UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

THÁRSIS SALATHIEL DE SOUZA VIANA

UMA BIBLIOTECA PARA INTERAÇÃO COM OBJETOS VIRTUAIS EM APLICAÇÕES DE REALIDADE AUMENTADA PARA DISPOSITIVOS MÓVEIS

QUIXADÁ – CEARÁ

2017

THÁRSIS SALATHIEL DE SOUZA VIANA

UMA BIBLIOTECA PARA INTERAÇÃO COM OBJETOS VIRTUAIS EM APLICAÇÕES DE REALIDADE AUMENTADA PARA DISPOSITIVOS MÓVEIS

Monografia apresentada no curso de Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Ciência da Computação.
Área de concentração: Computação.

Orientador: Dr. Cristiano Bacelar de Oliveira

Coorientador: Dr. Rubens Fernandes Nunes

QUIXADÁ – CEARÁ

2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

V668b Viana, Thársis Salathiel de Souza.
Uma biblioteca para interação com objetos virtuais em aplicações de realidade aumentada para dispositivos móveis / Thársis Salathiel de Souza Viana. – 2017.
48 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2017.
Orientação: Prof. Dr. Cristiano Bacelar de Oliveira.
Coorientação: Prof. Dr. Rubens Fernandes Nunes.

1. Realidade aumentada. 2. Biblioteca (computação). 3. Visão computacional. 4. Interação homem-máquina. I. Título.

CDD 004

THÁRSIS SALATHIEL DE SOUZA VIANA

UMA BIBLIOTECA PARA INTERAÇÃO COM OBJETOS VIRTUAIS EM APLICAÇÕES DE REALIDADE AUMENTADA PARA DISPOSITIVOS MÓVEIS

Monografia apresentada no curso de Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Ciência da Computação.
Área de concentração: Computação.

Aprovada em: ____/____/_____

BANCA EXAMINADORA

Dr. Cristiano Bacelar de Oliveira (Orientador)
Universidade Federal do Ceará – UFC

Dr. Rubens Fernandes Nunes (Coorientador)
Universidade Federal do Ceará - UFC

Dr. Paulo de Tarso Guerra Oliveira
Universidade Federal do Ceará - UFC

Aos meus pais.

AGRADECIMENTOS

A minha família, principalmente meus pais e minhas irmãs, por todo apoio e incentivo.

Ao Prof. Dr. Cristiano Bacelar de Oliveira, pela excelente orientação. E por ter ministrado a disciplina que me motivou a realizar esse trabalho.

Ao Prof. Dr. Rubens Fernandes Nunes, pela co-orientação.

Ao Prof. Dr. Paulo de Tarso Guerra Oliveira participante da banca examinadora, pelo tempo, pelas valiosas colaborações e sugestões.

Aos professores Luis Rodolfo Rebouças Coutinho e Marcos Antonio de Oliveira pela orientação nas bolsas que participei.

Aos professores Ticiana Linhares Coelho da Silva, Marcos Antonio de Oliveira, Enyo José Tavares Gonçalves e a Mário Sérgio Rodrigues Falcão Júnior pela contribuição para minha vida acadêmica.

A todos meus colegas e amigos da universidade, em especial para meus colegas de turma, por todo apoio e solidariedade.

A todos que participaram da minha formação.

“Nada é tão inseguro ou instável quanto a fama
de poderio não fundada sobre a própria força”
(Maquiavel, O Príncipe)

RESUMO

É notável o rápido crescimento de aplicações que utilizam a tecnologia de realidade aumentada. Atualmente essa tecnologia está presente mesmo em dispositivos móveis. Porém, as aplicações que a utilizam frequentemente se limitam a projeção de objetos virtuais, com os quais o usuário não consegue interagir. Assim, o objetivo deste trabalho é propor uma biblioteca que auxilie as aplicações de dispositivos móveis que utilizam a tecnologia de realidade aumentada, no desenvolvimento da interação com objetos virtuais. Neste trabalho é apresentada a maneira que o usuário pode interagir fazendo uso da biblioteca, além de uma discussão sobre a eficiência e precisão dessa interação. Por fim, são mostradas algumas aplicações desenvolvidas com o uso da biblioteca, como por exemplo uma versão em realidade aumentada do jogo de corrida do minigame Brick Game e um piano em realidade aumentada.

