MARINA DE LARA

GERAÇÃO DINÂMICA DE AMBIENTES IMERSIVOS EM CAVE COM ESTEREOSCOPIA: USANDO MECANISMOS DE JOGOS DIGITAIS

CURITIBA

2021

MARINA DE LARA

GERAÇÃO DINÂMICA DE AMBIENTES IMERSIVOS EM CAVE COM ESTEREOSCOPIA: USANDO MECANISMOS DE JOGOS DIGITAIS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Pontifícia Universidade Católica do Paraná - PUCPR
Programa de Pós-Graduação em Informática - PPGIA

Orientador: Prof. Dr. Edson Emílio Scalabrin

CURITIBA
2021

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Sônia Maria Magalhães da Silva – CRB 9/1191

L318g 2021

Lara, Marina de
Geração dinâmica de ambientes imersivos em CAVE com estereoscopia : usando mecanismos de jogos digitais / Marina de Lara ; orientador: Edson Emílio Scalabrin. – 2021
91 f. ; il. : 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2021
Bibliografia: f. 88-91

1. Realidade virtual. 2. Estereoscopia. 3. Sistemas de indicação visual tridimensional. 4. Informática. I. Scalabrin, Edson Emílio. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título.

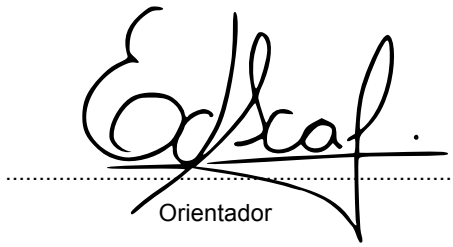
CDD. 20. ed. – 004

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
ESCOLA POLITECNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

AUTORIZAÇÃO PARA ENTREGA DA VERSÃO FINAL DA DISSERTAÇÃO DE MESTRADO

Eu, **Edson Emílio Scalabrin**, orientador da aluna de mestrado **MARINA DE LARA**, certifico que foram feitas todas as alterações solicitadas pela Banca Examinadora e indicadas na ata de defesa, e que a dissertação intitulada **GERAÇÃO DINÂMICA DE AMBIENTES IMERSIVOS EM CAVE COM ESTEREOSCOPIA: USANDO MECANISMOS DE JOGOS DIGITAIS,** na presente forma, cumpre todas as normas de formatação definidas pelo Programa, podendo, portanto, ser entregue na secretaria do Programa.

Curitiba (PR), 06 de Outubro de 2023



.....
Orientador

Em memória de Vicente.

Agradecimentos

Em primeiro lugar, quero agradacer a minha família. Minha avó Zenaide, meu tio Cleverson, meu avô Nelson e principalmente minha mãe Cristiane, por todos os sacrífícios que sempre fizeram por mim, por nunca deixarem de acreditar no meu sucesso e potencial e por sempre estarem ao meu lado nos momentos felizes e tristes. Vocês são tudo pra mim!

Também quero agradecer aos meus melhores amigos Allan e Mariana, por serem os irmãos que a vida me deu.

A todos os professores do PPGIA e do curso de Jogos Digitais, em especial, ao Prof. Dr. Edson Justino por me proporcionar tantas oportunidades e, ao meu orientador, Prof. Dr. Edson Scalabrin, por toda a credibilidade que sempre me deu e por ser calmaria em tempos de tempestade. Vocês são a minha inspiração!

Aos meus colegas de laboratório e de faculdade, por todo o companheirismo e por sempre estarem dispostos a ajudar. Sobretudo, aos colegas Flávio e Diogo, por me ajudarem a compreender o processo de aquisição e segmentação, ao Aramis por sempre compartilhar seus conhecimentos e ao Lucas, por sempre resolver todos os problemas no laboratório. Obrigada por tudo pessoal.

Por fim, agradeço a todas as mulheres da área de computação que servem de modelo para mim e para outras meninas e mulheres. É a coragem, inteligência e graciosidade de vocês que nos estimulam a seguir em frente.

"O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001"

"Homens fortes criam tempos fáceis
e tempos fáceis geram homens fracos,
mas homens fracos criam tempos difíceis
e tempos difíceis geram homens fortes."

Provérbio Oriental

Resumo

A virtualização de ambientes naturais é uma prática cada vez mais presente na sociedade contemporânea e ela tem encontrado um campo de aplicação ainda mais fértil em função, por exemplo, de restrições sociais ou simplesmente por questões práticas e econômicas. Neste contexto, esse trabalho apresenta um gerador dinâmico de ambientes imersivo e colaborativos em *CAVE*, com estereoscopia, com foco (a) na aquisição de imagens de pequenos objetos complexos em termos de detalhes; (b) no pós-processamento que mantenha a acuidade visual de tais objetos; (c) na projeção de imagens em *3D* em *CAVE* cilíndrica 360°. O ambiente possui capacidade de interação em tempo real com os elementos projetados. A base de imagens é composta por insetos reais, e sua aquisição é feita com o *scanner F2S2* e o processamento com o software *BipApp*. As estratégias de aquisição, pós-processamento e projeção de imagens em CAVE cilíndrica foram colocas em prática em *C++* e *SDL*. Os resultados do projeto encerram (a) um sistema de geração dinâmica de ambientes virtuais em *CAVE* cilíndrica 360°, e (b) um sistema de projeção de modelos volumétricos com estereoscopia em *CAVE* cilíndrica 360°; ambas as contribuições podem ser colocadas em prática com esforços humanos reduzidos.

Palavras-Chave: *CAVE*, Estereoscopia, Ambientes Imersivos.

Abstract

The virtualization of natural environments is an increasingly present practice in contemporary society and it has found an even more fertile field of application due, for example, to social restrictions or simply for practical and economic reasons. In this context, this work presents a dynamic generator of immersive collaborative environments in *CAVE*, with stereoscopy, with a focus on (a) the acquisition of images of small objects that are complex in terms of details; (b) in post-processing that maintains the visual acuity of such objects; (c) in the projection of *3D* images in a 360° cylindrical *CAVE*. The environment is capable of real-time interaction with the projected elements. The image base is composed of real insects, and its acquisition is done with the *F2S2* scanner and processing with the *BipApp* software. The acquisition, post-processing and image projection strategies in a cylindrical *CAVE* were put into practice in *C++* and *SDL*. The project results include (a) a system for dynamic generation of virtual environments in a 360° cylindrical *CAVE*, and (b) a system for the projection of volumetric models with stereoscopy in a 360° cylindrical *CAVE*; both contributions can be put into practice with reduced human effort.

Keywords: *CAVE*, Stereoscopy, Immersive Environment.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

3D	Três Dimensões
2D	Duas Dimensões
BipApp	Batch Image Processing App
CAVE	CAVE Automatic Virtual Environment
CPU	Central Processing Unit
F2S2	Full Frames Semi-Spherical Scanner
FPR	Film-Type Patterned Retarder
GPU	Graphics Processing Unit
HDCP	High-bandwidth Digital Content Protection
HDMI	High-Definition Multimedia Interface
HDR	High Dynamic Range
HMD	Head-Mounted Display
HSV	Hue Saturation Value
ID	Identity
IDE	Integrated Development Environment
ISO	International Organization for Standardization
JPG	Joint Photographics Experts Group
PPGIA	Programa de Pós-Graduação em Informática
PNG	Portable Network Graphics
PUCPR	Pontifícia Universidade Católica do Paraná
RA	Realidade Aumentada

RAM	Random Access Memory
RGB	Red Green Blue
RV	Realidade Virtual
SDL	Simple DirectMedia Layer
SSD	Solid-State Drive
SVM	Support Vector Machine
VRAM	Video Random Access Memory
VRAT	Virtual Reality Exposure Therapy

Sumário

1 Introdução

Ao longo dos últimos anos a indústria do entretenimento vem almejando entregar ao público uma experiência mais palpável e próxima da realidade, proporcionando novas formas de inserir o espectador em um determinado contexto. Hoje é comum ir ao cinema assistir um filme em 3D, apreciar um show ou teatro com efeitos visuais holográficos e experimentar jogos de forma imersiva. Apesar destes recursos serem usados para proporcionar diversão, eles também podem ser uma alternativa inovadora para explorar novas formas de realizar atividades cotidianas

Uma ferramenta que vem se tornando popular por proporcionar experiências imersivas, é a RV (realidade virtual). Essa técnica permite, com a utilização de headsets – ou óculos – de realidade virtual, que uma pessoa seja transportada para um plano virtual podendo interagir com ele. Segundo (YU, 2017), a RV está sendo popularizada na medida que proporciona experiências imersivas que podem ser observadas e sentidas. Por conta disso, tem sido muito utilizada em videogames, mas também está ganhando espaço em outras áreas. A escola de inglês *Beetools*[1] desenvolveu um método de aprendizagem que proporciona diversas situações cotidianas criadas a partir de ambientes virtuais, onde o aluno precisa comunicar com pessoas dentro desse universo em inglês. A área de psicologia tem realizado terapias de exposição à realidade virtual (*Virtual Reality Exposure Therapy* – VRET) para medir a ansiedade por meio de indicadores fisiológicos e/ou verbais. Desta forma, o profissional pode analisar o indivíduo inserido em contextos que lhe causem desconforto, sem que o sujeito seja exposto ao julgamento social e a situações perigosas (JOSEANI; BENDER; KOCHHANN, 2016).

A desvantagem de experiências imersivas, que fazem uso de óculos de realidade virtual, é que elas ocorrem de forma individual. Como esse dispositivo oculta todo o ambiente real ao redor do indivíduo, a imersão fica limitada apenas ao plano virtual. Isso anula a possibilidade de interação em grupo, tornando sua usabilidade não tão adequada em determinadas situações.

Uma forma alternativa de obter experiências imersivas de forma colaborativa são os ambientes *CAVE*. A sigla vem do acrônimo *CAVE Automatic Virtual*

[1] <https://www.beetools.com.br/> Acessado em: 03/12/2021

Environment (Ambiente Virtual Automático *CAVE*) e foi pesquisado e desenvolvido pelo Laboratório de Visualização Eletrônica (*Electronic Visualization Laboratory*) da Universidade de Illinois em Chicago, para ser utilizado como uma ferramenta de visualização científica. Estes ambientes produzem uma sensação de imersão por serem feitos em formato de um grande cubo – pelo menos uma pessoa deve conseguir entrar no cubo – cercando o usuário com telas de projeção em todas as suas faces (MANJREKAR et al., 2014). Para obter uma visualização 3D, é preciso usar óculos com tecnologia estéreo ativos. (CREAGH, 2003)

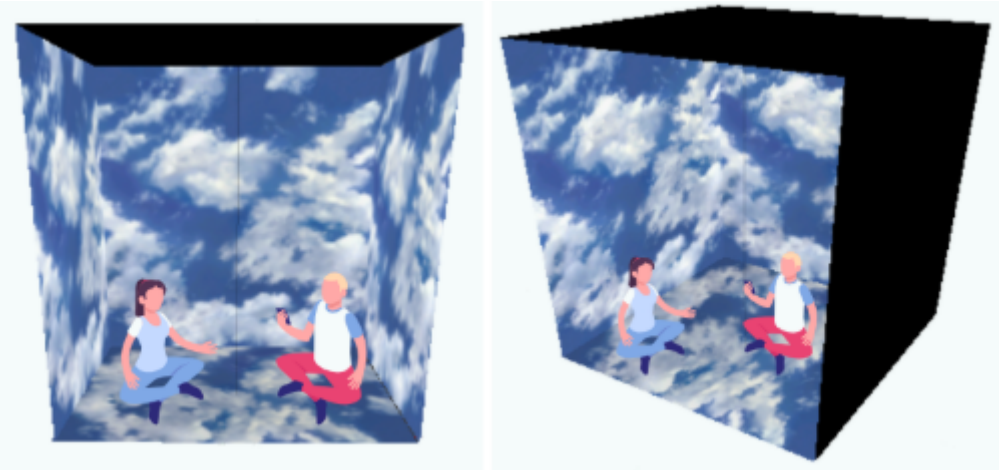


Figura 1 – Ilustração de um modelo de *CAVE* cúbica. Fonte: Própria

Existem *CAVEs* que possuem outros formatos. A companhia *Igloo Vision*[2] desenvolve há mais de dez anos ambientes imersivos 360° compartilhados, não só com design de cubo, mas também em formatos cilíndricos, de domos e personalizados. Há diversos tipos de aplicação, algumas delas são: simulações para treinamento e educação, vídeo conferências e aulas assistidas a distância, demonstração de produtos, entretenimento etc.

A realidade de um mundo 3D é transmitida para as pessoas por meio de estímulos dos sentidos fisiológicos de visão, olfato, paladar, audição e tato (ONIME; ABIONA, 2016). Em seu livro (ALMEIDA; JUSTINO, 2020) comentam que os ambientes imersivos podem utilizar diferentes tecnologias a fim de proporcionar uma experiência real, dentro de um espaço restrito e controlado. Os ambientes *CAVE* viabilizam, além de projeções, o uso de equipamentos capazes de emitir odores e sons. Também é possível implementar formas de interação com os elementos

[2] <https://www.igloovision.com/> Acessado em: 12/02/2020

projetados utilizando *joystick*, *Kinect*, telas *touch* ou qualquer outro dispositivo de *input*.

A *CAVE* pode ser aplicada em diversos contextos, um bom exemplo é a sua utilização como ferramenta de ensino. Com as diferentes técnicas que podem ser aplicadas em uma *CAVE*, é possível criar uma ambientação baseada em múltiplos estilos, que estimule simultaneamente diferentes formas de aprendizagem para que estudantes com dificuldades heterogêneas consigam atingir o mesmo nível de compreensão de um determinado conteúdo. Essa prática, não só traz maior significância e fixação no aprendizado, como é considerada motivante, encorajadora e, principalmente, empolgante. (PANTELIDIS, 2010)

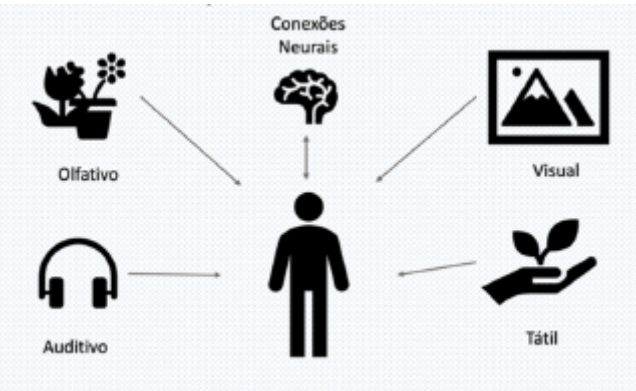


Figura 2 – Múltiplos estímulos. Fonte: (ALMEIDA; JUSTINO, 2020)

Apesar da popularização da realidade virtual e dos pontos positivos proporcionados pela imersão, ainda não é comum a existência desses recursos em locais que fazem parte da rotina das pessoas (e.g. escolas, universidade, empresas). As *CAVEs*, em especial, ainda são muito pouco conhecidas pelo público geral, e sua construção envolve diferentes processos de alta complexidade. Tornar essa ferramenta mais acessível fomenta sua utilização, e consequentemente proporciona uma alternativa inovadora na prática de atividades cotidianas.

Devido às condições de quarentena e restrições sociais impostas pela pandemia do *Covid-19*, a *PUCPR* teve suas atividades presenciais suspendidas e todos os recursos utilizados neste projeto ficaram inacessíveis durante este período. Por conta disso algumas alterações foram aplicadas para que o projeto pudesse continuar sendo desenvolvido em ambiente domiciliar.

1.1 Objetivo Geral

O projeto tem como objetivo geral a criação de um gerador de *CAVE* cilíndrica dinâmica com múltiplas projeções, que resulte em uma maquete digital interativa de um ambiente imersivo que pode ser usado como base de criação em um modelo físico de ambiente imersivo, independendo do foco para o qual a imersão será utilizada. Para a consecução deste objetivo foram delineados os seguintes objetivos específicos:

- Realização de estudo de diferentes ambientes imersivos e aplicações de projeção em *CAVE* cilíndrica.
- Realização de estudo de métodos de aplicação de projeções estereoscópicas e modelos volumétricos.
- Aquisição e pós-processamento de imagens de modelos reais em ultra definição para projeções volumétricas.
- Implementação de estratégias de controle (i) de projeção volumétrica estereoscópica via comandos de entrada do usuário (e.g., rotação de um objeto tridimensional a partir de imagens bidimensionais, transformações geométricas do modelo (translação e escala), e sincronização do sistema de controle com a projeção), e (ii) de projeção volumétrica estereoscópica visualizadas por meio de lentes polarizadas.
- Criação de um modelo digital de *CAVE* cilíndrica de forma dinâmica a partir de parâmetros variáveis de raio e a quantidade de projetores.

1.2 Desafios

Se tratando da aquisição de imagens a partir de modelos reais de insetos, algumas particularidades devem ser levadas em consideração. A maioria dos insetos possuem muitos detalhes, e se tratando de uma ferramenta que tem o aprendizado como foco, é de suma importância que todos eles estejam presentes nas imagens obtidas. Quanto menor for o inseto, mais difícil se torna obter nitidez nestes detalhes. Além disso, os insetos possuem assimetrias proporcionais às suas características, sua aquisição é diferente quando comparados à um objeto simétrico e/ou com

poucos detalhes (e.g. pelos, antenas, espinhos, camadas etc.) Deve-se destacar que, de um lado, nós contamos com um equipamento que executa o processo de aquisição de imagens de forma automática – explicação detalhada do equipamento e do processo no capítulo do método –, do outro lado, a configuração do foco é feita de forma manual, logo é demorado e em determinadas posições algumas imagens acabam ficando desfocadas, o que pode resultar em uma visualização não tão clara de alguns detalhes importantes.

Em alguns testes de segmentação realizados o nível de detalhes também se torna um desafio. Para obter um resultado mais refinado, aplica-se a técnica de suavização de bordas que remove os resquícios do fundo indesejado. Porém, desta forma, alguns detalhes como por exemplo antenas, podem ser removidos parcialmente ou completamente. Como cada parte do inseto é relevante para o estudo, pode-se optar por não aplicar a suavização de bordas, o que permite manter os detalhes, em contrapartida o modelo final pode manter o contorno da cor do fundo que fora removido.

Para obter uma movimentação fluida que cause a sensação de tridimensionalidade a partir de imagens bidimensionais, é preciso que todas as imagens do mesmo objeto sejam armazenadas em memória. Cada modelo de aquisição resulta em mais de 3000 imagens, todas em ultra alta definição *4K*. Deve-se notar que, tomando como base um computador com a seguinte configuração: 2 *SSDs*, um com 1 *terabyte* e outro com 256 *gigabytes* de memória computacional para armazenamento e 128 *gigabytes* de processamento, demora-se alguns minutos para que esse processo seja realizado. Portanto requer-se uma estratégia/algoritmo que realize o carregamento das imagens de forma assíncrona para que os diferentes modelos possam ser carregados e armazenados em um curto espaço de tempo.

Conforme citado anteriormente, devido às restrições de acesso presencial aos locais durante a pandemia do *Covid-19*, algumas adaptações foram aplicadas com relação à *CAVE*, responsável por entregar a principal parte da experiência imersiva. Ao invés do desenvolvimento do modelo físico da *CAVE*, a aplicação passou a ser simulada em um ambiente virtual utilizando a *engine Unity*. A ferramenta oferece vários componentes que podem ser utilizados de diferentes formas e para diferentes propósitos. Apesar disto, não existe um componente pronto que possa ser usado para representar a *CAVE*, é necessário criar toda a estrutura que será capaz de renderizar imagens em seu interior. Para isso, é preciso gerar uma malha

de triângulos, que tenha suas normais direcionadas para o interior do cilindro, desta forma as imagens serão renderizadas na parte interna do cilindro.

Como a proposta é obter uma *CAVE* dinâmica, ou seja, uma *CAVE* que possa assumir diferentes tamanhos em raio, resolução e graus de circunferência, a malha de triângulos deve ser calculada com base nestes parâmetros, acarretando o desenvolvimento de uma solução que comporte quaisquer valores que sejam fornecidos como entrada. Além disso, para obter a face de renderização voltada para dentro, os índices correspondentes aos vértices dos triângulos devem ser definidos no sentido horário, o que implica diretamente na ordem em que os vértices foram instanciados. Em um modelo de *CAVE* com 360º, os vértices finais da malha devem possuir os mesmos índices dos vértices iniciais, desta forma a imagem ou vídeo 360 que forem renderizados não apresentarão fissuras ou incoerências entre as duas extremidades. Por fim, outro desafio é realizar a renderização do vídeo 360 na malha de triângulos do cilindro da *CAVE*, mantendo as proporções corretas para que não fique distorcido, levando em consideração as várias formas de se projetar conteúdo na *CAVE*. Pode-se também considerar um desafio a renderização e manipulação de diversos elementos (e.g. imagens, vídeo, estereoscopia, texto etc.) em uma única textura.

1.3 Motivação

Com o avanço da tecnologia, muitos processos cotidianos passaram a ser modernizados. Instituições financeiras e públicas, lojas, mercados, transporte, aquisição de conteúdo de entretenimento, como por exemplo, jogos, filmes, séries e música, entre outras coisas passaram a funcionar de maneira muito mais prática por meio da tecnologia. Com isso, a tecnologia que era considerada complexa e de uso exclusivo para "aqueles que a compreendiam", passou a ser algo não só natural, mas essencial na vida de parte das pessoas e hoje é praticamente impensável uma sociedade sem essa ferramenta.

Tendo isso em vista e levando em consideração que a evolução surge na busca do que está adiante, do próximo passo com relação a algo, outras formas de entregar conteúdo, em diferentes áreas, passaram a ter maior relevância. Cada vez se torna mais comum notícias sobre Inteligência Artificial, Internet das Coisas, Realidade Virtual e Aumentada etc. Estes assuntos têm sido fomentados nas diferentes

áreas do conhecimento e de negócio, tornando-os mais comuns à população geral e instigando nas pessoas o desejo em experimentar essas novas experiências no cotidiano.

Um bom exemplo de como essas ferramentas podem impactar positivamente na vida das pessoas é a sua aplicação na área da educação (ALMEIDA; JUSTINO, 2020). comentam no capítulo sobre ambientes imersivos de aprendizagem de seu livro que, com base na neurociência, oferecer diferentes estímulos aos estudantes torna o aprendizado mais robusto e permanente. Ambos os pesquisadores também ressaltam como a educação atual vislumbra um grupo de estudantes de forma homogênea, sem levar em consideração os diferentes perfis e as individualidades de cada estudante.

Segundo a pesquisa *2018 VR/AR in Research and Education Survey*, da plataforma *Internet2's*[3], que conta com mais ou menos trezentas e cinquenta universidades cadastradas, 28% das instituições de ensino superior estão envolvidas com implementação de realidade virtual, 18% já contam com uma implementação completa e aproximadamente metade está em fase de testes ou ainda não implementou completamente[4].

Com foco nessa tendência, a motivação deste projeto surgiu na necessidade de elevar algumas experiências, como por exemplo a educação, para um próximo nível utilizando ambientes imersivos colaborativos. Este tipo de ambiente permite combinar diferentes tecnologias (e.g. estereoscopia, mesa interativa etc.) em uma única ferramenta proporcionando múltiplos estímulos nos usuários. São diferentes dos *Head Mounted Displays* – ou óculos de realidade virtual, como são comumente conhecidos – pois propiciam uma experiência imersiva não apenas de forma singular, mas também para um grupo de pessoas, o que possibilita que os indivíduos interajam com o ambiente e entre si.

A geração de um ambiente imersivo requer um conhecimento complexo. É preciso compreender e conhecer as diferentes aplicabilidades deles, entender os prós e contras, bem como as possibilidades e limitações deste tipo de ferramenta, e por fim, criar/conceber um ambiente imersivo colaborativo simulado, que possa ser utilizado de diversas maneiras, seja na educação, entretenimento, exibição em

[3] <https://www.internet2.edu/> Acessado em: 12/02/2020

[4] <https://meetings.internet2.edu/media/medialibrary/2019/03/08/20190306-Fineman-MetaverseWG.pdf/> Acessado em: 12/02/2020

feiras, vídeo conferências, treinamentos, e quaisquer outros tipos de experiências imersivas que possam surgir futuramente. Portanto, uma segunda motivação do projeto é prover um conjunto de ferramentas e modelos que permitam a realização desta tarefa com reduzido esforço de tecnologia.

