para cada ponto no espaço livre:
   defina Point_Number (≥ 1).
2. Defina os pontos inicial e final da trilha.
3. Selecione o ponto final, marque-o e faça-o ser o ponto atual. Faça Father_Number = 0 para o ponto final.
4. Para cada ponto (i,j), não marcado, no espaço livre vizinho ao ponto atual:
   calcule o custo $C_{i,j}$ do ponto.
5. Selecione o vizinho do ponto atual, que não esteja marcado, com menor custo. Faça seu Father_Number = Point_Number do ponto atual. Agora marque o ponto selecionado e faça-o ser o ponto atual.
6. O ponto atual é o ponto inicial da trilha ?
   Sim: Fim do algoritmo.
   Não: Volte ao passo 4.

Uma vez que a trilha "ótima" tenha sido calculada, o próximo estágio é a aplicação de algum modelo cinemático ou dinâmico ao movimento que deve ser executado de acordo com a trajetória gerada. Considerando que podemos estar modelando o movimento de objetos rígidos, aos invés de apenas entidades "puntiformes", sugere-se manter o maior eixo do objeto na mesma direção da trilha gerada.

Um exemplo para uma trajetória gerada é apresentado na Figura 7.2.

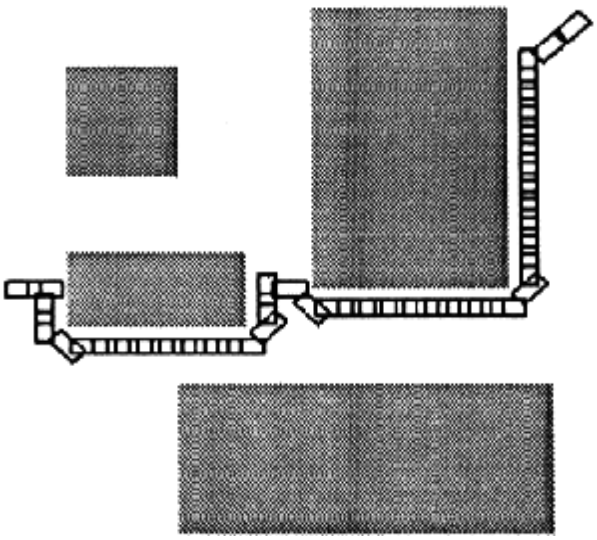

Figura 7.2: Exemplo de trilha gerada através do algoritmo de planejamento de trilhas.

Capítulo 8

CONTROLE GLOBAL ATRAVÉS DO MODELO DE DEDS

Como introduzido no capítulo 1, a subdivisão em dois níveis, denominados local e global, pode trazer benefícios no que diz respeito à simplificação da modelagem do sistema.

Dentro deste contexto, o controle local realizará o deslocamento efetivo de um determinado objeto; por outro lado, através do controle global controlamos a interacão entre atores e entre um ator e o animador. Como ilustracão da interação entre os métodos de controle local e global será discutido, nesta seção, o controle de uma estrutura bípede.

Podemos modelar um bípede através de uma estrutura articulada. De fato, são necessárias duas estruturas articuladas, idênticas no caso, para compormos o bípede. Analogamente, para um modelo que se assemelha a um ser humano devemos considerar mais duas estruturas articuladas para os seus braços.

Quando são consideradas cada uma das estruturas articuladas individualmente, vislumbramos o controle local do ator. Ou seja, devemos solucionar o problema do posicionamento de cada uma das "pernas" ao longo do tempo isoladamente. Este problema pode ser encarado do ponto de vista da cinemática inversa, ou seja, dadas as posições inicial e final de um objeto, devemos calcular as posições intermediárias para o objeto, de modo que o movimento pareça o menos robótico possível. Observemos a Figura 8.1.

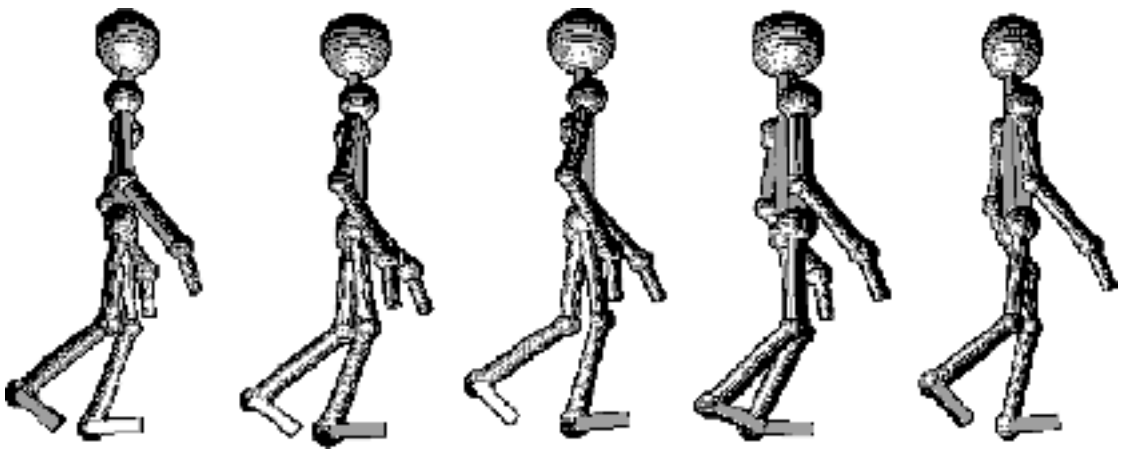

Figura 8.1: Um bípede composto por dois pares de estruturas articuladas.

Por sua vez, não basta controlarmos o movimento de cada perna isoladamente, visto que, em um bípede, deve haver coordenação entre as pernas para que a translação do ator seja efetivamente realizada. Por exemplo, enquanto caminha, não é permitido que ambas as pernas encontrem-se no ar simultaneamente. Dessa forma, quando estamos lidando com a interacão que existe entre as pernas, estamos diante do problema de controle global do movimento.

