Rafael Souza da Costa

DEFORMAÇÃO DE BIOMEMBRANAS COM RETORNO DE FORÇA PARA TREINAMENTO MÉDICO EM REALIDADE VIRTUAL

Brasília
julho de 2013

UNIVERSIDADE DE BRASÍLIA
FACULDADE UNB PLANALTINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DE MATERIAIS

Rafael Souza da Costa

DEFORMAÇÃO DE BIOMEMBRANAS COM RETORNO DE FORÇA PARA TREINAMENTO MÉDICO EM REALIDADE VIRTUAL

Dissertação apresentada ao programa de Pós-Graduação em Ciência de Materiais da Universidade de Brasília, Faculdade UnB Planaltina, como requisito parcial para a obtenção do título de Mestre em Ciência de Materiais (Área de Concentração: Modelagem e Simulação).

Orientador:

Ivan Ferreira da Costa

Coorientador:

Bernhard Georg Enders Neto

Brasília
julho de 2013

RAFAEL SOUZA DA COSTA

DEFORMAÇÃO DE BIOMEMBRANAS COM RETORNO DE FORÇA PARA TREINAMENTO MÉDICO EM REALIDADE VIRTUAL

Dissertação apresentada ao programa de Pós-Graduação em Ciência de Materiais da Universidade de Brasília, Faculdade UnB Planaltina, como requisito parcial para a obtenção do título de Mestre em Ciência de Materiais (Área de Concentração: Modelagem e Simulação).

Aprovada em: 26 de julho de 2013.

BANCA EXAMINADORA

_____

Professor Doutor **Bernhard Georg Enders Neto** - Presidente da Banca
Universidade de Brasília – Faculdade UnB Planaltina – (FUP)

_____

Professor Doutor **Armando de Mendonça Maroja** – Membro Efetivo
Universidade de Brasília – Faculdade UnB Planaltina – (FUP)

_____

Professora Doutora **Lourdes Mattos Brasil** – Membro Externo ao Programa
Universidade de Brasília – Faculdade Gama (FGA)

AGRADECIMENTOS

Agradeço primeiramente a Deus pela saúde no decorrer desse processo e pela materialização de mais esse sonho na minha vida.

Meus sinceros agradecimentos ao meu orientador, Prof. Ivan Ferreira da Costa, por sua paciência durante meu período de aprendizado, por suas explicações, sugestões, conselhos, por sua ajuda durante a redação desta dissertação e também pela sua amizade.

Ao meu coorientador, Prof. Bernhard Georg Enders Neto, pelos momentos de descontração no Laboratório de Computação Científica, pelos conselhos na formatação dessa dissertação e pela indicação de diversos programas computacionais que me auxiliaram na inserção de fórmulas matemáticas ou na elaboração de gráficos.

Ao Prof. Remis Balaniuk pelos inúmeros conselhos e explicações em programação C++ e por possibilitar que eu participasse como aluno ouvinte em suas aulas no curso de Realidade Virtual.

A todos os meus professores do Campus UnB Planaltina, que me ofereceram a base de todo o conhecimento que possuo hoje, em especial aos professores Alex Fabiano Cortez Campos e Renata Aquino da Silva de Souza por se colocarem sempre dispostos a sanar qualquer dúvida e orientar todo e qualquer aluno do programa sempre que solicitado.

Aos técnicos administrativos, Jorivê Sardinha da Costa e Aristides Alvares Dourado Júnior, pelo excelente serviço prestado à comunidade acadêmica do Programa de Pós-Graduação em Ciência de Materiais e também por sempre se colocarem dispostos a sanar quaisquer dúvidas a respeito do curso.

A todos os amigos e familiares que me incentivaram e fortaleceram na árdua caminhada rumo à conclusão deste trabalho, em especial, a minha mãe Joselita Maria de Souza, por ter dado a base necessária e toda a humildade possível para que eu me tornasse o homem que sou hoje, ao meu irmão, Felipe Souza da Costa, por sua amizade e por sempre estar disposto a me ajudar no que for preciso, e por fim, à minha futura esposa, Ludmilla Patrícia Marra de Souza, por entender minhas angústias, por me fortalecer quando já esgotado, por ter suportado toda a

minha ausência nesse período e por ser essa parceira tão formidável, muito obrigado a todos por fazerem parte da minha vida.

Meu muito obrigado ao Programa de Pós-Graduação em Ciência de Materiais, à CAPES e ao REUNI pelas bolsas de estudos sem as quais eu ficaria impossibilitado de concluir o curso.

Por fim, agradeço às agências de fomento CNPq através dos projetos de pesquisa N 474831/2012-4, N 478736/2010-0 e Nº 473301/2007-5 e a FINEP pelo projeto Nº 0619/07, que possibilitaram a aquisição e utilização dos equipamentos e materiais necessários à concretização dessa dissertação.

"O sonho sem uma ação, é simplesmente um sonho. A ação, desprovida de um sonho, não leva a lugar nenhum. Mas o sonho aliado à ação, poderá mudar o mundo."

Fred Polak

RESUMO

Este trabalho trata da obtenção de solução analítica para a deformação de membranas com emprego da realidade virtual em tempo real. A simulação de membranas em ambientes virtuais fornece sensações visuais e táteis. Neste estudo, a plataforma CHAI 3D é responsável pelas sensações visuais que são transmitidas por meio do monitor de um computador e a interface de retorno de força, *PHANTOM Omni*®, aliada a essa plataforma, fornece aos usuários as sensações táteis. O método de deformação criado e implementado leva em consideração a *Equação de Poisson* e impõe a *Condição de Contorno de Dirichlet* que, por sua vez, possui solução analítica quando utilizamos a abordagem das *Funções de Green*. Duas situações são levadas em consideração nos cálculos, a primeira considera uma membrana homogênea no espaço livre e a segunda, considera uma membrana também homogênea, agora com borda fixa. A implementação das duas situações foi realizada utilizando-se o compilador e editor Microsoft® Visual Studio® 2008 em linguagem de programação C++. A segunda situação foi submetida a uma validação, a partir da comparação com o experimento de deformação de uma membrana de borracha homogênea real, a qual obteve-se êxito, uma vez que os gráficos comparativos entre o experimento real e a simulação possuem perfeita simetria e sobreposição.

Palavras-chave: Deformação de membranas. Realidade Virtual. Retorno de força.

ABSTRACT

This paper deals with obtaining the analytical solution for the deformation of membranes with use of virtual reality in real time. The simulation of membranes in virtual environments provides visual and tactile sensations. In this study, the platform CHAI 3D is responsible for visual sensations that are transmitted through a computer monitor and interface force feedback, *PHANTOM Omni*®, combined with that platform, provides users with tactile sensations. The method of deformation created and implemented taking into account the *Poisson Equation* and imposes the *Dirichlet Boundary Condition* which, in turn, has an analytical solution when we use the approach of *Green's Functions*. Two situations are taken into account in the calculations, the first considering a homogeneous membrane in free space and the second considers a membrane also homogeneous, now with fixed border. The implementation of the two conditions was performed using the compiler and publisher Microsoft® Visual Studio® 2008 programming language C++. The second case was submitted to a validation from the comparison with the experiment of deformation of a homogeneous real rubber membrane, which was obtained successfully, since the comparison charts between the actual experiment and simulation have perfect symmetry and overlap.

Keywords: Deformation membranes. Virtual reality. Force feedback.

RESUMEN

En este trabajo se ocupa de la obtención de la solución analítica para la deformación de las membranas con el uso de la realidad virtual en tiempo real. La simulación de membranas en entornos virtuales proporciona sensaciones visuales y táctiles. En este estudio, la plataforma CHAI 3D es responsable de sensaciones visuales que se transmiten a través de un monitor de ordenador y regeneración de la fuerza de la interfaz, *PHANTOM Omni*®, combinada con la plataforma, proporciona a los usuarios con sensaciones táctiles. El método de la deformación creada y aplicarse teniendo en cuenta la *Ecuación de Poisson* y impone la *Condición de Contorno de Dirichlet* que, a su vez, tiene una solución analítica cuando usamos el enfoque de las *Funciones de Green*. Dos situaciones se tienen en cuenta en los cálculos, teniendo en cuenta el primero de una membrana homogénea en el espacio libre y el segundo considera también una membrana homogénea, ahora con borde fijo. La aplicación de las dos condiciones se ha realizado mediante el compilador y editor de Microsoft® Visual Studio® 2008 lenguaje de programación C++. El segundo caso fue sometido a la validación de la comparación con el experimento de la deformación de una membrana de goma verdadera homogénea, que se obtuvo con éxito, ya que los gráficos de comparación entre la experiencia real y la simulación tienen una simetría perfecta y se solapan.

Palabras clave: Membranas deformación. Realidad virtual. Back fuerza.

SUMÁRIO

Lista de Símbolos

u	deformação da membrana no eixo vertical	53
τ	coeficiente de tensão superficial da membrana	53
dF	força aplicada em um ponto da membrana	56
P	pressão aplicada sobre a membrana	56
dA_0	área (dx, dy)	56
$\delta(x)$	delta de Dirac	56
g	função de Green	57
f	força referente à pressão sobre a tensão da membrana	57
V_0	volume inicial para o cálculo de uma integral parcial tripla	57
θ	ângulo de r com a horizontal	57
θ_0	ângulo de r_0 com a horizontal	57
r	módulo das coordenadas *x* e *y* da membrana plana, onde $r = \sqrt{x^2 + y^2}$	57
r_0	posição central da membrana circular	57
b	raio da borda fixa da membrana	58
u_0	posição da prova com relação ao eixo vertical	59
a	raio do disco da prova	59
C	uma constante qualquer	59
g_a	função de Green para ra	73
θ_a	ângulo de r_a com a horizontal (tetaa)	74
x_a	coordenadas do eixo x que formam o disco a	74
y_a	coordenadas do eixo y que formam o disco a	74
r_a	módulo das coordenadas xa e ya do disco a, onde $ra = \sqrt{xa^2 + ya^2}$	74

1 Introdução

Apesar de os sistemas em realidade virtual terem início nos Estados Unidos com finalidade na construção de simuladores de voo após a Segunda Guerra Mundial (MACHADO, 2003, p.12), atualmente, devido aos avanços de *hardware* e *software*, a utilização de ambientes virtuais para fins na área médica tem crescido em quantidade e qualidade.

A viabilidade de reproduzir situações reais sem risco a pacientes, a diminuição de custos devido à redução de uso de objetos físicos, a possibilidade de simular e visualizar ações impossíveis de serem percebidas no mundo real, como o funcionamento de um órgão humano ou a trajetória de um medicamento no corpo, a diminuição do uso de cadáveres, cobaias e materiais de manutenção constituem motivos suficientes para que a realidade virtual aplicada à saúde constitua uma área de interesse crescente, com benefícios tanto para a saúde, quanto para o desenvolvimento das áreas tecnológicas (NUNES; MACHADO; COSTA, 2009; PAVARINI, 2006; KIMER; CISCOUTTO, 2007).

A Realidade Virtual é uma ciência que engloba conhecimentos de diversas áreas, tais como computação, engenharias, ciências de materiais, cognição, dentre outras, que oferece sistemas computacionais que integram características de imersão e interatividade para simular ambientes reais onde os usuários têm estimulados simultaneamente os sentidos visuais e auditivos, como já é bem conhecido em nossos televisores e microcomputadores, mas também, outros sentidos podem ser explorados com novas tecnologias, como táteis, motores ou olfativos, através do uso de interfaces computacionais específicos de entrada e saída. Os dispositivos de entrada permitem o envio de informações de interação ou movimentação do usuário com o sistema, enquanto que os dispositivos de saída recebem as respostas do sistema que visam estimular os sentidos do usuário (MACHADO, 2003, p.9).

Em uma aplicação de simulação de cirurgia, por exemplo, erros da ordem de milímetros podem ser inadmissíveis. A necessidade de realismo exige que muitos sistemas, principalmente aqueles que simulam procedimentos, forneçam precisão em relação à modelagem e manipulação de objetos. E este requisito pode vir de encontro à necessidade de tempo real. Desta forma, tempo e precisão são dois fatores quase que conflitantes (KIMER; CISCOUTTO, 2007).

1.1 OBJETIVO PRINCIPAL

O trabalho em questão tem como objetivo principal desenvolver tecnologia para a simulação, visualização e interação tátil tridimensional da deformação de membranas em ambiente virtual.

1.2 OBJETIVOS ESPECÍFICOS

- Obtenção de sensações visuais através da plataforma gráfica CHAI 3D (CHAI 3D, 2013), que serão geradas por meio da utilização do monitor de um computador;
- Obtenção de sensações táteis por meio da manipulação do dispositivo *PHANTOM Omni*, produzido pela empresa SensAble Technologies (PHANTOM OMNI, 2013), que proporciona retorno de força em tempo real;
- Validação do método utilizado, onde será realizada uma comparação entre resultados experimentais encontrados na literatura e a simulação produzida com a implementação do método para a segunda situação.

1.3 JUSTIFICATIVA

A futura realização de procedimentos cirúrgicos não presenciais e o treinamento médico envolvendo órgãos e membranas como tímpano, mama, bexiga, fígado, olhos, veias e artérias, bem como a identificação de neoplasias benignas ou malignas justificam a importância da realização desse trabalho.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

No Capítulo 2 deste trabalho são apresentados exemplos da utilização do dispositivo de retorno de força *PHANTOM Omni*® na área da saúde e da educação, o que por sua vez abre uma grande possibilidade de temas de estudos voltados à utilização desse tipo de interface

háptica. Também é feita uma rápida descrição dos tipos de métodos utilizados na deformação de objetos virtuais, particularmente na deformação de biomateriais.

O Capítulo 3 tem como meta orientar o leitor a respeito da aquisição dos arquivos referentes ao CHAI 3D em um computador que possua um sistema operacional da família *Windows*®, a instalação do *software Microsoft*® *Visual Studio*® 2008 e a utilização de alguns de seus recursos, e por fim, a instalação da interface de retorno de força ou interface háptica *PHANTOM Omni*® e sua utilização na estação de trabalho.

No Capítulo 4 é apresentado o formalismo matemático utilizado na deformação de membranas homogêneas. A seção 4.2 aborda os cálculos para uma membrana que não possui uma borda definida, tendendo dessa forma a infinito, primeira situação. Na seção 4.3 são abordados os cálculos para uma membrana que possui borda definida, segunda situação, como o tímpano do ouvido humano.

No Capítulo 5 estão descritos os passos e os resultados alcançados com a modificação das linhas de códigos escritas em C++ de um dos projetos exemplo da plataforma CHAI 3D, bem como a validação do método desenvolvido na dissertação. Essa validação é feita por meio da realização de uma comparação de resultados tomados na literatura da deformação real de uma membrana de borracha homogênea com os resultados produzidos pela simulação com a segunda situação implementada.

Finalmente, no Capítulo 6, são apresentadas as conclusões finais da dissertação, destacando os resultados mais importantes obtidos no decorrer do trabalho realizado e uma proposta para trabalhos futuros.

2 Trabalhos Correlatos

Neste capítulo são apresentados alguns dos trabalhos que podem ser realizados com a utilização do dispositivo de retorno de força *PHANTOM Omni*® na área da saúde, o que por sua vez, abre uma grande possibilidade de temas de estudos voltados à utilização desse tipo de interface háptica. Também é feita uma rápida descrição dos tipos de métodos utilizados na deformação de objetos virtuais, particularmente na deformação de biomateriais.

2.1 ALGUMAS APLICAÇÕES DO *PHANTOM OMNI*®

Com o intuito de mostrar o quão abrangente pode ser a utilização desse dispositivo, essa seção se destina a mostrar algumas das aplicações e seu uso nas áreas da pesquisa, da educação, da saúde e da medicina.

O primeiro *software* simula um modelo 3D de uma boca que foi separada em duas camadas: gengiva e dentes. Onde uma escova de dente virtual pode ser controlada pela caneta do dispositivo háptico, que fornece ao usuário, sensações táteis de toque nos objetos virtuais, sendo possível sentir a maciez desses objetos exibidos no ambiente virtual, veja Figura 1. Além disso, outras propriedades específicas como atrito estático e atrito dinâmico são inseridas na simulação, fazendo com que o usuário perceba, por meio do toque virtual, a sensação de que está tocando em áreas diferentes da boca, além da visualização da mesma, por meio do monitor do computador. (RODRIGUES; MACHADO; VALENÇA, 2009).

Esse *software* é uma ótima opção para área da educação e saúde, podendo ser aplicado para várias faixas etárias, uma vez que pode ser utilizado como meio de demonstração da realização de procedimentos corretos na hora de fazer a higiene bucal.

Outra aplicação odontológica, agora com o objetivo centrado no treinamento de profissionais dessa área, é um *software* que exibe um maxilar, com o qual o usuário pode interagir por meio de instrumentos como um escavador dental ou uma broca, que são

controlados pela interface háptica, veja Figura 2. Dependendo-se da tarefa a ser realizada carrega-se o objeto escavador ou o objeto broca.

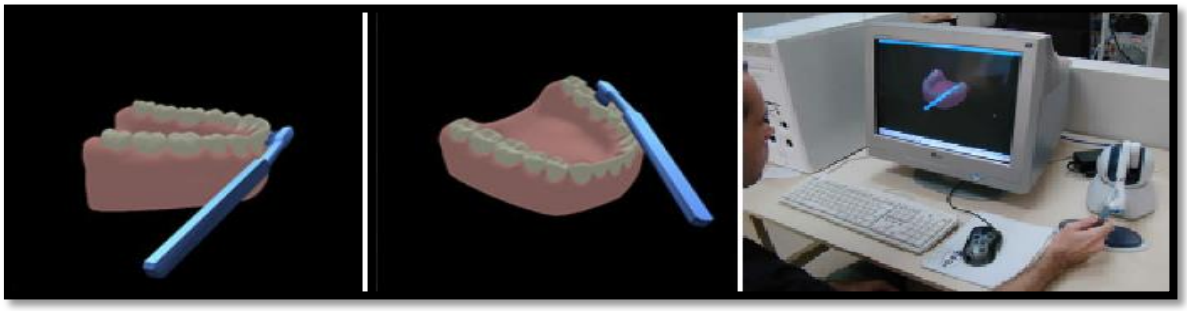

Figura 1 - *Software* que simula procedimento de uma escovação bucal.

Fonte: RODRIGUES; MACHADO; VALENÇA, 2009.

Com a utilização desses instrumentos virtuais pode-se realizar a retirada de partes da superfície de dentes em uma área específica ou então em várias áreas, apresentando variados níveis de desgaste. Nesse *software* parte da gengiva fica separada dos dentes, permitindo que a sensação de retorno de força ocorra de forma diferente em cada região do maxilar, sendo possível identificar as partes macias da gengiva e partes mais rígidas como os dentes **(**KÜLBERG; OLIVEIRA, 2012).

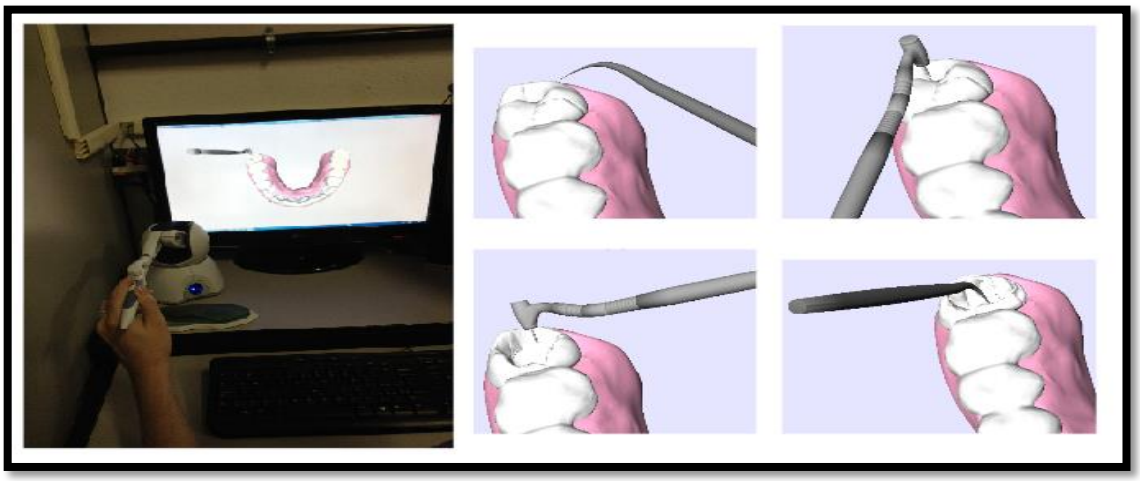

Figura 2 - *Software* para treinamento de profissionais na área odontológica.

Fonte: KÜLBERG; OLIVEIRA, 2012.

Moreira (2011) desenvolveu um simulador de corte craniano que utiliza o *Phantom* para gerar modificações na estrutura desse crânio, Figura 3. A imagem do crânio é gerada a

partir de tomografia computadorizada, que é posteriormente transformado em um arquivo de imagem 3D e carregado na simulação.

Por meio da manipulação da caneta do dispositivo háptico é possível tocar partes dos vértices que formam o crânio com um objeto na forma de bisturi, que realiza a fragmentação dessas estruturas gerando o corte. O autor ressalta em seu trabalho que o sistema apresentou limitações com relação ao tempo de resposta, sendo o simulador útil apenas para treinamento médico e o ensino de cirurgias.

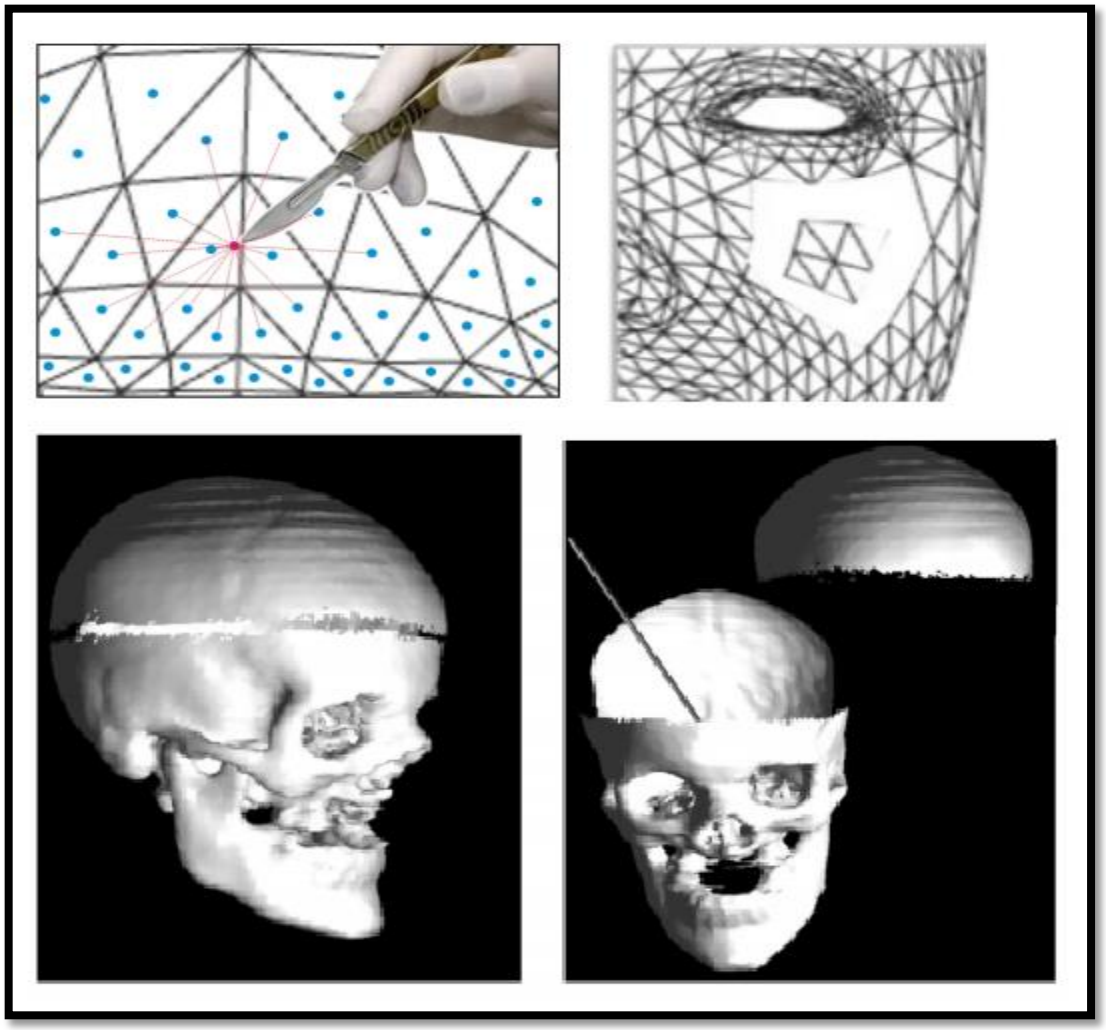

Figura 3 – Simulador para corte craniano.

Fonte: MOREIRA et al., 2011.

Outra aplicação desenvolvida com a utilização do *Phamtom omni*® e que pode ser usada no treinamento médico é mostrada em um trabalho preliminar desenvolvido por (COSTA, Rafael; COSTA, Ivan, 2012), onde foi criada uma simulação do tímpano do ouvido

humano, que na verdade é uma textura que colocada sobre uma membrana flexível pode ser deformada em tempo real. O usuário pode manipular a caneta do dispositivo, que por sua vez controla uma pequena esfera no ambiente virtual, a qual realiza as deformações do "tímpano" na cena, Figura 4.

Além do estímulo visual produzido pelo ambiente virtual, o usuário recebe ainda o estímulo tátil, que pode ser sentido com a manipulação do dispositivo háptico, que detecta o relevo na superfície por meio de colisões e deformações com retorno de força, o que faz o usuário ter a sensação de estar realmente deformando uma membrana flexível.

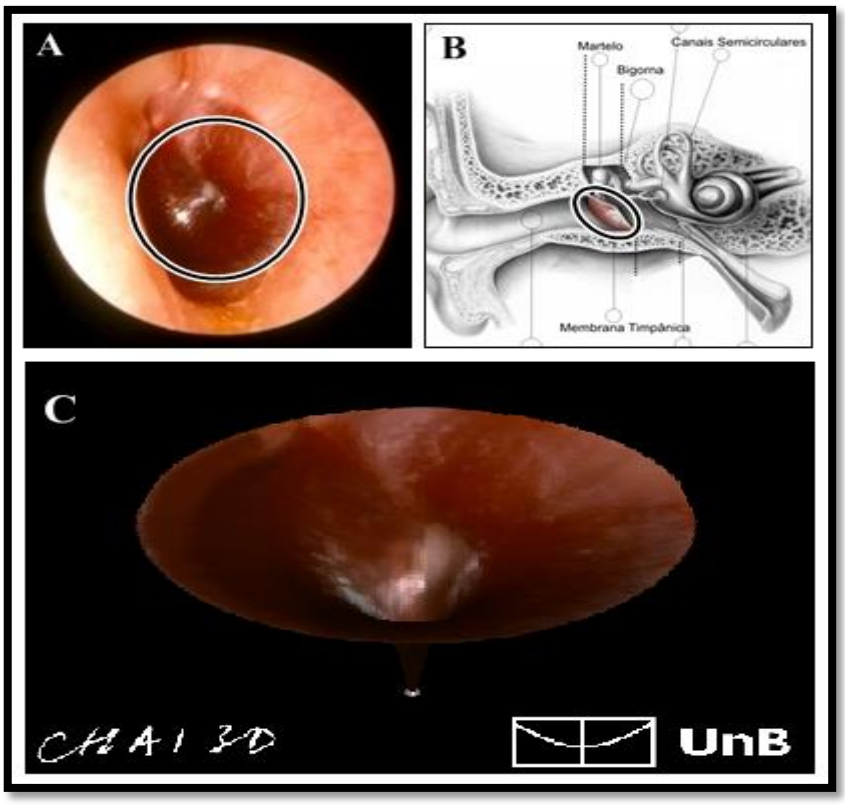

Figura 4 – Simulador de deformação de biomembranas.

Fonte: COSTA, R. S.; COSTA, I. F. Deformação de biomembranas com retorno de força para treinamento médico em realidade virtual. XXIII Congresso Brasileiro em Engenharia Biomédica - CBEB, 2012.

Como pode ser observado nos exemplos acima, o *Phamtom Omni*® possui diversas utilidades, sendo que essas possibilidades se concentram na sua maioria na área da educação, como com a realização de aulas práticas ou realização de procedimentos pré-cirúrgicos ou pré-operatórios. Note que a caneta do dispositivo pode ser utilizada como um corpo rígido (escova, broca, bisturi e esfera) que assume a forma de vários objetos e que possibilita a

interação do usuário com a simulação por meio desses objetos que são carregados para a cena virtual e que realizam modificações naquele ambiente.

Para entender melhor esses conceitos e procedimentos, veja na seção 2.2 como funcionam os principais métodos de deformação para objetos em realidade virtual.

2.2 MÉTODOS PARA DEFORMAÇÃO DE OBJETOS EM REALIDADE VIRTUAL

Segundo Pavarini (2006), em um ambiente virtual podem ser inseridos vários objetos e esses podem adquirir características de corpos estáticos (possuem forma fixa do início ao término de uma simulação) ou dinâmicos (possuem interação com outros objetos no decorrer da simulação, podendo modificar a sua forma, posição, cor ou textura).

Como no mundo real a grande maioria dos objetos possui comportamento dinâmico, ou seja, podem mudar de posição, cor e forma, os objetos dinâmicos costumam ser mais estudados devido às ações e reações que podem produzir em um ambiente virtual. Os objetos dinâmicos podem ter seus atributos modificados através de animações (processos pré-determinados) ou por meio da interação de um usuário que realiza essas modificações utilizando cálculos em tempo real. Dessa forma, as modificações da estrutura de um objeto dinâmico virtual podem gerar o que conhecemos como deformação.

A deformação de um objeto 3D pode ser implementada através de simulações geométricas (dados geométricos como vértices e pontos, que representam dobras e ondulações através de equações geométricas), físicas (baseia-se em leis da física para obter a forma do objeto sob condições envolvendo massa, aceleração e força física, seja ela interna ou externa) ou híbridas, que é a junção das técnicas geométricas e físicas. As técnicas mais conhecidas para realização de deformação de objetos em realidade virtual são: *Free Form Deformation* (FFD) ou Deformação de Livre Forma, *Finite Element Method* (FEM) ou Método dos Elementos Finitos e *Mass Spring* (MS) ou Massa Mola (PAVARINI, 2006, CAMPOS, MACHADO, 2008).

O método conhecido como *Deformação de Livre Forma*, ocorre pela modificação dos vários pontos e vértices que formam o objeto 3D carregado para a cena virtual, Figura 5. Como esse método não permite a manipulação dos objetos que compõem a cena, ele acaba

perdendo o seu realismo, principalmente quando empregado em treinamentos cirúrgicos (PAVARINI, 2006; CAMPOS, MACHADO,2008).

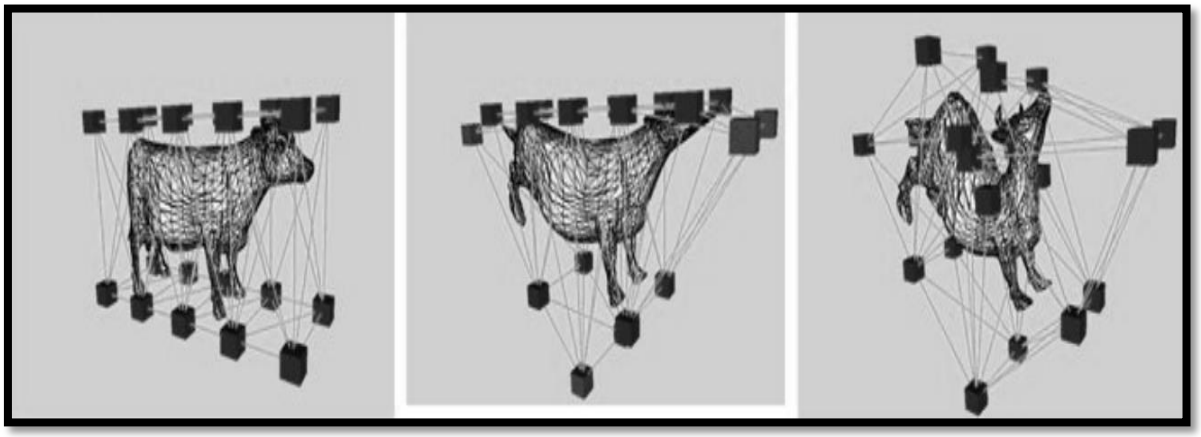

Figura 5 – Deformação de um objeto com o método de Deformação de Livre Forma.

Fonte: PAVARINI, 2006.

O *Método dos Elementos Finitos* pode ser entendido como um método matemático, no qual um meio contínuo é discretizado, ou seja, subdividido em elementos que mantém as propriedades do objeto que o originou. Na física esse método é amplamente utilizado na solução por aproximação de equações diferenciais parciais e também na solução de equações integrais (LOTTI, 2006; PAVARINI, 2006; CAMPOS, MACHADO, 2008). Dessa forma, a partir de elementos como linhas (1D), triângulos e quadrados (2D – elementos planos), tetraedros e hexaedro (3D - elementos com volume), é possível criar objetos formados pela repetição desses elementos básicos, Figura 6.

Como esse método emprega o uso de vetores que são utilizados para guardar os valores das massas e forças de cada objeto carregado, esse processo torna-se inviável para ser utilizado em simulação de deformação de objetos virtuais em tempo real, sendo necessária uma grande capacidade de processamento computacional (PAVARINI, 2006).

O terceiro método, também bastante utilizado e citado na literatura como método para deformação de objetos é o *Massa-Mola*. Esse método caracteriza-se pela modelagem de objetos deformáveis a partir de nós de massa conectados por molas. A Figura 7(a) mostra um objeto formado por pequenas esferas que representam as massas, e as molas são representadas pelas linhas em “zigue-zague”, que por sua vez podem conectar uma esfera a outras várias esferas. A Figura 7(b) mostra como é representado um único nó dos nós existentes em um

objeto formado por esse método. Como uma mola é uma estrutura que possui elasticidade, ao ser pressionada, será realizada a deformação daquela região em específico. Sendo assim, esse método simulação um sistema dinâmico governado pela Segunda Lei de Newton (PAVARINI, 2006; CAMPOS, MACHADO, 2008).

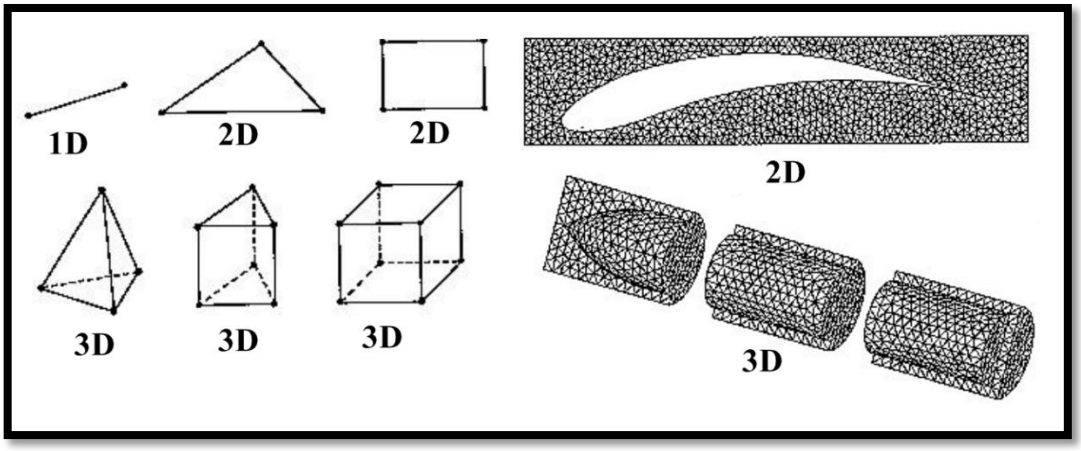

Figura 6 - Discretização de objetos contínuos a partir de elementos discretos básicos.

Fonte: PAVARINI, 2006.

Apesar de ser um método de fácil compreensão e implementação, e ainda possibilitar a simulação de deformações de objetos em tempo real, quando se deseja modelar ou criar propriedades muito complexas com uma grande quantidade de vértices, o processo torna-se novamente inviável, uma vez que a resposta do sistema leva um tempo considerável para realizar os cálculos envolvendo matrizes, o que acaba por diminuir o realismo da cena virtual (PAVARINI, 2006; CAMPOS, MACHADO, 2008).

Como constatado, os métodos usuais para deformação de objetos em realidade virtual da literatura possuem restrições quanto a capacidade de processamento de suas ações em ambientes virtuais.

De maneira resumida, os métodos convencionais necessitam de computadores com grande capacidade de processamento, maior intervalo de tempo para se chegar ao resultado final de uma deformação, melhor representação da realidade virtual simulada e maior realismo para serem empregados em aplicações que envolvam o treinamento médico. Uma maneira de contornar esses problemas é a utilização de solução analítica para realizar os cálculos da malha a ser deformada, obtendo-se assim uma solução bastante rápida do ponto de

vista computacional. O resultado final é a deformação de biomembranas em ambiente virtual ocorrendo em tempo real.

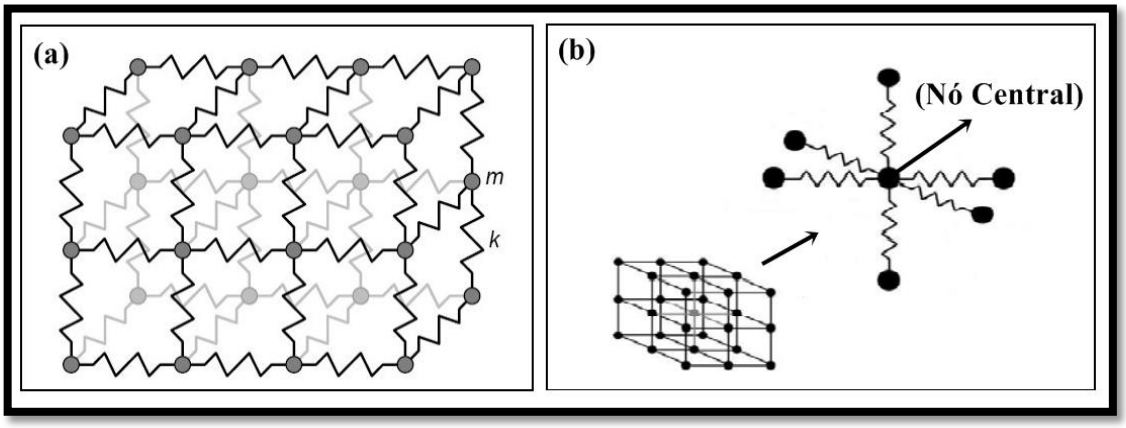

Figura 7 - Molas conectadas a um ponto de massa e que podem fornecer forças ao vizinho. (a) Objeto formado por massa de molas conectadas umas as outras. (b) Um dos muitos nós existentes em um objeto criado através do método massa-mola.

Fonte: PAVARINI, 2006.

Dessa forma, é proposto no capítulo 4 desse trabalho um método novo que tem como finalidade realizar a deformação em tempo real de materiais constituídos por membranas flexíveis, sem que para isso seja necessário solução de matrizes, o que acaba por deixar o resultado final de uma deformação um tanto demorado.

