AndVR: Uma biblioteca em OpenGL ES de Realidade Virtual para Android

Jordan Lira de Araújo Junior

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2018

Jordan Lira de Araújo Junior

AndVR: Uma biblioteca em OpenGL ES de Realidade Virtual para Android

Monografia apresentada ao curso Ciência da Computação
do Centro de Informática, da Universidade Federal da Paraíba,
como requisito para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Danielle Rousy Dias da Silva

Novembro de 2018

Catalogação na publicação
Seção de Catalogação e Classificação

J95a Junior, Jordan Lira de Araujo.
AndVR: Uma biblioteca em OpenGL ES de Realidade Virtual para Android / Jordan Lira de Araujo Junior. - João Pessoa, 2018.
45 f. : il.

Orientação: Danielle Rousy Dias da Silva.
Monografia (Graduação) - UFPB/CI.

1. Realidade Virtual; Motor Gráfico; Android. I. Silva, Danielle Rousy Dias da. II. Título.

UFPB/BC

widthheightfigure

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Ciência da Computaçãointitulado ***AndVR: Uma biblioteca em OpenGL ES de Realidade Virtual para Android*** de autoria de Jordan Lira de Araújo Junior, aprovada pela banca examinadora constituída pelos seguintes professores:

Danielle Rousy Dias da Silva
Prof. Dra. Danielle Rousy Dias da Silva
Universidade Federal da Paraiba

Jessica Maciel de Castro
Mestranda Jessica Maciel de Castro
Universidade Federal da Paraiba

Prof. Dr. Christian Azambuja Pagot
Universidade Federal da Paraiba

Gustavo Henrique Matos Bezerra Motta
Coordenador(a) do Departamento Departamento de Ciência da Computação
Gustavo Henrique Matos Bezerra Motta
CI/UFPB

João Pessoa, 24 de junho de 2018

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

Olhe para as estrelas e não para baixo, para os seus pés. Tente entender o que vê e questione-se sobre o que faz o universo existir. Seja curioso.

Stephen Hawking

Eu dedico esta monografia a minha mãe,
a pessoa mais esforçada, honesta e
simples que eu conheço nessa vida,
um exemplo a ser seguido.

AGRADECIMENTOS

Agradeço aos meus pais, Jordan e Lourdinha, por todo o suporte e paciência que tiveram comigo, a minha irmã Jordana, por sempre acreditar em mim mesmo quando ninguém mais parece acreditar.

A minha namorada Jéssica, por não só ser meu apoio nas horas boas e ruins, mas também, como uma referência acadêmica a ser seguida.

A minha orientadora Danielle Rousy, pelas correções, incentivos e pela atenção durante toda escrita do trabalho.

E a todos que me ajudaram diretamente e indiretamente neste trabalho, muito obrigado.

RESUMO

Com o advento e barateamento dos *smartphones* e seus sensores posicionais houve uma utilização destes para aplicações em realidade virtual. Porém, a escolha de qual motor gráfico utilizado para a produção de conteúdo em realidade virtual, ainda representa uma complexa decisão devido a alguns problemas. Neste trabalho, foi desenvolvido um motor gráfico de realidade virtual para o sistema operacional *Android*, em forma de biblioteca, afim de facilitar a criação e utilização de cenários virtuais em aplicações nativas *Android*. Abstraindo assim, para o desenvolvedor final, a utilização de noções complexas de computação gráfica e removendo o obstáculo de utilizar ferramentas externas das utilizadas por programadores de *Android* nativo. O resultado é uma biblioteca de simples utilização e importação, na IDE *Android Studio*, que permita o programador utilizar sem a necessidade de recorrer a outras linguagens ou ferramentas de desenvolvimento.

Palavras-chave: Realidade Virtual; Motor Gráfico; Android.

ABSTRACT

With the progress and cheapness of smartphones and their positional sensors, there was a increase of it's use for applications in virtual reality. However, the choice of which graphic engine use in the virtual reality content production still represents a complex decision due to some problems. In this work, a virtual reality graphic engine was developed for the Android operating system, in the form of a library in order to facilitate the creation and use of virtual scenarios in native Android applications. Thus, for the final developer, the use of complex notions of computer graphics and, removing the hurdle of using external tools, those used by native Android programmers. The result is a library of simple use and import, in the IDE textit Android Studio, that allows the programmer to use it without the necessity to resort to other languages or development tools.

Keywords: Virtual Reality; Graphic Engine; Android.

LISTA DE FIGURAS

1 Usuário utilizando o Sketchpad Fonte: SUTHERLAND (1963) 19

2 Usuária utilizando óculos de RV Fonte: CASTELVECCHI (2016) 19

3 Primeiro SBHMD Fonte: LOGAN (2011) 21

4 *Headset* recente da Google Fonte: AMAZON (2018) 22

5 Sistema com uma câmera Fonte: Autor (2018) 23

6 Sistema de câmeras em VR Fonte: Autor (2018) 23

7 Desenho de Pesquisa Fonte: Autor (2018) 27

8 Comparativo entre Java e C Fonte: ANDROID AUTHORITY (2018) . . . 28

9 Diagrama de Componentes Fonte: Autor (2018) 29

10 Marketshare das versões Android OS 2018 Fonte: STATISTA (2018) . . . 30

11 Diagrama de Atividades Fonte: Autor (2018) 32

12 Cenário Demonstrativo com poucos polígonos Fonte: Autor (2018) 35

13 Cenário Demonstrativo com muitos polígonos Fonte: Autor (2018) 35

14 Collada Duck no Blender Fonte: Autor (2018) 36

15 Collada Duck no AndVR Fonte: Autor (2018) 37

16 Camionete no Blender Fonte: Autor (2018) 37

17 Camionete no AndVR Fonte: Autor (2018) 38

LISTA DE TABELAS

1 Comparativo de Motores Gráficos de RV para Android 24

LISTA DE ABREVIATURAS

API – *Application Programming Interface*

HMD – *Head Monted Display*

SBHMD – *Smartphone Based Head Monted Display*

RV – Realidade Virtual

GLSL – *OpenGL Shading Language*

IDE – *Integrated Development Environment*

SDK - *Software Development Kit*

QPS - Quadros por Segundo

QoE - *Quality of Experience*

UX - *User Experience*

JVM -*Java Virtual Machine*

GC - *Garbage Colector*

VBO - *Vertex Buffer Object*

GPU - *Graphics Processing Unit*

POO - Programação Orientada a Objetos

AAR - *Android Archive Library Package*

Sumário

1 INTRODUÇÃO **14**

1.1 Definição do Problema 15

1.2 Objetivos 15

1.2.1 Objetivos Gerais 16

1.2.2 Objetivos Específicos 16

1.3 Estrutura da monografia 16

2 CONCEITOS GERAIS E FUNDAMENTAÇÃO TEÓRICA **18**

2.1 Realidade Virtual 18

2.2 Head-mounted Displays 19

2.3 Smartphone-Based Head Mounted Display 20

2.4 Engines Gráficas de Realidade Virtual para Android 22

3 METODOLOGIA **25**

3.1 Metodologia da Pesquisa 25

3.1.1 Fase Exploratória 25

3.1.2 Prototipação, avaliação e análise 26

3.1.3 Conversão para o formato de biblioteca 26

3.2 Visão geral do motor gráfico AndVR 27

3.3 Renderização Gráfica no AndVR 29

3.4 Processo de validação do AndVR 32

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS **34**

4.1 Hardware utilizado 34

4.2 A Aplicação Demonstrativa 34

4.3 Análise da Experiência e do Desempenho 36

5 CONCLUSÕES E TRABALHOS FUTUROS **39**

REFERÊNCIAS **39**

1 INTRODUÇÃO

Desde o advento do universo cinematográfico, a indústria de entretimento evoluiu, cada vez mais, em trazer a imersão de ambientes fictícios para o público. Apostando na utilização de sistemas sonoros mais envolventes, projeções em resoluções maiores, telas curvas, mais luzes, cheiros e movimentos para envolver cada vez mais o indivíduo no conteúdo proposto. Garantindo, não só o interesse dele no conteúdo, mas diminui as distrações que possam tirar a atenção e auxilia em um maior entretenimento ao consumir o conteúdo preparado para ele.

A ideia da realidade virtual (RV) surgiu por meio de pequenos experimentos realizados pelo cineasta Morton Heilig[1], com uma máquina chamada Sensorama[2] em meados de 1960, paralelamente com experimentos realizados em simuladores militares. Utilizando equipamentos que obstruem todo o campo de visão do usuário, a RV troca a realidade verdadeira por um cenário gerado computacionalmente que estimula uma experiência em três dimensões de alta imersão. Esta tecnologia também se adentrou nas pesquisas acadêmicas, se ramificando em diversos campos, como na área de saúde em simuladores cirúgicos e reabilitações[3]. Concomitantemente, existem estudos na área de negócios, na visualização de maquetes de edificações e seus interiores[4], como também nos jogos eletrônicos, onde segundo Mishra e Shrawankar [5], a realidade virtual é iminente e não é exagero prever que a RV será um dos gigantes avanços na remoção de barreiras de vídeo games e do mundo real.

Com o advento de *smartphones* de baixo custo e com especificações cada vez mais potentes de processamento, maiores resoluções de *displays* e inclusão de sensores orientacionais — como o giroscópio, o acelerômetro e a bússola — permitiram a entrada destes aparelhos na indústria de realidade virtual, utilizando o próprio *smartphone* como *display*, com o uso de todo o *hardware* e *software* autocontidos [6]. A RV, se integrando com o *Head-mounted Displays* (HMD), possibilitou atingir um novo patamar, pois as telas acompanham o usuário, realizando rastreamento da posição da cabeça e dividindo a tela em duas partes, possibilitando a sensação de profundidade tridimensional, técnica conhecida como estereografia. Esta técnica não só possibilita uma maior imersão visual, mas a sensação de perspectiva tridimensional. Barateando significativamente o acesso a RV, tanto para os desenvolvedores, quanto para usuários finais [7]. Os HMDs não apenas possibilitaram uma maior entrada da indústria de jogos, mas também na possibilidade de aplicações comerciais integradas com realidade virtual a um público maior[8].