1.4 Organização

Essa dissertação está organizada em cinco capítulos. No primeiro, e atual, capítulo, é apresentado uma breve contextualização do problema, os objetivos que se pretende alcançar, os desafios que serão enfrentados e as motivações do trabalho. No segundo capítulo discute-se algumas pesquisas sobre estereoscopia e ambientes imersivos de aprendizagem, seguido de uma fundamentação teórica sobre os principais assuntos abordados neste projeto. No terceiro capítulo constam as etapas do método, informações sobre a elaboração da ferramenta ICE-S, criação da base digital de imagens de insetos e desenvolvimento da pojeção de modelos volumétricos com estereoscopia. No quarto capítulo são apresentados os resultados obtidos. Por fim, o quinto e último capítulo apresenta as considerações finais bem como os trabalhos futuros. Seguidamente encontram-se as referências bibliográficas.

2 Pressupostos Teóricos

Neste capítulo é feita uma revisão bibliográfica dos principais tópicos abordados nesta pesquisa. A primeira parte aborda conceitos fundamentais de estereoscopia e modelos volumétricos. Na segunda parte é apresentado uma perspectiva geral sobre ambientes imersivos de aprendizagem e diferentes técnicas de implementação já utilizadas em outros projetos do mesmo gênero. No final é feito um comparativo entre o atual estado da arte e o que se pretende atingir nesse trabalho.

2.1 Estereoscopia

Geralmente quando o termo estéreo é citado, se faz associação a tecnologias de áudio. O Som estéreo funciona com canais de áudio ligados em caixas de som independentes que fazem com que diferentes sons cheguem aos ouvidos e sejam processados e interpretados pelo cérebro. Essa interpretação do cérebro é o que causa a sensação de imersão nos sons que estão sendo transmitidos. (RAPOSO et al., 2004)

O mesmo conceito utilizado com áudio pode ser estendido para visualização, e é denominado estereoscopia. É uma técnica de simulação visual onde duas imagens de uma mesma cena são projetadas uma para cada olho, cada uma delas corresponde a pontos de observação ligeiramente diferentes e são chamadas de par estéreo. (MALARD et al., 2008) Isso torna possível perceber aspectos de tamanho, profundidade e posição dos objetos em uma cena, causando uma sensação de tridimensionalidade.

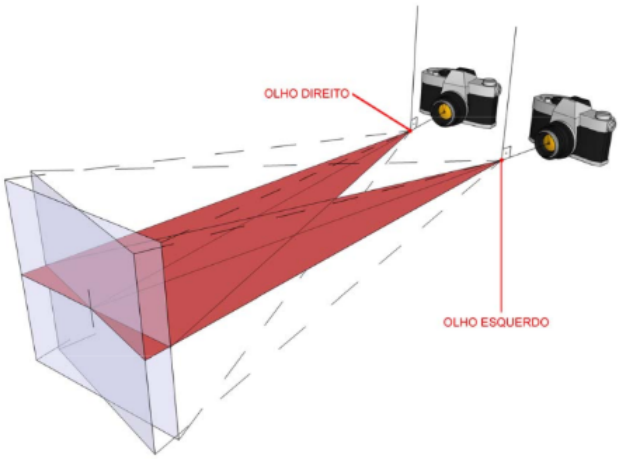


Figura 3 – Demonstração de como é feita a aquisição de imagens par estéreo.

Segundo o *The StereoGraphics Developer's Handbook*[5], um visor estereoscópico é um sistema ótico onde o componente final é o cérebro humano. É trabalho do cérebro fazer a junção das duas imagens e gerar essa sensação de tridimensionalidade, criando assim uma ilusão de ótica.

2.1.1 Estereoscópico

A palavra estereoscópio é uma combinação dos termos gregos *stereos*, que significa 'sólido' ou 'firme', e *skopeō*, que significa 'observar'. Apesar do conhecimento da visão binocular existir desde a antiguidade, foi só depois de 1830 que se tornou uma questão fundamental para a ciência. (CARVALHO, 2006) O primeiro cientista que sistematizou a sensação de profundidade, foi o físico britânico Sir Charles Wheatstone, ele inventou o primeiro estereoscópico no ano de 1838, um pouco depois da invenção da fotografia. O dispositivo funcionava da seguinte maneira: dois espelhos eram posicionados com ângulo de quarenta e cinco graus, a partir do centro correspondente ao foco visual do observador, um para cada olho. Paralelamente, em frente aos espelhos de ambos os lados, eram dispostos dois monitores que refletiam desenhos no espelho. (LUNAZZI; FRANÇA; MORI, 2015)

Em 1849, o físico Sir David Brewster, apresentou um modelo de estereoscópio portátil que utilizava lentes para ajudar na visualização devido a curta distância entre os olhos e o par estéreo de imagens. Também sugeriu que, além de desenhos,

[5] <http://www.cs.unc.edu/Research/stc/FAQs/Stereo/stereo-handbook.pdf> Acessado em: 17/02/2020

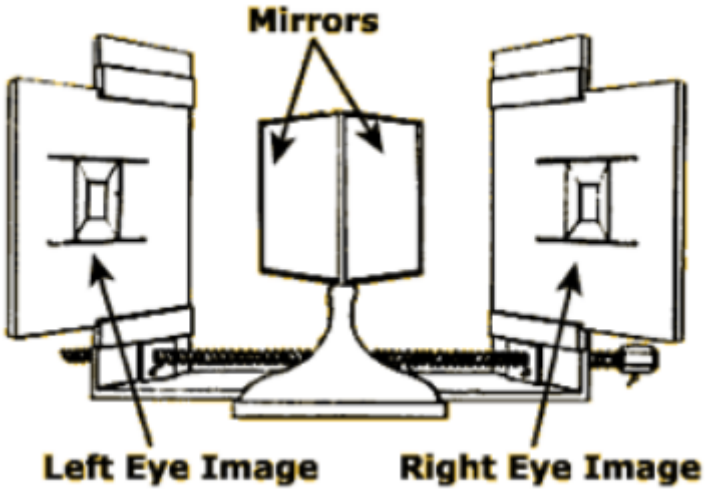


Figura 4 – Esboço do estereoscópico criado por Wheatstone.

fosse aplicado a fotografia. (PERES, 2016) O dispositivo foi apresentado ao público na *Exposição Universal de Londres* em 1851, e tornou-se um fenômeno, vendendo mais de mil cópias no Reino Unido nesse mesmo ano. (ADAMS, 1999)

O primeiro modelo estereoscópico com ilusão de movimento foi patenteado somente em 1861. Foi desenvolvido pelo engenheiro estadunidense Coleman Sellers, combinando as técnicas de visão binocular estereoscópica e discos rotativos. Esse mesmo conceito de discos rotativos foi anteriormente usado no *fantasmascopio* (do inglês - *Phantasmascope*), primeiro dispositivo de animação generalizada a criar uma ilusão de movimento. A invenção de Sellers foi batizada de *Kinematoscópio* (do inglês – *Kinematoscope*). (WESSON; LIPTON, 1984)

Muitos outros estudos foram realizados para aperfeiçoar a sensação de tridimensionalidade. Algumas técnicas posteriores tiveram bastante sucesso, e algumas são utilizadas até os dias de hoje.

2.1.2 Anáglifo

Uma das técnicas mais conhecidas na estereoscopia é o anaglifo. Foi inventada em meados de 1920 (WESSON; LIPTON, 1984) e fez muito sucesso comercialmente na exibição de filmes em 3D. Várias produtoras de filmes da época usavam essa tecnologia, e até hoje ainda é bastante utilizada.

Esse tipo de visualização é feita a partir da projeção do par estéreo em

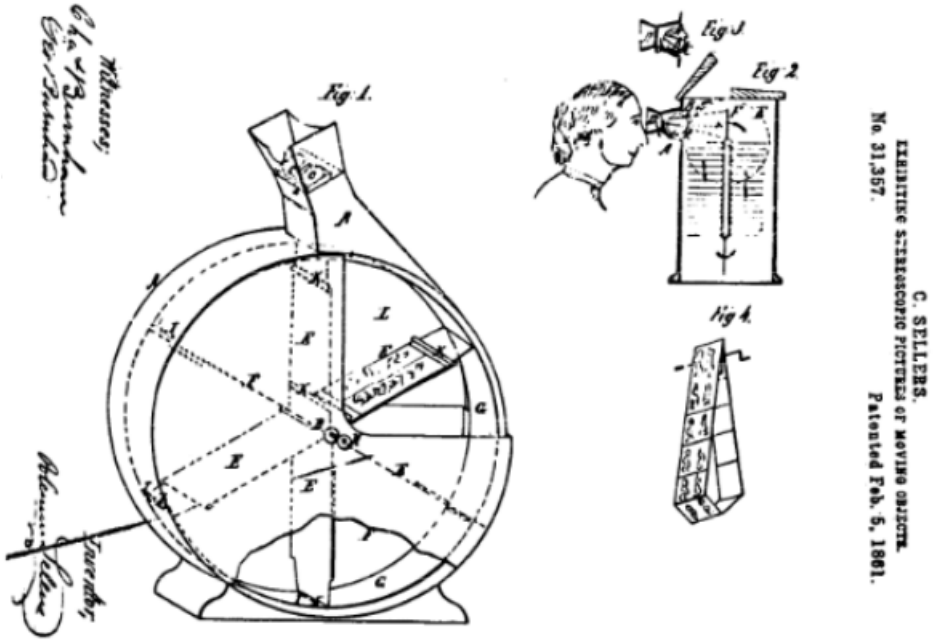


Figura 5 – Modelo de Kinematoscópio de Coleman Sellers. À esquerda, a primeira versão com cartões posicionados, e à direita, a segunda versão, com os cartões dispostos no formato de um cinto, movimentados por uma manivela.

duas cores diferentes, cada imagem de uma cor. Uma das imagens é projetada em azul, e a outra em vermelho. As imagens são projetadas de forma sobreposta e com uma leve distância entre as duas, para causar o efeito de profundidade. Isso é possível quando utilizado um óculos com filtro nas mesmas cores, pois cada olho recebe somente a imagem correspondente à cor complementar daquele olho, ou seja, a imagem em ciano será filtrada apenas pelo olho que estiver com a lente ciano, geralmente aplicada ao olho direito, e a vermelha pelo olho com a lente vermelha, usualmente aplicada ao olho esquerdo. (SANTOS; DIAS, 2011)

Mesmo com todo o sucesso do anaglifo antigamente, hoje não é tido como o melhor recurso existente. A projeção anáglifa, apesar de causar a sensação de profundidade, quando equiparada a tecnologias mais recentes, perde muito seu valor. Também não é uma projeção muito confortável para os olhos. A única vantagem desse método hoje é o seu baixo custo, tanto de produção do conteúdo, quanto do óculos, e é por isso que ainda é muito usado em impressões simples, produtos infantis, entre outros semelhantes.

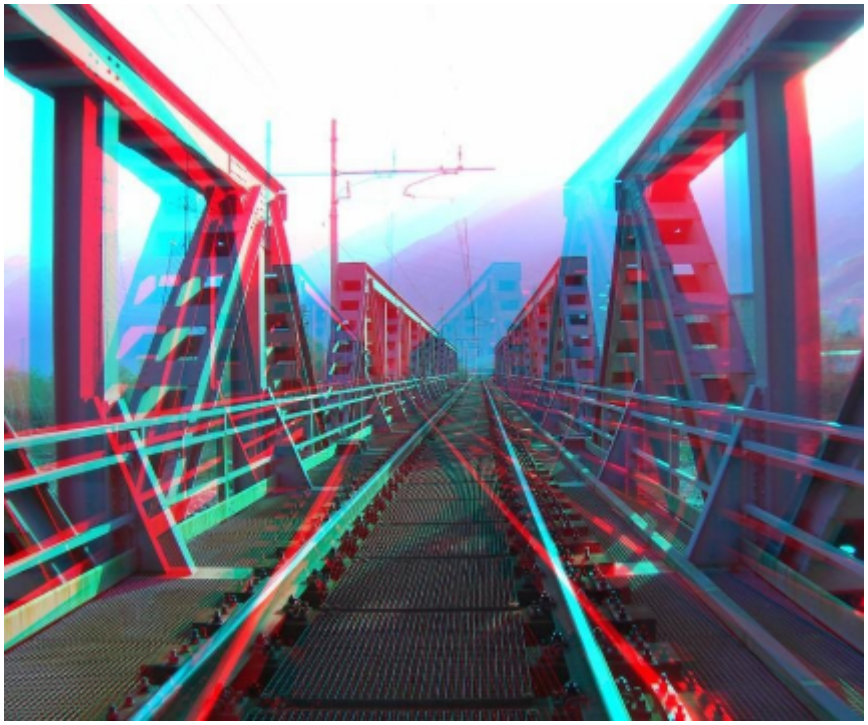


Figura 6 – Resultado de sobreposição do par estéreo anaglifo.

2.1.3 Polarização da Luz

Pensando na luz como um modelo ondulatório, é possível considerar que as ondas vibram em direções perpendiculares. Mas, também é viável forçar uma onda de luz a vibrar em apenas um dos planos. (KIRNER; SISCOUTTO, 2007) Quando várias ondas de luz são recebidas por um filtro polarizador, os eixos divergentes ao filtro são bloqueados, e somente o eixo restante passa a ser transmitido. No exemplo da Figura 7, um filtro polarizador vertical é aplicado, passando a vibrar somente a onda vertical de luz. O mesmo exemplo pode ser aplicado com um filtro horizontal, passando a emitir somente ondas de luz horizontais.

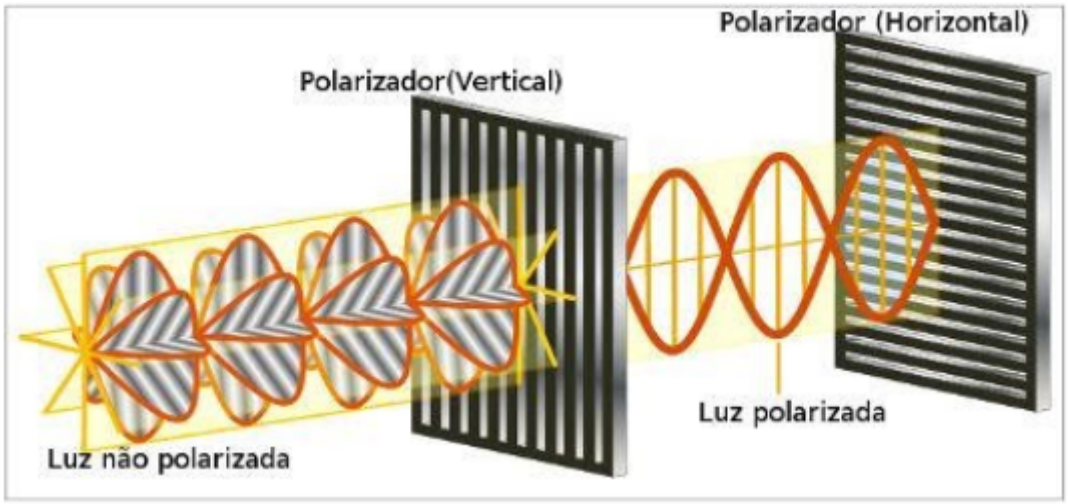


Figura 7 – Ilustração de ondas de luz perpendiculares que são filtradas por um filtro polarizador vertical e passam a emitir somente a onda vertical. Quando as ondas verticais chegam ao filtro horizontal, são bloqueadas.

Esse tipo de filtro é comumente usado em lentes de óculos de sol. Isso porque as ondas de luzes refletidas em uma determinada superfície, oscilam de acordo com o plano onde refletem. (NOGUEIRA, 2005) Se a luz do sol refletir em uma superfície plana, no caso de ser em um asfalto ou painel de carro, por exemplo, as ondas de luz serão horizontais, da mesma forma que o plano. Se um motorista quiser bloquear os raios de luz, basta usar um óculos polarizado com filtro polarizador vertical.

Dispondo desse meio, é possível criar uma projeção polarizada para obter a estereoscopia. Para isso, é preciso de dois projetores, cada um projetando a imagem de cada olho. Cada projetor tem um filtro polarizador posicionado em frente à lente, e emitem ondas de luz conforme a posição do filtro, na horizontal ou vertical. O usuário utiliza o óculos polarizado, que também possui lentes com filtros polarizadores verticais e horizontais, um eixo em cada lente. Ambos os projetores projetam suas imagens de forma sobreposta, dessa forma cada olho passa a enxergar somente a imagem que foi projetada pelo filtro polarizador equivalente. (SANTOS, 2000)

Nem sempre é preciso de dois projetores para gerar a estereoscopia. Hoje podemos controlar os dispositivos como TV's e projetores por meio de *software*. Basta dispor de duas imagens, com diferença angular entre elas, uma ao lado direito e outra ao lado esquerdo, cada uma ocupando metade da tela. O *software*

mistura as duas imagens, criando o mesmo efeito de sobreposição utilizado em dois projetores, e é trabalho da película polarizadora juntamente ao óculos, exibir a imagem correta para cada olho. Em algumas Tv's, além do filtro polarizador, é utilizada mais uma camada chamada de *FPR* (do inglês *Film-type Patterned Retarder*), responsável por polarizar a luz de forma circular, permitindo que o usuário possa inclinar a cabeça sem perder de vista a imagem projetada, fator causado pelo desalinhamento da onda de luz com o filtro polarizador da lente do óculos tradicional (polarização linear).

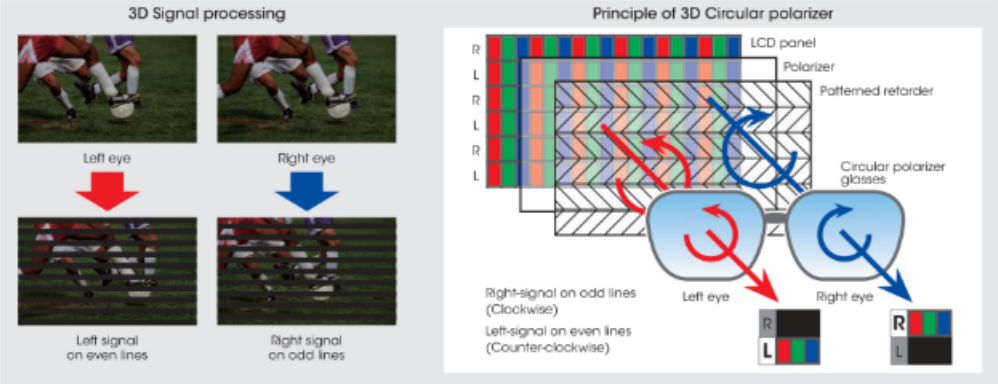


Figura 8 – Televisão com polarização circular: Na esquerda, demonstração dos sinais recebidos por cada olho. Linhas ímpares para o olho esquerdo e pares para o olho direito. À direita, ilustração dos princípios da polarização circular.

2.1.4 Óculos Obturador Sincronizado

A tridimensionalidade polarizada é conhecida como 3D passivo. Mas também existe o modelo de 3D ativo, que funciona utilizando um óculos obturador sincronizado, e ao contrário do modelo polarizado, não utiliza a polarização da luz para funcionar. As lentes do óculos são feitas de cristal líquido e ele funciona por meio de um controle eletrônico. A imagem de cada olho é exibida separadamente, ou seja, quando a imagem da direita é mostrada, a lente direita fica transparente e a da esquerda opaca, e vice-versa. (SISCOUTTO et al., 2004) Como essa troca acontece muito rápido, é imperceptível ao olho humano, fazendo com que o cérebro processe as duas imagens ao mesmo tempo, obtendo as características de profundidade e volume do 3D.

Uma das grandes dificuldades nesse método, é fazer com que as imagens apareçam na sequência de forma sincronizada. Quando há disparidade na sincronização da amostragem, a projeção se torna desconfortável, causando cansaço excessivo e dores de cabeça no usuário.

Alguns defeitos desse modelo, em comparação ao modelo polarizado, são: o óculos precisa de bateria, e para ser usado precisa estar sempre carregado. Poucos óculos são compatíveis com outros fabricantes, tornando a tecnologia mais exclusiva para cada marca. O custo do produto é superior ao polarizado, fazendo com que não seja acessível à todos. Como precisa estar bem sincronizado para o bom funcionamento, é muito mais fácil obter estabilidade com o modelo polarizado, tendo em vista que os pares estéreo são projetados simultaneamente e, geralmente o 3D ativo causa maior cansaço na vista dos usuários.

A grande vantagem é que, quando realizado com sucesso, esse tipo de projeção é muito mais fiel às cores e à alta resolução, tornando a experiência muito mais atrativa aos olhos do que a projeção por polarização de luz.

2.2 Ambientes Imersivos

Desde o início da exploração do ser humano sobre conceitos de profundidade da visão binocular em projeções de imagens e filmes, fica evidente a busca por um recurso que permita que pessoas possam imergir no ambiente virtual. Desta forma, acredita-se que a imersão teve início com o surgimento do primeiro dispositivo estereoscópico, porém, somente em meados dos anos 1950/1960 é que a realidade virtual foi inventada.

Assim como várias outras tecnologias, a *RV* também teve seu surgimento como recurso militar dos Estados Unidos. Devido aos muitos acidentes e por causar prejuízo ao governo norte americano, criou-se uma necessidade de ensinar os pilotos como manusear os controles de uma aeronave, bem como outras condições de voos. Com isso, por volta de 1910, inventores estadunidenses criaram treinadores de voos, hoje mais conhecidos por simuladores. (MACHADO, 2016)

Algumas invenções foram testadas desde então, a que teve maior destaque foi o modelo de Edwin Link. Externamente, possuía uma aparência de avião de brinquedo, e por mais que demonstrasse superioridade com relação aos simuladores anteriores, devido à falta de tecnologia da época, ainda não era visto como um

substituto de um voo real. Foi só nos anos 30, com a adição de uma cabine de pilotagem que o dispositivo foi levado a sério. Durante a Segunda Guerra Mundial, foram encomendados vários simuladores de Link para equipar o setor de treinamento dos Estados Unidos e seus aliados. (PAGE, 2000)

Figura 9 – Treinamento no simulador de Edwin Link, o Link Trainer, por pilotos da Royal Canadian Air Force.

A indústria de entretenimento também foi de grande contribuição para a evolução da imersão. Em 1962, Morton Heilig inventou o Sensorama, um simulador com visualização 3D em primeira pessoa, que emitia sensações como vento e aroma. O *design* ainda incluía uma luz ultravioleta que higienizava a superfície de visualização para o próximo usuário. Para utilizar o simulador, a pessoa se sentava em um banco, e posicionava a cabeça em uma espécie de cúpula. Uma das experiências simulava uma motocicleta, o espectador começava a sentir o banco e o guidão tremer, como se estivesse montado em um veículo real e os sons do motor e do ambiente eram emitidos em *full* estéreo. (TURI, 2014)

Figura 10 – Primeiro dispositivo de imersão focado em entretenimento: Sensorama.

2.2.1 HMD - Head-Mounted Display

Uma das primeiras coisas que vem na mente quando se fala em realidade virtual e imersão, é de uma pessoa usando um dispositivo no rosto, que cobre todo o seu olho. De fato, os óculos de realidade virtual roubaram a cena nesse assunto e são os dispositivos mais populares dentre todos os existentes. Mas se engana quem pensa que essa tecnologia é algo recente.

O primeiro produto que foi considerado um *HMD*, foi o *Headsight*, inventado em 1961, pela empresa Philco. Criado para enfrentar situações de risco, incorporava uma tela e sistema de rastreamento a um sistema de câmera de circuito fechado, dessa forma era possível observar um ambiente real remotamente. Foi usado em operações militares para treinamento, e por pilotos de helicóptero, permitindo que tivessem uma visão clara quando sobrevoavam no escuro. (SRIVASTAVA; CHAUDHURY; DAS, 2014)

Alguns anos depois, em 1968, o cientista da computação Ivan Sutherland apresentou o primeiro dispositivo da categoria que permitia que o usuário interagisse com o ambiente virtual. (SUTHERLAND, 1968) O sistema era conectado ao computador, transmitindo o cenário tridimensional para os olhos do utilizador,

fazendo a cena virtual parecer o ambiente real. O aparelho também era capaz de detectar a posição da cabeça e dos olhos da pessoa, e atualizava a visão estereoscópica conforme os movimentos realizados. (BOAS, 2013) Essa característica, e a de interagir com o ambiente, foi o grande diferencial dessa invenção, fazendo com que o usuário tivesse a sensação de fazer parte do cenário tridimensional. O óculos de Shuterland recebeu o nome de "The Sword of Damocles", em homenagem a história da mitologia grega de Dâmocles.

