

UNIVERSIDADE FEEVALE

FERNANDO HERMES HERNANDEZ LUSARDO

DESENVOLVIMENTO DE UM PROTÓTIPO VISUALIZADOR 3D  
PARA AUXÍLIO NA ÁREA DA EDUCAÇÃO E MEDICINA

Novo Hamburgo

2013

FERNANDO HERMES HERNANDEZ LUSARDO

DESENVOLVIMENTO DE UM PROTÓTIPO VISUALIZADOR 3D  
PARA AUXÍLIO NA ÁREA DA EDUCAÇÃO E MEDICINA

Trabalho de Conclusão de Curso apresentado  
como requisito parcial à obtenção do grau de  
Bacharel em Sistemas de Informação pela  
Universidade Feevale

Orientadora: Marta Rosecler Bez  
Co-orientador: Fabrício Henrique Rodrigues

Novo Hamburgo  
2013

## **AGRADECIMENTOS**

Agradeço a todos os que contribuíram de alguma forma para que este trabalho de conclusão fosse possível:

Ao meu amigo Fabrício Henrique Rodrigues, motivador incansável, fez valer a pena os sacrifícios da longa caminhada, a minha orientadora Marta Rosecler Bez, pelo suporte recebido e a dedicação que a transforma num apóstolo do ensino, a minha família que sustentou meus períodos mais difíceis.

Agradeço em especial a Deus que me deu a oportunidade de chegar até aqui e de ter conhecido tantos colegas que compartilharam seus sonhos e caminharam comigo para sermos hoje, melhores do que fomos ontem.

## RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo para visualização de objetos tridimensionais chamado “3Dboard”, no formato de um quadro negro 3D para emprego no ensino e na medicina. Para tanto, é apresentada a bibliografia referente a imagens 3D, as operações mais comumente utilizadas em sistemas de visualização e formas de utilização. As ferramentas analisadas para a proposta do trabalho são apresentadas, destacando-se a Unity, que sendo um motor para jogos. Esse apresentou facilidades para sua implementação. O visualizador foi desenvolvido levando em consideração pré-requisitos como simplicidade, intuitividade e objetividade, sendo apresentado e explicado em detalhes. Ao final, um teste foi realizado, demonstrando que o protótipo foi bem aceito pelos usuários, destacando-se a facilidade de uso e a contribuição que a ferramenta traz para a área do ensino e medicina, marcando alguns pontos para serem resolvidos em trabalhos futuros.

Palavras-chave: Imagens 3d. Visualizador 3D. Computação Gráfica. Informática Médica. Informática na Educação.

## **ABSTRACT**

This work presents the development of a prototype for visualization of three-dimensional objects - which was called "3Board" - in the form of a 3D blackboard for use in education - specially medical education. It started summarizing some bibliography about the 3D images field, covering the most commonly used operations in visualization systems, its ways of use and tools for developing such kind of application. Then, the viewer was developed, using Unity as the main tool. For the development, it was taken into account requirements such as simplicity, intuitiveness and objectivity. In order to assess how the development met those requirements, a test was performed with 28 volunteer users, showing that the prototype was well accepted, highlighting the ease of use, however pointing some issues to be solved in future works.

**Key words:** 3D Images. 3D Browser. Computer Graphic. Medical Informatics. Information Technology in Education.

## LISTA DE FIGURAS

Figura 1 – A Lição de Anatomia do Dr. Tulp (1632) Fonte: (UNIVERSITÁRIA, 2007)	13
Figura 2 – O plano $x y$ , $\mathbb{R}^2$ , e o ponto $(a, b)$ Fonte: (BUSS, 2003)	16
Figura 3 – Coordenadas nos eixos $\mathbb{R}^3$ , representando o ponto $(a, b, c)$ Fonte: (BUSS, 2003)	16
Figura 4 – Regra da mão direita para ver a polaridade do eixo $z$ num objeto 3D Fonte: do Autor	17
Figura 5 – Princípio de triangulação dos scanners 3D Fonte: (VICEDO, LINARES, 2010)	19
Figura 6 – Scanner Konica Minolta VI-910RV Fonte: (VICEDO, LINARES, 2010)	20
Figura 7 – Objeto modelado, mostrado nos eixos $\mathbb{R}^3$ Fonte: Imagem do coração adaptada de (SYSTEMS, 2011)	21
Figura 8 – Definição dos três ângulos de Euler em relação aos eixos $X, Y$ e $Z$ Fonte: Adaptado de: (AZEVEDO, CONCI, 2003)	23
Transformações retiradas de (GONZALES, WOODS, 2007)	24
Figura 9 – Coordenadas dos pontos projetados em perspectiva. Fonte: (AZEVEDO, CONCI, 2003)	25
Figura 10 – Coordenadas da posição da câmera e seus sete graus de liberdade: localização no espaço $(x, y, z)$ , ângulos de rotação em torno de cada um dos eixos (setas curvas) e foco. Fonte: Adaptado de (AZEVEDO, CONCI, 2003)	27
Figura 11 – Exemplo de renderização em OpenGL utilizando GLUT Fonte: (LAMARCHE, 2010)	30
Figura 12 – Código C++ usando biblioteca OpenGL Fonte: (WRIGHT, LIPCHAK, HAEMEL, 2007)	31
Figura 13 – Saída produzida pelo código apresentado na figura 12 Fonte: (WRIGHT, LIPCHAK, HAEMEL, 2007)	32
Figura 14 – a) pontos, retas e polígonos, b) pontos estruturados, c) malha não estruturada, d) malha estruturada. Fonte: (CARRANZA, FLORIAN, 2006)	34

Figura 15 – Código C# para visualização de uma esfera com o VTK Fonte: (KITWARE, 2008)	35
Figura 16 – Saída apresentada pelo código acima, com o uso de VTK. Fonte: PrintScreen da saída do programa	36
Figura 17 – Modelo conceitual da arquitetura WebGL Fonte: (GONZÁLEZ, 2012)	38
Figura 18 – Ambiente gráfico para desenvolvimento da Unity	40
Figura 19 – Ambiente gráfico para desenvolvimento da Unity	41
Figura 20 – Verificação de compatibilidade de hardware com o sistema BioDigitalHuman™. Fonte: (SYSTEMS, 2011)	44
Figura 21 – Visualização do rim direito com a descrição científica ao seu lado. Fonte: (SYSTEMS, 2011)	45
Figura 22 – Imagem 3D montada no Google body da seleção no ícone lateral. Fonte: (GOOGLE, 2012)	47
Figura 23 – Trecho de código utilizado para testar a rotação do objeto 3D usando Quaternion.	56
Figura 24 – Trecho de código adotado no protótipo, utilizando a função Rotate da classe Transform.	57
Figura 25 – Exemplo de tela capturada do “3Dboard” exemplificando a rotação nos eixos XY.	58
Figura 26 – Exemplo de tela capturada do “3Dboard” exemplificando a rotação no eixo Z+.	59
Figura 27 – Representação dos botões utilizados no “3Dboard” para movimentar o objeto.	60
Figura 28 – Exemplo de tela capturada do “3Dboard” exemplificando a movimentação no eixo X-.	61
Figura 29 – Botões para aproximação e afastamento do objeto.	62
Figura 30 – Exemplo de tela capturada do “3Dboard” exemplificando o afastamento o objeto usando o botão.	62
Figura 31 – Exemplo de tela capturada do “3Dboard” exemplificando a aproximação do objeto usando o botão.	62

Figura 32 – Botões representativos da rotação no eixo Z. _____	64
Figura 33 – Botões utilizados no “3Dboard” para representar a aproximação e o afastamento do objeto. _____	65
Figura 34 – Configurações para a construção do Deploy para o “3Dboard” Android. _____	65
Figura 35 – Configurações para a construção do Deploy para o “3Dboard” PC-Web. _____	66
Figuras 36 (esquerda) e 37 (direita) – Exemplo das configurações do engine para o “3Dboard” para smartphone - Android e para PC-Web. _____	67
Figura 38 – Gráfico representativo da afinidade com a internet e dispositivos móveis dos voluntários que realizaram o teste do protótipo. _____	75
Figura 39 – Gráfico representativo da afinidade com a internet e dispositivos móveis dos voluntários que realizaram o teste do protótipo. _____	76
Figura 40 – Resultado da avaliação geral do teste 1 _____	76
Figura 41 – Avaliação sobre o carregamento dinâmico dos objetos no teste 1 (agrupando-se as avaliações sobre os grupos de atividades I e IV). _____	77
Figura 42 – Resultado para a pergunta sobre a iluminação no teste 1. _____	78
Figura 43 – Avaliação sobre rotação dos objetos no teste 1 _____	79
Figura 44 – Avaliação sobre a movimentação dos objetos no teste 1. _____	81
Figura 45 – Avaliação sobre o afastamento dos objetos no teste 1. _____	82
Figura 46 – Resultado da avaliação geral do teste 2 _____	84
Figura 47 – Avaliação da iluminação na versão para dispositivos móveis. _____	85
Figura 48 – Avaliação sobre a rotação dos objetos no teste 2. _____	86
Figura 49 – Avaliação da atividade de contagem (instrução 24) que testa, entre outras coisas, a rotação do objeto no teste 2. _____	86
Figura 50 – Avaliação da movimentação do objeto no teste 2. _____	87
Figura 51 – Avaliação do afastamento e da aproximação do objeto no teste 2. _____	88



## LISTA DE ABREVIATURAS E SIGLAS

API	<i>Application Programming Interface</i>
AUX	Biblioteca Auxiliar
CCD	<i>Charge Coupled Device</i>
CPU	<i>Central Processing Unit</i>
DICOM	<i>Digital Imaging and Communications in Medicine</i>
DOM	<i>Document Object Model</i>
FOV	<i>Field of View</i>
GLUI	<i>Graphic Library User Interface</i>
GLUT	<i>Graphic Library Utility Toolkit</i>
GNUGPL	<i>General Public License</i>
GUI	<i>Graphical User Interface</i>
HTML	<i>Hyper Text Markup Language</i>
OpenGL	<i>Open Graphic Library</i>
PC	<i>Personal Computer</i>
PDA	<i>Personal Digital Assistant</i>
PSD	<i>Position Sensitive Device</i>
RGBA	<i>Red, Green, Blue, Alfa</i>
SO	Sistema Operacional
TI	Tecnologia da Informação
URL	<i>Uniform Resource Locator</i>
VTK	<i>Visualization Toolkit</i>
WebGL	<i>Web Graphic Library</i>
XHTML	<i>eXtensible Hypertext Markup Language</i>
XML	<i>eXtensible Markup Language</i>

## SUMÁRIO

<b>INTRODUÇÃO</b>	<b>12</b>
<b>1 IMAGENS 3D</b>	<b>15</b>
1.1 Sistema de coordenadas	15
1.2 Aquisição de imagens 3D	17
1.2.1 Digitalizador Tridimensional	17
1.2.2 <i>Scanners</i> Tridimensionais	18
1.3 Modelagem	20
1.4 Transformações	21
1.5 Câmera virtual	26
<b>2 FERRAMENTAS PARA DESENVOLVIMENTO DE APLICAÇÕES VISUAIS 3D</b>	<b>28</b>
2.1 OpenGL	28
2.1.1 Características gráficas	28
2.1.2 Extensões de OpenGL	30
2.2 A VTK	32
2.2.1 Classes gráficas	32
2.2.2 Modelo gráfico	33
2.2.3 Arquitetura do VTK	34
2.3 WebGL	36
2.4 A ferramenta Unity	38
<b>3 TRABALHOS CORRELATOS</b>	<b>43</b>
3.1 BioDigitalHuman™	43
3.2 Healthline Body Maps	45
3.3 Google Open Sources Zygote 3D Human Body Viewer	46
<b>4 PROPOSTA DA FERRAMENTA</b>	<b>48</b>
<b>5 IMPLEMENTAÇÃO DO PROTÓTIPO</b>	<b>51</b>
5.1 Carregamento dinâmico do objeto	52
5.2 Iluminação sobre o objeto	54
5.3 Rotação do objeto	55
5.4 Movimentação do objeto	59
5.5 Aproximação ou afastamento do objeto	61
5.6 Interface intuitiva	63
5.7 Deploy	65

<b>6 RESULTADOS E AVALIAÇÃO</b>	<b>68</b>
6.1 Projeto dos testes	68
6.2 Avaliação dos resultados para o roteiro 1 (versão para PC/Web)	76
6.2.1 Avaliação do requisito ‘a’ – Carregamento dinâmico do objeto	77
6.2.2 Avaliação do requisito (b) – Iluminação do objeto	78
6.2.3 Avaliação do requisito ‘c’ – Posicionamento do objeto à frente da câmera.	79
6.2.4 Avaliação do requisito ‘d’ – Rotação do objeto	79
6.2.5 Avaliação do requisito ‘e’ – Movimentação do objeto	81
6.2.6 Avaliação do requisito ‘f’ – Aproximação e afastamento do objeto	82
6.2.7 Avaliação do requisito ‘g’ – Utilizar os gestos de interface difundidos, elementos gráficos intuitivos e opções de hardware.	83
6.3 Avaliação dos resultados para o roteiro 2 (versão para dispositivos móveis).	84
6.3.1 Avaliação do requisito ‘a’ – Carregamento dinâmico do objeto	84
6.3.2 Avaliação do requisito ‘b’ – Iluminação do objeto	85
6.3.3 Avaliação do requisito ‘c’ – Posicionamento do objeto em frente à câmera	85
6.3.4 Avaliação do requisito ‘d’ – Rotação do objeto	86
6.3.5 Avaliação do requisito ‘e’ – Movimentação do objeto	87
6.3.6 Avaliação do requisito ‘f’ - Aproximação e afastamento do objeto	88
6.3.7 Avaliação do requisito ‘g’ - Utilizar os gestos de interface, elementos gráficos intuitivos e opções de hardware	89
<b>CONCLUSÃO</b>	<b>90</b>
<b>REFERÊNCIAS BIBLIOGRÁFICAS</b>	<b>94</b>
<b>APÊNDICE I – FORMULÁRIO DE TESTE</b>	<b>97</b>

## INTRODUÇÃO

Os seres humanos aprendem observando e essa é uma característica que nos acompanha desde a infância. Isso é visível no uso de desenhos de fatias de pizza ou palitinhos para que a visualização ajude no aprendizado de frações. O quadro negro ou quadro de giz, desde sua invenção, pelo professor escocês James Pillans (BUARQUE, 2007), é usado pelos educadores para reforçar seus ensinamentos com figuras e desenhos, exemplificando graficamente alguns conteúdos para ajudar os alunos a fixarem o conhecimento. Muitas vezes, usa-se o desenho para representar coisas que não são possíveis de se ter ao alcance. Enfim, a visualização sempre ajudou o homem no aprendizado de inúmeras disciplinas.

Um exemplo disso é o fato de, durante séculos, o uso de enciclopédias terem sido uma prática corrente no aprendizado. Nesse tipo de material, os alunos podem, apoiados pela visualização de fotografias e outras figuras, fixar o conhecimento recebido em aula ou construí-lo em estudos individuais.

No entanto, em certos domínios, a observação num plano bi-dimensional pode representar uma falta de informação considerável. Há casos em que a visualização do objeto de estudo em três dimensões é um acréscimo interessante. Isto pode ser visto no ensino da Medicina – mais especificamente, da anatomia humana. Tradicionalmente, o estudo dessa disciplina envolve o uso de órgãos humanos oriundos de cadáveres de pessoas cujos corpos não foram reclamados ou foram doados, no intuito de confrontar o aluno com a visualização do objeto em estudo.

A Figura 1 apresenta o interesse pela visualização como auxílio no aprendizado, no estudo da anatomia humana, retratado magistralmente pelo grande Artista Rembrant.



Figura 1 – A Lição de Anatomia do Dr. Tulp (1632) Fonte: (UNIVERSITÁRIA, 2007)

Entretanto, segundo Arger (2009), a falta de cadáveres para o ensino de anatomia tem sido um grande problema. O fato de haver uma atmosfera de medos, constrangimentos e receios em torno disso, faz alunos e professores enfrentarem muitos desconfortos no processo de ensino-aprendizagem (QUEIROZ, 2005). Nesse sentido, a Doutora Maryelena Seleme Dora (2003), professora de anatomia do Departamento de Ciências Morfobiológicas da Fundação Universidade Federal do Rio Grande, aconselha o uso de imagens obtidas através da técnica da ultra-sonografia para a exploração do corpo humano sem o efeito da deformação que a dissecação de cadáveres e a decomposição impõe aos órgãos manipulados.

Ainda, ao que parece, essa preocupação está bem disseminada na área, tendo havido avanços notáveis para a solução desse tipo de problema. Um exemplo é o Atlas do Corpo Humano em 3D BioDigitalHuman™ (SYSTEMS, 2011), gratuito, que permite ao aluno uma navegação tridimensional pelos diversos sistemas e órgãos do corpo humano. Nele é possível selecionar um modelo humano feminino ou masculino, observar detalhadamente a anatomia, movimentando-se entre os sistemas esquelético, muscular, visceral, nervoso, linfático e outros, permitindo ainda visualizar alguns em funcionamento, bem como, identificar algumas patologias.

Netter's 3D Interactive Anatomy (CYBER, 2010) é outro bom exemplo. Trata-se de um *software* pago, muito similar ao BioDigitalHuman™, que permite também a livre navegação pelos sistemas do corpo humano ou a parte selecionada para estudo. Seu tutor virtual vai ensinando o usuário os detalhes sobre o item em destaque, cuja visualização se dá no ângulo e rotação escolhido no manuseio do *mouse*.

Outros esforços também estão sendo feitos com um enfoque um pouco diferente, utilizando realidade virtual – uma área do conhecimento que vem trazer muitas oportunidades para a investigação científica. Nesse sentido, seu uso para treinamento de procedimentos médicos têm sido alvo de pesquisas em todo o mundo (WESTWOOD et al., 1999/2000/2001/2002) apud (MACHADO, 2003). Os trabalhos de BioDigitalHuman™ e o Netter's 3D Interactive Anatomy, citados anteriormente, estão procurando preencher a necessidade de observação tri-dimensional.

Como já foi dito, a visualização é importante no estudo e, em algumas situações, se for feita com recursos 3D, tem-se um valioso acréscimo de significado para o observador. Um exemplo notório é a medicina, cujos recursos para visualização 3D tradicionalmente consistem em exemplares de órgãos reais – o que é limitado, pois acarreta dificuldades para a obtenção dessas peças. Assim, esforços têm sido realizados no sentido de criarem-se alternativas computacionais para contornar tais dificuldades. Isso demonstra que essa é uma área de pesquisa atual e em franca evolução. Com isso, trabalhos que contribuam para essa evolução têm grande valor. Ainda, com o advento de novas tecnologias para tratamento de imagens, tais como as bibliotecas gráficas OpenGL, WebGL e VTK e ferramentas como o Unity, a possibilidade de contribuição torna-se bastante acessível.

Com isso, este trabalho apresenta os esforços no sentido de construir um protótipo de visualizador 3D, de fácil utilização, que possa complementar o que vem sendo realizado pela comunidade que utiliza a computação gráfica para o auxílio visual na educação. Esses podem ser provenientes da área médica, arqueológica, ou de qualquer outra área que considere relevante a observação tri-dimensional com o objetivo exploratório.

Este trabalho está dividido em sete capítulos. No primeiro explora-se o universo das imagens 3D. No capítulo dois é desenvolvido um estudo sobre as bibliotecas OpenGL, WebGL e VTK e da ferramenta Unity, abrangendo suas características e virtudes, focando, principalmente no seu aproveitamento para o objetivo do presente trabalho. No terceiro capítulo são apresentados trabalhos correlatos, suas características e os aspectos nos quais o trabalho aqui apresentado pode vir a complementar ou melhorar o que já existe. No quarto capítulo, apresenta-se uma proposta de ferramenta para auxiliar na solução do problema identificado neste trabalho, bem como os aspectos do desenvolvimento de seu protótipo. O quinto capítulo expõe a implementação do protótipo. O sexto capítulo traz os resultados do trabalho, oriundos da avaliação do uso do protótipo. Por fim, no sétimo capítulo, são apresentadas as conclusões e trabalhos futuros.

## 1 IMAGENS 3D

Falar em imagens 3D lembra a reflexão sobre o que se entende como percepção tridimensional. Como o nome sugere, tridimensional refere-se a três dimensões que, no plano físico, são representadas pela altura, largura e profundidade.

De acordo com Azevedo e Conci (2003), o ser humano percebe a espacialidade de uma imagem pela distinção da sua forma, cor, textura e pela relação espacial que o objeto apresentado guarda com outros conhecidos do mundo real.

Segundo esses mesmos autores, existem três categorias de estímulos visuais pelos quais o cérebro reconhece uma imagem em três dimensões:

- Informações Monoculares, que são as noções de perspectiva linear ou conhecimento prévio do objeto, a densidade das texturas e o volume sugerido pela reflexão da luz e as sombras sobre ele;
- Informações Visuais óculo-motoras, alimentadas pela movimentação dos olhos através da musculatura responsável e que trazem ao cérebro detalhes sobre a distância dos objetos apurada pelas alterações focais;
- Informações visuais estereoscópicas, que se produz pela dualidade das imagens obtidas pelos olhos, cuja separação produz a chamada disparidade binocular.

Tendo isso em consideração, para representar a tridimensionalidade de um objeto na tela do computador, visto que a mesma estampa as imagens num plano bidimensional, é usada a noção de perspectiva, ou seja, a manutenção da proporcionalidade dos objetos em referência a distância que estes se encontram no cenário e o ângulo de visão do observador.

Este capítulo abordará o sistema de coordenadas que estabelece como se relaciona o mundo em três dimensões, a aquisição de dados 3D (i.e. processo pelo qual se obtém informação tridimensional a partir de objetos do mundo real), a modelagem, que permite a criação de objetos tridimensionais e as transformações, sem as quais não seria possível a movimentação e rotação dos objetos tridimensionais.

### 1.1 Sistema de coordenadas

Quando se representa formas graficamente, usa-se o sistema de coordenadas para determinar seu posicionamento. Por exemplo, para representar um triângulo em termos do seu

posicionamento, indica-se o ponto correspondente nas coordenadas  $(x, y)$  para cada um de seus vértices. No plano bidimensional, também referenciado como  $\mathbb{R}^2$ , usa-se convencionalmente o eixo  $x$  horizontal apontando para a direita para representar a largura e o eixo  $y$  vertical apontando para cima representa a altura (BUSS, 2003).

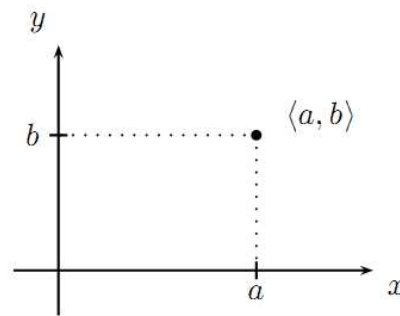


Figura 2 – O plano  $x, y$ ,  $\mathbb{R}^2$ , e o ponto  $(a, b)$  Fonte: (BUSS, 2003)

Em representações tridimensionais ( $\mathbb{R}^3$ ), acrescenta-se o eixo  $z$ , perpendicular ao plano  $xy$ , convencionalmente apontando em direção ao observador. Nelas, o posicionamento é representado por pontos  $(a, b, c)$ , com  $a, b$  e  $c$  informando as coordenadas no eixo  $x, y, z$ , conforme apresentado na Figura 3 a seguir.

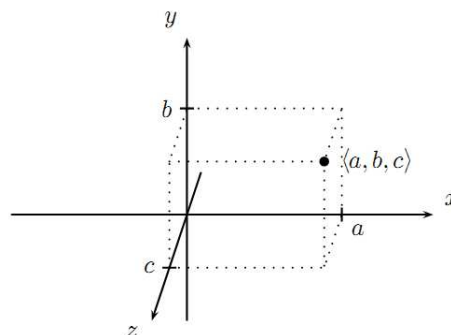


Figura 3 – Coordenadas nos eixos  $\mathbb{R}^3$ , representando o ponto  $(a, b, c)$  Fonte: (BUSS, 2003)

Os sistemas 3D podem ser positivos ou negativos. Seus eixos podem assumir diferentes posições, não necessariamente as convencionais. Então, mesmo considerando o eixo  $x$  como horizontal e o  $y$  como vertical, seu terceiro eixo ( $z$ ) pode apontar para duas direções. A direção positiva será a que obedecer a regra da mão direita. Tal regra consiste em estender a mão direita aberta, fazendo coincidir o indicador com o eixo  $y$  e o polegar com o eixo  $x$ . Se a palma da mão apontar para o eixo  $z$ , o sistema de eixos é positivo, caso contrário, negativo (Figura 4).



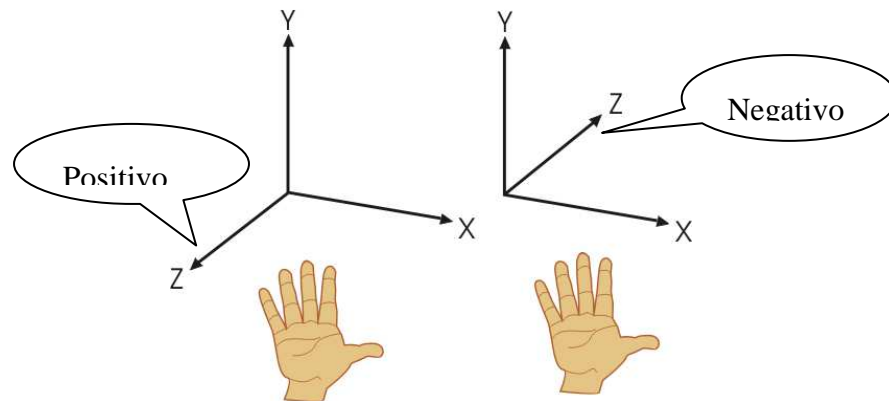


Figura 4 – Regra da mão direita para ver a polaridade do eixo z num objeto 3D Fonte: do Autor

Para poder representar qualquer forma de maneira não ambígua em um sistema de coordenadas, esse deverá ter origem e uma orientação que determine seu posicionamento em relação aos eixos. Nesse sentido, todo objeto sólido que for representado no sistema cartesiano tridimensional  $\mathbb{R}^3$ , vai formar uma nuvem de pontos cuja origem se dá na interseção dos três eixos ( $x$ ,  $y$ ,  $z$ ) e cujos valores variarão positivamente quando mais se afastarem da origem no sentido apontado pelos eixos.

## 1.2 Aquisição de imagens 3D

Um modelo 3D virtual é a representação numérica de um objeto de tal forma que permita sua representação posterior (FEITO ET AL, 2006).

Com o avanço da tecnologia surgem cada vez mais dispositivos de entrada 3D, com os quais pode-se propiciar interação dentro de ambientes virtuais tridimensionais. Ainda em concordância com Azevedo e Conci (2003), entre tais dispositivos pode-se destacar:

### 1.2.1 Digitalizador Tridimensional

É um tipo de braço mecânico em cuja ponta um sensor informa as coordenadas de um determinado ponto de um objeto qualquer, que tenha sido detectado pelo sensor. O ponto é identificado, em um sistema cartesiano de três dimensões, para que o computador vá montando a imagem através de uma aglomeração destes pontos devidamente situados no sistema de coordenadas (i.e.  $x$ ,  $y$  e  $z$ ) (VICEDO, LINARES, 2010) conforme será explicado na sequência.

### 1.2.2 *Scanners* Tridimensionais

Existem dois tipos de *scanner* 3D: os que medem por contato físico com o objeto e os que não fazem uso desse contato. Os *scanners* de contato físico são muito precisos e bem lentos quando comparados aos sem contato. A grande desvantagem destes é a possibilidade de danificar as peças medidas pela necessidade do contato físico para a captura da grande quantidade de pontos realizados na varredura (VICEDO, LINARES, 2010). Já entre as diversas tecnologias de *scanners* sem contato disponíveis no mercado, podem-se destacar dois grupos: as tecnologias mais baratas, que utilizam câmeras digitais para fornecerem as coordenadas ao sistema, e os sistemas mais caros, que utilizam laser.

Independente do modelo escolhido para a captação de imagens tridimensionais, o resultado será uma nuvem de pontos 3D que representem o objeto obtido pelo *scanner* (FEITO ET AL, 2006).

A partir de diversas varreduras, é possível obter uma nuvem de pontos completa sobre o objeto a reproduzir, após o qual devem ser reconstruídas as superfícies para cobrir falhas oriundas da captura.

Uma classe de *scanners* baseia sua aferição no tempo de viagem do feixe de luz emitido, medindo a distância através do tempo que a luz demora no trajeto de ida e volta. O telêmetro *laser*, usado para medir distâncias, detecta somente a que existe no ponto da direção do *laser*. Com isso, obtém-se todo o campo de visão ponto a ponto, movimentando-se o telêmetro ou com auxílio de espelhos que direcionem os feixes (VICEDO, LINARES, 2010).

