UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas de São Carlos
Departamento de Ciência de Computação e Estatística

**Animação de Simulações de
Sistemas Mecânicos
Multicorpos**

Rogério Toshiaki Kondo

Profa. Dra. Maria Cristina Ferreira de Oliveira

Dissertação apresentada ao Instituto de
Ciências Matemáticas de São Carlos da
Universidade de São Paulo como parte dos
requisitos para obtenção do Título de
Mestre - Área: Ciências de Computação e
Matemática Computacional

São Carlos, SP
Novembro de 1997

*À minha família e aos meus
amigos*

Agradecimentos

Meus sinceros agradecimentos

- À Profa. Dra. Maria Cristina Ferreira de Oliveira pela orientação, paciência, compreensão e confiança depositada para alcance do objetivo.
- Ao Prof. Dr. Álvaro Costa Neto pelo apoio e iniciativa para o desenvolvimento deste trabalho.
- Aos professores do ICMSC/USP pelo estímulo e valiosos ensinamentos durante a graduação e pós-graduação.
- Aos meus pais, Katsumi e Helena, pela dedicação, empenho e preocupação pela minha formação.
- À minha irmã Angélica e aos meus irmãos Cézar e Clóvis.
- Ao amigo Evair.
- Aos amigos da graduação e pós-graduação pelos momentos de amizade e companheirismo.
- Aos amigos do CISC/USP e SIn/UFSCar pelo apoio durante o trabalho do dia-a-dia.

Resumo

Este trabalho apresenta o **Animbs (Animation for MBS)**, um sistema capaz de visualizar dados gerados por um sistema de simulação de engenharia (SD/FAST) na forma de animações por computador. SD/FAST é um sistema utilizado para o modelamento e a simulação de sistemas mecânicos multicorpos (MBS). O sistema **Animbs** permite a associação de uma geometria ao MBS sendo simulado e utiliza os dados produzidos pela simulação do SD/FAST para criar uma animação do comportamento do sistema mecânico e, dessa forma, melhorar a análise de dados feita pelos usuários do SD/FAST.

Abstract

This work presents **Animbs** (**A**nimation **f**or **M**BS), a software that enables the visualization of data generated by an engineering simulation system (SD/FAST) in the form of computer animation. SD/FAST is a system for modeling and simulation of multibody systems (MBS). The **Animbs** system allows the association of a geometry to the MBS being simulated, and uses the data produced by the SD/FAST simulation to create an animated view of the MBS behavior, thus providing support for enhanced data analysis by users of SD/FAST.

Índice

Introdução	1
 Capítulo 1	
Animação Convencional	4
1.1 Considerações Iniciais.....	4
1.2 Histórico.....	5
1.3 Princípios Básicos da Animação Convencional.....	9
1.4 Técnicas e Efeitos Especiais de Câmera	13
1.5 Considerações Finais.....	15
 Capítulo 2	
Animação por Computador	16
2.1 Considerações Iniciais.....	16
2.2 O Computador na Animação.....	17
2.3 Classificação de Sistemas de Animação por Computador.....	18
2.4 Elementos da Animação Tridimensional por Computador.....	22
2.5 Tipos de Objetos e seus Movimentos	24
2.6 Técnicas de Controle de Movimentos.....	27
2.6.1 Manipulação Direta.....	27
2.6.2 Interpolação de <i>Keyframes</i>	28

2.6.3 Interpolação Paramétrica.....	28
2.6.4 Sistemas de <i>Script</i>	29
2.6.5 <i>Rotoscoping</i>	29
2.6.6 Controle por Procedimentos ou Procedimental	30
2.6.7 Cinemática Inversa.....	30
2.6.8 Análise Dinâmica.....	31
2.7 Considerações Finais.....	32
 Capítulo 3	
Sistemas Multicorpos	33
3.1 Considerações Iniciais.....	33
3.2 Técnicas de Modelamento de Sistemas Multicorpos	33
3.3 Implementação Computacional.....	38
3.4 Considerações Finais.....	41
 Capítulo 4	
Sistema de Simulação SD/FAST.....	42
4.1 Considerações Iniciais.....	42
4.2 Arquivo de Descrição do Sistema	43
4.2.1 Preâmbulo	44
4.2.2 Corpos e Juntas da Árvore	44
4.2.3 Juntas de <i>Loop</i>	46
4.2.4 Restrição do Usuário.....	46
4.3 Juntas.....	47
4.4 Vetor de Estados	48
4.5 Arquivos Gerados pelo SD/FAST.....	49
4.6 Exemplo de Utilização	51
4.7 Considerações Finais.....	54
 Capítulo 5	
Geração da Animação - Animbs.....	55

5.1 Considerações Iniciais.....	55
5.2 Biblioteca de Extensão.....	56
5.2.1 Rotinas da Biblioteca	57
5.2.2 Formato dos Arquivos de Dados Gerados pelas Rotinas da Biblioteca...	60
5.3 O Sistema Animbs	61
5.3.1 Biblioteca OpenGL	62
5.3.2 Funcionalidade do Sistema Animbs.....	63
5.3.3 Implementação do Animbs	70
5.4 Exemplo de Utilização	75
5.5 Considerações Finais.....	86
 Conclusão	 87
Referências	90
 Apêndice A	
Sintaxe do Arquivo de Descrição do Sistema	94
 Apêndice B	
Rotinas do SD/FAST	96
 Apêndice C	
Diagrama de Fluxo de Dados do Animbs.....	99
 Apêndice D	
Projeto Modular do Animbs	101

Lista de Figuras

1.1 Lanterna mágica	5
1.2 Uma das invenções para ilusão de movimento.....	6
1.3 Quadro do desenho animado <i>Gertie, the Dinosaur</i>	7
1.4 O <i>rotoscope</i>	8
1.5 Organização de um filme animado.....	9
1.6 <i>Storyboard</i> do desenho <i>Beauty and the Beast</i> , de Bosutow Entertainment.....	10
1.7 Um exemplo de formulário do modelo da personagem <i>Hootie Owl</i> , em <i>Heidi's Song</i> , de Hanna-Barbera.....	11
1.8 Filme de 35 mm.....	13
1.9 Tipos do efeito <i>wipe</i>	14
2.1 Uma taxonomia das técnicas de modelagem.....	22
2.2 Uma taxonomia das técnicas de controle de movimento	23
2.3 Uma taxonomia das técnicas de renderização	23
2.4 Interpolação linear para <i>keyframes</i>	28
3.1 Exemplos de sistemas mecânicos que podem ser modelados pelas técnicas MBS....	35

3.2	Topologias de sistemas MBS	35
3.3	Tipos de juntas.....	36
3.4	Origem das forças que agem sobre os sistemas mecânicos.....	37
3.5	Restrições nos modelamentos MBS	38
3.6	Três modos de gerar uma simulação MBS	40
4.1	Estado dos referenciais durante uma simulação	44
4.2	Vetores <code>bodyToJoint</code> e <code>inbToJoint</code>	45
4.3	Eixos genéricos de uma junta.....	45
4.4	Estados do sistema SD/FAST.....	50
4.5	Modelo de um mecanismo de quatro barras.....	51
4.6	Arquivo de descrição do sistema do mecanismo de quatro barras	52
4.7	Programa principal do usuário para o mecanismo de quatro barras.....	53
4.8	Exemplo de resultado da simulação do mecanismo de quatro barras	53
4.9	Exemplo de resultado gráfico do mecanismo de quatro barras, fornecendo a posição e a velocidade da manivela.....	54
5.1	Propriedades de um MBS, visualizadas no Animbs	63
5.2	Propriedades de um corpo do MBS, visualizadas no Animbs	64
5.3	Propriedades de uma junta do MBS, visualizadas no Animbs	65
5.4	Janela de topologia de um MBS, visualizada no Animbs	65
5.5	Juntas de <i>loop</i> de um MBS	66
5.6	Operações de transformação sobre a geometria dos corpos (rotação e escalonamento) e sobre os planos de visualização (<i>pan</i> , <i>zoom</i> e rotação para a <i>trackball</i> virtual)	67
5.7	Localização das projeções ortogonais do Animbs	68

5.8	Selecionando os planos de visualização no Animbs	69
5.9	Plano de projeção em perspectiva em relação ao referencial global, para a simulação da <i>trackball</i> virtual.....	69
5.10	Painel de controle da animação no Animbs	70
5.11	Propriedades da animação, visualizadas no Animbs	70
5.12	Objetos do Animbs , em <i>wireframe</i> à direita e sólido à esquerda.....	72
5.13	Sequência de multiplicação das matrizes de transformação.....	73
5.14	Janela principal do Animbs	76
5.15	Menu File	76
5.16	Janela para obtenção de arquivos de extensão inf	77
5.17	Menu system	77
5.18	Menu Body	78
5.19	Janela de criação de geometria de corpos no Animbs	78
5.20	Menu Edit	78
5.21	Geometria inicial do corpo	79
5.22	Rotação do corpo em 90 graus em relação ao eixo horizontal do plano de visualização	80
5.23	Escalonamento da geometria do corpo rotacionado.....	81
5.24	Janela para remoção de uma geometria de um corpo no Animbs	81
5.25	Menu Joint	82
5.26	Menu Rendering	82
5.27	Janela para armazenar a geometria em arquivos de extensão geo	82
5.28	Menu Animation	83
5.29	Instante da animação com ponto de observação fixo.....	84
5.30	Outro instante da animação com o ponto de observação fixo.....	85

5.31	Menu help	85
5.32	Exemplo de avisos do sistema Animbs	86

Lista de Tabelas

2.1 Funções dos sistemas de animação por computador	20
5.1 Rotinas da biblioteca de extensão.....	58

Introdução

Ao longo dos anos, pesquisadores nas áreas de Computação Gráfica e Visualização Científica vêm desenvolvendo técnicas computacionais para transformar dados em imagens. Muitos destes dados são melhor visualizados quando apresentados através de animações. Afinal, se é verdadeira a clássica afirmação de que *uma imagem é equivalente a mil palavras*, então é razoável supor que uma imagem em movimento seja capaz de revelar informações de maneira ainda mais eficiente. Este fato parece ser confirmado pela utilização de diversas técnicas de animação em várias atividades e aplicações, desde a indústria de entretenimento à visualização de dados científicos. Muitos produtos gerados nessas áreas utilizam técnicas de animação por computador, devido em grande parte aos enormes avanços nas tecnologias de *hardware* e *software*.

Uma das aplicações mais frequentes da animação por computador é na visualização de simulações de fenômenos científicos. Os resultados dessas simulações, em geral, são enormes quantidades de dados, que são posteriormente convertidos em imagens que constituirão a animação. Um exemplo de utilização é a visualização de resultados de simulações de sistemas mecânicos multicorpos (ou *Multibody Systems*, MBS).

Um MBS caracteriza-se por ser um sistema mecânico que pode ser modelado como um conjunto de corpos rígidos conectados por articulações, os quais podem apresentar um alto grau de movimentação relativa entre si. O sistema está sujeito à ação de forças, obedecendo a um

conjunto de restrições, e tem seu comportamento governando por equações dinâmicas de movimento. Exemplos típicos são veículos e robôs.

Técnicas de modelamento MBS são utilizadas para analisar o comportamento de tais sistemas, uma tarefa que geralmente é apoiada por programas de computador capazes de modelar sistemas dinâmicos MBS e simular o seu comportamento. Esses programas constituem uma classe dos programas conhecidos como de *Engenharia Apoiada por Computador* (ou *Computer-Aided Engineering*, CAE), que são atualmente ferramentas indispensáveis nos processos de projeto e manufatura em engenharia.

Muito dos sistemas CAE disponíveis atualmente foram construídos como "pacotes fechados" que incluem programas de modelagem, análise e simulação, além de módulos para pré e pós processamento dos dados (esse último pode ou não incluir a geração de animações). Entretanto, essa filosofia de sistemas altamente integrados dificulta o desenvolvimento de técnicas adequadas de visualização e integração entre diferentes sistemas.

A tendência atual na área de CAE certamente aponta para a construção de programas compostos por vários módulos específicos capazes de operar de maneira independente. A filosofia modular favorece o surgimento de ferramentas mais poderosas dedicadas à modelagem, análise e simulação de sistemas dinâmicos, bem como de módulos específicos para a visualização dos resultados. Um exemplo dessa filosofia é o sistema de simulação de MBS SD/FAST que, devido à sua arquitetura aberta, possibilita ao usuário do sistema obter e concentrar sua análise em dados específicos da simulação. Estes dados precisam ainda ser manipulados por alguma ferramenta para a geração de gráficos estatísticos.

Seguindo essa filosofia modular, este trabalho consistiu no desenvolvimento de um sistema para gerar animações a partir de dados gerados pelo SD/FAST, com o objetivo de acelerar e complementar o processo de análise dos dados estatísticos gerados pelo sistema. Este trabalho foi assessorado pelo Prof. Dr. Álvaro Costa Neto, do Departamento de Engenharia Mecânica da Escola de Engenharia de São Carlos, na área referente a sistemas dinâmicos MBS.

Esta dissertação está organizada da seguinte maneira:

O capítulo 1 apresenta um histórico da animação convencional, desde as primeiras tentativas da criação da ilusão do movimento através de aparelhos espectrais até a animação moderna. Apresenta também as terminologias utilizadas na animação convencional e a técnica que deu origem à animação por computador.

O capítulo 2 apresenta o papel do computador na animação e as principais técnicas de animação tridimensional. Além disso, apresenta algumas classificações dos sistemas de animação por computador comumente encontradas na literatura.

Os sistemas multicorpos são apresentados no capítulo 3. É introduzida a terminologia utilizada nestes sistemas mecânicos, as técnicas de modelamento e suas implementações computacionais.

O capítulo 4 apresenta o sistema computacional para simulação de sistemas mecânicos multicorpos, o SD/FAST.

O capítulo 5 apresenta o sistema **Animbs**, desenvolvido para gerar animações dos resultados de simulações do SD/FAST. Finalmente, as conclusões são apresentadas no último capítulo.

Capítulo 1

Animação Convencional

1.1. Considerações Iniciais

Literalmente, *animar* significa *dar vida ou movimento*. Embora esta definição seja, em geral, adequada, segundo [SOLOM83], ela falha na essência da própria animação. A animação pode ser mágica, fazendo o público rir toda vez que um infeliz coiole cai de um penhasco e chorar diante da situação de um pequeno e solitário elefante separado de sua mãe. A animação pode ser uma ferramenta de comunicação extremamente poderosa, podendo transcender barreiras culturais e lingüísticas.

De maneira mais objetiva e técnica, Magnenat-Thalmann [MAGTH85a] exprime o conceito de animação através de duas definições:

- Animação é a técnica em que a ilusão do movimento é criada fotografando-se uma série de desenhos em sucessivo quadros. A ilusão é obtida projetando o filme a uma certa velocidade.
- Animação refere-se ao processo de gerar uma série de quadros de um conjunto de objetos, em que cada quadro é uma alteração do quadro anterior.

Estas definições descrevem o princípio da animação como foi concebido há mais de 80 anos e ainda continuam válidas até hoje, em muitas aplicações. Em outras, como os *video-games*,

estas definições são imprecisas. Além disso, não se deve limitar a animação apenas ao conceito de movimento, uma vez que pode haver animação sem movimento, por exemplo,

- nas metamorfoses, quando um objeto é transformado em outro;
- nas mudanças de cores, como em um rosto ruborizado;
- nas mudanças de intensidade de luz, como em um pôr do sol.

Neste capítulo apresenta-se um histórico do desenvolvimento da animação e a terminologia utilizada na animação convencional. É apresentada também uma das técnicas de animação mais utilizadas, amplamente simulada na animação por computador.

1.2. Histórico

Solomon [SOLOM83] fornece um histórico detalhado sobre animação convencional que serviu de base para o resumo apresentado nesta seção. A animação começou no século XVII com a invenção da *lanterna mágica* pelo padre jesuíta Athanasius Kircher, ilustrada na figura 1.1. A lanterna mágica consistia de uma pequena caixa de metal com uma fonte de luz interna e uma lente que cobria um pequeno orifício em um dos lados da caixa. A luz da fonte passava através de uma imagem pintada em uma lâmina de vidro colocado atrás da lente, e então as lentes projetavam a imagem. Ainda no final deste século, Johanes Zahn mostrou que, adaptando na lanterna mágica um disco giratório contendo várias lâminas pintadas, era possível criar a ilusão de movimento.

Figura 1.1 Lanterna mágica.

Durante os séculos XVIII e XIX, as lanternas mágicas foram popularizadas. Nesta atmosfera, Peter Roget publicou seu artigo intitulado *Persistence of Vision with Regard to Moving Objects*, em 1824. Embora algumas partes de seu trabalho tenham sido demonstradas imprecisas, o conceito de que o olho humano pode fundir imagens estáticas quando apresentadas a uma certa velocidade e com uma quantidade de luz adequada influenciou o desenvolvimento da animação.

O trabalho de Roget incentivou uma grande quantidade de invenções e experimentos. Em 1831, Joseph Antoine Plateau inventou o *phenakistiscope*, na qual uma série de desenhos eram colocados em um disco giratório e eram observados através de uma abertura. Em 1834, o inglês Horner estendeu o conceito do *phenakistiscope* e inventou o *zoetrope*, que consistia de um tambor giratório com fendas regularmente espaçadas, com desenhos no seu lado interno. Quando o tambor é rotacionado, a ilusão de movimento pode ser vista através das fendas. A figura 1.2a ilustra um *zoetrope* e a figura 1.2b ilustra uma tira de desenhos utilizados no interior do tambor. Em 1877, Emile Reynaud refinou o *zoetrope*, desenvolvendo o *praxinoscope*, no qual as fendas foram substituídas por espelhos colocados no centro do tambor.

