Edivaldo José da Silva

SIMULAÇÃO ANALÍTICA DA DEFORMAÇÃO DE SUPERFÍCIES COM REALISMO: ESTUDO DE CASO DO OLHO HUMANO

BRASÍLIA
JUNHO, 2017.

Universidade de Brasília
FUP – Faculdade UnB Planaltina
Programa de Pós-Graduação em Ciência de Materiais

Edivaldo José da Silva

SIMULAÇÃO ANALÍTICA DA DEFORMAÇÃO DE SUPERFÍCIES COM REALISMO: ESTUDO DE CASO DO OLHO HUMANO

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 Ciências de Materiais (Área de Concentração: Modelagem e Simulação).

Orientador: Ivan Ferreira da Costa

BRASÍLIA
JUNHO, 2017.

Ficha catalográfica elaborada automaticamente, com
os dados fornecidos pelo(a) autor(a)

SSI586s

Silva, Edivaldo José da
SIMULAÇÃO ANALÍTICA DA DEFORMAÇÃO DE SUPERFÍCIES COM REALISMO: ESTUDO DE CASO DO OLHO HUMANO / Edivaldo José da Silva; Orientador Ivan Ferreira da Costa. -- Brasília, 2017.
84 p.

Dissertação (Mestrado - Mestrado em Ciência de Materiais) -- Universidade de Brasília, 2017.

1. Realidade virtual. 2. Solução Analítica. 3. Deformação de Membrana. 4. Olho Humano. 5. Realismo Físico em Tempo Real. I. Costa, Ivan Ferreira da, orient. II. Título.

Edivaldo José da Silva

SIMULAÇÃO ANALÍTICA DA DEFORMAÇÃO DE SUPERFÍCIES COM REALISMO: ESTUDO DE CASO DO OLHO HUMANO

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 junho de 2017.

BANCA EXAMINADORA

Professor Doutor Ivan Ferreira da Costa - Presidente da Banca
Universidade de Brasília – Faculdade UnB Planaltina – (FUP)

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

Professor (a) Doutor (a) Ceres Nunes Resende Oyama – Membro Externo ao Programa
Universidade de Brasília – Campus Darcy Ribeiro – (UnB)

A Deus, por ser extremamente paciente e piedoso comigo.

As mulheres da minha vida. Maria minha mãe e minha adorável filha Lara.

AGRADECIMENTOS

Ao Professor Orientador Dr. Ivan Ferreira da Costa, braço amigo de todas as etapas deste trabalho que sempre me incentivou a estudar mais para dar maior qualidade à minha dissertação.

E ao Professor Dr. Armando Mendonça Maroja, incentivador e apoiador que clareou meus pensamentos durante os momentos de escuridão.

A minha família, meus irmãos e sobrinhos, pela confiança e motivação. Em especial a minha mãe Maria de Moura Silva, pessoa que foi a maior responsável por todos os passos que dei e darei na vida, pois com ela aprendi o mais difícil: o primeiro passo. Minha filha Lara, pela paciência nos momentos de ausência e pelos sorrisos que facilmente brotavam no rosto quando estamos juntos.

Ao Cristo Jesus, força e motivo de nossa incessante busca por sabedoria.

Aos amigos e colegas, em especial posso citar Fernanda Lourenço, Carlos Alberto, Rafael Costa, pela força e pela vibração em relação a esta jornada.

Aos professores da UnB Campus Planaltina, em especial Alex Fabiano Cortez Campos e Paulo Eduardo Brito, sempre dispostos a ajudar. Aos colegas de Curso, pois juntos trilhamos uma etapa importante de nossas vidas.

A secretaria do Programa de Pós-Graduação em Ciência de Materiais, em especial Jorivê Sardinha da Costa e Aristides Álvares Dourado Júnior, sempre dispostos e atendendo prontamente quando solicitados.

A todos que, com boa intenção, palavras amigas e orações, colaboraram para a realização e finalização deste trabalho. Aos companheiros de cavalgada pelos momentos de descontração e muitos sorrisos.

Agradeço também ao Programa de Pós-Graduação em Ciência de Materiais.

“Há um ditado chinês que diz que, se dois homens vêm andando por uma estrada, cada um carregando um pão, ao se encontrarem eles trocam os pães; cada um vai embora com um. Porém, se dois homens vêm andando por uma estrada, cada um carregando uma idéia, ao se encontrarem, trocam as idéias; cada um vai embora com duas. Quem sabe, é esse mesmo o sentido do nosso fazer: repartir idéias, para todos terem pão...”

Mário Sérgio Cortella.

RESUMO

Este trabalho apresenta a simulação analítica utilizada para deformação de superfícies esféricas preenchidas com líquido em seu interior com realismo físico. A solução analítica aqui desenvolvida garante o realismo físico e a preservação de volume em superfícies fechadas cheias de liquido. Implementamos esta solução analítica em realidade virtual que foi aplicada a um modelo tridimensional do olho humano. O modelo do olho é aproximado de uma superfície fechada cheia de líquido e com volume constante. Através da utilização de ambiente virtual, deformamos a superfície do olho, com a simulação, visualização e interação tátil tridimensional. A simulação consiste na aplicação de força sobre a superfície do olho, que por sua vez pode ser manipulado e deformado com a utilização de uma plataforma gráfica. Já as imagens serão geradas por meio da utilização de um monitor de computador, com isso, obtemos sensações visuais. O retorno de força (sensações táteis) em tempo real é obtido por meio da manipulação do dispositivo háptico virtual. O presente trabalho de mestrado está em consonância com uma tendência mundial que é a busca de ferramentas eficientes com uso de simuladores para a educação, aperfeiçoamento, treinamento médico, processos de tomada de decisão em pré-cirurgias e cirurgias guiadas por imagem. Podemos a partir desse trabalho, como aplicação futura, desenvolver simuladores com execução em tempo real aceitável, uma vez que o modelo apresentado demonstra-se eficiente para aplicações médicas e biológicas utilizando a tecnologia de realidade virtual com a finalidade de ensino em procedimentos cirúrgicos e de diagnóstico de patologias.

Palavras chave: membrana, deformação, solução analítica, conservação de volume.

ABSTRACT

This work focuses on the analytical solution for presentation filled spherical surfaces with or without liquid. The analytical solution developed here guarantees physical realism and the preservation of volume on closed surfaces filled with liquid. We implemented this virtual reality analytical solution that was applied to a three-dimensional model of the human eye. The model of the eye is approximated to a closed surface filled with liquid and with a constant volume. Through the use of virtual environment, we deform the surface of the eye, with simulation, visualization and tactile three-dimensional interaction. The simulation consists of the application of force on the surface of the eye, which in turn can be manipulated and deformed using a graphic platform. The images will be generated through the use of a computer monitor, with this, we obtain visual and tactile sensations. The real-time force return is obtained through the manipulation of the virtual haptic device. This project is in line with a global trend that is the search for effective tools to use simulators for education, improvement and medical training, decision-making processes in pre-surgeries and image-guided surgeries. From this work, as a future application, we can develop simulators with acceptable real-time execution, since the model presented is efficient for medical and biological applications using virtual reality technology for the purpose of teaching surgical procedures and diagnosis of pathologies.

Keywords: membrane, deformation, analytical solution, volume conservation.

RESUMEN

Este trabajo presenta la simulación analítica utilizada para deformación de superficies esféricas rellenadas con líquido en su interior con realismo físico. La solución analítica aquí desarrollada garantiza el realismo físico y la preservación de volumen en superficies cerradas llenas de líquido. Hemos implementado esta solución analítica en realidad virtual que se ha aplicado a un modelo tridimensional del ojo humano. El modelo del ojo es aproximado a una superficie cerrada llena de líquido y con volumen constante. A través del uso de ambiente virtual, deformamos la superficie del ojo, con la simulación, visualización e interacción táctil tridimensional. La simulación consiste en la aplicación de fuerza sobre la superficie del ojo, que a su vez puede ser manipulada y deformada con la utilización de una plataforma gráfica. Las imágenes se generarán mediante el uso de un monitor de ordenador, con lo que obtenemos sensaciones visuales. El retorno de fuerza (sensaciones táctiles) en tiempo real se obtiene por medio de la manipulación del dispositivo háptico virtual. El presente trabajo de maestría está en consonancia con una tendencia mundial que es la búsqueda de herramientas eficientes con uso de simuladores para la educación, perfeccionamiento, entrenamiento médico, procesos de toma de decisión en pre-cirugía y cirugías guiadas por imagen. Podemos a partir de ese trabajo, como aplicación futura, desarrollar simuladores con ejecución en tiempo real aceptable, una vez que el modelo presentado se demuestra eficientemente para aplicaciones médicas y biológicas utilizando la tecnología de realidad virtual con la finalidad de enseñanza en procedimientos quirúrgicos y de diagnóstico de patologías.

Palabras clave: membrana, deformación, solución analítica, conservación de volumen.

LISTA DE SIMBOLOS

u	deformação da membrana no eixo vertical
a	raio do disco da prova
b	raio da borda fixa da membrana
ΔF	força aplicada em um ponto da membrana
P	pressão aplicada sobre a membrana
ΔA_0	área (dx, dy)
τ	coeficiente de tensão superficial da membrana
θ	ângulo de r com a horizontal.
C	uma constante qualquer
r	módulo das coordenadas x e y (vetor posição) da membrana, onde $r = \sqrt{x^2 + y^2}$
V	volume do objeto
L	espessura da membrana
∇	operador de divergência para o cálculo de vetores
u_0	posição da aresta da prova que empurra a membrana com relação ao eixo vertical
$u(r)$	deformação da membrana em *r*
u_H	solução homogênea para *u*
u_P	solução particular para *u*

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

Um dos objetivos atuais da computação gráfica, da interação háptica, da robótica, e da realidade virtual (RV) é simular o comportamento das membranas elásticas (MEIER et., al., 2005).

Na simulação de membranas para aplicações interativas é exigido que os tecidos moles reajam às forças aplicadas de forma estável, realista e em tempo real. No entanto, é difícil lidar com esses requisitos conflitantes. Por exemplo, a taxa de processamento de um dispositivo háptico deve ser executada com estabilidade a taxas de simulação superiores a 1 kHz e com deformações o mais real possível (HO, A. K. et. al., 2012).

As frequências visual e tátil (30 imagens, 1000 amostras de força por segundo) devem ser alcançadas para evitar atrasos ou lentidão na animação. Além disso, o tempo de simulação deve sempre ser sincronizado com o tempo físico, independentemente da plataforma computacional.

Diversas abordagens foram desenvolvidas com o intuito de animar objetos deformáveis (GIBSON, 1997; NEALEN 2006), mas apenas alguns modelos podem ser usados quando se pretende um desempenho em tempo real.

Animar membranas deformáveis em tempo real é um assunto importante para muitas aplicações interativas, como a simulação de tecidos, sistemas para simulação de cirurgias e treinamento médico e jogos (vídeos-game), bem como sua utilização em pesquisas na área de ciências de materiais.

Muitos órgãos humanos podem ser modelados essencialmente por biomembranas elásticas para algum grau de precisão. Bons exemplos deste tipo de órgãos são a vesícula biliar (MOUTSOPOULOS, 1997, WEBSTER, 2003), a bexiga urinária (BOUBAKER, 2009, CHI, 2006), o ducto biliar (BASDOGAN, 2001), o estômago (KUHNAPFEL, 2000), o intestino (PHAN-THIEN, 1989), o olho (WEBSTER, 2004), sistema vascular (LUBOZ, 2009, ALDERLIESTEN, 2007).

Um modelo preciso de tecidos moles pode ser usado eficazmente em várias aplicações diferentes, como planejamento cirúrgico, cirurgia guiada por imagem, cirurgia minimamente invasiva, telecirurgia e treinamento médico.

Para a cirurgia guiada por imagem, a deformação do tecido mole pode comprometer a precisão da cirurgia com base em imagens pré-operatórias. Uma maneira de superar essas limitações é combinar modelos biomecânicos com escassos dados intra-operatórios, a fim de realisticamente deformar a imagem pré-operatória para coincidir com a situação cirúrgica. Um modelo de deformação com base na física é útil para atualizar as imagens concorrentes com a cirurgia, a fim de conseguir um registro não rígido entre o órgão e as imagens pré-operatórias.

No planejamento cirúrgico a simulação pode possibilitar a visualização dos resultados potenciais ou auxiliar na tomada de decisões pré-operatórias sobre as opções cirúrgicas (DEL PALOMAR, 2008, ROOSE, 2005, BALANIUK, 2006, WILLIAMS, 2003). Por exemplo, na cirurgia plástica a simulação pode ser útil para visualizar o resultado da cirurgia reconstrutiva da mama (mamoplastia). A capacidade de visualizar os resultados potenciais da cirurgia e tomar decisões sobre suas opções cirúrgicas é muito importante para pacientes e para os cirurgiões em cirurgia plástica.

Outra aplicação médica que a deformação realista de membranas virtuais pode ser útil é na formação médica. Os simuladores médicos permitem aos alunos praticar e ajudar a prever e planejar alguns procedimentos cirúrgicos.

Apresentamos nesta dissertação a solução analítica para deformação de superfícies com realismo físico e conservação de volume. Temos o modelo de um olho tridimensional, que pode ser aproximado de uma superfície fechada cheia de líquido e que manipulamos em ambiente virtual. Com a aplicação de força sobre a superfície do olho podemos deformá-lo e seu volume ser conservado. Para garantir a conservação de volume, após a imposição de uma prova, o volume retirado pela prova é distribuído por toda a superfície.

Do ponto de vista de processamento computacional, quando utilizamos uma solução analítica para deformação de superfícies em ambiente virtual temos um ganho em realismo físico. Na solução analítica aqui apresentada, não precisamos fazer inversão de grandes matrizes, ou incorporar ao modelo virtual soluções complicadas, através de métodos tradicionais de engenharia ou modelos matemáticos.

1.1. Objetivos do Trabalho

Este trabalho possui como objetivo principal desenvolver um modelo virtual do olho humano para simular a deformação de superfície com realismo físico.

Assim temos como objetivos específicos para se atingir o objetivo principal:

- Utilizar a plataforma gráfica CHAI3D para desenvolvimento de um modelo virtual de uma membrana semi-esférica.
- Incorporar retorno de força e obter sensações táteis, com realismo físico, por meio da manipulação do dispositivo háptico *PHANTOM Omni®*.
- Determinar a solução analítica para deformação superfícies na forma do olho humano preenchidas com líquido e que mantém volume constante.
- Incorporar soluções analíticas (extremamente rápidas do ponto de vista computacional) na plataforma de simulação virtual.
- Inserir textura no modelo virtual de forma a assemelhar-se graficamente a um olho humano real.

1.2. Justificativa

O presente projeto de mestrado está em consonância com uma tendência mundial. Os novos paradigmas de pesquisa em biomateriais buscam desenvolvimento de ferramentas eficientes com uso de simuladores para a educação, aperfeiçoamento e treinamento médico. Essa busca vem sendo realizada por meio dos conselhos e ou colegiados relacionados à prática médica, como a Conferência MICCAI[1] *(Medical Image Computing and Computer Assisted Intervention)*. Esta conferência visa reunir cientistas, engenheiros, médicos, cirurgiões, educadores e estudantes do mundo inteiro para contribuir e participar na missão e atividades da sociedade, com a preservação e promoção de pesquisa, educação e prática no

[1]A Sociedade MICCAI foi formada como uma corporação sem fins lucrativos em 29 de julho de 2004. O nome corporativo oficial é *The Medical Image Computing and Computer Assisted Intervention Society* (“The MICCAI Society”) (http://www.miccai.org/, 2015).

campo da computação de imagens médicas e assistida por computador intervenções médicas (MICCAI, 2015).

Na conferência MICCAI 2015, onde se realizou o 2º Workshop sobre Análise de Imagem Médica Oftálmica (OMIA 2015) deu-se grande relevância ao tema de estudo deste projeto ressaltando que:

> *"O custo da cegueira para a sociedade e indivíduos é enorme, e muitos casos pode ser evitado por uma intervenção precoce. Estratégias de diagnóstico precoce e de confiança e tratamentos eficazes são, portanto, uma prioridade mundial."* (JIANG, Jimmy Liu, *Institute for Infocomm Research*, A * STAR, Singapura – 0MIA – MICCAI, 2015 – MUNICH- Alemanha)

É crescente a utilização de simuladores cirúrgicos como uma alternativa no processo de educação e treinamento médico. Diversos profissionais em conferências buscam afirmação de que o treinamento deve ocorrer primeiro em simuladores (sala de operação virtual). Quando adquiridas e comprovadas certas habilidades (corte, sutura, punção, entre outras) o estudante estará apto para iniciar o treinamento em centro cirúrgico (BASDOGAN et. al., 2007).

Temos como justificativa para esse trabalho o treinamento e melhor capacitação cirúrgica antes e durante programas de treinamento de residência médica, sem a utilização de cobaias e cadáveres, bem como realização de procedimentos cirúrgicos à distância. Ainda o treinamento médico com a manipulação de órgãos humanos e biomembranas, tais como tímpano, mama, bexiga, fígado, olhos, veias e artérias e possível identificação e detecção de melanomas benignos ou malignos.

1.3. Disposição do Trabalho

Esta dissertação possui, além desta introdução, seis capítulos, a saber:

Capítulo 2 – Apresenta o estado da arte da deformação de membranas e os conceitos básicos de realidade virtual.

Capítulo 3 – Conceitua o olho humano apresentando suas características físicas e propriedades elásticas.

Capítulo 4 – Apresenta uma solução analítica para deformação de membranas homogêneas preenchidas com líquido em seu interior com preservação de volume.

Capítulo 5 – Apresenta os materiais, métodos utilizados e resultados alcançados no desenvolvimento e implementação do modelo virtual de deformação de membranas com realismo físico para o olho humano.

Capítulo 6 – Apresenta as discussões e conclusões.

Em seguida são apresentadas as referências que serviram de embasamento teórico para a elaboração desta dissertação e os apêndices que documentam o modelo de simulação construído.

2. ESTADO DA ARTE EM DEFORMAÇÃO DE MEMBRANAS

2.1. Introdução

Apresentamos neste capítulo, o estado da arte para em deformação de membranas, abordando as principais técnicas de deformação. Como se trata de um modelo para deformação de superfície com conservação de volume, utilizamos inicialmente um objeto deformável em configuração de equilíbrio, ou seja, em repouso e sem atuação de forças sobre ele. Ao aplicarmos uma força sobre a superfície do olho, esta se deforma assim, a partir de um monitor de computador, se obtém por meio da plataforma gráfica CHAI3D[2] e com interação do dispositivo de retorno de força *PHANTOM Omni*®[3] sensações visuais e táteis.

2.2. Técnicas de Deformação

Quando desenvolvemos uma simulação em um ambiente virtual, podemos inserir vários objetos deformáveis e rígidos. Os objetos deformáveis podem interagir com outros objetos durante a simulação mudando sua forma, já os rígidos possuem forma fixa desde o princípio até o fim da simulação. Ambos podem ter outras características modificadas durante a simulação como cor, posição ou textura.

Para a deformação utilizamos técnicas de modelagem geométrica (simulação computacional), onde ao se deformar um objeto, estaremos aplicando força nos pontos que formam o objeto produzindo dobras, compressão ou estiramento, os objetos podem ser gerados em 1D (linhas), 2D (quadrados e triângulos) e elementos com volume em 3D (hexaedros e tetraedros). (PAVARINI, 2006).

COSTA (2013) cita que a maioria de objetos do mundo real possui comportamento dinâmico e podem mudar de posição, cor e forma, sofrendo ações e reações quando estudados em um ambiente virtual. Sendo assim, para possibilitar a interação com o usuário em RV, estes objetos podem ter seus atributos modificados utilizando cálculos em tempo real,

[2]CHAI 3D é um código aberto que possui um conjunto de bibliotecas em linguagem de programação C++.
[3]Dispositivo háptico produzido pela empresa *SensAble Technologies (PHANTOM Omni, 2017)* ®.

proporcionando modificações na estrutura de acordo com as propriedades elásticas e constantes de cada material, gerando o que chamamos de deformação.

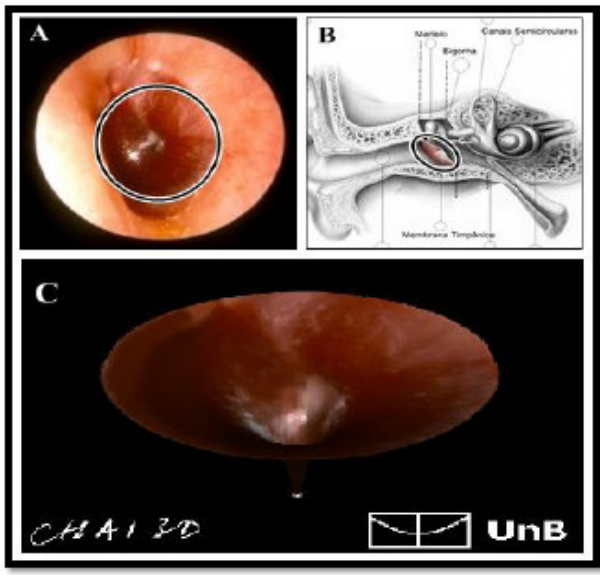


Figura 1: Simulador virtual de deformação de biomembranas (COSTA, R. S.; COSTA, I. F., 2012).