Ainda em conformidade com Vicedo e Linares (2010), outra classe de *scanners* mede a distância do ponto por triangulação. Essa utiliza uma câmera para localizar a posição do ponto *laser*. Em função da distância, o *laser* aparece em diferentes posições no campo de visão da câmera. Esse processo chama-se de triangulação porque o feixe do emissor *laser* e a câmera forma um triângulo. A figura 5 exemplifica o princípio de triangulação utilizado:

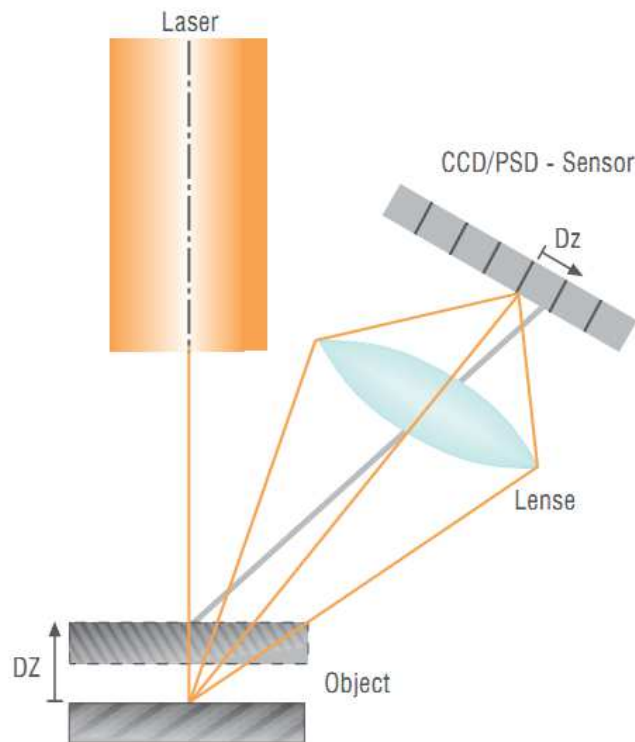


Figura 5 – Princípio de triangulação dos scanners 3D Fonte: (VICEDO, LINARES, 2010)

O comprimento de um lado do triângulo e a distância entre o emissor *laser* e a câmera são conhecidos, bem como o ângulo formado pelos dois lados adjacentes ao emissor. Finalmente, o ângulo dos lados adjacentes à câmera pode ser determinado observando a posição do ponto do *laser* no campo de visão da câmera. Com estas informações pode-se determinar a forma e tamanho do triângulo e a distância em que se encontra o objeto.

Uma terceira classe de *scanners* utiliza luz estruturada, projetando um padrão sobre o objeto e medindo a deformação do mesmo com uma câmera e uma técnica similar à de triangulação. A vantagem desse tipo de *scanner* é a velocidade: em lugar de obter um ponto de cada vez, podem-se capturar múltiplos pontos, com a possibilidade de pegar todo o campo de visão em uma única captura. Existem *scanners* deste tipo que são capazes até de captar objetos em movimento em tempo real (VICEDO, LINARES, 2010).

A Figura 6 a seguir mostra um scanner sem contato, que utiliza a técnica de triangulação:



Figura 6 – Scanner Konica Minolta VI-910RV Fonte: (VICEDO, LINARES, 2010)

A nuvem de pontos que se obtém como saída do *scanner* normalmente é convertida em polígonos para unir os pontos adjacentes numa superfície homogênea, processo conhecido como reconstrução ou modelagem de acabamento (VICEDO, LINARES, 2010).

Há no mercado, ainda, dispositivos como luvas, capacetes, 3D *controllers* e roupas (AZEVEDO, CONCI, 2003), com diferentes objetivos que não serão abordados no presente trabalho.

### 1.3 Modelagem

O processo de criação para imagens 3D pode ser dividido em três fases básicas:

- modelagem;
- configuração e *layout* do ambiente onde os objetos são modelados;
- apresentação ou visualização.

A modelagem pode ser descrita como a formatação de objetos que serão utilizados posteriormente num ambiente virtual chamado de cena, edição da superfície e propriedades do material modelado, luminosidade, sombras, reflexos, transparência ou opacidade, texturas e demais detalhes para a sua posterior visualização. A configuração e *layout* da cena envolvem a iluminação, câmera, ponto de vista do observador, entre outros e a apresentação ou visualização permite a manipulação do objeto selecionado e sua observação de diferentes pontos de vista e ângulos, como apresentado na Figura 7 a seguir:

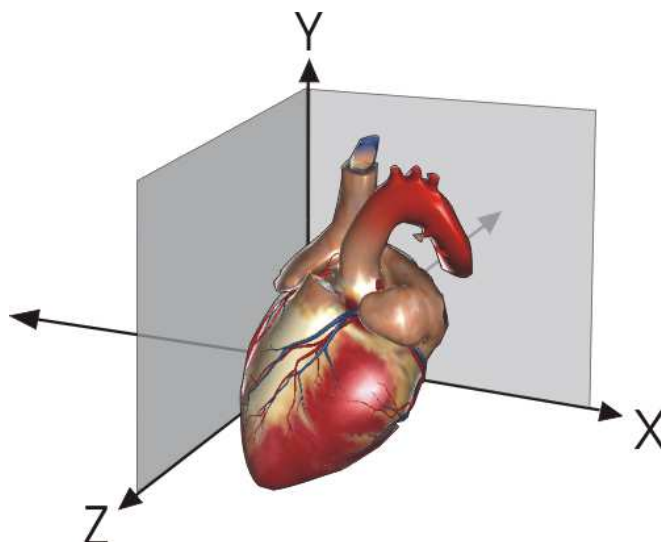


Figura 7 – Objeto modelado, mostrado nos eixos  $\mathbb{R}^3$  Fonte: Imagem do coração adaptada de (SYSTEMS, 2011)

Além da pura criação de imagens, alguns dos métodos de captura apresentados também requerem a intervenção de modeladores para a obtenção de um melhor acabamento na peça digitalizada.

Atualmente, existem diversos *softwares* para auxiliar no processo de modelagem. Entre os mais populares, é possível destacar o Blender (ROSENDAAL ET AL, 2004) e o Maia (AUTODESK, 2012) que permitem a modelagem de objetos 3D e o acabamento das peças modeladas em perfeita harmonia com o cenário montado e iluminação escolhida.

Blender é um programa que integra uma série de ferramentas para manipulação de conteúdos 3D, mantido como “*Software Livre*” com o código fonte disponível sob a licença GNU GPL. O programa pode ser utilizado para criar visualizações 3D, imagens estáticas ou vídeos de alta qualidade, inclui modelagem, mapeamento, texturização, animação, entre outros (ROSENDAAL et al., 2004).

O Maia 3D, é um software proprietário cuja característica é a de facilitar tarefas de criação, modelagem, visualização e composição de ambientes gráficos 3D, sendo muito utilizado em filmes, jogos, televisão e *design* (AUTODESK, 2012).

#### 1.4 Transformações

As transformações desempenham um papel de vital importância no processamento de imagens devido a oferecerem uma aproximação da realidade, refletindo num plano bidimensional o que seria visto no mundo tridimensional (GONZALES, WOODS, 2007).

As transformações geométricas para manipulação de objetos gráficos envolvem muitas operações de aritmética e o uso de matrizes que, por serem mais fáceis de utilizar, tem se tornado o caminho natural. Através de matrizes e de sua multiplicação, é possível representar as transformações lineares 2D e 3D que combinadas resultam numa única matriz de transformação (AZEVEDO, CONCI, 2003).

Uma das grandes utilidades das transformações é permitir a observação de um objeto em diversos ângulos. Para que essa observação seja possível, é necessário girá-lo, o que é feito pela aplicação de uma série de transformações sobre ele. Em concordância com Gonzales e Woods (2007) as principais transformações são:

**Translação:** Seu objetivo é o de movimentar um ponto de coordenadas  $(X, Y, Z)$  para uma nova posição, utilizando o deslocamento  $(X_0, Y_0, Z_0)$  e realizada pelas equações:

$$X^* = X + X_0$$

$$Y^* = Y + Y_0$$

$$Z^* = Z + Z_0$$

Onde  $X^*$ ,  $Y^*$  e  $Z^*$  representam as coordenadas do novo ponto transladado. Essa operação também pode ser expressa de forma matricial como:

$$\begin{bmatrix} X^* \\ Y^* \\ Z^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X^0 \\ 0 & 1 & 0 & Y^0 \\ 0 & 0 & 1 & Z^0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Para facilitar a tarefa, usam-se matrizes quadradas, ficando assim definidas:

$$\begin{bmatrix} X^* \\ Y^* \\ Z^* \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X^0 \\ 0 & 1 & 0 & Y^0 \\ 0 & 0 & 1 & Z^0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

**Mudança de escala:** Para fazer com que uma imagem definida por um conjunto de pontos mude seu tamanho, é necessário multiplicar todos os valores de suas coordenadas (i.e. de cada um dos pontos da imagem) por um fator de escala (AZEVEDO, CONCI, 2003).

No caso de imagens 3D, multiplicam-se cada ponto (representado, para fins da multiplicação, através de  $[X_n, Y_n, Z_n, 1]$ ), pela matriz apresentada a seguir (que representa o fator de escala):

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**Rotação:** Esta transformação permite que os objetos sejam observados de diferentes ângulos. Em 3D é mais fácil realizar esta tarefa individualmente sobre cada um dos eixos, usando os chamados ângulos de Euler (ADORNO, 2007). Cada uma das rotações possíveis pode ser definida a partir da análise de operações realizadas nos planos  $XY$ ,  $YZ$  e  $ZX$ .

Os ângulos de Euler facilitam uma definição apurada das rotações em relação a um sistema de eixos, pois definem a rotação em um plano pelo giro em torno de um vetor normal a esse plano.

Olhando o sistema de eixos mostrado na Figura 8, podem-se definir três ângulos de Euler, em relação aos eixos  $X$ ,  $Y$  e  $Z$ . Um deles define o giro em torno do eixo  $X$  afetando os pontos do plano  $YZ$ . Outro que define o giro em torno do eixo  $Y$  para os pontos do plano  $XZ$ . Por último, tem-se o que define o giro em torno do eixo  $Z$  para pontos no plano  $XY$  (AZEVEDO, CONCI, 2003).

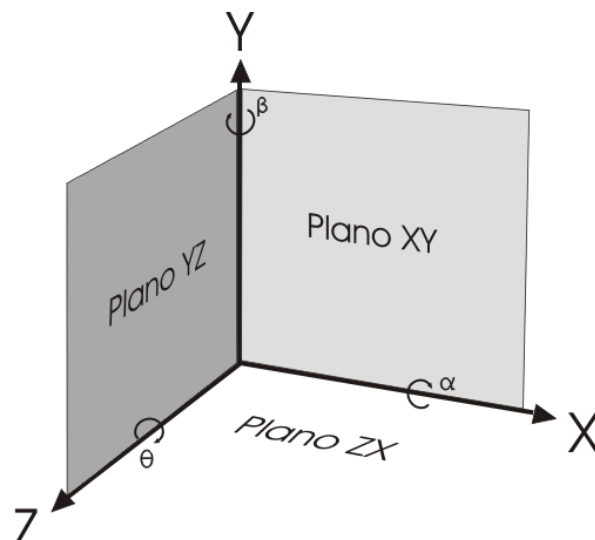


Figura 8 – Definição dos três ângulos de Euler em relação aos eixos  $X$ ,  $Y$  e  $Z$  Fonte: Adaptado de: (AZEVEDO, CONCI, 2003)

Como exposto, a rotação de um ponto no espaço tridimensional pode ser obtida pela multiplicação dos ângulos de rotação em torno dos eixos ao ponto.

No mundo tri-dimensional, têm-se três possíveis matrizes de rotação, com sua forma dependendo do eixo sobre o qual vai ser realizado o movimento.

Exemplificando a dinâmica da Figura 8, a rotação de um ponto em torno do eixo de coordenadas Z, por um ângulo  $\theta$  é realizada usando a transformação:

$$R_{\theta} = \begin{bmatrix} \cos \theta & \text{sen } \theta & 0 & 0 \\ -\text{sen } \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

O ângulo de rotação  $\theta$  é medido no sentido horário olhando para a origem a partir de um ponto sobre o eixo +Z. Por esse motivo, a transformação afeta os valores das coordenadas X e Y como exemplificado na Figura 8.

Analogamente ao eixo Z, para o eixo X, num ângulo  $\alpha$ , usa-se a transformação:

$$R_{\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & \text{sen } \alpha & 0 \\ 0 & -\text{sen } \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finalmente, para o eixo Y, a rotação do ângulo  $\beta$  tem uma matriz de transformação:

$$R_{\beta} = \begin{bmatrix} \cos \beta & 0 & -\text{sen } \beta & 0 \\ 0 & 1 & 0 & 0 \\ \text{sen } \beta & 0 & \cos \beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformações retiradas de (GONZALES, WOODS, 2007)

**Transformações de perspectiva:** Esta é uma transformação dentro do espaço tridimensional cujas projeções representam a cena vista de um ponto de observação a uma distância finita (AZEVEDO, CONCI, 2003).

No raciocínio desenvolvido por Azevedo e Conci (2003), utiliza-se a Figura 9 para exemplificar as transformações de perspectiva.



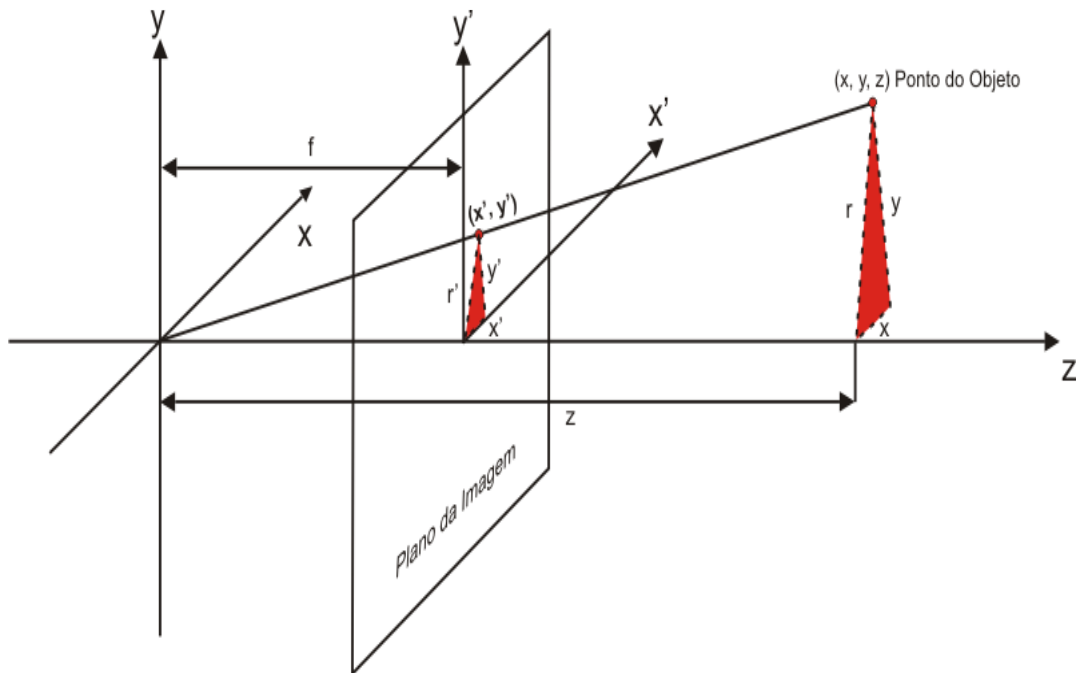


Figura 9 – Coordenadas dos pontos projetados em perspectiva. Fonte: (AZEVEDO, CONCI, 2003)

Assumindo que o centro de projeção coincide com a origem do sistema de eixos, desejando a projeção de um ponto  $(x, y, z)$  qualquer do espaço de coordenadas em um plano  $z=f$ , como mostra a Figura 9, as coordenadas do ponto projetado  $(x', y', z')$ , são obtidas da relação:

$$\frac{f}{z} = \frac{r'}{r}$$

Onde  $f$  é a distância do ponto  $(x', y', z')$  para a origem, no eixo  $Z$ ,  $z$  é a distância do ponto  $(x, y, z)$  para a origem no eixo  $Z$ ,  $r'$  é a altura da projeção e  $r$  a altura do objeto. No caso particular do triângulo tem-se:

$$\begin{aligned} r'^2 &= x'^2 + y'^2 \\ r^2 &= x^2 + y^2 \end{aligned}$$

Sabendo que o triângulo formado pelas coordenadas  $x$  e  $y$  e a distância  $r$ , assim como formado pelas coordenadas  $(x', y')$  e a perpendicular ao eixo  $z$ ,  $r'$ , obtém-se por semelhança:

$$\frac{x'}{x} = \frac{y'}{y} = \frac{r'}{r}$$

Combinando as expressões anteriores obtém-se:

$$\frac{x'}{x} = \frac{f}{z} e \frac{y'}{y} = \frac{f}{z}$$

Assim, as coordenadas  $(x', y', (z=f))$  do ponto no plano da imagem são obtidas pelas equações a seguir:

$$x' = \frac{f}{z} x y' = \frac{f}{z} y$$

De forma matricial, a projeção de um ponto de vista sobre o eixo perpendicular ao plano da imagem é:

$$[x \quad y \quad z \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{f_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} [x' \quad y' \quad z' \quad 1]$$

## 1.5 Câmera virtual

É inevitável, quando se trata de representar um objeto 3D num plano 2D (como efetivamente é o caso da tela que recebe o resultado da imagem escaneada), fazer analogia com a máquina fotográfica. No mundo tridimensional, imagina-se o observador que vê a cena através das lentes da câmera, definindo, além da sua posição, sua orientação, foco, tipo de projeção e posicionamento dos planos que limitam a visibilidade da cena.

O vetor que vai da posição da câmera ao ponto focal, (i.e. para onde ela está apontando), é denominado direção de projeção. A orientação da câmera é controlada pela sua posição  $(x, y, z)$ , seu ponto focal (ponto  $D$  no caso da Figura 10) e pelo vetor que indica o “lado de cima” da cena 3D, parâmetros que definem a câmera (AZEVEDO, CONCI, 2003).

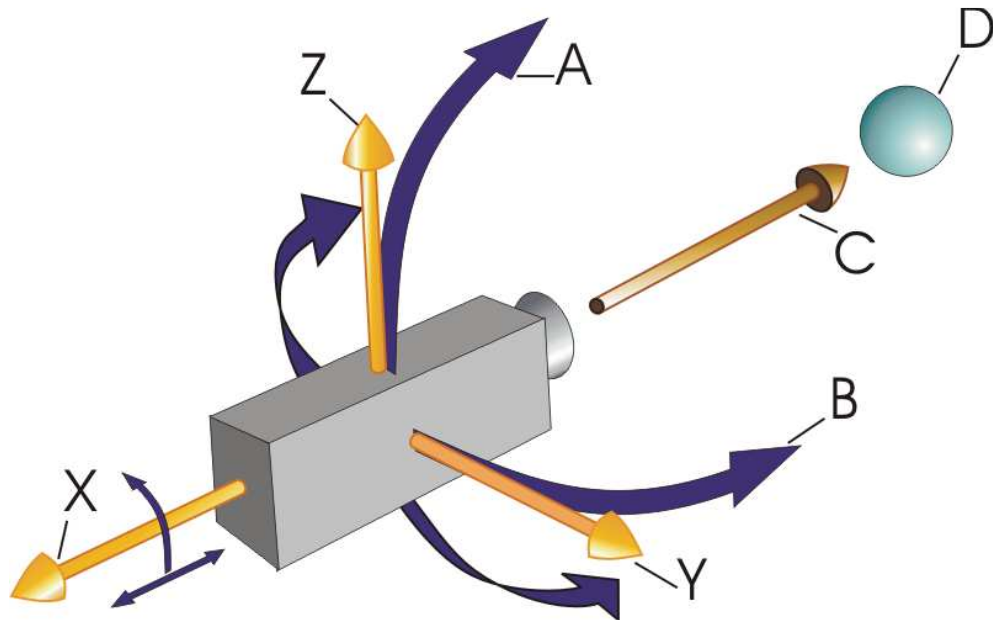


Figura 10 – Coordenadas da posição da câmera e seus sete graus de liberdade: localização no espaço ( $x, y, z$ ), ângulos de rotação em torno de cada um dos eixos (setas curvas) e foco.

Fonte: Adaptado de (AZEVEDO, CONCI, 2003)

Visto o apresentado, é necessário fazer uma revisão de algumas ferramentas disponíveis para construção de aplicações gráficas, cujas características e facilidade de uso possam ser interessantes para este trabalho.

## 2 FERRAMENTAS PARA DESENVOLVIMENTO DE APLICAÇÕES VISUAIS 3D

Tendo como objetivo a criação de um protótipo para observação de imagens 3D, convém falar-se de ferramentas que auxiliem no seu desenvolvimento. Tais ferramentas incluem tanto bibliotecas de *software* como OpenGL, VTK e WebGL – quanto ambientes completos de desenvolvimento – como Unity. Ainda, considerando uma aplicação prática, o enfoque dado a essas ferramentas afasta-se de sua formalização, aproximando-se da perspectiva do desenvolvedor que as utiliza para aplicar nas suas próprias técnicas.

### 2.1 OpenGL

A biblioteca gráfica aberta OpenGL é uma *interface* de *software* desenvolvida para tirar o melhor aproveitamento do *hardware* gráfico. Em essência, é uma API de procedimentos gráficos, com a qual o programador detalha os mecanismos para conseguir uma determinada forma ou efeito, utilizando comandos da biblioteca.

A biblioteca não inclui funções para manipulação de janelas ou interação do usuário, assim como tratamento de entrada e saída de arquivos. Os sistemas operacionais onde a biblioteca será utilizada já têm suas próprias rotinas para tais propósitos que devem ser utilizadas para manipulação da biblioteca gráfica (WRIGHT, LIPCHAK, HAEMEL, 2007).

Do ponto de vista do desenvolvedor, a biblioteca é um conjunto de comandos que permitem especificar objetos geométricos em duas ou três dimensões. Ela oferece funções de mapeamento de texturas dentro de polígonos na cena, que aceleram o processo de representação de objetos.

No seu início, a OpenGL criou a biblioteca AUX para facilitar o aprendizado e escrita dos programas sem ter que se preocupar com os detalhes do sistema operacional e, com o tempo, a biblioteca GLUT veio a substituí-la (WRIGHT, LIPCHAK, HAEMEL, 2007).

Pode-se dizer que a OpenGL é uma biblioteca estilizada de alto rendimento, já que existem muitas placas aceleradoras que implementam suas primitivas a nível de *hardware*.

#### 2.1.1 Características gráficas

A OpenGL permite a descrição matemática de objetos a partir de suas primitivas, organizando-os no espaço, modificando o ponto de vista e as propriedades da câmera. A biblioteca permite calcular a cor dos *píxels* pela designação direta, cálculo de iluminação ou por mapeamento de texturas (CARRANZA, FLORIAN, 2006).

Em relação às suas qualidades, Carranza e Florian (2006) destacam:

- Primitivas geométricas que descrevem matematicamente os objetos, tais como pontos, linhas, polígonos, imagens e mapas de bits;
- Codificação da cor pelo sistema RGBA (*Red, Green, Blue, Alfa*);
- Visualização e modelagem que permitem dispor objetos numa cena tridimensional e movendo a câmera no espaço, selecionar a visualização mais conveniente na composição desejada;
- Mapeamento de textura, aproximando os objetos da realidade pelo desenho da superfície dos modelos;
- Iluminação dos modelos, permitindo o cálculo da cor em qualquer ponto por meio das propriedades do material e as fontes de iluminação da cena;
- *Double buffering* que contribui com a diminuição das piscadas de imagem nas animações;
- Redução de bordas escalonadas nas linhas desenhadas sobre a tela através da modificação da cor e intensidade dos *pixels* próximos a linha afetada;
- Sombreado de *Gouraud*, utilizado para o sombreamento suave de objetos 3D, produzindo uma diferença sutil na cor da superfície dos objetos;
- *Z-Buffering* que mantém os registros da coordenada Z de um objeto 3D, usado para registrar a proximidade do objeto em questão com o observador. Utilizado também para a eliminação de superfícies ocultas;
- Efeitos atmosféricos tais como neblina e fumaça para trazerem mais realismo às cenas modeladas, agregando profundidade à imagem. Sem esses efeitos, as imagens aparecem muito nítidas, tirando seu aspecto real;
- *Alpha blending* que permite combinar a cor do fragmento a ser processado com a do *pixel* que já está no *buffer*. Isso permite simular a transparência do objeto;
- Avaliadores polinômicos que suportam *B-splines* racionais, ou seja, ajudam no desenho de curvas suaves mantendo a referência de alguns pontos. Poupa-se assim a necessidade de acumular grandes quantidades de pontos intermediários;

- Características de *feedback*, seleção e eleição de uma região da tela ou objeto apresentado nela;
- Primitivas de *raster* (*bitmaps* e retângulos de *pixels*);
- Operações com *pixels*;
- Transformações: rotação, escala e perspectiva em 3D;

### 2.1.2 Extensões de OpenGL

GLU (*Graphic Library Utility*) é a sigla utilizada para representar a biblioteca de utilitários do OpenGL, composta por funções de desenho de alto nível que se baseiam nas rotinas primitivas da biblioteca, distribuídas com a extensão. As funções GLU são reconhecidas facilmente já que todas começam com o prefixo “glu”, (e. g. *gluOrtho2D()*, que define uma matriz de projeção de duas dimensões).

GLUT (*OpenGL Utility Toolkit*) é uma API de fácil assimilação, multiplataforma, e prove o usuário com funcionalidades para o manuseio de janelas e interação pelo uso do teclado ou mouse.

GLUI é uma biblioteca de interface com o usuário, escrita em C++. É o acrônimo de “*OpenGL user Interface Library*”. Esta biblioteca oferece elementos de controle como botões, caixas de seleção, *spinners* para as aplicações que utilizem OpenGL. A Figura 11 apresenta uma renderização realizada com funções da biblioteca OpenGL.