Por simplicidade, ao modelar-se o movimento da estrutura articulada bípede, será considerado que o mesmo pode apenas deslocar-se em linha reta. O controle local do movimento pode ser modelado com o auxílio de técnicas de cinemática inversa estudadas anteriormente. Quanto ao controle global, este será modelado como uma Máquina de Estados Estendida (ESM - *Extended State Machine*), pois este tipo de sistema pode ser encarado como um Sistema Dinâmico a Eventos Discretos (DEDS- *Discrete Event Dynamic System*). Neste trabalho abordaremos alguns conceitos relevantes sobre DEDS, sendo que maiores detalhes podem ser encontrados em **[Ho (1989)]**, **[Cohen (1989)]** e **[Ostroff (1989)]**.

8.1 Projeto do mecanismo de controle global para um objeto bípede

A tarefa do mecanismo de controle global é coordenar o movimento das "pernas" em termos de suas respectivas posições e velocidades. O modelo de locomoção proposto em **[Girard (1985)]** não se baseia em DEDS, porém são descritos alguns parâmetros importantes tanto para o modelo cinemático quanto para o modelo de sistema a eventos discretos. Nas próximas subseções serão discutidos em detalhe estes modelos.

8.1.1 Modelo de locomoção cinemático

O modelo de locomoção discutido em **[Girard (1985)]** utiliza alguns parâmetros para descrever o deslocamento de um personagem dotado de pernas. São eles:

- Um **padrão de locomoção** descreve uma sequência de suspensões e descidas do pé. O padrão se repete à medida em que o personagem se move. Cada repetição de uma sequência 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.
- A **fase relativa da perna i**, R_i, descreve a fração do período do ciclo de locomoção gasto antes que a perna "i" seja suspensa.
- Durante cada período do ciclo de locomoção qualquer uma das pernas passará uma parte deste no solo. Tal fração é chamada de **fator de suporte** da perna "i". Este fator pode ser utilizado para distinguirmos entre o caminhar e o correr de um bípede.
- O tempo em que um pé permanece no solo é chamado de **tempo de suporte**.
- O tempo que uma perna permanece no ar é chamado de **tempo de transferência**.

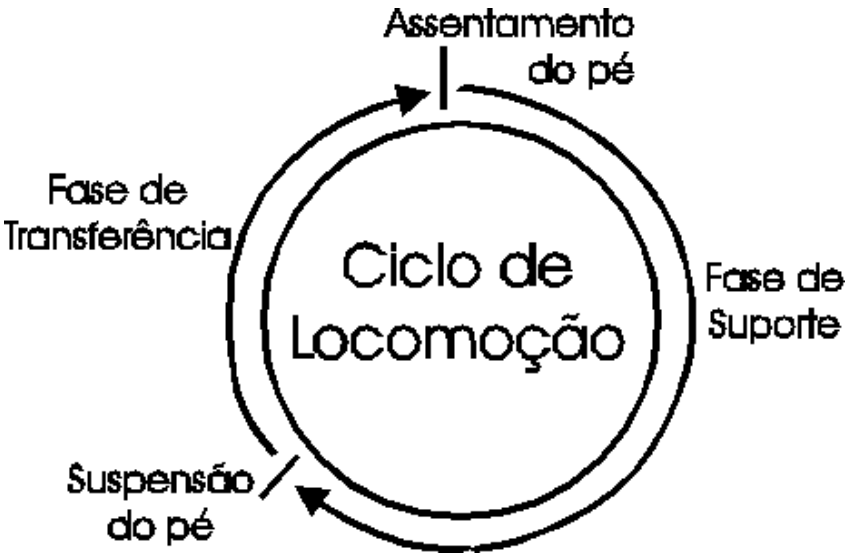

Figura 8.2: Ciclo de locomoção de uma perna [Girard (1985)].

- O **deslocamento** é definido como a distância percorrida pelo personagem durante a fase de suporte do pé.

A Figura 8.2 ilustra o ciclo de locomoção de uma perna de um bípede.
As seguintes equações descrevem o modelo cinemático de locomoção:

$$Tempo\_de\_Suporte = \frac{deslocamento}{Velocidade\_do\_Personagem} \tag{8.1}$$

$$Fase\_de\_Suporte = \frac{Tempo\_de\_Suporte}{P} \tag{8.2}$$

$$Tempo\_de\_Transferencia = P - Tempo\_de\_Suporte \tag{8.3}$$

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 das duas pernas. Na seção seguinte será estruturado o controle global do sistema através de um modelo a eventos discretos, **[Camargo (1994)]** e **[Camargo (1995)]**.

8.2 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 as pernas do bípede. As relações desenvolvidas e os parâmetros especificados na seção anterior são úteis ao modelo, em especial, devemos reconsiderar a Figura 8.2.

Visto que o sistema será modelado como um conjunto de ESMs, é interessante apresentar, inicialmente, o método para se modelar sistemas com o auxílio de ESMs.

8.2.1 Definição da síntaxe de ESMs

Definição: Uma ESM básica é uma quíntupla (X,Y,C,L,A) onde, segundo **[Ostroff (1989)]**:

X é um conjunto singular [x] onde "x" é uma variável de atividade com uma classe associada tipo(x). Os elementos de tipo(x) são chamados atividades ou estados.

Y é um conjunto de variáveis de dados onde cada variável de dado $y \in Y$ tem associada uma classe tipo(y).

C é um conjunto de canais de comunicação. Um canal de comunicação pode ser considerado como sendo uma linha de comunicação unidirecional conectando dois processos, através do qual um sinal ou dado pode ser transmitido.

L é um conjunto de rótulos de eventos.

A é um conjunto de ações básicas.

AÇÕES BÁSICAS: Cada ação básica em A é dada por [(x,E)], onde "E" é um evento dado pela quádrupla (a_s , guarda , operação , a_d), onde:

a_s é uma atividade (ou estado) de origem no tipo(x).

a_d é uma atividade (ou estado) de destino no tipo(x).

guarda é uma expressão booleana com as variáveis de dados.

operação é descrita em detalhe abaixo.