Figura 11 – The Sword of Damocles, primeiro dispositivo *HMD* com interação e movimento. Desenvolvido por Ivan Shuterland.

Voltando para a *RV* aplicada a treinamentos militares, entre os anos de 1986 a 1989, o inventor estadunidense Thomas Furness dirigiu o programa da Força Aérea Americana, o Super Cockpit. Muito mais avançado do que os equipamentos da década de 30, o programa de Furness contava com um dispositivo *HMD* para atingir seus objetivos. O sistema era capaz de projetar mapas 3D, imagens infravermelhas e radares, em um ambiente tridimensional imersivo, e permitia que o piloto escutasse e falasse em tempo real (LENOIR; LOWOOD, 2002)

A partir dos anos 90, com a explosão dos videogames, muitas empresas do setor passaram a investir nessa tecnologia. Porém, visto que jogos de videogame são *softwares* complexos e necessitam de hardwares potentes para entregar uma boa experiência ao usuário, não se obteve muito sucesso na época, a realidade virtual ainda não estava pronta para os jogos eletrônicos.

Até então, a realidade virtual e aumentada eram vistas como uma coisa só. Foi só em 1990 que o termo realidade aumentada foi usado pela primeira vez pelo pesquisador da Boeing, Tom Caudell. Caudell desenvolveu um sistema de *RA* para auxiliar mecânicos a encontrar as conexões corretas entre os cabos nos motores de aeronaves. (CURTIS et al., 1999) Desde então, a *RV* e a *RA* passaram a ser independentes uma da outra. Apesar de serem tecnologias muito semelhantes, possuem características distintas, e podem ser utilizadas para diferentes propósitos.

Hoje, na realidade virtual, o *HMD*, também popularmente conhecido como óculos de *RV* e *headset*, faz com que o usuário fique totalmente imerso ao ambiente, isso se deve ao fato de o dispositivo cobrir completamente a área dos olhos e ser fechado, não permitindo que a pessoa interaja com elementos do cenário real. A visão do ambiente tridimensional pelo utilizador é feita utilizando as técnicas de estereoscopia, por meio de duas câmeras, ou seja, quando se desenvolve um *software* para um óculo de *RV*, é preciso criar dois ambientes de visualização, um para o olho esquerdo, e outra para o olho direito. Assim como em qualquer aplicativo de visão binocular, ambos os olhos enxergam a mesma cena, contando apenas com uma pequeno deslocamento entre uma e outra. O cérebro faz a associação de ambas as cenas, criando, então, o ambiente virtual tridimensional.

Figura 12 – Cena do jogo Battlefield 1. Imagem demonstrando a aplicação da estereoscopia para gerar profundidade.

Existem muitos tipos de óculos, alguns funcionam até mesmo com o celular.

Estes são basicamente uma casca vazia, com um espaço para colocar o celular, de forma que a tela fique virada para os olhos. O funcionamento é feito com o auxílio de aplicativos do próprio aparelho, fazendo com que o *HMD* funcione apenas como suporte. Outros funcionam de forma dependente à um dispositivo específico, como por exemplo o *PlayStation VR*, que precisa do console para rodar os jogos, ou o *Oculus Rift* (entre muitos outros), que precisam de um computador com jogos compatíveis. Recentemente, a *Oculus*, lançou seu primeiro modelo integrado, o *Quest*. Esse modelo pode ser conectado ao computador para jogar os jogos compatíveis com o modelo, mas é opcional. O diferencial do aparelho é justamente seu funcionamento de forma independente.

É comum que esses dispositivos possuam algum tipo de controlador, pois são eles que possibilitam a interação do usuário com o ambiente virtual, fator que em jogos e ambientes de treinamento profissional passa a ser fundamental. Os controladores possuem sensores de movimento, que realizam o reconhecimento da posição das mãos do utilizador, assim como botões de *input*, o que, como em qualquer outro controle, possibilita configurar determinadas ações conforme programado no *software*. Além disso, a maioria conta com interação de voz, através de microfone.

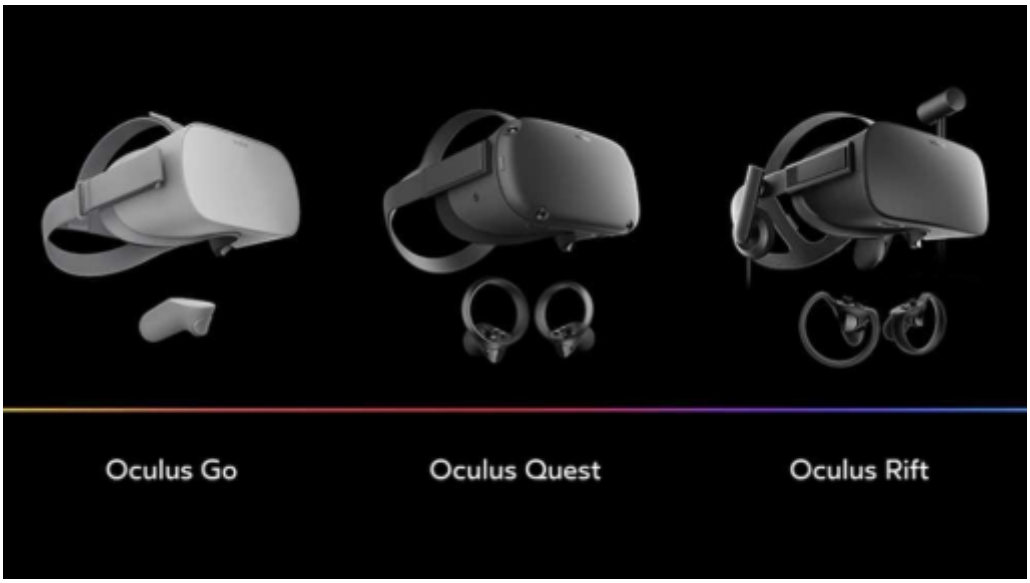


Figura 13 – Foto dos diferentes modelos de *headsets* da marca *Oculus*.

A realidade aumentada ainda é menos popular do que a realidade virtual, isso porque durante muitos anos essa tecnologia ficou mais restrita dentro das

universidades e filmes de ficção científica. Mas nos últimos anos esse recurso vem ganhando cada vez mais popularidade, principalmente após o lançamento do *Google Glass*, considerado um dos maiores momentos da realidade aumentada com relação ao público geral.

Um pouco diferente dos *headsets* de realidade virtual, os óculos de realidade aumentada são muito mais semelhantes à um óculos de grau. Possuem uma lente transparente, que permite que o usuário visualize o ambiente real, e a projeção virtual é feita de forma sobreposta à realidade. Dessa forma é possível, criar telas flutuantes de transmissão, textos flutuantes com informações sobre alguma coisa, objetos tridimensionais, entre outras várias coisas.

O *HMD* mais popular da categoria é o *Google Glass*, que depende, quase totalmente, de um *smartphone* para as principais funcionalidades. A conexão é feita utilizando tecnologia *Bluetooth* e é necessário baixar um aplicativo próprio para ativar as funcionalidades. A projeção dos elementos virtuais é feita somente em um olho, e são projetadas no canto superior direito do campo de visão. A parte frontal do óculos possui uma tela, e dentro dela existe um prisma que reflete a imagem do *Glass* para a nossa retina. Como o óculos possui um sensor que detecta o movimento dos nossos olhos, a imagem projetada não atrapalha a visão real, basta mudar o foco de visão para o plano real que a projeção fica opaca e diminui. (PHILIPE, 2014)

O *Glass* possui um sensor lateral que ativa várias funcionalidades, como atender o telefone ou tirar fotos, mas também é possível acessar ferramentas por comando de voz. Um grande destaque do modelo é como emite som para o usuário, pois não possui fones de ouvido e nem auto falantes. O *Google* usa uma tecnologia que emite vibrações ao cérebro que se transformam em som, dessa forma, somente a pessoa que estiver com o *Glass* pode ouvir.

Outro modelo interessante de realidade aumentada é o *Moverio*, da marca *Epson*. A projeção virtual funciona um pouco diferente do aparelho do *Google*, pois assim como na estereoscopia, é feita de forma binocular. Cada lente possui uma microtela de projeção, e a inserção dos elementos tridimensionais é semelhante a realidade virtual, porém, sem perder o ambiente real de vista. Assim como o *Glass*, ele possui câmeras e sensores, e funciona integrado ao sistema *Android*. A diferença é que no Moverio, é possível conectar um controle ao óculos, possibilitando a interação por meio dele.

O aparelho da *Epson* é bastante utilizado na integração com *Drones* aéreos, pois conta com um aplicativo próprio de voo, onde o usuário é capaz de enxergar o ambiente real e a visualização da câmera do *drone* simultaneamente. Além disso, alguns modelos como o *BT-2000*, são projetados especificamente para serem fixados em capacetes de obra, possibilitando a utilização do produto em ambientes de trabalho que exigem proteção, sem que o produto fique posicionado fragilmente no rosto do usuário.

Figura 14 – À esquerda, modelo convencional do *Google*, o *Glass*. À direita, modelo de trabalho da *Epson*, o *Moverio PRO BT-2000*.

2.2.2 CAVES - CAVE Automatic Virtual Environment

Com objetivo de trazer a imersão, mas com uma abordagem diferente, as *CAVES* permitem que o usuário, literalmente, entre no ambiente de projeção. Existem diferentes formatos desses ambientes, mas os primeiros modelos foram criados em forma de um cubo. Uma *CAVE* pode ser pequena ou grande, dependendo da quantidade desejada de pessoas no local. De forma geral, é um ambiente de projeção, de vídeo e áudio, tridimensional, que proporciona uma experiência de imersão para mais de uma pessoa ao mesmo tempo.

Assim como o próprio nome sugere, uma *CAVE* cúbica possui seis lados de projeção, podendo, ou não, utilizar todos os lados em uma simulação. A escolha dos lados que serão projetados varia muito conforme o objetivo da aplicação, por exemplo, em um modelo criado na América Latina optou-se por utilizar as quatro paredes e o piso, pois o modelo é voltado para projeções de ambientes automotivos, petrolíferos, energético, aeronáutico, entre outros. (ZUFFO et al., 2001) Para essas simulações, é importante que a pessoa tenha uma visão partindo da sua perspectiva

para baixo, tornando-se desnecessária, na maioria dos casos, projeções acima do usuário.

As paredes do ambiente são semitransparentes e projeção é feita de fora para dentro, em alta definição, com o auxílio de espelhos. Os espelhos e projetores ficam do lado externo, e são posicionados de forma estratégica. Cada projetor tem seu foco direcionado para seu respectivo espelho, e o espelho reflete a imagem para a parede semitransparente do ambiente. Dentro do cubo, o usuário consegue visualizar os planos de projeção de forma unificada, ou seja, todas as imagens projetadas se tornam uma única cena de simulação.

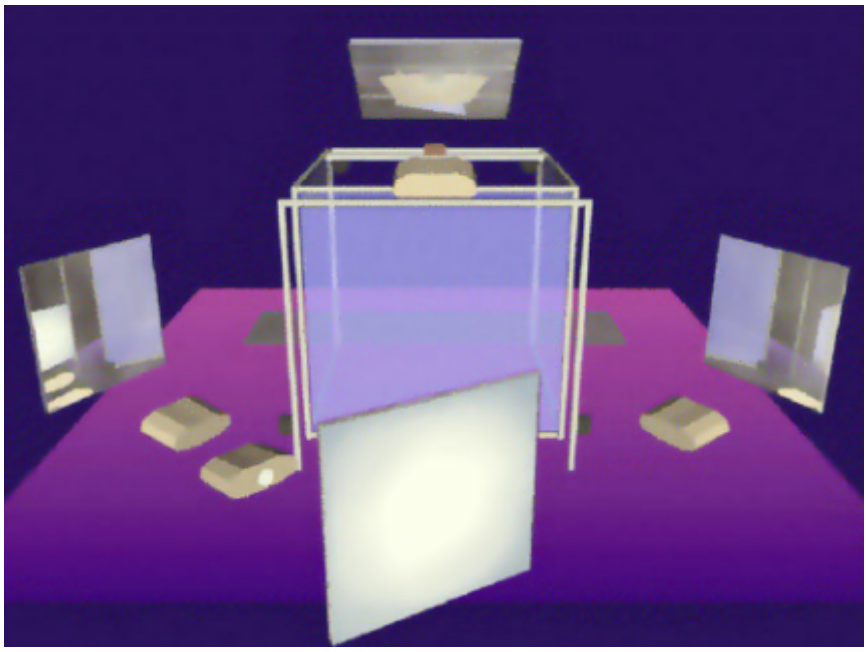


Figura 15 – Representação da *CAVE Automatic Virtual Environment* da Universidade de Illinois.

Para obter a sensação de profundidade, é utilizada a técnica de estereoscopia, ativa ou passiva, dependendo da preferência dos desenvolvedores do sistema. Como toda aplicação estereoscópica, os visualizadores devem usar óculos compatíveis para obter a sensação de profundidade. A movimentação do ambiente pode ser feita por meio de um controle de *input*, como um de videogame, por exemplo, ou com dispositivos que possuem sensores de movimento, como um *Kinect*[6]. Nessa abordagem, é importante salientar, que apenas um usuário deve ser reconhecido pelos sensores, pois vários movimentos diferentes podem confundir o sistema.

[6] <https://news.xbox.com/en-us/2013/06/06/privacy/> Acessado em: 11/03/2020

Conforme o usuário líder (pessoa portando controle ou detectada pelos sensores) se movimenta dentro da *CAVE*, as projeções estéreo e suas perspectivas são atualizadas, causando uma sensação de movimento na cena a partir do condutor. Para os outros espectadores dentro do ambiente, a sensação é semelhante à de um passageiro dentro de um ônibus (KENYON, 1995).

Conforme essa tecnologia foi evoluindo, outros formatos foram sendo aplicados, hoje já existem *CAVES* em formatos de domos e cilindros, e alguns até são construídos com material inflável, para que possam ser montados e desmontados conforme a necessidade de uso. O objetivo continua sendo o mesmo, o que muda um pouco é a forma como se faz a projeção, e a visualização dos usuários.

Os ambientes imersivos em domo, são construídos no formato de uma meia-esfera, com projeção em 360°. São bem semelhantes à teatros *fulldome*, o que muda é que nos teatros a projeção é feita somente na parte superior, pois a cúpula assume o lugar do teto do ambiente, e em uma *CAVE* a semiesfera é construída a partir do plano do piso e exclui o topo da semiesfera, abrangendo todo o ambiente somente nas laterais.

Diferente do modelo cúbico, os projetores ficam posicionados na parte interna, excluindo a necessidade de espelhos e paredes semitransparentes. Para obter uma cena completa nas laterais de uma cúpula, quatro, ou mais, projetores são necessários, e ficam posicionados na parte superior, cada um com o foco de projeção abrangendo um espaço equivalente ao número de projetores dividido pelo tamanho total em graus. O maior desafio desse formato, é corrigir os erros de perspectiva da cena, uma vez que uma projeção normal é feita em um plano reto, é preciso calcular as deformidades do plano circular, de outra forma, não seria possível simular uma cena contínua.

Para delimitar o espaço onde cada pedaço da cena deve ser renderizada, a semiesfera é dividida dentro do *software*. Essas divisões são feitas a fim de obter quatro lados, porém, diferente de uma *CAVE* cúbica, o domo não possui uma quinta face, pois não existe projeção no piso e nem no teto. Dessa forma, o plano de visualização ainda é contínuo em 360°, mas dentro do programa o domo possui divisões laterais. Para obter visão binocular estereoscópica, basta projetar o par estéreo de imagens de forma sobreposta, e que, como sempre, os usuários estejam utilizando óculos compatíveis com o tipo de estereoscopia programada.

A *CAVE* de formato cilíndrico é praticamente igual ao formato em domo,

Figura 16 – Modelo de *CAVE* no formato domo produzida pela empresa Igloo Vision.

a única diferença é que o plano de projeção possui menos deformidade, pois não assume o formato de uma semiesfera. O eixo vertical é reto, e a curvatura só acontece no eixo horizontal, isso faz com que a complexidade na renderização da cena seja menor, por não ser preciso fazer muitas correções.

Figura 17 – Modelo de *CAVE* no formato cilindro produzida pela empresa Igloo Vision.

2.3 Considerações Finais

O atual capítulo apresentou um referencial teórico e uma revisão bibliográfica com foco na estereoscopia e *CAVES* imersivas. Foram apresentados a evolução na técnica de estereoscopia e suas utilizações mais recentes, focando no desenvolvimento de aplicações para *Head-Mounted Displays* e *Cave Automatic Virtual Environment*. Esse estudo foi fundamental para o desenvolvimento deste projeto, pois permitiram adquirir um conhecimento aprofundado sobre cada uma das tecnologias que serão utilizadas neste trabalho. A revisão bibliográfica possibilitou conhecer diferentes trabalhos já realizados de temas similares, observando as dificuldades enfrentadas em cada época e o quanto ainda é possível evoluir no assunto.

O próximo capítulo apresenta o método proposto para o desenvolvimento deste projeto, incluindo as etapas que foram realizadas e uma descrição detalhada de cada um dos processos de desenvolvimento.

3 Método

3.1 Considerações Iniciais

Neste capítulo é apresentado o processo de desenvolvimento deste trabalho. Sendo assim, a Seção 3.2 apresenta uma visão geral do método proposto. A Seção 3.3 descreve e discute as principais informações relativas à base de imagens. Na Seção 3.4 são apresentadas as características do desenvolvimento do ICE-S e de outras ferramentas adicionais, descrevendo cada um dos processos realizados durante esta etapa. Na Seção 3.5 são apresentadas as ferramentas que foram utilizadas na criação do *software*. Finalmente, na Seção 3.6 são feitas as considerações finais sobre o atual capítulo.

3.2 Visão Geral

As etapas que serão realizadas durante o desenvolvimento são:

- Aquisição – realizar captura de imagens da semiesfera superior de insetos em 360° utilizando o *F2S2* (*Full Frames Semi-spherical Scanner*);
- Pós-processamento – remover plano de fundo das imagens adquiridas com *software BipApp* baseado em classificadores e aplicação de técnicas de aprimoramento;
- Desenvolvimento da estereoscopia – criar algoritmo aplicando técnicas de estereoscopia em imagens dos insetos para gerar uma projeção volumétrica do modelo;
- Desenvolvimento de interface de controle – elaborar uma interface de usuário que simule um modelo tridimensional a partir de imagens bidimensionais e que permita aplicar as transformações geográficas no modelo de visualização simetricamente ao modelo volumétrico (estereoscópico);
- Desenvolvimento de simulador de *CAVE* – implementar ferramenta que possibilite a criação de uma *CAVE* dinâmica com base no raio, resolução e tamanho em graus de circunferência.

A aquisição de imagens foi realizada em parceria com o grupo de trabalho da área de Entomologia da Escola de Ciências da Vida e Medicina da *PUCPR*, que forneceu os insetos utilizados nesta etapa, além de informações sobre as características importantes para o estudo de cada modelo.

Na Figura 18 é mostrado em fluxograma, todas as etapas do método proposto, com intuito de simplificar a visualização das diferentes fases de desenvolvimento do projeto.

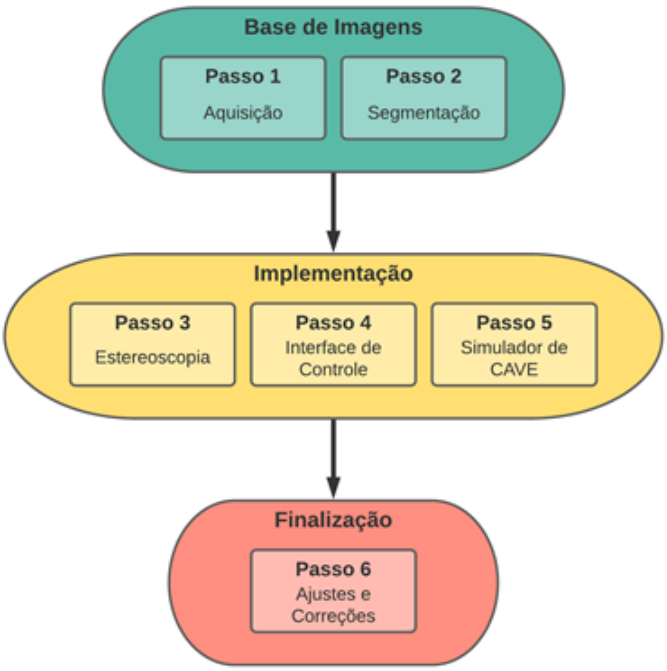


Figura 18 – Fluxograma do ciclo de desenvolvimento. Fase inicial: produção da base de imagens; Fase intermediária: implementação de soluções/algoritmos de visualização e controle; Fase final: refinamento dos processos. Fonte: Própria.

3.3 Base de Imagens

Um dos pontos mais importantes para criar uma base de imagens com acuidade visual é a qualidade no processo de aquisição das imagens. Se a captura não for bem executada, as imagens podem apresentar falta de foco, excesso ou escassez de iluminação, encobrir detalhes relevantes no modelo, entre outros problemas. Caso isso ocorra, todo o processo deve ser recomeçado. Esse ciclo deve ser realizado

até que todas as imagens apresentem uma visualização sem perda de detalhes e/ou nitidez.

3.3.1 Aquisição

A aquisição das imagens foi feita no *F2S2*. Deve-se ressaltar que o *F2S2* foi desenvolvido pelo colega de laboratório Diogo Olsen (OLSEN, 2020), no contexto do seu projeto de doutorado. O *F2S2* é um *scanner* tridimensional que gera modelos digitais interativos em alta definição de objetos reais. Segundo (SILVA et al., 2019), o termo "scanner semiesférico" refere-se ao fato de que o *F2S2* tira fotos enquanto move a câmera, de modo a formar uma semiesfera ao redor do objeto, cobrindo todos os ângulos de acordo com um intervalo predefinido; e o termo "quadros completos" refere-se à aquisição fotográfica de todos os ângulos do objeto.

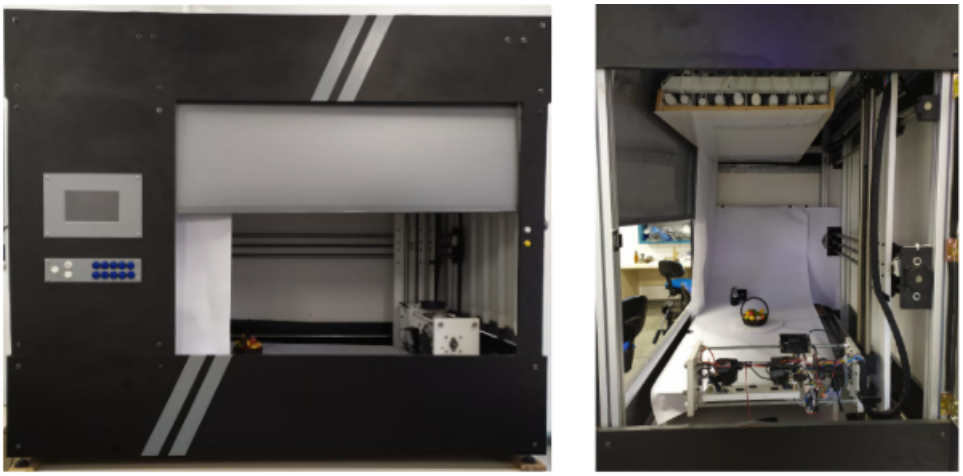


Figura 19 – À esquerda, imagem da lateral do *F2S2*. À direita, imagem da parte interna do *F2S2*. Fonte: Própria

O *F2S2* realiza realiza a aquisição do modelo utilizando fotografias, de forma que seja possível visualizá-lo digitalmente com os mesmos detalhes do objeto real, preservando os aspectos de cor, forma e textura (SILVA et al., 2019). O resultado do processo de aquisição é uma coleção de imagens (*pack* de imagens) compostas por *streams* e *frames*, sendo *stream* o nome que se dá ao conjunto de fotos adquiridas na mesma variação vertical, nas diferentes posições horizontais, e *frame* cada imagem armazenada em um *stream2D*, ou seja, cada *frame* está contido em um *stream* (OLSEN, 2020).

