UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Simulação de Movimentos Para o Membro Superior Humano Baseado nas Forças Aplicadas Pelos Músculos

Orientador: Edgard Lamounier Júnior, PhD

Co-Orientador: Alcimar Barbosa Soares, PhD

Orientanda: Marlene Ferreira Marques

Novembro

2005

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Simulação de Movimentos Para o Membro Superior Humano Baseado nas Forças Aplicadas Pelos Músculos

Dissertação apresentada à Universidade Federal de Uberlândia, perante a banca de examinadores abaixo, como parte dos requisitos necessários à obtenção do título de Mestre em Ciências.

Edgard Lamounier Júnior, PhD (UFU) - Orientador
Alcimar Barbosa Soares, PhD (UFU) - Co-Orientador
Adriano Alves Pereira, PhD (UFU)
Marcio Serolli Pinho, Dr (PUC-RS)

FICHA CATALOGRÁFICA

Elaborada pelo Sistema de Bibliotecas da UFU / Setor de Catalogação e Classificação

M357s Marques, Marlene Ferreira, 1971-
Simulação de movimentos para o membro superior humano baseado nas forças aplicadas pelos músculos / Marlene Ferreira Marques. - Uberlândia, 2005.
129f. : il.
Orientador: Edgard Lamounier Jr.
Dissertação (mestrado) – Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Elétrica.
Inclui bibliografia.
1. Engenharia biomédica - Teses. 2. Realidade virtual - Teses. 3. Simulação por computador - Teses. I. Lamounier Júnior, Edgard. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

CDU: 61:62 (043.3)

Simulação de Movimentos do Membro Superior Humano Baseado nas Forças Aplicadas Pelos Músculos

Marlene Ferreira Marques

Dissertação apresentada à Universidade Federal de Uberlândia como parte dos requisitos para obtenção do título de Mestre em Ciências.

Edgard Lamounier Júnior, PhD
Orientador

Aos meu pais Mauro e Conceição,

Aos meus irmãos Marina, Carminha,

Aroldo, Arnaldo e Armando

pelo carinho e apoio durante minha jornada.

Agradecimentos

Agradeço primeiramente a Deus por iluminar os meus caminhos, concedendo-me força para superar todos os obstáculos.

Ao meu orientador, Professor Edgard Lamounier pela dedicação, paciência, amizade e orientação com extrema competência.

Ao Professor Márcio Pinho pela atenção, presteza e colaboração durante toda a fase do meu projeto.

Ao Professor Alcimar Soares pelo carinho, apoio, contribuição e paciência.

Ao Professor Alexandre Cardoso pelo carinho e apoio oferecidos sempre com entusiasmo.

À minha irmã Marina e sua família pelo acolhimento nos momentos de dor e alegria.

Às minhas amigas Luzia, Maria Cândida, Antônia e Renata pelo companheirismo nos momentos de dificuldade e de alegria.

Aos meus amigos de laboratório, Arquimedes(meu terceiro orientador), Marlene, Márcio, Paula, Luciano, Kenedy, Flávio, Maria Emília e Wneiton, e também Élvio, pelo carinho, amizade, companheirismo e troca de conhecimento.

Aos demais amigos da pós-graduação pela amizade, colaboração e pelos momentos de lazer.

Aos professores Reny Alan Araújo, Cláudio Cardoso e Luiz Alberto Corticione da Faculdade de Física e Edson Agustini da Faculdade de Matemática pela troca de conhecimento.

Aos meus *"pupils"* Nayara, Natália, Caio e Victor pelo carinho, atenção e interesse pela transferência de conhecimentos.

À Marly pelo auxílio e presteza na secretária da pós-graduação.

A todos aqueles que de alguma forma contribuíram direta ou indiretamente para a realização deste trabalho.

À CAPES pelo suporte financeiro a esta pesquisa.

Resumo

Marques, M. F. *Simulação de Movimentos do Membro Superior Humano Baseado nas Forças Aplicadas pelos Músculos*, Faculdade de Engenharia Elétrica - UFU, Uberlândia, 2005.

A modelagem e simulação computacional do movimento e da postura de seres humanos vem sendo empregados há muitos anos na medicina e fisioterapia em vários aspectos tais como, auxílio em cirurgias, reabilitação de deficientes físicos, bem como em muitas outras atividades desempenhadas por profissionais da área de saúde.

Entretanto, a maioria dos sistemas baseados nesta tecnologia não apresentam características primordiais ao treinamento de uso de próteses, como exemplo simulação de movimentos dos membros por meio de valores de força aplicadas pelos músculos. Portanto, este trabalho tem como objetivo investigar técnicas computacionais que suportam eficientemente as simulações acima referidas, bem como possibilitem ao profissional da área uma interface intuitiva para manipulação dos parâmetros necessários neste tipo de sistema.

Um estudo dos princípios biomecânicos é apresentado, destacando-se os músculos, articulações e ossos (propostos neste trabalho como biblioteca de classes genéricas) relacionados aos movimentos do membro superior humano, bem como as equações necessárias à análise de equilíbrio estático do corpo. Também é apresentado um estudo sobre os principais sistemas de modelagem e simulação de membros do corpo humano disponíveis no meio científico. Com base nestes estudos, procurou-se definir as limitações dos sistemas existentes em termos de simulações por meio de forças aplicadas pelos músculos em tempo de execução e, partindo destas limitações, propor um sistema que permitisse a modelagem músculo-esquelética e a simulação de movimentos do membro superior a partir da alteração dos valores das forças aplicadas pelos músculos em tempo de execução do movimento. Objetivando uma interface com alto nível de interação, técnicas de Realidade Virtual foram investigadas no desenvolvimento do sistema.

Os parâmetros necessários para simulação do movimento são obtidos por meio de objetos de interface ou por um arquivo que contém valores de forças aplicadas pelos músculos, aplicadas em um dado intervalo do tempo. Após uma série de experimentos, observou-se que o sistema permite, além da alteração do valor das forças aplicadas pelos músculos durante a simulação do movimento, a interação do usuário com o modelo do membro superior humano, podendo avaliar sob vários pontos de vista, o efeito destas no membro modelado.

Palavras-chave:

Simulação de músculos, realidade virtual, modelo biomecânico, movimento, modelo músculo-esquelético.

Abstract

Marques, M. F. *Simulation of Upper Limb Motion Considering Applied Forces by Muscles as Parameters*, Faculty of Electrical Engineering - UFU, Uberlândia, 2005.

The computational modelling and simulation of the motion and gait of human beings have been employed for many years in medicine and physiotherapy in several aspects such as, aid in surgeries, amputee treatments, as well as in many other activities performed by professionals of the health area.

However, most of the systems based on this technology do not present the main characteristics to prosthesis training, for example, the simulation of limb motion responding to forces applied by the muscles. Therefore, this work has the objective of investigating computational techniques that efficiently support the simulations cited above, and also to provide an intuitive interface for manipulation of the necessary parameters in this type of system to the health professional and Biomedical engineering .

A study of biomechanics principles is presented emphasizing the muscles, joints and bones (considered in this work as a generic class library) related to the human upper limb motion, associated to the necessary equations of the static equilibrium analysis for the human body. Besides, it is presented a study on the main modelling and simulation systems of the human body limbs available in the scientific area. Considering these studies, it has been defined the limitations of the existing systems in terms of simulation by run-time muscle forces and then, to consider a system that allows to the musculoskeletal modelling and the simulation of upper limb motion answering the changes of the forces values applied by the muscles in execution time of the motion. In order to provide an interface with high level of interaction, Virtual Reality techniques had been investigated in the development of the system.

The necessary parameters for the motion simulation are obtained by objects of interface or from a file that contains values of forces applied by the muscles in known time interval. After a set of experiments, it was observed that the system allows the changing of value of applied forces by muscles during the motion simulation as well as user interaction with human upper limb model, being able to evaluate from different points of view, the effect of applied forces by muscles of the modelled limb.

Keywords:

Muscle simulation, virtual reality, biomecanic model, motion, musculoskeletal model.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

O movimento é uma das atividades mais naturais e importantes que um ser humano pode praticar. Tal atividade já é evidenciada mesmo antes do nascimento, quando somos capazes de realizar movimentos, *in utero*, de modo perceptível pela nossa mãe [Flehmig 1987]. Quase sempre o valor que essa habilidade representa para o ser humano não é percebido. Raras vezes, quando se permite parar e assistir um espetáculo de dança ou ginástica, executado com leveza e precisão, é que se tem noção da grandeza dessa capacidade. Um exemplo é a apresentação da brasileira Daiane dos Santos ao som de "Brasileirinho", que a fez conquistar a medalha de ouro no solo na etapa de Stuttgart, na Alemanha, da Copa do Mundo de ginástica artística em 2005, após apresentar o salto duplo twist carpado.

Uma outra forma de evidenciar essa fascinante habilidade designada ao ser humano é quando se presencia uma roda de capoeira. O mestre de capoeira e doutor em sociologia Luiz Renato costuma dizer durante as aulas dessa arte-luta que, na roda de capoeira assim como na vida, é muito importante saber cair e levantar [Vieira 1995].

Algumas vezes as pessoas levam quedas tão grandes que sentem-se incapazes de se levantar e enfrentarem a vida novamente. Seria o caso de um portador de deficiência física que, diante da dificuldade de enfrentar o preconceito e tentar recuperar parte dos movi-

mentos perdidos, se sente desencorajado, ou melhor, desmotivado a usar uma prótese. Na maioria dos casos, o deficiente precisa de um prolongado acompanhamento de profissionais para ajudá-lo nesta nova empreitada. Muitas dificuldades são encontradas na fase de treinamento para o uso de prótese, mas a maior delas é o grande esforço mental necessário nos primeiros estágios de treinamento. Quando trabalham com prótese mioelétrica, este esforço aumenta consideravelmente [Soares et al. 2003].

Considerando o cenário de reabilitação de deficientes físicos, existem vários sistemas destinados à simulação de movimentos dos membros. No entanto, estes sistemas apresentam limitações quando se pretende simular movimentos de membros, respondendo aos valores de forças aplicadas pelos músculos em tempo real. Esta limitação dificultaria o treinamento mais natural do uso de próteses por meio de softwares conectados a sensores que captam os sinais de forças aplicadas músculos.

Um bom programa de reabilitação é igualmente importante a uma prótese bem feita [Lake 1997]. Logo, a fim de colaborar com programas dedicados à reabilitação de pacientes com deficiência física, é proposto neste trabalho um sistema que, ao receber o sinal de estímulo dos músculos ainda ativos nos pacientes, simula o movimento que ele seria capaz de fazer com o uso de prótese de membro superior. Este trabalho é uma parte de um projeto planejado para dar suporte aos usuários de próteses no estágio de aprendizagem, quando ainda não estão usando uma prótese real. O projeto completo é composto por sensores de sinais eletromiográficos(EMG), um software para detecção e processamento destes sinais, um software que converte os sinais EMG em valores de força(N) para cada músculo envolvido no movimento, um software de simulação dos movimentos baseado nos valores das forças aplicadas pelos músculos e um sistema que utiliza técnicas de Realidade Aumentada para propiciar um maior realismo no treinamento.

1.2 Objetivo e Metas

O objetivo deste trabalho é desenvolver técnicas e algorítmos computacionais e bibliotecas que permitem o desenvolvimento de modelos músculo-esqueléticos para humanos,

bem como a simulação de movimentos realizados pelo membro superior para auxiliar usuários de próteses no treinamento. Além disso, técnicas de interação humano-computador mais intuitivas também são desenvolvidas a fim de proporcionar um ambiente de trabalho mais natural para profissionais da referida área. Para atingir o objetivo acima, as seguintes metas são propostas:

- Analisar, destacando suas principais características e limitações, projetos destinados à criação de modelos músculo-esqueléticos e à simulação de movimentos do corpo humano;
- Pesquisar tecnologias de apoio designadas ao desenvolvimento de um sistema interativo com ambiente virtual para possibilitar a modelagem e transformações geométricas atendendo às solicitações do usuário;
- Desenvolver as bibliotecas necessárias para o modelo músculo-esquelético;
- Desenvolver um sistema de simulação de movimentos de prótese para membro superior humano;
- Validar o sistema de simulação de movimentos do membro superior por meio de estudos de caso.

1.3 Simulação de movimentos humanos

Nestes últimos anos, pesquisadores das áreas de engenharia, biologia, educação física e medicina iniciaram o uso de ferramentas computacionais dentro da prática da medicina. Devido ao aumento da capacidade de processamento dos computadores, modelagem e simulação computacional tem se tornado cada vez mais importante no desenvolvimento tecnológico e científico [Székely e Satava 1999], permitido assim o desenvolvimento e análise de modelos mecânico-biológicos altamente sofisticados e também a aquisição de novos conhecimentos sobre o desempenho do corpo humano [Wojcik 2003].

Um dos grandes desafios na modelagem de humanos virtuais é conseguir uma representação das principais características humanas com o maior realismo possível. Para isto, muitas pesquisas envolvendo modelagem de movimentos humanos, deformação de corpos, dentre outros assuntos, que possibilitam análises e simulação de muitas estruturas envolvendo humanos estão em desenvolvimento. Estas simulações são especialmente utilizadas para gerar informações de modelos ou reproduzir o comportamento que poderia ser observado em situações reais [MAUREL 1998].

Um outro fator importante envolvendo modelagem e simulação é o uso de técnicas de realidade aumentada, uma tecnologia que permite misturar imagens dos mundos real e virtual que visa uma maior imersão do usuário com o ambiente ou cenário da aplicação. Entretanto, pode-se considerar que a curto e médio prazo "todo o tipo de informação médica venha a ser visualizado diretamente sobre o paciente, como sobreposição de estruturas geradas por computador ao paciente real" [Truppe 1995].

Existem diversas aplicações para o uso de modelagem e simulação computacional na engenharia biomédica, medicina, fisioterapia, e em outras áreas relacionadas a saúde. Seria praticamente impossível enumerar todas as aplicações disponíveis, porém, como exemplo, são citados:

- Sistemas de análise de movimento para facilitar a avaliação e análise da postura humana [Ditunno 1997];
- Uso de controle muscular computadorizado para gerar simulações dinâmicas da caminhada humana [Thelen e Anderson 2005];
- Modelos tridimensionais do fluxo sanguíneo no sistema vascular cerebral [Moore et al. 2005];
- Modelagem do joelho completo por elementos finitos [Halloran, Petrella e Rullkoetter 2005];
- Propriedades mecânicas da massa cerebral in-vivo [Miller et al. 2000];
- Simulações dinâmicas de movimento usando controle muscular [Thelen, Anderson e Delp 2003];

- Estudo de salto vertical partindo de diferentes posturas [Selbie e Caldwell 1996];
- Visualização por meio de técnicas de Realidade Aumentada para cirurgia de Laparoscopia [Fuchs et al. 1998].

1.4 Organização da Dissertação

Este trabalho está organizado da seguinte forma: o próximo capítulo apresenta o estado atual de programas computacionais para visualização, análise e modelagem de sistemas músculo-esquelético. O Capítulo 3 apresenta um estudo sobre a anatomia do movimento humano e o sistema de equilíbrio de corpos. No Capítulo 4 são apresentadas as tecnologias de apoio para desenvolvimento do projeto e uma proposta de arquitetura do sistema. A seguir, no Capítulo 5 são descritos a especificação e implementação do sistema baseada na arquitetura definida no Capítulo 4. No Capítulo 6 são apresentados estudos de caso para análise e validação do sistema proposto. As conclusões e trabalhos futuros são apresentados no último Capítulo dessa dissertação.

Capítulo 2

Trabalhos Relacionados

2.1 Introdução

Muito esforço científico foi dedicado a modelagem músculo-esquelético e de movimento. De acordo com [Huijing 1995] este esforço iniciou-se no século XVII com Stensen em 1667 e Borelli em 1680, e continua até o presente momento com o desenvolvimento das facilidades computacionais.

Com o intuito de mostrar as características de projetos destinados à criação de modelos músculo-esqueléticos e à simulação de movimentos do corpo humano, este capítulo apresenta um estudo de programas computacionais para visualização, análise e modelagem dos referidos modelos. Ao final do capítulo, um quadro comparativo elaborado após esta revisão, mostra as principais características e limitações dos sistemas estudados e que nortearam o direcionamento desta dissertação.

2.2 Características dos Sistemas Estudados

2.2.1 SIMM (Software for Interactive Musculoskeletal Modeling)

SIMM TM (Musculographics Inc., USA) é um software comercial para desenvolvimento e análise de modelos músculos-esqueléticos realísticos [DELP e LOAN 2000]. Este sistema ajuda os usuários na criação de modelos que quantificam os efeitos da geometria músculo-esquelético, cinemática de articulações e parâmetros músculo-tendão nos tamanhos dos músculos, braços de momento, forças dos músculos e momentos das articulações [DELP e LOAN 1995].

O SIMM (Figura 2.1) foi desenvolvido em linguagem C e, requer ao menos 16MB de memória. Suas principais características são:

- ser genérico o suficiente para que uma variedade grande de estruturas músculo-esquelético sejam modeladas,
- prover modelos realísticos de músculo, tendão, e ligamentos, e permitir uma especificação acurada da cinemática das articulações,
- prover um ambiente gráfico interativo permitindo que modelos sejam visualizados, alterados, testados e analisados, e
- ser extensível para que novas capacidades possam ser adicionadas ao software.

O sistema SIMM permite ao usuário avaliar um ou mais modelos músculo-esqueléticos através da leitura de um conjunto de arquivos de ossos, arquivos de articulações e arquivos de músculos (Figura 2.2). O módulo *model maker* percorre as informações nesses arquivos e cria uma estrutura de dados que representa o modelo músculo-esquelético.

Os músculos atuadores desenvolvem força e o torque sobre as articulações. O SIMM calcula o tamanho e o braço de momento de cada músculo modelado. O músculo tem um braço de momento para cada plano de movimento em todas as articulações que o músculo

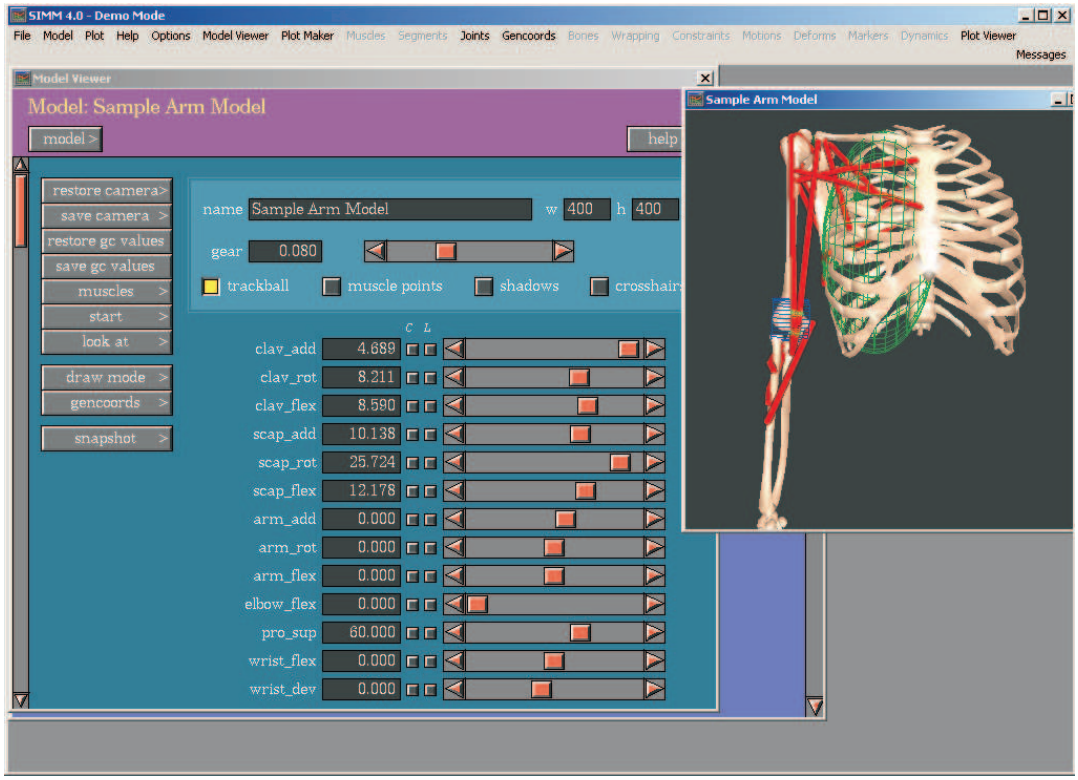

Figura 2.1: Tela do sistema SIMM

percorre. Por exemplo, devido ao músculo bíceps braquial tangenciar o cotovelo, o ombro e a articulação rádio-ulnar, ele tem um braço de momento para o ângulo do cotovelo e para cada grau de liberdade no ombro e na articulação rádio-ulnar.

Para produzir uma animação, o usuário cria um arquivo de movimentos contendo uma seqüência de ângulos das articulações descrevendo um movimento. Uma vez carregado pelo programa, o arquivo de movimentos pode ser usado para alterar múltiplos graus de liberdade, animar o modelo e prover novas variáveis para geração de relatórios.

Entretanto, o sistema SIMM apresenta as seguintes limitações:

- Não permite alterações dos valores de excitação do músculo em tempo real durante uma animação. Ele permite ao usuário carregar o modelo de movimento escrito em linguagem C para posterior análise. Esta limitação impede a simulação de movimentos respondendo às forças aplicadas pelos músculos através da análise de equilíbrio dos membros.

- O usuário precisa ter um conhecimento da linguagem C para produzir os modelos

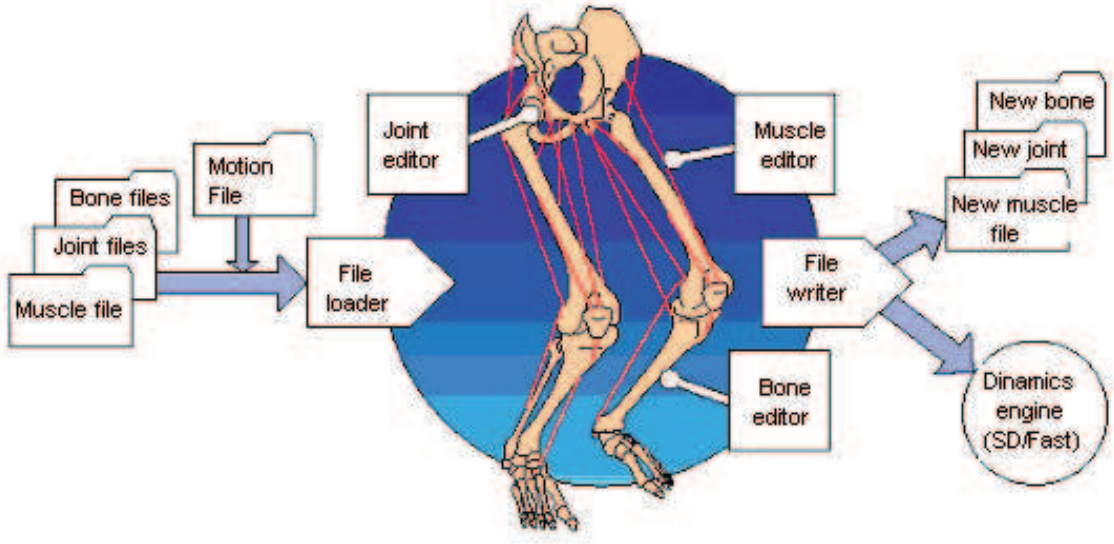

Figura 2.2: Estrutura do SIMM [DELP e LOAN 2000]

músculo-esquelético, dificultando o uso do sistema por profissionais da área de saúde.

2.2.2 CHARM - Comprehensive e Human Animation Resource Model

O European Esprit Project CHARM, envolvendo um grande número de universidades, propôs o desenvolvimento de uma completa base de dados de recursos humanos, permitindo a simulação dinâmica do complexo sistema músculo-esquelético [CHARM 1993]. Foram escolhidos os modelos do braço humano e ombro na fase inicial do projeto.

O modelo biomecânico do membro superior humano foi modelado incluindo propriedades biomecânicas como ossos, articulações e linha de ação dos músculos como requerimento para análise dinâmica [Helm et al. 1992].

Este módulo do projeto foi criado para auxiliar médicos e fisioterapeutas na visualização de movimentos pré-definidos. A entrada de dados é uma solicitação ao sistema em linguagem natural usando vocabulário médico (ex.: pronation, abduction, etc) . O sistema analisa o movimento solicitado e mostra o resultado na tela (veja Figura 2.3). O resultado é a nova posição do membro superior humano mostrado através de visualização 3D.

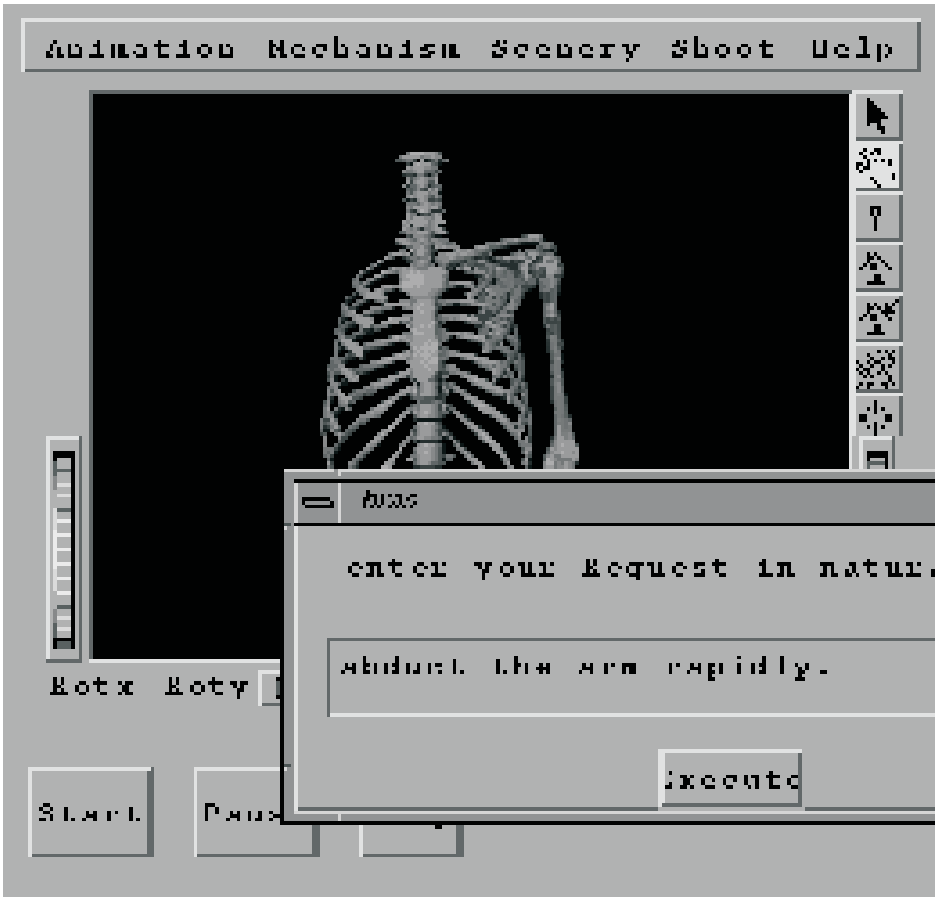

Figura 2.3: Interface do CHARM, módulo para médicos e biomecânicos [Multon et al. 1999]

Cada movimento pré-definido é determinado por: partes do corpo, amplitude, velocidade, direção e subseqüência de movimentos elementares. Um exemplo de entrada de dados é a sentença: *raise the arm rapidly* (levante o braço rapidamente). Os atributos para velocidade e amplitude podem ter valores qualitativos através de um conjunto de funções *slowly, normally, rapidly, strongly, completly, moderally, sligtly.*(lentamente, normalmente, rapidamente, mais rapidamente, completamente rápido, moderadamente, um pouco). Cada um desses atributos tem uma função correspondente que expressa o grau de ajuste como mostrado na Figura 2.4

O modelo simula movimentos do braço com 5 DOFs (degrees of freedom - graus de liberdade) e tem a limitação de apresentar apenas movimentos pré-definidos, o que inviabiliza o acompanhamento do movimento do usuário em tempo real.

O sistema tem interface amigável utilizando linguagem natural, mas não permite

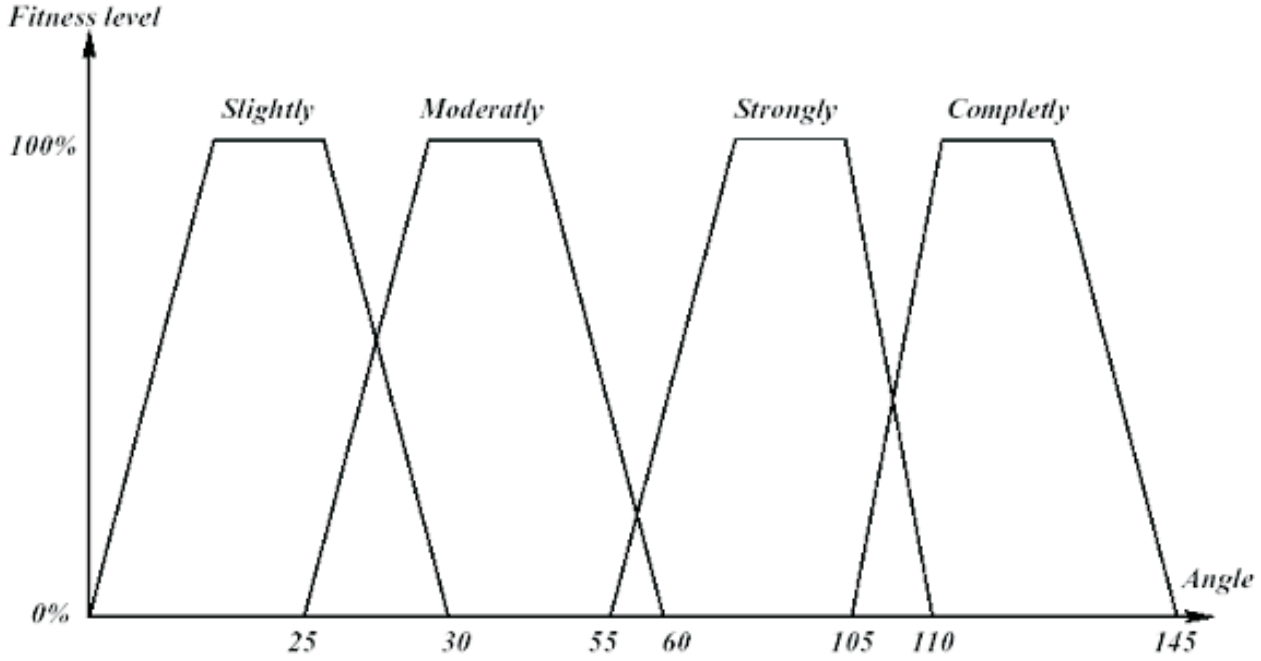

Figura 2.4: Variáveis lingüísticas usadas para movimentar o cotovelo [Multon et al. 1999]

alteração dos parâmetros de movimento em tempo real e também não simula os movimentos em função da força aplicada pelo músculo. Essas limitações fazem com o que o usuário deva previamente fazer um estudo das forças aplicadas pelos músculos, calcular os ângulos obtidos em cada evento de alteração da força e só então gerar o modelo para uma posterior análise do movimento.

2.2.3 MMS (Musculoskeletal Modeling in SimulinkTM)

MMS é um conjunto de scripts para MATLAB(software para cálculo numérico e análise de funções [MATLAB 1994]) e arquivos C que são adicionados ao processo normal de construção de modelos no SIMM para gerar modelos para o Simulink e para acrescentar uma funcionalidade adicional que permite ao usuário definir os modelos sem que tenha que escrevê-los em código C [Davoodi e Loeb 2001](Figura 2.5).

Os modelos músculo-esqueléticos são gerados pelo sistema SIMM, sendo eles : a) O modelo esquelético incluindo os segmentos, articulações, DOF's; b) O modelo muscular contém as dimensões do músculo, pontos de inserção, linha de ação e modelos de produção

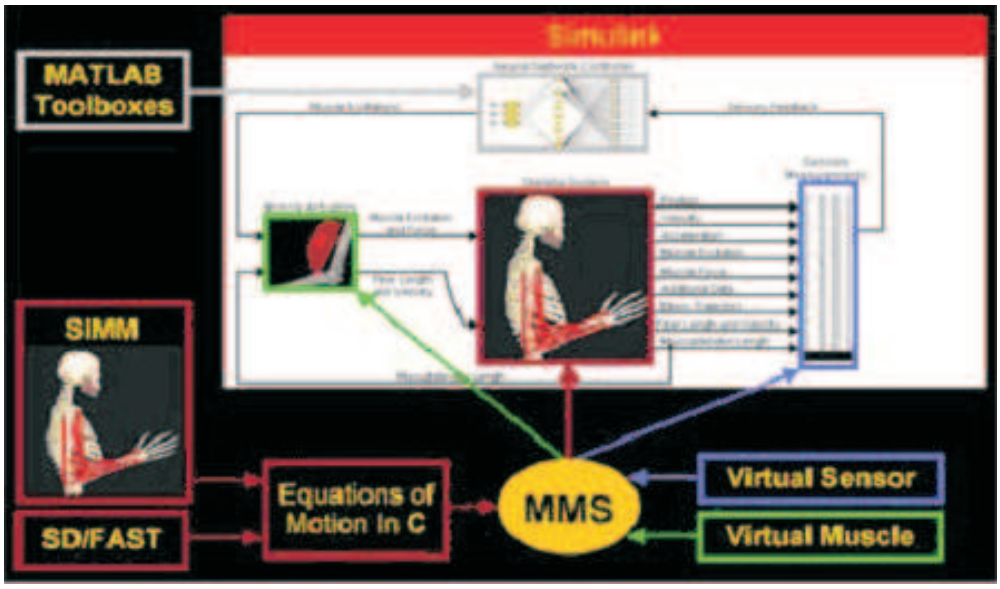

Figura 2.5: MMS é um software de construção de modelos computacionais do sistema músculo-esquelético no Simulink (Mathworks Inc., USA)

de força. As forças externas e os torques são definidos nos arquivos de movimento, preparados de acordo com o sistema SIMM. Nesse modelo são gravados os valores de força e torque para cada quadro. A quantidade de quadros, os espaços de tempo entre os quadros e tempo total são fixados.

Uma vez preparados os arquivos de modelo de entrada no SIMM, os scripts MMS são executados na janela de comando do Matlab, resultando na criação de arquivos contendo os modelos de simulação dinâmica para o Simulink. Esses modelos contêm as entradas e saídas de acordo com o arquivo lido e as questões no processo de construção de modelo interativo.