Diversas são as técnicas que podem ser usadas para deformação, com solução analítica implementada através de simulações geométricas e físicas: solucionando equações geométricas; estudo de aplicação de força física; conceitos de massa e aceleração ou através da junção destes dois tipos de simulação dando origem a técnicas híbridas. Diversas são as técnicas de simulação em ambientes virtuais dentre as quais: Deformação de Livre Forma *(Free Form Deformation* – FFD), Método dos Elementos Finitos (*Finite Element Method* – FEM) ou e Massa Mola (*Mass Spring* – MS). (LOTTI, PAVARINI, 2006, PAMPLONA, 2014).

O método FFD é uma técnica geométrica onde o objeto da cena é envolvido em uma grade 3D que sofrerá a deformação, e por consequência o objeto envolvido por ela também deformará de acordo com a deformação da grade que o envolve. Este método em ambientes virtuais pode gerar a deformação da imagem e, para o caso específico de treinamento médico virtual, ocasiona na distorção ou modificação das proporções do órgão no qual se modela. (PAVARANI, 2006, COSTA, 2013). Dessa forma esse método não se mostra adequado para aplicações médicas.

A utilização do Método de Elementos Finitos é bem aceita como uma ferramenta de solução rápida para deformação em ambientes virtuais para simulação e aplicação de modelos na área de Ortodontia (LOTTI, 2006). Sua aplicação ocorre por meio de uma análise

matemática em que consiste a divisão do meio contínuo em pequenos elementos, onde se mantém as propriedades do problema original. Os elementos são resolvidos por modelos matemáticos e descritos por equações diferenciais para se chegar ao resultado desejado (PAVARINI, 2006, COSTA, 2013). PAMPLONA (2014) utiliza o Método de Elementos Finitos por se tratar de um bom modelo de solução de resposta rápida para execução de sucessivas expansões de tecido da pele humana.

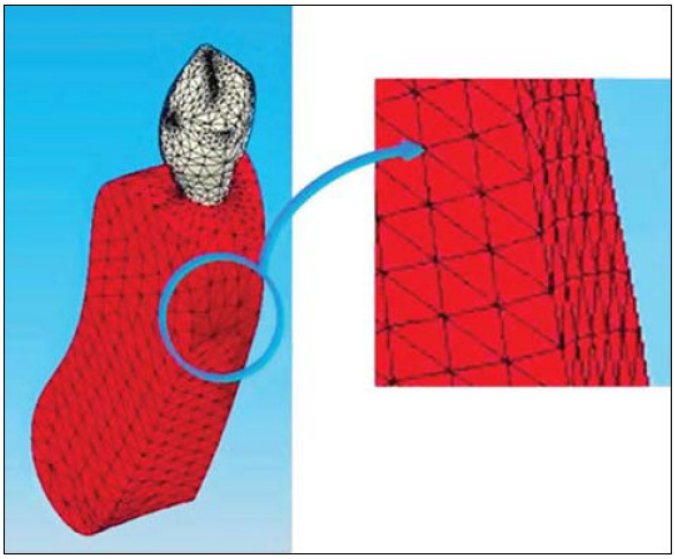


Figura 2: Malha do canino e osso alveolar implementada com a utilização de elementos finitos. Cada triângulo representa divisão do meio contínuo em pequenos elementos (LOTTI, 2006).

O terceiro método citado é muito utilizado na literatura é conhecido como método Massa Mola, sendo caracterizado pela abordagem de objetos conectados por molas e são deformáveis a partir de nós, cada um com sua massa e que podem fornecer forças ao nó vizinho. (PAVARINI, 2006, DOS SANTOS MACHADO, 2008, COSTA, 2013).

Utilizamos neste trabalho com solução analítica, já que nos permite alcançar uma solução rápida, ou seja, que produza resultados em tempo real. A utilização de modelos baseados em elementos finitos, modelos de massa-molas e abordagens híbridas, quando utilizadas em deformação de objetos virtuais possuem restrições quanto à utilização em ambientes virtuais, implicando diretamente na capacidade de processamento e diminuição do realismo na cena virtual.

2.3. Deformação de Membranas

A deformação de membranas tem sido um tema amplamente estudado na área de ciências de materiais com diversas aplicações em materiais biomédicos, tais como a expansão da pele humana (PAMPLONA, 2014) deformação de membrana do tímpano (COSTA e COSTA, 2013), aplicações em simuladores de cirurgias oftalmológicas (CARVALHO, 2012).

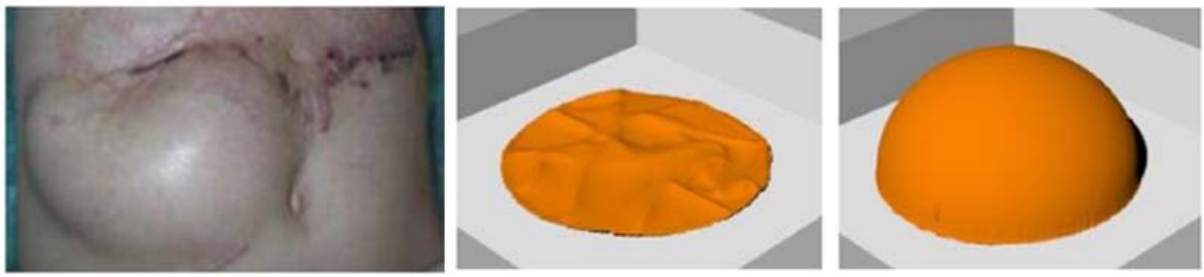


Figura 3: Fotografia de uma expansão de pele (esquerda), superfície digitalizada de expansor de pele sem líquido (centro) e inflado com liquido (direita). (PAMPLONA, 2014)

OLIVEIRA FILHO (2014), realizou em seu trabalho estudo analítico para deformação de membranas com realismo físico, com deformação de membranas elásticas circulares preenchidas ou não com líquido. Através de ensaios experimentais validou o estudo, que busca servir como base no desenvolvimento de sistemas para deformação em diversas aplicações que representem tecidos moles a fim de obter características elásticas de membranas.

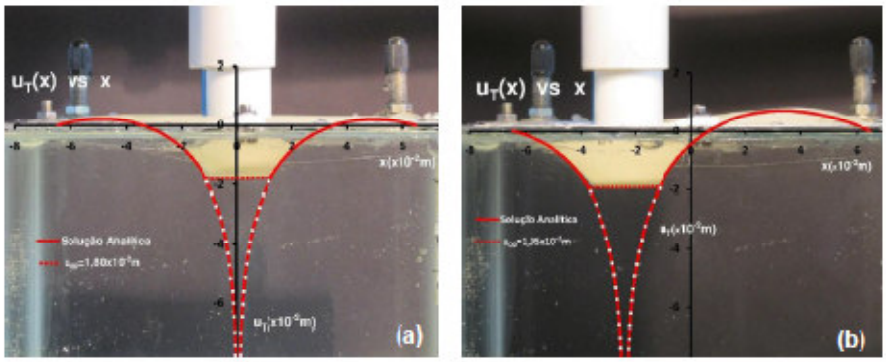


Figura 4: Estudo da deformação da membrana com a caixa transparente cheia de água. Fonte: G. B. OLIVEIRA FILHO, A. M. MAROJA, R. S. COSTA, I. F. COSTA– Deformação de membranas para uso em sistemas de simulação: uma validação experimental de soluções analíticas com realismo físico – Congresso Brasileiro de Engenharia e Ciência dos Materiais, CBECIMAT, 2014.

2.4. Simulação em Realidade Virtual

O conceito de realidade virtual parte da idéia de interação homem-computador, onde através de sensações visuais tenta-se reproduzir uma situação real utilizando meios tecnológicos, com retornos através de interfaces como o monitor de computador, hápticas (retorno de força), auditivas e até olfativas (CARVALHO, 2012). Para aplicações médicas têm sido amplamente utilizado como uma ferramenta de apoio no ensino e treinamento médico. O uso de simuladores em diversos campos tem possibilitado uma maior fidelidade nas técnicas de ensino, onde além do treinamento, pode ser diminuído o uso de cobaias e de cadáveres. O aluno de medicina consegue através de um simulador virtual treinar em situações que se assemelhem ao ambiente real de trabalho (DOS SANTOS MACHADO, 2006).

COSTA (2007) ressalta a importância da utilização de simuladores virtuais com o intuito de reproduzir situações do mundo real. A simulação de objetos tridimensionais permite planejar e testar novas técnicas de treinamento em profissionais de diversas áreas. Na área médica, por exemplo, o treinamento e reciclagem de profissionais pode ser possível com o uso de simulação, o que já ocorre no treinamento de pilotos de aviões e astronautas. Além disso, a simulação torna-se um método eficiente, seguro e de baixo custo no treinamento e planejamento de cirurgias.

Figura 5: Cirurgia real no olho humano (esquerda) e simulação de cirurgia em modelo de olho virtual (direita) (WAGNER et. al., 2002)

WAGNER et. al. (2002), aborda o conceito de utilização de simuladores em realidade virtual, como a construção de sistemas em que o usuário possa esquecer que está trabalhando em um ambiente virtual e executar suas ações naturalmente como se estivesse trabalhando no mundo real. Com isso, torna-se necessário o envolvimento de todos os sentidos do usuário, a fim de alcançar este resultado, onde ocorra a fusão entre a percepção sensorial e real do indivíduo, criando uma total interação entre o real e o computacional.

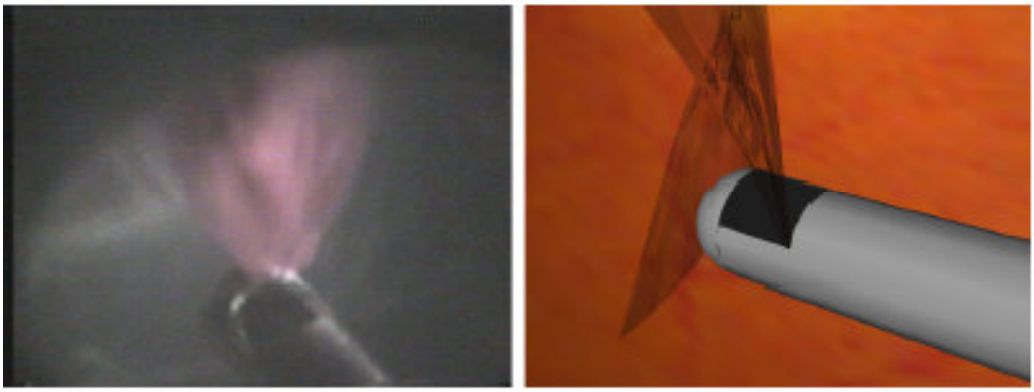


Figura 6: Imagem real capturada de procedimento cirúrgico (esquerda) e a visão durante simulação de procedimento cirúrgico (direita) (WAGNER et. al., 2002).

Realidade virtual é conceituada por DOS SANTOS MACHADO (2006), como a utilização de interface humano-computador com múltiplos canais sensoriais, onde é criado um ambiente artificial que oferece imersão e interação do indivíduo, com a utilização de computadores, aplicações em física, gráficos eletrônicos e explorando os sentidos da audição, visão e tato (COSTA, 2013).

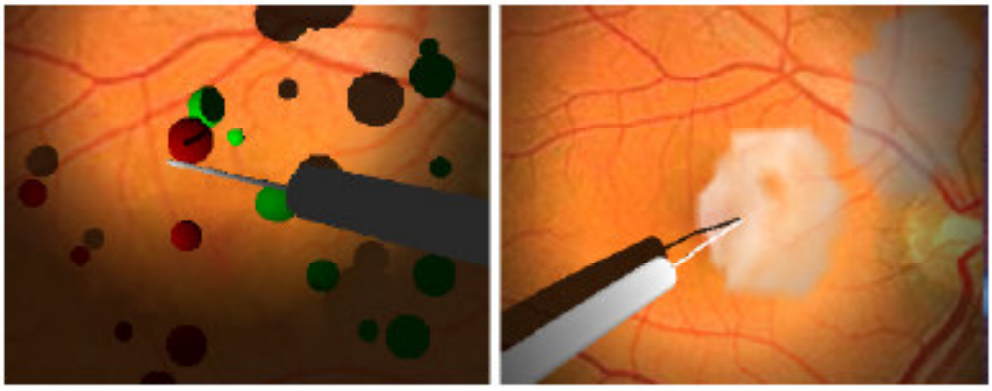


Figura 7: Treinamento de cirurgia de retina no simulador virtual (WAGNER et. al., 2002)

2.5. Dispositivos Hápticos

Dispositivos hápticos, são braços robóticos que permitem a interação com ambientes tridimensionais, que rastreiam a posição das coordenadas (*x, y* e *z*) de acordo com o movimento do usuário e tem forças de retorno calculadas pelo simulador em enviadas ao disposto háptico. Simula diversas aplicações, por exemplo, o dedo do médico, na etapa tátil do exame de colo de útero (DOS SANTOS MACHADO, 2006).

Podemos utilizar a interação háptica em várias modalidades de procedimentos médicos que envolvam manipulação de instrumentos específicos, como cateter, endoscópio e agulhas, entre elas, como aborda em seu trabalho CORRÊA (2008), sua utilização na simulação de anestesia, e demais problemas.

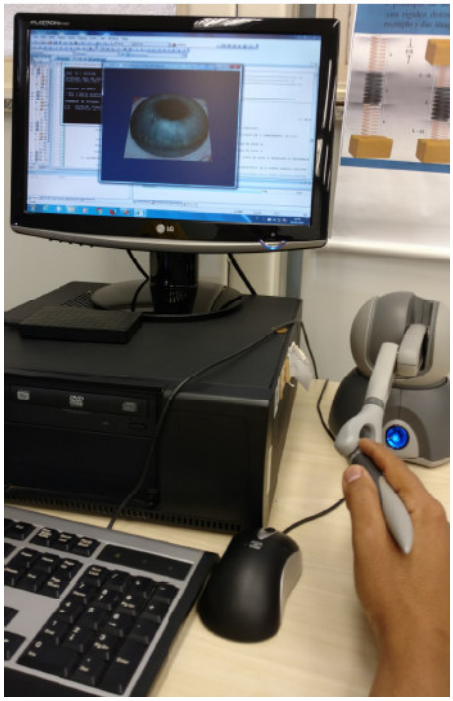


Figura 8: Manipulando olho em 3D com dispositivo háptico (Acervo pessoal)

Uma importante ferramenta para simulação em realidade virtual é o dispositivo háptico da *SensAble Technologies PHANTOM*®, dispositivo tátil que torna possível aos usuários tocar e manipular objetos virtuais. O modelo *PHANTOM Omni* é o mais utilizado por apresentar hoje o melhor custo-benefício em se tratando de dispositivos hápticos (PHANTOM, 2015).

2.6. Chai3D

O CHAI3D segundo MELO, BRASIL e BALANIUK (2007), além de ser capaz de incorporar diversos conceitos de tratamentos de objetos 3D, ainda disponibiliza um conjunto de funcionalidades para manipulação de imagens via interface háptica. Possibilita a conexão com dispositivo háptico, possui algoritmos de colisão, que capturam o cálculo da força aplicada no objeto, no ambiente virtual de simulação a partir de uma área delimitada no ambiente virtual.

Por possuir código aberto, o CHAI3D tem um conjunto de bibliotecas em linguagem de programação C++, que podem, por meio da sua manipulação em computador e através de um compilador, *Microsoft Visual Studio 2008 – MSVC 2008* (como o que foi utilizado nesse trabalho, por exemplo), pode-se criar visualizações e simulações interativas em tempo real. Além disso, o CHAI3D suporta vários dispositivos hápticos hoje comercializados, e que podem possuir três, seis ou até sete graus de liberdade táteis. Sua finalidade é especialmente adequada para fins de educação e pesquisa, onde, oferece uma plataforma com muitos recursos que podem ser modificados pelo usuário conforme a sua necessidade. Tais justificativas nos levaram a escolha do CHAI3D para sua utilização neste trabalho.

Segundo informações do site que disponibiliza acesso ao pacote do CHAI3D na internet, (https://www.chai3d.org/, 2017) ele foi originalmente iniciado no laboratório de inteligência artificial da Universidade de Stanford em 2003. Foi desenvolvido para fornecer um quadro simples e compacto para a introdução da experimentação de dispositivos hápticos para a comunidade de pesquisa.

Com a sincronização do visual e tátil no ambiente virtual e ainda com a possibilidade de informações auditivas, é possível efetuar simulações em alto nível que envolva coordenação olho-mão. O CHAI3D é utilizado como ferramenta de apoio na construção de simuladores cirúrgicos, tem como objetivo ensinar os alunos gestos delicados e avaliar seu desempenho.

Há ainda no pacote do CHAI3D vários modelos prontos, que são fornecidos para que usuário possa manipular e testar cada um em ambiente virtual de forma a facilitar o seu entendimento e modificação. Todos são nomeados e numerados, com a finalidade de indicar sua complexidade, para que à medida que o usuário efetue os testes vá evoluindo em sua utilização.

Figura 9: Treinamento virtual com interface háptica utilizando o Chai3D (CHAI 3D, 2017)

2.7. Aplicações em Realidade Virtual para Estudo de Medicina

Atualmente, existe uma série de aplicações para treinamento médico envolvendo a realidade virtual, dentre os quais trabalhos recentes de simulação para exame de colo de útero (MORAES, 2006), deformação da membrana do tímpano (COSTA, 2013), treinamento em cirurgias oftalmológicas (WEBSTER, 2004 e WAGNER, 2002), simulador de cirurgia otológica (DE SOUSA, OKADA e SUZUKI, 2011), procedimentos de anestesia, (ULLRICH, 2011), e ainda procedimentos de expansão de pele para correção de defeitos e cicatrizes (PAMPLONA, 2014). Devido à complexidade de treinamentos com utilização de cobaias, cadáveres e ainda para segurança dos profissionais de saúde, a construção de simuladores que proporcionem realismo físico aos estudantes de medicina têm ganhado um grande espaço, sendo de grande importância na no ensino e no desenvolvimento de técnicas para procedimentos cirúrgicos.

WAGNER et. al. (2002), afirma que quando se utiliza da simulação de tecido em simuladores médicos o objetivo principal é cumprir dois requisitos básicos: precisão e tempo de resposta computacional. Neste sentido, as imagens são baseadas em aproximações, buscando imitar a propriedade desejada, onde o comportamento modelado esteja ajustado ao observado com menor custo computacional e que possa refletir as propriedades físicas de processos medidos do mundo real.

Ainda se tratando de simuladores em realidade virtual, podemos citar o trabalho de MORAES et. al. (2006), onde através da utilização de um sistema interativo que efetua o treinamento para identificar e capacitar o estudante de medicina no diagnóstico de câncer de colo de útero em mulheres, possibilitando a estes tanto a fase de visualização, quanto de palpação (exame de toque).

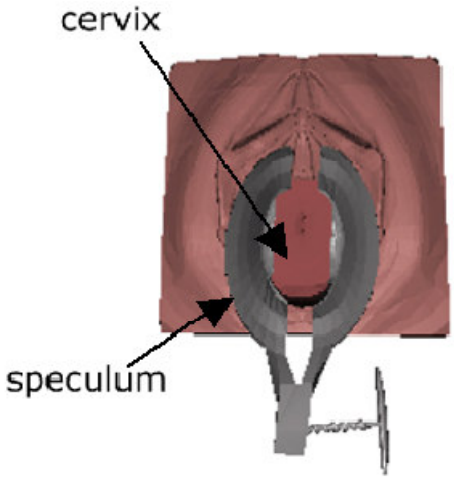


Figura 10: Modelo virtual desenvolvido para observação do colo do útero (DOS SANTOS MACHADO, 2006).

Como ferramenta de treinamento médico em realidade virtual, ULLRICH (2011), apresenta um simulador de anestesia em paciente virtual, onde, diferente da maioria dos softwares comerciais com esse fim, apresenta um banco de dados contendo informações de pacientes, cada um com seu biótipo específico, proporcionando aos médicos residentes uma grande proximidade do real, utilizando dispositivos hápticos para retorno de força e sensações táteis. As imagens são segmentadas e apresentam regiões especificas para treinamento, onde ocorre desde a fase de palpação e escolha do local onde será efetuado o procedimento de anestesia através de uma plataforma de realidade virtual intuitiva, que consiste em guiar o usuário para auto-execução dos procedimentos.

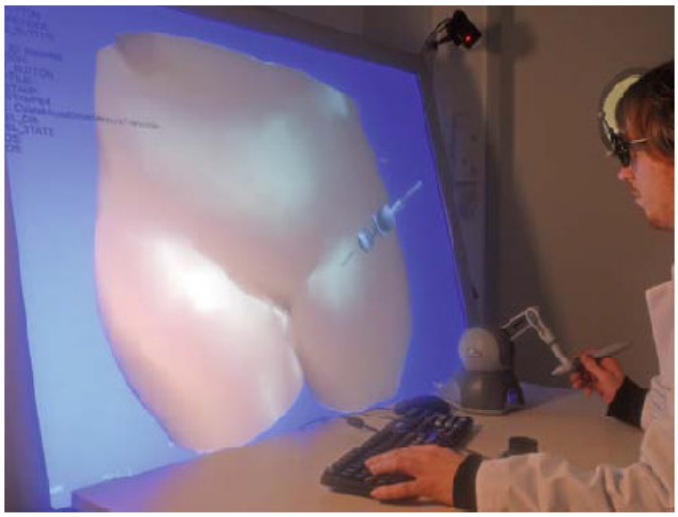


Figura 11: Protótipo de simulador de anestesia baseado em realidade virtual com utilização de dispositivo háptico. (ULLRICH, 2011).

2.8. Jogos de Computador

Nos jogos, apesar da história de mais de três décadas de pesquisa em modelagem deformável em computação gráfica, os resultados obtidos por esta pesquisa raramente foram aplicados em jogos de computador. Como resultado disto, corpos rígidos dominam jogos em geral.