Entretanto, os HMDs que utilizam *smartphones* trouxeram uma nova metodologia de desenvolvimento. Devido ao uso de dispositivos de pouca performance, a utilização de métodos tradicionais acaba se tornando uma problemática. A utilização de engines gráficas, isto é, ferramentas que permitem a manipulação de objetos tridimensionais no

cenário virtual para a renderização, precisa ser adaptada para rodar nestes dispositivos, de um desempenho satisfatório e facilidade na manutenção.

1.1 Definição do Problema

O desenvolvimento atual de aplicações para *smartphones Android* em RV exige a utilização de motores de jogos, bibliotecas e *Application Programming Interfaces* (APIs), que possuem alguns problemas. Um desses problemas, é na utilização de APIs que exige um conhecimento de computação gráfica avançada para utilizá-las. Isto é devido ao *Android* utilizar o *OpenGL ES* e o *Vulkan* como API de renderizações gráficas, e que elas exigem o conhecimento avançado de computação gráfica é dificilmente encontrado fora de ambientes acadêmicos, como o uso de *pipelines* gráficos, matrizes de transformações e de entendimento da linguagem *OpenGL Shading Language* (GLSL)[9]. Afastando assim, a entrada de desenvolvedores menos experientes de seguir esta metodologia.

Outra problemática encontrada é em relação as *Integrated Development Environments* (IDEs) de motores gráficos que possuem outras linguagens nas quais o usuário pode não estar habituado a desenvolver, o que impossibilita a produtividade quando o desenvolvedor desconhece linguagens como *C#*, utilizada no *Unity*, ou *C++* utilizada na *Unreal Engine*. Um dos motivos deste problema afetar o desenvolvimento são as restrições comerciais devido as custosas licenças utilizadas no ato do desenvolvimento.

Além disso, a própria necessidade de utilizar uma outra plataforma de desenvolvimento, que seja diferente da que está sendo trabalhada, já pode ser considerada um obstáculo que pode tomar proporções de horas ou dias. Pois, tudo depende da curva de aprendizado da ferramenta a ser utilizada e do tempo de download e instalações das mesmas.

Esse desafio das IDEs também intensifica quando as aplicações entram no ramo comercial, pois quando é preciso a interação com uma ou mais aplicações *Android* preexistentes, o desenvolvedor pode precisar de funcionalidades e recursos do *Android Software Development Kit* (SDK) que não estarão disponíveis nas IDEs escolhidas.

Logo, é fundamental a necessidade de uma API que forneça uma interface simples de desenvolvimento e que permita uma fácil integração com uma aplicação existente.

1.2 Objetivos

O objetivo deste projeto é propor uma possível opção de desenvolver uma biblioteca de RV, com o intuito de solucionar as problemáticas de diferentes linguagens e compatibilidades com projetos *Android* pré-existentes. Além disso, que seja simples, *open-source* e rápida o sulficiente para a utilização comercial.

Em outras palavras, uma API que esteja apta para o desenvolvimento de aplicação comercial ou jogo digital com RV, pode ser criada na plataforma *Android*, utilizando o seu *SDK* e o *Android Studio*. Fornecendo assim a capacidade de reaproveitar códigos, texturas, imagens e as funcionalidades disponíveis no *Android SDK*, acelerando o desenvolvimento de aplicações com funcionalidades de RV e até em jogos digitais.

No intuito de atingir o objetivo supracitado, tem-se os seguintes *objetivos gerais* e *objetivos específicos* analisados neste trabalho:

1.2.1 Objetivos Gerais

- Propor o desenvolvimento de uma API que possibilite a construção de aplicativos em realidade virtual 3D estereográfica na plataforma *Android.*;

1.2.2 Objetivos Específicos

- Desenvolver uma arquitetura de fácil manutenção;
- Suportar objetos exportados de aplicações de modelagem tridimensional;
- Permitir, facilmente, operações como translação, rotação e escala de objetos tridimensionais;
- Posicionar fonte de luz com potência customizável;
- Preservar um desempenho aceitável e uma taxa estável de quadros por segundo;
- Suportar diversos óculos e lentes diferentes.

1.3 Estrutura da monografia

Esta monografia está estruturada em cinco capítulos. Onde este atual descreve o conteúdo da introdução é categorizado como o Capítulo 1, segue-se então o Capítulo 2, com os conceitos gerais e fundamentação teórica empregados na produção deste trabalho, onde serão detalhados alguns conceitos básicos sobre realidade virtual, engines de jogos, renderização 3D estereográfica, API's de renderização gráficas e como o seus estados da arte atual.

No Caítulo 3, será abordada detalhadamente a metodologia utilizada no desenvolvimento da engine gráfica, especificando as suas estruturas e chamada de renderização de API gráfica.

No Capítulo 4, será apresentada a biblioteca desenvolvida implementada em um aplicativo demonstrativo e logo em seguida será feita uma análise de utilização e comparativo de desempenho em relação as opções existentes no mercado.

Por fim, o Capítulo 5 será apresentado as considerações finais acerca do trabalho desenvolvido, isto é, seus aspectos positivos e negativos, possíveis melhorias, bem como perspectivas de trabalhos futuros a partir do desenvolvimento desta monografia.

2 CONCEITOS GERAIS E FUNDAMENTAÇÃO TEÓRICA

Para entender a problemática e a solução abordadas neste trabalho é preciso o conhecimento de alguns conceitos básicos envolvidos, dentre eles, destacamos o conhecimento básico sobre engines de renderização gráficas como também a criação de uma realidade virtual esterográfica, discutindo desde seu processo de concepção e desenvolvimento, a sua validação e aplicação através dos processos de testes e de comparação.

2.1 Realidade Virtual

Segundo Kirner a "realidade virtual é uma interface computacional que permite ao usuário interagir em tempo real, em um espaço tridimensional gerado por computador, usando seus sentidos, através de dispositivos especiais"[12][1]. Já Burdea e Coiffet, que definem a realidade virtual como "uma interface computacional avançada que envolve simulação em tempo real e interações, através de canais multisensoriais"[10].

A história da realidade virtual diverge entre autores. Kirner [12] sugere que surgiu em uma aplicação denominada Sketchpad [13] de 1963. Esta aplicação permitia a manipulação de figuras tridimensionais no monitor de um computador em tempo real. Já Rheingold[17] cita a origem em referências de uma patente datada de 1962, de uma máquina que permitia a visualização de cenários animados estereogŕaficos, o Sensorama Simulator do Morton Heilig [2]. Este simulador permitia a emissão de odores, sistema de som estéreo e uma cadeira que se movimentava afim de gerar a sensação de movimento.

Desde sua origem, a realidade virtual é tentada em diversos tipos de tecnologias nas diversas formas de aplicações possíveis, Kirner[14] define dois grandes grupos de realidade virtual: a realidade virtual não imersiva, que é projetada em meio de uma janela, ou um projetor fixo e não possui capacidade de abranger todo aspecto visual do usuário. Este grupo é mais comum no início da RV, como podemos ver na aplicação de Sketchpad [13] e no Sensorama do Heilig [2]. Como é possível observar na Figura 1, os sistemas que utilizam RV não imersiva, não conseguem cobrir todo campo visual do usuário.

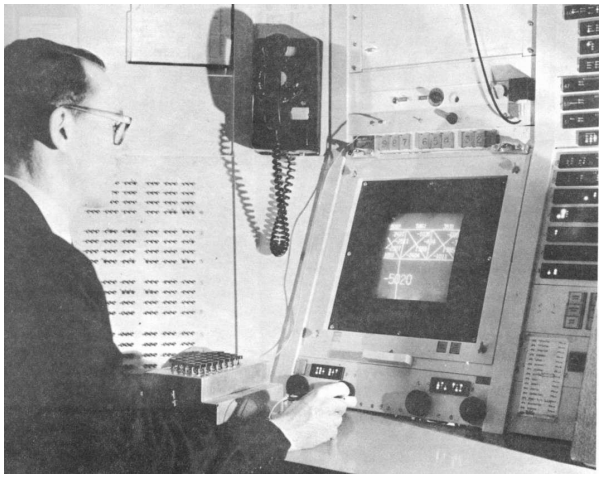

Figura 1: Usuário utilizando o Sketchpad
Fonte: SUTHERLAND (1963)

Já a implementação de realidade virtual imersiva, como mostra na Figura 2, gera uma experiência que permite deixar o usuário totalmente no domínio da aplicação, fazendo com que ele se sinta completamente imerso no mundo virtual gerado digitalmente, podendo interagir com objetos e realizar ações que resultam em mudanças em todo o aspecto visual e auditivo do usuário.

Figura 2: Usuária utilizando óculos de RV
Fonte: CASTELVECCHI (2016)

2.2 Head-mounted Displays

Como foi discutido anteriormente, a RV imersiva proporciona uma experiência mais realista. Segundo Froehlich e Azhar[4], as pessoas recebem 70% da informação do mundo por meio da visão, logo os HMD, tradução direta para "Displays Montados na Cabeça"são exemplos interessante de RV imersivos, pois ao fixar o aparelho na cabeça do usuário, além de cobrir todo o campo de visão, esta tecnologia permite a adição de sensores posicionais,

taís como o giroscópio, bússula e acelerômetro. Estes sensores possibilitam a simulação de vários aspectos multisensoriais impactando, desta forma, no sentimento de imersão de quem está usando o HMD. Os sensores não só permitem a capacidade de determinar a direção em que a cabeça do usuário está apontando, possibilita também, a renderização sincronizada com a câmera em primeira pessoa do mundo renderizado. Além disso, estes *headsets* de realidade virtual utilizam lentes biconvexas para adequar a visão do usuário ao display que está próximo de seus olhos. O próprio fato que os HMD se posicionam bem na frente dos olhos do usuário, auxilia a ofuscar a realidade, sem a remoção da liberdade de seus movimentos, alcançando algo próximo da *imersão total*, como cita Hezel [24].

A ideia de se utilizar os HMD comercialmente é explorada desde 1994, com o surgimento de um dos primeiros óculos de realidade virtual comercial, o *Forte VFX1*, mas com um alto custo e com a capacidade computacional limitada da época, como cita Psotka, [3], os primeiros óculos não conseguiam renderizar uma quantidade de polígonos suficientes para uma imersão realista, nem uma taxa de quadros por segundo (QPS) rápida o suficiente para acompanhar naturalmente os deslocamentos da cabeça do usuário. Entretanto, com o avanço da tecnologia, esses problemas foram se amenizando dando surgimento a displays com taxas de atualização cada vez mais altas.