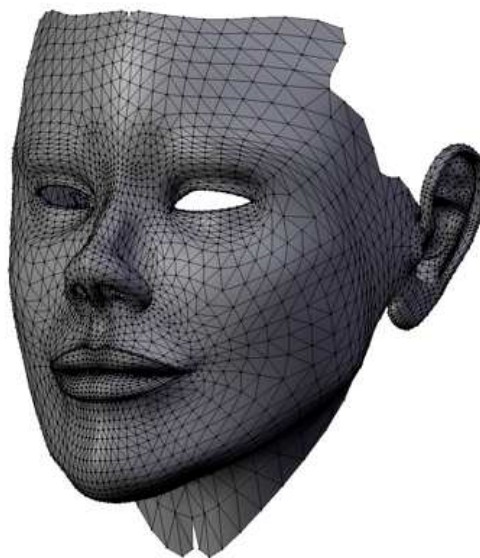
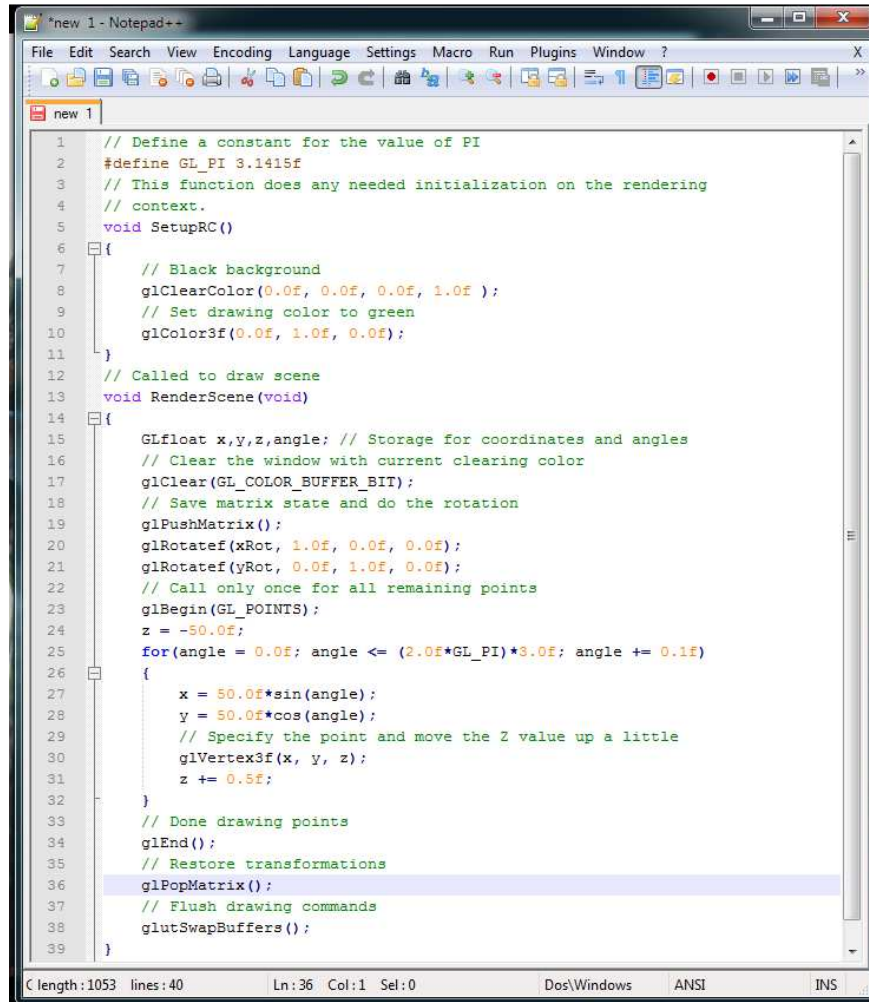


Figura 11 – Exemplo de renderização em OpenGL utilizando GLUI Fonte: (LAMARCHE, 2010)

O pequeno trecho de código a seguir (Figura 12), codificado em C++ e retirado do livro OpenGL Bible (WRIGHT, LIPCHAK, HAEMEL, 2007), exemplifica uma espiral de pontos:



```

1 // Define a constant for the value of PI
2 #define GL_PI 3.1415f
3 // This function does any needed initialization on the rendering
4 // context.
5 void SetupRC()
6 {
7     // Black background
8     glClearColor(0.0f, 0.0f, 0.0f, 1.0f );
9     // Set drawing color to green
10    glColor3f(0.0f, 1.0f, 0.0f);
11 }
12 // Called to draw scene
13 void RenderScene(void)
14 {
15     GLfloat x,y,z,angle; // Storage for coordinates and angles
16     // Clear the window with current clearing color
17     glClear(GL_COLOR_BUFFER_BIT);
18     // Save matrix state and do the rotation
19     glPushMatrix();
20     glRotatef(xRot, 1.0f, 0.0f, 0.0f);
21     glRotatef(yRot, 0.0f, 1.0f, 0.0f);
22     // Call only once for all remaining points
23     glBegin(GL_POINTS);
24     z = -50.0f;
25     for(angle = 0.0f; angle <= (2.0f*GL_PI)*3.0f; angle += 0.1f)
26     {
27         x = 50.0f*sin(angle);
28         y = 50.0f*cos(angle);
29         // Specify the point and move the Z value up a little
30         glVertex3f(x, y, z);
31         z += 0.5f;
32     }
33     // Done drawing points
34     glEnd();
35     // Restore transformations
36     glPopMatrix();
37     // Flush drawing commands
38     glutSwapBuffers();
39 }

```

Figura 12 – Código C++ usando biblioteca OpenGL Fonte: (WRIGHT, LIPCHAK, HAEMEL, 2007)

O código da Figura 12 calcula as coordenadas  $X$  e  $Y$  para um ângulo entre  $0^\circ$  e três vezes  $360^\circ$ . Cada vez que um ponto é desenhado, o valor de  $Z$  é levemente incrementado. O resultado da execução desse código, é apresentado na Figura 13 a seguir:

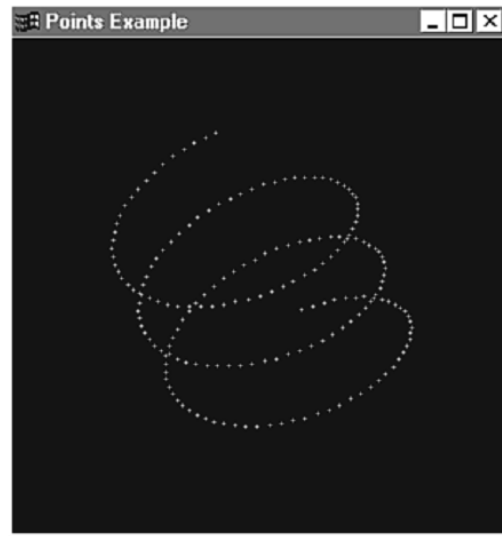


Figura 13 – Saída produzida pelo código apresentado na figura 12 Fonte: (WRIGHT, LIPCHAK, HAEMEL, 2007)

## 2.2 A VTK

A VTK (*Visualization Toolkit*) é uma ferramenta *open-source*, disponível gratuitamente para processamento de imagens 3D, utilizada por milhares de pesquisadores e desenvolvedores em todo o mundo (KITWARE, 2012). As aplicações que utilizam VTK podem ser escritas diretamente em C++, Java, Python, entre outras linguagens.

A VTK suporta uma ampla variedade de algoritmos de visualização, tais como: escalar, vetor, tensor, textura e volume, assim como técnicas avançadas de modelagem (KITWARE, 2012). Ainda, redução de polígonos, recorte, contorno e triangulação, entre outras, o transformam numa ótima opção para manipulação de conteúdos gráficos 3D (CARRANZA, FLORIAN, 2006).

### 2.2.1 Classes gráficas

De acordo com Schroeder, Martin e Lorensen (2002), na biblioteca oferecida no VTK há sete classes básicas que são utilizadas para visualização de cenas. Entre muitas outras que compõem a biblioteca, essas são usadas com mais frequência:

- *vtkRenderWindow* – Controla uma janela no dispositivo de visualização com um ou vários objetos;
- *vtkRenderer* – Coordena a apresentação e modelagem de objetos, envolvendo luzes, câmeras e atores;
- *vtkLight* – Fornece uma fonte de iluminação para a cena;



- *vtkCamera* – Define o ponto de vista da cena, o centro do foco e outras propriedades da visualização;
- *vtkActor* – Representa um objeto inserido na cena, suas propriedades e sua posição;
- *vtkProperty* – Define a aparência de um objeto, sua cor, transparência e luminosidade;
- *vtkMapper*– É a representação geométrica de um ator (objeto), com a característica que mais de um ator pode se referir ao mesmo *vtkMapper*;

Objetos instanciados da classe *vtkRenderWindow*, além de controlarem a construção dos objetos contidos neles e armazenar suas características, utilizam a técnica de *Double buffering*, onde a janela é dividida logicamente entre dois buffers. Enquanto o primeiro buffer é usado para a imagem em processamento, o segundo pode ser usado para desenhar a próxima imagem numa animação.

Quando a imagem processada no segundo *buffer* fica pronta, estes podem ser trocados facilmente, mostrando a continuação do movimento. Isso permite que o usuário veja animações sem perceber as primitivas geométricas (polígonos) (SCHROEDER, MARTIN & LORENSEN, 2002).

### 2.2.2 Modelo gráfico

O modelo gráfico captura as principais características de um sistema gráfico 3D. Os objetos básicos que compõem este modelo foram apresentados anteriormente e são compostos de funções e procedimentos que gerenciam o modelo e sua representação 3D ou 2D na tela do computador.

Deve-se destacar que o caráter dos dados a apresentar deve ser conhecido. Sem um perfeito entendimento do tipo de dados que se quer visualizar, corre-se o risco de gerar um sistema inflexível e limitado.

Dados de visualização são discretos porque são utilizados computadores para a aquisição e visualização daquilo que representam através de um número finito de pontos, e serão representados num formulário de tela limitado pelo seu tamanho.

O papel do modelo gráfico do VTK é transformar os dados em imagens, enquanto que as funções de visualização constroem a representação geométrica dos dados que serão

apresentados. Entre os diferentes tipos de dados que podem constituir um objeto estão, entre outros, pontos, retas, polígonos, pontos estruturados, malhas estruturadas e não estruturadas, como apresentado na Figura 14.

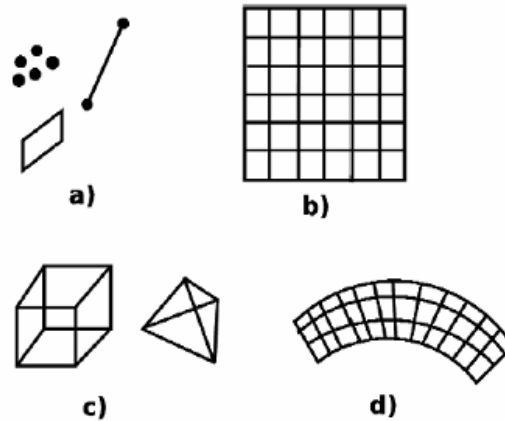


Figura 14 – a) pontos, retas e polígonos, b) pontos estruturados, c) malha não estruturada, d) malha estruturada. Fonte: (CARRANZA, FLORIAN, 2006)

### 2.2.3 Arquitetura do VTK

O VTK tem um núcleo compilado, implementado em C++ e uma interface interpretada gerada automaticamente. O *pipeline* de processamento de dados, transforma-os para que possam ser exibidos pelo modelo gráfico descrito anteriormente. Os objetos de dados representam e permitem o acesso a eles através de filtros.

Desde que se usem os *wrappers* necessários, o VTK pode ser utilizado praticamente com qualquer linguagem. *ActiViz.net*, por exemplo, permite seu uso com o Microsoft.NET *framework*. Isso significa que se podem utilizar linguagens como C# ou Visual Basic para usufruir das vantagens gráficas que o VTK oferece.

O pequeno trecho de código a seguir apresenta uma esfera numa malha poligonal, codificado em C# no Visual Studio 2010. O *framework* ajuda no uso do VTK para mostrar e manipular a esfera com opções de movimentar, ampliar e reduzir o objeto sem muito esforço de programação (KITWARE, 2008).

```

Form1.cs* X Form1.cs [Design]*
HelloFormVTK.Form1 VTKRender_Load(object sender, Ev
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Kitware.VTK;

namespace HelloFormVTK
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void VTKRender_Load(object sender, EventArgs e)
        {
            // Create a simple sphere. A pipeline is created.
            vtkSphereSource sphere = vtkSphereSource.New();
            sphere.SetThetaResolution(8);
            sphere.SetPhiResolution(16);

            vtkShrinkPolyData shrink = vtkShrinkPolyData.New();
            shrink.SetInputConnection(sphere.GetOutputPort());
            shrink.SetShrinkFactor(0.9);

            vtkPolyDataMapper mapper = vtkPolyDataMapper.New();
            mapper.SetInputConnection(shrink.GetOutputPort());

            // Link the data pipeline to the rendering subsystem
            vtkActor actor = vtkActor.New();
            actor.SetMapper(mapper);
            actor.GetProperty().SetColor(1, 0, 0);

            // Create components of the rendering subsystem
            //
            vtkRenderer ren1 = VTKRender.RenderWindow.
                GetRenderers().GetFirstRenderer();
            vtkRenderWindow renWin = VTKRender.RenderWindow;

            // Add the actors to the renderer, set the window size
            ren1.AddViewProp(actor);
            renWin.SetSize(250, 250);
            renWin.Render();
            vtkCamera camera = ren1.GetActiveCamera();
            camera.Zoom((double)1.5);
        }
    }
}
75 %

```

Figura 15 – Código C# para visualização de uma esfera com o VTK. Fonte: (KITWARE, 2008)

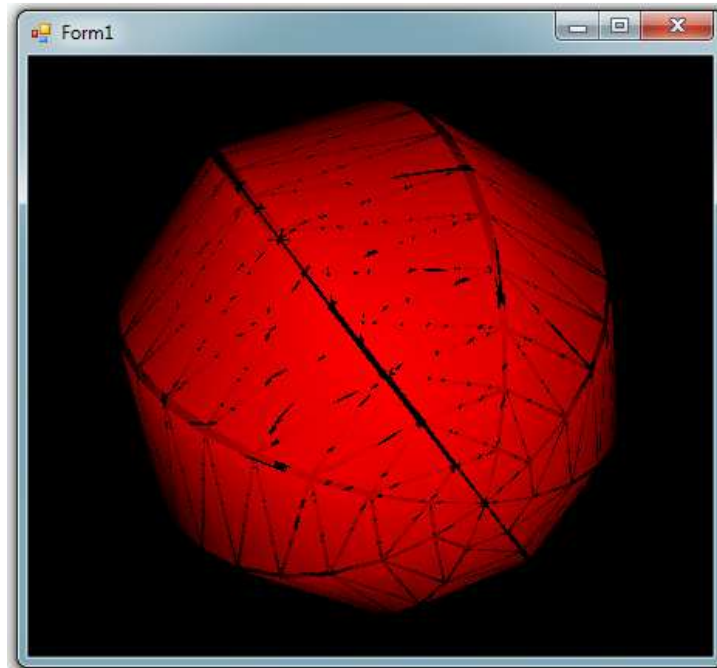


Figura 16 – Saída apresentada pelo código acima, com o uso de VTK. Fonte: PrintScreen da saída do programa

### 2.3 WebGL

A WebGL é uma DOM (*Document Object Model*) API (*Application Programming Interface*) gráfica desenhada para manipulação de gráficos 3D na web e derivada do OpenGL ES 2.0. Quando integrada no *browser*, a WebGL pode tirar vantagem da infraestrutura do *JavaScript* e do DOM em qualquer documento HTML. Em se tratando de uma tecnologia que se relaciona com diversas outras, são necessárias algumas breves considerações sobre estas.

Inicialmente, é preciso mencionar o HTML5 (*Hyper Text Markup Language*) quinta versão. O seu principal objetivo é manipular a maioria dos conteúdos multimídia sem a necessidade de utilizar *plugins*. Entre as muitas características interessantes que a linguagem oferece, a WebGL utiliza particularmente o *canvas* (GONZÁLEZ, 2012).

O *canvas* é um elemento de HTML5 que permite a geração de gráficos dinamicamente utilizando *scripting*<sup>1</sup>. Seu acesso se dá por meio do uso do JavaScript e na atualidade é suportado pela maioria dos navegadores. Entre outras coisas, o *canvas* permite a

---

<sup>1</sup>*Scripting* são pequenos trechos de código executados do interior de programas ou de outras linguagens de programação.

criação de retângulos, linhas, arcos, curvas, manuseio de cores e estilos, como também, transformações, composições e animações sem o uso de *plugins* externos.

Também é interessante mencionar o OpenGL ES 2.0. Esse é uma variação simplificada do OpenGL visto anteriormente, para dispositivos integrados, como *smartphones* e PDA. A biblioteca permite a programação de gráficos 3D de maneira similar ao OpenGL na sua versão completa, mantendo uma característica muito importante que é a aceleração por *hardware* (*Hardware acceleration*).

Essa técnica consiste no uso do *hardware* para desempenhar algumas funções mais rapidamente do que seria em um *software* rodando na CPU. Usa-se, ao invés disso, a GPU (*Graphic Processing Unit*) da placa gráfica para o processamento de grandes cargas gráficas. A GPU é uma unidade de processamento de gráficos dedicada a operações de ponto flutuante – por este motivo usada para processamento de gráficos, os quais exigiriam muito do processador.

Finalmente, uma palavra sobre *JavaScript* e *DOM*. *JavaScript* é uma linguagem de *script* multiparadigma, que suporta os estilos de programação orientada a objetos. Por ser uma linguagem do lado cliente (*client-side*), é implementada como parte do navegador web, oferecendo melhoras na interface do usuário e nas páginas web dinâmicas.

Todos os navegadores web modernos interpretam o código *JavaScript* – integrado nas páginas. Assim ele dá condições aos navegadores para reconhecer objetos numa página HTML através do DOM – uma API multiplataforma para interagir com documentos HTML, XHTML e XML – facilitando o acesso dinâmico ao conteúdo e estrutura, assim como, ao estilo dos documentos.

De acordo com González (2012), a WebGL foi criada inicialmente pela Mozilla, e posteriormente padronizado pelo grupo Khronos, que é o responsável pelo OpenGL e o OpenGLES. Pode ser dito que WebGL é uma biblioteca de *software* que complementa o *JavaScript* para abrir possibilidades de geração de gráficos 3D interativos em qualquer navegador compatível. Pode-se observar um modelo conceitual da arquitetura WebGL na Figura 17.

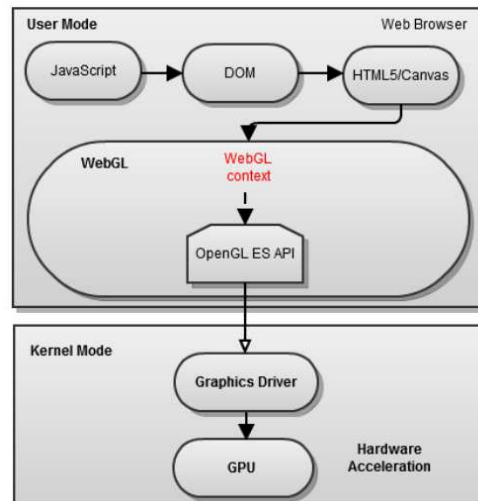


Figura 17 – Modelo conceitual da arquitetura WebGL Fonte: (GONZÁLEZ, 2012)

No modelo apresentado, utiliza-se JavaScript para obter através do DOM o elemento *Canvas* do HTML5. Nele se define o contexto WebGL, tendo acesso à sua API, baseada naquela da OpenGL ES, a qual se comunica com o *driver* da placa gráfica para poder realizar a aceleração por *hardware* na GPU.

Outra característica importante da WebGL é que, por ser um conjunto de especificações técnicas em constante evolução, *standard web*, facilita sua manutenção e interoperabilidade. Utilizar padrões (*standard*) garante vida longa aos projetos que os utilizam e usabilidade independente do navegador utilizado (GONZÁLEZ, 2012).

As bibliotecas de código aberto apresentadas até aqui facilitam ao usuário a oportunidade de implementar sistemas gráficos de alta qualidade, cabendo ao desenvolvedor a escolha da biblioteca a ser utilizada, conforme peculiaridades inerentes ao gosto e adaptabilidade referentes a seu uso.

Entretanto, existem ferramentas que oferecem, além das funcionalidades fornecidas por essas bibliotecas, ambientes completos para desenvolvimento de aplicações gráficas – o que pode facilitar a construção do protótipo proposto neste trabalho. Assim uma dessas ferramentas – a Unity será apresentada a seguir.

## 2.4 A ferramenta Unity

Nos últimos anos, o desenvolvimento de jogos tem impulsionado o mercado de TI (Tecnologia da Informação) de uma forma contundente. Nada mais motivador que um *software* desenvolvido especificamente para facilitar a criação de jogos 2 e 3D para proporcionar às mentes criativas, o incentivo ideal que tem levado milhares de jovens a

procurar os caminhos da computação gráfica. Esse é precisamente o caso da ferramenta Unity (TECHNOLOGIES, 2013).

Mesmo tendo sido lançada em 2005, já se encontra na quarta versão, considerada como um poderoso motor de renderização para jogos. Sua facilidade de publicar os projetos para várias plataformas confere especial interesse nela para uso na implementação do protótipo.

Desenvolvido pela Unity Technologies, a ferramenta é muito similar ao Blender, ela é disponibilizada em duas versões: Unity Pro que é paga e a versão gratuita, simplesmente chamada Unity, que pode ser utilizada com propósito educacional e até com fins comerciais (TECHNOLOGIES, 2013).

A ferramenta disponibiliza recursos para manipulação dos elementos básicos específicos de jogos virtuais, tais como – roteiro, personagens, câmeras, luzes, terreno e sons. Além disso, ela também provê funcionalidades úteis para qualquer aplicação gráfica, tais como movimentação e seleção de objetos e de câmera.

Sua programação é, na maior parte, resolvida criando-se cenas, colocando câmeras, objetos, alterando parâmetros, movimentando e posicionando todos no espaço desejado, num ambiente gráfico, intuitivo e de fácil utilização.

Para isso, a Unity possui uma área de trabalho muito rica, composta por uma variedade de *views* ou janelas cuja funcionalidade permite o manuseio das propriedades e métodos dos objetos. Na Figura 18 podem ser observados objetos distribuídos na cena, tais como a câmera, a iluminação e o controlador de primeira pessoa.

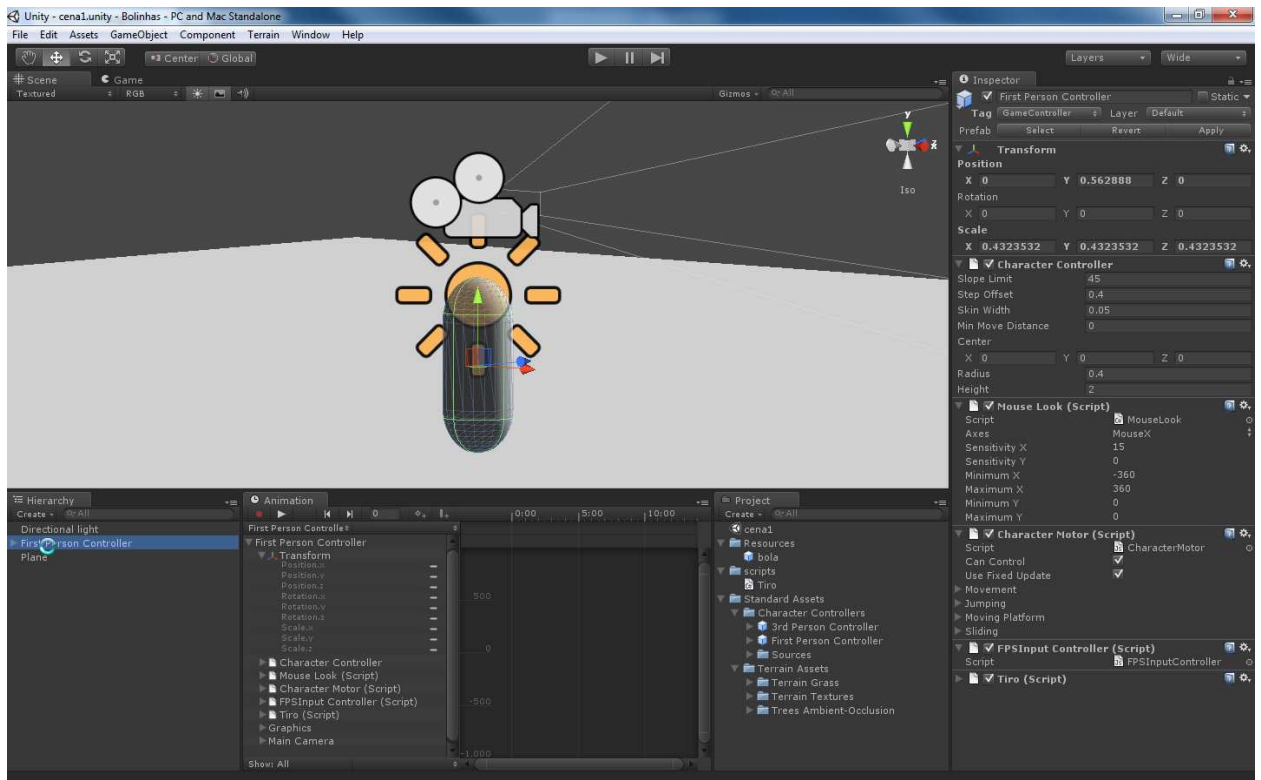


Figura 18 – Ambiente gráfico para desenvolvimento da Unity

Obviamente, não se pode abrir mão da codificação tradicional, tendo em vista que é com ela que se consegue complementar o uso destes objetos, importando estruturas, movimentando dinamicamente a câmera ou os objetos na cena e iniciando animações, mudando a iluminação e interagindo com o usuário, entre outras coisas.

Para isto, a Unity dá suporte a *scripts* em *Boo*, *C#* ou *JavaScript*, sendo possível escolher a que melhor atenda às necessidades do desenvolvedor. Os *scripts* na Unity são considerados *game objects* e uma vez criados, podem ser anexados a qualquer outro objeto que faça parte da cena, inclusive a câmera. Isto permite uma grande interação entre os objetos e uma maleabilidade que pode ser muito bem aproveitada, compreendendo sua biblioteca de objetos, classes e funções.

Aliado a isso, o editor da Unity permite o desenvolvimento da aplicação de maneira fácil e intuitiva, testando e observando as modificações enquanto são feitas e integrando seus *scripts* de forma dinâmica ao seu editor e compilador, que permite *debug* e reporte de *logs* para acompanhar o desenvolvimento.

A biblioteca da Unity utiliza uma arquitetura de objetos para o jogo baseada em composição, ou seja, é composto por várias funções que vão sendo adicionadas ou retiradas



conforme a necessidade de um objeto *container*, chamado de *Game Object*, que funciona como um repositório de funcionalidades.

É importante destacar que todos os *Gameobjects* possuem o componente *Transform* que se responsabiliza pelo posicionamento, escala e orientação dele na cena. Na Figura 19, a seguir, pode ser observado um *script* que implementa uma funcionalidade a um objeto.

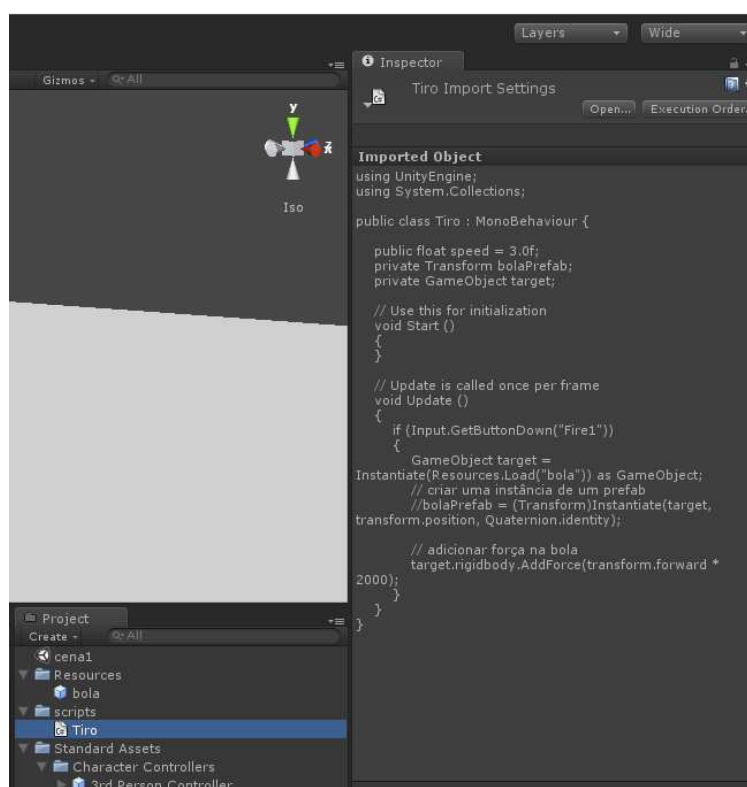


Figura 19 – Ambiente gráfico para desenvolvimento da Unity

A Unity suporta os mais conhecidos formatos de arquivos para seus objetos renderizados. Por exemplo, ela importa modelos 3D nos formatos: “.mb” e “.ma” (Autodesk Maya), “.max” (Autodesk 3D Max), “.blend” (Blender) e outros como XSL, *Lightwave*, Cinema4D e “.obj”.

A ferramenta suporta também fontes *TrueType* “.ttf” bastando arrastá-las para o ambiente, permite o trabalho com camadas no formato “.psd” e outros formatos como “.jpg”, “.png”, “.bmp” e “.tga”.

Unity possui uma ampla biblioteca de classes, subclasses e métodos que manipulam de maneira transparente todas as operações de translação, mudança de escala, rotação e transformações de perspectiva (apresentados teoricamente no capítulo 2).

Ela trabalha com translação através de funções como *Translate*, que é um método da classe *Transform*. Todo objeto do cenário tem um *Transform* que é utilizado para armazenar e manipular seu posicionamento, rotação e escala.

Assim, o método *Transform.translate*:

- Move o objeto na direção e distância informadas;
- Move o objeto no eixo *X* distância *x*, eixo *Y* distância *y* e eixo *Z* distância *z*;

Para rotacionar um objeto, a Unity tem a função *Transform.rotate* que aplica a rotação dos ângulos de Euler *z* graus sobre o eixo *Z*, *x* graus sobre o eixo *X* e *y* graus sobre o eixo *Y*, nessa ordem.

A mudança de escala na Unity se obtém com a função *Transform.localscale*, que muda o tamanho do objeto relativamente àquele que o contém, utilizando a estrutura *Vector3*, que é usada para passar posições 3D e direcionamento.

Já as transformações de perspectiva são tratadas na classe *Câmera*. Todas as transformações podem ser realizadas através da utilização de *scripts* ou diretamente na interface gráfica da ferramenta, clicando no objeto e utilizando os botões de transformação ou as teclas de atalho pré-definidas.

Apresentadas as bibliotecas e ferramentas qualificadas para a realização do protótipo, objetivo do presente trabalho, se faz necessário uma revisão mais apurada dos esforços que vem sendo feitos para colaborar com a problemática equacionada no primeiro capítulo.

Foram analisados trabalhos correlatos já existentes e cujo uso tem sido uma realidade promissora. Em função do alinhamento deste trabalho junto ao CAS (Grupo de Computação Aplicado em Saúde da Universidade Feevale), restringiu-se o escopo a visualizadores relacionados com a área médica – tanto por estarem em conformidade com a proposta aqui apresentada, quanto por, em virtude da complexidade de representação do corpo humano, explorarem de forma consistente a área de visualizadores 3D. Tais trabalhos são apresentados a seguir.

### 3 TRABALHOS CORRELATOS

#### 3.1 BioDigitalHuman™

O BioDigitalHuman™ é uma plataforma que simplifica o aprendizado da anatomia, doenças e tratamentos. Com ferramentas interativas para exploração, dissecação e compartilhamento de visões, combinadas com detalhes médicos, uma excelente ferramenta para o estudo (SYSTEMS, 2011).

A plataforma 3D foi desenvolvida em WebGL e utilizando o *canvas* HTML5 como DOM (*Document Object Model*) como apresentado na seção 2.3 deste trabalho. Como o sistema é totalmente baseado na Web, não há necessidade de instalação, podendo ser acessado diretamente do *browser* no endereço <http://www.biodigitalhuman.com>.