O *F2S2* possui uma base central giratória onde deve ser posicionado o objeto que será fotografado. A peça precisa estar o mais fiel possível ao centro do prato. Desta forma, o foco será mantido em todas as imagens garantindo maior acuidade visual. O ajuste de posicionamento é feito com o auxílio de *lasers* niveladores e no total foram utilizados 3 *lasers*, um deles posicionado na parte frontal ou traseira e os outros dois nas laterais (cf. Figura 20). Posteriormente, a base é rotacionada em 180° e em caso de descentralização deve-se movimentar o objeto 50% da distância de desajuste, e os *lasers* a outra metade para o lado onde houver maior divergência (SILVA, 2020). O processo deve ser repetido até que todos os 3 *lasers* estejam alinhados em ambas as faces (traseira ou dianteira), aos pontos de interesse definidos no início do processo de alinhamento.

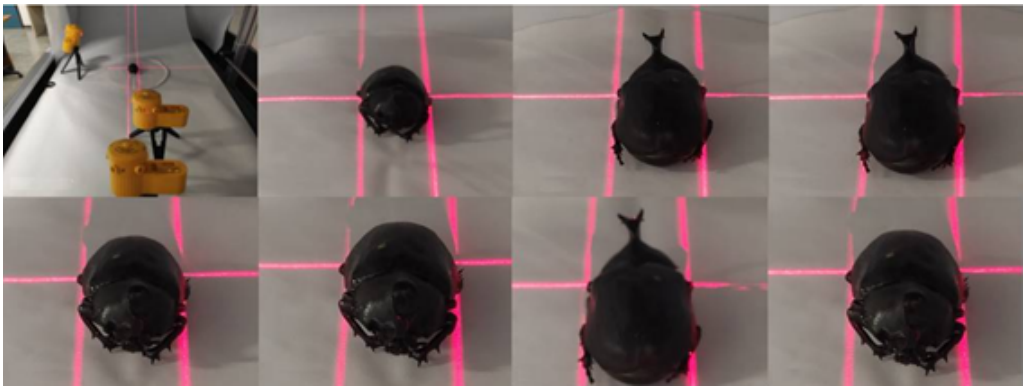


Figura 20 – Demonstração do processo de alinhamento do inseto com lasers niveladores no centro da base giratória. Fonte: Flávio de Almeida e Silva.

Os insetos são modelos particularmente difíceis de ser alinhados corretamente por possuírem muitos detalhes e assimetrias. Em modelos pequenos, como por exemplo joaninhas e abelhas, os *lasers* paralelos ficam muito próximos um do outro e a movimentação do inseto e do *laser* tem de ser feita minuciosamente. Além disso é preciso tomar muito cuidado para alguns detalhes não passarem despercebidos durante o processo, pois muitos deles são quase imperceptíveis a olho nu.

Com o modelo alinhado ao centro da base, deve-se configurar manualmente a câmera e a iluminação do próprio *scanner*. Os parâmetros relevantes da câmera que devem ser verificados são: foco, *ISO*, abertura e *zoom* (quando necessário). Os ajustes destes valores são feitos em paralelo à iluminação do *scanner* e deve-se realizar algumas tomadas de teste para verificar se a nitidez e a exposição à luz estão satisfatórias. Por mais que em alguns casos as configurações possam

ser reaproveitadas, é importante que cada modelo seja ajustado conforme suas particularidades, por exemplo, alguns insetos são mais escuros e podem necessitar de mais iluminação, alguns possuem um aspecto mais brilhante, que refletem a luz, e podem ter seus detalhes ofuscados se muita iluminação externa for utilizada, insetos muito pequenos ou com muitos detalhes normalmente precisam de *zoom* para ficarem mais visíveis, dependendo do tamanho do modelo o *zoom* não é suficiente e é preciso utilizar uma lente macro para obter todos os detalhes.

O *F2S2* possui um *software* de controle que oportuniza realizar o escaneamento de 3 formas diferentes, sendo eles: único, parcial e completo. O modo de aquisição único, como o próprio nome sugere, realiza a captura de uma única imagem dado um determinado ângulo em x e y e costuma ser utilizado para testes de acuidade. O escaneamento parcial possibilita capturar uma parte do objeto em um intervalo de ângulo xy determinado. Por fim, o escaneamento completo, executa a aquisição de toda a parte visível do objeto, capturando todas as suas faces em 360°, obtendo uma espécie de semiesfera do modelo conforme exibido na Figura 20.

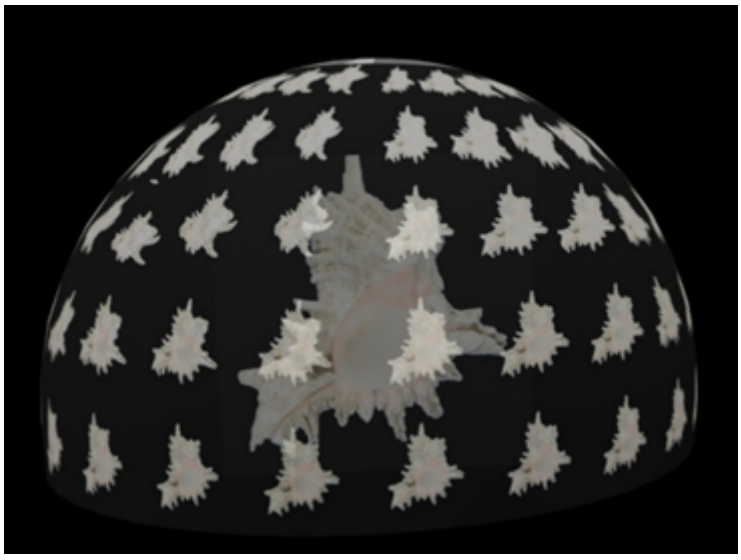


Figura 21 – Representação da semiesfera de aquisição da imagem no modo de escaneamento completo. Fonte: Autoria de Diogo Olsen e Flávio de Almeida e Silva com design de Felipe Teixeira de Almeida e Silva.

A última etapa antes de iniciar o processo automático de aquisição das imagens, é efetuar as configurações no *software* de controle do *F2S2*. Os parâmetros que devem ser fornecidos são: largura e altura (em milímetros) do objeto que está

sendo escaneado, altura (em milímetros) da base onde a peça está posicionada, distância (em milímetros) da câmera com relação ao modelo, intervalo angular entre as imagens de aquisição que deve ser definido separadamente para x e y, por fim, o tempo (em segundos) para realizar o disparo da câmera.

Estes ajustes definem a posição inicial da câmera, o quanto a base giratória deve rotacionar, o quanto a câmera deve se locomover e o tempo que a câmera tem que esperar para realizar a captura da imagem. A etapa de configuração é importante para obter resultados com alta acuidade visual, uma vez que os parâmetros não estejam corretamente configurados o resultado será comprometido.

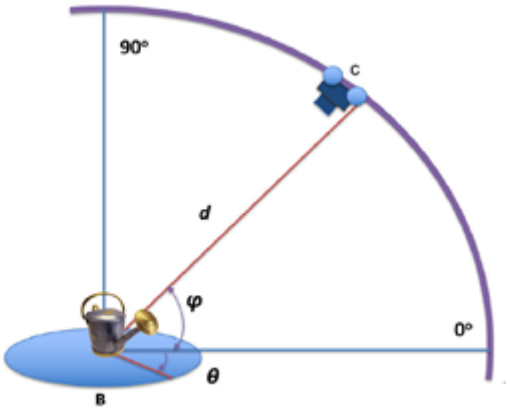


Figura 22 – Desenho de posicionamento dos elementos no processo de aquisição do F2S2. Fonte: Diogo Olsen.

Na aquisição dos insetos dois modelos foram escaneados especificamente para este projeto, sendo um gafanhoto e uma borboleta. Alguns outros modelos também foram adquiridos para validação e testes em outros projetos. Os valores aproximados que foram utilizados na configuração do *F2S2* estão dispostos na Tabela 1.

Inseto	Alt.	Larg.	Alt. base	Dist. câmera	Ângulo	Tempo
Gafanhoto	20mm	55mm	150mm	250mm	3°	4 seg.
Borboleta	10mm	40mm	150mm	250mm	3°	4 seg.

Tabela 1 – Parâmetros de configuração utilizados para cada inseto escaneado.

A aquisição completa de ambos os modelos, resulta em um *pack* de 3720 imagens sendo 120 *frames* e 31 *streams*. Todas as imagens possuem resolução 4k

e são armazenadas no formato *RAW*. Cada uma tem em média 25*megabytes*, e o conjunto totaliza em média 90 *gigabytes*. O tempo médio de aquisição de um objeto completo com intervalo angular de 3 graus, é de 6 a 8 horas.

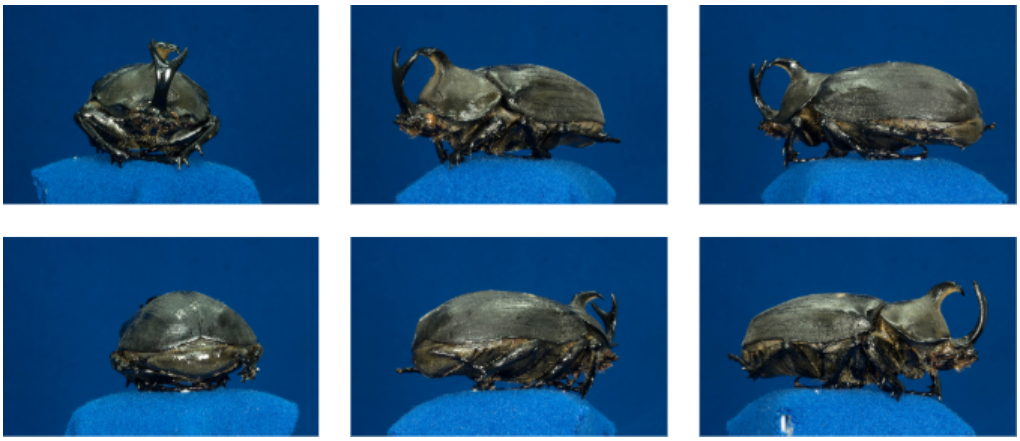


Figura 23 – Amostra de alguns frames da aquisição feita com escaravelho. Fonte: Flávio de Almeida e Silva e Diogo Olsen.

3.3.2 Pós-processamento

Após concluir a etapa de aquisição das imagens, dá-se início a etapa de pós processamento, que tem como foco principal a remoção do fundo, para que apenas o objeto de interesse seja mantido. Apesar dessa técnica ser comum em *softwares* de edição de imagem, deve-se levar em consideração que – com base nas configurações anteriormente citadas – o *pack* de imagens de um único modelo possui 3720 imagens, e por este motivo é ideal que a segmentação seja feita em lotes e de forma automática.

Para isso o processo foi realizado com o *software BipApp* (*Batch Image Processing App*) que foi desenvolvido juntamente ao *F2S2* pelo colega de laboratório, Diogo Olsen (OLSEN, 2020). O programa oferece algumas técnicas de pós-processamento, sendo elas: correção, redução de contexto, segmentação e aprimoramento. Além de aplicar estas técnicas em grande escala, o *BipApp* foi desenvolvido em conjunto ao *F2S2*, o que o torna a ferramenta ideal para realizar essa etapa com as imagens adquiridas para este projeto. A ferramenta também oferece três formas de processamento: o modo *preview*, *subset* e *full*. Neste projeto o modo *preview* e *subset* foram utilizados apenas para testes, pois executam o

processamento em determinadas imagens do *pack*. Os resultados foram obtidos a partir do modo *full*, que aplica o processamento no *pack* completo.

O *software* recebe como entrada a pasta que contém as imagens do modelo escaneado. As imagens de entrada foram fornecidas no formato *RAW*, sem passar por nenhum outro processo desde a etapa de aquisição. O primeiro passo foi aplicar a redução de contexto, que exclui as extremidades da imagem, da mesma forma que um recorte. A região a ser mantida é selecionada manualmente, a partir de uma pré-visualização de uma das imagens do lote. É importante levar em consideração que a mesma região selecionada manualmente para ser mantida será aplicada em todas as imagens, portanto é preciso fazer esta seleção de forma que, em nenhuma das imagens, parte do objeto acabe sendo removida. Outra ressalva é de que este processo não é obrigatório e em alguns casos não existe a necessidade de aplicá-lo.

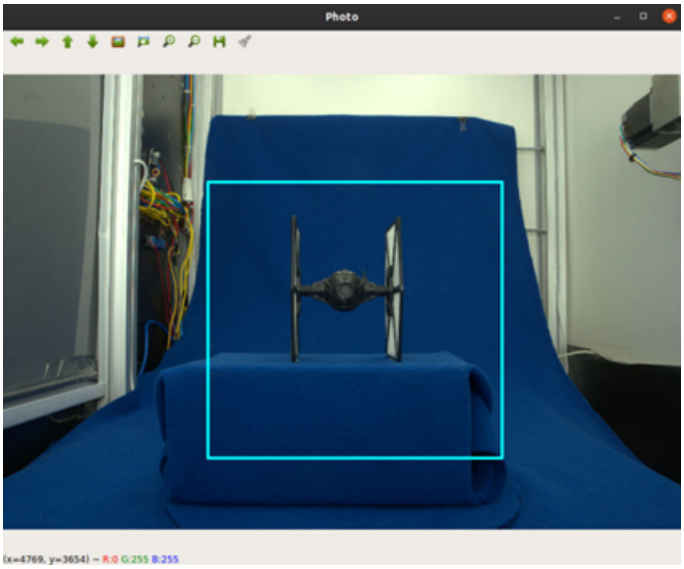


Figura 24 – Pré-visualização da imagem no software BipApp. O quadrado representa a região que será mantida. Fonte: Diogo Olsen.

O segundo passo é realizar a segmentação nas imagens. Este passo é o mais relevante, pois efetua a remoção dos *pixels* do fundo azul das imagens, substituindo-os por *pixels* pretos ou transparentes, dependendo do formato de saída que for selecionado. O *BipApp* realiza a segmentação aplicando-se um classificador *SVM*. O programa é treinado para reconhecer qual *pixel* corresponde ao fundo e qual corresponde ao objeto. É possível escolher o *kernel* que será utilizado no *SVM* entre: *linear*, *gaussiano* e *polinomial*, e as imagens podem ser processadas em *RGB*

ou *HSV* (OLSEN, 2020). O *kernel* usado na segmentação dos insetos foi o *linear* e as imagens foram processadas em *RGB*.

A quantidade de sementes para alimentar a base de imagens de treinamento é definida pelo usuário e então são selecionadas de forma aleatória, mas mantendo um intervalo considerável entre uma e outra para que diferentes ângulos do objeto possam ser classificados. Em caso de necessidade, mais imagens podem ser selecionadas manualmente para melhorar a qualidade dos resultados. O dado de entrada para definir a quantidade das imagens usadas no treinamento foram 4, tanto para o gafanhoto quanto para a borboleta. Porém, na borboleta, devido à perda parcial das antenas, mais imagens precisaram ser adicionadas.

A definição das classes correspondentes ao fundo e ao objeto é feita de forma manual em uma pré-visualização de cada uma das imagens que foram selecionadas. A classe referente ao fundo é visualizada em rosa e a classe referente ao objeto é visualizada em amarelo.

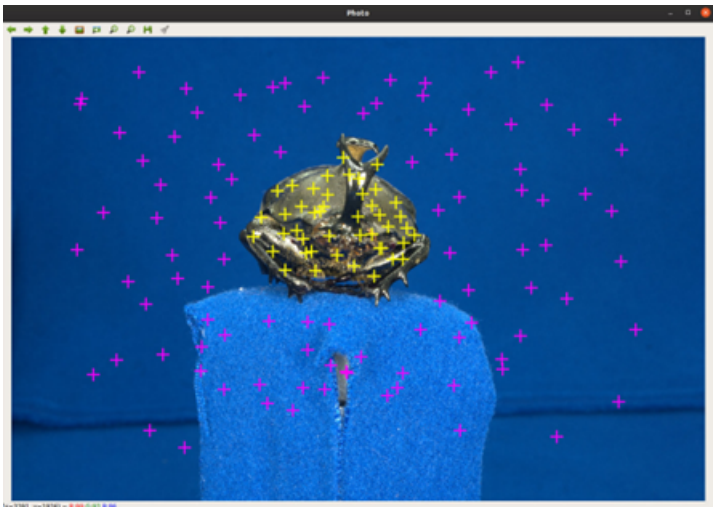
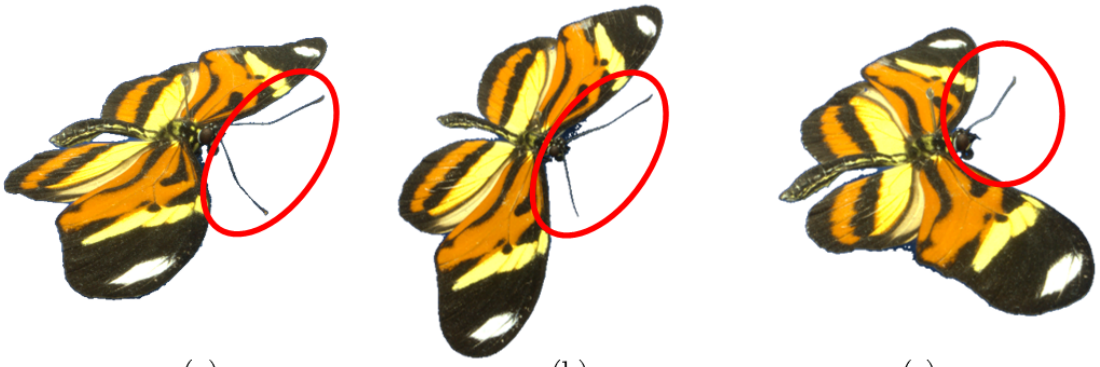


Figura 25 – Pré-visualização de classificação de regiões na imagem utilizada para alimentar a base de treinamento. Fonte: Diogo Olsen.

Para aprimorar os resultados, optou-se por aplicar as técnicas de suavização de bordas, remoção de *pixels* esparsos e erosão. De forma geral, a suavização de bordas remove a sensação de serrilhado nas extremidades do objeto, a remoção de *pixels* esparsos elimina resíduos remanescentes de fundo e a erosão remove os resquícios de borda, reduzindo uma linha de *pixel* ao redor do objeto. As técnicas foram inicialmente aplicadas em ambos os insetos. No modelo do gafanhoto,

que não possui detalhes ínfimos, todos os detalhes foram mantidos depois do processamento das imagens. Em compensação, a borboleta teve suas antenas parcialmente ou completamente removidas em algumas imagens. Portanto, como alternativa para minimizar esta perda, a erosão e a suavização de bordas foram excluídas no processamento da borboleta.



(a) (b) (c)

Figura 26 – Amostra de alguns frames da borboleta obtidas no pós-processamento. A primeira imagem mostra um exemplo de antenas que permaneceram completas (a), a segunda opção exemplifica a perda parcial de uma das antenas (b) e no último exemplo uma das antenas foi completamente excluída (c). Fonte: Própria.

O tempo de processamento varia conforme as configurações e a quantidade de imagens. O tempo total para capturar as 3720 imagens de um inseto foi, em média, seis a oito horas, dependendo do nível de detalhes de cada inseto. A remoção do fundo diminui significativamente o tamanho em *bytes* das imagens, sendo que o espaço que ocupam em memória varia dependendo da quantidade de *pixels* remanescentes. Por exemplo, uma imagem da borboleta vista de frente/lado não possui muitos *pixels*, então seu tamanho é menor, já em uma imagem da perspectiva de cima, onde é possível visualizar toda a asa da borboleta, a quantidade de *pixels* é maior e consecutivamente o seu tamanho também será maior. Ainda com relação ao tamanho de armazenamento, por mais que os dados de entrada no *BipApp* tenham sido em formato *RAW*, a saída das imagens é *JPG* ou *PNG*, que são formatos consideravelmente mais leves se comparados ao *RAW*. A fim de manter a transparência no fundo dos insetos, optou-se pelo formato de saída *PNG*. A relação de tamanhos está disposta na Tabela 2.

Devido as necessidades que serão discutidas na Seção 3.4, mais uma etapa

Inseto	Menor img.	Maior img.	Tam. total
Gafanhoto	467kb	1664kb	4.45gb
Borboleta	272kb	3245kb	6.71gb

Tabela 2 – Amostra de tamanhos (menor e maior imagem e tamanho total) resultantes no processo de pós-processamento.

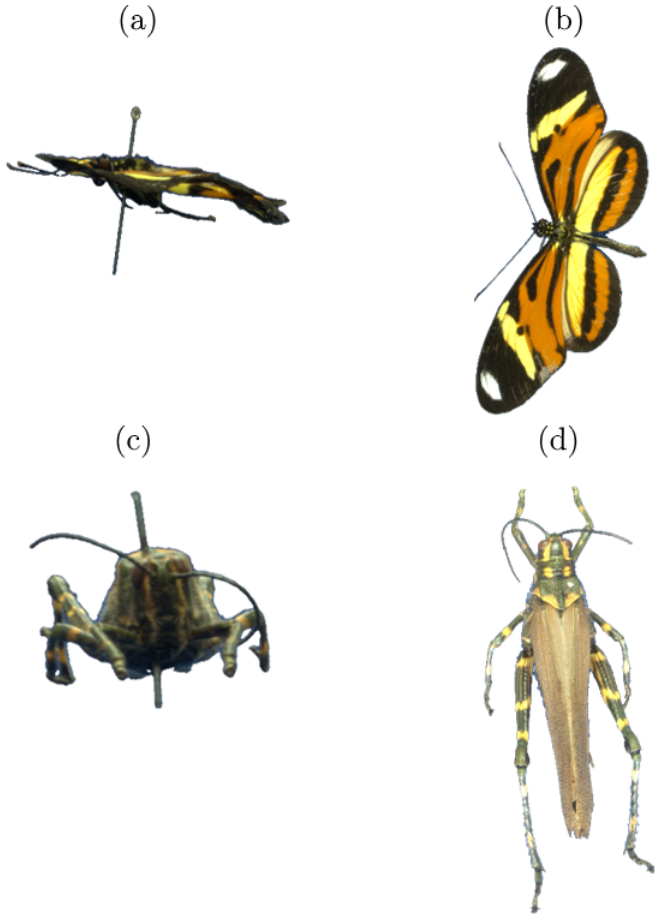


Figura 27 – Menor e maior imagem de cada inseto. Menor imagem da borboleta (a) com 272kb e maior (b) com 3.245kb. Menor imagem do gafanhoto (c) com 467kb e maior (d) com 1.664kb. Fonte: Própria.

de redução de contexto foi aplicada. No *BipApp* a seleção da região que deve ser mantida, precisa levar em consideração todas as perspectivas do objeto em cada *frame*, o que acarreta a geração de imagens com uma quantidade desnecessária de *pixels* transparentes. Portanto, foi preciso desenvolver uma solução própria que aplica a redução de contexto em todas as imagens levando em consideração

o primeiro *pixel* encontrado em cada uma das paralelas, horizontal e vertical, formando um quadrado fictício ao redor da imagem. O resultado são imagens que contêm apenas o objeto de interesse.

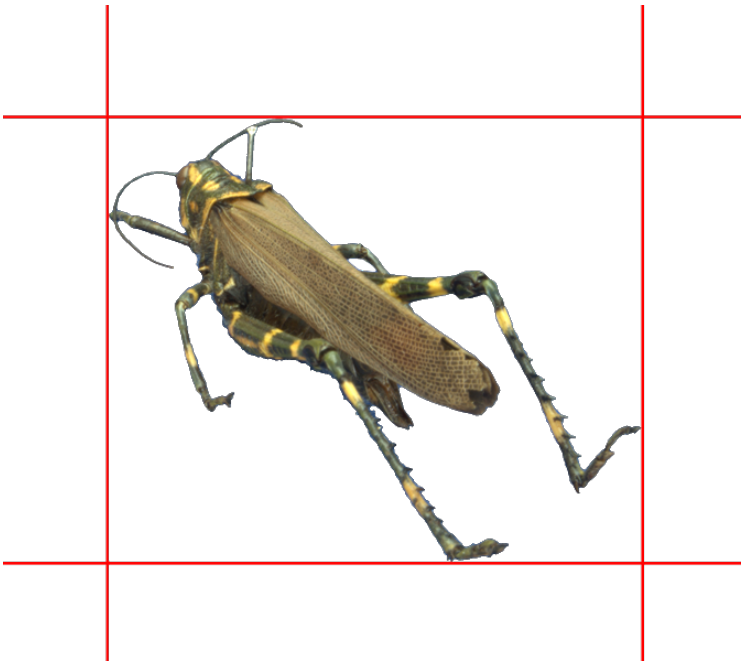


Figura 28 – Redução de contexto aplicada posteriormente ao processamento realizado no *BipApp*. Fonte: Própria.