O MMS produz os quatro tipos de saídas, mas as proeminentes a esse projeto são: a) saídas para músculos - para cada músculos obtém-se a excitação, força e o tamanho; b) saídas para registro e animação - nesse caso o sistema gera um arquivo Matlab chamado "motdata.mat", usado para gerar o arquivo de movimento compatível com o SIMM para visualização da simulação.

Depois de ter gerado os blocos para o Simulink, o usuário somente precisa conectar as entradas e saídas do bloco para as fontes e destinos apropriados e executar a simulação.

Embora o MMS seja uma ferramenta que apresenta facilidades na simulação e aná-

lise de movimentos dos membros do corpo humano, ele apresenta as seguintes limitações: a) faz-se necessário o uso de dois ambientes, o SIMM e o Simulink, para criar os modelos músculo-esqueléticos e obter os resultados para análise e simulação gráfica do movimento; b) somente é possível ver a animação do movimento ao final da simulação através da leitura de um arquivo gerado pelo simulink.

2.2.4 BodySim

Maciel A., Nedel L. e Freitas C. em 2002, desenvolveram um modelo de articulações, embasado em características anatômicas, para representar modelos esqueléticos com o auxílio de técnicas de projeto orientado a objetos. Este modelo foi criado com o objetivo de disponibilizar um conjunto de classes para modelagem de articulações no contexto do projeto Vpat (Virtual Patients) (Figura 2.6) do Grupo de Pesquisa em Computação Gráfica e Processamento de Imagens da UFRGS (Figura 2.7). O projeto Vpat tem por objetivo a construção de um ambiente para modelagem, visualização e simulação de pacientes virtuais.

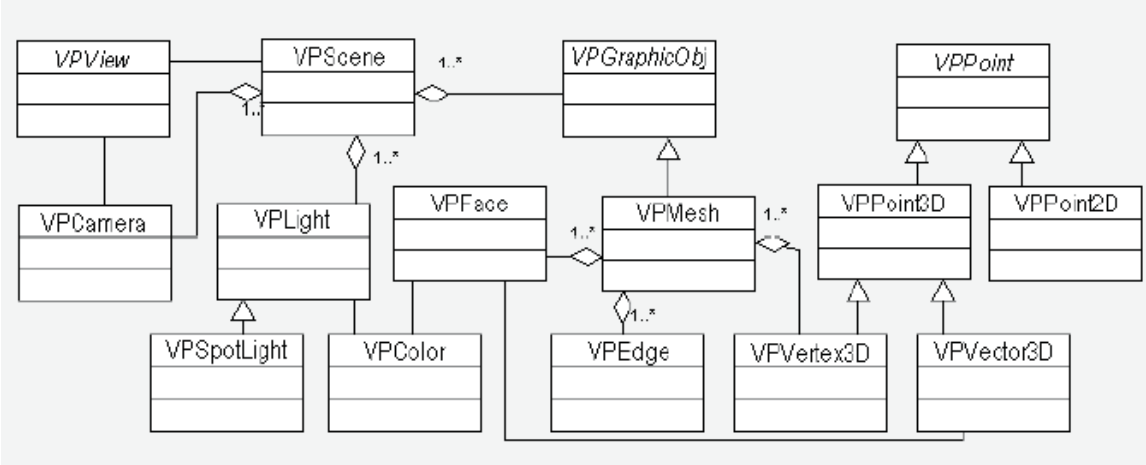

Figura 2.6: Diagrama de classes básicas do framework. Fonte: [Freitas et al.]

Para que pudessem verificar o funcionamento do modelo proposto, desenvolveram o BodySim (Figura 2.8) em ambiente Linux Red Hat utilizando a linguagem C++. Esse aplicativo recebe como entrada a descrição de um corpo humano articulado - ou parte de um corpo - e a especificação de uma seqüência de movimentos. A partir disso, o programa

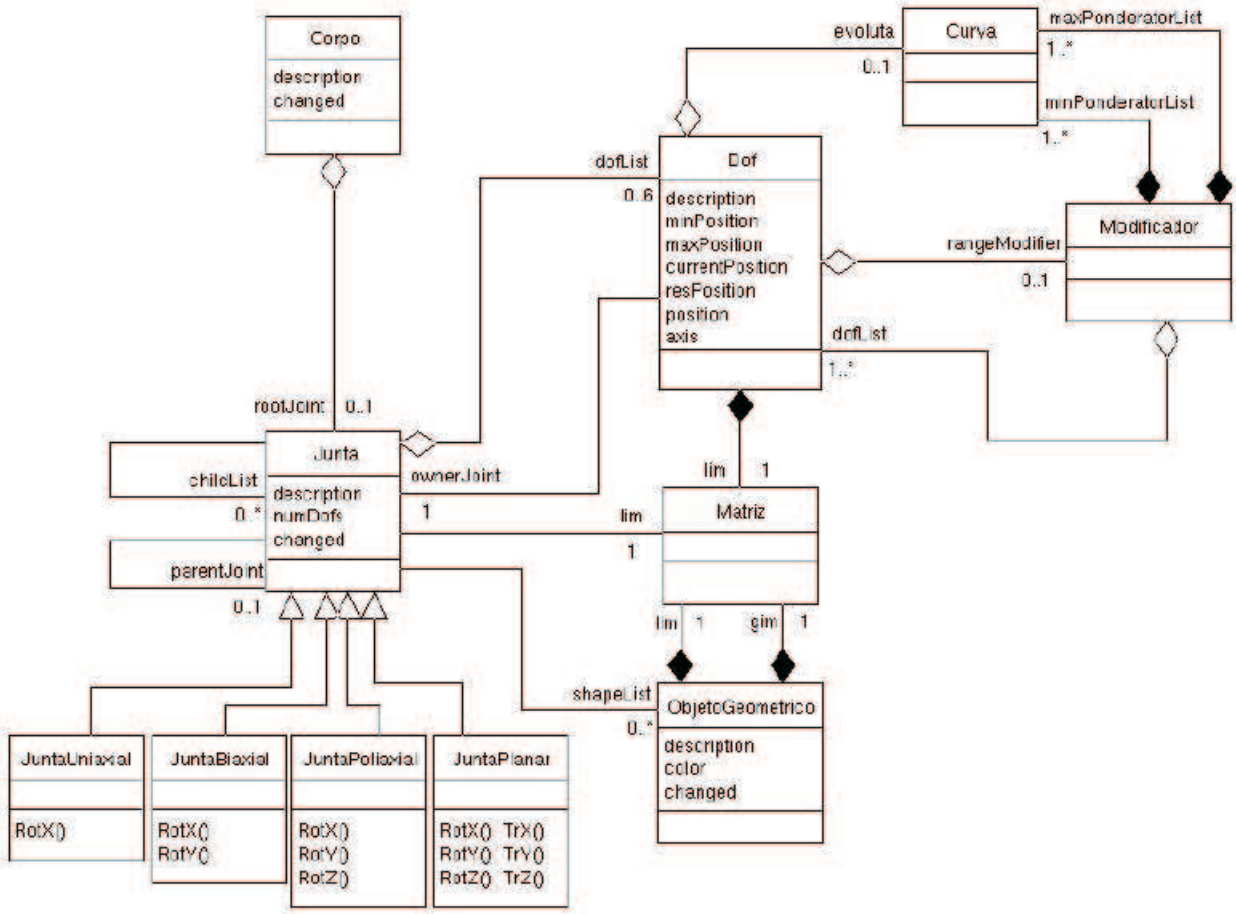

Figura 2.7: Diagrama de Classes BodySim [Maciel, Nedel e Freitas 2002]

realiza a simulação dos movimentos especificados, apresentando uma animação do modelo 3D em uma janela gráfica onde o usuário pode modificar interativamente os parâmetros de câmera [Maciel, Nedel e Freitas 2002].

O BodySim permite uma boa visualização da simulação dos movimentos, mas não permite ao usuário a alteração dos parâmetros em tempo real e também não simula os movimentos em função da força aplicada pelo músculo.

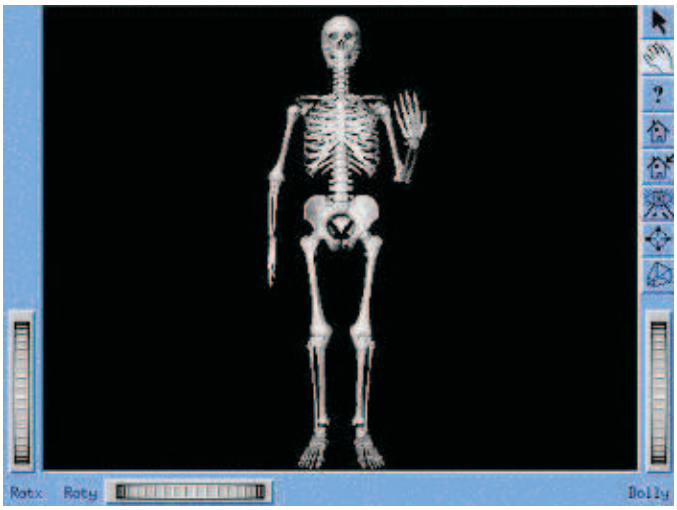

Figura 2.8: Interface do BodySim [Maciel, Nedel e Freitas 2002]

2.3 Quadro Comparativo dos Sistemas

Para melhor assimilar as diferenças dos projetos apresentados, foi elaborado um quadro comparativo (Tabela 2.1) com as principais características de cada um em relação à simulação de movimentos do membro superior humano.

Tabela 2.1: Comparação dos Projetos estudados

Principais Características	**SIMM**	**CHARM**	**MMS**	**BodySim**
Visualização gráfica em 3D	x	x	x	x
Ambiente gráfico interativo	x	x	x	x
Reprodução de arquivos de movimentos do membro superior humano.	x	x	x	x
Permissão para alterações nos parâmetros de movimento em tempo real			x	
Simulação dos movimentos a partir do valor da força aplicada pelo músculo	x		x	
Criar os modelos e visualizar o resultado da simulação no mesmo ambiente	x	x		x

Tabela 2.1: Comparação dos Projetos estudados(continuação)

Principais Características	**SIMM**	**CHARM**	**MMS**	**BodySim**
Exibição da animação em tempo real respondendo às alterações dos valores das forças aplicadas pelo músculo				

2.4 Sumário e Conclusões

A análise dos trabalhos citados nesse capítulo, permite evidenciar algumas características relevantes à concepção de projetos de sistemas de simulação de movimentos.

O SIMM apresenta um ambiente gráfico interativo, modelos músculo-esqueléticos realísticos e permite ao usuário criar arquivos de movimentos contendo uma seqüência de ângulos das articulações descrevendo movimentos. O CHARM tem uma interface amigável e utiliza linguagem natural na simulação dos movimentos. O MMS utiliza os modelos músculo-esqueléticos do SIMM e faz a simulação no Simulink por meio dos arquivos contendo modelos de simulação dinâmica.

Analisando as características de cada software e considerando as limitações apontadas na Tabela 2.1, pode-se perceber que alguns deles não permitem alterações dos valores de estimulação dos músculos em tempo real na simulação. Tomando-se como referência as informações obtidas nesta análise, verifica-se a necessidade de desenvolver um sistema que crie um modelo músculo-esquelético e possibilite a simulação dos movimentos em tempo real, ou seja, realizando movimentos no mesmo instante em que ocorrem alterações nos valores de forças aplicadas pelos músculos.

Capítulo 3

Princípios Biomecânicos

3.1 Introdução

Este capítulo apresenta um estudo das estruturas anatômicas do corpo humano que permitem variados tipos de movimentos, examina de que maneira os músculos atuam para realizar a diversidade de movimentos de que é capaz e, determina o modelo do sistema de equilíbrio de forças.

3.2 Anatomia do Movimento

Como a estrutura anatômica afeta as capacidades de realizar movimentos das articulações da extremidade superior do corpo humano? Para responder essa pergunta são apresentados os estudos das articulações e dos músculos e, os fatores antropomórficos dos segmentos corporais dessa estrutura.

3.2.1 Biomecânica das Articulações

As articulações do corpo humano são responsáveis pela orientação dos movimentos dos segmentos corporais. Nas seções seguintes, serão descritos os aspectos anatômicos e biomecânicos das articulações do membro superior.

Classificação das Articulações

Planar, Deslizante ou Artródia

As articulações desse tipo (Figura 3.1) permitem movimentos de translação e rotação de um osso contra o outro. As superfícies ósseas articuladas são quase planas e o único movimento permitido é o deslizamento de umas sobre as outras. São permitidos seis graus de liberdade de movimento: rotação em torno dos três eixos de coordenadas e translação em três direções, porém, todos com muito pouca amplitude.

Exemplo: articulações planares estão entre os ossos do carpo (na mão) e do tarso e do metatarso (no pé). Na mão são permitidos os movimentos: flexão, extensão, abdução, adução,e pequenas rotações nos sentidos horário e anti-horário. No pé, os movimentos são: flexão, extensão, abdução, adução, supinação e pronação.

Figura 3.1: Articulação Planar, Fonte:[Anatomy of Human Organ Systems 2004]

Uniaxial

As articulações deste tipo possuem apenas um eixo de rotação no movimento, sendo assim permitem somente um grau de liberdade. Dois sub-tipos podem ser considerados:

Dobradiça, Trocleartrose ou Gínglimo Uma articulação deste tipo (Figura 3.2) permite rotação em torno de um eixo perpendicular ao comprimento dos ossos envolvidos. A superfície óssea articulada é côncava, enquanto que a outra é convexa. Essa forma é responsável pelo movimento, tipo dobradiça, que é limitado a flexão e extensão dos cotovelos, dedos das mãos e dos pés

Exemplo: articulação ulnoumeral (cotovelo).

Figura 3.2: Articulação tipo dobradiça. Fonte:[Anatomy of Human Organ Systems 2004]

Pivô ou Trocóide A articulação do tipo pivô (Figura 3.3) possibilita o movimento angular em torno de um único eixo, permitindo que um osso se movimente girando em relação ao outro

Figura 3.3: Articulação tipo pivô. Fonte:[Anatomy of Human Organ Systems 2004]

Exemplo: rádio-ulnar (antebraço);

Biaxial

Permitem movimentação em torno de dois eixos, suportando dois graus de liberdade no movimento. Três sub-tipos podem ser considerados:

Esferoidal Apresentam duas superfícies paralelas e arredondadas que se articulam com superfícies planas. Nesse tipo de articulação é permitida a rotação em torno dos três eixos de coordenadas.

Exemplo: articulação do quadril e do ombro;

Elipsóide Uma superfície óssea convexa em um dos ossos se articula com outra superfície com formato côncavo (Figura 3.4). Esta classe de articulação é muito parecida e permite todos os tipos de movimento de uma articulação poliaxial esferóide, exceto rotação, que é evitada pela forma elipsoidal das superfícies articulantes.

Figura 3.4: Articulação tipo Elipsóide. Fonte:[Anatomy of Human Organ Systems 2004]

Exemplo: Articulação que envolve o tubérculo articular do osso temporal, a fossa mandibular e o côndilo da mandíbula.

Sela Uma junta do tipo sela permite os mesmos tipos de movimento das juntas elipsoidais. A diferença está na forma da articulação. Cada extremidade articulante em uma junta deste tipo tem dupla curvatura, e quando colocadas juntas, a curvatura convexa de uma extremidade encaixa-se na curvatura côncava da outra(Figura 3.5).

Exemplo: carpo-metacarpiana do polegar (permite o movimento de oposição do polegar aos outros dedos);

Figura 3.5: Articulação tipo Sela. Fonte:[Anatomy of Human Organ Systems 2004]

Poliaxial

Este tipo de articulação (Figura 3.6) apresenta três graus de liberdade. As articulações poliaxiais humanas são formadas por uma superfície óssea articulada esférica em um dos ossos e outra côncava no outro. Permite amplo movimento em todos os planos, incluindo flexão/extensão, abdução/adução, rotação e circundação. O alcance do movimento depende muito da profundidade do encaixe côncavo. Um encaixe menos profundo, como o do ombro, aumenta a variedade de movimentos possíveis. Por outro lado,

um encaixe mais profundo, como o do quadril, aumenta a estabilidade da articulação [Maciel 2001].

Figura 3.6: Articulação Poliaxial. Fonte:[Anatomy of Human Organ Systems 2004]

Exemplo: Escápulo-umeral (ombro).

3.2.2 As articulações do membro superior

Essa seção exibe uma breve descrição das articulações do membro superior humano(Figura 3.7).

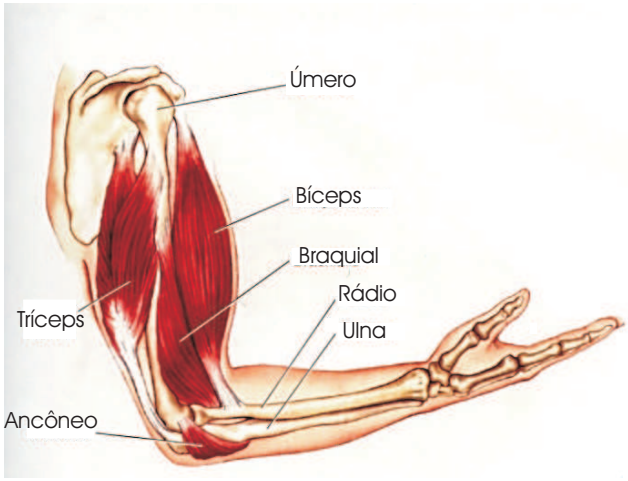

Figura 3.7: Músculos do braço (Bíceps e Tríceps). Fonte: [Les muscles 1999]

- Ombro

  É um conjunto funcional que permite unir o membro superior ao tórax. Os movimentos globais do ombro são: elevação, abaixamento, abdução, adução, báscula medial, báscula lateral, flexão, extensão, rotação externa e rotação interna. Porém, para este trabalho, apenas os movimentos: flexão, extensão, abdução e adução e os

músculos relacionados à articulação escapuloumeral, foram escolhidos como objeto de estudo.

A Tabela 3.1 mostra os músculos envolvidos nessa articulação, bem como seus pontos de origem e inserção, e os movimentos realizados por cada um;

Tabela 3.1: Músculos e movimentos do ombro

Músculo	**Origem**	**Inserção**	**Movimento**
Supra-espinhal	-Escápula	- Úmero (tubérculo maior)	Abdução do braço
Bíceps branquial	-Parte inferior da cavidade glenóide da escápula - Processo coracóide	- Parte alta do rário	Flexão do braço e antebraço
Peitoral maior	- clavícula - externo - cartilagens costais	Úmero (sulco intertubercular)	Adução do braço
Deltóide	- acrômio - escápula - clavícula	Úmero (face lateral)	Flexão do braço Abdução do braço Extensão do baço

- Cotovelo

  É uma articulação que realiza dupla função: a)flexão-extensão que permite ao membro superior dobrar-se sobre si mesmo ou estender-se; b)prono-supinação que é, em parte, a sede dos movimentos que permitem ao antebraço girar sobre seu eixo [Calais-German 1992].

  Os músculos e movimentos relacionados ao cotovelo estão descritos na Tabela 3.2.

Tabela 3.2: Músculos e os movimentos do cotovelo

Músculo	**Origem**	**Inserção**	**Movimento**
Braquial	- Úmero (face anterior, metade anterior)	-Ulna(processo coronóide)	Flexão do Antebraço
Braquiorradial	- Úmero (borda lateral, terço inferior)	- Rádio (processo estilóide)	Flexão do Antebraço Supinação do Antebraço
Bíceps braquial	-Escápula (cavidade glenóide) -Escápula (processo coracóide)	- Rádio (parte alta)	Flexão do Antebraço
Tríceps braquial	- Escápula (cavidade glenóide) - Úmero (face posterior, metade superior) - Úmero (face posterior, metade inferior)	- Ulna (olecrano)	Extensão do Antebraço
Ancôneo	- Úmero (faze posterior)	- Ulna (face posterior, quarto superior)	Extensão do Antebraço
Pronador redondo	- Úmero (epicôndilo medial) - Ulna (face anterior do processo coronóide)	- Rádio (face lateral)	Pronação do antebraço
Pronador quadrado	- Ulna (face anterior)	- Rádio (face anterior)	Pronação do Antebraço
Supinador	- Ulna (epicôndilo lateral e parte alta)	- Rádio (extremidade superior)	Supinação do Antebraço

- Punho e Mão

  Nesta região existe um grande número de articulações. Começando pelo punho, articulação radiocarpal do tipo elipsóide. Nessa articulação ocorrem os movimentos do tipo abdução e adução, e parte dos movimentos do tipo flexão e extensão do punho, o restante ocorre na articulação mediocarpal.

  Nas articulações carpometacapianas(planas) do segundo ao quinto dedos, os movimentos de flexão, extensão e rotação promovem alteração no arco transverso da mão.

  Os movimentos do polegar, oponência, superfície palmar do polegar da ponta do polegar é dirigida para a superfície palmar dos outros dedos, e reposição, inverso à oponência, são permitidos pelas articulações carpometacarpiana do polegar(sela).

  Os movimentos flexão, extensão, abdução e adução dos dedos são permitidos pela articulação metacarpofalangeanas, porém, com amplitude de movimento mais limitada no polegar [Maciel 2001].

  Os músculos e os movimentos relacionados ao punho e mão estão descritos na Tabela 3.3.

Tabela 3.3: Músculos e os movimentos do punho e mão

Músculo	**Origem**	**Inserção**	**Movimento**
Flexor radial do carpo	- Epicôndilo medial	-Segundo metacárpio	Flexão do Punho
Palmar longo	- Epicôndilo medial	- Retináculo flexor do carpo	Flexão do Punho
Flexor ulnar do carpo	- Epicôndilo medial, - Olécrano	- Pisiforme	Flexão do Punho

Tabela 3.3: Músculos e os movimentos do punho e mão(continuação)

Músculo	**Origem**	**Inserção**	**Movimento**
Radiais	- Úmero	-Face dorsal da mão	Extensão do Punho
Extensor radial do carpo	- Borda lateral do úmero	-Base do segundo metacarpo	Extensão do Punho
Extensor radial curto do carpo	- Epicôndilo lateral	-Base do terceiro metacarpo	Extensão do Punho
Extensor ulnar curto do carpo	- Epicôndilo lateral - Borda posterior - da ulna	-Base do quinto metacarpo	Extensão do Punho
Flexor profundo dos dedos	- Face anterior da ulna - Membrana interóssea	-Base do falange distal dos dedos	Flexor da falange distal dos dedos
Flexor superficial dos dedos	- Epicôndilo medial.	- Face palmar da falange	Flexão da falange média
	- Borda anterior do rádio	média	
Extensor dos dedos	- Parte baixa do úmero	- Base da falange proximal - Base da falange média - Base da falange distal	Extensão dos dedos

Tabela 3.3: Músculos e os movimentos do punho e mão(continuação)

Músculo	**Origem**	**Inserção**	**Movimento**
Extensor do do indicador	- Face posterior da ulna	- Indicador	Extensão do indicador
Extensor do do dedo mínimo	- Parte inferior do úmero	- Dedo mínimo	Extensão do dedo mínimo
Interósseos	- Próximo ao dorso da mão	- Base da falange proximal - Base da falange proximal - Falange média	Abdução dos dedos
Lumbricais	- Tendões do flexor profundo dos dedos	Tendões do externo dos dedos	Flexão das articulações metacarpofalangeanas
Oponente do do dedo mínimo	- Hamato - Retináculo flexor do carpo	- Quinto metacarpo	Rotação externa
Flexor curto do dedo mínimo	- Hamato - Retináculo flexor do carpo	- Base da falange proxima do dedo mínimo	Flexor da falange proxima do dedo mínimo
Adutor do dedo mínimo	- Pisiforme - Retináculo flexor do carpo	- Base da falange proxima do dedo mínimo	Adução do dedo mínimo

Amplitude dos Movimentos

A amplitude dos movimentos mede-se a partir da posição anatômica de repouso das articulações do membro superior, pendendo ao longo do corpo (0°). Cada tipo de movimento tem um limite de amplitude para cada plano de referência: a) plano sagital - divide o corpo verticalmente em metades direita e esquerda; b) plano coronal ou frontal - divide o corpo verticalmente em metades anterior e posterior e; c) plano horizontal ou transversal - separa o corpo em metades superior e inferior. Os limites da amplitude normal para os vários movimentos do ombro (com cotovelo em extensão) são:

- No plano sagital:

  Flexão do braço de 0° a 180°;

  Extensão do braço de 0° a 60°;

- No plano coronal:

  Adução do braço de 0° a 45°;

  Abdução do braço de 0° a 180°;

- No plano horizontal:

  Flexão horizontal do braço de 0° a 135°;

  Extensão horizontal do braço de 0° a 45°.

A medição da amplitude dos movimentos do cotovelo faz-se a partir da posição anatômica de repouso já descrita para o braço, ou seja, membro superior pendente ao longo do corpo (0°). A amplitude de flexão vai desde 0° até 145° (flexão completa do antebraço sobre o braço) [Hall 2000].

Tipos de Movimento

Flexão: Movimento de uma articulação que permite rotação em torno de um eixo perpendicular ao comprimento dos ossos envolvidos. Este movimento ocorre no plano

sagital e leva uma parte do corpo para frente da posição anatômica. Exemplo: úmero-ulnar (cotovelo);

Extensão: Movimento de uma articulação que permite rotação em torno de um eixo perpendicular ao comprimento dos ossos envolvidos. Este movimento ocorre no plano sagital e leva uma parte do corpo para trás da posição anatômica. Exemplo: úmero-ulnar (cotovelo);

Pronação: Movimento angular em torno do eixo dado pelo comprimento de um osso, permitindo que um osso se movimente girando em relação ao outro. Este movimento ocorre no plano transversal (horizontal) e leva uma parte do corpo para fora. Exemplo: rádio-ulnar (antebraço);

Supinação: Movimento angular em torno do eixo dado pelo comprimento de um osso, permitindo que um osso se movimente girando em relação ao outro. Este movimento ocorre no plano transversal (horizontal) e leva uma parte do corpo para dentro. Exemplo: rádio-ulnar (antebraço);

Adução: Movimento de rotação no plano frontal (coronal) que leva a região do corpo para a linha mediana do corpo. Exemplo: Um dos movimentos da articulação rádio-cárpica (punho);

Adução: Movimento de rotação no plano frontal (coronal) que leva a região do corpo para longe da linha mediana do corpo. Exemplo: Um dos movimentos da articulação rádio-cárpica (punho);

Circundução: Movimento que circunscreve uma área cônica, realizando em seqüência movimentos de flexão/extensão e abdução/adução. Este movimento ocorre nos planos: sagital, frontal (coronal) e transversal (horizontal) [Calais-German 1992]. Exemplo: Ombro.

3.2.3 Biomecânica do Músculo Esquelético Humano

O músculo é o único tecido capaz de desenvolver tensão ativamente. Essa característica permite ao músculo esquelético, ou estriado, realizar funções importantes para o desempenho eficiente do corpo humano.

Propriedades Comportamentais do Tecido Muscular

Irribitabilidade: É a capacidade para responder a estimulação;

Contratibilidade: É a capacidade de um músculo para encurtar-se através da estimulação do tecido muscular;

Extensibilidade: É a capacidade do músculo para alongar-se através de uma força externa;

Elasticidade: É a capacidade da fibra muscular para retornar ao seu comprimento de repouso após o alongamento do músculo [Hamill e Knutzen 1999].

Organização Estrutural do Músculo

Quando um músculo é tensionado, suas características anatômicas e fisiológicas são afetadas pela força gerada e pelo período de tempo que a força pode ser mantida. Para melhor compreensão do funcionamento da anatomia muscular, seus elementos básicos estão descritos a seguir:

Fibras Musculares A célula muscular é denominada fibra muscular , Figura 3.8 A membrana que circunda a fibra é chamada de sarcolema e o citoplasma (a parte interna da fibra, excluindo-se o núcleo) recebe a designação de sarcoplasma. O sarcoplasma de cada fibra contém inúmeras miofibrilas que estão alinhadas paralelamente umas as outras, conferindo ao músculo a designação de esquelético ou estriado.

O sarcômero é a unidade estrutural básica de fibra muscular onde ocorre a contração muscular. Cada sarcômero é dividido ao meio por uma linha M e é limitado por

duas linhas Z. As faixas A contém os filamentos grossos e rugosos de miosina e as faixas I contêm os filamentos lisos de actina. As linhas Z se movimentam na direção das faixas A, que mantém seu tamanho original, enquanto as faixas I tornam-se mais estreitas e a zona H desaparece (Figura 3.9). As projeções dos filamentos de miosina chamadas de pontes cruzadas formam elos físicos com os filamentos de actina durante a contração muscular, sendo que a quantidade desses elos é diretamente proporcional à produção de força e ao dispêndio de energia [Hall 2000].

O retículo sarcoplástico está associado externamente com cada fibra. Internamente, as fibras são atravessadas por túbulos transversos que passam através da fibra e se abrem externamente. O retículo sarcoplasmático e os túbulos transversos suprem os canais para o transporte dos mediadores eletroquímicos da ativação muscular [Hall 2000];

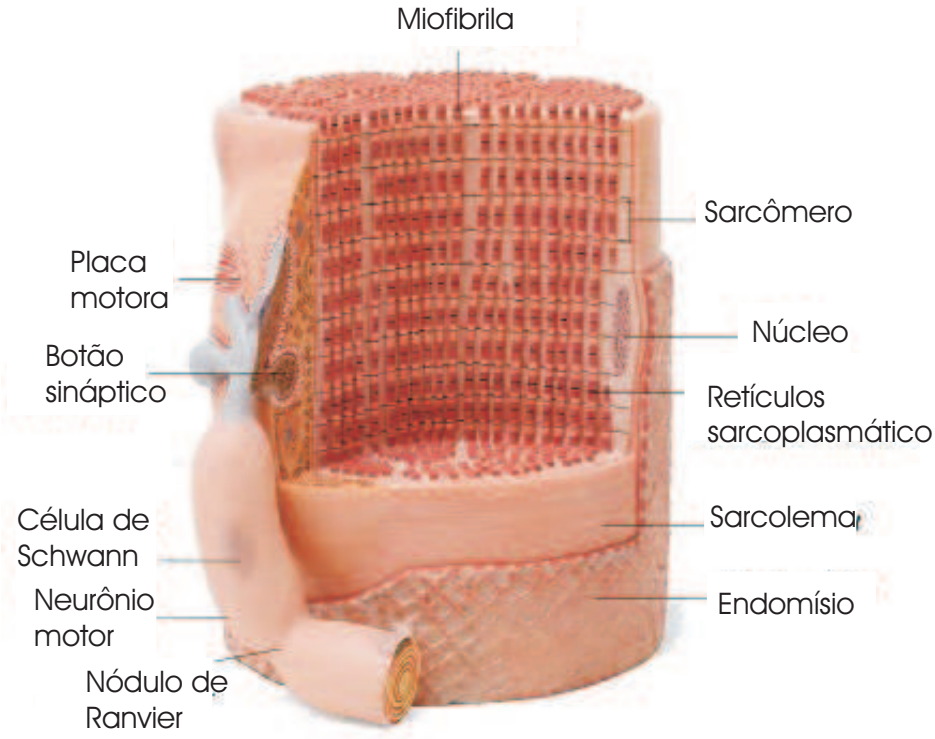

Figura 3.8: Fibra de músculo esquelético Fonte: [Rocha 2000].

Unidades Motoras As unidades motoras são grupos funcionais constituídos por um neurônio motor, fibra nervosa e por todas as fibras musculares por ele inervadas.

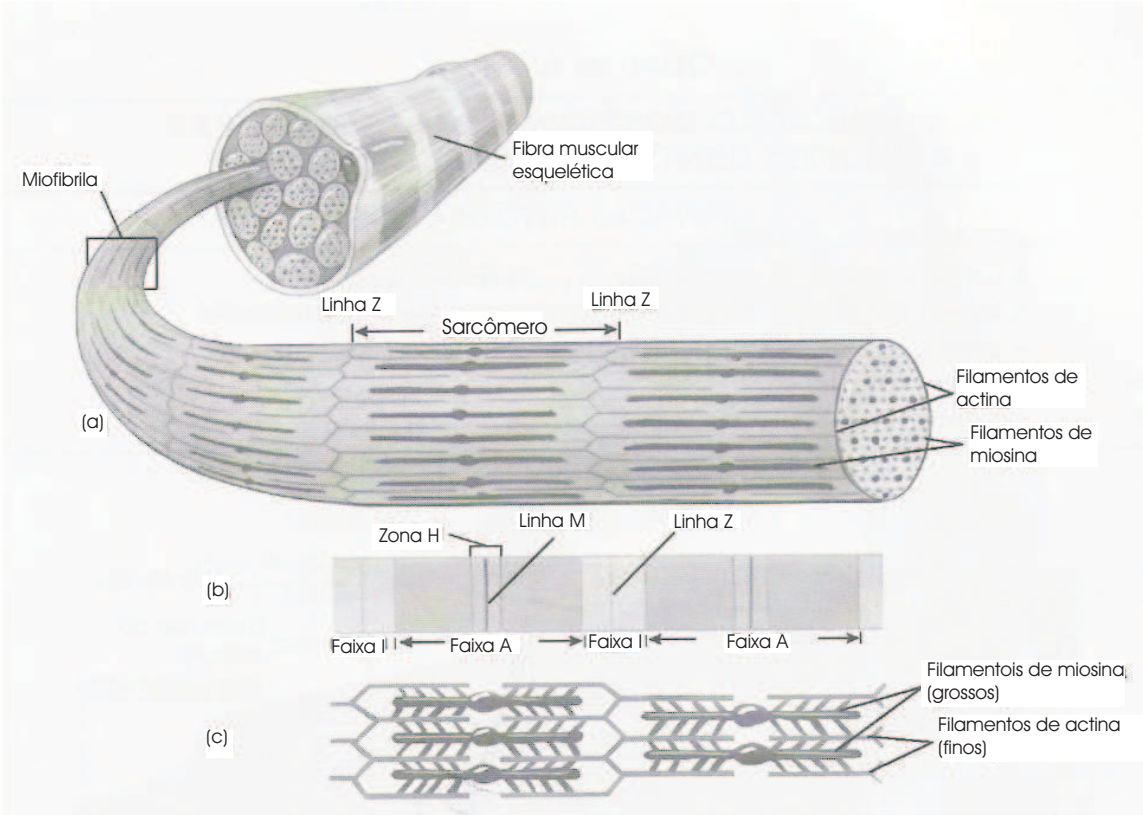

Figura 3.9: O sarcoplasma de uma fibra muscular. Fonte: [Hall 2000]

Cada fibra nervosa individual possui uma placa motora terminal devido ao fato de o axônio de cada uma se subdividir muitas vezes;

Arquitetura da Fibra Muscular O arranjo das fibras no músculo tem influência na função muscular no que diz respeito à capacidade de gerar força ou de encurtar-se. As orientações das fibras dentro de um músculo, bem como a forma pela qual elas se fixam aos tendões musculares, variam consideravelmente entre os músculos do corpo humano. Essas considerações estruturais afetam a força da contração muscular e a amplitude de movimento. Os dois tipos básicos de arranjo de fibras musculares são denominados paralelo (fusiforme) e oblíquo (peniformes).

