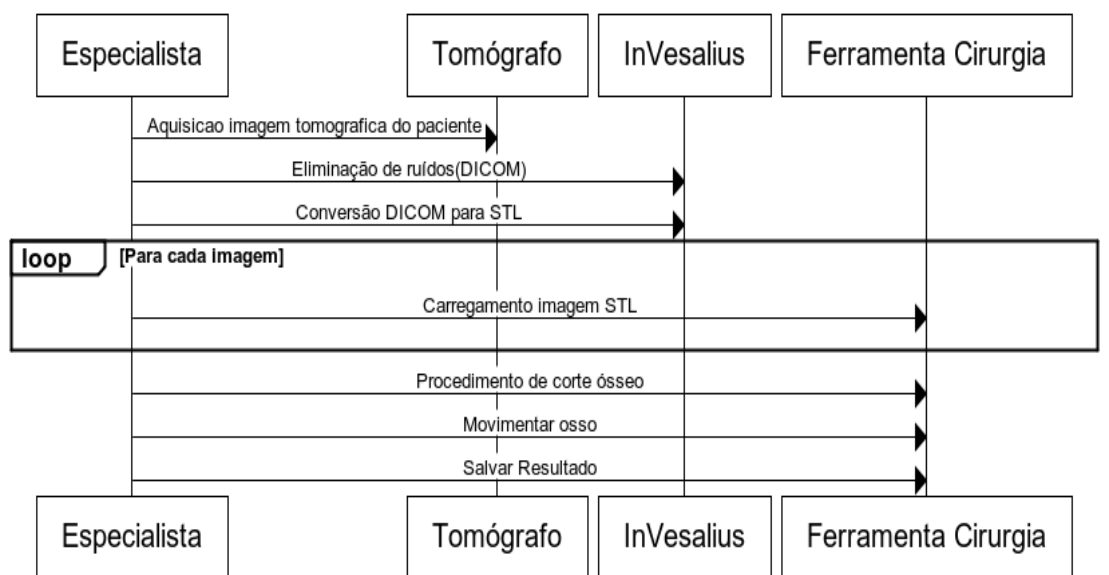


é exibido no diagrama UML de sequencia, de forma a identificar os principais elementos da solução, bem como a relação entre os mesmos.

Observando o diagrama da Figura 29, o bloco (ator) com a denominação Ferramenta Cirurgia, para fins práticos representa o sistema Protótipo VTO – UFU – Visualização do Tratamento Ortognático aqui desenvolvido nessa pesquisa.

FIGURA 29: Diagrama de Sequencia do Sistema



www.websequencediagrams.com

Nota-se a participação do ator tomógrafo, nesse caso podendo representar igualmente tanto um dispositivo de tomografia, ou um médico no papel de operador desse dispositivo.

O diagrama de sequencia acima apresenta também uma forma de loop, com o objetivo de deixar claro que mais de uma imagem (ou parte) poderá ser utilizada dentro do ciclo de vida do sistema desenvolvido.

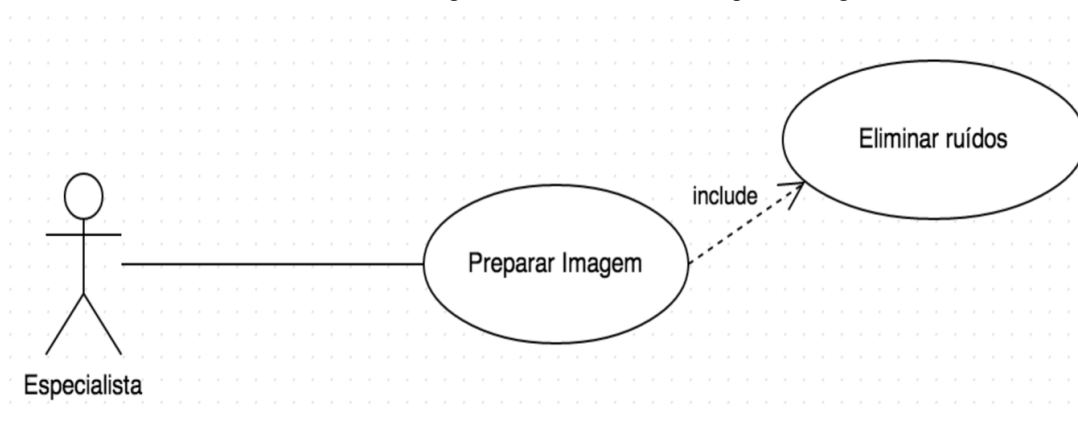
4.2.3 Casos de Uso

Foram criados casos de uso para atender esses requisitos identificados anteriormente nesse capítulo. Por esse trabalho buscar trazer contribuições para o âmbito acadêmico, são apresentados a seguir os diagramas de caso de uso, bem como a sua descrição para o fluxo principal.

4.2.3.1 - Casos de Uso – Preparar Imagem

Por meio do diagrama de caso de Uso “Preparar Imagem” verifica-se os vários passos que são necessários até obter a imagem tomográfica do paciente com todos os requisitos tratados, conforme é possível visualizar na Figura 30, entender detalhes na descrição textual desse caso de uso.

FIGURA 30: Diagrama de Caso de Uso Preparar Imagem



Fonte: Acervo Próprio

Segue a tabela 3 a descrição do caso de uso Preparar Imagem.

TABELA 3 - Descrição caso de Uso Preparar Imagem

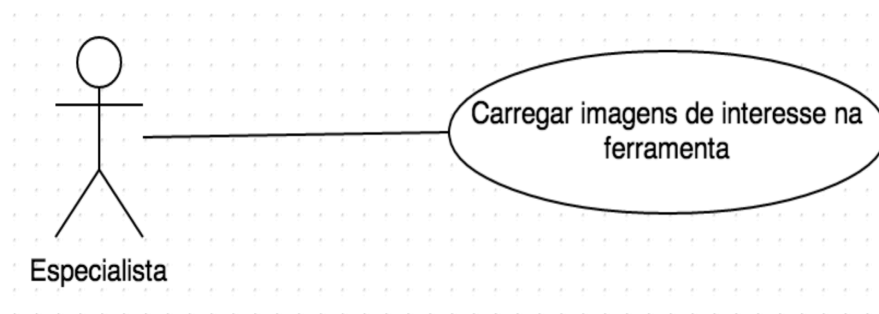
Caso de Uso	Preparar Imagem
Ator	Especialista
Fluxo Principal	1. O especialista identifica a existência de ruído na imagem DICOM
	2. O especialista aplica filtros para remover esses ruídos utilizando softwares auxiliares
	3. O sistema aplica filtros na imagem.
	4. O especialista avalia a necessidade de desacoplar mandíbula e maxila na imagem DICOM original
	5. O especialista de forma manual editora a imagem eliminando pontos de união entre as partes
	6.O especialista aplica a conversão do imagem no formato DICOM para STL
	7.O sistema gera novos arquivos com essas imagens convertidas

Fonte: Acervo Próprio

4.2.3.2 - Casos de Uso – Carregar Imagem

O caso de uso Carregar Imagem, simbolizado pela Figura 31, apresenta os passos e interações entre o especialista e o sistema de forma que o especialista determina nesse momento com quais imagens o procedimento será realizado.

FIGURA 31: Diagrama de Caso Carregar Imagem



Fonte: Acervo Próprio

Segue a tabela 4 a descrição do caso de uso Carregar Imagem.

TABELA 4 - Descrição Caso de Uso Carregar Imagem

Caso de Uso	Carregar Imagem
Ator	Especialista
Pré-Requisito	Execução do caso de uso Preparar Imagem
Fluxo Principal	1. O especialista separa quais arquivos STL são de interesse
	2. Para cada arquivo STL de interesse o especialista solicita o carregamento da imagem na área de trabalho gráfica
	3. O sistema carrega do sistema de arquivos e apresenta as imagens carregadas em cores diferentes no <i>canvas</i>
	4. O especialista confere se as imagens corretas foram carregadas

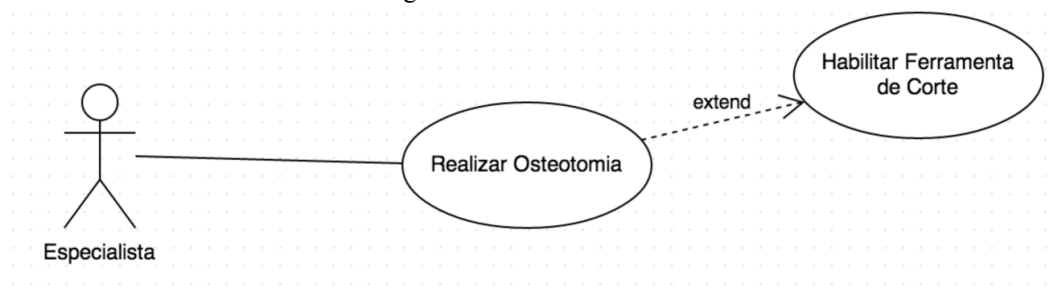
Fonte: Acervo Próprio

4.2.3.3 - Casos de Uso – Realizar Osteotomia

O caso de uso Realizar Osteotomia, certamente é o mais importante para o sistema Protótipo VTO – UFU – Visualização do Tratamento Ortognático. Por meio da descrição desse caso de uso, pode-se compreender de forma simples e ao mesmo tempo detalhada a troca de mensagens entre o especialista e o sistema.

A fim de realizar o procedimento cirúrgico por meio de movimentos no sistema que promovem a osteotomia. Pode-se observar que o caso de uso Realizar Osteotomia, estende o comportamento do caso de uso que aborda o controle da ferramenta de corte exibido na Figura 32.

FIGURA 32: Diagrama de Caso de Uso Realizar Osteotomia



Fonte: Acervo Próprio

Segue a tabela 5 a descrição do caso de uso Realizar Osteotomia.

TABELA 5 - Descrição de Caso de Uso Realizar Osteotomia

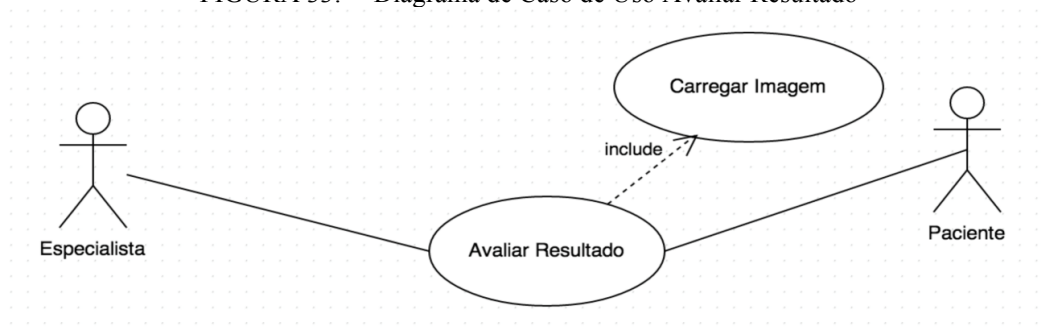
Caso de Uso	Realizar Osteotomia
Ator	Especialista
Pré-Requisito	Execução do caso de uso Carregar Imagem
Fluxo Principal	1. O especialista avalia e analisa qual procedimento será realizado
	2. O sistema apresenta a ferramenta de corte
	3. O especialista movimenta a câmera para uma posição adequada
	4. O sistema apresenta a imagem a partir de um outro ponto de visualização
	5. O especialista movimenta a ferramenta de corte para a área de interesse
	6. O sistema apresenta pontos de movimento (expansão e retração) na ferramenta de corte
	7. O sistema aplica o corte na área em que esta posicionada a ferramenta de corte, dividindo a imagem em duas partes, o que esta dentro e fora da área de corte
	8. Para cada parte, com base em sua experiência, o especialista realiza o movimento
	9. O sistema salva as novas partes e suas novas posições

Fonte: Acervo Próprio

4.2.3.4 - Casos de Uso – Avaliar Resultado

O caso de uso Avaliar Resultado, exibido na Figura 33, e explicado aqui, é fundamentalmente o mais relevante para os envolvidos, nesse caso o especialista e o paciente. Observando a descrição desse caso de uso, entende-se que após a realização da osteotomia, ambos: especialista e paciente avaliam o resultado. Nesse momento o especialista tem a oportunidade de explicar o resultado para o paciente de forma a facilitar o entendimento do mesmo.

FIGURA 33: Diagrama de Caso de Uso Avaliar Resultado



Fonte: Acervo Próprio

Segue a tabela 6 a descrição do caso de uso Avaliar Resultado.

TABELA 6 - Descrição Caso de Uso Avaliar Resultado

Caso de Uso	Avaliar Resultado
Ator	Especialista, Paciente
Pré-Requisito	Execução do caso de uso Realizar Osteotomia
Fluxo Principal	1. O especialista carrega na área de trabalho todas as partes de interesse para a sua avaliação, inclusive partes que não sofreram quebra, como por exemplo a caixa craniana.
	2. O sistema apresenta as imagens na área de trabalho
	3. O especialista solicita o desativação da ferramenta de corte
	4. O sistema desabilita a ferramenta de corte
	5. O especialista e o paciente, interagem com o sistema rotacionando a câmera
	6. O especialista explica e avalia o resultado da simulação virtual com o paciente como uma forma de comunicação do procedimento

Fonte: Acervo Próprio

As seções subsequentes desse capítulo permitem que seja compreendido de forma mais consistente essas interações com o sistema desenvolvido (Protótipo VTO – UFU –

Visualização do Tratamento Ortognático) por meio de *screenshots* e explicações detalhadas sobre cada passo e tela do sistema. Embora essa seção subsequente tenha ligeira semelhança com um guia ou tutorial, o objetivo final é expressar de forma mais realista o que foi visto nos casos de uso desta seção, bem como um pequeno background para a compreensão dos formatos de imagens citados nos casos de uso, são eles DICOM e STL. O texto a seguir contempla essas explicações.

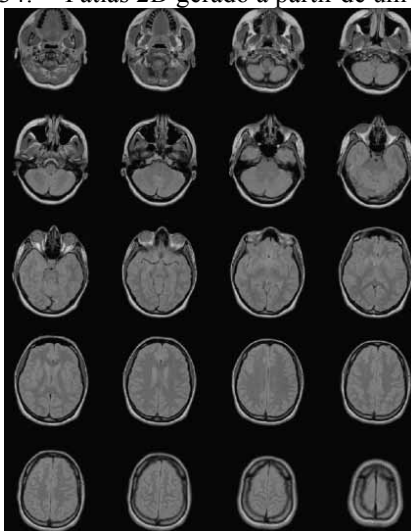
4.3 Formatos Digitais para Imagens Médicas

O padrão de indústria DICOM (Digital Imaging and Communications in Medicine) foi desenvolvido de acordo com os Procedimentos do Comitê de Normas DICOM, é estruturado como um multi - documento de peça com as diretrizes estabelecidas no seguinte documento:

Directivas ISO/IEC, 1989 Parte 3: Elaboração e Apresentação das Normas Internacionais utilizada pela maioria dos equipamentos que fazem obtenção de imagens médicas tais como tomografia computadorizada e ressonância magnética, define um conjunto de dados sobre o paciente, o equipamento, dados das imagens adquiridas, etc.

Do ponto de vista desse trabalho, o interesse é somente no conteúdo referente às imagens adquiridas.

FIGURA 34: Fatias 2D gerado a partir de um tomógrafo



Fonte: Accorsi e Velasco (2011)

Mesmo que a definição do padrão DICOM permita que informações da imagem em 3D sejam representadas, isso ainda é raro na família de equipamentos atualmente disponíveis,

dessa forma, o que é obtido e armazenado de fato conforme Figura 34, é uma série de imagens 2D (duas dimensões), imagens estas que podem ser interpretadas simplesmente como fatias.

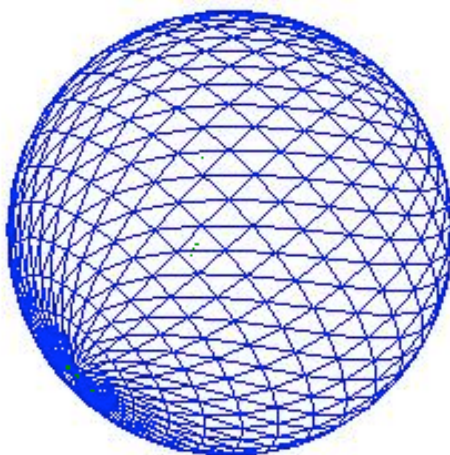
Estes pixels formadores das fatias 2D, ainda possuem informações sobre densidade, representadas através de tons de cinza que variam de 0 (preto) a 255 (branco), conhecido por escala de Hounsfield (ACCORSI E VELASCO, 2011). Esse assunto já havia sido brevemente discutido no capítulo 2 (Fundamentos).

A cadeia de fatias 2D, poderiam ser combinadas a fim de buscar uma rica representação em três dimensões. Fatores como a grande quantidade de informações presentes nos arquivos criados seguindo o padrão DICOM, além das imagens originalmente adquiridas (*raw data*) estarem em 2D, dificultam a sua utilização direta em softwares de RP (*rapid prototyping*) e CAD (*Computer Aided Design*).

Uma alternativa a isso, é fazer a conversão do *dataset* (conjunto de dados da imagem) presente no arquivo DICOM para o formato STL (*STereoLithography*), também conhecido por *Structure Triangular Language*, uma vez que esse formato representa uma malha de polígonos que representam a superfície de um objeto em 3D. (Ideyama e Carvalho, 2011)

O formato (STL) tem se tornado largamente utilizado como fonte para prototipagem rápida para a criação de biomodelos físicos.

FIGURA 35: Esfera composta por polígonos



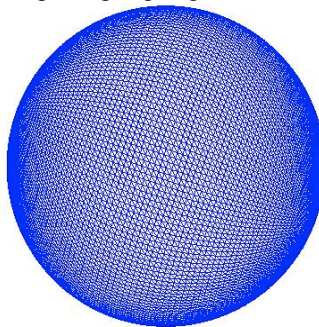
Fonte: AcervoPróprio

A quantidade de pontos presentes na malha e facetados como triângulos, chamados aqui por polígonos, pode variar, determinando a suavização do modelo 3D gerado, Figura 35.

As Figuras 35 e 36 apresentam uma esfera com um menor e maior numero de polígonos respectivamente. Em contra partida um excesso de suavização pode fazer com que

um número exagerado de polígonos aumente o tamanho do modelo final, dificultando a sua manipulação, como apresentado na Figura 36.

FIGURA 36: Esfera composta por polígonos com maior número de pontos



Fonte: Acervo Próprio

A fim de permitir trabalhar de forma mais simples e eficiente, no sistema proposto a imagem original oriunda de uma tomografia computadorizada no formato DICOM será transformada para o formato STL, dessa forma a composição de fatias em 2D presentes no arquivo DICOM, serão combinadas e transformadas em uma malha de polígonos conforme determinado pelo padrão STL.

Na sequência, será usado o software *InVesalius*TM para realizar essa transformação. É importante destacar que o procedimento de transformação poderia também ser realizado com auxílio de outras ferramentas diferentes do *InVesalius*TM.

Esse processo de transformação se faz necessário uma vez que o Protótipo VTO – UFU – Visualização do Tratamento Ortognático, sistema proposto, irá trabalhar em ambiente 3D, com isso a necessidade da transformar a origem composta por um conjunto de imagens em 2D para um modelo em 3D utilizando o formato STL, representado por uma malha de polígonos.

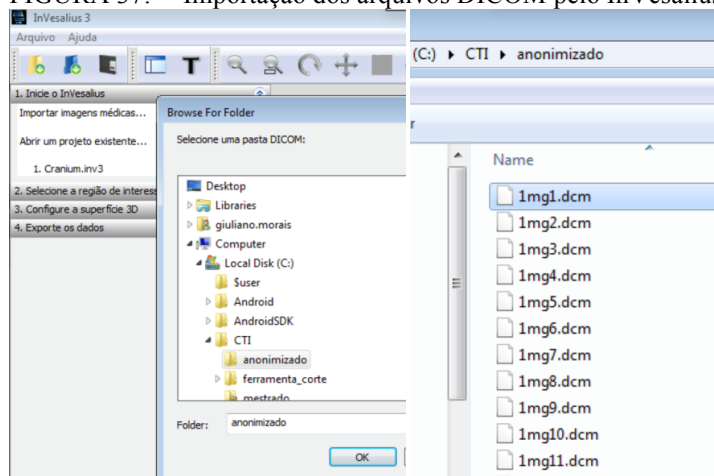
4.4 Conversão DICOM para STL

Inicialmente deve-se realizar a importação do arquivo DICOM para então transformá-lo em um arquivo no formato STL, como já citado para essa tarefa utiliza-se nesse trabalho o sistema *InVesalius*TM. A fim de obter melhores resultados para o objeto de estudo desse trabalho, é recomendável que a boca esteja semiaberta buscando obter uma separação suave entre os dentes. Caso não seja possível obter uma imagem nessas condições, ainda assim é possível chegar a um resultado próximo fazendo a separação dessas partes por meio de um

procedimento puramente manual utilizando mais uma vez o software *InVesalius*TM (ou similar) para a edição da imagem, conforme apresentado pelo fluxo na sequência.