No entanto, mesmo com a metodologia atual, os algoritmos e modelos têm visto aplicação um pouco limitada em ambientes de produção e jogos de vídeo. Uma das razões para isso é a falta de poder computacional: muitas das técnicas apresentadas são inerentemente *off-line* e podem levar horas ou dias para produzir resultados. No campo do entretenimento interativo, já estão sendo implementados pequenos modelos deformáveis com base física, e graças a desenvolvimentos como a Unidade de Processamento de Física (PPU – *Physics Processing Unit*) [4], essa tendência provavelmente continuará.

Uma segunda razão que vemos para uma relutância em adotar esses novos métodos deformáveis é o controle limitado que os usuários têm sobre as animações resultantes.

[4] A Unidade de processamento de Física ou PPU *(Physics Processing Unit)* é um processador dedicado criado especialmente para realizar o processamento de algoritmos de Física complexos em tempo real. Utilizados em modernos jogos eletrônicos que usam cenários em 3 dimensões (FRUSTACI et. al., 2007).

3. OLHO HUMANO

3.1. Características e Propriedades Físicas.

Este capítulo apresenta as principais características do olho humano, bem como suas propriedades físicas e estruturais.

O olho humano pode ser visto como uma esfera com diâmetro aproximado de 20 mm, composto por membranas que o envolvem: a camada externa da esclerótica (branco do olho) que tem a função de proteção e mantém a forma do olho; a córnea que é composta por um tecido resistente e transparente que recobre a superfície anterior do olho; a coróide que é a membrana conjuntiva que recobre e nutre a retina, que por sua vez é a membrana mais interna do olho se estendendo por toda porção posterior do olho humano.

3.2. Estrutura do Olho Humano

A figura abaixo mostra a estrutura básica do olho humano, onde podemos fazer uma analogia inicial apenas por uma esfera preenchida com fluido transparente, na qual a luz incidente penetra por um pequeno orifício (pupila) e incide na superfície oposta a ele, onde está a retina. (HELENE, 2011).

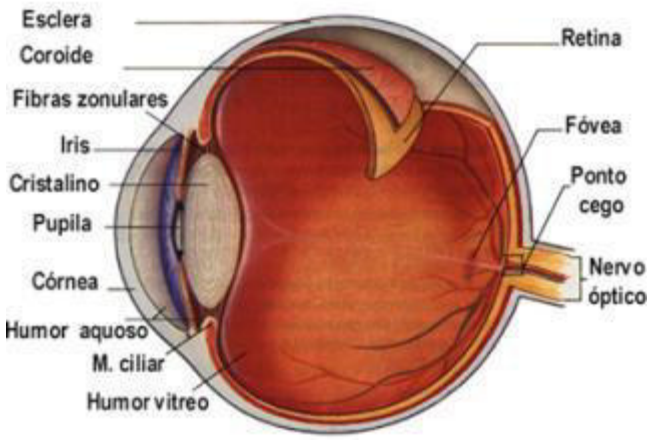


Figura 12: Corte lateral do olho humano (imagem adaptada da internet http://www.museuescola.ibb.unesp.br/subtopico.php?id=2&pag=2&num=2&sub=1*)*

O modelo do olho humano pode ser simplificado como um conjunto de lentes, onde uma lente biconvexa, denominada cristalino está posicionada na região anterior do globo ocular *(Figura 12)*. No fundo do globo ocular encontramos a retina que funciona como um filme sensível a luz, nela que se localizam os receptores de luz (cones e bastonetes). A imagem do mundo externo é percebida através de sensações luminosas que são convertidas em impulsos elétricos e que são levadas ao cérebro onde as imagens são processadas, por meio do nervo óptico.

No interior do olho existem ainda dois tipos de líquido. O humor aquoso é um líquido incolor que existe entre a córnea e o cristalino. O humor vítreo é uma substância gelatinosa que preenche todo o espaço interno do globo ocular entre o cristalino e a retina. Tudo isso funciona para manter a forma esférica do olho. Ambos os humores têm índices de refração muito próximos ao da água, da ordem de 1,34 (HELENE, 2011).

Como o olho pode ser entendido como uma superfície esférica repleta de líquido em seu interior, e o tema central deste trabalho é estudar as configurações de equilíbrio para deformação de membranas sob carga externa, podemos aproximar nosso modelo de olho virtual a uma membrana fechada repleta de líquido em seu interior. Com essa aproximação, podemos utilizar a solução analítica para deformação de objetos volumétricos definidos por membranas fechadas, quando submetidos a uma força aplicada na superfície.

3.3. Simuladores de Realidade Virtual no Estudo da Oftalmologia

O uso de simuladores cirúrgicos para formação de médicos oftalmologistas é sem dúvidas uma revolução no processo de ensino médico e treinamento de profissionais residentes, trazendo maior segurança tanto aos pacientes e professores, quanto para os alunos ainda sem habilidades para executar procedimentos complexos de cirurgias oftalmológicas. Reduz-se consideravelmente o risco de complicações e traumas tanto físicos quanto psicológicos, produzidos por erros dos inexperientes alunos.

Considera-se ainda que estamos em uma época em que a exigência de melhores resultados é crescente (REZENDE et. al. 2012). Afinal, cirurgias inadequadas, quando efetuadas em pacientes reais e realizadas por profissionais ainda inabilitados, podem produzir desafios tanto ao paciente quanto ao aluno e professor, por se tratar em geral de um procedimento complexo (GARCIA, 2003).

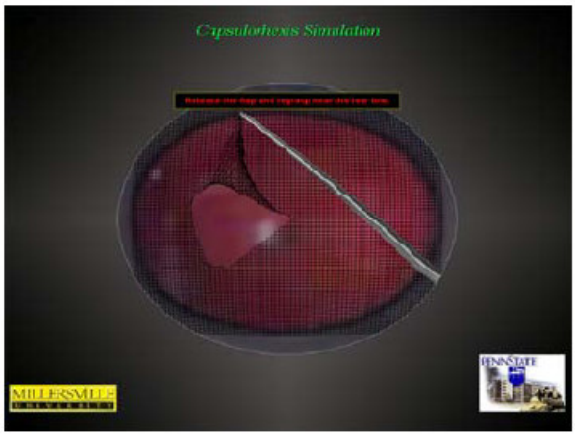


Figura 13: Modificado em massa-molas modelo usado para rasgar a membrana. (WEBSTER et. al., 2004).

Procedimentos de microcirurgia para região do olho são vistos como uma das tarefas cirúrgicas mais exigentes, por ser um procedimento que requer uma precisão submilimétrica, exigindo grande interação entre o olho e a mão do profissional, em um pequeno espaço disponível para trabalho.

O simulador para cirurgia de catarata – *Continuous Curvilinear Capsulorhexis* (CCC) [5]– citado por WEBSTER et. al., (2004), consiste em um modelo capaz de utilizar gráficos 3D, onde, através da utilização de dispositivos hápticos com retorno de força de alta fidelidade por meio de um modelo matemático e com o treinamento através de um monitor de computador, tem-se o intuito de fornecer um treinamento e ambiente de aprendizagem sem risco para pacientes reais.

O modelo do simulador para utilização de uma técnica específica para a cirurgia de catarata apresentado por WEBSTER et. al., (2004), do ponto de vista de simulação e treinamento é uma ferramenta válida, uma vez que dispensa o uso de cobaias e cadáveres. No modelo de simulador descrito em seu artigo (WEBSTER et. al., 2004), utiliza modelos híbridos (modificado de massa-molas) para modelagem de tecido mole em simulação cirúrgica.

[5] *Continuous Curvilinear Capsulorhexis* (CCC) ou *Capsulorrex*é uma técnica utilizada para remover a cápsula da lente durante a cirurgia da catarata. (GARCIA, 2003).

4. SOLUÇÃO ANALÍTICA PARA DEFORMAÇÃO DE MEMBRANAS ESFÉRICAS PREENCHIDAS COM LÍQUIDO EM SEU INTERIOR COM PRESERVAÇÃO DE VOLUME

4.1. Introdução

Na simulação da deformação de membranas em realidade virtual se faz necessário que os tecidos moles, ao serem manipulados, deformem e reajam às forças neles aplicadas. Para que a deformação da superfície em ambiente 3D ocorra em tempo real, faz-se necessário que novos métodos sejam desenvolvidos e implementados para se obter velocidade e realismo físico.

Neste capítulo apresentamos uma nova abordagem para a simulação de objetos deformáveis em tempo real. Calculamos as configurações de equilíbrio de uma membrana 2D elástica tocada por uma prova circular na direção perpendicular à superfície. Objetos 3D também são tratados através da distorção da membrana deformável na forma de órgãos.

A idéia principal de nosso modelo deformável é obter equações analíticas fisicamente motivadas para a simulação de deformações plausíveis. A versatilidade da abordagem em termos de deformação em tempo real, a eficiência em termos de memória e complexidade computacional e a estabilidade incondicional da simulação tornam a abordagem particularmente interessante para jogos e sistemas de simulação cirúrgica.

Ao se tratar de simulação do comportamento mecânico de biomembranas encontramos um grande desafio quando utilizamos os métodos mais tradicionais usados em engenharia para cálculo de deformação de superfície. Podemos citar, por exemplo, método de elementos finitos, onde não se consegue atingir frequência de processamento aceitável para se trabalhar com interface gráfica e dispositivos de retorno de força (COSTA, 2013, GB. OLIVEIRA FILHO. et. al., 2014).

Nesse trabalho escolhemos uma solução analítica com o objetivo de se alcançar uma solução rápida, ou seja, em tempo real. Nessa solução não é necessária efetuar inversões de grandes matrizes, frequentemente utilizadas em outros métodos de deformação de engenharia. O modelo matemático considera a deformação analítica de membranas para a superfície e impõe a condição de conservação de volume para o olho tridimensional.

4.2. Equação de Equilíbrio para Membranas

As soluções para a teoria de pequenas deflexões em membranas são bem conhecidas, mas são resumidas aqui por conveniência. (Adaptado de R. P. FEYNMAN, R. LEIGHTON e M. SANDS, The Feynman Lectures on Physics, Vol. 2, página 12-5, Addison-Wesley, Reading, MA, 1963).

Partiremos do exemplo de uma membrana de material homogêneo e elástico esticada sobre uma armação horizontal, como a pele que recobre um tambor. Suponha agora que a membrana é empurrada para baixo por uma prova com a força sobre a superfície da membrana, a fim de se obter uma deformação. Devido à aplicação da força ocorre a deflexão da membrana. (COSTA, 2013, FEYNMAN, 2008).

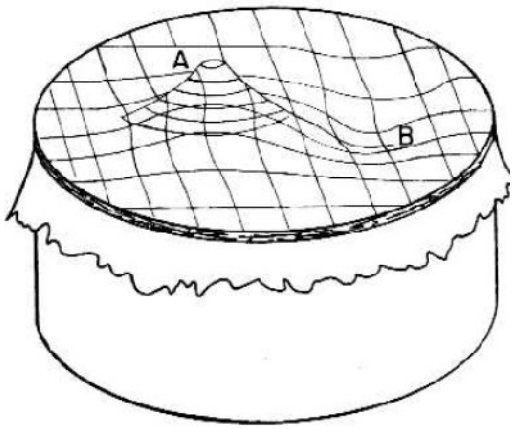


Figura 14: Membrana fina de borracha sobre uma armação cilíndrica (como em um tambor) – a membrana é empurrada para cima no ponto A e para baixo no ponto B. (Adaptado de FEYNMAN, 2008).

A superfície mostrada (Figura 14) apresenta borda fixa e circular. Uma força ΔF no sentido perpendicular ao plano é aplicada por uma prova circular de área transversal ΔA (Figura 15). Portanto, a membrana sofre uma pressão P, dada pela equação $P = \frac{\Delta F}{\Delta A}$. Antes da deformação um ponto da membrana pode ser localizado pelas coordenadas x_0, y_0, z_0. Após a deformação esse ponto se move para a posição x, y e z. A deformação da membrana para um ponto da superfície é aqui definida por u, como $u = z - z_0$.

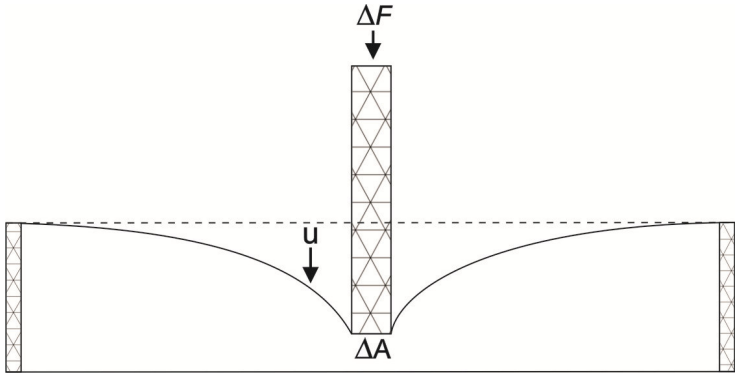
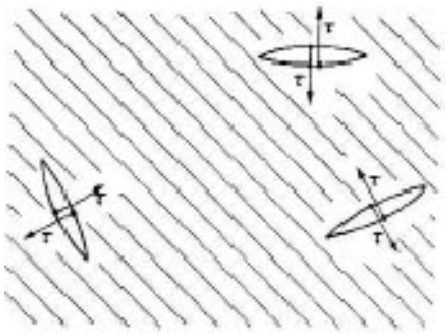


Figura 15: Seção reta de uma membrana esticada quando empurrada para baixo por uma prova circular. (Adaptado de FEYNMAN, 2008).

Há forças na membrana que recobre o tambor, pois ela está esticada. Definimos a magnitude da tensão superficial τ como a força por unidade de comprimento ao longo da membrana. Essa tensão superficial é necessária para manter juntos os dois lados da membrana esticada. Se realizássemos um corte em qualquer ponto, os dois lados da membrana se afastariam, sendo puxados em sentidos opostos (Figura 16). Seja *x* e *y* a posição da membrana e *u* o deslocamento vertical da membrana de sua posição relaxada (não empurrada).

Figura 16: Tensão superficial ao longo de uma folha de borracha, esticada, que é a força por unidade de comprimento através de uma linha. (Adaptado de FEYNMAN, 2008)



Considere um pequeno pedaço da superfície de comprimento *Δx* e largura *Δy*, (Figura 17). Haverá forças na peça devido à tensão superficial τ ao longo de cada ponto (FEYNMAN, 2008).

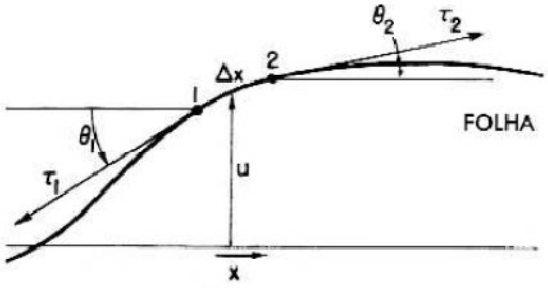


Figura 17: Corte lateral de seção transversal defletida. (Adaptado de FEYNMAN, 2008)

Partiremos de dois pontos 1 e 2 a uma distância Δx um do outro, que são a componente vertical da força resultante para cima. Traçamos então um vetor tangente em cada ponto, ambos de sentidos opostos, teremos um ângulo θ para cada um dos pontos, com ângulos θ_1 e θ_2. Em cada tangente temos a indicação de uma tensão superficial, τ_1 e τ_2 (responsável por manter as duas extremidades unidas ao se efetuar um corte em qualquer região da membrana), (Figura 17).

A força que atua ao longo da extremidade 1 da figura será $\tau_1 \Delta y$, direcionada tangencialmente à superfície, isto é, em um ângulo θ_1 com a horizontal. O mesmo ocorrerá na extremidade 2 da figura a força será $\tau_2 \Delta y$ com um ângulo θ_2 (outras forças que atuam sobre a membrana aqui não são consideradas).

Em pequenas deformações da membrana (pequenas curvaturas), podemos substituir o $sen\theta$ por $tan\theta$[6], garantindo uma boa aproximação que por sua vez pode ser escrito como $\partial u/\partial x$, devido à relação trigonométrica da equação (4.1), limitando assim nossas considerações.

$$sen\theta = tan\theta = \frac{\partial u}{\partial x} \quad (4.1)$$

Logo, a variação de força aplicada em um ponto da membrana será ao longo da extremidade Δy nos dois pontos dada pela relação

$$\Delta F = \tau_2 . (\Delta y) . sen\theta_2 - \tau_1 . (\Delta y) . sen\theta_1 \quad (4.2).$$

Substituindo na equação (4.1) e considerando a tensão superficial constante temos que a força é

$$\Delta F = \tau \left[\left(\frac{\partial u}{\partial x}\Big|_{x=x_2} \right) - \left(\frac{\partial u}{\partial x}\Big|_{x=x_1} \right) \right] \Delta y \quad (4.3),$$

e lembrando a definição de derivada,

[6]Para uma boa aproximação usamos a conversão de graus em radianos, visto que, $\pi/180° = 0{,}017453$ em radianos = 1°, portanto tomando como exemplo um ângulo $\theta = 10°$ possui tangente 0,176326 em radianos e que o mesmo ângulo θ em radianos é igual a 0,174532, o que nos dá uma precisão de duas casas após a vírgula. Considerando a solução apenas para pequenas curvaturas ou deformações em uma membrana (COSTA 2013, FEYNMAN 2008).

$$\frac{\partial f(x)}{\partial x} \equiv \lim_{x_2 \to x_1} \frac{f(x_2) - f(x_1)}{x_2 - x_1} = \lim_{\Delta x \to 0} \frac{f(x_2) - f(x_1)}{\Delta x} \quad (4.4).$$

A quantidade entre colchetes na equação (4.3) pode ser escrita no limite em que $\Delta x \rightarrow 0$ como

$$\left[\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)\right] \Delta x \ (4.5);$$

logo, reescrevemos a equação (4.3)

$$\Delta F = \tau \left[\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)\right] \Delta x \Delta y \ (4.6).$$

Assim, teremos uma equação análoga para *y* devido à contribuição das duas direções nas extremidades, portanto,

$$\Delta F = \tau \left[\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right)\right] \Delta x \Delta y \ (4.7).$$

Para uma pressão constante por unidade de área $\Delta x \Delta y$, considerando a membrana em equilíbrio, a partir da pressão $P = \Delta F / \Delta x \Delta y$ a deformação *(u)* obedece à equação (4.8). A força deve ser balanceada pela força interna da membrana, ou seja, contrária a força externa aplicada. Logo, o fator negativo na equação (4.8) deve-se a uma força de mesmo módulo exercida pela membrana por unidade de área contrária a força imposta pela prova (Terceira Lei de Newton), assim $P = -\Delta F / \Delta x \Delta y$, onde, P representa uma carga externa sendo exercida sobre a membrana por unidade de área.

$$\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right] = -\frac{\mathrm{P}}{\tau} \ (4.8),$$

se a tensão for considerada constante em toda a membrana.

Essa equação pode ser escrita como

$$\nabla^2 u = -\frac{P}{\tau} \ (4.9).$$

Onde:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \nabla^2 u \text{ (4.10).}$$

Finalmente chegamos então à chamada equação de Poisson (4.9), que mede a deformação transversal proporcional a um esforço aplicado e com uma tensão superficial τ considerada constante em toda a membrana.

4.3. Formulação de uma Solução Analítica para Equação de Poisson

Com o objetivo de obter uma solução analítica para a Equação de Poisson (4.9), se faz necessário escolher uma com alto grau de simetria (geometria circular). Essa escolha não imporá grandes restrições para deformar um grande objeto (muito maior do que a área deformada), visto que, as deformações são desprezíveis longe da posição tocada. Assim, na extremidade da região simétrica escolhida, a solução poderá ser conectada a outra superfície, onde a deformação poderá ser considerada nula.

Para uma membrana esticada como na Figura 15 (membrana pressionada em um único ponto) e considerando que conhecemos os limites de borda podemos utilizar uma solução analítica. Se estes limites não fossem conhecidos teríamos somente uma solução numérica para a equação (4.9).

4.4. Solução da Equação de Poisson para Simetria Radial

Esse caso trata de membrana circular de raio *r*, tocada em seu centro por uma prova também circular com raio *a*.

Quando a geometria é circular e a deformação se dá no centro dessa membrana não teremos dependência no ângulo θ, portanto $u(x, y) = u(r)$.

Estando a membrana em equilíbrio (o caso estático) e para pequenas distorções da membrana, temos a equação diferencial que relaciona o deslocamento *u* à tensão da membrana dada pela bem conhecida equação 2D de Poisson (Equação 4.9).