3.4 Implementação

Esta etapa sofreu algumas mudanças durante o percurso por conta da pandemia do *Covid-19*. Portanto, a implementação da aplicação foi dividida em duas partes, o modelo volumétrico — estereoscopia — e a interface de controle foram desenvolvidos em mesa digital com projeção estereoscópica e a *CAVE* foi feita de forma simulada em *game engine*.

A aplicação para mesa digital com projeção estereoscópica foi desenvolvida em *C++* com a biblioteca *SDL* (*Simple DirectMedia Layer*). Os fatores levados em consideração para a escolha da linguagem foram: performance, plataforma, ferramentas adicionais disponíveis, popularidade e familiaridade. Além disso, por ser uma linguagem orientada a objetos, possibilita aplicar técnicas de encapsulamento, herança, abstração, polimorfismo etc. que são relevantes para o reuso de código e aplicação de boas práticas de programação.

O *SDL* é uma biblioteca multimidia que facilita lidar com gerenciamento de janelas, imagens, vídeos, áudios, *inputs* e *threads* (SUPERVISOR; JúNIOR, 2017). Sua utilização é comum no desenvolvimento de jogos, pois são aplicações que normalmente utilizam estes tipos de recursos. Foi escrita por Sam Lantinga em 1998 (HIETALA; KOPONEN, 2011). Por ser uma biblioteca com mais de 20 anos, pode ser considerada estável e durante o desenvolvimento deste projeto está na sua *versão 2.0*. A biblioteca possibilita a renderização de imagens com a *GPU*, característica fundamental para o desenvolvimento deste projeto que trabalha com grande escala de imagens.

O desenvolvimento da simulação da *CAVE* foi feito na *game engine Unity*. De acordo com o gráfico (cf. Figura 29), disponível na plataforma *Game Developer*[7], que considera a quantidade de jogos lançados por ano, a *Unity* é a *game engine* mais utilizada no mercado. É conhecida entre os desenvolvedores de jogos por possuir uma comunidade grande que disponibiliza diversos conteúdos sobre a usabilidade da ferramenta por meio de vídeos, artigos, fóruns, tutoriais etc.

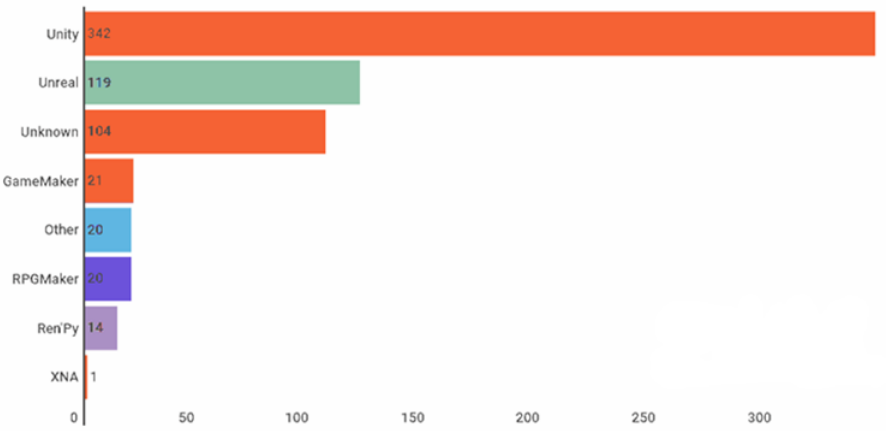


Figura 29 – Quantidade de jogos lançados por ano de cada game engine (2010 – 2021). O gráfico considera somente os jogos lançados na plataforma Steam com preço superior a $ 4.99 e pelo menos 50 avaliações. Fonte: Game Developer.

A ferramenta possui um editor com interface visual do ambiente que está sendo implementado e diversas seções de componentes prontos, entre elas: objetos 3D, objetos 2D, Efeitos, Luzes, Áudio, Vídeo, UI e Câmera. Cada uma das seções

[7] <https://www.gamedeveloper.com/business/game-engines-on-steam-the-definitive-breakdown> Acessado em: 27/12/2021

possui elementos condizentes que podem ser manipulados de diferentes formas. A *Unity* usa o *C#* como linguagem de programação e promove um design orientado a componentes. A abstração do mundo real para o código é feita da mesma forma que na POO, porém, a orientação a componentes permite que cada parte do objeto seja implementada de forma independente, ou seja, os componentes são capazes de executar suas funções independente da conexão com o objeto pai (NICOLL; KEOGH, 2019).

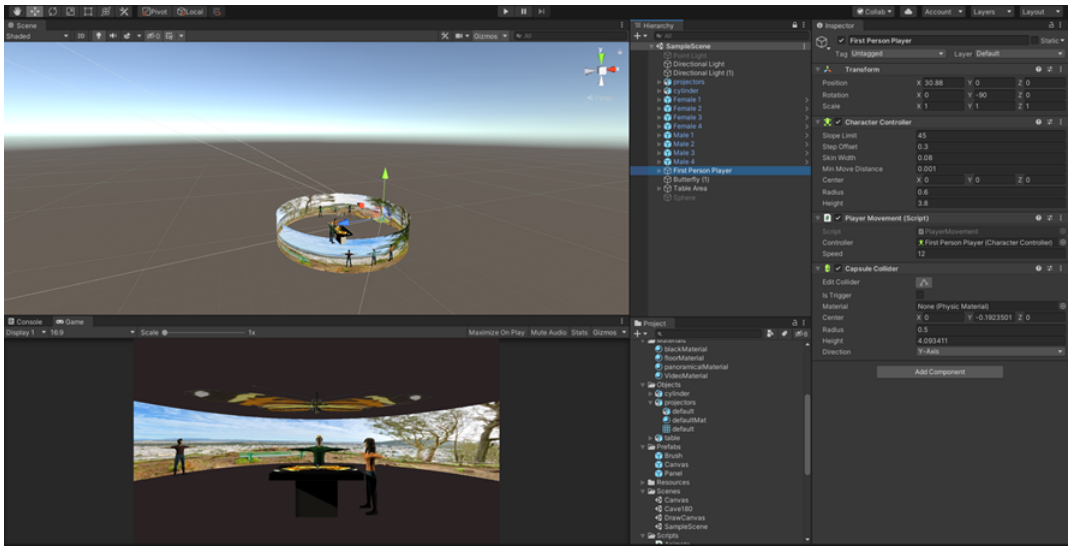


Figura 30 – Editor da Unity. Fonte: Própria.

3.4.1 ICE-S *(Immersive CAVE Environment - Simulator)*

Esta Seção apresenta o desenvolvimento de uma ferramenta que simula um ambiente imersivo em *CAVE* com tamanho e resolução dinâmicos, denominada ICE-S (*Immersive CAVE Environment – Simulator* – Ambiente *CAVE* Imersivo - Simulador). O ICE-S foi desenvolvido em *Unity* e a geração do modelo da *CAVE* é feito dentro do editor da própria *game engine*.

Os parâmetros definidos para possibilitar a criação de uma cave dinâmica são: raio, altura (altura das paredes), resolução (quantidade de faces) e grau de circunferência (360 para uma cave totalmente fechada, 180 para uma meia lua etc.). Com esses parâmetros é possível criar a geometria de uma *CAVE* em múltiplos formatos simétricos (e.g. cubo, triângulo, meia lua etc.).

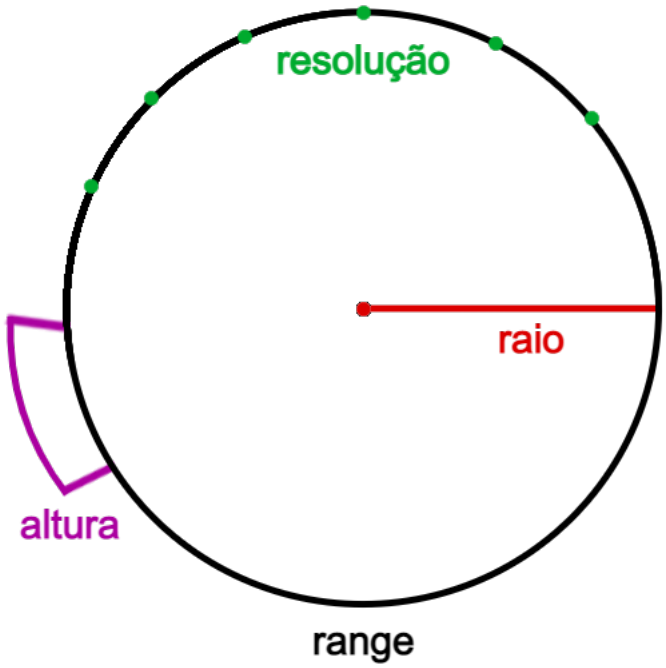


Figura 31 – Esboço do cilindro visto de cima com representação de parâmetros utilizados. Fonte: Própria.

A geração da geometria da *CAVE* é dada pela criação de uma malha de triângulos no formato desejado. Uma malha de triângulos, ou *mesh* como é mais conhecida, é muito utilizada em aplicações gráficas pois permite representar objetos tridimensionais a partir de uma coleção de vértices e polígonos.

Conforme citado anteriormente na Seção de Implementação, a *Unity* fornece uma série de componentes gráficos prontos que podem ser manipulados de acordo com a necessidade de utilização. Alguns componentes comuns utilizados na *Unity* são: *Camera*, que gera a perspectiva de visualização da cena ao ser compilada; *GameObject*, qualquer objeto que esteja instanciado na cena, assumindo diferentes formas (e.g. cubo, esfera etc.), podendo também ser vazio; *Transform*, existente em todos os *GameObjects* (inclusive os vazios) para guardar dados de posição, rotação e escala.

Para renderizar objetos tridimensionais, a *Unity* utiliza os componentes *MeshFilter* e *MeshRenderer*. O *MeshFilter* contém a informação geométrica do

objeto, ou seja, vértices e índices que compõem a malha de triângulos (*Mesh*). O *MeshRenderer* é responsável por renderizar a geometria do *MeshFilter* fazendo os processamentos visuais necessários. Criar uma malha – em termos de programação – envolve criar uma Mesh, popular sua lista de vértices, índices e *UV's*, e atribuir a *Mesh* ao *MeshFilter*.

A geometria da *CAVE* é construída em volta de um ponto central. A distância angular entre os vértices é dada pela divisão do grau de circunferência com a resolução. Para cada ângulo obtido, um par de vértices é posicionado na direção do ângulo, com distância raio a partir do centro. A posição do par de vértices é dada pela fórmula $P_(x,y,z)) = V(\cos(\alpha) * R, y, \sin(\alpha) * R)$, onde $P_(x,y,z))$ é a posição xyz do vértice, V é a representação do vetor resultante, $\cos(\alpha) * R$ é referente a posição x, y é referente a posição y e $\sin(\alpha) * R$ é referente a posição z. O valor de y é 0 para o vértice inferior, e *altura* para o vértice superior. As posições de todos os vértices são armazenadas em uma lista, onde todos os índices pares da lista correspondem aos vértices inferiores e os índices ímpares aos vértices superiores.

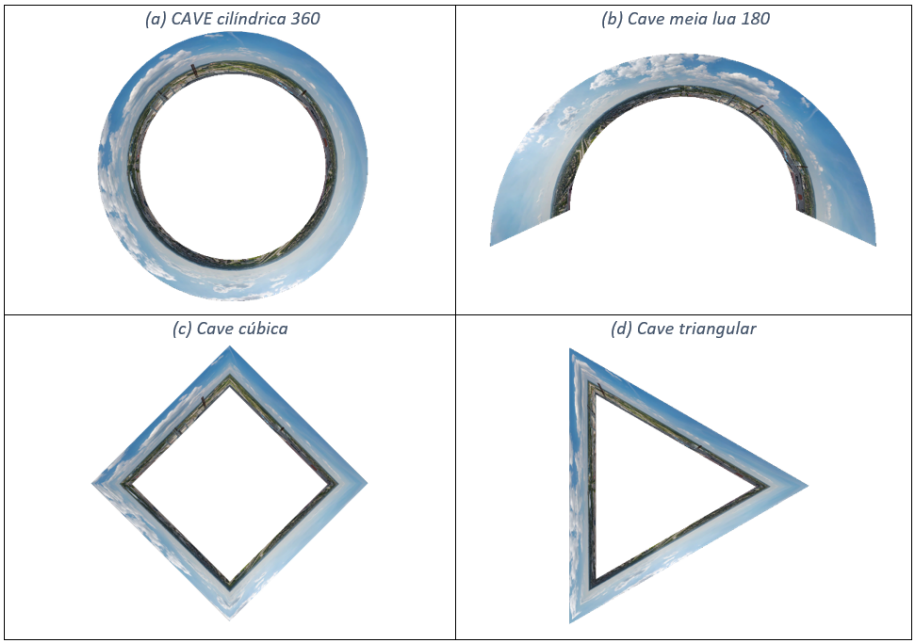


Figura 32 – Formatos de *CAVEs* de mesma altura. (a) resolução: 50, raio: 50, range: 360; (b) resolução: 30, raio: 50, range: 180; (c) resolução: 4, raio: 30, range: 360; (d) resolução: 3, raio: 40, range: 360. Fonte: Própria.

Para criar os triângulos a partir dos vértices que foram definidos deve-se informar quais índices da lista de vértices compõem cada triângulo. A forma de ordenação dos índices determina para qual direção as faces estarão dispostas, ou seja, qual será a direção das normais de cada face. Para que as normais fiquem direcionadas para dentro, os índices devem ser informados no sentido horário. Por exemplo, para obter o primeiro triângulo, os índices informados são: 0, 3 e 2; para desenhar o segundo triângulo os índices informados são: 0, 1 e 3, e assim sucessivamente. Este processo é exemplificado melhor na Figura 33.

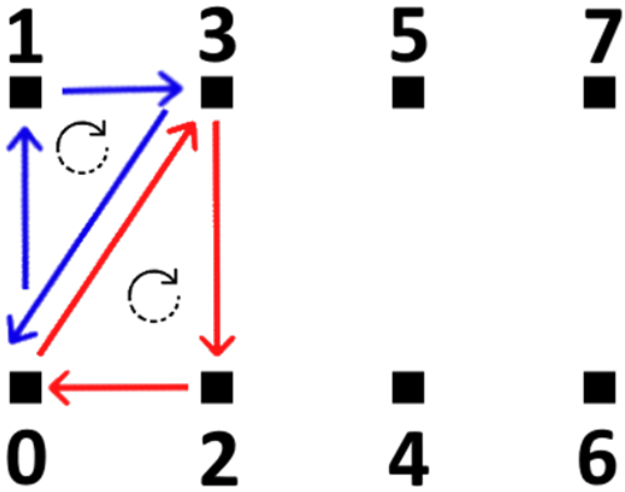


Figura 33 – Ilustração de ordenação de desenhos de triângulos. Cada ponto preto representa um vértice e os números representam o índice da lista em que foram armazenados. As setas correspondem ao sentido de ordenação dos índices no desenho de cada triângulo. Fonte: Própria.

O mapeamento de *UVs* é necessário para que texturas 2D possam ser aplicadas em um modelo 3D. O processo é feito com base nos vértices e consiste em informar ao *software* como a textura deve ser aplicada. As coordenadas *UV* ocorrem a partir do primeiro vértice armazenado na lista de vértices e são adicionadas em pares, uma para o vértice inferior e uma para o superior. O processo de mapeamento, diferente da criação dos vértices, é feito a partir dos valores normalizados de altura e largura do objeto, ou seja, ambos os valores são convertidos em uma escala entre 0 e 1. Portanto, o valor que determina o intervalo entre os pares de vértices adotados no mapeamento das *UVs* é obtido mediante o resultado da divisão de 1 pela *resolução* (quantidade de faces). A definição das coordenadas é feita iterando pela quantidade de faces – valor anteriormente utilizado para definir o tamanho da

lista de vértices – e cada ponto do par de *UVs* é definido por $U = \frac{1}{res} * i$, onde i corresponde ao índice da face atual e sendo $V = 0$ para o vértice inferior e $V = 1$ para o vértice superior.

As características que mais diferenciam a projeção em um ambiente imersivo de uma projeção comum são: a viabilidade em projetar imagens e vídeos panorâmicos e a possibilidade de projetar diferentes conteúdos simultaneamente em um único ambiente. Tanto é possível planejar o ambiente para exibir uma cirurgia em tempo real realizando a transmissão em 360 na *CAVE*, como também pode-se planejar o ambiente para dar uma aula ou palestra, onde é preciso exibir diversos conteúdos, como por exemplo, slide, imagens, vídeos etc.

Na ferramenta ICE-S desenvolvida, a simulação de projeção na *CAVE* ocorre atribuindo uma textura a malha de triângulos gerada. Para obter uma projeção de elementos estáticos, atribui-se à *CAVE* uma única textura que contenha todos estes elementos que serão exibidos. O mapeamento das *UVs* explicado anteriormente se encarrega de posicionar a textura corretamente na malha. Porém, para que o conteúdo seja projetado de forma dinâmica, ou seja, para exibir vídeos, imagens animadas ou apresentação de slides, torna-se necessário gerar uma textura dinamicamente que represente o estado mais recente da projeção. Por exemplo, para projetar um vídeo, a textura deve ser atualizada de forma que cada frame do vídeo seja exibido sucessivamente; para movimentar uma imagem ao redor da *CAVE*, é preciso gerar uma nova textura com a imagem para cada posição em que ela será renderizada.

Na *Unity*, texturas são armazenadas em instâncias de *Texture2D*, que permitem manipular as características de uma textura, incluindo os *pixels*. Para criar a textura que será renderizada na *CAVE*, uma *Texture2D* é gerada – em termos de programação – unificando os diversos elementos que serão exibidos.

Para customizar o que será exibido no simulador, deve-se especificar a imagem que será o plano de fundo da projeção, os elementos renderizadas por cima e a posição e tamanho em que cada um destes elementos será renderizado. A textura final é criada copiando primeiramente os *pixels* da imagem de fundo, depois são copiados os *pixels* dos elementos fornecidos na posição e tamanho em que foram especificados. Caso o tamanho informado seja diferente do tamanho original da imagem, duas instâncias de textura são armazenadas, uma com os valores da textura original e outra com os valores da textura redimensionada com base no

tamanho fornecido. Com isso as informações da textura original são mantidas na memória, mas não é preciso aplicar o redimensionamento da imagem toda vez que ela for renderizada. Os elementos são renderizados seguindo a ordem que foram especificados, de forma que aqueles que foram renderizados por último são os que ficarão por cima (cf. Figura 34)

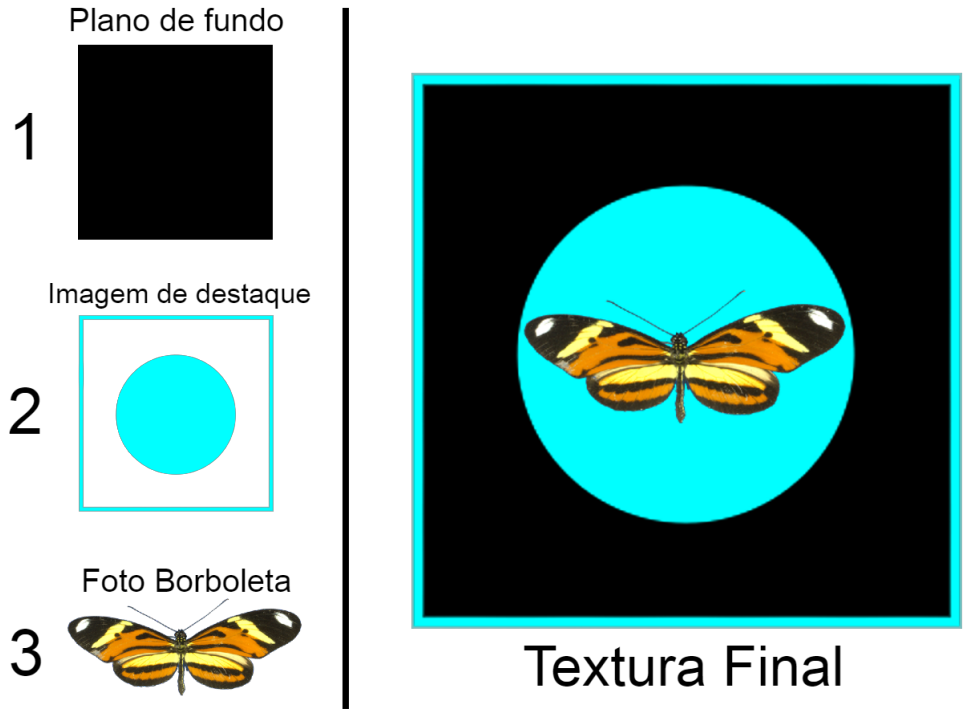


Figura 34 – Ordem de renderização das texturas. Primeira textura definida renderizada ao fundo e última textura definida renderizada à frente. Fonte: Própria.

Em uma *CAVE* cilíndrica o ideal é que as texturas possam ser renderizadas em qualquer posição. Apesar do cilindro ser construído de forma contínua em 360°, existe uma posição em que os pontos inicial e final da malha de triângulos se encontram. Para que uma textura possa ser renderizada nesta intersecção é preciso que a renderização ocorra de forma circular, ou seja, os *pixels* que ultrapassam uma das extremidades da textura final, devem ser renderizados na extremidade oposta (cf. Figura 3.25). Portanto, a posição xy de cada *pixel* é obtida a partir da fórmula $x = x\ mod\ largura$ e $y = y\ mod\ altura$, onde x e y correspondem à posição em que o *pixel* será posicionado na textura final e $largura$ e $altura$ correspondem ao tamanho da textura final. Desta forma, a imagem pode ser renderizada em

qualquer posição da *CAVE* sem gerar obstruções.

Figura 35 – Textura final renderizada na *mesh*. A textura do gafanhoto ultrapassa a extremidade da textura final e os *pixels* que seriam excluídos, passam a ser renderizados no início. Na *CAVE* as duas extremidades são conectadas de forma a se obter uma projeção 360. Fonte: Própria.

A criação de uma textura dinâmica possui um custo de processamento computacional alto, pois as texturas com movimento (e.g. vídeos, imagem com animação) são redesenhadas a todo instante. Isso causa uma queda na taxa de atualização de quadros por segundo (no inglês *frames per second* – FPS), reduzindo a frequência de produção de imagens pelo dispositivo – que geralmente variam de 30 a 60 imagens por segundo – e consequentemente, impactando negativamente na fluidez do movimento. Para reduzir o custo computacional deste processo, foi criada uma estrutura/estratégia que permite gerenciar o processo de renderização de texturas na *mesh* da *CAVE*. A estrutura é composta pelas estratégias *CaveElement* e *CaveTextureRenderer*. Cada estratégia é representada e implementada no paradigma da orientação a objetos.

Cada elemento que é renderizado na *CAVE* é representado pelo objeto *CaveElement*, que possui as informações de textura (*Texture2D*), posição (*Vector2D*), tamanho (*Vector2D*) e estado – modificado/não modificado – (*Booleano*). O *CaveTextureRenderer* é uma fábrica de objetos responsável por criar instâncias de *CaveElement* e gerenciá-las. Este processo acontece da seguinte forma: toda vez que o programa é compilado, o *CaveTextureRenderer* cria um *CaveElement* para cada imagem que será renderizada, armazena-o em uma lista e gera a textura que será inicialmente projetada. Quando um dos parâmetros (posição, tamanho ou textura) de um *CaveElement* é modificado, o estado do elemento é alterado para modificado (*true*). A cada quadro por segundo, o *CaveTextureRenderer* verifica

o estado de todas as instâncias de *CaveElements* contidas na lista, caso encontre uma ou mais instância(s) que tenha(m) sido modificada(s), a textura de projeção é gerada novamente, considerando os valores atualizados para cada objeto. Essa solução apresenta um resultado melhor na renderização de texturas com movimento, pois as texturas só são redesenhadas caso tenham sido modificadas, o que exclui a necessidade de atualizar a projeção a cada quadro por segundo. Vale ressaltar que *CaveElements* podem ser adicionados ou removidos durante a execução do programa, e estas ações também determinam quando a textura de projeção deve ser recriada.