Como esperado, o primeiro passo a ser realizado é a importação dos arquivos que compõem a imagem DICOM conforme apresentado na Figura 37.

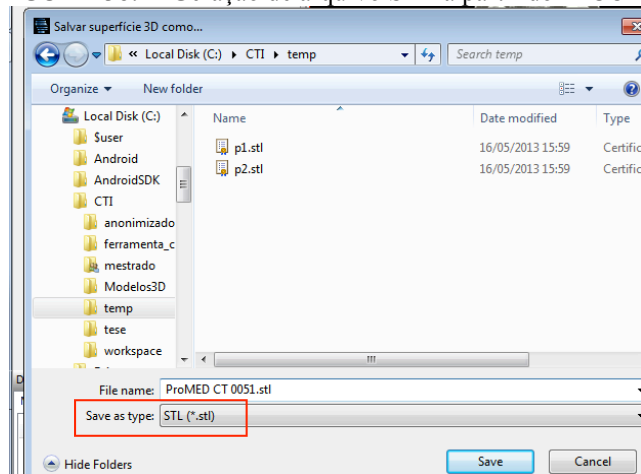
FIGURA 37: Importação dos arquivos DICOM pelo InVesalius



Fonte: Acervo Próprio

Uma vez que a importação foi concluída, pode-se simplesmente exportar a imagem médica para o formato STL como destacado pela Figura 38.

FIGURA 38: Geração de arquivo STL a partir do DICOM



Fonte: Acervo Próprio

No exemplo apresentado aqui, a fim de apresentar um fluxo mais completo, utilizou-se uma imagem tomográfica, onde há presença de ruídos. Percebe-se que a maxila esta grudada

com a mandíbula, e considerando que o objetivo final do procedimento certamente implicará em algum tipo de movimentação, se faz, portanto necessário separar esses dois conjuntos ósseos, nesse caso a mandíbula e a maxila.

FIGURA 39: Editoração das fatias sagitais para separação da mandíbula

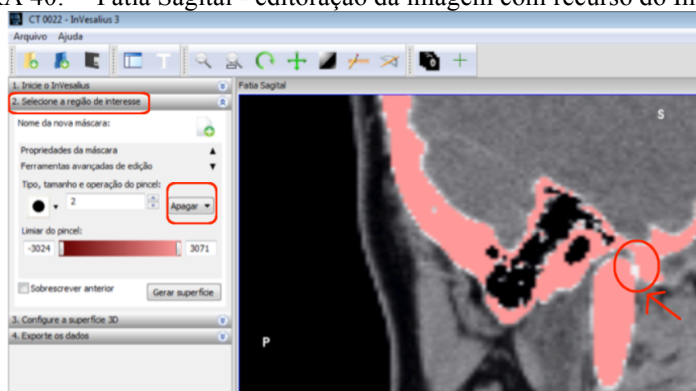


Fonte: Acervo Próprio

A Figura 39 destaca pelo círculo em vermelho o local onde a imagem apresenta uma junção da maxila com a caixa craniana e mandíbula chamada de Articulação temporo-mandibular (ATM).

Dessa forma, por meio de visualização das fatias sagital, esta sendo utilizado o modo de edição avançado do *InVesalius*TM para realizar essa separação, conforme destacado na Figura 40.

FIGURA 40: Fatia Sagital - editoração da imagem com recurso do InVesalius

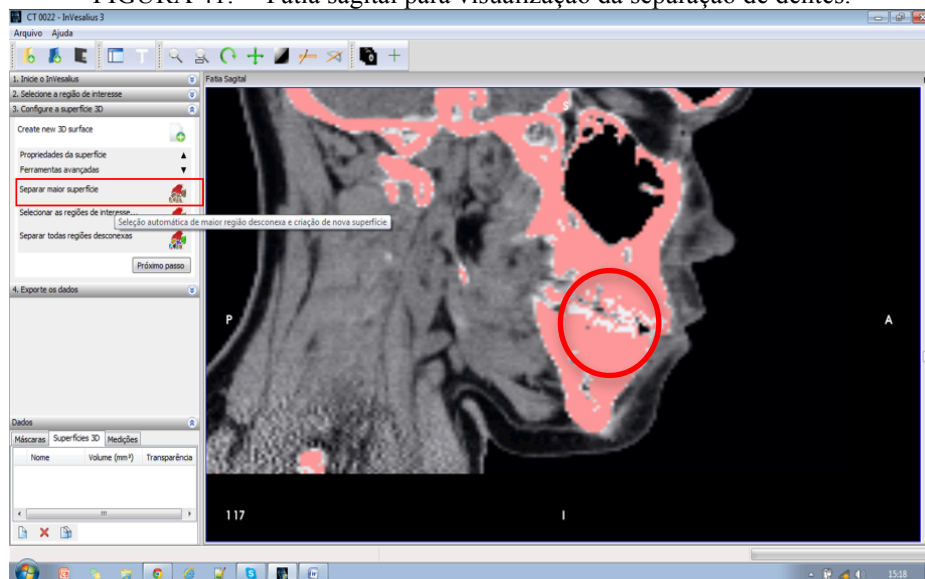


Fonte: Acervo Próprio

Repetindo o procedimento, dessa vez utilizando outra fatia de visualização, pode-se perceber que ainda se faz necessário separar os dentes entre a maxila e a mandíbula. Esse

passo facilmente não seria necessário caso a tomografia tivesse sido realizada com a boca semi-aberta.

FIGURA 41: Fatia sagital para visualização da separação de dentes.



Fonte: Acervo Próprio

A Figura 41 apresenta uma visão diferente, de forma a facilitar a identificação desses dois conjuntos ósseos, maxila e mandíbula, conectadas por meio de pontos entre os dentes.

Conforme apresentado até aqui, o procedimento de separação somente é necessário em algumas situações, particularmente quando a imagem tomográfica original no formato DICOM não apresenta separação suficiente entre as partes (Crânio-maxila e mandíbula) para uma fácil manipulação entre as partes no sistema proposto, a ser detalhado funcionalmente na sequência.

Como visto, apoia-se no uso do software livre de imagens médicas *InVesalius*TM, para reconstrução tridimensional de estruturas anatômicas e edição das mesmas. Por ser simples e intuitivo, esse software se apresenta como uma forma viável para a utilização de especialistas que exercem o procedimento de correção da maloclusão dentária por meio do procedimento cirúrgico, considerando para isso o enriquecimento de suas capacidades, com as funcionalidades desenvolvidas nesse trabalho.

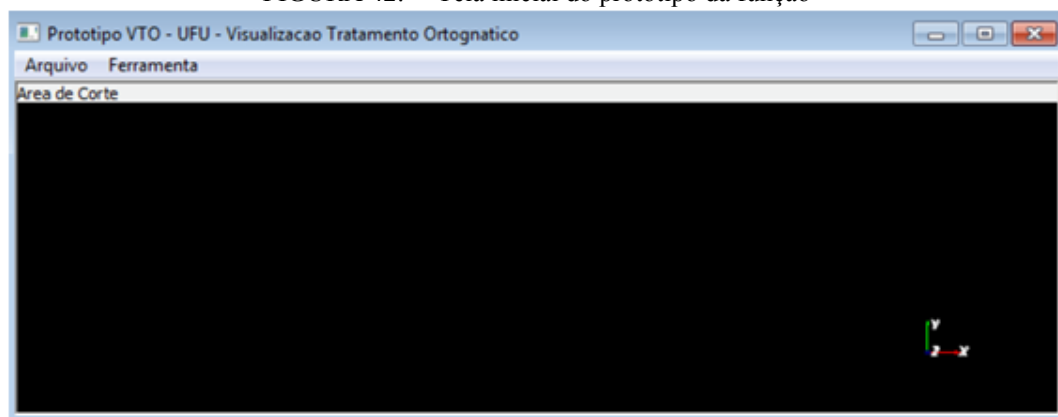
Considera-se que o uso imagens de tomográficas computadorizadas para obter um maior realismo e precisão no estudo dos movimentos que ocorrerão durante o procedimento, trazem significativas vantagens para a realização de análise e simulação do procedimento cirúrgico, mas precisamente para o processo de osteotomia.

4.5 Fluxo de Utilização do Sistema Proposto

Considerando que a imagem tomográfica nesse estágio já foi devidamente convertida para STL com o *InVesalius*TM, os pontos de contato entre a mandíbula e a maxila já estão separados, o especialista nesse momento possui os requisitos mínimos para a utilização da ferramenta de análise da simulação e VTO desenvolvida pelo autor dentro do que está sendo proposto.

Ao iniciar a ferramenta Protótipo VTO – UFU – Visualização do Tratamento Ortognático, uma área de trabalho minimalista é apresentada conforme Figura 42.

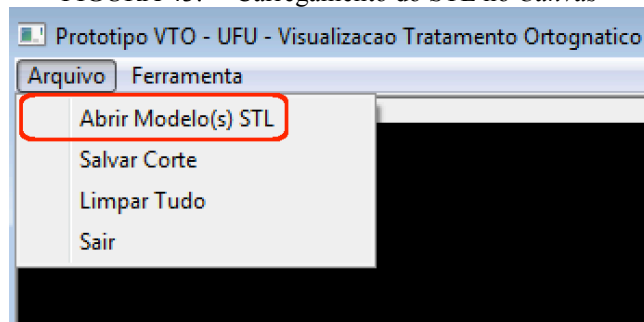
FIGURA 42: Tela inicial do protótipo da função



Fonte: Acervo Próprio

No passo seguinte, o médico deverá fazer o carregamento da imagem STL, já convertida do formato DICOM. Esse procedimento de carregamento é realizado através menu Arquivo e na sequência Abrir Modelo(s) STL, como destacado na Figura 43.

FIGURA 43: Carregamento do STL no *Canvas*

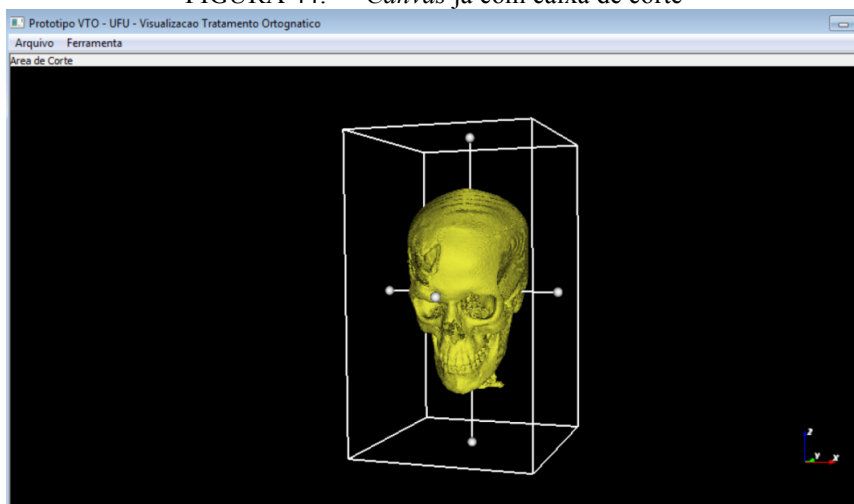


Fonte: Acervo Próprio

Uma vez que o carregamento do arquivo STL foi realizado, na área de trabalho da ferramenta (*Canvas*) a imagem 3D é apresentada dentro de um cubo. Esse cubo como será

apresentado mais adiante é a ferramenta de corte ósseo que o especialista (usuário) irá utilizar, através da manipulação das arestas que determinam os eixos de orientação das imagens.

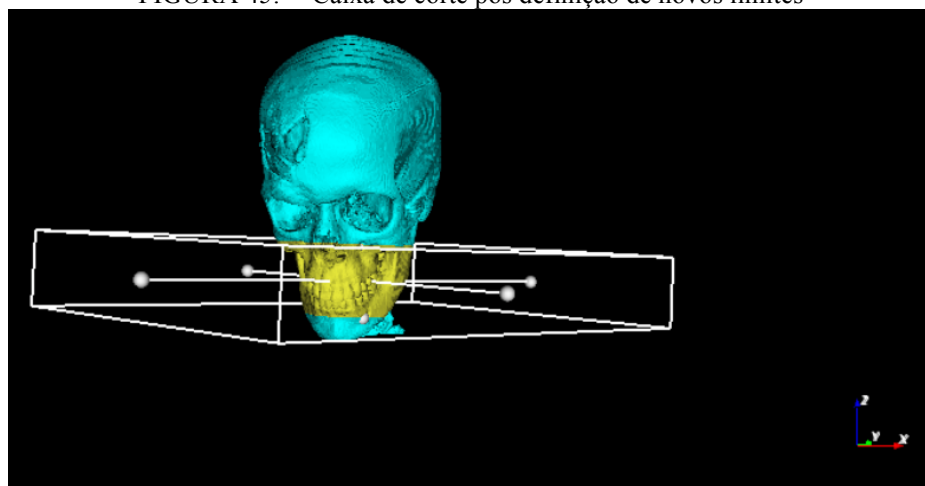
FIGURA 44: *Canvas* já com caixa de corte



Fonte: Acervo Próprio

A ferramenta de corte é iniciada por padrão tão logo a imagem STL seja carregada no *canvas*, conforme Figura 44, e pode facilmente ser manipulada por meio dos pontos específicos destacados no cubo representado por pequenas esferas, onde pode-se expandi-lo, comprimi-lo, e movimenta-lo em qualquer um dos eixos.

FIGURA 45: Caixa de corte pós definição de novos limites



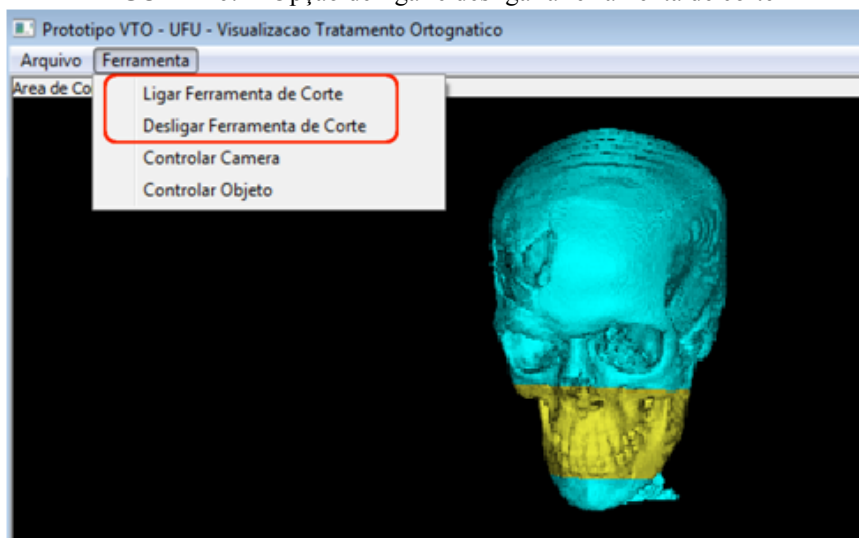
Fonte: Acervo Próprio

Na Figura 45, a ferramenta de corte foi redimensionada e movimentada de forma que a imagem foi dividida em duas partes distintas, a parte interna à ferramenta, e a parte externa.

Essa diferenciação é visualizada de forma fácil por meio de cores distintas. Salienta-se que nesse momento não o autor não simulando um procedimento cirúrgico, e sim apenas a demonstração da utilização da ferramenta.

Dessa forma, o especialista pode movimentar a ferramenta de corte para a região e tamanho de preferência de acordo com o procedimento cirúrgico que esta sendo planejado. Nesse ponto a experiência trazido pelo especialista é de fundamental importância, visto que o protótipo não traz assistentes ou guias para auxílio nesse sentido. O sistema proposto Protótipo VTO – UFU – Visualização do Tratamento Ortognático, oferece ainda outras facilidades visuais, puramente com o objetivo de tornar o trabalho do médico mais simples e objetivo.

FIGURA 46: Opção de ligar e desligar a ferramenta de corte



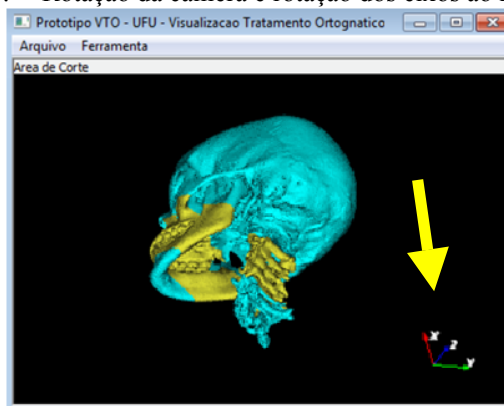
Fonte: Acervo Próprio

Para simplificar a visualização e manipulação das partes, por meio de um simples comando é possível ativar e desativar a ferramenta de corte (Figura 46), dessa forma a ferramenta de corte quando desativada, permite que o *canvas* de forma mais clara, apresente somente os objetos salvos e consequentemente cortados referentes à parte óssea, eliminando assim um elemento visual extra na tela (ferramenta de corte), simplificando com isso a visualização e movimentação das partes pelo usuário.

De forma similar é possível também determinar se o controle do usuário em determinado momento será na câmera ou nos objetos. Utilizando o controle da câmera, o médico pode visualizar os objetos (partes ósseas) de outras formas, simplesmente clicando sobre a área de trabalho e movimentando-a para a posição/eixo desejados.

Perceba ainda, que no canto inferior direito é apresentada uma pequena guia para orientação do médico em relação aos eixos X,Y e Z, conforme Figura 47.

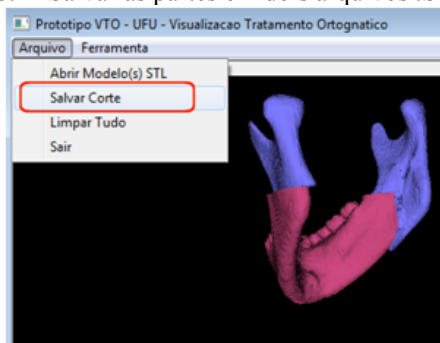
FIGURA 47: Rotação da câmera e rotação dos eixos ao mesmo tempo



Fonte: Acervo Próprio

A alteração do posicionamento da câmera para rotacionar o *canvas* para a posição desejada, é de fato importante para a observação dos eixos de direcionamento, uma vez que são através deles que tornam-se possível a execução da função de movimentação do objeto com controle total e ciência de como o movimento está ocorrendo. Um outro tipo de controle e movimento possível dá-se por meio da opção Controlar Objeto também presente no menu Ferramenta. Essa funcionalidade permite movimentar as partes (objetos) distintas na área de trabalho. Essa ferramenta de movimentação será extensivamente utilizada pelo especialista para realizar o reposicionamento óseo, porém para que seja possível a utilização da ferramenta é fundamental ter as partes ósseas salvas em arquivos independentes.

FIGURA 48: Salvar as partes em dois arquivos .STL distintos.

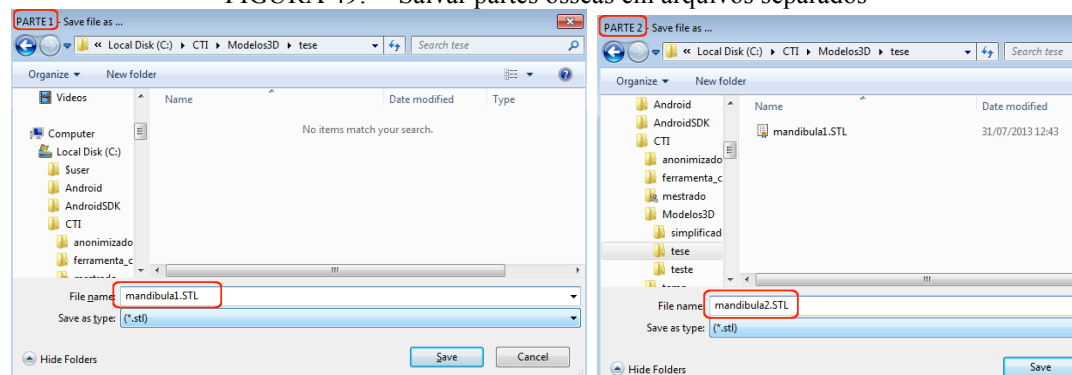


Fonte: Acervo Próprio

Para salvar as partes separadas em arquivos distintos, o especialista deve utilizar a opção Salvar Corte disponível no menu Arquivo conforme Figura 48.