Figura 1.2 Uma das invenções para ilusão de movimento: (a) *Zoetrope*. (b) Tira de desenho interno.

Outra manifestação desta fascinação por ilusões óticas foi a popularidade da fotografia. As primeiras fotografias foram tiradas durante as décadas de 20 e 30 do século XIX. Exatamente quando, onde e por quem as primeiras projeções de movimento de figuras foram realizadas ainda é objeto de debate. Porém, com certeza os trabalhos e experiências anteriores levaram ao

desenvolvimento de filmagem de movimento de imagens como conhecemos atualmente. Eadweard Muybridge começou experimentando com fotografias sequenciais de corridas de cavalos já em 1873. Estes estudos levaram a dois conjuntos de fólios de seqüências fotográficas, *Animals in Motion* (1899) e *The Human Figure in Motion* (1901), que foram mais tarde publicadas em livros e permanecem como referências padrão.

Figura 1.3 Quadro do desenho animado *Gertie, the Dinosaur*.

Um dispositivo para registrar imagens sequenciais foi inventado por Thomas Edson em 1888. Nesta época, a emulsão sensível à luz era fixada em lâminas de metal ou vidro, nas quais apenas uma exposição poderia ser feita. Estas chapas tornavam o recarregamento do dispositivo uma operação desajeitada e impediam a fotografia em seqüência. O filme em rolo só ficou disponível em 1888 através de um processo patenteado por George Eastman. Uma base de celulósido flexível e leve tornou possível manter filme suficiente na câmera para obter várias fotografias sucessivas sem o recarregamento. Então, no começo de 1895, em Paris, os irmãos

Lumiere mostraram a primeira projeção pública de movimento de imagens, surgindo assim o cinema.

Em 1914, Winsor McCay produziu *Gertie the Dinosaur*, que pode ser considerado como sendo o primeiro desenho animado. A figura 1.3 ilustra um dos quadros desse desenho. Embora o desenho animado seja curto, McCay utilizou pelo menos 5.000 desenhos em papel arroz. Durante os anos de 1913 a 1917, várias séries americanas foram produzidas, entre elas o desenho *Felix the Cat*, de Pat Sullivan.

Em 1915, Earl Hurd patenteou a técnica de usar folhas de celulóide ao invés do papel, poupando os animadores de redesenharem a cena de fundo para cada cena da animação. Max Fleischer patenteou, em 1915, o *rotoscope*, ilustrado na figura 1.4, um dispositivo que permitia aos animadores desenhar a partir de filmes verdadeiros para garantir movimentos realísticos. Com o *rotoscope*, Fleischer começou uma série de desenhos (1916 - 1929) intitulados *Out of the Ink-well*.

Figura 1.4 O rotoscope.

Naturalmente, o pai da animação comercial é certamente Walt Disney. Durante os anos de 1928 a 1938, Disney criou vários personagens famosos, como Mickey Mouse e o pato Donald. O primeiro curta de Mickey Mouse, em 1928, foi o primeiro filme com som sincronizado. Então veio o desenho animado comercial de longa metragem, *Branca de Neve e os Setes Anões*, no qual foi utilizado o *multiplano*, aparelho inventado por Disney que permitia o movimento de vários celulóides de acetatos em relação aos outros. Enquanto isso, outros pioneiros foram desenvolvendo desenhos animados em diferentes países, como a Rússia, a antiga Tchecoslováquia, França e Canadá.

1.3. Princípios Básicos da Animação Convencional

Filmes de desenho animado são produzidos em estúdios que utilizam diferentes metodologias de produção, envolvendo diversas pessoas. Entretanto, uma breve descrição dos principais passos na produção de um desenho animado pode ser apresentada:

Figura 1.5 Organização de um filme animado.

1. História

Como em qualquer filme comum, um filme de desenho animado geralmente possui uma história. Para descrever essa história, três tipos de documentos são necessários, sendo que cada um detalha o documento anterior. A figura 1.5 ilustra a organização de um filme.

- A **sinopse** é um resumo da história em poucas linhas.
- O **cenário** é um texto detalhado que descreve a história completa sem referências cinematográficas.

Figura 1.6 Storyboard do desenho *Beauty and the Beast*, de Bosustow Entertainment [SOLOM83].

- O **storyboard** é um filme com traços de desenho simples. Consiste de um número de ilustrações arranjadas em forma de quadrinhos com legendas apropriadas. O número de ilustrações individuais dentro de um *storyboard* varia enormemente. O mais importante é que o *storyboard* representa os quadros chaves do filme, como ilustra a figura 1.6. É importante notar que um

filme é composto de *seqüências* que definem ações específicas. Cada seqüência consiste de uma série de *cenas* que são genericamente definidas por uma certa posição e conjunto de personagens. As cenas são divididas em *fotogramas*, que são consideradas as unidades de imagens (figura 1.5).

Figura 1.7 Um exemplo de formulário do modelo da personagem *Hootie Owl*, em *Heidi's Song*, de Hanna-Barbera [SOLOM83].

2. Layout

Este passo consiste principalmente do esboço das personagens a serem animadas e suas ações, gerando um documento chamado *formulário do modelo*, como ilustrado na figura 1.7. Baseado no *storyboard*, o relacionamento entre as formas e formatos no plano de fundo e no primeiro plano são decididos.

3. Trilha sonora

Na animação convencional, a gravação da trilha sonora deve preceder o processo da animação, uma vez que o movimento da animação deve ser igual ao diálogo e/ou à música.

4. Animação

O processo de animação é realizado por animadores que desenhavam *keyframes*. Frequentemente, um animador é responsável por uma personagem específica.

5. In-betweening

In-betweens são definidos como desenhos que são colocados entre duas posições ou *keyframes*. Assistentes de animação desenhavam alguns *in-betweens* e *in-betweeners* desenhavam os quadros restantes.

6. Xeroxing e inking

Croquis são usualmente desenhados à lápis. Eles devem ser transferidos para as celulóides de acetato utilizando câmeras Xerox modificadas. As linhas de desenho devem ser traçadas à mão utilizando-se tinta (*inking*).

7. Pintura

Como os desenhos animados são geralmente coloridos, devem passar por um estágio de pintura, que exige paciência e precisão. Acetatos devem ter o grau correto de opacidade e planos de fundo estáticos devem também ser pintados.

8. Checagem

Animadores precisam checar a ação em suas cenas antes da filmagem.

9. Câmeras

A fotografia da animação geralmente é feita em filmes coloridos ou *videotapes*. A figura 1.8 ilustra um filme de 35 mm.

Figura 1.8 Filme de 35 mm.

10. Pós-produção

Esta fase inclui todos os passos para transformar a filmagem no produto final e inclui o *processamento* e *edição*. O processamento do filme envolve uma série de operações de laboratório em que as imagens expostas no filme são envolvidas com soluções químicas. Uma vez que as falhas de produção podem complicar a fase de edição, a pós-produção pode implicar em refilmagem. Editar consiste principalmente em montar, ordenar e unir o filme. A sincronização do som também faz parte da pós-produção.

1.4. Técnicas e Efeitos Especiais de Câmera

Os efeitos especiais de câmera são usados em animação convencional, mas alguns deles são difíceis de produzir e, algumas vezes, muito caros:

- **Pan**

Um pan (contração da palavra panorama) é um efeito do movimento horizontal da câmera de um ponto ao outro.

- **Tilt**

Um *tilt* é um efeito do movimento vertical da câmera de um ponto ao outro.

- **Zoom**

Um *zoom* é um efeito que torna aparentemente grande ou pequeno um objeto. Em desenhos animados bidimensionais, o efeito é produzido movendo a câmera para perto ou longe do objeto. Este efeito é impossível em três dimensões, devido ao efeito da perspectiva. Porém, lentes de *zoom* podem ser utilizadas na câmera para obter o efeito tridimensional.

- **Spin**

Um *spin* é um efeito produzido rotacionando a câmera.

- **Fade-in**

Um *fade-in* é um efeito usado no começo de uma cena: a cena gradualmente surge do preto.

- **Fade-out**

Um *fade-out* é o efeito usado no final de uma cena: a cena gradualmente desaparece no preto.

- **Cross-dissolve**

Um *cross-dissolve* é um efeito que é freqüentemente utilizado na transição entre as cenas. Corresponde a um *fade-out* de uma cena e a um *fade-in* da próxima cena sobre o mesmo comprimento de filme.

Figura 1.9 Tipos do efeito *wipe*.

- **Wipe**

Um *wipe* é um efeito em que uma cena desliza sobre a outra. A figura 1.9 ilustra diferentes possibilidades de um *wipe*.

Para outros efeitos especiais, como sobreposição ou efeitos de múltiplas imagens, uma *impressora ótica* é essencial na animação. Este aparelho é uma câmera de cinema que é focalizada na abertura da lente de um projetor de cinema para duplicar um pedaço de filme. Ela é utilizada para:

- fazer impressões em negativos;
- converter um formato de filme para outro, como por exemplo de 8 para 35 mm;
- modificar a velocidade de uma ação;
- melhorar a qualidade de um filme;
- regravar vários ciclos de um movimento repetido;
- fazer sobreposição de títulos e logotipos;
- realizar os efeitos de *fade* e *wipe*.

1.5. Considerações Finais

Este capítulo apresentou uma visão geral sobre a animação convencional — suas origens e as técnicas utilizadas. O próximo capítulo aborda as técnicas de animação por computador.

Capítulo 2

Animação por Computador

2.1. Considerações Iniciais

Com o desenvolvimento da tecnologia, os computadores tornaram-se suficientemente rápidos para permitir um desenvolvimento da computação gráfica e, principalmente, da animação por computador. Segundo Wyvill [WYVIL90], a indústria da animação por computador cresceu por volta de 150 milhões de dólares durante o decênio 77-87. Esta tecnologia permitiu que curtas-metragem de animação fossem produzidos por computador, como *La Faim* (1974) de Peter Foldes, com *software* produzido por Nestor Burtnyk e Marcelli Wein e *Luxo Junior* (1986) de John Lasseter, utilizando o *software* de Bill Reeves et al. [WYVIL90].

A técnica utilizada em *La Faim* era essencialmente bidimensional e estendia o conceito de *keyframes* da animação convencional. Em contraste, *Luxo Junior* utilizou a animação tridimensional por computador, em que os modelos das personagens e os planos de fundo são descritos no espaço tridimensional e são movidos utilizando uma variedade de técnicas, incluindo a simulação dinâmica de cada movimento. O computador permitiu aos animadores produzir seus projetos de forma nova e excitante.

Existem muitas aplicações nas quais a animação por computador é utilizada, além das indústrias de entretenimento e propaganda. Simuladores de voo têm se tornado cada vez mais sofisticados, apresentando imagens em tempo real em resposta a entradas do usuário. A visualização científica utiliza as técnicas de animação por computador para apresentar dados

complexos de uma forma mais compreensível e agradável. Além disso, a animação tem sido aplicada na educação, uma vez que muitos conceitos fundamentais podem ser facilmente explicados aos estudantes através da utilização de efeitos visuais envolvendo movimentos [FOLEY90][MAGTH85a][WYVIL90].

Este capítulo aborda as diversas classificações dos sistemas de animação por computador encontradas na literatura. O capítulo é voltado para a animação tridimensional, apresentando as três fases que a compõem, os tipos de objetos envolvidos e as várias técnicas de controle neste tipo de animação.

2.2. O Computador na Animação

Da mesma forma que é utilizado em diversas áreas da atividade humana, o computador possui diferentes papéis na animação [MAGTH85a]:

- **Criação de desenhos**

Os desenhos chaves podem ser digitalizados ou criados por um editor gráfico interativo. Objetos complexos podem ser produzidos por programação.

- **Criação de movimento**

Inbetweens podem ser completamente calculados, ou movimentos complexos podem ser gerados diretamente pelo computador.

- **Coloração**

Desenhos podem ser pintados usando um sistema interativo por computador, ou imagens realísticas e complexas podem ser geradas.

- **Filmagem**

Uma câmera de filmagem real pode ser controlada pelo computador, ou câmeras virtuais podem ser completamente programadas.

- **Pós-produção**

A edição e sincronização do som podem ser controladas pelo computador.

2.3. Classificação de Sistemas de Animação por Computador

Desde o surgimento do computador, sistemas foram desenvolvidos que o utilizam para simular tarefas humanas, com a finalidade de minimizar custos e tempo. Dessa forma, sistemas de animação por computador foram criados para facilitar as etapas da animação convencional, além de estimular o desenvolvimento de novas técnicas e metodologias da animação.

Existem diferentes maneiras de classificar os sistemas de animação por computador. Magnenat-Thalmann e Thalmann [MAGTH85a] classificam-nos por níveis:

- **Nível 1**

São basicamente editores gráficos utilizados por projetistas e somente para criar, pintar, armazenar, recuperar e modificar desenhos interativamente, como o editor para pintura *Paint* [SMITH82].

- **Nível 2**

São sistemas utilizados para ser usados por ou para substituir *inbetweeners*. Podem calcular *inbetweens* e mover objetos ao longo de uma trajetória, como a técnica apresentada por [BURTN82].

- **Nível 3**

São sistemas que permitem ao animador fazer operações sobre objetos, como translação e rotação. Também podem incluir operações de câmeras virtuais, como *zoom*, *pan* ou *tilt*.

- **Nível 4**

São sistemas que permitem um meio de definir atores, ou seja, objetos que possuem animação própria.

- **Nível 5**

São sistemas que podem ser ampliados e podem aprender à medida que trabalham. Em cada uso, tais sistemas podem se tornar mais poderosos e "inteligentes". Tais sistemas ainda não estão disponíveis.

Uma outra classificação, mais popular e simples, distingue os sistemas de animação por computador em duas categorias [CAMAR93][MAGTH85a]: animação *auxiliada por computador*, quando as máquinas apenas auxiliam os animadores na criação de desenhos, no cálculo de movimentos, na escolha de cores, no controle da câmera, entre outros (níveis 1 e 2); e animação *modelada por computador*, quando o animador determina os atores, ambientes e ações a serem executadas, cabendo ao computador o controle da animação como um todo (níveis 3 ou 4). A tabela 2.1 compara as funções de sistemas de animação auxiliada por computador com sistemas de animação modelada por computador.

Estas classificações estão baseadas na forma como os sistemas computacionais atuam no processo de criação da animação. Uma outra consideração é o modo de produção. A animação por computador em *tempo real* é limitada pela capacidade da máquina, sendo que a imagem deve ser mostrada a cada 1/15 segundo, pelo menos, para criar ilusão de movimento. Esta limitação é muito severa, já que muitos cálculos precisam ser realizados neste período de tempo. As restrições são impostas pela velocidade do ciclo do computador, capacidade de armazenamento, tamanho da palavra de memória e o conjunto de instruções. O outro modo de produção é *quadro-a-quadro*. Os quadros são calculados e armazenados previamente e projetados posteriormente a uma velocidade de 24 ou 30 quadros por segundo. O cálculo de um quadro pode levar alguns minutos ou várias horas, dependendo da complexidade das imagens.

	Auxiliada por Computador	Modelada por Computador
Criação de objetos	<ul style="list-style-type: none"> • Digitalização de personagens. • Uso de um editor gráfico. 	<ul style="list-style-type: none"> • Programas de reconstrução 3D. • Editor gráfico 3D. • Programas de modelamento de objetos 3D.
Movimento	<ul style="list-style-type: none"> • Cálculo de <i>inbetweens</i>. • Movimento ao longo de um caminho. 	<ul style="list-style-type: none"> • Programação de movimentos 3D. • Sistemas de atores
Coloração	<ul style="list-style-type: none"> • Sistemas de pintura. 	<ul style="list-style-type: none"> • Sistemas de tonalização 3D.
Câmera	<ul style="list-style-type: none"> • Controle físico da câmera. 	<ul style="list-style-type: none"> • Câmeras virtuais
Pós-Produção	<ul style="list-style-type: none"> • Sistemas de edição • Sincronização 	<ul style="list-style-type: none"> • Na teoria, estes sistemas de animação poderiam eliminar a fase de pós-produção através da atualização automática.

Tabela 2.1 Funções dos sistemas de animação por computador.

Os primeiros sistemas de animação por computador desenvolvidos eram específicos e utilizavam linguagens de programação ou eram sistemas interativos acessíveis somente a cientistas em computação. Mais tarde, foram desenvolvidos sistemas interativos amigáveis, permitindo que artistas, sem uma formação computacional, produzissem filmes de animação sem a necessidade da presença de uma pessoa com experiência em computação. Contudo, estes sistemas impõem restrições na criatividade daqueles que gostariam de explorar toda a capacidade de um computador. Dessa forma, linguagens de programação ainda desempenham um importante papel na animação.

Existem muitas linguagens diferentes para descrever uma animação e novas linguagens são constantemente desenvolvidas. Estas linguagens são divididas em três categorias, segundo Foley et al. [FOLEY90]:

- **Notação de listas lineares**

Na notação de listas lineares, cada evento na animação é descrito por um número inicial e final de quadro e uma ação a ser tomada (o *evento*). As ações tipicamente requerem parâmetros, tais como o exemplo abaixo

```
42, 53, B ROTATE "PALM", 1, 30
```

que significa que "*entre os quadros 42 e 53, rotacione o objeto chamado PALM em torno do eixo 1 de 30 graus, determinando a quantidade de rotação em cada quadro de acordo com a tabela B*". Outras linguagens baseadas na notação de listas lineares incorporam características de linguagens de programação de alto nível, como variáveis e fluxo de controle [HACK80] [TALB80].

- **Linguagens de propósito geral**

As animações podem ser descritas através de extensões de linguagens de propósito geral. Dessa forma, estruturas de dados e fluxo de controle podem ser utilizadas em rotinas para gerar animações. Na verdade, em alguns sistemas, as animações são efeitos colaterais de simulações geradas através da linguagem. Estas linguagens possuem grande potencial, mas exigem conhecimentos de programação por parte do usuário. O desenvolvimento das linguagens de programação tem levado a novos conceitos fundamentais no controle do movimento e de eventos temporais [MAGTH85a].