3.4.2 Interface de Controle

A interface de controle foi especificamente pensada para a mesa digital, que foi construída pelo *CIIM* (Centro de Inovação em Imagens Médicas) juntamente ao colega de laboratório Lucas Murbach Pierin. As especificações técnicas de hardware da mesa serão discutidas posteriormente na Seção 3.5. Vale ressaltar que o protótipo desenvolvido teve inspiração na aplicação desenvolvida para o projeto de mestrado do colega de laboratório Andrei Rafael Brongel (BRONGEL, 2020). A interface foi idealizada para permitir ao usuário realizar interações com o modelo digitalizado, de forma que simulasse um objeto 3D a partir de imagens bidimensionais e para que fosse possível realizar a troca do modelo de exibição. Com essas características em mente, a proposta inicial da mesa teve seu design conforme exibido na Figura 36.

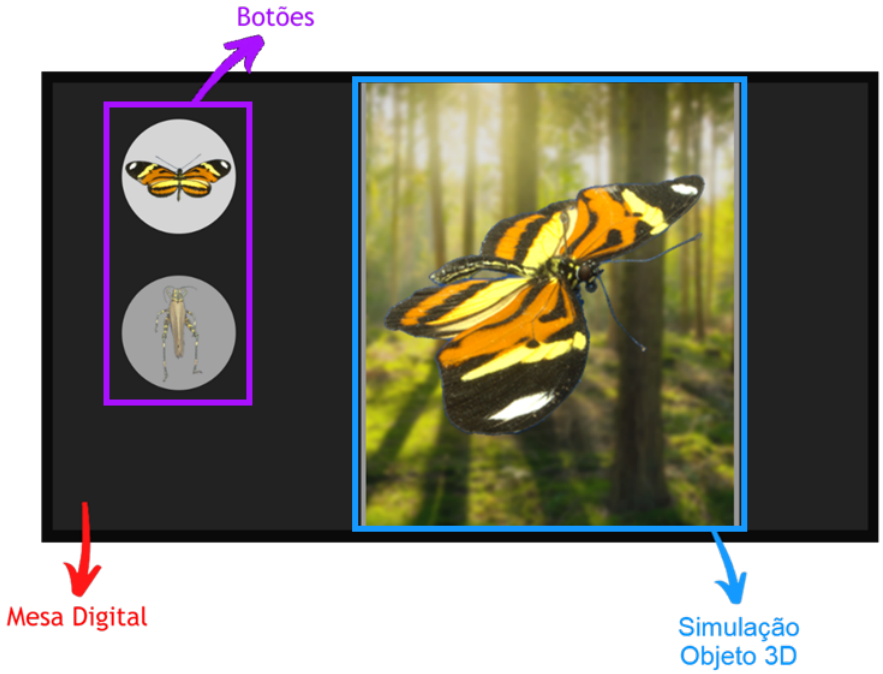


Figura 36 – Esboço da primeira versão da interface de controle em mesa digital. Fonte: Própria.

O desenvolvimento teve seu início na simulação 3D das imagens. De forma a obter todas as imagens em uma estrutura condizente à visualização proporcionada pelo scanner *F2S2*, optou-se por realizar o armazenamento em uma matriz de streams $Stream_(i, j)$. Essa estrutura foi adotada visando a simulação do objeto 3D que necessita alternar entre os *frames* adjacentes conforme o dado de entrada que for fornecido pelo usuário.

O carregamento das imagens pode ser feito de diversas formas, algumas abordagens cogitadas foram: carregamento em janelas deslizantes, carregamento único e carregamento completo. A primeira alternativa foi implementada, em um caso semelhante, no projeto de doutorado do colega Flávio de Almeida e Silva, oportunizando um carregamento com mais de 32 mil imagens ou em computadores que não possuam capacidade para armazenar todas as imagens em memória (SILVA, 2020). Com esse mesmo problema em mente, a primeira abordagem implementada foi o carregamento único, que recebia como entrada o caminho da pasta com todas as imagens e realizava o carregamento e renderização de uma única imagem por

vez. Por exemplo, se uma imagem (IMG0) estiver sendo exibida e logo em seguida um input for recebido solicitando que a próxima imagem (IMG1) seja renderizada, a imagem (IMG1) é carregada e renderizada e a imagem (IMG0) é excluída da memória. Fica evidente que essa abordagem, se tratando de custo de memória, é mais eficiente. Porém, como o programa realiza o carregamento e renderização em tempo real, quando ocorre a troca de uma imagem para outra é possível observar um *delay*, o que causa a sensação de um programa com travamentos, impactando negativamente na fluidez da aplicação.

Levando em consideração que o *hardware* da mesa digital possui 2 *SSDs*, um com 1 *terabyte* e outro com 256 *gigabytes* de memória computacional para armazenamento e visto que a interface foi projetada com um propósito fixo, o de renderizar 3720 imagens de insetos adquiridos com o *F2S2*, assumiu-se a que não é necessário realizar otimização com relação ao armazenamento do lote de imagens. Portanto, o carregamento foi realizado de forma completa, armazenando em memória as 3720 imagens simultaneamente.

Para obter um carregamento eficiente foi utilizado o recurso *future* da biblioteca padrão do *C++*, o *std*. Esse recurso autogerencia *multithreads* e executa funcionalidades de forma assíncrona. Dessa forma, foi possível carregar várias imagens simultaneamente, resultando em um carregamento rápido e de fácil implementação. O tempo de carregamento que levava em média cinco minutos sem a utilização deste recurso, passou a ser feito em menos de um minuto.

O *SDL* oferece duas estruturas para armazenar as informações das imagens, o *SDL_Surface* e o *SDL_Texture*. Ambas possuem a mesma finalidade, a diferença entre elas é que o *SDL_Surface* utiliza a memória *RAM* da *CPU* para armazenar os dados da imagem, enquanto o *SDL_Texture* armazena as informações em *VRAM* (memória *RAM* do dispositivo gráfico) e o processo de renderização é acelerado pela *GPU*. Devido ao fato de o programa trabalhar com a renderização de múltiplas imagens em tempo real, é imprescindível que o modelo adotado seja o *SDL_Texture*.

Cada textura (*SDL_Texture*) que é renderizada precisa estar atribuída à um renderizador específico, por exemplo, na Figura 36 mostrada anteriormente, a seção em azul denominada “Simulação Objeto 3D” é uma área de renderização, toda vez que uma imagem for exibida na tela da mesa, sua textura deve estar atribuída a este renderizador específico. O *SDL* não permite que uma mesma textura seja atribuída a mais de um renderizador. Isso adiciona um nível de complexidade

quando levado em consideração que a mesma imagem que está sendo renderizada na mesa digital, deve ser renderizada na projeção estereoscópica.

Em uma primeira abordagem, foi considerado realizar o carregamento de um *array* bidimensional com todas as imagens para cada um dos renderizadores. No entanto, isso implicaria em um consumo dobrado de memória, impactando negativamente no funcionamento do programa/estratégia. Para fazer com que as imagens fossem carregadas apenas uma vez em memória, ao invés de armazenar as imagens em uma estrutura *SDL_Texture*, os dados foram armazenados no formato *SDL_Surface*. Desta forma, toda vez que uma imagem for renderizada, duas texturas (*SDL_Texture*) são criadas a partir da mesma *surface* com o método *SDL_CreateTextureFromSurface*, que permite criar texturas a partir de uma estrutura do tipo *surface*. A mesma lógica é aplicada no par estéreo na projeção estereoscópica; essa questão será mais aprofundada na Seção Estereoscopia.

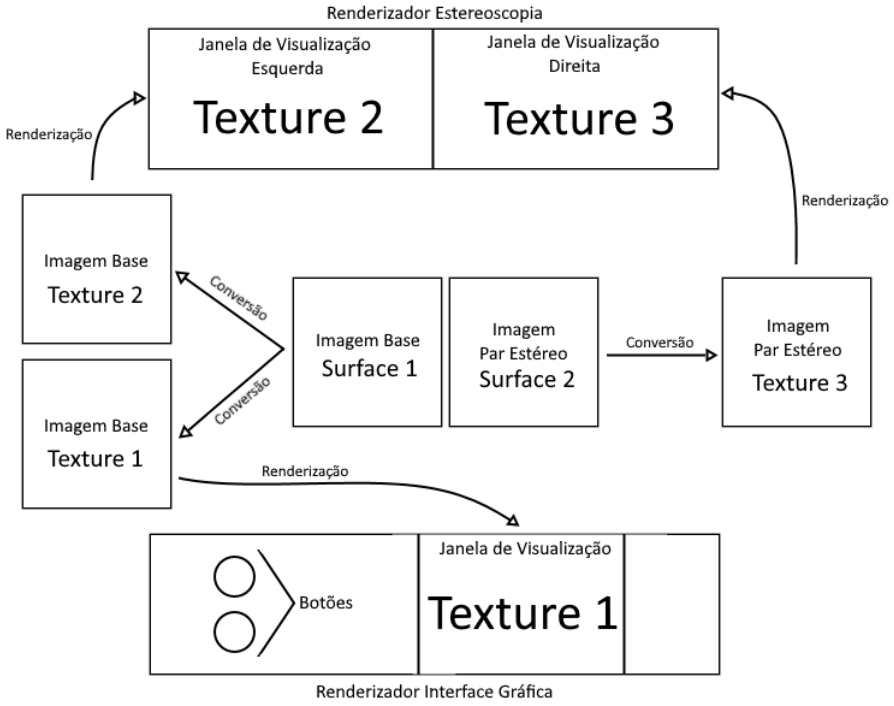


Figura 37 – Fluxo de armazenamento, conversão e renderização de imagens no programa. Fonte: Própria.

A mesa digital possui tela com sensor de movimentos *touch* e por isso todos os eventos de *input* responsáveis por aplicar as transformações geográficas no modelo 3D simulado foram implementados por comandos *touch* e foram definidos

da seguinte maneira: movimentação de um único dedo na horizontal e vertical rotaciona o objeto, movimentação de dois dedos na horizontal e vertical translada o objeto e o movimento de pinça escala o objeto (*zoom in zoom out*).

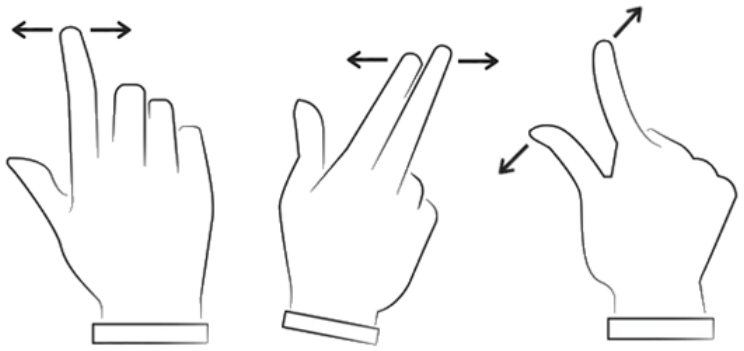


Figura 38 – Ilustração da movimentação touch utilizada no projeto. Da esquerda para a direita, rotação, translação e escala. Fonte: Colourbox por BigP com edição própria.

Alguns padrões de projeto foram aplicados para processar os eventos de entrada com o intuito de simplificar a implementação e usabilidade no código. Os padrões foram:

Command: Utilizado para encapsular todas as funcionalidades de transformações geográficas no modelo, fazendo com que o processamento de cada evento possa ser alterado sem que cada funcionalidade precise ser reimplementada.

Singleton: Como a aplicação possui mais de uma janela, foi criado um *Singleton* para organizar e gerenciar o sistema de eventos de *inputs* excluindo a necessidade de se criar uma instância para cada uma das janelas do programa.

Observer: Utilizado em conjunto com o padrão *Command* para acionar todas as classes interessadas em determinado comando.

A transformação geométrica mais simples de ser implementada é a escala. Tendo o usuário realizado o movimento de pinça distanciando os dedos (movimento de abertura), o tamanho da imagem é escalado em 1.1, se caso o movimento for de aproximação dos dedos (movimento de fechamento) o tamanho da imagem é escalado em 0.9.

A translação da imagem é feita com base na distância percorrida do *input* do usuário. A posição inicial do *input* é armazenada em um vetor bidimensional $P_1(x, y)$ e a posição final é armazenada em um vetor bidimensional $P_2(x, y)$. O

cálculo de distância é dado pela fórmula $d(x, y) = P_2(x, y) - P_1(x, y)$. A posição do objeto é modificada, incrementando o valor do vetor de distância à sua posição atual toda vez que o usuário atualiza o movimento dos dedos na tela até que o *input* seja finalizado.

A rotação do objeto é a que possui maior complexidade dentre as transformações geométricas, uma vez que precisa ser simulada pois não se trata de um modelo tridimensional. Para realizar a operação a partir dos graus de rotação, foi implementada a estratégia *getImageFromAngle*, que dado um determinado vetor angular xy retorna a imagem correspondente ao ângulo na matriz $Stream2D_(i, j))$. A fórmula que realiza essa conversão é aplicada em ambas as dimensões (i, j) da matriz e se dá por $I = N * \frac{\alpha}{R}$, onde I corresponde ao índice, N é o tamanho da lista, α é o ângulo fornecido e R é o raio de aquisição (*frames = 360° e streams = 90°*).

3.4.3 Estereoscopia

A projeção do modelo volumétrico com estereoscopia foi baseada nos princípios teóricos descritos na tese do colega de laboratório Flávio de Almeida e Silva (SILVA, 2020).

A geração de modelos volumétricos pode ser obtida de diferentes maneiras e em diferentes dispositivos conforme citado anteriormente na Seção 2.1. Levando em consideração que a visualização do modelo volumétrico neste projeto é feita através de óculos polarizados, a técnica utilizada na construção da estereoscopia foi feita com pares estéreo. Um par estéreo é obtido a partir de duas imagens de um mesmo objeto que possuam uma disparidade angular entre elas. Segundo estudos realizados por Silva, a diferença angular entre um par estéreo deve estar entre 1° a 6°, ultrapassando este limite surge a necessidade de realizar correções da geometria epipolar para que a visualização não cause desconforto (SILVA, 2020).

Um dos motivos da aquisição das imagens com o *F2S2* ter sido realizada com diferença angular de 3° foi possibilitar a obtenção de um par estéreo com o menor número possível de imagens, sem obter perdas de fluidez na movimentação do modelo em sua visualização tridimensional. Desta forma a rotação do objeto ocorre de forma fluída e é possível obter um par estéreo que gere uma visualização estereoscópica confortável ao olho humano.

O par estéreo é obtido a partir das imagens armazenadas em memória no $Stream2D_(i, j)$ e é composto por uma imagem chamada de *imagem base* e a imagem *par correspondente*. A *imagem base* é sempre a mesma imagem que está sendo exibida na interface de controle da mesa digital e seu *par* é obtido pela fórmula $P_i = i + \frac{G}{A}$. Onde P_i corresponde ao índice da *imagem par*, i é o índice da *imagem base*, G é a diferença angular desejada e A é a diferença angular de aquisição. Por exemplo, se a *imagem base* for a do índice 0, a diferença angular desejada for 6 e o grau de aquisição for 3, a *imagem par* será a do índice dois (Figura 39 (a)). Se a *imagem base* for a do índice 1, então a *imagem par* será o índice 3 (cf. Figura 39 (b)), e assim sucessivamente.

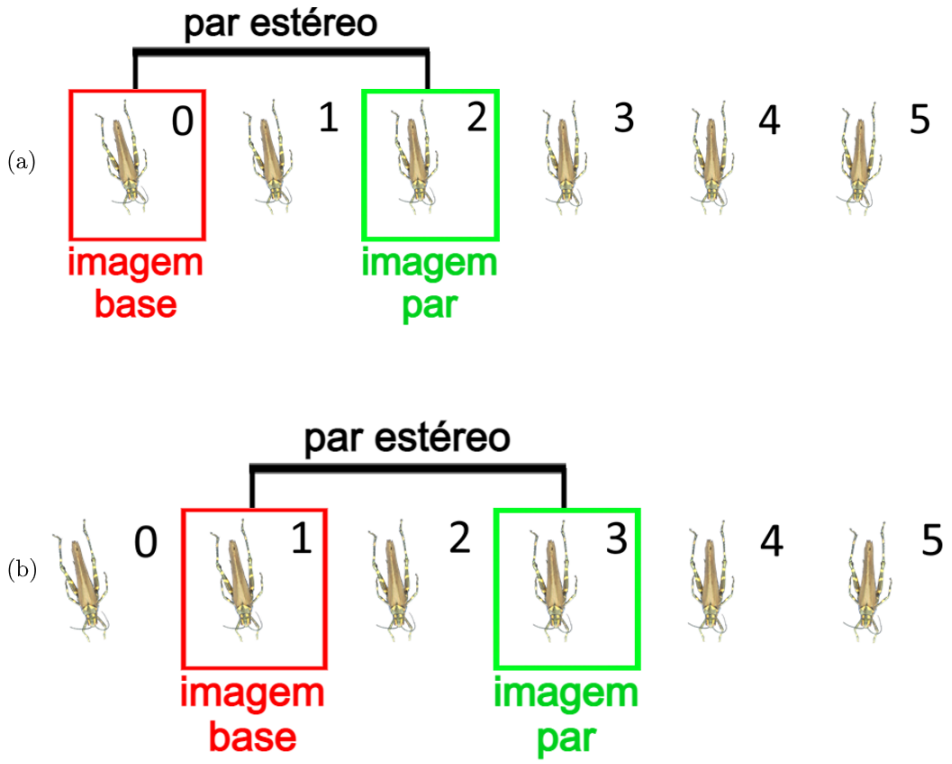


Figura 39 – Exemplo de aplicação da fórmula de obtenção do par estéreo no eixo horizontal na matriz $Stream2D_(i, j)$. Fonte: Própria.

A renderização das imagens foi feita em uma segunda janela a partir da mesa digital, ou seja, a aplicação foi desenvolvida para funcionar em duas janelas, uma com a interface de controle na mesa digital e outra com a projeção do modelo volumétrico no projetor, que foi conectado à mesa.

Para que a visualização de projeção estéreo seja obtida é preciso renderizar

duas imagem de forma que cada uma seja enxergada por apenas um olho. Isso pode ser criado a partir do deslocamento entre as imagens, chamado de paralaxe. Existem três formas de definição da paralaxe com relação ao *display*: negativo (em frente a tela), zero (no plano da tela) e positivo (atrás da tela) (WOLDEGIORGIS; LIN; LIANG, 2018). A paralaxe zero é praticamente bidimensional, na medida que o ponto de encontro de visão dos dois olhos ocorre na tela. A paralaxe positiva é obtida a partir de uma única imagem duplicada que quando sobreposta por um dispositivo estereoscópico (e.g. óculos de realidade virtual) gera uma pequena distorção causando a sensação de profundidade (SILVA, 2020). Como a interseção da linha de visão ocorre atrás da tela, a profundidade é obtida dentro do plano de projeção. A técnica para aplicar a paralaxe negativa é semelhante à positiva, mas ao invés de duplicar a mesma imagem, a visualização é obtida a partir de imagens que possuam entre si um deslocamento de aproximadamente 6.5cm (distância média de separação entre os olhos) (SILVA, 2020). Isso faz com que a imagem da direita seja vista pelo olho esquerdo e vice-versa, tendo seu ponto de interseção em frente ao plano de projeção, ocasionando na visualização de um objeto flutuante.

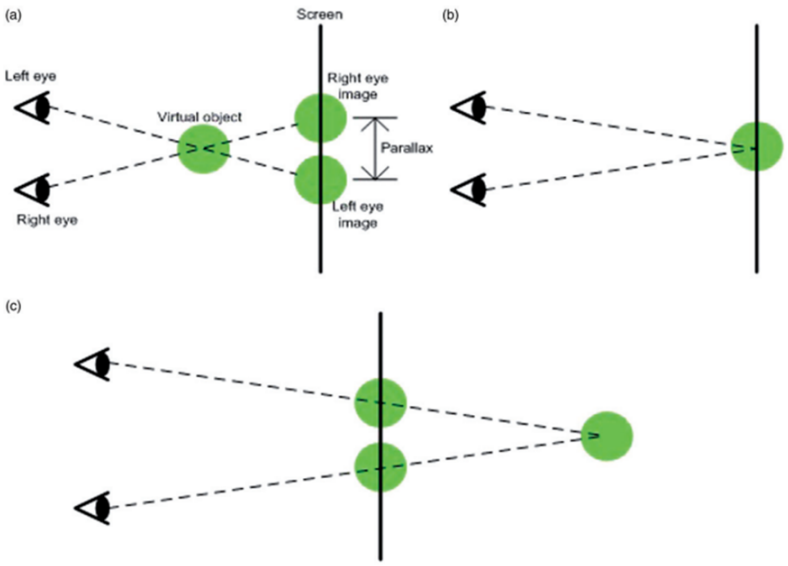


Figura 40 – Ilustração de paralaxe. (a) Paralaxe negativa, onde as linhas de visão se cruzam em frente a tela; (b) paralaxe zero, interseção é feita na tela; (c) paralaxe positiva, interseção ocorre atrás da tela. (Baseado em (Kim, G. 2005) e (Seigle, D. C. et al 2009). Fonte: (Eunhee, C. et al 2020).

O modelo que se encaixa nas premissas deste trabalho é o de paralaxe

negativa, pois o interesse é que se obtenha um modelo flutuante do objeto real escaneado. A obtenção das imagens que se enquadram neste modelo foi obtida a partir do par estéreo que possui o deslocamento em graus entre as imagens, conforme dito anteriormente. Desta forma, a implementação da janela de projeção pode ser visualizada na Figura 41.

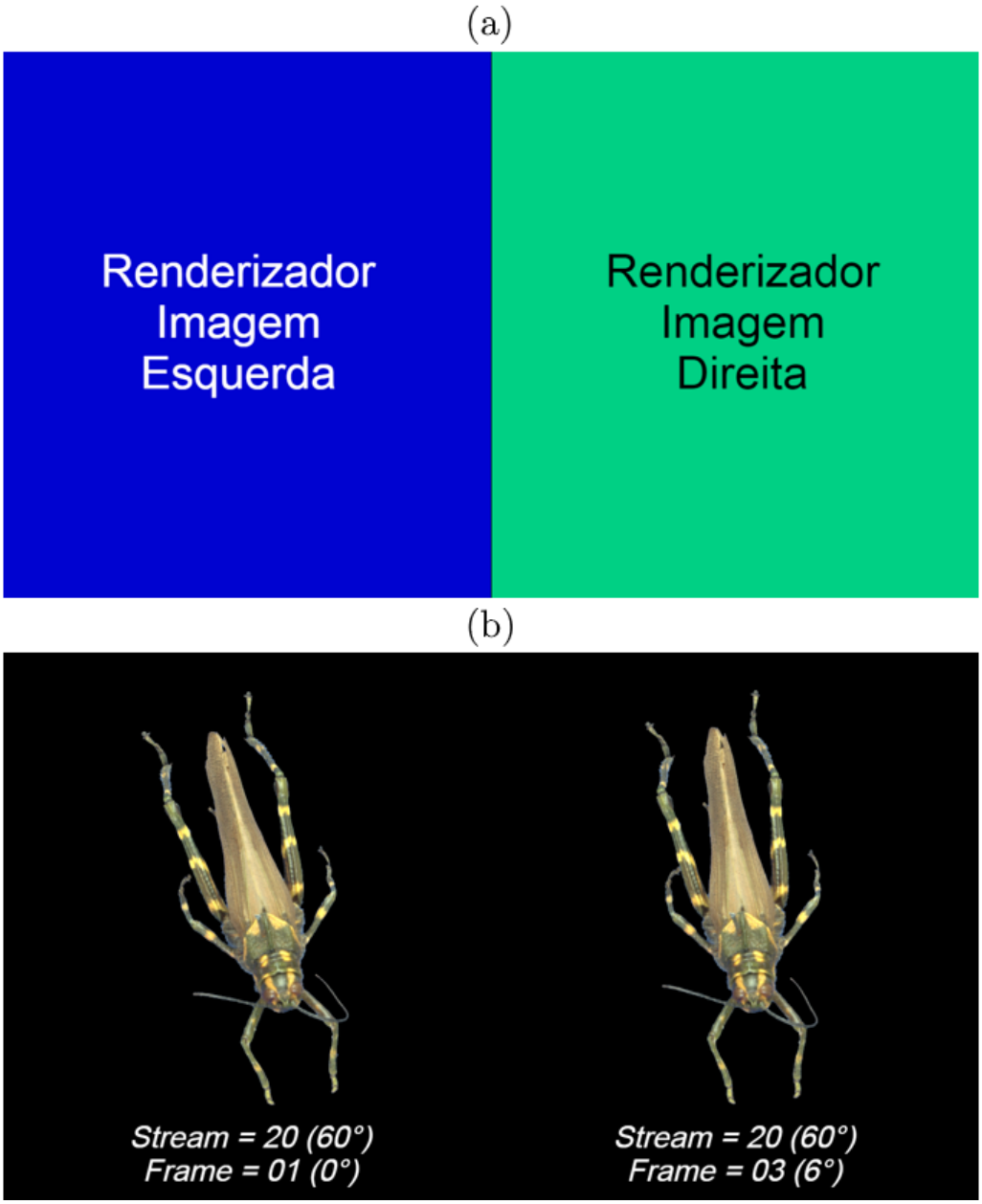


Figura 41 – Janela de projeção estereoscópica. (a) Esboço da disposição dos renderizadores, (b) disposição do par estéreo do gafanhoto. Fonte: Própria.