Utilizando esse comando, o sistema Protótipo VTO – UFU – Visualização do Tratamento Ortognático, apresenta duas caixas de dialogo para o médico salvar as duas partes de forma separadas, conforme a Figuras 49.

FIGURA 49: Salvar partes ósseas em arquivos separados

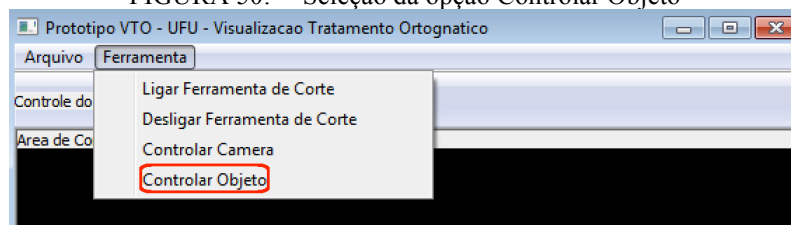


Fonte: Acervo Próprio

Após ter gerado dois arquivos contendo cada um a parte do *canvas* referente a parte interna e outro referente a parte externa da caixa de corte, o sistema tem então dois novos objetos que representam as partes ósseas recém segmentadas.

Com as partes salvas, deve-se então recarrega-la de forma a ter a área de trabalho completamente prepara para o especialista proceder com os movimentos individuais em cada osso. Esse movimento individual é possível por meio da opção Controlar Objeto presente no menu Ferramenta. A Figura 50 exibe como ativa-se o controle sobre objetos, e na sequencia um exemplo da movimentação das partes ósseas.

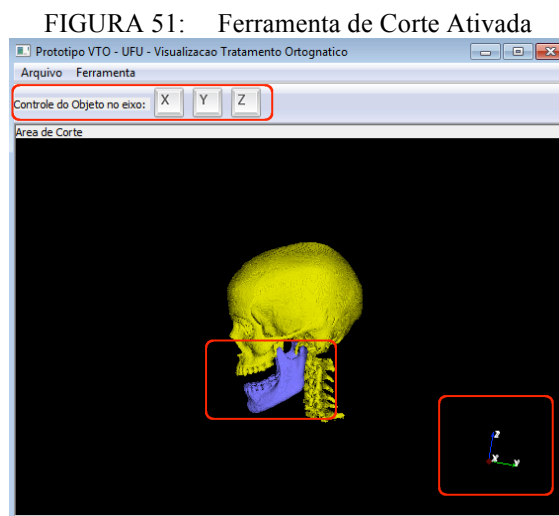
FIGURA 50: Seleção da opção Controlar Objeto



Fonte: Acervo Próprio

Uma vez que o controle de objetos foi ativado, e a fim de evitar a incorreta movimentação das partes nos três eixos ao mesmo tempo de forma pouco ordenada, visto que o sistema por meio da sua área de trabalho pode trazer a sensação da movimentação para um determinado eixo, quando na verdade o objeto poderá estar movendo-se por um eixo não desejado naquele momento, foi inserido no sistema um guia de movimentação para cada eixo,

conforme Figura 51, de forma independente, fixando o movimento da parte somente no eixo designado através da seleção do usuário. Tendo o eixo selecionado e fixado, pode-se livremente mover a parte pelo eixo em qualquer sentido.

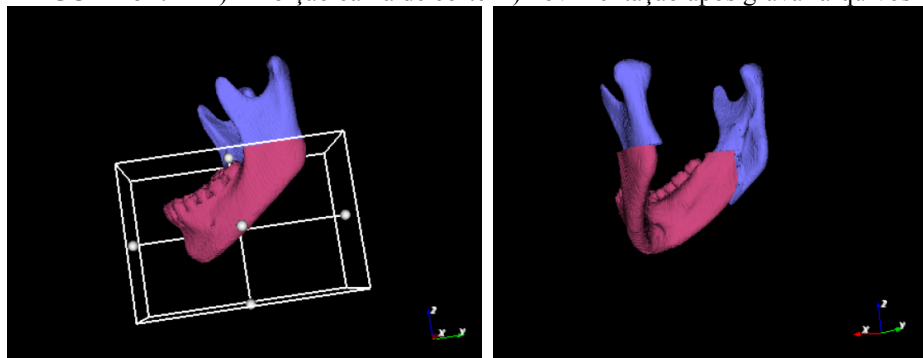


Fonte: Acervo Próprio

A Figura 51 apresenta os botões de seleção que permite a seleção exclusiva de um eixo ante do movimento.

Por fim, visto que o especialista realizou o movimento, é possível salvar novamente a área cortada, assumindo sua nova posição, como apresentado na Figura 52A. Nessa imagem, percebe-se que um pequeno pedaço da mandíbula foi segmentado e movimentado no eixo Y, de acordo com a seleção do eixo efetuada pelo usuário.

FIGURA 52: A) Exibição caixa de corte B) movimentação após gravar arquivos

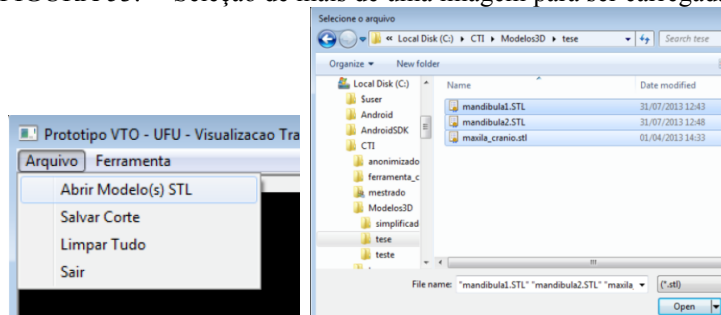


Fonte: Acervo Próprio

Como consequência da segmentação e movimento (Figura 52A), observa-se pela Figura 52B, que houve a separação de mandíbula em duas partes. O especialista pode,

portanto decidir trabalhar com todo o conjunto ósseo, ou seja, todo o crânio e maxila, ou também com partes separadas. Essa escolha e decisão são garantidas ao especialista com base em sua experiência, no procedimento e preferência. A opção por trabalhar com partes separadas pode implicar em uma maior velocidade no uso da ferramenta, visto que há um conjunto menor de polígonos disponíveis na área de trabalho.

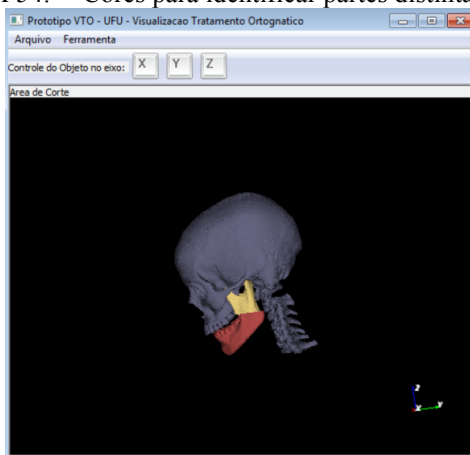
FIGURA 53: Seleção de mais de uma imagem para ser carregada.



Fonte: Acervo Próprio

A Figura 53 apresenta o sistema carregando 3 partes distintas na mesma cena. Mesmo que o especialista decida trabalhar com somente uma parte óssea por vez na área de trabalho, nada o impede de em um momento posterior incluir na mesma cena outras partes ósseas para que tenha uma melhor compreensão da análise e simulação em andamento.

FIGURA 54: Cores para identificar partes distintas na cena

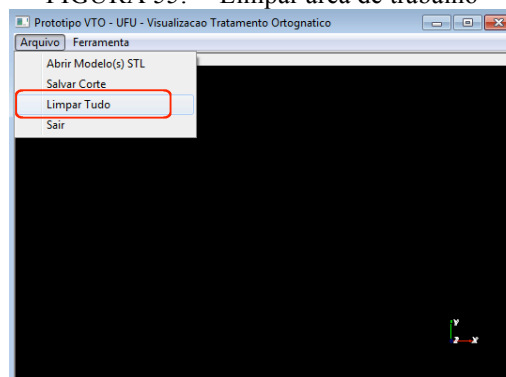


Fonte: Acervo Próprio

Nesse momento os 3 pedaços são carregados área de trabalho, muito embora continuem sendo partes separadas e passíveis de serem manipuladas de forma independente por meio do controle e seleção de objeto. Na Figura 54 é possível compreender isso observando as diferentes cores para cada parte separada e carregada na área de trabalho.

É importante destacar que para o profissional simular a boa oclusão, obrigatoriamente se faz necessário à arcada dentária superior e inferior com ao menos um conjunto de referência craniana. É essencial que a imagem tomográfica obtida traga a parte da mandíbula, está fixa na caixa craniana através do conjunto da ATM.

FIGURA 55: Limpar área de trabalho



Fonte: Acervo Próprio

Por fim, para limpar a área de trabalho (Figura 55), a fim de iniciar uma nova análise e simulação para a cirurgia, o usuário pode utilizar a função para limpar toda a área de trabalho. Isso é realizado por meio da função Limpar Tudo no menu Arquivo.

Como observado nesse capítulo, à ferramenta Protótipo VTO – UFU – Visualização do Tratamento Ortognático, busca trazer ao médico um ambiente de trabalho simples e bastante objetivo na preparação para uma cirurgia com base na sua própria experiência, evitando desviar sua atenção com atividades e tarefas burocráticas como oferecidas em outros softwares com funções similares.

A partir do processo descrito, é possível realizar cortes em qualquer imagem médica adquirida no padrão DICOM conforme apresentado no capítulo 2, a proposta da dissertação como descrita no capítulo 1 é atender a profissionais que tratam as deformidades faciais causadas por maloclusão. Considerando o que foi visto sobre o Protótipo VTO – UFU – Visualização do Tratamento Ortognático nesse capítulo, na sequência o autor aborda detalhes técnicos e de tecnologia aplicados no sistema apresentado nesse capítulo.

CAPÍTULO 5

5. DETALHES DE IMPLEMENTAÇÃO

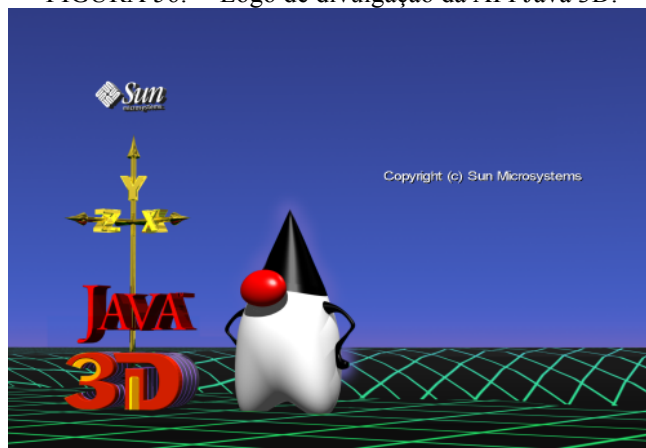
Considerando o exposto nos capítulos anteriores neste capítulo é apresentado o detalhamento do sistema proposto, Protótipo VTO – UFU – Visualização do Tratamento Ortognático, descrevendo as tecnologias que foram utilizadas na sua implementação, além das tentativas em utilizar outras tecnologias mais conhecidas, o que posteriormente provou-se não atender as necessidades da proposta.

5.1 Java 3D

Por ser amplamente utilizada, inicialmente foi escolhido o Java como plataforma e linguagem base para o desenvolvimento do sistema.

Por padrão o Java possui somente uma API para trabalho com gráficos em modo 2D, como parte do pacote `java.awt`, entretanto devido a natureza do sistema proposto, em manipular e promover cortes em imagens 3D, a versão padrão do Java não seria suficiente, motivo esse para ampliar a pesquisa para uma API (biblioteca) que oferecesse suporte a gráficos em 3D (Brown,1999).

FIGURA 56: Logo de divulgação da API Java 3D.



Fonte: Sun Microsystem, 2012

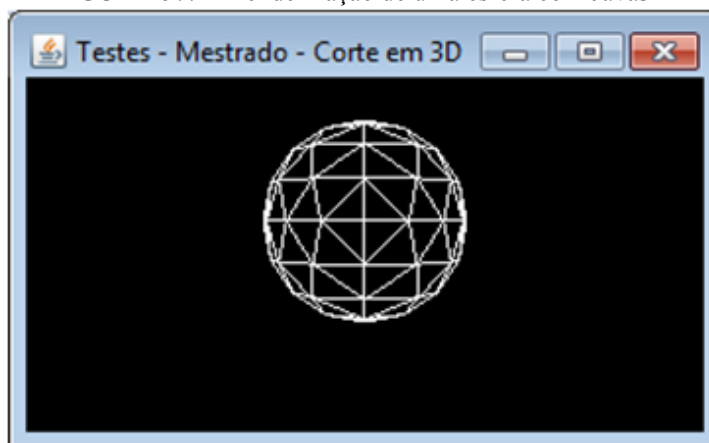
O caminho natural foi iniciar a pesquisa em torno do API Java 3D, (Figura: 56), uma biblioteca distribuída separadamente, como uma extensão do Java para manipulação de gráficos em 3D.

O Java3D oferece uma API para programas desenvolvidos na linguagem Java. Em sua essência o Java3D é uma interface para tecnologias já existentes como OpenGL e DirectX, o que permite a programas escritos em Java3D terem um bom desempenho. Adicionalmente o Java3D possui a capacidade de trabalhar com os objetos gerados em sistemas de modelagem como o VRML.

Uma programação em 3D por si só, é uma tarefa extremamente árdua considerando a matemática envolvida, bem como técnicas já amplamente exploradas de gráficos em três dimensões (Brown,1999). Utilizando uma biblioteca como o Java3D, soma-se então uma API simples a uma linguagem totalmente orientada a objetos. De forma geral as capacidades conhecidas do Java, como multi plataforma e segurança, somadas a simplicidade da biblioteca Java 3D se firmou como o caminho natural para programadores Java interessados no desenvolvimento de aplicações gráficas 3D.

Uma aplicação típica Java 3D é projetada a partir de um grafo de cena contendo diversos objetos, entre eles luz, comportamento, câmera, etc (Brown,1999).

FIGURA 57: Renderização de uma esfera com Java3D



Fonte: Acervo Próprio

Na Figura 57, é apresentada uma tela onde uma esfera é renderizada utilizando o objeto *Sphere* da biblioteca Java3D (BOUVIER ,2010).

A seguir é apresentado o código (Figura 58), utilizado para a criação da esfera, com comentários inseridos no decorrer do código.

FIGURA 58: Código para gerar uma esfera de polígonos

```
public class Esfera {
    public Esfera() {
```

```

        Frame frame = new Frame("Testes - Mestrado - Corte em 3D");
        GraphicsConfiguration config = SimpleUniverse
            .getPreferredConfiguration();
        Canvas3D canvas = new Canvas3D(config);
        canvas.setSize(300, 300);
        SimpleUniverse universo = new SimpleUniverse(canvas);

        // criacao do grafo

BranchGroup group = new BranchGroup();
        Vector3f vector = new Vector3f(0.0f, 0.0f, 0.0f);
        Transform3D transform = new Transform3D();
        transform.setTranslation(vector);
        TransformGroup transformGroup = new TransformGroup(transform);
        Appearance appearance = new Appearance();
        appearance.setPolygonAttributes(new PolygonAttributes(
            PolygonAttributes.POLYGON_LINE, PolygonAttributes.CULL_BACK,
0.0f));

        // Nesse momento uma instancia da classe Sphere é criada
        // onde o parametro appearance criado acima determina sua aparência
        // ligando os vertices dentro da esfera

        Sphere sphere = new Sphere(0.3f, appearance);
        transformGroup.addChild(sphere);
        group.addChild(transformGroup);
        universo.getViewingPlatform().setNominalViewingTransform();
        universo.addBranchGraph(group);
    }

```

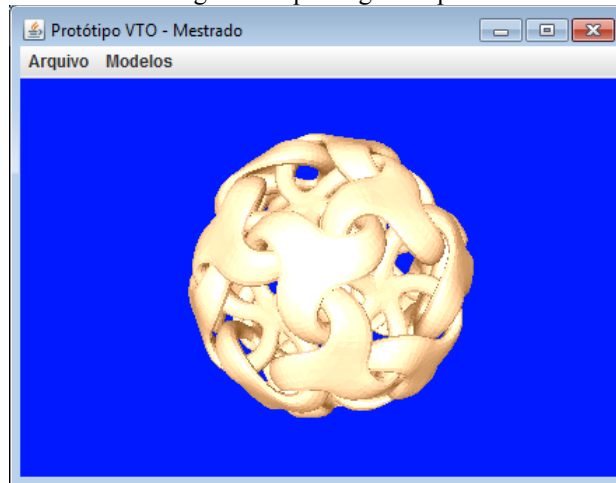
Fonte: acervo próprio

Na sequência da pesquisa, iniciou-se a construção de o Protótipo VTO – UFU – Visualização do Tratamento Ortognático, cujo principal objetivo buscava efetivar um corte em objetos 3D, para então executar a posterior manipulação (clipping) das imagens 3D.

Durante o desenvolvimento do protótipo, após um período de pesquisas e testes, foi possível construir o código fonte para carregar um arquivo STL, e manipular uma imagem 3D por meio de movimentos de rotação e translação, no entanto, ainda não era possível efetivar a

segmentação desse objeto em duas partes independentes, como se fazia necessário para atender um dos principais requisitos da proposta deste trabalho (Figura 59).

FIGURA 59: Imagem complexa gerada por meio do Java3D

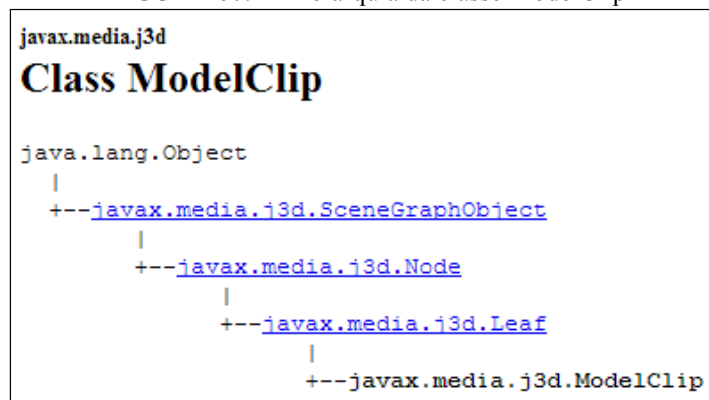


Fonte: Acervo Próprio

Em busca do objetivo maior na manipulação da imagem para esse estudo (segmentação), as pesquisas foram direcionadas para uma classe existente na biblioteca do Java3D que pudesse auxiliar no corte de objetos (clipping) (BOUVIER ,2010) (SUN MICROSYSTEMS, Inc.,2006).

A classe ModelClip foi identificada e testada em busca desse objetivo. Essa classe, parte da API Java3D tem sua hierarquia apresentada na Figura 60.

FIGURA 60: Hierarquia da classe ModelClip



Fonte:Sun Microsystems, Inc.,(2006)

A classe ModelClip realiza cortes em um objeto de forma arbitrária. Seu funcionamento acontece por um corte configurado em até 6 planos definidos arbitrariamente.

O código abaixo (Figura 62) utiliza a classe ModelClip para realizar alguns cortes na esfera apresentada anteriormente na Figura 57.

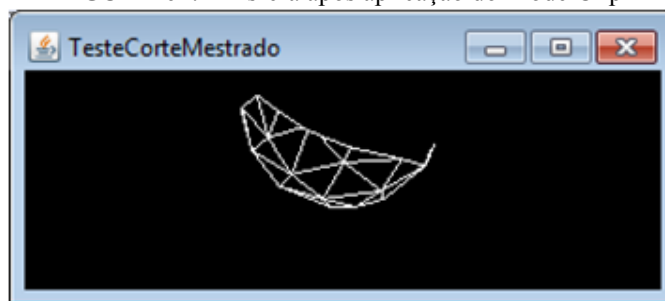
FIGURA 61: Código para uso da função ModelClip

```
//Realiza o corte
ModelClip mc = new ModelClip();
boolean planos[] = { false, false, false, false, false, false };
Vector4d p1 = new Vector4d(0.0, 1.0, 0.0, 0.0);
Vector4d p2 = new Vector4d(1.0, 1.0, 0.0, 0.0);
mc.setEnables(planos);
mc.setPlane(1, p1);
mc.setPlane(2, p2);
mc.setEnable(1, true);
mc.setEnable(2, true);
mc.setInfluencingBounds(bounds);
objTrans.addChild(mc);
```

Fonte: Acervo Próprio

Esse código produz a esfera cortada exibida na Figura 62. Apesar do corte ter sido realizado por meio da classe ModelClip, não foi atingido o principal objetivo do estudo, que era separar imagens por meio do corte, e manter ainda na cena as partes separadas, pois somente assim, as partes então separadas poderiam sofrer de forma independentes os movimentos de rotação e translação, simulando um reposicionamento ósseo realizado em uma procedimento cirúrgico.