- **Linguagens gráficas**

Um dos problemas com linguagens de propósito geral é a dificuldade encontrada pelo animador em visualizar a animação apenas olhando o texto. Linguagens gráficas descrevem a animação de uma forma mais visual. Tais linguagens substituem o paradigma textual por um paradigma visual, ou seja, ao invés de escrever descrições das ações, o animador fornece um esboço da ação. Um dos primeiros trabalhos neste sentido foi feito por Baeck [BAECK80].

Os sistemas de animação por computador também podem ser diferenciados pelo modo de apresentação. Alguns sistemas utilizam técnicas de *frame buffers*, que conceitualmente podem ser vistos como sendo matrizes bidimensionais. Uma dessas técnicas é a *animação por manipulação de tabela de cores*. O valor de um pixel é geralmente um índice para uma posição em uma tabela de cores (*lookup table*). A animação é obtida modificando a tabela de cores, de

forma cíclica, alternada ou seletiva. Um bom exemplo, é a animação de encanamentos de água. Detalhes desta técnica podem ser encontrados em [SHOUP82].

Outra forma de apresentação é o *playback* em tempo real, na qual um sistema gera e armazena todos os quadros da animação previamente, os quais posteriormente são mostrados por um programa em tempo real. Com o desenvolvimento dos computadores e das interfaces gráficas, a animação também pode ser apresentada totalmente em tempo real, sem necessidade do armazenamento dos quadros.

2.4. Elementos da Animação Tridimensional por Computador

Da mesma forma que a animação tradicional com modelos requer a construção de um pequeno mundo em miniatura e o uso de câmeras *high-tech* com lentes periscópicas para fotografar os modelos, e o cálculo de cuidadoso de cada movimento de cada personagem, a animação tridimensional também requer a criação de um mundo tridimensional.

A animação tridimensional por computador é genericamente descrita como sendo composta por três fases [MAGTH85a] [WYVIL90]:

Figura 2.1 Uma taxonomia das técnicas de modelagem.

1. Modelagem de objetos

Consiste na descrição ou construção de objetos tridimensionais. A figura 2.1 apresenta uma taxonomia das principais técnicas de modelagem [WYVIL90].

Figura 2.2 Uma taxonomia das técnicas de controle de movimento.

2. Controle de movimento

A animação de um objeto consiste em fazê-lo mover, mudar sua forma ou, ainda, sua posição em um período de tempo. A figura 2.2 apresenta uma taxonomia das principais técnicas de controle de movimento [WYVIL90].

Figura 2.3 Uma taxonomia das técnicas de renderização.

3. Renderização

É o processo de produzir uma imagem realista através da remoção de superfícies escondidas e da adição de efeitos realísticos, como tonalização, transparência e textura.

A figura 2.3 apresenta uma taxonomia das principais técnicas de renderização [WYVIL90].

A segunda fase é fundamental para a animação por computador, sem a qual esta não existiria. As técnicas de modelamento e renderização estão além do escopo deste trabalho, e não serão tratadas aqui. As técnicas de controle de movimento são discutidas na seção 2.6.

2.5. Tipos de Objetos e seus Movimentos

Objetos variam de acordo com os movimentos que possuem. Tipicamente, o movimento é descrito em termos de *graus de liberdade*, ou seja, o número de coordenadas independentes necessárias para especificar as posições de todos os componentes do sistema [WILH87]. Para um sistema com n graus de liberdade mostrados em t quadros de animação, um total de $n \times t$ números devem ser especificados. Um exemplo realista é uma animação de vídeo a 30 quadros por segundo. Uma animação de três minutos de um objeto com um total de 50 graus de liberdade requer 4500 números para especificar completamente o movimento. Naturalmente, utilizando-se as técnicas de *keyframe* e de interpolação é possível reduzir a percentagem de números que o animador deve especificar, mas a quantidade de dados necessários para controlar o movimento ainda é considerável. Entre os tipos de objetos possíveis, tem-se:

- **Partículas**

Uma partícula pode ser descrita por um ponto no espaço tridimensional (x, y, z) . O posicionamento e movimento de tal ponto são descritos por três variáveis e, portanto, um ponto tem três graus de liberdade de movimento. A animação de um ponto requer uma tripla de números para cada quadro da animação, ou três funções descrevendo a variação de x , y e z em relação ao tempo.

- **Corpos rígidos**

Um corpo rígido é definido por um número de pontos que devem ser movimentados juntamente. Um ponto não pode se mover em relação ao outro, mas os pontos movem-

se como um todo, como, no caso de pontos que definem polígonos e superfícies de forma livre. O movimento de um corpo rígido é especificado por seis graus de liberdade, translação e rotação em x , y e z .

- **Corpos flexíveis**

Um corpo flexível consiste de um número infinito de pontos que se movem relativamente uns aos outros em relação ao tempo. Na prática, um corpo flexível pode ser definido através de um conjunto de pontos, como, por exemplo, os vértices de um polígono ou os pontos de controle de superfícies. O movimento de um corpo flexível definido como uma superfície de forma livre usando p pontos de controle tem $3p$ graus de liberdade em relação ao tempo. Dessa forma, explica-se porque muitas animações por computador utilizam corpos rígidos.

- **Corpos articulados**

Corpos articulados são compostos por segmentos cujos movimentos em relação aos outros segmentos obedecem a certas restrições. Por exemplo, o corpo humano é freqüentemente representado como segmentos rígidos unidos por articulações (juntas) que possuem três graus de liberdade. O número total de graus de liberdade que deve ser especificado é a soma dos números dos graus de liberdade de cada junta.

Corpos articulados também podem ser flexíveis. Neste caso, a junta entre dois segmentos flexíveis pode ser modelada como uma junta entre dois sistemas de coordenadas, cada um correspondendo a um dos segmentos adjacentes. Dois tipos de movimento são possíveis. O movimento na articulação consiste na mudança de seis números especificando a relação entre os sistemas. O movimento dentro de cada segmento flexível consiste do movimento dos pontos que definem o segmento, relativamente ao sistema local. O número total de graus de liberdade necessários para especificar um corpo articulado flexível é o número de graus de liberdade de cada junta, mais três vezes o número dos pontos que definem os segmentos.

- **Outros objetos e corpos de metamorfoses**

Corpos podem ser também especificados algebricamente. Uma esfera pode ser modelada com um centro (três números) e um raio, e uma esfera pulsante pode ser animada mudando-se o raio. O número de graus de liberdade depende da natureza das equações definindo estes objetos algébricos.

O movimento se torna mais complexo quando os corpos sendo modelados mudam seus graus de liberdade em relação ao tempo. Exemplos de corpos de metamorfose incluem a formação de uma cadeia de montanhas fractais, crescimento de plantas e sistemas de partículas.

Como a complexidade do movimento depende do número de graus de liberdade, restrições impostas aos objetos podem diminuir essa quantidade [WILH87]. Por exemplo, uma junta articulada em um joelho pode ser modelada como tendo somente um grau de liberdade de rotação. Restrições também podem ser aplicadas com limites dentro de um grau de liberdade. Por exemplo, uma cabeça pode girar em relação ao pescoço, mas não pode fazer um giro de 360 graus.

Estas restrições parciais podem ser difíceis de modelar, uma vez que os limites de um grau de liberdade podem variar de acordo com a posição em relação aos outros graus de liberdade. Por exemplo, um braço não pode balançar na direção do centro do corpo quando o corpo estiver ereto. Porém, se o corpo estiver curvado para frente, este movimento é possível.

Restrições podem ser aplicadas ocasionalmente, como no movimento de um pulo, no qual as pernas são algumas vezes restritas ao chão. As restrições podem assumir muitas formas. Uma mão de robô pode ser restrita ao contato com uma superfície ou mover-se em relação a um ponto. As restrições também podem ser aplicadas à velocidade e à aceleração e a natureza das restrições pode variar com o tempo.

Um sistema de controle de movimento deve levar em conta as restrições ou, pelo menos, fazer a verificação das restrições e responder razoavelmente quando estas são violadas. Um

resposta razoável pode informar ao animador a violação da restrição e possivelmente incluir uma recusa na realização do movimento.

2.6. Técnicas de Controle de Movimentos

O campo do controle de movimento na animação por computador é ainda muito novo, embora várias técnicas tenham sido desenvolvidas. Como ilustra a figura 2.2, existe uma grande variedade de técnicas utilizadas no controle do movimento e muitas delas são combinações de outras técnicas. Definir o movimento para cenas tridimensionais complexas é complicado e consome tempo, devido à dificuldade em escolher entre uma grande variedade de técnicas e à enorme quantidade de informação que deve se especificada [WILH87][WYVIL90].

Grosseiramente, existem duas classes de controle de movimento [WYVIL90]: *simulação* e *ilusão*. Para parecer natural, a animação tem de representar o movimento do mundo físico. Neste sentido, a melhor animação é baseada em simulações detalhadas que consideram a dinâmica do movimento. Numa simulação, um modelo matemático representando as leis físicas que governam o movimento produziria, como consequência, o efeito desejado. Entretanto, a menos que a animação esteja voltada para visualizar os resultados de uma simulação, a animação só precisa *parecer* realista, criando uma ilusão de movimento.

2.6.1. Manipulação Direta

Esta técnica de animação é o tipo mais simples de controle de movimento. Aqui, o animador fornece uma descrição completa de tudo que ocorre na animação, ou especificando mudanças simples, tais como escalas, translação e rotação, ou fornecendo informações de *keyframes* e métodos de interpolação para gerar os *inbetweens*.

2.6.2. Interpolação de *Keyframes*

Na animação convencional, um animador desenha dois *keyframes* e então produz os *inbetweens*. Uma técnica computacional equivalente a este processo consiste em interpolar os

keyframes. Um método simples seria a utilização de interpolação linear entre os *keyframes* e até mesmo entre os objetos da cena. A figura 2.4 ilustra os princípios utilizados para criar *inbetweens* através da interpolação linear com início no objeto ABCD e término no objeto $A_1B_1C_1D_1$. Quando as imagens inicial e final não possuem o mesmo número de vértices, é necessário adicionar ou remover vértices extras (*metamorfose*).

Figura 2.4 Interpolação linear para *keyframes*.

A interpolação linear produz efeitos indesejáveis, tais como falta de suavidade no movimento, descontinuidade na velocidade do movimento e distorções na rotação. Métodos alternativos foram propostos, mas não existe uma solução totalmente satisfatória para os efeitos colaterais da interpolação de *keyframes* [THALM89]. Um dos métodos mais simples é a utilização de polinômios cúbicos.

A interpolação de *keyframes*, muito utilizada na animação bidimensional, pode ser estendida para objetos tridimensionais. O princípio é o mesmo quando os objetos são modelados em *wireframe*. Entretanto, a técnica é muito mais complexa quando os objetos são baseados em faces, uma vez que uma correspondência entre o número de faces e o número de vértices deve ser encontrada.

2.6.3. Interpolação Paramétrica

Sistemas paramétricos usam a interpolação de *keyframes* em um nível mais elevado que sistemas de *keyframes*. Cada objeto é definido por um conjunto de parâmetros e os valores destes parâmetros são interpolados entre os *keyframes*. Se um modelo é definido por *patches* de superfícies, por exemplo, os pontos de controle podem ser interpolados e os *patches* podem ser refeitos para os novos dados.

Tão importante como garantir que o movimento seja suave através do tempo, é garantir que a câmera também execute um movimento suave através do espaço. Usando uma interpolação com splines, o animador pode colocar a câmera em um certo número de *keyframes* e então interpolar suavemente entre eles, através do espaço e tempo. Um controle similar pode ser aplicado a qualquer objeto da cena.

2.6.4. Sistemas de *Script*

A animação por *keyframes* tem a vantagem de que um animador pode interagir com o movimento e especificar movimentos por manipulação direta. Entretanto, muitas animações são algorítmicas por natureza e difíceis de serem especificadas interativamente. Por esta razão, muitos sistemas permitem ao usuário especificar a animação através de *scripts*. Um *script* é uma sequência de comandos capazes de produzir um movimento. Entre estes sistemas incluem-se os que utilizam as notações de listas lineares e as diversas linguagens que foram criadas especificamente para animação, como CINEMIRA e MIRANIM [MAGTH85a].

Uma forma mais poderosa, porém em um nível mais baixo que os sistemas de *scripting*, é a extensão das linguagens de programação de propósito geral, que pode fornecer um ambiente no qual a animação algorítmica é especificada facilmente, oferecendo também o poder de uma linguagem de programação.

2.6.5. *Rotoscoping*

Quando a animação dos movimentos humanos foi primeiramente tentada, os dados dos movimentos eram freqüentemente obtidos através da gravação de um movimento real (seção 1.1). Baseando-se nesta técnica, o *rotoscoping* é um método no qual as coordenadas dos membros eram obtidas manualmente através de pelo menos duas visões ortogonais de uma cena previamente gravada (em filme ou vídeo). Embora fosse uma técnica tediosa, foi o começo da análise do movimento humano na animação computacional. Mais tarde, outras técnicas foram desenvolvidas, utilizando goniômetros e equipamentos de vídeo de *scanning* para registrar os

movimentos tridimensionais. Os resultados permitiram a produção das primeiras seqüências razoáveis de animação de movimentos humanos.

2.6.6. Controle por Procedimentos ou Procedimental

Esta técnica é derivada da modelagem por procedimentos, na qual vários elementos de um modelo comunicam-se entre si com a finalidade de determinar suas propriedades. Os sistemas de partículas são bons exemplos da utilização desta técnica. Por exemplo, a proximidade das partículas de vento determina a posição das folhas da grama em um cenário de campo aberto. Assim, o sistema de partículas descrevendo a grama é afetado por aspectos de outros objetos da cena. Este tipo de interação por procedimentos entre objetos pode ser utilizado para gerar movimentos que podem ser difíceis de serem produzidos por outras técnicas. Por outro lado, exige que o animador tenha conhecimentos de programação.

2.6.7. Cinemática Inversa

Na *cinemática direta*, o movimento é obtido através da especificação da posição em relação ao tempo. Por exemplo, quando se quer coçar a orelha, na cinemática direta, as seqüências de posições da mão, do pulso e do cotovelo devem ser fornecidas pelo animador. Ao invés de calcular a posição de um segmento (mão) com base nas rotações da junta (pulso e cotovelo), na *cinemática inversa* as rotações da juntas são calculadas com base na posição do segmento [WILH87]. Por exemplo, para coçar a orelha, deve-se mover a mão para a orelha. Mas, quando a mão chega na orelha, o cotovelo pode estar em qualquer uma de diversas posições (perto ou longe do corpo). Assim, os movimentos de levantar e abaixar o braço e o pulso não são completamente determinados pela instrução *mover a mão para a orelha*. A cinemática inversa permite uma especificação de movimento mais natural.

A cinemática inversa fornece um método para restringir corpos em relação ao seu mundo [WILH87]. Por exemplo, é utilizada para manter um corpo realisticamente em contato com o chão quando ele se move. A tendência neste tipo de movimento é que o balanço do movimento faça com que os pés do corpo fiquem abaixo do nível do chão.

O problema de alcançar alvos, freqüente em robótica, também pode ser resolvido através da cinemática inversa, sendo que os cálculos das posições das articulações são feitos a partir do posicionamento que o braço do robô deve alcançar.

Os problemas de cinemática inversa são difíceis de serem resolvidos, pois envolvem equações complexas que podem resultar em múltiplas soluções. É bem mais fácil resolver equações com solução única do que equações com diversas soluções. Para isto, restrições (por exemplo, a energia potencial do braço deve ser a mínima possível para cada fase do movimento) devem ser impostas para que a solução se torne única.

A cinemática inversa tem sido largamente utilizada na descrição de movimentos de corpos humanos, na tentativa de torná-los reais. Entretanto, ainda não é uma solução adequada, por exemplo, para a criação de movimentos humanos no computador [WYVIL90].

2.6.8. Análise Dinâmica

Na cinemática, o movimento é especificado pelas posições em relação ao tempo. Porém, ela não leva em consideração as causas do movimento, que são afetados por forças e torques agindo sobre massas. A *dinâmica* leva em conta as leis físicas que governam a cinemática.

A dinâmica tem sido utilizada no modelamento computacional em poucos casos. Alguns sistemas CAD/CAM incluem pacotes de engenharia para análise dinâmica de máquinas. Estudos do comportamento de veículos podem incluir simulações dinâmicas envolvendo a computação gráfica [ADAMS92].

As equações de movimento dinâmico são utilizadas para relacionar a aceleração da massa (objeto) às forças e torques agindo sobre essa massa. As forças produzem movimentos de translação, enquanto que os torques produzem movimentos de revolução. A segunda lei de Newton é a base para a simulação usada na dinâmica:

$$\mathbf{F} = m \cdot \mathbf{a}$$

onde $\mathbf{F}$ é o vetor força agindo sobre a massa m , e $\mathbf{a}$ é a aceleração dessa massa. Para massas constantes, esta equação pode ser resolvida de duas maneiras: (1) dada a força, pode-se obter a aceleração; (2) dada a aceleração, pode-se obter a força.

O segundo modo de resolução da equação é geralmente de interesse da robótica (*dinâmica inversa*), já que o movimento desejado é conhecido e a quantidade de força que deve ser aplicada aos motores dos robôs para produzir este movimento deve ser determinada. Para a computação gráfica, o primeiro modo de resolução é mais interessante (*dinâmica direta*): dadas as forças e torques internos e do ambiente, deve-se determinar a aceleração que o objeto sofrerá. Dada a aceleração, a posição atual e a velocidade, o caminho percorrido no movimento pode ser encontrado.