3 Plataforma de Simulação e Interface de Retorno de Força

Esse capítulo tem como meta orientar o leitor a respeito da aquisição dos arquivos referentes ao CHAI 3D, a sua utilização em um computador que possua o sistema operacional *Windows*®, e também, a instalação da interface de retorno de força e sua utilização na estação de trabalho.

Assim, esse capítulo tem um estilo bastante peculiar, como o de um tutorial. Note que esse capítulo é apresentado nesse formato com o intuito de servir de referência para usuários ou pesquisadores que necessitem instalar a interface háptica e *software* utilizados nessa dissertação. Isso será de grande importância, por exemplo, para trabalhos futuros em nosso laboratório, tendo em vista que foi nesse trabalho que pela primeira vez esses equipamentos e *software* foram instalados e utilizados. No entanto, se faz necessário esclarecer que a leitura pormenorizada de todos os detalhes contidos nesse capítulo só é de interesse para aqueles que efetivamente irão realizar tal instalação.

3.1 INTRODUÇÃO

O CHAI 3D é um código aberto que possui um conjunto de bibliotecas em linguagem de programação C++, que podem, por meio da renderização de um computador, criar visualizações e simulações interativas em tempo real. O CHAI 3D suporta vários dispositivos hoje comercializados, sendo que eles podem possuir três, seis ou até sete graus de liberdade táteis. O CHAI 3D é especialmente adequado para fins de educação e pesquisa, oferecendo uma plataforma com muitos recursos e que podem ser modificados à medida que for necessário.

Em particular, podemos elencar algumas características que serão bastante utilizadas nessa dissertação, tais como:

- Acesso Total ao Código Fonte;
- Multi-plataforma (*Windows*®, Mac®, *OS-X*, Linux®);
- Renderização de Gráficos Através da Biblioteca Open GL[1];
- Proxy-Modelo com Raio Variável;
- Modelos de Objetos Estáticos e Dinâmicos;
- Detecção de Colisão;
- Suporta Arquivos de Imagem em .BMP e .TGA;
- Suporta Arquivos de Malha em .OBJ e .3DS;
- Possui um Dispositivo Háptico Virtual;
- Suporta 06 Tipos de Dispositivos Hápticos, Inclusive o *PHANTOM Omni*® da SenSable Technologies.

Em suma, o CHAI 3D dá um passo importante em direção ao desenvolvimento e a criação de mundos virtuais multimodais, com a integração da tecnologia háptica que possibilitam o retorno de força ou *force feedback* e representações visuais de objetos estáticos e dinâmicos em 2D e 3D (CHAI 3D, 2013).

A aquisição dos arquivos para utilização da plataforma CHAI 3D poderá ser feita seguindo-se as orientações da seção 3.2 desse capítulo.

Entre as características descritas acima, cabe ressaltar o que são dispositivos hápticos físicos (*PHANTOM Omni*®, Figura 8) e dispositivos hápticos virtuais (aplicativo que simula o *PHANTOM Omni*®, Figura 12, janela 2). A palavra háptico refere-se à capacidade de sentir um ambiente mecânico, natural ou sintético por meio do toque. Háptico também inclui sinestesia (junção ou união de planos sensoriais diferentes: por exemplo, a visão com o tato), a capacidade de perceber a posição do corpo de uma pessoa, o movimento e o peso. É comum falar do canal tátil para designar coletivamente os componentes sensoriais e motores. Isto é, certas partes do corpo humano (em particular a mão), são órgãos unitários que ajudam a perceber o mundo e agir sobre ele, sendo dessa forma, atividades que ocorrem em conjunto. Por exemplo, para agarrar um objeto desconhecido é necessário que identifiquemos

[1] O *OpenGL*® (*Library Open Graphics*) é uma API (*Application Programming Interface*) para *hardware* gráficos. A API consiste em um conjunto de várias centenas de procedimentos e funções que permitem que um programador crie programas de sombreamento, objetos e operações envolvidas na produção de imagens de alta qualidade gráfica, imagens coloridas e objetos tridimensionais (OPENGL, 2013).

ativamente as suas dimensões com as nossas mãos. Canais táteis e cinestésicos trabalham juntos para fornecer aos seres humanos os meios para que possamos perceber e agir em determinado ambiente (HAYWARD, 2004).

Apesar do CHAI 3D ser uma plataforma de código aberto, para que possamos criar alterações nos códigos fontes de cada projeto disponibilizado, devemos instalar o *software Microsoft® Visual Studio®* na estação de trabalho, para essa dissertação iremos utilizar apenas a versão 2008 desse *software*, por isso, *Microsoft® Visual Studio®* 2008 - (MSV2008). Sendo assim, as instruções contidas nesse documento servem apenas para sistemas operacionais pertencentes à plataforma *Windows®*, uma vez que o *software* MSV2008 funciona apenas nesse sistema operacional. O *software* não é gratuito, por isso, deve ser adquirida uma licensa para sua utilização.

Para utilizar um sistema operacional diferente do *Windows®*, visite a página do CHAI 3D no endereço eletrônico:<www.chai3d.org/documentation.html>, onde se encontra toda a documentação e procedimentos para utilização de um sistema operacional diferente, cabe ressaltar que todo o sítio encontra-se em língua inglesa.

A seção 3.3 auxilirá o leitor na instalação da interface de retorno de força e manipulação da cena virtual *PHANTOM Omni®* da SensAble Technologies no computador, sua calibragem e também a sua utilização.

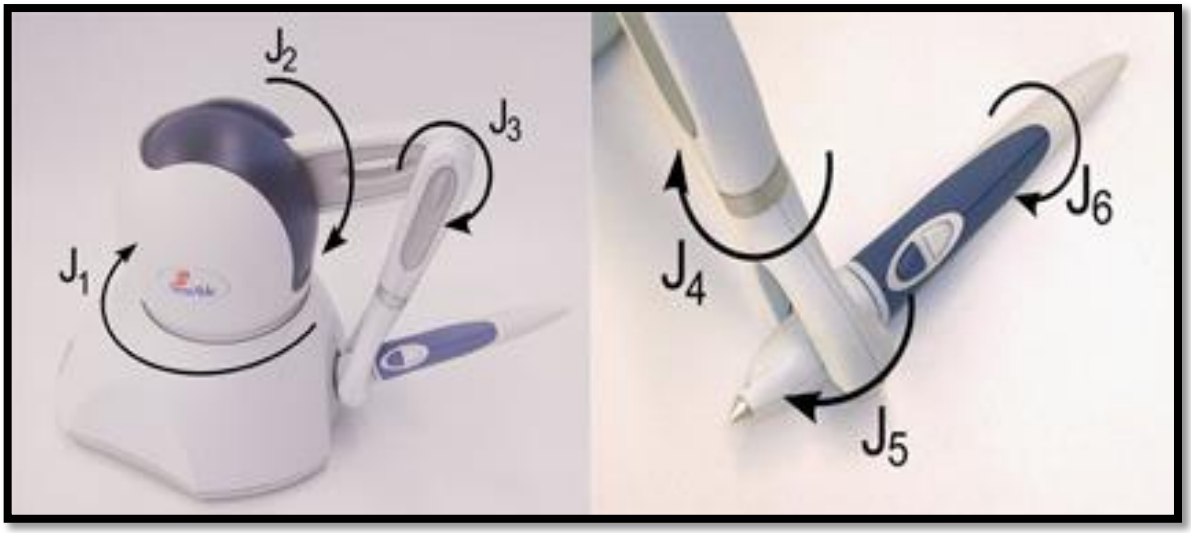

Figura 8 – PHANTOM Omni® e seus 6 graus de liberdade (dispositivo háptico físico).

Fonte: Disponível em: <www.quarcservice.com/ReleaseNotes/files/sensable_phantom_omni.html>. Acessado em: 23 jun. 2013.

Esse dispositivo possibilita movimentação em 6 graus de liberdade na simulação (movimentos de translação e rotação nos eixos *x,y,z*), Figura 8, sendo que as informações captadas são adquiridas através da leitura do posicionamento da caneta física do dispositivo.

Além disso, este tipo de dispositivo oferece um mecanismo de *force feedback* nos eixos, *x,y,z,* permitindo dessa forma, uma interação entre o usuário da simulação e a aplicação (SARAIVA, 2010; PHANTOM, 2013).

O *PHANTOM Omni*® é conectado ao computador por meio de uma entrada FireWire®, que é a única alternativa para se realizar essa comunicação, dessa forma, oferece comunicações de alta velocidade e serviços de dados em tempo real.

3.2 AQUISIÇÃO DOS ARQUIVOS DO CHAI 3D

1º Passo - Acesse o endereço eletrônico:< www.chai3d.org>;

2º Passo - Na página inicial clique no botão "DOWNLOAD" com o botão esquerdo do *mouse* no menu principal, opção destacada em vermelho na Figura 9;

Figura 9 – Página inicial do CHAI 3D. Botão download em destaque na barra de menu principal.

3º Passo: Baixe os pacotes de arquivos compatíveis com o seu sistema operacional, plataformas: *Windows*® 32 e 64 bits, *Linux*® 32 e 64 bits e *Mac*®, Figura 10. Se você está utilizando o sistema operacional *Windows*® e gostaria de checar qual tipo de plataforma seu computador está utilizando, consulte o APÊNDICE A e siga os procedimentos descritos**.**

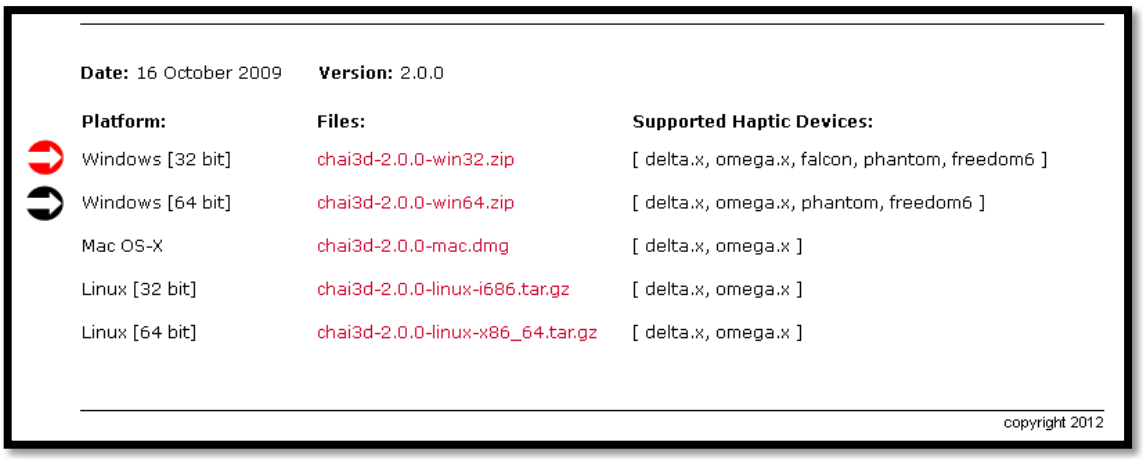

Figura 10 – Arquivos da aba *Download*. Setas indicando arquivos para plataforma 32 *bits* (seta vermelha) e 64 *bits* (seta preta) para o sistema operacional Windows®.

4º Passo: Nesse passo devemos descompactar o arquivo adquirido no 3º passo. Deixe o arquivo descompactado na mesma pasta onde está o arquivo compactado, posteriormente pode ser necessário para gerar um novo arquivo sem alterações. O procedimento para se descompactar um arquivo ou pasta pode ser acompanhado através do APÊNDICE B.

5º Passo: Checar qual deve ser a versão do *Microsoft® Visual Studio®* a ser instalada para utilização do CHAI 3D adequadamente. Na pasta "chai3d-2.0.0" que acabamos de descompactar no passo anterior existe uma pasta com o nome de "*projects*", e por sua vez, *projects* possui outras três pastas, chamadas respectivamente de: MSVC7, MSVC8, MSVC9. Cada uma das pastas anteriormente citadas funciona apenas com uma versão do *Visual Studio®* específica. Sendo assim, temos as seguintes compatibilidades:

- **MSVC7** – deve ser executado no *Microsoft® Visual Studio®* 2003.
- **MSVC8** – deve ser executado no *Microsoft® Visual Studio®* 2005.
- **MSVC9** – deve ser executado no *Microsoft® Visual Studio®* 2008.

Dessa forma, se você possui o *Microsoft® Visual Studio®* 2005 instalado, deverá obrigatoriamente utilizar os arquivos da pasta MSVC8 exclusivamente, e se por acaso tentar utilizar outra pasta ocorrerá erro ou então o programa poderá funcionar com algumas restrições. Essas informações podem ser acessadas em inglês na página do CHAI 3D, página

citada no 1º passo, na aba *support* e no link *Getting started* (Como começar), onde se encontra disponível toda a documentação[2] do programa.

É importante salientar que nesse trabalho foram utilizados apenas os arquivos da pasta MSVC9, que por sua vez são compatíveis apenas com o *Microsoft® Visual Studio®* 2008. Após a aquisição do *software*, consulte o APÊNDICE C desse trabalho que mostrará o passo a passo para realização da instalação do MSV2008.

3.3 VISUALIZANDO AS SIMULAÇÕES PRODUZIDAS PELO CHAI 3D

Depois de realizar os comandos da seção 3.2 (Aquisição dos Arquivos do CHAI 3D) e ter seguido as instruções para a instalação do *Microsoft® Visual Studio®* 2008 através do APÊNDICE C, agora é o momento de utilizar o CHAI 3D e visualizar algumas de suas simulações.

Existem duas maneiras de utilizar o CHAI 3D, a primeira delas é apenas executar um dos 15 projetos de simulações prontas que acompanham a plataforma. Para executar uma dessas simulações basta ir até a pasta em que salvou os arquivos do CHAI 3D, lembrando que essa pasta foi criada no APÊNDICE B e está localizada na "Área de Trabalho", pasta "CHAI 3D". Dentro da pasta "CHAI 3D" está localizada a pasta "chai3d-2.0.0" que descompactamos também no APÊNDICE B. Dê um duplo clique com o botão esquerdo do *mouse* nessa pasta e novas opções de pasta serão disponibilizadas, procure uma pasta com o nome de "*bin*" e dê um duplo clique com o botão esquerdo do *mouse* sobre ela, a Figura 11 mostra o conteúdo dessa pasta.

Agora basta escolher um dos 15 projetos e executá-lo com um duplo clique com o botão esquerdo do *mouse*. Não é necessário dar clique duplo no dispositivo háptico virtual indicado pelo ponto número 2, ponto destacado na Figura 11, pois ele será executado automaticamente com o ponto número 1. O resultado da execução do projeto "40-ODE-cube" pode ser observado na Figura 12.

---

[2] Documentação do programa. Disponível em: <www.chai3d.org/documentation.html>. Acessado em fev. 2013. Cabe ressaltar que tudo está escrito em lingua inglesa.

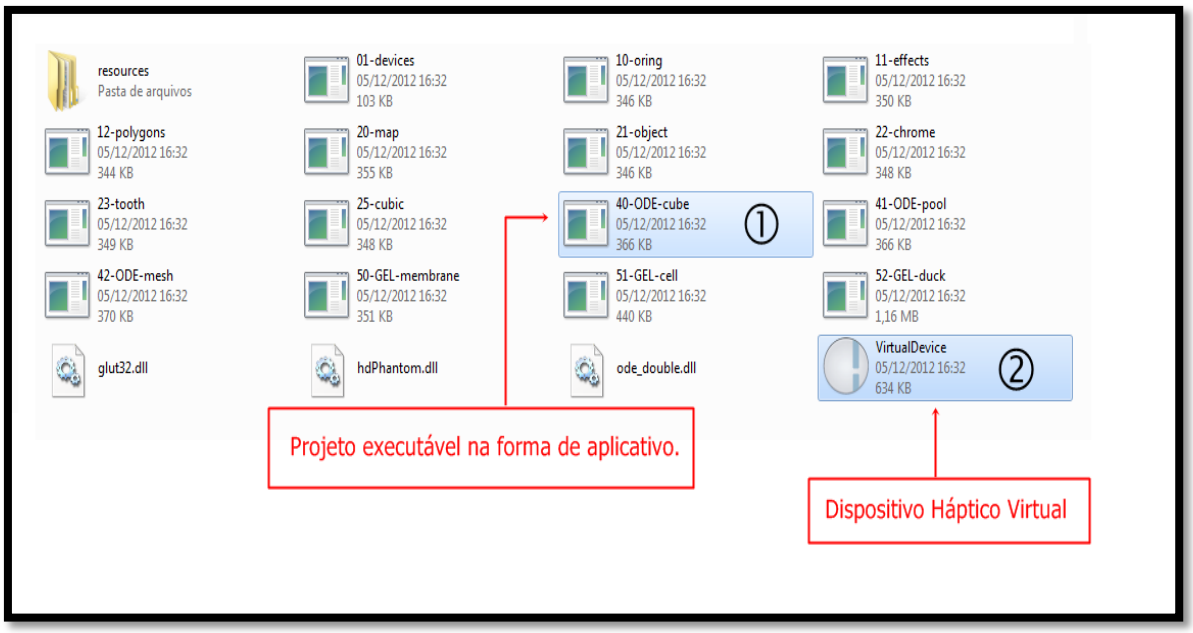

Figura 11 – Conteúdo da pasta "bin". O ponto 1 destaca um dos 15 projetos prontos disponibilizados junto com o CHAI 3D e o ponto 2 destaca o executável do dispositivo háptico virtual.

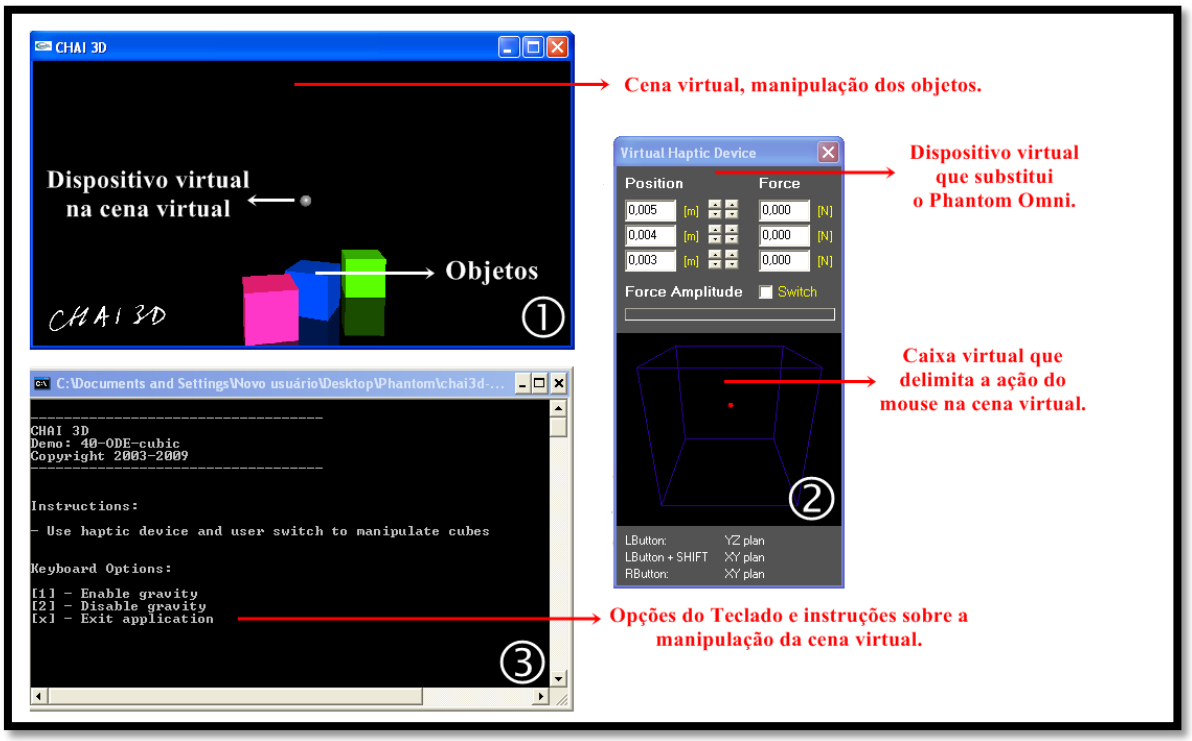

Figura 12 – Execução do projeto "40-ODE-cube", três janelas serão acionadas simultaneamente: janela (1) cena virtual, janela. (2) dispositivo háptico virtual e janela (3) instruções sobre a simulação em questão e opções do teclado.

Cada simulação tem seus objetivos em específicos, nesse caso o objetivo é simular a ação de forças física como a gravidade, a falta de gravidade e colisões entre os cubos. Para

saber mais sobre cada exemplo[3] pronto do CHAI 3D entre no sítio e veja as peculiaridades de cada projeto.

A segunda forma de acessar os projetos do CHAI 3D é por meio da utilização do programa MVS2008. Acessar os projetos pelo MVS2008 é interessante porque podem ser realizadas modificações no código original, o que por sua vez possibilita a produção de simulações da forma desejada, uma vez que esse programa possibilita a edição das linhas de código em C++.

Abra o programa que instalamos no APÊNDICE C, que provavelmente está nos ícones da "Área de Trabalho" ou no menu iniciar. No primeiro acesso o programa vai solicitar que o usuário especifique com que tipo de projetos gostaria de trabalhar, marque o campo "*Visual C++ Development Settings*" em "*Choose your default environment settings:*", marque a opção "*Allow Visual Studio to download and display online RSS content*", dessa forma o usuário do computador onde está instalado o MVS2008 será informado sobre atualizações disponíveis sem precisar procura-las manualmente, e posteriormente clique com o botão esquerdo do *mouse* em "*Start Visual Studio*" para iniciar a utilização do programa, Figura 13.

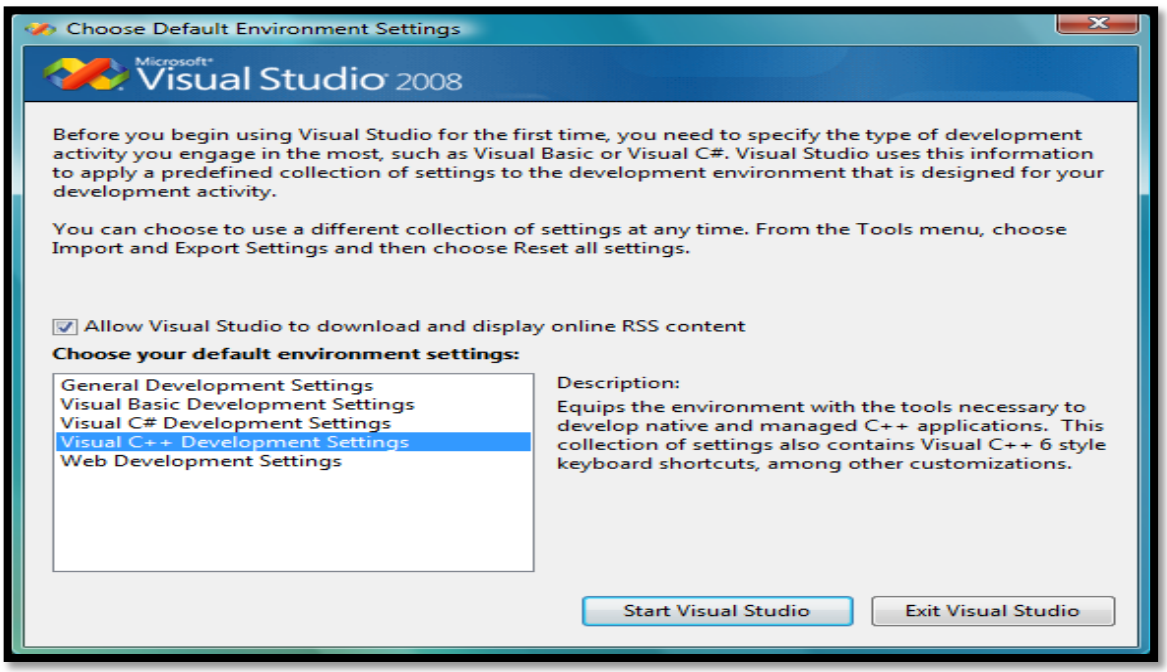

Figura 13 – Caixa de diálogo inicial (essa caixa de diálogo aparecerá apenas no primeiro acesso).

[3] Peculiaridade de cada projeto exemplo do CHAI 3D. Disponível em língua inglesa no sítio: <www.chai3d.org/examples.html>. Acessado em fev. de 2013.

Na tela inicial do programa clique em abrir arquivo (nesse momento deve ser indicado o local onde foi salvo o arquivo que adquirido na seção 3.1), porém, ao invés de procurar a pasta "*bin*", agora o objetivo é localizar a pasta "*projects*", que se encontra na mesma pasta onde está a pasta "bin", ou seja, a pasta "chai3d-2.0.0". Na Figura 14 são demonstradas três formas de abrir um arquivo no MVS2008, todas possuem o mesmo efeito e estão enumeradas de 1 à 3, outra alternativa para executar a mesma tarefa ocorre pressionando-se simultaneamente as teclas "Ctrl + O". Após realizar a ação para solicitar a abertura de um arquivo, surgirá uma nova janela com o nome de "*Open Project*" (Abrir Projeto), veja a Figura 14. Agora, indique a localização do arquivo. Tendo a "Área de Trabalho" (ponto número 4 da Figura 14) como ponto de partida, a seguinte sequência de pastas devem ser percorridas: Área de Trabalho\ Chai3D \ chai3d-2.0.0 \ projects \ msvc9. Na pasta "msvc9" localize o arquivo "CHAI 3D", execute um clique duplo com o botão esquerdo do *mouse* ou selecione-o e finalmente clique com o botão esquerdo do *mouse* em abrir (ponto número 5 da Figura 14).

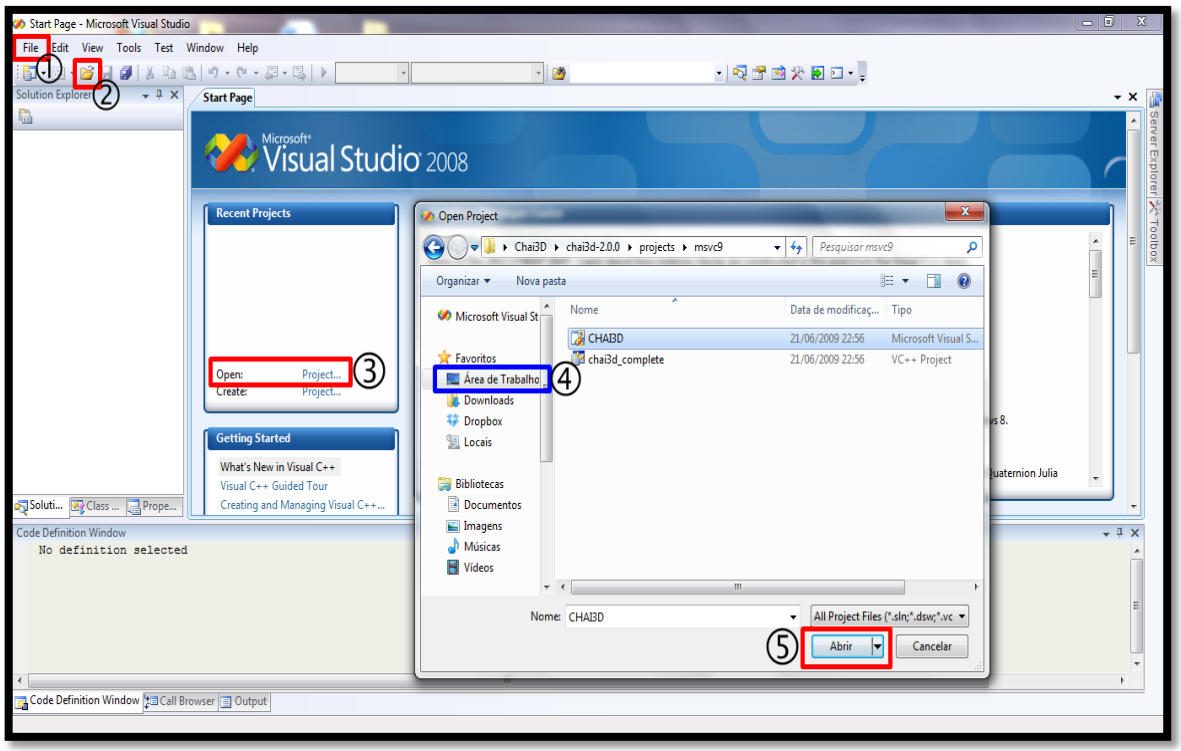

Figura 14 – Janela inicial do MSV2008 "Star Page" (em destaque botões de (1) a (3) para solicitar a abertura de um arquivo) e janela "Open Project" para abertura de projetos prontos em destaque o ponto (4), que marca a posição inicial do arquivo no computador e ponto (5), botão que finaliza a ação de abrir o arquivo.

Por consequência, surgirá a janela da Figura 15, onde são identificadas algumas funcionalidades importantes.

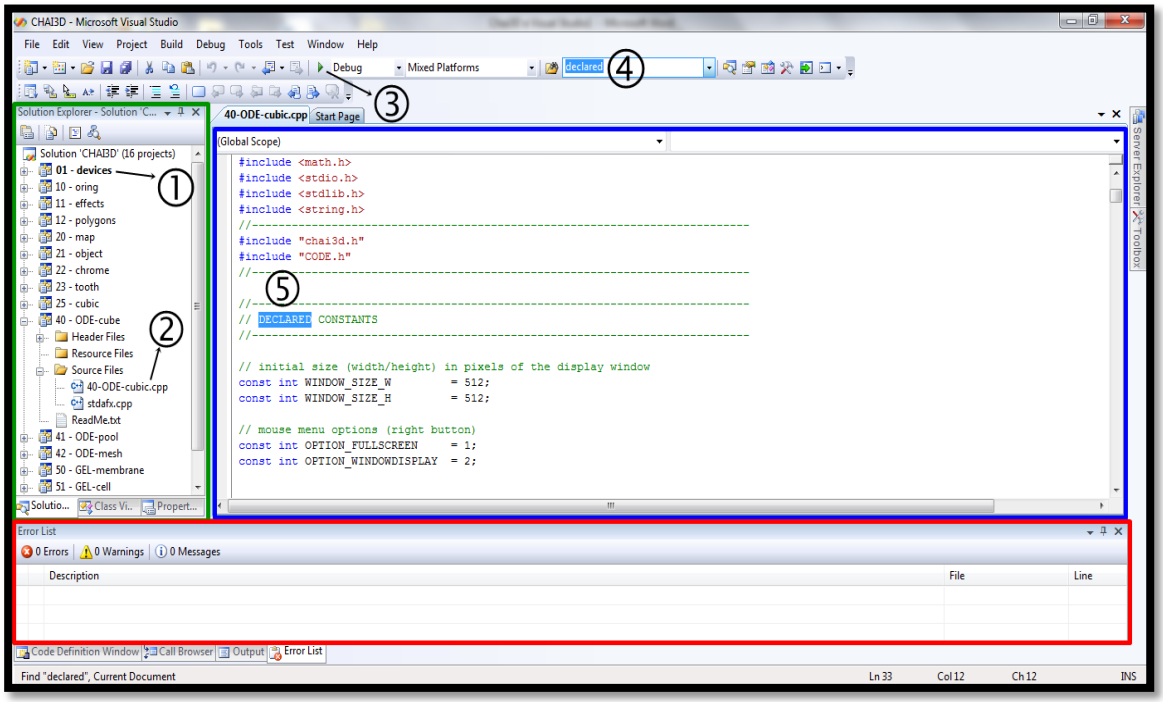

Figura 15 – Janela principal do MSV2008. Em destaque, retângulo verde (projetos), retângulo azul (linhas de código do projeto 40-ODE-cube, referente ao arquivo "40-ODE-cubic.cpp"), retângulo vermelho (lista de erros), ponto número 1 (projeto definido para inicialização), ponto número 2 (arquivo em edição no retângulo azul), ponto número 3 (botão para inicialização a simulação), ponto 4 número (campo para realizar pesquisa de palavras no código do projeto) e ponto número 5 (palavra localizada no código do projeto).

Na área delimitada pelo retângulo verde estão agrupados os projetos do CHAI 3D, dentre eles, observe um projeto destacado em negrito, ponto número 1, quando abrir esse conjunto de projetos pela primeira vez, por padrão, esse projeto fica selecionado automaticamente. Para selecionar outro projeto basta clicar com o botão direito do mouse sobre o projeto desejado e selecionar a opção "*Set as StartUp Project*" - (Definir Como Projeto de Inicialização) com o botão esquerdo do *mouse*, dessa forma, o projeto desejado aparecerá em negrito, enquanto que o projeto anterior será exibido sem o negrito.

Apesar de estar em negrito, o projeto "01-devices" não está em edição, como pode ser percebido na Figura 15, o arquivo em edição no retângulo azul é o arquivo "*40-ODE-cubic.cpp*", destacado no ponto número 2. Independentemente do arquivo que possa estar sendo editado, é o projeto que estiver em negrito que será inicializado.

Na barra de ferramentas, o ponto número 3 indica o botão que executa a inicialização de um dos projetos selecionados no retângulo verde. A mesma tarefa pode ser realizada pressionando-se a tecla “F5”.

Ainda na barra de ferramentas, existe um campo bastante útil para pesquisas, ponto número 4, que localiza trechos do texto escrito no corpo do código que está em desenvolvimento ou sendo editado no retângulo azul.

O ponto número 5 mostra uma palavra que foi localizada utilizando-se o recurso de pesquisa comentado no ponto anterior.

Para gerar uma simulação semelhante a da Figura 12, clique sobre o projeto “*40-ODE-cube*” com o botão esquerdo do *mouse* e selecione a opção “*Set as StartUp Project*”. Depois que o arquivo estiver selecionado e em negrito pressione a tecla “F5” para executar a simulação. Uma caixa de diálogo entrará em ação perguntando se é esse o projeto que você realmente deseja executar, antes de clicar em “*Yes*”, clique no campo abaixo para marcar a opção “*Do not show this dialog again*”, dessa forma essa caixa de diálogo não aparecerá novamente, caso seja sua vontade. Nesse momento, o MVS2008 irá carregar todos os projetos e no final irá mostrar apenas aquele que você selecionou. Note que o dispositivo háptico virtual não aparece, ponto número 2 da Figura 12, isso ocorre porque aqui ele não executa manualmente, sendo assim, aperte a tecla “ESC” para finalizar a simulação ou então clique sobre o “x” no canto superior direito da janela. Localize o conteúdo da pasta “bin” novamente, Figura 11, ponto número 2, dê um clique duplo com o botão esquerdo do *mouse* sobre o arquivo “*VirtualDevice*” para executar o dispositivo háptico virtual. Com o dispositivo háptico em execução volte no MVS2008 e pressione a tecla “F5”, o resultado da execução dos comandos acima pode ser observado na Figura 16.

Uma opção para que não seja necessário fazer o trajeto até a pasta “*bin*”, sempre que fechar o dispositivo háptico virtual, é clicar com o botão direito sobre o ícone na pasta “*bin*” e enviá-lo para “Área de Trabalho” ou então arrastá-lo até a barra de tarefas e fixá-lo, garantindo dessa forma agilidade para acessá-lo em outras oportunidades.

Recomendo bastante cuidado na execução de modificações nos projetos e posterior visualização, mesmo que existam linhas de códigos erradas, o MVS irá executar a última simulação como se nada estivesse errado, porém, no campo lista de erros, destacado no retângulo vermelho da Figura 15, estarão listados todos os erros ocorridos na simulação solicitada. Caso esse campo não esteja visível, no menu principal, clique com o botão

esquerdo do *mouse* em "*View*" (ver) e navegue até "*Other Windows*" (Outras Janelas) e posteriormente "*Error List*" (Lista de Erros), feito isto, o campo se tornará visível.

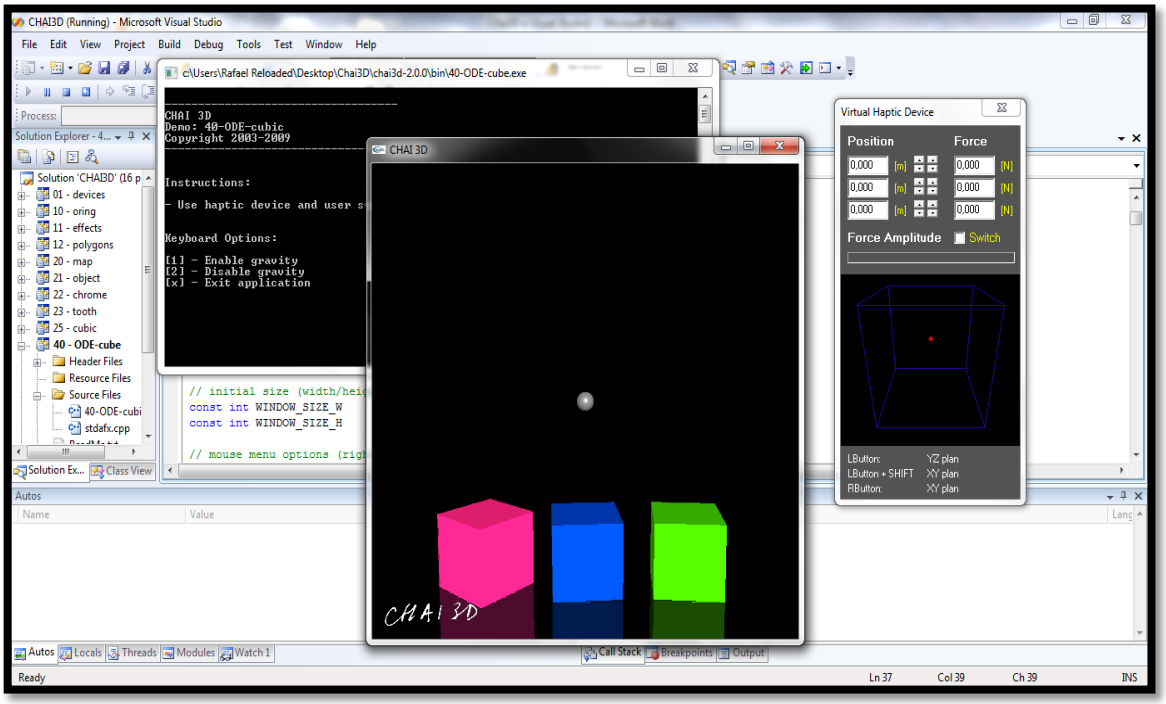

Figura 16 – Simulação em execução por meio do MVS2008.

Outro ponto bastante importante que ajuda a descobrir se suas modificações estão entrando em vigor é localizar a pasta "*bin*" e procurar por um arquivo com um nome igual ao do projeto que você está editando e verifique se a data e hora da última modificação coincidem com o momento em que você pressionou a tecla "F5", ou seja, com o momento em que você executou a simulação.

Por último, aconselho que no momento em que estiver executando a simulação, a mesma consumirá muita capacidade de processamento do computador, se for possível feche todos os programas que não estiver utilizando. E também, toda vez que for compilar dados, pressionar "F5", é necessário fechar a última simulação para evitar sobrecarga no processador.

3.4 INSTALAÇÃO DA INTERFACE DE RETORNO DE FORÇA *PHANTOM OMNI*®

O primeiro procedimento a ser tomado é a instalação física de uma das placas (*hardware*) que possibilitem a conexão da interface com um computador de mesa, *notebook* ou *laptop*.

Mesmo que a instalação da placa ocorra perfeitamente, o dispositivo háptico ainda não vai entrar em funcionamento, isto porque devemos acessar a página da SensAble para localizar também os *drivers* do *software* para a instalação do dispositivo háptico.