FIGURA 62: Esfera após aplicação do ModelClip



Fonte: Acervo Próprio

Observou-se que a classe ModelClip realizou o corte, porém uma das partes segmentadas era simplesmente descartada.

Devido à escassez de alternativas dentro da biblioteca Java3D, além de limitada literatura no assunto, o autor buscou uma alternativa para separar uma imagem (ou objeto) em partes distintas.

Em discussões com profissionais da área, como por exemplo o Sr. Mark McKay da empresa norte americana Kitfox, conclui-se que o Java3D se apresenta como uma boa ferramenta para criação de cenários 3D mais elaborados, porém para técnicas 3D mais complexas ou avançadas, ele não se mostra como uma solução viável.

Existem ainda outras bibliotecas alternativas que poderiam trazer ganhos referentes ao que esta sendo buscado nessa pesquisa como o JOGL (MARK MCKAY,2010).

5.2 Invesalius

É um software desenvolvido para o auxílio ao diagnóstico e planejamento cirúrgico. Capaz de trabalhar com imagens obtidas por meio de equipamentos de tomografia computadorizada, ressonância magnética ou qualquer equipamento que gere arquivo no padrão internacional médico.

Com essas imagens, são criados modelos em 3D correspondentes às estruturas anatômicas dos pacientes. É um software público (Figura 63), e vem sendo difundido em diversas áreas, tais quais medicina, odontologia, veterinária e engenharia. O software foi desenvolvido pelo CTI (Centro de Tecnologia da Informação Renato Archer), uma unidade filiada ao Ministério da Ciência e Tecnologia.

FIGURA 63: *Splash Screen* do InVesalius



Fonte: Software público Divisão 3D, 2013

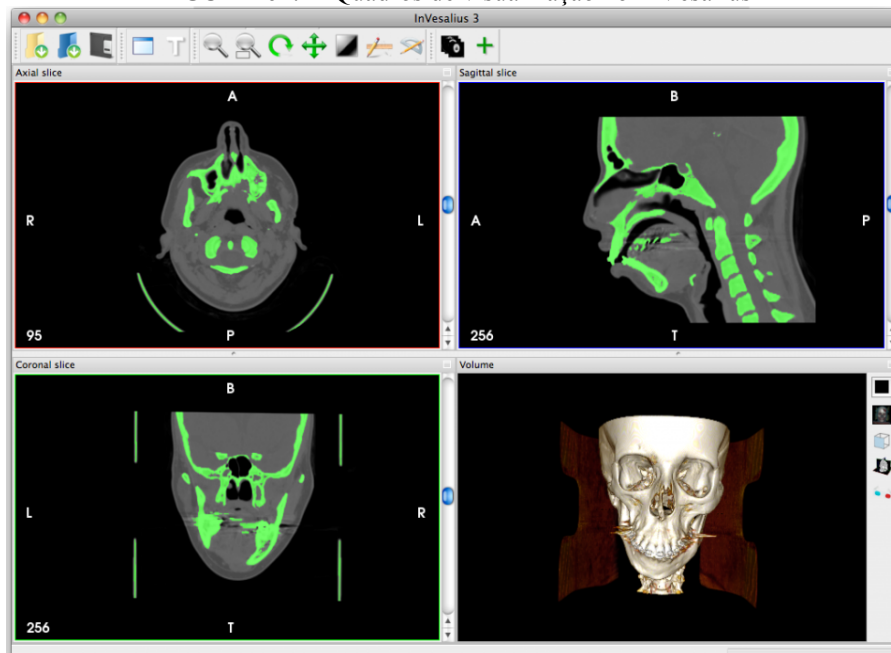
O nome *InVesalius*TM nasceu como uma homenagem ao médico belga Andreas Vesalius (1514-1564), considerado o pai da anatomia moderna. A maior demanda que o

software atende atualmente está no planejamento cirúrgico detalhado e no diagnóstico de patologias, foco no qual foi criado, porém também tem sido utilizado na geração de biomodelos físicos construídos com mecanismos de prototipagem rápida, parafusamento de implantes e geração de próteses. Por ser desenvolvido no Brasil, é na língua portuguesa que se consegue o suporte principal, facilitando assim sua adoção dentro do território nacional.

O *InVesalius*TM trabalha com a memória RAM de forma muito eficiente utilizando o sistema *Map*, o que permite que somente os dados da matriz com três dimensões referentes as imagens convertidas sejam mantidos em memória. Existindo a necessidade em se trabalhar um conjunto de dados diferentes dos quais estejam carregados na memória naquele momento, novos dados são carregados do disco. Essa estratégia permite trabalhar com arquivos de imagens médicas grandes, com muitas fatias, sem que toda a imagem esteja na memória. (AMORIM et al, 2011).

Dentre as ferramentas oferecidas pelo software, pode-se destacar a visualização volumétrica, medições, reconstrução 3D, visualização 2D e 3D, visualização oblíqua, entre outras funcionalidades, conforme exibido na Figura 64.

FIGURA 64: Quadros de visualização no InVesalius

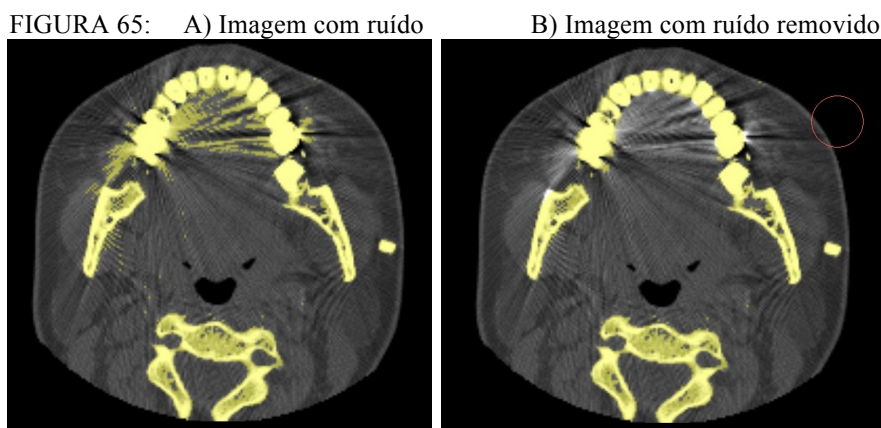


Fonte: Software InVesalius - Guia do Usuário-Amorim,2011

Além da construção 3D por composição de fatias, é possível ainda visualizar uma imagem tomográfica navegando por cada fatia nas orientações axial, sagital e coronal, isso devido à implementação de uma matriz de três dimensões (BARTOLO et al,2008).

O modelo virtual gerado pelo *InVesalius*TM também pode ser utilizado para projetar próteses personalizadas para cada paciente antes do procedimento cirúrgico.

As Figuras 65A e 65B abaixo, apresentam uma situação onde o *InVesalius*TM é utilizado para remover um ruído gerado na captação de um imagem tomográfica.



Fonte: Manual do InVesalius

Fazendo uso de algumas funcionalidades oferecidas pelo *InVesalius*TM, o sistema proposto Protótipo VTO – UFU – Visualização do Tratamento Ortognático, nesse estudo utiliza a conversão de uma imagem adquirida no formato DICOM para o formato STL como já explanado no capítulo 4, além da eliminação de ruídos e separação das imagens não obtidas com a boca semi aberta.

Por meio do estudo da arquitetura desse software (*InVesalius*TM), observou-se como linguagem de programação utilizada o Python. De acordo com Paulo Jorge 2008, Python não seria a escolha natural para uma linguagem que envolvesse processamento de imagens, entretanto é uma linguagem de fácil aprendizagem, com uma sintaxe simples, que pode ser usada para construir sistemas grandes em um período curto de tempo (BARTOLO, 2008).

Além de Python, foi empregado no *InVesalius*TM o uso da biblioteca VTK (Visualization Toolkit) como o motor de processamento, o wxPython como gerenciador de janelas, como também outras bibliotecas para cálculos matemáticos, notoriamente o NumPy. Como base nos estudos envolvendo o *InVesalius*TM, e além do estreito contato com o time de desenvolvimento do mesmo, foi decidido utilizar a mesma linguagem e bibliotecas no desenvolvimento do sistema proposto nesse trabalho.

Nas seções subsequentes desse capítulo serão apresentados mais detalhes dessas tecnologias.

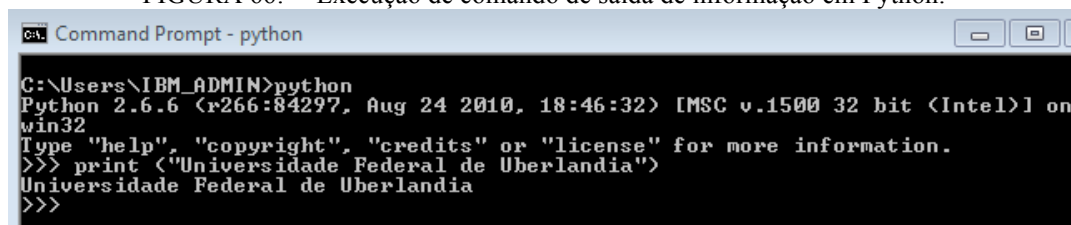
5.3 Python

A linguagem Python é uma linguagem de alto nível, aberta e que pode ser usada para diferentes tarefas computacionais. Desde seu ano de criação em 1990 pelo programador Guido Van Rossum, a adoção do Python tem crescido a largos passos, incluindo grandes institutos como a NASA e o Google.

O crescimento do Python tem ocorrido principalmente pela facilidade no seu aprendizado, e o seu grande poder na manipulação de estrutura de dados. É uma linguagem de natureza interpretada, o que a torna ideal para scripts e desenvolvimento rápido de aplicações (KINSER,2009). Por ter seu código fonte aberto, o próprio interpretador do Python pode ser facilmente extensível com novas funções e tipos de dados. Quando se escreve programas através do Python, um programador não deve se preocupar com detalhes de baixo nível, como manipulação explícita de memória. A orientação a objetos é um dos modelos de programação suportados pelo Python, que oferece também suporte ao tradicional modelo procedural (KINSER,2009).

Uma vez instalado o aplicativo Python no sistema, pode-se por meio do próprio programa interpretador (modo interativo) ou de qualquer editor de textos ASCII, escrever um *script* nessa linguagem. Considerando a simplicidade deste processo é possível observar na Figura 66 a execução de um *script* Python no modo interativo desenvolvido com função mais básica de saída de dados, o comando *print*.

FIGURA 66: Execução de comando de saída de informação em Python.



```

C:\Users\IBM_ADMIN>python
Python 2.6.6 <r266:84297, Aug 24 2010, 18:46:32> [MSC v.1500 32 bit <Intel>] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print <"Universidade Federal de Uberlandia">
Universidade Federal de Uberlandia
>>>

```

Fonte: Acervo Próprio

Muito embora a biblioteca padrão do Python seja grande e poderosa, existe a possibilidade de se utilizar bibliotecas adicionais de forma a complementar o rico conjunto de funcionalidades oferecido por meio da biblioteca padrão. Atualmente o Python está disponível em diversas plataformas entre elas Windows, OSX e Linux.

Sendo o *Python* uma linguagem orientada a objetos, o trecho de código apresenta um *script* escrito em *Python* para a criação de uma classe Aluno no modo interativo. Segundo

(MELO, 2002), uma classe é a representação de um conjunto de objetos que compartilham a mesma estrutura de atributos, operações e relacionamentos dentro de um mesmo contexto (semântica). No código apresentado na Figura 67, uma instancia da classe Aluno é criada e alguns dados de exemplo são fornecidos como parâmetros no método construtor.

FIGURA 67: Exemplificação da “classe Aluno” no modo interativo

```
>>> class Aluno:
...     def __init__ (self, nome, idade, sexo):
...         self.nome = nome
...         self.idade = idade
...         self.sexo = sexo
>>> Giu = Aluno("Giuliano Moraes", 37, "m")
>>> print Giu
<__main__.Aluno instance at 0x80a94cd>
>>> print Giu.idade
37
```

Fonte: Acervo Próprio

Na Figura 68 uma lista (*array*) chamada *alunos* é criada com 3 elementos, e na sequência esses elementos são ordenados pela função *sorted*. Nesse exemplo, a lista é ordenada pelo atributo *idade*. Uma consulta mais expansiva acessando o link <http://docs.python.org/2/howto/sorting.html> na documentação do Python traz mais detalhes e exemplos de como utilizar tal comando de diferentes formas.

FIGURA 68: Código para ordenação em Python

```
>>> alunos = [
    ('joao', 'A', 35),
    ('janaina', 'B', 22),
    ('davi', 'B', 11),
]
>>> sorted(alunos, key=lambda aluno: aluno[2])
[('davi', 'B', 11), ('janaina', 'B', 22), ('joao', 'A', 35)]
```

Fonte: Acervo Próprio

Entre todos os pontos sobre Python apresentados até aqui, ressalta-se também a grande adoção da linguagem pela comunidade técnica, bem como farta literatura e documentação sobre o tema.

O suporte da comunidade (Figura 69), e a grande quantidade de documentação formal, além da influencia da arquitetura do *InVesalius*TM, fortaleceram a decisão da escolha do Python como linguagem a ser utilizada no desenvolvimento do sistema proposto nesse estudo.

FIGURA 69: Ranking de Linguagens de Programação

Programming Language	Position Nov 2012	Position Nov 2007	Position Nov 1997	Position Nov 1987
C	1	2	1	1
Java	2	1	4	-
Objective-C	3	38	-	-
C++	4	3	2	6
C#	5	7	-	-
PHP	6	5	-	-
(Visual) Basic	7	4	5	5
Python	8	6	29	-
Transact-SQL	9	45	-	-
JavaScript	10	10	14	-
Lisp	15	21	17	3
COBOL	19	17	3	12
Ada	22	23	18	2

Fonte: TIOBE (2013)

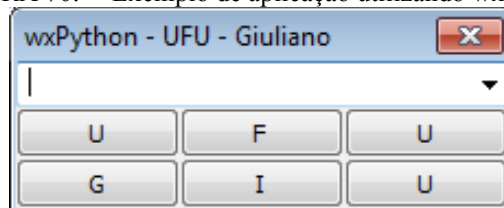
De acordo com a empresa Tiobe Software, direcionada a pesquisa e métricas de linguagem de programação, a linguagem Python ocupa atualmente a 8 posição no *ranking*, sendo que em 1997, a mesma ocupava a 29, conforme dados apresentados na Figura 64.

5.3.1 wxPython

O wxPython é um *wrapper* (empacotamento) para uma plataforma de GUI (Graphical User Interface), com base no wxWidgets para a linguagem de programação Python. Esse é um tipo de biblioteca também conhecido por *Widget Toolkit*. Ser multiplataforma foi fundamental

para o sucesso do wxPython, pois dessa forma uma programa em Python que utiliza essa biblioteca pode ser compilado e executado em diversas plataformas (RAPPIN, 2006).

FIGURA 70: Exemplo de aplicação utilizando wxPython



Fonte: Acervo Próprio

Conforme indicado pelo próprio nome, o wxPython permite que programadores Python criem interfaces gráficas com o usuário de forma simples e robusta. Na Figura 70 é apresentado uma janela gráfica simples criada por meio do wxPython usando componentes gráficos comuns no desenvolvimento de interfaces gráficas, tais como botão e campo de texto.

FIGURA 71: Código utilizando o wxPython, criação inclusive da Janela Principal

```
#!/usr/bin/env python

import wx

class TesteGUI(wx.Dialog):
    def __init__(self):
        __titulo = 'wxPython - UFU - Giuliano'
        __wx.Dialog.__init__(self, None, -1, titulo)
        __sizer = wx.BoxSizer(wx.VERTICAL)
        __self.display = wx.ComboBox(self, -1)
        __sizer.Add(self.display, 0, wx.EXPAND)

        __gsizer = wx.GridSizer(2, 3)
        __for row in (('U', 'F', 'U'),
                      ('G', 'I', 'U')):
            __for label in row:
                __b = wx.Button(self, -1, label)
                __gsizer.Add(b)
                __self.Bind(wx.EVT_BUTTON, self.OnButton, b)
            __sizer.Add(gsizer, 1, wx.EXPAND)

        __self.SetSizer(sizer)
```

```

__sizer.Fit(self)

def OnButton(self, evt):
    '''Handle button click event'''

if __name__ == '__main__':
    # Run the application
    _app = wx.PySimpleApp()
    _dlg = TesteGUI()
    _dlg.ShowModal()
    _dlg.Destroy()

```

Fonte: Acervo Próprio

No código apresentado na Figura 71, o wxPython é o responsável pelos componentes visuais, incluindo a própria janela, representada aqui pela classe wx.Dialog, bem como seus botões e uma caixa combo, representados por wx.Button e wx.ComboBox respectivamente.

5.3.2 VTK

Influenciado pelo modelo de arquitetura usado no desenvolvimento do *InVesalius*TM, a solução proposta faz uso do VTK (*Visualization Toolkit*). O VTK é uma biblioteca desenvolvida em C++, e que possui um conjunto de funcionalidades para o processamento e visualização de imagens 3D e superfícies. O *toolkit* VTK suporta uma gama de diferentes algoritmos de visualização, incluindo: escalar, vetorial, tensor, textura e métodos volumétricos (SCHROEDER, MARTIN, e LORENSEN,1998).

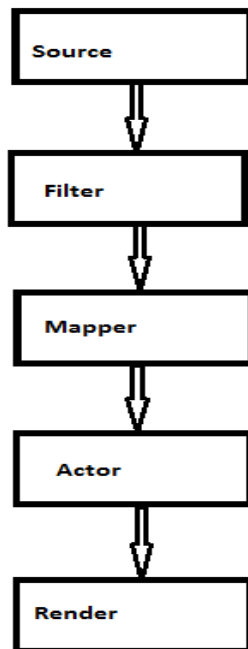
Sua implementação também suporta diferentes técnicas de modelagem tais como redução de polígonos, triangulação de Delaunay, entre outras. Originalmente desenvolvido em C++, o VTK possui também interfaces para TCL, Java e Python (PAPADEMETRIS E JOSHI,2009). Embora seja uma biblioteca de código aberto, o VTK possui um modelo de comercialização que inclui suporte oferecido pela empresa KitWare. Devido adoção de Python na criação do sistema proposto, foi utilizado a interface do VTK para Python (SCHROEDER, MARTIN, E LORENSEN,1998).

O ponto chave para entender o VTK é a compreensão da sua estrutura de classes dentro da metodologia da orientação a objetos, e a sua arquitetura de *pipeline*, similar a existente em outros *framework* ou sistemas operacionais como o Linux, onde múltiplos

elementos são sequencialmente agrupados (ou encadeados) em busca de um resultado de uma tarefa maior e mais complexas.

Uma arquitetura típica de *pipeline* no VTK é apresentada na Figura 72

FIGURA 72: Diagrama de Blocos do pipeline do VTK



Fonte: Acervo próprio

Os principais componentes do VTK são:

Source: Determina a origem inicial dos dados. É vinculado a uma classe do VTK que carrega os dados, como por exemplo a classe `vtkJPEGReader`, que lê um arquivo no formato jpeg e o utiliza como entrada de informação no *pipeline* (PAPADEMETRIS E JOSHI,2009).

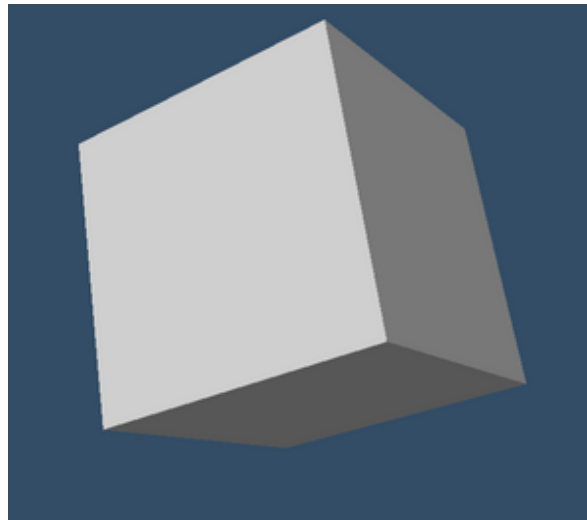
Filter: São classes do VTK que aplicam alguma modificação sobre o conjunto de dados oriundos do componente anterior na cadeia. A classe `vtkImageGaussianSmooth` é um exemplo de filtro que realiza função gaussiana para reduzir o nível de ruído na imagem.