Para simular um sistema dinâmico são necessárias *restrições dinâmicas*, que são forças ajustadas para agir sobre o objeto, de tal forma a manter algumas condições. Em sistemas simples, a integração analítica é utilizada. À medida que a complexidade do sistema aumenta, técnicas de integração numérica são necessárias para se obter os resultados.

2.7. Considerações Finais

Como em outras áreas, o computador tem desempenhado um importante papel no desenvolvimento da animação. Este capítulo apresentou um resumo das possíveis classificações de sistemas computacionais de animação, bem como uma visão dos elementos da animação tridimensional por computador. Uma animação tridimensional é composta de três fases: modelamento, controle de movimento e renderização, sendo que neste capítulo foram enfatizadas as técnicas tridimensionais de controle de movimento. As demais fases estão além do escopo do trabalho proposto.

O próximo capítulo apresenta uma das possíveis aplicações da animação por computador na visualização de resultados de simulações de engenharia, mais especificamente a simulação de sistemas mecânicos multicorpos.

Capítulo 3

Sistemas Multicorpos

3.1. Considerações Iniciais

A animação por computador é muito útil no processo de desenvolvimento de projetos em engenharia e vem sendo utilizada em diversas ferramentas de *Engenharia Apoiada por Computador*, ou *Computer-Aided Engineering* (CAE). A animação oferece uma ferramenta de apoio comumente utilizada na fase de pós-processamento em sistemas computacionais de engenharia. Entretanto, essa tendência está sendo modificada com o desenvolvimento de novos equipamentos [PORT94] que têm permitido que animações deixem de ser utilizadas unicamente em atividades de pós-processamento para tornarem-se realmente uma ferramenta de desenvolvimento.

A engenharia mecânica tem utilizado muito os computadores no processo de desenvolvimento de projetos, desde a geração automática de desenhos de projeto até a simulação de possíveis produtos mecânicos. Uma das técnicas de modelamento empregada em sistemas mecânicos é a técnica de *Sistemas de Multicorpos* ou *Multibody Systems* (MBS), implementada em vários sistemas de *software* disponíveis comercialmente.

Neste capítulo, apresentamos a terminologia utilizada em sistemas mecânicos multicorpos e exemplos de pacotes de *software* que utilizam a técnica de modelamento MBS para o desenvolvimento de produtos.

3.2. Técnicas de Modelamento de Sistemas Multicorpos

Esta seção apresenta um resumo das técnicas de modelamento de sistemas de multicorpos apresentadas por Costa [COSTA92].

Um MBS pode ser definido como um sistema mecânico com muitos graus de liberdade [COSTA92]. O movimento de um MBS é governado por equações denominadas *equações dinâmicas de movimento*, que compreendem um conjunto de equações diferenciais, juntamente com algumas equações algébricas. As equações diferenciais são expressões de leis físicas (Leis de Newton) que descrevem o movimento de corpos rígidos e as equações algébricas exprimem as restrições da geometria do sistema ou de seus movimentos, como por exemplo sistemas de articulações conectando corpos adjacentes e características particulares do contato entre os corpos.

As técnicas de modelamento MBS podem ser utilizadas na análise e projeto de qualquer sistema mecânico que possa ser modelado como um conjunto de corpos rígidos interconectados por juntas, influenciados por forças, dirigidos por movimentos determinados e limitados por restrições. Exemplos de sistemas mecânicos que podem ser estudados através destas técnicas incluem naves espaciais articuladas, veículos terrestres, mecanismos e máquinas, manipuladores, dispositivos eletromecânicos de alta velocidade, entre outros. A figura 3.1 ilustra alguns exemplos de sistemas mecânicos modelados pelas técnicas de MBS.

Qualquer conjunto de corpos rígidos conectados por juntas pode ser considerado topologicamente equivalente a uma "árvore" (cadeia de corpos com ramos) que pode ter alguns ramos conectados para formar laços. Várias juntas de rotação ou de translação conectam esses corpos. A topologia de um sistema pode ser *aberta* (árvore) ou *fechada* (laço). Além disso, os sistemas podem ter *livre movimento*, como satélites, ou ser *presos ao chão*, como veículos. A figura 3.2 ilustra algumas possíveis topologias dos sistemas mecânicos.

Figura 3.1 Exemplos de sistemas mecânicos que podem ser modelados pelas técnicas MBS: (a) mecanismos e máquinas; (b) veículos terrestres; (c) veículos espaciais articulados; (d) engrenagens; (e) manipuladores; (f) cabos; (g) dispositivos eletromecânicos de alta velocidade.

Figura 3.2 Topologias de sistemas MBS: (a) árvores abertas; (b) *loops*.

Figura 3.3 Tipos de juntas: (a) junta de pino; (b) junta de translação; (c) junta esférica; (d) junta soldada; (e) junta universal; (f) junta *gimbal*; (g) junta planar; (h) junta cilíndrica; (i) junta *sixdof*.

As juntas podem ser baseadas na construção de três primitivas: juntas de rotação unidimensional (juntas de pino), juntas de rotação bidimensional (juntas de translação) e juntas de rotação tridimensional (juntas esféricas). Estas três primitivas podem ser utilizadas para construir outros tipos de juntas, tais como *juntas universais* (junta de rotação bidimensional com um eixo articulável), *juntas gimbal* (junta de rotação tridimensional), *juntas sixdof*, (juntas livres ou juntas com seis graus de liberdade utilizando três juntas de translação e uma junta esférica), *juntas cilíndricas* (combinação de juntas de translação e de pino), *juntas planares* (duas juntas de translação mais uma junta de pino) e *juntas soldadas* (uma junta com zero grau de liberdade, muito útil para fechar certos tipos de laços). A figura 3.3 ilustra os tipos de juntas encontradas em sistemas mecânicos.

Em geral, as *forças* e *torques* (que são freqüentemente chamados de *cargas*) que influenciam os sistemas mecânicos podem ser de três tipos, ilustrados na figura 3.4.

Figura 3.4 Origem das forças que agem sobre os sistemas mecânicos: (a) amortecedores; (b) molas; (c) gravidade; (d) motores e máquinas; (e) controle de realimentação; (f) controle direto; (g) pressão; (h) pneus.

- **Aplicadas externamente**

Forças externas podem ser aplicadas em qualquer posição em qualquer corpo. Do mesmo modo, torques externos podem ser aplicados em qualquer corpo. Exemplos de cargas incluem impulsores, pneus, fricção externa, forças de fluidos e molas.

- **Aplicadas internamente**

Forças internas e torques podem ser aplicadas em qualquer eixo de junta. Exemplos de cargas incluem sistemas de motores, acionadores hidráulicos, molas dobráveis e amortecedores.

- **Gravitacional**

Sistemas mecânicos podem ser influenciados por um campo gravitacional uniforme ou não-uniforme.

Os tipos de juntas de qualquer sistema mecânico restringem seus movimentos. Além disso, os movimentos podem ser determinados por fatores internos, como motores, ou externos,

como perfil de estradas, movimentos orbitais, entre outros. A figura 3.5 ilustra os fatores que restringem os movimentos de um sistema mecânico.

Figura 3.5 Restrições nos modelamentos MBS: (a) engrenagens; (b) distância; (c) trilhos; (d) correias e polias.

As técnicas de modelamento MBS podem ser classificadas de várias formas. Uma classificação comumente adotada divide as técnicas de acordo com o formalismo utilizado para descrever o sistema e a natureza da implementação computacional do algoritmo relacionado. Maiores detalhes podem ser obtidos em [COSTA92].

3.3. Implementação Computacional

No passado, a dedução das equações dinâmicas dos sistemas mecânicos era realizada manualmente. Entretanto, com o aumento da complexidade destes sistemas, este processo tornou-se incômodo, tedioso e propenso a erros. Além disso, mudanças de projeto, mesmo aquelas com simples variações, ficavam difíceis de serem aplicadas. Por estas razões, programas de computadores para gerar equações de movimentos para MBS começaram a ser desenvolvidos. O desenvolvimento do modelamento MBS tem origem nas áreas de máquinas e mecanismos e de pesquisa espacial. Somente mais tarde as áreas de transporte terrestre, veículos, estradas e ferrovias passaram a utilizar essa técnica de MBS.

Formalismos matemáticos foram desenvolvidos para que as equações de movimentos para um MBS pudessem ser geradas por computador baseando-se em dados simples sobre os corpos (topologia e inércia) e suas interconecções (restrições cinemáticas e leis de forças). O comportamento dinâmico de sistemas mecânicos, com velocidades envolvidas de pequena ordem de magnitude, pode ser adequadamente descrito através das Leis de Movimento de Newton, provenientes da mecânica clássica. Estas equações podem ser descritas através de equações de Newton-Euler e de Lagrange, conhecidas como formalismos *sintéticos* e *analíticos*, respectivamente. Os formalismos sintéticos, também denominados *métodos Eulerianos*, são baseados em equações que descrevem o equilíbrio de forças em um sistema. Já os formalismos analíticos, também conhecidos como *métodos Lagrangeanos*, descrevem o equilíbrio da energia dentro de um sistema. Além disso, Kane desenvolveu um método para derivar equações dinâmicas que tornou-se a base teórica do modelamento MBS, sendo implementado em muitos sistemas [COSTA92].

Em relação à implementação computacional, programas MBS podem ser classificados como *numéricos*, *simbólicos* ou uma combinação de ambos. O termo simbólico é um tanto errôneo, já que o código resultante não é, em termos computacionais, realmente simbólico, mas uma subrotina em FORTRAN, C ou Pascal.

Programas numéricos podem possuir código especializado, desenvolvido especificamente para a simulação de um modelo ou de uma classe de modelos, ou código genérico. Os códigos genéricos são desenvolvidos em torno de uma forma geral de equações de movimento, capazes de simular sistemas com uma grande variedade de topologias, permitindo corpos com distribuição de massas arbitrária, conectados em pontos arbitrários por diversos tipos de juntas. Códigos deste tipo são geralmente enormes e possuem execução lenta, comparando com os códigos especializados. Exemplos de programas numéricos que trabalham com os formalismos sintético e analítico, respectivamente, são o SD/FAST [SDFAST91] e o ADAMS [ADAMS92].

Programas que utilizam técnicas de manipulação simbólica surgiram da necessidade de gerar código eficiente, específico para um problema, mas que fossem capazes de simular uma ampla classe de sistemas, evitando a ineficiência de executar uma simulação genérica completa

para todas as configurações dos sistemas. Programas de manipulação simbólica de propósito geral, tais como REDUCE [RAYNA87], foram utilizados para gerar equações de movimentos [COSTA92]. Entretanto, estes programas continham muitas funções além das necessárias para gerar as equações e exigiam técnicas especiais, como alocação de memória dinâmica, para gerar um código mais eficiente. No sentido de superar estes problemas, métodos específicos de manipulação simbólica foram desenvolvidos. Um exemplo de programa com manipulação simbólica é o SD/FAST.

Figura 3.6 Três modos de gerar uma simulação MBS[SDFAST91]: (a) código de propósito específico desenvolvido por engenheiros; (b) programas MBS convencionais; (c) código de propósito específico desenvolvido por manipulação simbólica.

A figura 3.6 ilustra uma comparação entre os programas MBS de geração por manipulação simbólica e outros métodos. Manualmente, os engenheiros trabalham com a especificação dos parâmetros do sistema para desenvolver equações explícitas do movimento (figura 3.6a). Estas equações são então codificadas e incorporadas a um grande *framework* de simulação, como SIMULINK [MATLAB93]. Para um código de propósito geral, os parâmetros

específicos do sistema são fornecidos ao código MBS em tempo de execução, e são utilizados para gerar um modelo genérico de multicorpos para produzir as respostas dinâmicas do sistema (figura 3.6b). Um problema sério do código de propósito geral de multicorpos é o enorme tempo de execução, uma vez que simplificações encontradas em qualquer sistema físico não são exploradas para reduzir a complexidade das equações. Por outro lado, os códigos simbólicos, ao invés de ser uma substituição para as equações de movimentos geradas na fase de desenvolvimento, são uma substituição dos *engenheiros*, no sentido de que são capazes de produzir o conjunto mais simplificado possível de equações de movimentos, a partir de uma descrição e, a seguir, gerar um código específico empregando todas as informações do sistema (figura 3.6c).

3.4. Considerações Finais

Este capítulo apresentou um pequeno resumo da terminologia utilizada na técnica de modelagem de sistemas multicorpos, bem como uma visão geral das diversas implementações computacionais de ambientes de simulação desses sistemas. O modelamento matemático envolvido na geração de equações de movimentos de sistemas mecânicos não foi apresentado por estar além do escopo do trabalho proposto. Contudo, Costa [COSTA92] apresenta os fundamentos matemáticos utilizados na geração de equações.

O próximo capítulo apresenta o sistema SD/FAST para simulação de sistemas MBS, cujo dados gerados fornecem a entrada para o sistema de animação proposto e implementado neste projeto.

Capítulo 4

Sistema de Simulação SD/FAST

4.1. Considerações Iniciais

O SD/FAST, desenvolvido pela Symbolic Dynamics Inc.[SDFAST91], é um sistema de simulação utilizado no apoio à análise e projeto de sistemas mecânicos que possam ser modelados como MBS. Este sistema gera códigos otimizados de subrotinas da simulação a partir de uma descrição dos corpos rígidos, suas conexões e tipos das juntas que compõe o sistema. O usuário do SD/FAST é responsável pela criação do programa principal que acessa as subrotinas geradas pelo SD/FAST, cuja execução resulta na execução da simulação. A escolha dos resultados a serem gerados e o controle da simulação ficam a critério do próprio usuário. O programa principal deve ser escrito obedecendo a certas regras, pois o SD/FAST pode ser visto como uma "máquina de estados". Determinadas chamadas de rotinas somente têm sentido se o SD/FAST estiver no "estado" correto. O sistema métrico utilizado é definido pelo próprio usuário, que deve especificar os valores de forma coerente com o sistema métrico adotado.

Na seção 4.2 é apresentado o formato de um arquivo típico de entrada para o SD/FAST. A seguir, na seção 4.3 são ilustrados os tipos de juntas suportados pelo sistema. A seção 4.4 descreve o vetor de estados da simulação, que contém informações importantes sobre a mesma. Na seção 4.5 são descritos os arquivos gerados pelo SD/FAST, que devem ser compilados e *linkeditados* com o programa principal. Finalmente, na seção 4.6, um exemplo de utilização do

SD/FAST é apresentado. Os apêndices A e B apresentam, respectivamente, a sintaxe do *arquivo de descrição do sistema* e as rotinas geradas pelo SD/FAST.

4.2. Arquivo de Descrição do Sistema

Fundamentalmente, o sistema SD/FAST é um gerador de código de subrotinas em C ou FORTRAN, com base em um **arquivo de descrição do sistema**. Neste arquivo, o usuário deve descrever cada corpo (e suas propriedades) e cada junta (e suas propriedades) que conecta este corpo com o corpo anterior, para formar uma árvore do MBS, como descrito na seção 3.1.

O arquivo de descrição do sistema MBS possui sintaxe e semântica próprias (Apêndice A), e está dividido em quatro seções:

- **Preâmbulo**

Consiste de especificações que afetam o sistema como um todo.

- **Corpos e juntas da árvore**

Consiste na descrição de cada corpo do sistema e sua junta de conexão ao corpo anterior.

- **Juntas de *Loop***

Consiste na descrição de juntas de loop do sistema. Estas juntas são utilizadas para restringir determinados movimentos.

- **Restrição do usuário**

Consiste na declaração das restrições do sistema impostas pelo usuário.

Com exceção da seção **Corpos e Juntas da Árvore**, as demais seções podem ser omitidas do *arquivo de descrição do sistema*.

4.2.1.Preâmbulo

Nesta seção da descrição, valores globais que afetam o MBS como um todo podem ser especificados, como a gravidade que é exercida no MBS (**gravity**), a linguagem de programação a ser gerada (**language**), a precisão dos cálculos (**single** ou **double**) e o prefixo do MBS (**prefix**), que define o prefixo das subrotinas SD/FAST.

Cada corpo possui um referencial situado no seu centro de massa e, inicialmente, os eixos deste referencial estão alinhados com um referencial global, geralmente localizado no corpo pré-definido **ground**. No SD/FAST, toda geometria, propriedades de massa e localização das juntas e sua orientação estão definidos em relação ao referencial local de cada corpo. A figura 4.1 mostra uma representação dos referenciais de cada corpo no instante inicial e em algum movimento subsequente. A rotação destes referenciais obedece à regra da mão direita para rotações positivas.

Figura 4.1 Estado dos referenciais durante uma simulação: (a) estado inicial, na qual cada referencial de cada corpo está alinhado ao referencial global **N**; (b) uma instância durante uma simulação.

4.2.2.Corpos e Juntas da Árvore

Nesta seção da descrição, cada corpo (**outboard body**) é definido pela palavra-chave **body**, e é conectado ao corpo anterior (**inboard body**) definido pela palavra-chave **inboard**, através da junta da árvore **joint**. A massa (**mass**) e a matriz de inércia (**inertia**) do corpo devem ser especificadas, além dos vetores **bodyToJoint** e **inbToJoint**. O vetor **bodyToJoint** é

definido em relação ao referencial do corpo **body** e **inbToJoint** é definido em relação ao referencial **inboard**, como mostra a figura 4.2.

Figura 4.2 Vetores **bodyToJoint** e **inbToJoint**.

Através destes vetores, é possível localizar a posição do **centro de massa** do corpo em relação ao referencial do corpo anterior:

$$\vec{com} = \vec{inbToJoint} - \vec{bodyToJoint} \quad (\text{Equação 4.1})$$

Portanto, o centro de massa de um corpo em relação ao referencial global é dado por:

$$\vec{com}_{global} = \sum (\vec{inbToJoint} - \vec{bodyToJoint}) \quad (\text{Equação 4.2})$$

para todos os corpos conectados entre o corpo em estudo e o corpo **ground**.