A aplicação permite o estudo da anatomia humana e os diversos sistemas funcionais do corpo, como o esquelético, digestivo, urinário, reprodutivo, respiratório, endócrino, linfático, nervoso, cardiovascular, muscular, entre outros, facilitando o aprofundamento no estudo, por parte do usuário, que pode navegar pelos diversos sistemas. Tudo isso é apresentado numa interface agradável, onde o estudante, acompanhado de explicações que evoluem conforme o ponto de navegação em que se encontra, sente-se a vontade para procurar as informações do seu interesse.

Na primeira visita é realizada uma verificação para saber se o computador do usuário comporta os gráficos 3D requeridos pelo sistema BioDigitalHuman™, informando o resultado. Se os requisitos forem preenchidos, os campos aparecem checados e é realizado um teste de *performance*, também informado na tela, como pode ser visto na Figura 20.

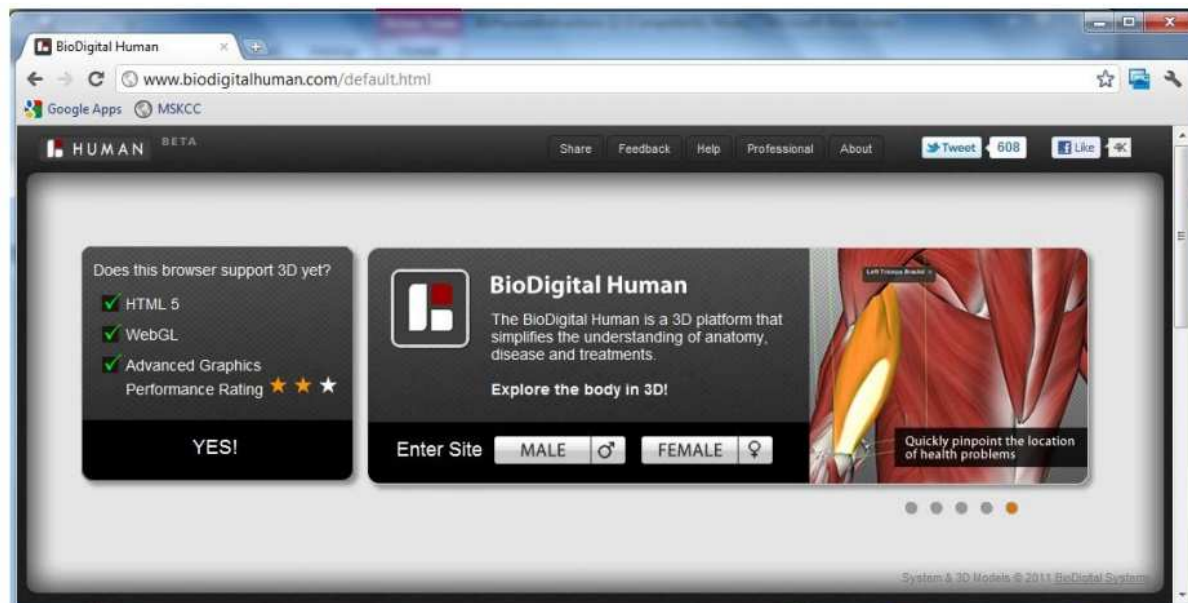


Figura 20 – Verificação de compatibilidade de hardware com o sistema BioDigitalHuman™.  
Fonte: (SYSTEMS, 2011)

O sistema possui aproximadamente 4000 objetos representando peças anatômicas. De maneira intuitiva, todas estas informações são organizadas com o propósito de facilitar a navegação em uma árvore de relacionamento de objetos, cuja profundidade vai se mostrando a medida que a seleção de uma ramificação é realizada.

Para localizar as peças de interesse, o usuário pode esconder diferentes partes utilizando a referida árvore. O usuário pode receber informação detalhada da(s) peça(s) em observação acompanhando a descrição pedagógica numa janela aberta com esse propósito. Esta característica tem a finalidade de servir como tutor ou instrutor específico que vai valorizando as visualizações com descrições esclarecedoras sobre a peça em foco.

Na Figura 21, a seguir, mostra-se a seleção do rim direito de um ser humano e a explicação que aparece ao seu lado, vai auxiliando o aluno no seu aprendizado.

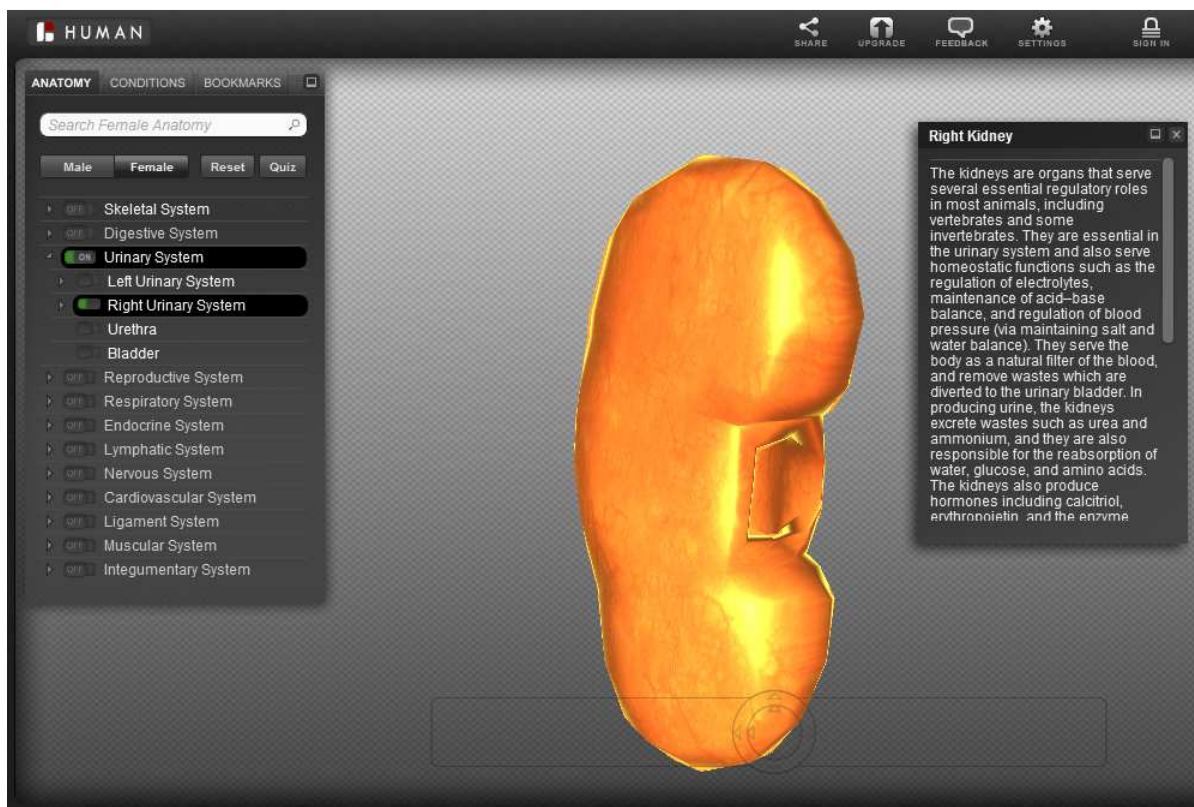


Figura 21 – Visualização do rim direito com a descrição científica ao seu lado. Fonte: (SYSTEMS, 2011)

Outro aspecto interessante do sistema é a facilidade de manipulação da imagem selecionada, pelo *mouse* ou pelo teclado. O uso das teclas e da roda central do *mouse* para movimentar o objeto, selecionar, aumentar ou diminuir, é uma opção de fácil assimilação por parte do usuário. Isso se dá porque o usuário geralmente procura selecionar e movimentar os objetos com o ponteiro, assim como, o uso da roda central que é muito comum na utilização de *scroll* e *zoom* de conteúdos.

### 3.2 Healthline Body Maps

O *Healthline Body Maps* é outra plataforma que ajuda no aprendizado da anatomia humana (HEALTHLINE, 2005). Muito mais simplificada que a apresentada anteriormente, utiliza os recursos da computação gráfica 3D para mostrar também os diversos sistemas de funcionamento do corpo. O sistema é gratuito e encontra-se disponível no site da *Healthline* e de forma direta no endereço <http://www.healthline.com/human-body-maps>. Permite a seleção do sexo a ser estudado e imediatamente abre uma barra de ferramentas com as opções para navegação dentro dos sistemas mencionados.

O mouse pode ser usado para girar os objetos selecionados, assim como uma barra apresentada no rodapé da janela, que permite o giro gradual da imagem ou a animação da mesma num giro de 360°.

No passeio do *mouse* pela figura, o sistema mostra o nome do objeto em cuja área encontra-se o ponteiro. Um *click* no botão dinâmico criado para mostrar esse nome, faz com que o sistema abra outra tela do navegador com o objeto em foco, ocupando toda a janela de visualização e com as opções já descritas, abrangendo a nova área ampliada.

Desta forma ocorre a navegação pelos diversos sistemas do corpo. No segundo nível de navegação, o *click* numa determinada área, apresenta uma janela *popup* com a descrição da peça (HEALTHLINE, 2005).

### 3.3 Google Open Sources Zygote 3D Human Body Viewer

A *Google* disponibilizou como *opensource* sua plataforma de visualização 3D que utiliza WebGL e que foi construída nos 20% de tempo livre que a companhia disponibiliza aos seus funcionários para a livre criação. O código está hospedado pela *Zygote Media Goup* que providenciou as imagens para o *Google Body*, como é chamada a plataforma (BRIDWATER, 2012).

Muito popular entre professores e estudantes de anatomia, seus usuários podem usufruir de ampla galeria de imagens 3D modeladas para representar os diversos sistemas do corpo, já mencionados anteriormente.

Simple e intuitivo, o *Google Body* apresenta o corpo selecionado pelo usuário na barra de menu lateral, cujos ícones mostram o tipo de sistema a escolher na navegação tridimensional que será apresentada na janela 3D.

O *mouse* permite o giro da imagem apresentada, a roda do *mouse* ajusta seu aumento ou diminuição como os demais *softwares* analisados e o clique sobre determinada área da figura, realizado num ícone apresentado na barra lateral, apresenta uma nova visão ampliada do sistema selecionado.

A imagem a seguir mostra a seleção ampliada do sistema escolhido, determinado na região demarcada no ícone lateral.

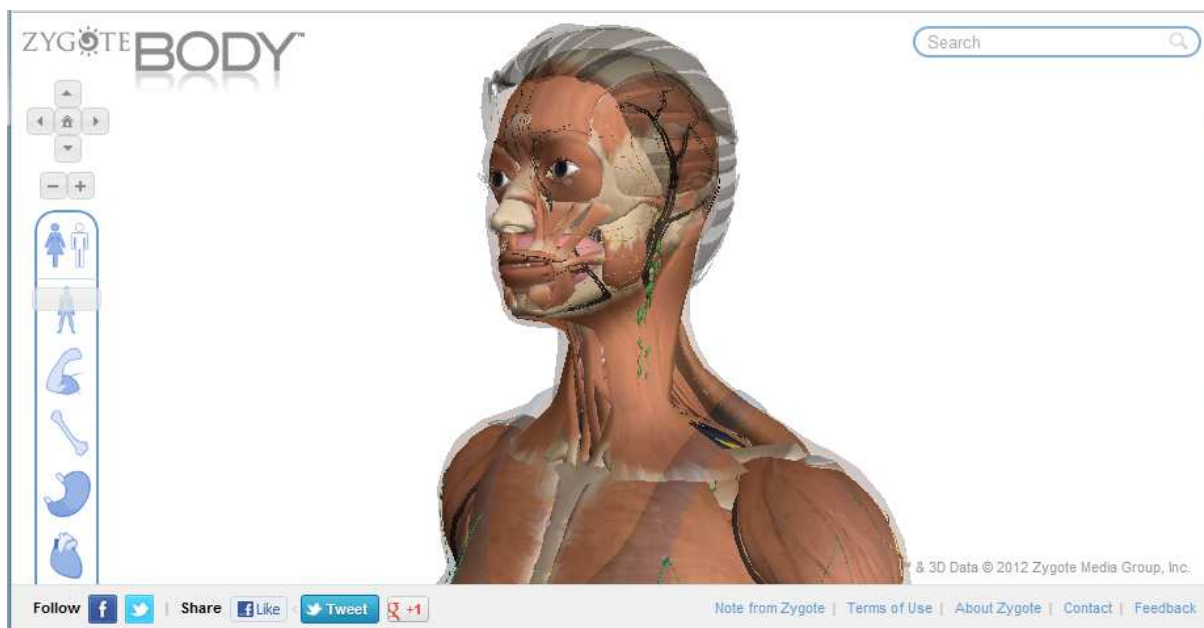


Figura 22 – Imagem 3D montada no Google body da seleção no ícone lateral. Fonte: (GOOGLE, 2012)

Os sistemas apresentados têm uma ampla gama de menus e funções que levam um tempo para serem assimiladas e um tempo ainda maior para poder tirar deles o melhor proveito na sua utilização.

Lembrando a invenção do quadro negro ou de giz, pode-se fazer uma analogia de o quanto um quadro virtual que permita visualizar em 3D apenas um objeto sobre o aspecto do seu estudo, pode contribuir na área, tanto da medicina como da educação em geral.

A praticidade de poder acessar esse quadro virtual 3D de qualquer equipamento portátil ou computador pessoal na sala de aula, pode ser interessante, agregando valor ao vasto material existente no mercado e apresentado no presente trabalho.

A seguir apresenta-se uma síntese do protótipo idealizado para sua utilização no auxílio do estudo de peças cuja complexidade ou característica justifique sua análise tridimensional por parte do aluno ou pessoa interessada.

#### 4 PROPOSTA DA FERRAMENTA

Ante o exposto, com a intenção de contribuir num campo onde já se realizaram avanços com projetos na área médica (a exemplo dos já apresentados), busca-se desenvolver o protótipo de uma ferramenta que será de utilidade para as áreas de ensino em que a visualização 3D seja necessária ou desejada.

Trazendo a tona o que foi mencionado sobre a invenção do quadro negro ou quadro de giz no século XVIII, pode-se deduzir que essa foi uma ideia simples e prática. O professor que o idealizou possivelmente deve ter pensado em uma ferramenta com que seus alunos pudessem fixar visualmente o que era colocado verbalmente na sala de aula. Fica claro que o quadro negro, pela facilidade de uso e a possibilidade de ser utilizado em vários locais, cumpriu seu propósito, pois sua utilização se estende até os dias atuais.

A propósito disso, de acordo com Bill Jensen, presidente da Jensen Group, numa entrevista dada à revista Exame, fazer as coisas mais fáceis de usar é o valor da simplicidade (EXAME, 2013). As pessoas, de certa forma, estão tendo um choque do futuro, como refere Jensen. Assim, ele realizou uma pesquisa que envolveu mais de 350 mil pessoas de 17 países. Nela descobriu que a sobrecarga cognitiva originada pelo choque do futuro é de grandes proporções. Fala também que a inabilidade de processar rapidamente muitos tipos de informação força as pessoas a adotar certos comportamentos. Elas estão se esforçando para descobrir o que realmente merece atenção, num mundo em que o tempo e a atenção são finitos e a quantidade de informação e escolhas não (EXAME, 2013). Essas pessoas estão se esforçando na luta diária para decidir o que é urgente e o verdadeiramente importante e, na maioria das vezes, o urgente termina vencendo.

Isso nos faz pensar que oferecer uma ferramenta que se proponha a mostrar um objeto 3D para ser observado de todos os ângulos que a tridimensionalidade permita, pode ser apenas isso, sem subsistemas ou menus elaborados. Apenas um visualizador, de fácil manuseio, de utilização não restrita apenas a computadores, que possa ser usado em *tablets* ou *smartphones* e de fácil assimilação – enfim, um acréscimo pela simplicidade.

Por conseguinte, este trabalho propõe o desenvolvimento de uma ferramenta chamada “3D Board”, que possa ser utilizada em *smartphones* ou *tablets* e que consiga manipular um objeto 3D do ponto de vista de quem está observando.



Assim, simplicidade pode ser a chave para agregar valor ao que já existe. Um quadro negro 3D ou “3D Board”, como foi batizado o protótipo, pode ser definitivamente um acréscimo, tanto na área médica para o estudo de uma peça anatômica, como para os alunos do ensino fundamental ou médio, observarem algum objeto de difícil obtenção presencial.

A grande diferença que o “3D Board” trará será a objetividade. Não haverá menus nem tutoriais longos para aprender a usá-lo. Será uma ferramenta simples e de fácil assimilação para cumprir o propósito de ser um quadro virtual, onde os objetos podem ser vistos de diferentes ângulos, permitindo um estudo morfológico de grandes vantagens quando comparado ao que pode ser apresentado no antigo quadro bidimensional.

Nesse sentido, outro aspecto importante a considerar no mundo de aplicações 3D é o manuseio da ferramenta. É importante, na medida do possível, preservar as características de movimentação dos objetos, seja utilizando o *mouse*, seus botões e sua roda central, seja utilizando toques e gestos em dispositivos móveis, de forma consistente com o que já está consolidado entre os usuários. Com a manutenção desses padrões de operação pretende-se preservar a relação cognitiva já estabelecida pelo usuário com a forma de manipulação de objetos virtuais.

Por fim, poderá ser acionada na sala de aula e o professor poderá submeter um objeto preparado para o estudo em um endereço eletrônico. Uma vez carregado no equipamento disponível, o aluno pode manipular o objeto e responder a perguntas formuladas.

Para alcançar tal objetivo, o protótipo deverá atender aos seguintes requisitos:

- a) O objeto deve ser carregado dinamicamente;
- b) Ter uma iluminação direcionada ao objeto carregado;
- c) O objeto deverá estar posicionado à frente de uma câmera fixa, a partir da qual o expectador possa examiná-lo, movimentá-lo, aproximá-lo ou afastá-lo de uma forma intuitiva e fácil de assimilar;
- d) Deverá ser possível rotacionar o objeto nos seus 3 eixos do sistema de coordenadas  $\mathbb{R}^3$ ;
- e) Deverá ser possível movimentar o objeto vertical e horizontalmente (o que equivale a movimentações nos eixos *Y* e *X* respectivamente);
- f) Deverá poder aproximar ou afastar o objeto do ponto de vista do observador (o que equivale à movimentação no eixo *Z*);

- g) Utilizar os gestos de interface já bem difundidos entre os usuários de PCs, *smartphones* e *tablets*, bem como outros elementos gráficos intuitivos e opções de hardware como teclado e mouse.

O próximo capítulo apresenta a forma como foram implementados esses requisitos, abordando cada requisito, sua problemática, uma possível solução e a forma como foi implementada no protótipo.

## 5 IMPLEMENTAÇÃO DO PROTÓTIPO

Tendo como objetivo a criação do protótipo proposto no capítulo anterior, primeiramente houve a necessidade de escolher entre as bibliotecas e ferramentas apresentadas, aquela que melhor permitisse alinhar o trabalho dentro dos objetivos propostos.

A preocupação com a interface de qualquer aplicativo vem ganhando relevância no contexto atual. A grande diversificação de equipamentos com os quais os usuários teriam acesso ao “3DBoard” complementa a justificativa para a atenção especial a este aspecto. Por conseguinte, a preocupação em desenvolver o protótipo em uma ferramenta que permita sua implementação nessa ampla gama de equipamentos ganhou um peso respeitável na hora da escolha.

Além disso, procurou-se uma ferramenta que permitisse a implementação do protótipo de forma relativamente rápida e fácil de construir. Por último, também levou-se em conta a preferência pessoal do autor deste trabalho quanto à linguagem utilizada para a codificação. Assim, após o estudo, uso e avaliação das ferramentas e bibliotecas descritas anteriormente no trabalho, foi possível fazer algumas observações sobre seus resultados.

Primeiramente, avaliou-se comparativamente o uso isolado da WebGL e da API gráfica OpenGL em relação à sua utilização por meio da ferramenta Unity. Com isso, constatou-se que o uso da Unity permite tirar um maior proveito da OpenGL. Ela a utiliza para ter acesso ao *hardware* gráfico de uma forma transparente, fazendo-o através de suas classes e métodos. Também constatou-se que, em alguns casos, é possível exportar para WebGL. Além disso é possível com ela desenvolver o protótipo uma única vez e exportá-lo para as plataformas pretendidas neste trabalho.

Em seguida, avaliou-se a biblioteca VTK. Essa tem uma ampla gama de funções para manipulação de objetos. Considerando as preferências do autor deste trabalho em relação à linguagens e ambientes de desenvolvimento, foram realizados diversos testes instalando a biblioteca no *Visual Studio* e utilizando C#. Entretanto, nesse ambiente, existe a limitação de só poderem ser geradas aplicações para *Windowsform*, inviabilizando seu uso, nessas condições, para atender à diversidade de plataformas pretendida no trabalho.

Assim, considerando que a VTK não atende plenamente aos objetivos do protótipo no ambiente de preferência do autor deste trabalho e que com a Unity é possível obter, de

forma facilitada, os recursos necessários da OpenGL e WebGL (bibliotecas equivalentes à VTK em funcionalidade), atendendo, ainda, aos requisitos de portabilidade, essa foi a ferramenta escolhida para ser utilizada na construção do protótipo proposto. Entretanto, apesar de ser a que melhor atende os propósitos estabelecidos, a Unity apresenta um desafio particular: o de utilizar uma *engine* para jogos com o propósito de fazer uma ferramenta de manipulação para apenas um objeto a ser estudado em 3D – uma abordagem totalmente diferente se comparada à dinâmica de um jogo.

Feita essa escolha, a seguir serão abordados cada um dos requisitos elencados para o protótipo, sua problemática, uma possível solução e, finalmente, a forma como foram implementados.

### 5.1 Carregamento dinâmico do objeto

Na proposta falou-se em realizar o carregamento dinâmico dos objetos que serão manipulados pelo usuário. Considerando a crescente disponibilidade de acesso à *internet* e o eventual custo de, no caso de um professor em sala de aula, ter-se de disponibilizar os objetos a serem estudados no dispositivo de cada aluno, decidiu-se por realizar essa funcionalidade utilizando a plataforma *web*. Isso quer dizer que os objetos deverão ser colocados em uma *URL (Uniform Resource Locator)* ou endereço eletrônico, de onde o “3D board” fará sua leitura. Ainda, para ler e importar os objetos para o ambiente gráfico do protótipo, deve haver uma espécie de protocolo, por assim dizer – uma série de regras para catalogá-los, ordená-los e transmiti-los para a ferramenta.

Na Unity, a importação de objetos não é algo comum de acontecer. Ela apresenta duas possibilidades de carregamento dinâmico: *assetbundles* (grupo de objetos) e *resource folders* (diretório de recursos). Entretanto, nas duas abordagens os objetos devem ser criados dentro do projeto, importando-os, agrupando-os em pacotes e exportando-os novamente, o que limita muito a liberdade de disponibilizar mais amostras 3D para visualização.

Procurando a solução, achou-se uma abordagem muito interessante em (DROZDZ, 2010). É uma série de *scripts*, contendo classes, que se encarregam de ler as informações dos objetos e montá-los num *Game Object* vazio, previamente criado para receptá-lo. Tais *scripts* foram objeto de um estudo para viabilização de seu uso no protótipo. Uma característica importante deles é a utilização do formato “.obj” criado para trabalhar no *Wavefront’s Advance Visualizer* (FILEFORMAT.INFO, 2013).

Nos diversos testes realizados com os *scripts* encontrados, o objeto da *URL* que vem gravado neles foi carregado no protótipo. Contudo, ao mudar-se a *URL* do objeto (a fim de poder armazená-lo em algum lugar desejado), a importação não ocorria. Então, descobriu-se que, para poder usar um endereço externo na Unity, deve ser disponibilizado um arquivo *crossdomain.xml* no site remoto (onde será armazenado o objeto a ser carregado) com o seguinte formato:

```
<?xml version="1.0"?>
  <cross-domain-policy>
    <allow-access-from domain="*" />
  </cross-domain-policy>
```

Com esse acréscimo, o objeto disponibilizado foi importado sem problemas. Assim, essa abordagem passou a fazer parte da solução gerada no presente trabalho.

Apesar de útil, essa abordagem tem a clara limitação de importar apenas objetos no formato citado e no padrão Blender, o que forçou a necessidade de obtenção de amostras compatíveis para uso no protótipo. Muitas amostras tiveram que ser convertidas entre diversas ferramentas modeladoras de objetos 3D – no caso, utilizaram-se as ferramentas Blender (BLENDER, 2013) e Autodesk 3ds Max (AUTODESK, 2013).

No protótipo foi inserido um *GameObject* vazio para o qual foi movido o *script* principal do exemplo estudado, adaptado ao projeto. Como a *URL* utilizada nele era fixa, foi criado um novo *script* que agrupa *Gameobjects* do tipo *GUI* (*Graphic User Interface*) para permitir que o usuário carregue um novo objeto:

```
usingUnityEngine;
usingSystem.Collections;

public class criaOBJ : MonoBehaviour {

    privateGameObject of;
    privateGameObjectObjectFactory;
    private string nomeObj = " ";
    private string newGOID;

    voidOnGUI (){

        GUI.BeginGroup (new Rect (Screen.width / 2 - 160, Screen.height - 45, 500, 40));
        GUI.Box (new Rect (5,10,150,22), "Informe o Objeto: ");
        nomeObj = GUI.TextField(new Rect(160, 10, 100, 25), nomeObj, 25);

        if (GUI.Button(new Rect(280, 10, 80, 25), "Carregar" )){
            if (of != null){
                Destroy (of);
            }

            // Instancia o ObjectFactory na mesma posição
            ObjectFactory = GameObject.FindGameObjectWithTag("goFactory");
            Of= (GameObject) Instantiate(ObjectFactory,
            ObjectFactory.transform.position,
            ObjectFactory.transform.rotation);
```

```

// Pega os scripts e passa os parâmetros nos campos public
of.GetComponent<OBJ>().nomeObjeto = nomeObj.Trim();
of.AddComponent<MoveBtns>();
of.GetComponent<MoveBtns>().btnUp = Resources.Load("btnUp") as Texture;
of.GetComponent<MoveBtns>().btnLeft = Resources.Load("btnLeft") as Texture;
of.GetComponent<MoveBtns>().btnRight = Resources.Load("btnRight") as Texture;
of.GetComponent<MoveBtns>().btnDown = Resources.Load("btnDown") as Texture;
of.GetComponent<MoveBtns>().btnMais = Resources.Load("btnZMais") as Texture;
of.GetComponent<MoveBtns>().btnMenos = Resources.Load("btnZMenos") as Texture;
of.GetComponent<MoveBtns>().btnZRotate = Resources.Load("btnZRotate") as
Texture;
of.GetComponent<MoveBtns>().btnZRotateM = Resources.Load("btnZRotateM") as
Texture;

}

GUI.EndGroup ();

}

}

```

Esse código cria um campo texto para digitar o nome do objeto a carregar, que deverá ser informado na ferramenta desenvolvida, e cria também um botão que realiza a operação destruindo o objeto anterior se não for nulo – ou seja, se existir algum objeto previamente criado.

O código, ainda, cria uma cópia do *GameObject* que contém os *scripts* de leitura e importação do objeto 3D. Por fim, ele associa ao objeto instanciado o *script* que permite a movimentação dele através de botões e do teclado (no caso da versão para *PC*) e associa dinamicamente as imagens aos botões da interface gráfica do programa (*GUI*).

Deve ser ressaltado ainda que o acesso à *URL* remota na Unity é liberado apenas na versão paga da ferramenta.

## 5.2 Iluminação sobre o objeto

O primeiro requisito do protótipo trata do aspecto da iluminação, afim de que o objeto a ser examinado tenha uma boa visualização. Seus detalhes devem ser percebidos pelo usuário, o qual, com a movimentação do objeto, terá acesso a todas as suas faces.

A primeira problemática que se apresenta é saber que tipo de iluminação deve ser utilizada para permitir uma observação adequada. Pelo uso esperado para o protótipo, seria necessária uma luz que ficasse distante o suficiente do objeto para não ser captada pela câmera, encobrindo o objeto e atrapalhando a visão do usuário. Também seria necessário, essa não ofuscasse a câmera com seu brilho. Para tanto, a solução seria ser criar uma fonte

luminosa com grande intensidade e raio de abrangência. Desta forma, esta poderia ser posicionada bem longe da cena para visualização do objeto.

A Unity tem quatro tipos de iluminação: direcionada (*Directionallights*), ponto (*Point*), *spot* (luz dirigida sobre uma área circular) e *área* (*Area*). A luz direcionada assemelha-se à luz do sol e independentemente da sua distância, ilumina todos os objetos da cena. A luz do tipo ponto brilha igual em todas as direções, tendo um alcance limitado. A luz do tipo *spot* cria um cone de iluminação que afeta somente os objetos que estiverem na sua área de abrangência. Finalmente, a luz do tipo *área* ilumina em todas as direções numa área retangular, utilizada mais para efeitos de cena do tipo contraluz.

Existe também a possibilidade de anexar uma delas ao próprio objeto, o que não seria adequado para o tipo de resultado que se espera, por tratar-se de um visualizador de apenas um objeto por vez. Assim, como tal objeto pode mudar dinamicamente, essa opção significaria ter que criar a fonte de iluminação junto com ele toda vez que esse fosse carregado.