Mapper: São classes do VTK cuja tarefa é converter um conjunto de dados em algum objeto tangível e concreto dentro do VTK para posterior transformação. A super classe é `vtkMapper`, e algumas de suas sub classes são `vtkPolyDataMapper` e `vtkGraphMapper` (PAPADEMETRIS E JOSHI,2009).

Actor: Atores no VTK são classes permitem alterar propriedades dos objetos gráficos gerados por meio do `vtkMapper`. É por meio de um `vtkActor` que pode-se trabalhar no nível de detalhe do objeto, como transparência, cor, etc. Este é o ultimo passo de preparação para um objeto gráfico ser renderizado na cena.

Render: A classe `vtkRender` é quem controla o processo de renderização de objetos. Vinculados ao renderizador (Render) estão a câmera, como também elementos de iluminação. É durante a renderização que ocorre a conversão do objeto gráfico por meio da definição da sua geometria em uma imagem formada na cena. A classe `vtkRender` também trabalha com as coordenadas dos objetos para a correta renderização na cena (PAPADEMETRIS E JOSHI,2009).

FIGURA 73: Cubo renderizado usando o VTK



Fonte: Acervo Próprio

O cubo exibido na Figura 73 é resultado do código escrito em Python utilizando como suporte gráfico o VTK. O código fonte detalhado na Figura 74, onde todos os componentes descritos anteriormente na definição de *pipeline* com a sua devida aplicação em uma sequência lógica e necessária a sua apresentação. (SCHROEDER, MARTIN, E LORENSEN,1998).

FIGURA 74: Código do cubo em VTK usando técnica de *pipeline*

```
import vtk
```

```

P0 = [0.0, 0.0, 0.0]
P1 = [1.0, 0.0, 0.0]
P2 = [1.0, 1.0, 0.0]
P3 = [0.0, 1.0, 0.0]
P4 = [0.0, 0.0, 1.0]
P5 = [1.0, 0.0, 1.0]
P6 = [1.0, 1.0, 1.0]
P7 = [0.0, 1.0, 1.0]

# Create the points
points = vtk.vtkPoints()
points.InsertNextPoint(P0)
points.InsertNextPoint(P1)
points.InsertNextPoint(P2)
points.InsertNextPoint(P3)
points.InsertNextPoint(P4)
points.InsertNextPoint(P5)
points.InsertNextPoint(P6)
points.InsertNextPoint(P7)

# Create a hexahedron from the points
hex = vtk.vtkHexahedron()
hex.GetPointIds().SetId(0,0)
hex.GetPointIds().SetId(1,1)
hex.GetPointIds().SetId(2,2)
hex.GetPointIds().SetId(3,3)
hex.GetPointIds().SetId(4,4)
hex.GetPointIds().SetId(5,5)
hex.GetPointIds().SetId(6,6)
hex.GetPointIds().SetId(7,7)

# Add the hexahedron to a cell array
hexs = vtk.vtkCellArray()
hexs.InsertNextCell(hex)

# Add the points and hexahedron to an unstructured grid
uGrid = vtk.vtkUnstructuredGrid()
uGrid.SetPoints(points)
uGrid.InsertNextCell(hex.GetCellType(), hex.GetPointIds())

```

```

# Visualize
mapper = vtk.vtkDataSetMapper()
mapper.SetInput(uGrid)
actor = vtk.vtkActor()
actor.SetMapper(mapper)

renderer = vtk.vtkRenderer()
renderWindow = vtk.vtkRenderWindow()
renderWindow.AddRenderer(renderer)
renderWindowInteractor = vtk.vtkRenderWindowInteractor()
renderWindowInteractor.SetRenderWindow(renderWindow)

renderer.AddActor(actor)
renderer.SetBackground(.2, .3, .4)

renderWindow.Render()
renderWindowInteractor.Start()

```

Fonte: Acervo Próprio

Observa-se no código da Figura 74 a aplicação do pipeline, por meio do *source*, nesse caso o *vtkPoints*, seguido pelo *vtkDataMapper*, *vtkActor* e *vtkRender*, todos encadeados. Como complemento as facilidades e benefícios do VTK apresentados até aqui, o VTK possui vasta literatura disponível, e uma forte comunidade de usuários.

Cabe ao autor aqui destacar também que diversos softwares opensource e comerciais fazem o uso dessa biblioteca, entre eles 3DSlicer e Mayavi.

5.4 Detalhes de Implementação do Sistema Proposto

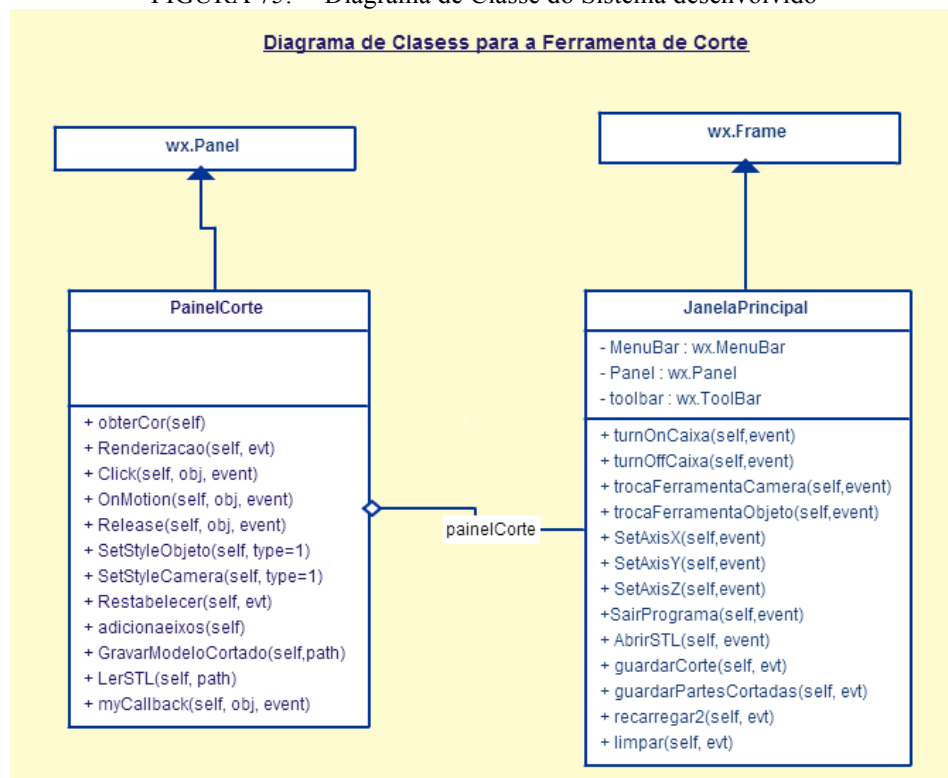
Nesta seção será descrito com detalhes o código fonte, bem como sua explicação para o sistema proposto desenvolvido nesse estudo. Destaca-se o por todo o código desenvolvido em Python o uso das bibliotecas wxPython e VTK apresentadas nas seções anteriores.

5.4.1 Diagrama de Classes

Como citado anteriormente, o Python suporta programação dentro do paradigma de orientação a objetos. O diagrama de classes apresenta as principais classes do sistema

desenvolvido, (Figura 75). As classes `PainelCorte` e `JanelaPrincipal` que serão explicadas a seguir nesse mesmo capítulo.

FIGURA 75: Diagrama de Classe do Sistema desenvolvido



Fonte: Acervo Próprio

O ponto de partida do sistema passa pelo método *main* da classe `PainelCorte`, onde uma instância da classe `JanelaPrincipal` é criada.

FIGURA 76: Código da Janela Principal

```

if __name__ == '__main__':
    app = wx.App(0)
    JanelaPrincipal(None, -1, 'Prototipo VTO - UFU - Visualizacao Tratamento
    Ortognatico')
    app.MainLoop()
  
```

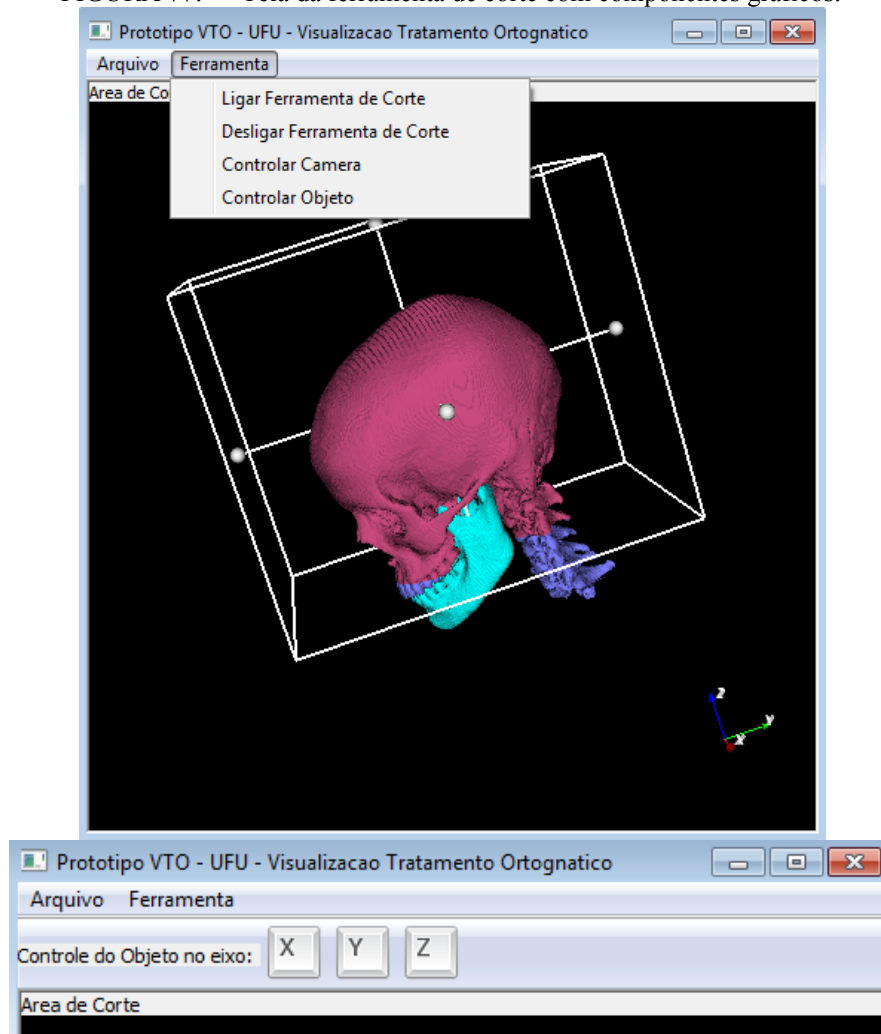
Fonte: Acervo Próprio

No momento da criação desse objeto, o título da janela é passado como um parâmetro, conforme apresentado na Figura 76, e na sequência o *loop* principal é executado.

5.4.2 Interface Gráfica

Para suportar a área de trabalho gráfica (*canvas*), bem como forma para o usuário interagir com a ferramenta (GUI – Graphical User Interface), foi definido que a criação desses elementos da interface seriam feitos por meio da biblioteca wxPython apresentada anteriormente. Na Figura 77 é possível ver alguns desses elementos gráficos, tais como a janela, menus de seleção, etc.

FIGURA 77: Tela da ferramenta de corte com componentes gráficos.



Fonte: Acervo Próprio

Fazendo uso do princípio da herança em programação orientada a objetos, a classe *JanelaPrincipal* estende a classe *wx.Frame*. Essa é a classe responsável por toda a GUI (*Graphical User Interface*), o que inclui o *frame*, menus, botões, seleção, etc.

5.4.3 Classe JanelaPrincipal

O trecho de código apresentado na Figura 78 exibe a definição da classe JanelaPrincipal. É possível identificar que a JanelaPrincipal é uma subclasse de wx.Frame

FIGURA 78: Código da definição da classe JanelaPrincipal

```
class JanelaPrincipal(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, size=(1300, 700))
```

Fonte: Acervo Próprio

A classe wx.Frame fornece a estrutura básica de uma janela, sendo essa classe uma subclasse de Window, a qual por sua vez possui diversos métodos e atributos para o gerenciamento do ciclo de vida de uma janela no ambiente gráfico.

5.4.4 Elementos Visuais

Nesta seção são descritos e exemplificados por meio de código a composição do ambiente gráfico de janelas.

Para essa exemplificação, utiliza-se detalhes da classe JanelaPrincipal que é uma subclasse de wx.Frame, onde são definidos vários controles visuais, tais como:

- a) Menu - O menu acoplado (Figura 79) à JanelaPrincipal é um instancia da classe wx.Menu().

O método Append da classe wx.Menu() é utilizado para acrescentar elementos no menu e sua respectiva descrição textual.

FIGURA 79: Criação do menu na JanelaPrincipal

```
menu=wx.Menu()
AbrirModelo=menu.Append(-1, "&Abrir Modelo(s) STL")
guardar4=menu.Append(-1, "&Salvar Corte")
limpar=menu.Append(-1, "&Limpar Tudo")
sair=menu.Append(-1, "&Sair")
MenuBar.Append(menu, "Arquivo")
```

```

menuFerramenta=wx.Menu()
ligarcaixa=menuFerramenta.Append(-1, "&Ligar Ferramenta de Corte")
desligarcaixa=menuFerramenta.Append(-1, "&Desligar Ferramenta de Corte")
ferramentaCamera=menuFerramenta.Append(-1, "&Controlar Camera")
ferramentaObjeto=menuFerramenta.Append(-1, "&Controlar Objeto")
MenuBar.Append(menuFerramenta, "Ferramenta")

self.SetMenuBar(MenuBar)

```

Fonte: Acervo Próprio

- b) Barra de Ferramentas – Da mesma forma que o menu foi adicionado à Janela principal, a Figura 80 exibe como uma barra de ferramentas (ou *tool bar*) é inserida na janela.

FIGURA 80: Barra de Ferramentas

```

self.toolbar1 = wx.ToolBar(self, id=-1, style=wx.TB_HORIZONTAL |
wx.NO_BORDER | wx.TB_FLAT | wx.TB_TEXT)

texto = wx.StaticText(self.toolbar1, -1, 'Controle do Objeto no eixo: ',
(1, 1))
self.toolbar1.AddControl(texto)

self.toolbar1.AddSimpleTool(808, wx.Image('x.png',
wx.BITMAP_TYPE_PNG).ConvertToBitmap(), 'Eixo X', '')
self.toolbar1.AddSimpleTool(809, wx.Image('y.png',
wx.BITMAP_TYPE_PNG).ConvertToBitmap(), 'Eixo Y', '')
self.toolbar1.AddSimpleTool(810, wx.Image('z.png',
wx.BITMAP_TYPE_PNG).ConvertToBitmap(), 'Eixo Z', '')

```

Fonte: Acervo Próprio

A classe `wx.ToolBar` é usada para definir a barra de ferramentas. Nota-se que logo após o objeto da classe `wx.ToolBar` ser criado, tendo sua orientação horizontal já definida no momento da criação, uma série de botões e suas respectivas imagens são adicionados, compondo dessa forma a barra de ferramentas utilizado para o protótipo.

- c) Manipuladores de Eventos – Mesmo tendo os elementos visuais criados, ainda se faz necessário vincular ações individuais para cada item do menu, ou botão da barra de ferramentas em resposta a ações do usuário. O método *Bind* da classe *JanelaPrincipal*, herdado da classe *wx.Frame* é utilizado para criar esse vínculo. Por meio desse método associadas as funções, chamadas aqui de manipuladores de evento, as quais serão acionados pelo sistema de janelas em respostas a ações de comportamento do usuário, tais como um evento de click em um botão ou item do menu conforme apresentado na Figura 81.

FIGURA 81: Manipulação de Eventos

```
self.Bind(wx.EVT_MENU, self.SairPrograma, sair)
self.Bind(wx.EVT_MENU, self.AbrirSTL, AbrirModelo)
self.Bind(wx.EVT_MENU, self.guardarPartesCortadas, guardar4)
self.Bind(wx.EVT_MENU, self.limpar, limpar)
self.Bind(wx.EVT_MENU, self.turnOnCaixa, ligarcaixa)
self.Bind(wx.EVT_MENU, self.turnOffCaixa, desligarcaixa)
self.Bind(wx.EVT_MENU, self.trocaFerramentaCamera, ferramentaCamera)
self.Bind(wx.EVT_MENU, self.trocaFerramentaObjeto, ferramentaObjeto)
```

Fonte: Acervo Próprio

Alguns desses métodos manipuladores de eventos são explicados e detalhados a seguir.

5.4.5 Métodos Manipuladores de Eventos

Nesta seção foram destacados alguns dos principais métodos manipuladores dos eventos da classe *JanelaPrincipal*, que atuam em resposta a ações do usuário.

- a) *AbrirSTL* – Método responsável por permitir que o usuário selecione um ou mais arquivos no formato STL que serão carregados na área de trabalho. Utiliza-se para essa tarefa a classe *wx.FileDialog*. Essa classe é responsável por apresentar uma caixa de dialogo padrão do sistema operacional, permitindo que o usuário selecione um ou mais arquivos no formato STL presentes no sistema de arquivos. O trecho de código onde é possível observar que após a seleção e confirmação da caixa de dialogo, os arquivos STL são carregados na área de trabalho gráfica, nesse texto também tratada

por *canvas*. Essa função é realizada por meio da chamada para o método `LerSTL` que se encontra no código da Figura 82, com sua chamada acontecendo para cada arquivo selecionado na caixa de diálogo. Essa chamada acontece dentro do método `AbrirSTL`, onde o arquivo representa a imagem selecionada na caixa de diálogo.

FIGURA 82: Código para Abrir arquivo no formato STL

```
def AbrirSTL(self, event):
    dlg_abrir=wx.FileDialog(self, message="Selecione o arquivo",
        defaultDir=self.currentDirectory,
        defaultFile="",
        wildcard=wildcard,
        style=wx.OPEN | wx.MULTIPLE | wx.CHANGE_DIR
    )

    if dlg_abrir.ShowModal()==wx.ID_OK:
        paths=dlg_abrir.GetPaths()

        for arquivo in paths:
            self.PainelCorte.LerSTL(arquivo)
            #self.PainelCorte.LerSTL(paths[0])
        dlg_abrir.Destroy()
```

Fonte: Acervo Próprio

- b) GuardaCorte – De forma análoga ao manipulador detalhado na seção anterior, o método `GuardaCorte` exibido na Figura 83, apresenta a caixa de diálogo que permite ao usuário gravar novos arquivos referentes as partes segmentadas por meio da ferramenta de corte.

FIGURA 83: Manipulador de evento responsável por gravar as partes cortadas

```
def guardarCorte(self, evt):
    dlg = wx.FileDialog(
        self, message="Save file as ...",
        defaultDir=self.currentDirectory,
        defaultFile="", wildcard=wildcard, style=wx.SAVE
    )
```

```

if dlg.ShowModal() == wx.ID_OK:
    path = dlg.GetPath()
    self.PainelCorte.GravarModeloCortado(path)
dlg.Destroy()

```

Fonte: Acervo Próprio

O método que efetivamente faz a gravação da(s) parte(s) no sistema de arquivos, é o método GravarModeloCortado. O caminho e nome do arquivo definido pelo usuário na caixa de dialogo são passados como parâmetro para esse método.

- c) turnOnCaixa / turnOffCaixa – O sistema em algumas situações pode permitir que o usuário desligue a ferramenta (ou caixa) de corte. O que de fato ocorre é que a caixa de corte é removida de área visual. Isso é interessante para o usuário em situações onde deseja por meio do mouse manipular as partes ósseas na tela, sem que a ferramenta de corte receba o evento de click no lugar da parte óssea, uma vez que pode estar posicionada em primeiro plano. O código da Figura 84 apresenta o código para isso.