Palavras-chave: Realidade aumentada. Biblioteca (computação). Visão computacional. Interação homem-máquina.

ABSTRACT

It is evident the rapid growth of application using the technology of augmented reality. Nowadays this technology is present even in mobile devices. However, the applications that use it often are limited to the projection of virtual objects, which the user can not interact with. Thus, the objective of this work is to propose a library that supports the applications of mobile devices that use augmented reality technology, in the development of interaction with virtual objects. This work presents the way the user can interact using the library, as well as a discussion about the efficiency and accuracy of this interaction. Finally, some applications developed with the use of the library are shown, such as an augmented reality version of the minigame racing game Brick Game and a piano in augmented reality.

Keywords: Augmented reality. Library (computing). Computer vision. Man-machine interaction.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
TUIs	Tangible User Interfaces
JAR	Java ARchive
APK	Android Package
ARR	Android Archive
HMD	Head-mounted display
UML	Unified Modeling Language
FPS	Frames Per Second

SUMÁRIO

1 INTRODUÇÃO

A realidade aumentada permite que o usuário veja o mundo real com objetos virtuais integrados a ele, complementando a realidade em vez de substituí-la completamente, dando a impressão para o usuário que objetos reais e virtuais coexistem no mesmo espaço. Essa tecnologia possui potencial para aplicações em várias áreas, algumas já exploradas, como por exemplo: visualização médica, fabricação e reparo de produtos, anotação e visualização, planejamento de caminho para um robô, aviação militar e entretenimento (AZUMA, 1997).

Algumas bibliotecas auxiliam no trabalho de desenvolver uma aplicação que utilize a realidade aumentada, como as bibliotecas: ARToolKit[1], Vuforia[2], Wikitude[3], ARCore[4], entre outras. Um fato a se destacar é que geralmente essas bibliotecas possibilitam integração com diversas plataformas, como: Android, iOS e Unity.

Apesar dos avanços de pesquisadores na área de realidade aumentada, a maioria das pesquisas se concentram na área de rastreamento, que consiste em posicionar o objeto sobre a cena real (BILLINGHURST et al., 2015) (ZHOU; DUH; BILLINGHURST, 2008). A interação com os objetos virtuais normalmente é colocada em segundo plano. Esse é o caso de algumas bibliotecas de realidade aumentada, como as citadas anteriormente. Apesar de algumas oferecerem métodos para facilitar a interação com objetos virtuais, o principal foco está na projeção do objeto virtual no mundo real.

Assim, as aplicações desenvolvidas com uso exclusivo dessas bibliotecas por muitas vezes acabam se limitando a projeção de objetos virtuais. Caso métodos para interação com objetos virtuais fossem oferecidos, por outra biblioteca por exemplo, os aplicativos poderiam tirar melhor proveito do real potencial oferecido pela realidade aumentada. A possibilidade de interação pode levar a maior motivação por parte das pessoas ao uso dessa tecnologia (NUNES et al., 2006).

Atualmente, é notável a rápida popularização de dispositivos móveis. Tais dispositivos fornecem viabilidade para diversos tipos de aplicação, entre essas, aplicações com foco na realidade aumentada. A capacidade de integração da realidade aumentada a dispositivos móveis proporciona um aumento na quantidade de pessoas que podem ser atingidas por essa tecnologia. É importante ressaltar que a integração com dispositivos móveis nem sempre é

[1] Disponível em: <https://artoolkit.org/>. Acesso em: 28 abr. 2017.
[2] Disponível em: <https://vuforia.com/>. Acesso em: 17 out. 2017.
[3] Disponível em: <https://www.wikitude.com/>. Acesso em: 19 out. 2017.
[4] Disponível em: <https://developers.google.com/ar/>. Acesso em: 19 out. 2017.

tratada nas propostas existentes.