OPERAÇÕES:
Existem três tipos de operação:

1. **Uma operação de atribuição é dada por:**
   α**[y:a]** (lê-se "a" é atribuido a "y") onde "α" é o rótulo de um evento, "y" é uma variável de dados e "a" é uma expressão. A expressão "a" deve ser do mesmo tipo da variável "y" . Atribuições simultâneas devem ser representadas na forma $\alpha[y_1 : a_1$, $y_2 : a_2]$.

2. **Uma operação de envio é dada por:**
   c!m onde "c" é um canal de comunicação em "C" e "m" é uma mensagem. Uma mensagem pode ser tanto um **termo** (isto é, uma expressão) com variáveis de dados ou de atividade, como um rótulo de um evento.

3. **Uma operação de recebimento é dada por:**
   c?r onde "c" é um canal de comunicação e "r" pode ser tanto uma variável de dados como um rótulo de um evento.

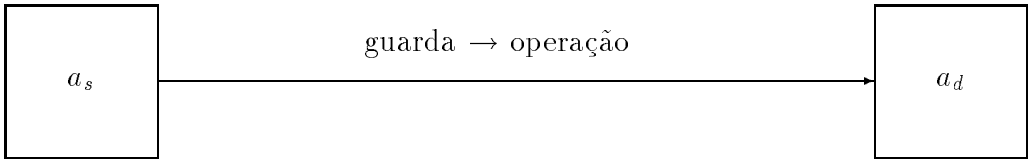

Figura 8.3: Representação de uma transição entre dois estados.

Por sua vez, o gráfico da Figura 8.3 é uma representação de um evento. Se o **guarda** é omitido, então este é assumido como VERDADEIRO. Uma interpretação informal do evento é: *Se a ESM encontra-se atualmente na atividade a_s e, se o* **guarda** *é avaliado como sendo VERDADEIRO, então saltamos para o estado a_d enquanto executamos a* **operação**.

Operações de comunicação podem ser utilizadas tanto para comandar a execução de um evento específico como para comunicar uma mensagem. Uma descrição informal desses tipos de comunicação é apresentada a seguir:

COMANDO PARA EXECUTAR UM EVENTO:
A operação de envio **c!**α em algum processo M_1 da ESM é lida como: *envie o comando* **faça** α *pelo canal "c" (de M_1) para algum outro processo (M_2, por exemplo).* A operação de recebimento **c?**α no processo M_2 é lida como: *receba o comando* **faça** α *(de M_1) pelo canal "c" e execute o comando* α. As operações de envio e recebimento, quando sincronizadas, resultam neste **comando e execução do evento** α.

COMUNICANDO UMA MENSAGEM:
Seja "m" um termo (o valor de "m" é uma mensagem) e seja "r" uma variável de dado. Então, a operação **c!m** em algum processo M_1 é lida como: *envie a mensagem "m" pelo canal "c" (para algum processo M_2).* A operação correspondente **c?r** em M_2 é lida como: *receba uma mensagem (de M_1) pelo canal "c" e armazene a mensagem em "r".* As operações de envio e recebimento em conjunto resultam em uma atribuição distribuída de "m" para "r".

Uma operação de envio $c_1!m$ "casa-se" a uma operação de recebimento $c_2?r$ se $c_1 = c_2$ e, tanto "m" como "r" são um único rótulo de evento, ou "r" é uma variável de dados e "m" é um termo do mesmo tipo de "r" (isto é, tipo(m)=tipo(r)). Uma **ação de envio** (isto é, uma ação básica contendo uma operação de envio) em uma ESM básica **M** não pode ter uma ação de recebimento correspondente em **M**, pois não existe comunicação interna dentro da própria ESM (um canal representa uma conexão de um-para-um de **M** para alguma outra ESM).

As definições anteriores não completam toda a sintaxe de uma ESM, contudo, são suficientes para o escopo deste trabalho. Dessa forma, a seguir será apresentado o modelo de locomoção para o bípede.

8.2.2 Planejamento das ESMs para o modelo de locomoção

Podemos considerar cada uma das pernas do bípede como sendo uma ESM básica, sendo que, associados a uma perna "i", temos os seguintes estados:

S_i - Fase (estado) de suporte da perna "i".

T_i - Fase (estado) de transferência da perna "i".

Também temos associados a cada perna "i" os seguintes eventos:

α_i - início da suspensão do pé "i".

β_i - fim da descida do pé "i".

Como especificação básica de controle temos:

Não é permitido que ambas as pernas encontrem-se na fase de transferência simultaneamente.

Para as pernas direita, P_D, e esquerda, P_E, podemos formalizar as seguintes ESMs básicas:

ESM P_E:
$P_E = [(x_E), \emptyset, (m_E, c_E), (\alpha_E, \beta_E), A_E]$
$\text{tipo}(x_E) = (S_E, T_E)$

$$A_E = [\ [(x_E, (T_E, \beta_E, \emptyset, S_E))]\ ,\ [(x_E, (S_E, c_E?\alpha_E, \emptyset, T_E))]\ ,\ [(x_E, (S_E, TRUE, m_E!x_E, S_E))]\ ,\ [(x_E, (T_E, TRUE, m_E!x_E, T_E))]\ ]$$

ESM P_D:
$P_D = [(x_D), \emptyset, (m_D, c_D), (\alpha_D, \beta_D), A_D]$
$\text{tipo}(x_D) = (S_D, T_D)$

$$A_D = [\ [(x_D, (T_D, \beta_D, \emptyset, S_D))]\ ,\ [(x_D, (S_D, c_D?\alpha_D, \emptyset, T_D))]\ ,\ [(x_D, (S_D, TRUE, m_D!x_D, S_D))]\ ,\ [(x_D, (T_D, TRUE, m_D!x_D, T_D))]\ ]$$

Caso seja permitido ao ator mover-se apenas em linha reta indefinidamente, então seu controlador será:

ESM CONTROLADOR:
CONTROLADOR =
$[(x_C), (y_E, y_D), (m_E, m_D, c_E, c_D), (\alpha_E, \alpha_D), A_C]$
$\text{tipo}(x_C) = (L_1, L_2, L_3, L_4)$

$$
\begin{aligned}
A_C = [\ & [(x_C, (L_1, m_E?y_E, \emptyset, L_2))]\ , \\
& [(x_C, (L_2, y_E = T_E, \emptyset, L_1))]\ , \\
& [(x_C, (L_2, y_E = S_E, c_D!\alpha_D, L_3))]\ , \\
& [(x_C, (L_3, m_D?y_D, \emptyset, L_4))]\ , \\
& [(x_C, (L_4, y_D = T_D, \emptyset, L_3))]\ , \\
& [(x_C, (L_4, y_D = S_D, c_E!\alpha_E, L_1))]\ ]
\end{aligned}
$$

Graficamente, podemos representar o conjunto planta + controlador conforme indica a Figura 8.4.