Figura 4.3 Eixos genéricos de uma junta.

Cada eixo (**pin**) da junta deve ser especificado no arquivo de descrição. O sistema SD/FAST interpreta estes eixos como sendo fixos no *inboard body*. O *outboard body* possui somente um eixo, o qual está sempre alinhado ao último eixo do *inboard body*, como mostra a figura 4.3.

Se o movimento é prescrito (**prescribed**), isto é, determinado pelo usuário, o número de prescritos deve ser igual número de graus de liberdade (**degrees of freedom** ou **dof**) de cada junta.

4.2.3. Juntas de *Loop*

Esta difere da seção anterior pelo fato do corpo definido na palavra-chave **body** já existir. Ao contrário das juntas da árvore, que sempre conectam os corpos, as juntas de *loop* podem não estar inicialmente conectando dois corpos. Neste caso, outros valores são necessários para definí-las (**bodypin**, **bodyref** e **inbreff**). Além disso, existem várias restrições sobre a perpendicularização dos eixos destes tipos de juntas [SDFAST/91].

4.2.4. Restrição do Usuário

Esta seção permite ao usuário impor restrições ao movimento do MBS sendo simulado, com o objetivo de fazer uma análise mais realista do mesmo. O SD/FAST trabalha com três tipos de restrições:

- **Restrição da junta**

Os movimentos dos corpos estão condicionados aos movimentos permitidos pela junta.

- **Movimento prescrito**

Ao contrário de permitir que o movimento dos corpos ocorra devido ao movimento natural da junta, o usuário pode definir um movimento de tal forma que a junta fique sempre em uma determinada posição. Isto permite simular *dinâmica inversa*, na qual o movimento é prescrito por forças e torques necessários para o cálculo do movimento.

- **Equações de restrição genéricas**

Permite ao usuário descrever restrições mais genéricas como engrenagens, trilhas ou atrito. Estas restrições podem ser ligadas e desligadas durante a simulação.

Uma análise mais detalhada de cada seção do *arquivo de descrição do sistema* está além do escopo deste trabalho. Maiores detalhes podem ser obtidos no manual do usuário do SD/FAST [SDFAST91].

4.3. Juntas

O sistema SD/FAST trabalha com nove tipos de juntas, ilustradas na figura 3.3, que podem ser combinadas entre si para criar tipos adicionais. Os nove tipos básicos são:

1. **Pin**: junta de grau de liberdade 1 e um eixo de rotação.
2. **Slider**: junta de grau de liberdade 1 e um eixo de translação.
3. **Cylindrical**: junta de grau de liberdade 2 e um eixo de translação e rotação.
4. **Universal (Hooke's joint)**: junta de grau de liberdade 2 e dois eixos de rotação.
5. **Planar**: junta de grau de liberdade 3, com dois eixos de translação e um eixo de rotação.
6. **Gimbal**: junta de grau de liberdade 3 e três eixos de rotação.
7. **Ball**: junta de grau de liberdade 3 e nenhum eixo.
8. **Sixdof**: junta de grau de liberdade 6, com três eixos de translação e uma junta **ball**.
9. **Weld**: junta de grau de liberdade 0.

Somente a junta **weld** não pode ser utilizada como junta da árvore. Maiores detalhes sobre os tipos de juntas podem ser encontrados no manual do usuário do SD/FAST [SDFAST91].

4.4. Vetor de Estados

O SD/FAST utiliza um vetor de estados para representar a posição e a velocidade do MBS em estudo a cada iteração da simulação. Os elementos do vetor de estados são denominados de **coordenadas generalizadas** e **velocidades generalizadas**. As coordenadas generalizadas especificam o centro de massa e a orientação espacial de cada corpo durante a simulação. As velocidades generalizadas determinam a velocidade linear do centro de massa de cada corpo, e também a velocidade angular do corpo.

O tamanho do vetor de estados varia conforme o número de corpos e juntas da árvore do MBS. Mais precisamente, depende do número de equações geradas na simulação. Juntas do tipo *loop* possuem um vetor próprio denominado **pseudo-estados**. Esse tipo de junta influencia no comportamento do sistema MBS, impondo restrições ao movimento.

O vetor de estados é implementado como um vetor de valores em ponto flutuante. Cada índice deste vetor tem significado próprio, dependendo do tipo da junta. Além disso, juntas do tipo BALL e SIXDOF utilizam parâmetros de Euler [SDFAST91] para especificar a orientação do corpo no espaço. Maiores detalhes sobre o vetor de estados podem ser obtidos no manual do usuário do SD/FAST [SDFAST91].

4.5. Arquivos Gerados pelo SD/FAST

Depois de analisar sintática e semanticamente o *arquivo de descrição do sistema*, o SD/FAST gera quatro arquivos:

- **Arquivo de Dinâmica**

Contém todas as equações e códigos específicos do sistema.

- **Arquivo de Análise Simplificada**

Contém análises padrões específicas para o sistema, voltadas para operações básicas do SD/FAST.

- **Arquivo de Informação**

Contém informações específicas do sistema MBS.

- **Arquivo de Biblioteca do SD/FAST**

Contém rotinas puramente numéricas e independentes do sistema, utilizadas nos arquivos de *Dinâmica* e *Análise Simplificada*.

Os arquivos de *Dinâmica*, *Análise Simplificada* e de *Biblioteca* incluem código gerado na linguagem (C ou FORTRAN) especificada no *arquivo de descrição do sistema* ou na execução do SD/FAST. O arquivo de *Informação* é do tipo texto, e inclui informações detalhadas para a composição do programa principal da simulação a ser preparado pelo usuário, como identificação de cada corpo, número de corpos, número de juntas da árvore e de *loop*, entre outros.

A estrutura do programa principal definida pelo usuário diz respeito à simulação a ser realizada e está além do escopo deste trabalho. Entretanto, vale ressaltar que o usuário precisa estar atento ao estado em que se encontra o sistema SD/FAST ao criar o programa principal. O sistema pode ficar em um de quatro estados, como mostra a figura 4.4. O apêndice B apresenta as rotinas utilizadas pelo SD/FAST, organizadas em classes.

No estado **New System**, o usuário pode utilizar somente rotinas *Utilitárias*, rotinas de *Mudança de Parâmetros* e `SDINIT()`. Esta última resulta na mudança do sistema para o estado **Initialized** e deve ser chamada uma única vez. No estado **Initialized**, somente rotinas *Utilitárias*, rotinas de *Análise Simplificada* e `sdstate()` devem ser chamadas. Esta última rotina sempre coloca o programa no estado **Kinematics Ready**. Neste estado, todos os cálculos relativos a posições e velocidades são realizados através de rotinas *Utilitárias*, rotinas de *Informação de Posição e Velocidade*, rotinas de *Forças e Movimento* e rotinas de *Derivação*. Se a rotina `SDDERIV()` é chamada, o sistema passa para o estado **Dynamics Ready**. Neste estado, os cálculos relativos à aceleração e forças são realizados por rotinas *Utilitárias*, rotinas de *Informação de Posição e Velocidade* e rotinas de *Informação de Aceleração e Forças*.

Figura 4.4 Estados do sistema SD/FAST.

O usuário também é responsável pela criação de rotinas padronizadas do SD/FAST para a execução da simulação. O conjunto de rotinas do sistema SD/FAST é listado no Apêndice B. Maiores detalhes sobre a criação do programa principal podem ser encontrados no manual do usuário do SD/FAST [SDFAST91].

4.6. Exemplo de Utilização

Esta seção ilustra um exemplo simples de simulação MBS realizada com o sistema SD/FAST. O exemplo consiste na junção de um mecanismo comum de quatro barras e seu movimento transiente para esta junção, como mostra a figura 4.5. O mecanismo consiste de um terra (*ground*), uma manivela (*crank*), um balancim (*rocker*) e uma barra de conexão (*connect rod*).

Figura 4.5 Modelo de um mecanismo de quatro barras.

Os parâmetros deste modelo são descritos através do *arquivo de descrição do sistema*, como ilustra a figura 4.6. O sistema opera sob gravidade e possui uma viscosidade de amortecimento (atrito) na junta de conexão entre a manivela e a terra de 3 Nm/(rad/sec).


```

Gravidade ————— gravity = 0 -9.8 0

body = crank inb = $ground joint = pin
mass = 1                      inertia = 1 0 1
bodytojoint = 0 -.5 0         inbtojoint = 0 0 0
pin = 0 0 1

Descrição de
cada corpo ————— body = rocker inb = $ground joint = pin
mass = 2                      inertia = 2 0 2
bodytojoint = 0 -1 0         inbtojoint = 3 0 0
pin = 0 0 1

body = connect inb = rocker joint = pin
mass = 3                      inertia = 0 3 3
bodytojoint = 1.5 0 0        inbtojoint = 0 1 0
pin = 0 0 1

Linha de comentário ————— # This is the loop joint.

Descrição da junta
de conexão ————— body = connect inb = crank joint = pin
bodytojoint = -1.5 0 0       inbtojoint = 0 .5 0
pin = 0 0 1

```

Figura 4.6 Arquivo de descrição do sistema do mecanismo de quatro barras.

O programa SD/FAST é responsável pela montagem do sistema, assumindo que a manivela está na posição vertical e somente o balancim e a barra de conexão podem mover durante a junção. Em seguida, as velocidades iniciais são computadas para todos os corpos com a manivela possuindo uma velocidade angular inicial de 10 rad/seg. Finalmente, as equações de movimentos são geradas em linguagens de alto nível como FORTRAN ou C, permitindo ao sistema iniciar a manivela a uma velocidade de 10 rad/seg, e então mover e cair devido à viscosidade de amortecimento e à gravidade (figura 4.6). Forças de reação no pino da manivela-terra também podem ser computadas.

Todas as propriedades das massas e a geometria básica, bem como a gravidade, são passadas ao sistema SD/FAST através do arquivo de entrada. Quaisquer outras características do modelo que dependam do estado do sistema, tais como a força da viscosidade de amortecimento, são passadas durante a execução da simulação para as rotinas SD/FAST através de rotinas escritas pelo usuário e *linkeditadas* com o código da simulação. A figura 4.7 ilustra parte do código em FORTRAN escrito pelo usuário para o mecanismo de quatro barras.

```

Dimensões fornecidas pelo SD/FAST      real*8 state(6), dstate(6), t, dt, ctol, tol, frc(4,3), trq(4,3)
                                         integer lock(3), fcnt, err, i, flag
                                         data dt, ctol, tol / 0.05d0, 1d-5, 1d-6 /
                                         data lock, flag / 1, 0, 0, 1 /

Inicialização do modelo SD/FAST      c Set initial conditions to a starting guess and initialize sd model.
                                         data t, state / 0d0, 0d0, 0d0, 0d0, 10d0, 1d0, -1d0 /
                                         call sdinit

Mecanismo de montagem                c Do assembly and velocity analysis. Only q(1) and u(1) are locked.
                                         call sdassemble (t, state, lock, 1d-7, 1000, fcnt, err)
Cálculo da compatibilidade das velocidades  call sdinitvel (t, state, lock, 1d-7, 1000, fcnt, err)
                                         print 5, state
                                         5 format (' ', 6f12.5)

Integração E. O. M.                  c Do motion analysis. Constraints maintained to 'ctol'; integration to 'tol'
                                         do 10 i = 1, 200
                                         call sdmotion (t, state, dstate, dt, ctol, tol, flag, err)
Cálculo das cargas de reação          call sdreac (frc, trq)
                                         print 5, t, state(1), state(4), frc(1,1), frc(1,2)
                                         10 continue

Checagem de erro                      call sdprinterr (6)
                                         end

Modelo de viscosidade de amortecimento  c Apply viscous damping torque to the crank joint
                                         subroutine sduforce (t, q, u)
                                         double precision t, q(3), u(3)
                                         call sdhinet (1, 1, -3d0*u(1))
                                         return
                                         end

```

Figura 4.7 Programa principal do usuário para o mecanismo de quatro barras.

Depois de compilar, *linkeditar* e executar a simulação, os resultados também podem ser obtidos em forma de arquivos textos. A figura 4.8 ilustra somente uma linha de um arquivo de saída da simulação.

	0.00000	0.00463	0.25268	10.00000	4.87298	-5.16398
Ângulo inicial da manivela (como desejado)						
Ângulos iniciais do balancim e da barra de conexão exigidos para a junção do mecanismo						
Velocidade inicial da manivela (como desejado)						
Velocidades angulares iniciais do balancim e da barra de conexão exigida para prevenir o mecanismo de uma separação						

Figura 4.8 Exemplo de resultado da simulação do mecanismo de quatro barras.

Ao invés de listar um histórico temporal da análise do movimento subsequente, os engenheiros geralmente mostram este histórico em forma de gráficos, como na figura 4.9, na qual são mostradas a posição e velocidade da manivela.

Figura 4.9 Exemplo de resultado gráfico do mecanismo de quatro barras, fornecendo a posição e a velocidade da manivela.

4.7. Considerações Finais

Uma visão superficial do funcionamento do sistema de simulação SD/FAST é importante para uma melhor compreensão do sistema de animação apresentado no capítulo 5. Embora este capítulo não apresente detalhes da operação do SD/FAST, os principais tópicos necessários para uma compreensão mais objetiva foram abordados. Fundamentalmente, a filosofia de operação do SD/FAST dirigiu a criação do sistema **Animbs**, que permite animar os resultados das simulações, descrito no próximo capítulo.

Capítulo 5

Geração da Animação - Animbs

5.1. Considerações Iniciais

No desenvolvimento do sistema **Animbs - *Animation for MBS***, procurou-se seguir a tendência atual no desenvolvimento de *software*: a utilização de bibliotecas gráficas e de interface padronizadas e certificadas por organizações compostas por empresas e entidades de pesquisa, garantindo maior portabilidade do código entre diversas plataformas. O Animbs foi desenvolvido na linguagem C padrão ANSI, para utilização em estações de trabalho RISC com sistema operacional UNIX. Para a interface gráfica foi utilizado o *toolkit* Motif, versão 1.2 [BERL92], e o X Window System [JOHNS89], da Open Software Foundation. Para a manipulação dos objetos tridimensionais foi utilizada a biblioteca gráfica OpenGL [ARB93] [KILG96], da OpenGL Architectural Review Board.

O **Animbs** foi desenvolvido com base na operação do sistema SD/FAST [SDFAST91] e serve de apoio à interpretação dos resultados da simulação. A partir da descrição de um MBS, o SD/FAST gera códigos compiláveis em C ou FORTRAN que permitem ao usuário do simulador inserir código que impõe restrições ao MBS ou apenas código para obter os dados que considera relevantes. Levando-se em conta a necessidade de manipulação do código fonte para a geração do programa de simulação e a garantia de que os resultados sejam sempre coerentes com os esperados pelo sistema **Animbs**, foi desenvolvida uma biblioteca de extensão do SD/FAST para a geração de animação de resultados da simulação produzidos pelo sistema. Esta biblioteca

fornece várias rotinas necessárias para a utilização correta do **Animbs**, as quais devem ser inseridas no programa principal da simulação, caso o usuário deseje gerar animações.

O **Animbs** permite ao usuário associar uma geometria ao MBS simulado pelo SD/FAST, que o interpreta apenas como um conjunto de corpos e juntas. A mesma geometria pode ser utilizada para animar diversas instâncias da simulação de um mesmo MBS.

O *software* gerado nesse projeto anima somente sistemas mecânicos aterrados (presos ao chão), limitando-se aos veículos automotivos, embora o sistema SD/FAST possa trabalhar com qualquer sistema mecânico que possa ser modelado como um MBS [SDFAST91]. Esta restrição foi imposta para limitar o escopo dos sistemas MBS (figura 3.1) e, assim, restringir o universo de objetos geométricos manipulados em cena.

Na seção 5.2 é apresentada a biblioteca de extensão do SD/FAST desenvolvida para gerar os dados da animação exigidos pelo **Animbs**. Na seção 5.3 é apresentado o sistema **Animbs** e, finalmente, na seção 5.4 é ilustrado um exemplo da utilização do sistema **Animbs**. O Apêndice C apresenta o *diagrama de fluxo de dados* e o Apêndice D, o diagrama do projeto do **Animbs**.

5.2. Biblioteca de Extensão

A biblioteca de extensão para a geração de animações dos resultados gerados pelo SD/FAST foi criada com o objetivo de evitar que o usuário cometa erros na geração dos dados necessários ao **Animbs**, uma vez que este tem total controle sobre o código do programa principal da simulação. Além disso, esta biblioteca garante uma independência em relação à simulação em estudo e a coerência dos dados necessários para a animação.

A biblioteca foi criada nos mesmos moldes das rotinas usuais do SD/FAST e utiliza somente rotinas de inspeção de dados do próprio SD/FAST, não interferindo na geração dos dados pertinentes à simulação. O usuário deve inserir o cabeçalho da biblioteca e suas rotinas no seu programa principal, obedecendo aos critérios de criação e respeitando o estado em que se encontra o sistema SD/FAST (seções 4.5 e 5.3). A biblioteca foi desenvolvida em C, padrão

ANSI, e está configurada para qualquer número de corpos e juntas, pois a alocação de memória é feita dinamicamente.

A biblioteca de extensão foi compilada para ser utilizada com precisão dupla e prefixo **sd**. O usuário deve somente referenciar a biblioteca durante a compilação de seu programa, na linha de comando do compilador. Com a utilização da biblioteca, o usuário terá, como resultado, quatro arquivos adicionais aos já gerados pelo SD/FAST:

- **Arquivo de informação**

Contém informações sobre o sistema simulado. Sua extensão é **inf**. Os dados sobre o sistema são obtidos utilizando-se rotinas de inspeção dos objetos da simulação (como corpos e juntas) do próprio sistema SD/FAST.