Onde τ é uma tensão superficial constante ao longo da folha, e na simetria circular sem dependência no ângulo θ o termo em $\nabla^2 u$ é:

$$\nabla^2 u = \frac{\partial^2 u}{\partial r^2} \quad (4.11)$$

De forma a facilitar a solução da equação fazemos a seguinte substituição, $\vec{\nabla}\mathrm{u} = \vec{f}$, onde $\vec{\nabla} u = \frac{\partial u}{\partial r}\hat{r}$ assim podemos dizer que,

$$du = f.dr \quad (4.12)$$

com *f* e *u* na direção radial e

$$\vec{\nabla}.\vec{f} = -\frac{P}{\tau} \quad (4.13).$$

Como não existe uma força infinita atuando sobre a membrana, temos apenas uma força distribuída por uma área, podemos então substituir em $P = -F/A$ e na direção radial, reescrevemos a equação (4.13),

$$\vec{\nabla}.\vec{f} = \frac{F}{A\tau} \quad (4.14).$$

Com esta definição, multiplicamos e dividimos o lado esquerdo dessa equação pela espessura da membrana L e a equação (4.14) toma a seguinte forma:

$$\vec{\nabla}.\vec{f} = \frac{FL}{A\tau L} \quad (4.15),$$

teremos em *AL* o volume *V* interno à membrana

$$\vec{\nabla}.\vec{f} = \frac{\mathrm{FL}}{\tau \mathrm{V}} \quad (4.16).$$

Passando V multiplicado para o lado direito da equação

$$\overrightarrow{\nabla}.\vec{f}V = \frac{\text{F.L}}{\tau} \text{ (4.17).}$$

Assim, podemos dizer que obtivemos todo o volume *V* da membrana. Com isso, vemos que o volume *V* é o somatório de vários volumes menores, onde essa soma é a própria integral de volume e, podemos substituir a variável *V* por *dV* e integrar

$$\int \overrightarrow{\nabla}.\vec{f}dV \ = \frac{\text{F.L}}{\tau} \text{ (4.18).}$$

Para solucionar o problema, iremos aplicar o Teorema de Gauss presente no livro Cálculo 2.

$$\int \overrightarrow{\nabla}.\vec{f}dV \ = \int \vec{f} \ . \text{d}\vec{\text{A}} \text{ (4.19).}$$

Substituindo equação (4.18) a equação (4.19)

$$\int \overrightarrow{f}.d\vec{A} = \frac{\text{F.L}}{\tau} \text{ (4.20)}$$

os vetores $\vec{f}$ e $d\vec{A}$ são paralelos e $dA = 2\pi rdL$

$$\int f \ 2\pi r dL \ = \frac{\text{F.L}}{\tau} \text{ (4.21)}$$

Para uma espessura independente de *r*, podemos escrever:

$$2\pi f \ = \frac{1}{r}\frac{\text{F}}{\tau} \text{ (4.22),}$$

multiplicando os dois lados da equação por *dr* e integrando

$$\int 2\pi f dr = \int \frac{1}{r}\frac{F}{\tau} dr \text{ (4.23),}$$

usando a equação (4.12) podemos escrever

$$2\pi u = \frac{F}{\tau}\int \frac{dr}{r} \quad (4.24).$$

O termo em F foi considerado constante.

Por solução logarítmica a partir dos livros de Cálculo 1 temos que $\int \frac{dr}{r}$ é o mesmo que *log (r) + C,* onde *C* é uma constante, assim,

$$u(r) = \frac{F}{2\pi\tau}\ln r + C \quad (4.25),$$

A constante *C* é definida pela condição de borda que é dada pela posição u_0 da aresta da prova que empurra para baixo a membrana. Estas terão dependência de uma força constante aplicada em toda a membrana e aumentando o valor em *r* obteremos uma forma da membrana em logaritmo.

Consideramos uma prova com forma de extremidade que obedece a uma das seguintes características:

- com uma pequena área de extremidade (aproximadamente pontual) ou;
- não muito rígida para que sua borda pode se encaixar na forma de membrana circular ou;
- com uma forma circular;

Portanto, para o caso de espaço livre, o efeito da prova pode ser considerado como uma deformação circular de raio *a* em uma posição constante u_0, ou seja,

$$u(a) = u_o = \frac{F}{2\pi\tau} ln\left(\frac{1}{a}\right) + C \ (4.26).$$

Esta equação nos dá a constante *C* em podemos reescrever em *u(r)* como:

$$u(r) = \frac{F}{2\pi\tau} ln\left(\frac{a}{r}\right) + u_o \ (4.27).$$

Assim a partir da equação (4.27), onde dada uma posição u_0 podemos descobrir as posições *u(r).*

A equação (4.27) é válida para $r \geq a$, isto é, para $u \leq u_0$.

Para $r \leq a$ a superfície do fundo da prova está em contato com a membrana e sua posição *u(r)* é dada pela forma da extremidade da prova. Em nossas simulações esta forma foi escolhida para ser plana $u = u_0 = \text{constante}$ para $r \leq a$. Note que u_0 não é uma constante no tempo porque a altura da prova varia.

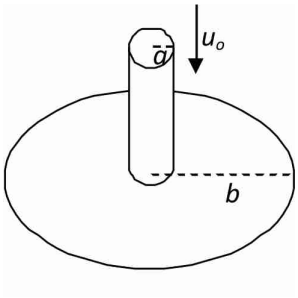


Figura 18: Membrana circular de raio b, tocada por uma prova de raio a.

Se impusermos que $u = 0$ em um círculo de raio b, ou $u(b) = 0$ (Figura 18) então, podemos calcular a força F na prova devido à membrana deformada como

$$F = -2\pi\tau u_0 / \ln\frac{a}{b} \quad (4.28).$$

Devido à lei de ação-reação, uma força de mesmo tamanho e direção oposta deve ser aplicada pela interface háptica.

Usando a equação (4.28) na equação (4.27) podemos desenhar $u\ (r,\ \theta)$. Para um ângulo constante θ fizemos um gráfico *2D* de *u* como uma função de *r* (Figura 19).

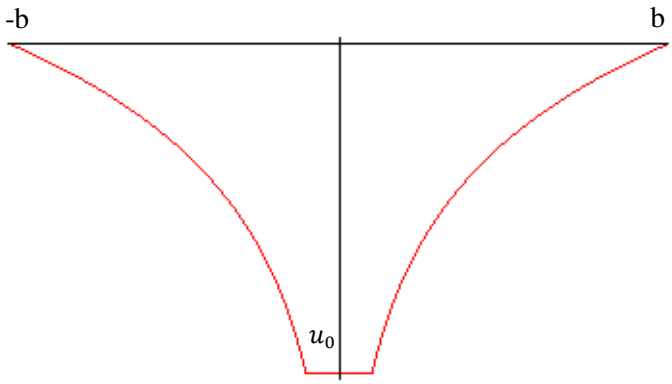


Figura 19: u como uma função de r para um valor constante de θ.

4.5. Conservação de Volume para um Objeto 3D

4.5.1. Objetos Repletos de Fluido - distorcendo uma membrana 2D em um objeto 3D

Podemos impor a conservação do volume para um objeto 3D formado por uma membrana deformável cheia de fluido. Exemplos deste tipo de objetos são vesícula biliar (MOUTSOPOULOS, 1997; WEBSTER, 2003), bexiga urinária (BOUBAKER, 2009; CHI, 2006), ducto biliar (BASDOGAN, 2001), estômago (KUHNAPFEL, 2000), intestino (PHAN-THIEN, 1989), olho (WEBSTER, 2004), mama (AZAR, 2002) sistema vascular (LUBOZ, 2009; ALDERLIESTEN, 2007), etc.

Tomamos como exemplo um objeto 3D, formado por uma superfície elástica, semelhante a balões de festa repletos de fluido. Para pequenas deformações a superfície do objeto pode ser considerada plana, próximo a região da prova, quando submetida a uma força *F*.

Para um objeto cheio com líquido, a conservação do volume pode ser imposta se o fluido é considerado incompressível. Esta é geralmente uma boa aproximação para muitos tecidos biológicos. Em seguida, o volume que é removido em uma parte do objeto, devido à ação da prova, deve ser adicionado igualmente às outras partes do objeto. Assim, a soma dos volumes removidos e adicionados deve ser igual a zero.

Nós podemos impor a conservação do volume para um objeto 3D, formado por uma membrana deformável cheia de líquido. Uma vez encontrada a equação (4.27), que pode ser aplicada a uma membrana circular com o centro de prova, também circular, na origem e se deforma quando tocamos em um ponto da membrana, podemos expandir essa solução para outros objetos.

Cobrimos então parte desse objeto com uma membrana circular de raio b e centro na posição da prova, nos possibilitando o cálculo da deformação a partir da equação (4.25).

$$u(r) = \frac{F}{2\pi\tau}\ln\left(\frac{1}{r}\right) + C$$

Existe a continuidade da superfície, ou seja, a superfície é curva, contínua e sem quebra. Assim na junção na borda da membrana circular $r = b$ com a superfície do objeto não deformada devemos ter a posição e a primeira derivada nesta posição igual para ambas as superfícies. Assim, quando u é contínuo $r = b$, ou seja, $u(b)^+ = u(b)^-$

$$\left.\frac{du}{dr}\right|_{r=b} = 0 \quad (4.29)$$

na borda circular de raio b. Essa solução onde não existe descontinuidade da membrana é denominada condição de contorno homogênea de Neumann.

Essa equação (4.29) será satisfeitas se houver continuidade entre a superfície do objeto e da membrana.

Para nosso problema inicial $\nabla^2 u_H = -\frac{P}{\tau}$, sempre a equação diferencial, admite duas soluções: a solução particular (u_P) e a solução homogênea (u_H).

A equação (4.27) já nos dá a solução particular, onde podemos substituir $u(r)$ por u_P,

$$u_P = \frac{F}{2\pi\tau} \ln\left(\frac{a}{r}\right) + u_0 \quad (4.30).$$

Precisamos satisfazer a solução homogênea para função u_H, isto é, $\nabla^2 u_H = 0$, quando temos condições de contorno.

Logo a equação (4.30) pode ser adicionada à solução homogênea de forma a encontrarmos uma nova solução geral u, que continuará sendo solução, pois a solução homogênea da equação soma apenas um termo nulo. Mas a vantagem é que podemos usar essa solução homogênea para que as condições de contorno sejam obedecidas.

Assim,

$$u = u_P + u_H.$$

Podemos substituir a solução geral na equação (4.29) de forma a definir a condição de continuidade em b.

$$\left.\frac{du_H}{dr}\right|_{r=b} = -\left.\frac{du_P}{dr}\right|_{r=b} \quad (4.31).$$

Substituindo a equação (4.30) em (4.31) e resolvendo, temos que:

$$\frac{du_H}{dr} = \frac{F}{2\pi\tau}\frac{1}{\mathrm{b}} \text{ (4.32).}$$

Integrando e resolvendo a equação (4.32) teremos que

$$u_H = \frac{F}{2\pi\tau}\frac{r}{\mathrm{b}} \text{ (4.33).}$$

Assim, a solução geral,

$$u = u_P + u_H$$

é deste modo igual a:

$$u(r) = \frac{F}{2\pi\tau}\ln\left(\frac{a}{r}\right) + C + \frac{F}{2\pi\tau}\frac{r}{\mathrm{b}} \text{ (4.34)}$$

Como a deformação pode ser considerada constante no ponto onde a prova é aplicada, podemos considerar uma condição física de forma que equação (4.34) seja satisfeita para o valor da constante.

Essa constante pode ser escolhida de forma que sejam mantidas as condições de continuidade da superfície $u(a) = u_0$ na posição da borda da prova circular de raio $r=a$.

$$C = u_o + \frac{F}{2\pi\tau}\left(-\frac{a}{b}\right) \text{ (4.35)}$$

Substituindo 4.35 em 4.34 obtemos

$$u(r) = \frac{F}{2\pi\tau}\ln\left(\frac{a}{r}\right) + u_o - \frac{F}{2\pi\tau}\frac{a}{b} + \frac{F}{2\pi\tau}\frac{r}{\mathrm{b}} \text{ (4.36)}$$

4.5.2. O Objeto Inteiro se Deforma – Conservação de Volume.

Para um objeto fechado e preenchido com líquido, o fluido deve ser considerado incompreensível, assim, a conservação de volume pode ser imposta.

Além disso, quando o volume é retirado de certa região, ou seja, quando uma parte da membrana é empurrada para baixo por uma prova, esse volume da região empurrada deve se deslocar para a superfície como um todo. Todo o volume colocado em uma região deve ser retirado de outras regiões e dividido por igual em toda a membrana. Logo, a soma dos volumes retirados e colocados na membrana tem que ser igual zero, assim, $V = 0$, definindo a conservação do volume.

Bexigas urinárias, entre outros, podem ser consideradas um exemplo para essa situação.

Figura 20: Bexiga Artificial (Fonte: https://artificialorgans.files.wordpress.com/2012/03/bladder.jpeg)

Entre a origem $r = 0$ e o raio da haste cilindrica $r = a$ volume removido é igual ao de um cilindro de raio a. Do raio da haste para $r = b$, o volume removido é igual ao de um sólido de revolução com contorno u (STEWART, James - Cálculo, volume I, 7ª edição, 2013). Estes volumes removidos devem ser compensados pelo volume adicionado pela inflação da superfície do objeto deformável fora do círculo de raio b.

$$V = \pi u_o a^2 + \int_a^b 2\pi u(r) r dr + Au(b) - \pi u(b) b^2 = 0 \text{ (4.37).}$$

Os volumes devem ser considerados desde o cilindro de prova que pressiona a membrana, de raio a. Pressionando a membrana de raio b, onde o primeiro volume retirado é $\pi u_0 a^2$, onde u_0 é a distância da prova com relação ao plano antes da aplicação de uma pressão, isto é, quanto a prova penetrou na membrana (Figura 21 – a).

Consideramos que o volume retirado até a borda b, deve ser adicionado em todo o sólido (diminuindo a posição da prova, pois esta é fixa). Observamos ainda a condição de continuidade da membrana, para que não haja deformação fora do local de aplicação da

prova. O volume retirado pela prova deve ser somado à deformação e depois diminuído da membrana deformada, para que ao final da nossa solução tenhamos $V = 0$.

Substituindo a equação 4.34 e 4.35 na equação 4.37 temos o valor da força F que resulta em:

$$F = -12\,\frac{A\pi u_o \tau b}{(\pi b^3 - 3\pi a^2 b + 2\pi a^3 + 6Ab\,(\log a - \log b) - 6Aa + 6Ab} \quad (4.38).$$

Assim utilizando as equações (4.34), (4.35), e (4.38), podemos obter a partir de um deslocamento u_0, imposto por uma prova de raio circular sobre um objeto 3D a deformação dessa superfície com conservação de volume (Figura 21 – b). Essas equações serão usadas no simulador com o intuito de deformar a membrana e obter sensações táteis no dispositivo háptico, garantindo assim o realismo físico. Definimos um valor para área de todo o objeto como sendo $A = 2\pi b^2$, uma vez que temos uma superfície esférica.

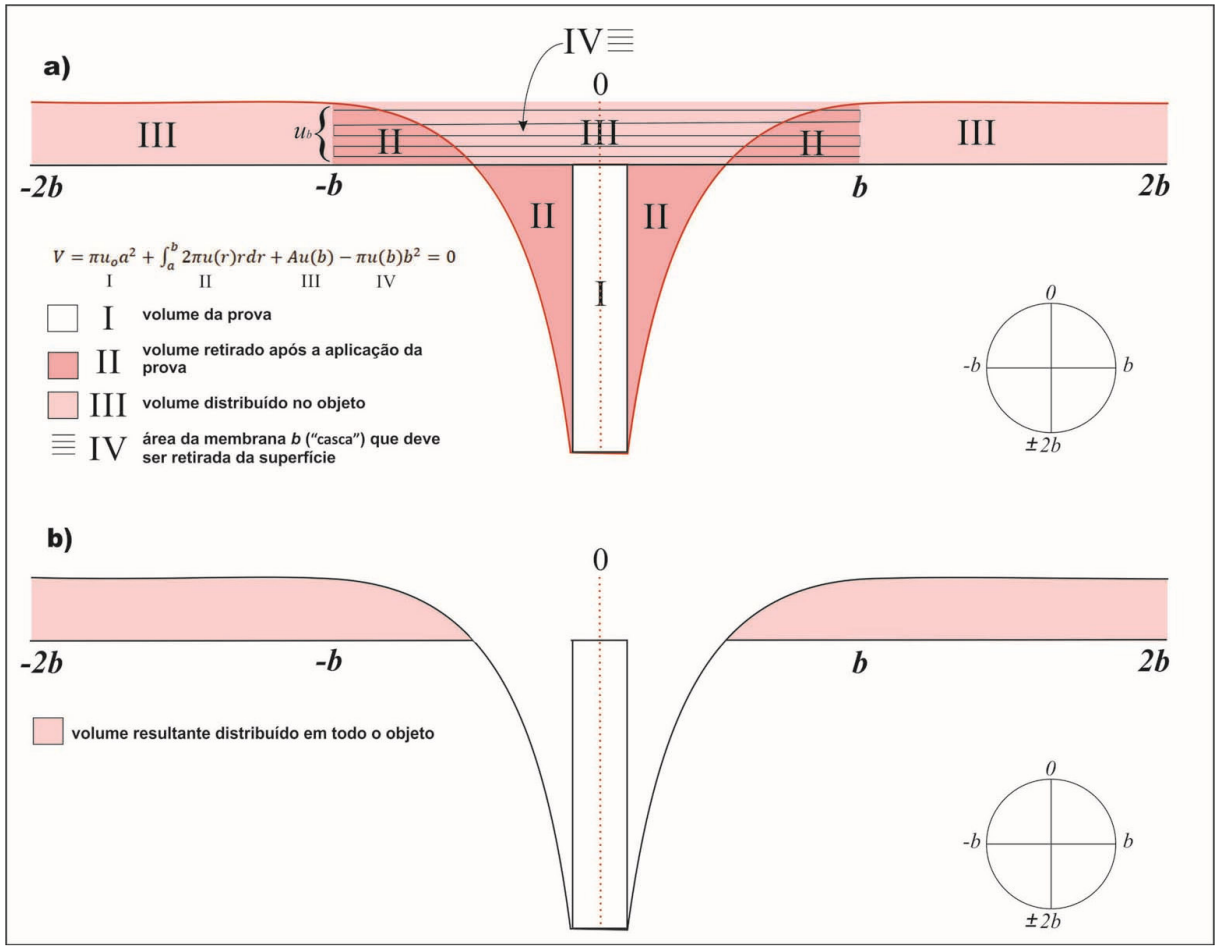


Figura 21: Aplicação da equação de conservação de volume – seção u como função de r para A =2π b²: a) explicação de termos da equação 4.37 e volumes destacados; b) resultado final após solução da equação 3.37.

5. MATERIAIS, MÉTODOS E RESULTADOS

5.1. Introdução

Neste capítulo detalhamos os materiais utilizados, métodos e os resultados alcançados durante a execução do trabalho de pesquisa.

O olho humano foi modelado em ambiente virtual e quando deformado temos de realismo físico em tempo real, com sensações visuais e táteis. As deformações são efetuadas com retorno de força. Utilizamos a plataforma gráfica de código aberto CHAI3D. A edição do código e simulação foi feita através do ambiente de desenvolvimento *Microsoft Visual Studio – MSVC 2008,* que é executado apenas através do sistema operacional *Windows®.* Podemos manipular uma interface háptica (dispositivo de retorno de força) que nos dá a sensação de tocar um objeto que observamos através de um monitor de computador.

5.2. Materiais, Métodos e Resultados

Utilizamos duas etapas para atingir os objetivos do trabalho. Através da codificação e programação em linguagem C++ obtivemos um modelo 3D do olho virtual deformável, onde implementamos a solução analítica, a fim de obter o retorno de força com sensação visual e tátil. A solução analítica considera objetos preenchidos com líquido com conservação de volume (Seção 4.5.2 do Capítulo 4). Por fim de forma a aproximar as características visuais do modelo próximas a de um olho real, efetuamos a inserção da forma e da textura do olho 3D. A superfície obtida com características que buscam simular o olho humano que pode ser deformada quando uma pressão é exercida em qualquer ponto da membrana.

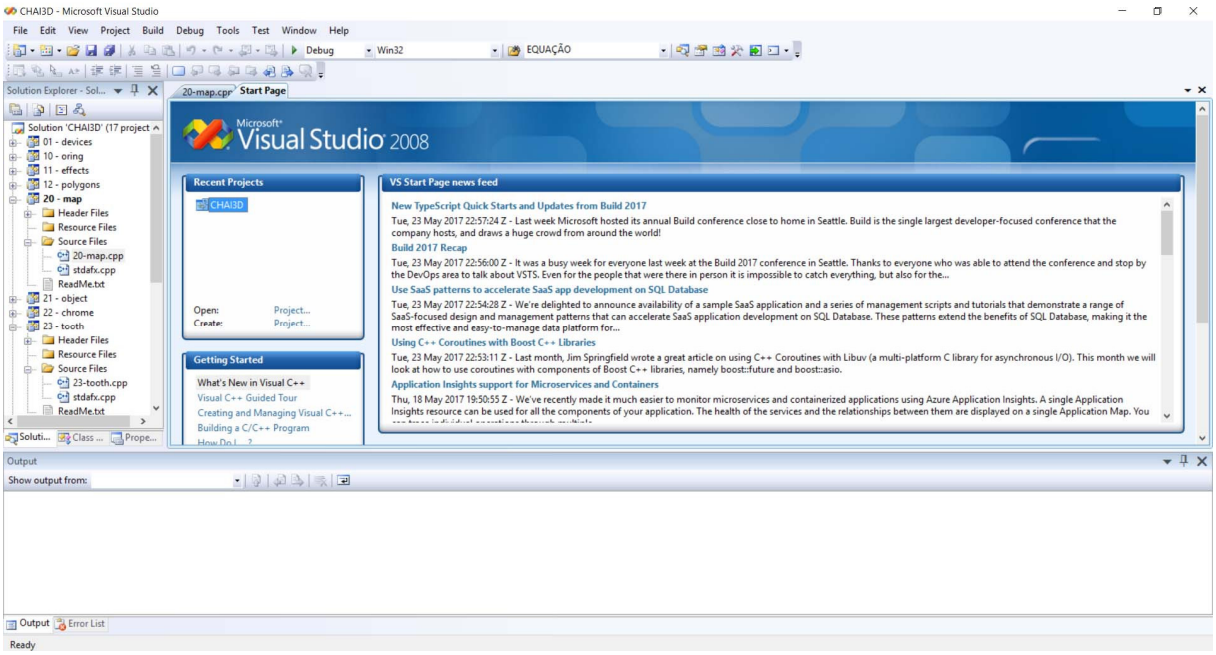


Figura 22: Janela do Microsoft Visual Studio 2008- seleção no painel "Solution Explorer" do arquivo "20_map. cpp".