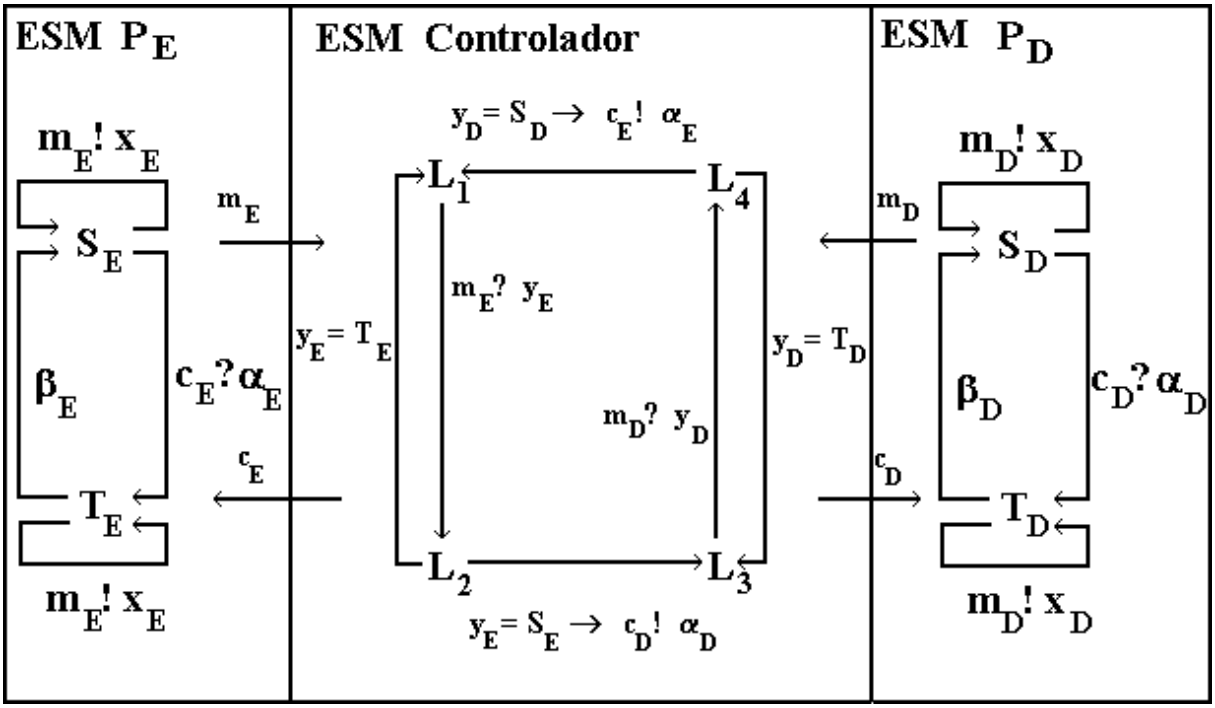

Figura 8.4: Representação gráfica para o sistema.

Capítulo 9

EXEMPLOS DE SISTEMAS DE ANIMAÇÃO

Neste capítulo faremos uma breve descrição de 3 sistemas utilizados para o desenvolvimento de animações: o ProSIm (Prototipação e Síntese de Imagens Foto-Realistas e Animação), o AutoDesk Animator Pro© e o AutoDesk 3D Studio©.

9.1 ProSIm - Prototipação e Síntese de Imagens Foto-Realistas e Animação

O ProSIm [**Magalhães (1991)**] é um projeto desenvolvido pelo Grupo de Computação de Imagens (CGI), do Departamento de Engenharia de Computação e Automação Industrial (DCA), da Faculdade de Engenharia Elétrica (FEE) da UNICAMP. É um projeto bastante abrangente, para a implementação de diversos módulos/sistemas relacionados à Computação de Imagens. Estes módulos são independentes entre si, mas interdependentes de um sistema de funções básicas, de forma a facilitar e incentivar a integração dos mesmos.

O TOOKIMA (*TOOl Kit for scripting computer Modeled Animation*) é o módulo do ProSIm que define o conjunto de ferramentas para a descrição algorítmica de animações (animação procedimental) de objetos modelados por computador – [**Hounsell (1992)**] e [**Raposo (1995a)**].

O TOOKIMA pode ser esquematizado como mostra a Figura 9.1.

9.1.1 Sistema de modelagem geométrica

O sistema permite a modelagem geométrica de superfícies poligonais, sobre as quais podem ser aplicadas as transformações geométricas básicas (translação, rotação e escalamento). É possível também o agrupamento de polígonos, em um único objeto de saída.

A modelagem é feita por meio de uma interface gráfica interativa.

O formato de saída do modelador é o *b-rep* (*boundary representation*). Para cada objeto modelado, é criado um arquivo binário contendo a lista de vértices do objeto e a maneira como estes vértices estão organizados para comporem as faces.