Considerando o contexto atual, o objetivo deste trabalho é desenvolver uma biblioteca que visa auxiliar aplicações de dispositivos móveis, que utilizam a tecnologia de realidade aumentada, oferecendo diversos métodos para interação com os objetos virtuais. Tendo em vista que as aplicações de realidade aumentada geralmente já utilizam alguma biblioteca para realizar a projeção dos objetos virtuais, a biblioteca desenvolvida neste trabalho pode ser vista como uma complementar, que ao invés de auxiliar na projeção, auxilia na interação com os objetos virtuais. Esta biblioteca tem como característica ser independente da projeção, assim pode ser utilizada independentemente do método utilizado para projetar os objetos virtuais.

Os capítulos deste trabalho estão organizados como: o Capítulo 2 apresenta um resumo sobre os conceitos necessários para realização deste trabalho; o Capítulo 3 apresenta os trabalhos relacionados; o Capítulo 4 apresenta a biblioteca desenvolvida detalhadamente; o Capítulo 5 apresenta os resultados de desempenho e precisão dos métodos de interação propostos, além de algumas aplicações desenvolvidas com auxílio da biblioteca; por fim, o Capítulo 6 apresenta as considerações finais sobre este trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

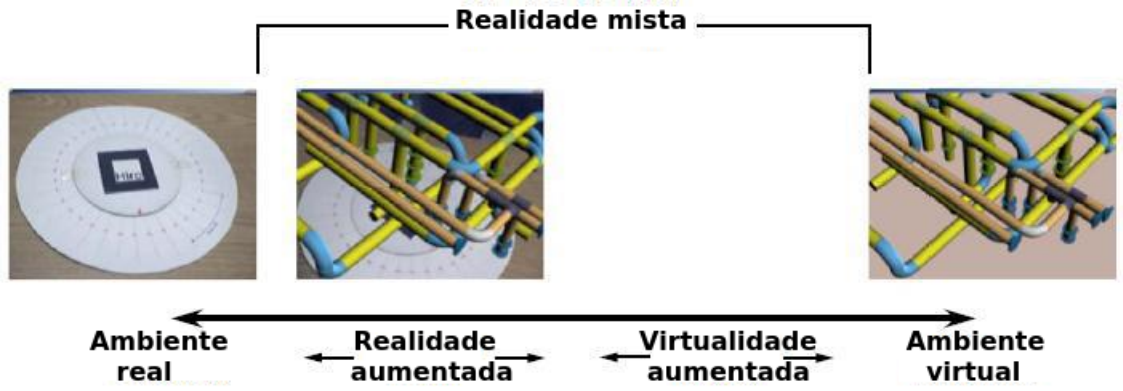
A seguir são apresentados os principais conceitos relacionados ao presente trabalho.

2.1 Realidade aumentada

A realidade aumentada é uma técnica que consiste na sobreposição de objetos virtuais tridimensionais a um ambiente físico real, visualizado através de um dispositivo tecnológico (ELISEU, 2017). A realidade aumentada começou a ser conceituada nos anos 60, na década de 1990 foi categorizada como uma subárea da realidade virtual. Enquanto a realidade virtual transporta completamente o seu utilizador para outro ambiente, a realidade aumentada mantém o utilizador no mesmo ambiente (ELISEU, 2017). Ou seja, na realidade virtual o utilizador não consegue ver o mundo real, ele está totalmente imerso em um ambiente virtual, que substitui o mundo real. Enquanto que na realidade aumentada o ambiente real não é completamente substituído, mas sim complementado com objetos virtuais que coexistem com objetos reais.