5.2.1. Interface Háptica

No desenvolvimento deste trabalho utilizou-se a interface háptica *PHANTOM Omni® da SensAble®* (Figura 23). Este dispositivo de retorno de força pode medir com precisão a posição espacial 3D, esquerda e direita, para cima e para baixo, para trás e para frente (ao longo dos eixos x, y e z) e a orientação (avanço e recuo, translação e rotação) da ponta do dispositivo (semelhante a uma caneta). O dispositivo háptico possui 6 graus de liberdade (*Six Degrees of Freedom – 6DOF*) e utiliza motores para criar as forças que fazem recuar a mão do utilizador para simular o toque e a interação com objetos virtuais.

Figura 23: Phantom com 6 graus de liberdade: movimentos possíveis de serem realizados com o dispositivo háptico.

O dispositivo háptico *PHANTOM Omni*® é conectado ao computador por meio de uma entrada *FireWire*®[7] (Figura 24), sendo esta a única forma de comunicação deste dispositivo para comunicações em tempo real e em alta velocidade.

Figura 24: Dispositivos para conexão FireWire – da esquerda para direita: cabo FireWire, placa de conexão FireWire PCI. (Fonte: http://lista.mercadolivre.com.br/firewire).

5.2.2. Ambiente Virtual

Para utilização do CHAI3D é necessário efetuar o download do aplicativo no site do fabricante e descompactar os arquivos de acordo com a versão escolhida, de forma que esta seja compatível com o sistema operacional presente no computador do usuário.

Para manipular e simular a aplicação de força no ambiente virtual o CHAI3D dispõe de uma interface háptica virtual (*Virtual Haptic Device*) que pode ser encontrada na pasta descompactada *(\chai3d-2.0.0\bin)* com o nome *"VirtualDevice.exe"*. Através da interface háptica virtual, podemos verificar a posição da prova nos três eixos (x, y, z) do ambiente virtual e a força exercida também nos três eixos (x, y, z). Quando um projeto está em execução, temos a abertura de uma janela, que apresenta informações sobre o modelo escolhido e os comandos de mouse e teclado que podem ser utilizados durante a simulação (Figura 25).

[7] O *FireWire*® desenvolvido pela *Apple Computer* é uma tecnologia de entrada/saída de dados em alta velocidade para conexão de dispositivos digitais até computadores portáteis e *desktops*. Oferece comunicações de alta velocidade e serviços de dados em tempo real (Disponível em: https://www.tecmundo.com.br/usb/1968-o-que-e-firewire-.htm. Acesso em 25/05/2017).

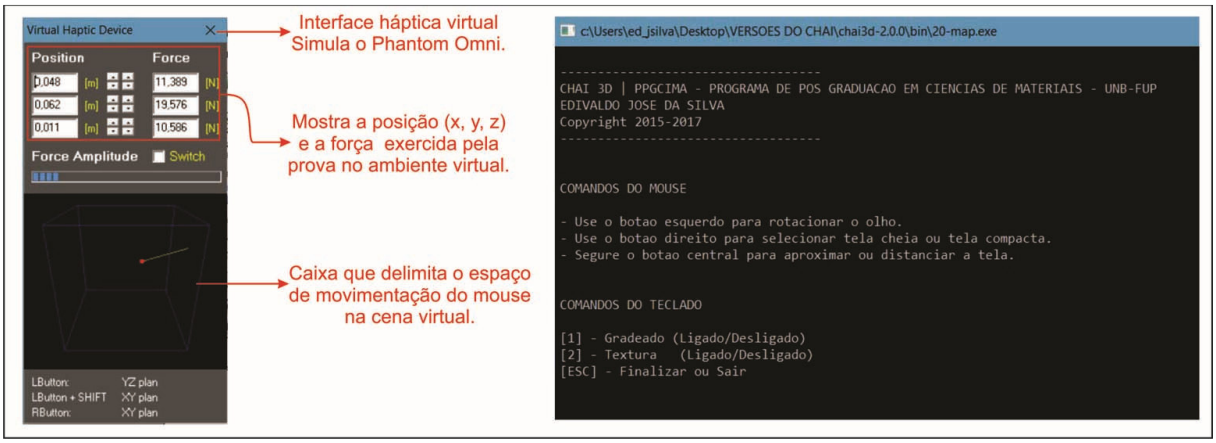


Figura 25: Dispositivo háptico virtual, que mede o deslocamento e a força exercida. Janela do projeto em execução com comandos do mouse e de opções de teclado.

5.2.3. Modelagem da Membrana Plana

O olho humano modelado em ambiente virtual foi desenvolvido a partir da membrana plana deformável com realismo físico e retorno de força real, COSTA, (2013). A modificação do código partiu do projeto exemplo do Chai3D denominado *"20_map"*, onde este apresentava características que possibilitavam a deformação (Figura 26).

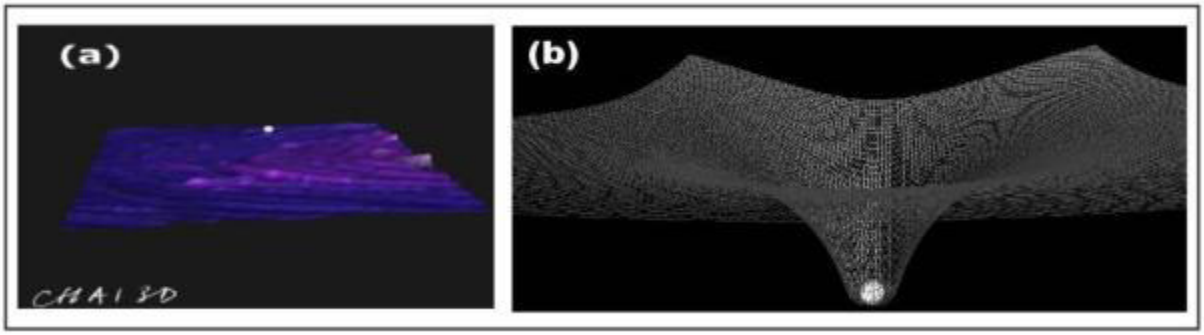


Figura 26: Simulação do projeto "20 – map": a) projeto executado a partir do arquivo original do CHAI3D; b) membrana plana deformável de COSTA, 2013).

5.2.4. Modelagem da Superfície Semi-esférica

Modelamos o olho humano com base no plano deformável apresentado por COSTA, (2013), uma vez que o código já se encontrava traduzido e comentado. A partir do plano inicial, o primeiro passo foi transformar a membrana em uma superfície semi-esférica. A Figura 27 mostra essa transformação passo a passo.

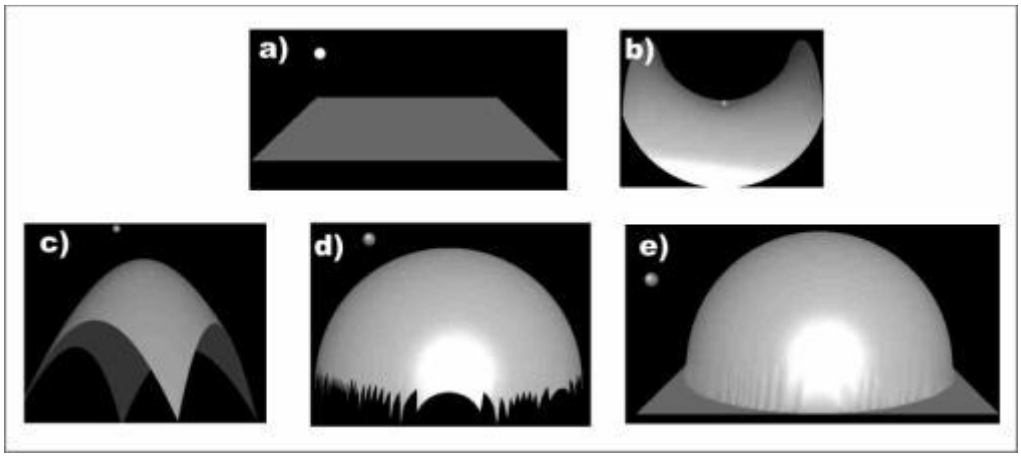


Figura 27: Etapas de transformação do plano em uma superfície semi-esférica: a) plano inicial (COSTA, 2013); b) modificação no código na tentativa de converter o plano a uma semi-esfera com a formação de uma "sela"; c) plano curvo aproximando de uma semi-esfera; d) semi-esfera sem correção de contorno; e) semi-esfera deformável aproximando-se da superfície do olho.

A inserção da equação da esfera possibilita a construção da superfície como vemos na figura 27, que a partir da geometria analítica, com centro (*0, 0, 0*) e raio r é $x^2 + y^2 + z^2 = r^2$. O valor para o raio da membrana foi definido como sendo 0.95, para que a superfície semi-esférica seja totalmente fechada. Quando este valor está muito próximo de 1.0 (limite de borda da superfície), gera-se um buraco nas extremidades da superfície (Figura 28).

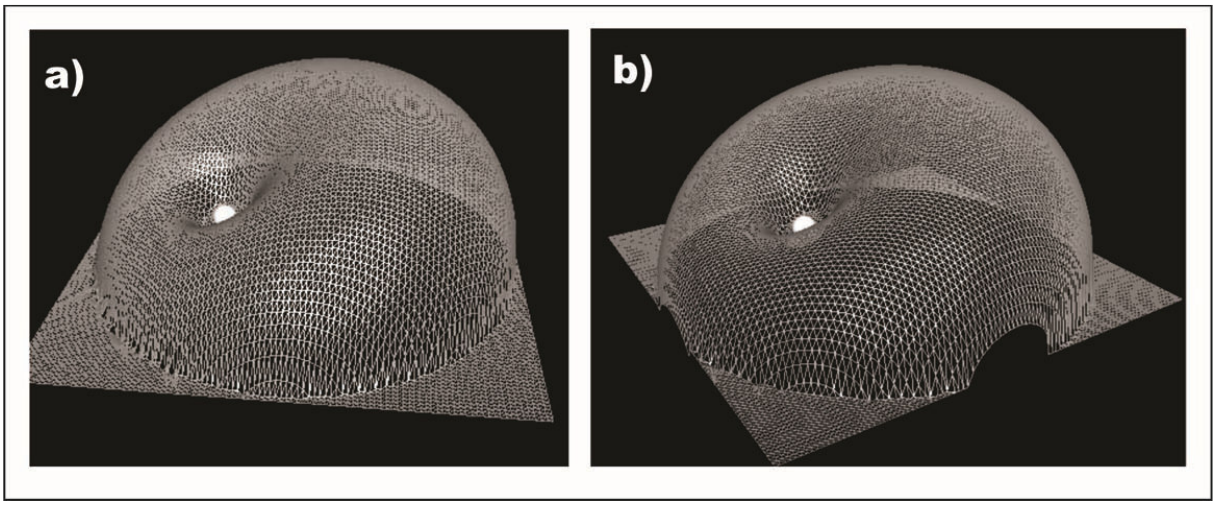


Figura 28: Prova aplicada sobre a superfície da semi-esfera com malha triangular: a) membrana com raio r=0.95; a) membrana com raio r=1.0.

Na codificação desse trabalho, utilizamos o valor padrão de 0.05 como sendo o raio da prova (esfera branca que toca o plano), que de acordo com os objetivos que se deseja alcançar pode ser alterado facilmente. Ainda, o coeficiente de tensão superficial τ foi definido

como sendo 1.0 e também pode ser alterado de acordo com a tensão superficial para o objeto real.

5.2.5. Desenhando e Aplicando Deslocamento da Superfície

Para gerar a (Figura 28) foi necessário modificar o código, uma vez que tínhamos inicialmente um plano e este deveria ser transformado em uma semi-esfera. Fizemos a inserção do trecho abaixo de forma a aplicar o deslocamento à membrana, considerando a equação da esfera.

```
      // APLICAR DESLOCAMENTO DO OLHO:
if ((cSqr(posVertex.x) + cSqr(posVertex.y))<0.95)
 posVertex.z = u+ sqrt(0.95 - (cSqr(posVertex.y)) -(cSqr(posVertex.x)
      // LEITURA DA POSIÇÃO DE U.
else posVertex.z=0.0;
```

Este comando faz o teste de comparação se $x^2 + y^2 < 0.95$ (se as posições dos vértices de *x y* somados são menores que o diâmetro da esfera). Caso o teste seja verdadeiro, a linha seguinte é executada, ou seja, é construído o desenho da superfície semi-esférica. Somamos ainda ao desenho uma deformação *u* imposta pela aplicação da prova sobre a semi-esfera na direção do vértice no eixo *z*.

Assim, a variável *posVertex* trata da posição dos vértices que formam a malha que forma a membrana, esses comandos fizeram modificação no desenho do plano transformando em uma superfície esférica.

Nesse ponto, é inserido o valor para o módulo de *r,* que nos dá uma posição *(x, y)* no plano.

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

5.2.6. A Detecção de Colisão

No intuito de criar um modelo que fosse o mais próximo possível do olho, trabalhamos de forma a codificar a detecção de colisão[8].

[8] Segundo DOS SANTOS MACHADO et. al. (2006), a detecção de colisão consiste no processamento em tempo-real da intensidade do retorno de força quando do toque do dispositivo háptico com os objetos virtuais.

Como se trata de um modelo com superfície radial, antes de inserir diretamente as equações no modelo virtual do olho humano fez-se necessário encontrar a posição de quanto a prova entra no objeto quando tocamos a área deformável, isto é, qual a posição da esfera branca (prova) que toca a membrana na cena virtual. Para isso utilizamos as seguintes linhas de código:

```
double modPosicao = sqrt(cSqr(posicao.x)+cSqr(posicao.y)+cSqr(posicao.z));
// MÓDULO DA POSIÇÃO DOS EIXOS X, Y E Z.
```

Assim quando a prova se aproximar dessa região calculada, o detector de colisão reconhece que a prova está em uma área deformável e podemos iniciar a manipulação do objeto, bem como a imagem do olho sendo deformado.

```
posVertex.z= u+(sqrt(cSqr(0.62)-cSqr(posVertex.x)-cSqr(posVertex.y))+0.45);
```

Assim nesse trecho do código é calculado o módulo da posição da prova na direção da superfície, ou seja, o quanto a prova penetra no objeto. Nosso modelo tem diâmetro igual a 0.95, precisamos calcular qual é a distância do centro do olho até o limite onde a prova penetrou. A (Figura 29) mostra a solução do problema para esta situação e em seguida apresentamos quais foram os cálculos efetuados para chegarmos a esta conclusão.

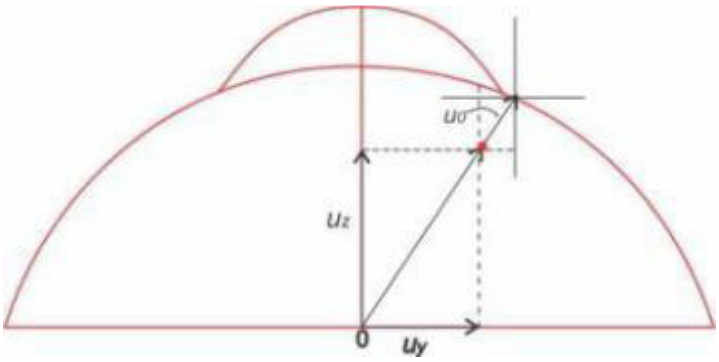


Figura 29: Cálculo do módulo de posição – representa a solução para encontrar a posição da prova quando esta entra na superfície da membrana, o ponto vermelho é a localização da prova, ou seja, a posição u.

Era necessário que localizássemos a posição de u_0 nas três direções (u_0x, u_0y, u_0z), ou seja, qual a posição da prova dentro da superfície (profundidade). Calculamos as componentes dos vetores, o que nos possibilitou encontrar o módulo das posições. Para cálculo das forças (*Fx, Fy, Fz*) fizemos o mesmo procedimento, uma vez que já havíamos obtido a direção dos vetores, o que nos resultou nas seguintes equações:

a) para cálculo em módulo das posições (simbolizado pela letra grega α, no código apresentamos a variável como *modPosicao*), o termo em $u_z{}^2$, temos dois

diferentes valores, um para protuberância do olho e outro para o restante da superfície, como mostrado no trecho que aborda a detecção de colisão.

b) $\alpha = \sqrt{{u_x}^2 + {u_y}^2 + {u_z}^2}$; para cálculo de $u_{0,}$ que também leva em consideração a protuberância do olho no teste de detecção de colisão,

$$u_0 = 0.95 - \alpha.$$

No trecho abaixo é testada a detecção de colisão que faz o a leitura da posição de onde estamos tocando no olho. A partir do teste da posição de u_{0z}, é possível encontrar em que ponto está localizado a prova com relação à superfície da membrana. Cria ainda, uma condição para que a deformação não ocorra caso o raio da membrana seja menor que o raio da prova, movendo assim toda a superfície.

```
if ((cSqr(posicao.x) + cSqr(posicao.y))<0.22)// TESTE DA DETECÇÃO DE
COLISÃO PARA PROTUBERÂNCIA (ELEVAÇÃO DA SUPERFICIE DO OLHO).
u0 = (modPosicao)-sqrt(cSqr(posicao.x) + cSqr(posicao.y)
+cSqr(sqrt(cSqr(0.62)-cSqr(posicao.x)-cSqr(posicao.y))+0.45)); // CALCULO
EM MÓDULO DA POSIÇÃO DA PROVA (U0) A PARTIR DA PROTUBERÂNCIA (ELEVAÇÃO DA
SUPERFICIE DO OLHO).
else
u0 = (modPosicao)-(0.95);// CASO A ESFERA BRANCA ESTEJA FORA DA
PROTUBÊNCIA, MAS NO OLHO, O MÓDULO DA POSIÇÃO PROVA É DIMINUIDO DE
APROXIMADAMENTE 1.
if (u0 <= 0.0){   // CONDIÇÃO PARA DEFORMAÇÃO DO OLHO - CASO ESTEJA ABAIXO
DE 0.0 OU SEJA TOCANDO O OLHO
if ((cSqr(posicao.x) + cSqr(posicao.y))<0.22)// LEITURA DA POSIÇÃO DA PROVA
(U0) EM Z A PARTIR DA PROTUBERÂNCIA (ELEVAÇÃO DA SUPERFICIE DO OLHO).
u0z = posicao.z-(sqrt(cSqr(0.62)-cSqr(posicao.x)-cSqr(posicao.y))+0.45);//
LEITURA DA POSIÇÃO DE U SOMANDO A DEFORMAÇÃO E DESENHANDO O OLHO.
else u0z = posicao.z - sqrt(0.95 - cSqr(posicao.x)-cSqr(posicao.y));
if (r <= a )
u = u0z; // FAZ A ESFERA ACOMPANHAR A PROVA CONDIÇÃO PARA QUE NÃO OCORRA A
DEFORMAÇÃO DO OLHO E LIMITA QUE DEFORMAÇÃO TENDA AO INFINITO.
```

5.2.7. Modelando o Olho a partir da Superfície Semi-esférica

No trecho destacado abaixo, inicialmente fizemos o teste de onde se encontrava posicionada a prova no eixo *z*, ou seja, se $x^2 + y^2$ é menor que a área do plano. Essas inserções no código é o que nos possibilita efetuar a aplicação da força no eixo *z*, visto que se trata de um modelo desenvolvido para uma membrana 2D, aplicada em um objeto 3D (Capítulo 4, Seção 4.5.1.).

A partir deste ponto criamos a segunda superfície sobre a superfície semi-esférica assemelhando se ao olho virtual. Porém, neste passo, ainda não havíamos feito a aplicação da

solução analítica para conservação de volume, bem como inserção de textura para recobrir a superfície criada.

```
if ((cSqr(posVertex.x)+ cSqr(posVertex.y))<0.95){// TESTE DA POSIÇÃO DOS
VERTICES DA MALHA QUE FORMA O OLHO
if ((cSqr(posVertex.x) + cSqr(posVertex.y))<0.22)// CRIA PROTUBERÂNCIA
(ELEVAÇÃO DA SUPERFICIE DO OLHO).
posVertex.z= u+(sqrt(cSqr(0.62)-cSqr(posVertex.x)-
cSqr(posVertex.y))+0.45);// LEITURA DA POSIÇÃO DE U SOMANDO A DEFORMAÇÃO E
DESENHANDO O OLHO.
else posVertex.z = u+ sqrt(0.95 - cSqr(posVertex.x)-cSqr(posVertex.y));
// CASO O TESTE NÃO SEJA VALIDO CRIA A SUPERFICIE SEM PROTUBERÂNCIA.
```

O resultado da implementação dos trechos de código apresentados a partir da idéia do plano inicial é mostrado (Figura 30).