Em um arranjo em paralelo, as fibras estão orientadas em paralelo à linha de tração dos músculos. Exemplos de músculos com fibras em paralelo é o bíceps braquial. Em um arranjo peniforme as fibras formam um ângulo com o eixo longitudinal do músculo, de modo que a força da fibra é em direção diferente da força muscular.

Cada fibra em um músculo peniforme se insere em um ou mais tendões. Exemplos de músculos com fibras peniforme são o deltóide e o reto femoral.

Em um músculo com fibras em paralelo, qualquer encurtamento causado por uma tensão representa o encurtamento de suas fibras. À medida que as fibras de um músculo peniforme se encurtam, o ângulo de inserção aumenta. Quanto maior o ângulo de inserção, menor será a força efetiva.

Os músculos peniformes têm a capacidade de produzir mais força que os músculos com fibras em paralelo de mesmo tamanho, pois os peniformes contém mais fibras por unidade de volume muscular. Por outro lado, os músculos com fibras em paralelo podem movimentar os segmentos corporais com amplitudes de movimento maiores que os peniformes de mesmo tamanho [Hall 2000].

Inserção Muscular

Existem três maneiras para o músculo inserir-se no osso: diretamente no osso (ex. coracobraquial), por um tendão (ex. bíceps braquial) ou por aponeurose (ex. extensor longo dos dedos), uma bainha do tecido fibroso, encontrada nos abdominais e na inserção do grande dorsal no tronco (Figura 3.10).

O músculo liga-se ao osso por suas duas ou mais extremidades. A ligação mais proximal em relação á linha média ao corpo é denominada origem. A ligação mais distal é denominada inserção [Hamill e Knutzen 1999].

Funções Desempenhadas pelo Músculo Esquelético

Os músculos esqueléticos realizam várias funções importantes ao desempenho eficiente do corpo humano, mas quando ativados só conseguem uma coisa: desenvolver tensão. Entretanto, considerando que um único músculo quase nunca atua isoladamente, são apresentadas as funções que os músculos realizam quando em combinação com outros músculos que atravessam a mesma articulação [Hamill e Knutzen 1999]:

Movimentador Primário vs. Movimentador Assistente: O músculo primariamente

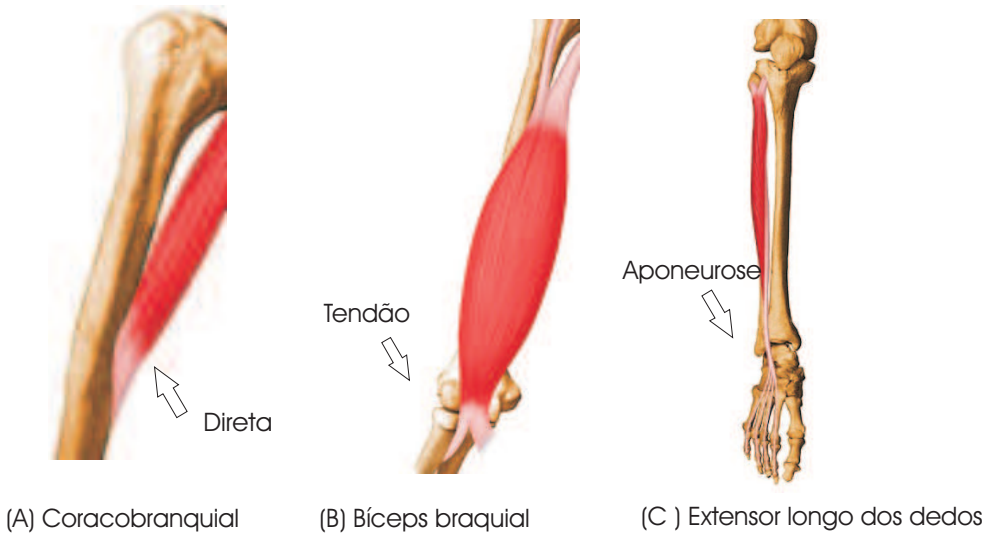

Figura 3.10: Um músculo pode inserir-se (A) diretamente no osso (B) no tendão ou (C) na aponeurose, Fonte: [Valdivia 2004]

responsável pela produção de um determinado movimento é chamado movimentador primário, e quando maior força é requerida outros músculos podem contribuir como movimentadores assistentes;

Agonistas e Antagonistas: Quando um músculo se contrai e acarreta o movimento de um segmento corporal em uma articulação, está agindo como agonista ou motor. Os músculos com ações opostas àquelas dos agonistas podem agir como antagonistas. Agonistas e antagonistas estão posicionados tipicamente em lados opostos de uma articulação;

Estabilizadores e Neutralizadores (Sinergistas): Outro papel desempenhado pelos músculos envolve a estabilização de uma parte do corpo contra uma força em particular. A força pode ser interna, proveniente da tensão em outros músculos, ou externa, como o peso de um objeto que está sendo levantado. Os neutralizadores, ou sinergistas, contraem-se para eliminar uma ação articular indesejada causada por outro músculo.

3.3 Equilíbrio e Movimento Humano

Quando os músculos em lados opostos de uma articulação desenvolvem tensão, o que determina a direção do movimento articular? A cinética tem a resposta a essa pergunta. Quando um músculo, passando por uma articulação, desenvolve tensão ele cria um torque na articulação através da produção de força sobre o osso onde é inserido. A maioria dos movimentos humanos envolvem a tensão simultânea nos músculos agonistas e antagonistas. A tensão nos antagonistas, além de controlar a velocidade do movimento, produz estabilidade da articulação no qual o movimento está ocorrendo. Como a tensão produzida pelo antagonista cria um torque na direção oposta ao torque produzido pelo agonista, o movimento resultante na articulação representa uma função do torque efetivo [Hall 2000].

Através da mensuração dos torques resultantes, pode-se determinar a direção do movimento na articulação. Vários fatores, como peso dos segmentos corporais, o movimento desses segmentos e a ação das forças externas, podem contribuir para os torques articulares efetivos. O torque é obtido através do produto da força pelo braço de momento dessa força [Hall 2000].

Na seção seguinte, é apresentado um modelo de geometria muscular, utilizado para capturar o valor do braço de momento de um músculo, que é a menor distância perpendicular entre a linha de ação do músculo e o centro da articulação (Figura 3.11). Em seguida serão determinados o braço de momento da força peso dos segmentos, e então apresentada a análise feita das somas de todas os torques na articulação envolvida até que atinja a condição de equilíbrio a cada movimento realizado pelo membro superior humano.

3.3.1 Cálculo do braço de momento e tamanho dos músculos

No momento em que o músculo desenvolve tensão, tracionando os ossos para sustentar ou deslocar a resistência criada pelo peso dos segmentos corporais, o músculo e o osso estão funcionando mecanicamente como uma alavanca. O tamanho e braço de

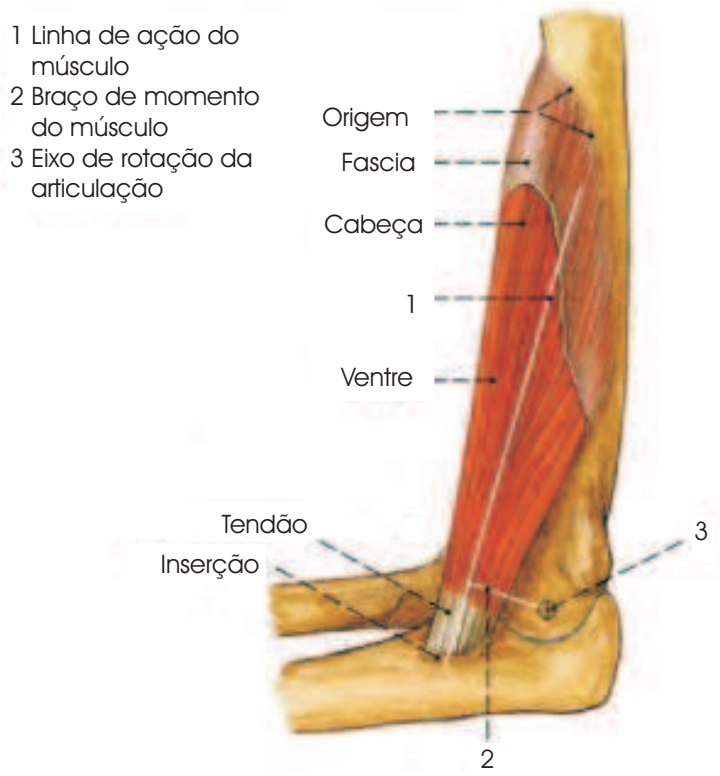

Figura 3.11: Linha de ação do músculo e braço de momento (braço de alavanca). Fonte: [Rocha 2000]

momento de um músculo afeta a capacidade do músculo de gerar força, produzir torque na articulação e realizar movimentos [Delp 2005]. No corpo humano, o osso atua como uma haste rígida, a articulação constitui o eixo, e os músculos aplicam força. A eficiência mecânica (ou vantagem mecânica) de uma alavanca em movimentar uma resistência é medida através da relação entre o braço de momento da força e o braço de momento da resistência. Essa vantagem mecânica é, junto ao tamanho do músculo e do nível de ativação, um fator importante na ação dos músculos em uma articulação.

[Fagg 2001] descreve um modelo de geometria muscular para vários músculos envolvidos na atuação do cotovelo e ombro. Esse modelo captura as alterações no braço de momento. Os músculos estudados nesse modelo são: Peitoral (flexor mono-articular do ombro), Deltóide (extensor do ombro), Bíceps cabeça longa (flexor mono-articular do cotovelo), Tríceps cabeça curta (flexor bi-articular) e Tríceps cabeça longa (extensor bi-articular). Todos os três extensores contornam a superfície da articulação todo o tempo. Os flexores mono-articulares do ombro e cotovelo seguem o caminho apresentado na Fi-

gura 3.12.

As duas condições de cálculos do braço de momento, importantes neste projeto, estão ilustradas na Figura 3.12. Primeiramente, é determinado o limite entre os dois casos que ocorrem quando o músculo toca a superfície da articulação num só ponto. A seguir, são mostradas as duas condições de tangência. Nas três condições é efetuado o cálculo do braço de momento e tamanho dos músculos a serem tratados nesse trabalho.

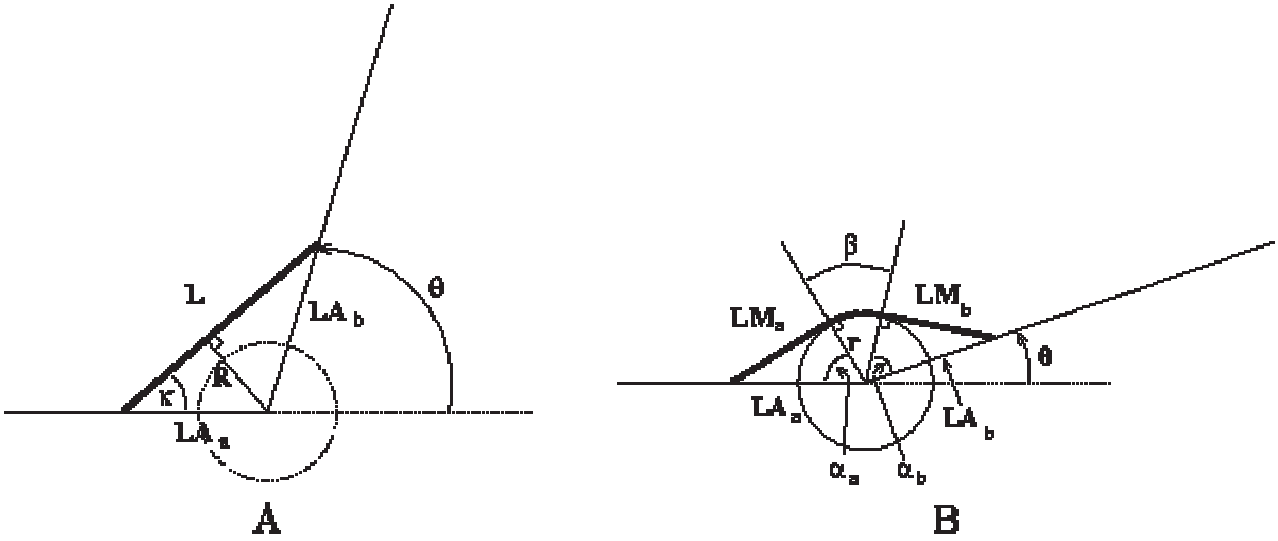

Figura 3.12: Modelo de caminho do músculo monoarticular : Caminho reto (A) e caminho de contorno na articulação (B). Fonte: [Fagg 2001].

Os parâmetros da Figura 3.12 são: θ (orientação da articulação), $LA_a e LA_b$ (distância do centro de rotação da articulação aos pontos de origem e inserção do músculo, respecivamente), r (o raio da cápsula da articulação), R (o braço de momento do músculo, L (comprimento do músculo), LM_a (distância do ponto de origem do músculo ao ponto de tangência na capsula da articulação) e LM_b (distância do ponto de inserção do músculo ao ponto de tangência na capsula da articulação).

Primeira condição: Ocorre quando o músculo tangencia a superfície da articulação num único ponto, como ilustrado na Figura 3.13. Os valores são calculados usando as seguintes equações:

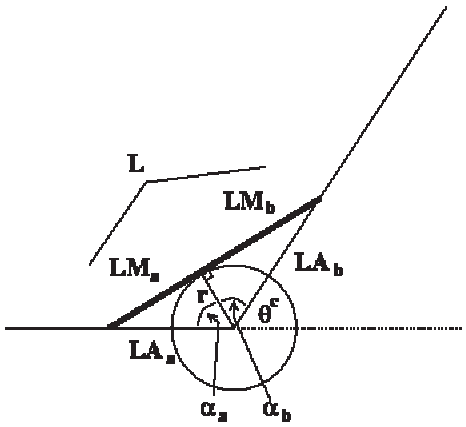

Figura 3.13: O músculo no limite entre as duas condições. Fonte: [Fagg 2001].

$$\alpha_a = \arccos(\frac{r}{LA_a}), \tag{3.1}$$

$$\alpha_b = \arccos(\frac{r}{LA_b}), \tag{3.2}$$

$$\theta^c = \Pi - \alpha_a - \alpha_b, \tag{3.3}$$

$$LM_a = LA_a * \sin(\alpha_a), \tag{3.4}$$

$$LM_b = LA_b * \sin(\alpha_b), \tag{3.5}$$

$$L = LM_a + LM_b, \tag{3.6}$$

onde r é o raio da superfície da articulação e LAa e LAb são as distâncias do ponto de origem e inserção do músculo ao centro de rotação da articulação. θ^c é o ângulo da articulação onde a transição entre as duas condições ocorrem. α_a, α_b , Lma e LMb são propriedades constantes do triângulo formado pelo centro de rotação, o ponto de origem e inserção e, o primeiro ponto onde o músculo tangencia na superfície da articulação,serão usados para a condição em que o músculo tangencia a superfície da articulação.

Segunda Condição: Músculo não tangencia a superfície da articulação, aplica-se se $\theta \geq \theta^c$ (Figura 3.12 A) . Os valores são calculados da seguinte forma:

$$L = \sqrt{LA_a^2 + LA_b^2 + 2LA_aLA_b\cos(\theta)} \tag{3.7}$$

$$K = \arccos(\frac{LA_a + LA_b\cos(\theta)}{L}) \tag{3.8}$$

$$R = LA_a\sin(K) \tag{3.9}$$

Terceira Condição Quando o músculo tangencia a superfície da articulação (Figura 3.12 B). Nesse caso o cálculo é feito com as seguintes equações:

$$\beta = \pi - \theta - \alpha_a - \alpha_b, \tag{3.10}$$

$$L = LM_a + r\beta + LM_b, \tag{3.11}$$

$$R = r. \tag{3.12}$$

onde α_a, α_b, Lma e LMb são calculados nas equações 3.1 a 3.6.

3.3.2 Cálculo do braço de momento da força Peso dos segmentos

Em adição ao cálculo dos torques produzidos pelos músculos na articulação, para uma completa análise do equilíbrio do membro superior humano, faz-se necessário o cálculo do braço de momento da força peso dos segmentos envolvidos para cada tipo de movimento. Devido a dificuldade de ilustrar as posições dos segmentos em três dimensões, é mostrado o membro superior nos planos sagital e coronal separadamente (Figura 3.14).

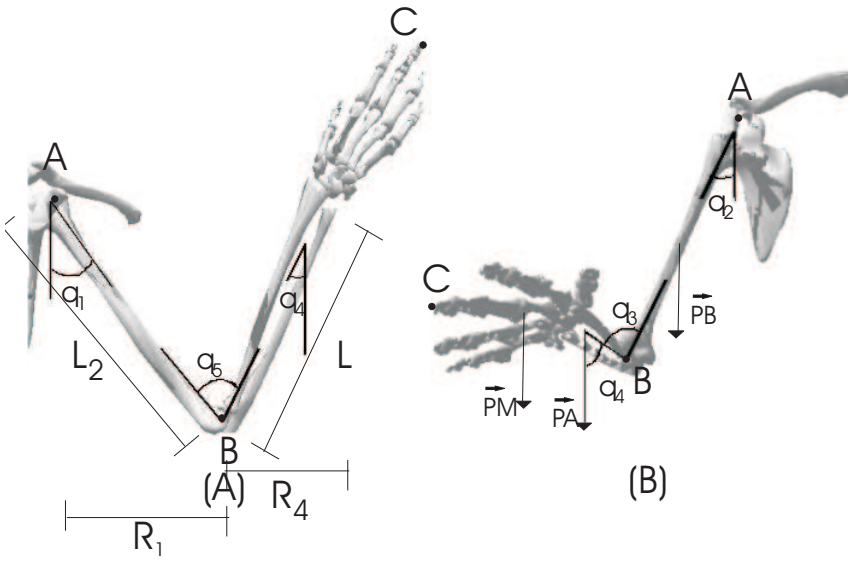

Figura 3.14: Posição do membro superior nos planos (A) Sagital e (B) Coronal

O mecanismo de movimento será descrito em termos do conjunto de parâmetros:

A: ponto ao redor do qual o braço rotaciona,

B: ponto ao redor do qual o antebraço rotaciona,

C: Extremidade distal do membro superior,

$\theta 1$: Ângulo entre o braço e corpo no plano sagital,

$\theta 2$: Ângulo entre o braço e corpo no plano coronal,

$\theta 4$: Ângulo entre a direção da força peso e o antebraço,

$\theta 5$: Ângulo entre o braço e antebraço,

$\theta 3$: 180^o - $\theta 5$;

α: Ângulo de movimento a cada iteração do processo,

L1: Comprimento do antebraço,

L2: Comprimento do braço,

R1: Projeção horizontal da distância entre o ombro e cotovelo,

R4: Projeção horizontal do braço de momento da força peso do antebraço,

PB: Peso do braço,

PA: Peso do antebraço,

PM: Peso da mão.

Para calcular o braço de momento da força peso do antebraço, para o movimento de flexão/extensão do antebraço, primeiro é necessário determinar a componente da força peso no eixo x. Para isso inicialmente é calculado $\theta 4$, (ângulo que a força peso do antebraço faz em relação ao próprio antebraço). Os valores de $\theta 1$, $\theta 2$ e $\theta 3$, previamente definidos, serão usados para calcular $\theta 4$, como mostra a Tabela 3.4. Para movimentos realizados no planos sagital ou no plano coronal, não simultaneamente, os valores foram obtidos baseando-se no modelo mecânico e deduções matemáticas. Um protótipo (Figura 3.15) foi criado para facilitar a visualização de algumas posições do membro superior humano. Usou-se uma armação de arame simples mas que permite as simulações dos movimentos desejados em três dimensões para estudo da direção e sentido das forças peso dos segmentos. Este protótipo foi desenvolvido pela autora do projeto.

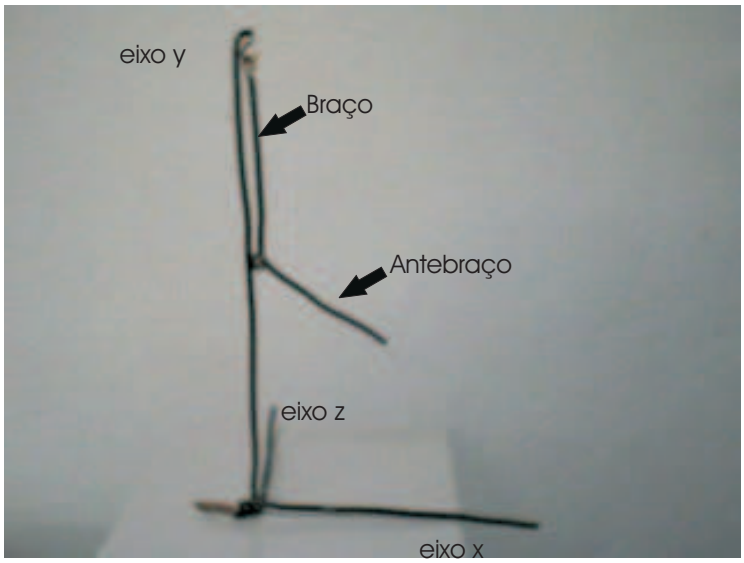

Figura 3.15: Protótipo do modelo mecânico do braço

Tabela 3.4: Valores dos ângulos e braço de momento da força peso do antebraço para cada posição das articulações do ombro e cotovelo.

θ**1**	θ**2**	θ**3**	θ**4**	**R4**
0^o	0^o	se $\theta 3 \leq 90°$	$\theta 3$	$(\frac{L1}{2})\sin(\theta 4)$
0^o	0^o	se $\theta 3 > 90°$	$180° - \theta 3$	$(\frac{L1}{2})\sin(\theta 4)$
$0^o < \theta 1 < 90^o$	0^o	se $\theta 3 \leq 90°$	$180° - \theta 1 - \theta 3$	$(\frac{L1}{2})\sin(180° - \theta 4)$
$0^o < \theta 1 < 90^o$	0^o	se $\theta 3 > 90°$	$\theta 1 + \theta 3 - 180°$	$(\frac{L1}{2})\sin(\theta 4)$
$\theta 1 = 90^o$	0^o	se $\theta 3 \leq 90°$	$90° - \theta 3$	$(\frac{L1}{2})\sin(\theta 4)$
$\theta 1 = 90^o$	0^o	se $\theta 3 > 90°$	$\theta 3 - 90°$	$(\frac{L1}{2})\sin(\theta 4)$
$\theta 1 > 90^o$	0^o	se $\theta 3 \leq 90°$	$180° - \theta 1 - \theta 3$	$(\frac{L1}{2})\sin(\theta 4)$
$\theta 1 > 90^o$	0^o	se $\theta 3 > 90°$	$\theta 1 - 90°$	$(\frac{L1}{2})\sin(\theta 4)$
0^o	$0^o < \theta 1 \leq 180^o$		$\theta 2$	$(\frac{L1}{2})\cos(\theta 4)$

Quando o movimento ocorre nos planos sagital e coronal simultaneamente, $\theta 4$ e $theta5$ (ângulo do braço nessa posição com o eixo z) são calculados da seguinte forma (ver Figura 3.16):

1. Define-se a equação da reta que passa pelo ponto C e é paralela a v, vetor diretor do antebraço:

$$R = C + \lambda\overrightarrow{v}, \tag{3.13}$$

$$(x, y, z) = (x2, y2, z2) + \lambda(x1 - x2, y1 - y2, z1 - z2), \tag{3.14}$$

onde $\overrightarrow{v} = (x1 - x2, y1 - y2, z1 - z2)$.

2. Define-se a equação da reta que passa pelo ponto C e é paralela ao vetor u (eixo y):

$$(x, y, z) = (x2, y2, z2) + \lambda(0, 0, -1), \tag{3.15}$$

u = (0,0,-1).

3. Fazendo-se uso da equação de produto escalar, calcula-se o ângulo entre os vetores $\overrightarrow{u}$ e $\overrightarrow{v}$:

$$\cos\alpha = \frac{(\overrightarrow{u} \bullet \overrightarrow{v})}{(|\overrightarrow{u}| * |\overrightarrow{v}|)}, \tag{3.16}$$

$$\cos\alpha = \frac{(z2 - z1)}{(1 * L1)}, \tag{3.17}$$

$$\alpha = \arccos(\frac{(Z2 - Z1)}{d}), \tag{3.18}$$

$$\theta 4 = \pi - \alpha. \tag{3.19}$$

Após ter sido definido $\theta 4$, o próximo passo é determinar o braço de momento da força Peso em relação à articulação envolvida com o movimento.

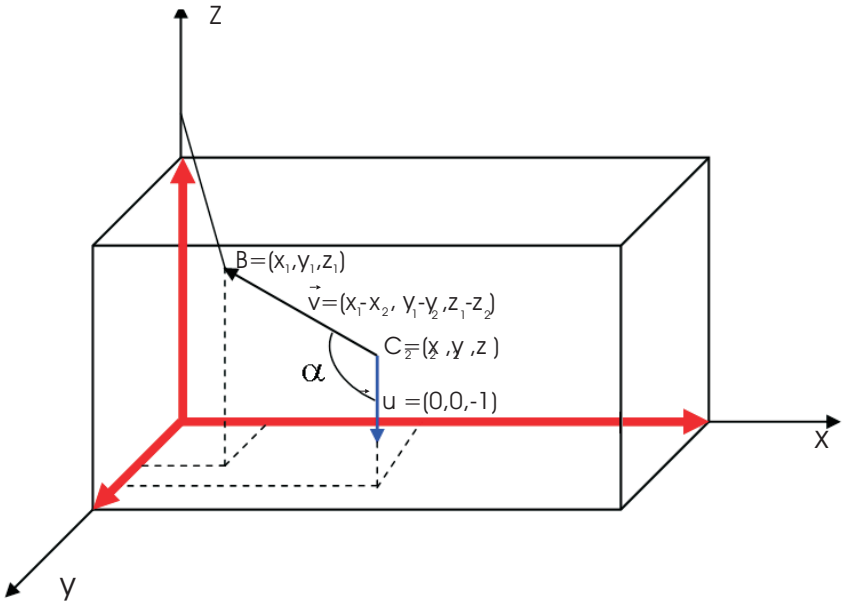

Figura 3.16: Ângulo entre a força Peso e o antebraço

- $Se(\theta 2 > 0^o)$

  $R4 = \frac{L1}{2} * \cos(\theta 4)$

- Se $(\theta 2 = 0^o)$ e $(\theta 1 > 0^o)$ e $(\theta 3 \leq 90^o)$

  $R4 = \frac{L1}{2} * \sin(180^o - \theta 4)$

- Se $(\theta 2 = 0^o)$ e $(\theta 1 > 0^o)$ e $(\theta 3 > 90^o)$

  $R4 = \frac{L1}{2} * \sin(\theta 4)$

Para calcular braço de momento da força peso do membro superior, se o antebraço estiver flexionado, ou seja $\theta 3 < 180^o$, é necessário analisar os efeitos dos segmentos correspondentes ao braço e antebraço separadamente (Figura 3.17).

Sendo $\theta 5$ o ângulo entre o braço e o corpo, ou seja, o eixo y, o cálculo do braço de momento da força Peso é determinado da seguinte forma:

- Se $\theta 3 = 180°$

  $R2 = \frac{L1+L2}{2} * \cos(\theta 5)$

- Se $\theta 3 < 90°$

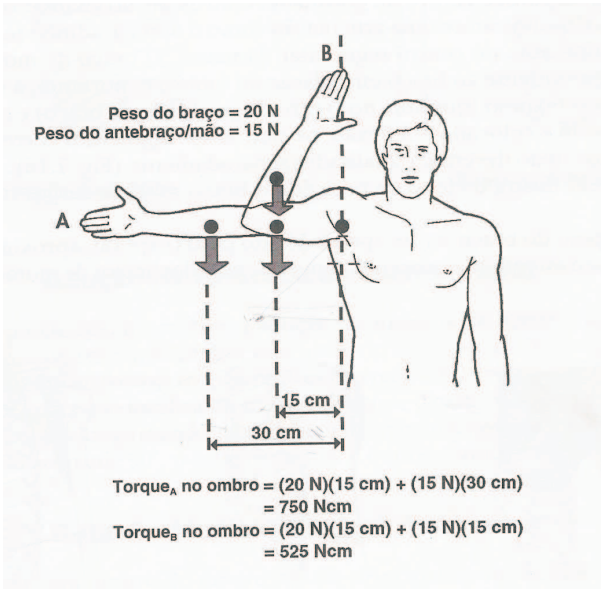

Figura 3.17: Torque criado no ombro por cada segmento do braço. Fonte: [Hall 2000]

$$R2 = \frac{L2}{2} * \cos(\theta 5) - R4$$

- Se $\theta 3 \geq 90°$ e $\theta 4 < 180°$

$$R2 = \frac{L2}{2} * \cos(\theta 5) + R4$$

O cálculo de $\theta 5$ é feito de forma análoga a $\theta 4$.

Após calcular o braço de momento, é calculado o torque resultante na articulação envolvida para determinar o sentido do movimento [Lemay e Crago 1996].

3.4 Conclusão

Este capítulo apresentou um estudo sobre os princípios biomecânicos utilizados na análise dos movimentos do membro superior humano. No próximo capítulo, baseando-se nos princípios aqui discutidos, é apresentada uma proposta de arquitetura de um sistema e as tecnologias de apoio, que permitem a simulação dos movimentos realizados pelo membro superior humano, a serem utilizados no desenvolvimento de próteses de membros superiores.

Capítulo 4

Tecnologias de Apoio e Arquitetura do Sistema

4.1 Introdução

A partir do estudo realizado no Capítulo 2, é proposta uma arquitetura de um sistema que permita a simulação do movimento do membro superior, com animação em tempo real, respondendo às variações dos valores de forças(N) aplicadas pelos músculos.

Para implementação do projeto foi escolhida a linguagem Visual C++ versão 7.0 por permitir a interação com bibliotecas gráficas, viabilizando a modelagem e manipulação do modelo virtual do membro superior humano. Como tecnologias de apoio foram selecionadas as bibliotecas OpenGL, SmallVR e FLTK, descritas a seguir.

4.2 Tecnologias de Apoio

Nessa seção, são apresentadas as tecnologias de apoio selecionadas para a implementação do sistema proposto. Posteriormente, será mostrado como essas tecnologias se integram para suportar a arquitetura proposta.

4.2.1 OpenGL e GLUT

OpenGL é uma biblioteca de rotinas gráficas e de modelagem bidimensional (2D) e tridimensional (3D). A maior vantagem em sua utilização é a portabilidade. Essa biblioteca usa algoritmos cuidadosamente desenvolvidos e otimizados pela Silicon Graphics, Inc., líder mundial em computação gráfica e animação [Manssour 2004].

Atualmente, OpenGL é reconhecida e aceita como um padrão API (Application Programming Interface) para desenvolvimento de aplicações gráficas 2D e 3D.

OpenGL é designada como uma interface independente não só de hardware , mas também de sistema operacional. Devido a essas características, nenhuma função de gerenciamento de janelas, interação com usuário ou arquivos de entrada e saída são inclusas no OpenGL. Cada ambiente, como exemplo o Microsoft Windows, possui suas próprias funções para esse propósito. Similarmente, OpenGL não fornece funções de alto nível para construção de modelos tridimensionais. Com OpenGL é possível a construção de modelos usando um pequeno conjunto de primitivas geométricas - pontos, linhas e polígonos. A biblioteca GLUT (OpenGL Utility Library) é que fornece várias funções de modelagem, tais como primitivas geométricas - esfera, cone e o famoso teapot (pote de chá)(ilustrado na Figura 4.1 com o exemplo do código fonte em c++), usado na maioria dos exemplos de programas OpenGL [Davis 1999].

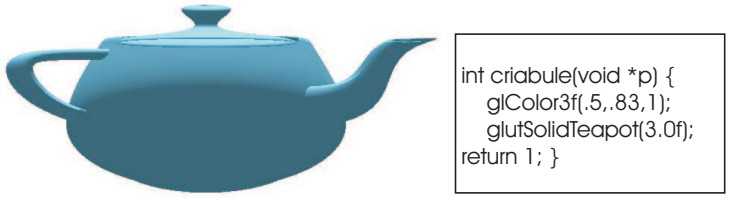

Figura 4.1: Ilustração e exemplo de código em c++ da primitiva *teapot*

4.2.2 SmallVR

SmallVR é uma ferramenta orientada a objetos para o desenvolvimento de aplicações de Realidade Virtual, desenvolvida pelo grupo de Realidade Virtual da PUCRS

(Pontifícia Univesidade Católica do Río Grande do Sul) [Santos, Kopper e Pinho 2003].

A ferramenta SmallVR é baseada nas bibliotecas OpenGL e GLUT e permite que o usuário use a estrutura básica de um programa com a GLUT e apenas adicione os objetos e comandos necessários à aplicação sem perder o controle da execução da aplicação.

Para a inicialização da biblioteca, utiliza-se a função *SmVR_Init*. Esta função devolve um apontador para um objeto que representa toda a cena tridimensional (chamado aqui de *RootObject)*. Para garantir a exibição da cena, o programador deve colocar dentro da função informada na *glutDisplayFunc* uma chamada do método Render do objeto referido *RootObject* (ver Figura 4.2).

```
SmVR_GeometricObject *RootObject;
void display( void )
{
    glClear(  GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT
);
RootObject->Render()
    glutSwapBuffers();
}
void InitGLUTCallbacks(void)
{
    // definição das funções de callback da GLUT
    glutDisplayFunc ( display );
    glutReshapeFunc ( reshape );
    glutKeyboardFunc ( keyboard );
    glutSpecialFunc ( arrow_keys );
}
void main()
{
    // inicialização da GLUT
    glutInit( &argc, argv );
    glutInitDisplayMode     (GLUT_DOUBLE     |     GLUT_RGB     |
GLUT_DEPTH );
    glutInitWindowPosition (0,0);
    glutInitWindowSize( 700, 500 );
    glutCreateWindow( "SmallVR Test" );

    RootObject = SmVR_Init();
```

Figura 4.2: Rotina de Inicialização da Biblioteca. Fonte:[Santos, Kopper e Pinho 2003]

Grafo de Cena

O grafo de cena (GS) é uma estrutura de dados que permite montar hierarquias entre os objetos da cena e aplicar transformações geométricas a estas hierarquias. As relações de hierarquia em um GS definem, resumidamente, quais as partes formadoras de um objeto tridimensional composto de outros objetos. Na modelagem do braço virtual, por exemplo, pode-se especificar o braço como sendo uma estrutura em que cada segmento tem um nível de hierarquia (são filhos) em relação ao outro. Com isto, as transformações geométricas aplicadas a um nível mais alto de uma hierarquia afetam a todos os objetos existentes em níveis inferiores da hierarquia. No braço virtual, o objeto que representa o antebraço é filho do objeto que representa o braço, logo, se houver uma transformação geométrica como uma rotação no objeto braço, essa transformação será automaticamente aplicada ao objeto antebraço.