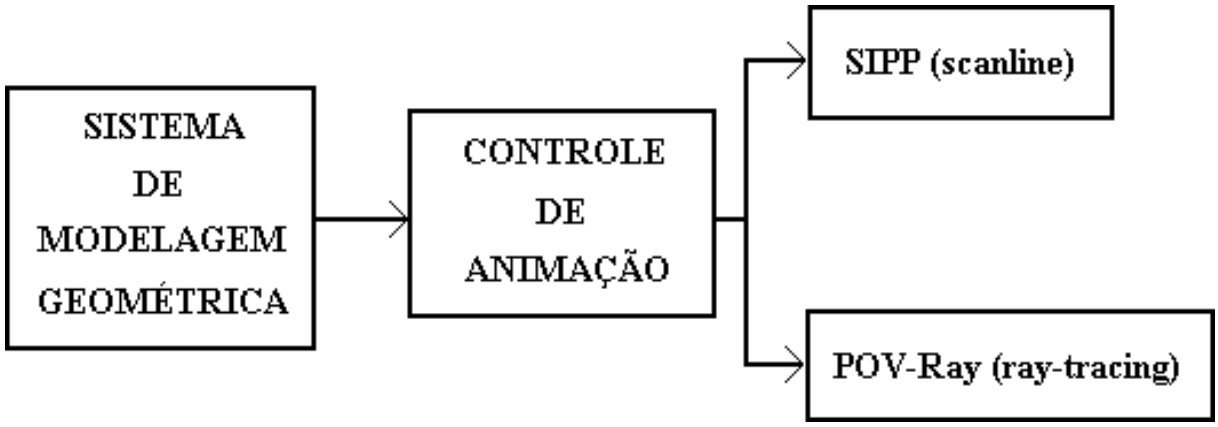

Figura 9.1: Esquematização do TOOKIMA.

9.1.2 Sistema de controle da animação

A animação é descrita por um roteiro, que deve ser escrito numa linguagem própria [**Raposo (1995b)**], que será mostrada a seguir. Entretanto, esta não é a única maneira de desenvolvermos a animação. Num nível mais baixo, é possível construir a animação com a linguagem do TOOKIMA, que é uma extensão da linguagem C. Num nível mais alto, será possível construir a animação com o auxílio de uma interface gráfica (em desenvolvimento). Esta hierarquia de linguagens, é mostrada na Figura 9.2.

Figura 9.2: Hierarquia de linguagens para a descrição de animações no ProSIm.

Um roteiro de animação no ProSIm é dividido em 8 módulos:

GENERAL: este módulo contém os parâmetros gerais da animação, tais como: tempo

total, taxa de quadros/segundo e variáveis globais – para leitura de valores resultantes de uma simulação, por exemplo.

OUTPUT: contém o formato de saída da animação (em quadros PPM, TGA, animação MPEG, *wireframe*, etc.), a resolução desta saída (320x200, 640x480, etc), a definição da existência ou não de sombras e o padrão de superamostragem (cada *pixel* pode ser renderizado internamente como uma matriz de *subpixels*, cuja cor média representará o pixel na imagem final; isto é feito para evitar *alias*).

RENDER: define o intervalo da animação a ser renderizado.

ACTORS: contém todas as informações sobre todos os atores (objetos) que comporão a animação: arquivo *b-rep* com as características geométricas, arquivo com as texturas e descrição dos movimentos a serem realizados por eles.

GROUPS: permite o agrupamento dos atores em conjuntos, para que os movimentos possam ser realizados por vários atores ao mesmo tempo.

CAMERA: contém os parâmetros de câmera (posição do observador, do centro de interesse, distância focal, etc.) e os movimentos de câmera (*zoom*, *pan*, *spin*, etc.).

LIGHTS: contém parâmetros de iluminação (cor de fundo, posição e cor das fontes, etc.) e os *fades* da iluminação.

TRACKS: define trajetórias a serem seguidas por atores, câmera ou fonte de luz. A trajetória é definida pelo cálculo de uma *spline* cúbica a partir de pontos de controle dados.

9.1.3 SIPP - *SImple Polygon Processor*

O SIPP [**Yngvesson (1994)**] é uma biblioteca de domínio público para a renderização de cenas tridimensionais, usando o algoritmo de *scan-line Z-buffer*. O SIPP possui, dentre outros, os seguintes recursos:

- Mapeamento de texturas.
- Sombras.
- Superamostragem.
- Pré-visualização em *wireframe*.
- Três tipos de *shading*: *flat shading*, método de Gouraud e método de Phong.

Algoritmos de *scan-line* são mais rápidos que os de *ray tracing*. Entretanto, os últimos apresentam resultados muito mais realistas. Por esta razão, o ProSIm tem evoluído no sentido de possibilitar uma nova opção de *renderer*, além do SIPP. Esta nova opção é o POV-Ray, que será visto a seguir.

9.1.4 POV-Ray - *Persistence Of Vision Ray tracer*

O POV-Ray [**Young (1994)**] é um *ray tracer* de domínio público [1], que utiliza um formato próprio de roteiro para a descrição da imagem a ser renderizada. Um exemplo de animação construída com o POV-Ray foi vista na seção 2.1.

O POV-Ray apresenta, resumidamente, os seguintes recursos:

- Formas pré-definidas
  - Primitivas sólidas finitas (esferas, paralelepípedos, cilindros, cones, toróides, *blobs* – esferas flexíveis, levando a um formato parecido com bolhas e *Height Fields* – usados para definição de montanhas).
  - Superfícies finitas (triângulos, disco e bicúbicas com 16 pontos de controle).
  - Primitivas infinitas (planos e superfícies polinomiais).
- CSG (*Construtive Solid Geometry*): permite a construção de novos objetos a partir de operações booleanas (união, interseção, diferença, etc) em objetos pré-existentes.
- Alterações de objetos por *clipping*.
- Biblioteca de texturas, com possibilidade de criação de texturas em camadas, transparentes, turbulências, mapas de texturas, etc.
- Fontes de luz puntuais, extensas, direcionadas, etc.

9.2 AutoDesk Animator Pro©

O AutoDesk Animator Pro© é um sistema para o desenvolvimento de animações bidimensionais, utilizando a técnica de quadros-chave (animação *keyframe*).

As animações serão produzidas nos formatos *flic* (.FLI ou .FLC), em qualquer resolução permitida pelo *hardware*.