Existem certas características descritas por Hezel [24], que definem a qualidade de um HMD, estes critérios auxiliam na redução de tonturas e náuseas nos usuários, que são causados por problemas na utilização de RV usando HMD, como cita Tanaki e Takagi [23] e aumentam a capacidade de imersão do usuário. Estas características de qualidade definidas por Hezel podem ser utilizadas como critérios na avaliação de casos de teste para averiguação da *Quality of Experience* (QoE), e estão descritas abaixo:

1. Tridimensão (3D)/Visão binocular
2. Campo de Visão
3. Resolução
4. Abrangência de cores
5. Inexistência de distorções ou aberrações na imagem
6. Acomodação de distância de visualização

2.3 Smartphone-Based Head Mounted Display

Com o criação e popularização [28] dos celulares com tela sensível ao toque como o Iphone [27] e o LG Prada em 2007, houve um acesso maior a dispositivos com grandes telas touchscreen para o grande público, isso proporcionou aos desenvolvedores e as pessoas em

geral a um hardware capaz de renderizar gráficos tridimensionais, com sensores posicionais e com acesso a redes sem fio. Estas funcionalidades juntas, fomentaram diversas pesquisas e produtos, entre elas, a de realidade virtual em dispositivos móveis.

Uma das primeiras idéias nesse sentido foi utilizar os smartphones como dispositivo de HMD. Este caminho foi popularizado com *Smartphone-Based Head Mounted Display* (SBHMD). Esta ideia surgiu com a pesquisa de Logan [6] na qual ele utiliza dois *smatphones* Iphones 4 em um suporte de madeira para representar um cenário virtual gerado no software *Unity 3D*, que é uma engine de jogos multi-plataforma, aliado a um software em um desktop para realizar sincronização das imagens nos *displays*.

Este protótipo inicial de SBHMD, como mostra na Figura 3 abaixo, possuia alguns problemas técnicos, mas conseguiu baratear significativamente os *headset* de realidade virtual pois permitiu que o usuário utilizasse o(s) próprio(s) celular(es) como hardware.

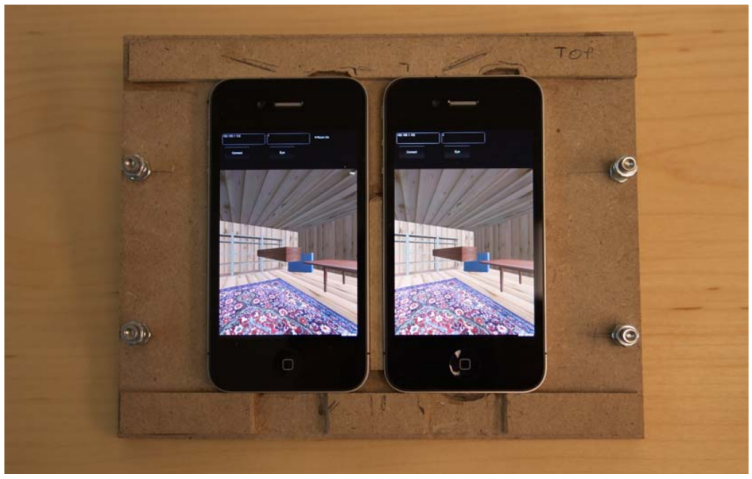

Figura 3: Primeiro SBHMD
Fonte: LOGAN (2011)

Com o avanço tecnológico, maiores displays e chips cada vez mais poderosos, como os SBHMD foram evoluindo ao ponto de não necessitar mais de um *desktop* para renderizar o cenário virtual, podendo ser simplificado em apenas um dispositivo, já que as telas alcançaram atualmente resoluções Full HD, com resolução de 1920x1080 pixels, tamanhos de 4,7 a 6 polegadas. Outro grande avanço foi com o Google Cardboard em 2015 [25], onde foi proposto o suporte a RV em uma forma barata e simples, o que possibilitou uma acessibilidade ainda maior a tecnologia, sendo a porta de entrada da RV para muitas pessoas. Esta nova geração de *headsets* de baixo custo promoveram o aparecimento de diversos outros tipos de *headsets* feitos de diversos materiais e de várias marcas, como cita Amer e Castelvecchi [26][22]. Como o *Headset* VR da Google representado na Figura 4 abaixo.

Figura 4: *Headset* recente da Google
Fonte: AMAZON (2018)

Uma característica notável que diferencia as aplicações de realidade virtual para SBHMD é a interação homem-máquina. Como os *headsets* SBHMD comumente ofuscam a tela *touchscreens* e os botões físicos do *smartphone*, há uma necessidade de se desenvolver aplicações que possuem uma interação virtual, com botões virtuais ou controles no cenário virtual que permitam o usuário interagir com o aparelho, ou alguma forma de controlar a aplicação externamente, como cita Steed [18]. Em uma aplicação RV com botões virtuais por exemplo, quando o usuário necessita clicar neste botão, ele necessita direcionar a sua cabeça para este botão por algum tempo ou clicar utilizando algum *gamepad*.

2.4 Engines Gráficas de Realidade Virtual para Android

Segundo Trenholme[21], a necessidade de se utilizar um motor gráfico (ou engines) se dá pela complexidade de se construir um cenário virtual, receber as interações dos usuários, comportamentos de objetos entre outas dificuldades. Já Mishra e Shrawankar, dizem que motores gráficos são um conjunto de módulos reutilizáveis que podem ser aplicados e manipulados para levar um jogo em direção ao realismo[5]. Para desenvolver uma aplicação de RV normalmente são utilizados *engines* gráficos pois as *engines* representam uma camada de abstração no uso de serviços de renderização gráfica, para a criação do cenário virtual e de seu comportamento. Em outras palavras, os motores gráficos são *middlewares*, ou seja, um grupo de funções de biblioteca que abstraem a maioria dos detalhes de implementação de baixo nível, fornecendo ao desenvolvedor, uma interface para a programação e desenvolvimento em um nível mais alto de abstração[16].

A utilização de *engines* gráficas na criação em RV para Android são estudadas há vários anos [19][20][8], já que elas possibilitam um desenvolvimento fácil e ágil na criação de conteúdos em RV. Esta facilidade se dá devido à abstração do desenvolvedor

de trabalhar utilizando diretamente as APIs de renderização gráficas, como as *OpenGL*, *DirectX* ou *Vulkan*.

Em um cenário virtual comum, sem estereografia, apenas uma câmera é necessária para representar o cenário no *display*, mas há dificuldades que se acentuam quando é preciso adicionar o efeito estereográfico, pois, para realizar este efeito, adiciona-se uma câmera a mais no cenário virtual a uma pequena distância da primeira, que também, duplica a imagem no *display*. A utilização de duas câmeras provoca a sensação de profundidade pois simula o que acontece nos nossos olhos, que enxergam duas imagens em posições distintas para que o cérebro possa identificar a profundidade na diferença entre as duas. Segue a demonstração de exemplo na Figura 5 e na Figura 6.

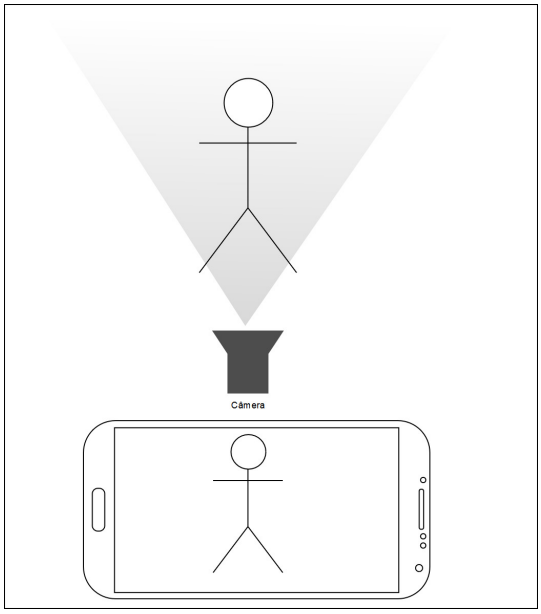

Figura 5: Sistema com uma câmera
Fonte: Autor (2018)

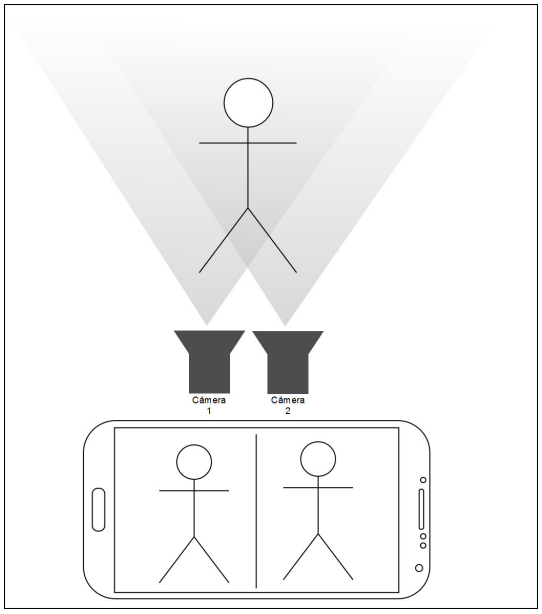

Figura 6: Sistema de câmeras em VR
Fonte: Autor (2018)

A duplicação de imagens geradas na câmera gera também, uma renderização duplicada no motor gráfico, sendo esta renderização uma das problemáticas ao se construir um motor gráfico para VR estereográfico em comparação a se desenvolver um motor gráfico comum. Somado a isso se reúne a dificuldade de se sincronizar os movimentos da cabeça do usuário as câmeras virtuais e a dificuldade de se adaptar a imagem as lentes dos *headsets* utilizados, já que é possível variar tanto as lentes do *headset* quanto o tamanho do display do celular.

Ao análisar diversos motores gráficos de RV com suporte ao Android, notamos que estes possuem algumas características que podem influenciar positivamente ou negativamente o desenvolvimento de uma aplicação de realidade virtual. Podemos visualizar algumas destas características de forma comparativa na Tabela 1 abaixo.