Para a montagem de um grafo de cenas são necessários basicamente dois métodos. Um que insira um objeto como filho de outro e um que o retire desta condição. A Figura 4.3 mostra um exemplo de código utilizando os métodos *AddChild* e *RemoveChild*. Esse programa cria um triângulo e insere ele na cena como filho da raiz, fazendo-o parte do grafo de cena (Figura 4.4).

Objetos Gráficos

SmallVR é capaz de suportar dois tipos de objetos. Um deles é representado pelos objetos lidos de arquivos. O outro é definido por uma rotina do próprio programa. Essa rotina deve conter somente funções de desenho de primitivas OpenGL, ficando vetado o uso de funções de transformações geométricas. Estas transformações geométricas devem ser definidas fora desta rotina através de funções específicas da biblioteca.

Transformações Geométricas

- *SetTranslation*(tx, ty, tz) - Define um deslocamento para o objeto a partir de sua posição atual,

```
#include <GL\gl.h>
#include "SmVR.h"
// Root Object to represent the entire scene
SmVR_CGeometricObject *RootObject;


int DrawTest(void *p)
{
// Draw a triangle
        glBegin (GL_TRIANGLES);
                glColor3f(1,0,0);
                glVertex3f(0,0,0);
                glColor3f(0,0,1);
                glVertex3f(10,0,0);
                glColor3f(0,1,0);
                glVertex3f(5,10,0);
        glEnd();
return 1;
}


void main ( int argc, char** argv )
{
        SmVR_CGeometricObject *TestObject;
// GLUT initialization
        glutInit ( &argc, argv );
        glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowPosition (0,0);
        glutInitWindowSize ( 400, 400);
        glutCreateWindow ( "SmallVR Test" );
// initialize OpenGL parameters
        init ();
// initialize the library
        RootObject = SmVR_Init(NULL); // no scene to load
// Create a new object
        TestObject = new SmVR_CGeometricObject("Test Object", DrawTest);
// Add the new object to the scene
        RootObject->AddChild(TestObject);
.....
// start rendering loop
glutMainLoop ( );
}
```

Figura 4.3: Parte do código do programa FirstSmallVR.cpp. Fonte:[Santos, Kopper e Pinho 2003]

- *SetRotation*(ang, x,y,z) - Define uma rotação sobre o objeto a partir de um ângulo e do eixo ao redor de rotação,

- *SetScale*(ex, ey, ez) - Define um fator de escala a ser aplicado ao objeto.

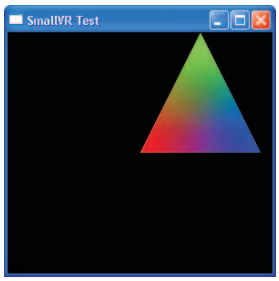

Figura 4.4: Tela do programa FirstSmallVR.cpp. Fonte:[Santos, Kopper e Pinho 2003]

4.2.3 FLTK

O FLTK (Fast Light Tool Kit), pronunciado como "fulltick", é um kit de desenvolvimento de interface gráfica para a linguagem C++. FLTK foi originalmente desenvolvido por Bill Spitzak e é mantido por um pequeno grupo de desenvolvedores ao redor do mundo com um repositório central nos Estados Unidos [Spitzak 1991].

FLTK pode ser usado em diversas plataformas (GNU Linux através do sistema gráfico X11, Windows, Mac OS X, OS 2 e Solaris) e, também, contém o FLUID(*Fast Light User Interface Designer*), que permite ao usuário desenvolver aplicações gráficas, widget (item de uma interface gráfica, ex.: exemplo: janelas, botões, menus) e classes para a GUI (interface gráfica, do inglês *Graphical User Interface*) [Spitzak 1991].

Para criar programas usando o FLTK é necessário que seja incluído o arquivo <FL/Fl.H>. Adicionalmente, os programas devem conter um arquivo de cabeçalho para cada classe FLTK em uso. Na Figura 4.5 é mostrado um exemplo de código de um simples programa que usa o FLTK para mostrar uma janela com uma caixa de texto escrito "Hello, World". Para a criação da janela, utiliza-se a função *Fl_Window*. A função *Fl_Box* mostra uma caixa de texto (Figura 4.6).

Usando OpenGL em programas FLTK

O modo mais fácil de criar uma janela OpenGL é através da subclasse *Fl_Gl_Window*. Essa subclasse deve implementar o método *draw()* que usa as chamadas às funções OpenGL para desenhar a tela. No programa principal deve conter uma chamada à função *redraw()* para as atualizações da cena, e o FLTK chamará a função *draw()* posteriormente.

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Box.H>

int main(int argc, char **argv) {
  Fl_Window *window = new Fl_Window(300,180);
  Fl_Box *box = new Fl_Box(20,40,260,100,"Hello, World!");
  box->box(FL_UP_BOX);
  box->labelsize(36);
  box->labelfont(FL_BOLD+FL_ITALIC);
  Box->labeltype(FL_SHADOW_LABEL);
  window->end();
  window->show(argc, argv);
  return Fl::run();
}
```

Figura 4.5: Exemplo de código de programa FLTK. Fonte [Spitzak 1991]

Figura 4.6: Tela do programa Hello World. Fonte [Spitzak 1991]

A Subclasse FL_GL_Window

Para criar a subclasse Fl_Gl_Window, o código deve conter:

- A definição da classe,
- O método *draw()*, e
- O método *handle()*, caso seja necessário receber dados através da interação com o usuário.

O exemplo do código para criação dessa classe está mostrado na Figura 4.7. Os métodos *draw()* e *handle()* são detalhados no próximo capítulo onde são mostrados os detalhes de implementação do sistema.

```
class MyWindow : public Fl_Gl_Window {
  void draw();
  int handle(int);

public:
  MyWindow(int X, int Y, int W, int H, const char *L)
    : Fl_Gl_Window(X, Y, W, H, L) {}
};
```

Figura 4.7: Exemplo de código de programa FLTK. Fonte [Spitzak 1991]

4.3 Arquitetura do Sistema

A Figura 4.8 apresenta a visão geral da arquitetura do sistema, composta basicamente por quatro(4) módulos: a GUI - Interface Gráfica do Usuário, o Gerenciador de Leitura e Simulação dos Movimentos, Modelos Virtuais do Membro Superior Humano e a Biblioteca de Classes, descritos a seguir.

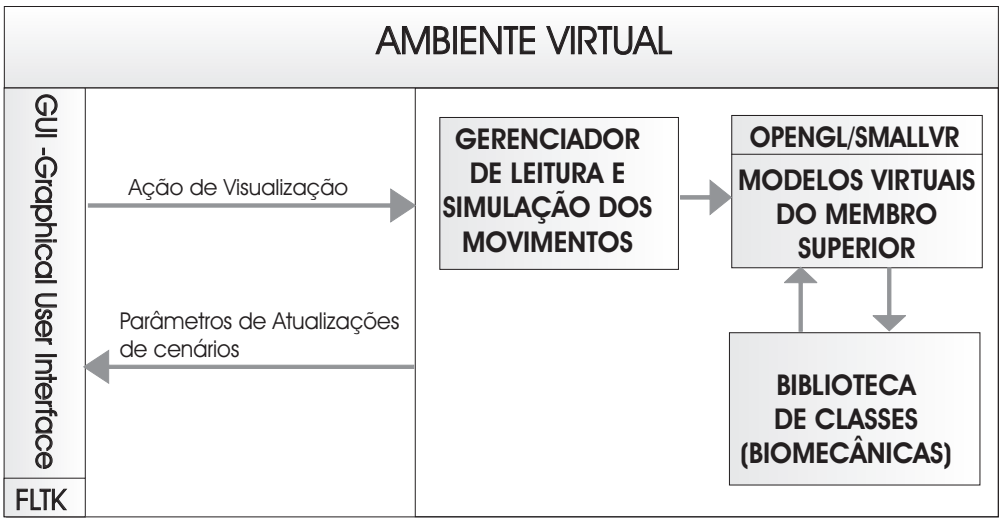

Figura 4.8: Arquitetura do Sistema

4.3.1 GUI - Interface Gráfica

A GUI - Interface Gráfica com usuário utiliza as classes da biblioteca **FLTK** para fazer a comunicação do usuário com o sistema (Figura 4.9). A utilização de objetos de interface (*widgets*) possibilita aos usuários ações para entrada de dados e manipulação dos objetos da cena como: navegação no cenário através das opções de zoom, rotação e

translação do objeto e também a alteração da posição da câmera no ambiente virtual, possibilitando a visualização da cena em três planos cardinais (Sagital, Coronal e Horizontal), que são planos perpendiculares imaginários de referência que dividem o corpo pela metade em termos de massa ([Hall 2000]).

No sistema proposto, o usuário informa os dados do paciente: idade, gênero e massa corporal. Usando esses dados, o sistema faz uma análise na tabela antropomórfica (Apêndice A) e gera os modelos virtuais do membro superior com os atributos necessário ao estudo dos movimentos do braço correspondentes aos movimentos de uma prótese.

O **FLTK** permite que a interface com o usuário e a janela contendo os modelos virtuais sejam mostradas na mesma janela, o que possibilita a visualização da cena durante a manipulação dos dados (Os limites de valores de forças presentes na interface formam definidos de forma aleatória, não seguindo um padrão pré-definido).

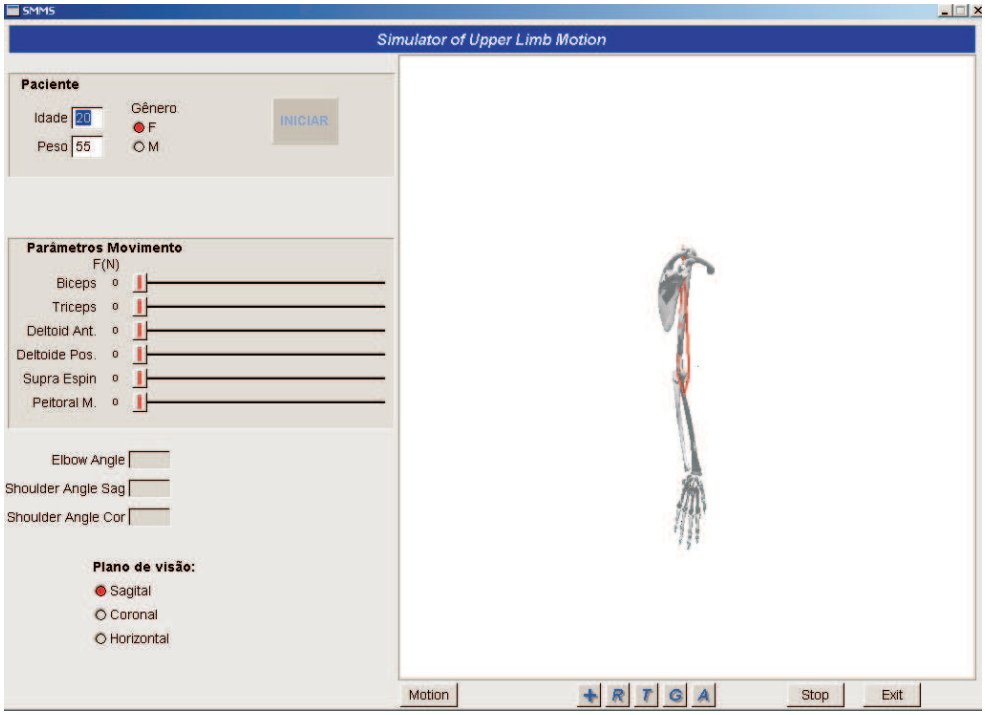

Figura 4.9: Tela do Sistema Simulador de movimentos do Membro Superior

4.3.2 Modelos Virtuais do Membro Superior Humano

O componente de Modelos Virtuais do Membro Superior Humano representa os objetos gráficos tridimensionais criados através do uso de funções de desenho de primitivas **OpenGL**. Para gerar cada objeto, o sistema fez uma leitura num arquivo contendo vértices e triângulos formados pelos vértices, gravados na ordem exata para que cada primitiva fosse criada seguindo a ordem necessária para modelagem do objeto. Esse arquivo de vértices foi obtido através da conversão de um objeto criado pela ferramenta 3D Visual Studio Max, feita através do programa 3DWin.

A biblioteca **SmallVR** permite a criação de um grafo de cena, descrito na seção anterior, onde é criada uma hierarquia entre os objetos da cena, ou seja, para o modelo virtual do membro superior tem-se o objeto Ombro pai do objeto Braço, que é pai do objeto Antebraço que tem a Mão como objeto filho.

4.3.3 Gerenciador de Leitura e Simulação dos Movimentos

O Gerenciador de Leitura e Simulação dos Movimentos faz a leitura dos parâmetros de realização do movimento (Força(N) aplicada por cada músculo e a força Peso dos segmentos), através de dados recebidos ou da manipulação dos objetos da interface. O usuário pode alterar o valor de uma força aplicada através da manipulação direta da barra de rolagem correspondente a cada força. Então o modelo virtual responde em tempo de execução a mudança das forças, olhando sua velocidade e posição em tempo real.

A cada leitura de dados feita, o sistema mostra o movimento gerado pelo membro superior virtual e a posição resultante do novo esquema de forças. Cada ângulo entre os segmentos Ombro e Braço (nos planos sagital e coronal) e entre o Braço e o Antebraço, são atualizados e mostrados ao usuário.

4.3.4 Biblioteca de Classes

Algumas características indispensáveis a um modelo músculo-esquelético são a facilidade de utilização e a especificação de seus movimentos. Por conseguinte, tendo noção

da complexidade em reproduzir um modelo muscúlo-esquelético humano, verificou-se a necessidade de desenvolvimento de uma biblioteca de classes para representação deste modelo.

A biblioteca de classes é composta pelas classes: Membro, Segmento, Articulação, Músculo e Movimento, mostradas na Figura 4.10. O membro, através da propriedade de agregação, é composto por segmentos, as articulações entre eles e músculos. Um segmento tem seu comprimento, largura, o seu peso e a ele associadas, tem-se as articulações e os músculos, que influenciam diretamente na realização dos movimentos. Os métodos *SetPosicaoDistal()* e *SetPosicaoProximal()*, ao sinal de cada movimento realizado pelo membro, atualizam os atributos: coordenadaProximal e coordenadaDistal, registrando a posição dos segmentos na cena durante toda a fase de simulação. A articulação tem os atributos: nome, tipo de articulação e os segmentos a ela associados. Por meio da associação com a classe movimento, tem a capacidade de restringir amplitude dos movimentos realizados pelo membro em cada plano cardinal, pelo método *ValidaAmplitude()*. O músculo tem seu nome, comprimentos nas condições : em repouso, extendido e contraído, peso, tipo de movimento, ponto de origem e inserção no segmento e, associado a classe segmento, também restringe os movimentos do membro por meio do método *ValidaTamanhoMusculo()*, obedecendo os limites expressos nos atributos: comprimento extendido e comprimento contraído.

Para elucidar as razões para criação destas classes e suas relações com este projeto, foi criado um exemplo de uma instância deste modelo, mostrado na Figura 4.11. Nesta figura, é mostrado o membro superior e um esquema da sua relação com o modelo de classes. O objeto *Bícepes* com seu ponto de origem a 2% do ponto proximal do segmento *Braço* e ponto de inserção a 10% do ponto proximal do segmento *Antebraço*. Entre estes segmentos, situa-se a articulação *Cotovelo* que, associada à classe Movimento, realiza o movimento *FlexaoCotovelo* seguindo as restrições determinadas.

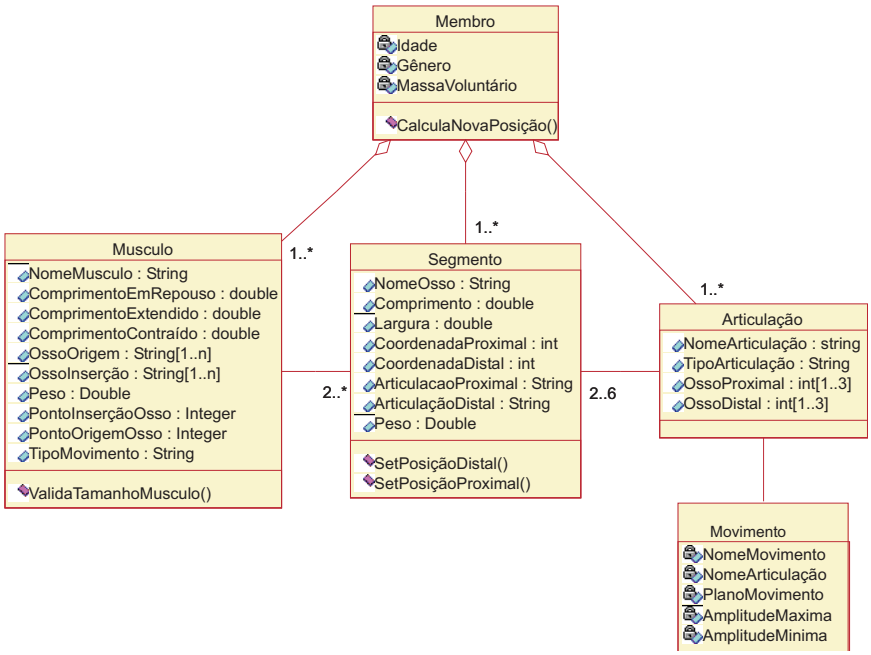

Figura 4.10: Classes genéricas de membros humanos

4.4 Sumário e Conclusões

A arquitetura aqui proposta foi construída com base nas análises dos sistemas apresentados no Capítulo 2 desse trabalho. A linguagem Visual C++ e as tecnologias de apoio apresentadas se adequaram perfeitamente para a realização desse projeto para produzir animação, interação e realização de cálculos de torques para análise dos movimentos. A biblioteca de classes apresenta as principais classes do sistema proposto e atua como referência para criação do modelo músculo-esquelético, assim como para a simulação dos movimentos de membros humanos.

No próximo capítulo, são apresentados os detalhes da implementação da arquitetura do Simulador de Movimentos do Membro Superior.

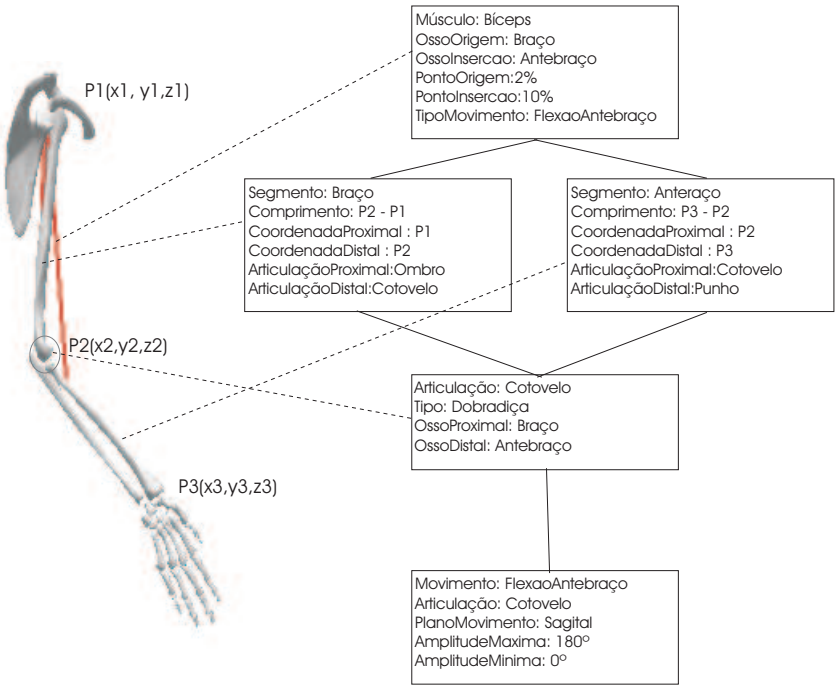

Figura 4.11: Diagrama de objetos do membro superior humano

Capítulo 5

Especificação e Implementação do Sistema Simulador de Movimentos do Membro Superior

5.1 Introdução

O software Simulador de Movimentos do Membro Superior - SMMS proposto no Capítulo 4, por meio da arquitetura do sistema, foi idealizado para simular, através da análise de equilíbrio estático, alguns movimentos correspondentes à uma prótese do membro superior. A partir dessa proposição, este capítulo apresenta os detalhes de especificação, análise e implementação do sistema.

5.2 Especificação do Sistema

A partir da análise comparativa de trabalhos relacionados apontada no final do Capítulo 2, esta seção apresenta a proposição de um sistema capaz de atender simultaneamente às seguintes limitações:

- Permissão para alterações nos parâmetros de movimento em tempo real;

- Simulação dos movimentos a partir do valor da força aplicada pelo músculo;
- Mostrar a animação em tempo real respondendo às alterações dos valores das forças aplicadas pelo músculo.

Para a fase de especificação do sistema foram utilizados modelos de caso de uso, diagramas de classes e de seqüência, ferramentas UML(*Unified Modelling Language*), uma linguagem gráfica para visualização, especificação e documentação dos artefatos de um sistema. O modelo de caso de uso descreve a funcionalidade do sistema e o diagrama de classes é uma padrão UML usado para detalhar o modo como os objetos serão produzidos. Uma classe é uma especificação/abstração - um objeto é uma instância de uma classe [Microsystems 2001].

5.2.1 Casos de Uso

O modelo de caso de uso descreve uma proposta de interação do usuário com o sistema. A Figura 5.1 apresenta os atores e os casos de uso a eles relacionados. Este projeto não implementa a participação dos atores: Sensor de sinal EMG, definido no diagrama, e PIPE, uma conexão entre a saída de um programa e a entrada de outro, bem como as funcionalidades por eles desempenhadas. Na seqüência serão descritos os casos de usos para os atores: Profissional de Biomecânica, Usuário de Prótese e PIPE.

Profissional de Biomecânica

As funcionalidades desempenhadas por esse ator são as seguintes:

Informa dados do usuário de prótese Para gerar o modelo virtual correspondente às medidas do usuário de prótese, primeiramente o profissional de biomecânica informa ao sistema os dados do usuário de prótese: idade, peso e gênero.

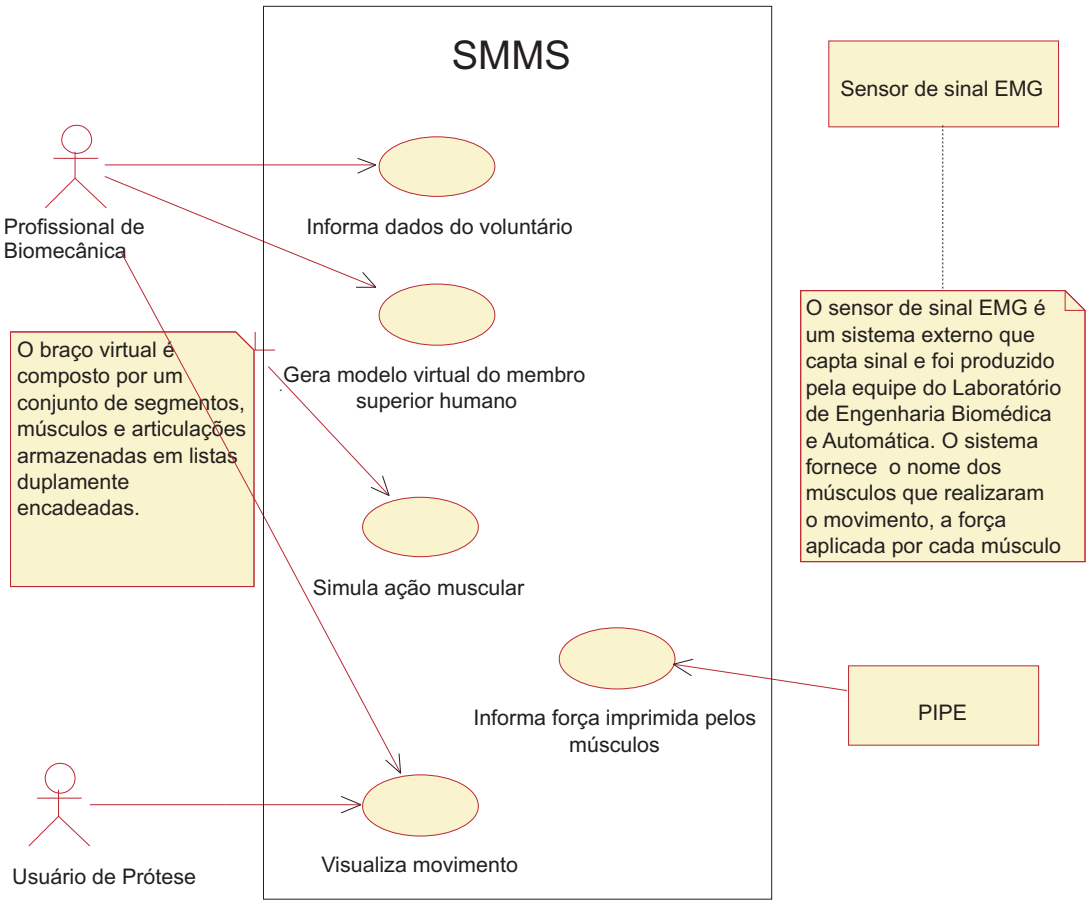

Figura 5.1: Modelo de casos de uso do sistema SMMS

Gera modelo virtual do membro superior humano O profissional de biomecânica solicita a criação do modelo virtual por meio de um click no botão *Iniciar*. O sistema faz uma consulta na tabela antropomórfica (Apêndice A), disponível no sistema em forma de matriz, e busca as medidas para cada segmento de acordo com a idade, o peso e o gênero do usuário de prótese. Tendo essas medidas como parâmetros, o sistema gera o modelo virtual do membro superior humano em três dimensões, incluindo o modelo do caminho percorrido pelos músculos bíceps e tríceps, conforme mostrado no Capítulo 3, representados por linhas vermelhas.

Simula forças musculares O profissional de biomecânica tem a opção de simular forças aplicadas pelos músculos por meio da alteração das forças correspondentes a cada músculo.

Visualiza movimento Esse caso de uso é detalhado no diagrama de estado mostrado na Figura 5.2. Esse diagrama mostra um processo de iteração. Após a ocorrência de um dos eventos mencionados no ítem anterior, o sistema faz várias análises e movimenta o braço virtual até que a condição de equilíbrio do membro seja alcançada ou, até que os limites de amplitude do movimento e de tamanho dos músculos sejam atingidos. Essas análises são detalhadas a seguir:

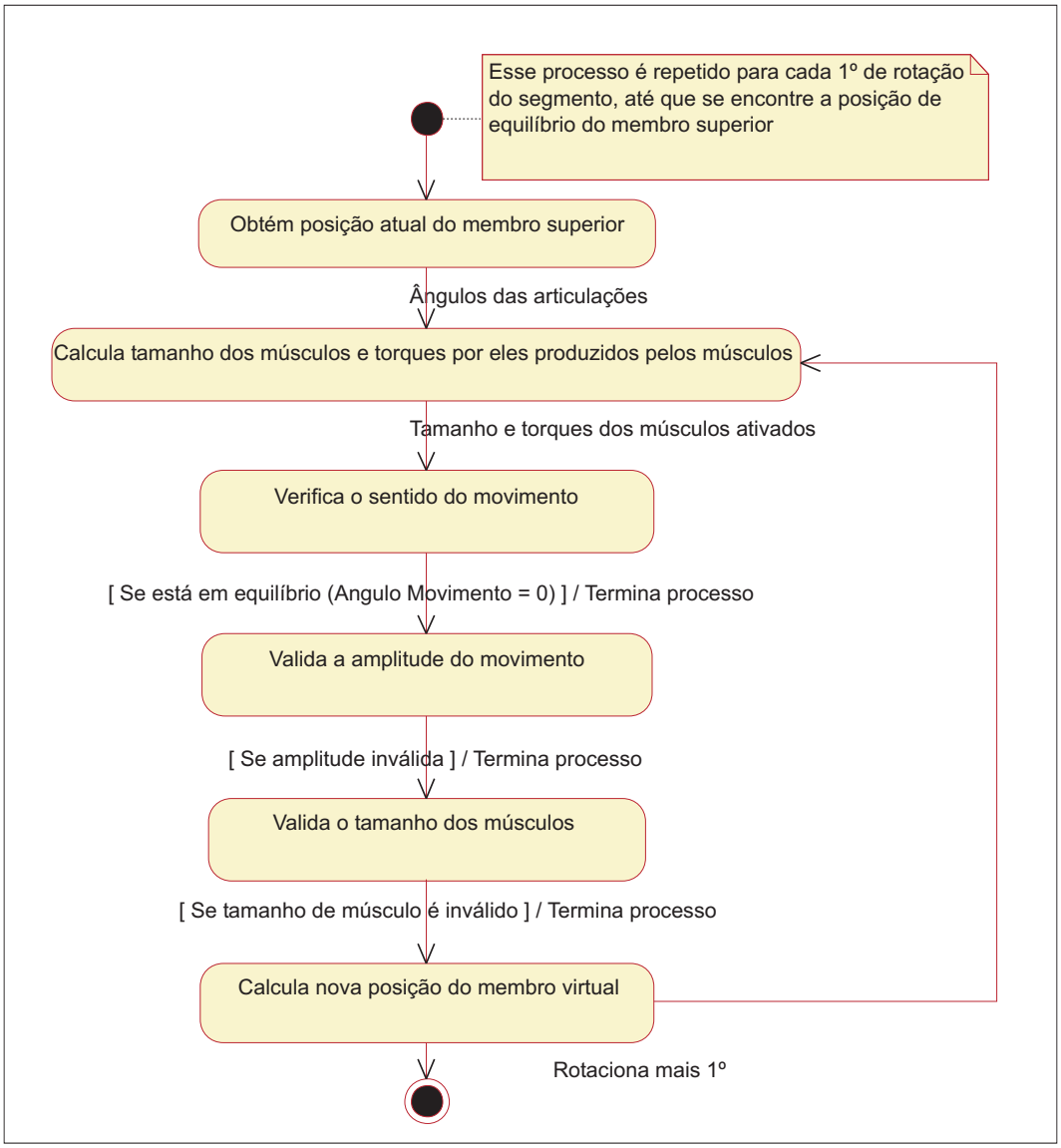

Figura 5.2: Diagrama de estado para o caso de uso: Valida nova posição do braço virtual

1. Obtém posição atual do membro superior

   O sistema busca nos campos existentes na tela os ângulos entre os segmentos

de cada articulação e no caso do ombro, os ângulos nos planos sagital e coronal.

2. Calcula tamanho dos músculos e torques por eles produzidos

   O Capítulo 3 mostra as equações para cálculo do braço de momento e tamanho dos músculos para cada uma das forças que estão sendo aplicadas no membro. O cálculo do torque (σ), para cada uma dessas forças, incluindo a força peso exercida pelas massas dos segmentos, é efetuado pela a equação 5.1.

$$\sigma = F * R, \tag{5.1}$$

   onde F é a força aplicada e R é o braço de momento dessa força em relação à articulação envolvida no movimento.

3. Verifica o sentido do movimento

   O sentido do movimento do segmento ou membro é definido pelo torque articular resultante.

   Após ter calculado os torques produzidos pelos músculos na articulação, é determinada a direção do torque produzido pela força peso dos segmentos em questão. Considera-se como positiva a direção dos torques produzidos pelos músculos agonistas (sentido horário) e negativa a direção do torques produzidos pelos músculos antagonistas (sentido anti-horário).

   O torque articular resultante é determinado pela soma de todos os torques efetivos na articulação. O valor deste torque determina a direção do movimento por meio da variação do ângulo de movimento: a) soma maior que zero(0), o ângulo de movimento assume o valor igual a 1º; b) soma menor que zero(0), o ângulo será igual a -1º; e c) soma igual a 0, o ângulo de movimento será igual a 0º.

4. Valida a amplitude do movimento

   Soma-se o ângulo de movimento ao ângulo anterior da articulação. Caso o ângulo resultante esteja fora dos limites de amplitude, o movimento é interrompido.

5. Valida o tamanho dos músculos

   Verifica se o tamanho do músculo está dentro dos limites de contração e extensão determinados para cada músculo. Caso o tamanho do músculo ultrapasse os limites, o movimento é interrompido

6. Calcula nova posição do membro virtual.

   Caso as duas condições anteriores tenham sido satisfeitas e o valor do ângulo de movimento seja igual a 0, rotaciona o segmento ou membro e atualiza as coordenadas dos pontos proximal e distal de cada segmento e reposiciona os músculos.

Usuário de Prótese

A funcionalidade desempenhada por esse ator é a seguinte:

Visualiza movimento O usuário de prótese, assim como o profissional de biomecânica, acompanha os movimentos gerados para o modelo virtual. Este caso de uso está detalhado nos casos de uso do Profissional de Biomecânica.

PIPE

Informa força aplicadas pelos músculos No projeto futuro, o usuário de prótese conectado aos sensores que captam os sinais EMG para informar as forças exercidas pelo membro superior humano, apenas realizaria movimentos, e esses sinais seriam

convertidos em valores de força(N) e velocidade realizados por cada músculo e informados ao sistema através do uso de um PIPE.

Como o projeto atual ainda não contempla a recepção dos valores de forças, esse processo será substituído por uma matriz contendo os valores das forças aplicadas por cada músculo para simular um movimento desejado. Nesse caso, o sistema faz a leitura desses dados e mostra a seqüência dos movimentos. Estes valores de forças foram gerados de forma aleatórias, não obedecendo a nenhum padrão, somente observando o resultado durante os testes para melhor conseguir mostrar a simulação.

5.3 Diagrama de Classes

O diagrama de classes é uma ferramenta utilizada para a fase de análise do sistema. Esse diagrama contém as principais classes do sistema proposto e mostra a interação entre os objetos. O diagrama de seqüência mostra a seqüência dessas interações. A Figura 5.3 mostra quais são os classes que fazem parte do sistema e a relação entre elas.

O *Membro* contém os atributos do usuário de prótese que são necessários à criação do modelo virtual do membro superior: idade, gênero e massa prótese, e as classes Segmento, Articulação e Músculo a ele agregados, compõem a estrutura do membro. Essas classes compõem a biblioteca de classes da arquitetura do sistema e foram detalhadas no Capítulo 4.

A classe *Objeto3D* é utilizada para modelar o membro superior através dos métodos: *InitObjetos(), DesenhaBraco(), DesenhaAntebraco(), DesenhaMao(), DesenhaMusculoBiceps(), DesenhaMusculoTriceps() e PosicionaPontosControle()*. O método *InitObjetos()* insere e posiciona o observador e o alvo na cena e também busca os arquivos que contém os vértices dos modelos virtuais. O métodos *DesenhaBraco(), DesenhaAntebraco() e DesenhaMao()* chamam funções que geram os modelos virtuais do braço, ombro, antebraço e mão, inserindo-os posicionando-os na cena. A inserção é feita de forma a montar a hierarquia entre eles. Os métodos *DesenhaMusculoBiceps() e DesenhaMusculoTriceps()*

chamam as funções que criam os músculos baseando-se nas equações utilizadas para cálculos do braço de momento na Seção 3.3. Finalizando, o método *PosicionaPontosControle()* insere pontos do tipo *SmVR_CPoint* entre os segmentos para facilitar as transformações geométricas utilizadas para simular os movimentos do membro.

O *EstruturaObj* tem os atributos: *Shape* e quantidade de shapes. Essa classe tem um método *loadFromMs3dAsciiFile* que faz a leitura do arquivo que contém os shapes do modelo, buscando os shapes e número de shapes. *EstruturaObj* é formado pelo *Shape* que tem os atributos: *NumeroTriangulos, NumeroVertices e NumeroNormals* e o método *loadFromMs3dAsciiSegment* que faz a leitura do arquivo, buscando os vértices, os triângulos e as normais para cada shape.

Figura 5.3: Diagrama de classes

5.4 Detalhes de Implementação

O SMMS foi implementado em Visual C++ 7.0 [Olsen e Templeman 2003]. Para criar a interface optou-se pelo uso do FLTK, por ele permitir o desenvolvimento de in-

terface gráfica e, principalmente, possibilitar a criação de uma janela OpenGL através da subclasse FL_GL_Window. Essa subclasse permite a interação da interface com o ambiente virtual. Para inserção, controle e transformações geométricas dos objetos na cena, optou-se pelo uso do SmallVR.

5.4.1 Criação do membro superior virtual

Quando o profissional de biomecânica informa os dados do usuário de prótese, os seguintes objetos: *Segmento, Articulação, Músculo e Movimento* são criados e armazenados respectivamente nas listas duplamente encadeadas: *listaSegmento, listaArticulacao, listaMusculo* e *listaMovimento*, implementadas nas classes *DbList* e *Lista*, através do método *Insert*. Esta dissertação inclui um cd(em sua contracapa) com os código do sistema desenvolvido.

Na fase de criação dos objetos do tipo *Segmento*, é feita uma consulta na Tabela Antropomórfica (Apêndice A) através da função *ObtemComprimento()*, que, tendo como parâmetro a idade, gênero e o nome do segmento, obtém-se o tamanho do segmento. Também é feita uma consulta na Tabela 5.1, registrada no sistema em uma matriz, para obter a massa de cada segmento relativo à massa do usuário de prótese.

Tabela 5.1: *Tabela de Medidas de Massa de Membro Superior em relação ao corpo. Fonte: [Zatsiorsky 1983]*

Corpo	**Braço**	**Antebraço**	**Mão**
100%	2.55%	1.38%	0.56%

Após o armazenamento desses objetos, os métodos *DesenhaBraco(), DesenhaAntebraco()* e *DesenhaMao()* da classe *Objeto3D* são chamados para criação dos modelos virtuais e inserção no grafo de cena.

O objeto3D é obtido pela instância da classe *SmVR_CGeometricObject* pertencente à biblioteca de classes *SmallVR*. Essa classe cria um objeto, para cada segmento, através da chamada de uma rotina do próprio programa que contém funções OpenGL. Os parâmetros para essa rotina são passados por meio do método *SetRenderFunctionData()*.

Para implementar as rotinas de criação dos objetos, foi necessário utilizar as classes: *Model3D, Shape, Triangulo* e *Vertice* para buscar os vértices do objeto. A classe *EstruturaObj* tem um método chamado *loadFromMs3dAsciiFile()* que faz a leitura de um arquivo de vértices, faz a instância da classe *Shape* e obtém as quantidades de triângulos por shape.

A classe *Shape*, através do método *loadFromMs3dAsciiSegment()*, faz a instância das classes *Vertices, Normal* e *Triangulo*, obtém os vértices, os conjuntos de vértices que formam os triângulos na ordem correta para gerar o objeto e os vetores normais, vetores perpendiculares aos planos definidos pelos triângulos utilizados tanto para e executar transformações geométricas bem como em modelos de iluminação.

Para cada segmento, braço, antebraço e mão, é criada uma instância da classe *EstruturaObj*. Os arquivos de vértices lidos pelos métodos *loadFromMs3dAsciiFile()* e *loadFromMs3dAsciiSegment()*, para gerar os modelos virtuais para estes segmentos são: ModeloBraco.txt, ModeloAntebraco.txt e ModeloMao.txt. O arquivo ModeloAntebraco.txt é apresentado no Apêndice F.

Após obter os vértices, são definidos os limites de referência da tela 3DS para o OpenGL, para que se possa converter o objeto para o tamanho desejado e, por último, é feita a leitura dos vértices e gerado o membro superior virtual.

Os códigos do método *DesenhaBraco()* e da rotina *DesenhaObjBraco()* chamada para desenhar o objeto Braco, são mostrados no Apêndice B.

Os objetos do tipo Articulação são criados através da função *InsereArticulacao()*. Essa rotina recebe os parâmetros: NomeArticulacao, TipoArticulaca, SegmentoProximal e SegmentoDistal e insere o objeto na *listaArticulacao*. O código dessa rotina é mostrado no fragmento de código 5.4.1.

Os objetos do tipo Musculo são criados através da função *InsereMusculo()*. Essa rotina chama a função *CalculaPosMusculo()* para calcular o comprimento do músculo de acordo com as equações 3.6, 3.7 ou 3.11 apresentadas na Seção 3.3.1.

No caso da inserção dos músculos bíceps e tríceps, chama-se os métodos *Dese-*