Acesse o endereço eletrônico: <www.sensable.com>. No menu principal do sítio, logo abaixo da logo marca da empresa, arraste o ponteiro do *mouse* sobre o botão "*products and services*" (produtos e serviços) que dará acesso a um sub menu, nele encontre a opção "*OpenHaptics Toolkit*" - (Kit de Ferramentas Hápticas Abertas) e clique com o botão esquerdo do *mouse* sobre ela, veja Figura 17.

Na próxima página, localize[4] o *link* para *download* do arquivo para desenvolvedores acadêmicos "Academic Developers", veja a Figura 18, que mostra o local onde deverá ser clicado.

Após clicar com o botão esquerdo do *mouse* sobre o *hiperlink* para realização do *download* mencionado no parágrafo anterior e visualizado na Figura 18, você será direcionado à página de suporte da SensAble, Figura 19. Como as pesquisas estão concentradas nessa área, clicar com o botão esquerdo do *mouse* no *hiperlink* que está destacado com o retângulo vermelho "*OpenHaptcis Academic Edition Developers*" (Edição Acadêmica para Desenvolvedores). Mais uma vez o sítio irá direcionar o usuário a outra página, agora uma página de cadastro de usuários, onde será fornecer nome e sobrenome, país, nome da universidade, número serial do dispositivo háptico e também, aceitar o termo de licença. Feito isso, dentro de aproximadamente 30 minutos, o suporte da SensAble lhe enviará um nome de usuário, que poderá ser o endereço de e-mail fornecido e uma senha de cinco dígitos para realizar a aquisição dos arquivos necessários para o funcionamento do *PHANTOM Omni*®.

Acesse a conta de e-mail fornecida no cadastro e localize o e-mail enviado pelo suporte da SensAble, de posse do nome de usuário e senha, navegue até a página de suporte da SensAble, veja Figura 19, preencha com seu nome de usuário e senha enviados pela SensAble e clique com o botão esquerdo do *mouse* em "*Log in*".

[4] Uma maneira mais eficiente de localizar uma palavra ou trecho escrito em uma página da internet, independentemente de qual navegador esteja utilizando, é pressionar simultaneamente as teclas de atalho Ctrl + F, digitar a palavra de interesse e depois pressionar a tecla "*Enter*".

Figura 17 – Página inicial da SensAble, em destaque menu "PRODUCTS AND SERVICES" e sub menu "OpenHaptics Toolkit".

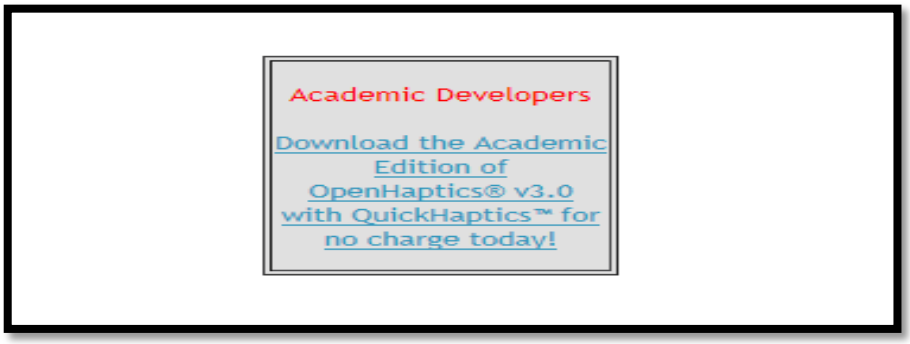

Figura 18 – Hiperlink para *download* da versão acadêmica para desenvolvedores.

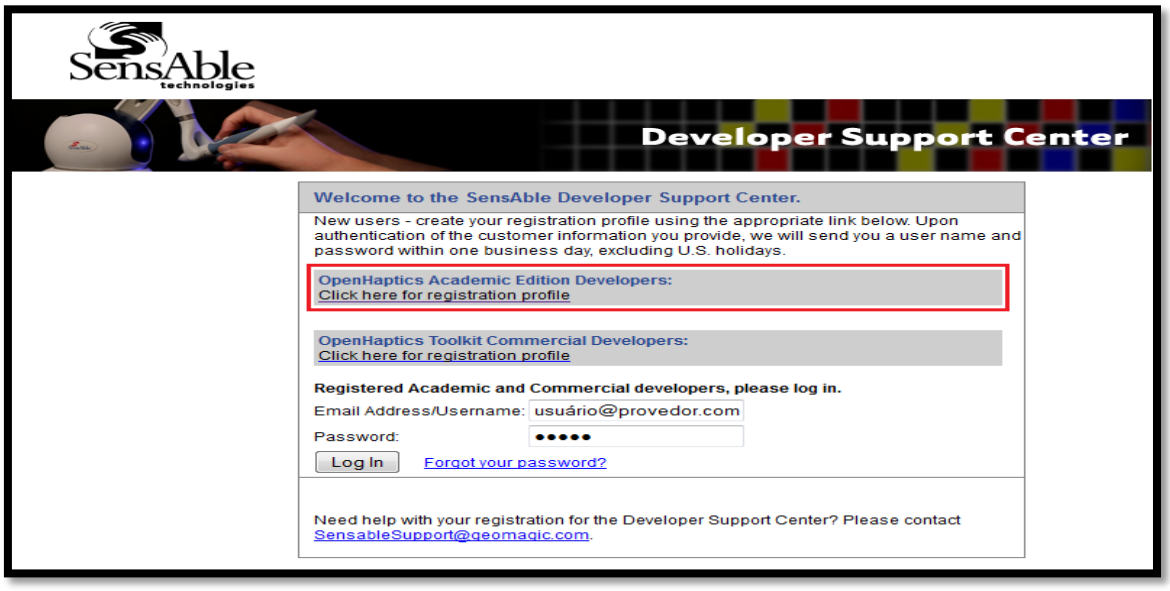

Figura 19 – Página de suporte e *Lon in* da SensAble.

Ao fazer o "*Log in*", o usuário é encaminhado para uma nova página, a Figura 20 mostra essa página, onde estão dispostos os arquivos para três sistemas operacionais

diferentes: *Windows*®, *Linux*® e *Mac*®. Marque a opção delimitada pelo retângulo vermelho, referente ao sistema operacional *Windows*®.

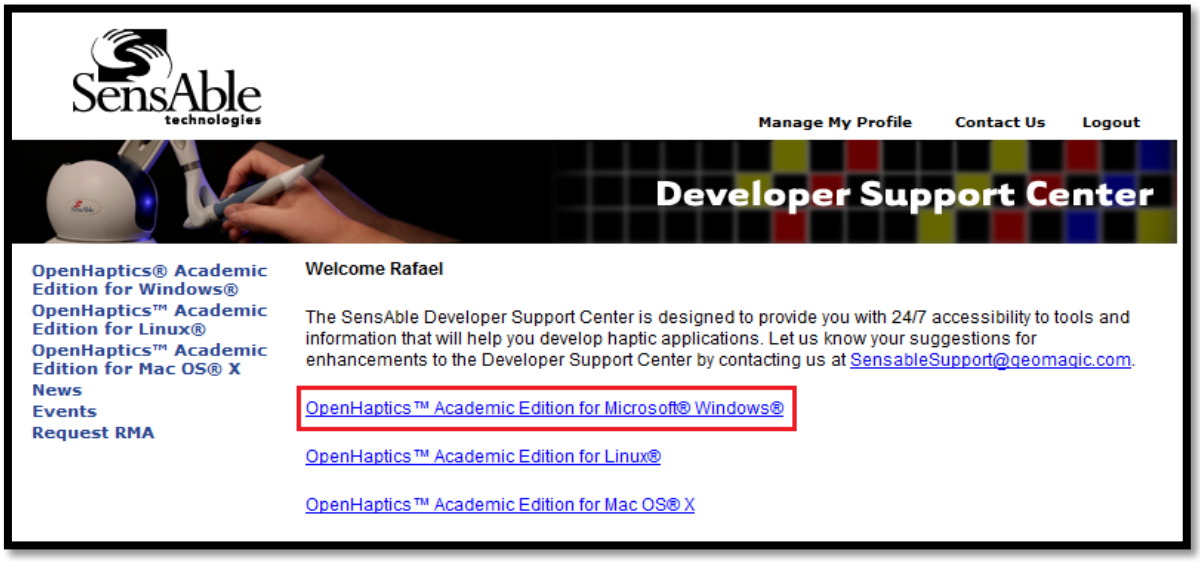

Figura 20 – Página para seleção do sistema operacional utilizado.

Após clicar com o botão esquerdo do *mouse* sobre o *hiperlink* para aquisição dos arquivos referentes ao sistema operacional *Windows*®, destacado pelo retângulo vermelho na Figura 20, o usuário será novamente encaminhado para uma nova parte do sítio, onde agora ele deverá clicar no *hiperlink* "*Software Downloads*", destacado com o retângulo vermelho na Figura 21.

Figura 21 - Página de suporte e *download* do *software* para utilização do PHANTOM Omni®.

Assim que abrir a nova página, clique com o botão esquerdo do *mouse* na versão mais nova do *software*, fazendo isto, evita-se adquirir um *software* com problemas relacionados à instalação, e também, um *software* eventualmente mais rápido ou eficiente na sua configuração ou disposição das ferramentas. Até o início do ano de 2013, a versão mais recente é a "v.3.1", a qual deverá ser adquirida, localize o retângulo vermelho na Figura 22, que indica a versão mais recente do *software*. Assim que efetuado um clique com o botão esquerdo do *mouse* no *hiperlink* destacado na Figura 22, aparecerá uma janela de diálogo, onde deverá ser especificado o local para armazenar o arquivo, caso possua uma pasta de sua preferência especifique nesse momento, caso não possua, direcione o arquivo para "Área de Trabalho", clique em "*OK*" com o botão esquerdo do *mouse* e aguarde até que o *download* seja concluído.

Apesar de existir uma forma de descompactar o arquivo já descrita na seção 3.2 "Aquisição dos Arquivos do Chai3D", ensinarei a descompactar novamente de uma forma que o arquivo possa ser gerado em uma pasta com o mesmo nome do arquivo de origem, dessa forma, localize o arquivo na "Área de Trabalho" ou no local armazenado no seu computador e clique com o botão direito do *mouse* sobre ele, certificando-se que tenha instalado no seu computador o programa "*Winrar*", e depois clique com o botão esquerdo sobre "Extrair para OpenHaptics_3.1_Academic_Edition\", Figura 23, essa ação fará com que uma pasta com o nome de "OpenHaptics_3.1_Academic_Edition" seja criada também na "Área de Trabalho", contendo os arquivos descompactados.

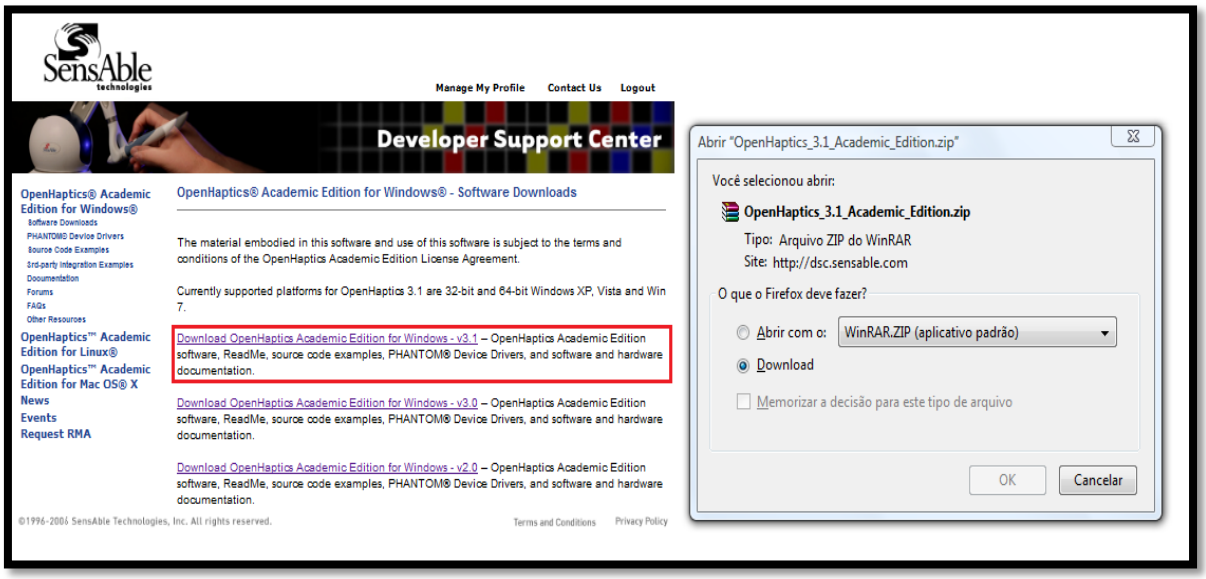

Figura 22 - Página de suporte e seleção da versão do *software* para *download* e janela de diálogo para *download*.

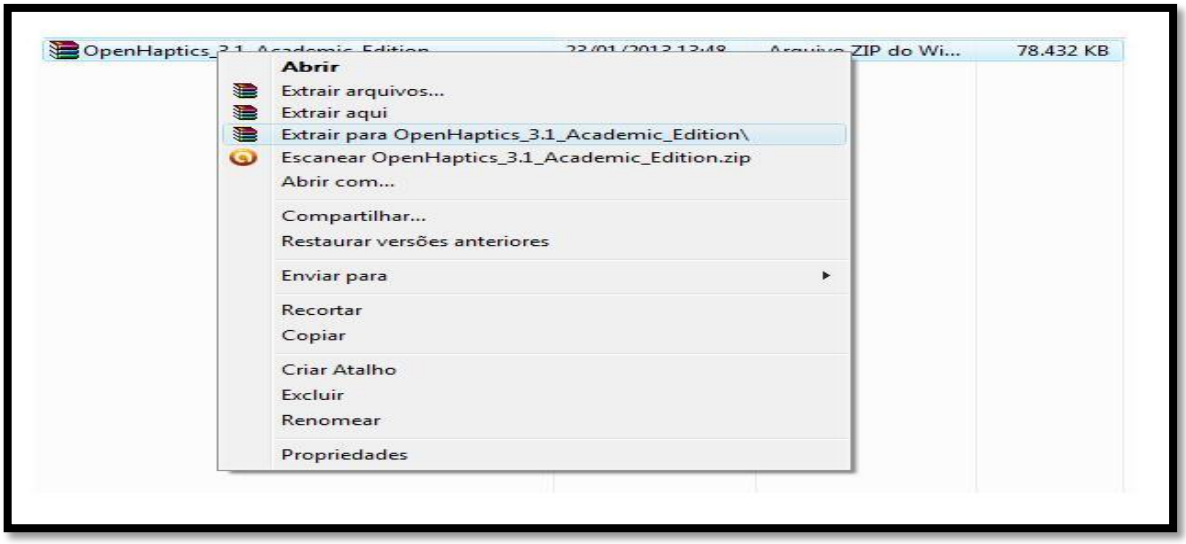

Figura 23 – Procedimentos para descompactar os arquivos para instalação do *software*.

Na pasta "OpenHaptics_3.1_Academic_Edition", consta a documentação em formato PDF, em língua inglesa, para a instalação do *software* e dos *drivers* da interface háptica, além de dois arquivos executáveis que deverão ser instalados para a correta utilização da interface de retorno de força: "OpenHaptics_Academic_3.1" e "Phantom_Device_Drivers_5.1.7_Release".

3.4.1 Instalação do OpenHaptics_Academic_3.1

Na pasta "OpenHaptics_3.1_Academic_Edition" procure o arquivo "OpenHaptics_Academic_3.1", dê um clique duplo com o botão esquerdo do *mouse* sobre o arquivo e aguarde até que a janela de diálogo seja iniciada. Assim que a primeira janela for iniciada, clique na caixa "*I agree to the above terms and conditions*" – (Eu concordo com os termos e condições acima), prossiga a instalação, clique com o botão esquerdo do *mouse* em "*Next*" – (Próximo), Figura 24.

Na segunda janela, Figura 25, apenas clique com o botão esquerdo do *mouse* em "*Next*" e não modifique nenhum campo.

Na terceira e última janela, Figura 26, por padrão, é sugerido pelo sistema que a instalação seja feita no *driver* "C" do computador, caso queira informar outro local realize esse procedimento nesse momento, caso não seja necessário, apenas clique com o botão

esquerdo do *mouse* sobre "*Install*" – (Instalar) e aguarde até que o processo seja finalizado e clique com o botão esquerdo do *mouse* em "*Finish*" – (Terminar).

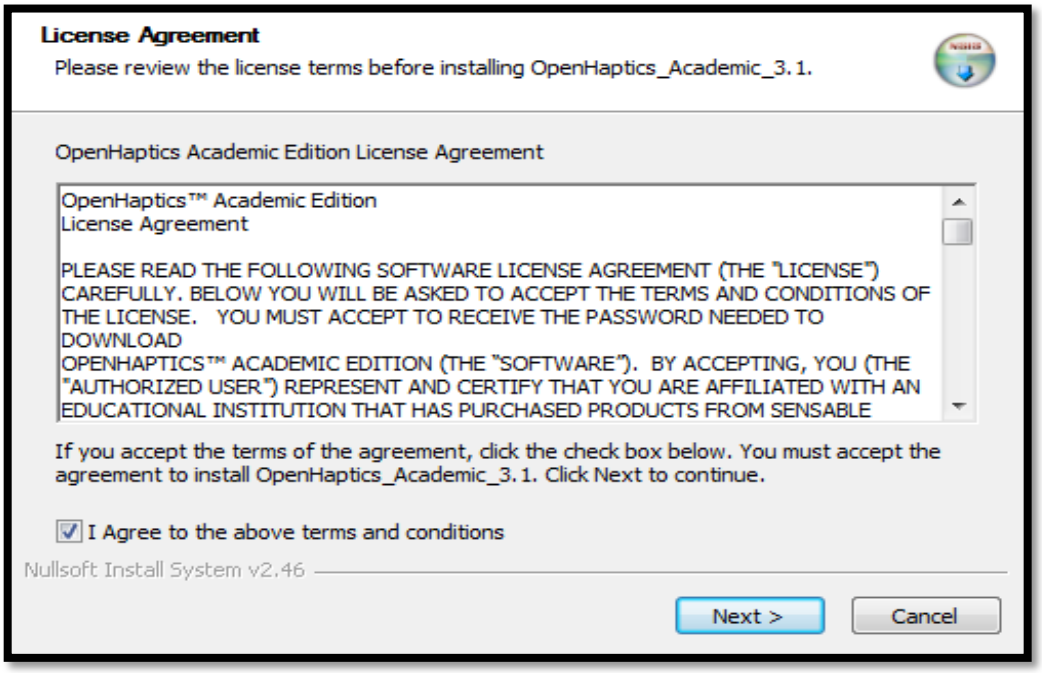

Figura 24 - Janela de diálogo para instalação do *OpenHaptics Academic*, termo de licença.

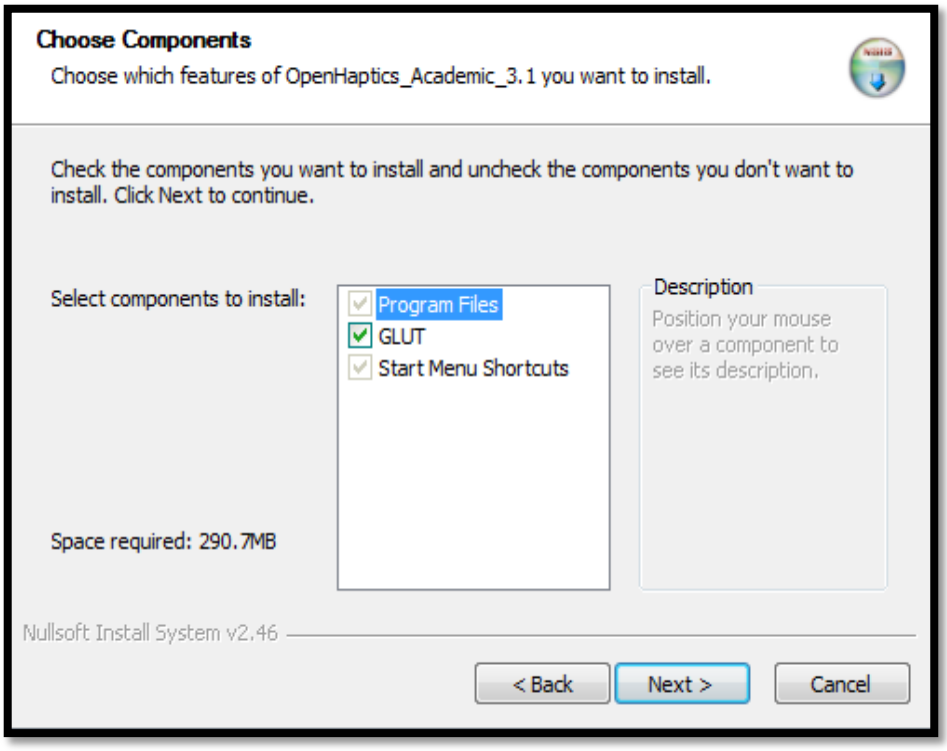

Figura 25 - Janela de diálogo para instalação do *OpenHaptics Academic*, escolha dos componentes.

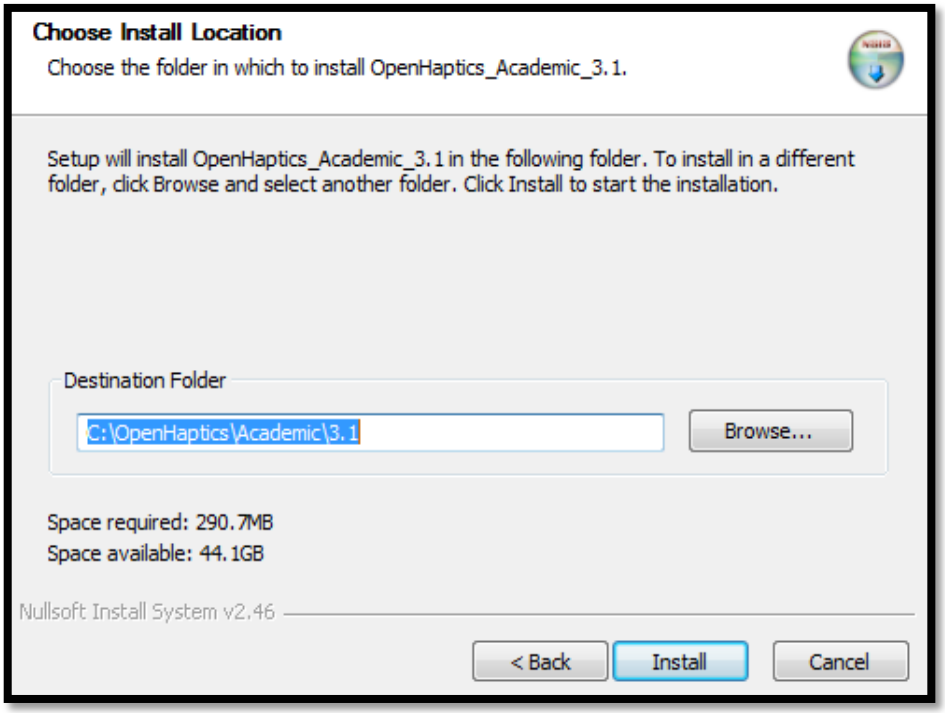

Figura 26 - Janela de diálogo para instalação do *OpenHaptics Academic*, escolha o destino da instalação.

Prossiga com a instalação dos *drivers* para o correto funcionamento da interface de retorno de força *PHANTOM Omni*®.

3.4.2 Instalação Phantom_Device_Drivers_5.1.7_Release:

A instalação do *driver* para o *PHANTOM Omni*® segue a mesma lógica utilizada para instalação do *software* anterior. Na pasta "OpenHaptics_3.1_Academic_Edition" procure o arquivo "Phantom_Device_ Drivers_5.1.7_Release", dê um clique duplo com o botão esquerdo do *mouse* sobre o arquivo e aguarde até que a janela de diálogo seja iniciada. Assim que a primeira janela for iniciada, clique na caixa "*I agree to the above terms and conditions*", prossiga a instalação, clique com o botão esquerdo do *mouse* em "*Next*", Figura 27.

Na janela posterior, segunda janela, Figura 28, apenas clique com o botão esquerdo do *mouse* em "*Next*" e não modifique nenhum componente.

Na última janela, Figura 29, por padrão, será sugerido pelo sistema que a instalação seja feita no *driver* "C" do computador, caso deseje informar outro local realize esse procedimento nesse momento, caso não seja necessário, apenas clique com o botão esquerdo

do *mouse* sobre "*Install*" e aguarde até que o processo seja finalizado e clique com o botão esquerdo do *mouse* em "*Finish*".

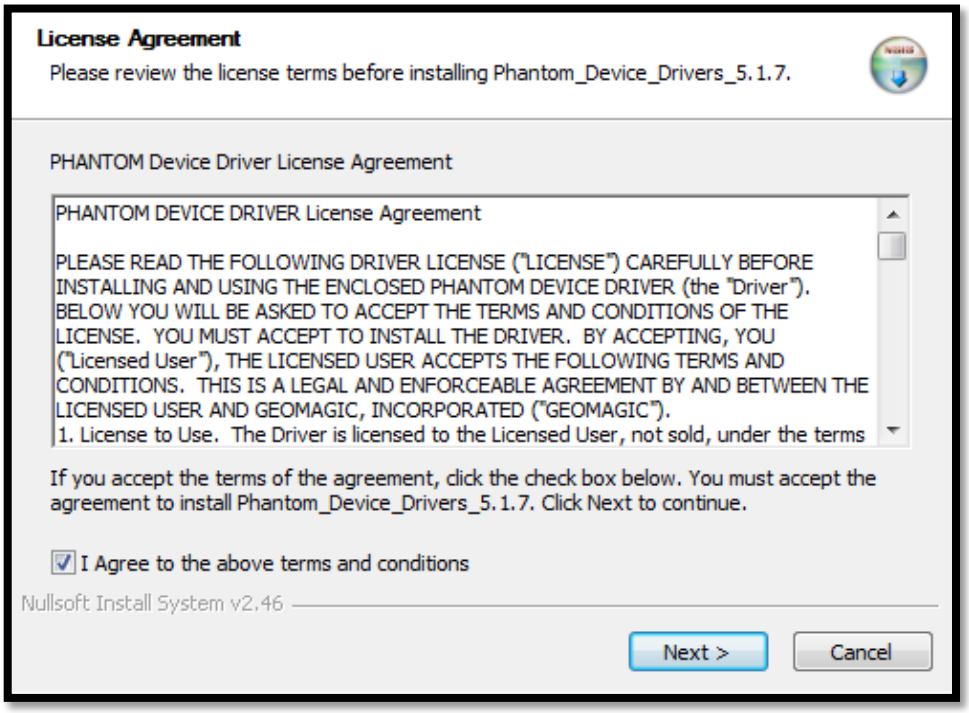

Figura 27 - Janela de diálogo para instalação do *driver* para o *PHANTOM Omni*®, aceitação do termo de licença.

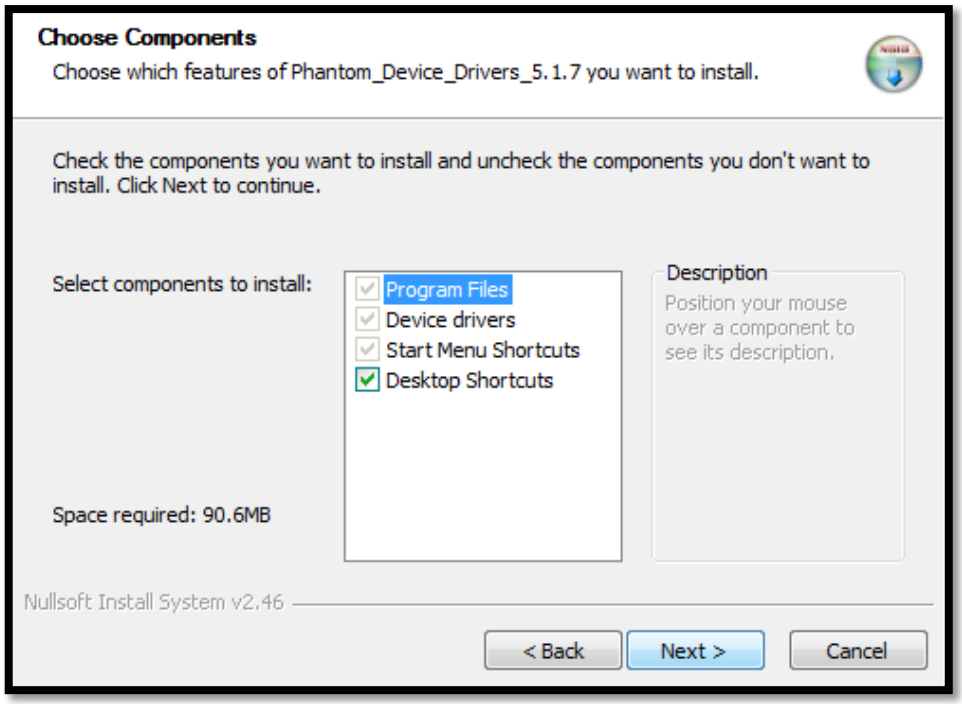

Figura 28 - Janela de diálogo para instalação do *driver* para o *PHANTOM Omni*®, escolha dos componentes da instalação.

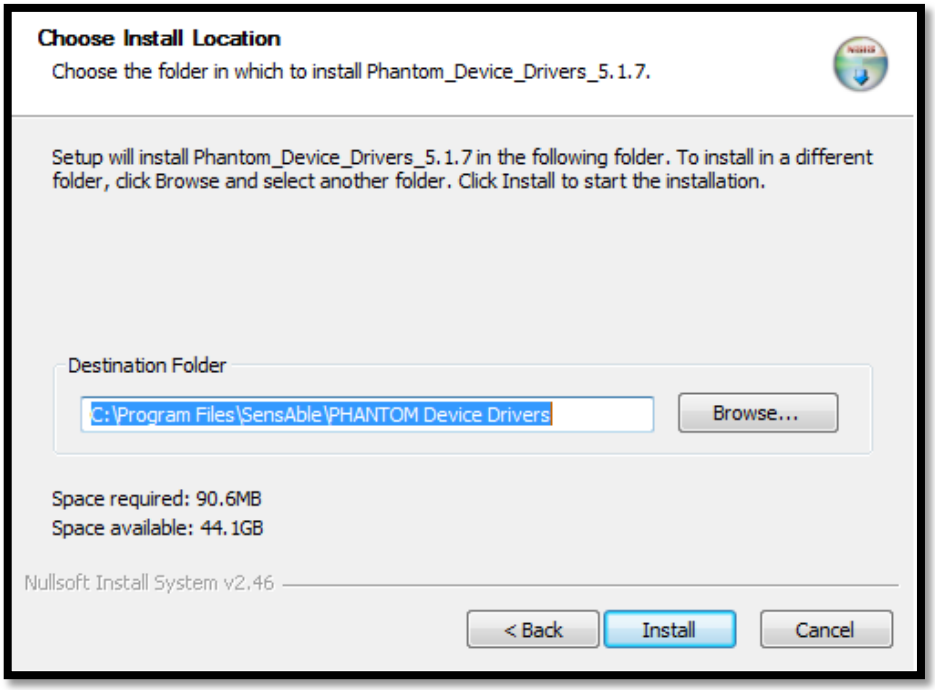

Figura 29 - Janela de diálogo para instalação do *driver* para o *PHANTOM Omni*®, escolha do destino da instalação.

Reinicie o computador, conecte o *PHANTOM Omni*® a uma fonte de energia e ligue o cabo *FireWire*® ao dispositivo e ao computador para que possamos configurá-lo.

3.4.3 Configuração do *PHANTOM Omni*®

Antes de começar esse procedimento certifique-se de que a interface háptica esteja conectada ao computador e também a uma fonte de energia, feito isso, prossiga com o procedimento abaixo.

A instalação do *software* do *PHANTOM Omni*® fará com que dois ícones apareçam na "Área de Trabalho", respectivamente "PHANToM_ConFiguration" e "PHANToM_Test" e também surgirá uma pasta no menu iniciar com o nome de *SansAble*, contendo além dos dois ícones anteriormente citados, a documentação do *Phantom*, em língua inglesa, e um ícone para desinstalar o *software*, Figura 30.

Primeiramente deve ser feita a configuração do dispositivo, para tanto, execute um duplo clique com o botão esquerdo do *mouse* sobre o ícone "PHANToM_ConFiguration" na "Área de Trabalho" ou um clique com o botão esquerdo do *mouse* sobre o ícone de mesmo nome na pasta "SensAble" existente no menu iniciar, essa ação fará com que a janela

mostrada na Figura 31 surja na tela. Se os passos anteriores foram executados exatamente como descritos nesse capítulo, espera-se que o número de série do dispositivo apareça no campo "*Serial Number*", esse campo é preenchido por 11 dígitos numéricos, esses dígitos podem ser encontrados ou visualizados na parte inferior do dispositivo. Faça com que os outros campos destacados na janela sejam preenchidos exatamente como os da Figura 31, clique em "*Apply*" – (Aplicar) com o botão esquerdo do *mouse e* posteriormente em *OK* com o mesmo botão. Feche a janela.

Figura 30 – Pasta que ficará disponível no menu iniciar após a instalação do *software* do *PHANTOM Omni*®.

Se você não obteve um resultado como o mostrado na Figura 31 não continue o procedimento, volte algumas etapas e realize-as novamente até que o resultado seja equivalente ao da figura em questão.

O procedimento seguinte será iniciar o "PHANToM_Test" para realizar as últimas configurações no dispositivo. Execute um duplo clique com o botão esquerdo do *mouse* sobre o ícone "PHANToM_Test" na "Área de Trabalho" ou um clique com o botão esquerdo do *mouse* sobre o ícone de mesmo nome na pasta "SensAble" existente no menu iniciar, essa ação fará com que a janela mostrada na Figura 32 surja na tela. Marque o campo "*User*" – (Usuário), que fará com que o modo usuário seja ativado e clique com o botão esquerdo do *mouse* na próxima aba "*Select*" – (Selecionar) ou apenas no botão "*Next*" disposto na parte inferior da janela que também realiza o mesmo procedimento.

Na aba "*Select*" – (Selecionar), Figura 33, preencha os campos como mostrado na figura, observe que o campo "*Serial Number*" aparece novamente, nesse campo deve aparecer o mesmo número que aparece no campo *Serial Number* da Figura 31. Observe que no último campo existe a mensagem "*Phantom inicialization Ok...*" – (O *Phantom* iniciou da maneira correta), clique com o botão esquerdo do *mouse* na próxima aba "*Calibrate*" – (Calibrar) ou apenas no botão "*Next*" disposto na parte inferior da janela.

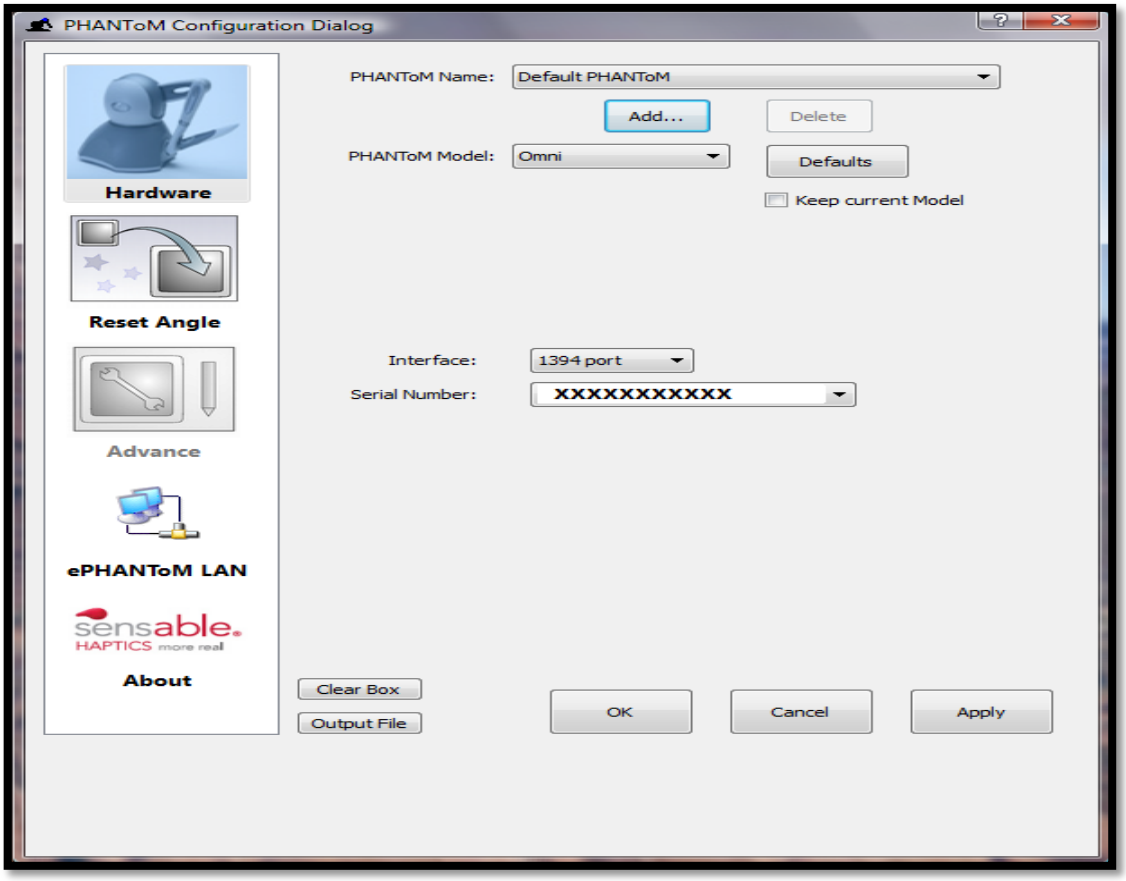

Figura 31 – Janela de configuração do *PHANTOM Omni*®.

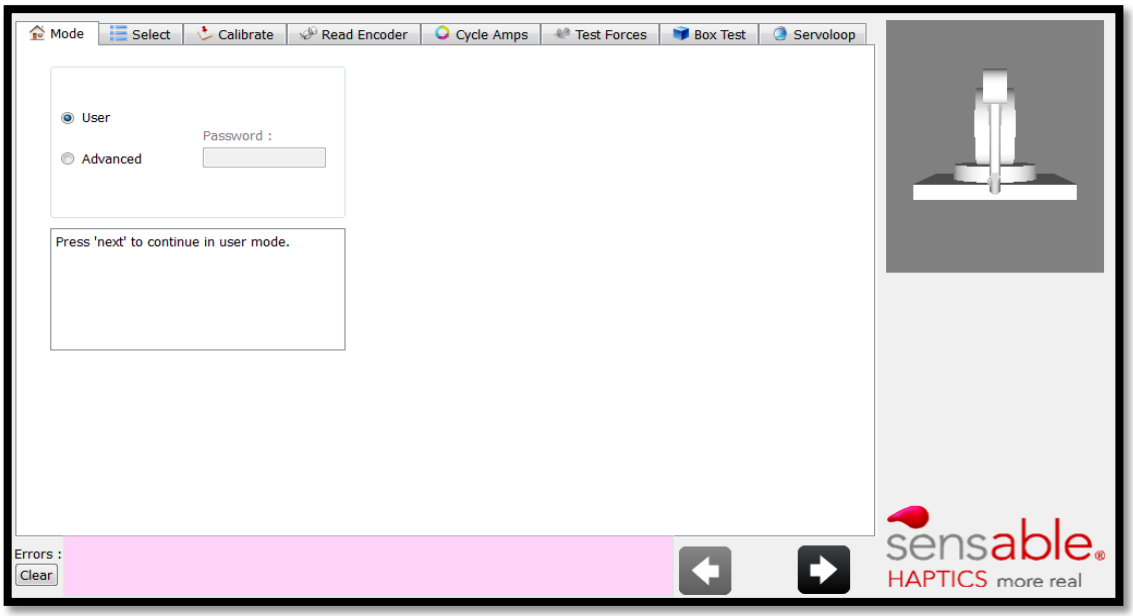

Figura 32 – Janela e aba inicial do "PHANToM_Test", seleção do modo a ser utilizado.