O sistema possui ferramentas de desenho, para a elaboração da animação quadro a quadro, mas seu recurso mais importante é a possibilidade de construir a animação através de *cels*. O conceito de animação por *cels* deriva da animação tradicional, na qual cada elemento móvel da cena é desenhado em folha transparente de celulóide, que é colocada sobre o fundo fixo e reposicionada ao longo da sequência de quadros, para simular o movimento [**Freiwald (1992)**]. No Animator, o *cel* tem mais ou menos o mesmo propósito que na animação tradicional; ele é um elemento móvel que pode ser movimentado sobre o fundo.

A seguir, listamos os principais recursos do Animator.

- Ferramentas de desenho.
- Animação por *cels* (*cels* podem ser figuras desenhadas no próprio Animator ou qualquer imagem *gif*).

[1] O POV-Ray pode ser encontrado em http://www.povray.org.

- Definição de uma cor-chave, que determina a transparência sobre o fundo.
- Definição de trajetórias curvas (*splines*) para os *cels*.
- Ferramentas para a criação dos créditos da animação, embora isso também possa ser feito através de *cels*.
- Transfigurações polimorfas (o sistema automatiza a metamorfose de uma figura poligonal em outra).
- Ilusão de terceira dimensão. O Animator não é um programa tridimensional, pois não modela os objetos em 3 dimensões, mas com ele podemos criar a ilusão de uma terceira dimensão, simulando, por exemplo, o movimento de afastamento (ou aproximação) ao longo do eixo z, ou a rotação de imagens bidimensionais em torno do eixo x ou y (efeito semelhante ao de girar uma folha de papel).

9.3 AutoDesk 3D Studio©

O AutoDesk 3D Studio© é um sistema que permite desde a modelagem da forma de objetos tridimensionais até sua animação. Ele é um sistema de animação muito utilizado por animadores profissinais.

O 3D Studio é composto de 5 módulos praticamente independentes, que interagem entre si:

2D Shaper: permite a criação e edição de figuras bidimensioniais.

3D Lofter: converte os polígonos criados no *2D Shaper* em objetos tridimensionais. Também permite a edição, distorção e deformação dos objetos a serem convertidos.

Materials Editor : cria e edita materiais para as superfícies dos objetos.

3D Editor: edita cenas estáticas de objetos convertidos pelo *3D Lofter* ou de primitivas tridimensionais criadas por ele mesmo (permitindo também alterações nesses objetos). Este módulo é o responsável pela determinação da iluminação, do pano de fundo (*background*)e do posicionamento da câmera.

Keyframer: anima a cena criada no *3D Editor*. Também permite a edição de objetos, a criação de luzes e câmeras.

A organização do 3D Studio é vista na Figura 9.3.

9.3.1 2D Shaper

Módulo que modela polígonos bidimensionais, que servirão como base para a construção de malhas tridimensionais com o *3D Lofter*. As figuras bidimensionais também podem ser usadas como superfícies planas na cena.

Permite a modelagem de primitivas (círculo, retângulo, elipse, etc.), de caracteres (em várias fontes) ou de qualquer linha poligonal.

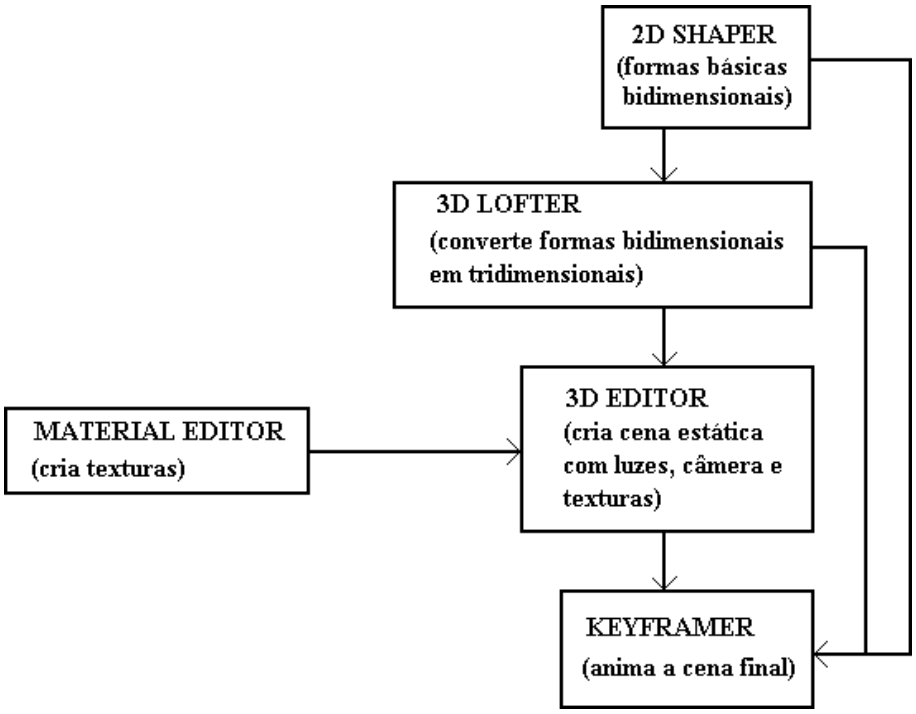

Figura 9.3: Organização do 3D Studio [Malheiros (1995)].

Os polígonos, uma vez construídos, podem ser editados neste módulo, que apresenta uma grande variedade de ferramentas de ajuste de vértices, de segmentos e de polígonos.

9.3.2 3D Lofter

Nesse módulo, os polígonos são importados do *2D Shaper* e locados em um *path*, para serem convertidos em objetos tridimensionais.

O *path* é uma linha guia que determina o direcionamento em que os polígonos são convertidos em objetos tridimensionais. Ele pode ser um polígono (geralmente aberto) construído no *2D Shaper* e pode ser editado no *3D Lofter*. É possível também criar *paths* para objetos de revolução e *paths* helicoidais.

O *path* tem vários níveis e, em cada um, podemos ter um polígono diferente, determinando as secções transversais do objeto tridimensional.