A Figura 1 ilustra bem onde está a realidade aumentada em relação a realidade virtual. Na extrema esquerda está um ambiente real e na extrema direita está um ambiente completamente virtual. Entre as duas extremidades estão ambientes resultantes da mistura de ambiente real e ambiente virtual, chamados de ambientes de realidade mista, onde a realidade aumentada se encontra (ELISEU, 2017).

Figura 1 – Continuum da realidade mista



Fonte – Wang e Dunston (2006)

O projeto mostrado em (BILLINGHURST; KATO; POUPYREV, 2001) é um bom exemplo para ilustrar essas diferentes classificações. O projeto possui como principal componente um livro com imagens e textos. Sem auxílio de tecnologia, o livro pode ser utilizado como livro

normal (ambiente real). Com auxílio de um *display* de realidade aumentada, o projeto apresenta mais duas formas de utilização. O livro pode ser utilizado como um livro que contém objetos virtuais anexados às páginas (realidade aumentada), e o *display* de realidade aumentada pode funcionar como um imersor a um ambiente virtual (realidade virtual).

2.2 Realidade aumentada tangível (Tangible AR)

Realidade aumentada tangível se origina do conceito de interfaces de usuário tangíveis (Tangible User Interfaces ou TUIs). Interfaces de usuário tangíveis são interfaces onde o usuário pode manipular informação digital com objetos físicos (ZHOU; DUH; BILLINGHURST, 2008). O uso de objetos físicos pode facilitar a interação do usuário, pois determinado objeto pode ter sua função descoberta de forma intuitiva devido ao seu uso familiar pelo usuário.

Essa forma de interface pode ser integrada à realidade aumentada, assim a projeção dos objetos virtuais poderia ser combinada com a entrada de objetos físicos (ZHOU; DUH; BILLINGHURST, 2008). Normalmente em realidade aumentada esse conceito está associado ao uso de marcadores, de forma que o marcador seria o objeto físico capaz de manipular os objetos virtuais. Geralmente a posição, rotação e a escala do marcador definem a posição, rotação e escala do objeto virtual. Assim, ao manipular o marcador o usuário está também manipulando o objeto virtual. Utilizando essa estratégia é possível definir alguns métodos de interação baseada nos marcadores, que são objetos físicos. Por exemplo, ao aproximar dois marcadores que contêm dois objetos virtuais associados, um dos objetos poderia desaparecer.

2.3 Biblioteca

Bibliotecas têm como função dar suporte na execução de uma tarefa, fornecendo solução para determinado problema genérico de uma área, através de funções, classes, métodos e tipos especializados (PIZETTA, 2014). Desta forma, o usuário da biblioteca não precisa partir do zero ou reinventar soluções já conhecidas para determinados problemas. Para utilizar uma biblioteca junto a sua aplicação o usuário deve importar a biblioteca ao projeto da aplicação, assim a biblioteca vira um módulo da aplicação.

É possível que uma biblioteca utilize outra biblioteca como auxiliar. Por exemplo, uma biblioteca pode ter o objetivo de exibir objetos virtuais sobre um marcador. Para isso é

necessário primeiramente encontrar o marcador na cena. Isso poderia ser feito com auxílio de outra biblioteca que seria capaz de encontrar um marcador na cena. Desta maneira o problema inicial seria reduzido a desenhar um objeto virtual sobre marcador encontrado.

Uma biblioteca em Java geralmente fornece apenas classes e métodos, através de um arquivo JAR (Java ARchive). Já uma biblioteca Android é igual a um módulo de aplicativo, podendo conter tudo que é necessário para a criação de um aplicativo, inclusive código-fonte, arquivos de recursos e um manifesto do Android. Entretanto a compilação não resulta em um arquivo APK (Android Package), como em um aplicativo, e sim em um arquivo ARR (Android Archive).