Tabela 1: Comparativo de Motores Gráficos de RV para Android

Recursos	Unity 5	Unreal Engine 4	LibGDX	Godot 3
Custo	Gratuito se a receita for menor que U$100.000	Gratuito se a receita for menor que U$3.000	Completamente Gratuito	Completamente Gratuito
Linguagens Suportadas	C#, JavaScript e Boo	C++, C#, GLSL, CG e HLSL	Java	GDScript, C# e C++
Licença de código	Código Fechado	Código Fechado	Código Aberto, Apache 2.0	Código Aberto, MIT
Plataformas	BlackBerry, Win Phone, Win, OS X, Android, iOS, Apple TV, PS3/4, PS Vita, Xbox 360, Xbox One, Wii U, Wii.	Windows, OS X Linux, Xbox 360/One, PS3/4, Wii U, Android, iOS, WinRT e PS Vita	Windows, Mac, Linux, Android, iOS, BlackBerry e HTML5	Windows, Linux, OSX, Android, iOS e Web
Suporte a RV	Suporte Nativo	Suporte Nativo	Suporte por plugins de terceiros	Suporte Nativo
Integração com apps preexistentes	Não	Não	Sim	Não
Necessita de uma outra IDE	Sim	Sim	Não	Sim

Fonte: Sites oficiais de cada motor gráfico [29][30][31][32] (2018)

É possível observar que essas *engines* são capazes de criar e manter um jogo ou aplicação com RV de forma satisfatória. Mas devido a algumas características em falta, pode ser que elas não sejam decisivas na escolha de um motor gráfico para RV.

3 METODOLOGIA

Como foi descrito anteriormente no capítulo de introdução, este trabalho foca em resolver alguns problemas destrinchados nos capítulos anteriores, relacionados a motores gráficos de realidade virtual. Que é basicamente o desenvolvimento de uma API de motor gráfico 3D de RV estereogŕafico para SBHMD, que seja de fácil utilização, que tenha uma performance aceitável e de fácil integração com aplicações preexistentes. Neste capítulo, será abordado como foi desenvolvidaa API, aprofundando-se na arquitetura, na leitura de geometria de objetos 3D e em como o processo de renderização na API gráfica foi implementado.

Também será abordado o processo de como foi planejado e realizado os testes, avaliando o desempenho e a UX (User Experience) da API desenvolvida comparada com outras APIs e frameworks para o desenvolvimento de realidade virtual em SBHMD para Android.

3.1 Metodologia da Pesquisa

Foi utilizado o Desenho de Pesquisa na Figura 7 para demonstrar a metodologia utilizada no desenvolvimento do trabalho. Descrito por MARCONI et al [34], o desenho de pesquisa tem como objetivo a demonstração de forma lógica e cronológica dos procedimentos metodológicos utilizados na criação e execução de uma pesquisa.

A pesquisa foi dividida em 3 etapas, onde cada etapa denota uma fase de tempo e especifica as tarefas realizada: onde na etapa inicial foi focado no estudo bibliográfico, a etapa seguinte foi determinada para o desenvolvimento e avaliação, e por último, a etapa final que foi destinada para a conversão para o formato de biblioteca e publicação.

3.1.1 Fase Exploratória

A Fase Exploratória, onde foi realizada uma carga maior de estudo bibliográfico e análise nas API e bibliotecas já existentes no mercado. Esta fase envolveu os seguintes passos:

- Identificação de autores, publicações e bibliotecas sobre construção e validação do processo de desenvolver uma RV: visando a criação da estrutura base do projeto, onde foram investigados os principais autores sobre RV, sendo realizado um estudo sobre construção de RV e formas de validar um projeto de RV.
- Busca e análise de Motores gráficos de RV para *Android*: para efeitos comparativos, foi realizado um mapeamento das principais bibliotecas e *API* existentes para Android sobre RV.

- Mapeamento das necessidades encontradas nos motores gráficos: visando a identificação de requisitos para uma RV, foi realizado um mapeamento das necessidades existentes nos motores gráficos do mercado.

- Estudo de processo dos critérios de análise para validação e avaliação: para validar e assegurar uma *QoE*, foi selecionado um critério de avaliação para o motor gráfico.

3.1.2 Prototipação, avaliação e análise

- Documentação inicial e elicitação de requisitos dos protótipos preliminares: Foi determinado uma documentação inicial para o projeto com os requisitos mínimos para um protótipo preliminar.

- Criação da arquitetura inicial do projeto: foi desenvolvida uma arquitetura que definisse uma base no funcionamento do motor gráfico.

- Identificação de bibliotecas e softwares externos para o auxílio no desenvolvimento do projeto: visando a criação facilitada do motor gráfico, foi selecionado algumas bibliotecas e ferramentas, tais como programa de modelagem e API de rastreamento posicional da cabeça do usuário, entre outras.

- Elaboração de cenário de teste para avaliação e validação: afim de averiguar o desempenho e experiência na utilização do motor gráfico, foi realizado um cenário que, hipoteticamente, utiliza-se todas as funcionalidades disponíveis no motor gráfico.

- Codificação do projeto: Foi realizado a programação do projeto.

- Execução do cenário de testes e avaliação de sua execução: visando avaliar o projeto, foi executado e analisado o seu desempenho no cenário de testes

3.1.3 Conversão para o formato de biblioteca

- Conversão para o formato *Android Archive Library Package* (AAR): foi realizado a conversão para o formato de biblioteca padrão do *Android* afim de facilitar a utilização por outros desenvolvedores.

- Preparação de documentação online e adição nos repositórios de bibliotecas online: Foi preparado a documentação e adição em repositórios online de bibliotecas *Android* (*JitPack*)

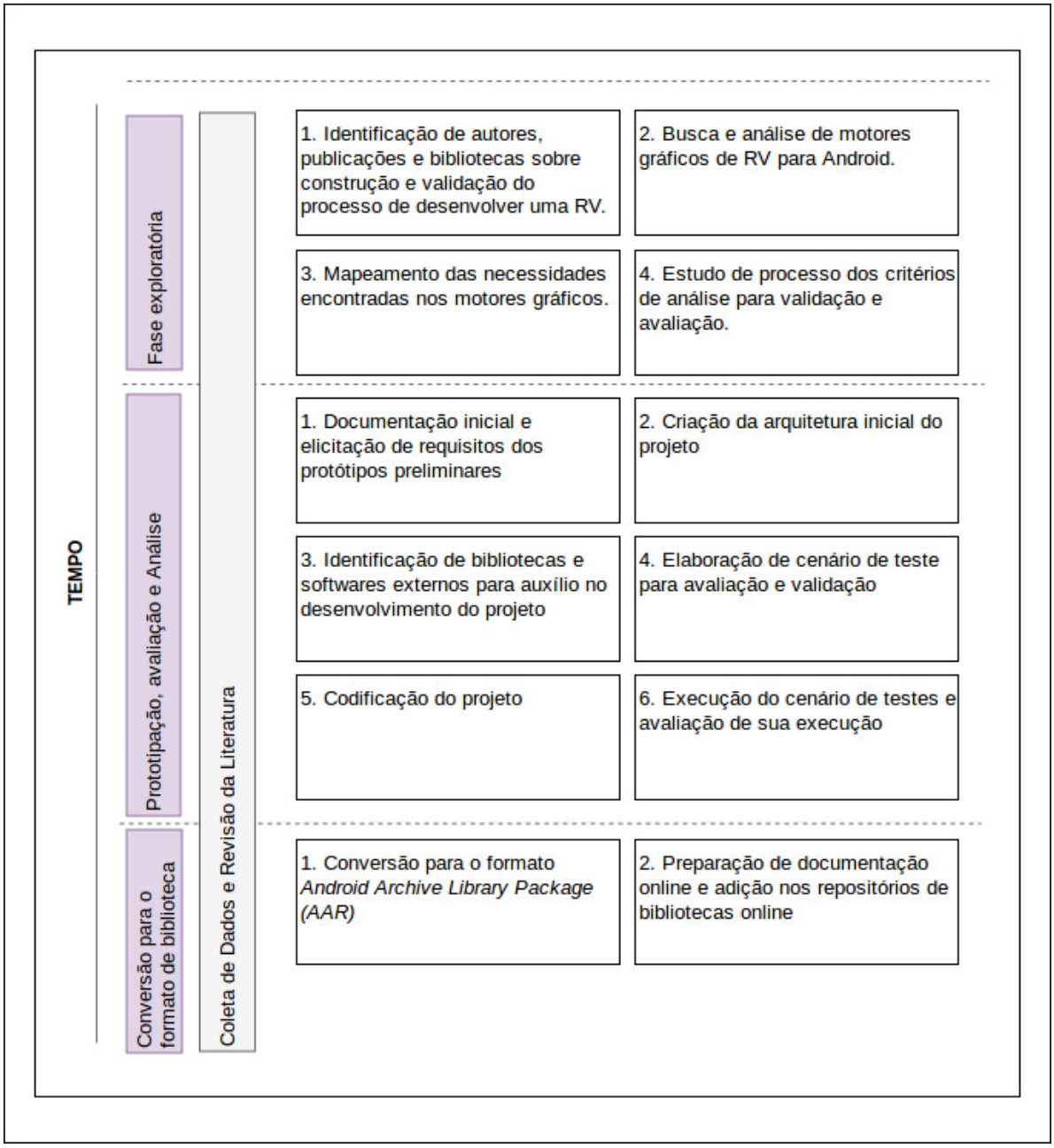

Figura 7: Desenho de Pesquisa
Fonte: Autor (2018)

3.2 Visão geral do motor gráfico AndVR

O AndVR foi desenvolvido utilizando a linguagem Java, inicialmente, foi pensado utilizar a linguagem C++, mas devido a grande oferta de bibliotecas em repositórios *Maven* e *JCenter* disponíveis, a facilidade de se desenvolver devido ao GC (*Garbage Colector*) e as otimizações feitas pela *Google* na Máquina Virtual ART (*Android RunTime*) não se faz necessidade emergente de se utilizar a linguagem C++ sobre a Java devido a performance. Principalmente quando comparamos a performance de Java a C++ nas versões *Android* e arquiteturas mais recentes como mostra Sims [33] no Gráfico 8 abaixo.