FIGURA 84: Manipulador de evento para ligar e desligar a ferramenta de corte

```

def turnOnCaixa(self,event):
    self.PainelCorte.BoxWidget.On()
    self.Bind(wx.EVT_IDLE, self.PainelCorte.Restabelecer(event)) #para
funcionar o editor de corte

def turnOffCaixa(self,event):
    self.PainelCorte.BoxWidget.Off()
    self.Bind(wx.EVT_IDLE, None) #para funcionar o editor de corte

```

Fonte: Acervo Próprio

Na Figura 85 são apresentados os métodos On() e Off() sobre o objeto BoxWidget, este instancia da classe vtk.vtkBoxWidget2() usada nesse trabalho como a ferramenta de seleção e corte.

5.4.6 Classe PaineiCorte

A classe PaineiCorte tem por principal objetivo, criar a área de trabalho onde o usuário irá interagir com as partes ósseas, tal interação permite movimentação do osso e da câmera, bem como o corte (osteotomia) das partes ósseas.

O código dessa classe é apresentado a seguir na Figura 85.

FIGURA 85: Código da Classe Painei de Corte

```
class PaineiCorte(wx.Panel):

    """
    Painei onde os cortes são feitos
    """

    def __init__(self, parent, id, style):
        wx.Panel.__init__(self, parent, id, style=style)
        self.renderer = vtk.vtkRenderer()
        self.Interactor = wxVTKRenderWindowInteractor(self, -1, size =
self.GetSize())
        self.Interactor.GetRenderWindow().AddRenderer(self.renderer)
        self.Interactor.Render()
        self.currentSelectedAxis = "x"

        # moviemnta a camera
        istyle = vtk.vtkInteractorStyleTrackballCamera()
        self.Interactor.SetInteractorStyle(istyle)

        # inicializar o plano
        self.VerPlano= False #
        self.SelectActor=None
        hbox=wx.BoxSizer(wx.VERTICAL)
        hbox.Add(wx.StaticText(self, -1, 'Area de Corte'))
        hbox.Add(self.Interactor, 1, wx.EXPAND)
        self.SetSizer(hbox)
        self.adicionaeixos()
        self.Picker=vtk.vtkPropPicker()
        pub.subscribe(self.Renderizacao, 'Renderizar')
```

Fonte: Acervo Próprio

Inicialmente a declaração da classe apresenta uma relação de herança com a `wx.Panel`, nesse caso sua super classe.

Essa classe é de grande importância para o sistema, pois é por meio da mesma que a interação do usuário com as partes ósseas no ambiente gráfico ocorre. É nessa classe que o renderizador é criado (`vtk.vtkRenderer`), como também a legenda representativa dos eixos na área de trabalho gráfico.

5.4.7 Métodos PainelCorte

Sendo a classe `PainelCorte` responsável por toda interação do usuário com o área de trabalho (*canvas*), destaca-se o uso de uma classe importada da biblioteca VTK chamada `vtk.vtkBoxWidget2()`. Essa classe é utilizada como a caixa de corte, capaz de separar a imagem em diferentes partes, podendo assim realizar o procedimento de osteotomia, para então manipular cada parte óssea de forma independente, como apresentado no capítulo 3.

- a) `LerSTL` – Uma vez que o arquivo STL foi selecionado pela `JanelaPrincipal`, a operação é delegada para o método `LerSTL` (Figura 86) da classe `PainelCorte`. Esse método é fundamental para o correto funcionamento do protótipo. Entre suas funções destaca-se a criação e inclusão do objeto (ator) na cena (*canvas*), além da criação e posicionamento inicial da caixa de corte.

FIGURA 86: Código para carregar a imagem na área de trabalho

```
def LerSTL(self, path):
    mesh= vtk.vtkSTLReader()
    mesh.SetFileName(path)
    mesh.Update()
    self.planos_do_cubo = vtk.vtkPlanes()

    # Clip da parte de dentro do cubo
    self.clipper_in= vtk.vtkClipPolyData()
    self.clipper_in.SetInput(mesh.GetOutput())
    self.clipper_in.SetClipFunction(self.planos_do_cubo)
    self.clipper_in.InsideOutOn()

    # Clip da parte de fora do cubo
    self.clipper_out = vtk.vtkClipPolyData()
```

```

self.clipper_out.SetInput(mesh.GetOutput())
self.clipper_out.SetClipFunction(self.planos_do_cubo)
self.clipper_out.InsideOutOff()

    # Mapeamento da regioao de corte
inMapper = vtk.vtkPolyDataMapper()
inMapper.SetInput(self.clipper_in.GetOutput())
inMapper.ScalarVisibilityOn()
inActor = vtk.vtkActor()
inActor.SetMapper(inMapper)
inActor.GetProperty().SetColor(self.obterCor())
outMapper = vtk.vtkPolyDataMapper()
outMapper.SetInput(self.clipper_out.GetOutput())
outActor = vtk.vtkLODActor()
outActor.SetMapper(outMapper)
outActor.GetProperty().SetColor(self.obterCor())

    # Representação do plano de corte
rep = vtk.vtkBoxRepresentation()
rep.SetPlaceFactor(1.50)
rep.PlaceWidget(mesh.GetOutput().GetBounds())
rep.VisibilityOn()

# Widget usado para fazer o corte.
self.BoxWidget = vtk.vtkBoxWidget2()
self.BoxWidget.SetInteractor(self.Interactor)
self.BoxWidget.SetRepresentation(rep)

# Faz o corte quando o usuário parar de mexer no plano de corte.
self.BoxWidget.AddObserver("EndInteractionEvent", self.myCallback)
self.BoxWidget.On()
self.renderer.AddActor(outActor)
self.renderer.AddActor(inActor)
self.renderer.ResetCamera()
self.Interactor.Render()
self.VerPlano= True

```

Fonte: Acervo Próprio

Os dois outros métodos que restam, e explicados aqui estão relacionados a movimentação da parte óssea na área de trabalho gráfica.

- b) Click – O primeiro reage ao evento de clique no mouse pelo usuário selecionando a parte que sofrerá a ação. Essa seleção acontece por meio da verificação das coordenadas X e Y onde ocorrera o evento de click. Isso ocorre por meio da classe `vtkPropPicker` apresentada na Figura 87, e representada no código pelo objeto `Picker`. É possível perceber por meio do código uma verificação pelo evento de click sobre um elemento gráfico, nesse caso sobre uma parte óssea, aqui também denominado ator. Caso o click tenha ocorrido fora do ator, nada acontece na tela.

FIGURA 87: Código para seleção e movimento da parte óssea

```
self.Picker=vtk.vtkPropPicker()

def Click(self, obj, event):
    iren =self.Interactor
    ren = self.renderer
    x, y = iren.GetEventPosition()
    actualY = y
    actualX = x

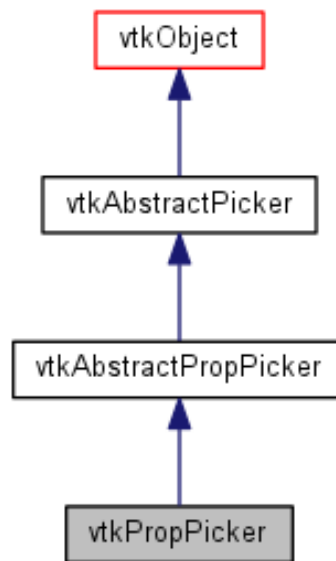
    self.Picker.Pick(actualX , actualY, 0, ren)
    actor=self.Picker.GetActor()
    print actor

    if actor is not None:
        polidata=actor.GetMapper().GetInput()
        cx, cy, cz = polidata.GetCenter()
        self.renderer.SetWorldPoint(cx, cy, cz, 0)
        self.renderer.WorldToDisplay()

        # dz é a profundidade do objeto com relação à camera
        dx, dy, dz = self.renderer.GetDisplayPoint()
        self.renderer.SetDisplayPoint(x, y, dz)
        self.renderer.DisplayToWorld()
        w = self.renderer.GetWorldPoint()
        self.LastPosition= w
        self.SelectActor=actor
```

Uma vez determinado pelo *click* sobre uma parte óssea, e não somente no *canvas*, o elemento clicado (Ator) é definido como valor para o atributo `SelectActor`. Esse ator selecionado e referenciado pelo `SelectActor` será usado no código para processar o movimento de arrastar e soltar da parte óssea definido no método `OnMotion` também parte da classe `PaineCorte` aqui explicada e detalhado mais adiante. A Figura 88 apresenta a hierarquia de classes do VTK para a classe `vtkPropPicker`.

FIGURA 88: Diagrama de Classes



Fonte: <http://www.vtk.org/doc/nightly/html/classvtkAbstractPropPicker.html>

- a) `OnMotion` – Esse método reage ao evento de arrastar e soltar de uma parte óssea através do mouse, sendo invocado pelo ambiente gráfico, e então processando a transformação linear por meio da classe `vtkTransform` para realizar o movimento de translação da parte tendo como input as coordenadas determinadas pelo evento `GetEventPosition()` conforme indicado pela Figura 89.

FIGURA 89: Definição do Método `OnMotion`

```

def OnMotion(self, obj, event):
    if self.SelectActor is not None:
        polidata=self.SelectActor.GetMapper().GetInput()
        x, y = self.Interactor.GetEventPosition()
  
```

```

# giu Lx,Ly= self.LastPosition[0], self.LastPosition[2]
Lx,Ly,Lz= self.LastPosition[0], self.LastPosition[1],
self.LastPosition[2]
cx, cy, cz = polidata.GetCenter()
self.renderer.SetWorldPoint(cx, cy, cz, 0)
self.renderer.WorldToDisplay()
dx, dy, dz = self.renderer.GetDisplayPoint()
self.renderer.SetDisplayPoint(x, y, dz)
self.renderer.DisplayToWorld()
w = self.renderer.GetWorldPoint()
transform = vtk.vtkTransform()

#transform.Translate(w[0]-Lx,0,w[2]-Ly)
if self.currentSelectedAxis == "x":
    transform.Translate(w[0]-Lx,0,0)

if self.currentSelectedAxis == "y":
    transform.Translate(0,w[1]-Ly,0)

if self.currentSelectedAxis == "z":
    transform.Translate(0,0,w[2]-Lz)
transformFilter=vtk.vtkTransformPolyDataFilter()
transformFilter.SetTransform(transform)
transformFilter.SetInput(polidata)
transformFilter.Update()

self.SelectActor.GetMapper().SetInput(transformFilter.GetOutput())
self.LastPosition=w
self.SelectActor.Modified()
pub.sendMessage('Renderizar')

```

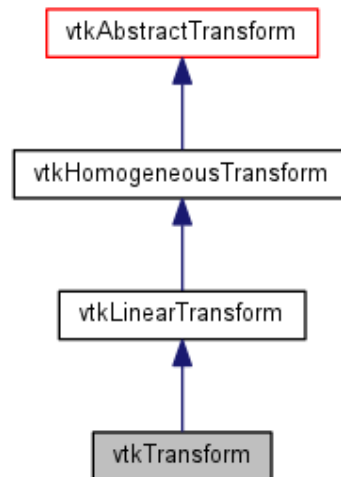
Fonte: Acervo Próprio

Nota-se aqui mais uma vez que seguindo a abordagem de encadeamento do VTK (*pipeline*), as classes `vtkTransform` e `vtkTransformPolyDataFilter` estão sequenciadas nessa respectiva ordem.

Por fim, é enviada uma mensagem para o sistema gráfico para que ocorra uma nova renderização tendo em conta as novas coordenadas, recebidas no momento em que o usuário

soltou o botão do *mouse*. Na Figura 90 visualiza-se a estrutura hierárquica do `vtkTransform`, aplicado neste código.

FIGURA 90: Hierarquia de classe de `vtkTransform`



Fonte: <http://www.vtk.org/doc/nightly/html/classvtkTransform.html>

No próximo capítulo, o autor aborda os resultados e limitações conhecidos por meio de uma pesquisa direcionada com especialistas da área médica de estudo, onde os mesmos tiveram a experiência no uso do Protótipo VTO – UFU – Visualização do Tratamento Ortognático, desenvolvido pelo autor.

CAPÍTULO 6

6 RESULTADOS E LIMITAÇÕES

O sistema foi testado por 10 especialistas da área de cirurgia buco maxilo facial, sendo 3 provenientes da cidade de Curitiba/PR e 7 da Baixada Santista (Santos e São Vicente)/SP, com uma média de 21 anos de experiência específica na área alvo do estudo desse trabalho. As entrevistas e experimentações ocorreram no período de Setembro a Novembro de 2013. É interessante salientar que todos os profissionais atuam na área e, além disso, 7 deles também lecionam em universidades particulares e públicas.

Após a introdução do tema desse trabalho para os entrevistados, foi apresentada uma simulação de uma osteotomia, realizando procedimentos de impactação e disjunção na imagem 3D, com o objetivo de demonstrar a ferramenta.

Na sequência são apresentados os resultados desses questionários por meio de gráficos e conclusões do autor.

6.1 Procedimentos

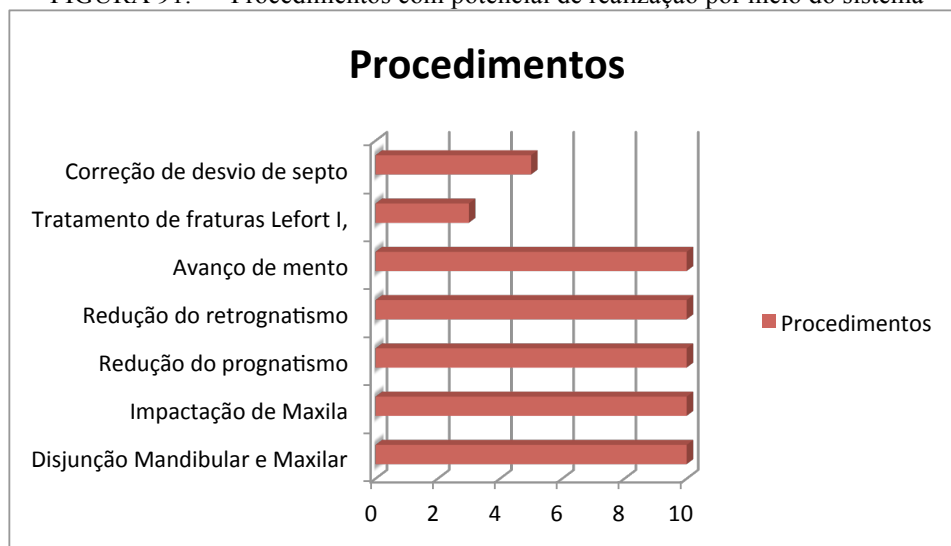
Após a apresentação do software e da experimentação total pelo especialista, considerando que o mesmo assumiu o controle do sistema e realizou algumas simulações arbitrárias, o especialista foi convidado a responder um questionário (Apêndice A).

Como abordado nos capítulos anteriores, à cirurgia buco maxilo facial pode envolver diferentes tipos de procedimentos. Conforme os gráficos a seguir, os especialistas entendem que o software (Protótipo VTO – UFU – Visualização do Tratamento Ortognático) desenvolvido, permite realizar alguns procedimentos com diferentes variações de relevância e pontos de vista distintos. Todos consideram que são importantes as funcionalidades apresentadas, podendo estas ser aplicadas também, quando o foco está no componente tegumentar, funcionalidade esta que não é contemplada no desenvolvimento desta dissertação, e sim sugerida como forma de extensão como trabalhos futuros.

Observando a Figura 91, nota-se que alguns desses procedimentos, tais como avanço de mento, redução de retrognatismo mandibular ou maxilar, redução do prognatismo mandibular ou maxilar, impactação e disjunção da maxila foram aceitos como procedimentos possíveis de serem realizados através do sistema. Por meio de uma análise simplista, para tais

procedimentos é necessário apenas fazer a movimentação das osteotomias destacadas em direções específicas, sem a necessidade do movimento de rotação.

FIGURA 91: Procedimentos com potencial de realização por meio do sistema



Fonte: Acervo Próprio

Dos profissionais entrevistados, 3 deles não consideram viável a utilização da aplicação nos processo de Lefort I, II e III, e esse resultado é devido a falta da função de rotação das osteotomias destacadas em uma impactação e/ou na disjunção, pois para que o procedimento possa ser efetivamente analisado, é necessário o movimento de leve inclinação nas partes ósseas destacadas no sistema, sendo assim, somente após esse movimento seria possível realizar a fixação com os parafusos, para posteriormente obter uma análise mais realista do que poderá ocorrer durante o tratamento de fraturas Lefort I, II e II, salienta-se outro ponto de vista em relação a não considerarem necessário uma análise prévia do procedimento, já que nos procedimentos que envolve Lefort trata-se principalmente do reposicionamento dos ossos já quebrados por um acidente. Quanto ao procedimento de correção de desvio de septo, não é um procedimento comumente realizado pelos profissionais entrevistados.

6.1.2 Procedimento Cirúrgico

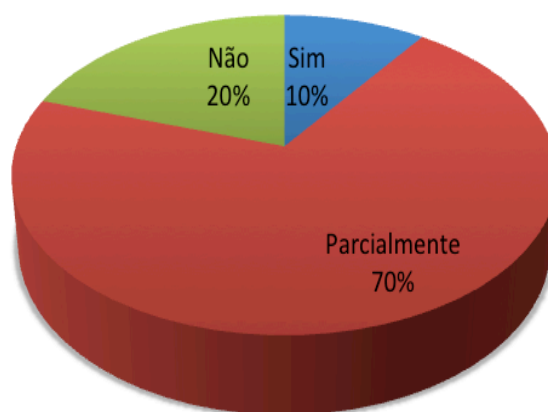
Os resultados apresentados a seguir (Figura 92), indicam a percepção dos entrevistados, quanto à maturidade do protótipo em relação a sua aplicabilidade em uma preparação para um procedimento cirúrgico. Por se tratar de um sistema criado em caráter de

protótipo, não se esperava a aceitação imediata de forma unanime para uso no preparo e simulação do procedimento cirúrgico.

Observando os resultados obtidos expostos pela Figura 92 percebe-se que uma adaptação ou evolução ainda é necessária para preencher algumas lacunas (funcionalidades) desejadas pela maioria dos especialistas, como citado anteriormente. As funcionalidades apontadas como ausentes ou a evoluir serão abordadas ainda nesse capítulo.

FIGURA 92: Apto para usar na preparação ou simulação do procedimento cirúrgico

Utilizariam em Procedimento Cirurgico



Fonte: Acervo Próprio

Mesmo com todas as considerações e limitações presentes no sistema Protótipo VTO – UFU – Visualização do Tratamento Ortognático desenvolvido, o saldo é considerado positivo unindo os resultados escolhidos como sim, e parcialmente, totalizando 80% dos resultados.

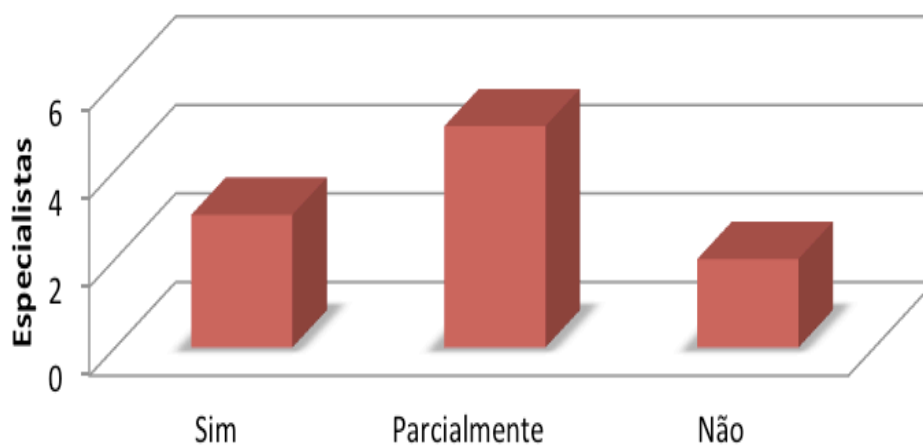
6.1.3 Projeção de Resultado

Com uma questão mais específica, os entrevistados foram questionados quanto à capacidade do protótipo em projetar o resultado de procedimento cirúrgico.

Embora a projeção do resultado tenha sido considerada viável ou parcialmente viável pelos entrevistados a maioria do grupo de entrevistados complementou de forma verbal ou escrita que a parte tegumentar, movimento do tecido mole, estaria fazendo falta para que a análise e projeção do resultado fosse completa conforme indica a Figura 93.