Com essas considerações, foram testadas todas as alternativas. Então, decidiu-se por usar um *Gameobject* de iluminação tipo direcionada, já que esse tipo de iluminação foi o que melhor destacou os detalhes em foco.

Dessa forma, uma vez criada a fonte de iluminação – que na Unity é também um *Gameobject* – ela foi direcionada e posicionada de forma que não atrapalhasse o conjunto. Como esse tipo de iluminação assemelha-se com a do sol, foi colocado bem distante do lugar onde o objeto permanece para visualização.

### 5.3 Rotação do objeto

Falou-se da rotação do objeto como requisito do protótipo. Obviamente, para poder examinar um objeto em três dimensões, deve-se poder rotacioná-lo de forma que as partes que ficam escondidas da câmera possam ser visualizadas pelo observador.

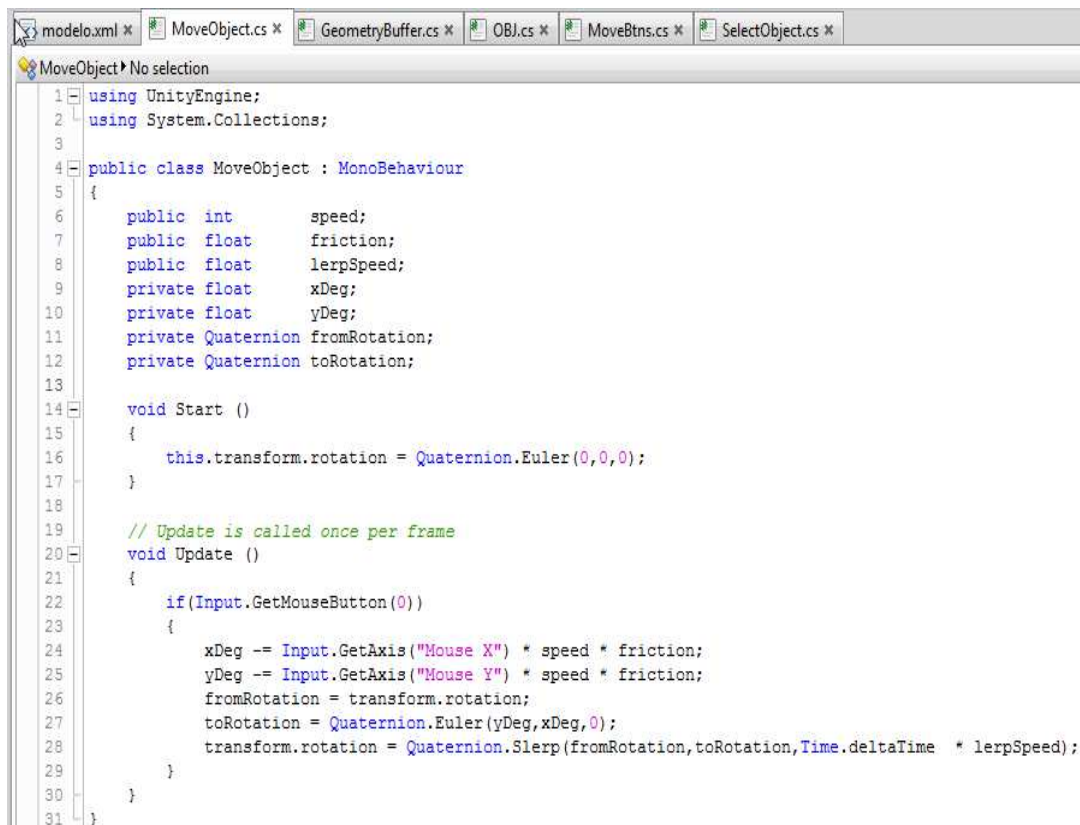
Como a tela onde os objetos são observados é plana, ela pode delimitar apenas dois eixos do plano  $\mathbb{R}^3$  – a saber, *X* e *Y*. Então, não é possível determinar com o clique do *mouse* ou com o pressionamento do dedo num ponto da tela, qual a profundidade no eixo *Z* que esse ponto se encontra. Dessa forma, as rotações nos eixos *X* e *Y* podem ser obtidas diretamente pela movimentação do *mouse* ou toque e arraste na tela dos dispositivos móveis, ao passo que a rotação no eixo *Z* deve ser possibilitada por outro recurso.

Para cumprir o requisito mencionado, deveria ser criada uma forma de rotacionar o objeto na direção e ângulo apontados pelo clique do mouse ou o toque na tela, aplicando no objeto as transformações de rotação já apresentadas no primeiro capítulo do presente trabalho.

Na Unity há várias formas de rotacionar objetos. Dentre elas destacam-se o *Quaternion*, que possibilita a rotação por seu método *rotation*. Seu uso mais comum implica na utilização de uma rotação existente para construir outras.

Outra forma de obter a rotação do objeto na Unity é utilizar, no *GameObject* que representa o objeto a ser visualizado, o método *rotate* da classe *Transform*. Ele necessita a informação dos três eixos que serão movimentados.

Então, fizeram-se testes com ambas alternativas (figuras 23 e 24). Neles observou-se que a movimentação obtida com a classe *Quaternion* não foi melhor que a obtida com a classe *Transform* – que foi, então, escolhida para atender ao requisito.



```

1  using UnityEngine;
2  using System.Collections;
3
4  public class MoveObject : MonoBehaviour
5  {
6      public int     speed;
7      public float   friction;
8      public float   lerpSpeed;
9      private float  xDeg;
10     private float  yDeg;
11     private Quaternion fromRotation;
12     private Quaternion toRotation;
13
14     void Start ()
15     {
16         this.transform.rotation = Quaternion.Euler(0,0,0);
17     }
18
19     // Update is called once per frame
20     void Update ()
21     {
22         if(Input.GetMouseButton(0))
23         {
24             xDeg -= Input.GetAxis("Mouse X") * speed * friction;
25             yDeg -= Input.GetAxis("Mouse Y") * speed * friction;
26             fromRotation = transform.rotation;
27             toRotation = Quaternion.Euler(yDeg,xDeg,0);
28             transform.rotation = Quaternion.Slerp(fromRotation,toRotation,Time.deltaTime * lerpSpeed);
29         }
30     }
31 }

```

Figura 23 – Trecho de código utilizado para testar a rotação do objeto 3D usando Quaternion.

A implementação final é apresentada na Figura 24 a seguir.



```

1  using UnityEngine;
2  using System.Collections;
3
4  public class MoveObject : MonoBehaviour
5  {
6      public int      speed;
7      private float   x;
8      private float   y;
9      // private Vector2 touch;
10
11     void Start ()
12     {
13         this.transform.rotation = Quaternion.Euler(0,0,0);
14     }
15
16     // Update is called once per frame
17     void Update ()
18     {
19         if(Input.GetMouseButton(0))
20         {
21             // touch = Input.GetTouch(0).deltaPosition;
22             x = Input.GetAxis("Mouse X");
23             y = Input.GetAxis("Mouse Y");
24             // x = touch.x;
25             // y = touch.y;
26             transform.Rotate (-y * speed, -x * speed, -0f);
27         }
28     }
29 }
30

```

Figura 24 – Trecho de código adotado no protótipo, utilizando a função Rotate da classe Transform.



Os trechos de código comentados na figura 24 (aqueles em verde, precedidos por ‘//’) fazem parte da solução para a plataforma móvel como *smartphone* ou *tablet* já que neles não é usado o *mouse*.

Por ser um motor para jogos, a Unity repete *frames* (quadros) por segundo. A função *Update* é chamada a cada quadro. Então para rotacionar o objeto, utiliza-se a função *Rotate* da classe *Transform* do *GameObject* dentro dela, passando como parâmetros o valor do eixo *X*, do *Y* e zero para o eixo *Z*.

A função *Rotate* aplica a rotação de *z* graus sobre o eixo *Z*, *x* graus sobre o eixo *X* e *y* graus sobre o eixo *Y*, nessa ordem, e é executada a cada frame tantas vezes por segundo como estiver configurada a *engine*. Então a rotação é repetida esse número de vezes com as informações ajustadas.

Quadro a quadro, as informações do eixo *X* e do eixo *Y*, são passadas para a função que por sua vez aplica a rotação, fazendo com que o objeto gire na direção da seta do *mouse* ou do toque na tela *touchscreen*. Como as informações que podem ser obtidas a partir do

clique com o *mouse* numa posição da tela ou pressionando um ponto com o dedo se o equipamento for de toque, são realizadas num plano 2D, pode-se apenas passar dados sobre o eixo *X* e *Y*, ficando as informações do eixo *Z* zeradas. Em outras palavras, com clique e arraste do *mouse* (na versão *PC*) ou toque e arraste na tela (na versão para dispositivos móveis) é possível realizar rotações apenas nos eixos *X* e *Y*.

Para informar o “3Dboard” da intenção de movimentar o objeto sobre o eixo *Z* foi criado um botão , que gira o objeto no sentido positivo – conforme explicado no capítulo um do presente trabalho – e um botão , que gira no sentido oposto (negativo).

Desta forma, acredita-se haver atingido o objetivo do requisito, dando liberdade ao usuário para rotacionar o objeto em todas as direções.

A Figura 25 mostra o efeito da rotação exemplificando o giro do objeto na direção do ponteiro do mouse pressionado.

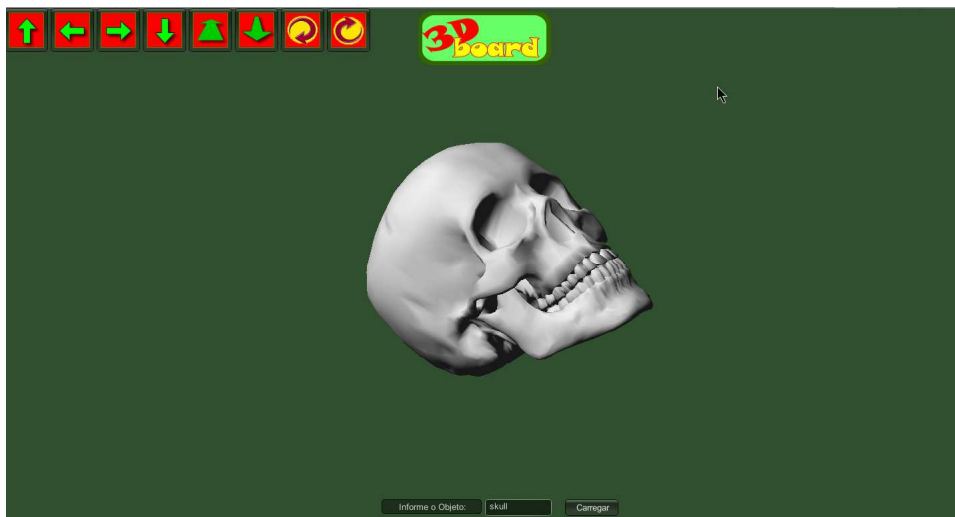


Figura 25 – Exemplo de tela capturada do “3Dboard” exemplificando a rotação nos eixos *XY*.



Figura 26 – Exemplo de tela capturada do “3Dboard” exemplificando a rotação no eixo Z+.

#### 5.4 Movimentação do objeto

Movimentar o objeto no sentido do eixo X e Y, tanto para o lado negativo quanto para o positivo, é outro requisito do protótipo, estipulado no capítulo 4.

Sendo a origem dos objetos de diversas fontes, muitas vezes esses podem não apresentar posicionamento desejado pelo usuário. Mesmo durante a utilização, é possível que algumas tarefas requeiram um posicionamento do objeto diferente do apresentado. Isso torna o requisito muito importante para que o usuário tenha a liberdade de manipular o objeto para alinhá-lo no centro da tela ou colocá-lo em outra posição que lhe agrade.

Lembrando-se mais uma vez do capítulo um, constata-se que a translação é a transformação apropriada para ser aplicada ao objeto visualizado a fim de atender-se ao requisito. Relembrando, o objetivo da translação é movimentar pontos do objeto da posição X, Y, Z no sistema de coordenadas para X', Y', Z'.

Enfrenta-se aqui a mesma problemática que na rotação, já que a tela onde será indicada a movimentação tem apenas duas dimensões (largura e altura), sendo possível indicar diretamente apenas as movimentações nos eixos X e Y. Desse modo, movimentar o objeto no eixo Z, acaba representando aproximação e afastamento do objeto em referencia à câmera, que é o foco do requisito discutido na próxima seção.

Para implementar a movimentação do objeto no protótipo, utilizou-se o método *translate*, da classe *GameObject*, oferecida pela Unity. Esse método encarrega-se da

matemática envolvida na translação de objetos e, em testes realizados durante o trabalho, apresentou o comportamento esperado para a tarefa. Na implementação, ele foi utilizado junto com a Classe *Vector3* e suas propriedades *up*, *left*, *right* e *down* para realizar as movimentações.

Contudo, além da simples aplicação da movimentação, foi preciso decidir quanto à escolha dos limites da movimentação (e.g. se seria possível ao objeto, inteiro ou em parte, posicionar-se fora dos limites da tela). Nesse ponto, considerando o usuário como quem tem o controle do objeto, decidiu-se por não impor tal tipo de limitação, deixando que ele decida se mantém o objeto em seu campo de visão ou não.

Ainda há que se pensar como movimentar o objeto. Utilizar o *mouse* ou o toque na tela para arrastar ele no sentido que se quer movimentar poderia causar problemas com o toque para rotacionar. Então, a solução implementada foi elaborada com a apresentação de quatro botões de movimentação (figura 27) que representam os quatro movimentos possíveis nos eixos *X* e *Y*.



Figura 27 – Representação dos botões utilizados no “3Dboard” para movimentar o objeto.

Na versão para PC, também foi implementada a movimentação com o uso das teclas de movimentação (i.e. setas do teclado), que realizam o movimento para a direção indicada por seu símbolo enquanto estiverem pressionadas. A Figura 28 apresenta uma imagem que sofreu movimentação para a esquerda utilizando-se os botões da tela.

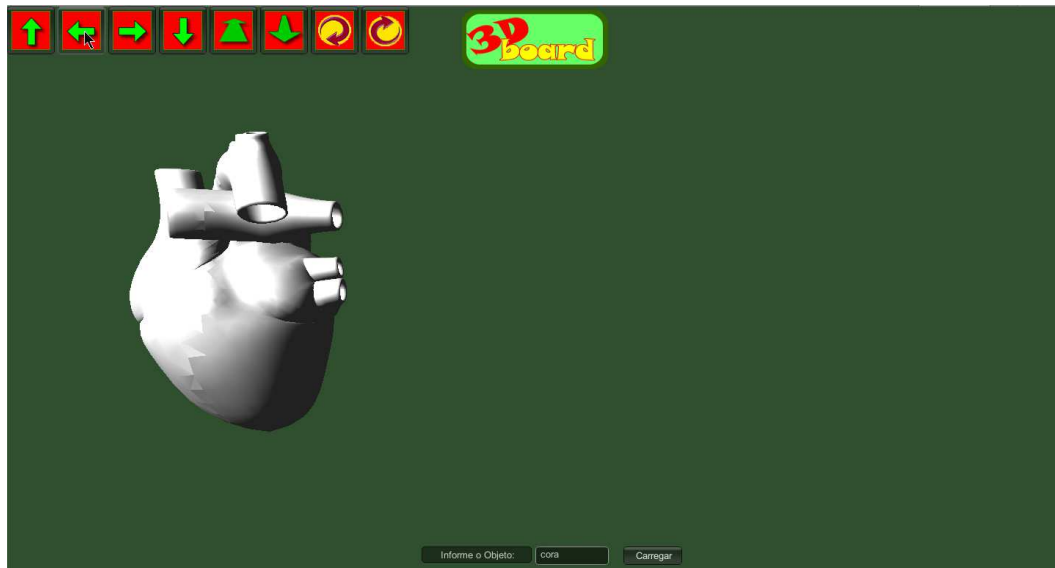


Figura 28 – Exemplo de tela capturada do “3Dboard” exemplificando a movimentação no eixo X-.

### 5.5 Aproximação ou afastamento do objeto

O próximo requisito estabelecido para o protótipo indica que deve ser possível afastar ou aproximar o objeto para seu estudo. Dado a dinâmica da projeção em perspectiva, esse tipo de movimentação tem efeito semelhante ao do *zoom*, fazendo aumentar ou diminuir o tamanho do objeto alvo na tela.

Entretanto, no ambiente simulado pelo protótipo, o que ocorre é a mudança da distância que o objeto mantém da câmera que o apresenta, ou do ângulo visual dela, o que causa a impressão de aumento ou diminuição. Assim, o que poderia ser feito para aplicar tal movimentação, tendo em consideração a existência de uma câmera fixa, seria uma translação do objeto no eixo *Z* – o eixo para o qual a câmera está posicionada, – o que representaria uma aproximação ou afastamento do objeto em referencia a ela. Uma alternativa seria movimentar a câmera de alguma forma em direção ao objeto, para aproximá-lo, ou na direção contrária, para afastá-lo.

Na Unity, o *Gameobject* câmera, tem a propriedade *Field of view (FOV)* que realiza a tarefa facilmente. É como se fosse acionada a opção *zoom* da própria câmera. Apesar de não mover o objeto em si, essa foi a solução adotada no protótipo por causar o efeito equivalente (i.e. deixar o objeto mais próximo ou mais distante da câmera). Por fim, a implementação deu-se pela inclusão de dois botões – um para aproximação, outro para afastamento. A Figura 29 mostra os botões utilizados para esse requisito.



Figura 29 – Botões para aproximação e afastamento do objeto.

As Figuras 30 e 31 exemplificam a diminuição e o afastamento do objeto usando os botões.

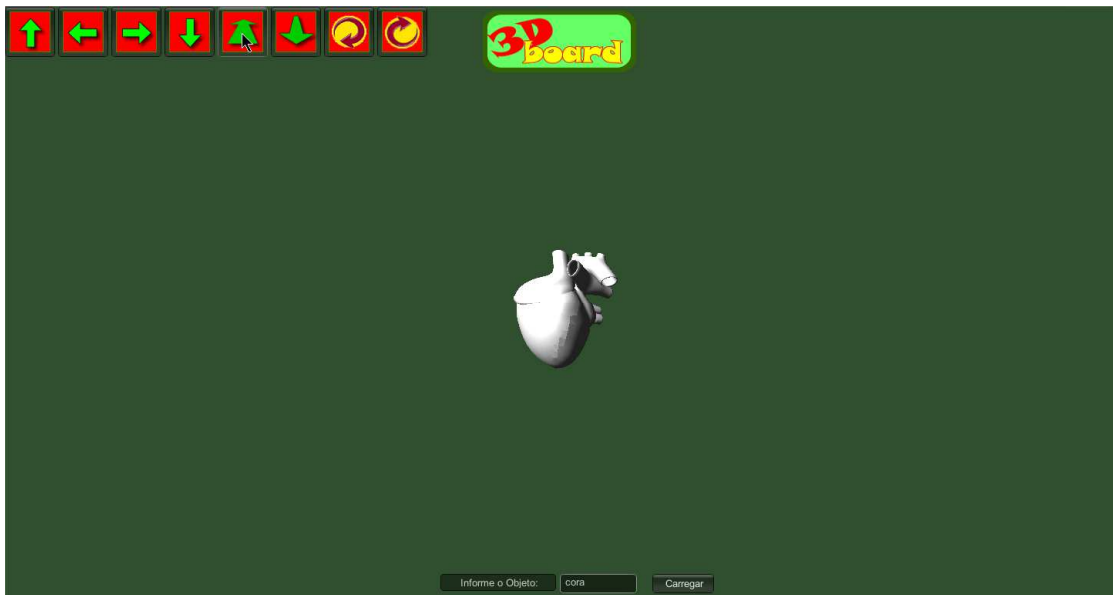


Figura 30 – Exemplo de tela capturada do “3Dboard” exemplificando o afastamento o objeto usando o botão.

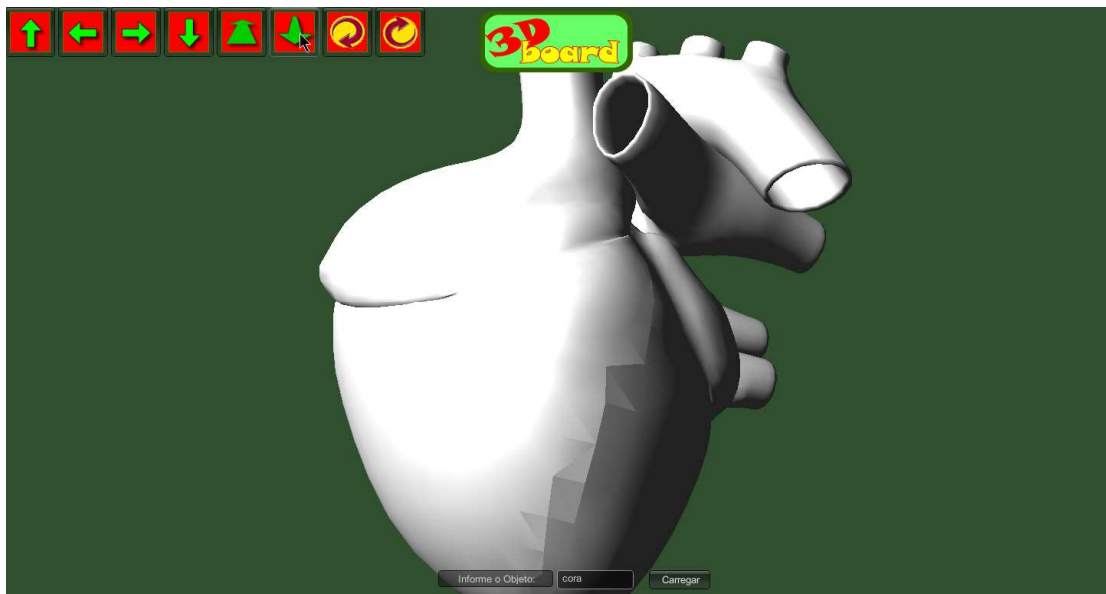


Figura 31 – Exemplo de tela capturada do “3Dboard” exemplificando a aproximação do objeto usando o botão.

## 5.6 Interface intuitiva

No último requisito, fala-se de obter uma interface intuitiva. A palavra intuição, vem do latim *intuitio* do verbo *intueor*, que significa olhar atentamente. Isso representa um conhecimento construído de forma imediata, sem a participação de terceiros. No caso, um conhecimento baseado em experiência, adquirido pelo uso.

Como fazer um protótipo de visualização 3D de forma que o usuário não necessite ler tutoriais para aprender a usá-lo? Ou ainda, que tipo de recursos utilizar para chamar a atenção dele no sentido de acioná-lo intuitivamente?

Deve-se utilizar conceitos amplamente assimilados pelos usuários de computadores, de *smartphones* e *tablets*. Quando se pensa em um usuário na frente de um objeto apresentado na tela do dispositivo eletrônico, observando ele flutuar no espaço como se não existisse gravidade, deve-se questionar: “Qual seria a primeira reação lógica do usuário?”.

Na maioria dos casos, seria a de tocar o objeto para ver o que acontece. No mundo real, se observamos uma folha em cima de uma mesa e apoiamos o dedo sobre ela, movimentando ele para um lado, a folha acompanhará o movimento. Se esse movimento for realizado em uma bola suspensa no ar, ela tenderá a girar a partir do ponto onde foi tocada. Considerando a natureza tridimensional do objeto no protótipo, a segunda metáfora fica mais plausível de aplicar. Assim, o toque e arraste na tela foi reservado para a rotação do objeto, oferecendo-se o recurso de movimentação por botões com setas – um símbolo que pode ser considerado universal para indicar movimento.

Entretanto, considerando a rotação, deve-se refletir sobre um efeito peculiar a essa metáfora: ao tocar e arrastar a bola suspensa no ar, ela irá girar e continuar girando mesmo depois de cessar o toque. Por questões de controle do objeto pelo observador, essa não é uma abordagem muito apropriada para um visualizador, devendo, portanto, o movimento ocorrer somente enquanto o objeto estiver sendo tocado. Como não há representação física do observador na cena, também não há representação do toque. Considerando, ainda, a tela de duas dimensões e a representação em três dimensões do objeto, não há como estabelecer uma relação não ambígua entre a posição do dedo enquanto toca a tela e a posição em que efetivamente ocorre o toque no objeto representado. Assim, para resolver esse impasse, decidiu-se que toda movimentação terá sempre o mesmo ponto de referência, como se o observador tocasse sempre o mesmo ponto do objeto, não importando suas posições (do observador e do objeto).

Fazendo-se uma comparação com um crânio, isso significaria rotacioná-lo sempre como se o usuário tivesse colocado seu dedo num ponto específico dele – digamos, do ponto imaginário que fica situado no meio das cavidades das orbitas oculares. Assim, sempre que ele quiser repetir o movimento, será como se o tivesse feito a partir desse mesmo ponto, ainda que o crânio esteja com a face voltada para o lado oposto ao da câmera.

Tem-se consciência de que essa seria uma dentre diversas soluções para esse problema. Entretanto, considerando a definição de um ponto fixo, formula-se a hipótese de que o usuário não tenha grandes dificuldades em constatar que as rotações nos eixo *X* e *Y* são sempre realizadas do mesmo ponto original – o que será objeto de avaliação e discussão no próximo capítulo.

Há que lembrar que o protótipo apresentado em realidade é virtual, apresentando três dimensões numa tela de duas. Isso impede o reconhecimento do toque no eixo *Z*. Assim, para haver uma rotação nesse eixo, o protótipo deveria prover uma forma de interação alternativa, sem ligação com as dimensões da tela, o que foi feito com a inclusão de dois botões na tela – um para cada sentido da rotação no eixo *Z*.

O objetivo de uma metáfora visual é o de representar alguma coisa que o usuário possa reconhecer e, por conseguinte, compreender seu significado e função. Assim, os símbolos utilizados na identificação dos botões foram selecionados de forma a induzir a ideia de movimento rotacional. Além disso, utilizou-se diferenciação do esquema de cores desses botões em relação àquela dos botões destinados à movimentação do objeto, a fim de enfatizar a diferença entre suas funcionalidades. A Figura 32 mostra os botões utilizarm o conceito metafórico referido para conseguir estabelecer uma ligação entre a imagem e a rotação no eixo mencionado.



Figura 32 – Botões representativos da rotação no eixo *Z*.

Quanto à de aproximação e afastamento, considerou-se que, ao utilizar o *mouse*, a primeira coisa que surge no pensamento é o uso da roda central – um padrão bastante utilizado, que parece já fazer parte do imaginário coletivo e que também foi utilizado neste trabalho. Entretanto, no caso do *smartphones* ou *tablets*, não há essa opção. Neste caso, pode-se pensar no movimento de “pinça” utilizado para ampliar e reduzir fotos.

Entretanto, como a ideia de *zoom* apresentada não é tratada como aumento ou diminuição do objeto, mas sim aproximação e afastamento, acreditou-se não ser essa uma



metáfora adequada, razão pela qual ela foi descartada. Em seu lugar, optou-se pela utilização de botões, um com o símbolo de uma seta apontando para o fundo da tela (para ser usado para afastar o objeto) e o outro com um símbolo apontando para frente da tela (para ser usado para aproximação do objeto). A Figura 33 apresenta os botões utilizados no trabalho para aproximação e afastamento do objeto.



Figura 33 – Botões utilizados no “3Dboard” para representar a aproximação e o afastamento do objeto.

Novamente, tem-se a hipótese que, pela natureza da interação, esses botões passarão a ideia de forma adequada – o que também será objeto de avaliação no próximo capítulo.

## 5.7 Deploy

Na Unity, a instalação do protótipo nas plataformas escolhidas é realizada de forma relativamente fácil. Existe toda uma parametrização que muda conforme a plataforma, e a ferramenta se encarrega de fazer um *build* apropriado para o resultado final.

No caso do protótipo, optou-se por manter duas versões, uma parametrizada para plataforma *WEB/PC* e outra para *smartphone* e *tablet*. As Figuras 34 a 37 exemplificam as opções da Unity no sentido de configurar o *Deploy* conforme o tipo de plataforma.

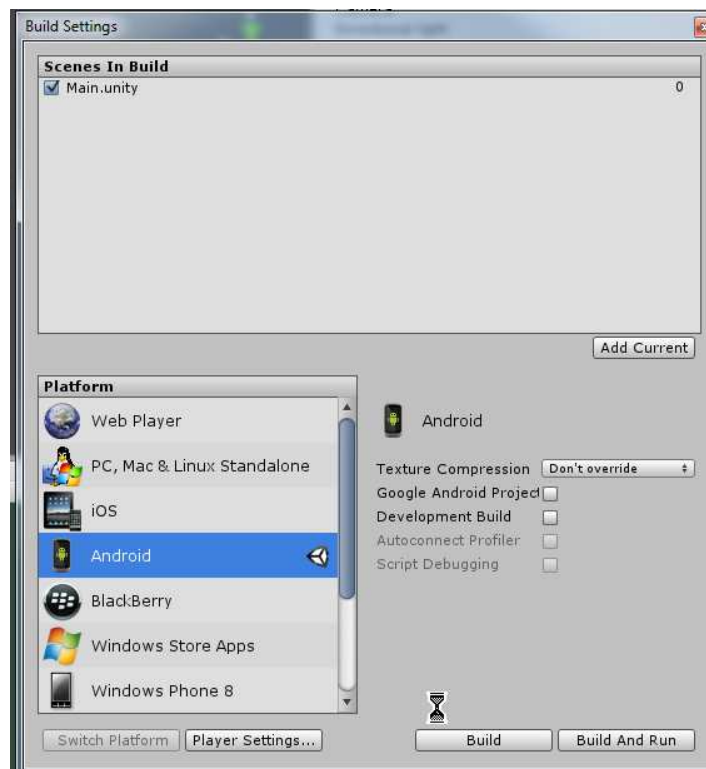


Figura 34 – Configurações para a construção do Deploy para o “3Dboard” Android.