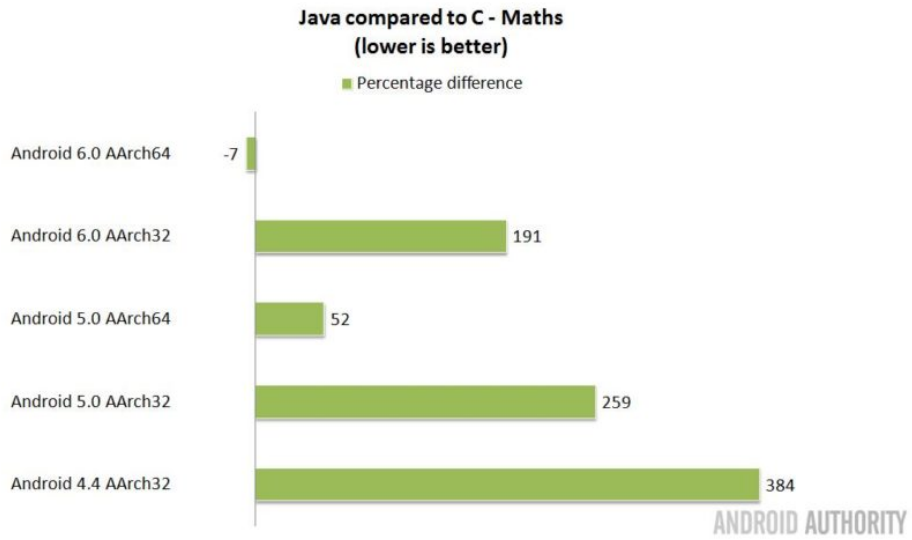

Figura 8: Comparativo entre Java e C
Fonte: ANDROID AUTHORITY (2018)

A arquitetura de funcionamento do motor foi baseado num sistema de Entidades e Componentes, onde uma entidade representa algo no cenário virtual e as componentes são as propriedades das entidades, como geometria 3D, textura, fonte de luz, transformações geométricas entre outras características que descrevem o que é a entidade e o que ela pode fazer. Este modelo de arquitetura é baseado na análogia ao paradigma de Programação Orientada a Objetos (POO), que possuem classes, atributos e objetos. Nesta analogia, a Entidade representa uma Classe, os Componentes representariam seus Atributos e a instanciação de uma entidade na execução seria a criação de um Objeto.

Além das entidades e componentes temos uma classe de carregamento de recursos chamada *GameResources*, onde se adiciona todas os arquivos de geometria 3D, áudio e textura, podendo assim, carregar todas as estruturas necessárias antes de abrir o cenário virtual. No código a ser implementado pelo usuário, é chamada a interface *GameUpdates*, que é responsável por implementar o método "gameFrame()", esse método é o encarregado por implementar o *game loop* do projeto, que é função da qual o jogo capta as interações do usuário, muda os estados do cenário virtual e faz modificações nas entidades. Podemos visualizar melhor o funcionamento do motor gráfico neste Diagrama de Componentes na Figura 9 abaixo:

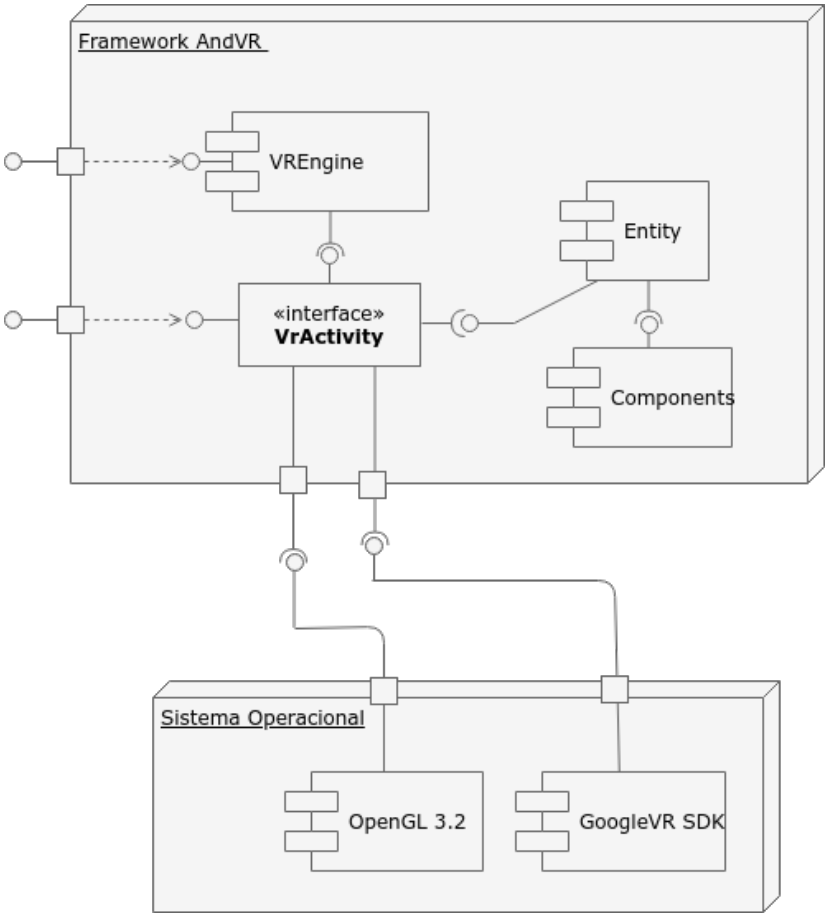

Figura 9: Diagrama de Componentes
Fonte: Autor (2018)

3.3 Renderização Gráfica no AndVR

No *Android*, há apenas duas *APIs* de renderização gráficas, a *OpenGL ES* e suas subversões, definida pelo *Kronos Group* [35] e a API Vulkan, também definida pelo *Kronos Group* [36]. Para a renderização das geometrias 3D no *AndVr*, é utilizado o *OpenGL ES* 3.2 devido ao suporte a versões mais antigas do *Android*, pois a *API Vulkan* só é suportada a partir da *API* 24 para cima e segundo o Statista [37], versões da API 24 ou maiores do Android não chegam a ultrapassam 29,6% dos smartphones ativos em Fervereiro de 2018.

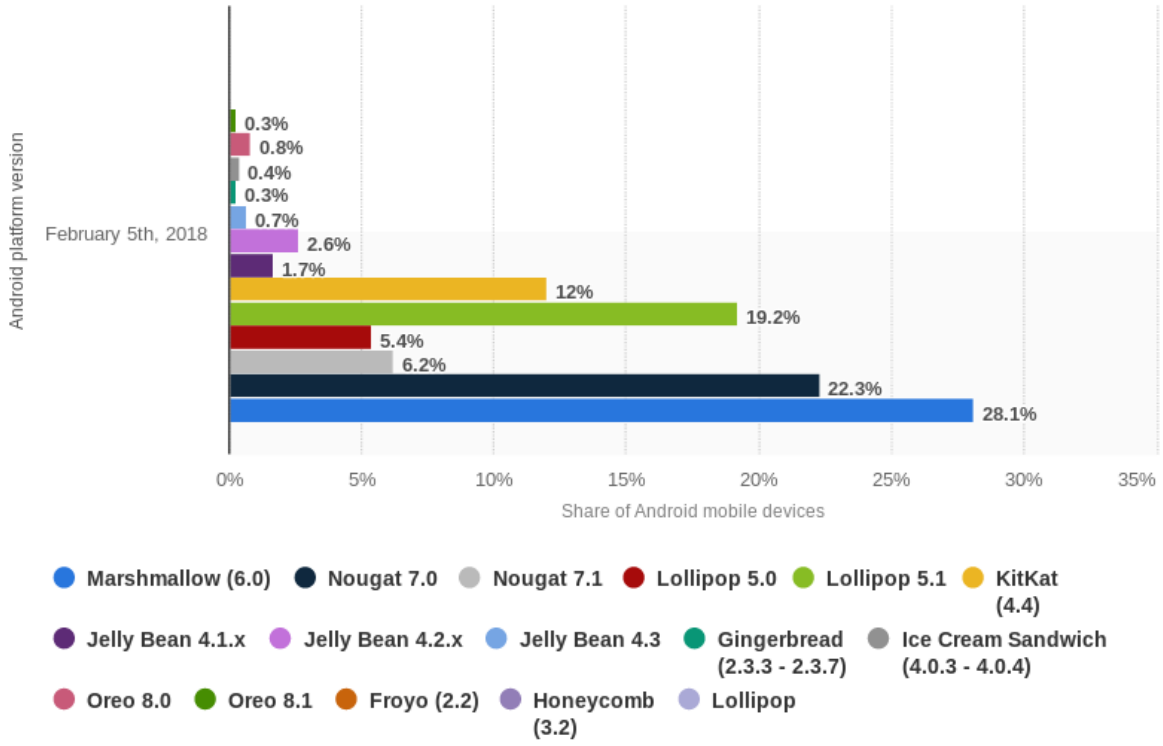

Figura 10: Marketshare das versões Android OS 2018
Fonte: STATISTA (2018)

No *AndVR*, o componente Model3D de uma entidade, é a classe incumbida da renderização das geometrias 3D utilizando a API de renderização *OpenGL ES*. O *Model3D* utiliza a geometria carregada previamente pela classe GameResource, no formato .OBJ *Wavefront*. Estas geometrias podem ser feitas em programas de modelagem 3D, como por exemplo, o *Blender*, o *3ds Max* ou o *SketchUp Pro*.

Na instanciação do *Model3D*, é realizada a alocação e carregamento da geometria do objeto na memória de vídeo, junto das referências das normais e coordenadas das texturas. Este processo é realizado usando *Vertex Buffer Object (VBO)*, o VBO garante que esses valores só sejam carregados na memória gráfica da GPU uma única vez, procedimento conhecido como *Non-Immediate-Mode Rendering*.

As texturas são realizadas usando o Componente *Texture*, nele é usado uma referência a uma imagem que também é previamente carregada com a classe *GameResource*. As imagens carregadas podem ser em qualquer formato aceito pelo *Android*, entre elas estão os formatos .JPEG, .GIF, .PNG, .BMP e o .WebP. Como o Componente *Texture* só possui a referência da imagem, é possível trocar a textura de um *Model3D* durante a execução do engine. Possibilitando assim uma maior dinamicidade na modelagem e execução do cenário virtual. Utilizando este método também é possível reaproveitar uma textura em vários *Model3D*, preservando assim, a utilização de memória.

Na execução de cada quadro, as entidades que possuem o componente *Model3D* são executadas e renderizadas sequencialmente. Este processo de execução é divido em três passos:

1. Inicialmente é executado as operações de modificação das entidades pelo usuário. Estas modificações podem ser movimento de entidades, rotações, adição ou remoção de entidades, translação de câmera, alterações na posição das iluminações entre outras operações que o usuário deseje que aconteça entre quadros.