```
1  // Insere as articulações através da entrada dos dados da prótese e do
2  //paciente
3  void InsereArticulacao(char nomeArticulacao[30], char tipoArticulacao[30],
4                         char ossoProximal[30], char ossoDistal[30]){
5          sprintf(ObjA->NomeArticulacao, "%s", nomeArticulacao);
6          sprintf(ObjA->TipoArticulacao, "%s", tipoArticulacao);
7          sprintf (ObjA->OssoProximal, "%s", ossoProximal);
8          sprintf (ObjA->OssoDistal, "%s", ossoDistal);
9          listaArticulacao.Insert(*ObjA);
10         pA = listaArticulacao.end;
11 }
```

Código 5.4.1: *Função que cria o objeto tipo Articulacao e o insere na listaArticulacao.*

nhaMusculo() e *DesenhaMusculoT()*, respectivamente, da classe Objeto3D, para inserir os músculos no grafo de cena. Esses métodos desenham os músculos através das rotinas *DesenhaLinha()* e *DesenhaDiscoParcial()*.

A rotina *DesenhaLinha()* gera as partes do músculo que não tangenciam a superfície da articulação. Os parâmetros que indicam os pontos de origem e inserção do músculo são passados através do método *SetRenderFunctionData()*. A rotina *DesenhaDiscoParcial()* gera a parte do músculo que tangencia a articulação. Os parâmetros α_a e β, apresentados na Seção 3.3.1, são passados através do método *SetRenderFunctionData()*.

O método *PosicionaPontosControle()* da classe Objeto3D, é usado para inserir pontos entre os segmentos no grafo de cena, a fim de facilitar a aplicação das transformações geométricas que produzem os movimentos realizados pelo membro.

A função *InsereMovimento()* restringe o tipo de movimento da articulação, bem como as amplitudes máxima e mínima do movimento para cada plano cardinal.

5.4.2 Visualização dos movimentos do membro superior virtual

Foram criados dois modos de visualização dos movimentos realizados pelo membro superior virtual: a) Leitura de um arquivo que contém os valores das forças aplicadas por cada músculo para simulação de um movimento, e b) Alteração do valor da força aplicada pelos músculos por meio das barras de rolagem apresentadas na interface do SMMS.

Nestes dois modos, quando o sistema recebe o evento de alteração de valor de força de algum músculo, é feita uma análise do sistema de equilíbrio de forças, como mostrado no diagrama de estado (Figura 5.2, Seção 5.2.1). Esse diagrama mostra um processo de iteração que ocorre enquanto o valor de forças aplicadas por um ou mais músculos são alterados.

O cálculo dos torques das forças aplicadas pelos músculos agonistas e antagonistas que participam do movimento, dos torques da força peso e a determinação do ângulo de movimento para cada parte do processo de iteração são efetuados na função *VerificaSentidoMovimento()*.

Essa função faz a análise do equilíbrio de forças para o membro superior quando o movimento está sendo realizado a partir das seguintes considerações: a) os valores do braço de momento e do ângulo que a força peso do antebraço faz com o antebraço(mostrados na Seção 3.3.2), b) os valores das forças aplicadas pelos músculos, c) os valores dos braços de momentos calculados de acordo com as equações 3.9 ou 3.12, utilizadas conforme o ponto de tangência dos músculos. O fragmento de código referente à essa função é apresentado no Apêndice D.

Após a verificação do sentido do movimento, são feitas as validações de amplitude do movimento e tamanho do músculo através dos métodos *ValidaAmplitude()* da classe *Movimento* e *ValidaTamanhoMusculo()* da classe *Musculo* respectivamente.

Satisfeitas todas as condições para realização do movimento, sua visualização ocorre pela chamada do método *RotateBy()* da classe *SmVR_CGeometricObject*. Considerou-se como ponto de rotação do ombro, a coordenada proximal do braço e, como ponto de rotação do cotovelo, a coordenada proximal do antebraço. Os valores dos ângulos entre os segmentos e os pontos, proximais e distais, de cada segmento são atualizados. As novas posições dos músculos são determinadas e os músculos são redesenhados do mesmo modo descrito na criação dos objetos na Seção 5.4.1.

O processo de iteração é continuado enquanto as condições acima mencionadas sejam satisfeitas. Caso uma delas não seja satisfeita, o membro superior virtual permanece

em equilíbrio.

5.4.3 Navegação no ambiente virtual

A SmallVR possibilita a navegação no ambiente virtual deslocando-se o observador e o seu ponto de interesse, o alvo. O controle do observador é feito a partir da chamada da rotina *gluLookAt*, da biblioteca GLU. No SMMS, essa rotina é chamada no método *draw* da classe OpenGL, como mostrado no fragmento de código 5.4.2.

```
1  // Obtém a posição do usuário no sistema de coordenadas da raiz do
2  // grafo de cena
3  Obj->user->GetPointInOCS(Zero, &pt_user, Obj->Root);
4
5  // Obtém a posição do alvo no sistema de coordenadas da raiz do
6  // grafo de cena
7  Obj->alvo->GetPointInOCS(Zero, &pt_target, Obj->Root);
8
9
10 Obj->normal->GetPointInOCS(Zero, &pt_normal, Obj->Root);
11 gluLookAt(pt_user.X,pt_user.Y,pt_user.Z,  // usuário
12           pt_target.X,pt_target.Y,pt_target.Z,   // alvo
13             pt_normal.X, pt_normal.Y, pt_normal.Z);
```

Código 5.4.2: *Fragmento de código do método draw que contém a chamada da rotina gluLookAt.*

O método *draw*, além de posicionar o observador e o alvo na cena, é usado também para: a) determinar a cor do fundo da tela, b) criar e configurar as luzes para a cena, e c) exibir a cena.

Foram criados dois modos de navegação no ambiente virtual para o SMMS. Um deles é a seleção do plano de visão, permitindo ao usuário a escolha de uma das opções: Plano Sagital, Plano Coronal ou Plano Horizontal.

Para cada opção escolhida, as posições do observador, do alvo e do vetor normal são alteradas através dos métodos *SetLocalTransformationMatrixToIdentity()*, que volta o usuário para a posição original, *TranslateByOnOBJCS()*, que move o observador e alvo para a posição relativa ao plano de visão selecionado, e o método *TranslateBy()*, que

move o vetor normal. Como exemplo, o código 5.4.3 mostra como esses objetos são reposicionados na cena para que ela possa ser vista no plano coronal.

O outro modo de navegação implementado no sistema, foi a navegação no próprio ambiente, através da seleção das opções: zoom, rotacionar, transladar, girar e atualizar a cena, essa última, move o observador e o alvo para suas posições originais.

O FLTK capta os eventos do mouse na janela do ambiente virtual através das seguintes funções: a) FL_PUSH, que capta a posição exata onde o botão do mouse foi pressionado, b) FL_DRAG, que capta as posições onde o botão do mouse está sendo arrastado enquanto pressionado.

Após verificar a opção de navegação selecionada pelo usuário e captar os eventos do mouse na janela do ambiente virtual, o método *handle* move o observador no ambiente, em relação ao alvo. A movimentação do usuário na cena é feita pelo método *TranslateByOnOBJCS()* da classe it SmVR_CGeometricObject. Essa opção de navegação foi implementada no método *handle* da classe OpenGL e o seu código é mostrado no Apêndice E.