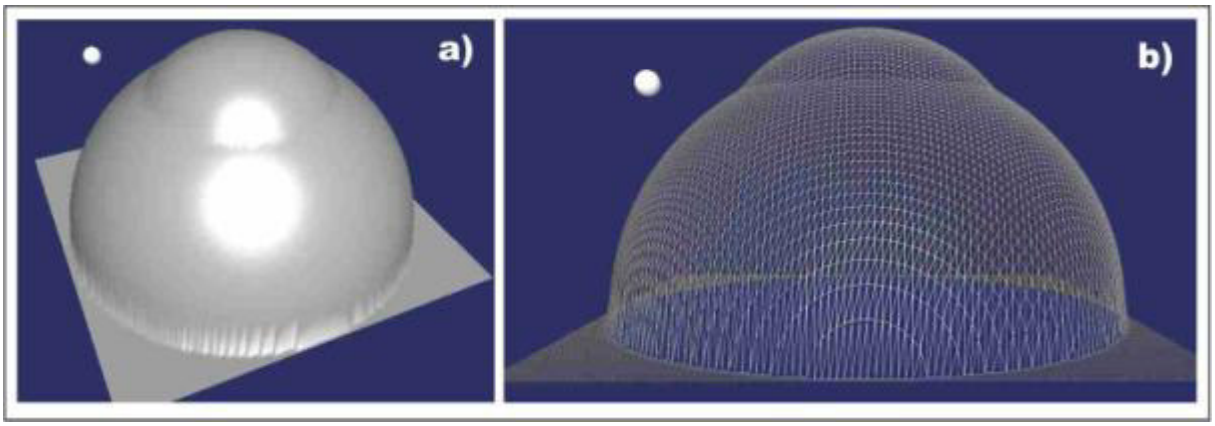
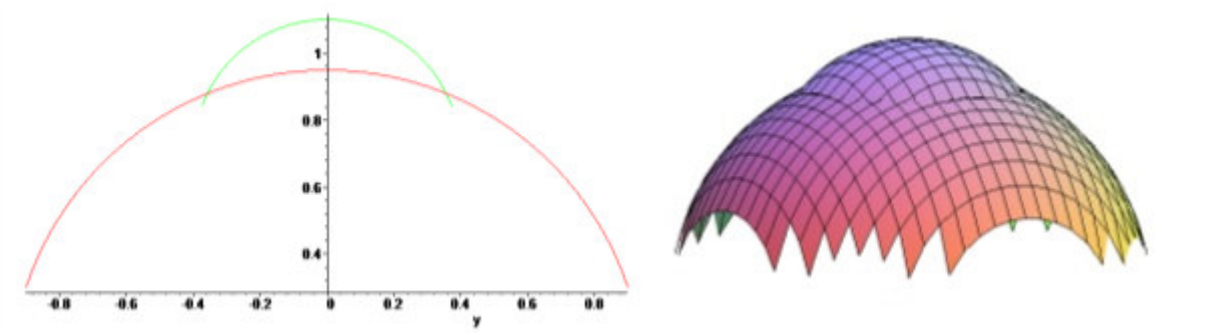


Figura 30: Resultado da implementação de linhas de código – superfície do olho: a) superfície sem recurso de textura e sem gradeado (malha triangular); b) superfície com recurso gradeado ativado.

Posteriormente este trecho foi melhorado de forma a trazer uma proximidade do desenho do olho humano. Na (Figura 31) podemos observar a combinação das equações que foram utilizadas para criar a superfície do olho virtual. Temos a equação da esfera ($\sqrt{0.95 - x^2 - y^2}$) e uma variação da mesma equação ($\sqrt{(0.62)^2 - y^2} + 0.45$), esta última calcula a área da superfície maior do olho e acrescenta o deslocamento da para cima (protuberância da córnea).

No simulador colocamos ainda um teste de forma a verificar qual o tamanho da protuberância do olho.

```
if ((cSqr(posVertex.x) + cSqr(posVertex.y))<0.22)// CRIA PROTUBERÂNCIA
(ELEVAÇÃO DA SUPERFICIE DO OLHO).
```




```
z3:=sqrt(0.95-x^2-y^2);
z4:=sqrt(0.62^2-x^2-y^2)+0.45;
plot3d({z3,z4},x=-0.9..0.9,y=-0.9..0.9,scaling=constrained);
```

$$z3 := \sqrt{.95 - x^2 - y^2}$$

$$z4 := \sqrt{.3844 - x^2 - y^2} + .45$$

Figura 31: Imagens e código gerados no Maple para semi-esfera com protuberância.

5.3. Inserção da Textura no Modelo do Olho Virtual

Nesta seção apresentamos as linhas de comando utilizadas para inserção de textura no modelo do olho virtual. A textura vem como forma de propiciar a sensação de estar visualizando o olho em 3D. A imagem da textura foi criada com o auxílio de algumas ferramentas para edição de imagens, considerando que a imagem deve estar no formato "*bmp*" [9]. A (Figura 32) apresenta a textura inserida depois de editada.

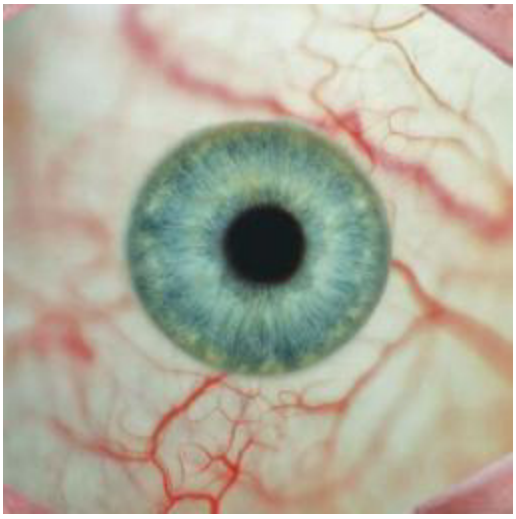


Figura 32: Textura utilizada para recobrir o modelo do olho virtual, nomeada como olho5.1.bmp.

[9] É um formato de imagem gráfica composto por pixels (menor ponto para formar uma imagem, medido em *dpi* ou *ppi* – pontos por polegada ou pixel por polegada.), tem como sigla: *Device Independent Bitmap (DIB)* ou *Windows Bitmap (BMP)*. (MIOT; PAIXÃO; PASCHOAL, 2006, p. 174-75, apud COSTA (2013, p. 64).

No corpo do código inserimos alguns trechos para que a textura fosse incorporada ao modelo. Através da análise de diversos exemplos de modelos disponíveis na biblioteca do CHAI3D, possibilitou sua adaptação ao que nos era necessário. Isso resultou nas seguintes linhas de código:

```
int plano()
{
cTexture2D* newTexture = new cTexture2D();
// CRIAR UM ARQUIVO DE TEXTURA
mundo->addTexture(newTexture);
bool fileload = newTexture->loadFromFile("resources/images/olho5.1.bmp");
// LEITURA DE ARQUIVO DE TEXTURA
if (!fileload)
{
#ifdefined(_MSVC)
fileload = newTexture-
>loadFromFile("../../../bin/resources/images/olho5.1.bmp");
// LEITURA DE ARQUIVO DE TEXTURA
#endif
}
if (!fileload)
{
printf("Erro - Textura não foi lida corretamente.\n");
// TESTE SE A LEITURA FOI FEITA CORRETAMENTE
fechar();
return (-1);

        }
```

No ambiente virtual inserimos ainda algumas imagens, tais como logotipo da universidade e logotipo do programa de Pós-Graduação e do CHAI3D, que seguem basicamente o mesmo princípio para inserção de textura, graças às facilidades oferecidas pela plataforma CHAI3D. O resultado da textura e logotipos implementados pode ser visualizado na Figura 33.

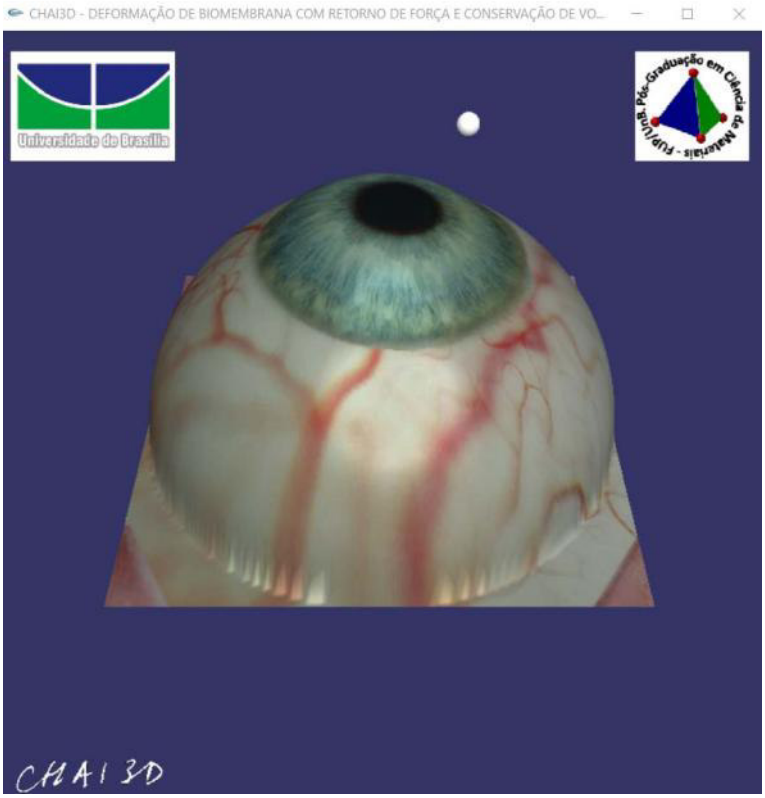


Figura 33: Execução do modelo do olho virtual após inserção de texturas e logotipos.

5.4. Aplicação da Solução Analítica para Conservação de Volume no Modelo Virtual

Nesta seção demonstramos como as soluções analíticas (Capítulo 4) foram inseridas no modelo virtual através do *MSVC 2008*. Utilizamos algumas etapas antes de colocar diretamente a solução das equações no nosso olho virtual. Isso nos proporcionou a compreensão do comportamento das equações quando fossem implementadas na plataforma.

Os trechos abaixo correspondem à transposição das equações (4.34; 4.35 e 4.40) para linguagem C++:

$$u(r) = \frac{F}{2\pi\tau}\ln\left(\frac{a}{r}\right) + C + \frac{F}{2\pi\tau}\frac{r}{\mathrm{b}} \quad (4.34)$$

```
        u  =  Forca/(2*Pi*tau)*log(a/r)+C+r*Forca/(2*tau*Pi*b);  //  EQUAÇÃO
PARA DEFORMAÇÃO DA MEMBRANA COM CONSERVAÇÃO DE VOLUME(EQUAÇÃO 4.34)
```

$$C = u_o + \frac{F}{2\pi\tau}\left(-\frac{a}{b}\right) \quad (4.35)$$

```
        double  C  =  u0  +  Forca/(2*Pi*tau)*(-a/b);// EQUAÇÃO PARA CÁLCULO DA
CONSTANTE, OU SEJA, c (EQUAÇÃO 4.35)
```

$$F = -12\,\frac{A\pi u_o \tau b}{(\pi b^3 - 3\pi a^2 b + 2\pi a^3 + 6Ab\,(\log a - \log b) - 6Aa + 6Ab} \quad (4.40).$$

```
double Forca = -12*A*Pi*u0*tau*b/(Pi*pow(b,3.0)-
3*Pi*cSqr(a)*b+2*Pi*pow(a,3) +6*A*b*log(a/b)-6*A*a+6*A*b);// EQUAÇÃO PARA
CÁLCULO DA FORÇA SOBRE A MEMBRANA PARA CONSERVAÇÃO DE VOLUME (EQUAÇÃO 4.40)
```

Para o olho tridimensional, a deformação que ocorre mediante a aplicação de uma força será apenas na direção do eixo z.

No próximo passo inserimos as equações no modelo do plano deformável, onde a membrana apresenta área infinita para deformação. Com essa aplicação notamos que as equações se comportavam conforme o esperado, e ainda que quando tocado ao centro estava semelhante aos gráficos apresentados nas Figuras 4 e Figura 21.

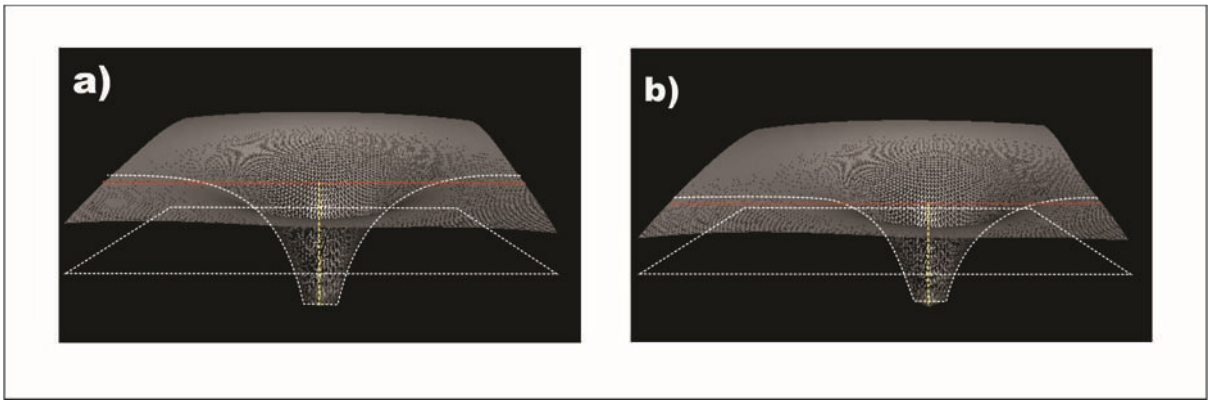


Figura 34: Equações de conservação de volume inseridas no plano deformável com sobreposição de perspectiva do plano e da deformação: a) com deformação ao centro do plano; b) deformação fora do centro do plano.

A aplicação das equações para conservação de volume no plano deformável tornou possível verificar que ao ser exercida uma força perpendicular ao plano (*eixo z*) parte do plano se desloca para cima da posição de equilíbrio. A membrana quando pressionada em um ponto separa uma interface de dois meios, ar do lado externo e água em seu interior. As linhas tracejadas na Figura 34 foram inseridas para ressaltar a deflexão da membrana: a linha vermelha representa a posição deslocada do plano sem deformação; a linha branca apenas contorna a deformação, bem como mostra o sentido do plano sem deslocamento; a linha amarela por sua vez mostra o deslocamento da prova. Assim, conseguimos verificar a validade da conservação do volume nas equações. Em seguida adaptamos essa solução ao modelo do olho virtual.

Para cálculo das forças inserimos o seguinte trecho no código do programa:

```
Fx = Forca*posicao.x/modPosicao;
// FORÇA APLICADA EM UMA REGIÃO DO OLHO NO EIXO X
Fy = Forca*posicao.y/modPosicao;
// FORÇA APLICADA EM UMA REGIÃO DO OLHO NO EIXO Z
Fz = Forca*posicao.z/modPosicao;
// FORÇA APLICADA EM UMA REGIÃO DO OLHO NO EIXO Y
```

Isso nos possibilitou que as forças fossem calculadas de forma independente, isto é, para cada posição que a prova estiver dentro da membrana, o valor da força será apresentado em cada eixo.

Quando a simulação está em execução é mostrado na interface háptica virtual (*Virtual Device*), o quanto foi exercido de força nesses eixos. Isso pode ser notado mais claramente (Figura 35 – olho sem deformação e Figura 36 – deformação no eixo z).

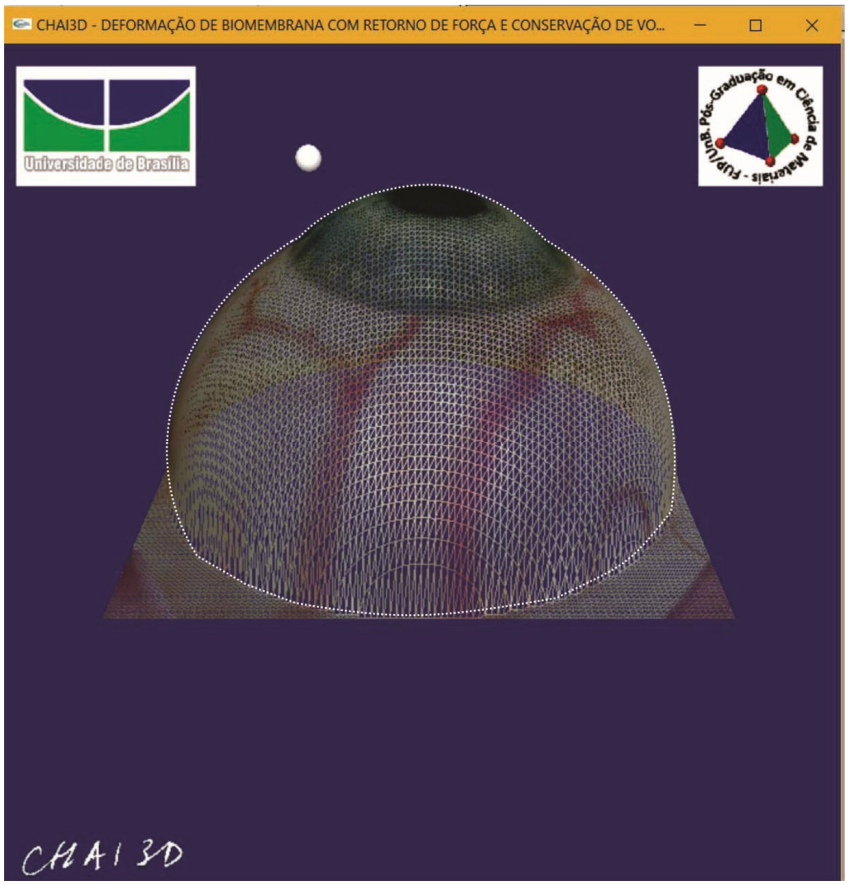


Figura 35: Olho sem Deformação – nenhuma força incide sobre a superfície do olho.

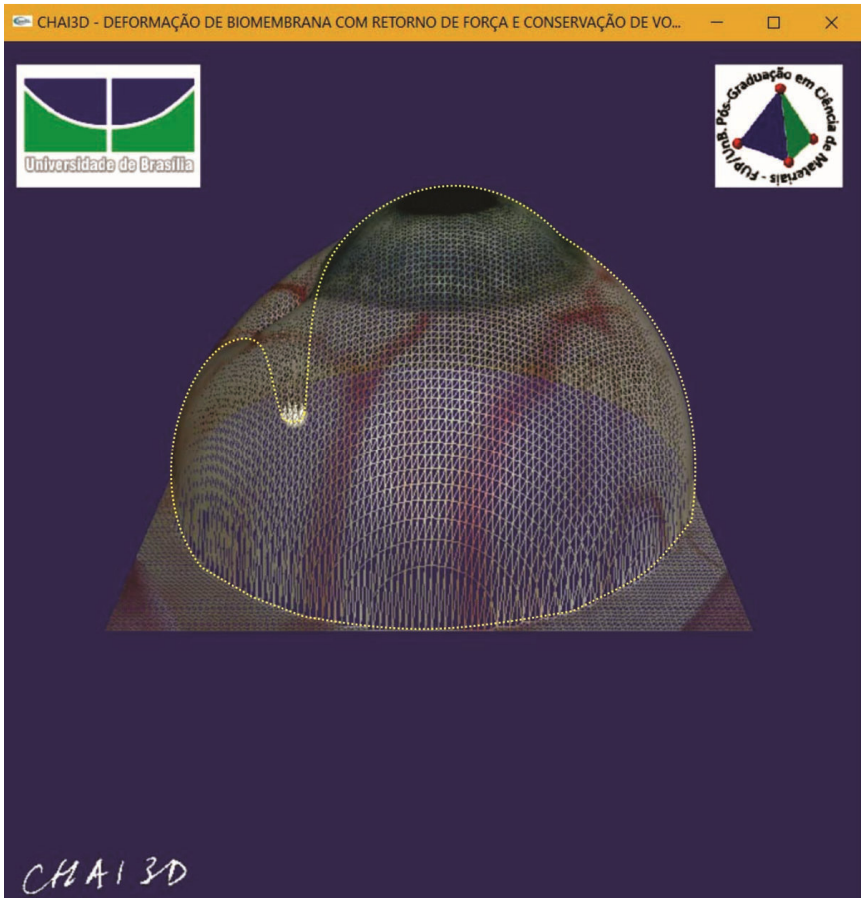


Figura 36: Olho com Deformação no eixo z – Forças aplicadas no eixo z

5.5. Utilização da Solução Analítica para Conservação de Volume no Olho Virtual

A imagem abaixo (Figura 37) demonstra a expansão da superfície do olho. Quando é aplicada a força em um ponto, o volume retirado pela prova é redistribuído em todo o olho. Fizemos aqui um comparativo através de editores que trabalham com a vetorização de imagens, assim, nos possibilitou criar linhas que contornam toda a superfície e podemos comparar o resultado obtido em uma superfície deformada e uma superfície sem deformação. Linhas tracejadas indicam a superfície deformada com aumento de volume e sem deformação.