Na aba "*Calibrate*", Figura 34, basta inserir a caneta na base do dispositivo, assim como mostrado na Figura e aguardar até que a mensagem "CALIBRATION DONE!" –

(CALIBRAÇÃO REALIZADA!) apareça no retângulo verde. Clique com o botão esquerdo do *mouse* na próxima aba "*Read Encoder*" – (Ler Codificadores) ou apenas no botão "*Next*" disposto na parte inferior da janela para continuar.

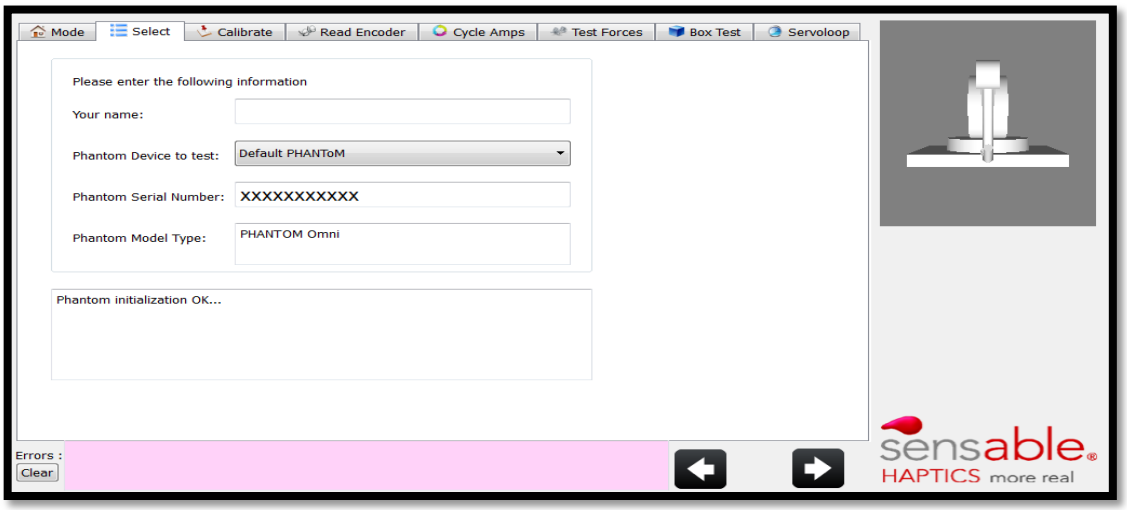

Figura 33 – Aba *Select*, seleção do dispositivo.

Na aba "*Read Encoder*", Figura 35, manipule a caneta do dispositivo e observe os valores em cada campo variando à medida que se realiza algum movimento. Descubra a posição em que o valor correspondente deveria ser zero e verifiquei se o campo realmente apresenta esse valor. Realizado esse procedimento, clique com o botão esquerdo do *mouse* na próxima aba "*Cycle Amps*" – (Amplitude os Ciclos) ou apenas no botão "*Next*" disposto na parte inferior da janela.

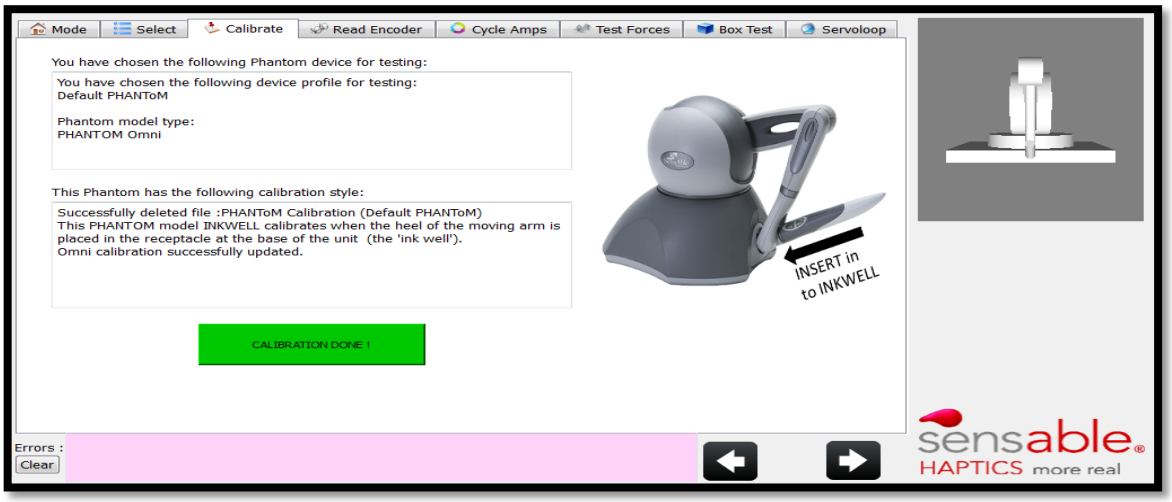

Figura 34 – Aba *Calibrate*.

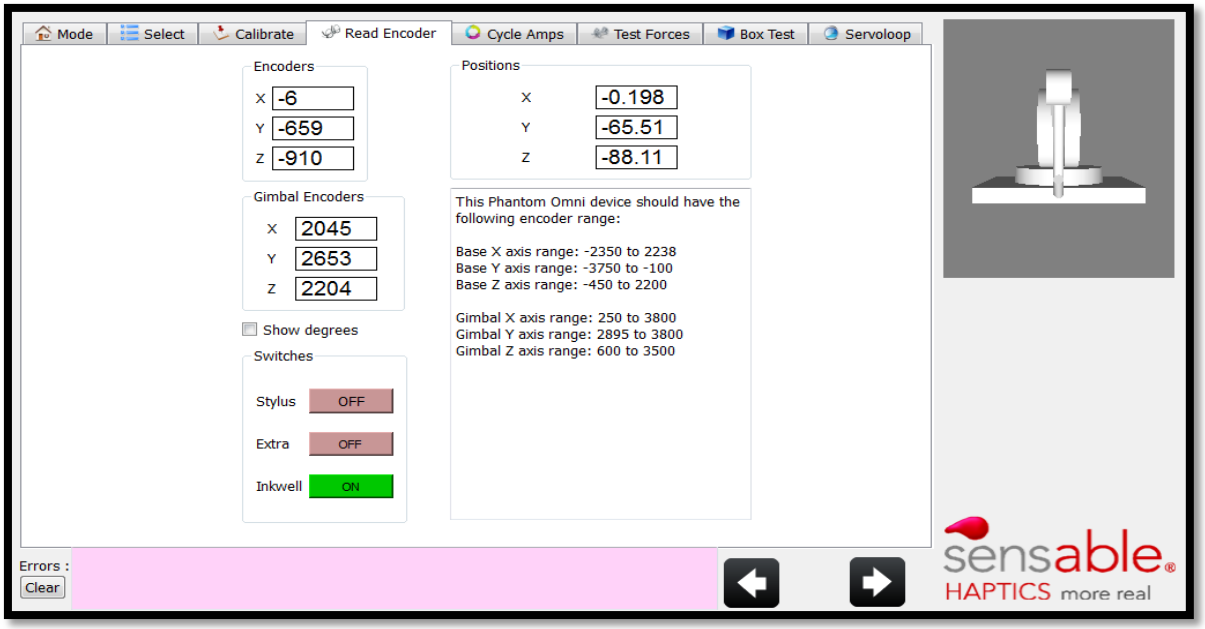

Figura 35 – Aba *Read Encoder*, faz a leitura da posição espacial de cada eixo do dispositivo.

Na aba *Cycle Amps*, Figura 36, é feito um teste na chave de liga/desliga do aparelho, o que verifica se o dispositivo está sendo ligado e desligado no momento correto, observe que o luz existente na base do dispositivo, onde pode ser inserida a caneta, liga quando o estado do campo "*STATUS*" está em "*ON*" – (ligado) e desliga quando o estado do campo está em "*OFF*" (Desligado). Prossiga o teste, clique com o botão esquerdo do *mouse* na próxima aba "*Test Forces*" – (Teste Forças) ou apenas no botão "*Next*" disposto na parte inferior da janela.

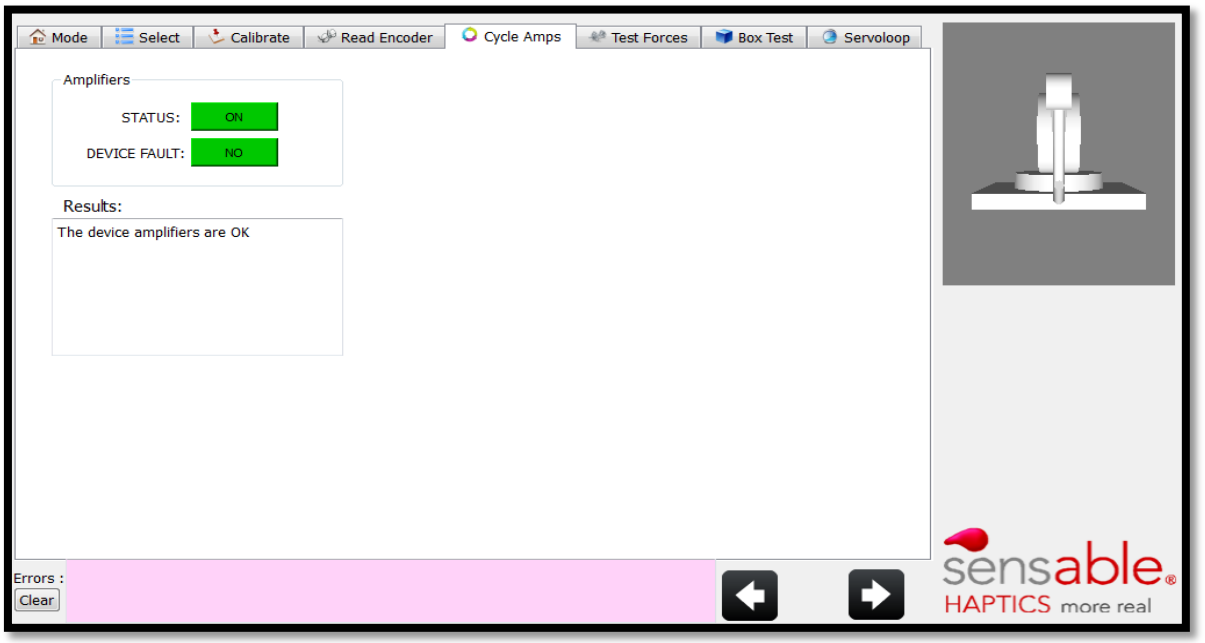

Figura 36 - Aba *Cycle Amps*, faz a leitura do estado do dispositivo (ligado ou desligado).

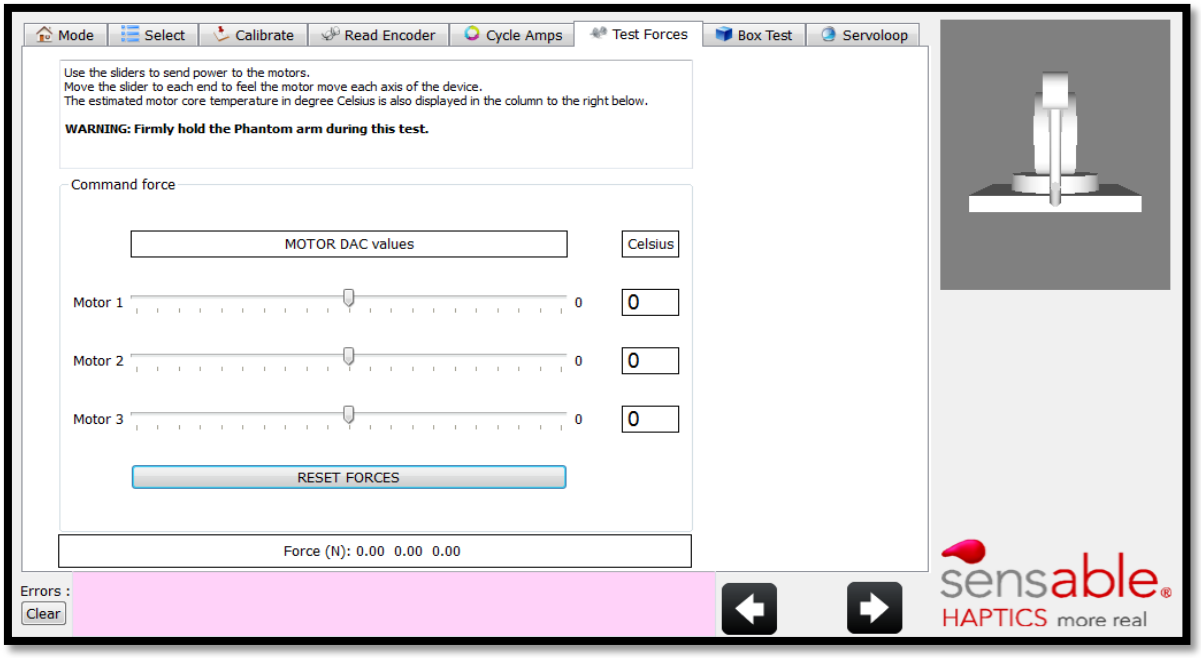

Figura 37 - Aba *Test Forces*, faz com que o dispositivo responda com um movimento em cada eixo.

Na aba *Test Forces*, Figura 37, pode ser observado e sentido o retorno de força do dispositivo em cada eixo, para tanto, mantenha uma distância segura do aparelho a fim de evitar choques físicos do dispositivo com o seu corpo, principalmente com a região do seu rosto. Modifique o marcador do "Motor 1", "Motor 2" e "Motor 3", cada um deles correspondendo a um eixo (x, y, z), note que estão sendo executados movimento exatamente nesses três eixos. Para continuar com o teste clique com o botão esquerdo do *mouse* na próxima aba "*Box Test*" – (Teste de Caixa) ou apenas no botão "*Next*" disposto na parte inferior da janela.

Na aba *Box Test*, Figura 38, existe uma caixa tridimensional que simula a posição virtual do dispositivo háptico em uma cena virtual, além de informar a força em newtons para cada eixo e a porcentagem de força naquele momento. Mova a caneta do *Phantom* para esquerda e para direita (eixo x), mova para cima e para baixo (eixo y) e mova para trás e para frente (eixo z). Quando o usuário move a caneta do *Phantom*, simultaneamente pode-se observar uma esfera se movendo na caixa virtual, à medida que o usuário desloca essa esfera para as bordas da caixa, percebe-se por meio do tato e de colisões geradas pelo dispositivo e pela simulação, as paredes dessa caixa virtual. Dessa forma, observe na Figura 38 que o usuário exerceu uma força no eixo y equivalente a 1.3 N, o que também equivale a 40% da força suportada nesse eixo. Caso o usuário exerça uma força maior do que o equipamento suporta, automaticamente o mesmo deixará de responder, até que o usuário faça movimentos

suportados e o dispositivo volte a fazer as leituras de retorno de força, isso se faz necessário para que sua estrutura física não se danifique rapidamente.

Para finalizar o teste, clique com o botão esquerdo do *mouse* na próxima aba "*Servoloop*" – (Sem tradução!) ou apenas no botão "*Next*" disposto na parte inferior da janela.

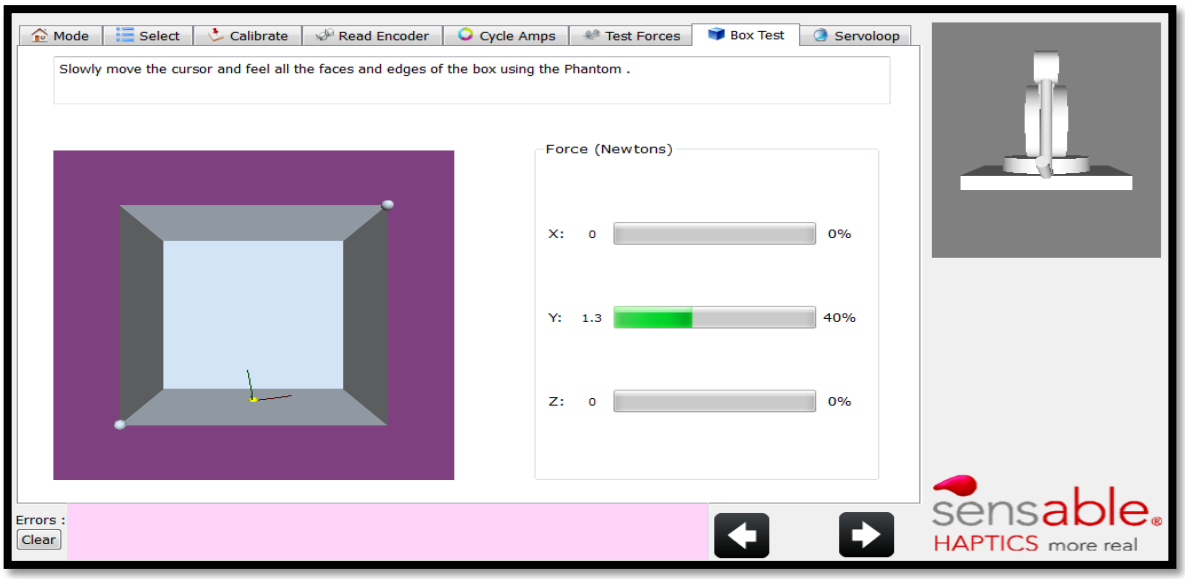

Figura 38 – Aba *Box Test*, mostra uma caixa tridimensional que simula uma esfera amarela com pequenos vetores.

As informações da aba *Serveloop*, Figura 39, são referentes ao tempo de resposta das ações que o dispositivo físico realiza em relação ao dispositivo virtual, nessa aba não é necessário fazer nenhuma alteração ou configuração diferente. Caso seja necessário saber esse tempo de resposta em *hertz*, basta apertar o botão "*START*" – (Começar).

Terminado o processo de instalação da interface de retorno de força PHANTOM Omni®, volte na seção 3.3 e execute um dos modelos prontos do CHAI 3D sem executar o dispositivo háptico virtual, dessa forma o PHANTOM Omni® (dispositivo háptico físico) entrará em funcionamento. Por fim, verifique se à medida que se realizam movimentos com a caneta física, algo é alterado ou movido na cena virtual, por exemplo, a esfera branca que aparece na maioria dos projetos exemplos.

Em alguns desses projetos é possível utilizar tanto o dispositivo físico quando o virtual na mesma cena gráfica ou na mesma simulação, um exemplo dessa funcionalidade do MVS2008 e do CHAI 3D pode ser observado ao se executar o projeto "01 - devices" com o dispositivo háptico físico conectado ao computador e também com o dispositivo háptico

virtual em execução, note que aparecerão duas esferas, e as duas podem ser manipuladas simultaneamente, uma pelo dispositivo virtual e outra pelo dispositivo físico.

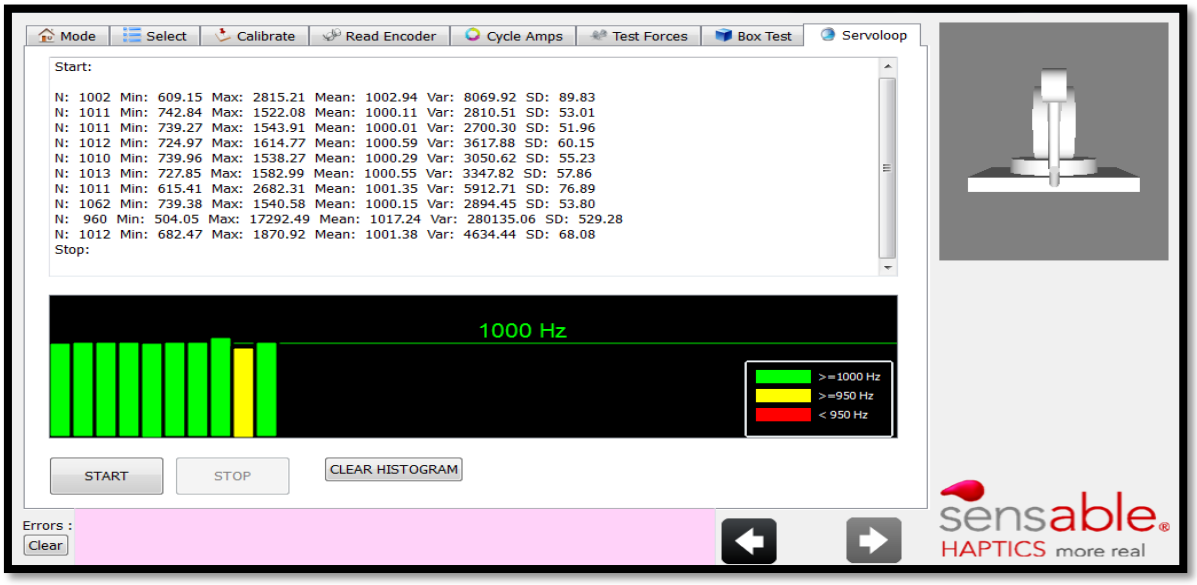

Figura 39 – Aba *Servoloop*, informa o tempo de resposta das ações realizadas com o dispositivo.

Caso nenhuma das ações informadas nos dois últimos parágrafos funcione, refaça todo o procedimento a fim de detectar o que ocorreu de errado na execução de cada um deles.

4 Solução Analítica Para Deformação de Membranas em Duas Dimensões

Um desafio corrente em realidade virtual é o cálculo da deformação com realismo físico e rapidez suficiente para serem utilizados em interfaces hápticas. Enquanto taxas de 30 Hz em geral são suficientes para as interfaces gráficas, já para interfaces hápticas como as utilizadas nesse trabalho são necessárias taxas maiores que 1000 Hz para um funcionamento adequado do dispositivo. No entanto, os métodos usuais para o cálculo de deformações com realismo físico em geral são lentos e estão longe de alcançar tais taxas de processamento. Dessa forma, faz-se necessário criar novos métodos que obtenham velocidade e realismo suficientes para aplicações específicas.

Nesse capítulo é apresentado o formalismo matemático utilizado na deformação de membranas circulares homogêneas. A seção 4.2 aborda os cálculos para uma membrana que não possui uma borda fixa, tendendo, dessa forma, o infinito. Na seção 4.3 são abordados os cálculos para uma membrana que possui borda fixa, por exemplo, no caso do tímpano do ouvido humano.

4.1 INTRODUÇÃO

Imagine um aquário vazio e que apresenta a forma de um retângulo com comprimento, profundidade e altura. Sobre a parte superior desse aquário foi afixada uma fina membrana de material homogêneo e elástico Figura 40(a), semelhante ao material utilizado para produzir balões de festa. Posteriormente foram aplicadas forças na superfície da membrana. Devido à aplicação dessas forças a membrana se deforma. Um exemplo desse tipo de deformação pode ser observado na Figura 40(b).

Em seguida é mostrada uma série de cálculos que tem como objetivo descreverem o comportamento dessa membrana, ou seja, as forças que atuam perpendicularmente à região da

membrana, com suas intensidades e posições, além de ser calculado também, qual a deformação resultante da superfície da membrana. Esses cálculos serão válidos desde que as deformações geradas na membrana não ultrapassem ângulos maiores do que 15º, pois essa será uma condição para a solução das equações.

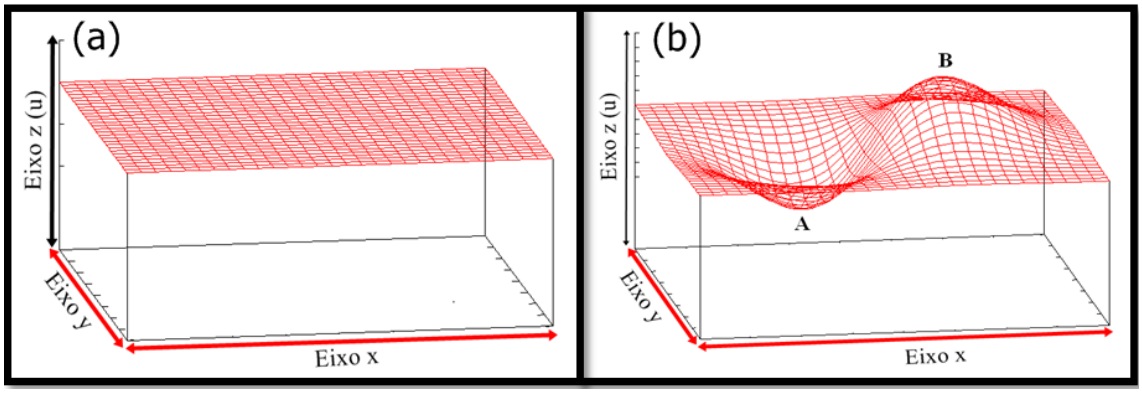

Figura 40 - Aquário com uma membrana esticada sobre sua parte superior. (a) Membrana sem nenhuma força atuando sobre sua superfície (eixo ***z***). (b) Forças sendo aplicadas para baixo em A e para cima em B no eixo ***z***.

Reveja o exemplo da Figura 40(a), a membrana esticada sobre a parte superior do aquário. Por estar esticada existem forças em todo o plano x e y. Devido a essas forças, quando feito um pequeno corte em qualquer lugar, os dois lados do corte serão puxados e se afastarão. Portanto, sobre essa membrana existe uma tensão superficial, que é conhecida como a magnitude da tensão superficial τ, e é entendida como a força por unidade de comprimento necessária para manter juntos os dois lados de um corte, se o mesmo fosse realizado em uma membrana como a da figura em questão.

Imagine uma situação de uma força externa exercida de baixo para cima, gerando uma deformação como o mostrado na Figura 41(a). Definimos u como o deslocamento vertical da membrana em relação à sua superfície e à sua posição normal, e x, y são as coordenadas no plano horizontal.

Considere um pequeno pedaço da superfície de comprimento Δx, Figura 41(b) e largura Δy. A tensão superficial exercerá forças em cada extremidade deste pedaço. A força ao longo da extremidade 1 da figura será $\tau \cdot \Delta y$, direcionada tangencialmente à superfície – isto é, em um ângulo θ_1 com a horizontal. Ao longo da extremidade 2, a força também será $\tau \cdot \Delta y$, uma vez que a tensão superficial é a mesma em toda a membrana, mas a direção será

diferente sob um novo ângulo θ_2. A força resultante para cima devido às extremidades 1 e 2 será a componente vertical. A componente vertical pode ser calculada multiplicando-se $\tau \cdot \Delta y$ pelo seno do ângulo:

$$dF = \tau \cdot \Delta y \cdot \mathrm{sen}\,\theta_2 - \tau \cdot \Delta y \cdot \mathrm{sen}\,\theta_1 \tag{4-1}$$

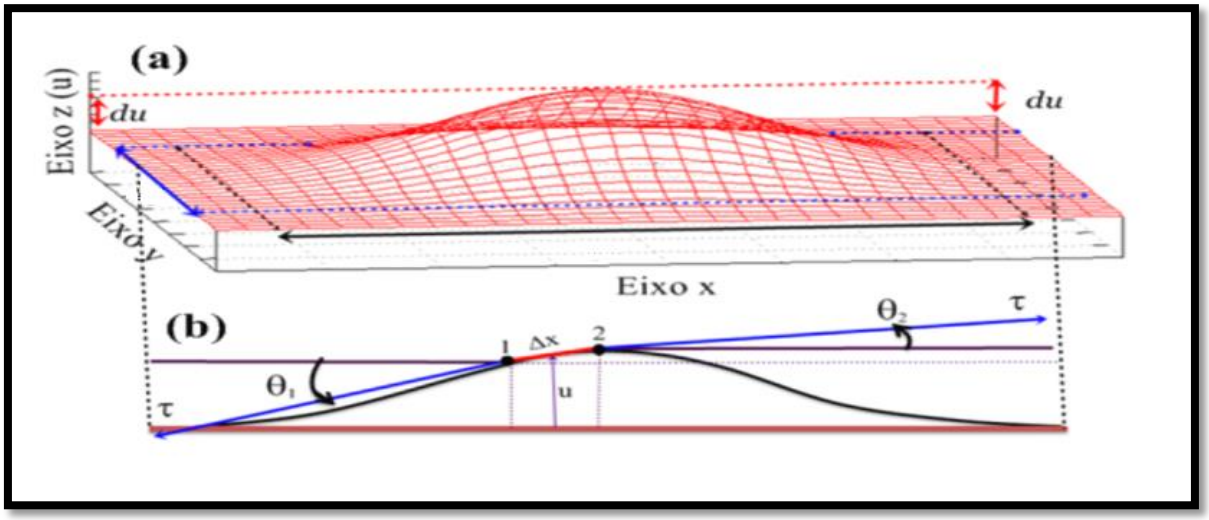

Figura 41 - Aquário com uma membrana esticada sobre sua superfície. (a) Membrana deformada pela ação de uma força exercida de baixo para cima e seus respectivos eixos em ***x***, ***y*** e ***z***. (b) Reflexo da deformação ocorrida em (a), (seção paralela ao eixo ***x***).

Limitando-se nossas considerações a pequenas deformações da membrana, ou seja, para pequenas curvaturas[5]. Dessa forma, pode ser feita a substituição de $sen\theta$ por $\tan\theta$, garantindo uma boa aproximação, que por sua vez pode ser escrito como du/dx, devido a seguinte relação trigonométrica,

$$\tan\theta = \frac{\partial u}{\partial x} \tag{4-2}$$

e devido à análise da Figura 42, que é uma ampliação nos pontos 1 e 2 da Figura 41. Realizando-se a substituição da equação (4-2) em (4-1) nas extremidades 1 e 2, fará com que a variação da força seja dada por:

[5] Como $\pi/180^o$ = 0,017453292 radianos = 1º, podemos dizer, por exemplo, que um ângulo $\theta = 10°$ possui tangente igual a 0,176326 radianos e que o valor de θ convertido diretamente para radianos é igual a 0,174532, o que nos da uma precisão de duas casas após a vírgula.

$$dF = \tau \left[\left(\frac{\partial u}{\partial x}\bigg|_{x=x_2} \right) - \left(\frac{\partial u}{\partial x}\bigg|_{x=x_1} \right) \right] dy \tag{4.3}$$

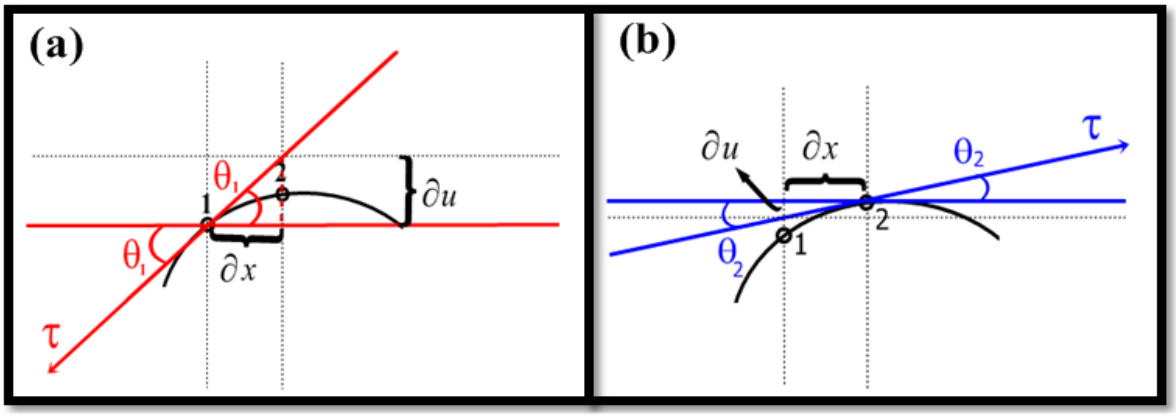

Figura 42 - Ampliação da Figura 41. (a) Ampliação e relações no ponto 1. (b) Ampliação e relações no ponto 2.

Definindo-se,

$$f(x) = \frac{\partial u(x)}{\partial x} \tag{4-4}$$

e lembrando que a definição de derivada é,

$$\frac{\partial f(x)}{\partial x} \equiv \lim_{x_2 \to x_1} \frac{f(x_2) - f(x_1)}{x_2 - x_1} = \lim_{\Delta x \to 0} \frac{f(x_2) - f(x_1)}{\Delta x} \tag{4-5}$$

A quantidade entre colchetes na Equação (4-3) pode ser escrita no limite em que $\Delta x \to 0$ como:

$$\left[\frac{\partial}{\partial x} \left(\frac{\partial u}{\partial x} \right) \right] dx \tag{4-6}$$

Dessa forma, a Equação (4-3) pode ser reescrita como:

$$dF = \tau \left[\frac{\partial}{\partial x} \left(\frac{\partial u}{\partial x} \right) \right] dxdy \tag{4-7}$$

Como haverá outra contribuição para as forças na direção vertical dF devido às duas extremidades na direção y, a equação final para a força em função do deslocamento irá adquirir o seguinte formato:

$$dF = \tau\left[\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right)\right]dxdy \tag{4-8}$$

As deformações no plano são causadas por forças externas atuando para baixo em uma área $dA_0 = dx_0 \cdot dy_0$ na membrana, ou seja, uma pressão. Essa pressão é representada por $P = dF / dA_0$. Mas, segundo a Terceira Lei de Newton, uma força de mesmo módulo e sentido oposto é aplicada pela membrana, que é a força na superfície da membrana por unidade de área para cima, devido às forças exercidas externamente. Quando a membrana estiver em equilíbrio (o caso estático) esta força que a membrana faz deve contrabalancear a força externa que foi calculada por meio da Equação (4-8), logo, a pressão aplicada pela membrana pode ser escrita como:

$$P = -\frac{dF}{dA_0} \tag{4-9}$$

Juntando-se as Equações (4-8) e (4-9) obtêm-se:

$$\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right) = -\frac{P}{\tau} \tag{4-10}$$

A Equação (4-10) é conhecida como *Equação de Poisson*, e a solução mais conveniente para essa equação, no caso em particular, é por meio das *Funções de Green*. Assim, o termo após a igualdade deve ser substituído pelo *Delta de Dirac* que é definido nas Equações (1.2.1) e (1.2.2) em DUFFY (2001), por:

$$\delta(x) = \begin{cases} \infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{4-11}$$

onde,

$$\int_{-\infty}^{\infty} \delta(x)dx = 1 \tag{4-12}$$

Além dessa substituição, a técnica para solucionar a *Equação de Poisson*, impõe a substituição da função u pela chamada *Função de Green*, que aqui será chamada de g, como comumente é designada. Considerando-se então que $u = g$ na Equação (4-10) e substituindo-se o termo após a igualdade por δ, a seguinte expressão é formada:

$$\left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right) = -\delta \tag{4-13}$$

onde, g representa a *Função de Green*. Dessa forma, a solução de u para um caso em três dimensões pode ser encontrada por meio da integral 5.0.14, solucionada por DUFFY (2001):

$$u(\vec{r}, \vec{r}_0) = \iiint_{V_0} f(\vec{r}_0) g(\vec{r}) dV_0 \tag{4-14}$$

onde $f = -P / \tau$.

Porém, como o caso em questão encontra-se em duas dimensões, a integral da Equação (4-14) adquire o seguinte formato:

$$u(r, \theta, r_0, \theta_0) = \iint_{A_0} f(r_{0,} \theta_0) g(r, \theta) dA_0 \tag{4-15}$$

onde, r_0 e θ_0 representam a posição (em coordenadas polares) onde a força é aplicada, e r e θ representam uma posição da membrana, ou seja, o mesmo que x e y.

No entanto, para utilizar as *Funções de Green* é necessário definir condições de contorno para o problema. Esse problema é sanado impondo-se a *Condição de Contorno de Dirichlet* ou *Condição de Contorno de Primeiro Tipo*, ou seja, que a posição da membrana seja conhecida para um conjunto de posições.

A solução desse problema foi realizada nesse trabalho para duas situações distintas: a primeira situação foi nomeada “Espaço Livre”, na qual é considerada uma membrana circular com raio infinito $(b=\infty)$, Figura 43(a). A segunda situação ocorre em uma membrana circular de raio b finito $(b=1)$, Figura 43(b), por isso nomeada “Borda Fixa”. Nos dois casos a deformação é devida a uma força pontual infinita aplicada na posição r_0.

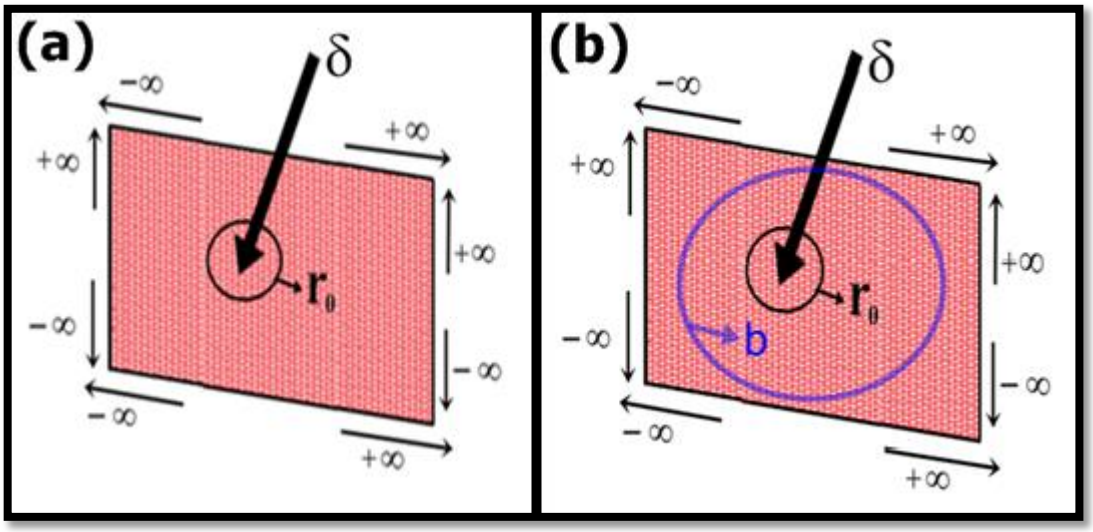

Figura 43 - Membrana, representação da força pontual e infinita aplicada sobre a membrana na posição r_0. (a) Primeira Situação: membrana circular com bordas infinitas, toda a região delimitada pelo retângulo preto pode ser deformada. (b) Segunda Situação: membra circular com bordas fixas, a região dentro do círculo azul pode ser deformada, enquanto a região fora do círculo é rígida, ou seja, não possibilita deformação. Nesse caso a condição de contorno é deslocamento zero para a borda $r = b$.