A mistura das imagens que resulta na projeção estereoscópica foi feita pelo próprio projetor *Optoma HD27HDR*, excluindo a necessidade de se realizar os cálculos para esta finalidade.

3.5 Ferramentas Utilizadas

A aquisição das imagens foi realizada com o *Full Frames Semi-Spherical Scanner*, e o processamento foi feito com o *BipApp* (*Batch Image Processing App*), ambos projetados e desenvolvidos pelo colega de laboratório Diogo Olsen.

O desenvolvimento foi feito na *IDE Visual Studio*, da *Microsoft*, utilizando a linguagem de programação *C++*, com o auxílio da biblioteca multimídia *SDL*. As ferramentas de auxílio para renomear e cortar arquivos em grande escala, foram desenvolvidas na *IDE Visual Studio Code* com linguagem de programação *C#*.

A mesa digital possui *hardware* com processador *Intel Core i9*, 128 *gigabytes* de memória *RAM*, um *SSD* de 1 *terabyte* mais 1 *SSD* de 256 *gigabytes* e placa de vídeo *NVidia GTX1080TI*. O projetor utilizado foi o *Optoma HD27HDR*, *Full HD*, 3D, com entrada *HDMI 2.0* com conectividade *HDCP 2.2* para fontes de entrada *4K HDR*.

A implementação do ICE-S (*Immersive CAVE Environment – Simulator*) foi feito na *game engine Unity v. 2019.4.18f1* no *Notebook Gamer Dell G3 15* com processador *Intel Core i7*, memória de armazenamento *SSD* de 500 *gigabytes*, memória de processamento *RAM* de 16 *gigabytes* e placa de vídeo *NVidia Geforce RTX2060*.

3.6 Considerações Finais

Este capítulo apresentou as etapas do método proposto bem como as particularidades de cada um dos processos que compõem este trabalho. Na Subseção 3.3 foram destacados todos os pontos relevantes no processo de aquisição dos insetos, levantando os principais pontos para obter imagens com boa acuidade visual levando em conta as particularidades de realizar a aquisição de imagens de insetos com o *F2S2*. Além disso, foram apresentadas todas as técnicas de processamento aplicadas nos insetos considerando as peculiaridades de cada um dos modelos escaneados. Na Subseção 3.4 foi comentado sobre as etapas de implementação realizadas, mostrando

as soluções que foram propostas para cada etapa e o processo de criação do ICE-S e as ferramentas adicionais que podem ser usadas em um ambiente imersivo. Por fim, a Subseção 3.5 apresenta as ferramentas que foram utilizadas no desenvolvimento das aplicações e especificações de *hardware* que se fizeram necessárias.

O próximo capítulo apresenta os resultados obtidos e será feita uma discussão sobre os principais desafios durante o desenvolvimento e as limitações do projetou.

4 Resultados e Discussões

Neste capítulo são apresentados os resultados obtidos e uma discussão sobre os desafios e limitações de cada uma das etapas. A Seção 4.1 discorre sobre os resultados da ferramenta ICE-S e alguns exemplos de modelos que podem ser gerados com o simulador. Na Seção 4.2 são discutidos os resultados com relação a base de imagens e as lições aprendidas durante a aquisição e processamento das imagens. A Seção 4.3 apresenta a aplicação da interface de controle da mesa digital com estereoscopia e as barreiras encontradas durante o desenvolvimento. Ao final, na Seção 4.4 são realizadas as considerações finais sobre o atual capítulo.

4.1 Análise da ferramenta ICE-S e dos modelos de CAVE gerados pelo simulador

O Simulador ICE-S foi projetado para funcionar dentro da própria *Unity*, de forma que as ferramentas proporcionadas pela *game engine* pudessem ser aproveitadas na construção dos ambientes imersivos e personalização dos elementos que são renderizados na *CAVE*. Portanto, é imprescindível que a *Unity* esteja instalada para que o simulador possa ser utilizado.

Outra abordagem seria a criação de um programa a parte que utiliza a *Unity* apenas no processo de desenvolvimento, de modo que o simulador não dependa da *engine* no seu funcionamento. Neste modelo, o sistema seria mais simples, com algumas configurações pré-estabelecidas e pouco espaço/necessidade para customização dos elementos de exibição na *CAVE*.

Uma das características que torna possível a criação de ferramentas acopladas à *Unity*, é a possibilidade de personalizar o editor via código, além da criação de parâmetros públicos que podem ter seus valores definidos no editor. Por exemplo, no ICE-S os parâmetros que são utilizados no código são fornecidos por *input* de texto no editor. A *CAVE* é gerada com os valores fornecidos ao invocar a funcionalidade *Generate*. Normalmente seria preciso executar a cena para visualizar a *CAVE* gerada, mas por conta dessa flexibilidade disponibilizada pela *Unity*, é possível implementar funcionalidades como esta.

Na Figura 42 são destacadas as diferentes partes que compõem a *game engine* e que são relevantes para a usabilidade do ICE-S, são elas: *Scene* (em vermelho), onde a *CAVE* é exibida; *Game* (em roxo), visualização da cena na perspectiva da câmera, o que será exibido ao ser executado; *Hierarchy* (em verde), ordem hierárquica dos componentes instanciados na cena; *Project* (em azul), organização dos arquivos que estão sendo utilizados no projeto (e.g. imagens, vídeos, áudios, etc.); *Inspector* (em amarelo), exibe todos os componentes contidos em determinado objeto.

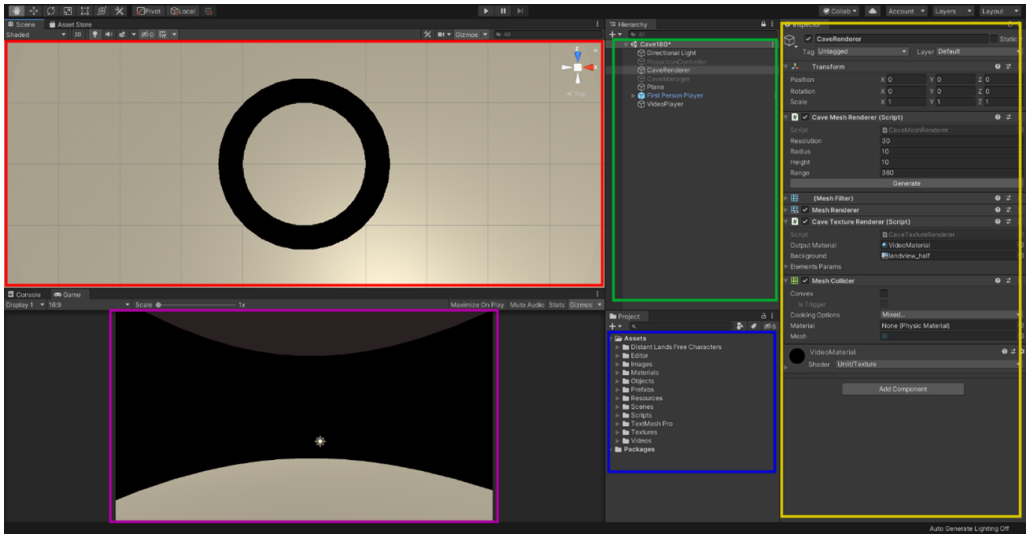


Figura 42 – Visualização seccionada do ICE-S no editor da Unity. Fonte: Própria.

Para visualizar no editor as variáveis que foram definidas como públicas no código, é preciso selecionar o componente que contém o *script* em que as variáveis foram declaradas. Com o componente selecionado, os parâmetros públicos são exibidos no *Inspector*. No caso do ICE-S, o componente que é utilizado para gerar uma *CAVE* e os elementos que são projetados nela é o *CaveRenderer*. A visualização do *Inspector* do componente é exibida na Figura 43.

O *script CaveMeshRenderer* é o responsável por criar a *CAVE* de forma dinâmica, com base nos valores definidos nos *inputs* exibidos editor. O *script CaveTextureRenderer* renderiza uma imagem estática como plano de fundo e outros elementos estáticos que forem fornecidos por meio do editor.

Para a realização de testes de projeções com movimento na *CAVE*, foram implementados dois modelos de exemplo: animação de rotação 3D a partir de imagens bidimensionais e apresentação de slides. Os parâmetros utilizados no

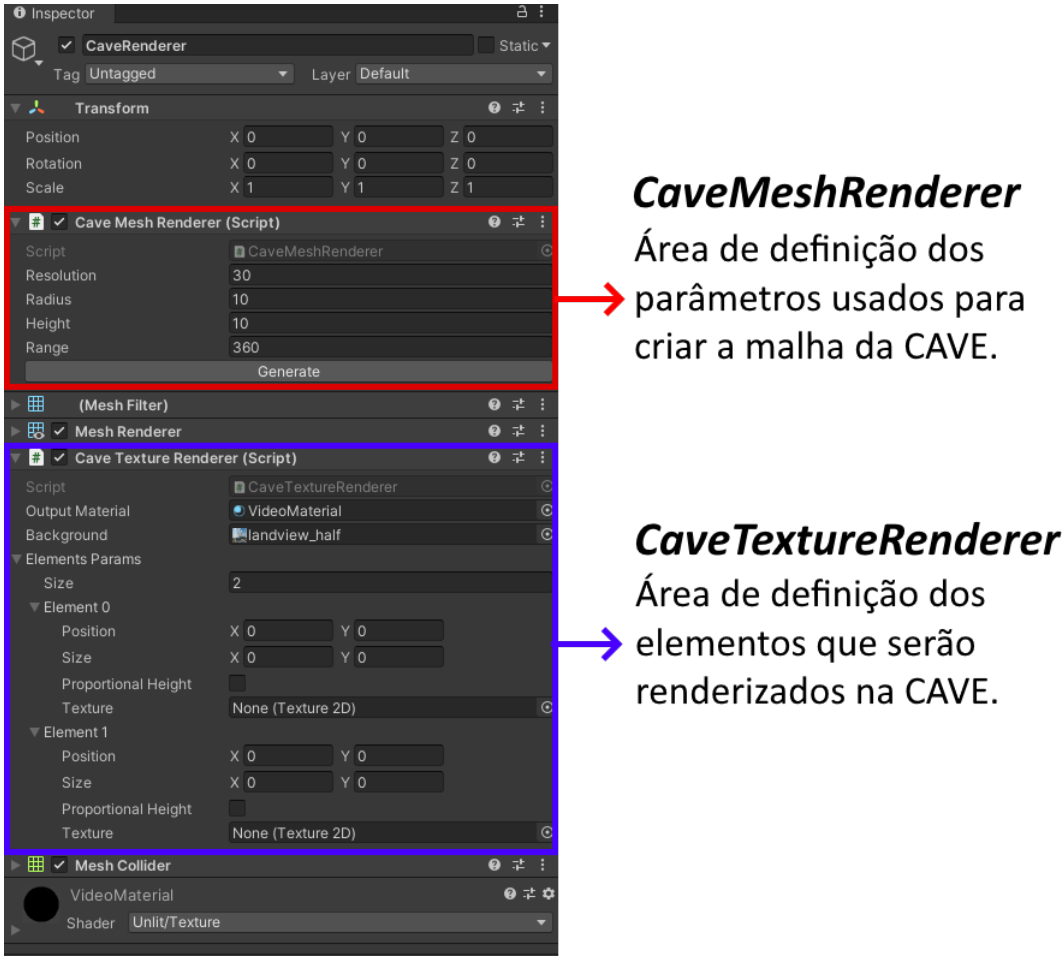


Figura 43 – Inspetor exibido ao ter o componente CaveRenderer em foco. A seção em vermelho corresponde aos parâmetros de criação da CAVE dinâmica e a seção azul aos parâmetros de renderização. Fonte: Própria.

código são exibidos no editor conforme ilustrado na Figura 43. Vale ressaltar que outros modelos poderiam ser implementados neste mesmo formato (e.g. animação de escala em imagens, animação de posição em textos, partículas etc.).

Conforme explicado anteriormente na Seção 3.4.3, o ICE-S é capaz de gerar diversos formatos de *CAVE*. A proposta inicial do simulador era gerar apenas modelos de *CAVE* em formato cilíndrico, seja completa (360) ou meia lua (180). Porém, no decorrer do desenvolvimento notou-se a possibilidade de gerar modelos em forma de cubo e triângulo. Como estes modelos não haviam sido previstos alguns ajustes são necessários para aplicar as texturas nestes modelos. Por exemplo, ao renderizar uma textura 360 em uma *CAVE* cúbica, de forma a obter uma

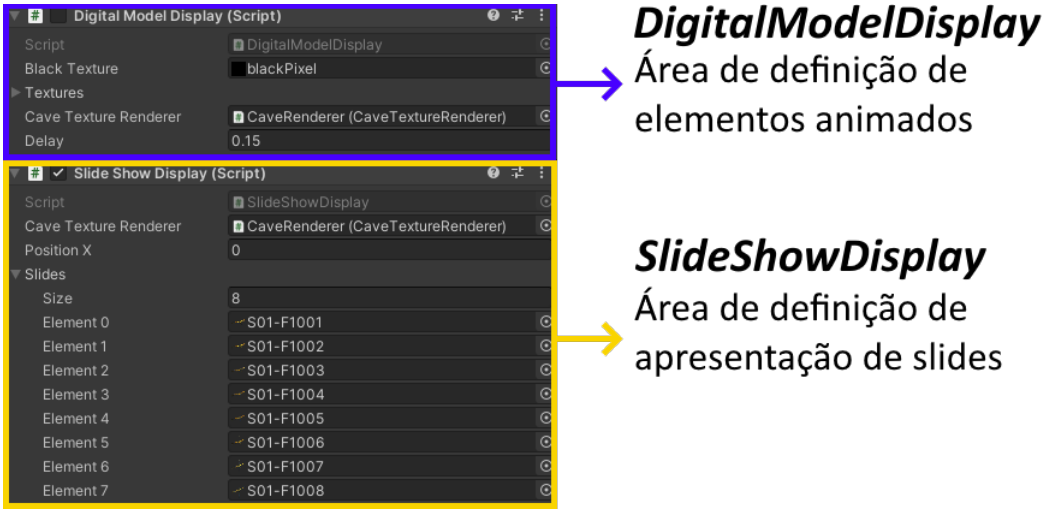


Figura 44 – Visualização no editor dos parâmetros utilizados para renderizar elementos com movimento. Fonte: Própria.

visualização contínua da imagem, faz-se necessária a aplicação de cálculos de correção da textura nos pontos de encontro das faces do cubo. O mesmo conceito se aplica para outras imagens que sejam renderizadas na mesma região.

Nas figuras abaixo são exibidas algumas *CAVEs* que foram geradas com o ICE-S para ilustrar a capacidade da ferramenta. Para cada modelo de exemplo, foi gravado um vídeo mostrando todo o entorno da *CAVE*. A Figura 45[8,9] mostra dois modelos de *CAVE* 360 com imagem estática de plano de fundo panorâmica 360°. Na Figura 46[10,11] são apresentados dois modelos de *CAVE* 360 com projeção de vídeo 360°. Por fim, a Figura 47[12,13] ilustra a utilização de elementos com movimento em uma *CAVE* 360 e em uma meia *CAVE* 180.

[8] <https://youtu.be/S5NqTXPWRWk>
[9] <https://youtu.be/1zafp4lKRLU>
[10] <https://youtu.be/YjiC25VttwM>
[11] <https://youtu.be/MSh4i9Qt8QY>
[12] <https://youtu.be/VgLVmzW4Iro>
[13] <https://youtu.be/cVml0d6sseQ>

Figura 45 – Modelos de *CAVE* 360 com projeção de plano de fundo estático de imagem panorâmica 360°. Fonte: Própria.

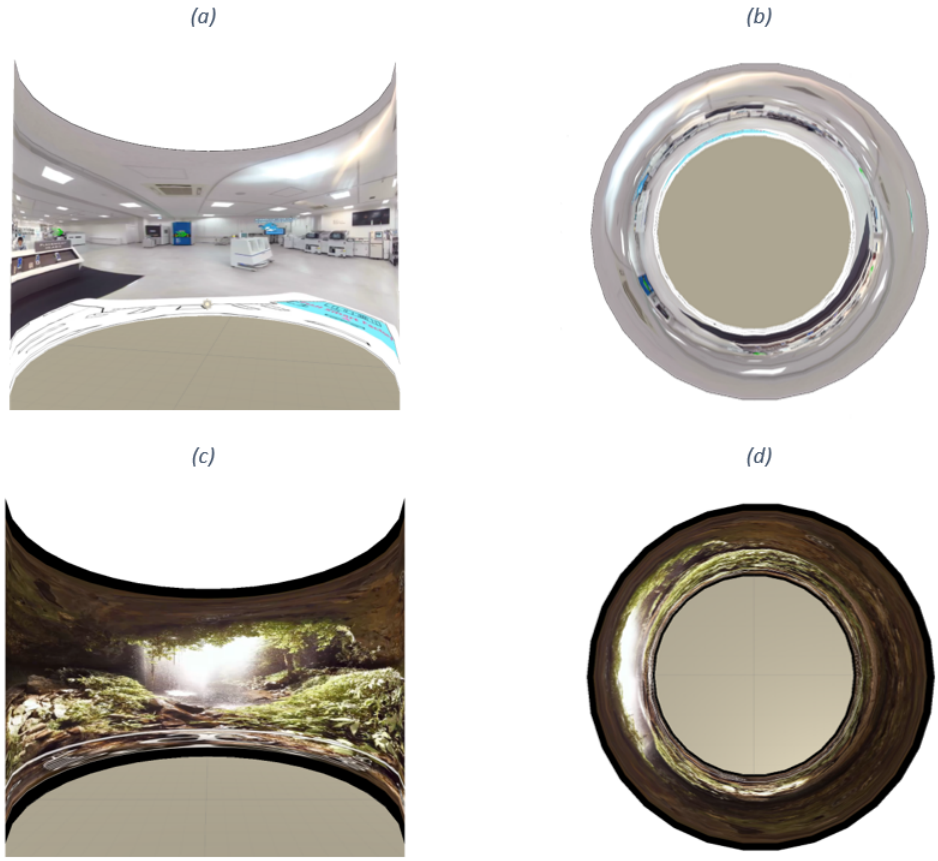


Figura 46 – Modelos de *CAVE* 360 com projeção de plano de fundo em vídeo 360°. Fonte: Própria.

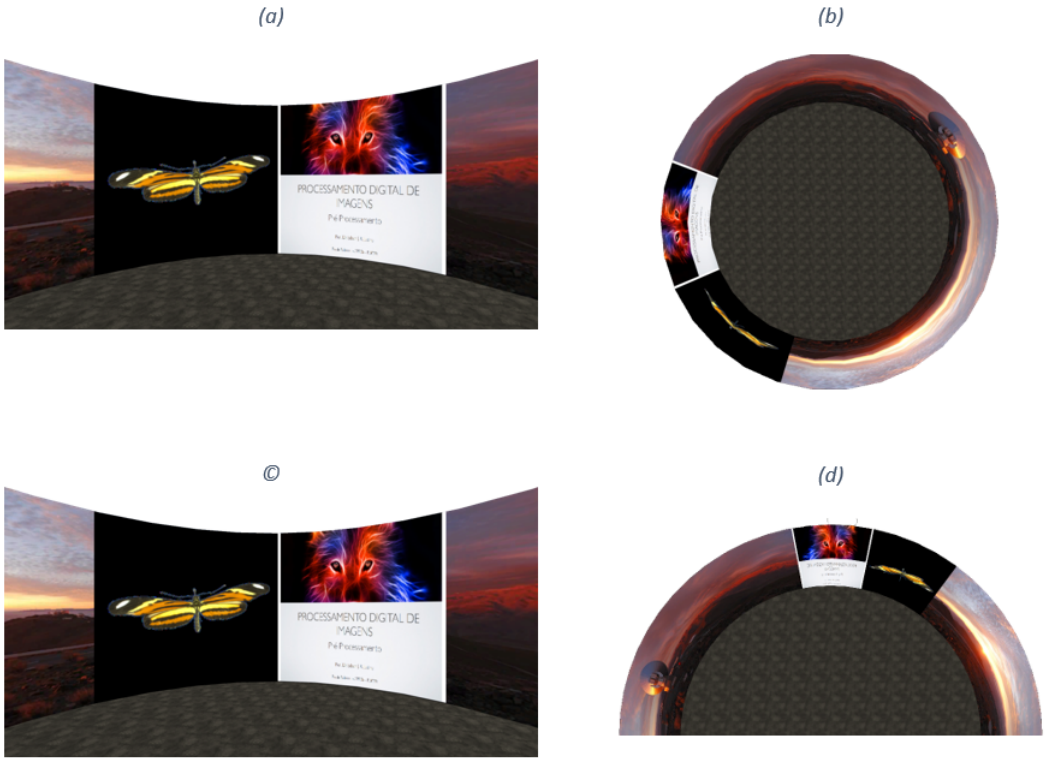


Figura 47 – Modelo de *CAVE* 360 (a)(b) e meia *CAVE* 180 (c)(d) com imagem estática de plano de fundo, apresentação de slides e animação de rotação de borboleta. Fonte: Própria.

Com a finalidade de demonstrar a performance da aplicação em modelos com diferentes tipos de projeção, foi realizada um comparativo entre os valores de FPS (*frames per second*). Os modelos de *CAVE* utilizados como base foram: resolução = 30, raio = 15, altura = 12 e range = 360. Os resultados estão dispostos na Tabela 3.

Elementos Renderizados	FPS
Img estática 360	2700 à 2900
Vídeo 360	2600 à 2800
Img estática 360 e borboleta com rotação 360	1200 à 1800
Img estática 360 e apres. de slides	2000 à 2300
Img estática 360, apres. de slides e borb. c/ rotação.	1600 à 2100

Tabela 3 – Oscilação de FPS em cada modelo.

Com relação às limitações da ferramenta, nos modelos em que são renderizados vídeos 360° a visualização apresenta deformações pois os vídeos que estão

sendo utilizados possuem projeção cilíndrica equidistante e geralmente são usados em domos e experiências de realidade virtual. Este tipo de vídeo, quando projetado em um ambiente próprio, proporciona a visualização em 360° de ambos os eixos xy e em uma *CAVE* cilíndrica, a visualização em 360° acontece apenas na horizontal. Portanto, para projetar vídeos equirectangulares sem deformações na *CAVE* é necessário realizar um ajuste nos pontos de deformidade dos *frames* do vídeo, de forma que se obtenha um vídeo panorâmico 360 plano sem distorções.

Como o ICE-S ainda está em sua primeira versão, existem alguns aspectos que devem ser melhorados. A usabilidade de modo geral é funcional, ou seja, todos os recursos anteriormente citados na etapa de implementação funcionam, porém, alguns processos ainda precisam de ajustes. Por exemplo, a geração do vídeo como plano de fundo não é feita no mesmo componente utilizado para gerar a imagem como plano de fundo. Toda vez que é feita a troca entre estes dois elementos é preciso realizar alguns ajustes que poderiam ser implementados diretamente no código, o que tornaria o processo de geração do plano de fundo mais acessível. Visto isso, alguns processos podem ser melhor estruturados, de forma a se obter uma melhor intuitividade na utilização de alguns recursos.