Este módulo também permite deformações na transformação de polígonos em objetos tridimensionais, alterando, por exemplo, a escala ou a posição de rotação dos polígonos em relação ao *path*.

9.3.3 Materials Editor

Permite a criação e alteração de materiais usados nas superfícies de objetos (texturas). Durante a edição destes materiais, é possível a pré-visualização dos mesmos em esferas ou cubos-exemplos.

Os materiais são criados controlando parâmetros tais como: componentes ambiente, difusa e especular da cor da superfície, transparência, brilho, nitidez da reflexão sobre ele

(*reflection blur*) e iluminação própria (efeito de "fonte de luz").

Também podem ser criadas texturas por mapeamento de uma imagem. Isto é, uma imagem pode ser "aplicada" à superfície do objeto. O mapeamento pode ser feito com uma imagem estática (.TGA, .GIF, .TIF, .JPG, .BMP ou .CEL – do Animator), com uma animação *flic* (.FLI ou .FLC) ou com uma lista numerada de imagens.

9.3.4 3D Editor

Este é o módulo mais abrangente do 3D Studio. Trabalha com aspectos que vão desde a modelagem geométrica (primitivas tridimensionais) até a renderização, passando pela determinação das câmeras e da iluminação da cena.

A seguir listaremos as principais funções deste módulo:

- Modelagem de primitivas sólidas: esfera, cone, cilindro, toróides, etc.
- Edição de formas tridimensionais: qualquer objeto pode ser editado pelo *3D Editor*, seja ele uma primitiva criada neste módulo ou criado pelo *3D Lofter*. A edição pode ser feita em nível de vértices, de faces ou de objetos. Novos objetos tridimensionais podem ser criados, conectando vértices e construindo faces.
- Ajuste nas coordenadas de mapeamento de texturas: o *3D Editor* trabalha em conjunto com o *Materials Editor*. É o *3D Editor* que aloca uma determinada textura a um objeto e ajusta as coordenadas de mapeamento (que pode ser plano, esférico ou cilíndrico).
- Iluminação: cria e edita fontes de luz, que podem ser do tipo luz ambiente (só existe uma na cena e não produz sombra), luz onidirecional (tem posição no espaço e ilumina em todas as direções, não produz sombra e podem existir várias delas na cena) ou do tipo *spot* (direcionada, com intensidade atenuada pela distância, produz sombras e pode existir mais de uma na cena).
- Câmera: cria e altera parâmetros de câmera. Pode existir mais de uma câmera na cena.
- Sombreamento (*shading*): pode ser do tipo *flat*, *Gouraud*, *Phong* ou *metal* (semelhante a *Phong*, mas dá características metálicas ao objeto).
- Parâmetros para a renderização de objetos: define se cada objeto produzirá ou não sombra, se será ou não visível, se será visto em *wireframe*, se terá a textura aplicada também ao lado interno das faces, etc.
- Parâmetros gerais de renderização: define o formato e dispositivo de saída, além de outros parâmetros que são úteis na conversão para vídeo.

Até aqui, os módulos trataram de modelagem e renderização (*3D Editor*). Só resta, portanto, dar movimentação à cena, o que será feito pelo *Keyframer*, visto a seguir.

9.3.5 Keyframer

Este módulo desenvolve a animação da cena criada, a partir de quadros-chave (animação *keyframe*). Cada objeto da cena (além das fontes de luz e das câmeras) pode sofrer uma transformação (rotação, translação, etc.), cujo resultado final será armazenado em um quadro-chave. Os *in-betweens* são calculados automaticamente.

Objetos podem ser animados através de transformações de posição, rotação, escalamento, *morphing* (inclusive com a possibilidade de transformar a textura) ou *hide* (aparecer/desaparecer). A fonte de luz ambiente pode sofrer transformação de cor. As luzes onidirecionais podem ter cores e posições transformadas. Os *spots* podem ter cores, posições e ângulos de abertura transformados, além de poderem ter as posições dos alvos mudadas. Câmeras podem sofrer transformações de posição, rotação e tamanho do campo de visão, além de também poderem ter as posições dos alvos mudadas.

Este módulo também permite ajustes na polilinha que controla cada uma das transformações, controlando a velocidade e aceleração das mesmas.

É também possível realizar movimentos em *paths*, que podem ser polígonos do *2D Shaper* ou malhas tridimensionais do *3D Lofter*. Um *path* determina a trajetória de um objeto (isto é, ele vai "andar" sobre o *path*). É possível controlar a velocidade e aceleração do movimento sobre o *path*.

Outro recurso interessante é a possibilidade de criação de uma hierarquia de atores. Dessa maneira, é possível realizar transformações em grupos de atores (uma transformação pode ser aplicada a todos os "filhos" e "descendentes" de um objeto).

O *Keyframer* permite também a criação de dois tipos especiais de objetos: objetos *dummy* (objetos que "não existem", servindo apenas para serem pais de outros objetos, auxiliando a realização de movimentos complexos, como a rotação em torno de dois eixos ao mesmo tempo) e objetos instanciados (que são cópias de um objeto e podem ser movimentadas independentemente, mas alterações feitas no original pelo *3D Editor* afetará todos os seus instanciados).

Podem ser criadas câmeras e fontes de luz, exatamente como o *3D Editor*.

Finalmente, o *Keyframer* permite a pré-visualização da animação produzida (em *bounding boxes*, em *wireframe*, ou em superfícies faceadas – *flat shading*) e a renderização final, em quadros numerados ou em animação no formato *flic* (com possibilidade de ajustes na paleta de cores, controle para gravação em vídeo, composição de imagens/animações e efeitos de transição – *fades*, cor-chave, etc.).

Capítulo 10

BIBLIOGRAFIA

[**Armstrong (1985)**] - ARMSTRONG, W. W. et alli; "*The dynamics of articulated rigid bodies for the purpose of animation*"; The Visual Computer, v. 1, n. 4, pp. 231-240; Springer-Verlag, December, 1985.