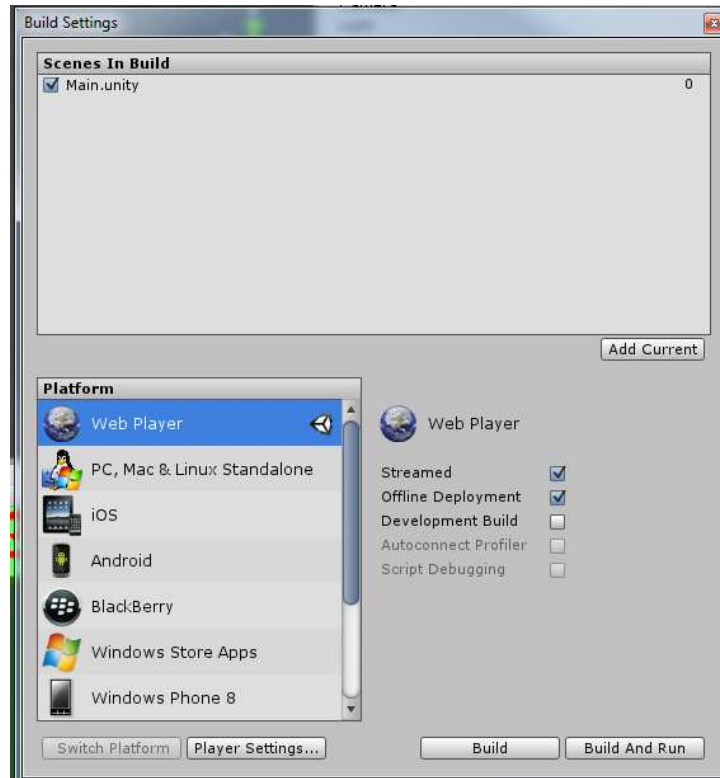
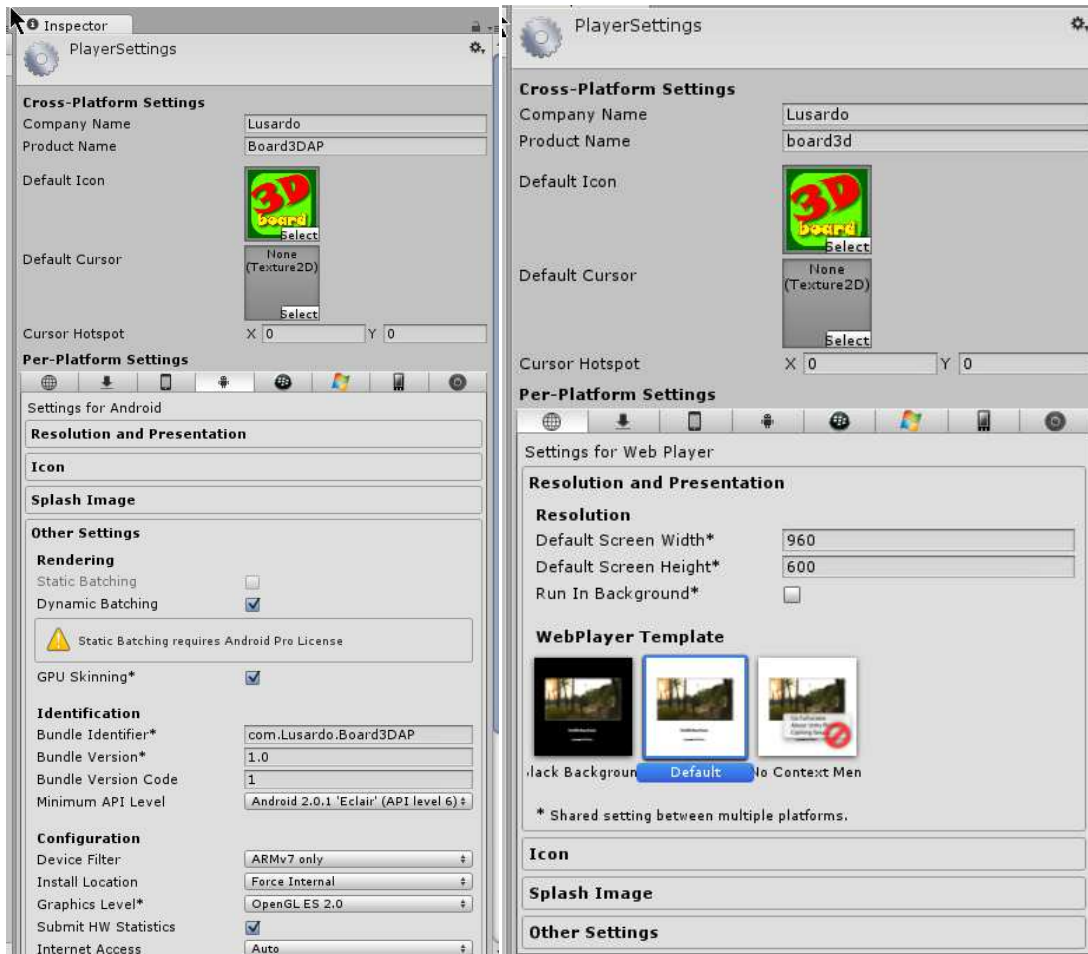


Figura 35 – Configurações para a construção do Deploy para o “3Dboard” PC-Web.



Figuras 36 (esquerda) e 37 (direita) – Exemplo das configurações do engine para o “3Dboard” para smartphone - Android e para PC-Web.

A seguir será apresentado o projeto e aplicação dos testes do protótipo, bem como a avaliação dos resultados no sentido de comprovar ou refutar as hipóteses levantadas no presente capítulo.

## 6 RESULTADOS E AVALIAÇÃO

No capítulo quatro do presente trabalho fixou-se como foco, a construção de um aplicativo intuitivo e fácil de usar, evitando-se, com isso, a necessidade de leitura de tutoriais para seu manuseio. Definiu-se também uma lista de requisitos que foram objeto de atenção durante seu desenvolvimento. A implementação de cada um dos requisitos foi apresentada no capítulo anterior. Entretanto, para ter-se condições de avaliar o quanto esses requisitos foram realizados de forma a alcançar a simplicidade e intuitividade desejada, foi necessária a realização de testes.

Para avaliar cada um dos requisitos do ponto de vista da intuitividade, foi preciso aplicar-se o teste com usuários reais. Além disso, procurou-se conseguir um número de voluntários suficiente para que a informação fosse representativa e permitisse tirar conclusões. A próxima seção apresenta o projeto dos testes aplicados e a seção seguinte os resultados obtidos com sua aplicação.

### 6.1 Projeto dos testes

Durante o projeto dos testes, a ideia central foi a de que os usuários deveriam manipular o protótipo sem instruções de como fazê-lo. Assim, se eles conseguirem realizar os movimentos sem ajuda, isso demonstraria que há intuitividade na operação do protótipo.

Uma alternativa para isso seria solicitar a realização de uma série movimentos fáceis, tarefas diretas como “gire o objeto para a direita” ou “movimente o objeto para cima”, mas esse tipo de instrução não simula condições reais. Uma situação real com certeza envolveria uma série de passos que deveriam ser executados para observar certo aspecto de um objeto e, possivelmente, responder a uma pergunta – elaborada por um professor ou de curiosidade própria, por exemplo.

Assim, considerando que a operação real da aplicação seria guiada por objetivos maiores em lugar de instruções diretas, pode-se afirmar que a intuitividade não fica restrita a apenas descobrir como fazer determinada ação, mas também intuir que ações podem ser realizadas a fim de atingir-se um objetivo.

Ou seja, se um dos itens do teste fosse, por exemplo, girar o objeto para a direita, a própria instrução já estaria dando uma informação que comprometeria o teste da intuitividade: o usuário ainda teria que descobrir como fazer para girar o objeto, mas a instrução já estaria dizendo que existe esta funcionalidade.

Por outro lado, se a instrução mandasse informar quantos dentes há num determinado crânio, o usuário deveria pensar o que necessitaria fazer para responder. Assim, se ele concluísse que deveria girar o objeto para facilitar a contagem, o protótipo fornecesse essa facilidade, e ele descobrisse por si só que isso é possível, e como se faz, então poder-se-ia dizer que a aplicação é intuitiva nesse aspecto.

Pensando-se nesse sentido, decidiu-se confeccionar roteiros para testes que observassem essas condições e que testassem cada um dos requisitos do protótipo. Considerando que uma das propostas do protótipo e seu funcionamento em *PC* na *web* e dispositivos móveis, construiu-se dois roteiros, um para cada plataforma.

As instruções do roteiro de testes para PC são as seguintes:

- 1) Carregue o objeto chamado *monkey*.
- 2) Movimente o objeto para que ele fique no canto superior direito da sua tela.
- 3) Mova o objeto para que ele fique encostado no botão carregar.
- 4) Faça com que o objeto fique encostado no botão com a seta verde apontada para a direita.
- 5) Coloque o objeto novamente no centro da tela.
- 6) Coloque-o olhando para a direita.
- 7) Coloque-o de perfil olhando para a esquerda.
- 8) Coloque o macaco olhando para você.
- 9) Faça o macaco ficar olhando para baixo.
- 10) Faça o macaco ficar olhando para cima.
- 11) Coloque o macaco de cabeça para baixo, olhando para você.
- 12) Aproxime o objeto o máximo que puder e deixe-o assim.
- 13) Carregue o objeto chamado *cora* e clique sobre ele.

Para a atividade 14, foi imposta uma restrição ao usuário: “sem clicar nos botões da tela”, a fim de obter-se um retorno quanto ao uso dos recursos que o hardware oferece como o teclado e o mouse com sua roda central.

- 14) Afaste o objeto o máximo que puder.

15) Conte quantos buracos existem no objeto.

Cada atividade foi projetada para testar um ou mais requisitos e foram agrupadas desta forma:

- I. A atividade (1) testa o requisito 'a' pedindo para carregar um objeto. Ressalte-se que aqui há uma violação à regra de não revelar o que o protótipo pode fazer. Nesse caso, porém, não há como incluir a atividade numa seqüência maior e desta forma obrigar o usuário a deduzir como é realizado o carregamento dinâmico dos objetos no protótipo.
- II. As atividades (2) a (5) testam a movimentação do objeto nos eixos *X* e *Y* (requisito 'e').
- III. As atividades (6) a (11) testam a rotação (requisito 'd').
- IV. A atividade (12) testa a aproximação do objeto (parte do requisito 'f') e, em seguida, a atividade (13) verifica, novamente, o carregamento dinâmico dos objetos (requisito 'a'). Aqui novamente não se respeitou a regra de não dar ordens diretas, mas isso em razão de preparar-se a cena para a atividade (14).
- V. A atividade (14) testa o afastamento do objeto (parte do requisito 'f') e o emprego de meios intuitivos para a funcionalidade (requisito 'g'), já que este exercício deve ser realizado sem clicar nos botões.
- VI. A atividade (15) testa o requisito 'd' e o requisito 'f', pois para realizar essa instrução o usuário deverá rotacionar, aproximar ou afastar o objeto de forma que facilite a identificação deles.

Interessante ressaltar que a atividade (14), no grupo VI, ao impedir o pressionamento de botões, quebram as ligações intuitivas que vinham sendo construídas anteriormente já que o usuário terá que concluir que deve haver outros meios para realizar o exercício, tentando descobri-los e experimentá-los.

Também é importante frisar que não há atividades que testem o requisito 'b' (iluminação adequada) e 'c' (posicionamento do objeto à frente da câmera). No caso do requisito 'b' isso se deve ao fato de a iluminação ser um aspecto presente em todas as atividades, não necessitando de uma específica.

Quanto ao requisito ‘c’, por ser estritamente técnico, tem-se como garantida sua realização pela própria implementação. Da mesma forma, apesar de o requisito ‘g’ ser o foco do teste – já que diz respeito justamente à intuitividade da interface – foi dado destaque a ele em algumas tarefas, projetadas especialmente para possibilitar o uso de interfaces específicas.

Para o teste em *smartphones* e *tablets* foi desenvolvido um segundo roteiro utilizando-se um objeto diferente. Nessa versão o protótipo já abre com o primeiro objeto posicionado no centro da tela. Aqui, ainda, a mudança de objetos não é dinâmica dado que a versão gratuita da Unity bloqueia essa opção. Assim, a troca de objetos ocorre com o pressionamento de um botão com essa função. As atividades do segundo teste são (a numeração segue a do primeiro roteiro, pois todas as atividades foram consideradas como participantes de um único teste da aplicação com um todo):

- 16) Faça com que o objeto apresentado saia da visão na tela.
- 17) Agora, faça com que o objeto fique debaixo do logotipo.
- 18) Carregue o próximo objeto.
- 19) Afaste o objeto o máximo que puder.
- 20) Aproxime o objeto e centralize-o.
- 21) Observe a parte de cima do crânio.
- 22) Coloque o objeto olhando para a direita.
- 23) Faça o objeto girar de forma que ele permaneça olhando para a direita.
- 24) Informe quantos dentes tem esse crânio.

Novamente as atividades objetivam testar um ou mais requisitos, sendo agrupadas desta forma:

- I. Atividades (16) e (17) testam a movimentação do objeto nos eixos *X* e *Y* (requisito ‘e’).
- II. Atividade (18) testa o carregamento do objeto (requisito ‘a’).
- III. Atividades (19) e (20) testam o afastamento e a aproximação do objeto (requisito ‘f’).
- IV. Atividades (21) a (23) testam a rotação do objeto (requisito ‘d’).

- V. Atividade (24), que tem potencial para testar rotação do objeto (requisito ‘d’), bem como afastamento e aproximação (requisito ‘f’), já que são essas atividades necessárias para completar a tarefa.

Pelos mesmos motivos apresentados na descrição do roteiro anterior, não há atividades específicas para testar os requisitos ‘b’, ‘c’ e ‘g’.

Tendo roteiros capazes de testar cada um dos requisitos, esses podem ser submetidos a usuários voluntários. Os usuários, após a manipulação dos objetos, têm todas as respostas necessárias para avaliar os resultados. Assim, é necessário obter deles essas informações de maneira que se tenha uma compreensão quanto ao nível de atendimento aos requisitos. Para obter essa informação, foram criados questionários, cujas perguntas dividem-se em quatro tipos, com os objetivos de:

- 1) Perguntas de perfil.
- 2) Avaliar a intuitividade de cada tarefa.
- 3) Verificar a compreensão do usuário sobre o que ele deveria fazer.
- 4) Avaliar aspectos gerais sobre o uso do protótipo.

As perguntas do tipo 1 tem o objetivo de avaliar a amostra de voluntários, com perguntas sobre idade e afinidade do usuário com a internet e com dispositivos móveis. Acredita-se que essas variáveis podem ter grande influência na percepção da aplicação pelos voluntários, sendo, então, importantes para ponderar corretamente as respostas e observações que esses venham a fazer durante o teste.

As perguntas do tipo 2 avaliam cada requisito e envolvem tanto tarefas simples quanto as mais elaboradas. Como já dito, os requisitos foram avaliados em relação a sua intuitividade. Entende-se algo como intuitivo se for fácil de utilizar. Então, considerou-se uma escala de dificuldade como adequada para medir as respostas às perguntas do tipo 2.

Decidiu-se por usar três níveis nessa escala (i.e. “Fácil”, “Razoável” e “Difícil”), por considerar suficiente esse nível de especificação e para facilitar, dessa forma, a avaliação pelo usuário, que teria menos classes para enquadrar sua resposta. Isso resultaria em um padrão de avaliação mais consistente entre usuários e menor fadiga durante a realização do teste – o que se considera útil para preservar a qualidade das respostas.

As do tipo 3 tem como objetivo saber se o usuário compreendeu o que foi solicitado. Peritindo assim o descarte de informações sem relevância, a exemplo de respostas dadas por



quem não compreendeu o que tinha que ser feito e, por conseguinte, realizaram uma operação diversa do esperado, não podendo sua resposta ser mapeada com segurança a algum requisito do protótipo. Essas perguntas são de resposta aberta, então, não tem alternativas.

Finalmente, as perguntas do tipo 4 tem o objetivo de captar a opinião geral sobre o protótipo e a intuitividade de seu uso.

Para compor o formulário para avaliação do protótipo (apêndice I), as perguntas foram dispostas ao longo do roteiro de testes. No início do formulário foram colocadas as perguntas do tipo 1, solicitando que o usuário informe sua idade e afinidade com o uso da internet e de dispositivos móveis. As perguntas do tipo 2 foram colocadas ao final de cada grupo de atividades (de I a VI no teste 1 e de I a V no teste 2).

Como exemplo, após o grupo II do teste 1 foi feita a seguinte pergunta “o que você achou da tarefa, instruções (2) a (5)?”, oferecendo as opções “Fácil”, “Razoável” e “Difícil” e duas linhas abaixo da pergunta para que o usuário escrevesse e justificasse sua resposta, caso essa não tivesse sido “Fácil”. Analogamente, foi colocada a pergunta “O que você achou da iluminação?”, com as opções “Boa”, “Razoável” e “Ruim”, ao final de cada teste 1, também com um espaço de duas linhas para justificar respostas diferentes de “Boa”.

As perguntas do tipo (3) foram colocadas após atividades específicas, da seguinte forma:

- Após a atividade 8 perguntou-se “Qual o formato do nariz dele?” (do objeto que estava sendo analisado) para verificar se o usuário realizou completa e corretamente as atividades anteriores. Para essa pergunta, a resposta esperada é “coração” ou “arredondado”;
- Após a atividade 11 perguntou-se “Que você usou para cumprir com a última instrução?”, para verificar se o usuário realmente utilizou o botão de rotação no eixo Z. A resposta esperada para essa pergunta é o uso de qualquer combinação de rotações;
- A atividade 15 em si representa uma pergunta do tipo 3, pois, em se tratando de um desafio simples, uma resposta que divirja muito do número real de buracos indica que não se realizou a atividade proposta corretamente. O valor esperado para essa pergunta é 9, aceitando-se variações de até 20% (que podem ser atribuídas a confusões de visualização devido à própria construção do objeto);

- De forma análoga, a atividade 24 também representa um pergunta do tipo 3. A resposta esperada para essa pergunta é 32, também tolerando-se até 20% de variação, pelos mesmos motivos anteriores.

Finalizando o formulário, as perguntas do tipo 4 foram colocadas após cada teste. Foram feitas as seguintes perguntas:

- “O que você achou do teste 1 como um todo (questões 1 a 15)?” e “O que você achou do teste 2 como um todo (questões 16 a 24)?”. Para essas, utilizou-se uma escala de 5 níveis, de “Fácil” a “Difícil”, incluindo os valores “Razoavelmente fácil” e “Razoavelmente difícil” para captar manifestações intermediárias que pudessem ocorrer (e.g. voluntários que tivessem marcado quantidades semelhantes de respostas “Fácil” e “Razoável” ou “Razoável” e “Difícil” ao longo das atividades);
- “Você considera que a forma de interação (botões, toques e gestos) com o sistema na versão para *smartphone* e intuitiva e fácil de assimilar?”, ao final do teste 2. Para essa questão, foram oferecidas as opções “Concordo plenamente”, “Concordo parcialmente”, “Não concordo nem discordo”, “Discordo parcialmente” e “Discordo plenamente”. Também foram acrescentadas duas linhas para que o usuário explicasse o motivo de sua eventual discordância.

No final do formulário, também pediu-se para que o voluntário avaliasse o protótipo (“Por favor de uma nota de 1 a 10 para o programa no que se refere a usabilidade e interface amigável”).

É importante fazer menção à finalização da construção do formulário do apêndice I. Este não foi concebido diretamente dessa forma. Primeiramente fez-se uma versão do formulário, que foi aplicado em um teste piloto, com sete pessoas do CAS (Grupo de Computação Aplicada a Saúde da Feevale), para comprovar a eficiência dos roteiros, avaliar sua clareza e a utilidade do formulário.

Com esse piloto, foram reformulados os textos de algumas atividades e perguntas e resolvidos alguns problemas. Dentre esses problemas, destaca-se o de ter utilizado para as perguntas do tipo (4) do teste piloto uma escala uma escala de 5 níveis assim distribuída: “Muito fácil”, “Fácil”, “Médio”, “Difícil” e “Muito difícil”.

Percebeu-se que os valores extremos não foram utilizados (provavelmente porque mesmo alguém marcando todas as atividades como “Fácil” ou “Difícil”, no máximo iria utilizar “Fácil” ou “Difícil” para classificar o teste como um todo).

Também se notou a necessidade dos valores intermediários, que passaram a ser utilizados (como descrito anteriormente). Além disso, foram ajustados os textos de algumas instruções que estavam induzindo os usuários a erro.

Com o formulário de avaliação pronto, foi realizado o teste definitivo. Esse foi realizado com uma amostra de 28 estudantes de graduação da Universidade Feevale. Desses, 3 são do sexo feminino e 25 do sexo masculino, com idades e afinidade com o uso de internet e dispositivos móveis distribuídos conforme apresentado nos gráficos a seguir.

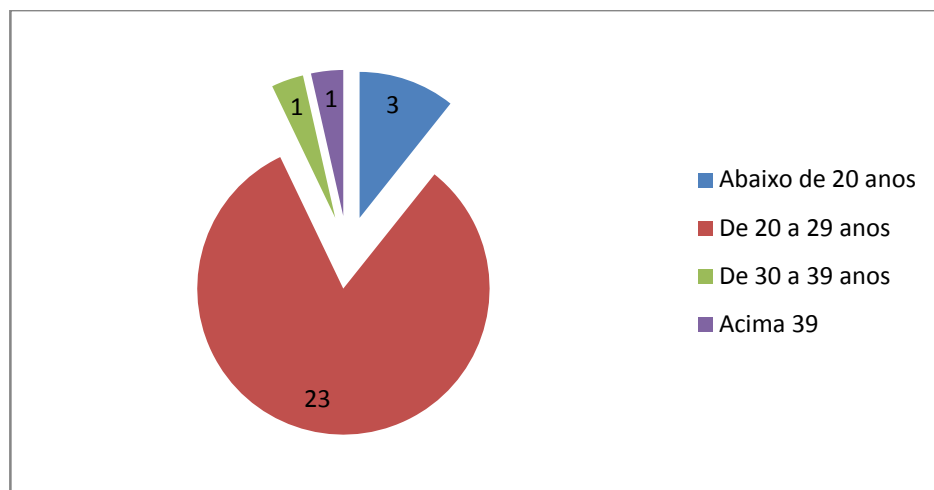


Figura 38 – Gráfico representativo da afinidade com a internet e dispositivos móveis dos voluntários que realizaram o teste do protótipo.

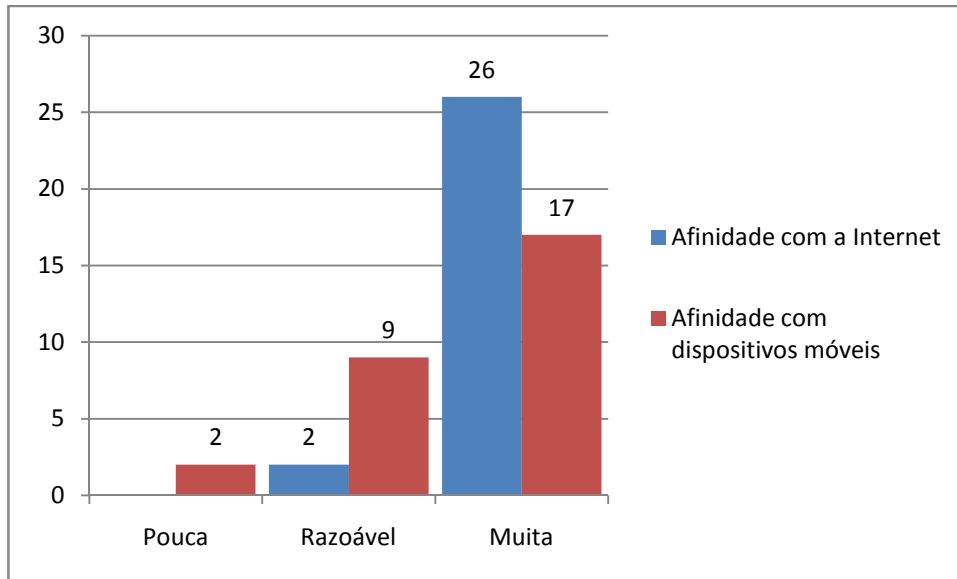


Figura 39 – Gráfico representativo da afinidade com a internet e dispositivos móveis dos voluntários que realizaram o teste do protótipo.

A avaliação dos resultados para cada um dos roteiros é apresentada a seguir.

## 6.2 Avaliação dos resultados para o roteiro 1 (versão para PC/Web)

Na avaliação geral do teste 1, feita por uma pergunta do tipo 4 ao final desse teste, obteve-se o resultado apresentado na figura a seguir.

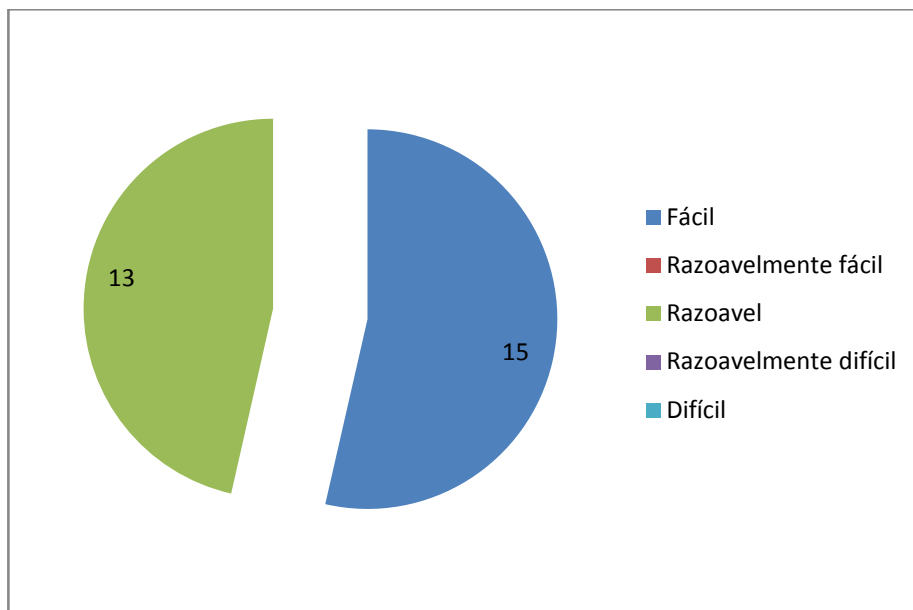


Figura 40 – Resultado da avaliação geral do teste 1

Primeiramente, é necessário dizer que todos os voluntários responderam corretamente às perguntas de compreensão (tipo 3), evidenciando que todos compreenderam a tarefa a ser realizada e, portanto, tem suas respostas válidas.

Nota-se que a maioria dos usuários (53,6%) considerou fácil a realização das atividades propostas, o que é uma boa avaliação para o protótipo. Entretanto, uma parcela considerável assinalou razoável a dificuldade do teste como um todo.

Para averiguar a razão dessas opiniões, bem como avaliar individualmente o sucesso da implementação de cada um dos requisitos, a seguir serão analisadas as respostas para as tarefas, agrupadas por requisito da aplicação.

### 6.2.1 Avaliação do requisito ‘a’ – Carregamento dinâmico do objeto

No teste 1, o requisito ‘a’ foi avaliado nas atividades (1) (grupo I) e (13) (grupo IV). O gráfico da Figura 41 agrupa as respostas dadas às perguntas que avaliaram os grupos dessas atividades.