```
1  void cb_planoC(){
2          SmVR_CPoint pt_aux;
3          int xAux = 0;
4          int yAux = 0;
5          int zAux = 0;
6
7          if (B->anime) B->anime = false;
8          else B->anime = true;
9
10         // Translada o observador para a posição original e
11         // depois os reposiociona
12         B->Obj->user->SetLocalTransformationMatrixToIdentity();
13         B->Obj->user->TranslateByOnOBJCS(90, 0, 00, B->Obj->Root);
14         B->Obj->alvo->SetLocalTransformationMatrixToIdentity();
15         B->Obj->alvo->TranslateByOnOBJCS(-90, 0, 00, B->Obj->Root);
16
17
18         // Calcula nova posição do vetor normal
19         xAux = yAux = zAux = 0;
20         B->Obj->normal->GetPointInOCS(Zero, &pt_aux, B->Obj->Root);
21         pt_normal.X = 0;
22         pt_normal.Y = 1;
23         pt_normal.Z = 0;
24
25         if (pt_aux.X > pt_normal.X)
26                 xAux = -1;
27         else if (pt_aux.X < pt_normal.X)
28                         xAux = 1;
29
30         if (pt_aux.Y > pt_normal.Y)
31                 yAux = -1;
32         else if (pt_aux.Y < pt_normal.Y)
33                         yAux = 1;
34
35         if (pt_aux.Z > pt_normal.Z)
36                 zAux = -1;
37         else if (pt_aux.Z < pt_normal.Z)
38                         zAux = 1;
39
40         B->Obj->normal->TranslateBy(xAux, yAux, zAux);
41         B->Obj->LimpaBuffer();
42         B->invalidate();
43         B->redraw();
44         Fl::flush();
45 }
```

Código 5.4.3: *Função cb_planoC que altera a posição do observador na cena.*

5.4.4 Tela de interface com o usuário (GUI)

A Figura 5.4 apresenta a interface do sistema SMMS. Na seqüência, são descritos todos os ítens da interface.

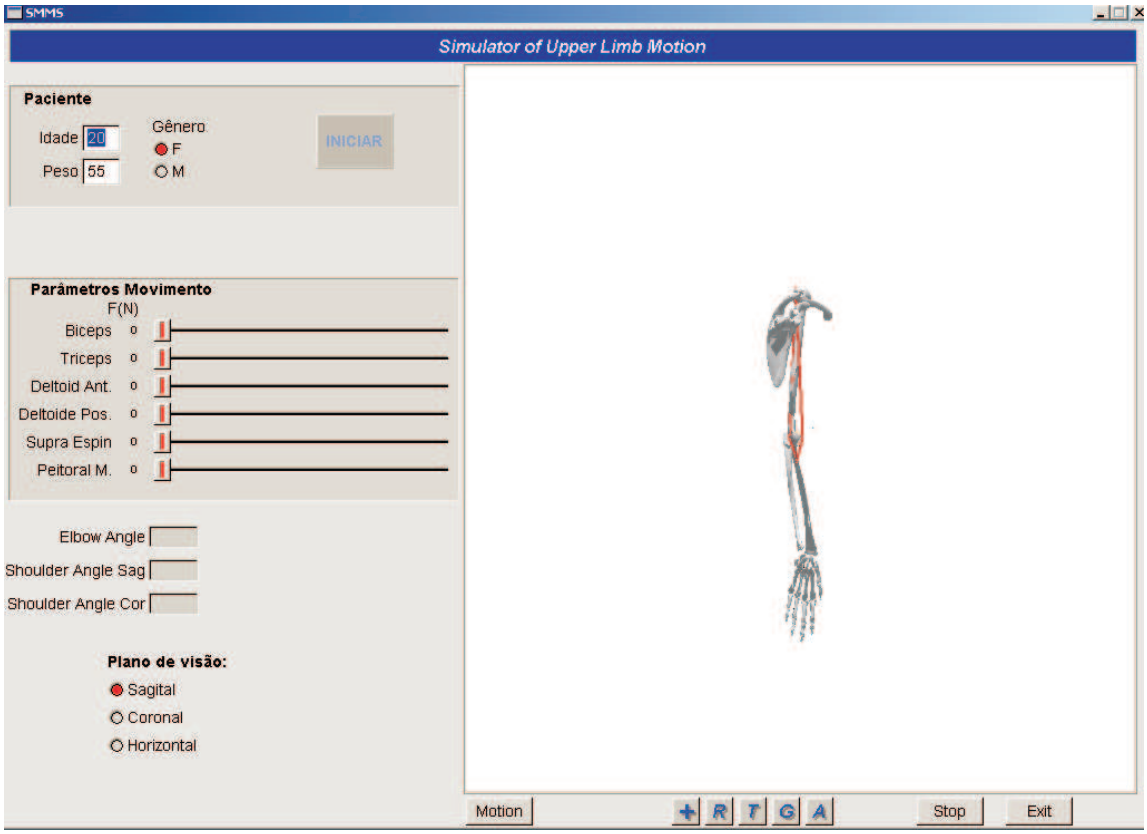

Figura 5.4: Tela do sistema SMMS

Dados do paciente: Informações pessoais do paciente

Idade: Idade do usuário de prótese em anos;

Peso: Massa do candidato em quilogramas;

Gênero: Gênero do usuário de prótese. F - Feminino e M - Masculino.

Botão Iniciar: Botão utilizado para gerar o modelo virtual do membro superior humano.

Parâmetros de Movimento: Esses objetos de interface são denominados barras de rolagem. O sistema contém uma barra de rolagem para cada músculo, para permitir que o usuário informe a força(N) aplicada por ele. As barras existentes são as seguinte:

Bíceps: informa a força aplicada pelo músculo bíceps, e produz o movimento de flexão do antebraço;

Tríceps: informa a força aplicada pelo músculo tríceps, e produz o movimento de extensão do antebraço;

Deltóide: informa a força aplicada pelo músculo deltóide, e produz o movimento de flexão do braço;

Peitoral: informa a força aplicada pelo músculo peitoral maior, e produz o movimento de extensão do braço;

Supra-espinhal: informa a força aplicada pelo músculo supra-espinhal, e produz o movimento de adução do braço;

Grande dorsal: informa a força aplicada pelo músculo grande dorsal, e produz o movimento de abdução do braço.

Ângulo Cotovelo: Mostra, em graus, o ângulo de rotação do antebraço em relação à linha do braço.

Ângulo Ombro Sag: Mostra, em graus, o ângulo de rotação do braço em relação à linha do corpo no plano sagital.

Ângulo Ombro Sag: Mostra, em graus, o ângulo de rotação do braço em relação à linha do corpo no plano coronal.

Plano de visão: Por meio da seleção de uma das opções abaixo, é possível visualizar o modelo virtual do membro superior humano nos três planos cardinais: sagital,

coronal e horizontal.

Botões de navegação: Por meio desses botões é possível navegar no ambiente virtual do sistema. Cada botão tem uma função específica descrita a seguir:

+ - Esse botão tem a função zoom no ambiente. Após selecionar essa função por meio de um clique no botão, se arrastar o *mouse* no ambiente de baixo para cima, o objeto é ampliado e, de cima pra baixo, é reduzido.

R - Esse botão tem a função de rotacionar o objeto na cena. Após selecionar essa função por meio de um clique no botão, o objeto é rotacionado no ambiente de quatro modos:

- se arrastar o *mouse* no ambiente da esquerda pra direita, o objeto é rotacionado em sentido anti-horário ao redor do eixo y,
- se arrastar o *mouse* no ambiente da direita para esquerda , o objeto é rotacionado em sentido horário ao redor do eixo y,
- se arrastar o *mouse* no ambiente de baixo pra cima, o objeto é rotacionado em sentido contrário ao mouse ao redor do eixo x, e
- se arrastar o *mouse* no ambiente de cima para baixo, o objeto é rotacionado em sentido contrário ao mouse ao redor do eixo x.

T - Esse botão tem a função de transladar o objeto na cena. Após selecionar essa função por meio de um clique no botão, o objeto é transladado no ambiente de quatro modos:

- se arrastar o *mouse* no ambiente da esquerda pra direita, o objeto é transladado em sentido contrário ao mouse, ou seja, da direita para esquerda,

- se arrastar o *mouse* no ambiente da direita para esquerda, o objeto é transladado em sentido contrário ao mouse, ou seja, da esquerda pra direita,
- se arrastar o *mouse* no ambiente de baixo pra cima, o objeto é transladado em sentido contrário ao mouse, de cima para baixo,
- se arrastar o *mouse* no ambiente de cima para baixo, o objeto é transladado em sentido contrário ao mouse, de baixo pra cima.

G - Esse botão tem a função de rotacionar o objeto na cena. Após selecionar essa função por meio de um clique no botão, o objeto é girado em torno do próprio eixo no ambiente de quatro modos:

- se arrastar o *mouse* no ambiente da esquerda pra direita, o objeto é rotacionado em sentido anti-horário ao redor do eixo y,
- se arrastar o *mouse* no ambiente da direita para esquerda , o objeto é rotacionado em sentido horário ao redor do eixo y,
- se arrastar o *mouse* no ambiente de baixo pra cima, o objeto é rotacionado em sentido contrário ao mouse ao redor do eixo x, e
- se arrastar o *mouse* no ambiente de cima para baixo, o objeto é rotacionado em sentido contrário ao mouse ao redor do eixo x.

A - Esse botão tema função de atualizar o objeto na cena, ou seja desfazer todas as transformações geométricas produzidas pela seleção das opções propostas nos botões anteriores.

Stop - Esse botão tem a função de interromper o movimento do modelo virtual do membro superior.

Exit - Esse botão tem a função de fechar a tela do sistema.

5.5 Considerações Finais

O sistema desenvolvido permite a modelagem do modelo músculo-esquelético do membro superior humano, a visualização dos movimentos: flexão/extensão do braço, abdução/adução do braço e flexão/extensão do antebraço e a navegação no ambiente virtual.

As fases de especificação e análise do sistema foram imprescindíveis ao processo de desenvolvimento do sistema, por mostrar, com clareza, a visão geral do modelo e as associações existentes entre os objetos do sistema.

A integração das bibliotecas OpenGL, SmallVR e FLTK à linguagem C++ contribui muito para se conseguir uma interface gráfica com o usuário integrada ao ambiente virtual que possibilitou, de uma forma amigável, a manipulação dos objetos na cena e a solicitação de ações, como exemplo realização de movimentos a partir da alteração dos valores de excitação muscular.

O próximo capítulo apresenta os resultados e limitações do sistema proposto, baseado em experimentos realizados com o mesmo.

Capítulo 6
Resultados e Limitações

6.1 Introdução

Este capítulo apresenta dois estudos de caso relacionados à simulação de movimentos realizados pelo membro superior humano, empregando as equações apresentadas no Capítulo 3 e o protótipo detalhado no Capítulo 5. Através destes estudos de caso, um conjunto de resultados e limitações apresentados pelo sistema são discutidos.

6.2 Estudos de Caso

Nos estudos de caso deste trabalho, cada simulação mostra uma composição dos três tipos de movimentos propostos no Capítulo 3. O primeiro estudo de caso trata de uma simulação efetuada para um suposto paciente de vinte anos e o segundo, demonstra a simulação de movimentos correspondente à uma criança fictícia com sete anos de idade. Estes dois estudos de caso foram escolhidos para evidenciar a diferença no sistema de equilíbrio causada pelas medidas antropomórficas e pelas forças peso dos segmentos dos usuários de prótese selecionados. As medidas antropomórficas foram obtidas de tabelas do livro [Hall 2000] e os valores de massa, do artigo [**?**].

6.2.1 Estudo de Caso 1: Paciente de vinte anos de idade

Este estudo de caso é realizado considerando as medidas de um paciente de vinte anos de idade, do gênero feminino e com massa igual a 55kg. Para possibilitar uma análise mais detalhada dos movimentos realizados na simulação, o resultado é mostrado, na seqüência, em três fases, uma para cada tipo de movimento:

Primeira fase: Esta fase mostra a realização do movimento flexão-extensão do antebraço. Considerou-se, primeiramente, que o braço se encontrava em posição de repouso. Em seguida o valor das forças aplicadas pelos músculos bíceps e tríceps foram alterados respectivamente para 56 N(Newton - unidade padrão de medida de acordo com o Sistema Internacional de Unidades) e 28 N, por meio da barra de rolagem. Neste momento, o sistema chama a função *VerificaSentidoMovimento()* e o sentido do movimento é definido seguindo as condições de equilíbrio estático (de acordo com as equações apresentadas na Seção 3.3). Enquanto houver diferença de torque na direção do movimento, é validada a amplitude do movimento pelo método *ValidaAmplitude()* da classe Movimento e também o tamanho do músculo pelo método *ValidaTamanhoMusculo()* da classe Músculo. Determinado o sentido do movimento e validado as condições acima citadas, o antebraço foi rotacionado, pelo método, *RotateBy()* da classe SmVR_CGeometricObject do SmallVR, de 0.5º em 0.5º até que atingiu a condição de equilíbrio estático, depois de ser rotacionado 16.5º, como Figura 6.1.

À medida que o antebraço é rotacionado, as coordenadas dos segmento são atualizadas e, consequentemente, os pontos de inserção dos músculos são alterados. Por meio da função *CalculaPosMusculo* foram calculados os tamanhos dos músculos e é feito uma chamada aos métodos *DesenhaMusculo()* e *DesenhaMusculoT()* da classe Objeto3D para redesenhar o músculo na posição nova.

O método *handle()* da classe Ogl faz a leitura da posição do mouse no ambiente virtual. Usando este método e as transformações geométricas realizadas por meio dos métodos da classes SmallVR, foi possível a navegação no ambiente virtual. Para ilustrar esta facilidade, ainda na primeira fase, através do clique no botão **G**, que indica a atividade "*Girar*", o sistema fez uma chamada ao método *RotateByOnOBJCS()* da classe SmVR_CGeometricObject para rotacionar o Observador em relação ao Alvo. Isto permitiu uma visualização da cena como se o braço virtual estivesse sendo rotacionado no seu próprio eixo. O resultado desta transformação geométrica é a visão

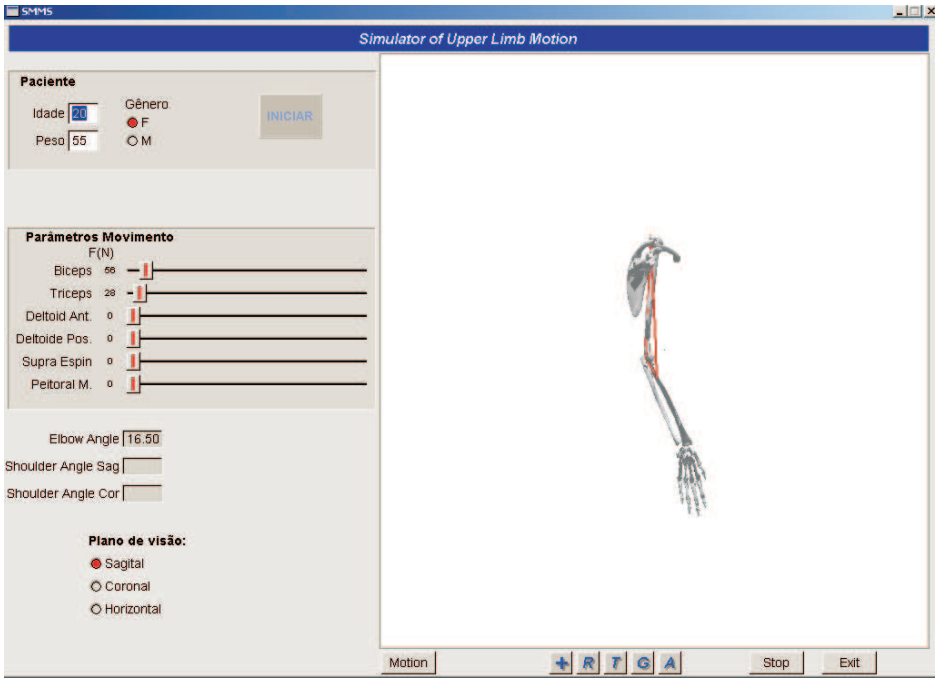

Figura 6.1: Flexão do antebraço

do braço virtual, após o giro de 180º, mostrado na Figura 6.2.

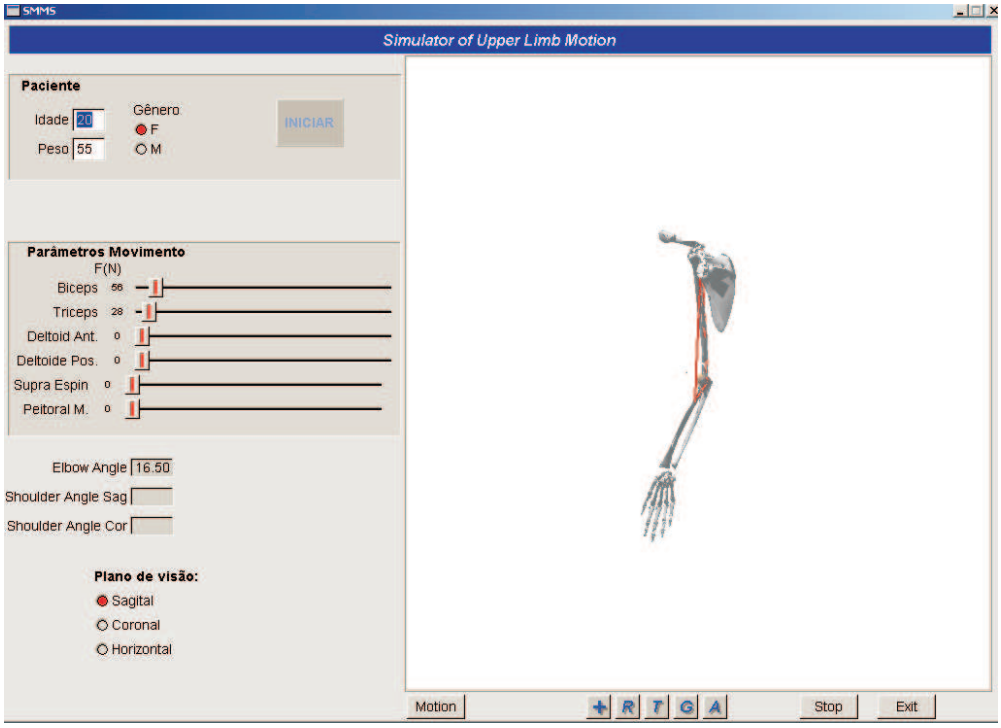

Figura 6.2: Navegação no ambiente virtual através da seleção da opção *Girar*

Segunda fase: A partir da posição de equilíbrio definida na primeira fase, o valor da força imprimida pelo músculo deltóide anterior é alterada para 204 N, por meio da

barra de rolagem. Neste momento, o sistema chama a função *VerificaSentidoMovimento()* para definir o sentido do movimento seguindo as condições de equilíbrio estático. Além de verificar a diferença dos torques das forças aplicadas no ombro para flexionar o braço, verificou-se também a nova posição do antebraço no espaço e sua condição de equilíbrio, ocasionando uma rotação do antebraço até que a condição de equilíbrio para a nova posição seja alcançada. Enquanto verificava-se a condição de equilíbrio para rotação do braço e antebraço, a cada 0.5º de rotação, foram validados a amplitude e tamanho dos músculos pelos métodos *ValidaAmplitude()* e *ValidaTamanhoMusculo()* respectivamente, para cada articulação envolvida. O resultado deste movimento foi uma flexão de 19º no braço e, conseqüentemente, numa extensão no antebraço de 15º. A posição de equilíbrio estático encontrada após a aplicação dessas forças é mostrada na Figura 6.3.

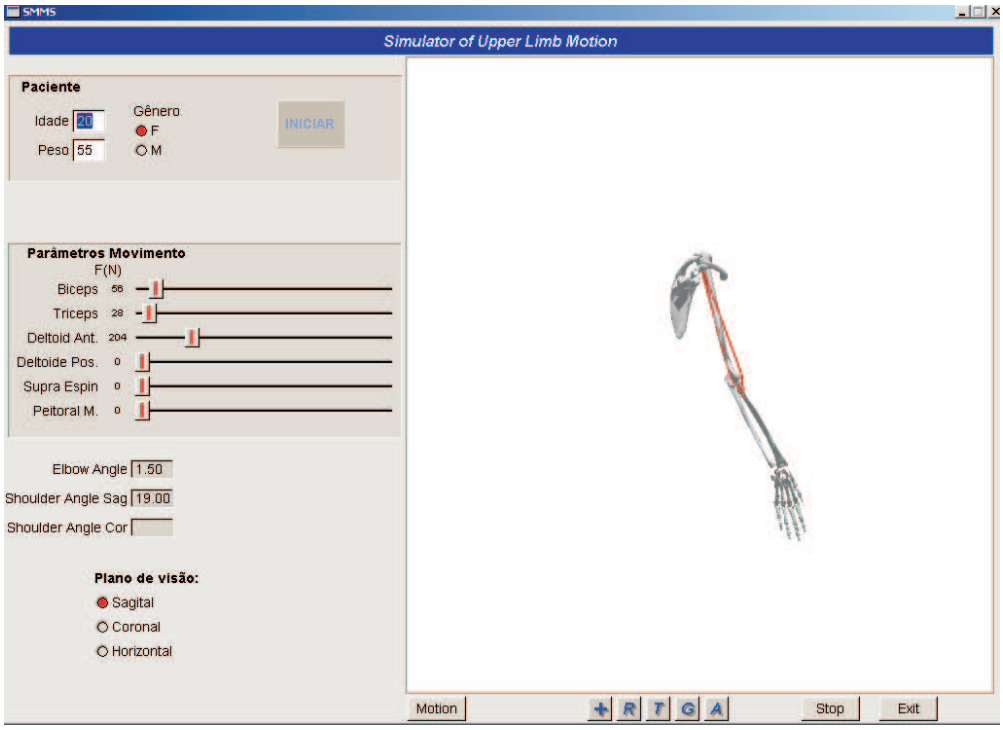

Figura 6.3: Flexão do braço

Para melhor visualização do ângulo resultante após a flexão do braço, foi selecionada a opção de navegação zoom por meio de um clique no botão +. Para conseguir uma ampliação da cena, arrastou-se o mouse de baixo pra cima para que se conseguisse

ver com nitidez o ângulo formado entre o braço e o eixo y (Figura 6.4).

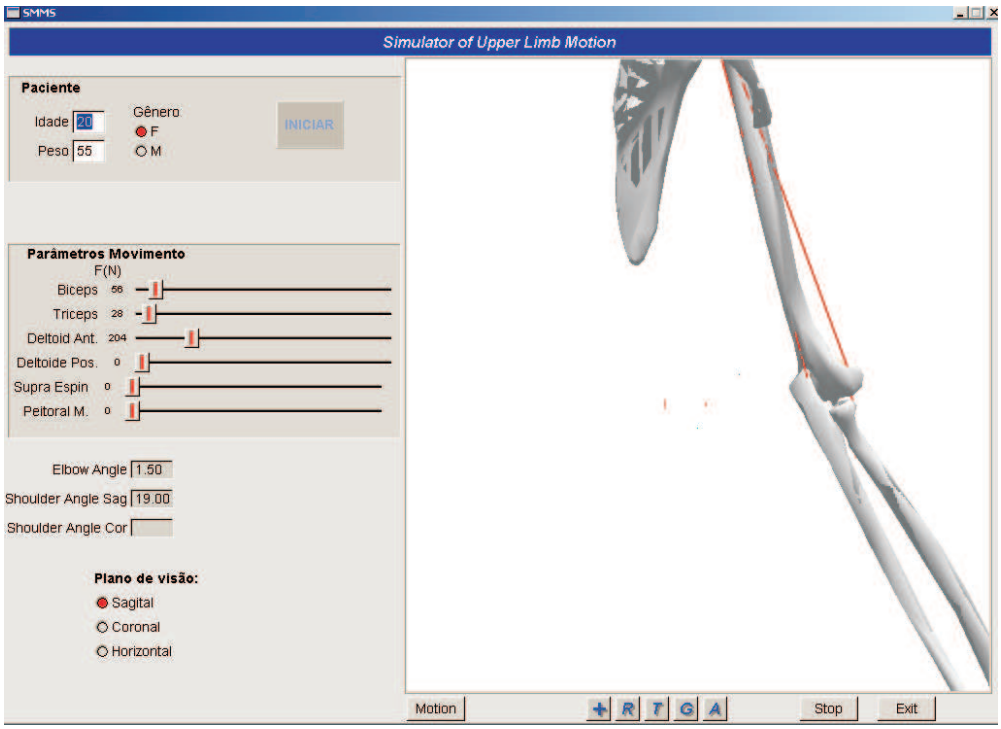

Figura 6.4: Navegação no ambiente: Zoom.

Terceira fase: Após a flexão do antebraço e braço, foi imprimida uma força de 240 N pelo músculo supra-espinhal e outra de 40N pelo seu antagonista, o peitoral maior. Quando o sistema recebeu estes novos valores para os músculos, chamou a função *VerificaSentidoMovimento()* definindo o sentido do movimento de acordo com as condições de equilíbrio estático. Através desta função, além de encontrar a posição de equilíbrio estático do braço também foi determinada a posição de equilíbrio do antebraço obedecendo à condição de equilíbrio estabelecida para a nova posição. Neste momento, enquanto verificava a condição de equilíbrio para rotação do braço no plano sagital, e do antebraço, a cada 0.5º de rotação, foram validados a amplitude e tamanho dos músculos pelos métodos *ValidaAmplitude()* e *ValidaTamanhoMusculo()* respectivamente, para cada músculo envolvido. O resultado deste movimento foi uma abdução no braço de 25º e uma extensão de 1.5º no antebraço, fazendo com o cotovelo não mostrasse mais flexão alguma. O resultado obtido é mostrado no plano de visão coronal (Figura 6.5).

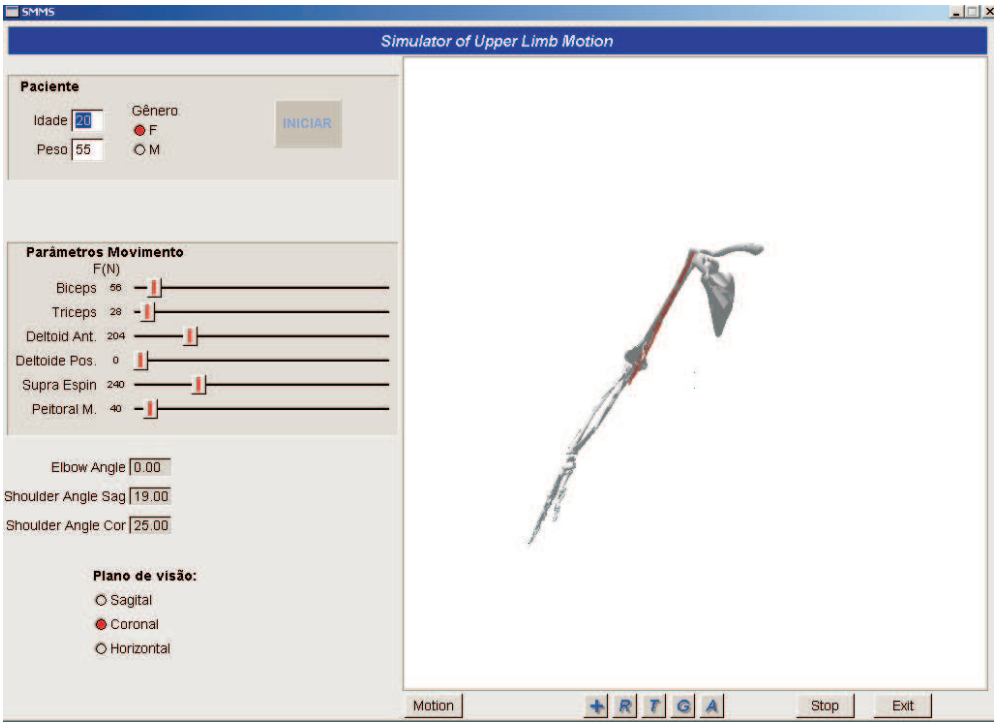

Figura 6.5: Abdução do braço - Plano de Visão: Coronal

A posição resultante, após as três fases, pode ser vista com mais detalhes na Figura 6.6.

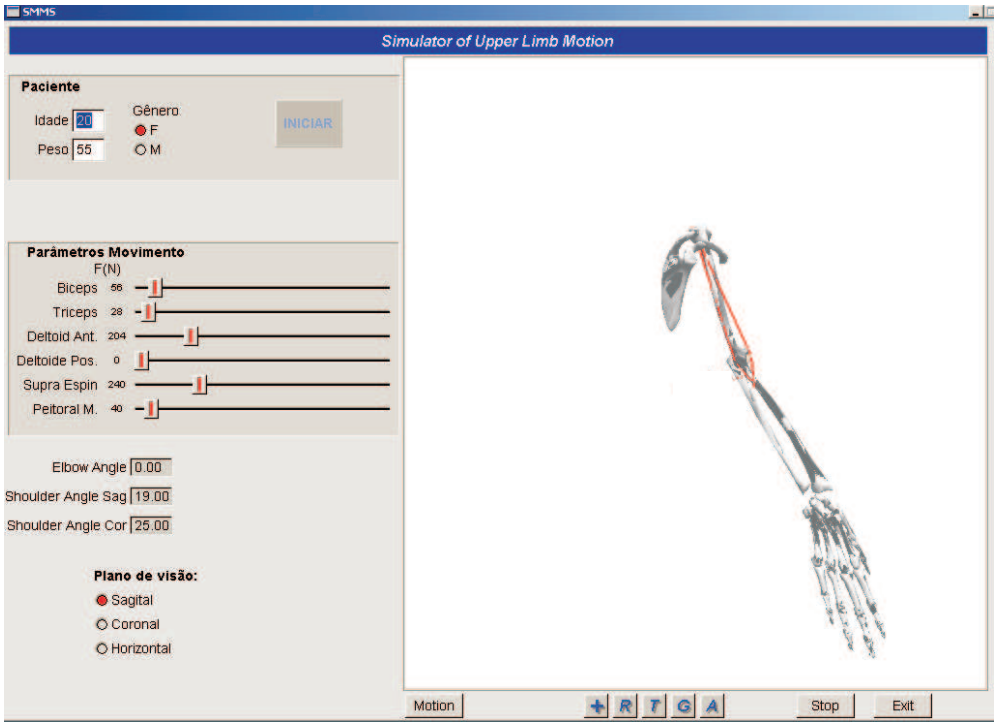

Figura 6.6: Abdução do braço no plano de visão Sagital, vista de uma forma ampliada.

Neste caso, após a realização do movimento, selecionou-se a opção transladar, clicando no botão **T** para levar o objeto da cena para uma posição mais baixa e, logo após, selecionou-se a opção + para aproximar a cena. Esta navegação no ambiente virtual resultou numa visualização mais detalhada do braço.

6.2.2 Estudo de Caso 2: Paciente de sete anos de idade

Para este estudo de caso foram utilizadas as medidas de um paciente de sete anos de idade, do gênero masculino e com massa igual a 20 kg. Para possibilitar uma análise mais detalhada dos movimentos realizados na simulação, o resultado é mostrado na seqüência em três fases, uma para cada tipo de movimento:

Primeira fase: Esta fase mostra a realização do movimento flexão-extensão do antebraço. Considerando, primeiramente, que o braço se encontrava em posição de repouso. Logo em seguida aplicou-se uma força de valor igual a 20 N imprimida pelo bíceps juntamente à uma força de 8 N imprimida pelo tríceps, como mostrado na Figura 6.7, o antebraço foi flexionado 14.5º, permanecendo a partir daí, em condição de equilíbrio estático.

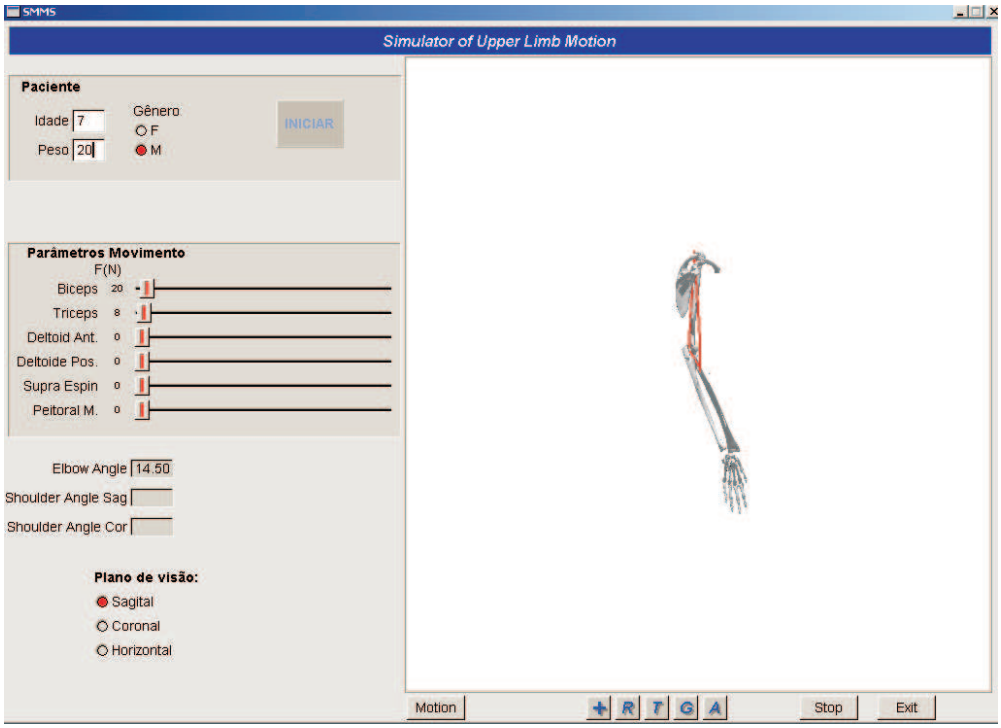

Figura 6.7: Flexão do antebraço

Segunda fase: A partir da posição de equilíbrio resultante na primeira fase, aplicou-se uma força de valor igual a 92 N pelo músculo deltóide anterior, ocorrendo uma flexão de 29º no braço e, conseqüentemente, uma extensão no antebraço de 14.5º, fazendo com o cotovelo não mostrasse mais flexão alguma. A posição de equilíbrio estático encontrada após a aplicação dessas forças é mostrada na Figura 6.8.

Terceira fase: Após a flexão do antebraço e braço, foi imprimida uma força de 100 N pelo músculo supra-espinhal. A aplicação desta força gerou uma abdução no braço de 43º. O resultado obtido é mostrado em dois planos de visão: Figura 6.9 - plano sagital e Figura 6.10 - plano coronal.

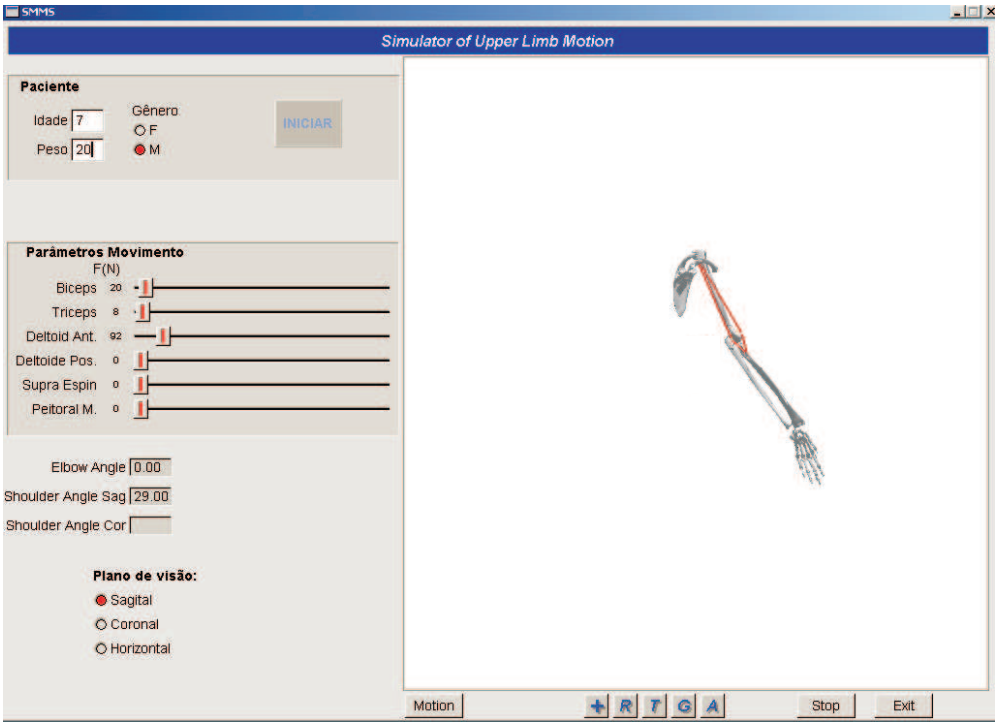

Figura 6.8: Flexão do braço

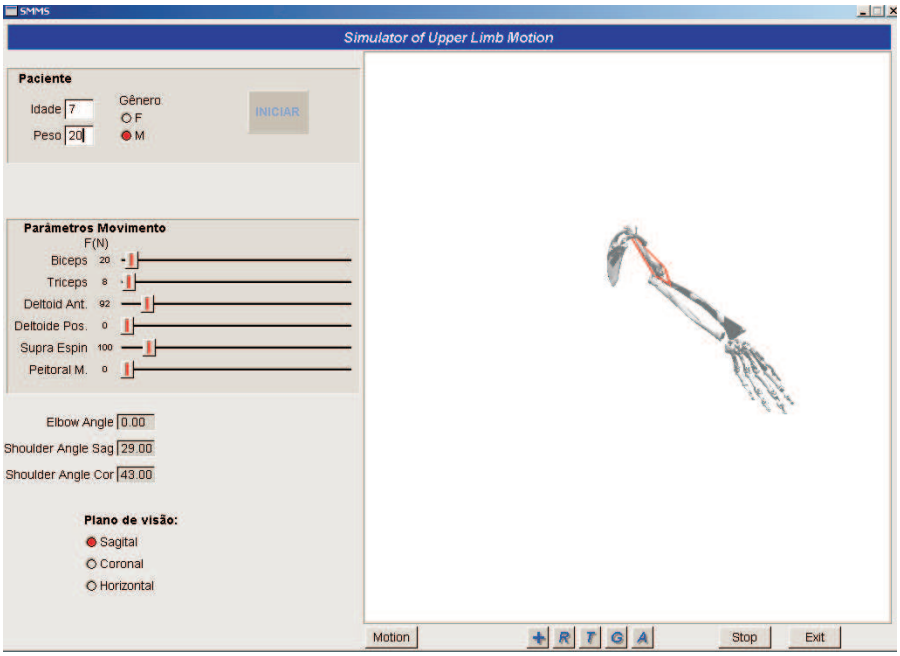

Figura 6.9: Abdução do braço - Plano de Visão: Sagital

6.3 Limitações do SMMS

Neste trabalho, foi realizada a simulação de movimentos do membro superior apenas para um número limitado de músculos não considerando todos os músculos que par-

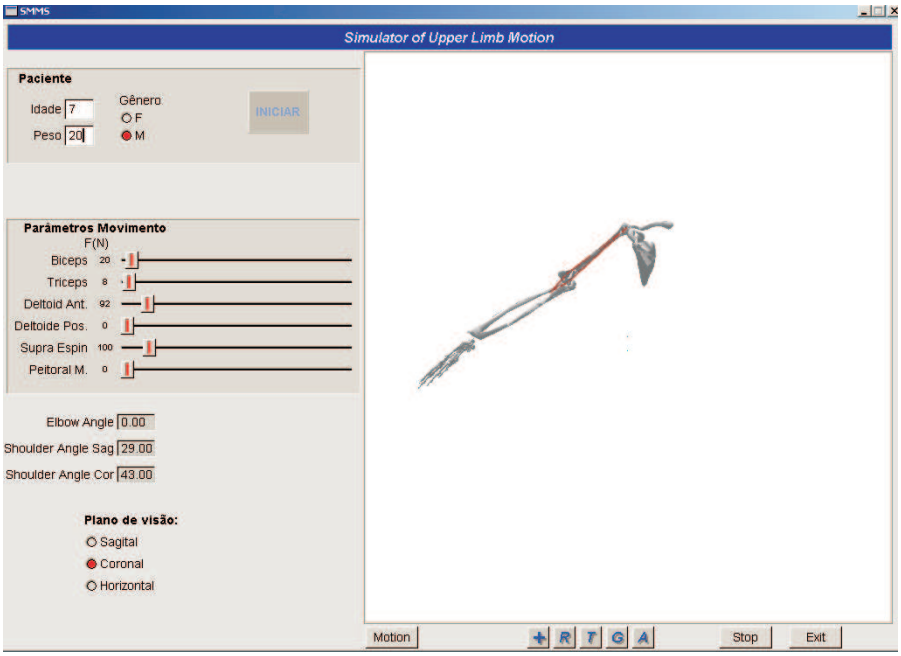

Figura 6.10: Abdução do braço - Plano de Visão: Coronal

ticipam de um mesmo tipo de movimento como mostrado na Seção 3.2.2.

Uma outra limitação a ser considerada é a quantidade de movimentos simulados. Este sistema limita-se a simular os movimentos de flexão-extensão do antebraço, flexão-extensão e adução-abdução do braço. Para uma simulação completa de movimentos de uma prótese de membro superior, seria necessário a simulação dos movimentos: pronação-supinação do antebraço, flexão-extensão do punho e o movimento de pinça.

6.4 Considerações Finais

Este capítulo apresentou o funcionamento do sistema proposto por meio de dois estudos de caso. Foi apresentada a modelagem do sistema músculo-esquelético, gerada a partir da biblioteca de classes proposta no Capítulo 4 e, também, a facilidade de interação com o usuário através da interface apresentada. A navegação no ambiente virtual foi útil para uma análise mais detalhada dos resultados do movimento. Foi mostrado também o resultado da análise do sistema de equilíbrio estáticos das forças musculares e gravitacionais como mostrados na seção 3.3.

Com o intuito de evidenciar a influência das medidas antropomórficas no sistema de

equilíbrio, foram escolhidos para cada estudo de caso dois pacientes com idades distintas. O próximo capítulo conclui este trabalho e apresenta os trabalhos futuros.

Capítulo 7
Conclusões e Trabalhos Futuros

7.1 Introdução

Esta dissertação teve como principal objetivo a criação de uma biblioteca de classes genérica(ApêndiceG) para modelagem e simulação de membros do corpo humano. Um outro objetivo foi a implementação de um sistema de simulação de movimentos do membro superior humano, que ao receber o valor das forças aplicadas pelos músculos exibe no ambiente virtual os movimentos efetuados.

Para uma análise mais detalhada da modelagem e do movimento gerado, foram implementadas funções que permitem navegação no ambiente por meio técnicas de realidade virtual. A seguir, neste capítulo são apresentados os principais tópicos abordados nesta dissertação, as contribuições científicas relacionadas com o trabalho e por fim, os possíveis trabalhos futuros.

7.2 Conclusões

Durante a fase de pesquisa verificou-se a existência de vários sistemas de modelagem e simulação de membros do corpo humano, porém constatou-se que a maioria deles não permitem simulações de movimentos a partir dos valores de forças aplicadas pelos músculos, e dentre os que permitem, não suportam alterações nos valores destas forças em tempo de execução. Também não ficou evidenciado o uso de um modelo de classes abrangente o suficiente para a simulação dos movimentos por estes softwares.

Este trabalho apresentou um estudo realizado de forma multidisciplinar envolvendo as áreas Anatomia Humana, Biomecânica, Física, e Matemática. Este estudo possibilitou

a criação de uma biblioteca de classes, parametrização de estímulos de forças musculares e desenvolvimento de equações para um sistema de equilíbrio estático.

A biblioteca de classes, uma das principais contribuições deste projeto, é parte da arquitetura do sistema e foi proposta de forma genérica, para dar suporte não só a projetos de simulação de movimentos do membro superior, mas também a modelos de simulação de outros tipos de movimentos realizados pelo corpo humano. As equações do sistema de equilíbrio foram criadas por meio de um modelo de geometria muscular para músculos envolvidos na atuação das articulações: cotovelo e ombro.

Uma outra contribuição importante foi a implementação de um software de simulação de movimentos para uso em prótese de membro superior humano. Este sistema, ao receber o sinal de alteração dos valores de forças aplicadas pelos músculo, faz uma análise da condição de equilíbrio estático e mostra o movimento realizado pelo membro superior humano no ambiente virtual. É importante ressaltar que este sistema, além de simular um movimento realizado por estímulo de forças informadas pelos objetos de interface, também possibilita a leitura de um conjunto de forças musculares previamente definidas para cada intervalo de tempo de uma tabela de valores. Nestas duas situações, o sistema permite a alteração dos valores destas forças em tempo real, ou seja, simultaneamente à execução do movimento.

Destaca-se como contribuição também, o uso de técnicas de Realidade Virtual na implementação do sistema. Isto permitiu que profissionais da área pudessem interagir e navegar no ambiente virtual desenvolvido assimilando com mais naturalidade os movimentos simulados.

Os estudos de caso apresentados no capítulo anterior mostram que o sistema funciona de um forma satisfatória.

7.3 Trabalhos Futuros

Considerando a arquitetura proposta, a modelagem e simulação computacional desenvolvidas neste trabalho, pôde-se visualizar os trabalhos futuros relativos à implementação do sistema proposto:

- Inclusão da participação de outros músculos no modelo desenvolvido, bem como a simulação de outros tipos de movimentos, como exemplo pronação e supinação do braço e antebraço, e também os movimentos do punho. Isto permitiria a completa simulação de movimentos realizados por uma prótese;

- Desenvolvimento de técnicas que permitam a simulação de próteses virtuais, através de captura de sinais eletromiográficos usando sensores acoplados aos músculos de pacientes reais. Estas técnicas permitiriam o envio dos valores de forças aplicadas pelos músculos do paciente durante o tempo de estimulação dos músculo, permitindo ao paciente com deficiência física, a visualização do movimento que seria realizado pela prótese real;

- A visualização da prótese virtual por meio do uso de técnicas de realidade aumentada agregaria mais realismo na simulação do uso de próteses;

- Modelagem dos músculos envolvidos no sistema para substituir a representação por linha de ação, trazendo mais realismo no modelo músculo-esquelético apresentado;

- Análise da influência da fadiga muscular na simulação dos movimentos.

Referências Bibliográficas

[Anatomy of Human Organ Systems 2004]ANATOMY of Human Organ Systems. [S.l.]: Acesso em: 25/04/2004, 2004. Disponível em: <http://www.bms.abdn.ac.uk/undergraduate/bi25b3cn.htm>.

[Calais-German 1992]CALAIS-GERMAN, B. *Anatomia para o movimento: Introdução à análise das técnicas corporais*. Editora Manole Ltda.: [s.n.], 1992.

[CHARM 1993]CHARM, E. P. . Part i : Technical annex. 1993.

[Davis 1999]DAVIS, M. W. J. N. T. *OpenGL Programming Guide*. 3rd edition. ed. Reading, MA: Addison-Wesley, 1999.

[Davoodi e Loeb 2001]DAVOODI, R.; LOEB, G. E. A software tool for faster development of complex musculoskeletal models in simulink. *American Society of Biomechanics Conference*, 2001.

[DELP e LOAN 1995]DELP, S. L.; LOAN, J. P. A graphics-based software system to develop and analyze models of musculoskeletal structures. *Comput. Bid. Med.*, v. 25, n. 1, p. 21–34, 1995.

[DELP e LOAN 2000]DELP, S. L.; LOAN, J. P. A computational framework for simulating and analyzing human and animal movement. *Computing in Science & Engineering*, v. 2, n. 5, p. 46–55, Setembro/Outubro 2000.

[Delp 2005]DELP, S. S. B. S. L. Three-dimensional representation of complex muscle architectures and geometries. *Annals of Biomedical Engineering*, v. 33(5), p. 661–673, 2005.

[Ditunno 1997]DITUNNO, J. F. *3rd Annual Gait and Clinical Movement Analysis (GCMA) Meeting*. 1997.

[Fagg 2001]FAGG, A. H. *A Model of Muscle Geometry for a Two Degree-Of-Freedom Planar Arm*. [S.l.], 2001.

[Flehmig 1987]FLEHMIG, I. *Desenvolvimento Normal e seus Desvios no Lactente*. [S.l.: s.n.], 1987. Disponível em.

[Freitas et al.]FREITAS, C. M. D. S. et al. *Framework para Construção de Pacientes Virtuais: Uma*. [S.l.: s.n.].

[Fuchs et al. 1998]FUCHS, H. et al. Augmented reality visualization for laparoscopic surgery. In: *First International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI '98)*. Cambridge, MA, USA: Massachusetts Institute of Technology,, 1998.

[Halloran, Petrella e Rullkoetter 2005]HALLORAN, J. P.; PETRELLA, A. J.; RULLKOETTER, P. J. Explicit finite element modeling of total knee replacement mechanics. *Journal of Biomechanics*, v. 38(2), p. 323–331, 2005.

[Hall 2000]HALL, S. J. *Biomecânica Básica*. Terceira edição. [S.l.: s.n.], 2000.

[Hamill e Knutzen 1999]HAMILL, J.; KNUTZEN, K. M. *Bases Biomecânicas do Movimento*. Primeira edição. [S.l.: s.n.], 1999.

[Helm et al. 1992]HELM, F. V. der et al. Geometry parameters for musculoskeletal modelling of the shoulder system. *J. Biomechanics*, v. 25, n. 2, p. 129–144, Fevereiro 1992.

[Huijing 1995]HUIJING, P. A. Parameter interdependence and success of skeletal muscle modelling. *Human Movement Science*, v. 14, p. 443–486, 1995.

[Lake 1997]LAKE, C. Effects of prosthesis training on upper-extremity prosthesis use. *Journal of Prosthesis and Orthotics*, v. 9(1), p. 3, 1997.

[Lemay e Crago 1996]LEMAY, M. A.; CRAGO, P. E. A dinamical model for simulating movements of the elbow, forearm, and wrist. *Journal of Biomechanics*, v. 29(10), p. 1319–1330, 1996.

[Les muscles 1999]LES muscles. [S.l.]: Acesso em 24/10/2005, 1999. Http://www.medecine.unige.ch/ bertrand/cours1/muscle/muscle.html.

[Maciel, Nedel e Freitas 2002]MACIEL, A.; NEDEL, L. P.; FREITAS, C. M. D. S. Anatomy-based joint models for virtual human skeletons. In: *CA '02: Proceedings of the Computer Animation*. Washington, DC, USA: IEEE Computer Society, 2002. p. 220. ISBN 0-7695-1594-0.

[Maciel 2001]MACIEL, A. *Modelagem de Articulações para Humanos Virtuais Baseada em Anatomia*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Agosto 2001.

[Manssour 2004]MANSSOUR, I. H. *Introdução à OpenGL*. www.inf.pucrs.br/ manssour/OpenGL/Introducao.html, Abril, 24 2004.

[MATLAB 1994]MATLAB. *MATLAB - The Language of Technical Computing*. [S.l.]: The MathWorks, Inc., Acesso em 20/09/2005, 1994. Disponível em: <http://www.mathworks.com/products/matlab/>.

[MAUREL 1998]MAUREL, W. *3D Modeling of the Human Upper Limb Including the Biomechanics of Joints, Muscles And Soft Tissues*. Tese (Doutorado) — ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE, 1998.

[Microsystems 2001]MICROSYSTEMS, S. *Object-Oriented Application Analysis and Design for Java TM Technology (UML) OO-226 Student Guide*. [S.l.]: Sun Microsystems, 2001.

[Miller et al. 2000]MILLER, K. et al. Mechanical properties of brain tissue in-vivo: experiment and computer simulation. *Journal of Biomechanics*, v. 33(11), p. 1369–1376, Novembro 2000.

[Moore et al. 2005]MOORE, S. et al. 3d models of blood flow in the cerebral vasculature. *Journal of Biomechanics*, In Press, Junho 2005.

[Multon et al. 1999]MULTON, F. et al. A software system to carry-out virtual experiments on human motion. In: *CA '99: Proceedings of the Computer Animation*. Washington, DC, USA: IEEE Computer Society, 1999. p. 16. ISBN 0-7695-0167-2.

[Olsen e Templeman 2003]OLSEN, A.; TEMPLEMAN, J. *Microsoft Visual C++ .NET Step by Step–Version 2003*. [S.l.: s.n.], 2003.

[Rocha 2000]ROCHA, B. *O Corpo Humano by Breno Rocha*. [S.l.]: Acesso em: 22/09/2004, 2000. Disponível em :<http://www.corpohumano.hpg.ig.com.br/sist_muscular/membros.html>.

[Santos, Kopper e Pinho 2003]SANTOS, M. C. C. dos; KOPPER, R. A. P.; PINHO, M. S. *Minicurso de Realidade Virtual*. [S.l.], 2003.

[Selbie e Caldwell 1996]SELBIE, W. S.; CALDWELL, G. E. A simulation study of vertical jumping from different starting postures. *Journal of Biomechanics*, v. 29(9), p. 1137–1146, Setembro 1996.

[Soares et al. 2003]SOARES, A. et al. The development of a virtual myoeletric prosthesis controlled by an emg pattern recognition system based on neural networks. *Journal of Intelligent Information System*, v. 21, n. 2, p. 127–141, 2003.

[Spitzak 1991]SPITZAK, M. S. C. P. E. B. *Programming Manual FLTK*. 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA, Junho 1991.

[Székely e Satava 1999]SZÉKELY, G.; SATAVA, R. M. *Virtual reality in medicine. Interview by Judy Jones.* November 1999.

[Thelen, Anderson e Delp 2003]THELEN, D. G.; ANDERSON, F. C.; DELP, S. L. Generating dynamic simulations of movement using computed muscle control. *Journal of Biomechanics*, v. 36(3), p. 321–328, 2003.

[Thelen e Anderson 2005]THELEN, D. G.; ANDERSON, F. C. Using computed muscle control to generate forward dynamic simulations of human walking from experimental data. *Journal of Biomechanics*, In Press, 2005.

[Truppe 1995]TRUPPE, M. Interventional vídeo tomography. In: *Laser in Surgery*. [S.l.]: CA: SPIE, 1995. p. 150–152.

[Valdivia 2004]VALDIVIA, R. *Anatomia Funcional*. [S.l.]: Acesso em: 23/09/2005, 2004. Disponível em: <http://www.ugr.es/ dlcruz/>.

[Vieira 1995]VIEIRA, L. R. *O jogo da capoeira: corpo e cultura popular no Brasil*. Rio de Janeiro: [s.n.], 1995.

[Wojcik 2003]WOJCIK, L. A. Modeling of musculoskeletal structure and function using a modular bond graph approach. *Journal of the Franklin Institute*, v. 340, p. 63–76, February 2003.

[Zatsiorsky 1983]ZATSIORSKY, S. The mass and inertia characteristics of the main segments of the human body. *Biomechanics*, V-IIIB, p. 1152–1159, 1983.

Apêndice A

Tabela Antropomórfica

A.1 Dimensões dos Membros Superiores Masculinos

Tabela A.1: Dimensões dos Membros Superiores Masculinos. Fonte: [Hall 2000]

Idade Inicial	**Idade Final**	**Comprimento do braço**	**Comprimento do antebraço**	**Comprimento da mão**
0	1	12.9	11.77	4.33
2	2	14.96	13.66	5.03
3	3	16,34	14.91	5.49
4	4	17.37	15.85	5.84
5	5	18.4	16.8	6.18
6	6	19.6	17.89	6.59
7	7	20.64	18.84	6.93
8	8	21.67	19.78	7.28
9	9	22.53	20.56	7.57
10	10	23.22	21.19	7.8
11	11	23.9	21.82	8.03
12	12	24.76	22.6	8.32
13	16	26.5	24.7	10.5
16	19	28.6	27.5	14.3
19	65	31.3	30.0	19.7

A.2 Dimensões dos Membros Superiores Femininos

Tabela A.2: Dimensões dos membros superiores femininos. Fonte: [Hall 2000]

Idade Inicial	**Idade Final**	**Comprimento do braço**	**Comprimento do antebraço**	**Comprimento da mão**
0	1	12.63	11.68	4.2
2	2	14.88	13.76	4.95
3	3	16,44	15.20	5.46
4	4	17.65	16.32	5.87
5	5	18.68	17.28	6.21
6	6	19.55	18.08	6.50
7	7	20.59	19.04	6.84
8	8	21.63	20.0	7.19
9	9	22.49	20.8	7.48
10	10	23.36	21.6	7.76
11	11	24.39	22.56	8.11
12	12	25.43	23.52	8.45
13	16	27.3	24.3	10.3
16	19	27.9	25.2	13.9
19	65	28.5	25.7	18.1

Apêndice B

Método *DesenhaBraco*