[**Boyce (1985)**] - BOYCE, W. E. et alli; "*Equações diferenciais elementares e problemas de valores de contorno*"; Guanabara Dois, pp. 315-353; 1985.

[**Camargo (1994)**] - CAMARGO, J. T. F., MAGALHÃES, L. P. e RAPOSO, A. B.; "*Modeling Motion Simulation with DEDS*; Proc. of the 13^{th} World IFIP Congress; v. 2, pp. 162-167; Hamburg, Germany, September, 1994.

[**Camargo (1995)**] - CAMARGO, J. T. F.; "*Animação Modelada por Computador - Técnicas de Controle de Movimento em Animação*"; Dissertação de Doutorado apresentada à FEE/UNICAMP; Janeiro, 1995.

[**Cohen (1989)**] - COHEN, G. et alli; "*Algebraic tools for the performance evaluation of discrete event systems*"; Proc. IEEE, v. 77, n. 1, pp. 39-57; January, 1989.

[**Craig (1989)**] - CRAIG, J. J.; "*Introduction to robotics: mechanics and control*"; 2^{nd} Ed., Addison-Wesley Publishing Company, 1989.

[**Freiwald (1992)**] - FREIWALD, L. et alli; "*Autodesk Animator - Guia Completo de Animação no PC*"; Berkeley Brasil Editora, 1992.

[**Gewali (1990)**] - GEWALI, L. P. et alli; "*Path Planning in the Presence of Vertical Obstacles*"; IEEE Trans. on Robotics and Automation, v. 6, n. 3, pp. 331-341; June, 1990.

[**Gilbert (1985)**] - GILBERT, E. G. et alli; "*Distance Functions and Their Application to Robot Path Planning in the Presence of Obstacles*"; IEEE Journal of Robotics and Automation, v. RA-1, n. 1, pp. 21-31; March, 1985.

[**Gill (1981)**] - GILL, P. E. et alli; "*Practical Optimization*"; Academic Press Inc. Ltd., 1989.

[**Girard (1985)**] - GIRARD, M. et alli; "*Computational modeling for the computer animation of legged figures*"; ACM Computer Graphics, v. 19, n. 3, pp. 263-270; July, 1985.

[Goldenberg (1985)] - GOLDENBERG, A. A. et alli; "*A complete generalized solution to the inverse kinematics of robots*"; IEEE Journal of Robotics and Automation, v. RA-1, n. 2, pp. 14-20; March, 1985.

[Gondran (1986)] - GONDRAN, M. et alli; "*Graphs and algorithms*"; John Wiley and Sons; 1986.

[Hayt (1981)] - HAYT Jr., W. H.; "*Engineering electromagnetics*"; McGraw-Hill Inc., 1981.

[Ho (1989)] - HO, Y.; "*Dynamics of Discrete Event Systems*"; Proc. IEEE, v. 77, n. 1, pp. 3-6; January, 1989.

[Hounsell (1992)] - HOUNSELL, M. S.; "*TOOKIMA: Uma Ferramenta para Animação Modelada por Computador*"; Dissertação de Mestrado apresentada à FEE/UNICAMP; 1992.

[Hwang (1992)] - HWANG, Y. K. et alli; "*A Potential Field Approach to Path Planning*"; IEEE Trans. on Robotics and Automation, v. 8, n. 1, pp. 23-32; February, 1992.

[Isaacs (1987)] - ISAACS, P. M. et alli; "*Controling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics*"; ACM Computer Graphics, v. 21, n. 4, pp. 215-224; July, 1987.

[Isaacs (1988)] - ISAACS, P. M. et alli; "*Mixed methods for complex kinematic constraints in dynamic figure animation*"; Visual Computer, v. 4, n. 6, pp. 296-305; 1988.

[Korein (1982)] - KOREIN, J. U. et alli; "*Techniques for generating the goal-directed motion of articulated structures*"; IEEE Computer Graphics and Applications, pp. 71-81; November, 1982.

[Magalhães (1992)] - MAGALHÃES, L. P. et alli; "*ProSIm - Um Sistema para Prototipação e Síntese de Imagens Foto-Realistas e Animação*"; Relatório Interno - DCA - 030/31 - FEE - UNICAMP, 1992.

[Malheiros (1995)] - MALHEIROS, P.; "*3D Studio 4.0 - Guia Completo*"; Berkeley Brasil Editora, 1995.

[Moore (1988)] - MOORE, M. et alli; "*Collision Detection and Response for Computer Animation*"; Computer Graphics, v. 22, n. 4, pp. 289-298; August, 1988.

[Ostroff (1989)] - OSTROFF, J. S.; "*Temporal logic for real time systems*"; Research Studies Press Ltd., John Wiley & Sons Inc.; 1989.

[Raposo (1995a)] - RAPOSO, A. B.; "*TOOKIMA - Uma Ferramenta para Animação Modelada por Computador*"; Relatório Interno - DCA - 001/95 - FEE - UNICAMP; 1995.

[Raposo (1995b)] - RAPOSO, A. B.; "*Uma Linguagem para Desenvolvimento de Roteiros de Animação*"; Relatório Interno - DCA - 002/95 - FEE - UNICAMP; 1995.

[Takahashi (1989)] - TAKAHASHI, O. et alli; "*Motion Planning in a Plane Using Generalized Voronoi Diagrams*"; IEEE Trans. on Robotics and Automation, v. 5, n. 2, pp.

143-151; April, 1989.

[Thalmann (1985)] - THALMANN, N. M. et alli; "*Computer Animation: Theory and Practice*"; Springer-Verlag; 1985.

[Wilhelms (1988)] - WILHELMS, J. et alli; "*Dynamic animation: interaction and control*"; Visual Computer, v. 4, n. 6, pp. 283-295; 1988.

[Yngvesson (1994)] - YNGVESSON, J. et alli; "*User's Guide to SIPP - A 3D Rendering Library - Version 3.1*"; 1994.

[Young (1994)] - YOUNG, C. et alli; "*Persistence of Vision Ray-Tracer (POV-Ray) - Vesion 2.0 - User's Documentation*"; 1994.