- **Arquivo de Dados Cinemáticos**

Contém dados cinemáticos da simulação, necessários à animação. Sua extensão é **kin**. Os dados são obtidos inspecionando-se o vetor de estados da simulação (descrito na seção 4.4) a cada iteração.

- **Arquivo de Forças de Reação**

Contém informações sobre a ação de forças e torques sobre cada junta do MBS. Sua extensão é **rec**. Os dados são obtidos inspecionando-se as forças e torques calculados pelo SD/FAST a cada iteração da simulação.

- **Arquivo de Forças**

Contém informações da ação de forças em cada eixo de cada junta do MBS. Sua extensão é **kin**. Os dados são obtidos inspecionando-se a ação de forças, em cada eixo da junta, calculadas pelo SD/FAST a cada iteração.

5.2.1. Rotinas da Biblioteca

A biblioteca de extensão possui apenas cinco rotinas, e todas começam com o prefixo **asd** (*animation for SD/FAST*), seguindo o padrão das bibliotecas existentes, com descrito na

tabela 5.1. Dessa forma, o usuário tem a possibilidade de criar rotinas com os mesmos nomes sem referenciar inadvertidamente uma rotina da biblioteca de extensão.

<code>asdInit()</code>	Inicializa o sistema para gerar dados da animação.
<code>asdClose()</code>	Finaliza a geração de dados da animação.
<code>asdKin()</code>	Gera os dados cinemáticos.
<code>asdReactionLoad()</code>	Gera os dados de forças de reação.
<code>asdHingeLoad()</code>	Gera os dados de forças.

Tabela 5.1 Rotinas da biblioteca de extensão.

As duas rotinas mais importantes se referem à inicialização e finalização da utilização da biblioteca de extensão: `asdInit()` e `asdClose()`, respectivamente.

A rotina `asdInit()` tem como parâmetro o nome do arquivo no qual os dados serão armazenados. Não é necessário especificar a extensão do arquivo, pois a rotina se encarrega de criar os arquivos já com sua respectiva extensão. Através da rotina do SD/FAST denominada `SDINFO()`, são obtidas as informações sobre o sistema sendo simulado, como o número de corpos, o número de juntas da árvore e o número de juntas de *loop*, além de outros dados. A gravidade a ser utilizada na simulação é obtida pela rotina SD/FAST `SDGETGRAV()`. Para cada junta do sistema utiliza-se a rotina SD/FAST `SDJNT()`, que retorna informações sobre as propriedades da junta, como o seu tipo, se é junta da árvore ou junta de *loop* e o número de graus de liberdade, entre outras. Além disso, as rotinas SD/FAST `SDGETBTJ()` e `SDGETITJ()` são utilizadas para obter, respectivamente, os vetores **bodyToJoint** e **inbToJoint**, necessários para o cálculo do referencial do corpo em relação ao referencial global. A rotina SD/FAST `SDGETPIN()` é utilizada para obter os eixos da junta. Os dados dos corpos são obtidos através das rotinas SD/FAST `SDGETMASS()`, para obter a massa de um corpo, e `SDGETINER()`, para obter a matriz de inércia de um corpo. Estes dados são armazenados em arquivos com extensão **inf**. Além de obter as informações sobre o sistema sendo simulado, a rotina `asdInit()` aloca memória suficiente para armazenar dados sobre as forças de reação, que dependem exclusivamente do número de corpos do sistema. Esta rotina deve ser chamada logo após a inicialização do SD/FAST, feita através da rotina `SDINIT()` na transição do estado *New System* para o estado *Initialized*, como ilustrado na figura 4.4.

A rotina `asdClose()` apenas finaliza os arquivos criados e libera o espaço de memória alocado. Não possui parâmetro algum, e deve ser a última a ser chamada no final da simulação.

As demais rotinas são responsáveis pela geração dos arquivos utilizados na criação da animação. A rotina `asdKin()` tem como parâmetro de entrada o instante de tempo da simulação e o vetor de estados (seção 4.3). Além disso, a rotina obtém, através da rotina SD/FAST `sdsys()`, a massa e a matriz de inércia de todo o sistema. A rotina armazena estes dados em arquivos com extensão **kin**. Esta rotina deve ser chamada a cada iteração da simulação, após uma chamada à rotina SD/FAST `sdderiv()`, responsável pela transição do estado *Kinematics Ready* para o estado *Dynamics Ready* (figura 4.4).

A rotina `asdReactionLoad()` tem como parâmetro apenas o instante de tempo da simulação, e utiliza a rotina SD/FAST `sdreac()` para obter as forças e torques exercidas em cada junta do sistema. A rotina armazena estes dados em arquivos com extensão **rec**. Esta rotina deve ser chamada a cada iteração da simulação e somente após a execução da rotina SD/FAST `sdderiv()`, na transição do estado *Kinematics Ready* para o estado *Dynamics Ready* (figura 4.4).

A rotina `asdHinge()` tem como parâmetro apenas o instante de tempo da simulação, e utiliza a rotina SD/FAST `sdgetht()` para obter os torques sobre cada eixo de cada junta. A rotina armazena estes dados em arquivos com extensão **hin**. Esta rotina deve ser chamada a cada iteração da simulação, após uma chamada à rotina SD/FAST `sdderiv()`.

Cabe ao usuário que está gerando o programa de simulação inserir corretamente as chamadas às rotinas responsáveis pela geração dos dados para a animação no estado correto do sistema SD/FAST (seção 4.5). A inserção incorreta dessas rotinas pode abortar a execução da simulação, por tentar obter dados ainda não disponíveis ou obter dados inconsistentes.

5.2.2.Formatos dos Arquivos de Dados Gerados pelas Rotinas da Biblioteca

Os dados contidos nos arquivos gerados pela biblioteca de extensão são armazenados em determinada ordem e, portanto, devem ser lidos nesta mesma ordem para a criação da animação.

Arquivos de extensão **inf** possuem o seguinte formato:

1. Informação sobre o sistema (vetor de valores em ponto fixo com 12 elementos):
 1. Sistema aterrado ou de movimento livre
 2. Número de corpos do sistema
 3. Número de graus de liberdade da árvore
 4. Número total de restrições
 5. Número de juntas de *loop* do sistema
 6. Número de graus de liberdade das juntas de *loop*
 7. Número de restrições das juntas de *loop*
 8. Número de juntas **sixdof** e **ball** da árvore
 9. Número de juntas **sixdof** e **ball** que são juntas de *loop*
 10. Número total de restrições de movimentos prescritos
 11. Número de restrições do usuário
 12. Número de restrições não obviamente redundantes.
2. Gravidade do sistema (vetor de valores em ponto flutuante com precisão dupla com 3 elementos)
3. Para cada junta do sistema:
 1. Informação da junta (vetor de valores em ponto fixo com 11 elementos):
 1. Tipo da junta (pin, ujoint, gimbal, ball, slider, sixdof, cylindrical, planar, weld)
 2. Junta da árvore ou junta de *loop*
 3. Número do *inboard body*
 4. Número do *outboard body*
 5. Número de graus de liberdade da junta
 6. Número de restrições usadas para sua implementação
 7. Número de forças prescritas ou geradas em tempo de execução
 8. Índice do vetor de estados ou pseudo-estados do primeiro eixo
 9. Índice do quarto parâmetro de Euler, para juntas **sixdof** e **ball**
 10. Índice do primeiro multiplicador, para qualquer junta de *loop*
 11. Índice do primeiro multiplicador, para qualquer movimento prescrito.
 2. Vetor **bodyToJoint** (vetor de valores em ponto flutuante com precisão dupla com 3 elementos)
 3. Vetor **inbToJoint** (vetor de valores em ponto flutuante com precisão dupla com 3 elementos)
 4. Para cada eixo da junta:
 1. Vetor do eixo (vetor de valores em ponto flutuante com precisão dupla com 3 elementos)
4. Para cada corpo do sistema:
 1. Massa do corpo (valor em ponto flutuante com precisão dupla)
 2. Matriz de inércia do corpo (matriz de valores em ponto flutuante com precisão dupla com 9 elementos)

Arquivos com extensão **kin** possuem o seguinte formato:

1. Instante de tempo (valor em ponto flutuante com precisão dupla)
2. Vetor de estados (vetor de valores em ponto flutuante com precisão dupla com número de elementos dependente do número de equações no sistema)
3. Massa do sistema (valor em ponto flutuante com dupla precisão)
4. Matriz de inércia do sistema (matriz de valores em ponto flutuante com precisão dupla com 9 posições)

Arquivos com extensão **rec** possuem o seguinte formato:

1. Instante de tempo (valor em ponto flutuante com dupla precisão)
2. Forças aplicadas em cada junta (valor em ponto flutuante com precisão dupla com 3 elementos)
3. Torques aplicados em cada junta (valor em ponto flutuante com precisão dupla com 3 elementos)

Arquivos com extensão **hin** possuem o seguinte formato:

1. Instante de tempo (valor em ponto flutuante com precisão dupla)
2. Para cada junta do sistema
 1. Para cada eixo da junta (junta **ball** possui 3 forças e junta **sixdoff**, 6)
 1. Torque da junta (vetor de valores em ponto flutuante com precisão dupla)

5.3. O Sistema Animbs

Um dos principais problemas encontrados na animação das simulações de MBS produzidas pelo sistema SD/FAST é a ausência de informação geométrica, essencial em qualquer animação. Apenas a posição do centro de massa do corpo pode ser obtida a partir dos valores iniciais e do vetor de estados. Isso porque, para a simulação no SD/FAST, o formato do corpo não é importante, bastando conhecer o valor de sua massa e a matriz de inércia, se existir.

A partir dos dados gerados pela biblioteca de extensão do SD/FAST, o usuário deve criar uma geometria para o MBS e, em seqüência, animá-la em tempo real ou, simplesmente, visualizar passo a passo a simulação do MBS.

O **Animbs** foi desenvolvido na linguagem C, padrão ANSI, tendo sido utilizado o *toolkit* **Motif**, versão 1.2, e a biblioteca do **X Window System** para a interface baseada em janelas. A biblioteca gráfica **OpenGL** foi utilizada para a manipulação e visualização dos objetos em cena.

5.3.1. Biblioteca OpenGL

O **OpenGL** [ARB93] é uma biblioteca gráfica padrão de alto desempenho, independente do sistema de janelas e do *hardware* gráfico utilizado, voltada para manipulação e renderização de objetos tridimensionais, além de manipular também gráficos bidimensionais e imagens. A biblioteca OpenGL foi desenvolvida pela Silicon Graphics, Inc., tendo sido

padronizada em 1991 por um consórcio de empresas, a OpenGL Architectural Review Board (ARB), que vem, desde então, ditando suas especificações.

O sistema OpenGL deve ser visto como uma máquina de estados que controla um conjunto de operações bem especificadas. O OpenGL oferece uma biblioteca de rotinas para a manipulação desta máquina de estados de forma a gerar corretamente uma saída gráfica. Assim, comandos específicos do OpenGL colocam o sistema em determinado estado, e comandos subsequentes são realizados sobre este estado.

O modelo abstrato para a interpretação dos comandos do OpenGL é baseado no modelo **Cliente/Servidor**, embora uma arquitetura cliente/servidor não seja uma exigência para sua implementação. Este modelo cliente/servidor cria um limite bem definido entre o aplicativo e a implementação OpenGL, especificando claramente como os dados serão passados. Além disso, permite operações sobre protocolos de rede, da mesma forma utilizada pelo X Window System. Dessa forma, comandos podem ser agrupados e enviados através de uma rede de computadores. No caso do X Window System, o OpenGL é implementado como uma extensão X [KILG96], significando que existe um protocolo de extensão X para o OpenGL (**GLX protocol**). Assim como aplicativos X podem ser executados em máquinas remotas e seus resultados mostrados em máquina locais, programas utilizando o OpenGL fazem o mesmo.

5.3.2. Funcionalidade do Sistema Animbs

O **Animbs** foi criado para utilizar, como dados de entrada para produzir a animação, os resultados gerados pela biblioteca de extensão do SD/FAST. Através do **Animbs**, o usuário deve definir a geometria do MBS, a partir dos arquivos de informação. O escopo do sistema **Animbs** foi especificado para abranger somente a animação de simulações de veículos terrestres, mais especificamente carros automotivos, embora sua implementação possa ser estendida para outros tipos de veículos, bastando apenas criar novos objetos geométricos. O Apêndice C apresenta o Diagrama de fluxo de Dados de alto nível do sistema e o Apêndice D apresenta os diagramas de decomposição hierárquica, segundo a notação usada no projeto estruturado de sistemas [PAGE88].

Figura 5.1 Propriedades de um MBS, visualizadas no **Animbs**.

A animação somente será realizada se o usuário possuir uma especificação geométrica para o MBS, que pode ser criada pelo usuário ou obtida de um arquivo já criado. Uma mesma especificação geométrica pode ser reutilizada para visualizar resultados de diferentes simulações de um mesmo MBS. A animação pode ser exibida em tempo real, ou passo a passo para uma análise mais detalhada. Informações sobre velocidades de corpos e juntas, forças aplicadas nas juntas e eixos das juntas podem ser obtidas para cada instante de tempo da simulação.

Todas as informações contidas no arquivo de descrição do sistema MBS podem ser visualizadas no **Animbs** através de janelas de propriedades. A figura 5.1 ilustra uma janela de propriedades do MBS na qual o número de corpos e o número de juntas da árvore e de *loop* podem ser verificados. Além disso, o usuário pode especificar um nome para o MBS a ser modelado, e uma breve descrição do sistema.

Uma informação que o **Animbs** não possui são os nomes dos corpos, que devem ser especificados pelo usuário de forma consistente com os nomes utilizados na simulação. O sistema SD/FAST utiliza os nomes definidos pelo usuário no *arquivo de descrição do sistema* para criar valores numéricos e constantes nas rotinas geradas, e não existe nenhuma rotina SD/FAST que permita recuperar o nome de cada corpo. Através de uma janela de propriedades,

como ilustra a figura 5.2, o usuário pode especificar o nome do corpo, verificar a matriz de inércia e o seu centro de massa em relação ao referencial global.

Figura 5.2 Propriedades de um corpo do MBS, visualizadas no **Animbs**.

As propriedades de uma junta (da árvore ou de *loop*) são visualizadas através de sua janela de propriedades, como ilustra a figura 5.3. Informações sobre os vetores **bodyToJoint** e **inbToJoint** podem ser obtidas, bem como os eixos da junta.

Figura 5.3 Propriedades de uma junta do MBS, visualizadas no **Animbs**.

O usuário tem acesso também a uma janela que descreve a topologia do MBS, que representa a hierarquia entre os corpos e as juntas que os conectam. Nesta janela, o usuário pode selecionar o corpo ou a junta da árvore cujas propriedades deseja visualizar. A figura 5.4 ilustra a janela que exibe a topologia de um MBS.

Figura 5.4 Janela de topologia de um MBS, visualizada no **Animbs**.

As juntas da árvore que conectam os corpos são visualizadas na janela de topologia como linhas de conexão entre os corpos. Juntas de *loop* têm suas propriedades apresentadas, mas não são visualizadas na janela de topologia. Suas propriedades podem ser obtidas a partir de uma lista de juntas de *loop*, como ilustra a figura 5.5. Estes tipos de juntas apenas restringem o movimento do MBS durante a simulação e seus efeitos podem ser observados durante a animação.

Figura 5.5 Juntas de *loop* de um MBS.

O **Animbs** somente permite criar uma geometria para corpos que realmente tenham massa (superior a zero). Os tipos de geometria admitidos atualmente para os corpos são três: **eixos**, **chassis** e **rodas**. Estas geometrias englobam quase todas as especificações possíveis para sistemas automotivos. Estes objetos geométricos podem ser escalonados e rotacionados pelo usuário para obedecer à especificação da topologia do corpo. O sistema automaticamente fixa o objeto geométrico ao centro de massa do corpo, e o referencial do corpo é utilizado em sua geometria para o cálculo das transformações de escala e rotação. A figura 5.6 ilustra o acesso às transformações de um corpo.

As transformações de *pan* e *zoom* não são aplicáveis à geometria dos corpos, mas aos planos de visualização. A primeira permite que o centro de referência da visualização seja modificado para melhor modelar a geometria. A segunda permite a aproximação ou o afastamento do centro de referência da visualização.

Figura 5.6 Operações de transformação sobre a geometria dos corpos (rotação e escalonamento) e sobre os planos de visualização (*pan*, *zoom* e rotação para a *trackball* virtual).

Seis planos de projeção ortogonal estão disponíveis no **Animbs** para a visualização das geometrias dos corpos: **front view** (projeção ortogonal do plano xy na direção positiva de z), **rear view** (projeção ortogonal do plano xy na direção negativa de z), **left view** (projeção ortogonal do plano zy na direção negativa de x), **right view** (projeção ortogonal do plano zy na direção positiva de x), **top view** (projeção ortogonal do plano xz na direção positiva de y) e **bottom view** (projeção ortogonal do plano xz na direção negativa de y). Estas projeções permitem ao usuário visualizar a geometria sendo criada em todas as direções, uma facilidade importante em qualquer sistema de modelamento geométrico. Além disso, a projeção ortográfica

permite que distâncias e ângulos sejam preservados na visualização. A figura 5.7 ilustra as projeções ortogonais em relação ao referencial global.

Figura 5.7 Localização das projeções ortogonais do **Animbs**.

O **Animbs** possui apenas uma janela de projeção em perspectiva na qual o usuário pode definir sua posição de observação, uma informação importante principalmente para a geração da animação. Inicialmente, o **Animbs** mostra o plano xy , na direção positiva de z . A figura 5.8 ilustra a forma de acesso aos planos de visualização.