```
1   // Esse metodo da classe Objetos3D solicita a criacao dos objetos braço e ombro, e os insere na cena nos pontos definidos
2   // nos seus parâmtros e na altura definiada no parâmetro "alt"
3
4   void Objetos::DesenhaBraco(float POrigemX,float  POrigemY, float PInsercaoX, float PInsercaoY, float alt)
5   {
6
7           SmVR_CPoint Zero(0, 0, 0);
8           SmVR_CPoint teste;
9           tamOssoB[0] = alt;
10          float LargB, ProfB, LargO, ProfO;
11
12          // criação do objeto Braço no modelo virtual
13          Braco = new SmVR_CGeometricObject ("Braco",DesenhaObjBraco);
14
15          // aqui se passa o parâmetro altura do braço para a função DesenhaObjBraco
16          Braco->SetRenderFunctionData(tamOssoB);
17
18          // posiciona o objeto na cena para que a cena seja criada no plano sagital
19          Braco->RotateBy(180,0,1,0);
20
21
22          tamOssoO[0]= tamOssoB[0] * (OmbroLimiteMax[1] - OmbroLimiteMin[1]) / (UmeroLimiteMax[1] - UmeroLimiteMin[1]);
23
24          // criação do objeto Ombro no modelo virtual
25          Ombro = new SmVR_CGeometricObject ("Ombro",DesenhaObjOmbro);
26
27          // aqui se passa o parâmetro altura do ombro para a função DesenhaObjOmbro
28          Ombro->SetRenderFunctionData(tamOssoO);
29
30
31          // Determina a largura do braço correspondente ao valor altura, fazendo a conversão do dados obtidos do arquivo
32          // gerado a partir do 3DS
33          LargB = tamOssoB[0] * (UmeroLimiteMax[0] - UmeroLimiteMin[0]) / (UmeroLimiteMax[1] - UmeroLimiteMin[1]);
34
35          // Do mesmo modo que determina a largura, determina-se a profundidade
36          ProfB = tamOssoB[0] * (UmeroLimiteMax[2] - UmeroLimiteMin[2]) / (UmeroLimiteMax[1] - UmeroLimiteMin[1]);
37
38
39
40          // Determinação da pontos nas extremidades do objeto Braço para facilitar as aplicaçõpes das tranformações
41          // geométricas,  por exemplo, para rotacionar o braço
42          PProximalB->TranslateBy(POrigemX,POrigemY,0);
43          PArtOmbro->TranslateBy(POrigemX-.9,POrigemY,0);
44          PProximalB->GetPointInOCS(Zero, &teste, Root);
45
46          // Determinação da posição do objeto Braço
47          Braco->TranslateBy(LargB/2,PInsercaoY,ProfB/2);
48
49          PDistalB->TranslateBy(PInsercaoX,PInsercaoY,0);
50          PArtCotovelo->TranslateBy(PInsercaoX-.9,PInsercaoY+1.8,0);
51
52          // Inserção dos pontos extremos do objeto brfaço na cena
53          Braco->AddChild(PDistalB);
54          Braco->AddChild(PArtCotovelo);
55
56          // Determina a largura do ombro correspondente ao valor altura, fazendo a conversão do dados obtidos do arquivo
57          // gerado a partir do 3DS
58          LargO = tamOssoO[0] * (OmbroLimiteMax[0] - OmbroLimiteMin[0]) / (OmbroLimiteMax[1] - OmbroLimiteMin[1]);
59
60          // Do mesmo modo que determina a largura, determina-se a profundidade
61          ProfO = tamOssoO[0] * (OmbroLimiteMax[2] - OmbroLimiteMin[2]) / (OmbroLimiteMax[1] - OmbroLimiteMin[1]);
62          Ombro->RotateBy(180,0,1,0);
63
64          Braco->GetPointInOCS(Zero, &teste, Root);
65
66          // Inserção dos objetos na cena
67          Ombro->TranslateBy(LargO/2,tamOssoB[0]+ teste.Y - tamOssoO[0] ,0);
68          Braco->AddChild(Ombro);
69          Root->AddChild(Braco);
70
71  }
```

Código B.0.1:

Apêndice C

Método *DesenhaObjBraco*

```
 1
 2  // Essa função cria o objeto Braço tridimensional
 3  int DesenhaObjBraco(void *p)
 4  {
 5          int i,j, ind, NumTriangulos;
 6
 7          // instância da classe Tri -  classe que contém a ordem dos triângulos (formados por três vértices) para
 8          // criação do braço
 9          Tri *tri;
10
11          // instância da classe Vec - classe que contém os todos os vértices do braço
12          Vec *vec;
13
14          // instância da classe Normal - classe que contém os vetores normais para cada vértice do triângulo
15          Normal *N;
16          float x, y, z, a, b, c;
17
18          // criação dos vetores que contém os valores máximos e mínimos dos vértices do objeto
19          float MaxMinX[2];
20          float MaxMinY[2];
21          float MaxMinZ[2];
22
23
24          // criação da variável altura que receberá o valor passado pelo parâmetro da função(p)
25          float *alt;
26          GLdouble altura;
27
28          alt = (float*)p;
29          altura = *alt;
30
31          //Definição da cor do objeto
32          glColor3f(1,1,1); // cor branca
33
34          // Variável que recebe a quantidade de triângulos do objeto
35          NumTriangulos = ObjModeloUmero.shapes[0].num_triangles;
36
37          // Determinação dos Limites do Objeto, é feita uma conversão dos limites definidos em 3D MAX, objeto inicial de
38          // onde se buscou os dados do objeto para o objeto a ser criado em OpenGL, a fim de se adequar ao cenário criado
39          // nesse projeto. As coordenas  Z's serão transformadas para Y, as Coordenadas Y's para X e as X's para Z.
40          MaxMinX[0] = UmeroLimiteMax[0];    //Limite max de X
41          MaxMinX[1] = UmeroLimiteMin[0];    //Limite min de X
42          MaxMinY[0] = UmeroLimiteMax[1];    //Limite max de Y
43          MaxMinY[1] = UmeroLimiteMin[1];    //Limite min de Y
44          MaxMinZ[0] = UmeroLimiteMax[2];    //Limite max de Z
45          MaxMinZ[1] = UmeroLimiteMin[2];    //Limite min de Z
46
47          // Conversão dos limites de referencia do objeto 3DS  para o objeto em OpenGL
48          // A variável "b" representa a altura do objeto em OpenGL
49          b = altura;
50
51          // A variável "a" representa a largura do objeto em OpenGL
52          a = (MaxMinX[0] - MaxMinX[1]) * b/(MaxMinY[0] - MaxMinY[1]);
53
54          // A variável "c" representa a profundidade do objeto em OpenGL
55          c = (MaxMinZ[0] - MaxMinZ[1]) * b/(MaxMinY[0] - MaxMinY[1]);
56
57          // As funções MapeiaPonto convertem as medidas orginais do objeto para a medida requerida no sistema determinada
58          // pelo parâmetro altura.
59          UmeroLimiteMinGL[0] = MapeiaPontoX(UmeroLimiteMin[0], MaxMinX, a);
60          UmeroLimiteMinGL[1] = MapeiaPontoY(UmeroLimiteMin[1], MaxMinY, b);
61          UmeroLimiteMinGL[2] = MapeiaPontoZ(UmeroLimiteMin[2], MaxMinZ, c);
62
63          UmeroLimiteMaxGL[0] = MapeiaPontoX(UmeroLimiteMax[0], MaxMinX, a);
64          UmeroLimiteMaxGL[1] = MapeiaPontoY(UmeroLimiteMax[1], MaxMinY, b);
65          UmeroLimiteMaxGL[2] = MapeiaPontoZ(UmeroLimiteMax[2], MaxMinZ, c);
66
67
68          // criação das primitivas triângulos em OpenGL para modelagem do objeto. Nesse trecho de código é feita
69          // uma busca por todos os triângulos e os vértices a ele correspondentes e então criados os vértices em OpenGL
70          // na ordem correta.
71          for (i = 0; i < NumTriangulos; i++)
72          {
73                  // inicializa a primiva triângulo
74                  glBegin(GL_TRIANGLES);
75
76                  // obtenção de cada triângulo do objeto Umero (correspondente ao segmento braço). Cada modelo, ou seja, objeto,
77                  // é formado por um conjunto de shapes (como exemplo, o objeto antebraço é formado por dois shapes: rádrio e ulno),
78                  // sendo esse objeto, o braço, formado um único shape. Cada shape é formado por um conjunto de triângulos na
79                  // seqüência definida no arquivo onde contém o modelo (ModeloUmero.txt) e cada triângulo contém três vértices
80                  // que também são lidos na seqüência definida no aquivo do modelo.
81
82                  tri = ObjModeloUmero.shapes[0].triangles + i;
83                  for (j = 0; j < 3; j++)
84                  {
85                          // obtenção do vetor normal para o vértice a ser criado
86                          N = ObjModeloUmero.shapes[0].normals + tri->n[j];
87                          glNormal3f(N->y, N->z, N->x);
88
89                          // obtenção de cada vértice para compor o triângulo
90                          vec = ObjModeloUmero.shapes[0].vertices + tri->v[j];
91                          x = MapeiaPontoX(vec->y, MaxMinX, a);
92                          y = MapeiaPontoY(vec->z, MaxMinY, b);
93                          z = MapeiaPontoZ(vec->x, MaxMinZ, c);
94                          glVertex3f(x, y, z);
95                  }
96
97          glEnd();
98          }
99          return 1;
100 }
```


Código C.0.2:

Apêndice D

Função *VerificaSentidoMovimento*

```
1   // Essa função controla todos o movimentos, propostos nesse projeto, realizados pelo membro superior humano. A
2   // cada movimento realizado, é feito um estudo do equilíbrio das forças para determinar o sentido do movimento
3   float VerificaSentidoMovimento(char nomeMovimento[30], float AngOmbSag, float AngOmbCor, float AngCot){
4
5           float AngMov, AngPeso, TorqueA, TorqueB, TorquePeso, teste;
6
7           // instâncias da classe Osso
8           Osso ObjOR, ObjOM, ObjOU;
9           float RB; //braço de momento da força peso do braço
10          float RA; //braço de momento da força peso do antebraço
11
12
13          // Essa função retorna o objeto Rádio (correspondente ao segmento antebraço) do tipo Osso (correspondente a classe
14          // Segmento)
15          ObjOR = BuscaOsso("Radio");
16
17          // Essa função retorna o objeto Mao do tipo Osso (correspondente a classe Segmento)
18          ObjOM = BuscaOsso("Mao");
19
20          // Essa função retorna o objeto Umero(correspondente ao segmento Braço) do tipo Osso (correspondente a classe]
21          // Segmento)
22          ObjOU = BuscaOsso("Umero");
23
24
25          // Verifica o tipo de movimento
26          switch (nomeMovimento[0]){
27
28          // Caso o tipo de movimento a ser verificado seja flexão do antebraço
29          case 'F':
30
31            if (nomeMovimento[6] == 'A'){   // Flexão do Antebraço
32
33              // Verifica se o braço está flexionado ou aduzido
34                  if ((AngOmbSag > 0) && (AngOmbCor > 0)){
35
36                          // Determina o ângulo que a força peso do antebraço faz com a linha do antebraço
37                          AngPeso = acos((ObjOR.YCoordDistal - ObjOR.YCoordProximal)/ObjOR.Comprimento);
38
39                          // Determina o ângulo da força peso do antebraço faz com a linha do antebraço, porém
40                          // considerando a direção da mão
41                          AngPeso = M_PI - AngPeso;
42
43                          // Determina o braço de momento da força peso do antebraço em relação a articulação do cotovelo
44                          RA = (ObjOR.Comprimento / 2) * cos(AngPeso);
45                  }
46                  else // Verifica se o braço está flexionado
47                          if (AngOmbSag > 0){
48
49                                  // Define ângulo da força peso quando o braço está flexionado com ângulo menor que 90°, mas
50                                  // considerando o ângulo do cotovelo também, se é menor ou maior que 90°
51                                  if (AngOmbSag < 90){
52                                          if (AngCot <= 90)
53                                                  AngPeso = M_PI - AngOmbSag*M_PI/180 - AngCot*M_PI/180;
54                                          else
55                                                  AngPeso = AngOmbSag*M_PI/180 + AngCot*M_PI/180 - M_PI;
56                                  }
57                                  else
58                                          // Define ângulo da força peso quando o braço está flexionado com ângulo igual a 90°, mas
59                                          // também considerando o ângulo do cotovelto.
60                                          if (AngOmbSag = 90)
61                                                  if (AngCot <= 90)
62                                                          AngPeso = (M_PI/2) - AngCot*M_PI/180;
63                                                  else
64                                                          AngPeso = AngCot - (M_PI/2);
65                                          else
66                                                  //Define ângulo da força peso quando o braço está flexionado com ângulo maior que 90°
67                                                  if (AngCot < 90)
68                                                          AngPeso = M_PI - AngOmbSag*M_PI/180 - AngCot*M_PI/180;
69                                                  else
70                                                          AngPeso = AngOmbSag*M_PI/180 - M_PI/2;
71
72                          } // Verifica se o braço está aduzido, se tiver, atribui o ângulo de adução ao ângulo da força peso
73                          else
74                            if (AngOmbCor > 0)
75                                  AngPeso = AngOmbCor*M_PI/180;
76                            else //Define ângulo da força peso do antebraço quando o braço se encontra na linha do corpo
77                                          if ( AngCot <= 90)
78                                                  AngPeso = AngCot*M_PI/180;
79                                          else
80                                                  AngPeso = M_PI - AngCot*M_PI/180;
81
82
83                  //Cálculo do torque da força peso do antebraço
84                  if (AngOmbCor > 0)
85                          TorquePeso = (ObjOR.Peso + ObjOM.Peso)* 9.8 * (ObjOR.Comprimento / 2) * cos(AngPeso);
86                  else
87                          if ((AngOmbSag > 0) & (AngOmbSag < 90) & (AngCot <= 90))
88                                  TorquePeso = (ObjOR.Peso + ObjOM.Peso)* 9.8 * (ObjOR.Comprimento / 2) * sin(M_PI - AngPeso);
89                          else
90                                  TorquePeso = (ObjOR.Peso + ObjOM.Peso)* 9.8 * (ObjOR.Comprimento / 2) * sin(AngPeso);
91
92
93                  // Torque exercido pela força Bíceps
94                  TorqueA = (SliderBic->value() * atof(outRB->value()));
95
96                  // Torque exercido pela força Tríceps
97                  TorqueB = (SliderTri->value() * atof(outRT->value()));
98
```


Código D.0.3:

```
                // Verifica o sentido do torque da força peso para inclusão no sistema de equilíbrio
                if (AngOmbCor = 0){
                        if (AngOmbSag > (180 - AngCot))
                                TorqueA = TorqueA + TorquePeso;
                        else
                                TorqueB = TorqueB + TorquePeso;
                }
                else
                        if (AngOmbCor <= 90)
                                TorqueB = TorqueB + TorquePeso;
                        else
                                TorqueA = TorqueA + TorquePeso;


                // Define a direção de rotação do antebraço
                if(TorqueA > TorqueB)
                        AngMov = 0.5;
                else if (TorqueA < TorqueB)
                                AngMov = -0.5;
                        else AngMov = 0;

                if ((TorqueA - TorqueB) > -5 && (TorqueA - TorqueB) < 5 )
                        AngMov = 0;
           }
          else
          {
                    //----- Caso o tipo de movimento a ser verificado seja flexão do braço, defini-se o sentido do torque de
                    //----- acordo com a posição que o braço se encontra
                 if (AngOmbSag <= 90){
                        AngPeso =  AngOmbSag * M_PI / 180;
                        TorquePeso = ObjOU.Peso * 9.8 * (ObjOU.Comprimento) * sin(AngOmbSag*M_PI/180) +
                                        (ObjOR.Peso + ObjOM.Peso)* 9.8 * ((ObjOU.Comprimento * sin(AngOmbSag*M_PI/180) +
                                        (ObjOR.Comprimento/2) * sin(AngPeso)));
                        }
                else {

                        // Definição do torque da força peso do braço e antebraço para variadas posições dos dois segmentos, pois
                        // a mudança do centro de gravidade, faz com o que o torque seja diferente devido a alteração no valor do
                        // braço de momento
                        if ((AngOmbSag + AngCot) >=  180){
                                AngPeso = M_PI - AngOmbSag*M_PI/180 - AngCot*M_PI/180;
                                TorquePeso = ObjOU.Peso * 9.8 * (ObjOU.Comprimento) * sin(AngOmbSag*M_PI/180) + (ObjOR.Peso +
                                                ObjOM.Peso)* 9.8 * ((ObjOU.Comprimento * sin(M_PI - AngOmbSag*M_PI/180) +
                                                        (ObjOR.Comprimento/2) * sin(AngPeso)));
                        }
                        else {
                                AngPeso = AngOmbSag + AngCot - M_PI;
                                TorquePeso = ObjOU.Peso * 9.8 * (ObjOU.Comprimento) * sin(AngOmbSag*M_PI/180) + (ObjOR.Peso +
                                                ObjOM.Peso)* 9.8 * ((ObjOU.Comprimento * sin(M_PI - AngOmbSag*M_PI/180) -
                                                        (ObjOR.Comprimento/2) * sin(AngPeso)));
                        }

                }


                // Determina o torque exercido pela força Deltóide
                TorqueA = SliderDel->value() * atof(outRD->value());

                // Determina o torque exercido pela força Peitoral + Peso
                TorqueB = SliderPei->value() * atof(outRP->value()) + TorquePeso; //ObjOU.Peso* 9.8 * (ObjOU.Comprimento) * sin(AngPeso)
                         + ObjOR.Peso + ObjOM.Peso)* 9.8 * (ObjOU.Comprimento + ObjOR.Comprimento/2) * sin(AngPeso);


                // Define a direção de rotação do braço
                if(TorqueA > TorqueB)
                        AngMov = 0.5;
                else if (TorqueA < TorqueB)
                                AngMov = -0.5;
                        else AngMov = 0;

           }
           break;


          // ------------ Caso o tipo de movimento a ser verificado seja adução e abdução do braço -------------
          case 'A':
          // Define o valor do torque e a direção da força peso do braço de acordo com o ângulo que o braço faz com a linha
          // do corpo no plano coronal
          if (AngOmbCor <= 90)
                        AngPeso =  AngOmbCor * M_PI / 180;
                else
                        AngPeso = M_PI - AngOmbCor * M_PI / 180;

                // Torque exercido pela força Supre Espinhal
                TorqueA = SliderSupE->value() * atof(outRSE->value());

                // Torque exercido pela força  + força Peso
                TorqueB = SliderDor->value() * atof(outRGD->value()) + ObjOU.Peso * 9.8 * (ObjOU.Comprimento/2) * sin(AngPeso)
                          + (ObjOR.Peso + ObjOM.Peso)* 9.8 * (ObjOU.Comprimento + ObjOR.Comprimento/2) * sin(AngPeso);

                // Define a direção de rotação do Antebraço
                if(TorqueA > TorqueB)
                        AngMov = 0.5;
                else if (TorqueA < TorqueB)
                                AngMov = -0.5;
                        else AngMov = 0;

            break;
          }

          return AngMov;
   }
```


Código D.0.4: continuação...

Apêndice E

Método *Handle*

```
1   // ********************************************************************
2   // void Ogl::handle(int event)
3   //    Função que verifica os eventos do mouse
4   // ********************************************************************
5   int Ogl::handle(int event){
6           double xGL1, yGL1, xGL2, yGL2, xaux, yaux, teste, dx, dy;
7           int transX, transY, transZ;
8           SmVR_CPoint pt_user;
9           SmVR_CPoint pt_userAux;
10          SmVR_CPoint Zero(0,0,0);
11          int ang;
12
13          xaux = yaux = 0;
14
15          // Capta o evento
16          switch (event){
17
18          // Caso o evento seja o clique do botão do mouse, guarda nas varíaves mxaux e myaus os valores
19          // de x e y referentes à posição que o mouse na janela do ambiente virtual se encontrava
20          // exatamente na hora que o botão foi pressionado.
21      case FL_PUSH:
22
23                  if (Fl::event_button() == FL_LEFT_MOUSE){
24                          mx = Fl::event_x();
25                          my = Fl::event_y();
26                          mxaux = Fl::event_x();
27                          myaux = Fl::event_y();
28  //                      teste = Fl::event_y();
29
30                          // os valores captados são então convertidos para os valores dos eixos x e y do ambiente 3D
31                          xaux = 0.2 * mx - 50;
32                          yaux = -0.23 * my + 50;
33                          xGL1 = xaux;
34                          yGL1 = yaux;
35
36          }
37          return 1;
38
39          // Caso o evento seja o arrastar do mouse, essa opção vefica a direção do mouse e a quantidade de
40          // pixels em que que o mouse Foi arrastado na tela
41          case FL_DRAG:
42                  SmVR_CPoint Zero(0, 0, 0);
43                  SmVR_CPoint pt_aux, pt_user, pt_trans(0,0,0);
44
45                  if (Ogl::anime) Ogl::anime= false;
46                                  else Ogl::anime = true;
47
48                  // Essa função atribui a pt_aux a posição do objeto braço na cena
49                  Obj->Braco->GetPointInOCS(Zero,&pt_aux,Obj->Root);
50
51                  // se o botão do lado esquerdo do mouse estiver pressionado, as posições do mouse são atribuídas
52                  // as variáveis mx, my.
53                  if (Fl::event_button() == FL_LEFT_MOUSE){
54                          mx = Fl::event_x();
55                          my = Fl::event_y();
56
57                          // altera os valores de dx e dy, que representam a quantidade de pixels que o mouse foi
58                          // arrastado na tela equanto o botão esquerdo estava sendo pressionado.
59                          dx = mx - mxaux;
60                          dy = my - myaux;
61
62                          // Verifica se a opção de navegação selecionada é Zoom
63                          switch (Navega){
64                          case 0:                                 // caso seja Zoom
65                                  // atribui a pt_userAux a posição do observador
66                                  Obj->user->GetPointInOCS(Zero, &pt_userAux, Obj->Root);
67                                  transX = transY = transZ = 0;
68
69                                  // verifica qual é o plano de visão do observador e determina a quantidade de pixels que
70                                  // o observador vai ser movido a medida que o mouse é arrastado
71                                  if (pt_userAux.X > 0)
72                                          transX = 5;
73                                  else if (pt_userAux.Y > 0)
74                                                  transY = 5;
75                                  else transZ = 5;
76                                  pt_trans.X = pt_trans.Y = pt_trans.Z = 0;
77
78                                  // verifica o sentido e direção que o mouse está sendo arrastado
79                                  if (dy > 0){                    // amplia o tamanho do objeto
80                                          pt_trans.X = transX;
81                                          pt_trans.Y = transY;
82                                          pt_trans.Z = transZ;
83                                  }
84                                  else{                                   // reduz o tamanho do objeto
85                                          pt_trans.X = -1 * transX;
86                                          pt_trans.Y = -1 * transY;
87                                          pt_trans.Z = -1 * transZ;
88                                  }
89
90                                  // move o observador em relação ao alvo para aproximar ou afastar o observador do alvo,
91                                  // conforme o direção e sentido que o mouse foi arrastado
92                                  Obj->user->TranslateByOnOBJCS(pt_trans, Obj->alvo);
93                                  Ogl::cursor(FL_CURSOR_CROSS);
94                                  break;
95
```

Código E.0.5:

```
                        // Verifica se a opção de navegação selecionada é Rotacionar
                        case 1:
                                pt_trans.X = pt_trans.Y = pt_trans.Z = 0;

                                // verifica o sentido e direção que o mouse está sendo arrastado
                                if (abs(dy) > abs(dx)){
                                        pt_trans.X = 1;
                                        if (dy > 0)
                                                ang = -5;
                                        else ang = 5;
                                }
                                else {
                                        pt_trans.Y = 1;
                                        if (dx > 0)
                                                ang = -5;
                                        else ang = 5;
                                }

                                // rotaciona o observador conforme o direção e sentido que o mouse foi arrastado
                                Obj->user->RotateBy(ang,pt_trans);
                                // altera o tipo do cursor na tela conforme transformação selecionada
                                Ogl::cursor(FL_CURSOR_ARROW);

                                break;

                        // Verifica se a opção de navegação selecionada é Translação
                        case 2:
                                        pt_trans.X = pt_trans.Y = pt_trans.Z = 0;
                                        // verifica a direção em que o mouse está se movendo
                                        if (abs(dx) > abs(dy)){         // eixo x
                                                // verifica o sentido em que o mouse está se movendo
                                                if (dx > 0)                             // direita
                                                        pt_trans.X = 5;
                                                else pt_trans.X = -5;}          // esquerda
                                        else if (dy > 0)
                                                        pt_trans.Y = -5;
                                                else pt_trans.Y = 5;
                                                // Move o observador em relação ao alvo
                                                Obj->user->TranslateByOnOBJCS(pt_trans, Obj->alvo);
                                        // altera o tipo do cursor na tela conforme transformação selecionada
                                        Ogl::cursor(FL_CURSOR_MOVE);
                                        break;


                        // Verifica se a opção de navegação selecionada é Girar
                        case 3:
                                pt_trans.X = pt_trans.Y = pt_trans.Z = 0;
                                // verifica a direção em que o mouse está se movendo
                                if (abs(dy) > abs(dx)){         // eixo x
                                        pt_trans.X = 1;
                                        // verifica o sentido em que o mouse está se movendo
                                        if (dy > 0)                                             // direita
                                                ang = -15;
                                        else ang = 15;
                                }
                                else {
                                        pt_trans.Y = 1;
                                        if (dx > 0)
                                                ang = -15;
                                        else ang = 15;
                                }

                                // Gira o observador ao redor do alvo
                                Obj->user->RotateByOnOBJCS(ang, pt_trans, Obj->alvo);
                                // altera o tipo do cursor na tela conforme transformação selecionada
                                Ogl::cursor(FL_CURSOR_ARROW);

                                break;
                        }

                        Obj->LimpaBuffer();
                        Ogl::invalidate();
                        Ogl::redraw();
                        Fl::flush();
                }
                return 1;

        }

        // se a captação dos eventos for válida, visualiza as transformações geométricas na tela
        if (!valid()) show();
        return 0;
};
```

Código E.0.6: ... continuação

Apêndice F

ModeloAntebraco.txt

```
//Marlene Ferreira Marques
//marlene@uu.br
//Vertices do Objeto Antebraço

Meshes: 2
"Ulna0" 0 0
81
0 -736.583313 -458.556793 1617.073853
0 -733.321960 -467.033142 1623.822144
0 -722.382202 -465.114136 1634.739014
0 -711.704590 -452.484497 1668.478760
0 -701.558105 -443.449341 1718.527222
0 -702.853149 -425.846710 1697.392334
0 -722.518433 -447.834778 1605.625366
0 -695.857300 -438.342590 1759.847168
0 -699.213989 -425.114319 1758.510498
0 -682.677002 -370.964874 2055.479248
0 -683.719238 -397.286560 2131.348633
0 -666.981995 -363.895111 2148.690430
0 -678.589294 -392.838654 2137.969238
0 -669.044617 -390.262665 2195.347412
0 -657.175659 -399.633423 2204.091064
0 -656.295715 -404.311859 2200.450928
0 -670.153442 -392.929321 2159.761475
0 -706.926208 -453.373047 1606.460938
0 -705.968628 -461.292114 1607.246094
0 -720.795044 -476.247314 1620.285645
0 -716.387451 -472.409973 1611.519775
0 -723.531494 -475.716644 1616.251099
0 -716.574768 -463.535614 1606.006226
0 -666.287354 -374.594574 2052.169678
0 -660.749390 -361.774200 2136.550781
0 -644.166138 -356.966370 2184.095459
0 -621.085144 -389.321075 2147.728027
0 -625.034607 -382.282593 2155.801758
0 -613.749512 -395.953583 2141.184082
0 -616.075806 -395.114471 2149.340820
0 -620.934204 -393.380951 2152.245605
0 -612.328979 -402.676910 2141.062744
0 -623.606201 -384.132782 2158.266602
0 -621.932129 -378.130127 2179.245117
0 -627.576721 -386.881805 2199.132324
0 -628.291260 -365.470581 2178.058594
0 -656.370911 -393.383575 2205.004150
0 -636.667603 -404.908447 2199.397217
0 -648.438232 -402.901459 2204.331543
0 -645.538086 -407.677826 2199.397705
0 -658.895935 -418.260193 2118.453125
0 -660.774719 -424.308838 2124.719971
0 -686.619507 -403.820984 2107.394775
0 -682.553772 -404.130280 2056.094971
0 -638.152466 -401.932190 2160.587402
0 -638.051270 -412.925293 2150.845215
0 -640.672424 -397.656647 2179.066650
0 -629.469727 -415.672943 2140.996582
0 -715.236938 -440.632019 1595.314453
0 -733.465759 -461.202209 1611.707764
0 -721.696411 -455.493042 1648.762451
0 -718.085205 -442.482513 1658.459961
0 -714.214111 -435.341766 1626.364258
0 -715.932983 -450.334961 1602.011597
0 -692.254578 -412.587463 1954.626099
0 -689.675049 -404.154480 1810.276367
0 -689.798462 -387.523407 1915.295776
0 -685.110352 -389.868622 2087.330322
0 -671.122253 -364.170868 2099.482178
0 -676.494934 -397.090668 2141.094727
0 -703.017456 -435.540649 1645.503906
0 -699.134338 -447.116089 1627.034546
0 -715.697815 -470.465607 1623.699341
0 -710.457153 -470.984772 1613.322266
0 -724.067505 -475.522766 1621.834961
0 -678.322021 -425.729828 1744.428589
0 -679.083374 -433.497314 1746.646362
0 -680.421448 -405.831268 1811.387939
0 -681.115051 -417.935822 1755.634155
0 -703.365295 -452.167664 1671.789063
0 -667.338013 -407.711761 1872.139893
0 -663.557251 -397.787659 1975.393799
0 -635.721436 -358.806885 2174.196045
0 -641.793762 -393.075439 2098.794189
0 -628.288757 -401.697266 2121.603516
0 -626.003235 -389.243530 2163.753662
0 -627.720581 -394.437103 2198.450684
0 -636.747253 -397.622528 2203.937012
0 -615.337158 -420.560150 2136.395020
0 -643.781738 -435.234070 2147.093262
0 -632.986511 -395.724701 2181.375488
```

Código F.0.7: *ModeloAntebraco.txt*

```
81
-0.592394 -0.708533 0.383472
-0.808869 -0.324477 0.490352
-0.288495 -0.790542 0.540199
-0.981810 -0.055810 0.181477
-0.934997 0.263153 0.237762
-0.834772 -0.536004 -0.125922
-0.924941 -0.096557 0.367642
-0.983107 0.168252 0.072056
-0.455368 0.885247 -0.094750
-0.292818 0.949437 -0.113257
0.463859 0.860025 -0.212586
0.280559 0.490685 -0.824933
0.987025 -0.033594 -0.157013
0.578402 -0.147459 -0.802313
0.608703 -0.374014 -0.699711
-0.149896 -0.422836 -0.893723
0.151861 -0.699238 -0.698573
-0.125961 -0.343898 -0.930520
-0.706037 -0.681374 0.192979
0.657184 -0.752772 0.037995
-0.667724 -0.736595 0.107575
-0.836341 0.547969 -0.016251
0.502455 0.848623 -0.165461
0.412943 0.876858 -0.246165
0.685756 0.309429 -0.658781
0.912717 0.288842 -0.288994
0.722674 0.690124 -0.038342
0.811448 0.546346 -0.207505
0.791266 0.611471 0.001583
0.975881 0.217162 0.022292
0.693787 0.639745 -0.330735
0.579118 0.325596 0.747402
-0.157130 0.208646 0.965286
0.315101 -0.243244 0.917357
-0.035556 -0.462051 0.886140
0.815637 -0.566027 -0.119785
-0.557244 -0.830048 -0.022346
-0.479197 -0.874196 -0.078437
-0.652640 -0.483048 0.583717
-0.854137 -0.437601 0.280988
0.190423 -0.885699 0.423411
-0.030573 -0.993765 -0.107218
-0.856605 -0.453520 0.246062
-0.994058 0.090211 -0.060917
-0.638404 -0.259401 -0.724673
-0.371859 -0.841379 -0.392177
-0.191787 -0.877643 0.439273
-0.737501 -0.619699 0.268449
-0.338409 -0.915714 0.216674
-0.726098 0.177807 -0.664204
0.174595 0.973521 -0.147561
-0.742832 0.669391 0.010799
-0.920400 0.250142 0.300487
-0.759070 0.634594 0.145272
-0.105025 0.991273 -0.079678
-0.458484 -0.353495 0.815373
-0.475244 -0.810565 0.342238
-0.034640 0.877956 0.477486
0.598082 -0.795816 -0.094736
0.388375 -0.806738 0.445353
0.337807 -0.927563 0.159727
0.250613 -0.940114 0.231038
0.938569 0.307736 -0.156162
0.993084 0.102108 -0.057944
0.578873 0.786704 -0.214481
0.483957 0.857783 -0.173185
0.186425 -0.841275 -0.507447
0.498279 0.858541 -0.120934
0.879959 -0.293382 0.373631
0.721116 -0.276334 0.635320
0.795823 0.267670 0.543156
0.832923 -0.531261 0.154919
0.993286 0.020724 0.113817
0.835529 -0.464447 0.293564
0.550424 -0.777909 0.303135
0.035042 -0.998766 0.035201
0.787000 -0.612810 -0.071377
0.166180 -0.813817 0.556853
0.476760 -0.372057 0.796413
0.156987 -0.457325 0.875333
0.486167 -0.863744 -0.132622
```

Código F.0.8: *ModeloAntebraco.txt (continuação...)*