Na próxima Seção será feita uma discussão sobre a aquisição e processamento dos insetos que compõem a base de imagens, apresentando os resultados que foram obtidos nos modelos adquiridos bem como os problemas enfrentados durante o processo.

4.2 Análise do método de aquisição e processamento dos insetos

Neste trabalho foi utilizado o *F2S2* para fazer a aquisição dos modelos. A câmera do *scanner* é em ultra alta definição *4k*, e realiza a aquisição de imagens com alta acuidade visual, possibilitando visualizar todos os detalhes quando as imagens são escaladas na projeção, sem que haja grande perda de qualidade. A câmera apresentou bons resultados nos modelos que foram escaneados, mas pode-se esperar que para insetos menores, como por exemplo abelhas, será necessário o uso de uma lente macro para capturar os mínimos detalhes.

Um dos maiores problemas do *F2S2* é o tempo que leva para fazer a aquisição de um único modelo. Quando o *pack* de imagens é visualizado no computador,

apesar dos vários testes que são realizados antes de iniciar o escaneamento completo, o modelo pode acabar não obtendo a acuidade visual esperada em algumas posições. Nestes casos é preciso realizar o processo de aquisição novamente, o que torna todo o processo lento. Se levado em consideração o tempo de processamento, alguns modelos podem levar dias para que se obtenha um bom conjunto de imagens com boa acuidade visual e sem perca de detalhes.

O *BipApp*, apesar das questões de tempo citadas acima, permitiu realizar todos os processamentos necessários nas imagens de forma eficaz. Os problemas que ocorreram durante esta etapa condizem mais aos modelos que estão sendo trabalhados – insetos – do que com a ferramenta propriamente dita.

Os insetos, conforme citado anteriormente, possuem muitos detalhes que são relevantes para o estudo da entomologia. Portanto, ao construir uma coleção digital de insetos reais é preciso que todas as imagens possuam todos estes detalhes. O Gafanhoto e borboleta são insetos que possuem características bem diferentes um do outro, apesar de ambos poderem ser considerados relativamente grandes quando comparados com outros insetos (e.g. joaninha e abelha). As imagens do gafanhoto apresentam boa acuidade visual e todos os seus detalhes foram mantidos depois de ter sido processado pelo *BipApp* (Figura 48). Por outro lado, por mais que as imagens da borboleta também possuem boa acuidade visual, algumas partes da antena foram perdidas durante a etapa de segmentação. Por mais que não foram aplicadas as técnicas de erosão e suavização de bordas, devido à espessura muito fina das antenas da borboleta – aproximadamente 1 *pixel* –, os *pixels* referentes ao fundo azul acabam sendo misturados com os *pixels* da antena, fazendo com que a antena fique levemente azulada. Por esse motivo, o *BipApp* não consegue diferenciar a antena do fundo em algumas imagens e acaba removendo parcialmente ou inteiramente essa característica da borboleta, conforme exibido anteriormente na Figura 26 da Seção 3.3.2.

Outra limitação na construção da base de imagens envolve a remoção do alfinete que os modelos possuem. Quando os insetos são coletados eles são alfinetados. Ao ser alfinetado, o inseto expele um líquido, que acaba apresentando a mesma característica de uma cola. Caso o alfinete seja removido, existe grande chance do inseto se despedaçar e o modelo pode ser perdido. Portanto, foi indicado pelos especialistas do grupo de trabalho da área de Entomologia da Escola de Ciências da Vida e Medicina da P*UCPR* que o alfinete não seja removido. Em

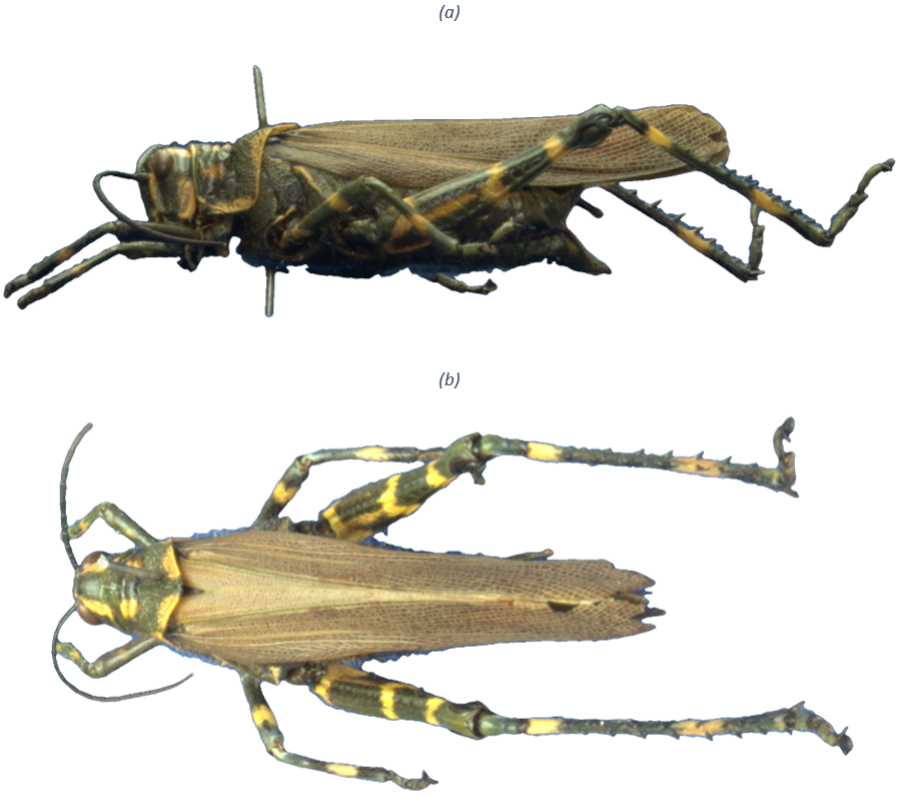


Figura 48 – Demonstração de qualidade de aquisição e processamento nas imagens do gafanhoto. Fonte: Própria.

alguns ângulos o alfinete sobrepõe o inseto, como por exemplo, em uma imagem com perspectiva da parte superior do inseto. Nestes casos não existe uma forma de remover o alfinete sem deixar um buraco vazio no meio do inseto. Portanto, optou-se por manter o alfinete nas imagens, visto que a visualização na lupa também ocorre com a existência do alfinete e o estudo não é afetado negativamente por isto.

O próximo capítulo aborda os resultados e características do processo de desenvolvimento da interface de controle em mesa digital com projeção estereoscópica, bem como os desafios e limitações descobertos durante a implementação.

4.3 Análise da interface de controle e projeção estereoscópica

A interface de controle com projeção do modelo volumétrico estereoscópico foi desenvolvida no *hardware* da mesa digital com projetor 3D *4K*. Quando o programa é compilado duas janelas são renderizadas, uma com a interface de

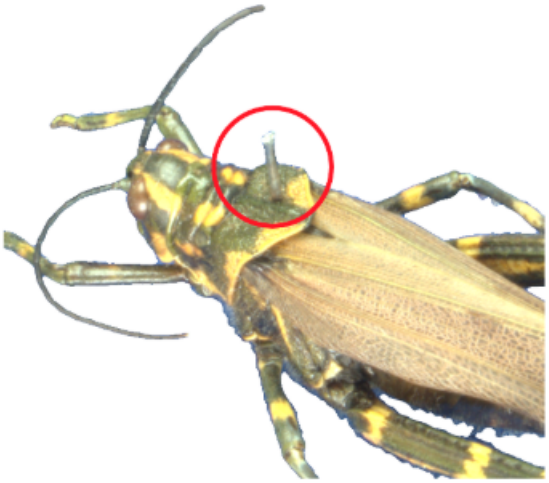


Figura 49 – Recorte de imagem segmentada do grilo onde a segmentação do alfinete resultaria em perda de parte do inseto. Fonte: Própria.

controle na mesa digital e a outra com a projeção estereoscópica no projetor.

Todos os mecanismos propostos para a aplicação funcionam corretamente. A simulação do modelo tridimensional exibido na mesa executa todas as transformações geométricas – translação, rotação e escala – conforme os *inputs* fornecidos pelo usuário e todas as transformações são aplicadas igualmente no modelo volumétrico projetado.

Os insetos que constam na aplicação são os mesmos que foram capturados no processo de aquisição (gafanhoto e borboleta). Além disso, foi adicionado o modelo de um crânio – adquirido anteriormente pelos colegas de laboratório – para testar as funcionalidades implementadas na interface de controle em outros modelos gerados pelo *F2S2*. Abaixo estão dispostas as figuras referentes ao modelo da aplicação implementada. A Figura 50 exibe a interface de controle; a Figura 51 contém ambas as projeções estereoscópicas dos insetos; e a Figura 52 apresenta a visualização de ambas as janelas da aplicação. É possível obter uma maior compreensão da ferramenta com o vídeo[14] que ilustra a usabilidade do programa.

Inicialmente o programa manipulava uma textura renderizada como plano de fundo, de forma que pudesse ser testada a sobreposição da projeção estereoscópica em imagens que futuramente estariam sendo projetadas na *CAVE*. Durante o processo de implementação e testes alguns colegas sentiram vertigem ao visualizar a

[14] <https://youtu.be/wcR0-hvJmt8>

Figura 50 – Interface de controle em mesa digital com gafanhoto sendo exibido. Fonte: Própria.

Figura 51 – Projeção de modelo volumétrico com visão estereoscópica. Gafanhoto à esquerda (a) e borboleta à direita (b). Fonte: Própria.

Figura 52 – Visualização de ambas as janelas, projeção estereoscópica e interface de controle. Fonte: Própria.

projeção com plano de fundo, alegando que realizar as transformações geométricas no modelo volumétrico em um fundo estático causava certa estranheza. Portanto, optou-se pela remoção da imagem de fundo e os modelos passaram a ser exibidos sobre um plano de fundo preto.

A primeira versão do protótipo foi exibida no evento *OX Inovação* realizado na *FTD Arena Digital* da *PUCPR*. Os convidados do evento puderam conhecer a ferramenta em questão, bem como os demais projetos desenvolvidos pelos colegas de laboratório. Com a interação de usuários que não conheciam a aplicação, foi possível observar os pontos positivos e negativos na usabilidade do *software* e mapear melhorias para tornar o programa mais intuitivo para os usuários em geral.

Figura 53 – Exibição do projeto no evento OX Inovação realizado na FTD Arena Digital. Fonte: Própria

4.4 Considerações Finais

Este capítulo mostrou os resultados obtidos em cada uma das etapas de desenvolvimento do projeto e discutiu suas principais características e limitações. As análises foram feitas considerando a ferramenta ICE-S e os modelos de *CAVE* gerados com o simulador ICE-S, a construção de uma base digital de imagens de insetos e a aplicação dos insetos na projeção estereoscópica com interface de controle em mesa digital. Sobre os resultados apresentados neste capítulo há espaço para melhorias, mas de forma geral, todas as funcionalidades propostas em cada aplicação foram alcançadas.

A próxima Seção apresenta algumas propostas de futuras implementações, tanto para o ICE-S como para a projeção estereoscópica e a interface de controle.

5 Considerações Finais

Esta pesquisa objetivou a criação de uma ferramenta capaz de gerar de forma dinâmica ambientes imersivos em CAVE com estereoscopia utilizando recursos aplicados em jogos digitais. O processo é constituído pelas etapas: (a) produção de uma base digital de imagens de insetos reais com acuidade visual; (b) aplicação de estereoscopia em modelos volumétricos gerados a partir da base de imagens, que permita aplicar as transformações geométricas de rotação, translação e escala a partir de uma interface de controle em mesa digital; e (c) desenvolvimento de um programa capaz de gerar ambientes imersivos em CAVE cilíndrica de forma dinâmica para simular a usabilidade de recursos imersivos em diferentes contextos.

Foram abordadas as particularidades em realizar a aquisição de imagens de insetos com o F2S2, de forma que as imagens geradas possuam acuidade visual para proporcionar uma boa visualização dos detalhes do inseto. Também foram apresentadas as técnicas de pós-processamento realizadas com o BipApp nas imagens dos insetos, de modo que todos os aspectos irrelevantes da imagem fossem removidos e apenas o objeto de interesse fosse mantido.

Outro ponto de discussão foi a implementação de técnicas de visão estéreo para obter uma projeção volumétrica flutuante dos insetos que possam ser usados como objetos de aprendizado em aula. A projeção estereoscópica foi desenvolvida juntamente à uma interface de controle em mesa digital para possibilitar a interação de usuários, de forma que este recurso possa ser utilizado como um meio de colaboração entre estudantes e professor. Esta implementação poderia ser adaptada para diferentes contextos, como por exemplo, treinamentos em grupo e exibição de produtos em feiras.

Por fim, foi apresentado o simulador ICE-S como uma proposta de redução de esforço humano na construção de ambientes imersivos colaborativos em CAVE. Os modelos gerados pelo simulador atuam como maquete digital do ambiente real e podem ser personalizados de diversas formas utilizando os recursos da game engine Unity.

A contribuição do projeto constitui na elaboração do ferramental desenvolvido que facilita a compreensão e construção de ambientes imersivos colaborativos

em CAVE. As ferramentas apresentadas neste trabalho fazem parte de um conjunto de soluções que seguem um mesmo contexto e que vem sendo estudadas por todos os colegas que fazem parte do grupo de pesquisa em questão.

Com isso, os objetivos específicos foram alcançados, de modo que foi obtida uma projeção volumétrica estereoscópica com os modelos gerados pelo F2S2, que podem ser manipulados pelos usuários por meio da mesa digital, viabilizando sua utilização como um objeto de aprendizagem. O simulador ICE-S realiza a criação de ambientes imersivos em CAVE de forma dinâmica e pode ser utilizado para elaboração de protótipos de diferentes modelos de CAVE com diferentes finalidades, tendo como resultado uma maquete digital do ambiente projetado.

Ao cumprir os objetivos específicos, o objetivo geral também foi atingido, que foi a criação de um gerador de CAVE cilíndrica dinâmica com múltiplas projeções. Em consequência disso, se obtém uma maquete digital que pode ser utilizada como base de construção do modelo físico com redução de esforço humano e que pode ser aplicada em diferentes contextos. Contudo, foram identificados alguns pontos de melhoria que não foram abordados neste trabalho, que serão listados como trabalhos futuros.

5.1 Trabalhos Futuros

1. Realizar a implementação de recursos para que o simulador seja capaz de renderizar os mesmos elementos do modelo digital em um modelo real.

2. Integrar a projeção do modelo volumétrico com estereoscopia e interface de controle ao ICE-S, para que também possa ser personalizada dentro do simulador.

3. Disponibilizar modelos pré-definidos para projeção de elementos com movimento, que possam ser aplicados sem a necessidade de implementação de código.

4. Alterar as dimensões da textura de projeção para um tamanho parametrizável.

5. Implementar reconhecimento de pixels com alfa diferentes de 0 ou 255, para viabilizar a renderização de elementos com transparência.

6. Revisar algumas estruturas implementadas e elaborar um modelo mais intuitivo que facilite a usabilidade da ferramenta.

Referências

ADAMS, G. *Um balanço bibliográfico e de fontes da estereoscopia*. [S.l.]: SciELO Brasil, 1999. v. 6-7. 207–225 p. ISSN 0101-4714. Citado na página 34.

ALMEIDA, M.; JUSTINO, E. *Como o Cérebro Processa a Matemática ? - Ensinamentos da Neurociência para uma Pedagogia Renovada (How Does the Brain Process Mathematics? - Teachings of Neuroscience for a Renewed Pedagogy)*. [S.l.: s.n.], 2020. ISBN 978-85-924793-4-3. Citado 4 vezes nas páginas 14, 25, 26 e 30.

BOAS, Y. Overview of Virtual Reality Technologies. In: *Interactive Multimedia Conference*. [S.l.: s.n.], 2013. v. 13, n. 1, p. 48–69. ISSN 18763308. Citado na página 42.

BRONGEL, A. R. *Uma Aplicação para Mesa Digital de Dissecação de Corpos Humanos para o Estudo de Anatomia*. 2020. Disponível em: <https://www.ppgia.pucpr.br/pt/arquivos/mestrado/dissertacoes/2020/Andrei_Rafael_2020.pdf>. Citado na página 71.

CARVALHO, V. de. Pontos de vista: modernidade e visão estereoscópica. 2006. Citado na página 33.

CREAGH, H. Cave automatic virtual environment. In: IEEE. *Electrical Insulation Conference and Electrical Manufacturing and Coil Winding Conference and Exhibition*. [S.l.], 2003. p. 499–504. ISBN 0941783235. Citado na página 25.

CURTIS, D. et al. Several devils in the details: making an AR application work in the airplane factory. In: *International workshop on Augmented Reality IWAR*. [S.l.: s.n.], 1999. p. 47–60. Citado na página 43.

HIETALA, O.; KOPONEN, J. Title developing a game engine with sdl vaihtoehtiset optional professional studies. 2011. Citado na página 63.

JOSEANI; BENDER, B. E. H. V.; KOCHHANN. Estratégias de terapias de exposição à realidade virtual: uma revisão discutida sob a ótica analítico-comportamental. *Psicologia ClÃnica*, scielopepsic, v. 28, p. 15 – 34, 12 2016. ISSN 0103-5665. Disponível em: <http://pepsic.bvsalud.org/scielo.php?script=sci_arttext&pid=S0103-56652016000300002&nrm=iso>. Citado na página 24.

KENYON, R. The CAVE(TM) automatic virtual environment: Characteristics and applications. *Illinois Univ, Human-Computer Interaction and*, p. 150–168, 1995. Citado na página 48.

KIRNER, C.; SISCOUTTO, R. Realidade virtual e aumentada: conceitos, projeto e aplicações. In: *Porto Alegre: Sociedade Brasileira de Computação*. [S.l.: s.n.], 2007. p. 28. Citado na página 36.

LENOIR, T.; LOWOOD, H. Theaters of war: The military-entertainment complex. *Collection-Laboratory-Theater: Scenes of knowledge in the 17th century*, p. 429–445, 2002. Citado na página 42.

LUNAZZI, J. J.; FRANÇA, M.; MORI, A. Revivendo o estereoscópio de Wheatstone. *Revista Brasileira de Ensino de Fisica*, SciELO Brasil, v. 37, n. 2, p. 2501, 2015. ISSN 01024744. Citado na página 33.

MACHADO, J. Os Primórdios dos Simuladores de Voo. Museu Aeroespacial, 2016. Citado na página 39.

MALARD, M. L. et al. Princípios Teóricos da Estereoscopia. p. 19, 2008. Citado na página 32.

MANJREKAR, S. et al. CAVE: An emerging immersive technology -a review. In: IEEE. *Proceedings - UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, UKSim 2014*. [S.l.], 2014. p. 131–136. ISBN 9781479949236. Citado na página 25.

NICOLL, B.; KEOGH, B. *The Unity Game Engine and the Circuits of Cultural Software*. [S.l.: s.n.], 2019. ISBN 978-3-030-25011-9. Citado na página 64.

NOGUEIRA, A. Suas Tecnologias. *Enem*, p. 22–25, 2005. Citado na página 37.

OLSEN, D. R. *Geração de Modelos Digitais Interativos em Alta Definição de Objetos do Mundo Real*. 2020. Disponível em: <https://www.ppgia.pucpr.br/pt/arquivos/doutorado/teses/2020/Diogo_Olsen_2020.pdf>. Citado 3 vezes nas páginas 53, 57 e 59.

ONIME, C.; ABIONA, O. 3d mobile augmented reality interface for laboratory experiments. *International Journal of Communications, Network and System Sciences*, Scientific Research Publishing, Inc„ v. 09, p. 67–76, 2016. ISSN 1913-3715. Citado na página 25.

PAGE, R. Brief history of flight simulation. *SimTecT 2000 Proceedings*, Citeseer, p. 1–11, 2000. Disponível em: <bit.ly/2UBi8UF>. Citado na página 40.

PANTELIDIS, V. S. Reasons to Use Virtual Reality in Education and Training Courses and a Model to Determine When to Use Virtual Reality. *Themes in Science and Technology Education*, v. 2, n. 1-2, p. 59–70, 2010. ISSN 1792-8788. Citado na página 26.

PERES, I. M. Aplicações da Fotografia Estereoscópica às Ciências : uma Perspectiva Histórica ESTEREOSCÓPICA ÀS CIÊNCIAS :. *Stereo & Immersive Media. Proceedings 2015*, n. November, p. 24–50, 2016. Citado na página 34.

PHILIPE, G. Como funciona o Google Glass. Oficina da net, 2014. Disponível em: <https://www.oficinadanet.com.br/post/12504-como-funciona-o-google-glass>. Citado na página 45.

RAPOSO, A. et al. Visão Estereoscópica, Realidade Virtual, Realidade Aumentada e Colaboração. *XXIII JAI–Jornada de Automatização em Informática, Capítulo*, v. 8, 2004. Citado na página 32.

SANTOS, D. dos; DIAS, F. Uso de Anaglifos como Alternativa para Práticas de Estereoscopia em Sensoriamento Remoto. *Anuário do Instituto de Geociências*, v. 34, n. 2, p. 105–111, 2011. ISSN 0101-9759. Citado na página 35.

SANTOS, E. T. Novas Tecnologias No Ensino De Desenho E Geometria. *Anais do I Encontro Regional do Vale do Paraíba de Profissionais do Ensino da Área de Expressão Gráfica*, p. 71–81, 2000. Disponível em: <bit.ly/39Aa36J>. Citado na página 37.

SILVA, F. de Almeida e. *Projeção interativa estereoscópica: visualização de modelos de objetos reais em ambiente virtual*. 2020. Disponível em: <https://www.ppgia.pucpr.br/pt/arquivos/doutorado/teses/2020/Flavio_Almeida_2020.pdf>. Citado 4 vezes nas páginas 54, 72, 76 e 78.

SILVA, F. e et al. Stereoscopic interactive objects: Acquisition, generation and evaluation. *CSEDU 2019 - Proceedings of the 11th International Conference on Computer Supported Education*, v. 2, p. 165–176, 2019. Citado na página 53.

SISCOUTTO, R. A. et al. Estereoscopia. *Realidade virtual: conceitos e tendências*, 2004. Citado na página 38.

SRIVASTAVA, K.; CHAUDHURY, S.; DAS, R. C. Virtual reality applications in mental health: Challenges and perspectives. *Industrial Psychiatry Journal*, Wolters Kluwer–Medknow Publications, v. 23, n. 2, p. 83, 2014. ISSN 0972-6748. Citado na página 41.

SUPERVISOR, P. P. M.; JúNIOR, E. A. da C. Unb games: A collaborative project. 2017. Disponível em: <https://bdm.unb.br/handle/10483/19790>. Citado na página 63.

SUTHERLAND, I. E. Head-Mounted Three Dimensional Display. In: *Proceedings of the December 9-11, 1968, fall joint computer conference, part I.* [S.l.: s.n.], 1968. v. 33, n. pt 1, p. 757–764. Citado na página 41.

TURI, J. The sights and scents of the Sensorama Simulator. *Engaget*, v. 26, 2014. Disponível em: <engt.co/2UQhNwb>. Citado na página 40.

WESSON, M.; LIPTON, L. *Foundations of the Stereoscopic Cinema: A Study in Depth*. [S.l.]: Van Nostrand Reinhold, 1984. v. 25. 689 p. ISSN 0040165X. Citado na página 34.

WOLDEGIORGIS, B.; LIN, C.; LIANG, W.-Z. Impact of parallax and interpupillary distance on size judgment performances of virtual objects in stereoscopic displays. *Ergonomics*, v. 62, p. 1–31, 09 2018. Citado na página 78.

YU, J. A light-field journey to virtual reality. *IEEE MultiMedia*, v. 24, n. 2, p. 104–112, 2017. Citado na página 24.

ZUFFO, J. A. et al. CAVERNA Digital - Sistema de Multiprojeção Estereoscópico Baseado em Aglomerados de PC's para Aplicações Imersivas em Realidade Virtual. In: *4th Brazilian Symposium on Virtual Reality - SVR'01*. [S.l.: s.n.], 2001. p. 139–147. Citado na página 46.