4.2 PRIMEIRA SITUAÇÃO – ESPAÇO LIVRE ($b = \infty$ e $r_0 = 0$)

Esse caso trata de uma deformação na posição u_0, que é também considerada a origem do sistema de coordenadas. Ou seja, quando a posição de aplicação da força varia, a posição da origem do sistema de coordenada muda! Nessa situação a solução do problema pela *Função de Green*, Equação (4-13), é dada na tabela 5.1.1 também em DUFFY (2001), onde:

$$g(r,\theta) = -\frac{\ln(r)}{2\pi} \tag{4-16}$$

Dessa forma, utilizando-se a Equação (4-15) para duas dimensões é possível encontrar o valor de u, onde:

$$u(r,\theta,r_0,\theta_0) = \iint_{A_0} \left(-\frac{P}{\tau}\right) \cdot \left(-\frac{\ln(r)}{2\pi}\right) dA_0 \tag{4-17}$$

$$u(r,\theta,r_0,\theta_0) = \left(-\frac{P}{\tau}\right) \cdot \left(-\frac{\ln(r)}{2\pi}\right) \iint_{A_0} dA_0 \tag{4-18}$$

Substituindo-se a Equação (4-9) no lugar de P, u será dado por:

$$u(r,\theta,r_0,\theta_0) = \frac{F}{\tau} \cdot \frac{\ln(r)}{2\pi} + C \tag{4-19}$$

Como no caso real não existe uma força infinita atuando em um ponto, mas sim uma força finita F distribuída em uma área A_0, na simetria circular desenvolvida nesse trabalho é considerado que a área A_0 é um círculo de raio a. Consequentemente, para posições onde r é menor do que a o deslocamento é um valor constante e conhecido, digamos u_0. Para posições onde a distância r é maior do que a a solução é dada pela Equação (4-19). Assim, a condição de continuidade da membrana impõe que quando $r = a$ o valor de $u = u_0$. Dessa maneira, é possível isolar a constante e descobrir seu valor:

$$u_0 = \frac{F}{\tau} \cdot \frac{\ln(a)}{2\pi} + C \tag{4-20}$$

$$C = u_0 - \frac{F}{\tau} \cdot \frac{\ln(a)}{2\pi} \tag{4-21}$$

Por fim, substituindo-se o valor da constante C encontrada na Equação (4-21) na Equação (4-19), é possível chegar a seguinte conclusão:

$$u = \frac{F}{\tau} \cdot \frac{\ln(r)}{2\pi} - \frac{F}{\tau} \cdot \frac{\ln(a)}{2\pi} + u_0 \tag{4-22}$$

Sabendo-se que o valor da tensão superficial é uma constante que depende da membrana utilizada, e como não foi aferida essa medida de uma membrana real, foi definido que a tensão superficial $\tau = 1/(2\pi)$. Usando-se a propriedade dos logaritmos, onde $\log(a) - \log(b) = \log(a/b)$, a Equação (4-22) é simplificada e adquiri o seguinte formato:

$$u = F \cdot \ln\left(\frac{r}{a}\right) + u_0 \tag{4-23}$$

Para definir uma única solução a condição de contorno deve ser imposta. Desta maneira, quando $r = 1$, o deslocamento é nulo, ou seja, $u = 0$. Essa imposição também nos permite obter o valor da força F aplicada na área A_0. Como logaritmo de 1 é igual a 0, a Equação (4-22) adquire a seguinte forma:

$$0 = \frac{F}{\tau} \cdot \frac{\ln(a)}{2\pi} + u_0 \tag{4-24}$$

$$-u_0 = \frac{F}{\tau} \cdot \frac{\ln(a)}{2\pi} \tag{4-25}$$

Realizando-se as devidas alterações e isolando-se a força F, teremos que:

$$F = -\frac{u_0 \cdot \tau \cdot 2\pi}{\ln(a)} \tag{4-26}$$

Lembrando-se que o valor da tensão superficial para membrana em questão foi definido como $\tau = 1/(2\pi)$, a equação final para força pode ser representada por:

$$F = -\frac{u_0}{\ln(a)} \tag{4-27}$$

Assim, dado o deslocamento u_0 que uma prova circular de raio a impõe sobre uma membrana, é possível calcular a forma da membrana pela Equação (4-23), sendo a força F dada pela Equação (4-27). Essas duas equações serão implementadas no simulador para deformar a membrana e para gerar as forças na interface háptica, conforme será discutido na seção 5.2 do próximo capítulo.

4.3 SEGUNDA SITUAÇÃO – BORDA FIXA ($b = 1$ *e* $u(b) = 0$).

Considere agora não mais uma membrana infinita, mas sim uma membrana de raio b finita. Para simplificar, defina b como sendo a unidade, de forma que essa condição possa ser relaxada caso seja necessário.

Agora a posição r_0, onde uma força infinita e pontual é aplicada pode ser diferente da origem. Assim, a deformação pode ocorrer em um raio de 0 a 1 dentro na membrana. Nesse caso, a solução pela *Função de Green* é dada por meio da Equação (5.2.36) também da referência DUFFY (2001), e é,

$$g(r,\theta,r_0,\theta_0) = -\frac{1}{4\pi}\ln\left[r^2 + {r_0}^2 - 2\cdot r\cdot r_0\cdot\cos\left(\theta-\theta_0\right)\right] + \frac{1}{4\pi}\ln\left[1+(r\cdot r_0)^2 - 2\cdot r\cdot r_0\cdot\cos\left(\theta-\theta_0\right)\right] \quad (4\text{-}28)$$

Utilizando-se novamente a Equação (4-15) para duas dimensões é possível encontrar o valor de u, onde:

$$u(r,\theta,r_0,\theta_0) = \iint_{A_0}\left(-\frac{P}{\tau}\right) g(r,\theta,r_0,\theta_0)\,dA_0 \quad (4\text{-}29)$$

Substituindo-se a Equação (4-28) em (4-29), surge a seguinte expressão:

$$u(r,\theta,r_0,\theta_0) = \iint_{A_0}\left(\frac{P}{\tau}\right)\cdot\left(-\frac{1}{4\pi}\ln\left[r^2 + {r_0}^2 - 2rr_0\cos\left(\theta-\theta_0\right)\right] + \frac{1}{4\pi}\ln\left[1+(rr_0)^2 - 2rr_0\cos\left(\theta-\theta_0\right)\right]\right)dA_0 \quad (4\text{-}30)$$

Para o caso de uma força pontual, existe um resultado diferente de zero dessa integral apenas onde a força aplicada é diferente de zero. Como nesse ponto é utilizada a função *Delta de Dirac* no lugar da força, apenas em um ponto essa integral é diferente de nula. Logo, a solução da integral é o próprio integrando multiplicado pela área A_0, ou seja:

$$u(r,\theta,r_0,\theta_0)=\left(\frac{P}{\tau}\right)\cdot\left(-\frac{1}{4\pi}\ln\left[r^2+{r_0}^2-2\cdot r\cdot r_0\cdot\cos\left(\theta-\theta_0\right)\right]+\frac{1}{4\pi}\ln\left[1+(r\cdot r_0)^2-2\cdot r\cdot r_0\cdot\cos\left(\theta-\theta_0\right)\right]\right)\iint_{A_0}dA_0 \qquad (4\text{-}31)$$

Por fim, o resultado final dessa integral será:

$$u(r,\theta,r_0,\theta_0)=\left(\frac{F}{\tau}\right)\cdot\left(-\frac{1}{4\pi}\ln\left[r^2+{r_0}^2-2\cdot r\cdot r_0\cdot\cos\left(\theta-\theta_0\right)\right]+\frac{1}{4\pi}\ln\left[1+(r\cdot r_0)^2-2\cdot r\cdot r_0\cdot\cos\left(\theta-\theta_0\right)\right]\right) \qquad (4\text{-}32)$$

Mas como no caso anterior, no caso real não temos uma força infinita atuando em um ponto, mas sim uma força finita F distribuída em uma área A_0. A força real não é infinita e nem pontual. Pode ser utilizada a solução (4-31) apenas para posições diferentes de onde a força é aplicada, ou seja, fora da região A_0. Na posição onde a força atua o deslocamento é definido pela prova. Sabendo-se que a prova utilizada é plana, ou seja, $u=$ constante e considerando-se $u=u_0$ em $u=a$, é possível encontrar a força F naquele ponto da deformação:

$$u_0(a,\theta,r_0,\theta_0)=\frac{F}{\tau}\cdot\frac{1}{4\pi}\ln\left[a^2+r_0-2\cdot a\cdot r_0\cdot\cos(\theta-\theta_0)\right]+\frac{1}{4\pi}\ln\left[1+\left(a\cdot r_0\right)^2-2\cdot a\cdot r_0\cdot\cos\left(\theta-\theta_0\right)\right] \qquad (4\text{-}33)$$

$$F=\frac{\tau\cdot u_0}{\frac{1}{4\pi}\ln\left[a^2+r_0-2\cdot a\cdot r_0\cdot\cos(\theta-\theta_0)\right]+\frac{1}{4\pi}\ln\left[1+\left(a\cdot r_0\right)^2-2\cdot a\cdot r\cdot_0\cos\left(\theta-\theta_0\right)\right]} \qquad (4\text{-}34)$$

Novamente, dado o deslocamento u_0 que uma prova circular de raio a impõe sobre uma membrana, é possível calcular a forma da membrana pela Equação (4-32), sendo a força F dada pela Equação (4-34). Essas duas equações são utilizadas no simulador para deformar

a membrana e para gerar as forças na interface háptica, conforme será discutido na seção 5.3 do próximo capítulo.

O diagrama da Figura 44 representa de maneira resumida as etapas para o desenvolvimento do método, a implementação das equações para a realização da deformação u e para a força F, até a obtenção das sensações táteis e visuais.

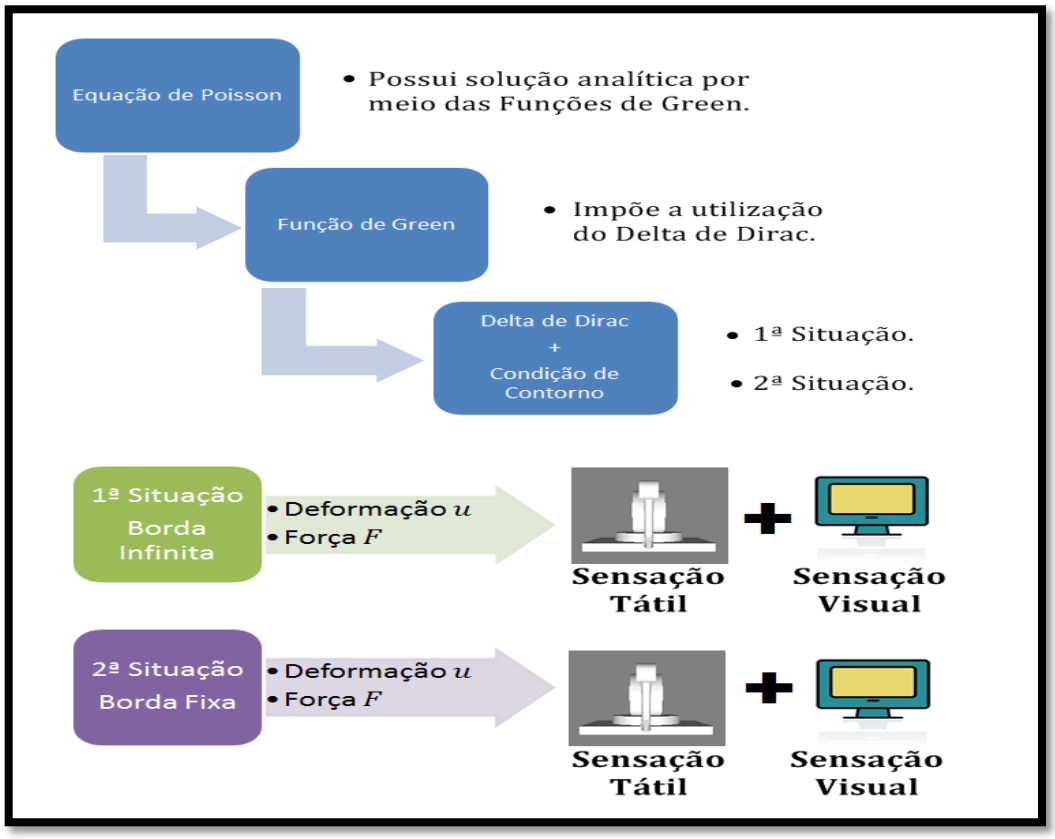

Figura 44 – Diagrama com os passos para o desenvolvimento do método, da implementação da deformação u, da força F, e obtenção das sensações táteis e visuais.

5 RESULTADOS E DISCUSSÕES

Nesse capítulo estão descritos os passos e os resultados alcançados com a modificação das linhas de códigos escritas em C++ de um dos projetos exemplo da plataforma CHAI 3D, bem como a validação do método desenvolvido no capítulo 4. Essa validação é mostrada por meio da realização de uma comparação de resultados tomados na literatura da deformação real de uma membrana de borracha homogênea com a simulação obtida com o CHAI 3D.

5.1 INTRODUÇÃO

Com o objetivo de implementar as equações de deformação de membrana em uma plataforma de simulação gráfica com retorno de força, optou-se por desenvolver esse trabalho com a plataforma CHAI 3D, pois essa já apresenta várias funcionalidades gráficas e hápticas, o que por sua vez também nos permite poupar tempo e trabalho na implementação.

Como o estudo em questão trata da deformação de membranas e devido ao fato de que dentre os códigos fonte exemplos fornecidos pela plataforma CHAI 3D, um desses códigos continha uma superfície que se deformava ao ser tocada pela interface háptica, optou-se por realizar a modificação das linhas de código desse projeto, denominado: "20 – map".

Resumidamente, esse exemplo gera um mapa de relevo a partir de uma imagem bitmap[6] 2D (imagem fixada sobre a superfície como uma textura[7]), convertendo-se cada *pixel*[8] colorido da imagem em um valor de altura. Esses *pixels* são posteriormente convertidos em uma matriz de triângulos que possibilita produzir uma malha virtual que pode ser deformada. Ao manipular o dispositvo háptico, o usuário pode livremente girar a câmera e navegar ao redor do objeto (mapa com relevo). Ao tocar a superfície do mapa e, simultaneamente pressionar o botão *switch* na caneta do dispositivo, o usuário pode modificar localmente a

[6] Imagens que contêm a descrição de cada *pixel*. Fonte: (LOPES, 2003, p. 13).

[7] Variações tonais repetitivas e organizadas que podem ser distinguidas em uma pequena região de uma imagem. Fonte: (LIBERMAN, 1997, p. 28).

[8] A menor parte do Monitor de vídeo é um pequeno ponto quadrado ou retangular. Os *pixels* podem ser uma combinação de duas palavras comuns, imagem e elemento. Um *pixel* é melhor descrito como uma unidade lógica, em vez de física, como o tamanho físico de um *pixel* individual é determinado pelo fabricante do monitor. Tamanho do *pixel* é medido em milímetros (mm). Fonte: (MIOT; PAIXÃO; PASCHOAL, 2006, p. 174-75).

topologia do mapa. Uma linha magnética virtual também é usada para guiar o usuário ao longo de um segmento vertical, durante o processo de deformação dessa malha virtual.

Para acessar o arquivo original sem que o mesmo tenha sofrido modificações em sua estrutura, inicie o *MSV2008* e siga os passos da seção 3.2 do capítulo 3 para encontrar os projetos. Depois que o programa estiver carregado, navegue até o arquivo "20-map.cpp" e dê um duplo clique com o botão esquerdo do *mouse*, a Figura 45 mostra o projeto "20 – map" selecionado, ao lado direito da figura podem ser notados trechos das linhas de código desse projeto em específico, linhas que podem ser alteradas de acordo com a necessidade do programador.

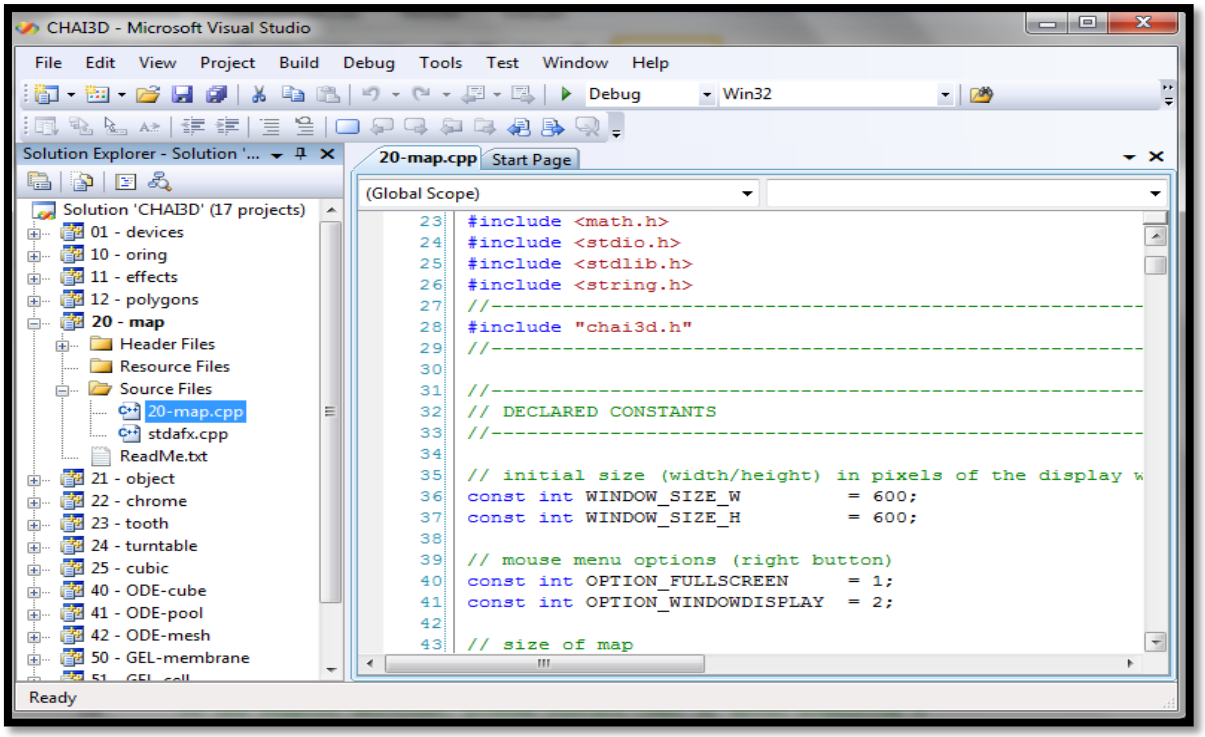

Figura 45 – Janela do MVS2008, em destaque no lado esquerdo da figura o painel "*Solution Explorer*" que mostra o projeto "20 – map" selecionado e a direita as linhas de código desse projeto.

Após selecionar o projeto "20 – map" e executar a simulação para o código em questão, surgirá na tela uma mapa com uma superfície irregular, assim como pode ser notado na Figura 46(a). A esfera branca (controlada pela caneta do dispositvo háptico), identifica as ondulações nessa superfície, o que faz com que o usuário do dispositivo tenha a sensação de que realmente está tocando nessas ondulações do mapa. Além disso, o usuário pode remover a textura (pressionando a tecla número 1 no teclado do computador) e visualizar a superfície com o recurso gradeado (malha de triângulos) acionado (pressionando a tecla número 2 no

teclado do computador), pressionando-se as mesmas teclas esses recursos são desativados, veja o resultado da ativação simultanea dos dois recursos na Figura 46(b) e (d). Quando o botão *switch* é pressionado (botão azul escuro da caneta do PHANTOM Omni®, veja a Figura 8 do capítulo 3), a esfera branca muda de cor e fica cinza, quando o mesmo botão é pressionado junto à superfície, outras duas esfera surgem, agora na cor vermelha, uma dessas esferas está localizada na parte superior e a outra outra está localizada na parte inferior do mapa, sendo que elas estão ligadas por uma linha branca, como mencionado acima, o que por sua vez, guia o usuário na sua trajetória vertical. Nessa linha branca surge uma força magnética que é transferida por meio do dispositivo háptico ao usuário, à medida que caneta do dispositivo é forçada para cima, o plano sofre alterações em seu relevo criando cristas, e à medida que a caneta do dispositivo é forçada para baixo, o plano sofre alterações em seu relevo criando vales, os resultados podem ser observados na Figura 46(c). Quando o botão *switch* é pressionado fora da superfície do mapa, é ativada a movimentação da câmera, o que nos faz crer que o mapa está sendo movimentado para cima e para baixo, além de poder também rotacioná-lo.

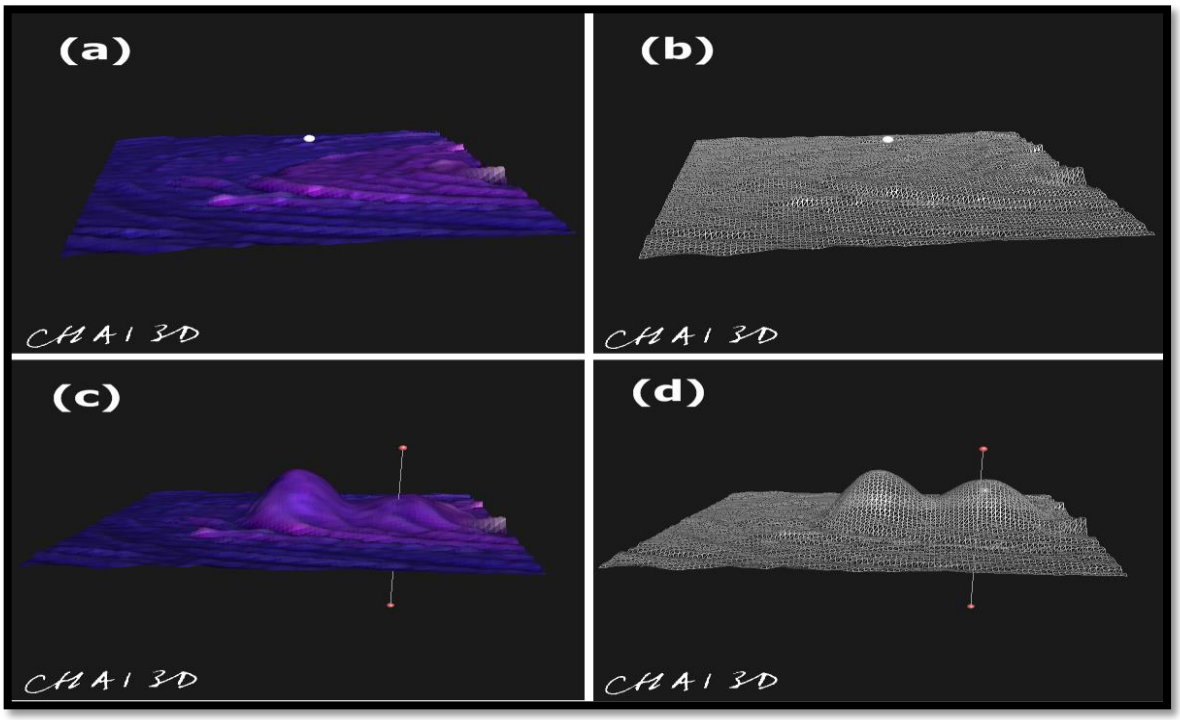

Figura 46 – Simulação do projeto "20 – map" sem modificações em suas linhas de código. (a) Superfície inicial da simulação. (b) Superfície inicial com o recurso textura desativado e gradeado acionado. (c) Superfície com alterações realizadas pelo usuário. (d) Superfície inicial com o recurso textura desativado e gradeado acionado. (c) Superfície com alterações realizadas pelo usuário. (d) Superfície com alterações realizadas pelo usuário, além do recurso textura desativado e gradeado acionado.

Fonte: CHAI 3D.

Realizadas as discussões a respeito das funções para esse exemplo em específico, podemos agora trabalhar nas modificações preliminares para implementação da primeira situação e posteriormente, da segunda situação, caracterizadas no capítulo 4 dessa dissertação. É importante mencionar que essas modificações preliminares servirão tanto para primeira, como para a segunda situação, sendo necessário alterar apenas as equações para cada uma delas.

Dentre as modificações[9] mais evidentes realizadas no código original estão: a tradução de todos os comentários, tradução de algumas variáveis declaradas, remoção das linhas magnéticas e de suas propriedades do código, e por fim, a remoção dos vários estados de manipulação do mapa, para apenas dois estados, rígido ou deformável.

O APÊNDICE B explica de maneira detalhada os procedimentos para que você possa executar o código alterado no seu computador, desde que todos os *hardware* e *software* mencionados no capítulo 3 estejam instalados e funcionando corretamente.

5.2 IMPLEMENTAÇÃO DA PRIMEIRA SITUAÇÃO – ESPAÇO LIVRE

A primeira situação ocorre para uma membrana que possui área infinita para deformação.

Localize o trecho do código referente às alterações para a primeira situação de deformação da membrana (linhas 380 a 407 do corpo do código):

```
/* ========================================================================== */
/* === ROTINA PARA DEFORMAÇÃO DA MEMBRANA - 1ª SITUAÇÃO (ESPAÇO LIVRE) === */
/* ========================================================================== */
```

Primeiramente, retire os comentários das linhas 383 e 405 para tornar esse trecho do código visível no momento da compilação, deixando todas as linhas ativas para essa rotina.

De início, as principais alterações no código podem começar com a inserção do valor do módulo de r, onde:

[9] Vide APÊNDICE D para ter acesso a todas as modificações realizadas no código original.

```
double r = sqrt(cSqr(posVertex.x-posicao.x)+cSqr(posVertex.y-posicao.y));
```

Além disso, pode ser inserida também a seguinte linha de código para evitar que um disco maior do que a própria membrana possa ser criado no momento da deformação, fazendo com que toda a membrana seja movimentada no eixo vertical e nenhuma deformação possa ser notada:

```
if (r < a) u = u0;
```

As equações necessárias à implementação do método para a primeira situação são as equações (4-23), que se refere ao cálculo da posição da membrana no momento da deformação e (4-27), que tem como resultado o cálculo da força F aplicada em um ponto qualquer da membrana. Como a deformação não pode ser realizada sem que uma força seja aplicada em um ponto da membrana, primeiro deve-se implementar a linha de código referente ao cálculo da força, onde:

```
F = -u0/log(a);
```

Seguindo-se a lógica, obtido o valor da força aplicada, pode ser implementada a Equação (4-23), a qual realizará a deformação da membrana por meio da linha de código que segue abaixo:

```
u = F*log(a/r)+u0;
```

O resultado final com a simulação da primeira situação implementada por meio da plataforma CHAI 3D pode ser observado na Figura 47.

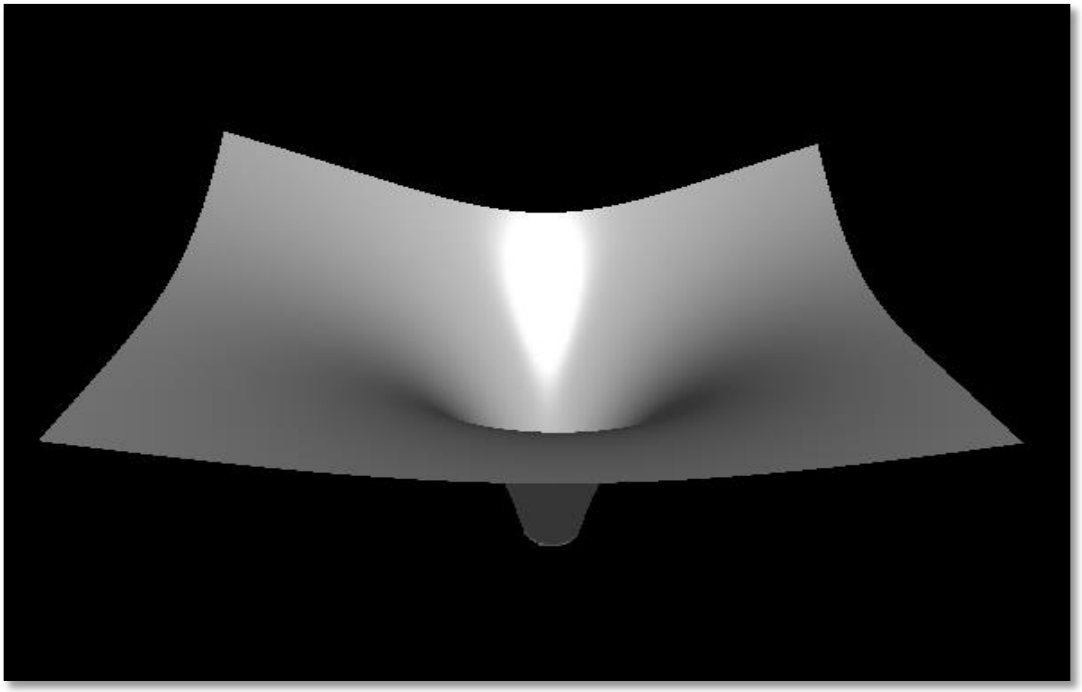

Figura 47 – Resultado da simulação da situação para espaço livre.

5.3 IMPLEMENTAÇÃO DA SEGUNDA SITUAÇÃO – BORDA FIXA

Localize o trecho do código referente às alterações para a segunda situação de deformação da membrana, descrito logo abaixo da rotina para a primeira situação (linhas 410 a 488 do corpo do código):

```
/* ========================================================================== */
/* ==== ROTINA PARA DEFORMAÇÃO DA MEMBRANA - 2ª SITUAÇÃO (BORDA FIXA) ==== */
/* ========================================================================== */
```

Novamente, retire os comentários das linhas 413 e 486 para tornar esse trecho do código visível no momento da compilação. Após a declaração de todas as variáveis, deve ser realizada a primeira alteração necessária à definição da borda da membrana, que no caso, possui diâmetro igual a (1,0), e como esse número é adimensional, ele não possui uma unidade física que o defina. Para a inserção desse trecho de código é necessário levar em conta as coordenadas de cada vértice (vertex) e cada ponto (posição) para as várias

coordenadas de x e y espalhadas pela membrana. Dessa maneira, a linha de código abaixo deverá ser criada:

```
if(sqrt(cSqr(posVertex.x) + cSqr(posVertex.y)) > 1.0 || sqrt(cSqr(posicao.x) +
cSqr(posicao.y)) > 1.0) u = 0.0;
```

assim, se a soma da raiz quadrada dos vértices x e y forem maiores que 1,0 ou a soma da raiz quadrada das coordenadas x e y forem maiores que (1,0), a deformação não ocorrerá e a membrana ficará rígida, retornando apenas a força da colisão de dois objetos rígidos para o dispositivo, sem que ocorra nenhuma deformação. O resultado da implementação dessa linha de código pode ser notado na Figura 48, onde um grande círculo é formado na membrana, na Figura 48(a) nota-se que não existe nenhuma deformação, mesmo que uma força seja exercida sobre aquela região, enquanto que na Figura 48(b), pode ser notada uma deformação na região interna delimitada pelo círculo branco[10]. Na região interna do círculo é possível realizar a deformação da membrana, enquanto que na região externa do círculo (as quatro bordas do plano) não é permitido executar a mesma ação, ou seja, mesmo que a prova seja pressionada sobre a membrana, exercendo uma força sobre essas regiões, elas permanecem rígidas, o que por sua vez, delimita o campo de ação das deformações e também a nossa borda fixa.

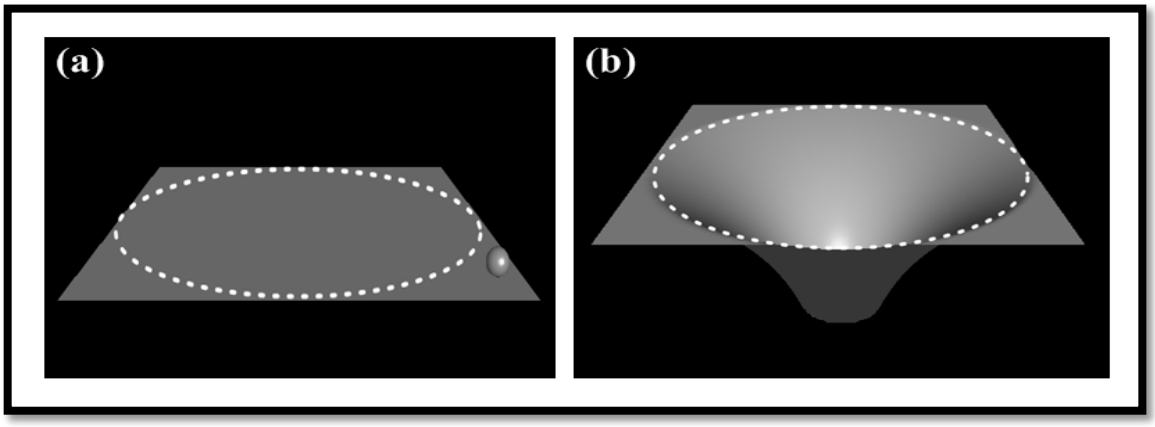

Figura 48 – Simulação com a implementação da borda fixa para a membrana. (a) Prova sendo pressionada contra a membrana, fora do círculo. (b) Prova sendo pressionada contra a membrana, dentro do círculo.

[10] O círculo branco foi inserido na figura através da edição da imagem para dar noção da região delimitada pela borda fixa e que não aparece na membrana quando a mesma não está sendo deformada.

Logo abaixo da linha de código referente à borda fixa, encontra-se a rotina para deformação da membrana, que inicia-se com 6 condições que irão realizar a correção de dois parâmetros para o funcionamento correto da deformação. Esses dois parâmetros são: a aquisição dos valores para os vários ângulos θ (variáveis teta0, teta e tetaa) e também a alteração do quadrante onde está sendo realizada a deformação, de negativo para positivo com relação às mesmas variáveis.

Com relação a aquisição dos valores para os ângulos θ é necessário a utilização da função inversa da tangente: $\tan^{-1}$ ou $arctg(x)$. Como esse caso possui apenas as várias posições das coordenadas x e y que formam a membrana, será utilizado o resultado da fração entre essas duas coordenadas para encontrar os valores dos ângulos θ, dessa forma, o $arctg(y/x) = \theta$. Como não existe fração em que o numerador seja igual a zero, foi necessário criar uma condição para quando as coordenadas de y chegassem próximas do intervalo de $-0{,}01$ a $0{,}01$ o resultado da função inversa fosse diretamente igual a 0. Abaixo seguem as linhas de código para implementação de cada uma das três condições referente à função inversa da tangente, que apesar de na rotina não se encontrarem dispostas uma abaixo da outra, aqui estão organizadas dessa maneira para dinamizar as explicações.

Implementação da condição para teta0:

```
if(posicao.y > 0.01 || posicao.y < -0.01)
{
	teta0 = atan((posicao.y)/( posicao.x));
}
else
{
	teta0=0.0;
}
```

Implementação da condição para teta:

```
if(posVertex.y > 0.01 || posVertex.y < -0.01)
{
	teta = atan((posVertex.y)/(posVertex.x));
}
else{
	teta=0.0;
}
```

Implementação da condição para tetaa:

```
if(ya > 0.01 || ya < -0.01)
{
```

```
        tetaa = atan((ya)/(xa));
}
else{
        tetaa=0.0;
}
```

Após a implementação dessas correções a simulação apresentará um novo problema ocasionado pela utilização da função inversa da tangente, Figura 49(a), que por sua vez, acarretará na duplicidade e na inversão dos fatos ocorridos nos quadrantes do plano, observe o resultado gerado na simulação por meio da Figura 49(b) após a inserção dessa linha de código, que realmente representa o que acontece com essa função, como pode ser notado na Figura 48(a).

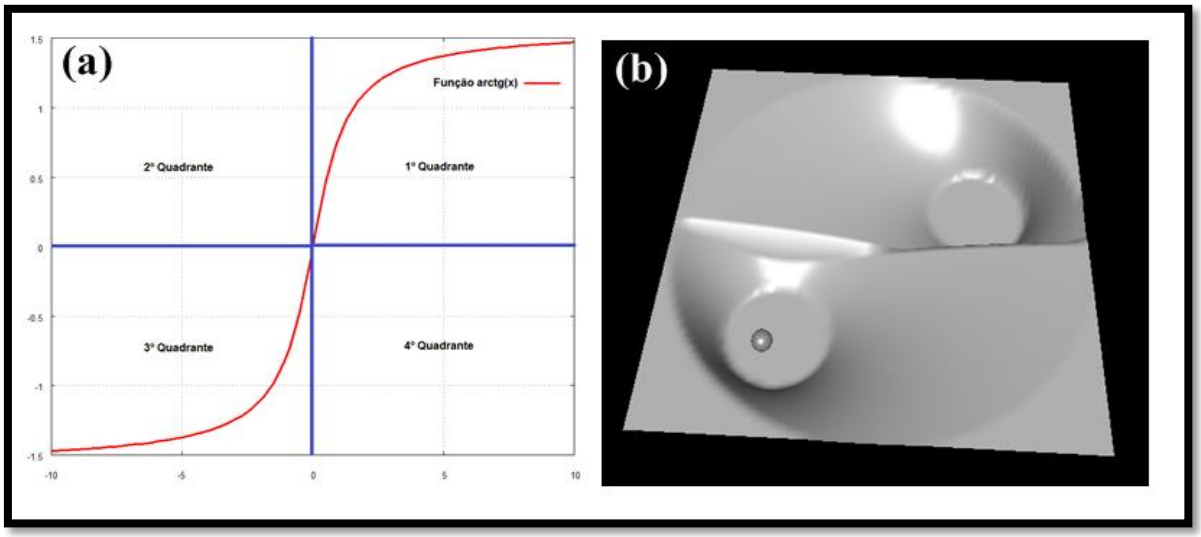

Figura 49 - Resultado da implementação da função inversa da tangente para gerar os valores dos ângulos θ para (teta0, teta e tetaa). (a) Gráfico de uma função $arctg(x)$ gerado através do aplicativo Gnuplot. (b) Resultado da inserção da função inversa na simulação.

O procedimento para correção desse problema passa pela inserção das seguintes linhas de código:

```
if(deslocamento.x < 0.0) teta0=teta0+Pi;
if(posVertex.x < 0.0) teta=teta+Pi;
if(xa < 0.0) tetaa=tetaa+Pi;
```

Sendo que cada linha representa a correção para um valor de θ utilizado (teta0, teta e tetaa). Dessa maneira, sempre que o valor de x for menor do que zero, ou seja, um número negativo,

será necessário acrescentar π radianos ao valor dos respectivos ângulos θ para que o mesmo possa ser representado por um valor de x positivo. Realizada a inserção dessas linhas no código, esse problema será sanado, no entanto, resta discutir a implementação das equações que realizam a deformação da membrana, apesar de que as simulações mostradas nas Figuras 48 e 49 já estão com essas equações implementadas.

Para que a deformação da membrana com a segunda situação possa acontecer, resgate as equações 4-32 e 4-34, que são as equações fundamentais para finalizar a implementação desse novo caso, onde:

$$u(r,\theta,r_0,\theta_0)=\left(\frac{F}{\tau}\right)\left(-\frac{1}{4\pi}\ln\left[r^2+r_0{}^2-2rr_0\cos\left(\theta-\theta_0\right)\right]+\frac{1}{4\pi}\ln\left[1+(rr_0)^2-2rr_0\cos\left(\theta-\theta_0\right)\right]\right) \quad (4\text{-}32)$$

é a equação para gerar a deformação da membrana e,

$$F=\frac{\tau u_0}{\frac{1}{4\pi}\ln\left[a^2+r_0-2ar_0\cos(\theta-\theta_0)\right]+\frac{1}{4\pi}\ln\left[1+\left(ar_0\right)^2-2ar_0\cos\left(\theta-\theta_0\right)\right]} \quad (4\text{-}34)$$

é a equação para calcular a força F que está sendo aplicada em uma determinada região da membrana.