FIGURA 93: Projeção do resultado

Utilização para Projeção do Resultado Cirurgico



Fonte: Acervo Próprio

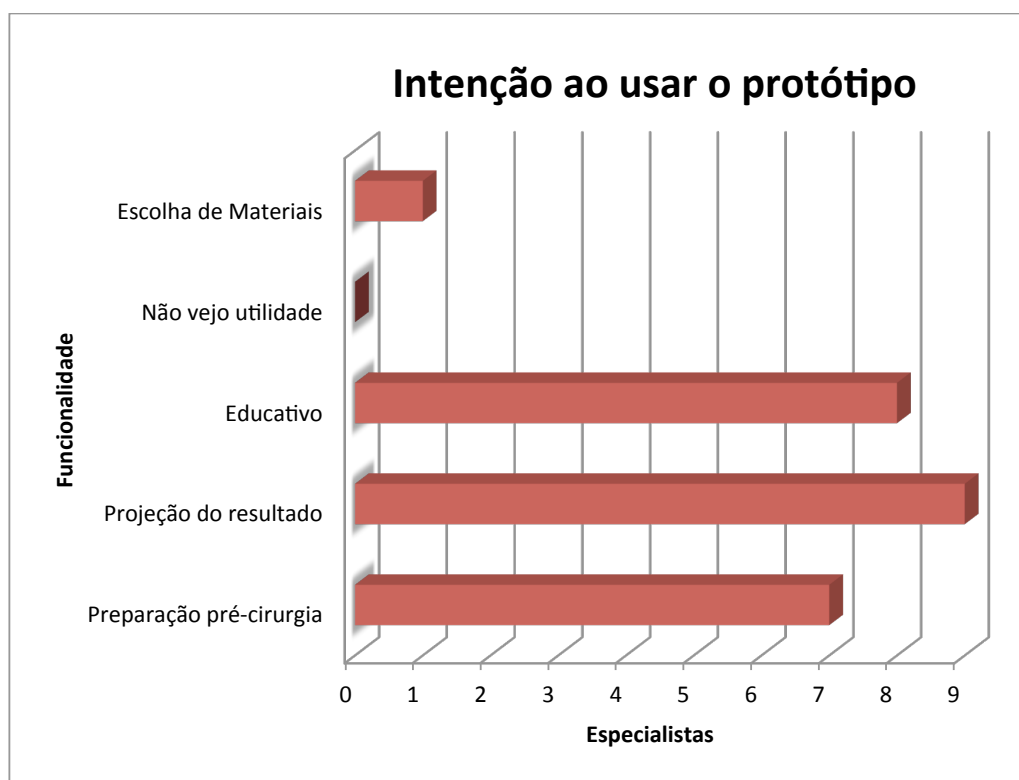
Mesmo com uma pequena variação, o resultado exibido na Figura 93, foi próximo ao resultado da Figura 92, onde a maioria dos entrevistados teve inclinação positiva, com algumas restrições.

6.1.4 Intenção no uso do software

Além da expectativa em utilizar o software para projeção de resultado, bem como análise cirúrgica, parte dos especialistas entrevistados mostrou interesse em utilizar o software para fins educacionais, salienta-se que dos 10 entrevistados, 7 são educadores, onde parte de suas funções esta em educar alunos da área de cirurgia buco maxilo facial. Destaca-se, que todos os entrevistados encontraram utilidade no protótipo como lhes foi apresentado.

Os diferentes interesses no uso do Protótipo VTO – UFU – Visualização do Tratamento Ortognático, sendo que somente um especialista entrevistado desejou que o software pudesse orientá-lo quanto à escolha de materiais a ser utilizado durante o procedimento cirúrgico. Essa solicitação jamais havia sido considerada pelo autor antes do *feedback* desse especialista, pode ser observado na Figura 94.

FIGURA 94: Intenção de uso do protótipo



Fonte: Acervo Próprio

6.1.5 Valor para o paciente

Como afirmado nos capítulos anteriores, um dos principais objetivos buscados pelo sistema proposto, é o de comunicar o procedimento cirúrgico ao paciente, inserindo-o no contexto, de forma que o paciente envolvido possa ter uma maior compreensão do procedimento e os resultados esperados do mesmo. Os especialistas expressaram-se positivamente quanto à capacidade do software em trazer real benefício de compreensão e aceite para o paciente envolvido.

Quase em sua totalidade, os especialistas entendem que o software atinge esses objetivos, sendo que somente um desses especialistas se posicionou de forma contrária com a alegação da falta de contemplar o movimento de tecido mole por meio do software, de acordo com a Figura 95.

FIGURA 95: Valor para o paciente

Traz valor para o paciente



Fonte: Acervo Próprio

Enumerando as entrevistas na ordem cronológica em que foram aplicadas, tem-se as seguintes afirmações de alguns profissionais. São elas:

- Profissional 1 afirma: “Sim. Seria uma maneira didática de mostrar aos pacientes os resultados dos procedimentos indicados.”

- Profissional 5 afirma: “Sim. Com a Projeção/Simulação de imagens o entendimento e aceitação da cirurgia se torna mais favorável.”

- O Profissional 6 afirma: “Sim. Para demonstrar como seria a cirurgia.”

- O Profissional 7 fez a seguinte ponderação: “Poderia ser usado, mas demandaria treino no uso do software”

- O Profissional 9 reforçou o que já havia sido comentado anteriormente com a seguinte afirmação: “Sim. A partir do momento em que as alterações tegumentares possam ser visualizadas.”

- O Profissional 10, indicou algo também já anteriormente exposto: “Sim, um valor ilustrativo e educacional, pois apresenta de forma simples as possibilidades de movimentação ortognática.”

6.1.6 Projeção em Tecido Mole

Observou-se também que todos os entrevistados consideram que a funcionalidade de projeção do resultado em tecido mole (função essa não suportada pelo software apresentado), seria de fundamental relevância, conforme Figura 96.



Fonte: Acervo Próprio

Profissionais que fizeram anotações justificando a necessidade da função movimentação de tecido mole:

- 1º: “A análise facial é fundamental para o resultado final.”.
- 5º: “A projeção de tecido mole aumenta a confiabilidade do procedimento proposto.”.
- 7º: “Relevante, pois seria um fator de argumentação, embora não se possa usar como previsão objetiva”.

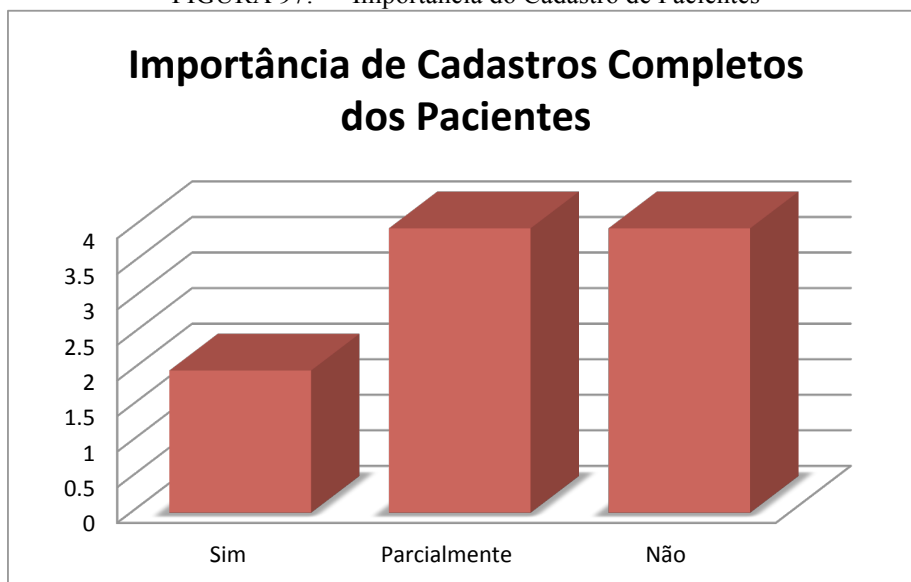
6.1.7 Importância do Cadastro de Pacientes

Observando-se os resultados da Figura 97, de forma comparativa, os especialistas foram arguidos quanto à necessidade de criar e manter cadastros informativos de pacientes, funcionalidade essa, presente na maioria dos softwares comerciais. Identificou-se que 60% dos

entrevistados manifestaram-se de forma favorável a existência desse tipo cadastro, indicando que um maior conhecimento do paciente é importante para compreendê-lo e planejar o procedimento cirúrgico adequado.

Os comentários a seguir foram feitos pelos entrevistados.

FIGURA 97: Importância do Cadastro de Pacientes



Fonte: Acervo Próprio

- Profissional 2 – “É importante ter acesso às informações do paciente e, especialmente aos dados provenientes do exame físico intra e extra bucal.”.

- Profissional 3 – “Acredito que um cadastro prévio seja importante não precisando ser muito extenso.”.

- Profissional 4 – “Acredito ser importante os dados completos do paciente.”.

- Profissional 7 – “Podem não ser essenciais, porém em algum momento podem ser úteis, portanto interessante a manutenção dos dados.”.

- Profissional 10 – “Não é importante, espera-se uma maior simplicidade e rapidez no uso, dessa forma fica mais agradável de ser utilizado pulando a etapa do cadastro”.

Os demais profissionais não exprimiram comentários adicionais sobre essa questão.

6.1.8 Interesse em utilizar o Protótipo VTO – UFU – Visualização do Tratamento Ortognático

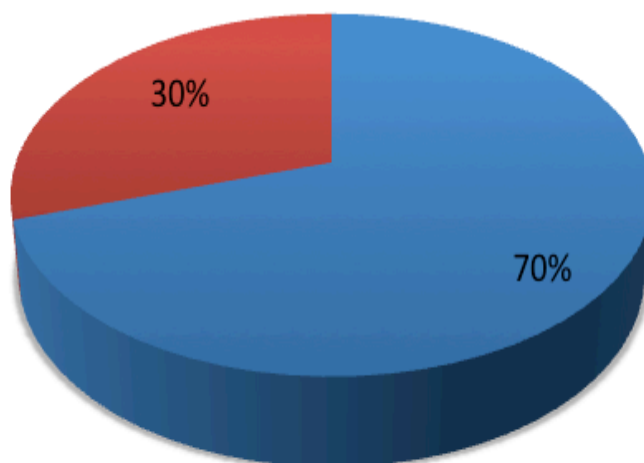
A entrevista também procurou conhecer o interesse dos entrevistados em utilizar o sistema desenvolvido da forma como está. Após as devidas explicações, orientações e testes

realizados por parte dos especialistas, 70% deles indicaram real interesse em utilizá-lo, já os 30% restantes indicaram que a ausência de projeção em tecido mole é o motivo por não terem interesse no protótipo sem essa funcionalidade.

Alguns comentários produzidos nas entrevistas, também foi notado a ausência de ferramenta para medições lineares e angulares, ferramentas essas já incluídas no sistema *InVesalius*TM, as quais portanto estarão disponíveis no futuro, quando houver a integração entre os sistemas. Os resultados são representados acima na Figura 98.

FIGURA 98: Interesse em usar o protótipo

Interesse do profissional na utilização do protótipo



Fonte: Acervo Próprio

6.1.9 Interesse em adquirir software

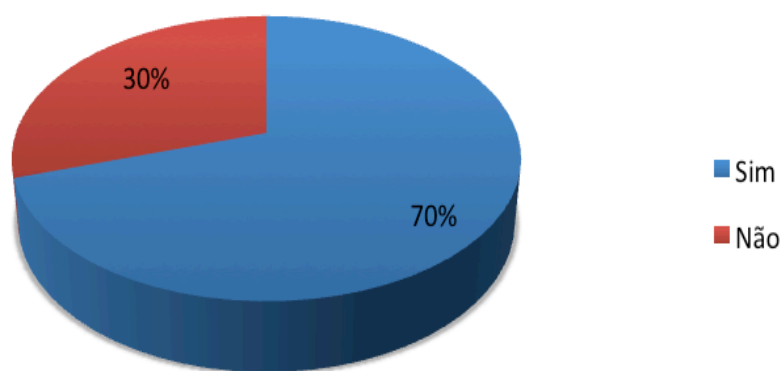
Para compreender ainda mais o interesse por ferramentas como a apresentada pelo autor, os sujeitos da pesquisa foram questionados sobre o interesse em comprar um software para esses propósitos, não necessariamente o protótipo aqui discutido, e nesse item, 70% indicaram interesse em realizar a compra de um software comercial como o sistema apresentado nesse estudo.

Observando as Figuras 98 e 99, percebe-se os mesmos números comparando os resultados entre profissionais interessados usar o protótipo e em adquirir um software similar

ou melhor. Inicialmente fez certo sentido entender que os mesmos 30% que não possuem interesse em utilizar o protótipo não queiram adquirir também, no entanto ao ler as respostas dos profissionais percebe-se uma divergência neste sentido. (Figura 99).

FIGURA 99: Interesse em comprar o software

Compraria Software com as funcionalidades do protótipo



Fonte: Acervo Próprio

O autor acredita que houve uma falha de interpretação na pergunta, após as seguintes conclusões:

Os profissionais 1, 6, 9 e 10 usariam e comprariam o protótipo, caso isso fosse possível.

Já os profissionais 2, e 5 usariam o protótipo, no entanto não possuem interesse imediato em comprar software similar, tal explicação foi verbalizada, pois esses profissionais já tem a possibilidade de utilizar softwares como esse em seus locais de trabalho ou de pesquisa, sem a necessidade de investimento.

O profissional 3, citou que não usaria o protótipo, devido as limitações do mesmo, particularmente para esse profissional, a limitação do tegumentar e medições, no entanto afirmou ter interesse em adquirir um software com essa finalidade.

Os profissionais 4 e 8 afirmaram que não usariam o protótipo por o mesmo não possuir a projeção em tecido mole. Afirmaram também ter interesse em adquirir um software para esta finalidade.

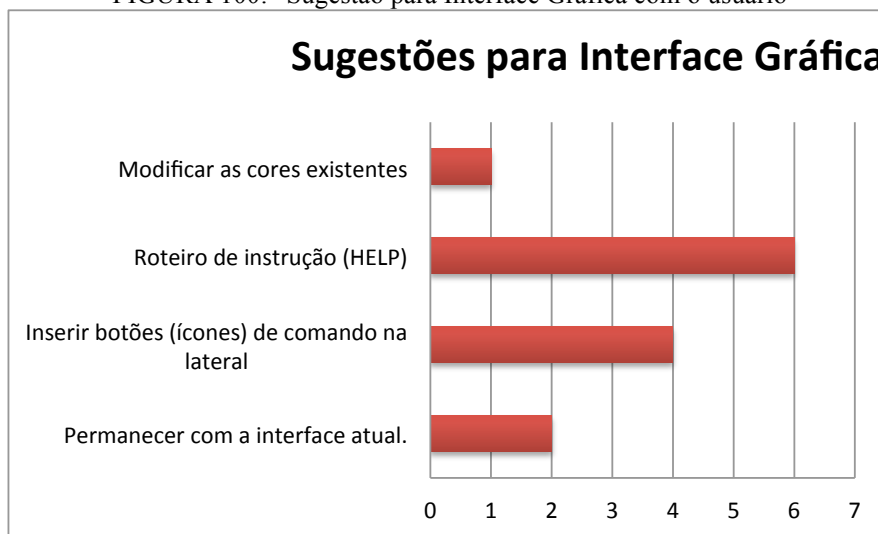
O profissional 7 indicou que utilizaria o protótipo, apesar não possuir interesse realizar investimentos neste sentido.

Seguindo a mesma linha de interesse no software, foram então questionados quanto a utilização desse software uma vez que fosse disponibilizado de forma gratuita. Nesse caso, 100% dos especialistas indicaram que fariam uso do software caso o mesmo seja oferecido sem custos.

6.1.10 Sugestões para a interface gráfica

No decorrer das entrevistas e da experimentação, foram solicitados aos especialistas que apontassem limitações e contribuíssem com sugestões de melhorias tanto para a interface gráfica, como também sugestões gerais para o Protótipo VTO – UFU – Visualização do Tratamento Ortognático. As Figuras 100 e 101, apresentam esses resultados.

FIGURA 100: Sugestão para Interface Gráfica com o usuário



Fonte: Acervo Próprio

Observando a Figura 100, notou-se que os possíveis usuários fizeram seus apontamentos somente sobre as opções que lhes foram apresentadas na questão (Apendice A), sendo essas opções de múltipla escolha. O Profissional 9 considerou a interface adequada, opinião essa compartilhada por outros 2 profissionais que assinalaram para permanecer com a interface gráfica atual.

Observou-se também que a ausência de um guia de instruções e botões de acesso rápido foram sugestões marcantes de melhorias para a interface gráfica, seja isso por uma experiência anterior com outro software ou simplesmente para trazer um maior conforto e segurança para o usuário.

6.1.11 Sugestões gerais

A figura 101 apresenta os resultados das sugestões gerais, onde ficou evidenciado por meio das entrevistas que entre as limitações do sistema proposto, estão a ausência do movimento de rotação, bem como ferramentas para medições (cefalometria) e projeção em tecido mole. Todas essas limitações foram indicadas pelos profissionais como sugestões para o protótipo.

FIGURA 101: Sugestões gerais



Fonte: Acervo Próprio

Todas as limitações foram incluídas na seção trabalhos futuros, presentes nesta dissertação, pois o complemento de parte destes apontamentos é de suma importância para a completa análise dos profissionais.

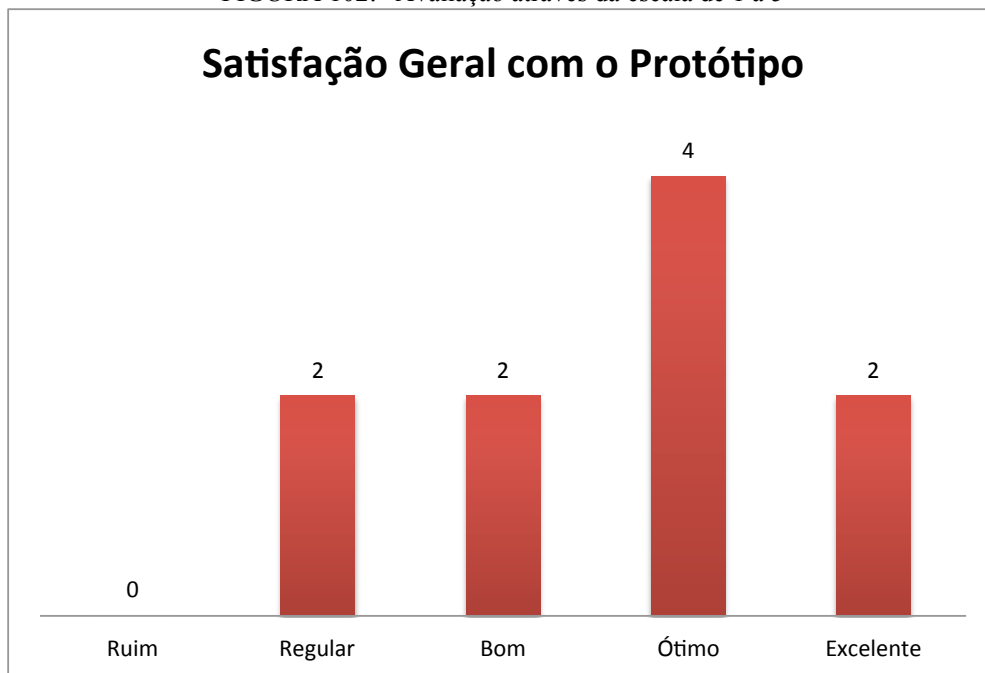
6.2 Avaliação

Ao final, os entrevistados foram convidados a avaliar o sistema proposto utilizando uma escala variando de 1 a 5, onde 1 é ruim e 5 excelente. Nota-se pelos resultados que apesar das limitações, o *feedback* foi positivo, sendo a maior parte das respostas entre bom, ótimo e excelente.

Levando-se em consideração os questionários e resultados obtidos, conclui-se que a interação entre o pesquisador, especialista e software trouxeram valiosas trocas de

experiências, sugestões e principalmente aprendizagem, dessa forma também determinantes para uma melhor compreensão dos questionários respondidos por esses especialistas, bem como o *feedback* dos mesmos. Os resultados dessa avaliação são apresentados na Figura 102.

FIGURA 102: Avaliação através da escala de 1 a 5



Fonte: Acervo Próprio

Satisfatoriamente notou-se que o sistema proposto pode trazer valor para todos os envolvidos no processo, incluindo principalmente o paciente e o especialista. Ficou claro também por meio dos números compilados o interesse dos entrevistados em acompanhar a pesquisa, a evolução do software e o principalmente o desejo em usá-lo no futuro.

CAPITULO 7

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo, destacam-se os aspectos da pesquisa e conclusões do trabalho apresentado nesta dissertação. Além disso, serão apresentadas algumas sugestões para trabalhos futuros e a contribuição científica do presente trabalho.