2. Após isso, é executado as operações de processamento de componentes não gráficos da etapa 1, como por exemplo, ações de física, transformações geométricas das matrizes com as novas posições dos modelos, verificação de colisão entre outras operações.

3. E por fim é executado duas vezes a função de renderização, uma para cada bissecção do *display*, esta operação é necessária, pois no RV há duas câmeras separadas numa pequena distância, cada câmera exibe a imagem de uma bissecção do *display*, a diferença de posição das câmeras influenciam levemente na forma que a geometria é renderizada de diversas formas, como no seus formatos, como a iluminação reflete em suas faces para cada câmera ou como a distorção da câmera afetará a posição.

Este passo 3 é um dos maiores impactos de desempenho em aplicações de RV comparado a uma renderização comum, pois para cada bissecção, há uma matriz *View*, e todas as matrizes transformações de objetos no cenário precisam ser multiplicadas por ela. Além disso, devido a natureza do HMD, há sempre micro movimentos na cabeça do usuário que evitam o reaproveitamento da posição anterior de câmera (e de sua matriz *ViewMatrix*), afetando também o desempenho.

Devido a forma que a API OpenGL ES foi implementada, não é possível uma chamada *multithread* assíncrona a API, ou seja, utilizando todo potencial de paralelismo dos processadores com mais de um núcleo. Logo, as chamadas de desenho na tela realizadas pela API, são realizadas sequêncialmente, uma bissecção depois a outra. O Diagrama de Atividades na Figura 11 abaixo mostra o fluxos conduzidos pela execução do AndVR.

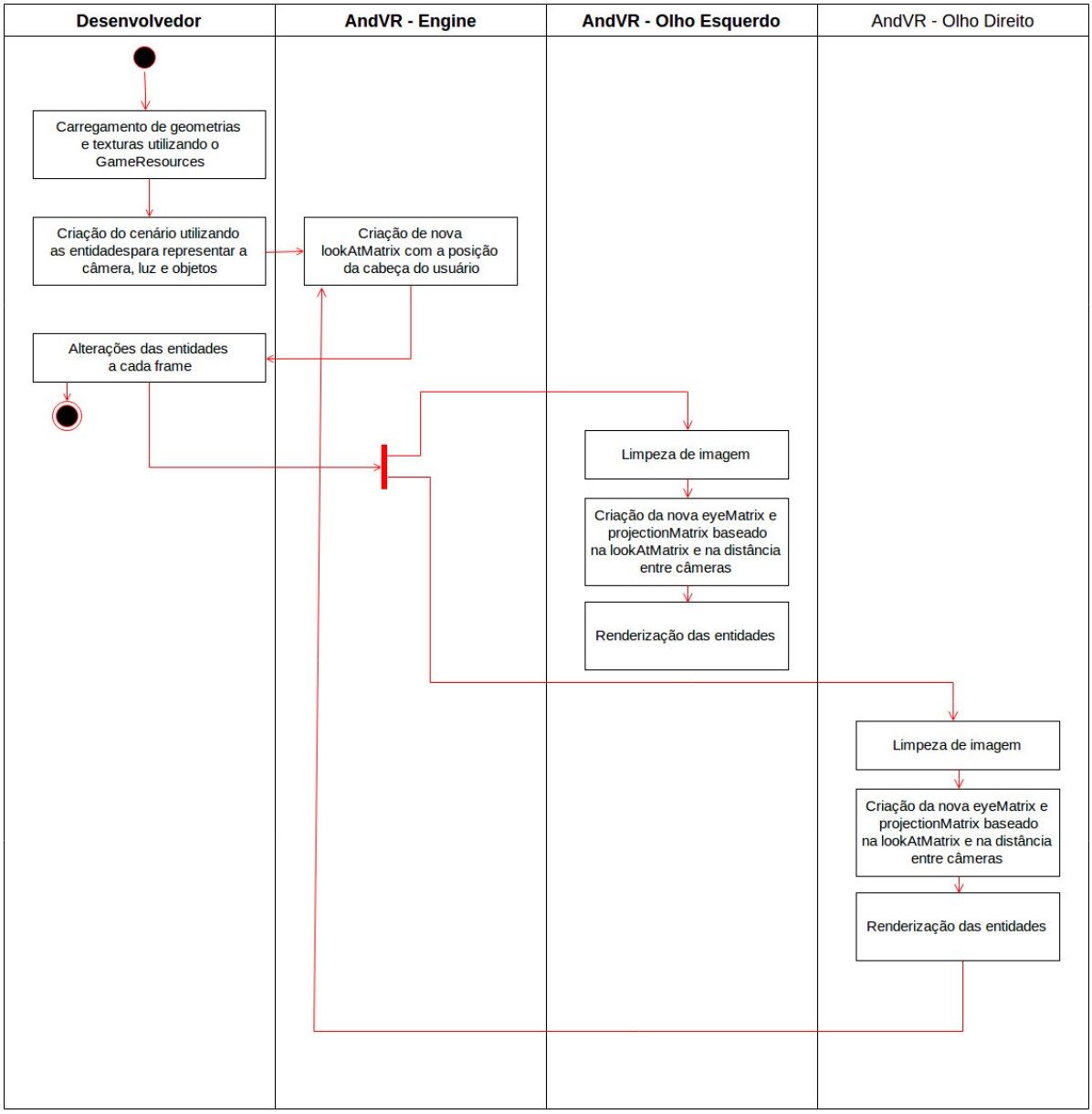

Figura 11: Diagrama de Atividades
Fonte: Autor (2018)

3.4 Processo de validação do AndVR

Para a realização do processo de validação da biblioteca AndVR foi utilizado as métricas de qualidade definidas por Hezel, aliado a reprodução de algumas demonstrações de cenários. As demonstrações que foram utilizadas são cenários com diversos objetos de tamanhos e posições aleatórias em um fundo de cor sólida. Podendo assim, ser averiguado minimamente, os seis critérios de Hezel, confirmando então, uma experiência de RV confortável.

Por fim, como trata-se de um motor gráfico, é necessário um critério de avaliação do desempenho nos quesitos de quadros por segundos e fluidez geral da renderização em

relação aos *smartphones* utilizados.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

A seguir, apresentam-se os trabalho realizado, demonstrando consigo, os resultados obtidos. Primeiramente, será descrito a forma que se foi realizado a aplicação demonstrativa, e após, será realizada a análise da demonstração a fim de se determinar a qualidade da realidade virtual utilizando os critérios definidos por Hezel el al.(1993).

4.1 Hardware utilizado

Foi utilizado de hardware um *smartphone Mi 5S Plus* da *Xiaomi*, modelo com processador Snapdragon 821 da *Qualcomm* de 4 núcleos, sendo 2 de 2,4GHz e 2 núcleos de 2,0GHz. De memórias, ele possui 6 *gigabytes* de memória RAM, 128 *gigabytes* de armazenamento. A placa gráfica integrada ao processador é a *Qualcomm Adreno* 530. A tela, é uma de 5.7 polegadas com resolução de 1920 por 1080 pixels.

Também foi testado no Moto X Force, com o processador *Snapdragon* 810 da *Qualcomm*, com 4 núcleos, sendo 2 de 2Ghz e 2 de 1.5Ghz. De memórias, ele possui 3 *gigabytes* de RAM e 64 *gigabytes* de armazenamento, tendo a placa gráfica a *Qualcomm Adreno* 430. A tela, é uma de 5.4 polegadas com resolução de 2560 por 1440 pixels.

E por último, o Galaxy S8, com o processador *Exynos* 8895 da *Samsung*, com 8 núcleos, sendo 4 de 2.3Ghz e 4 de 1.7Ghz. De memórias, ele possui 4 *gigabytes* de RAM e 64 *gigabytes* de armazenamento, tendo a placa gráfica a *Mali-G71 MP20*. A tela, é uma de 5.8 polegadas com resolução de 2960 por 1440 pixels.

O óculos utilizado foi o VR Box 2.0, que possui duas lentes 42 mm. Ele possui espaço para celulares de 4.7 a 6.0 polegadas.

4.2 A Aplicação Demonstrativa

Como o projeto foi desenvolvido em forma de biblioteca, ele foi facilmente importado utilizando o sistema de automação de compilação *Gradle*, sistema utilizado na compilação de aplicativos Android. Desta forma, foi desenvolvida uma aplicação demonstrativa a fim de averiguar a experiência da utilização do motor gráfico AndVR para utilização de realidade virtual.

Para a criação do cenário virtual, foi utilizado alguns modelos tridimensionais, em posições aleatórias em um raio ao redor da posição inicial do usuário. Estes modelos são 3 objetos *Wavefront*, com suas respectivas texturas, com variados tamanhos e complexidade poligonal. Foram utilizados estes 3 modelos originais para preencher 30 unidades no cenário. Um plano de fundo azul e um ponto de iluminação em movimento foi utilizado para poder visualizar o efeito de reflexão da iluminação nos modelos tridimensionais. Na

Figura 12 e Figura 13 é possível visualizar os cenários sendo executados em tempo real em um *smartphone*.