A posição de observação do usuário é definida através de uma **trackball virtual** simulada na tela. Esta simulação é realizada sobre o plano de projeção, fazendo com que este rotacione em relação ao referencial global, dando a impressão de uma *trackball*. A figura 5.9 ilustra o plano de projeção em relação a um referencial global. A rotação é sempre realizada sobre os eixos x_p , y_p e $-z_p$, calculando-se suas coordenadas em relação ao referencial global. Para este cálculo é necessário obter a matriz inversa da projeção.

O usuário pode salvar a geometria criada em arquivo. Quando uma geometria é obtida através de arquivo, o **Animbs** solicita os nomes dos arquivos gerados pela biblioteca de extensão. O MBS só é visualizado se as informações contidas no arquivo de geometria são compatíveis com os dados contidos no arquivo de extensão **inf**. Os arquivos de geometria possuem extensão **geo**.

Figura 5.8 Seleccionando os planos de visualização no Animbs.

Figura 5.9 Plano de projeção em perspectiva em relação ao referencial global, para a simulação do *trackball* virtual.

O usuário tem todo o controle sobre a animação do sistema, podendo executá-la em tempo real ou passo a passo através de um painel de controle, como ilustra a figura 5.10. O **Animbs** apresenta o tempo de cada quadro, obtido a partir de qualquer um dos arquivos de extensões **kin**, **rec** ou **hin**. Dados como velocidade de cada corpo são obtidos dos arquivos de extensão **kin**. Forças e torques sobre cada junta são obtidos de **rec**, caso esse arquivo exista. Dados sobre forças em cada eixo da junta são obtidos de **hin**, caso esse arquivo exista.

Figura 5.10 Painel de controle da animação no **Animbs**.

Existe também uma janela para definição de propriedades da animação, como ilustra a figura 5.11. O usuário pode especificar um ponto de observação fixo ou um ponto de observação móvel que acompanha a geometria do MBS em relação a um dos eixos **x**, **y** e **z**. O primeiro permite visualizar a trajetória do sistema, enquanto que o segundo permite visualizar o comportamento dos corpos e das juntas.

Figura 5.11 Propriedades da animação, visualizadas no **Animbs**.

5.3.3.Implementação do Animbs

O sistema foi implementado em diversos módulos, cada qual contendo um conjunto de funções específicas, sendo que o módulo principal é denominado **animbs.c** (ver Apêndice D) Os demais arquivos possuem como prefixo a palavra **asd**, para arquivos somente do aplicativo, **asdm** para arquivos que utilizam o *toolkit* Motif, **asdgl** para arquivos que tratam da manipulação

dos objetos tridimensionais com a biblioteca OpenGL e **asdmgl** para arquivos que tratam de janelas e objetos tridimensionais ao mesmo tempo.

O sistema **Animbs** foi implementado para manipular qualquer quantidade de corpos e juntas, estando limitado somente pela memória disponível. As principais estruturas de dados são implementadas por vetores de registros e estes vetores são alocados dinamicamente durante a execução do programa para compatibilizar com a implementação das bibliotecas Motif e OpenGL, que se utilizam deste recurso para seus objetos.

Estas estruturas contêm informações sobre o sistema a ser animado, informações de cada corpo do sistema e informações de cada junta, seja junta da árvore ou junta de *loop*, que podem ser acessadas no **Animbs** através das janelas de propriedades.

Devido às características de implementação do SD/FAST, o índice do vetor da estrutura de dados dos corpos corresponde ao número do corpo em questão. Além disso, o número de um corpo sempre é maior que o número de seu *inboard body*, pois este último sempre é declarado primeiro no *arquivo de descrição do sistema* (seção 4.2). Este mesmo índice é utilizado para acessar os dados da junta da árvore que conecta o corpo ao corpo anterior. Somente as juntas de *loop* são acessadas por índices maiores que a quantidade de corpos.

A estrutura que mantém os dados lidos dos arquivos ***.kin**, ***.rec** e ***.hin** também é alocada dinamicamente, pois depende do número de equações da simulação, número de juntas e número total de eixos das juntas, respectivamente.

Os corpos geométricos são definidos através de listas OpenGL. Estas listas permitem melhor desempenho na apresentação e renderização dos objetos em cena, e explora melhor o recurso oferecido pelo modelo Cliente/Servidor do OpenGL. O objeto **eixo** é representado por linhas amarelas, e corresponde aos vetores *inbToJoint* e *bodyToJoint*. O objeto **chassis** é composto de duas listas, uma representando a carcaça do chassis e outra suas laterais. O objeto **rodas** é composto de quatro listas. O **Animbs** permite ao usuário trabalhar com objetos em *wireframe* ou sólidos com faces preenchidas segundo a cor do objeto. A figura 5.12 ilustra os objetos geométricos do **Animbs**, em sua forma *wireframe* e sólida.

Figura 5.12 Objetos do **Animbs**, em *wireframe* à direita e sólido à esquerda: (a) objeto roda; (b) objeto chassis.

Por questão de eficiência para a visualização da animação em tempo real, o sistema não utiliza nenhum recurso de iluminação e tonalização. A animação como ferramenta de engenharia é muito diferente da animação tradicional, como aquela utilizada pela indústria de entretenimento e pela propaganda. A engenharia não requer imagens com realismo, como a propaganda e nem pode pagar pelo preço (custo e tempo) para produzir tais imagens. Exige outros tipos de propriedades [NOMA85], como exatidão e não ambigüidade dos objetos mostrados em cena, produção rápida e, principalmente, economia de custo. Nesse sentido, um dos objetivos ao desenvolver o **Animbs** foi de disponibilizar uma ferramenta capaz de reduzir o tempo de análise dos resultados de simulações SD/FAST.

Cada corpo possui três matrizes de transformação associadas: a **matriz de transformação global**, a **matriz de transformação local** e a **matriz de transformação de animação**. A *matriz de transformação global* corresponde à sequência de transformações de translação que definem a posição do centro de massa do corpo e, conseqüentemente, do referencial do corpo em relação ao referencial do corpo *inboard body*. Esta matriz é acumulativa

e, portanto, corresponde ao referencial global do sistema, pois a matriz de transformação global de seu *inboard body* é computada no cálculo dado pela equação 4.2.

A *matriz de transformação local* corresponde à sequência de transformações de escalonamento e rotação realizadas sobre o corpo para definir sua orientação em relação ao seu próprio referencial. O **Animbs** não realiza as transformações de escalonamento e rotação sobre a geometria dos corpos propriamente ditos. Os objetos geométricos permanecem sempre os mesmos. As transformações são realizadas sobre a matriz de transformação local que, computada com a matriz de visualização e projeção, produz o mesmo efeito.

A *matriz de transformação de animação* corresponde à sequência de transformações realizadas sobre o corpo a cada iteração da simulação, correspondendo a rotações e/ou translações. Estas matrizes são definidas de acordo com o formato das matrizes de transformação manipuladas pelo OpenGL, correspondendo a um vetor de 16 posições. Tratando-se de matrizes do espaço homogêneo, suas inversas podem ser obtidas facilmente [FOLEY90], através pela transposta da matriz 3x3, que corresponde aos movimentos de rotação, e pela inversão do sinal na coluna (ou linha) de translação.

$$\begin{bmatrix} \text{Objeto} \\ \text{projetado} \end{bmatrix} = \begin{bmatrix} \text{Matriz} \\ \text{de} \\ \text{Projeção} \end{bmatrix} \times \begin{bmatrix} \text{Matriz de} \\ \text{transformação} \\ \text{global} \end{bmatrix} \times \begin{bmatrix} \text{Matriz de} \\ \text{transformação} \\ \text{local} \end{bmatrix} \times \begin{bmatrix} \text{Objeto} \\ \text{geométrico} \end{bmatrix}$$

(a)

$$\begin{bmatrix} \text{Matriz de} \\ \text{transformação} \\ \text{de animação} \end{bmatrix} = \begin{bmatrix} \text{Matriz de} \\ \text{transformação} \\ \text{global} \end{bmatrix}$$

$$\begin{bmatrix} \text{Objeto} \\ \text{projetado} \end{bmatrix} = \begin{bmatrix} \text{Matriz} \\ \text{de} \\ \text{Projeção} \end{bmatrix} \times \begin{bmatrix} \text{Matriz de} \\ \text{transformação} \\ \text{de animação} \end{bmatrix} \times \begin{bmatrix} \text{Matriz de} \\ \text{transformação} \\ \text{local} \end{bmatrix} \times \begin{bmatrix} \text{Objeto} \\ \text{geométrico} \end{bmatrix}$$

(b)

Figura 5.13 Sequência de multiplicação das matrizes de transformação: (a) criação e manipulação da geometria do corpo; (b) animação da geometria do MBS.

Durante a composição da geometria do sistema MBS, um corpo é visualizado através de sua matriz de transformação global e de sua matriz de transformação local. Na animação, o corpo é visualizado utilizando a matriz de transformação de animação e, na sequência, a matriz de transformação local. A matriz de transformação de animação inicialmente assume os valores da matriz de transformação global. A figura 5.13 ilustra a sequência de utilização das matrizes de transformação.

As projeções ortogonais são calculadas automaticamente. A projeção em perspectiva possui uma **matriz de projeção**, que é calculada uma única vez no processo de inicialização, e uma **matriz de visualização**, que posiciona o observador em relação ao referencial global e faz a projeção em perspectiva. Esta matriz é calculada toda vez que o usuário utiliza a *trackball* virtual. A *trackball* virtual utiliza a *matriz de visualização* para calcular sua inversa, com o objetivo de obter as coordenadas, no referencial global, dos eixos **x_p**, **y_p** e **-z_p** do plano de projeção (figura 5.9). Uma vez calculados os novos eixos **x_p'**, **y_p'** e **-z_p'** no referencial global, as rotações são realizadas sobre esses novos eixos em ângulos correspondentes aos movimentos realizados pelo usuário na tela. Esta nova matriz de visualização é então armazenada novamente na *matriz de visualização* e o cenário geométrico é apresentado. Para os cálculos da rotação, utilizou-se as próprias rotinas do OpenGL.

Os arquivos de extensão **geo** possuem o seguinte formato:

1. Informação sobre o sistema (vetor de valores em ponto fixo com 12 elementos):
 1. Sistema aterrado ou de movimento livre
 2. Número de corpos do sistema
 3. Número de graus de liberdade da árvore
 4. Número total de restrições
 5. Número de juntas de *loop* do sistema
 6. Número de graus de liberdade das juntas de *loop*
 7. Número de restrições das juntas de *loop*
 8. Número de juntas **sixdof** e **ball** da árvore
 9. Número de juntas **sixdof** e **ball** que são juntas de *loop*
 10. Número total de restrições de movimentos prescritos
 11. Número de restrições do usuário
 12. Número de restrições não obviamente redundantes.
2. Gravidade do sistema (vetor de valores em ponto flutuante com precisão dupla com 3 elementos)
3. Nome do sistema (cadeia de caracteres com 32 elementos)
4. Descrição do sistema (cadeia de caracteres com 255 elementos)
5. Para cada corpo do sistema:
 1. Informação da geometria do corpo (valor em ponto fixo)

2. Matriz de transformação global (vetor de valores em ponto flutuante com precisão dupla com 16 elementos)
3. Matriz de transformação local (vetor de valores em ponto flutuante com precisão dupla com 16 elementos)

Assim que o usuário define sua posição de observação, e desde que exista pelo menos um corpo sendo visualizado, o **Animbs** irá permitir a exibição da animação. Esta exibição utiliza a janela de projeção em perspectiva.

5.4. Exemplo de Utilização

Esta seção ilustra a criação da geometria de um sistema MBS até a sua animação. Quando o sistema **Animbs** é executado (comando `animbs`), a janela principal do sistema é apresentada como ilustra a figura 5.14, com o plano de visualização em perspectiva como plano de visualização inicial. Alguns menus não estão disponíveis ao usuário no momento da execução inicial do **Animbs**. Estes menus ficam disponíveis de acordo com o contexto do sistema.

Primeiramente, o usuário deve em criar uma geometria para o sistema MBS ou, então obter a geometria a partir de arquivos **geo** já disponíveis. Estas operações podem ser realizadas através do menu **File**, ilustrado na figura 5.15.

Figura 5.14 Janela principal do Animbs.

Figura 5.15 Menu **F**ile.

Para a criação de uma nova geometria, o usuário deve fornecer o arquivo de informações do MBS, gerado pela biblioteca de extensão. Para isso, deve-se selecionar o comando **New...** e

uma janela é mostrada para que o usuário selecione o arquivo de informações. A figura 5.16 exemplifica a janela de obtenção de arquivos de extensão **inf**.

Figura 5.16 Janela para obtenção de arquivos de extensão **inf**.

Uma vez que os dados foram obtidos do arquivo de informações, os menus antes desabilitados ficam disponíveis para verificação dos dados do MBS e para a criação das geometrias dos corpos. A figura 5.17 ilustra o menu **system**, que permite ao usuário acessar a topologia do MBS (figura 5.4) para selecionar o corpo a ser modelado e a junta, bem como verificar as propriedades do sistema.

Figura 5.17 Menu **system**.

O menu **Body** permite ao usuário criar ou remover a geometria de um corpo selecionado na janela de topologia do sistema, além de verificar as propriedades do corpo, como ilustra a figura 5.18.

Figura 5.18 Menu **Body**.

O comando (**Create geometry...**) somente fica disponível se o corpo selecionado na janela de topologia possuir massa. O usuário pode modificar o nome do corpo e selecionar o tipo da geometria do corpo a partir de uma lista, com ilustra a figura 5.19.

Figura 5.19 Janela de criação de geometria de corpos no **Animbs**.

Quando o usuário pressiona o botão **Ok**, a geometria do corpo especificado na lista é criado no plano de visualização em seu tamanho padrão. O usuário deve utilizar as transformações de rotação e escalonamento para orientar a geometria corretamente. O menu **Edit** permite ao usuário desfazer uma transformação sobre uma geometria de um determinado corpo (**Undo**), como ilustra a figura 5.20. Somente a última operação poderá ser desfeita.

Figura 5.20 Menu **Edit**.

As figura 5.21, 5.22 e 5.23 ilustram as seqüências de rotação e escalonamento para a orientação de um corpo com geometria de **roda**.

Figura 5.21 Geometria inicial do corpo.

Figura 5.22 Rotação do corpo em 90 graus em relação ao eixo horizontal do plano de visualização.

Figura 5.23 Escalonamento da geometria do corpo rotacionado.

O comando (**Remove geometry...**) somente fica disponível se o corpo possuir alguma geometria. A figura 5.24 ilustra a janela de remoção de geometria do corpo.

Figura 5.24 Janela para a remoção de uma geometria de um corpo no **Animbs**.

O menu **Joint** permite ao usuário verificar o número de juntas de *loop* e suas propriedades, como ilustra a figura 5.25.

Figura 5.25 Menu **Joint**.

O menu **Rendering** permite ao usuário selecionar a opção de visualização da geometria através de *wireframe* ou sólido, como ilustra a figura 5.26.

Figura 5.26 Menu **Rendering**.

Para salvar uma nova geometria, uma janela semelhante à janela de obtenção dos arquivos é mostrada. A figura 5.27 ilustra essa janela.

Figura 5.27 Janela para armazenar a geometria em arquivos de extensão **geo**.

A figura 5.28 ilustra o menu **Animation**, que permite ao usuário visualizar a painel de controle (figura 5.10) e as propriedades da animação (figura 5.11). Este menu somente estará disponível se existir alguma geometria para o MBS.

Figura 5.28 Menu **Animation**.

Através do painel de controle o usuário pode manipular a animação de acordo com suas necessidades, como ilustram as figuras 5.29 e 5.30.

Figura 5.29 Instante da animação com ponto de observação fixo.

Figura 5.30 Outro instante da animação com o ponto de observação fixo.

Finalmente, o menu **H**elp apenas apresenta uma descrição mínima do sistema **A**nimbs, como ilustrado a figura 5.31.

Figura 5.31 Menu **H**elp.

Para sair do **Animbs**, o usuário deve selecionar o comando **Quit**, no menu **File**. Constantemente o **Animbs** emite janelas de aviso dependendo do comando utilizado pelo usuário, como ilustrado na figura 5.32.

Figura 5.32 Exemplo de avisos do sistema **Animbs**.

5.5. Considerações Finais

O **Animbs** foi desenvolvido para servir de ferramenta de apoio para análise dos resultados de simulações de sistemas multicorpos, utilizando o recurso da animação computadorizada. Ao invés de trabalhar unicamente com planilhas e gráficos estatísticos, a alternativa de gerar animações permite ao usuário deste sistema obter conclusões sobre o MBS de forma mais rápida. Deve-se ressaltar que o **Animbs** não substitui planilhas e gráficos estatísticos, mas apenas complementa estas informações.

Conclusão

A animação por computador é indiscutivelmente muito útil no processo de desenvolvimento de projetos em engenharia, sendo utilizada como ferramenta na Engenharia Apoiada por Computador, em diversas fases do desenvolvimento. A engenharia mecânica utiliza muito os computadores no processo de desenvolvimento, desde a geração de desenhos de projetos até a simulação de possíveis produtos mecânicos. A integração da animação com a visualização científica de dados obtidos de simulações tornou-se atraente com o surgimento de computadores capazes de manipular grandes quantidades de dados, permitindo aos engenheiros uma análise dos mesmos de forma mais rápida, e com menor custo associado.