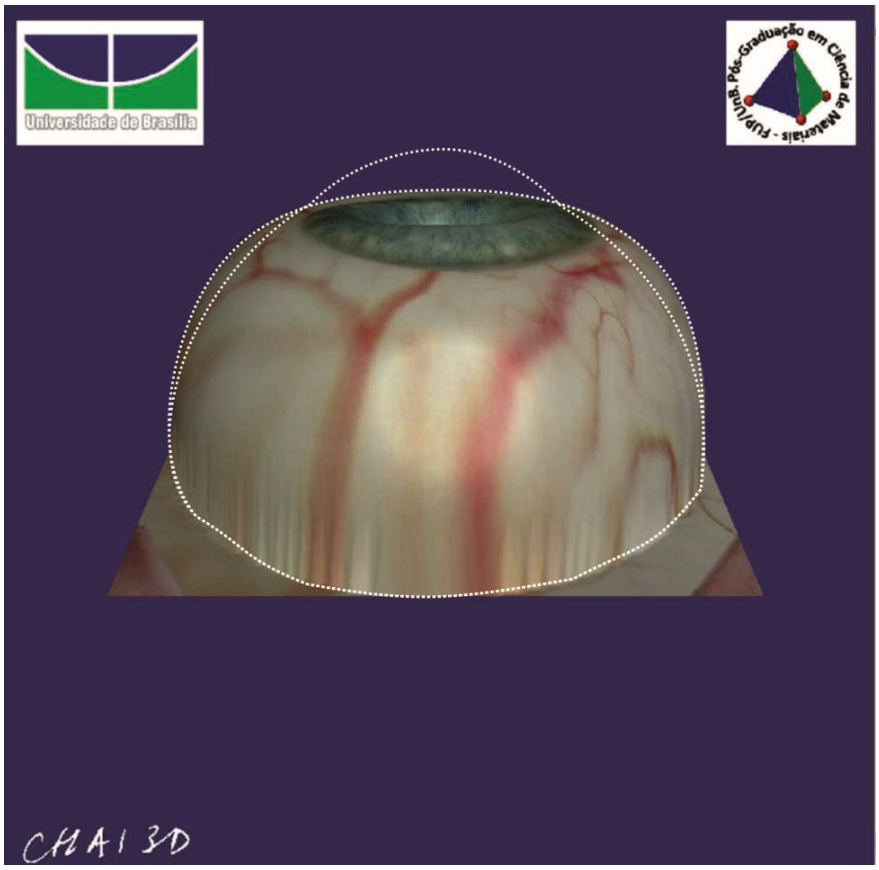


Figura 37: Olho com Deformação no eixo z – Forças aplicadas no eixo z na parte superior do olho e com contorno de forma a demonstrar o aumento de volume a partir da superfície original.

6. DISCUSSÃO E CONSIDERAÇÕES FINAIS

Esta seção apresenta alguns pontos que foram relevantes na construção deste trabalho. Aborda a discussão, as considerações finais e a possibilidade de trabalhos futuros a partir da utilização da solução analítica da deformação de superfícies com realismo físico: estudo de caso do olho humano.

6.1. Utilização de Interfaces Visuais e Táteis

A superfície de um objeto tridimensional pode ser aproximadamente descritas por um conjunto de vértices, pontos no espaço tridimensional.

Após os argumentos aqui apresentados, devemos nos convencer que a utilização de interfaces é de suma importância em aplicações de realidade virtual. Ressaltamos que estes dispositivos possibilitam o usuário ver, mas também sentir o objeto virtual no qual ele está interagindo. A adaptação do modelo deformável em uma situação real, de forma a assemelhar-se a um órgão, tecido ou material biológico tem que estar totalmente transparente para o usuário de forma inseri-lo no mundo virtual através dos seus sentidos do tato e da visão.

6.2. Retorno de Força

Nosso modelo que simula o olho humano com conservação de volume é executado em uma plataforma virtual, onde através de um dispositivo de retorno de força, o usuário interage, movendo uma ferramenta rígida que devolve a força aplicada. A interface háptica que usamos é o modelo *PHANTOM Omni* da *SensAble Technologies PHANTOM*®. Um dispositivo deste tipo requer resposta que obriga o cálculo em uma alta frequência (acima de 400 Hz), caso contrário o efeito ao toque não será natural para o usuário (respostas da interação não correspondem ao real manipulado com atrasos e sobressaltos artificiais). Com as soluções analíticas aqui apresentadas isto não é um problema, pois o nosso método se mostra rápido, pois não utiliza das lentas inversões de grandes matrizes.

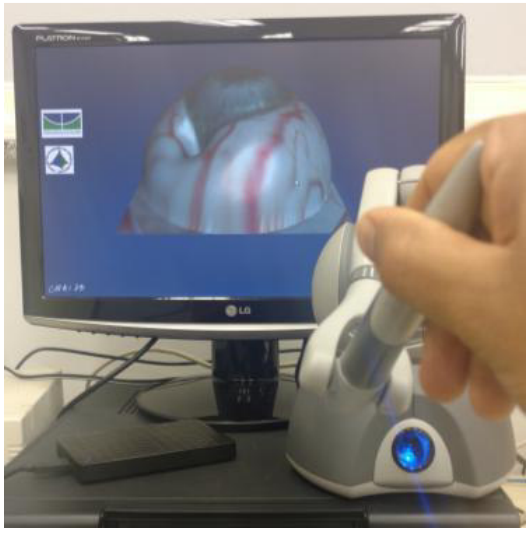


Figura 38: Retorno de força através da interface háptica.

Demonstramos ao decorrer deste trabalho com uma série de cálculos e testes visuais no modelo virtual que podem ser testados em ambiente virtual com manipulação da superfície. A validade do nosso método permite verdadeiras manipulações em tempo real de objetos deformáveis. Nós animamos com retorno de força o olho virtual em tempo real. Uma das vantagens desse modelo que o método não demanda grande capacidade de processamento, portanto não exige a utilização de grandes computadores.

Assim, dependendo da rigidez apresentada pelo objeto temos uma resposta realista para a ação do usuário, com deformações intuitivas quando a ferramenta é movida. Apresentamos abaixo algumas imagens de manipulação da cena virtual com utilização do dispositivo háptico.

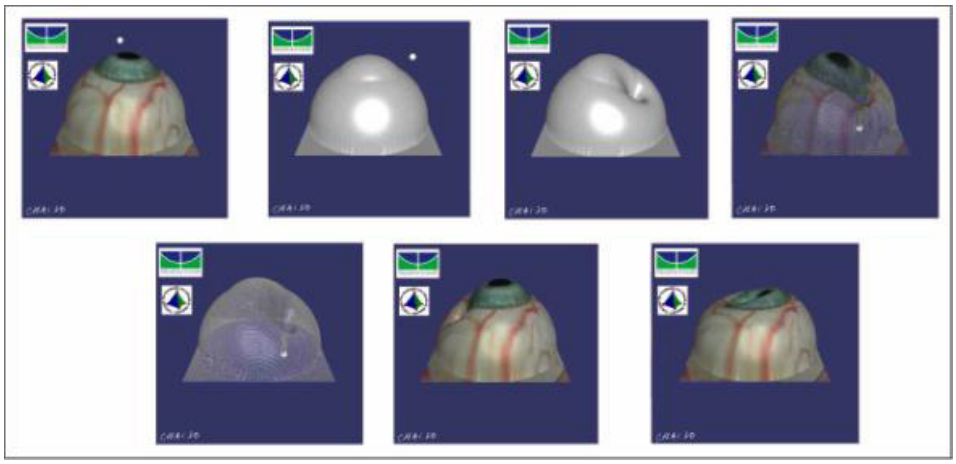


Figura 39: Sequência de imagens com e sem deformação – a textura foi retirada em algumas situações de forma a visualizar a interação da prova (esfera branca) que manipula a superfície da membrana.

Para a demonstração do olho em um ambiente virtual, usamos uma simulação física em tempo real de uma membrana de tensão uniforme. O uso da solução analítica e quase-estática em vez de equações do tipo EDP (Equações Diferenciais Parciais) evita todos os problemas relativos à integração numérica, garantindo estabilidade para a simulação. Não utilizamos inversão de matrizes ou cálculos que demandem grande capacidade de processamento, bem como acompanhamento a estrutura de dados, o que tornaria o modelo lento para utilização em realidade virtual.

Note que, para o estado de equilíbrio, as constantes de amortecimento e atrito são irrelevantes, pois todas as velocidades são zero. Os componentes da abordagem são simples de implementar e eficiente em termos de requisitos de memória e desempenho de tempo de execução.

O fluido que preenche o objeto é considerado incompressível. Isso significa que a conservação de volume deve ser garantida quando há algum contato externo ao objeto. O volume deslocado pelo contato causará o deslocamento de toda a superfície, ou em outras palavras, a variação de volume devido à superfície tocada pelo contato da prova com a superfície tem que ser igual à soma do volume criado pelo deslocamento de todos os elementos não tocados, para assegurar a conservação do volume.

As deformações globais são fisicamente plausíveis e fenômenos importantes, tais como o movimento de todas as partes do sólido devido à preservação do volume é produzido automaticamente. O método neste trabalho artigo pode ser parte de um simulador de realidade virtual mais geral, de forma a tornar mais acessível a manipulação de objetos deformáveis, podendo se adaptar a diversos órgãos do corpo humano. O método resolve o problema de deformação de forma elegante e simples, independentemente de outros aspectos de simulação como movimentos (tradução, rotação, integração dinâmica, etc.), alterações de topologia e manipulação de colisão. Para obter um simulador genérico, esse método deve ser integrado aos métodos adequados para lidar com estes e outros aspectos.

6.3. Considerações Finais

Com este trabalho, fomos capazes de modelar fisicamente o olho humano. Permitimos sua manipulação em ambiente virtual, atribuindo deformação elástica da

superfície, e preservando o volume, com as alterações de topologia em tempo real e sendo rapidamente computáveis.

Apresentamos a solução analítica e robusta de forma a trazer o conjunto de uma abordagem interessante para explorar o verdadeiro mundo das deformações tridimensionais da membrana e para ilustrar o potencial de aplicação da nossa abordagem. Visto que quando animamos membranas deformáveis elásticas quase estáticas temos uma taxa de quadros garantida, não sendo lenta a reprodução das imagens e a sensação tátil.

Nós desenvolvemos um método robusto e em tempo real para simulações. Os resultados são apresentados no contexto de um sistema de realidade virtual. O usuário interage em tempo real com o objeto dinâmico por meio do controle de uma ferramenta rígida, ligado a um dispositivo tátil conduzido com forças derivadas do método.

Além disso, mostramos que nosso modelo possibilita a simulação em tempo real, onde não é necessária grande carga computacional. Sendo assim, o método desenvolvido pode ser executado em computadores convencionais, tais como *desktops* e *notebooks*, implicando apenas que estes permitam a instalação de *hardware* e *softwares* específicos para manipulação de imagens e de dispositivos táteis.

7. REFERÊNCIAS BIBLIOGRÁFICAS

AZAR, Fred S.; METAXAS, Dimitris N.; SCHNALL, Mitchell D. Methods for modeling and predicting mechanical deformations of the breast under external perturbations. **Medical Image Analysis**, v. 6, n. 1, p. 1-27, 2002.

ALDERLIESTEN, Tanja; BOSMAN, Peter AN; NIESSEN, Wiro J. Towards a real-time minimally-invasive vascular intervention simulation system. **IEEE Transactions on Medical Imaging**, v. 26, n. 1, p. 128-132, 2007.

BASDOGAN, Cagatay; HO, C.-H.; SRINIVASAN, Mandayam A. Virtual environments for medical training: Graphical and haptic simulation of laparoscopic common bile duct exploration. **IEEE/Asme Transactions On Mechatronics**, v. 6, n. 3, p. 269-285, 2001.

BOUBAKER, Mohamed Bader et al. Finite element simulation of interactions between pelvic organs: Predictive model of the prostate motion in the context of radiotherapy. **Journal of biomechanics**, v. 42, n. 12, p. 1862-1868, 2009.

CHI, Y.; LIANG, J.; YAN, D. A material sensitivity study on the accuracy of deformable organ registration using linear biomechanical models. **Medical physics**, v. 33, n. 2, p. 421-433, 2006.

CORRÊA, Cléber Gimenez; DE LS NUNES, Fátima; BEZERRA, Adriano. Implementação de Interação em Sistemas Virtuais para Simulação de Exames de Biópsia. In: **VIII Workshop de Informática Médica-WIM2008**. 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.

COSTA, R. S. Deformação de biomembranas com retorno de força para treinamento médico em realidade virtual**.** Universidade de Brasília: Dissertação (Mestrado) **- Faculdade UnB Planaltina**, 2013.

COSTA, I. F. Bisturis virtuais. **Revista Ciência Hoje**, 40:79–81, 2007.

DEL PALOMAR, A. Pérez et. al. A finite element model to accurately predict real deformations of the breast. **Medical Engineering & Physics**, v. 30, n. 9, p. 1089-1097, 2008.

DOS SANTOS MACHADO, Liliane; DE MORAES, Ronei. VR-based simulation for the learning of gynaecological examination. **Advances in Artificial Reality and Tele-Existence**, p. 97-104, 2006.

FEYNMAN, Richard Phillips; LEIGHTON, Robert B.; SANDS, Matthew. **Feynman lectures on physics. vol. 1: Mainly mechanics, radiation and heat**. Addison-Wesley, 1963.

G. B. OLIVEIRA FILHO, A. M. MAROJA, R. S. COSTA, I. F. COSTA; Deformação de Membranas Para Uso em Sistemas de Simulação: Uma Validação Experimental de Soluções Analíticas com Realismo Físico, **Faculdade UnB Planaltina**, 2014.

GIBSON, Sarah FF; MIRTICH, Brian. **A survey of deformable modeling in computer graphics**. Technical Report, Mitsubishi Electric Research Laboratories, 1997.

GRIMM, Johannes PW; WAGNER, Clemens; MÄNNER, Reinhard. Interactive real-time simulation of the internal limiting membrane. **Lecture Notes in Computer Science**, v. 3078, p. 153-160, 2004.

HALLIDAY, RESNICK, WALKER. **Fundamentos de Física – volume 3: Eletromagnetismo**, 8ª Edição. (LTC – Rio de Janeiro, RJ, 2009. pg. 113-115).

HELENE, Otaviano; HELENE, André Frazao. Alguns aspectos da óptica do olho humano. **Revista Brasileira de Ensino de Física**, v. 33, n. 3, p. 3312, 2011.

HO, Andrew K. et al. Virtual reality myringotomy simulation with real□time deformation: Development and validity testing. **The Laryngoscope**, v. 122, n. 8, p. 1844-1851, 2012.

KOCH, Rolf M.; GROSS, Markus H.; BOSSHARD, Albert A. Emotion editing using finite elements. In: **Computer Graphics Forum**. Blackwell Publishers Ltd, 1998. p. 295-302.

KÜHNAPFEL, Uwe; CAKMAK, Hüseyin Kemâl; MAAß, Heiko. Endoscopic surgery training using virtual reality and deformable tissue simulation. **Computers & graphics**, v. 24, n. 5, p. 671-682, 2000.

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

LUBOZ, Vincent et al. Real-time guidewire simulation in complex vascular models. **The Visual Computer**, v. 25, n. 9, p. 827-834, 2009.

MORAES, Ronei M. et al. A virtual reality based simulator for gynecologic exam training. In: **Information Technology Based Higher Education and Training, 2006. ITHET'06. 7th International Conference on**. IEEE, 2006. p. 786-791.

MEIER, Ullrich et al. Real-time deformable models for surgery simulation: a survey. **Computer methods and programs in biomedicine**, v. 77, n. 3, p. 183-197, 2005.

MELO, Jairo Simão Santana; BRASIL, Lourdes Mattos; BALANIUK, Remis. Ambiente de Simulação Médica para WEB. In: **IX Symposium on Virtual And Augmented Reality**. 2007.

MOUTSOPOULOS, Konstantinos; GILLIES, Duncan. Deformable models for laparoscopic surgery simulation. **Computer networks and ISDN systems**, v. 29, n. 14, p. 1675-1683, 1997.

MUNEM, M. & FOULIS, D.J. **Cálculo. Vol. 2**. Rio de Janeiro: LTC, 1982. (LTC – Rio de Janeiro, RJ, 1982. pg. 1022-1023).

NEALEN, Andrew et al. Physically based deformable models in computer graphics. In: **Computer graphics forum**. Blackwell Publishing Ltd, 2006. p. 809-836.

DE OLIVEIRA, Ana Cláudia Melo Tiessi Gomes; DOS SANTOS NUNES, Fátima de Lourdes. Building a open source framework for virtual medical training. **Journal of digital imaging**, v. 23, n. 6, p. 706-720, 2010.

PAMPLONA, Djenane C.; VELLOSO, Raquel Q.; RADWANSKI, Henrique N. On skin expansion. **journal of the mechanical behavior of biomedical materials**, v. 29, p. 655-662, 2014.

PHAN-THIEN, N.; LOW, H. T. On the Flow of a Non-Newtonian Liquid Induced by Intestine-Like Contractions. **Journal of biomechanical engineering**, v. 111, n. 1, p. 1-8, 1989.

PAVARINI, L. Estudo e Implementação do Método Massa-Mola para Deformação em Ambientes Virtuais de Treinamento Médico usando a API Java 3D. Marília: Dissertação (Mestrado), 2006.

PICINBONO, Guillaume et al. Improving realism of a surgery simulator: linear anisotropic elasticity, complex interactions and force extrapolation. **Computer Animation and Virtual Worlds**, v. 13, n. 3, p. 147-167, 2002.

ROOSE, Liesbet et al. Validation of different soft tissue simulation methods for breast augmentation. In: **International Congress Series**. Elsevier, 2005. p. 485-490.

SORENSEN, Mads Solvsten; MOSEGAARD, Jesper; TRIER, Peter. The visible ear simulator: a public PC application for GPU-accelerated haptic 3D simulation of ear surgery based on the visible ear data. **Otology & Neurotology**, v. 30, n. 4, p. 484-487, 2009.

DE SOUSA, Ana Maria Almeida; OKADA, Daniel Mochida; SUZUKI, Fabio Akira. O uso de simuladores no aprendizado para cirurgia otológica.

ULLRICH, Sebastian et al. An intersubject variable regional anesthesia simulator with a virtual patient architecture. **International journal of computer assisted radiology and surgery**, v. 4, n. 6, p. 561-570, 2009.

WAGNER, Clemens; SCHILL, Markus A.; MANNER, Reinhard. Collision detection and tissue modeling in a VR-simulator for eye surgery. In: **ACM International Conference Proceeding Series**. 2002. p. 27-36.

WEBSTER, Roger et al. Elastically deformable 3D organs for haptic surgical simulation. **Studies in health technology and informatics**, p. 570-572, 2002.

WEBSTER, Roger et al. A haptic surgical simulator for laparoscopic cholecystectomy using real-time deformable organs. In: **Proceedings of the IASTED international conference on biomedical engineering**. 2003. p. 25-27.

WEBSTER, Roger et al. A haptic surgical simulator for the continuous curvilinear capsulorhexis procedure during cataract surgery. **Studies in health technology and informatics**, p. 404-406, 2004.

WILLIAMS, Celeste et al. Simulation studies for predicting surgical outcomes in breast reconstructive surgery. **Medical Image Computing and Computer-Assisted Intervention-MICCAI 2003**, p. 9-16, 2003.

8. APÊNDICES

Aqui apresentamos o código utilizado para construção do modelo, um projeto do CHAI3D que pode ser executado em um compilador. Utilizamos o *software Microsoft Visual Studio 2008* como ferramenta de compilação e para alteração das linhas do código durante a construção do modelo.

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

/* ================================================================================================ */
/* ======================== DECLARANDO AS CONSTANTES INCIAIS ============================== */
/* ================================================================================================ */

// TAMANHOS INICIAIS(LARGURA/ALTURA) EM PIXELS NA TELA DA JANELA

constint LARGURA_JANELA = 720; // LARGURA DA JANELA.
constint ALTURA_JANELA = 700; // ALTURA DA JANELA.


// MENU DE OPÇÕES DO MOUSE - (CLIQUE COM BOTÃO CENTRAL DO MOUSE NA TELA)

constint TELA_CHEIA= 1; // 1ª OPÇÃO DO BOTÃO CENTRAL DO MOUSE.
constint JANELA_COMPACTA= 2; // 2ª OPÇÃO DO BOTÃO CENTRAL DO MOUSE.
constdouble TAM_ESCALA_MALHA = 2.0;// TAMANHO DO PLANO DE DEFORMAÇÃO

/* ================================================================================================ */
/* ================== DECLARAÇÃO DAS VARIÁVEIS QUE CRIAM A CENA VIRTUAL ================== */
/* ================================================================================================ */

cWorld* mundo;// CLASSE QUE CRIA UMA O MUNDO DA CENA VIRTUAL.

cCamera* camera;// CLASSE QUE CRIA A CÂMERA.

cHapticDeviceInfo info;// INFORMAÇÕES SOBRE O DISPOSITIVO HÁPTICO NA CENA VIRTUAL

cMesh* olho;// CLASSE QUE CRIA O OLHO (OBJETO) ATRAVÉS DE UMA MALHA DE TRIÂNGULOS.

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

cBitmap* logoChai;// CLASSE QUE INSERE UM PEQUENO LOGO "CHAI3D.ORG" NO CANTO INFERIOR DA
JANELA.
cBitmap* logoUnB;// CLASSE QUE INSERE UM PEQUENO LOGO "UNB" NO CANTO SUPERIOR DA JANELA.
cBitmap* logoPPGCIMA;// CLASSE QUE INSERE UM PEQUENO LOGO "PPGCIMA" NO CANTO SUPERIOR DA
JANELA.

int displayW  = 0;// LARGURA DA JANELA.

int displayH  = 0;// ALTURA DA JANELA.

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

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

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.
```

```
string resourceRoot;//DECLARAÇÃO DO RECURSO RAIZ - USADO PARA ACHAR O CAMINHO DA PASTA ONDE
ESTÁ O ARQUIVOS DE TEXTURA E IMAGENS.