Como uma maneira de facilitar nossas equações, defina o termo dentro do segundo parêntese após a igualdade na Equação (4-32) como sendo g, o que resultará na seguinte linha de código:

```
g = - 1.0/4.0/Pi*(-log(cSqr(r)+cSqr(r0)-2.0*r*r0*cos(teta-
teta0))+log(1.0+cSqr(r*r0)-2.0*r*r0*cos(teta-teta0)));
```

no entanto, para que u seja calculado, é necessário que encontrar primeiramente o valor da força F que está sendo aplicada no momento da realização da deformação da membrana. Para tanto, é necessário tomar $u=u_0$, como explicado na seção 4.2 do capítulo 4, fazendo uso da Equação 4-34, é possível definir uma variável chamada ga, que se refere ao denominador da fração existente nessa equação, como resultado surge a seguinte linha de código:

```
ga = 1.0/4.0/Pi*(-log(cSqr(ra)+cSqr(r0)-2.0*ra*r0*cos(tetaa-
teta0))+log(1.0+cSqr(ra*r0)-2.0*ra*r0*cos(tetaa-teta0)));
```

note que essa linha de código é muito semelhante a linha anterior, porém, observe que o valor de r foi substituído por xa, onde xa representa o vetor posição das coordenadas x e y das bordas do disco a, que está sendo pressionado contra a membrana para gerar a deformação. Dessa maneira, ra é definido como:

```
ra=sqrt(cSqr(xa)+cSqr(ya));
```

onde, xa e ya dá o tamanho da prova utilizada. Na implementação em questão foi utilizado o valor padrão $0{,}02$ como sendo o raio da prova, que pode ser facilmente alterado de acordo com os objetivos que se deseja alcançar na simulação.

Sabendo-se que para esse caso o coeficiente de tensão superficial τ foi definido como sendo $1{,}0/2{,}0/\pi$, pode finalmente ser implementada a Equação (4-34) e então criar a linha de código abaixo para calcular a força F aplicada na deformação da membrana:

```
F = u0*Tau/ga;
```

Após calcular a força F, é possível também calcular a deformação gerada por u, implementando a Equação (4-32) por meio da linha de código descrita abaixo:

```
u = F/Tau*g;
```

Para finalizar a implementação dessa situação, deve-se acrescentar mais uma linha de código fazendo com que a membrana não siga sendo deformada até o infinito, como pode ser observado na Figura 50(a). Como o intuito é simular uma membrana real sendo deformada, a imposição de que sempre que os valores de u (deformação) forem menores que u_0 (a altura do disco a com relação à posição inicial da membrana), faça com que u seja igual a $u0$ com o auxílio da seguinte linha:

```
if (u < u0) u = u0;
```

por meio dessa imposição, a posição final do disco a será também a posição final de deformação da membrana. O resultado final da implementação de todas as equações e todas as correções podem ser contempladas na Figura 50(b).

Até o momento foi discutido apenas os resultados da implementação das equações para criar a deformação da membrana, porém, outro ponto bastante importante foi a decomposição das forças que o dispositivo lê e envia ao usuário que está manipulado a caneta do *phantom*. No código inicial, essa função era feita pela linha abaixo:

```
cVector3d offset = toolGlobalPos - prevToolGlobalPos;
```

onde, cVector3d é uma das classe do CHAI 3D que armazena o deslocamento nos eixo x, y e z feitos pela caneta do *phantom* no ambiente virtual, levando em conta a posição final menos a posição inicial da prova (esfera). Como é necessário saber apenas a posição exata da prova, foram removidos os termos após a igualdade e alterado o nome *offset* (deslocamento) para eixo e em seguida, esse vetor nomeado eixo foi decomposto em três variáveis diferentes como pode ser observado nas linhas abaixo:

```
cVector3d eixo (0,0,0);
eixo.x=0.0;
eixo.y=0.0;
eixo.z= -F*20.0;
```

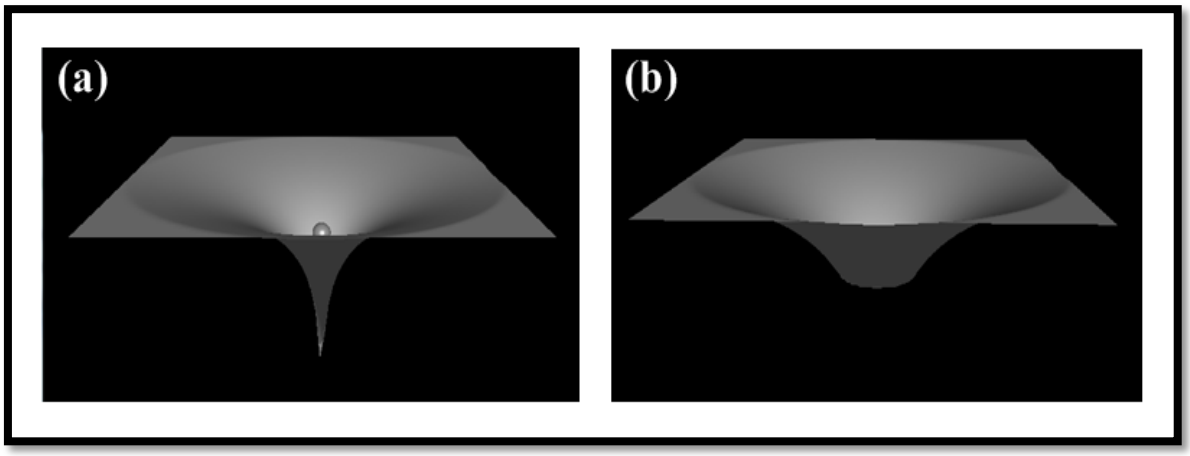

Figura 50 - Simulação com o resultado da implementação da linha que faz com que a membrana não vá até o infinito. (a) Membrana sendo deformada e indo até o infinito. (b) Membrana sendo deformada e com um corte na altura do disco $\boldsymbol{a}$.

Tendo em vista que não é necesário saber quais são as forças enviadas pelos eixos x e y, esses eixos enviam forças nulas para o dispositivo, ou seja, forças iguais a 0. Já o eixo z (eixo vertical) leva em consideração a força F calculada por meio da Equação (4-23) para a

primeira situação ou da Equação (4-34) para a segunda situação, sendo acrescentado ainda um fator qualquer para amplificar o retorno de força do dispositivo, onde nessa ocasião foi o fator 20.

Como essas alterações estão dispostas no corpo do código logo abaixo do final da implementação da segunda situação, esse trecho vai estar ativo tanto para a primeira como para a segunda situação.

Para finalizar, após a decomposição dos eixos, são acrescentadas duas linhas que tem como função fazer com que a prova (esfera) não fique ligada a membrana mesmo quando sua posição está localizada acima da membrana. As linhas de código que comandavam essa tarefa estavam ligadas aos vários estados de manipulação do projeto antigo, como eles foram retirados, foi necessário criar as seguintes linhas de código:

```
if (posEsfera.z <= 0.0)
phantom->setForce(eixo);
```

Essas duas linhas identificam a posição do phantom ou prova (esfera), se essa posição for menor ou igual a zero, o phantom lê a posição da prova e envia as forças exercido nesse ponto da membrana para o dispositivo, que por sua vez exerce uma força contrária ao do movimento realizado e que pode ser sentida pelo usuário.

5.4 VALIDAÇÃO DO MÉTODO

A validação do método tem como ponto de partida o trabalho de Selvadurai (SELVADURAI, 2006). Em seu trabalho, Selvadurai realizou experimentos reais e computacionais (simulação) com o intuito de observar o comportamento da deformação de membranas de borracha constituídas de material homogêneo, ou seja, que possuem mesma elasticidade e espessura. Além disso, foram realizadas comparações com vários outros modelos disponíveis na literatura. No entanto, apenas os resultados referentes ao experimento nos são úteis aqui.

No experimento, uma membrana de borracha é fixada em um painel retangular que possui um círculo no centro, isto é, a região compreendida entre o centro e a borda dessa

circunferência pode ser deformada, pois essa é a região em que é possível exercer uma pressão sobre a membrana alongada. A membrana de borracha é deformada à medida que uma esfera metálica (prova) é pressionada contra a sua superfície no sentido de cima para baixo. Simultaneamente à realização do experimento, são registradas por meio de fotografias, várias posições da prova que é pressionada contra essa membrana para posteriores análises quantitativas.

A câmara digital foi montada rigidamente no suporte de ensaio desenvolvido e fixada a uma distância de 1500 mm a partir do plano central do corpo de prova. Resumidamente, as imagens foram medidas em unidades de *pixels* e estes foram calibrados contra duas escalas que foram alinhados numa linha que passa através do plano de imagem e normal ao eixo da câmara digital. As escalas foram inseridas em cada lado da membrana e situadas à mesma distância do centro da membrana. Como pode ser observado na Figura 51, foram feitas três medições do grau de elasticidade da membrana, uma para cada posição da prova (esfera) que deforma a membrana. As posições são 25.4mm, 76.2mm e 127mm de distância com relação a superfície não deformada. Note que existem duas fotos capturadas para 25.4mm e 76.2mm, essas fotos duplicadas se devem ao fato de que uma delas foi capturada quando a deformação aumenta, ou seja, quando a prova está se movendo para baixo e a outra imagem é feita quando a deformação diminui, ou seja, quando a prova se move para cima. O ponto de 127mm é feito quando a membrana alcança a deformação máxima no experimento.

Além disso, na Figura 51, pode também ser observado que essa deformação foi realizada no centro da membrana. No entanto, Selvadurai realizou uma deformação semelhante também para uma posição diferente da região central da membrana, sendo que dessa vez a deformação foi realizada no espaço compreendido entre a borda e o centro da membrana, o resultado pode ser conferido na Figura 52.

A partir da análise dessas imagens foi possível construir um gráfico que demonstra o comportamento da membrana em que determinada pressão em Newtons realizada sobre a sua superfície, corresponde a uma determinada variação em milímetros, que por sua vez, produz uma deformação na sua estrutura flexível. Como o método em questão simula a deformação de uma membrana também circular e de borda fixa, seção 5.3, é possível também realizar uma comparação entre a deformação experimental realizada por Selvadurai (2006) e a simulação gerada na plataforma CHAI 3D.

Como temos conhecimento do diâmetro da membrana, que é de 250mm e das três posições às quais a membrana foi deformada (25.4mm, 76.2mm e 127mm), é possível montar

um gráfico com os dados exatos mostrados por Selvadurai (2006) em seu trabalho, uma vez que não foi possível ter acesso aos dados originais utilizados pelo próprio autor para construir os gráficos que demonstram o comportamento da deformação de sua membrana de borracha.

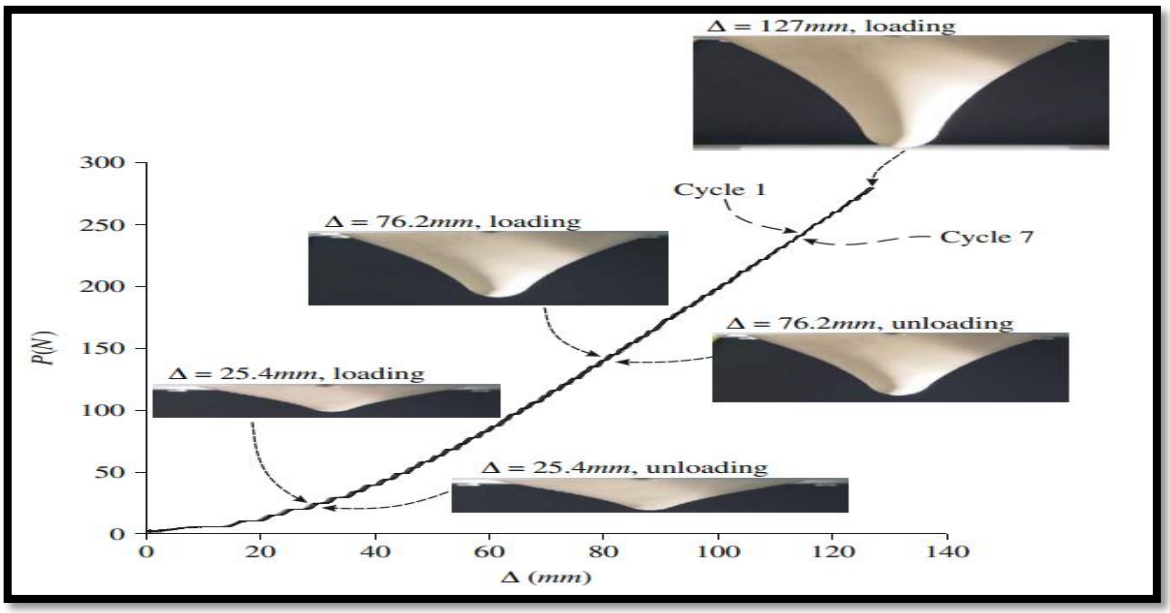

Figura 51 - Deformações na região central de uma membrana de borracha, relativas às três medições do grau de elasticidade da membrana de borracha.

Fonte: SELVADURAI, 2006, p. 1108.

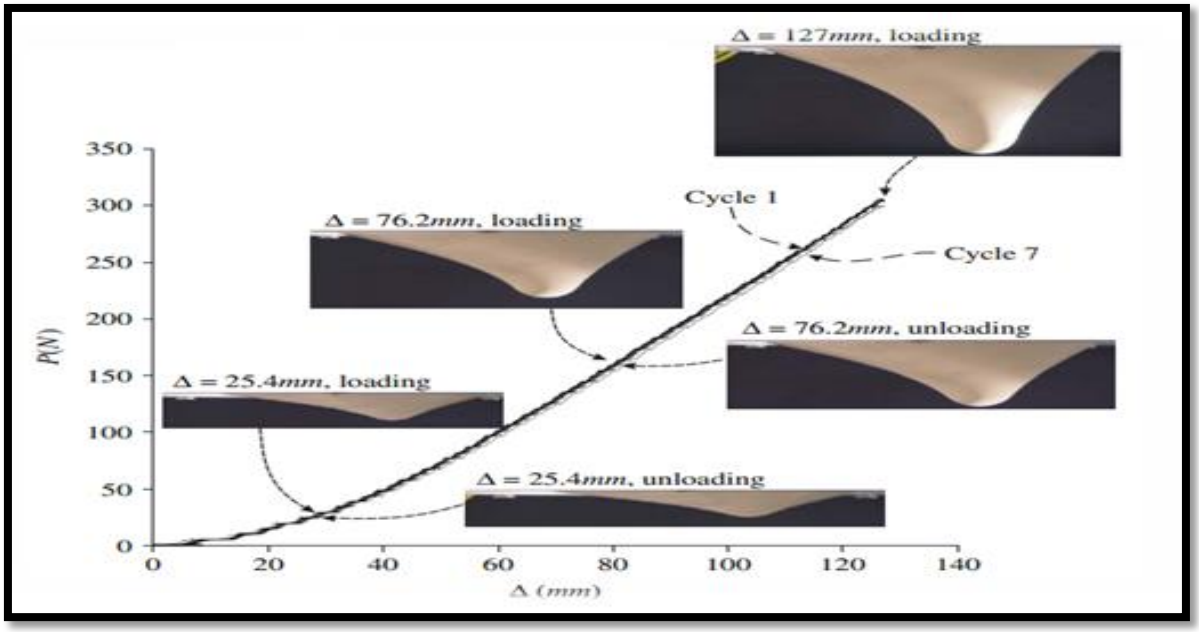

Figura 52 - Deformação na região compreendida entre a borda e o centro da mesma membrana de borracha, relativas às três medições do grau de elasticidade da membrana de borracha.

Fonte: SELVADURAI, 2006, p. 1109.

Uma explicação detalhada dos procedimentos para obtenção dos dados experimentais a partir do artigo de Selvadurai (2006) e também dos dados referentes à simulação podem ser consultados no APÊNDICE F dessa dissertação.

Depois de gerar vários arquivos com os dados referentes às posições x e y de cada imagem e também dos gráficos presentes no trabalho de Selvadurai, foi possível gerar dois gráficos. O primeiro gráfico plotado se refere à deformação realizada no centro da membrana, Figura 53 (Deformação Central), enquanto que o segundo gráfico se refere à deformação da membrana deslocada para região entre a borda o centro, Figura 54 (Deformação Deslocada).

Como pode ser observado nas Figuras 53 e 54, os dados referentes aos experimentos reais, membrana de borracha deformada no experimento de Selvadurai (2006) e os dados referentes à simulação na plataforma CHAI 3D realizados por meio da implementação do método aqui divulgado, possuem uma apreciável sobreposição, o que comprova que a simulação demonstra uma boa aproximação do comportamento de uma membrana real. Lembrando que não foram analisados os dados referentes às forças aplicadas na membrana, apenas os dados expostos na tela e que dão a sensação de estarmos manipulando uma membrana real.

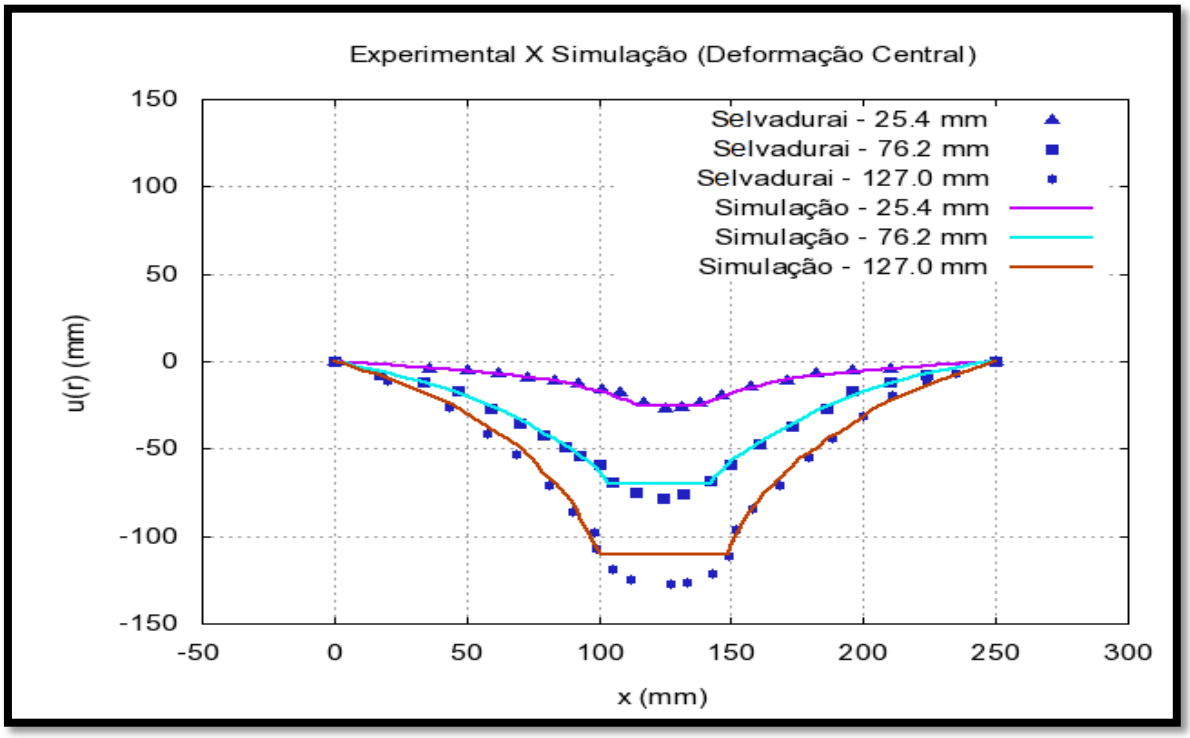

Figura 53 - Gráfico da comparação dos resultados obtidos pela deformação realizada no centro de uma membrana real (Selvadurai) e pela deformação na mesma posição de uma membrana em realidade virtual (Simulação).

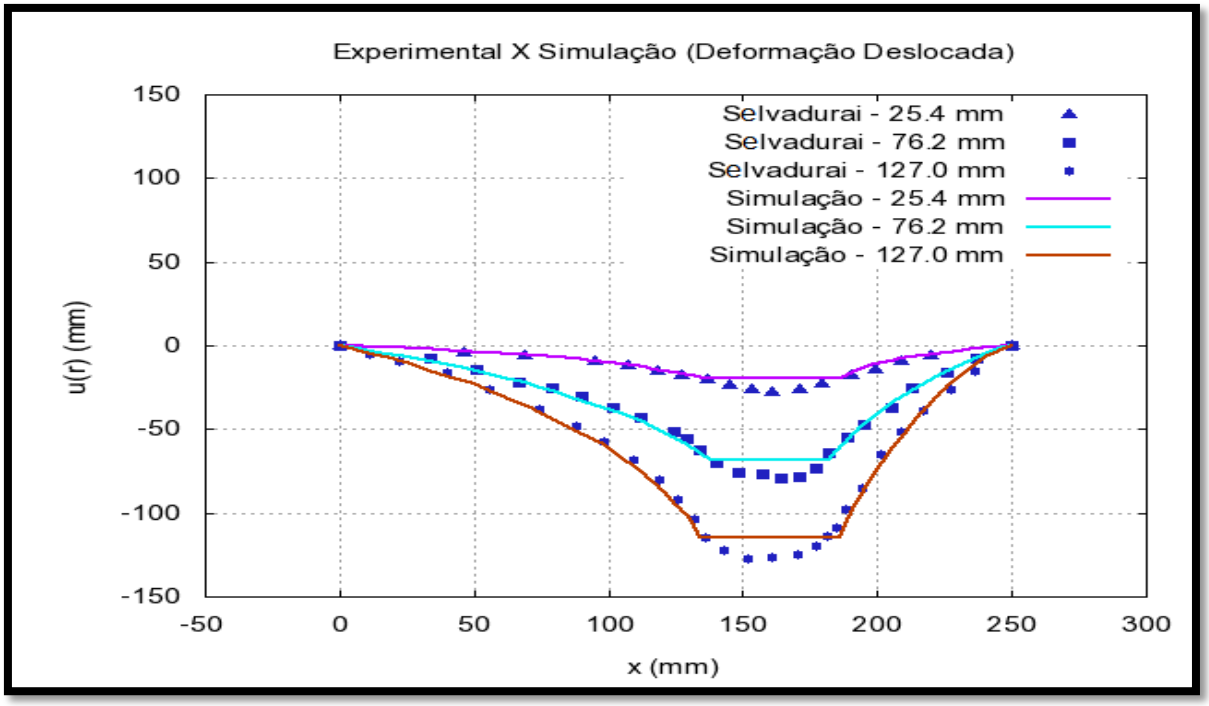

Figura 54 - Gráfico da comparação dos resultados obtidos pela deformação realizada próximo à borda de uma membrana real (Selvadurai) e pela deformação na mesma posição de uma membrana em realidade virtual (Simulação).

6 CONCLUSÕES

Neste trabalho, foi realizada com sucesso, a elaboração e a implementação de dois métodos para deformações de membranas homogêneas com a utilização da plataforma de simulação CHAI 3D e com o dispositivo de retorno de força PHANTOM Omni®, os quais possibilitam experimentem sensações visuais e táteis, respectivamente.

Tanto para a instalação da plataforma de simulação como para a instalação de *hardware* e *software*, dê preferência aos novos sistemas operacionais que possibilitam a instalação desses recursos de uma forma mais fácil e eficiente, como é o caso dos sistemas operacionais da família Windows®. Tanto a interface de retorno de força, como os *hardware* que possibilitam ligar a interface ao computador, bem como o *software* Microsoft® Visual Studio® 2008 instalaram muito bem nos sistemas operacionais Windows® Vista e Windows® Seven, e não apresentaram problemas na execução dessas ações e também em nenhum momento no decorrer da realização do trabalho em laboratório. Já o Windows® XP, em alguns momentos, apresentou lentidão e necessitou ser reiniciado à força, além de ser necessário procurar *drivers* de *hardware* que não se instalaram por conta próprio, como no caso das duas versões anteriormente citadas. A execução de todo o tutorial na sequência em que ocorre no capítulo 3 garantirá o sucesso quanto à utilização desses recursos nos seus estudos.

Apesar da linguagem de programação C++, que trabalha com classes de objetos, ser uma linguagem um tanto mais complicada de ser entendida do que a linguagem de programação C, não foi necessário aprofundar-se nesse tipo de linguagem, fato que proporcionou a escolha de um dos exemplos prontos do CHAI 3D, que por outro lado fez com que bastante tempo fosse poupado na elaboração de um novo código a partir do zero. No entanto, é necessário saber reconhecer pelo menos as estruturas básicas existentes na linguagem de programação C, por exemplo: conceito e definição de variáveis, conceito e condições para tomada de decisão, e também comandos para geração de rotinas repetitivas no corpo do código, ou seja, saber entender a lógica de programação em linguagem C. Outro fato fundamental relacionado à utilização da plataforma CHAI 3D é que ela já possui implementada a biblioteca gráfica do *OpenGL*®, quem de fato realiza as simulações apresentadas no monitor do computador, sendo novamente outro recurso que poupou bastante tempo com relação à programação.

Apesar de o tutorial da seção 3.3 explicar como podem ser acessadas e executadas as simulações do CHAI 3D de duas maneiras, é necessário estudar cada uma das simulações e entender como elas se relacionam, linha por linha de código, dando preferência aos códigos mais simples, que simulam esferas e objetos se movendo na cena virtual e depois passar a estudar códigos mais complexos, como os que realizam colisões, como os que possuem propriedades, tais como o magnetismo, ou aqueles que podem ser deformados. Dessa forma, além de se familiarizar com as linhas de código, também irá se familiarizar com as ferramentas do Visual Studio®.

A realização da deformação de objetos em ambiente virtuais é uma tarefa que requer computadores com grande capacidade de processamento, uma vez que muitos dos métodos empregados utilizam cálculos matemáticos complicados ou envolvem resolução numérica de matrizes como nos métodos usuais Massa Mola, Deformação de Livre Forma e Método dos Elementos Finitos, ocasionando em lentidão na obtenção de resultados finais da deformação, sendo que em algumas técnicas o resultado pode ser apresentado horas ou até dias depois. Nesse trabalho, a solução analítica implica em uma solução rápida, o que acarreta a deformação dessa membrana em tempo real por meio da plataforma de simulação. Além disso, as simulações podem ser executadas em computadores convencionais, como *notebooks* e *desktops*, desde que possibilitem a instalação dos *hardware* e *software* específicos.

A implementação da primeira e da segunda situação tiveram resultados bastante satisfatórios, principalmente a segunda situação, a qual foi comparada com a deformação de uma membrana de borracha real na literatura e obteve uma excelente aproximação, uma vez que os gráficos comparativos entre o experimento real e a simulação possuem perfeita simetria e sobreposição, tanto para deformações realizadas no centro da membrana, como para deformações realizadas entre a borda e o centro. Apesar de arduá, a obtenção dos dados para as deformações realizadas no experimento da literatura e para as deformações realizadas no CHAI 3D ocorreram por meio da utilização de dois *software*, o *Paint* do Windows® com suas ferramentas "régua" e "gradeado", que nos auxiliaram quanto ao ajuste das imagens para edição e com a ajuda do *software* Gnuplot, utilizado na geração dos gráficos comparativos entre os dois métodos e também na produção de algumas imagens do capítulo 4.

A presença dos computadores em nossas vidas tem possibilitado um crescente número de estudos envolvendo esse recurso, principalmente na realização de cálculos e simulações complexas que seriam impossíveis de serem realizadas sem tal ferramenta, o que por sua vez proporciona grandes avanços na área da tecnologia, visando cada vez mais conforto e

qualidade de vida, seja na área da educação, seja na área da medicina, seja no ramo da pesquisa.

6.1 TRABALHOS FUTUROS

Para os próximos passos, pretende-se finalizar a elaboração do método com a implementação da terceira situação. Essa última situação realizará a deformação de objetos virtuais que possuam volume constante, como é o caso de alguns órgãos do corpo humano, tais como: a bexiga, o estômago, os olhos, a mama, intestino, entre outros que possuem membranas flexíveis e que sejam repletos de fluidos, possibilitando dessa forma a sua deformação e a conversação do seu volume. Na plataforma de simulação esses órgãos poderão ser carregados como arquivos do tipo ".obj" ou ".3ds", criando dessa forma, bancos de dados que poderão ser carregadas de acordo com as necessidades do usuário, e que deverão ser utilizados efetivamente em diversas aplicações, como no planejamento de vários tipos de cirurgias, por exemplo: guiadas por imagem, minimamente invasivas e à distância. Além disso, também podem auxiliar na formação de médicos ou alunos de medicina.

Outro processo também bastante útil à medicina será o planejamento de cirurgias por meio da simulação, permitindo a visualização de resultados potenciais ou ajudar na tomada de decisões pré-operatória sobre as opções cirúrgicas. Por exemplo: na cirurgia plástica, a simulação pode ser útil para visualização do resultado final de uma cirurgia reconstrutiva da mama ou implantação de próteses de silicone, possibilitando dessa forma que o paciente veja o resultado final dessa plástica ou cirurgia.

Como visto, existem motivos de sobra para que possam continuar pesquisando e trabalhando na criação de novas tecnologias ligadas ao ensino e ao treinamento médico, sendo esses recursos meios que possibilitam cada vez mais uma maior qualidade no modo de vida da população em geral.

REFERÊNCIAS

CHAI 3D. Disponível em: <www.chai3d.org/>. Acessado em: 18 fev. 2013.

CAMPOS, F. S.; MACHADO, L. S. **Métodos de Deformação em Sistemas Interativos**. Departamento de Matemática / Departamento de Informática. Universidade Federal da Paraíba. João Pessoa – PB, Brasil, 2008.

COSTA, R. S.; COSTA, I. F. **Deformação de biomembranas com retorno de força para treinamento médico em realidade virtual**. XXIII Congresso Brasileiro em Engenharia Biomédica - CBEB, 2012.

DUFFY, D. G. **Green's functions with aplications p. cm. (Studies in advanced mathematics)**. 1 Green's functions. I Title. II. Series. Chapman & Hall/CRC. 2001.

FEYNMAN, R.P. Leighton, R.B and Sands, M. Sands, **The Feynman Lectures on Physics**, vol. 2 – APÊNDICE 1 - Análogos Eletrostáticos. (Addison-Wesley, Reading, MA 1963).

HAYWARD, V. ASTLEY, O. R. CRUZ-HERNANDEZ, M. GRANT, D. ROBLES-DE-LA-TORRE, G. **Haptic Interfaces and Devices.** Sensor Review. volume 24 · Number 1 · 2004 · pp. 16–29.

KIMER, C.; SISCOUTTO, R. **Realidade Virtual e Aumentada: Conceitos, projetos e Aplicações.** Livro do Pré-Simpósio – IX Symposium on Virtual and Augmented Reality. Petrópolis – RJ, 2007.

KÜLBERG, M.; OLIVEIRA, J. C. **Dental Procedure 3D Simulation Using a Hapic Device Software de Simulação de Procedimentos Dentários Utilizando Dispositivo Phantom Omni e Modelos 3D.** Laboratório Nacional de Computação Científica. Petrópolis- RJ, Brasil, 2012.

LIBERMAN, F. **Classificação de Imagens Digitais por Textura usando Redes Neurais.** 1997. 87 f. Dissertação submetida à avaliação para obtenção do grau de mestre em Ciência da Computação. Universidade Federal do Rio Grande do Sul. Instituto de Informática. Curso de Pós-Graduação em Ciência da Computação. Porto Alegre, 1997.

LOPES, J. M. B. **Formatos de Imagens.** 2003. Departamento de Engenharia Informática. Texto elaborado para a disciplina de Computação Gráfica. Licenciatura em Engenharia Informática e de Computadores Instituto Superior Técnico. Universidade Técnica de Lisboa. 2003, reeditado em Dezembro de 2008.

LOTTI, R. S. et al. **Aplicabilidade científica do método dos elementos finitos**. Rev. Dent. Press Ortodon. Ortop. Facial. v. 11, n. 2, p. 35-43, Maringá-PR, 2006.

MACHADO, L. S. **A Realidade Virtual no Modelamento e Simulação de Procedimentos Invasivos em Oncologia Pediátrica: Um Estudo de Caso no Transplante de Medula Óssea.** Tese de Doutorado – Departamento de Engenharia de Sistemas Eletrônicos. Escola Politécnica da Universidade de São Paulo, 2003.

MICROSOFT. Disponível em: <www.microsoft.com/brasil/servidores/64bit/overview.mspx>. Acessado em: 26 de jun. 2013.

MIOT, H. A.; PAIXÃO, M. P.; PASCHOAL, F. M. **Fundamentos da fotografia digital em Dermatologia.** An Bras de Dermatol. 2006; 81(2):174-80.

MOREIRA, L. B. et al. **Sistema Háptico para Corte Craniano**. Departamento de Ciência da Computação. Centro Universitário da FEI. São Bernardo do Campo – SP, Brasil, 2011.

NUMES, F. L. S.; MACHADO, L. S.; COSTA, R. M. E. M. **Aplicações de Realidade Virtual e Aumentada.** Capítulo 04: RV e RA Aplicadas à Saúde, em: XI SIMPÓSIO DE REALIDADE VIRTUAL E AUMENTADA. Porto Alegre-RS, Brasil, 2009.

OPENGL (Open Graphics Library). Disponível em: <www.opengl.org/documentation/current_version/>. Acessado em: 18. fev. 2013.

PAVARINI, L. **Estudo e Implementação do Método Massa-Mola para Deformação em Ambientes Virtuais de Treinamento Médico usando a API Java 3D.** Dissertação (Mestrado em Ciência da Computação) - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha. Marília, SP: [s.n.], 2006. 146 f.

PHANTOM Omni®. Disponível em: <www.sensable.com/haptic-phantom-omni.htm>. Acessado em: 15 fev. 2013.

RODRIGUES, H. F.; MACHADO, L. S.; VALENÇA, A. M. G. **Uma Proposta de Serious Game Aplicado à Educação em Saúde Bucal**. Departamento de Informática e Departamento de Clínica e Odontologia Social. Universidade Federal da Paraíba – UFPB. Cidade Universitária. João Pessoa – PB, Brasil, 2009.

SARAIVA, A. J. J. I. B. U. et al. **Sistema Integrado Para Incisão Cirúrgica com Bisturi Virtual Utilizando o Dispositivo Háptico Phantom Omni**. Trabalho de Conclusão de Curso - Centro Universitário da FEI. São Bernardo do Campo, 2010.

SELVADURAI, A.P.S. **Deflections of a rubber membrane**. Department of Civil Engineering and Applied Mechanics, McGill University, 817 Sherbrooke Street West, Montreal, Que., Canada H3A 2K6. Journal of the Mechanics and Physics of Solids nº 54. 2006. pgs. 1093–1119.

APÊNDICE A – Checar a Versão do Sistema Operacional do seu Computador

Os sistemas operacionais de 32 *bits* e de 64 *bits* se diferenciam um do outro, alguns aplicativos e *software* desenvolvidos para plataformas 32 *bits* podem funcionar em plataforma 64 *bits*, porém o inverso não é verdade. Os sistemas de 64 *bits* oferecem acesso direto a mais memória virtual e física do que os sistemas de 32 *bits* e processam mais dados por ciclo, permitindo soluções de computação mais escalonáveis e com maior desempenho (MICROSOFT, 2013).

Este APÊNDICE mostra os procedimentos para se checar qual é a versão do sistema operacional disponível no seu computador, desde que a mesma pertença à família *Windows*®.

A.1 VERSÃO DO SISTEMA OPERACIONAL

Em um dos sistemas operacionais da família *Windows*® (*XP*, *VISTA* ou *SEVEN*), vá até a "Área de Trabalho" e clique no ícone "Computador" ou "Meu Computador" com o botão direito do *mouse* e posteriormente em "propriedades" com o botão esquerdo do *mouse*. Caso não exista o ícone "Computador" ou "Meu Computador" na "Área de Trabalho" o mesmo resultado pode ser obtido ao clicar em "Iniciar" com o botão esquerdo do *mouse*, em "Computador" com o botão direito e posteriormente em "propriedades" com o botão esquerdo. Uma alternativa mais rápida e eficiente consiste apenas em pressionar simultaneamente as teclas "*Windows* + *Pause Break*" em qualquer janela que estiver.

Após clicar em "Propriedades" será aberta uma janela, nela estão listadas as informações a respeito do computador onde estiver trabalhando, inclusive em quantos bits o sistema operacional está sendo executado. Essa informação está destacada por meio do retângulo vermelho na Figura A.1, "Sistema Operacional de 32 bits", caso seja um sistema de 64 *bits* a descrição será então: "Sistema Operacional de 64 bits".

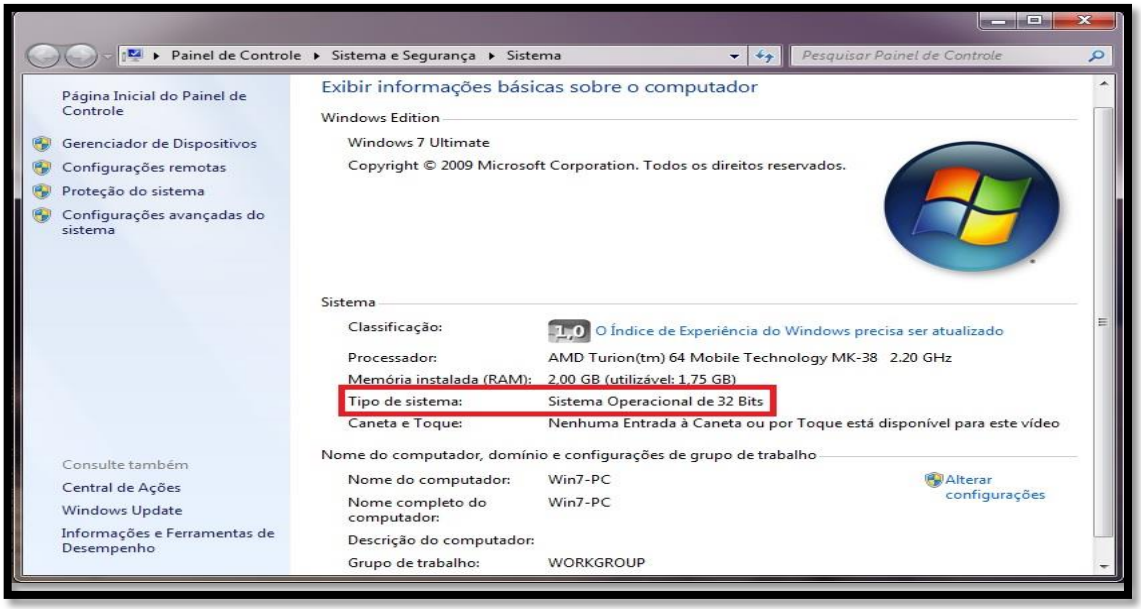

Figura A. 1 – Janela com informações básicas no *Windows*® *SEVEN*. Em destaque a quantidade de *bits* utilizados pelo sistema operacional.

Dessa forma, deve ser feito o *download* dos arquivos para *Windows*® 32 *bits*, como o programa não diferencia as versões do sistema operacional, esse mesmo pacote de arquivos funcionará nos sistemas operacionais *Windows*® *XP*, *VISTA*, *SEVEN* e outros da plataforma *Windows*®, desde que também operem em 32 *bits*.