7.1 Conclusão

O objetivo geral desse trabalho, que é a análise e simulação de um procedimento cirúrgico das partes ósseas para uma cirurgia ortognática foi cumprido. Conforme os resultados das pesquisas, a função de corte (osteotomia) atende de forma satisfatória as necessidades iniciais para o processo de análise e simulação do procedimento cirúrgico, no que se refere à quebra e reposicionamento ósseo da arcada dentária superior, inferior e queixo, de forma a compreender todos os cortes básicos para o procedimento cirúrgico ortognático, apresentado nas bibliografias estudadas, somando-se a isso a possibilidade de realizar a disjunção da maxila.

Após à aplicação do questionário a 10 profissionais da especialidade, com experiência média de 21 anos na área, ficou claro que a aplicação desenvolvida se mostrou adequada para simular o procedimento cirúrgico dos pacientes, embora ainda com algumas restrições. Apesar das restrições encontradas não estarem previstas na proposta inicial, conclui-se que algumas delas possuem um grau de importância grande, dando margem a continuação do estudo com trabalhos futuros (Seção 7.3), entretanto tais restrições não impediram que o resultado esperado pela proposta inicial fosse cumprido.

Todos as propostas do projeto foram alcançados, e são elencados aqui; corte e movimentação óssea, manuseio simples e intuitivo pelo especialista e ferramenta de preparação, planejamento e comunicação.

Com base no *feedback* dos especialistas, entendeu-se também que o Protótipo VTO – UFU – Visualização do Tratamento Ortognático, é capaz de beneficiar não somente, o cirurgião e o próprio paciente, mas outros profissionais envolvidos, como o ortodontistas e protéticos e por fim estudantes de disciplinas relacionadas, enfim atuando como uma ferramenta de comunicação.

Levando em consideração os objetivos de planejamento, preparação e simulação cirúrgica, buscou-se apresentar por meio das experimentações dos especialistas que é possível trabalhar em prol desses objetivos em um ambiente gráfico 3D, de forma absolutamente não intrusiva do ponto de vista do paciente uma vez que os dados gerados por meio do exame tomográfico são a única fonte de dados, sem qualquer outro procedimento que traga prejuízo ou incômodo ao paciente.

Sabendo que a manipulação das partes ósseas são oriundas de uma fonte fidedigna, nesse caso a tomografia, o sistema, aqui desenvolvido traz uma maior confiança ao especialista uma vez que a manipulação pelo mesmo, embora ainda que em ambiente virtual, é realizada na representação digital mais próxima possível do paciente, levando-se em consideração para essa afirmação, que a imagem tenha sido gerada por dispositivos tomográficos que produzam um *dataset* de qualidade. Espera-se também que alguns cuidados sejam tomados durante a aquisição dessas imagens, tais como obtenção da imagem com a boca semi-aberta, o que irá facilitar a importação das mesmas para o ambiente virtual em 3D usado pelo especialista.

As tecnologias adotadas foram fruto de estudos e testes. Como esperado, por meio de provas e experimentos, houve um refinamento nessas tecnologias com base em capacidade e resultados alcançados, além disso, considerou-se também as tecnologias usadas no desenvolvimento do software público *InVesalius*TM, o qual foi um grande fator de influencia para as escolhas feitas nesse trabalho, pois muitos princípios e objetivos são similares, tais como ambiente tridimensional, tecnologia de código aberto, e manipulação de imagens médicas.

Muitas das escolhas ao do desenvolvimento do trabalho como um todo, incluindo a isso o desenvolvimento do software, tiveram considerável influência visto que a integração plena com o *InVesalius*TM foi e ainda é algo desejado.

O uso da biblioteca gráfica VTK por meio da linguagem de programação Python foi de grande valor para os propósitos de manipulação das imagens em um ambiente gráfico 3D, de forma razoavelmente simples, dada a riqueza e poder dessa biblioteca no gerenciamento dos dados, modelos, imagens e funções gráficas.

Finalmente, pôde-se observar que as principais características de sistemas de Realidade Virtual (navegação, imersão e o envolvimento) demandaram pouco esforço de programação em função da arquitetura proposta e das ferramentas utilizadas para suportá-la.

7.2 Aprimoramentos do Sistema

Considerando o sistema desenvolvido como objeto principal dessa seção, são listados pontos de aprimoramento do sistema, levando-se em consideração limitações identificadas pelo autor ou por *feedbacks* dos especialistas durante as experimentações.

- Melhoria na Interface Gráfica ou GUI (Graphical User interface) de forma a oferecer diferentes visões do mesmo objeto na tela;
- Prover barra de ferramentas na lateral e contextualizada, ou seja, somente ativar; ferramentas que tenham aplicabilidade em determinado momento durante o uso do sistema;
- Disponibilizar documentação online para auxiliar o uso do sistema;
- Portar o sistema para diferentes sistemas operacionais;
- Incluir ferramenta de rotação, de forma a permitir o movimento em torno de um único eixo;
- Importar imagens diretamente no formato DICOM;
- Incluir ferramenta de régua para realizar medições e cefalometria;
- Portar o sistema para outras linguagens de programação como o Java.

7.3 Trabalhos Futuros

O presente estudo permite que outros desdobramentos sejam desenvolvidos. Notadamente como apresentado em alguns resultados do Capítulo 6, a alteração do tecido mole vinculado ao movimento ósseo, que foi muito citada pelos especialistas, como uma melhoria bastante esperada. Isto será capaz de completar o processo de VTO, de forma a apresentar resultados de movimento de pele, e consequentemente estéticos pós cirúrgicos por meio da ferramenta.

É possível também, aplicar guias restritoras dos movimentos ósseos, de forma a orientar o especialista sobre os limites de cada movimento. Esses movimentos poderiam ainda

ser padronizados na ferramenta como *templates* distintos para cada tipo de anomalia, dessa forma atuando como um assistente ao especialista.

Existe também a possibilidade de evoluir para uma pesquisa voltada para aplicações educacionais, onde um professor virtual orienta o aluno na preparação cirúrgica por meio de treinamentos guiados, buscando minimizar falhas em procedimentos reais.

Uma outra pesquisa poderia ser alavancada para geração de modelos de enxerto ósseo com base nos cortes e movimentos realizados pelo especialista, baseado na prototipagem rápida, assunto esse em grande expansão com a popularização das impressoras 3D, podendo ainda desenvolver a criação dos modelos das ferragens a ser utilizada no procedimento cirúrgico, de tal sorte, que diminua o tempo dos ajustes de tais peças no ato da cirurgia, colaborando para a diminuição do tempo de procedimento cirúrgico.

Além disso, as técnicas aqui apresentadas podem ser claramente utilizadas para a construção de outros objetos de estudo que não sejam necessariamente, da área de buco maxilo facial.

7.4 Considerações Finais

As conclusões expressas acima, relativas aos resultados obtidos por meio do sistema desenvolvido denominado Protótipo VTO – UFU – Visualização do Tratamento Ortognático, bem como o *feedback* satisfatório dos testes realizados com especialistas, de certa forma confirmam algumas contribuições desse trabalho.

Do ponto de vista científico, as principais contribuições desse trabalho são:

- A capacidade de planejar ou simular um procedimento cirúrgico em ambiente totalmente tridimensional usando informações reais e precisas dos pacientes obtidas por meio de equipamentos digitais como o tomógrafo.
- Dentro de uma perspectiva de desenvolvimento de software, foi possível identificar limitações na biblioteca Java3D que trouxeram obstáculos para a pesquisa, e fizeram com que outras tecnologias, linguagens e bibliotecas de programação fossem pesquisadas.
- Apresentar a possibilidade de usar e construir aplicações médicas utilizando tecnologias de código aberto.
- Compreender e comunicar por meio de um ambiente de computação gráfica, o procedimento cirúrgico para todas as envolvidos, sejam cirurgiões e/ou pacientes.

REFERÊNCIAS

- 3dmd – disponível em <<http://www.3dmd.com/>> - Acesso em 1 dezembro 2013
- Accorsi, M.; Velasco, L. “ **Diagnóstico 3D em Ortodontia: a tomografia cone-beam aplicada**”. 1ªed, Editora Napoleão Ltda, São Paulo – Nova Odessa 2011.
- Amorim, P. H. J. , et al, **Software InVesalius - Guia do Usuário** - CTI (Centro de Tecnologia da Informação Renato Archer), unidade do Ministério da Ciência e Tecnologia (MCT)
- Amorim, Paulo H. J.; Moraes, Thiago F. de; Azevedo, Fábio de S.; Silva, Jorge V. L. da **InVesalius: Software Livre de Imagens Médicas** -, Centro de Tecnologia da Informação Renato Archer - CTI, campinas/SP – 2011 - CSBC2011
- Anatomage, Inc - **Invivo5.1 Reference Manual - 2011** - San Jose, CA (2011)
- Andersen V. **Nombegriff und Optimumsbegriff**. Fortsch Orthod 1931;1:276-286
- Bartolo, P. J. da S. - **Virtual and Rapid Manufacturing: Advanced Research in Virtual and Rapid Prototyping (Balkema-Proceedings and Monographs in Engineering, Water and Earht Sciences)** [Hardcover] 2008 - Copyright (coffee) 2006 Sun Microsystems, Inc.
- Broadbent, B. H. **A New x-ray technique and its application to orthodontia**. Angle Ortho 1931;1:45-66
- Brons, A.A.; Scarfe, W C.; at al. “**Precisão da tomografia linear Cone Beam CT derivadas de imagens 3D**”. In: Angle Orthodontist, artigo de 2009, University of Louisville School of Dentistry, Louisville, Ky.
- Brown K.; Daniel, P.**Ready-to-Run Java 3D** (Wiley Computer Publishing, 1999)
- Cavalcanti, M.. “**Tomografia Computadorizada por feixe cônico – Interpretação e Diagnóstico para o Cirurgião-dentista**”, Livraria Santos Editora Ltda, São Paulo – SP, 2010
- Centro de cirurgia Odontológica – disponível em <<http://www.odontologiamt.com.br/procedimentos/item.asp?id=50> – 2010>. Acesso em 1 Novembro 2013
- Cericato G.; Garbin D.; Fernandes, A. “**Uso dos Sistemas Especialistas em Odontologia**”, CBIS'2006. Costão do Santinho, Florianópolis, SC, 14-18 de outubro,
- Cevidanes, Lucia H. C.,et al. **Three-dimensional surgical simulation**, 2010
- Costa, F., “**Utilização da Realidade Virtual em Odontologia. Sociedade Brasileira de Informática em Saúde**”, X Congresso Brasileiro de Informática e Saúde. Campos do Jordão, São Paulo, 2007. 29 – 3 dez.
- Dennis J Bouvier, **Getting Started with the Java 3D™ API - Chapter 2 Creating Geometry**, 2000

DICOM – Digital Imaging and Communications in Medicine – disponível em -
<<http://medical.nema.org/Dicom>> - Acesso 5 janeiro 2012

Dolphin Imaging Featuring 3D – disponível em <http://www.dolphinimaging.com/3d.html> –
Acesso em 15 fevereiro de 2012

Dows, W.B. **Analysis of the dentofacial profile**. Angle Orthod 1956;26;192-212

Farman, A. et al. **“Education in the round: multidimensional imaging in dentistry”**. Inside
Dentistry. Louisville, Kentucky January 2008, (1):39-41

Gimenez, C.M.M. ; et al. **“Avaliação cefalométrica do perfil mole de pacientes face longa submetidos à cirurgia ortognática: estudo retrospectivo”**. In: Rev. dental press ortodon. ortopedi. facial; vol.11(6): 91-103, Maringa - PR, nov.-dez. 2006.

Graber, Dan; Cevdanes, Lucia S.H.; Proffit, William R. - **Working with DICOM craniofacial images**, AJO-DO - **American Journal of Orthodontics and Dentofacial Orthopedics**, Vol. 136, Issue 3, Pg 460-470, 2009

Harrel Jr, Willian E. – disponível em <http://drharrell.com/images/d/645/Fig6_Camera-4000-th.gif> - Acesso em 1 Dezembro 2013

Hofrath, H. **Die Bedeutung der Röntgenfern und Abstandsaufnahme für die Diagnostik der Kieferanomalien**. Forstschr Orthod 1931; 1:231-258

IBGE - Instituto Brasileiro de Geografia e Estatística. Disponível em:
<<http://saladeimprensa.ibge.gov.br/>> – Acesso em 23 de Março de 2013

Jacobson, A.; Jaconson, R. **“Cefalometria Radiografica: Imagens Básicas e 3D”**. 2ªed, Quintessence editora Ltda., São Paulo – São Paulo 2010.

Kinser, J. **Python for Bioinformatics**. Jones and Bartlett Publishers, (2009).

Materialise dental – SimPlant O&O Disponível em -
<<http://www.materialisedental.com/materialise/view/en/600383-SimPlant+O%26O+.html>> -
Acesso em 10 novembro 2013

MediaWiki - Public Wiki - Disponível em:
<http://www.vtk.org/Wiki/VTK/Examples/Python/GeometricObjects/Display/Hexahedron> -
Acesso em 21 de abril 2012

Melo, Ana Cristina. **Desenvolvendo aplicações com UML: Do conceitual à implementação**, Ed Brasport 2002 Rio de Janeiro - RJ

Menezes, M.; Sforza, C.; **“Morfometria tridimensional (3D) da face”**. In: **Artigo 2010, Dental Press Journal Orthodontics**, Departamento de Morfologia Humana e Ciências Biomédicas da Universidade de Milão, Itália.

Metzger, M C; et al. **“Fabricação de Splints para Cirurgia Ortognática Utilizando Impressora Tridimensional”**. In: Artigo 2008, Revista Oral Surgery, Oral Medicine, Oral Pathology, Oral Radiology & Endodontics, Freiburg, Germany; and Aarau, Switzerland Albert-Ludwig University Freiburg and Hirslanden Medical Center.

Motta, A.T.S. da, Brunharo, I.H.P. et al., **“Simulação computadorizada do perfil facial em cirurgia ortognática: precisão cefalométrica e avaliação por ortodontistas”**. In: Rev Dental Press Ortodon Ortopedi Facial; Maringa – PR, set./out. 2007, vol.12, n. 5, p. 71-84.

Nunes, Fátima L. S. **Capítulo 2 - Introdução ao Processamento de Imagens Médicas para Auxílio ao Diagnóstico – Uma Visão Prática** -. () – Universidade de São Paulo, São Paulo, 2006

Ortodontia Contemporanea BLOG - Loiola, Marlos – disponível em -
<<http://www.ortodontiacontemporanea.com/2009/02/historia-da-ortodontia-sebastiao.html>> -
Acesso 17 agosto 2011

Paccini, A. J. **Roentgen ray antropometry of the skull. J Radiol** 1922;3:230,322-331,418-426.

Papademetris, Xenophon ; Joshi, Alark, **An Introduction to Programming for Medical Image Analysis for Medical Image Analysis with the Visualization Toolkit**, 2009, 2ª ed, BioImage Suite

Pianykh, O. S. (2007). **Digital Imaging and Communications in Medicine (DICOM)**. Springer Publishing, 1th edition.

Rappin, N. and Dunn, R. (2006). **wxPython in action. Manning Pubs Co Series. Manning.**

Ribas, M.de O. et al. **“Cirurgia ortognática: orientações legais aos ortodontistas e cirurgões bucofaciais”**. Dental Press Ortodon Ortop Facial. v. 10, n. 6, p. 75-83, Maringá – PR, nov./dez. 2005

Rodrigues, M.A.F.; Silva, W.B. , Barbosa Neto, M.E.. **“Um Sistema de Realidade Virtual para Tratamento Ortodôntico”**. (Mestrado em Informática Aplicada) – Universidade de Fortaleza, Fortaleza-CE, 2006.

Santin, R; Kirner, C. **“Detecção em tempo real de objetos em vídeo usando Realidade Aumentada”**. In: Anais do WRA`2005 – II Workshop de Realidade Aumentada, 2005, Piracicaba, SP, v1, p.25-28.

Schendel, Stephen A.; Lane, Chris, **3D Orthognathic Surgery Simulation Using** - Seminars in Orthodontics, Vol 15, No 1 (March), 2009: pp 48-56, Published by Elsevier Inc.

Schroeder, W; Martin, K. M. Lorensen, W. E., **The visualization toolkit (2nd ed.): an object-oriented approach to 3D graphics: Prentice-Hall, Inc., 1998.**

SERAGUSA, F. - Folha Online Home Page, disponível em
<<http://www1.folha.uol.com.br/fofha/ciencia/ult306u585704.shtml>> - Acesso em: 15 de julho de 2010.

TIOBE Software – disponível em:
<<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>> - Acesso 21 Novembro, 2013

van Loon, J.A.W. **A new method in demo-facial orthopedic, Parts 1 and 2. Dent Cosmos** 1915;57:1093-1101,1229-1235

Visualization ToolKit. - disponível em <<http://www.vtk.org/>>. – Acesso em 19 de dezembro 2011

Volume Viewer - disponível em <<https://volumeviewer.kenai.com/>>. Acesso em 1 Novembro 2013

ZAGOTTIS, D. L., “**Técnica, tecnologia e engenharia: conceituação**”, Notas de Aula, Escola Politécnica da USP, São Paulo, SP, 1987.

APÊNDICE A

TERMO DE CONSENTIMENTO

Eu, _____,
 RG _____ CPF _____, abaixo
 assinado, concordo em participar do questionário abaixo, referente ao trabalho de mestrado
 sobre preparação para cirurgias ortognáticas.

Esclareço que fui devidamente informado pelo pesquisador a respeito do projeto e os
 procedimentos nele envolvidos, estando ciente que minha colaboração acadêmica restringe-
 se ao trabalho de mestrado envolvendo Giuliano Diniz de Moraes, sem que isto traga
 qualquer responsabilidade pelo conteúdo fornecido ou compromisso com a qualidade do
 trabalho.

_____, ____ de _____ de 2013

Área de atuação: _____ Empresa: _____
 Tempo de atuação na área específica: _____

1) No seu entendimento, quais procedimentos poderão ser utilizados através desse
 software:

- ☐ Disjunção mandibular e maxilar
- ☐ Impactação de Maxila
- ☐ Redução do prognatismo mandibular ou maxilar
- ☐ Redução do retrognatismo mandibular ou maxilar
- ☐ Avanço de mento
- ☐ Tratamento de fraturas Lefort I, II e III
- ☐ Correção de desvio de septo

2) Da forma como o software se encontra:
 É possível utilizá-lo para preparação do procedimento cirúrgico?

- ☐ sim ☐ parcialmente ☐ não

3) É possível utilizá-lo para projetar como será o resultado do procedimento cirúrgico?

- ☐ sim ☐ parcialmente ☐ não

4) Qual seria a sua intenção ao utilizar esse software?

- ☐ preparação pré-cirurgia
- ☐ projeção do resultado
- ☐ educativo
- ☐ não vejo utilidade
- ☐ outros. Quais? _____

5) Você acredita que esse software também pode ter valor para seus pacientes ? Explique

6) Possuindo o software, a capacidade de apresentar a pele (tecido mole) do paciente modificada, refletindo as modificações realizadas na estrutura óssea, tais como quebra e movimentação. Seria essa característica relevante ou fundamental para você ? Explique.

☐ sim ☐ parcialmente ☐ não

7) Softwares similares possuem como pré-requisitos um cadastro prévio com um conjunto grande de informações do paciente, muitas vezes essas informações não estão relacionadas diretamente com a cirurgia. Você pensa que esses cadastros são essenciais também para esse software?

8) Você conhece ou já utilizou outro software que faça algo parecido? Quais?

9) Qual característica você gostou ?

10) E quais não gostou ?

11) Você tem interesse em usar o software apresentado pelo pesquisador da forma como esta ?

☐ sim ☐ não

Em caso, negativo. Por que ?

12) Considerando que você não utiliza nenhum software para esses propósitos, você compraria algum software com essas funcionalidades ou similares ?

☐ sim ☐ não

13) Em caso negativo, você estaria disposto a utilizar um software gratuito ?

☐ sim ☐ não

14) O que você sugere para o *Layout* do programa?

- ☐ Permanecer com a interface atual.
 - ☐ Inserir botões (ícones) de comando na lateral
 - ☐ Roteiro de instrução (HELP)
 - ☐ Modificar as cores existentes
 - ☐ Outros. _____
-

15) O que você poderia sugerir para esse software se tornar melhor ?

16) De forma geral, em uma escala de 1 a 5, onde 1 é pouco satisfeito e 5 muito satisfeito.
Como você avalia o software ?
☐1 ☐2 ☐3 ☐4 ☐5

Assinatura