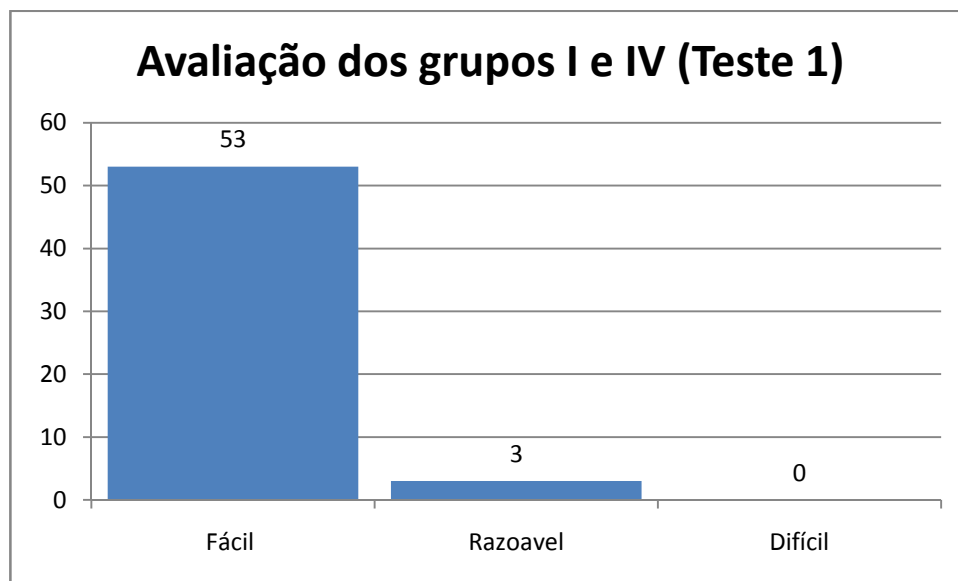


Figura 41 – Avaliação sobre o carregamento dinâmico dos objetos no teste 1 (agrupando-se as avaliações sobre os grupos de atividades I e IV).

Como pode ser visto, 95,6% das respostas consideraram fácil a utilização da funcionalidade do requisito ‘a’.

Em relação às 3 respostas que consideraram-no “Razoável”, uma das justificativas dos voluntários para esse julgamento dizem respeito à falta de uma lista de objetos para

escolha de qual carregar. Entende-se que o motivo é plausível e pode ser objeto de estudo para futuros trabalhos. Contudo, ele não afeta o teste do requisito do ponto de vista da intuitividade, pois o usuário conseguiu realizar a tarefa proposta, mesmo não existindo a lista sugerida.

Outra justificativa foi a utilização da palavra *monkey* para nomear o objeto – que, por estar em inglês, poderia não ser compreendida. Sendo livre a escolha dos nomes dos objetos pelo o usuário que os forneça, entende-se que essa cautela não faz parte do escopo do trabalho.

Assim, dado o expressivo número de avaliações positivas e considerando que as críticas não estão relacionadas a intuitividade na relação da tarefa, pode-se considerar que o requisito ‘a’ foi implementado de forma intuitiva na versão para *PC* do protótipo.

#### 6.2.2 Avaliação do requisito (b) – Iluminação do objeto

O requisito ‘b’ do protótipo foi avaliado por uma pergunta direta, ao final do roteiro do teste 1. Os resultados são apresentados no gráfico a seguir.

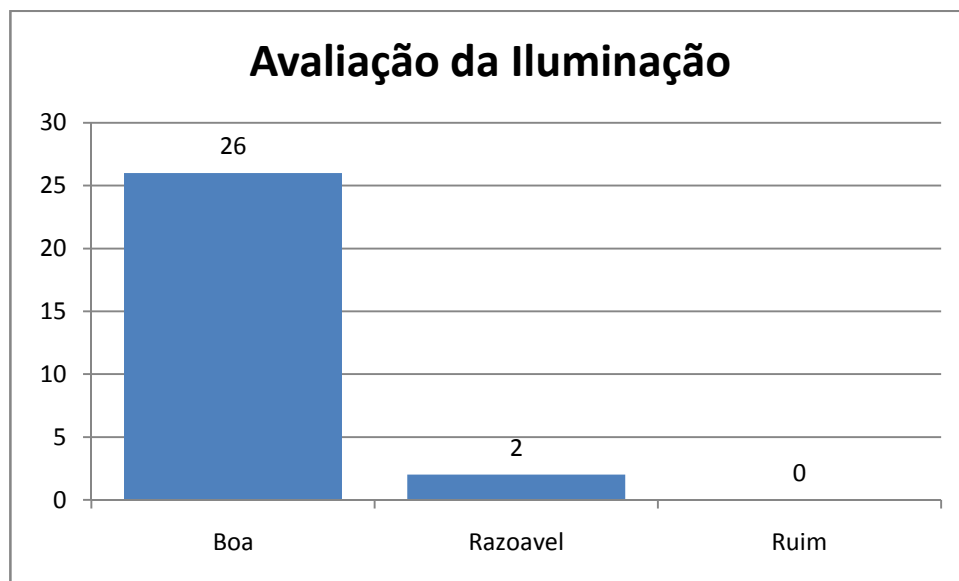


Figura 42 – Resultado para a pergunta sobre a iluminação no teste 1.

92,9% das respostas consideraram boa a iluminação do objeto no protótipo. As justificativas para as duas respostas que consideraram “Razoável” a iluminação dão a sugestão para que o usuário possa configurar o ponto de incidência e a intensidade da luz, assim como aplicar alguns efeitos básicos.

É fato que se houvesse mais pontos luminosos na cena, com regulagem ajustável, o objeto poderia ser visto com maiores detalhes, e isso enriqueceria a aplicação. Entretanto, isso poderia vir ao custo de perda da simplicidade que pautou o projeto.

De toda forma, considerando a predominância de avaliações positivas sobre a iluminação e o fato de que as observações dos voluntários não constituem uma reclamação, mas uma sugestão para aprimoramento do trabalho no futuro, tem-se por atingido o requisito 'b'.

### 6.2.3 Avaliação do requisito 'c' – Posicionamento do objeto à frente da câmera.

Como defendido anteriormente, esse requisito consiste em uma questão puramente técnica, cujo sucesso pode ser aferido diretamente de sua implementação, sem necessitar da avaliação dos usuários. Assim, esse requisito não foi contemplado no teste.

### 6.2.4 Avaliação do requisito 'd' – Rotação do objeto

O requisito 'd' é testado especificamente pelas atividades (6) a (11) (grupo III) e, também, pela atividade 14, ainda que em conjunto com outros requisitos. O gráfico a seguir mostra o resultado das avaliações para o grupo III.

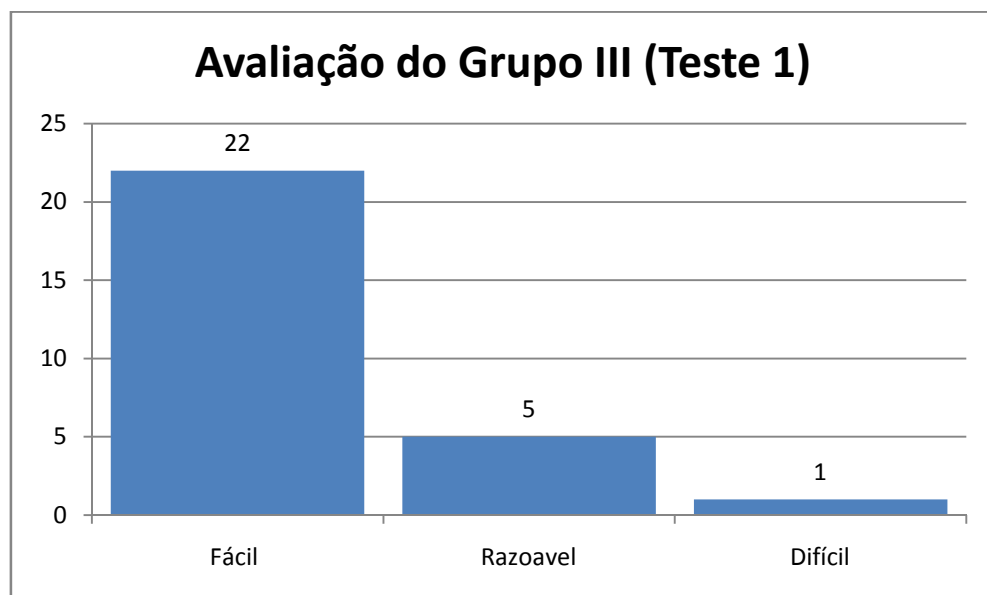


Figura 43 – Avaliação sobre rotação dos objetos no teste 1

Apesar de a maior parte (78,6) ter considerado fáceis as atividades que envolveram rotação, houve uma discordância expressiva quanto a esse ponto. Dentre as justificativas dadas por aqueles que não consideraram fáceis as tarefas, 2 disseram que não sabiam o que o objeto poderia fazer e outros 2 disseram ter que realizar vários movimentos e cliques e 1

relatou um comportamento que achou estranho na rotação do objeto (i.e. a rotação parece invertida, dependendo da posição do objeto).

Essas observações demonstram que tanto a descoberta de o quê a aplicação pode fazer quanto de como realizar a tarefa pode não ser tão fácil quanto se supôs na hipótese levantada na metáfora de rotação apresentada no capítulo 4.

O sexto usuário divergente sentiu falta de botões para rotação em todos os eixos ( $X$ ,  $Y$ ,  $Z$ ). Além de fazer coro com os demais, isso também indica uma possibilidade de incremento para o trabalho, a ser analisada no futuro.

Pelo exposto, ainda que mais de  $\frac{3}{4}$  dos voluntários tenham considerado a rotação intuitiva, as ressalvas apontadas pelos demais permitem concluir que o requisito 'd' não foi implementado de forma completamente intuitiva, o que contraria a hipótese levantada no capítulo 4.

Esse resultado ganha maior peso ao se considerar que, dentre os voluntários, 92,9% declararam ter bastante intimidade com a utilização de aplicações web. Assim, torna-se necessário o estudo de novas metáforas para a rotação do objeto (e.g. utilizar um ponto fixo em frente à câmera como origem para todo estímulo de rotação no objeto), a serem objeto de trabalhos futuros.



### 6.2.5 Avaliação do requisito ‘e’ – Movimentação do objeto

As atividades (2) a (5) testaram a movimentação do objeto nos eixos X e Y (requisito ‘e’) e fazem parte do grupo II do primeiro roteiro. O gráfico das avaliações sobre esse grupo é apresentado a seguir.

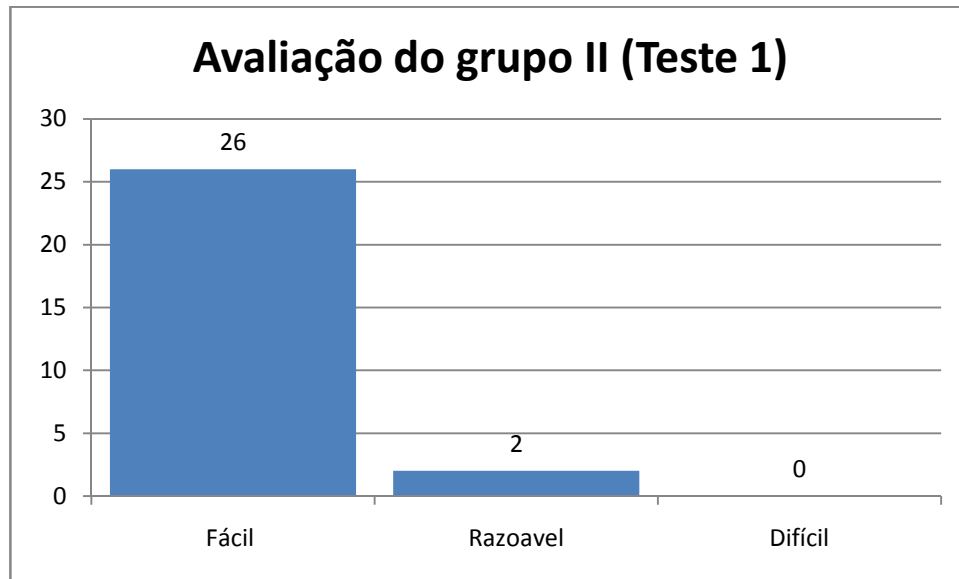


Figura 44 – Avaliação sobre a movimentação dos objetos no teste 1.

92,9% das avaliações consideraram fáceis as atividades de movimentação. Quanto aos comentários das respostas divergentes, um dos usuários observa que a posição dos botões de movimentação pode ser colocada em cruz. Isso que pode ser objeto de estudo futuro, mas não afeta a intuitividade nem a funcionalidade da movimentação.

A outra resposta divergente focou-se no objetivo da tarefa em si, justificando a resposta por a movimentação ser muito sensível ao toque do botão. É possível que esse aspecto realmente influencie no conforto e na realização das atividades e certamente deverá ser considerado em trabalhos futuros. Contudo, pela definição do que se entende por intuitividade, tem-se que não seja uma crítica que desabone o trabalho nesse quesito.

Tendo em consideração os comentários analisados e as respostas expostas no gráfico, pode se dizer que, na versão PC do protótipo, o requisito ‘e’, sobre o ponto de vista da intuitividade, foi atingido com sucesso. Todos os usuários pressionaram os botões na tela para movimentar o objeto.

### 6.2.6 Avaliação do requisito ‘f’ – Aproximação e afastamento do objeto

A atividade (12), que faz parte do grupo IV no teste 1, testou a aproximação do objeto (parte do requisito ‘f’), cujos valores estão contidos na figura 41. O único usuário que marcou “Razoável” colocou em seu comentário que deveria haver uma opção para listar os objetos que podem ser carregados. Isso isenta a aproximação do objeto de culpa pela avaliação “Razoável”, colocando seu *status* em 100% de respostas como “Fácil”.

A atividade (14), que integra o grupo VI do teste 1, testou o afastamento do objeto, (também parte do requisito ‘f’), acrescida de uma dificuldade pela limitação de não poder-se clicar nos botões na tela. Seu gráfico é apresentado a seguir:

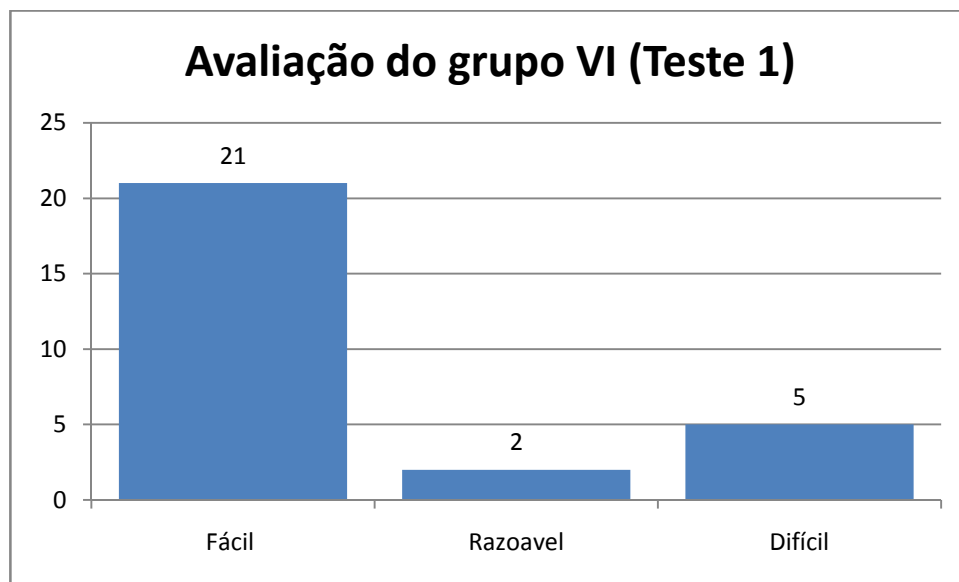


Figura 45 – Avaliação sobre o afastamento dos objetos no teste 1.

75% dos avaliaram a tarefa de afastamento como “Fácil”. Das sete avaliações que não consideraram a tarefa assim, duas vieram de usuários de *notebook*.

Observa-se aqui um problema sério. Os usuários que utilizaram *notebook* e que não tinham *scroll* no seu *touchpad* não conseguiram realizar a tarefa por causa da dificuldade imposta de não poder clicar os botões da tela. Um deles identificou a limitação e utilizou um mouse auxiliar. Assim, apesar de não impactar na intuitividade do recurso, pode-se apontar com essas observações uma falha na usabilidade. Entretanto, havendo o recurso de aproximação e afastamento nos botões da tela, não há necessidade de maiores preocupações.

Nos comentários dos outros usuários que não consideraram a tarefa fácil percebe-se que a tentativa geral foi a de utilizar o teclado. Atribui-se isso à algum problema induzido

pelo enunciado da atividade, que ao limitar o uso de cliques nos botões da tela, pode ter dado a entender que o *mouse* não deveria ser utilizado.

Além disso, tendo em vista que as dificuldades apontadas pelos usuários dizem respeito ao requisito ‘g’ (utilização de gestos, elementos gráficos e recursos de hardware), aqueles que marcaram “Difícil” e “Razoável” não estavam se referindo ao afastamento em si, mas a impossibilidade de realizar o movimento pela falta de um recurso no seu equipamento. Assim, não se consideraram essas respostas para a avaliação de intuitividade do requisito.

Assim, no julgamento do requisito ‘f’, pode-se dizer que foi atingido o seu objetivo, tendo sido esse implementado de forma intuitiva.

Abre-se um parêntesis, porém, para destacar o comentário de um usuário nas perguntas do grupo (4) sobre a intuitividade em si, que diz “Os botões de aproximar e afastar não estão claros. Os de movimentação poderiam estar melhor dispostos, em cruz”. Isso de certa forma contradiz os resultados até aqui expostos sobre o uso dos botões de afastamento e aproximação, levantando dúvidas sobre a confirmação da hipótese levantada no capítulo 4. Entretanto, o rotundo 100% de êxito na operação em que foi possível o uso dos botões se contrapõem a esse comentário. De qualquer forma, alternativas para a representação das ações de aproximação e afastamento deverão ser objeto de estudo para trabalhos futuros.

#### **6.2.7 Avaliação do requisito ‘g’ – Utilizar os gestos de interface difundidos, elementos gráficos intuitivos e opções de hardware.**

Repassando os comentários gerais obtidos no sentido de contribuir com a formação de um parecer sobre se a utilização de gestos de interface difundidos, elementos gráficos intuitivos e as opções de hardware para o roteiro 1, pode-se dizer que para a versão *PC/Web* o objetivo foi alcançado. Tendo apenas uma observação a fazer: deve ser visto mais um substituto para a utilização do *scroll* do *mouse* utilizado para aproximação e afastamento dos objetos sem o uso dos botões apropriados para isso.

No sentido de haver atingido um bom grau de intuitividade nesse quesito, pode se dizer que mesmo não estando presente no hardware de alguns usuários, esta ferramenta, eles deram conta da sua falta e a mencionaram nos comentários como uma limitação.

### 6.3 Avaliação dos resultados para o roteiro 2 (versão para dispositivos móveis).

A aplicação do formulário de avaliação na amostra também avaliou o protótipo em sua versão para dispositivos móveis, utilizando o roteiro 2. O resultado geral quanto às atividades foi obtido por uma pergunta do tipo 4 realizada ao final do teste, tendo-se obtido os resultados a seguir.

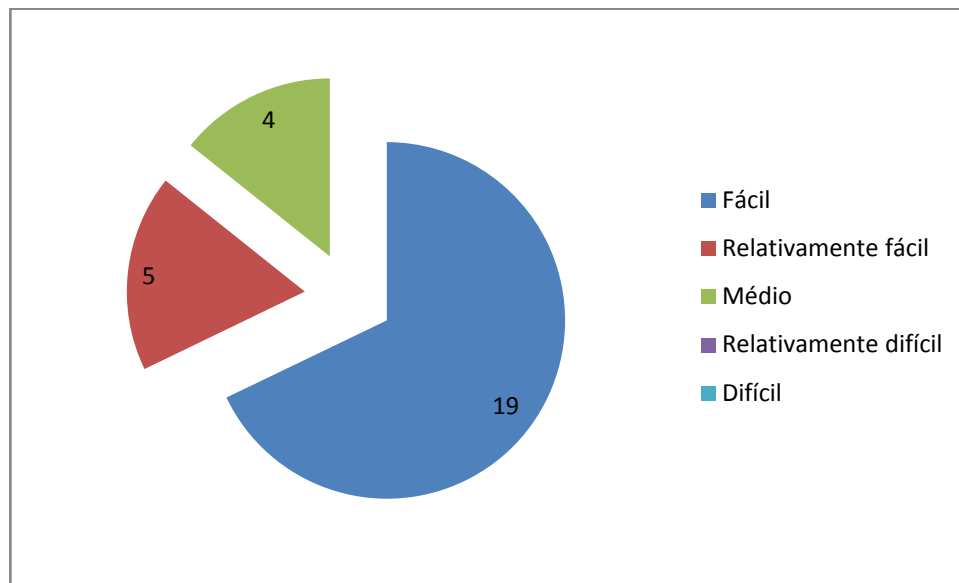


Figura 46 – Resultado da avaliação geral do teste 2

Da mesma forma que o teste 1, todos os voluntários responderam corretamente às perguntas de compreensão (tipo 3), sendo consideradas válidas suas respostas. Também observa-se que a maioria dos usuários considerou fácil a realização das atividades propostas – inclusive em proporção (67,9%) maior que naquele teste. Entretanto, aqui também houve uma parcela considerável que destoou desse resultado. Assim, no intuito de averiguar as causas dessas divergências e também para avaliar individualmente o sucesso de cada um dos requisitos em relação à intuitividade, a seguir serão analisadas as respostas para as tarefas, novamente agrupadas por requisito da aplicação.

#### 6.3.1 Avaliação do requisito ‘a’ – Carregamento dinâmico do objeto

Na avaliação sobre o grupo II de atividades do segundo roteiro, que analisa o requisito ‘a’ da aplicação, observa-se que a totalidade dos usuários marcou o carregamento do objeto como “Fácil”. No caso do roteiro 2 este carregamento foi alterado para ser acionado por um botão que lê e carrega o objeto seguinte disponível para o protótipo. Por limitações da

versão gratuita da Unity, os objetos envolvidos foram disponibilizados localmente, disponibilizando-se uma implementação diferente da adotada na versão para PC.

### 6.3.2 Avaliação do requisito ‘b’ – Iluminação do objeto

O requisito ‘b’ do protótipo foi avaliado diretamente por uma pergunta ao final do teste 2. O gráfico do resultado das respostas é apresentado em seguida.

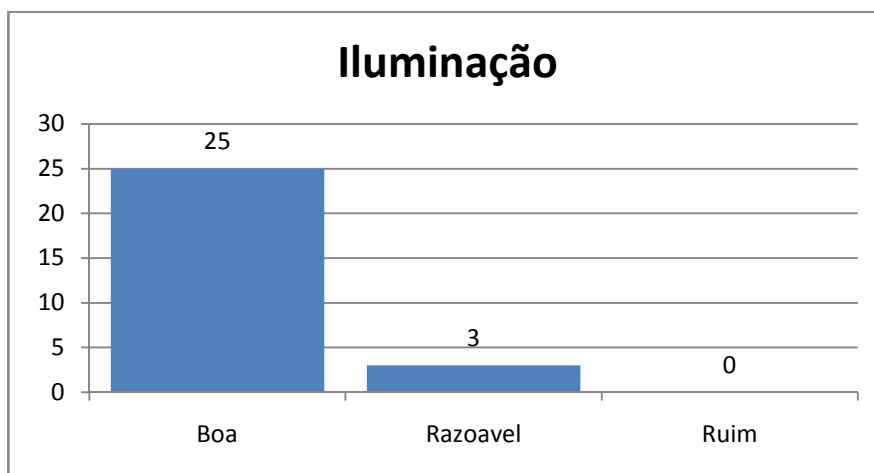


Figura 47 – Avaliação da iluminação na versão para dispositivos móveis.

89,3% das avaliações foram positivas. Para as divergentes, os voluntários justificaram não ser possível identificar os detalhes dos objetos (como os contornos dos dentes do crânio utilizado no teste).

Apesar de ser uma queixa da minoria dos participantes, ela evidencia um problema a ser resolvido sobre a iluminação do objeto utilizado no roteiro 2 e para a plataforma móvel. Entretanto, é possível que a limitação apontada esteja relacionada à forma como foi concebido o objeto em questão, o que isentaria o protótipo de responsabilidade.

Testes posteriores devem ser feitos para dirimir essa dúvida. Por ora, pode-se apenas considerar que o requisito ‘b’ foi apenas parcialmente atendido na versão para dispositivos móveis do protótipo. Todavia, trata-se de um aspecto de fácil ajuste, que não invalida a contribuição do trabalho.

### 6.3.3 Avaliação do requisito ‘c’ – Posicionamento do objeto em frente à câmera

Este requisito não foi objeto de avaliação, pelos motivos já apresentados.

#### 6.3.4 Avaliação do requisito ‘d’ – Rotação do objeto

O grupo IV do segundo roteiro engloba as atividades que utilizam as funcionalidades do requisito ‘d’. O resumo de sua avaliação é apresentado a seguir.

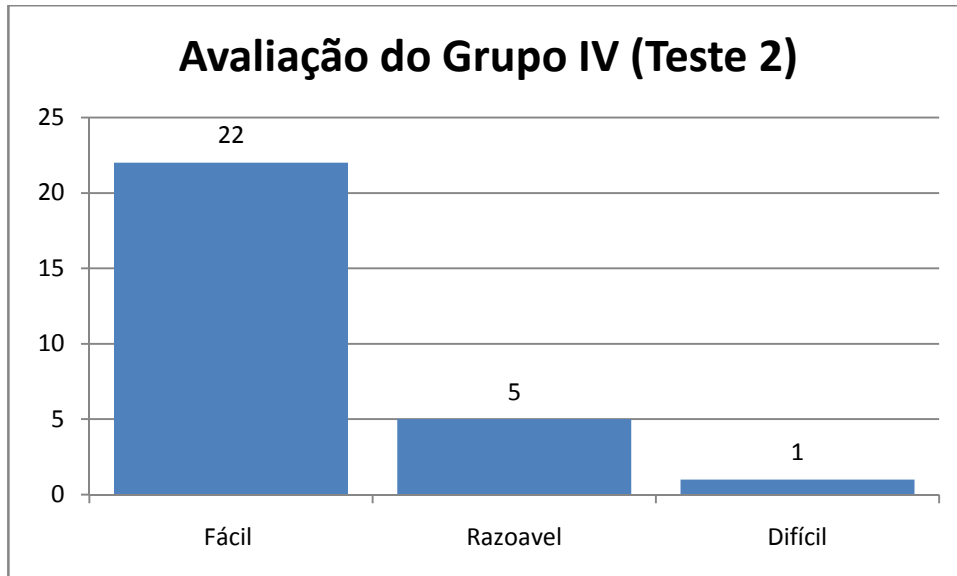


Figura 48 – Avaliação sobre a rotação dos objetos no teste 2.

78,6% dos voluntários avaliaram como “Fácil” as atividades do grupo. Como justificativa para as seis respostas divergentes, há queixas diretas à forma de rotação, como dificuldades de controlar a direção do toque e comportamento do objeto diferente do esperado.

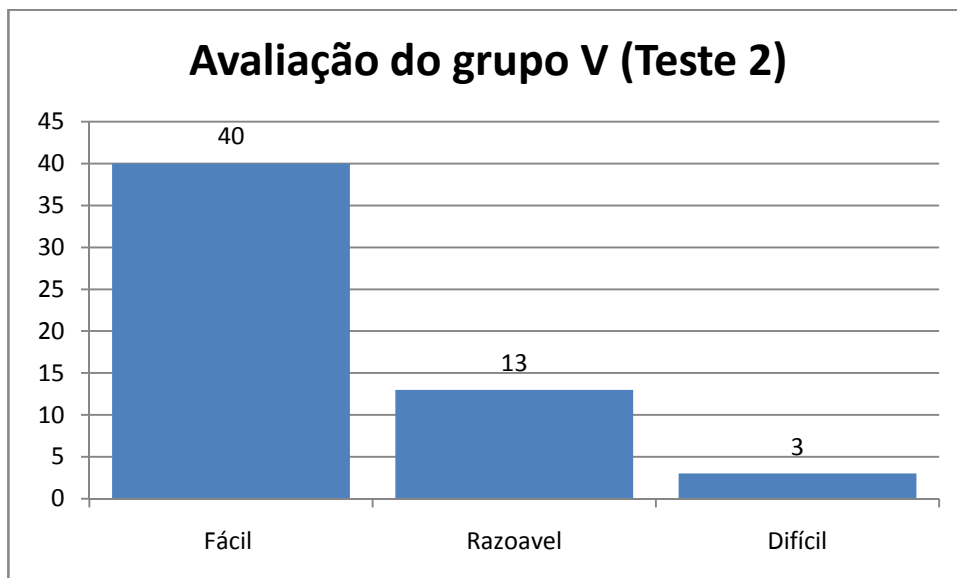


Figura 49 – Avaliação da atividade de contagem (instrução 24) que testa, entre outras coisas, a rotação do objeto no teste 2.

Ampliando-se um pouco mais a análise apresentada, pode-se observar, na figura 49, o gráfico das respostas sobre a atividade (24), pertencente ao grupo V de atividades, que requer a aplicação de rotação para sua realização. Das observações sobre essa atividade, elaboradas por aqueles que a consideraram “Razoável” ou “Difícil”, apenas 3 atribuíram a dificuldade a problemas de rotação, sendo duas queixas sobre a sensibilidade da rotação e apenas 1 mencionou “eixo invertido” – o que entende-se como dificuldade na compreensão de como a rotação se processa.

Assim, considerando-se que a inversão do eixo já tinha sido comentada no roteiro 1 do presente capítulo, tem-se que a dificuldade atribuída a rotação neste teste tem a mesma origem – a metáfora de rotação utilizada não é tão clara quanto se presumia.