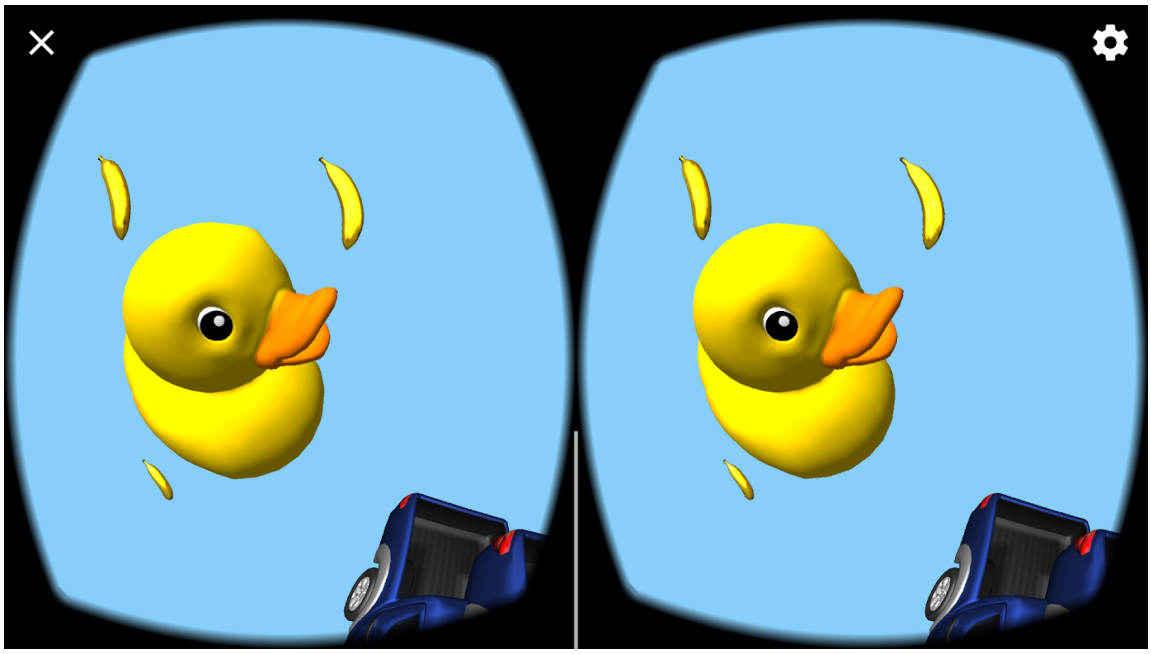

Figura 12: Cenário Demonstrativo com poucos polígonos
Fonte: Autor (2018)

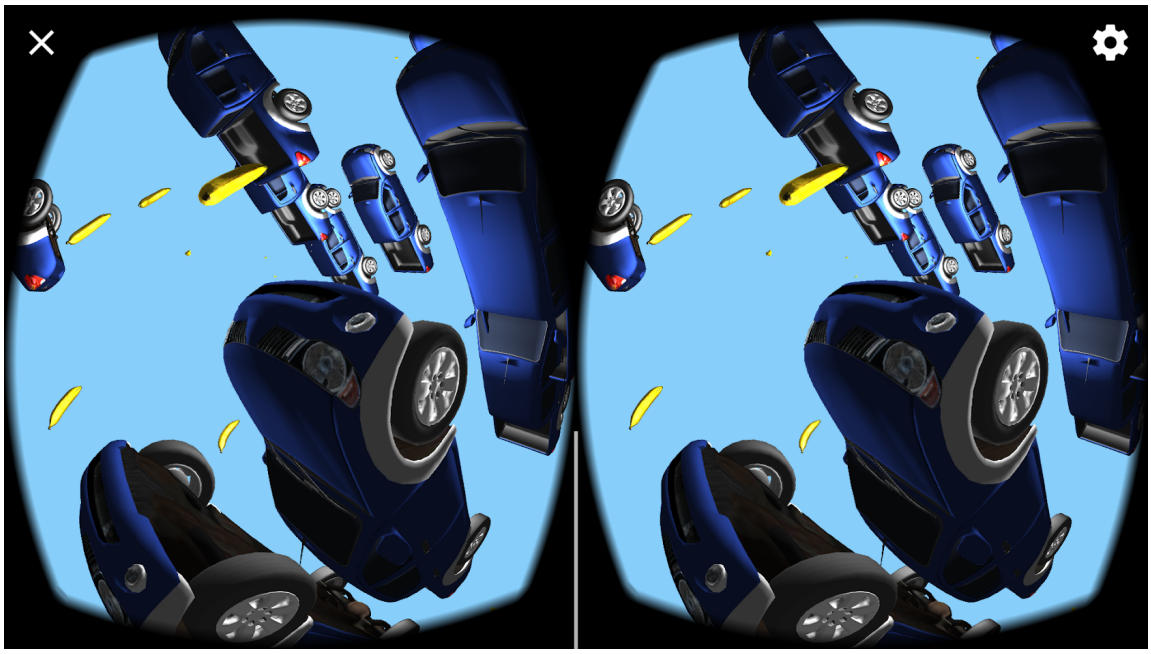

Figura 13: Cenário Demonstrativo com muitos polígonos
Fonte: Autor (2018)

Para locomoção no cenário, a fim de facilitar o acesso a melhores ângulos de visualização dos objetos, foi utilzado um controlador de jogo conectado ao celular por *Bluetooth*, uma forma de conexão sem fio, muito comum em *smartphones*.

4.3 Análise da Experiência e do Desempenho

Ao analisar a execução do AndVR no cenário de testes, podemos visualizar, na ótica do que era esperado, a renderização correta dos modelos tridimensionais apresentados. Como já foi descrito, foram utilizados 3 modelos tridimensionais para renderizar 30 elementos. Cada modelo com quantidade diferente de número de faces.

Como é possível ver na Figura 14 e Figura 15, o famoso modelo do pato de borracha *COLLADA Duck*, que apresenta 4212 faces triangulares, foi aberto tanto no programa de modelagem 3D *Blender*, quanto no *AndVR*, a fim de comparar se a renderização foi realizada com sucesso.

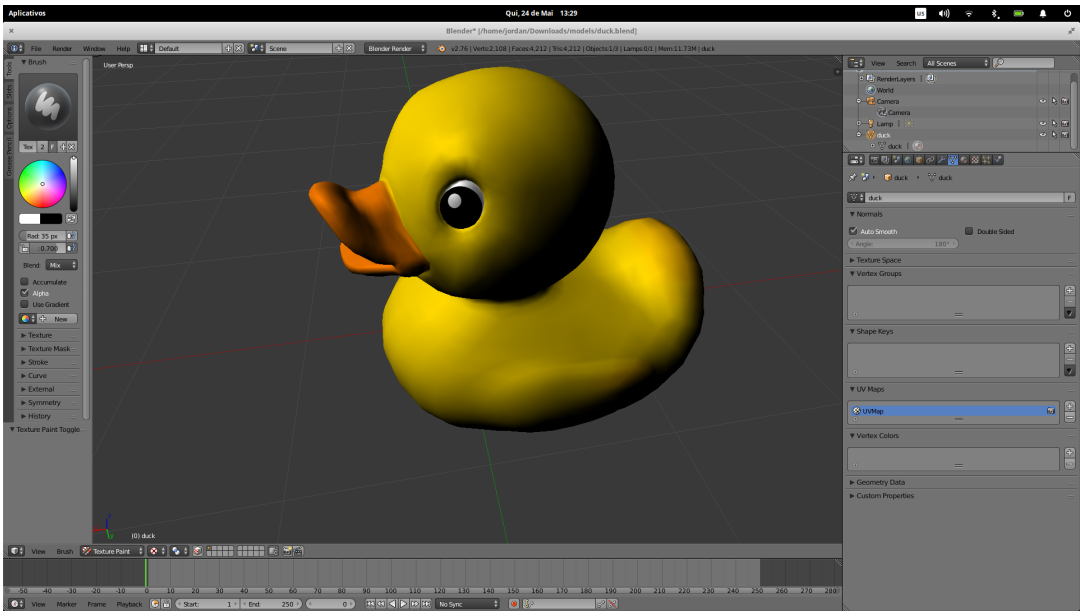

Figura 14: Collada Duck no Blender
Fonte: Autor (2018)

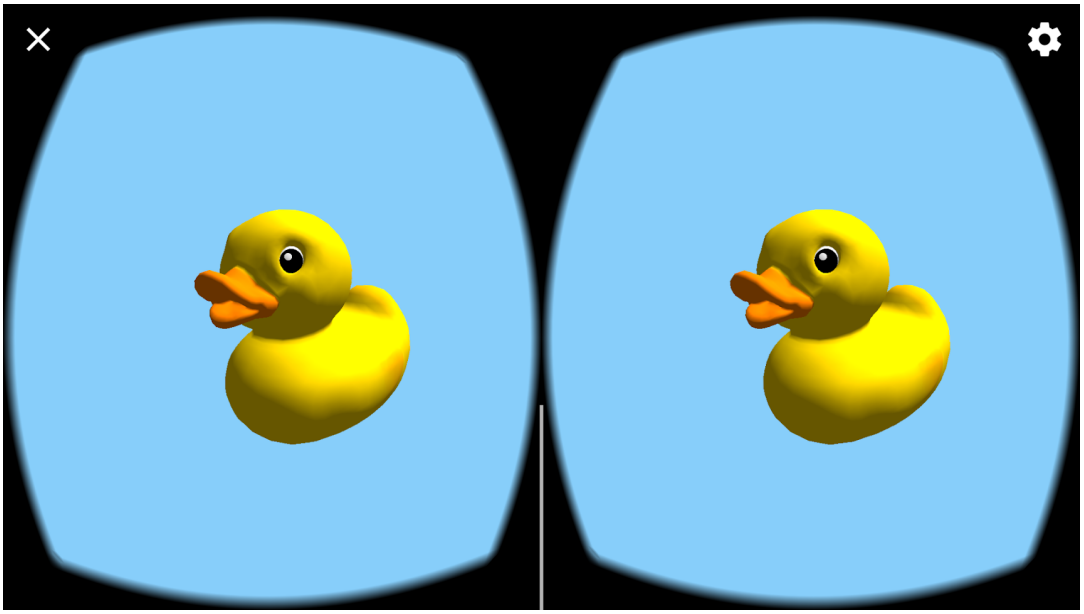

Figura 15: Collada Duck no AndVR
Fonte: Autor (2018)

Comparando com um modelo com mais faces na Figura 16 e Figura 17, o modelo da camionete possui 18536 faces. Neste modelo, as rodas e as palhetas são modelos separados da camionete principal e mesmo assim, conseguiu ser exibido corretamente. É notável também, a distorção modificada causada pela biblioteca utilizada, algo essencial para contrabalancear a distorção gerada pela lente dos óculos de realidade virtual.