/* ================================================================================================ */
/* ================================   DECLARAÇÃO DE MACROS========================== */
/* ================================================================================================ */

#define CAMINHO_RECURSO(p)(char*)((resourceRoot+string(p)).c_str())// CONVERTER PARA CAMINHO
DO RECURSO


/* ================================================================================================ */
/* ================== COORDENADAS ESFÉRICAS PARA POSIÇÃO DA CÂMERA: ====================== */
/* ================================================================================================ */

double cameraAngleH;// ANGULO HORIZONTAL DA CÂMERA

double cameraAngleV;// ANGULO VERTICAL DA CÂMERA

double distanciaCamera;// DISTÂNCIA DA CÂMERA

cVector3d posicaoCamera;// POSIÇÃO DA CÂMERA DO TIPO VETOR 3D

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).

constdouble Pi =3.14159265359;// DEFINIÇÃO DO VALOR DE PI.


/* ================================================================================================ */
/* ========================== DECLARAÇÃO DE FUNCOES E METODOS ============================== */
/*================================================================================================ */

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

void teclado(unsignedchar 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 opcaoMouse(int value);// CHAMA A FUNÇÃO CLIQUE COM OS BOTÕES DIRETO DO MOUSE PARA TELA
CHEIA OU COMPACTA.

void movimentoMouse(int x, int y);// CHAMA A FUNÇÃO QUE CONTROLA O OLHO 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 O OLHO INICIAL DA SIMULAÇÃO PARA SER DEFORMADO.

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

int main(int argc, char* argv[]){
printf ("\n");
printf (" ------------------------------------\n");
printf ("CHAI 3D | PPGCIMA - PROGRAMA DE POS GRADUACAO EM CIENCIAS DE MATERIAIS - UNB-FUP\n");
printf (" EDIVALDO JOSE DA SILVA\n");
printf (" Copyright 2015-2017\n");
printf (" ------------------------------------\n");
printf ("\n\n");
printf (" COMANDOS DO MOUSE\n\n");
printf (" - Use o botao esquerdo para rotacionar o olho. \n");
```

```
printf (" - Use o botao direito para selecionar tela cheia ou tela compacta. \n");
printf (" - Segure o botao central para aproximar ou distanciar a tela. \n");
printf ("\n\n");
printf (" COMANDOS DO TECLADO\n\n");
printf (" [1] - Gradeado (Ligado/Desligado)\n");
printf (" [2] - Textura   (Ligado/Desligado)\n");
printf (" [ESC] - Finalizar ou Sair \n");
printf ("\n\n");


 resourceRoot = string(argv[0]).substr(0,string(argv[0]).find_last_of("/\\")+1);// ANALISAR
PRIMEIRO ARG PARA TENTAR LOCALIZAR OS RECURSOS

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

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

mundo->setBackgroundColor(0.20, 0.2, 0.4);// DEFINE A COR DO FUNDO - PADRÃO RGB (VERMELHO,
VERDE, AZUL)

camera = new cCamera(mundo);// CRIA UMA CÂMERA E INSERE ESSA CÂMERA NO MUNDO PARA QUE ESSE
POSSA SER OBSERVADO.

 mundo->addChild(camera);// ADICIONA UMA CÂMERA AO MUNDO COMO UM DE SEUS FILHOS (ADCIONA A
CÂMERA NA CENA).


// DEFINIR A POSIÇÃO PADRÃO DA CÂMERA EM COORDENADAS ESFÉRICAS:

cameraAngleH = 0;// VERIFICAR A VARIAÇAO DA DETECCAO DE COLISAO COM O ANGULO DA CAMERA
cameraAngleV = 30;// VERIFICAR A VARIAÇAO DA DETECCAO DE COLISAO COM O ANGULO DA CAMERA
distanciaCamera = 2.0 * TAM_ESCALA_MALHA;// CALCULA DISTANCIA DA
CAMERAatualiza_posicao_camera();// ATUALIZA A POSIÇÃO DA 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.

/* ================================================================================================ */
/* ============================ INSERIR LOGOTIPOS 2D NA SIMULAÇÃO ============================ */
/* ================================================================================================ */


// CRIAR UMA IMAGEM BITMAP EM 2D
logoChai = new cBitmap();
logoUnB = new cBitmap();
logoPPGCIMA = new cBitmap();

// ADICIONAR O LOGOTIPO NA CENA DO OLHO
camera->m_front_2Dscene.addChild(logoUnB);
camera->m_front_2Dscene.addChild(logoChai);
camera->m_front_2Dscene.addChild(logoPPGCIMA);


bool fileload;// LEITURA DA IMAGEM BITMAP LOGOTIPOS
fileload = logoUnB->m_image.loadFromFile(CAMINHO_RECURSO("resources/images/unb.bmp"));
fileload = logoChai->m_image.loadFromFile(CAMINHO_RECURSO("resources/images/chai3d.bmp"));
fileload = logoPPGCIMA->m_image.loadFromFile(CAMINHO_RECURSO("resources/images/PPGCIMA.bmp"));

if (!fileload){
#ifdefined(_MSVC)
fileload = logoUnB->m_image.loadFromFile("../../../bin/resources/images/unb.bmp");
fileload = logoChai->m_image.loadFromFile("../../../bin/resources/images/chai3d.bmp");
fileload = logoPPGCIMA->m_image.loadFromFile("../../../bin/resources/images/PPGCIMA.bmp");
#endif
}
```

```
logoUnB->setPos(10, 580, 0);// POSICIONE O LOGOTIPO UNB NA PARTE SUPERIOR ESQUERDA DA TELA
(COORDENADAS DE PIXEL)
logoUnB->setZoomHV(0.5, 0.5);// DIMENSIONE O LOGOTIPO UNB AO LONGO DE SEU EIXO HORIZONTAL E
VERTICAL

logoChai->setPos(10, 10, 0);// POSICIONE O LOGOTIPO CHAI3D NA PARTE INFERIOR ESQUERDA DA TELA
(COORDENADAS DE PIXEL)
logoChai->setZoomHV(0.3, 0.3);// DIMENSIONE O LOGOTIPO CHAI3D AO LONGO DE SEU EIXO HORIZONTAL
E VERTICAL

logoPPGCIMA->setPos(20, 450, 0);// POSICIONE O LOGOTIPO PPGCIMA NA PARTE SUPERIOR ESQUERDA DA
TELA (COORDENADAS DE PIXEL)
logoPPGCIMA->setZoomHV(0.4, 0.4);// DIMENSIONE O LOGOTIPO PPGCIMA AO LONGO DE SEU EIXO
HORIZONTAL E VERTICAL

// AQUI NÓS SUBSTITUÍMOS TODOS OS PIXELS PRETOS (0, 0, 0) DO BITMAP CHAI3D
// LOGO COM PIXELS PRETOS TRANSPARENTES (0, 0, 0, 0).
// ISTO PERMITE-NOS FAZER O FUNDO DO LOGOTIPO PARECER TRANSPARENTE.

logoChai->m_image.replace(cColorb(0, 0, 0),// COR RGB ORIGINAL cColorb(0, 0, 0, 0)// NOVA COR
RGBA);

// HABILITAR TRANSPARÊNCIAS

logoChai->enableTransparency(true);
logoUnB->enableTransparency(true);
logoPPGCIMA->enableTransparency(true);

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

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

cGenericHapticDevice* dispositivoHaptico;// CLASSE QUE CHAMA O DISPOSITIVO HÁPTICO.
disp_haptic->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 ======================== */
/* ================================================================================== */

olho = new cMesh(mundo);// CRIA O OLHO NO MUNDO.

mundo->addChild(olho);// ANEXA O OLHO 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("CHAI3D - DEFORMAÇÃO DE BIOMEMBRANA COM RETORNO DE FORÇA E CONSERVAÇÃO DE
VOLUME");// TÍTULO NA BARRA DA JANELA

// CRIAR MENU PARA BOTÃO DIREITO DO MOUSE.

glutCreateMenu(opcaoMouse);
glutAddMenuEntry("TELA CHEIA", TELA_CHEIA);
glutAddMenuEntry("JANELA COMPACTA", JANELA_COMPACTA);
glutAttachMenu(GLUT_RIGHT_BUTTON);

/* ================================================================================== */
/* ================================== 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);
}

/* ================================================================================== */
/* ====================== MENU DE COMANDOS - FUNÇÕES DO TECLADO ====================== */
/* ================================================================================== */

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

```
exit(0);// SAIR DA SIMULAÇÃO.
}

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

// BOTÃO NÚMERO 2:
if (tecla == '2'){
bool useTexture = olho->getUseTexture();// LIGA A TEXTURA.
olho->setUseTexture(!useTexture);// DESLIGA A OPÇÃO TEXTURA.
}
}

/* ================================================================================== */
/* ================= 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;
}

elseif (estado == GLUT_UP) {// ESTADO PARA CIMA.
movimentaCamera = false;// MOVIMENTO DA CÂMERA PARA CIMA.
}
}
void opcaoMouse(int value)// FUNÇÃO QUE UTILIZA BOTÃO DIREITO PARA TELA CHEIA OU COMPACTA DA
CENA VIRTUAL
{
switch (value){

case TELA_CHEIA:// HABILITA MODO DE TELA CHEIA
glutFullScreen();
break;

case JANELA_COMPACTA:// REDEFINE A JANELA PARA TAMANHO ORIGINAL
glutReshapeWindow(LARGURA_JANELA, ALTURA_JANELA);
break;
}
}


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

void movimentoMouse(int x, int y) {// FUNÇÃO QUE USA DO MEIO PARA ZOOM NA TELA
if (movimentaCamera){
if (botaoMouse == GLUT_MIDDLE_BUTTON) {// BOTÃO DO MEIO.
distanciaCamera = distanciaCamera - 0.02 * (y - mouseY);// MODIFICA A DISTÂNCIA DO OLHO COM
RELAÇÃO A CÂMERAE ESQUERDO PARA GIRAR A CENA VIRTUAL.
}

elseif (botaoMouse == GLUT_LEFT_BUTTON) {// BOTÃO ESQUERDO (MODIFICA O ÂNGULO DA CÂMERA COM
RELAÇÃO AO OLHO.
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){
olho->computeAllNormals(true);// ATUALIZAÇÃO DA POSIÇÃO NORMAL DO OLHO.

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)
{
cVector3d posicao (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 O OLHO.

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

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

 {
cVector3d posicao;// CÁLCULA A POSIÇÃO DA PROVA (ESFERA) ENQUANTO A DEFORMAÇÃO ESTIVER
ACONTECENDO.

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

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


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

posicao = esfera->getDeviceGlobalPos();// POSIÇÃO GLOBAL DA PROVA (ESFERA)
cVector3d posVertex = vertices->getPos();// OBTÊM A POSIÇÃO DOS VÉRTICES QUE FORMAM O OLHO


/* ================================================================================== */
/* ====== DECLARAÇÃO DE VARIÁVIES PARA DEFORMAÇÃO DO OLHO COM CONSERVAÇÃO DE VOLUME ====== */
/* ================================================================================== */
double u=0.0,  u0z=0.0;
double Fx=0.0, Fy=0.0, Fz=0.0;// FORÇA APLICADA EM UM PONTO DO OLHO.
double modPosicao = sqrt(cSqr(posicao.x) + cSqr(posicao.y) + cSqr(posicao.z));// MÓDULO DA
POSIÇÃO DOS EIXOS X, Y E Z.
double u0 = 0.0;// POSIÇÃO DA PROVA
double a = 0.05;// RAIO DO DISCO A.
double b= 1.0;// RAIO DA MEMBRANA B
```

```
double r = sqrt(cSqr(posVertex.x-posicao.x)+cSqr(posVertex.y-posicao.y)); // MÓDULO DAS
COMPONENTES X E Y, ONDE R= RAIZ(X^2+Y^2).
double A = (Pi*cSqr(b));// ÁREA DO OBJETO DE APLICAÇÃO DE PROVA
double tau= 1.0;// COEFICIENTE PARA TENSÃO SUPERFICIAL DA MEMBRANA.


/* ============================================================================== */
/* ==========DETECÇÃO DE COLISÃO E DEFORMAÇÃO DO OLHO COM CONSERVAÇÃO DE VOLUME ======= */
/* ============================================================================== */

if ((cSqr(posicao.x) + cSqr(posicao.y))<0.22)// TESTE DA DETECÇÃO DE COLISÃO PARA
PROTUBERÂNCIA (ELEVAÇÃO DA SUPERFICIE DO OLHO).
u0 = (modPosicao)-sqrt(cSqr(posicao.x) + cSqr(posicao.y) +cSqr(sqrt(cSqr(0.62)-
cSqr(posicao.x)-cSqr(posicao.y))+0.45)); // CALCULO EM MÓDULO DA POSIÇÃO DA PROVA (U0) A
PARTIR DA PROTUBERÂNCIA (ELEVAÇÃO DA SUPERFICIE DO OLHO).
else
u0 = (modPosicao)-(0.95);// CASO A ESFERA BRANCA ESTEJA FORA DA PROTUBÊNCIA, MAS NO OLHO, O
MÓDULO DA POSIÇÃO PROVA É DIMINUIDO DE APROXIMADAMENTE 1.
if (u0 <= 0.0){// CONDIÇÃO PARA DEFORMAÇÃO DO OLHO - CASO ESTEJA ABAIXO DE 0.0 OU SEJA TOCANDO
O OLHO
if ((cSqr(posicao.x) + cSqr(posicao.y))<0.22)// LEITURA DA POSIÇÃO DA PROVA (U0) EM Z A PARTIR
DA PROTUBERÂNCIA (ELEVAÇÃO DA SUPERFICIE DO OLHO).
u0z = posicao.z-(sqrt(cSqr(0.62)-cSqr(posicao.x)-cSqr(posicao.y))+0.45);  // LEITURA DA
POSIÇÃO DE U SOMANDO A DEFORMAÇÃO E DESENHANDO O OLHO.
else
u0z = posicao.z - sqrt(0.95 - cSqr(posicao.x)-cSqr(posicao.y));

if (r <= a )
u = u0z;// FAZ A ESFERA ACOMPANHAR A PROVA CONDIÇÃO PARA QUE NÃO OCORRA A DEFORMAÇÃO DO OLHO E
LIMITA QUE DEFORMAÇÃO TENDA AO INFINITO.

else {
double Forca = -12*A*Pi*u0*tau*b/(Pi*pow(b,3.0)-3*Pi*cSqr(a)*b+2*Pi*pow(a,3)+6*A*b*log(a/b)-
6*A*a+6*A*b);  // EQUAÇÃO PARA CÁLCULO DA FORÇA SOBRE A MEMBRANA PARA CONSERVAÇÃO DE VOLUME
(EQUAÇÃO 4.40)
double ua = u0 + Forca/(2*Pi*tau)*(-a/b);        // EQUAÇÃO PARA CÁLCULO DA CONSTANTE, OU SEJA,
ua (EQUAÇÃO 4.38)
u =  Forca/(2*Pi*tau)*log(a/r)+ua+r*Forca/(2*tau*Pi*b);// EQUAÇÃO PARA CÁLCULO DA DEFORMAÇÃO
DA MEMBRANA COM CONSERVAÇÃO DE VOLUME (EQUAÇÃO 4.37)

Fx = Forca*posicao.x/modPosicao;     // FORÇA APLICADA EM UMA REGIÃO DO OLHO NO EIXO X
Fy = Forca*posicao.y/modPosicao;     // FORÇA APLICADA EM UMA REGIÃO DO OLHO NO EIXO Y
Fz = Forca*posicao.z/modPosicao;     // FORÇA APLICADA EM UMA REGIÃO DO OLHO NO EIXO Z
}
}

/* ============================================================================== */
/* ================= APLICAR DESLOCAMENTO DA MEMBRANA - FORMATO DO OLHO ================= */
/* ============================================================================== */

if ((cSqr(posVertex.x)+ cSqr(posVertex.y))<0.95)// TESTE DA POSIÇÃO DOS VERTICES DA MALHA QUE
FORMA O OLHO
{
if ((cSqr(posVertex.x) + cSqr(posVertex.y))<0.22)// CRIA PROTUBERÂNCIA (ELEVAÇÃO DA SUPERFICIE
DO OLHO).
posVertex.z= u+(sqrt(cSqr(0.62)-cSqr(posVertex.x)-cSqr(posVertex.y))+0.45);// LEITURA DA
POSIÇÃO DE U SOMANDO A DEFORMAÇÃO E DESENHANDO O OLHO.
else
 posVertex.z = u+ sqrt(0.95 - cSqr(posVertex.x)-cSqr(posVertex.y));       // CASO O TESTE NÃO
SEJA VALIDO CRIA A SUPERFICIE SEM PROTUBERÂNCIA.
}


else posVertex.z=0.0;// CASO O TESTE INICIAL NÃO SEJA VALIDO NÃO CRIA A SUPERFICIE DO OLHO.

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


/* ============================================================================== */
/* ====================== ENVIAR FORÇA PARA DISPOSITIVO ============================== */
/* ============================================================================== */

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

eixo.x=-Fx*20.0;// INTENSIDADE DA FORÇA NO EIXO X.
eixo.y=-Fy*20.0;// INTENSIDADE DA FORÇA NO EIXO Y.
```

```
eixo.z=-Fz*20.0;// INTENSIDADE DA FORÇA NO EIXO Z (EIXO NO QUAL É REALIZADO A DEFORMAÇÃO DO
OLHO).

if ((cSqr(posicao.x) + cSqr(posicao.y))<0.22)       // TESTA E ENVIA AO PHANTOM SE A FORÇA
ESTÁ APLICADA NA PROTUBERÂNCIA (ELEVAÇÃO DA SUPERFICIE DO OLHO).
{
if (posicao.z<= sqrt(cSqr(0.62)-cSqr(posicao.x)-cSqr(posicao.y))+0.45)// TESTA E ENVIA AO
PHANTOM SE A FORÇA ESTÁ SENDO APLICADA ABAIXO DA PROTUBERÂNCIA (ELEVAÇÃO DA SUPERFICIE DO
OLHO).
phantom->setForce(eixo);
}
else if (posicao.z<= sqrt(0.95 - (cSqr(posicao.x)) -(cSqr(posicao.y))))// ENVIAR FORÇAS PARA O
DISPOSITIVO SE A ESFERA BRANCA ESTIVER ABAIXO DA SUPERFICIE DA PROTUBEÂNCIA NO EIXO z.
phantom->setForce(eixo);
else esfera->applyForces();   // ENVIA FORÇA ZERO PARA O DISPOSITIVO HÁPTICO CASO TODOS OS
TESTES NÃO SEJAM VÁLIDOS.
}
}
terminarSimulacao = true;     // FINALIZA A SIMULAÇÃO.
}}
/* ============================================================================================ */
/* ================= FUNÇÃO PARA CRIAÇÃO DA MALHA QUE FORMA O OLHO ======================== */
/* ============================================================================================ */

int plano()
{
cTexture2D* newTexture = new cTexture2D();// CRIAR UM ARQUIVO DE TEXTURA

mundo->addTexture(newTexture);

bool fileload = newTexture->loadFromFile("resources/images/olho5.1.bmp");// LEITURA DE ARQUIVO
DE TEXTURA
if (!fileload)
{
#ifdefined(_MSVC)
fileload = newTexture->loadFromFile("../../../bin/resources/images/olho5.1.bmp");// LEITURA DE
ARQUIVO DE TEXTURA
#endif
}
if (!fileload)
{
printf("Error - Texture image failed to load correctly.\n");// TESTE SE A LEITURA FOI FEITA
CORRETAMENTE
fechar();
return (-1);
}

// get the size of the texture image (U and V)
// int texSizeU = newTexture->m_image.getWidth();
//  int texSizeV = newTexture->m_image.getHeight();
int texSizeU = 100; // LARGURA DO OLHO
int texSizeV = 100;// COMPRIMENTO DO OLHO

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 = 1.99 / (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 DO OLHO:

int u,v;

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

// CÁLCULA A POSIÇÃO DOS VÉRTICES

   cColorb color = newTexture->m_image.getPixelColor(u,v);

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

// CRIA UM NOVO VÉRTICE

unsignedint index=olho->newVertex (px, py, 0.0); // CRIA UM PLANO INICIAL PARA SER DEFORMADO.

cVertex* vertex = olho->getVertex(index);

// CALCULA COORDENADAS DA TEXTURA
tu = (double)u / texSizeU;
tv = (double)v / texSizeV;
vertex->setTexCoord(tu, tv);

}
}

// 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:
unsignedint index00 = ((v + 0) * texSizeU) + (u + 0);
unsignedint index01 = ((v + 0) * texSizeU) + (u + 1);
unsignedint index10 = ((v + 1) * texSizeU) + (u + 0);
unsignedint index11 = ((v + 1) * texSizeU) + (u + 1);

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

olho->setUseTexture(true);

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

olho->computeBoundaryBox(true);// CÁLCULA OS LIMITES DA CAIXA INVISÍVEL ONDE ESTÁ O OLHO.

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

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

cVector3d span = cSub(max, min);// TAMANHO DO OLHO.

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

double escala = TAM_ESCALA_MALHA / tamanho;// DECLARAÇÃO DA VARIÁVEL PARA CALCULAR A ESCALA DA
MALHA DO OLHO.

olho->computeBoundaryBox(true);// CÁLCULA NOVAMENTE O TAMANHO DO OLHO.

olho->createAABBCollisionDetector(1.01 * raioProxy, true, false);// DETECTA COLISÕES ENTRE O
OLHO E OUTROS OBJETOS NA CENA SIMULAÇÃO.

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

olho->setNormalsProperties(0.01, cColorf(0.95, 0.0, 0.0, 0.95), true);// OBTER TAMANHO DAS
NORMAIS.
```

```
olho->setUseCulling(false);// PROCESSAR GRAFICAMENTE AMBOS OS LADOS DOS TRIÂNGULOS.

olho->computeGlobalPositions();// ATUALIZAR POSIÇÃO GLOBAL PARA O OLHO.

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.

        }
```