No *Windows*® *XP* será necessário verificar na janela "Propriedades do Sistema" se existe a descrição "*Microsoft Windows*® *XP Profissional*" para sistemas que operam em 32 *bits* Figura A.2(a) ou "*Microsoft Windows*® *XP Profissional x64 Edition*" para sistemas que operam em *64 bits*, Figura A.2(b).

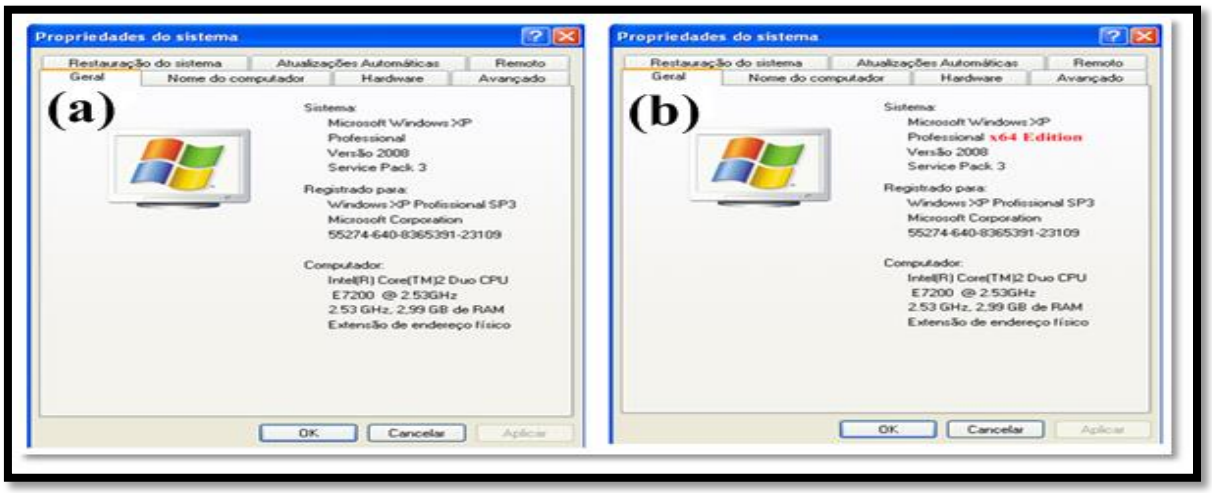

Figura A. 2 – Janela "Propriedades do Sistema" no *Windows*® *XP Profissional*. (a) – Sistema de 32 *bits*. (b) – Sistema de 64 *bits*.

APÊNDICE B – Descompactar Arquivo Baixado

Esse APÊNDICE tem como função auxiliar os interessados no processo para descompactação de arquivos por meio da plataforma *Windows*®.

B.1 DESCOMPACTAR UM ARQUIVO

Para extrair arquivos compactados em uma pasta, siga os passos abaixo:

1º Passo: Baixe o programa "*Winrar*" ou o programa "*Winzip*" para descompactar o arquivo.

2º Passo: Para facilitar, crie uma pasta na "Área de Trabalho" com o nome CHAI 3D e depois mova o arquivo do 3º passo da seção 3.2 para essa pasta. Após instalar o *Winrar* ou o *Winzip*, clique com o botão direito do *mouse* sobre o arquivo e marque "Extrair aqui" com o botão esquerdo do *mouse*, em aproximadamente um minuto será criada uma pasta com o nome do arquivo descompactado, Figura B.1.

Figura B. 1 – Pasta com arquivos descompactados do CHAI 3D e pasta compactada do CHAI 3D sendo descompactada pelo programa *Winrar*.

APÊNDICE C - Instalação do *Microsoft*® *Visual Studio*® 2008

Este APÊNDICE destina-se exclusivamente a informar os passos e procedimentos que devem ser seguidos para se realizar a correta instalação do *Microsoft*® *Visual Studio*® 2008 na sua estação de trabalho.

C.1 INSTALAÇÃO DO *MICROSOFT*® *VISUAL STUDIO*® 2008

Adquira o CD de instalação do programa *Microsoft*® *Visual Studio*® 2008 – (MSV2008), insira-o no *driver* de CD/DVD do computador ou notebook, unidade D, a instalação irá começar automaticamente e aparecerá uma tela como a da Figura C.1. Verifique se o computador está conectado à internet, pois a atualização de alguns arquivos importantes será realizada no momento dessa instalação.

Figura C. 1 – Caixa de diálogo inicial para instalação do MSV2008.

Clique com o botão esquerdo do *mouse* em "*Install Visual Studio 2008*" (Instale o Visual Studio 2008) para iniciar a instalação. Após esse comando, uma nova janela se abrirá, veja Figura C.2.

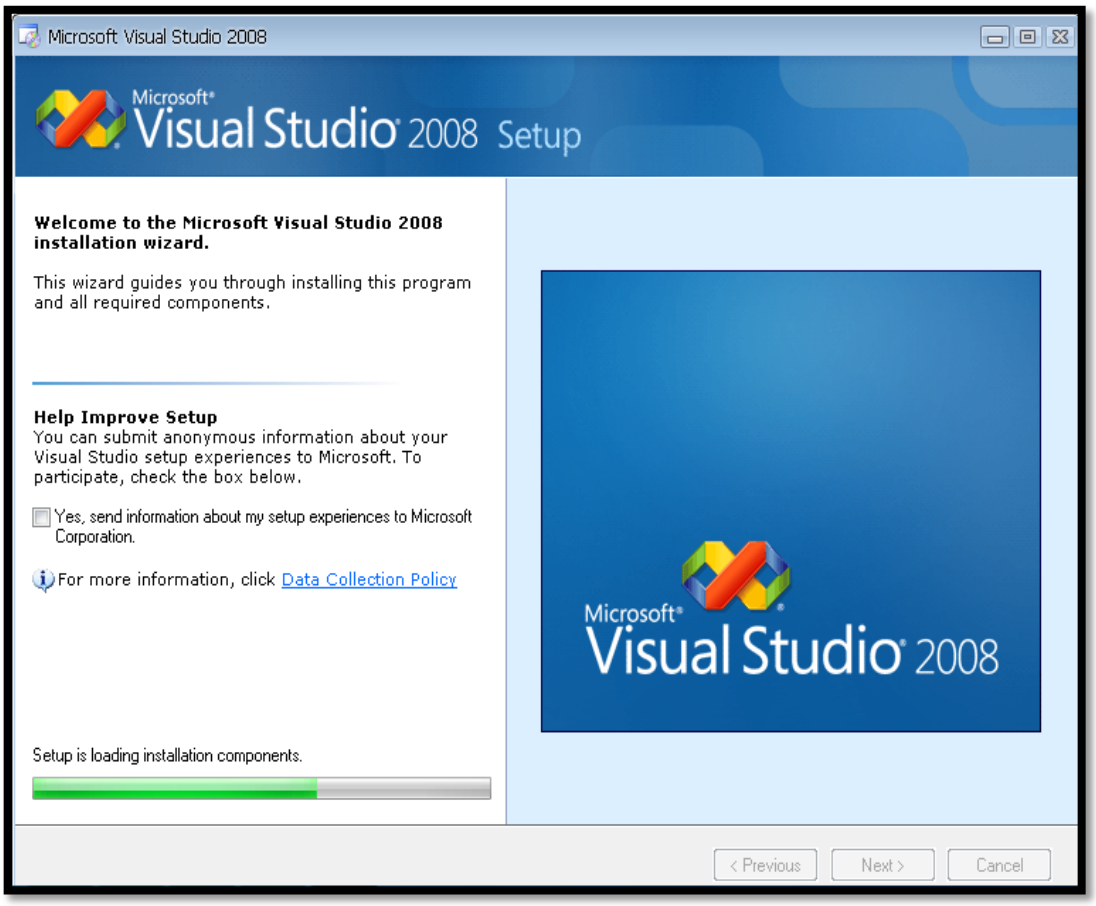

Figura C. 2 – Janela que informa o carregamento dos componentes já instalados no computador.

Nessa janela aguarde até que todos os arquivos sejam carregados e clique com o botão esquerdo do *mouse* em "*Next*" (próximo), que ficará disponível após o carregamento dos componentes instalados no computador. Não é necessário marcar a opção "*Yes, send information about my setup experiences to Microsoft Corporation.*". A próxima janela solicita que você leia o termo de licença, Figura C.3. Após ler o termo clique com o botão esquerdo do *mouse* na opção "*I have read and accept the license terms*", preencha a chave do produto "*product key*", com 25 caracteres em maiúsculo (números e letras). O campo "*Name*" - (nome) será preenchido automaticamente com o nome do usuário ou da conta em que estiver instalando o programa, caso não aconteça, preencha apenas com o seu primeiro e último nomes e posteriormente pressione "*Next*".

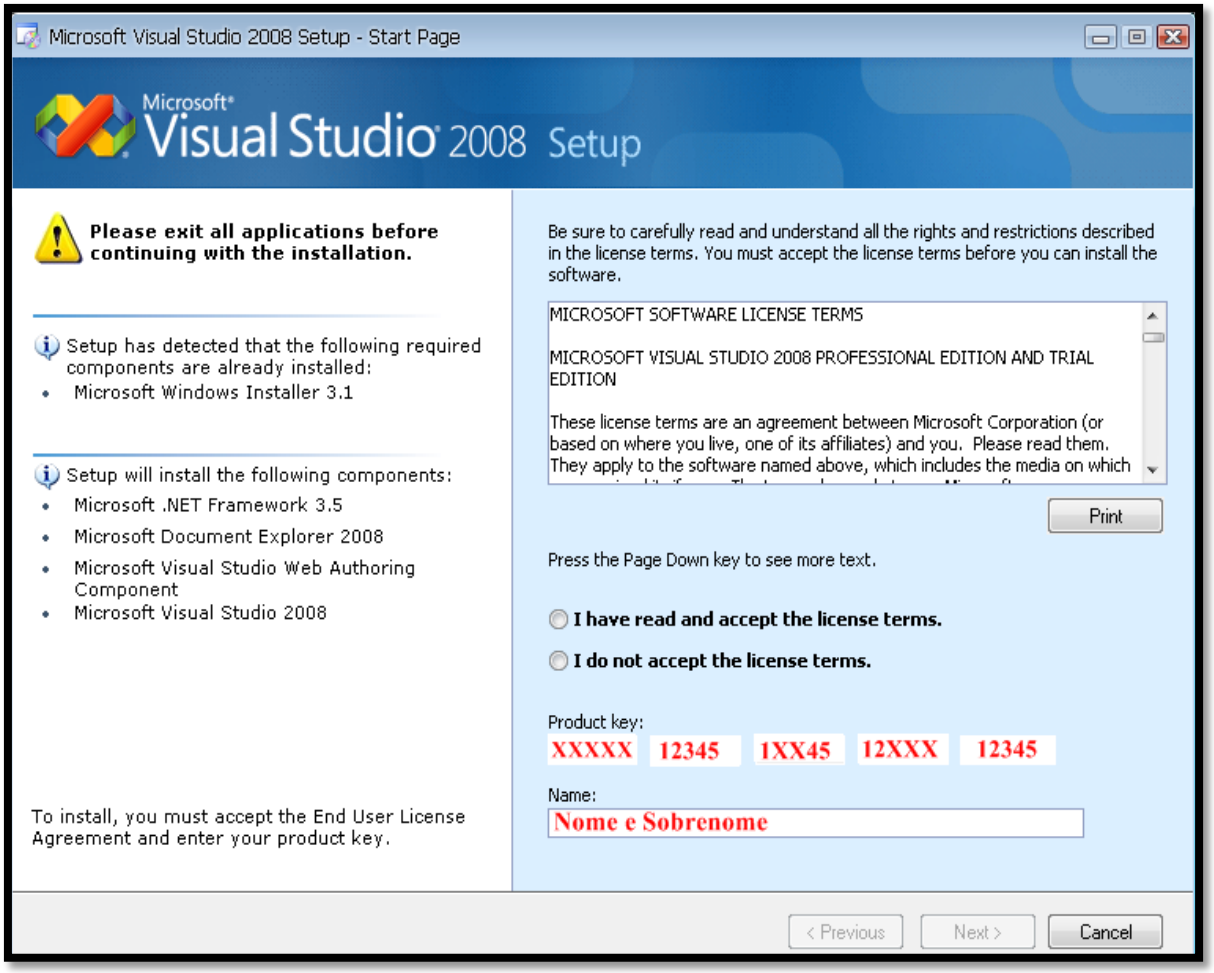

Figura C. 3 – Termo de Licença, chave do produto e nome do usuário.

A próxima janela diz respeito ao tipo de instalação e os componentes do programa que devem ser instalados. Três opções podem ser marcadas respectivamente de cima para baixo: "*Default*" (Padrão), "*Full*" (Completo) ou "*Custom*" (Personalizada), veja Figura C.4. Para que o processo ocorra com a maior eficiência possível, marque a opção completo. Ao lado, em "*Product install part*" é especificada a pasta onde será instalado o programa, por padrão as instalações são feitas na pasta "Arquivos de programas", que geralmente se localiza no driver "C". Caso seja necessário alterar o local de instalação, esse é o momento exato, para tanto, clique com o botão esquerdo do *mouse* em "Browse..." e informe um novo local no seu computador. Logo abaixo desse ponto são informadas a capacidade do disco rígido e a quantidade de memória requerida para armazenar o MVS2008, que no caso em questão foi de 4,1 *Gigabytes*. Após verificar todos esses passos clique com o botão esquerdo do *mouse* em "*Install*" (Instalar). Caso tenha esquecido algum passo, clique em "*Previous*" (Anterior) e realize as alterações necessárias.

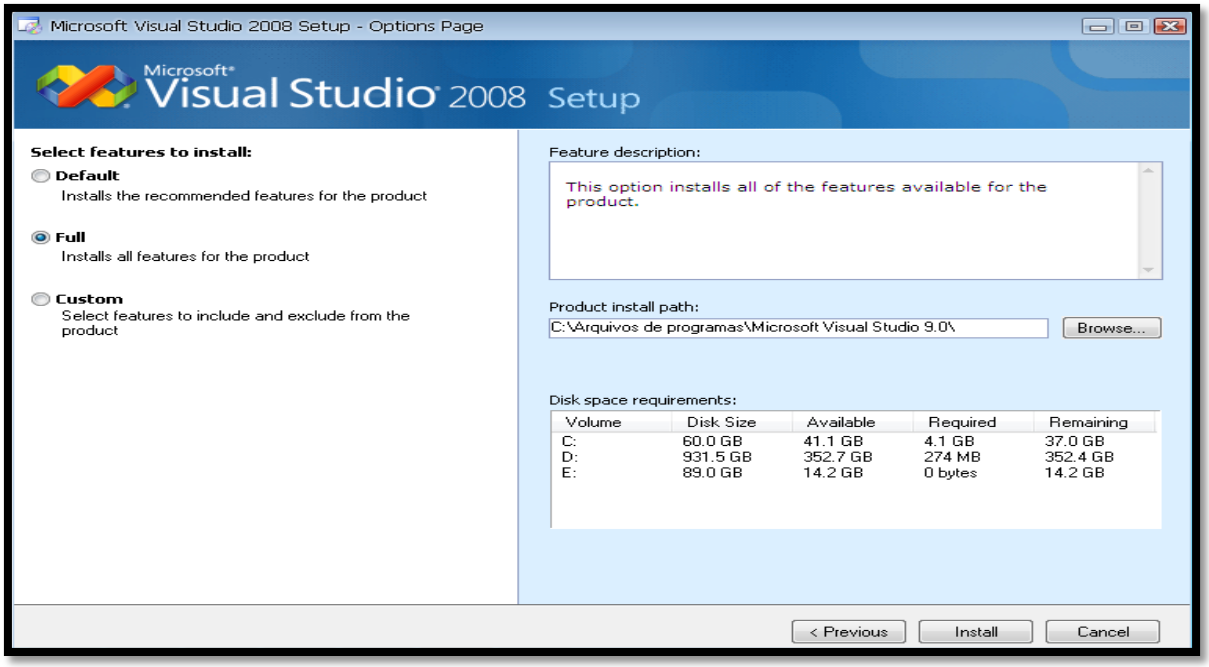

Figura C. 4 - Tipo de instalação do programa, local de armazenamento e capacidade de disco rígida requerida para instalar o programa.

Uma nova janela irá se abrir mostrando o processo de instalação, veja a Figura C.5. Essa etapa é um pouco demorada e pode variar de 20 a 40 minutos, dependendo da velocidade de processamento do seu computador e da velocidade da sua conexão com a internet.

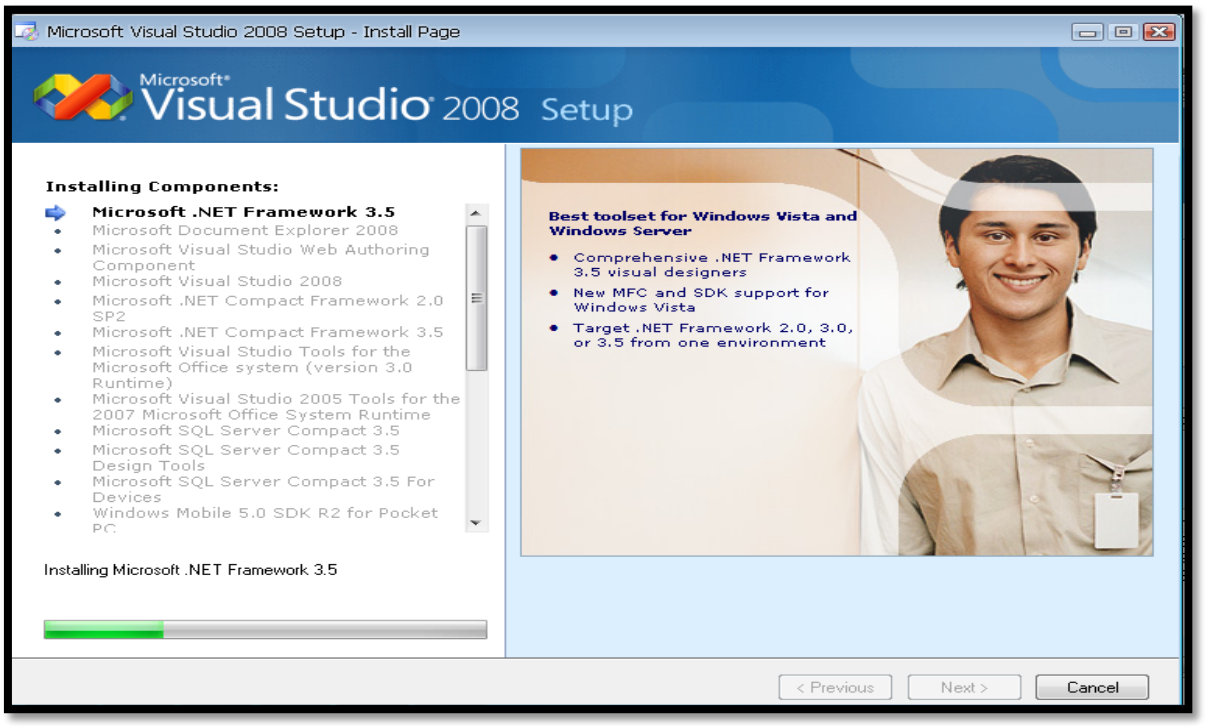

Figura C. 5 – Processo de Instalação.

Por fim, após a instalação de todos os itens listados na janela anterior, Figura C.5, será mostrada a última janela informando que o processo ocorreu com sucesso, veja Figura C.6. Pode ser que ocorra algum erro e que seja necessário instalar algum aplicativo manualmente, uma janela poderá informar a ausência desse aplicativo ou programa que impedi a instalação correta do MSV2008. Se nenhum erro ocorreu no processo, clique em "*Finish*" para finalizar a instalação e utilizar o MVS2008.

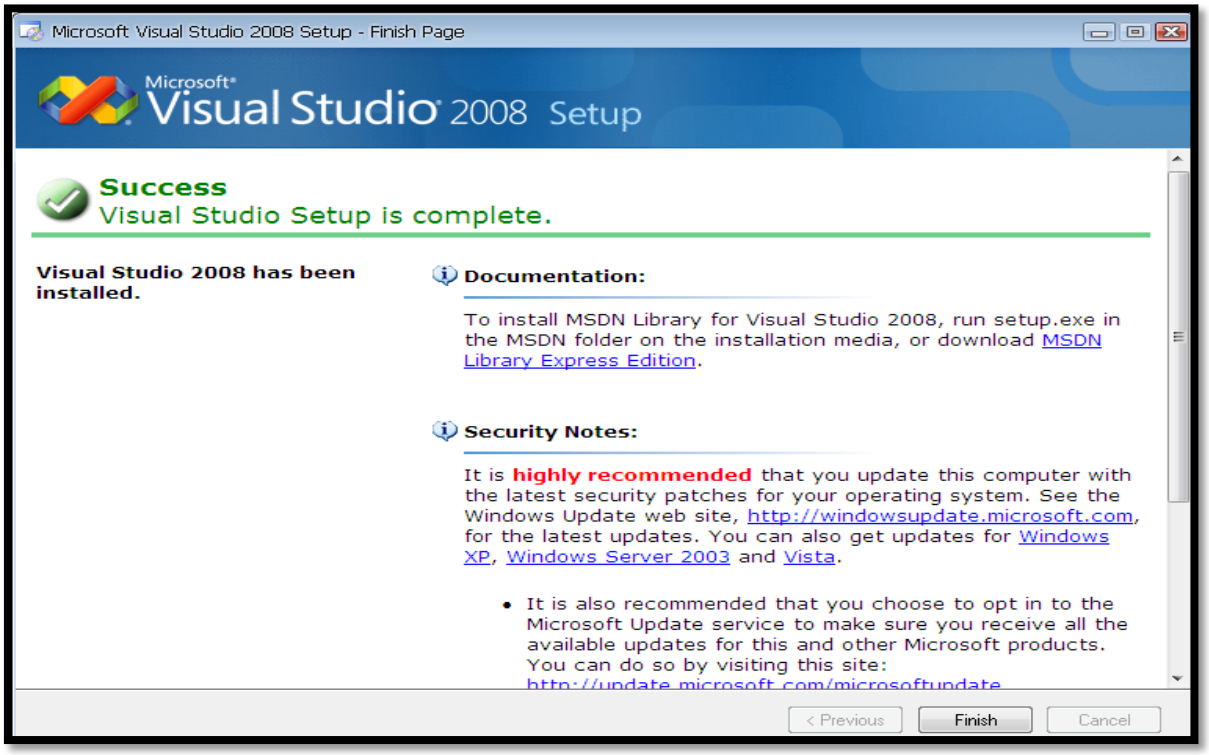

Figura C. 6 – Confirmação do sucesso da instalação do programa.