Assim, da mesma forma como o requisito ‘d’ foi avaliado no roteiro anterior, pode-se dizer que a restrições quanto sua intuitividade também na versão para dispositivos móveis do protótipo, devendo-se buscar metáforas alternativas para sua representação. Da mesma forma, as questões de sensibilidade devem ser objeto de trabalhos futuros.

#### 6.3.5 Avaliação do requisito ‘e’ – Movimentação do objeto

As atividades (16) a (17), que fazem parte do grupo I de atividades, testaram a movimentação do objeto nos eixos *X* e *Y* (requisito ‘e’) no segundo roteiro. O resumo das avaliações desse quesito é apresentado no gráfico a seguir.

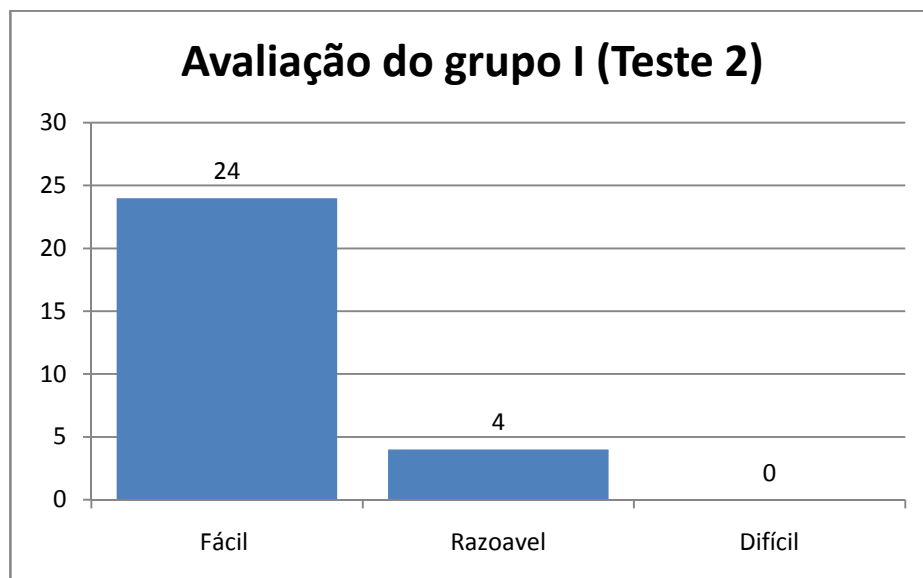


Figura 50 – Avaliação da movimentação do objeto no teste 2.

85,7% dos voluntários consideraram a movimentação fácil. Dos que tiveram uma impressão diferente, 2 atribuíram a dificuldades de entender o propósito da tarefa e outros 2 referiram problemas de interface – como botões pequenos e falta de gestos de movimentação – que serão considerados quando da análise do requisito ‘g’.

Com isso, observa-se que, no tocante a intuitividade da operação, não houve reclamação dos usuários na realização das tarefas. Desta forma, mesmo havendo considerações que serão observadas oportunamente, pode-se aceitar o requisito como realizado intuitivamente na versão para dispositivos móveis do protótipo (assim como o foi na versão PC).

### 6.3.6 Avaliação do requisito ‘f’ - Aproximação e afastamento do objeto

As atividades (19) e (20), que fazem parte do grupo III de atividades do roteiro 2, testaram o afastamento e a aproximação do objeto (requisito ‘f’). O gráfico com o resumo das avaliações segue.

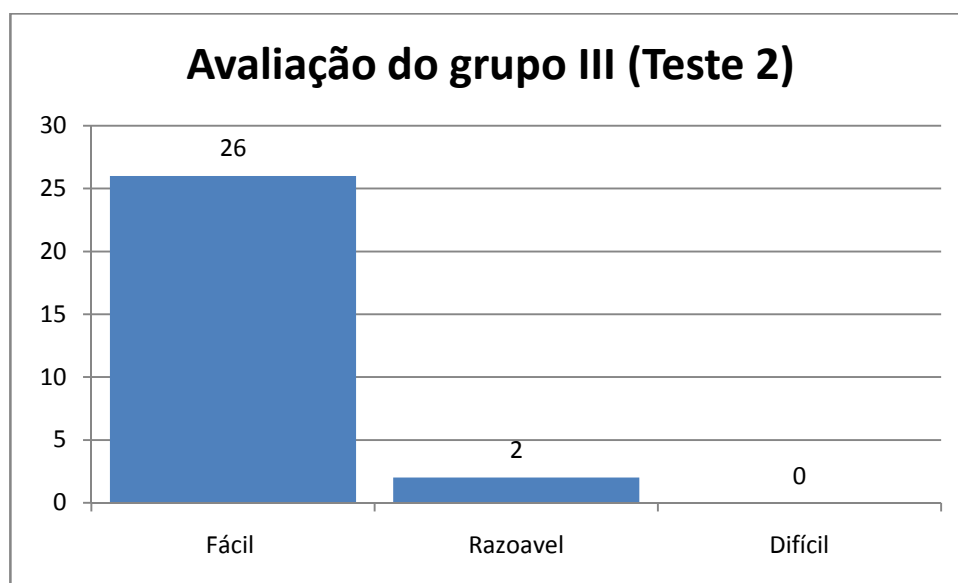


Figura 51 – Avaliação do afastamento e da aproximação do objeto no teste 2.

92,9% dos voluntários consideraram fáceis as tarefas de afastamento e aproximação do objeto. Das duas avaliações diferentes, uma foi justificada por o protótipo não utilizar algum tipo de gesto para controlar essa movimentação – o que faz parte do escopo da avaliação do requisito ‘g’.

Outro identificou um *bug* do protótipo: quando se afasta o objeto ao limite, ele começa a aumentar. Observa-se aqui que há um problema técnico na versão para dispositivos



móveis. Esse defeito apresenta-se apenas na versão da ferramenta utilizada para o roteiro 2 e haverá necessidade de verificar e corrigir o problema. Entretanto, não se tratando de uma “funcionalidade” intencional, não considera-se a ocorrência dessa falha como um comprometimento da intuitividade do requisito.

Então, sobre o julgamento de se o requisito ‘f’ foi atendido de forma intuitiva, pode-se dizer que sim. Mesmo tendo sido solicitado o uso de gestos particulares na plataforma (o que será considerado em trabalhos futuros), todos os usuários conseguiram realizar a tarefa sem problema algum.

#### **6.3.7 Avaliação do requisito ‘g’ - Utilizar os gestos de interface, elementos gráficos intuitivos e opções de hardware**

Pode se observar pelo exposto até o momento que o requisito ‘g’ não foi satisfatoriamente atendido na versão para dispositivos móveis do protótipo. Os usuários da plataforma, de uma forma geral, queixaram-se da falta dos gestos característicos da ferramenta e do tamanho dos botões utilizados.

Por outro lado, do ponto de vista da intuitividade, mesmo sem contar com os recursos característicos que a plataforma oferece, os botões utilizados para sugerir a idéia das movimentações possíveis foram amplamente utilizados por todos. Isso deixa a questão da intuitividade como atendida de uma forma geral.

Avaliando-se a versão como um todo, no aspecto de ter-se uma interface amigável, pode-se dizer que há trabalhos a serem implementados futuramente na ferramenta para disponibilizá-la para a comunidade de usuários dessa plataforma.

Estas considerações serão objeto de futuros trabalhos a serem abordados mais a frente, no capítulo que tratará desse assunto.

## CONCLUSÃO

O presente trabalho apresentou os esforços para criação de um protótipo para uma ferramenta chamada “3Dboard” que possa ser utilizada em PCs, *smartphones* e *tablets* para manipulação de objetos 3D. Propõe-se como diferencial da ferramenta a objetividade e simplicidade, sendo intuitiva o suficiente a ponto de não serem necessários menus ou tutoriais para aprender a utilizá-lo.

Após levantamento de referencial teórico e estudo do estado da arte desse tipo de aplicação, passou-se à fase de projeto, estabelecendo-se os objetivos da aplicação e 7 requisitos a serem atendidos, que envolveram movimentação e rotação em 3 eixos, aspectos de iluminação e obtenção de objetos para observação. Dada a proposta inicial, os requisitos também contemplaram a intuitividade da aplicação, estabelecendo o uso de padrões de operação consagrados, bem como o acréscimo meios intuitivos de interação.

Para a fase de implementação, escolheu-se a ferramenta Unity para implementação do protótipo, devido às facilidades oferecidas para agilizar o desenvolvimento e para gerar versões compatíveis com diversas plataformas.

Encerrado o desenvolvimento, obteve-se um protótipo em funcionamento em duas versões – uma para PC e outra para *smartphones* e *tablets*. Para testá-lo, foram criados roteiros que procurassem levar o usuário a utilizar as funcionalidades definidas em cada requisito do projeto. Também foi elaborado um questionário para avaliar os requisitos implementados do ponto de vista da sua intuitividade. Um formulário de avaliação foi construído utilizando os roteiros de teste e questionário. O formulário inicial foi submetido a um teste piloto, a partir do qual identificaram-se alguns problemas, que foram corrigidos na versão final - e essa, então, aplicada em uma amostra de voluntários para avaliar o protótipo. A amostra de voluntários foi formada por 28 pessoas.

As respostas para cada um dos sete requisitos foram analisadas, assim como, os comentários que os usuários colocaram nas suas folhas de perguntas. Apesar de algumas ressalvas, de um modo geral, os dados permitem concluir que o protótipo atingiu seus objetivos, oferecendo contribuições importantes. As ressalvas mencionadas constituem limitações do trabalho e podem dar origem a trabalhos futuros.

Por contrapartida, a grande contribuição que os usuários deixaram para a ferramenta são as sugestões que nortearão as novas implementações. Assim, as contribuições, limitações e possíveis trabalhos futuros são enumerados a seguir:

- Estudar e otimizar a rotação dos objetos de forma que não cause inversão de comandos, como os relatados nos roteiros apresentados no capítulo anterior.
- Maior aproveitamento dos gestos consagrados no uso de *smartphones* e *tablets* para facilitar as tarefas dos usuários com grande familiaridade nessa plataforma.
- Estudo e melhoria de tamanhos de botões para a plataforma *smartphone*, já que viu-se que há usuários que contam com telas de tamanhos diversos, nas quais os botões ficam realmente pequenos para serem pressionados com comodidade.
- Ajustar a sensibilidade de movimentação para a plataforma móvel, já que muitos usuários se queixaram de certas dificuldades de controlar o objeto.
- O fato de alguns usuários terem solicitado uma lista de objetos que poderiam ser carregados, sugere que pode ser implementado o uso de banco de dados para a disponibilização de objetos para estudo.

Acredita-se ter contribuído de alguma forma com a área de ensino ou da medicina, adicionando às opções existentes um aplicativo de fácil instalação na plataforma móvel e de acesso imediato na web, cujo uso, pela sua simplicidade e intuitividade, pode atender às necessidades de uma ampla faixa de usuários.

Além disso, por ser simples e, ainda, pensando na difícil arte de ensinar, contribui-se com um “quadro negro 3D”, ou “3Dboard”, que poderá ser utilizado amplamente para transmitir conhecimento anatômico para alunos de medicina ou de qualquer área interessada em observar objetos em 3 dimensões.

Uma das limitações do trabalho diz respeito à avaliação. Tendo em vista que os voluntários disponíveis declararam ter boa intimidade com a internet, os resultados podem não refletirem perfeitamente a realidade da ferramenta no sentido da sua intuitividade, já que, entre os possíveis usuários da aplicação, poderá haver diversos usuários com menos “experiência”. Assim, são necessários testes com amostras mais representativas para atestar definitivamente esse quesito.

Outra limitação que merece atenção é a de que o protótipo importa apenas objetos no formato “.obj” gerados pelo Blender. Como foi apresentado no capítulo 5, do presente

trabalho, há muitas variações nos formatos dos arquivos que contém os dados para os objetos tridimensionais e ampliar a capacidade de importação de objetos para a ferramenta é sem dúvida nenhuma uma futura premissa.

Pode-se citar, também, como limitação, o fato de não haver utilizado em favor da ferramenta todos os gestos característicos da plataforma móvel que os usuários estão tão acostumados a utilizar e que sua falta mereceu vários comentários por parte deles nos testes realizados.

Como fruto dos testes realizados, é possível enumerar alguns trabalhos futuros que podem servir para incrementar a ferramenta, corrigir as falhas identificadas e a torná-la mais útil para seus usuários no cumprimento do seu objetivo. Entre eles, destacam-se:

- 1) Trabalhar melhor a questão da rotação, no sentido de torná-la mais fácil de ser compreendida. Os testes realizados demonstraram que a hipótese levantada sobre a utilidade da metáfora de rotação implementada não é completamente verdadeira (essa forma de rotação não é tão intuitiva quanto se pensou inicialmente). Assim, deve-se estudar novas formas de rotação, como a de fazer com que qualquer movimento de rotação ocorra como se o usuário estivesse tocando a figura no central do foco da câmera. Além disso, também pode-se aplicar uma velocidade com que o usuário consiga se sentir mais confortável;
- 2) Trabalhar na importação de mais tipos de arquivos 3D, aperfeiçoando e expandindo os *scripts* utilizados no protótipo, adequando-os, a leitura de mais tipos de objetos provenientes de diversas outras ferramentas de modelagem e renderização;
- 3) Aperfeiçoar a interface da versão para *smartphone* e *tablet*, utilizando-se de todos os recursos que a plataforma oferece para movimentar e rotacionar o objeto através de gestos (como arrastar objetos utilizando mais de um dedo ao mesmo tempo), além dos botões aprovados nos testes. Além disso, pode-se trabalhar no *design* dos botões da versão para *smartphones*, modificando seu tamanho para tornar o seu pressionamento mais fácil por parte dos usuários;
- 4) Acrescentar opções de acessibilidade em ambas versões, utilizando comandos de voz para acionar os recursos para usuários com problemas;

- 5) Uso de banco de dados para o acesso a uma biblioteca de objetos para estudo, cuja manutenção seja realizada pela internet e alimentada pela rede de ensino que for utilizar a ferramenta.

Ao final pode se definir como concluído com êxito este trabalho, desenvolvendo o protótipo, avaliando-o e deixando disponíveis os requisitos para futuras implementações.

## REFERÊNCIAS BIBLIOGRÁFICAS

ADORNO, B. V. **Planejamento Probabilístico de Rotas no espaço de configuração e sua aplicação em Robótica Móvel**. Brasília - DF: Universidade de Brasília, 2007.

ARGER, M. E. B. F. Como funciona a Doação de Corpos para Instituições de Ensino. **bluelogs.net**, Belo Horizonte, 24 Agosto 2009. Disponível em: <<http://bluelogs.net/drexplica/artigos/como-funciona-a-doacao-de-corpos-para-instituicoes-de-ensino/>>. Acesso em: 09 Março 2012.

AUTODESK. Autodesk Maya. **Autodesk**, 2012. Disponível em: <<http://usa.autodesk.com/maya/>>. Acesso em: 20 Mar 2012.

AUTODESK. 3DS MAX. **Autodesk**, out. 2013. Disponível em: <<http://www.autodesk.com.br/products/autodesk-3ds-max/overview>>. Acesso em: 25 out. 2013.

AZEVEDO E., CONCI A. **Computação Gráfica**. 7. ed. Rio de Janeiro: Elsevier Editora Ltda., 2003.

BLENDER. Your Own 3D Software. **blender**, out. 2013. Disponível em: <<http://www.blender.org/>>. Acesso em: 25 out. 2013.

BRIDWATER, A. Open Source Insider. **ComputerWeekly**, 2012. Disponível em: <<http://www.computerweekly.com/blogs/open-source-insider/2012/01/google-open-sources-zygote-3d-human-body-viewer.html>>. Acesso em: 03 Jun 2012.

BUARQUE, S. C. Os Educadores do Futuro. **Isto é Comportamento**, p. Edição 1964, 2007.

BUSS, S. R. **3-D Computer Graphics - A Mathematical Introduction with OpenGL**. 2003. ed. New York: Cambridge University Press, 2003.

CARRANZA, FLORIAN. **OpenGL y VTK**. Escuela Académico Profesional de Informática, Universidad Nacional de Trujillo. Trujillo Perú, p. 11. 2006.

CYBER, A. interactelsevier.com. **www.interactelsevier.com**, 2010. Disponível em: <<http://www.interactelsevier.com/netter>>. Acesso em: 08 Março 2012.

DORA, M. S. ANATOMIA ARTE IMAGEM. **CADERNOS DA ABEM - ASSOCIAÇÃO BRASILEIRA DE EDUCAÇÃO MÉDICA**, p. 34, 35, 2003.

DROZDZ, B. LOADING 3D models at runtime in Unity3D. **EVERYDAY3D**, 01 maio 2010. Disponível em: <<http://www.everyday3d.com/blog/index.php/2010/05/24/loading-3d-models-runtime-unity3d/>>. Acesso em: 25 set. 2013.

EXAME, R. Fazer as coisas mais fáceis é o valor da simplicidade, diz especialista dos EUA. **exame.com**, 25 out. 2013. Disponível em: <<http://exame.abril.com.br/negocios/noticias/fazer-as-coisas-mais-faceis-e-o-valor-da-simplicidade-diz-especialista-dos-eua-m0076823>>. Acesso em: 25 out. 2013.

FEITO ET AL. **Proyectos de Investigación (2004-2005)**. Primera edición, diciembre de 2006. ed. TORREDONJIMENO (Jaén) - Espanha: © UNIVERSIDAD DE JAÉN, 2006.

FILEFORMAT.INFO. Wavefront OBJ File Format Summary. **FileFormat.Info**, 2013. Disponível em: <<http://www.fileformat.info/format/wavefrontobj/egff.htm>>. Acesso em: 07 nov. 2013.

GONZALES, WOODS. **Processamento de Imagens Digitais**. 2007. ed. [S.l.]: Editora Blucher, 2007.

GONZÁLEZ, C. S. C. **Web-based Graphics Library**. [S.l.]: [s.n.], 2012.

GOOGLE. ZygoteBody. **ZygoteBody**, 2012. Disponível em: <<http://www.zygotebody.com/>>. Acesso em: 03 Jun 2012.

HEALTHLINE. Healthline BodyMaps. **Visualize Better Health**, 2005. Disponível em: <<http://www.healthline.com/human-body-maps>>. Acesso em: 03 Jun 2012.

JUDENSNAIDE, I. J. Lição de Anatomia do Prof. Tulp - O Olhar de Rembrant. **arScientia**, 2005. Disponível em: <[http://www.arscientia.com.br/materia/ver\\_materia.php?id\\_materia=57](http://www.arscientia.com.br/materia/ver_materia.php?id_materia=57)>. Acesso em: 20 Mar. 2012.

KHRONOS. Connecting Software to Silicon. **KHRONOS Group**, 1012. Disponível em: <<http://www.khronos.org/webgl/>>. Acesso em: 31 Mai 1012.

KITWARE. **ActiViz.NET User's Guide - Version 5.2**. 2008. ed. U.S.A: Kitware, Inc., 2008.

KITWARE. [vtk.org](http://www.vtk.org/). **vtk Visualization Toolkit**, 2012. Disponível em: <<http://www.vtk.org/>>. Acesso em: 15 Mai 2012.

LAMARCHE. OpenGL ES 2.0 for iOS, Chapter 3 - Fundamentals of 3D Programming. **IPHONE DEVELOPMENT**, 2010. Disponível em: <<http://iphonedevlopment.blogspot.com.br/2010/10/opengl-es-20-for-ios-chapter-3.html>>. Acesso em: 27 Mai. 2012.

MACHADO, L. D. S. **A Realidade Virtual no Modelamento e Simulação de Procedimentos Invasivos em Oncologia Pediátrica: Um Estudo de Caso no Transplante de Medula Óssea** (Tese de Doutorado) Orientador: Prof. Dr. Marcelo Knörich Zuffo, São Paulo, 2003.

QUEIROZ, C. D. A. F. **O Uso de Cadáveres Humanos como Instrumento na Construção de Conhecimento a partir de uma Visão Bioética** (Dissertação de Mestrado) Orientador Prof. Dr. José Nicolau Heck, Goiânia, Fevereiro 2005.

ROOSENDAAL ET AL. **BLENDER 2.3 GUIA**. [S.l.]: [s.n.], 2004.

SCHROEDER, MARTIN & LORENSEN. **The Visualization Toolkit - An Object-Oriented Approach to 3D Graphics**. Third Edition. ed. New York - USA: Pearson Education, Inc., 2002.

SYSTEMS, B. biodigitalhuman.com. **www.biodigitalhuman.com**, 2011. Disponível em: <<http://www.biodigitalhuman.com/>>. Acesso em: 09 Março 2012.

TECHNOLOGIES, U. Unity. **unity3d.com**, 23 out. 2013. Disponível em: <<http://portuguese.unity3d.com/>>. Acesso em: 23 out. 2013.

UNIVERSITÁRIA. Sete Curiosidades na "Lição de Anatomia do Dr. Tulp" de Rembrandt van Rijn. **Medicina & Arte**, Mai 2007. Disponível em: <<http://medicinaearte.blogspot.com.br/2010/06/sete-curiosidades-na-licao-de-anatomia.html>>. Acesso em: 20 Mai 2012.

VICEDO, LINARES. Escaneado de objetos tridimensionales en el ITI. **Revista del Instituto Tecnológico de informática**, Valencia - Espanha, p. 04 - 07, Jan 2010.

WRIGHT, LIPCHAK, HAEMEL. **OpenGL Superbible**. Fourth Edition. ed. Boston, MA: Pearson Education Inc., 2007.



## Apêndice I – Formulário para testes

# Roteiro para teste do



Nome	
Idade	
Afinidade com a internet	Pouca <input type="checkbox"/> razoável <input type="checkbox"/> muita <input type="checkbox"/>
Afinidade com dispositivos	Pouca <input type="checkbox"/> razoável <input type="checkbox"/> muita <input type="checkbox"/>

Conto com você para realizar dois testes que servirão para avaliar meu trabalho de conclusão.

Para o primeiro teste você disporá de 20 minutos para sua realização. Para o segundo disporá de mais 20 minutos. Ao final de cada etapa de teste, haverá questões avaliando aspectos do trabalho. Preencha as informações acima e responda da forma mais direta e sincera possível. Agradeço desde já sua colaboração. Por favor, utilize caneta.

Este é o primeiro teste:

Siga às instruções abaixo e responda as perguntas ao final de cada etapa.

Você terá 20 minutos para realizar este teste.

Abra o endereço: <http://www.lusardo.com.br/feevale/Web3dboard.html>

Anote a hora: \_\_\_\_\_

1. Carregue o objeto chamado *monkey*.

O que você achou da tarefa ( instrução 1 )?

1 – Fácil	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Difícil	<input type="checkbox"/>

Se você informou Difícil ou Razoável, explique o motivo:


2. Movimente o objeto para que ele fique no canto superior direito da sua tela.
3. Mova o objeto para que ele fique encostado no botão carregar.
4. Faça com que o objeto fique encostado no botão com a seta verde apontada para a direita.
5. Coloque o objeto novamente no centro da tela.

O que você achou da tarefa ( Instruções de 2 a 5 )?

1 – Fácil	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Difícil	<input type="checkbox"/>

Se você informou Difícil ou Razoável, explique o motivo:

--

O objeto que você está manipulando é um macaco.

6. Coloque-o de perfil olhando para a direita.
7. Coloque-o de perfil olhando para a esquerda.
8. Coloque o macaco olhando para você.

Qual é o formato do nariz dele: \_\_\_\_\_

9. Faça o macaco ficar olhando para baixo.
10. Faça o macaco ficar olhando para cima.
11. Coloque o macaco de cabeça para baixo, olhando para você.

Que você usou para cumprir com a última instrução?


O que você achou da tarefa ( instruções de 6 a 11 )?

1 – Fácil	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Difícil	<input type="checkbox"/>

Se você informou Difícil ou Razoável, explique o motivo:


12. Aproxime o objeto o máximo que puder e deixe-o assim.

13. Carregue o objeto chamado *cora* e clique sobre ele.

O que você achou da tarefa ( instruções de 12 a 13 )?

1 – Fácil	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Difícil	<input type="checkbox"/>

Se você informou Difícil ou Razoável, explique o motivo:


**Atenção:**

*Sem clicar nos **botões da tela**:*

14. Afaste o objeto o máximo que puder.

O que você achou da tarefa ( instrução 14 )?

1 – Fácil	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Difícil	<input type="checkbox"/>

Se você informou Difícil ou Razoável, explique o motivo:


Para realizar a próxima tarefa, pode movimentar o objeto do jeito que quiser e tantas vezes quantas forem necessárias de maneira que facilite a contagem.

15. Conte quantos buracos existem no objeto? \_\_\_\_\_

O que você achou da tarefa ( instrução 15 )?

1 – Fácil	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Difícil	<input type="checkbox"/>

Se você informou Difícil ou Razoável, explique o motivo:


Acabou o teste 1, avise o avaliador e anote a hora: \_\_\_\_\_

O que você achou da iluminação?

1 – Boa	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Ruim	<input type="checkbox"/>

Se você informou Difícil ou Razoável, explique o motivo:


O que você achou do teste 1 como um todo (questões 1 a 15)?

1 – Fácil	<input type="checkbox"/>
2 – Razoavelmente fácil	<input type="checkbox"/>
3 – Médio	<input type="checkbox"/>
4 – Razoavelmente difícil	<input type="checkbox"/>
5 – Difícil	<input type="checkbox"/>

Se você não informou fácil, explique o motivo:


Você considera que a forma de interação (botões, teclas, uso do mouse) com o sistema na versão para PC/Web é intuitiva e fácil de assimilar?

1 – Concordo plenamente	<input type="checkbox"/>
2 – Concordo parcialmente	<input type="checkbox"/>
3 – Não concordo nem discordo	<input type="checkbox"/>
4 – Discordo parcialmente	<input type="checkbox"/>
5 – Discordo plenamente	<input type="checkbox"/>

Se você não informou Concordo plenamente, explique o motivo:

----------

# Roteiro para teste do



Esse teste será realizado no celular e você terá 20 minutos.

Este é o segundo teste:

Siga às instruções abaixo e responda as perguntas ao final de cada etapa.

Você tem 20 minutos para executar este teste.

Carregue o programa no seu celular a partir do endereço:

<http://www.lusardo.com.br/objetos/Board3DAP.apk>

Abra o programa 3Dboard;

Anote a hora: \_\_\_\_\_

16. Faça com que o objeto apresentado saia da visão na tela.

17. Agora, faça com que o objeto fique debaixo do logotipo.

O que você achou da tarefa ( Instruções 16 e 17 )?

1 – Fácil	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Difícil	<input type="checkbox"/>

Se você informou Difícil ou Razoável, explique o motivo:


18. Carregue o próximo objeto.

O que você achou da tarefa ( Instrução 18 )?

1 – Fácil	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Difícil	<input type="checkbox"/>

Se você informou Difícil ou Razoável, explique o motivo:


19. Afaste o objeto o máximo que puder.

20. Aproxime o objeto e centralize-o.

O que você achou da tarefa ( Instruções de 19 a 20 )?

1 – Fácil	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Difícil	<input type="checkbox"/>

Se você informou Difícil ou Razoável, explique o motivo:


21. Observe a parte de cima do crânio.

22. Coloque o objeto olhando para a direita.

23. Faça o objeto ***girar de forma que ele permaneça olhando para a direita.***

O que você achou da tarefa ( Instruções de 21 a 23 )?

1 – Fácil	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Difícil	<input type="checkbox"/>

Se você informou Difícil ou Razoável, explique o motivo:


Para realizar a próxima tarefa, pode movimentar o objeto do jeito que quiser e tantas vezes quantas forem necessárias de maneira que facilite a contagem.

24. Informe quantos dentes tem esse crânio? \_\_\_\_\_

O que você achou da tarefa ( Instrução 24 )?

1 – Fácil	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Difícil	<input type="checkbox"/>

Se você informou Difícil ou Razoável, explique o motivo:


Acabou o teste 2, avise o avaliador e anote a hora \_\_\_\_\_.

O que você achou da iluminação?

1 – Boa	<input type="checkbox"/>
2 – Razoável	<input type="checkbox"/>
3 – Ruim	<input type="checkbox"/>

Se você informou Ruim ou Razoável, explique o motivo:


O que você achou do teste 2 como um todo (questões 16 a 24)?

1 – Fácil	<input type="checkbox"/>
2 – Razoavelmente fácil	<input type="checkbox"/>
3 – Médio	<input type="checkbox"/>
4 – Razoavelmente difícil	<input type="checkbox"/>
5 – Difícil	<input type="checkbox"/>

Se não informou Fácil, explique o motivo:




Você considera que a forma de interação (botões, toques e gestos) com o sistema na versão para *smartphone* é intuitiva e fácil de assimilar?

1 – Concordo plenamente	<input type="checkbox"/>
2 – Concordo parcialmente	<input type="checkbox"/>
3 – Não concordo nem discordo	<input type="checkbox"/>
4 – Discordo parcialmente	<input type="checkbox"/>
5 – Discordo plenamente	<input type="checkbox"/>

Se você não informou Concordo plenamente, explique o motivo:


Por favor de uma nota de 1 a 10 para o programa no que se refere a usabilidade e interface amigável? \_\_\_\_\_

Obrigado pela sua participação.