2.4 API (Application Programming Interface)

Embora haja similaridade entre o conceito de API e de biblioteca, uma API está em um nível mais próximo do usuário, com pouca necessidade de implementação extra. A API oferece rotinas para acesso a um software ou plataforma web (PIZETTA, 2014). Uma API pode servir como interface com a biblioteca, facilitando assim a utilização da biblioteca por parte do usuário.

A API deste trabalho está baseada no controle de eventos do Java, principalmente com componentes SWING [1]. Um evento nesse contexto é a mudança de estado de um objeto. Eventos são gerados por alguma interação com um componente de interface gráfica, como um clique de botão, mover o mouse, selecionar um item de uma lista, etc. No caso da API desenvolvida durante este trabalho, a interação a partir de algum dos métodos definidos em 4.3 irá gerar um evento.

Para tratar os eventos, definimos dois agentes. São eles: a fonte e o ouvinte. A fonte é responsável por gerar os eventos e prover informações sobre eles. Essa tarefa é executada pela biblioteca. O ouvinte é responsável por gerar a resposta a um evento. O ouvinte espera que algum evento aconteça e então executa uma determinada função. A assinatura dessa função é definida por uma interface da API, entretanto o processamento que essa função executará é definido pelo usuário.

Este método é eficiente pois separa a lógica de geração de eventos da lógica responsável pela ação a ser tomada no acontecimento de algum evento. Dessa maneira o papel do usuário é apenas definir a resposta para determinado evento. Para que algum ouvinte passe a

[1] Disponível em: <https://docs.oracle.com/javase/tutorial/uiswing/events/intro.html>. Acesso em: 20 out. 2017.

receber os eventos, é necessário registra-lo em um determinado objeto. Assim esse ouvinte passa a tratar eventos apenas daquele objeto.

2.5 Visão computacional

Visão computacional é a transformação de dados obtidos de imagens ou vídeos em uma decisão ou nova representação, para atingir determinado objetivo (BRADSKI; KAEHLER, 2008). Uma imagem capturada por uma câmera salva em um computador é apenas uma matriz de números, onde cada número representa por exemplo uma cor ou intensidade. Detectar padrões, reconhecer objetos, detectar movimento, entre outros, são problemas difíceis de se resolver dispondo apenas de uma entrada composta de uma matriz de números.

Uma biblioteca bastante conhecida e utilizada para o desenvolvimento de aplicações na área de visão computacional é a OpenCV[2]. O número estimado de downloads ultrapassa os 14 milhões. Apesar de ser escrita em C/C++ a OpenCV possui interfaces para C++, C, Python e Java. A biblioteca desenvolvida por esse trabalho faz uso da OpenCV para o auxílio da implementação dos métodos de interação.

[2] Disponível em: <https://opencv.org/>. Acesso em: 10 set. 2017.

3 TRABALHOS RELACIONADOS

Alguns autores já buscaram soluções para facilitar a interação com objetos virtuais em um ambiente de realidade aumentada. Lee et al. (2004) desenvolvem um modelo de interação para aplicações de realidade aumentada baseado em componentes. Já Nunes et al. (2006) desenvolvem uma biblioteca para facilitar a interação, que deve ser usada em conjunto com a ARToolKit, uma biblioteca especifica para a projeção dos objetos virtuais. Por fim, Shim et al. (2016) propõem um sistema de interação via gestos, com um conjunto de operações que podem ser aplicadas a um objeto virtual.