APÊNDICE D – Código Modificado

Seguem abaixo todas as modificações realizadas no código do projeto exemplo “20-map” do CHAI 3D. Porém, utilize este APÊNDICE apenas para consultas de trechos do código, dê preferência aos arquivos que foram disponibilizados por meio do endereço eletrônico: <sites.google.com/site/biomembranadeformavel/dissertacao>, uma vez que as linhas de código abaixo não possuem mais a sua formatação original.

```
//------------------------------------------------------------------------------------------------

#include <assert.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "chai3d.h"

/* ============================================================================================== */
/* ============================== DECLARAÇÃO DAS CONSTANTES ===================================== */
/* ============================================================================================== */

const int LARGURA_JANELA       = 600;           // LARGURA DA JANELA
const int ALTURA_JANELA = 600;                  // ALTURA DA JANELA

const int TELA_CHEIA            = 1;            // 1ª OPÇÃO DO BOTÃO DIREITO DO MOUSE
const int JANELA_COMPACTA       = 2;            // 2ª OPÇÃO DO BOTÃO DIREITO DO MOUSE

const double MESH_SCALE_SIZE = 2.0;             // TAMANHO DO PLANO

/* ============================================================================================== */
/* ============================== DECLARAÇÃO DAS VARIÁVEIS ====================================== */
/* ============================================================================================== */

cWorld* mundo;          // CLASSE QUE CRIA O MUNDO

cCamera* camera;// CLASSE QUE CRIA A CÂMERA

cMesh* membrana;// CLASSE QUE CRIA A MEMBRANA (OBJETO) ATRAVÉS DE UMA MALHA DE TRIÂNGULOS

cLight *luz;                    // CLASSE QUE CRIA A LUZ (ILUMINAÇÃO DA CENA VIRTUAL)

int displayW  = 0;              // LARGURA DA JANELA

int displayH  = 0;              // ALTURA DA JANELA

cHapticDeviceHandler* manipulador;      // CLASSE QUE CRIA O MANIPULADOR DA CENA VIRTUAL (DISPOSITIVO
HÁPTICO VIRTUAL)

cGeneric3dofPointer* esfera;            // CLASSE QUE CRIA A FERRAMENTA FÍSICA NO AMBIENTE VIRTUAL
(ESFERA BRANCA QUE MANIPULA A MEMBRANA.

double raioProxy;                       // RAIO DA FERRAMENTA VIRTUAL (INVISÍVEL POR PADRÃO)

bool correndoSimulacao = false; // ESTADO DA SIMULAÇÃO (INÍCIO = DESLIGADO)

void atualiza_posicao_camera(); // ATUALIZAÇÃO DA POSIÇÃO DA CÂMERA

// COORDENADAS ESFÉRICAS PARA POSIÇÃO DA CÂMERA:

double cameraAngleH;
```

```
double cameraAngleV;
double distanciaCamera;
cVector3d posicaoCamera;

bool movimentaCamera;                    // ESTADO DA CÂMERA

int mouseX, mouseY;                      // POSIÇÃO DO MOUSE

int botaoMouse;                          // BOTÕES DO MOUSE

bool terminarSimulacao = false;          // FINALIZA A SIMULAÇÃO

cGenericHapticDevice* phantom;           // CLASSE PARA CRIAÇÃO DO DISPOSITIVO TÁTIL LIGADO AO
COMPUTADOR (PHANTOM OMNI)

const double Pi =3.14152926535;          // DEFINIÇÃO DO VALOR DE PI

/* ================================================================================================ */
/* ================================= DECLARAÇÃO DAS FUNÇÕES ======================================= */
/* ================================================================================================ */

void redimensionaJanela(int w, int h);           // CHAMA A FUNÇÃO PARA REDIMENCIONAR A JANELA.

void teclado(unsigned char key, int x, int y);           // CHAMA A FUNÇÃO TECLADO QUANDO ALGUMA TECLA
É PRESSIONADA

void cliqueMouse(int button, int state, int x, int y);   // CHAMA A FUNÇÃO CLIQUE COM OS BOTÕES MOUSE
QUANDO ALGUM BOTÃO É PRESSIONADO

void movimentoMouse(int x, int y);       // CHAMA A FUNÇÃO QUE CONTROLA O MEMBRANA COM AJUDA DO MOUSE

void fechar(void);                       // FUNÇÃO QUE REALIZA A SAÍDA DA SIMULAÇÃO

void atualizaCena(void);                 // ATUALIZA A CENA GRÁFICA

void atualiza_dispositivo_haptico(void);         // ATUALIZA A POSIÇÃO DO DISPOSITIVO HÁPTICO

int plano();                             // CRIA A MEMBRANA INICIAL DA SIMULAÇÃO PARA SER DEFORMADA

/* ================================================================================================ */
/* ==================== FUNÇÃO PRINCIPAL - INICIALIZAÇÃO DO PROGRAMA0 ============================= */
/* ================================================================================================ */

int main(int argc, char* argv[])
{
printf ("\n");
printf (" ----------------------------------\n");
printf (" CHAI 3D | UNB-FUP\n");
printf (" RAFAEL SOUZA DA COSTA\n");
printf (" Copyright 2011-2013\n");
printf (" ----------------------------------\n");
printf ("\n\n");
printf (" COMANDOS DO MOUSE\n\n");
printf (" - Use o botao esquerdo para rotacionar a membrana. \n");
printf (" - Use o botao direito para distanciar-se ou afastar-se da membrana. \n");
printf ("\n\n");
printf (" COMANDOS DO TECLADO\n\n");
printf (" [1] - Gradeado (Ligado/Desligado)\n");
printf (" [x] - Finalizar ou Sair \n");
printf ("\n\n");

/* ================================================================================================ */
/* ================================== CENA GRÁFICA EM 3D ========================================== */
/* ================================================================================================ */

mundo = new cWorld();            // CRIA O NOVO MUNDO VIRTUAL

camera = new cCamera(mundo);     // CRIA UMA CÂMERA E INSERE ESSA CÂMERA NO MUNDO

mundo->addChild(camera);         // ADICIONA UMA CÂMERA AO MUNDO COMO UM DE SEUS FILHOS

// DEFINI A POSIÇÃO PADRÃO DA CÂMERA EM COORDENADAS ESFÉRICAS:
cameraAngleH = 0;
cameraAngleV = 15;
```

```
distanciaCamera = 1.8 * MESH_SCALE_SIZE;
atualiza_posicao_camera();

luz = new cLight(mundo);                // CRIA A LUZ NO MUNDO

camera->addChild(luz);                  // ANEXA UMA LUZ NA CÂMERA COMO UM FILHO

luz->setEnabled(true);                  // DEIXA A LUZ LIGADA

luz->setPos(cVector3d( 0.0, 0.3, 0.3));         // POSIÇÃO DA FONTE DE LUZ

luz->setDir(cVector3d(-1.0,-0.1, -0.1));        // DEFINI A DIREÇÃO DO FEIXE DE LUZ EMITIDO PELA FONTE

luz->m_ambient.set(0.5, 0.5, 0.5);      // PROPRIEDADES PARA SOMBRA

luz->m_diffuse.set(0.8, 0.8, 0.8);      // PROPRIEDADES PARA SOMBRA

luz->m_specular.set(1.0, 1.0, 1.0);     // PROPRIEDADES PARA SOMBRA

/* ================================================================================================ */
/* ============================== DISPOSITIVO HÁPTICO E ESFERA ================================== */
/* ================================================================================================ */

manipulador = new cHapticDeviceHandler();       // CRIAÇÃO DO DISPOSITIVO HÁPTICO DA CENA VIRTUAL

cGenericHapticDevice* dispositivoHaptico;       // CLASSE QUE CHAMA O DISPOSITIVO HÁPTICO

manipulador->getDevice(dispositivoHaptico);     // IDENTIFICAÇÃO DOS DISPOSITIVOS HÁPTICOS

phantom = dispositivoHaptico;           // DEFINE A VALIÁVEL PHANTOM COMO SENDO UM DISPOTIVO HÁPTICO

esfera = new cGeneric3dofPointer(mundo);        // CRIA A FERRAMENTA (ESFERA BRANCA) E INSERE NO MUNDO

camera->addChild(esfera);                       // ANEXAR UMA CÂMERA PARA ESFERA

esfera->setPos(-distanciaCamera, 0.0, 0.0);             // POSIÇÃO DA ESFERA COM RELAÇÃO A CÂMERA

esfera->setHapticDevice(dispositivoHaptico);            // COMUNICA OU CONECTA O DISPOTIVO HÁPTICO COM
A FERRAMENTA (ESPERA BRANCA)

esfera->start();// INICIALIZA A FERRAMENTA (ESPERA BRANCA) PARA CONEXÃO COM DISPOSITIVO HÁPTICO

esfera->setWorkspaceRadius(1.0);// INFORMA A FERRAMENTA (ESFERA BRANCA) O TAMANHO TOTAL DA ÁREA DE
TRABALHO

esfera->setRadius(0.05);// DEFINE O RAIO DA ESFERA BRANCA

esfera->m_deviceSphere->setShowEnabled(false);  // ESCONDE A ESFERA DO PROXY (PARA MOSTRAR O PROXY
INSERA: ", true", dentro do parentese depois de false)

// DEFINIÇÃO DAS PROPRIEDADES DO PROXY:
raioProxy = 0.05;
// RAIO FÍSICO DO PROXY
esfera->m_proxyPointForceModel->setProxyRadius(raioProxy);
esfera->m_proxyPointForceModel->m_collisionSettings.m_checkBothSidesOfTriangles = false;

/* ================================================================================================ */
/* ============================== COMPOSIÇÃO DA CENA VIRTUAL ================================== */
/* ================================================================================================ */

membrana = new cMesh(mundo);            // CRIA UMA MEMBRANA NO MUNDO

mundo->addChild(membrana);              // ANEXA A MEMBRANA AO MUNDO COMO UM DE SEUS FILHOS

plano();                                // CARREGA O PLANO

double workspaceScaleFactor = esfera->getWorkspaceScaleFactor();        // LEITURA DA ESCALA DA PROVA
(ESFERA).

/* ================================================================================================ */
/* ============================== JANELA DE TRABALHO - OPEN GL ================================== */
/* ================================================================================================ */

glutInit(&argc, argv);          // INICIALIZA GLUT
```

```
// RECUPERA A RESOLUÇÃO DA JANELA DO COMPUTADOR E COLOCA A JANELA DO GLUT NA POSIÇÃO CENTRAL DO
MONITOR:
int larguraJANELA       = glutGet(GLUT_SCREEN_WIDTH);
int alturaJANELA= glutGet(GLUT_SCREEN_HEIGHT);
int windowPosX = (larguraJANELA - LARGURA_JANELA) / 2;
int windowPosY = (alturaJANELA  - ALTURA_JANELA) / 2;

// INICIALIZA A JANELA DO OPENGL GLUT:
glutInitWindowPosition(windowPosX, windowPosY);                 // POSIÇÃO DA JANELA
glutInitWindowSize(LARGURA_JANELA, ALTURA_JANELA);              // LARGURA E ALTURA DA JANELA
glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE);       // MODE DE INICIALIZA NA JANELA DE
TRABALHO.
glutCreateWindow(argv[0]);                          // CRIA A JANELA
glutDisplayFunc(atualizaCena);                      // ATUALIZA O GRÁFICO DA CENA VIRTUAL
glutMouseFunc(cliqueMouse);                         // CHAMA AS FUNÇÕES DE CLIQUE COM OS BOTÕES DO MOUSE
glutMotionFunc(movimentoMouse);                     // CHAMA AS FUNÇÕES DO MOVIMENTO DO MOUSE
glutKeyboardFunc(teclado);                          // CHAMA AS FUNÇÕES DO TECLADO
glutReshapeFunc(redimensionaJanela);                // DIMENSIONA A JANELA
glutSetWindowTitle("CHAI 3D - BIOMEMBRANAS DEFORMAVEIS");       // TÍTULO NA BARRA DA JANELA


/* ============================================================================================== */
/* ================================== COMEÇO DA SIMULAÇÃO ======================================= */
/* ============================================================================================== */

correndoSimulacao = true;                           // INICIE A SIMULAÇÃO

cThread* hapticsThread = new cThread(); // CRIA UM SEGMENTO QUE INICIA O CICLO DE PROCESSAMENTO DO
DISPOSITOVOS HÁPTICOS PRINCIPAIS

hapticsThread->set(atualiza_dispositivo_haptico, CHAI_THREAD_PRIORITY_HAPTICS);  // INICIALIZA OS
DISPOTIVOS HÁPTICOS.

glutMainLoop();         // MANTÉM O CICLO FUNCIONANDO NA JANELA PRINCIPAL

fechar();               // INTERROMPE O CICLO E FECHA A JANELA PRINCIPAL

return (0);             // FINALIZA A CENA E NÃO RETORNA NADA
}

/* ============================================================================================== */
/* ============== FUNÇÃO PARA ATUALIZAR / REDIMENCIONAR O TAMANHO DA JANELA ================== */
/* ============================================================================================== */

void redimensionaJanela(int largura, int altura)
{
displayW = largura;                                     // LARGURA DA JANELA
displayH = altura;                                      // ALTURA DA JANELA
glViewport(0, 0, displayW, displayH);
}

/* ============================================================================================== */
/* ================================== FUNÇÕES DO TECLADO ======================================== */
/* ============================================================================================== */

void teclado(unsigned char tecla, int x, int y)
{
// TECLA ESC OU ESCAPE:
if ((tecla == 27) || (tecla == 'x'))
{
fechar();       // FECHAR TUDO

exit(0);        // SAIR DA SIMULAÇÃO
}

// OPÇÃO 1:
if (tecla == '1')
{
bool useWireMode = membrana->getWireMode();     // LIGA A OPÇÃO GRADEADO
membrana->setWireMode(!useWireMode);            // DESLIGA A OPÇÃO GRADEADO
}
}
```

```
/* ================================================================================================ */
/* ======================== FUNÇÃO PARA CLIQUE COM OS BOTÕES DO MOUSE ============================= */
/* ================================================================================================ */

void cliqueMouse(int botao, int estado, int x, int y)
{
if (estado == GLUT_DOWN)                // ESTADO PARA BAIXO
{
movimentaCamera = true;                 // MOVIMENTO DA CÂMERA PARA BAIXO
mouseX = x;
mouseY = y;
botaoMouse = botao;
}
else if (estado == GLUT_UP)      // ESTADO PARA CIMA
{
movimentaCamera = false;         // MOVIMENTO DA CÂMERA PARA CIMA
}
}


/* ================================================================================================ */
/* ============================== FUNÇÃO PARA MOVIMENTOS DO MOUSE ================================= */
/* ================================================================================================ */

void movimentoMouse(int x, int y)
{
if (movimentaCamera)
{
if (botaoMouse == GLUT_RIGHT_BUTTON)                            // BOTÃO DIREITO
{
distanciaCamera = distanciaCamera - 0.01 * (y - mouseY);        // MODIFICA A DISTÂNCIA DA MEMBRANA
COM RELAÇÃO A CÂMERA
}

else if (botaoMouse == GLUT_LEFT_BUTTON)                // BOTÃO ESQUERDO (MODIFICA O ÂNGULO DA CÂMERA
COM RELAÇÃO À MEMBRANA
{
cameraAngleH = cameraAngleH - (x - mouseX);             // ÂNGULO NA HORIZONTAL
cameraAngleV = cameraAngleV + (y - mouseY);             // ÂNGULO NA VERTICAL
}
}

atualiza_posicao_camera();

mouseX = x;
mouseY = y;
}

/* ================================================================================================ */
/* ========================== FUNÇÃO PARA FINALIZAR A SIMULAÇÃO ================================== */
/* ================================================================================================ */

void fechar(void)
{
correndoSimulacao = false;                      // PARAR A SIMULAÇÃO

while (!terminarSimulacao) { cSleepMs(100); }   // AGUARDAR ATÉ QUE O DISPOSITIVO HÁPTICO E OS
GRÁFICOS FINALIZEM

esfera->stop();                                 // PARAR E DESLIGAR O DISPOSITIVO HÁPTICO
}

//---------------------------------------------------------------------------------------------------

void atualizaCena(void)
{
membrana->computeAllNormals(true);              // ATUALIZAÇÃO DA POSIÇÃO NORMAL DA MEMBRANA

camera->renderView(displayW, displayH);         // CRIA A VISUALIZAÇÃO DA JANELA INICIAL

glutSwapBuffers();                              // REALIZA O CARREGAMENTO DA JANELA

// PROCURAR POR ERROS NO OPEN GL:
GLenum err;
```

```
err = glGetError();
if (err != GL_NO_ERROR) printf("Error:  %s\n", gluErrorString(err));

// INFORMA AO GLUT SOBRE A ATUALIZAÇÃO DA CENA GRÁFICA PARA O PRÓXIMO FRAME:

if (correndoSimulacao)
{
glutPostRedisplay();
}
}

/* ================================================================================================ */
/* ================ FUNÇÃO PARA ATUALIZAR A POSIÇÃO DO DISPOSITIVO HÁPTICO ====================== */
/* ================================================================================================ */

void atualiza_dispositivo_haptico(void)
{
const int RIGIDO          = 1;                  // ESTADO RÍGIDO DA MEMBRANA
const int DEFORMAVEL      = 2;                  // ESTADO DEFORMÁVEL DA MEMBRANA
int estado = RIGIDO;

cVector3d posEsfera (0.0,0.0,0.0);              // POSIÇÃO INICIAL DA PROVA (ESFERA)

while(correndoSimulacao)                // ENQUANTO A SIMULAÇÃO ESTIVER SENDO EXECUTADA
{
mundo->computeGlobalPositions(true);    // CALCULAR A REFERÊNCIA GLOBAL PARA A MEMBRANA

esfera->updatePose();                   // ATUALIZA A POSIÇÃO E A ORIENTAÇÃO DA PROVA(ESFERA)

esfera->computeInteractionForces();     // CÁLCULA AS FORÇAS DE INTERAÇÃO ENTRE A PROVA (ESFERA) E A
MEMBRANA

bool userSwitch = esfera->getUserSwitch(0);     // LÊ A SITUAÇÃO DO BOTÃO SWITCH (ALTERNA ENTRE LIGADO
OU DESLIGADO PARA QUE HAJA A DEFORMAÇÃO)

if ((estado == DEFORMAVEL || estado == RIGIDO) && (userSwitch)) // SE O ESTADO DEFORMÁVEL OU FIXO
ESTIVER ATIVO E O BOTÃO SWITCH ESTIVER LIGADO, COMECE A DEFORMAÇÃO
{
cVector3d posicao = posEsfera;          // CÁLCULA A POSIÇÃO DA PROVA (ESFERA) ENQUANTO A DEFORMAÇÃO
ESTIVER ACONTECENDO

int numVertices = membrana->getNumVertices(true);       // DECLARA A VARIÁVEL INTEIRA "numVertices",
NÚMERO DE VÉRTICES PARA OBTER OS MESMO

for (int i=0; i<numVertices; i++)       // REFAZ A SUPERFÍCIE DA MEMBRANA A PARTIR DO NÚMERO DE
VÉRTICES
{
cVertex* vertices = membrana->getVertex(i, true);       // OBTÊM NOVOS VÉRTICES PARA A MEMBRANA ATÉ
QUE i SEJA MAIOR DO QUE A QUANTIDADE DE VÉRTICES EXISTENTES


// CÁLCULA A DISTÂNCIA EXISTENTE ENTRE OS VÉRTICES QUE COMPOEM A MEMBRANA E A PROVA (ESFERA):

posEsfera = esfera->getDeviceGlobalPos();       // POSIÇÃO GLOBAL DA PROVA (ESFERA)

cVector3d posVertex = vertices->getPos();       // OBTÊM A POSIÇÃO DOS VÉRTICES QUE FORMAM A MEMBRANA


/* ================================================================================================ */
/* ============ ROTINA PARA DEFORMAÇÃO DA MEMBRANA - 1ª SITUAÇÃO (ESPAÇO LIVRE) ================= */
/* ================================================================================================ */

/*
double F=0.0;                   // FORÇA APLICADA EM UM PONTO DA MEMBRANA

double u;                       // DEFORMAÇÃO DA MEMBRANA NO EIXO Z (EIXO VERTICAL)

double u0 = posicao.z;          // POSIÇÃO REAL DA ESFERA QUE DEFORMA A MEMBRANA OU PLANO

double a = 0.1;                 // RAIO DO DISCO A

double r = sqrt(cSqr(posVertex.x-posicao.x)+cSqr(posVertex.y-posicao.y));

if (r < a ) u = u0;             // SITUAÇÃO PARA QUE NÃO OCORRA A DEFORMAÇÃO DA MEMBRANA OU PLANO
```

```
else {

F = -u0/log(a);                 // FORÇA APLICADA EM UMA REGIÃO DA MEMBRANA (EQUAÇÃO 4-27)

u = F*log(a/r)+u0;              // EQUAÇÃO PARA DEFORMAÇÃO DA MEMBRANA NO ESPAÇO LIVRE (EQUAÇÃO 4-23)

}

if (posEsfera.z >= 0.00001) u= 0.0;     // SITUAÇÃO PARA QUE A ESFERA NÃO ACOMPANHEI O PLANO A PARTIR
DA PARTE POSITIVA DO EIXO VERTICAL
*/

//-----------------------------------------------------------------------------------------------
// ================== FIM DA ROTINA PARA IMPLEMENTAÇÃO DA 1ª SITUAÇÃO ==============================
//-----------------------------------------------------------------------------------------------


/* ============================================================================================== */
/* ============ ROTINA PARA DEFORMAÇÃO DA MEMBRANA - 2ª SITUAÇÃO (BORDA FIXA) ================== */
/* ============================================================================================== */
/*
// VARIÁVEIS

double g;                       // FUNÇÃO DE GREEN
double ga;                      // FUNÇÃO DE GREEN PARA A POSIÇÃO DO DISCO A
double F=0.0;                   // FORÇA APLICADA EM UM PONTO DA MEMBRANA
double teta;                    // ÂNGULO COM A HORIZONTAL COM RELAÇÃO A R
double tetaa;                   // ÂNGULO COM A HORIZONTAL COM RELAÇÃO A RA
double teta0;                   // ÂNGULO INICIAL DA PROVA OU ESFERA COM RELAÇÃO À MEMBRANA OU PLANO
double ya=0.0;                  // POSIÇÃO DA COMPONENTE Y COM RELAÇÃO A POSIÇÃO DO DISCO A
double xa;                      // POSIÇÃO DA COMPONENTE X COM RELAÇÃO A POSIÇÃO DO DISCO A
double a = 0.2;                 // RAIO DA PROVA (REPRESENTADO PELA POSIÇÃO DA ESFERA NA SIMULAÇÃO)
double u;                       // DEFORMAÇÃO DA MEMBRANA NO EIXO Z (EIXO VERTICAL)
double u0 = posicao.z;          // POSIÇÃO DA PROVA COM REAÇÃO AO EIXO VERTICAL
double Tau = 1.0/2.0/Pi;        // COEFICIENTE PARA TENSÃO SUPERFICIAL DA MEMBRANA

double r0 = sqrt(cSqr(posicao.x)+cSqr(posicao.y));      // POSIÇÃO INICIAL DA PROVA (ESFERA) NA
MEMBRANA

double r = sqrt(cSqr(posVertex.x)+cSqr(posVertex.y));   // MÓDULO DAS COMPONENTES X E Y, ONDE
R=v(X^2+Y^2)

double ra;                      // MÓDULO DAS COMPONENTES X E Y DO DISCO A,ONDE R=v(X^2+Y^2)

if(sqrt(cSqr(posVertex.x)+ cSqr(posVertex.y)) > 1.0 || sqrt(cSqr(posicao.x)+ cSqr(posicao.y)) > 1.0) u
= 0.0;                          // DEFINIÇÃO DA BORDA COMO SENDO IGUAL A 1

else
{
if(posicao.y > 0.01 || posicao.y < -0.01)
{
teta0 = atan((posicao.y)/(posicao.x));
}
else{
teta0=0.0;
}

if(posVertex.y > 0.01 || posVertex.y < -0.01)
{
teta = atan((posVertex.y)/(posVertex.x));

}
else{
teta=0.0;
}

if (posicao.x < 0.0) teta0=teta0+Pi;            // ALTERA O QUADRANTE DE NEGATIVO PARA POSITIVO

if (posVertex.x < 0.0) teta=teta+Pi;            // ALTERA O QUADRANTE DE NEGATIVO PARA POSITIVO

ya=posicao.y;                                   // COMPONENTE Y COM RELAÇÃO AO DISCO A

xa=posicao.x+a;                                 // COMPONENTE X COM RELAÇÃO AO DISCO A
```

```
if(ya > 0.01 || ya < -0.01)
{
tetaa = atan((ya)/(xa));
}
else{
tetaa=0.0;
}

g = -1.0/4.0/Pi*(-log(cSqr(r)+cSqr(r0)-2.0*r*r0*cos(teta-teta0))+log(1.0+cSqr(r*r0)-2.0*r*r0*cos(teta-
teta0)));

if (xa < 0.0) tetaa=tetaa+Pi;           // ALTERA O QUADRANTE DE NEGATIVO PARA POSITIVO

ra=sqrt(cSqr(xa)+cSqr(ya));             // RAIO DA POSIÇÃO DO DISCO COM RELAÇÃO ÀS COMPONENTE YA E XA

ga = 1.0/4.0/Pi*(log(cSqr(ra)+cSqr(r0)-2.0*ra*r0*cos(tetaa-teta0))+log(1.0+cSqr(ra*r0)-
2.0*ra*r0*cos(tetaa-teta0)));

F = u0*Tau/ga;                  // FORÇA APLICADA EM UMA REGIÃO DA MEMBRANA (EQUAÇÃO 4-33)

u = F/Tau*g;                    // EQUAÇÃO PARA DEFORMAÇÃO DA MEMBRANA COM BORDA FIXA (EQUAÇÃO 4-32)

if(u < u0) u=u0;                // FAZ COM QUE A DEFORMAÇÃO ACOMPANHE A POSIÇÃO DO DISPOSITIVO
(ESFERA) E NÃO VÁ PARA INFINITO

}

if (posEsfera.z >= 0.00001) u= 0.0;     // CONDIÇÃO PARA QUE A ESFERA NÃO ACOMPANHEI O PLANO A PARTIR
DA PARTE POSITIVA DO EIXO VERTICAL

*/

//-------------------------------------------------------------------------------------------------
// ================== FIM DA ROTINA PARA IMPLEMENTAÇÃO DA 2ª SITUAÇÃO ================================
//-------------------------------------------------------------------------------------------------

// APLICAR DESLOCAMENTO DA MEMBRANA:

posVertex.z = u;                        // LEITURA DA POSIÇÃO DE U

vertices->setPos(posVertex);            // LEITURA DA POSIÇÃO DOS VÉRTICES

cVector3d eixo (0,0,0);                 // DECOMPOEM A CLASSE cVECTOR3D EM 3 COMPONENTES (X,Y,Z).

eixo.x=0.0;                             // INTENSIDADE DA FORÇA NO EIXO X

eixo.y=0.0;                             // INTENSIDADE DA FORÇA NO EIXO Y

eixo.z= -F*20.0;                        // INTENSIDADE DA FORÇA NO EIXO Z (EIXO NO QUAL É REALIZADO A
DEFORMAÇÃO DA MEMBRANA).

if (posEsfera.z <= 0.0)
phantom->setForce(eixo);                // ENVIAR FORÇAS PARA O DISPOSITIVO SE A ESFERA BRANCA ESTIVER
ABAIXO DE 0.0 NO EIXO Z.
}
}
esfera->applyForces();                  // ENVIA FORÇAS PARA O DISPOSITIVO HÁPTICO
}
terminarSimulacao = true;               // FINALIZA A SIMULAÇÃO
}

/* ============================================================================================== */
/* ================= FUNÇÃO PARA CRIAÇÃO DA MALHA QUE FORMA A MEMBRANA ========================== */
/* ============================================================================================== */

int plano()
{
int texSizeU = 100;             // LARGURA DA MEMBRANA
int texSizeV = 100;             // COMPRIMENTO DA MEMBRANA

int x = texSizeU;               // LARGURA
int y = texSizeV;               // COMPRIMENTO
```

```
if ((texSizeU < 1) || (texSizeV < 1)) { return (false); }       // CHECAR O TAMANHO DA IMAGEM, SE
LARGURA E COMPRIMENTO FOREM MAIORES QUE 1 TUDO OCORRE NORMALMENTE

int largestSide;                // LOCALIZA O LADO MAIOR DA IMAGEM

if (texSizeU > texSizeV)        // SE A TEXTURA EM U FOR MAIOR QUE A TEXTURA EM V
{
largestSide = texSizeU;         // O LADO MAIOR É U
}
else                            // SENÃO
{
largestSide = texSizeV;         // O LADO MAIOR É V
}

double tamanho = 2.0 / (double)largestSide;     // CÁLCULO DO TAMANHO DE CADA PIXEL NO MUNDO VIRTUAL

// CÁLCULO DO TAMANHO DOS TRIÂNGULOS REFERENTES ÀS POSIÇÕES DE X E Y, PARA SE TER COMO REFERÊNCIA:

double offsetU = 0.5 * (double)texSizeU * tamanho;
double offsetV = 0.5 * (double)texSizeV * tamanho;

// CRIA UM VÉRTICE PARA CADA PIXEL DA MEMBRANA:

int u,v;


for (v=0; v<texSizeV; v++)
{
for (u=0; u<texSizeU; u++)
{
double px, py;

// CÁLCULA A POSIÇÃO DOS VÉRTICES

px = tamanho * (double)u - offsetU;
py = tamanho * (double)v - offsetV;

// CRIA UM NOVO VÉRTICE

unsigned int index=membrana->newVertex (px, py, 0.0); // CRIA UM PLANO INICIAL PARA SER DEFORMADO

cVertex* vertex = membrana->getVertex(index);
}
}

// CRIA UM TRIÂNGULO COMO BASE UTILIZANDO OS PIXELS DE TEXSIZEU E TEXSIZEV:

for (v=0; v<(texSizeV-1); v++)
{
for (u=0; u<(texSizeU-1); u++)
{
// OBTER O NÚMERO DE INDEXAÇÃO DOS PRÓXIMO QUATRO VÉRTICES:
unsigned int index00 = ((v + 0) * texSizeU) + (u + 0);
unsigned int index01 = ((v + 0) * texSizeU) + (u + 1);
unsigned int index10 = ((v + 1) * texSizeU) + (u + 0);
unsigned int index11 = ((v + 1) * texSizeU) + (u + 1);

// CRIA DOIS NOVOS TRIÂNGULOS:
membrana->newTriangle(index00, index01, index10);
membrana->newTriangle(index10, index01, index11);
}
}

membrana->computeAllNormals(true);      // CALCULAR TODAS AS NORMAIS

membrana->computeBoundaryBox(true);     // CÁLCULA OS LIMITES DA CAIXA INVISÍVEL ONDE ESTÁ A MEMBRANA

cVector3d min = membrana->getBoundaryMin();     // OBTER TAMANHO LIMITE MÍNIMO PARA CAIXA

cVector3d max = membrana->getBoundaryMax();     // OBTER TAMANHO LIMITE MÁXIMO PARA CAIXA

cVector3d span = cSub(max, min);                // TAMANHO DA MEMBRANA

tamanho = cMax(span.x, cMax(span.y, span.z));   // LIMITES DA CAIXA
```

```
double escala = MESH_SCALE_SIZE / tamanho;      // DECLARAÇÃO DA VARIÁVEL PARA CALCULAR A ESCALA DA
MEMBRANA.

membrana->computeBoundaryBox(true);             // CÁLCULA NOVAMENTE O TAMANHO DA MEMBRANA

membrana->createAABBCollisionDetector(1.01 * raioProxy, true, false);   // DETECTA COLISÕES ENTRE A
MEMBRANA E OUTROS OBJETOS NA CENA SIMULAÇÃO.

membrana->setFrameSize(0.2, true);              // OBTER TAMANHO DOS FRAMES

membrana->setNormalsProperties(0.01, cColorf(1.0, 0.0, 0.0, 1.0), true); // OBTER TAMANHO DAS NORMAIS

membrana->setUseCulling(false);         // PROCESSAR GRAFICAMENTE AMBOS OS LADOS DOS TRIÂNGULOS

membrana->computeGlobalPositions();     // ATUALIZAR POSIÇÃO GLOBAL PARA A MEMBRANA

return (0);                             // CASO TUDO OCORRE BEM NADA SERÁ INFORMADO
}

/* ================================================================================================ */
/* ====================== FUNÇÃO PARA ATUALIZAR A POSIÇÃO DA CÂMERA ============================== */
/* ================================================================================================ */

void atualiza_posicao_camera()
{
// VERIFICAR VALORES:

if (distanciaCamera < 0.1) { distanciaCamera = 0.1; }
if (cameraAngleV > 89) { cameraAngleV = 89; }
if (cameraAngleV < -89) { cameraAngleV = -89; }

// CALCULAR A POSIÇÃO DA CÂMERA NO ESPAÇO:

cVector3d posicao = cAdd(
posicaoCamera,
cVector3d(
distanciaCamera * cCosDeg(cameraAngleH) * cCosDeg(cameraAngleV),
distanciaCamera * cSinDeg(cameraAngleH) * cCosDeg(cameraAngleV),
distanciaCamera * cSinDeg(cameraAngleV)
)
);

cVector3d olhar = posicaoCamera;                // CALCULAR PARA ONDE A CÂMERA ESTÁ OLHANDO

cVector3d para_cima(0.0, 0.0, 1.0);             // DEFINIR A ORIENTAÇÃO DA CÂMERA

camera->set(posicao, olhar, para_cima);         // OBTER NOVA POSIÇÃO PARA CÂMERA

mundo->computeGlobalPositions(true);            // RECALCULAR POSIÇÕES GLOBAIS PARA O MUNDO

if (esfera != NULL)                             // SE O ESTADO DA ESFERA FOR DIFERENTE DE NULO

esfera->setPos(-distanciaCamera, 0.0, 0.0);     // ATUALIZA POSIÇÃO DA ESFERA
}
```

APÊNDICE E – Baixar e Executar o Código Modificado

Este APÊNDICE explica de maneira detalhada os procedimentos para que você possa executar o código alterado através do seu computador, desde que todos os *hardware* e *software* mencionados no capítulo 3 estejam instalados e funcionando corretamente.

E.1 BAIXAR O CÓDIGO MODIFICADO

Para facilitar ainda mais nossos trabalhos, disponibilizamos o código com suas alterações no sítio: <sites.google.com/site/biomembranadeformavel/dissertacao>, entre nesse endereço e baixe um dos arquivos contendo as linhas de código abaixo descritas, note que na página os arquivos são disponibilizados em duas extensões diferentes (.cpp e .txt), faça uso daquela que lhe for mais familiar ou conveniente.

E.2 EXECUTAR O CÓDIGO MODIFICADO

Após baixar o arquivo mencionado na seção E.1, dê um duplo clique com o botão esquerdo do *mouse* para abri-lo e posteriormente selecione o código completo e copie-o, ou simplesmente, dê um clique no corpo do código e pressione simultaneamente as teclas "Ctrl+A" (seleciona todo o código ou texto) e depois "Ctrl+C" (copia a região selecionada do código ou texto), que também executam as mesmas tarefas. Concluído esse procedimento, inicie o *MVS2008* e abra a janela de edição com os projetos exemplos do CHAI 3D, localize o projeto 20 - map, dê um único clique com o botão esquerdo do mouse no símbolo de "+" ao lado do projeto para visualizar a raiz, encontre a subpasta "*Source Files*", novamente dê um único clique com o botão esquerdo do mouse no símbolo de "+" e por fim, execute o

comando clique duplo sobre o arquivo 20 – map.cpp, clique no corpo do código com o botão esquerdo do *mouse*, pressione simultaneamente as teclas “Ctrl+A” e depois “Ctrl+V”, o que fará com que o código que você baixou e copiou acima possa ser colocado em edição, veja Figura E.1. Finalizada essa parte, clique com o botão direito do mouse sobre o projeto 20-map e selecione o comando “*Set as StardUp Project*”, o que faz com que o projeto seja selecionado para ser realizada sua inicialização. Note que ele vai ficar destacado entre os demais, como se estivesse com o recurso negrito ativado. Pressione a tecla “F5” para compilar e iniciar a simulação.

Provavelmente ocorrerão alguns erros no momento da execução da simulação, acompanhe as seções 5.2 e 5.3 no capítulo 5, para realizar as modificações necessárias no corpo do código que possibilitarão a execução correta da simulação.

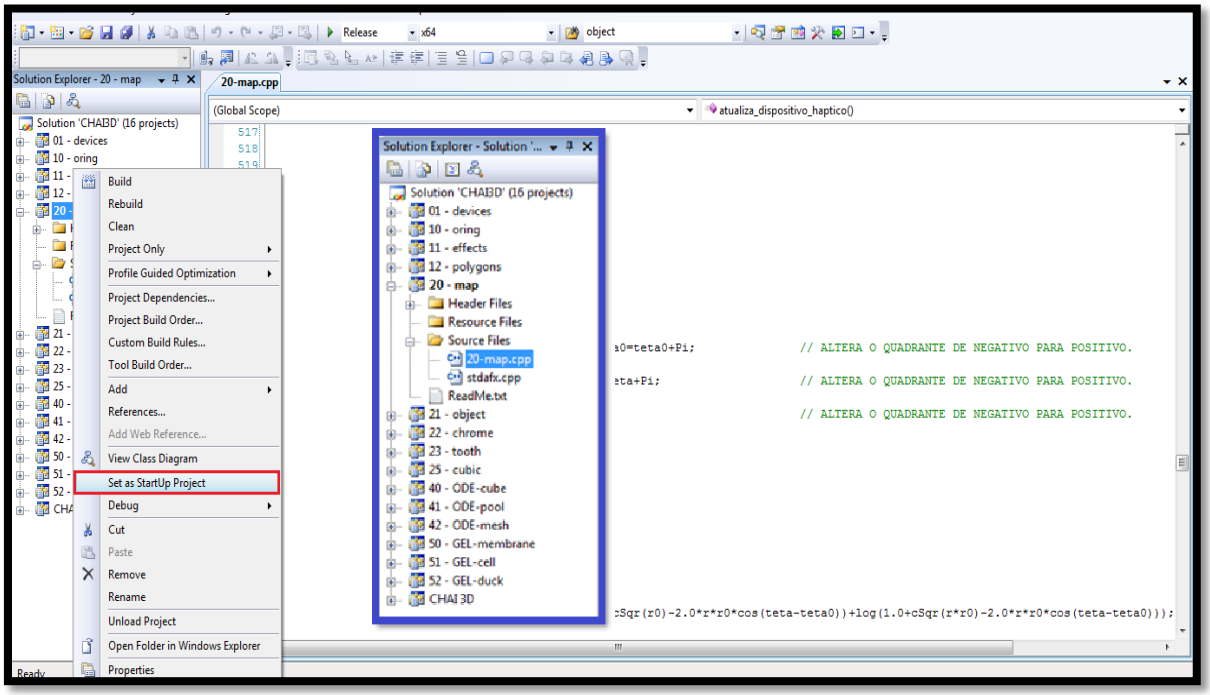

Figura E. 1 – Janela do MVS2008. Retângulo com borda azul mostra o painel “*Solution Explorer*” e a localização do arquivo 20-map.cpp enquanto que o retângulo vermelho marca o opção que deve ser selecionada para que o projeto selecionado seja prioridade no momento da inicialização.

APÊNDICE F – Aquisição dos Dados Para o Experimento Real e Para a Simulação

Este APÊNDICE fornece uma explicação detalhada dos procedimentos para obtenção dos dados experimentais a partir do artigo de Selvadurai e também dos dados referentes à simulação, o que por sua vez possibilitará a validação do método desenvolvido nessa dissertação por meio da comparação entre o experimento real e a simulação computacional.

F.1 AQUISIÇÃO DOS DADOS PARA O EXPERIMENTO REAL

Primeiramente devemos localizar os gráficos das Figuras 17 e 18 da publicação de Selvadurai (2006), depois devemos fazer uma cópia da tela pressionando a tecla "PrntScr" no teclado do computador, próximo a tecla delete. Quando for fazer a cópia, note que em cada uma das Figuras existem 5 gráficos comparando os diversos resultados com outros da literatura, dê preferência aquele gráfico que possui os três resultados experimentais de Selvadurai, é por eles que vamos conseguir criar gráficos iguais ou muito semelhantes. Após copiar a tela com a imagem desse gráfico, abra o programa *Paint* do *Windows*® e cole a imagem, ative as ferramentas régua e gradeado na aba "exibir", recorte a Figura e redimensione de modo a fazer com que ela possua as mesmas dimensões utilizadas no trabalho de Selvadurai, a saber, diâmetro de 250 (régua horizontal) e alturas respectivas de 25.4, 76.2 e 127 (régua vertical), não é necessário se preocupar com a unidade em que se encontra a régua no programa, pois de toda forma a proporção será mantida, o resultado pode ser observador na Figura F.1. Feito o ajuste, basta fazer a leitura das posições de cada *pixel* na barra inferior do programa para montar três tabelas que serão preenchidas com as várias coordenadas de x e y, tais coordenadas devem ser obtidas pelas linhas formadas pelos quadrados, triângulos e círculos, a tabela F.1 mostra um exemplo de como os dados poderão ser organizado, para na sequência serem plotados os gráficos. As curvas formadas pelos

quadrados, triângulos e círculo são na verdade os dados experimentais, as linhas continuas referem-se às comparações que Selvadurai fez com outros métodos na literatura, dessa forma, desconsidere-as. Note que na barra inferior do *Paint* existem duas referências aos *pixels*, a primeira destina-se às coordenadas exatas que determinado *pixel* possui na imagem, enquanto que a segunda indica a quantidade total de *pixels* que a imagem possui em cada eixo. Os valores de x e y deverão ser adquiridos através da primeira referência, sendo que o valor de x é o da esquerda e o valor de y o da direita.

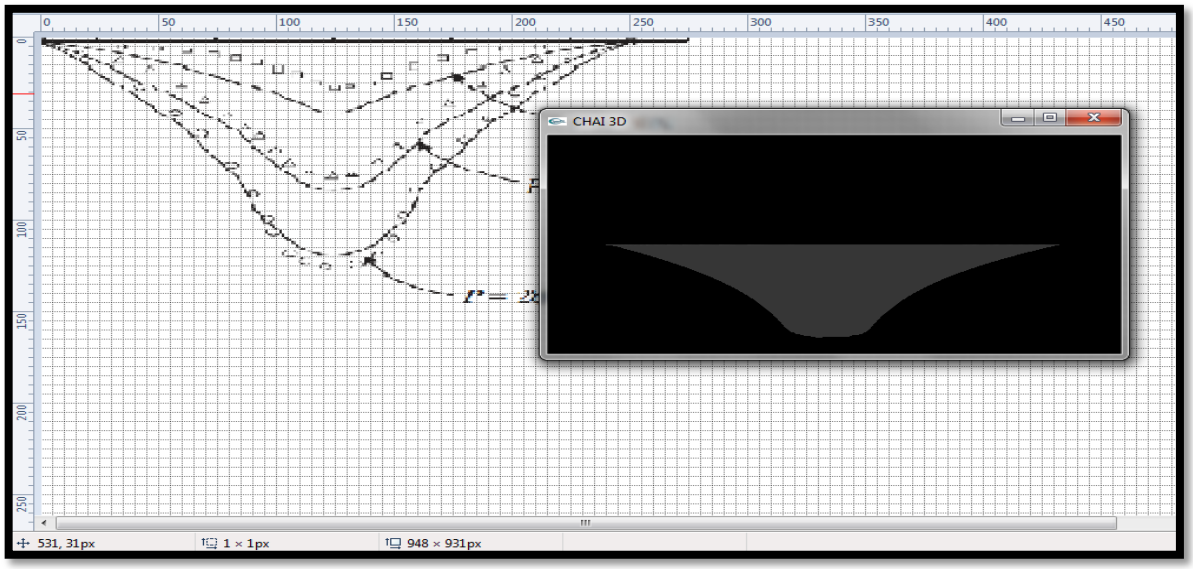

Figura F. 1 – Ajuste do gráfico de Selvadurai no programa Paint do *Windows*® e janela com a simulação em execução.

Tabela F.1 – Alguns dos valores dos *pixels* da Figura F.1.

Dados para gerar o gráfico das três curvas experimentais de Selvadurai						
Quadrados			Triângulos		Círculos	
X	Y		X	Y	X	Y
0	0		0	0	0	0
6	-1		4	-1	5	-2
19	-2		9	-3	10	-5
30	-3		16	-5	17	-8
41	-4		21	-7	24	-12
46	-5		26	-9	31	-16
54	-6		32	-11	37	-20
58	-7		39	-14	43	-24
66	-8		47	-18	48	-28

Feito o mapeamento por meio das coordenadas, basta plotar um gráfico com esses dados, onde quadrados, triângulos e círculos formam uma curva distinta cada, nessas

condições formaremos o gráfico da Figura F.2, esse gráfico demonstra o comportamento da membrana quando ela é deformada no centro, e o gráfico da Figura F.3, demonstra o comportamento da mesma membrana quando ela é deformada na região entre a borda e o centro. Como pode ser notado, o resultado é muito próximo do gráfico gerado por Selvadurai, e dessa maneira, podemos agora realizar uma comparação entre o experimento real e a simulação. No nosso caso, utilizamos o programa Gnuplot, que possui distribuição livre, para plotar todos os gráficos do trabalho. Feita a aquisição dos gráficos necessários à comparação entre o experimento e a simulação, deve-se fazer a leitura *pixel* a *pixel* também da simulação gerada pelo CHAI 3D para fazer de fato a comparação.

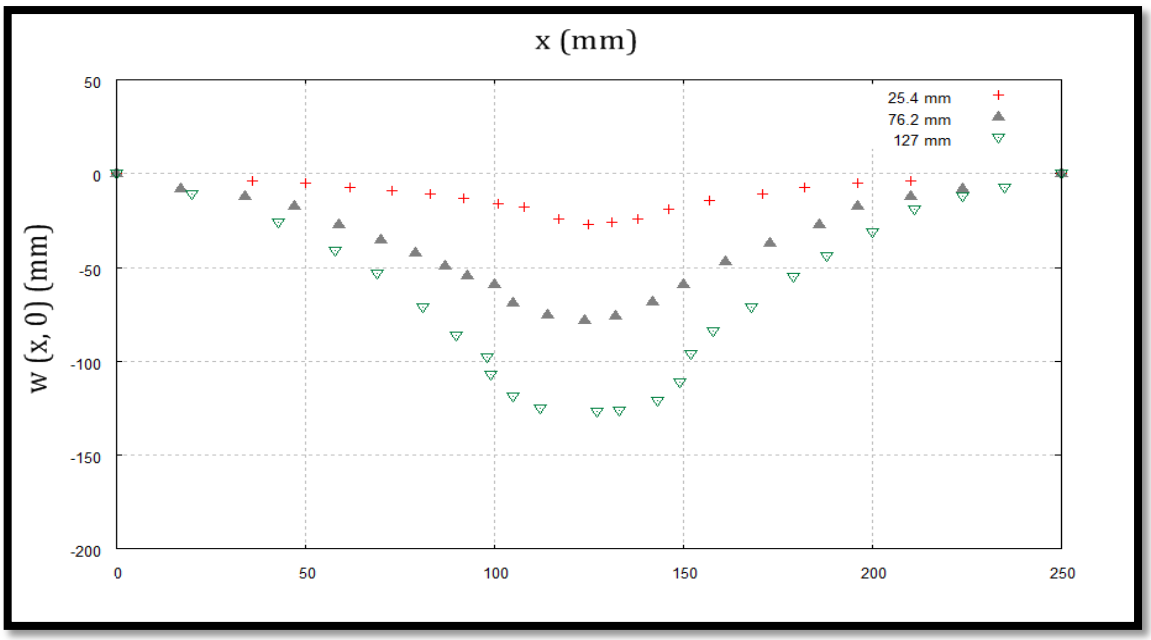

Figura F. 2 – Cópia do gráfico (leitura *pixel* a *pixel* da imagem) que foi gerado por Selvadurai, devido a uma deformação realizada no centro de uma membrana de borracha homogênea.

Antes de prosseguirmos com a finalização da comparação entre o experimento real e a simulação, geramos algumas simulações, recortamos e posicionamos lado a lado com as imagens capturadas no experimento real, o que gerou algo como o exposto na Figura F.4. Note que a nossa simulação não produz uma esfera que pressiona a membrana, ou seja, um objeto com volume, e sim, um disco, como uma pequena moeda que pressiona a membrana no seu centro. Dessa forma, a comparação ocorre com relação ao centro da esfera (prova) utilizada por Selvadurai em seu experimento, por isso, a linha tracejada delimita essa região do espaço entre as duas deformações. Feita essa observação, agora devemos montar uma tabela com os dados referentes ao contorno da simulação gerada no CHAI 3D.

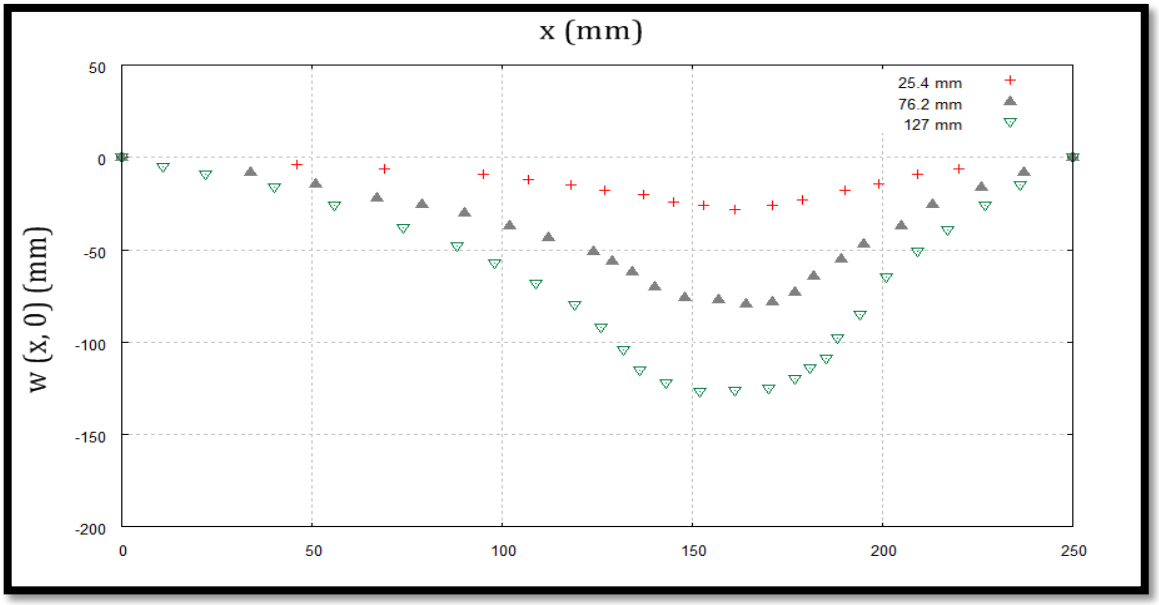

Figura F. 3 – Cópia do gráfico (leitura *pixel* a *pixel* da imagem) que foi gerado por Selvadurai, devido a uma deformação realizada fora do centro de uma membrana de borracha homogênea.

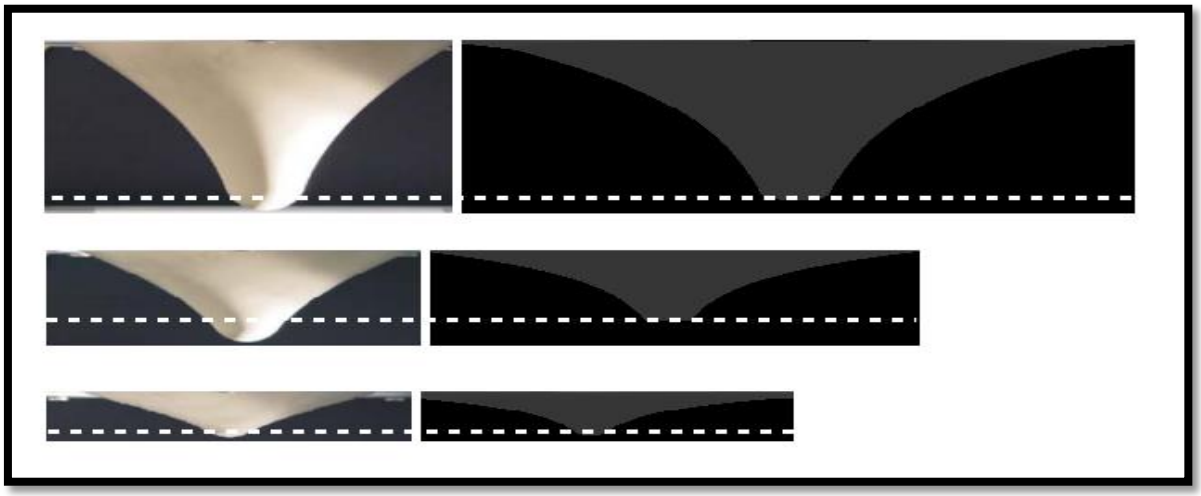

Figura F. 4 – Deformação real (membrana da esquerda) comparada com a deformação gerada pela simulação no CHAI 3D (membrana da direita).

F.2 AQUISIÇÃO DOS DADOS PARA A SIMULAÇÃO

Veja novamente a Figura F.1 e note que existe uma janela com a execução da simulação em andamento, posicione essa janela da simulação de forma a sobrepor o gráfico de Selvadurai, tomando o devido cuidado com os valores do diâmetro e das três deformações geradas no experimento real.

Após realizar a sobreposição entre a imagem experimental e a simulação, tire uma cópia da tela do computador pressionando novamente a tecla “PrintScr” no teclado do computador, abra uma nova janela do *Paint* e cole a imagem capturada da tela referente à sobreposição do experimento e da simulação. Observe na Figura F.5, o resultado da sobreposição da simulação da deformação de tamanho 25.2mm e do ajuste das medidas padrões. Ajuste a visualização da janela do programa em 600% para marcar facilmente cada *pixel* e delimitar a região compreendida pela deformação da membrana em cada imagem. Esse recurso nos permite mapear a figura pintando os *pixels* existentes na borda da membrana deformada de uma cor que os mesmos fiquem evidentes o suficiente para que possamos enxergá-los e distingui-los uns dos outros. Construímos novamente uma tabela com as sucessivas posições de x e y, e posteriormente plotamos os dados para gerar um gráfico para comparar os resultados da nossa simulação com os resultados obtidos por Selvadurai. Será necessário realizar esses passos para cada uma das três curvas de cada uma das duas posições em que Selvadurai realizou seu experimento, dessa forma, teremos produzido ao final, seis arquivos contendo os dados também para seis curvas diferentes, que deverão ser plotados e comparados com os dados referentes ao experimento real.

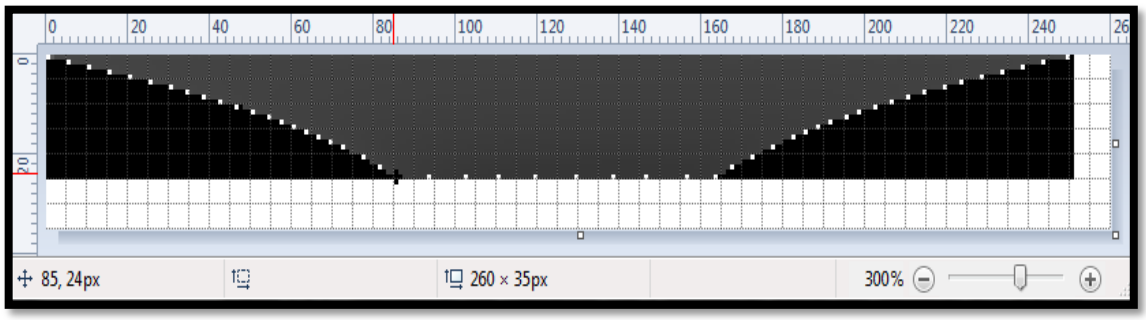

Figura F.5 – Recorte da janela do programa *Paint*, em edição a cópia da tela com a simulação da membrana sendo deformada até a posição 25.2mm.