```
158
0 3 50 2 0 1 2 2
0 3 8 51 0 3 4 2
0 42 10 57 5 6 7 2
0 52 5 60 8 9 10 2
0 48 60 61 11 10 12 2
0 17 61 18 13 12 14 2
0 48 17 53 11 13 15 2
0 20 22 18 16 17 14 2
0 7 66 54 18 19 20 2
0 55 67 5 21 22 9 2
0 24 74 73 23 24 25 2
0 74 26 28 24 26 27 2
0 35 32 27 28 29 30 2
0 34 36 77 31 32 33 2
0 77 36 38 33 32 34 2
0 54 71 43 20 35 36 2
0 41 59 10 37 38 6 2
0 16 59 44 39 38 40 2
0 44 46 13 40 41 42 2
0 0 49 21 43 44 45 2
0 64 1 21 46 47 45 2
0 0 21 1 43 45 47 2
0 1 2 50 47 2 1 2
0 3 4 8 0 48 3 2
0 2 1 64 2 47 46 2
0 51 0 50 4 43 1 2
0 1 50 0 47 1 43 2
0 3 51 50 0 4 1 2
0 8 5 51 3 9 4 2
0 51 5 52 4 9 8 2
0 0 51 52 43 4 8 2
0 6 0 52 49 43 8 2
0 48 53 6 11 15 49 2
0 48 6 52 11 49 8 2
0 6 49 0 49 44 43 2
0 8 7 55 3 18 21 2
0 55 7 54 21 18 20 2
0 55 54 56 21 20 50 2
0 54 43 57 20 36 7 2
0 5 8 55 9 3 21 2
0 56 54 9 50 20 51 2
0 9 54 57 51 20 7 2
0 7 8 4 18 3 48 2
0 42 57 43 5 7 36 2
0 12 57 10 52 7 6 2
0 57 12 9 7 52 51 2
0 11 58 9 53 54 51 2
0 9 12 11 51 52 53 2
0 59 12 10 38 52 6 2
0 11 12 16 53 52 39 2
0 13 11 16 42 53 39 2
0 36 13 14 32 42 55 2
0 15 14 13 56 55 42 2
0 25 11 13 57 53 42 2
0 60 48 52 10 11 8 2
0 63 61 62 58 12 59 2
0 69 62 61 60 59 12 2
0 3 2 62 0 2 59 2
0 62 69 3 59 60 0 2
0 17 48 61 13 11 12 2
0 63 18 61 58 14 12 2
0 63 62 19 58 59 61 2
0 2 64 62 2 46 59 2
0 19 62 64 61 59 46 2
0 18 22 17 14 17 13 2
0 18 63 20 14 58 16 2
0 20 63 21 16 58 45 2
0 63 19 21 58 61 45 2
0 53 17 22 15 13 17 2
0 66 61 65 19 12 62 2
0 70 66 65 63 19 62 2
0 7 4 66 18 48 19 2
0 60 5 68 10 9 64 2
0 68 5 67 64 9 22 2
0 61 68 65 12 64 62 2
0 68 67 65 64 22 62 2
0 65 67 70 62 22 63 2
0 68 61 60 64 12 10 2
0 61 66 69 12 19 60 2
0 66 4 69 19 48 60 2
0 69 4 3 60 48 0 2
0 56 70 67 50 63 22 2
0 70 56 71 63 50 35 2
0 71 56 73 35 50 25 2
0 55 56 67 21 50 22 2
0 73 56 23 25 50 65 2
0 73 23 24 25 65 23 2
0 23 56 9 65 50 51 2
0 70 71 54 63 35 20 2
0 40 43 71 66 36 35 2
0 66 70 54 19 63 20 2
0 71 73 40 35 25 66 2
0 25 72 24 57 67 23 2
0 35 72 25 28 67 57 2
0 25 34 35 57 31 28 2
0 26 74 27 26 24 30 2
0 24 27 74 23 30 24 2
0 35 27 24 28 30 23 2
0 31 74 28 68 24 27 2
0 30 29 26 69 70 26 2
0 28 26 29 27 26 70 2
```


Código F.0.9: *ModeloAntebraco.txt (continuação...)*

```
0 31 28 29 68 27 70 2
0 32 26 27 29 26 30 2
0 75 30 32 71 69 29 2
0 32 30 26 29 69 26 2
0 33 75 32 72 71 29 2
0 33 32 35 72 29 28 2
0 76 75 33 73 71 72 2
0 33 34 76 72 31 73 2
0 33 35 34 72 28 31 2
0 35 24 72 28 23 67 2
0 36 34 25 32 31 57 2
0 77 76 34 33 73 31 2
0 37 76 77 74 73 33 2
0 37 77 39 74 33 75 2
0 38 39 77 34 75 33 2
0 14 38 36 55 34 32 2
0 39 38 15 75 34 56 2
0 14 15 38 55 56 34 2
0 25 13 36 57 42 32 2
0 40 73 74 66 25 24 2
0 78 74 31 76 24 68 2
0 79 40 78 77 66 76 2
0 74 78 40 24 76 66 2
0 79 41 40 77 37 66 2
0 42 43 40 5 36 66 2
0 24 58 11 23 54 53 2
0 24 11 25 23 53 57 2
0 23 9 58 65 51 54 2
0 23 58 24 65 54 23 2
0 21 19 64 45 61 46 2
0 21 22 20 45 17 16 2
0 49 22 21 44 17 45 2
0 6 22 49 49 17 44 2
0 6 53 22 49 15 17 2
0 47 44 45 78 40 79 2
0 47 45 79 78 79 77 2
0 44 80 46 40 80 41 2
0 80 39 46 80 75 41 2
0 37 39 80 74 75 80 2
0 80 44 75 80 40 71 2
0 80 75 76 80 71 73 2
0 76 37 80 73 74 80 2
0 44 47 30 40 78 69 2
0 30 75 44 69 71 40 2
0 31 30 47 68 69 78 2
0 47 79 78 78 77 76 2
0 47 78 31 78 76 68 2
0 31 29 30 68 70 69 2
0 59 41 79 38 37 77 2
0 41 42 40 37 5 66 2
0 41 10 42 37 6 5 2
0 44 59 45 40 38 79 2
0 13 46 39 42 41 75 2
0 39 15 13 75 56 42 2
0 59 79 45 38 77 79 2
0 13 16 44 42 39 40 2
0 59 16 12 38 39 52 2
```

Código F.0.10: *ModeloAntebraco.txt (continuação...)*

```
"Radius0" 0 2
89
0 -690.255798 -481.802887 1596.010620
0 -684.761780 -489.721802 1595.523315
0 -684.291382 -498.595581 1593.781616
0 -709.038208 -540.382324 1579.931152
0 -717.355774 -540.474243 1575.875366
0 -688.133484 -481.204712 1604.482056
0 -683.994019 -489.528229 1601.485229
0 -683.332458 -500.440399 1599.206787
0 -700.734375 -541.539795 1596.492676
0 -714.984680 -544.432129 1578.201416
0 -713.234741 -547.106445 1583.844360
0 -703.830933 -544.314148 1603.087158
0 -706.239258 -487.574493 1627.714233
0 -710.616699 -536.094604 1629.071533
0 -721.241821 -482.393646 1624.132446
0 -703.720886 -507.811493 1642.606689
0 -715.639771 -484.676392 1673.577881
0 -708.845886 -488.046661 1691.235352
0 -707.472107 -511.824280 1695.039185
0 -712.838196 -473.403076 1787.414551
0 -713.813721 -521.120178 1679.650513
0 -694.029175 -468.613342 1957.222046
0 -683.448303 -410.009979 2067.284424
0 -692.322388 -409.254608 2068.164307
0 -679.164490 -412.621094 2075.016846
0 -693.625916 -408.019989 2087.434814
0 -672.580505 -423.964722 2058.002441
0 -678.085022 -429.892334 2089.755859
0 -685.312805 -433.002625 2089.765137
0 -683.842590 -403.198975 2104.271240
0 -677.214600 -406.723541 2104.321289
0 -680.965576 -399.306091 2110.087646
0 -674.313965 -403.216003 2109.168701
0 -679.042053 -395.383209 2130.357666
0 -665.767578 -412.222748 2132.102295
0 -683.127563 -430.351440 2110.457764
0 -670.432251 -431.152771 2121.224609
0 -678.835266 -437.577454 2123.220703
0 -720.978027 -545.134277 1583.534546
0 -721.940186 -541.999023 1579.740234
0 -739.782227 -507.917114 1603.119629
0 -739.481445 -503.322876 1601.155151
0 -726.271790 -518.748901 1582.233398
0 -724.573364 -491.734283 1582.731445
0 -724.050720 -478.307312 1589.137085
0 -730.304260 -499.836365 1710.027344
0 -726.596252 -490.992065 1812.381592
0 -702.112671 -468.304565 1957.847778
0 -692.395020 -430.207855 2090.701172
0 -695.204468 -430.491882 2111.722900
0 -704.784729 -437.424835 2041.041748
0 -702.255737 -423.218842 2059.730713
0 -700.533752 -426.463501 2110.503906
0 -703.840027 -413.037964 2108.579102
0 -704.560852 -401.316223 2119.040039
0 -708.192444 -430.857971 2129.677246
0 -695.265808 -439.668274 2124.285156
0 -693.000061 -436.311584 2136.292969
0 -705.655640 -429.420227 2135.514893
0 -704.227051 -399.357849 2126.449219
0 -708.536011 -404.697418 2133.535645
0 -709.606201 -420.561066 2135.284180
0 -697.147156 -395.960999 2125.457764
0 -696.075928 -396.107056 2132.376221
0 -704.902466 -406.169861 2135.889160
0 -682.137024 -417.330048 2131.948730
0 -696.708496 -529.588257 1589.775635
0 -696.315369 -490.850830 1614.342285
0 -704.703003 -482.291809 1617.072021
0 -704.702026 -496.632202 1700.190430
0 -710.857239 -510.455170 1736.082520
0 -702.720215 -496.011047 1802.953613
0 -694.474548 -431.151550 1986.289063
0 -692.361816 -457.739868 1990.346436
0 -697.926331 -423.365570 2023.031372
0 -666.140625 -416.733521 2115.815918
0 -673.873413 -417.507233 2107.102295
0 -723.238403 -541.188721 1594.653564
0 -739.269653 -506.876007 1608.884888
0 -730.088379 -502.718781 1584.319946
0 -721.273438 -514.917969 1705.070068
0 -726.454468 -506.171967 1735.173828
0 -719.088318 -499.275116 1813.961914
0 -713.551819 -486.641174 1885.465820
0 -718.952087 -466.613159 1894.432739
0 -697.236267 -439.500977 2057.317383
0 -712.351685 -417.996399 2121.387207
0 -694.630554 -398.594330 2115.273193
0 -671.458008 -427.343536 2135.600098
```

Código F.0.11: *ModeloAntebraco.txt (continuação...)*

```
89
0.529472 -0.067905 -0.845605
0.725933 0.277194 -0.629432
0.782133 -0.028297 -0.622468
0.966735 -0.202971 0.155647
0.793400 -0.584839 -0.168760
0.680198 -0.647865 0.342931
0.941189 0.269522 0.203768
0.813215 0.019004 0.581653
0.737629 0.675055 -0.014298
0.292296 0.948626 -0.121125
0.983993 0.178155 0.004395
0.875918 -0.480310 0.045493
0.471667 0.833653 -0.287322
0.632748 -0.763363 0.130026
0.313418 -0.920427 0.233631
0.993843 0.073904 -0.082553
0.647508 0.555167 -0.522037
0.905632 -0.147640 -0.397535
0.968123 -0.131501 -0.213180
-0.693558 -0.713573 0.098944
-0.452336 0.022983 -0.891551
-0.753051 -0.445592 -0.484109
0.489513 -0.826827 0.277011
0.174816 -0.957718 0.228508
0.338599 -0.929898 0.143666
-0.431838 -0.887178 0.162577
-0.828220 -0.551024 0.102096
-0.924566 -0.361878 0.119255
-0.918005 0.395520 0.028832
-0.440498 -0.866794 0.233731
-0.286888 -0.943318 0.166875
-0.921471 0.388377 -0.007387
-0.600747 0.778945 -0.179855
-0.853858 0.328463 -0.403781
-0.708867 -0.629470 -0.318237
-0.971810 -0.033187 -0.233417
-0.780837 -0.085316 0.618882
-0.764613 -0.637886 0.092024
-0.721748 0.599122 -0.346602
-0.791191 0.488096 0.368483
-0.540072 0.839433 -0.060621
0.495558 0.548140 -0.673769
-0.057717 -0.160364 -0.985369
0.620942 -0.243925 -0.744937
0.582821 0.746887 0.320125
0.133361 -0.986070 -0.099402
0.009383 -0.736253 -0.676642
0.465311 0.772063 0.432902
0.702387 0.694696 0.155081
0.983800 0.042573 0.174143
-0.488290 0.864594 0.118533
-0.307112 0.908723 -0.282674
-0.000945 0.999999 -0.000858
0.860711 -0.500667 0.092245
0.297366 0.941830 -0.156620
-0.267336 0.961513 -0.063429
-0.227093 0.939429 -0.256712
0.565087 0.822802 -0.060607
0.701432 -0.702309 0.121475
-0.007018 -0.987814 0.155482
-0.270146 0.891899 -0.362681
0.213422 0.815461 -0.538028
0.167934 0.898695 -0.405148
0.653676 0.628772 -0.421133
0.267714 0.925234 0.268835
0.759956 0.327604 0.561376
0.462982 -0.832414 -0.304524
0.744963 -0.616853 -0.254012
0.300002 -0.945086 -0.129656
0.640037 -0.379630 0.668007
-0.506094 -0.840830 -0.192026
-0.947239 -0.213247 -0.239300
-0.985444 -0.108619 0.130778
-0.929860 0.344367 -0.129506
-0.836571 0.202145 -0.509201
-0.068745 0.189693 -0.979434
-0.231634 -0.929588 0.286728
-0.535940 -0.785252 0.310077
-0.945404 -0.271466 0.180324
-0.575152 -0.815040 0.070074
-0.241620 -0.924729 -0.294102
-0.996925 0.076926 0.014930
-0.298942 -0.927696 -0.223638
-0.063546 -0.574465 0.816059
-0.424023 -0.347782 0.836213
-0.197186 0.722646 0.662495
-0.227386 0.966500 -0.119052
-0.089293 0.153216 0.984150
0.061038 0.149485 0.986878
```

Código F.0.12: *ModeloAntebraco.txt (continuação...)*

```
174
0 66 1 2 0 1 2 2
0 2 1 7 2 1 3 2
0 8 7 11 4 3 5 2
0 6 67 7 6 7 3 2
0 17 19 69 8 9 10 2
0 71 72 21 11 12 13 2
0 73 21 26 14 13 15 2
0 30 75 76 16 17 18 2
0 77 42 39 19 20 21 2
0 13 20 77 22 23 19 2
0 20 70 80 23 24 25 2
0 81 46 45 26 27 28 2
0 71 83 82 11 29 30 2
0 84 74 72 31 32 12 2
0 53 52 86 33 34 35 2
0 61 86 55 36 35 37 2
0 54 60 59 38 39 40 2
0 0 1 66 41 1 0 2
0 4 42 3 42 20 43 2
0 42 66 3 20 0 43 2
0 1 0 6 1 41 6 2
0 6 0 5 6 41 44 2
0 6 7 1 6 3 1 2
0 66 2 7 0 2 3 2
0 8 66 7 4 0 3 2
0 3 66 8 43 0 4 2
0 10 9 3 45 46 43 2
0 3 9 4 43 46 42 2
0 3 8 10 43 4 45 2
0 10 8 11 45 4 5 2
0 5 67 6 44 7 6 2
0 11 7 67 5 3 7 2
0 68 12 15 47 48 49 2
0 68 15 67 47 49 7 2
0 5 68 67 44 47 7 2
0 12 68 14 48 47 50 2
0 68 44 14 47 51 50 2
0 13 67 15 22 7 49 2
0 67 13 11 7 22 5 2
0 16 12 14 52 48 50 2
0 17 12 16 8 48 52 2
0 69 12 17 10 48 8 2
0 69 15 12 10 49 48 2
0 69 18 15 10 53 49 2
0 13 15 18 22 49 53 2
0 16 19 17 52 9 8 2
0 71 69 19 11 10 9 2
0 70 18 71 24 53 11 2
0 69 71 18 10 11 53 2
0 70 20 18 24 23 53 2
0 18 20 13 53 23 22 2
0 19 72 71 9 12 11 2
0 74 22 72 32 54 12 2
0 23 22 74 55 54 32 2
0 25 24 23 56 57 55 2
0 22 23 24 54 55 57 2
0 26 72 22 15 12 54 2
0 24 26 22 57 15 54 2
0 26 21 72 15 13 12 2
0 76 26 24 18 15 57 2
0 73 26 27 14 15 58 2
0 76 27 26 18 58 15 2
0 27 28 73 58 59 14 2
0 25 87 29 56 60 61 2
0 24 25 30 57 56 16 2
0 29 30 25 61 16 56 2
0 31 32 29 62 63 61 2
0 32 30 29 63 16 61 2
0 31 33 32 62 64 63 2
0 33 34 32 64 65 63 2
0 34 75 32 65 17 63 2
0 30 76 24 16 18 57 2
0 32 75 30 63 17 16 2
0 35 27 76 66 58 18 2
0 35 28 27 66 59 58 2
0 76 75 35 18 17 66 2
0 36 35 75 67 66 17 2
0 36 37 35 67 68 66 2
0 88 36 75 69 67 17 2
0 34 88 75 65 69 17 2
0 88 37 36 69 68 67 2
0 10 11 13 45 5 22 2
0 10 13 77 45 22 19 2
0 77 38 10 19 70 45 2
0 39 9 38 21 46 70 2
0 10 38 9 45 70 46 2
0 38 77 39 70 19 21 2
0 40 42 77 71 20 19 2
0 39 42 4 21 20 42 2
0 9 39 4 46 21 42 2
0 77 78 40 19 72 71 2
0 41 78 14 73 72 50 2
0 40 78 41 71 72 73 2
0 79 40 41 74 71 73 2
0 42 40 79 20 71 74 2
0 14 79 41 50 74 73 2
```

Código F.0.13: *ModeloAntebraco.txt (continuação...)*

```
0 43 79 44 75 74 51 2
0 43 44 0 75 51 41 2
0 80 77 20 25 19 23 2
0 77 80 78 19 25 72 2
0 45 14 78 28 50 72 2
0 16 14 45 52 50 28 2
0 14 44 79 50 51 74 2
0 82 80 70 30 25 24 2
0 70 71 82 24 11 30 2
0 80 81 78 25 26 72 2
0 80 82 81 25 30 26 2
0 45 78 81 28 72 26 2
0 19 16 45 9 52 28 2
0 46 81 82 27 26 30 2
0 84 45 46 31 28 27 2
0 71 21 83 11 13 29 2
0 47 83 21 76 29 13 2
0 85 47 73 77 76 14 2
0 47 21 73 76 13 14 2
0 83 47 50 29 76 78 2
0 47 85 50 76 77 78 2
0 83 46 82 29 27 30 2
0 84 46 83 31 27 29 2
0 50 84 83 78 31 29 2
0 19 45 84 9 28 31 2
0 19 84 72 9 31 12 2
0 73 28 85 14 59 77 2
0 48 85 28 79 77 59 2
0 49 28 35 80 59 66 2
0 48 28 49 79 59 80 2
0 51 50 85 81 78 77 2
0 52 51 48 34 81 79 2
0 85 48 51 77 79 81 2
0 84 50 74 31 78 32 2
0 50 51 74 78 81 32 2
0 25 23 51 56 55 81 2
0 53 51 52 33 81 34 2
0 52 48 49 34 79 80 2
0 52 49 56 34 80 82 2
0 49 35 37 80 66 68 2
0 52 56 55 34 82 37 2
0 86 52 55 35 34 37 2
0 54 53 86 38 33 35 2
0 56 49 37 82 80 68 2
0 37 57 56 68 83 82 2
0 57 37 88 83 68 69 2
0 55 56 57 37 82 83 2
0 57 58 55 83 84 37 2
0 59 60 63 40 39 85 2
0 86 60 54 35 39 38 2
0 60 86 61 39 35 36 2
0 55 58 61 37 84 36 2
0 43 66 42 75 0 20 2
0 66 43 0 0 75 41 2
0 0 44 5 41 51 44 2
0 68 5 44 47 44 51 2
0 23 74 51 55 32 81 2
0 25 51 53 56 81 33 2
0 53 87 25 33 60 56 2
0 33 31 87 64 62 60 2
0 33 87 62 64 60 86 2
0 31 29 87 62 61 60 2
0 63 33 62 85 64 86 2
0 62 59 63 86 40 85 2
0 87 59 62 60 40 86 2
0 87 54 59 60 38 40 2
0 87 53 54 60 33 38 2
0 43 42 79 75 20 74 2
0 63 60 64 85 39 87 2
0 60 61 64 39 36 87 2
0 57 64 58 83 87 84 2
0 64 61 58 87 36 84 2
0 64 57 65 87 83 88 2
0 65 34 33 88 65 64 2
0 65 88 34 88 69 65 2
0 57 88 65 83 69 88 2
0 33 63 65 64 85 88 2
0 63 64 65 85 87 88 2
```

Código F.0.14: *ModeloAntebraco.txt (continuação...)*

Apêndice G

Tutorial para uso da Biblioteca de Classes

G.1 Introdução

A biblioteca de classes foi desenvolvida com a finalidade de dar suporte à criação do modelo músculo esquelético do membro superior humano, bem como definir as restrições necessárias na simulação de movimentos.

Para uma melhor compreensão e usabilidade destas classes, este tutorial busca mostrar a forma como elas poderão ser usadas a partir do desenvolvimento de um projeto que permite aos usuários desenvolver uma aplicação de realidade virtual contendo um modelo músculo esquelético e a simulação de movimentos dos segmentos criados.

Este tutorial é baseado no minicurso de Realidade Virtual desenvolvido pelo Grupo de Realidade Virtual da PUC/RS e também utiliza o tutorial por ele proposto com algumas modificações para inserção do modelo músculo esquelético.

Pré-requisitos: programação em Linguagem C ++(necessário); conhecimentos de OpenGL e SmallVR.

G.2 Configuração do ambiente de desenvolvimento

Primeiramente, é necessário instalar o SmallVR conforme as instruções descritas no Minicurso de Realidade Virtual disponível no CD em anexo. Feito isso, substitua o programa Navegacao3.cpp copiado do minicurso pelo arquivo Simula.cpp encontrado no diretório Simula e insere no projeto SmallVR. Proceda da seguinte forma:

1.Abrir o Visual C++ 7.0,

2.Abrir o projeto SmallVR do tutorial do Minicurso de Realidade Virtual,

3.Na janela *Solution Explorer* clicar em *SmallVR,* situado na parte de cima da janela, clicar com o botão direito do mouse sobre *Test Files* e escolher a opção *Add Existing Item*,

4.Em seguida, adicionar o arquivo SimulaMov.cpp como mostrado na Figura G.1,

5.Excluir do projeto o arquivo Navegacao3.cpp.

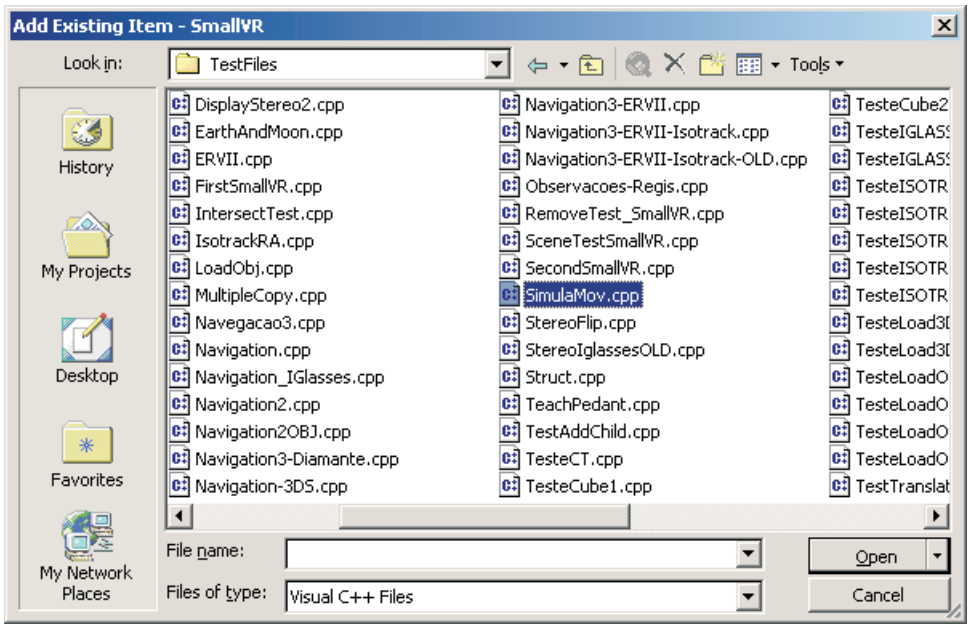

Figura G.1: Tela do sistema SIMM

Para iniciar o uso das classes propostas, deve-se incluí-las no projeto e então, no Simula.cpp acrescentar os includes:

$\sharp include < segmento.h >$

$\sharp include < musculo.h >$

$\sharp include < articulacao.h >$

$\sharp include < movimento.h >$

$\sharp include < DbList.h >$

G.3 Criando os objetos

No próximo passo, deve-se instanciar estas classes e criar uma lista para permitir o armazenamento dos atributos de cada objeto criado. A instância das classes é feita junta com a declaração das variáveis globais. Exemplo:

$Segmento * ObjS = newSegmento;$

$DbList < Segmento > listaSegmento;$

$Lista < Segmento > *pS;$

$Musculo *ObjM = newMusculo;$
$DbList < Musculo > listaMusculo;$
$Lista < Musculo > *pS;$

$Articulacao *ObjA = newArticulacao;$
$DbList < Articulacao > listaArticulacao;$
$Lista < Articulacao > *pS;$

Para inserir o objeto criado na lista correspondente, como exemplo o ObjS na listaSegmento, é necessário informar os valores de cada atributo, considerando o peso e as medidas antropomórficas correspondentes, conforme as tabelas em anexo. Para fazer isto, pode-se criar um procedimento para inserir os valores em cada atributo e chamá-lo no procedimento *CriaCenario* antes inserção dos objetos (Segmentos) na cena.

Exemplo do código para informação dos valores do objeto *Segmento:*

ObjS->XCoordProximal = 0;
ObjS->YCoordProximal = 0;
ObjS->ZCoordProximal = 0;

ObjM->NomeMusculo = ‘Biceps‘;
ObjM->SegmentoOrigem = ‘Braco‘;
ObjM->SegmentoDestino = ‘Antebraco‘;

ObjA->NomeArticulacao = ‘Cotovelo‘;
ObjA->TipoArticulacao = ‘Dobradica‘;
ObjA->SegmentoProximal = ‘Braco‘;

ObjA->SegmentoDistal = ‘Antebraco‘;

Exemplo da chamada do método *Insert* da classe *Lista,* para inserção do objeto na lista:

$listaSegmento.Insert(*ObjS)$;
$listaMusculo.Insert(*ObjM)$;
$listaArticulacao.Insert(*ObjA)$;

G.4 Inserindo os objetos na cena

Para inserir os objetos 3D na cena, é necessário calcular o tamanho de cada um para passar como parâmetro para a função. As funções que desenham os objetos já foram previamente criadas, sendo uma para cada objeto a ser criado como descrito a seguir:

Braço DesenhaObjBraco()

Antebraço DesenhaObjAntebraco()

Mão DesenhaObjMao()

A inserção dos objetos na cena pode ser feita em CriaCenário. A chamada destas funções é feita seguindo o exemplo :

$Braco = newSmVR_CGeometricObject("Braco", DesenhaObjBraco)$;
$Braco- > SetRenderFunctionData(tamOssoB)$;
$RootObject- > AddChild(Braco)$;

$Antebraco = newSmVR_CGeometricObject("Antebraco", DesenhaObjAntebraco)$;
$Antebraco- > SetRenderFunctionData(tamOssoA)$;
$RootObject- > AddChild(Antebraco)$;

$Mao = newSmVR_CGeometricObject("Braco", DesenhaObjMao);$

$Mao- > SetRenderFunctionData(tamOssoM);$

$RootObject- > AddChild(Mao);$

Obs.: As variáveis *tamOssoB, tamOssoA* e *tamOssoM* devem ser declarados como vetor de uma posição e devem variáveis globais para serem reconhecidas na chamada da função *SetRenderFunctionData.*

G.5 Posicionando os objetos na cena

Após inserir os objetos na cena, é importante posicioná-los e determinar os pontos de rotação de cada objeto, ou seja, cada segmento.

Entre os segmentos existem as articulação e, no modelo proposto, elas são representadas por estes pontos de rotação do tipo *SmVR_CPoint.* Com a finalidade de construir uma estrutura para o modelo músculo esquelético, é necessário determinar uma hierarquia entre os segmentos e os pontos de rotação da seguinte forma:

1. Criar um ponto que represente a coordenada proximal do braço, ou seja, ponto em torno do qual ocorrerá a rotação do braço, ex.:

   $PProximalB = newSmVR_CGeometricObject("PProximalB");$

2. Fazer com que o segmento Braco seja filho deste ponto,

   $RootObject- > AddChild(PProximalB);$

   $PProximalB- > AddChild(Braco);$

3. Criar um ponto represente a coordenada distal do braço, ou seja, ponto em torno do qual ocorrerá a rotação do antebraço, ex.:,

   $PDistalB = newSmVR_CGeometricObject("PDistalB");$

4.Fazer este ponto ser filho do segmento Braco e pai do segmento Antrebraco. Ex.:

$Braco- > AddChild(PDistalB);\ PDistalB- > AddChild(Antebraco);$

5.Para finalizar, faça a mão ser filha do objeto antebraço. Ex.:

$Antebraco- > AddChild(Mao);$

A criação desta hierarquia facilita a simulação de movimentos do braço seguindo a forma natural, pois, se o braço movimentar-se, o antebraço e a mão devem seguir o movimento. Se não for criado a hierarquia, sempre que for solicitado a movimentação do braço, deve-se também, movimentar o antebraço e a mão.

G.6 Analisando as restrições de movimento

Após a criação da hierarquia, deve-se determinar a forma com será simulada a movimentação dos segmentos. Pode-se fazer a simulação através do procedimento *Keyboard,* programado para responder às teclas pressionadas. Desta forma, por exemplo, cada vez que o usuário pressionar uma determinada tecla, pode-se programar a rotação de um segmento em uma quantidade de graus em um eixo determinado. Para isto, basta escolher a teclas e alterar seu código correspondente. como exemplo, uma linha de código que rotaciona o braco em cinco graus a cada vez que a tecla *a* for pressionada:

case ‘a‘: PProximalB->RotateBy(5,0,0,1);

Para cada movimento simulado, as restrições devem ser verificadas e os pontos distais e proximais dos segmentos devem ser atualizados. As restrições de tamanho dos músculos são analisadas através da chamada do método *ValidaTamanhoMusculo()* da classe *Musculo*. Para fazer a verificação, pode-se usar exemplo:

Declaração do objeto Músculo através da instância da classe Musculo:

Musculo ObjMB;

Determinação dos valores dos atributos:

ObjMB = BuscaMusculo(”Biceps”);

A função BuscaMusculo está descrita na figura G.2.

```
Musculo BuscaMusculo(char nome[30]){
  Musculo c;
  char nomeLista[30];
  bool existe = 0;
  bool ok;
  int j;
  int tamTexto;

  pM = listaMusculo.GetStart();
  while((pM) && (!existe)) {
    pM->GetInf(c);


    //nomeLista = c.NomeOsso;
      if (strlen(nome) > strlen(c.NomeMusculo))
            tamTexto = strlen(nome);
      else
            tamTexto = strlen(c.NomeMusculo);
      ok = true;
      j=0;
      while ((ok==true) && (j<tamTexto)){
            if (nome[j] == c.NomeMusculo[j]){
                        ok = true;
                  }
                  else{
                        ok = false;
                  }
                  j=j++;
            }
     if (ok)
             existe = 1;
       else
             existe = 0;
     pM = pM->next;
  }
  pM = listaMusculo.end;
  return c;
}
```

Figura G.2: Código da função BuscaMusculo.

Chamada do método de validação do tamanho do músculo:

ObjMB.ValidaTamanhoMusculo(TamanhoMusculo);

Obs.: O tamanho do músculo deve ser determinado a cada movimentação.

As restrições de amplitude de movimento são verificadas através da chamada *Va-*

lidaAmplitude() da classe *Movimento.* A verificação é feita do seguinte modo:

Declaração do objeto Movimento através da instância da classe Movimento:

Movimento ObjMov;

Determinação dos valores dos atributos:

ObjMov = BuscaMovimento(ObjMB.TipoMovimento);

A função BuscaMovimento está descrita na figura G.3.

```
Movimento BuscaMovimento(char nome[30]){
      Movimento c;
  char nomeLista[30];
  bool existe = 0;
  bool ok;
  int j;
  int tamTexto;

  pMov = listaMovimento.GetStart();
  while((pMov) && (!existe)) {
    pMov->GetInf(c);


    //nomeLista = c.NomeOsso;
      if (strlen(nome) > strlen(c.NomeMovimento))
            tamTexto = strlen(nome);
      else
            tamTexto = strlen(c.NomeMovimento);
      ok = true;
      j=0;
      while ((ok==true) && (j<tamTexto)){
            if (nome[j] ==  c.NomeMovimento[j]){
                        ok = true;
                  }
                  else{
                        ok = false;
                  }
                  j=j++;
            }
     if (ok)
             existe = 1;
       else
             existe = 0;
     pMov = pMov->next;
  }
  pMov = listaMovimento.end;
  return c;

}
```

Figura G.3: Código da função BuscaMovimento

.

Chamada do método de validação da amplitude do movimento:
ObjMov.ValidaAmplitude(Angulo);

Obj.: o valor do Angulo informado como parâmetro, deve ser o valor referente a posição final do movimento solicitado.

A atualização dos pontos distais e proximais na cena é feita por meio dos métodos *SetPosicaoDist()* e *SetPosicaoProx()*.