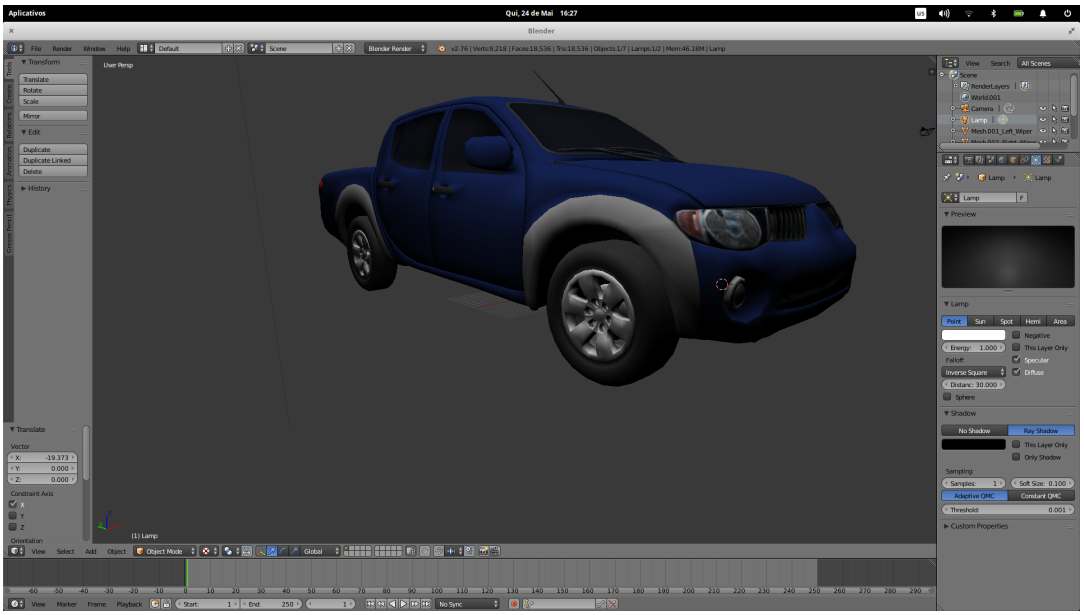

Figura 16: Camionete no Blender
Fonte: Autor (2018)

Figura 17: Camionete no AndVR
Fonte: Autor (2018)

Nos quesitos de experiência e desempenho, podemos observar que o AndVR executa o cenário criado, nos dispositivos testados, com uma taxa constante de 60 quadros por segundo. Sendo uma taxa ideal para uma experiência natural, como cita Hezel et al [23]. Para cada uma das 6 características de qualidade citadas, podemos analisar:

1. Tridimensão/Visão binocular: é presente.
2. Campo de Visão: também presente e editável de acordo com o óculos utilizado.
3. Resolução: depende da resolução do *smartphone* do usuário, a qualidade melhora proporcionalmente. Tendo uma melhoria considerável no *Samsung S8*, com resolução 2960 x 1440, em comparação aos outros dispositivos.
4. Abrangência de cores: nos dispositivos testados, as telas *IPS* e *AMOLED* presentes, exibem um espaço de cores que permitem uma qualidade visual aceitável.
5. Inexistência de distorções ou aberrações na imagem: as distorções identificadas foram causadas propositalmente, a fim de, contrabalancear a distorção gerada pela lente dos óculos de RV. Não há outras distorções ou aberrações nos modelos testados.
6. Acomodação de distância de visualização: no óculos testado, não houve incomodo na distância devido a lente e no contrabalanceamento de *software* utilizado.

Sendo assim, houve a execução confortável do cenário demonstrativo, e ele foi executado abrangendo as características definidas por Hezel et al.[23].

5 CONCLUSÕES E TRABALHOS FUTUROS

Diante do que foi exposto neste trabalho, podemos concluir que o AndVR possui a *capacidade* para ser aplicado ao desenvolvimento de aplicações e jogos de RV para Android. Podemos concluir também, que ao identificar os pontos positivos e negativos dos motores gráficos de RV, comparando com AndVR, chegamos ao desfecho que ele tem propriedades a adicionar no mercado de desenvolvimento Android.

Como direções futuras, podemos elaborar uma nova forma de renderização utilizando a nova API gráfica *Vulkan*, que por sua propriedade assíncrona, poderia renderizar as duas bisseções da tela paralelamente, podendo hipoteticamente, alcançar um desempenho superior. Outros possíveis trabalhos futuros seriam os de adicionar funcionalidades extras que são presentes em outros motores gráficos, tais como funções para facilitar o controle de translação de personagem, algumas implementações de colisões entre entidades, leitura de outros formatos de arquivos de geometria, entre outras funcionalidades.

Por fim, este estudo propôs um nova biblioteca de desenvolvimento de motores gráfico de RV para *Android*, visando sua utilização comercialização, auxiliando novos aplicações na industria de apps e jogos eletrônicos.

REFERÊNCIAS

[1] KIRNER, Cláudio et al. **Introdução à Realidade Virtual, Realidade Misturada e Hiper-realidade**. São Paulo: Senac-SP, 2004.

[2] HEILIG, Morton L et al. **SENSORAMA SIMULATOR**. United States patent US3050870A. 1962 .

[3] PSOTKA, Joseph. **Immersive training systems: Virtual reality and education and training.** Volume 23, Página 405-431, 1995.

[4] FROEHLICH, Mark A e AZHAR, Salman. Investigating virtual reality headset applications in construction. **Proceedings of the 52nd ASC International Conference**, Provo, UT. 2016.

[5] MISHRA, Prerna e SHRAWANKAR ,Urmila Comparison between Famous Game Engines and Eminent Games, **International Journal of Interactive Multimedia and Artificial Intelligence**, Volume 4, Páginas 69-77, Setembro, 2016

[6] OLSON, J. Logan et al. A design for a smartphone-based head mounted display. **Virtual Reality Conference (VR) 2011 IEEE**, Abril, 2011 .

[7] AMER, Ahmed et al. Affordable altered perspectives: Making augmented and virtual reality technology accessible. **Global Humanitarian Technology Conference (GHTC)**, Outubro, 2014 .

[8] WANG, Sa et al. A New Method of Virtual Reality Based on Unity3D. **18th International Conference on Geoinformatics**, Volume 16, Páginas 39-68, Junho, 2010.

[9] BROTHALER, Kevin et al. **OpenGL ES 2 for Android**. O'Reilly, 2013.

[10] Burdea, G.Coiffet, P., **Virtual Reality Technology**, John Wiley Sons, 1994.

[11] KOEFFEL, Christina et al. Using Heuristics to Evaluate the Overall User Ex-perience of Video Games and Advanced Interaction Games. **Evaluating user experience in games.** Springer London. Página 233-256, 2010.

[12] KIRNER, Cláudio. **Realidade Virtual e Aumentada**, Acesso em Março 2011, Disponível em <http://www.realidadevirtual.com.br>

[13] Sutherland, I.E. **Sketchpad: A Man-Machine Graphical Communication System**, PhD Thesis, MIT. Technical Report No. 574, University of Cambridge, UCAM-CL-TR-574, 1963.

[14] KIRNER, Cláudio et al. **Realidade Virtual e Aumentada: Aplicações e Tendências**. XIII SIMPÓSIO DE REALIDADE VIRTUAL E AUMENTADA, 2011.

[15] BISHOP, Lars et al. **Designing a PC Game Engine** IEEE Computer Graphics and Applications archive Volume 18, Páginas 46-53, 1998

[16] M. Lewis, J. Jacobson, "Game engines in scientific research - introduction," **Communications of the ACM**, Volume 45, no. 1, Páginas 27–31, 2002.

[17] RHEINGOLD, Howard. **Virtual Reality: The Revolutionary Technology of Computer-Generated Artificial Worlds-And How It Promises to Transform Society**, Agosto, 1992 .

[18] STEED, Anthony, at al. **Design and implementation of an immersive virtual reality system based on a smartphone platform. In 3D User Interfaces (3DUI)**,2013 IEEE Symposium, Página 43-46, 2013.

[19] SHIRATUDDIN, Mohd at al. **Utilizing a 3D game engine to develop a virtual design review system** Journal of Information Technology in Constructicon, Volume 16, Páginas 39-68. 2011

[20] JACOBSON, J, at al. **Game engine virtual reality with CaveUT.** Computer, Volume 38(4), Páginas 79-82. 2005.

[21] TRENHOLME, David at al. **Computer game engines for developing first-person virtual** Virtual Reality, Volume 12, Páginas 181-187. 2008

[22] CASTELVECCHI, Davide. **Low-cost headsets boost virtual reality's lab appeal.** Vol 533, Página 153, Maio, 2016.

[23] TANAKA et al. **Virtual Reality Environment Design of Managing Both Presence and Virtual Reality Sickness.** Journal of PHYSIOLOGICAL ANTHROPOLOGY and Applied Human Science, Volume 23, Página 313-317.

[24] HEZEL et al. **Head Mounted Displays for Virtual Reality.** Proceedings of SPIE - The International Society for Optical Engineering, Página 47, 1993.

[25] MACLSAAC, Dan **Google Cardboard: A virtual reality headset for $10?** The Physics Teacher, Volume 53, Página 125, 2015.

[26] AMER, Ahmed**Affordable altered perspectives: Making augmented and virtual reality technology accessible** IEEE Global Humanitarian Technology Conference (GHTC 2014), Página 603-608, 2014

[27] DOLAN, Brian. **Timeline of Apple "iPhone"Rumors (1999–Present)**. Disponível em: <https://www.fiercewireless.com/wireless/timeline-apple-iphone-rumors-1999-present>, FierceWireless . Acesso em: 29 de Março de 2018.

[28] ARTHUR, Charles. **The history of smartphones: timeline**. Disponível em: <https://www.theguardian.com/technology/2012/jan/24/smartphones-timeline>, TheGuardian . Acesso em: 29 de Março de 2018.

[29] UNITY. Unity 3D Website. Disponível em: <http://unity3d.com>. Acesso em: 30 de Março de 2018.

[30] UNREAL. Unreal Engine Website. Disponível em: <https://www.unrealengine.com/en-US/features>. Acesso em: 30 de Março de 2018.

[31] LIBDGX. LibGDX BadLogic Website. Disponível em: <https://libgdx.badlogicgames.com>. Acesso em: 30 de Março de 2018.

[32] GODOT. Godot Website. Disponível em: <https://godotengine.org>. Acesso em: 30 de Março de 2018.

[33] SIMS, Garry. Java vs C app performance – Gary explains. Disponível em:<https://www.androidauthority.com/java-vs-c-app-performance-689081/> Acesso em 31 de Março de 2018.

[34] MARCONI et al. **Fundamentos de metodologia científica**. São Paulo: Atlas 5. Ed., 2003.

[35] OpenGL ES. The Standard for Embedded Accelerated 3D Graphics. Disponível em:<https://www.khronos.org/opengles/> Acesso em 31 de Março de 2018.

[36] Vulkan. The Standard for Embedded Accelerated 3D Graphics. Disponível em:<https://www.khronos.org/opengles/> Acesso em 31 de Março de 2018.

[37] Statista . Distribution of Android operating systems used by Android phone owners in February 2018, by platform versions. Disponível em:<https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/> Acesso em 31 de Março de 2018.