O **Animbs** foi desenvolvido com o objetivo de reduzir o tempo de análise dos resultados de simulações de sistemas mecânicos multicorpos através da geração de animações desses resultados. A vantagem da visualização completa da simulação através da animação é dar ao engenheiro uma nova visão, esclarecendo qualitativas dúvidas do comportamento do sistema e inferindo mais rapidamente a influência de cada corpo (e sua junta) no sistema mecânico como um todo. A animação é complementação das planilhas e gráficos estatísticos, que dão uma visão qualitativa, mas requerem uma análise e um conhecimento mais profundo do sistema mecânico modelado.

A utilização do sistema SD/FAST exigiu a criação de uma biblioteca de extensão para a produção coerente dos dados de animação necessários ao **Animbs**. O uso dessa biblioteca,

entretanto, não impede o usuário de produzir os seus próprios dados para uma análise mais profunda e criteriosa. Um dos problemas encontrados no desenvolvimento da biblioteca de extensão foi o grande universo de informações geradas pelo SD/FAST, e foi necessário criar dados necessários à visualização da animação do sistema simulado, uma vez que um modelo MBS não possui geometria associada. O escopo dos dados gerados pelo SD/FAST pode ser ampliado de acordo com a necessidade da análise do MBS, e deve ser complementado no **Animbs**.

Ao contrário da simulação, na qual o corpo é definido apenas por uma massa e uma matriz de inércia, a animação por computador requer uma geometria que possa ser visualizada durante a animação. Neste sentido, o **Animbs** também foi desenvolvido para atuar como uma ferramenta de modelagem geométrica de sistemas mecânicos do SD/FAST. Entretanto, devido à grande quantidade de sistemas MBS que podem ser modelados pelo SD/FAST, o escopo do **Animbs** foi limitado aos veículos automotivos.

Em relação aos recursos computacionais, o **Animbs** está limitado pela utilização em sistemas operacionais UNIX que possuam gerenciadores de janelas compatíveis ao padrão Motif e da necessidade de utilização a biblioteca padronizada OpenGL. No desenvolvimento do **Animbs** todo cuidado foi tomado para tornar os módulos da aplicação, os módulos de janelas e os módulos da biblioteca gráfica o mais independente possível.

O sistema desenvolvido requer ainda complementações que podem ser facilmente integradas ao sistema, como:

- Ampliação do escopo dos sistemas mecânicos visualizados no **Animbs** através da ampliação dos objetos geométricos disponíveis através da definição e/ou utilização de bibliotecas de geometrias pré-definidas.
- Utilização de NURBS (*Non-Uniform Rational B-Spline*) para a criação de objetos mais flexíveis dos que os apresentados atualmente pelo **Animbs**, como o objeto **chassis**.

- Introdução de sistemas de gráficos bidimensionais para análise complementar das forças e torques atuantes no MBS, além de trajetórias de corpos e do sistema.
- Introdução de um módulo para criação de cenários para a geração de animações mais próximas da realidade, além de introdução de métodos de iluminação e tonalização, e utilização de técnicas de renderização mais realistas.
- Geração da animação em arquivo para animação em *playback* e em formato padronizado de vídeo, como o MPEG. Esta complementação, juntamente com as duas últimas, podem permitir que os resultados de uma simulação possam ser visualizadas e compreendidas por pessoas fora do escopo da área de engenharia mecânica.
- Implementação do sistema para gerenciadores de janelas alternativas como o Windows 95 e bibliotecas gráficas como PHIGS e PEX.

A integração de tecnologias de *hardware* e *software* permitiu o desenvolvimento de ferramentas poderosas que auxiliam o engenheiro no projeto e criação de seus produtos, com a redução de seus custos. Neste sentido, o **Animbs** pode passar a integrar o sistema SD/FAST como módulo de pós-processamento e ferramenta de apoio ao desenvolvimento, complementado análises obtidas a partir de outras ferramentas, como gráficos e planilhas.

Referências

- [ADAMS92] Mechanical Dynamics, *ADAMS/View: User's Guide*, versão 6.1, 1992.
- [AMBRO93] Ambrósio, J. A. C.; Nikravesh, P. E.; "Animation of multibody systems with an active observer and viewpoint strategy using Euler parameters", *Compugraphics'93, Third International Conference on Computational Graphics and Visualization Techniques*, Ed. Harold P. Santo, ACM, Algarve, Portugal, 1993, 27-28.
- [ARB93] OpenGL Architectural Review Board; Neider, J.; Davis, T.; Woo, M.; *OpenGL Programming Guide — The Official Guide to Learning OpenGL, Release 1*; Addison-Wesley, 1993.
- [BAECK80] Baecker, R. M.; "Picture-driven animation", *Tutorial and Selected Readings in Interactive Computer Graphics*, Ed. H. Freeman, IEEE Computer Society Press, 1980, 332-347.
- [BERL92] Berlage, T.; *OSF/Motif: Concepts and Programming*; Addison-Wesley, 1992.
- [BURTN82] Burtnyk, N.; Wein, M.; "Interactive skeleton techniques for enhancing motion dynamics in key frame animation", *Tutorial: Computer Graphics*, Eds. J. C. Beatty and K. S. Booth, Second Edition, IEEE Computer Society Press, 1982, 516-512.
-

- [CAMAR93] Camargo, J. T. F.; Magalhães, L. P.; *Técnicas de controle em animação — Um tutorial*, Relatório interno, UNICAMP, 1993.
- [CHEOK92] Cheok, K. C.; Nigjian, H.; "Real-time simulation and animation for dynamic control system", *Simulation*, vol. 59, n. 4, outubro 1992, 245-252.
- [COSTA92] Costa Neto, A.; *Application of Multibody System (MBS) — Techniques to Automotive Vehicle Chassis Simulation for Motion Control Studies*, Ph.D. Thesis, Engineering Department, University of Warwick, Coventry, UK, janeiro 1992.
- [FERG93] Ferguson, P. M; *Motif Reference Manual for OSF/Motif Release 1.2*, Motif Edition, Volume Six B, O'Reilly & Associates, 1993.
- [FOLEY90] Foley, J. D.; van Dam, A.; Feiner, S. K.; Hughes, J. F.; *Computer Graphics — Principles and Practice*, Second Edition, Addison-Wesley, 1990.
- [GLASS88] Glassner, A. S; "Space ray tracing for animation", *IEEE Computer Graphics & Applications*, vol. 8 , n. 2, março 1988, 60-70.
- [HACK80] Hackathorn, R. J.; "Anima II: A 3-D color animation system", *Tutorial and Selected Readings in Interactive Computer Graphics*, Ed. H. Freeman, IEEE Computer Society Press, 1980, 364-374.
- [HELL93] Heller, D.; Ferguson, P. M.; *Motif Programming Manual for OSF/Motif Release 1.2*; Motif Edition, Volume Six A, O'Reilly & Associates, 1993.
- [JOHNS89] Johnson, E. F; Reichard, K.; *X Windows — Applications Programming*, Advanced Computer Books, MIS Press, Portland, 1989.
- [KILG96] Kilgard, M. J.; *OpenGL — Programming for the X Window System*, Addison-Wesley, 1996.

-
- [LANSD85] Lansdown, J.; "Object and movement description techniques for animation — An informal review", *Fundamental Algorithms for Computer Graphics*, Ed. R. A. Earnshaw, NATO ASI Series, vol. F17, Springer-Verlag, Berlin Heidelberg, 1985, 1029-1037.
- [MAGTH85a] Magnenat-Thalmann, N.; Thalmann, D.; *Computer Animation — Theory and Practice*, Ed. T. L. Kunii, Computer Science Workbench, Springer-Verlag, Tokyo, 1985.
- [MAGTH85b] Magnenat-Thalmann, N.; Thalmann, D.; "Three-dimensional computer animation: More an evolution than a motion problem", *IEEE Computer Graphics & Applications*, vol. 5, n. 10, outubro 1985, 47-57.
- [MATLAB93] The MathWorks, Inc., *The MATLAB EXPO*, março 1993.
- [NOMA85] Noma, T.; Kunii, T. L.; "ANIMENGINE: An engineering animation system", *IEEE Computer Graphics & Applications*, vol. 5, n. 10, outubro 1985, 24-33.
- [PAGE88] Page-Jones, M.; *Projeto Estruturado de Sistemas*, McGraw-Hill, 1988.
- [PORT94] Porter, S.; "Engineering visualization", *Computer Graphics World*, vol. 17, n. 11, novembro de 1994, 23-33.
- [RAYNA87] Rayna, G.; *REDUCE — Software for Algebraic Computation*, Symbolic Computation, Springer-Verlag, New York, 1987.
- [REINH94] Reinhardt, A.; "Construindo a auto-estrada de dados", *Byte do Brasil*, vol. 3, n. 5, maio 1994, 84-104.
- [SDFAST91] Symbolic Dynamics, *SD/FAST: User's Manual*, versão B.1.1, 1991.
- [SHOUP82] Shoup, R. G.; "Color table animation", *Tutorial: Computer Graphics*, Eds. J. C. Beatty and K. S. Booth, Second Edition, IEEE Computer Society Press, 1982, 214-219.
- [SMITH82] Smith, A. R.; "Paint", *Tutorial: Computer Graphics*, Eds. J. C. Beatty and K. S. Booth, Second Edition, IEEE Computer Society Press, 1982, 501-515.
-

- [SOLOM83] Solomon, C; Stark, R.; *The Complete Kodak Animation Book*, Eastman Kodak Company, New York, 1983.
- [TALB80] Talbot, P. A.; Carr III, J. W.; Coulter Jr, R. R.; Hwang, R. C.; "Animator: An On-line two-dimensional film animation system", *Tutorial and Selected Readings in Interactive Computer Graphics*, H. Freeman, IEEE Computer Society Press, 1980, 348-356.
- [THALM89] Thalmann, D.; "Motion control: from keyframe to task-level animation", *State-of-the-art in Computer Animation*, Proceedings of Computer Animation '89, Eds. N. Magnenat-Thalmann and, D. Thalmann, Springer-Verlag, Tokyo, 1989.
- [WILH87] Wilhelms, J.; "Toward automatic motion control", *IEEE Computer Graphics & Applications*, vol. 7, n. 4, abril 1987, 11-22.
- [WYVIL90] Wyvill, B.; "A computer animation tutorial", *Computer Graphics Techniques — Theory and Practice*, Ed. D. F. Rogers and R. A. Earnshaw, Springer-Verlag, New York, 1990.

Apêndice A

Sintaxe do Arquivo de Descrição do Sistema

```
# Any text following a '#' is considered a comment

#
# Preamble (optional)
#
[ gravity = <vector> ]
[ language = { fortran | c } ]
[ prefix = <string> ]

#
# Body (1 or more)
#
{ body = <name> inboard = { $ground | <name> } joint = <jointtype>
  mass = <real>
  inertia = { <real> <real> <real> | <real> <real> <real>
              <real> <real> <real>
              <real> <real> <real> }

  bodyToJoint = <vector>
  inbToJoint = <vector>
  [ pin = <vector> ]*
  [ prescribed [ = { 1[?] | 0[?] | ? }* ] ] }*
```

```

#
# Loop joint (0 or more)
#
[ body = <name> inboard = { $ground|<name>} joint = <jointtype>
  bodyToJoint = <vector>
  inbToJoint = <vector>
  [ pin = <vector> ]*
  [ bodypin = <vector> ]
  [ inbref = <vector> ]
  [ bodyref = <vector> ]
  [prescribed [ = {1[?]|0[?]|?}* ] ] ]*

#
# User constraint (0 or more)
#
[ {constraint = <name>}|{constraints = <integer>} ]*

```

onde:

[] = opção { } = exigido * = repetição | = escolha ? = parâmetro
 <real> = [+|-][<digits>][.[<digits>]][e[+|-]<digits>][?]
 <jointtype> = pin|ujoint|gimbal|ball|slider|cylinder|planar|
 sixdof|weld
 <vector> = <real> <real> <real>
 <name> = seqüência até 32 caracteres de letras, dígitos e _. Deve começar com letra
 <string> = seqüência de caracteres legais ao compilador do usuário
 \$ground = nome pré-definido para o corpo inicial

Apêndice B

Rotinas do SD/FAST

As rotinas geradas pelo SD/FAST são mostradas abaixo, agrupadas em oito classes, e são adicionais as rotinas escritas pelo próprio usuário. Os parâmetros foram omitidos. Mais detalhes podem ser encontrados em [SDFAST91].

Mudança de parâmetro

SBTJ ()	Especifica o ponto de junção a partir do <i>outboard body</i> .
SDGRAV ()	Especifica a gravidade.
SDINER ()	Especifica a matriz de inércia de um corpo.
SDITJ ()	Especifica o ponto de junção a partir do <i>inboard body</i> .
SDMASS ()	Especifica a massa de um corpo.
SDPIN ()	Especifica as orientações dos eixos da junta.

Inicialização

SDINIT ()	Inicializa o sistema, utilizado no começo ou após uma mudança de parâmetro.
------------	---

Especificação do Estado do Sistema

SDSTATE ()	Especifica o estado e o tempo do sistema.
SDASSEMBLE ()	Ajusta as coordenadas generalizadas para satisfazerem as restrições de posição.
SDINITVEL ()	Ajusta as velocidades generalizadas para satisfazerem as restrições de velocidade.

Cálculo de Derivadas

SDDERIV()	Calcula as derivadas do vetor de estados.
SDSTATIC()	Ajusta as coordenadas generalizadas para encontrar uma configuração estática.
SDSTEADY()	Ajusta o vetor de estados para encontrar uma configuração estável.
SDMOTION()	Integração do vetor de estados de t para $t+dt$.

Movimentos prescritos e aplicação de cargas

SDBODYT()	Aplica torque em um corpo.
SDHINGET()	Aplica carga (força ou torque) no eixo da junta.
SDPOINTF()	Aplica força em um ponto do corpo.
SDPRESACC()	Aceleração prescrita para um eixo da junta.
SDPRESVEL()	Velocidade prescrita para um eixo da junta.
SDPRESPOS()	Posição prescrita para um eixo da junta.
SDPSSTATE()	Troca o pseudo-estado prescrito com valores integrados.

Posicionamento e Informações de Velocidade

SDANGVEL()	Velocidade angular inercial em relação ao referencial do corpo.
SDMOM()	Momento angular e linear inercial e energia cinética.
SDORIENT()	Orientação do corpo em relação ao referencial global.
SDPERR()	Erros de restrição de posicionamento.
SDPOS()	Posição inercial em um ponto de algum corpo.
SDPSEUDO()	Retorna variáveis do pseudo-estado.
SDPSQDOT()	Retorna as derivadas do pseudo-coordenadas
SDRESID()	Calcula os erros na integração DAE de derivadas e multiplicações.
SDTRANS()	Transforma um vetor de um referencial para outro referencial.
SDSYS()	Massa do sistema, centro de massa e matriz de inércia do sistema.
SDVEL()	Velocidade inercial em um ponto do corpo.
SDVERR()	Erros de restrição de velocidade.

Aceleração e Informações de Cargas

SDACC()	Aceleração inercial em um ponto do corpo.
SDAERR()	Erros de restrição de aceleração.
SDADGACC()	Aceleração angular inercial em relação ao referencial do corpo.
SDGETHT()	Obtém carga aplicada ou computada
SDMULT()	Informação dos multiplicadores de restrição.
SDPSUDOT()	Retorna as derivadas das pseudo-velocidades.
SDREAC()	Carga de reação do <i>outboard body</i> , em relação ao referencial do corpo.

Utilitários

SDANG2DC ()	Converte ângulos de Euler para cosenos diretores.
SDANG2ST ()	Converte ângulos de Euler em quatérnios.
SDCLEARERR ()	Zera erros de utilização, se existirem.
SDCONS ()	Obtém informação sobre uma restrição.
SDDC2ANG ()	Converte cosenos diretores para ângulos de Euler.
SDDC2QUAT ()	Converte cosenos diretores para quatérnios.
SDERROR ()	Retorna o número do erro de utilização, se existir.
SDFINTEG ()	Integrador RK4 de passo fixo.
SDGETBTJ ()	Obtém a localização do ponto de junção do <i>outboard body</i> .
SDGETGRAV ()	Obtém o valor corrente do vetor da gravidade.
SDGETINER ()	Obtém a matriz de inércia de um corpo.
SDGETITJ ()	Obtém a localização do ponto de junção do <i>inboard body</i> .
SDGETMASS ()	Obtém a massa do corpo.
SDGETPIN ()	Obtém a orientação dos eixos da junta.
SDGETPRES ()	Verifica se o movimento prescrito está ativo ou não.
SDGETSTAB ()	Obtém os valores correntes de Baumgarte.
SDINDX ()	Função que retorna o índice do vetor de estados.
SDINFO ()	Obtém a informação geral do sistema.
SDJNT ()	Obtém a informação sobre a junta.
SDLSSLV ()	Calcula as equações lineares de mínimos-quadrados.
SDNORMST ()	Normaliza os quatérnios no estado.
SDPRES ()	Ativa ou desativa movimentos prescritos.
SDPRINTERR ()	Imprime o erro de utilização, se existir.
SDQUAT2DC ()	Converte quatérnios para cosenos diretores.
SDROOT ()	Calcula as equações não-lineares.
SDST2ANG	Converte quatérnios para ângulos de Euler.
SDSTAB ()	Especifica as constantes de Baumgarte.
SDVINTEG ()	Integrador RK4 de passo variável.

Definidas pelo usuário

sduaerr ()	Calcula os erros de restrição da aceleração.
sduconsfrc ()	Aplica as cargas de reação de restrição.
sduderiv ()	Calcula as derivadas para o integrador.
sdueval ()	Avalia o erro residual para o cálculo da raiz.
sduforce ()	Aplica forças e torques.
sdumotion ()	Aplica movimentos prescritos.
sduperr ()	Calcula os erros de restrição do posicionamento.
sduverr ()	Calcula os erros de restrição da velocidade.

Apêndice C

Diagrama de Fluxo de Dados do Animbs

Apêndice D

Projeto Modular do Animbs