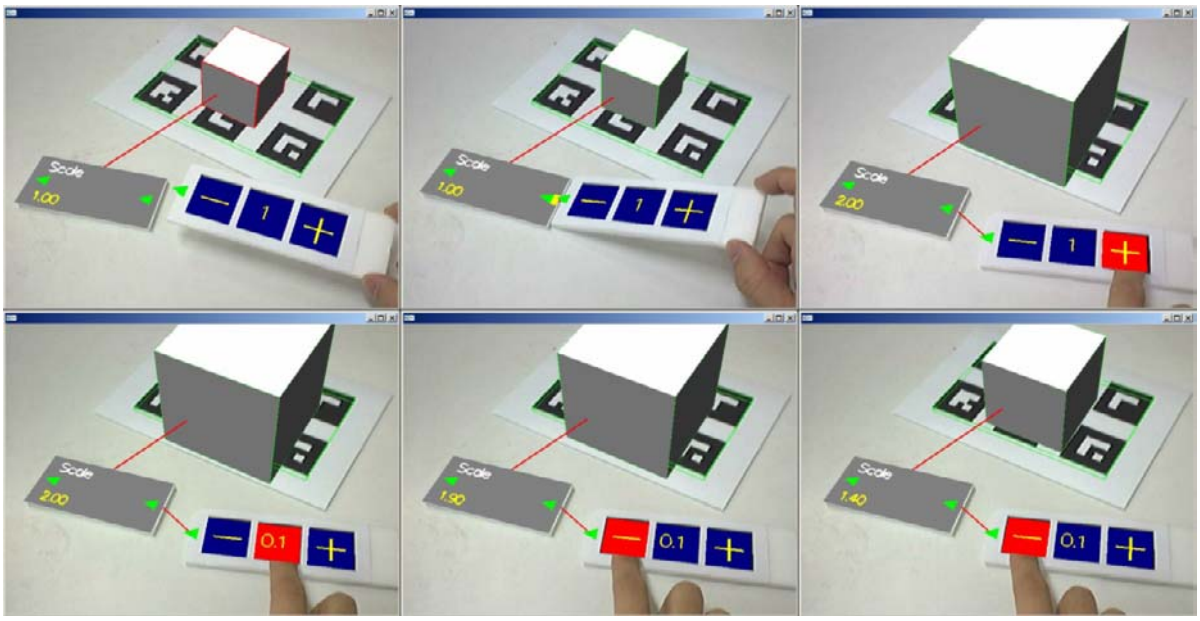
Esses trabalhos foram escolhidos porque propõem um modelo para as interações em realidade aumentada, ou ainda uma maneira de facilitar o processo de interação com os objetos virtuais. A seguir esses trabalhos são descritos mais detalhadamente.

3.1 Immersive Authoring of Tangible Augmented Reality Applications

O artigo de Lee et al. (2004) propõe um modelo de interação baseado em componentes. Cada componente possui um conjunto de atributos associados, dependendo do seu tipo. Os componentes são divididos em três tipos: objetos físicos, objetos virtuais e caixas lógicas. Tanto objetos físicos quanto objetos virtuais possuem atributos semelhantes, relacionados a suas características físicas, como posição e orientação. Porém, objetos físicos são objetos reais, por isso seus atributos podem ser apenas lidos mas não modificados, diferentemente de objetos virtuais, que podem ter seus atributos alterados. Uma caixa lógica é um componente que tem como função relacionar atributos entre dois componentes, definindo assim a lógica da aplicação. Também são definidas as operações possíveis a serem realizadas sobre componentes, são elas: criar, destruir, modificar e conectar (ou linkar). Neste ponto o artigo apresenta certa relação com o artigo de Poupyrev et al. (2002).

O usuário pode utilizar acessórios, que nada mais são que marcadores, os quais podem ser manipulados fisicamente para realizar as operações e alterar atributos dos componentes, como mostrado na Figura 2. Utilizando as operações, a leitura e alteração dos valores dos atributos dos componentes e o componente que define a lógica é possível construir uma cena e definir a interação que o usuário poderá ter com objetos da cena. Desta forma, o usuário não precisa ter conhecimento em programação para conseguir desenvolver uma aplicação de realidade aumentada que possua alguma forma de interação, bastando que ele manipule os

Figura 2 – Utilização dos acessórios para modificar a propriedade de escala do objeto virtual



Fonte – Lee et al. (2004)

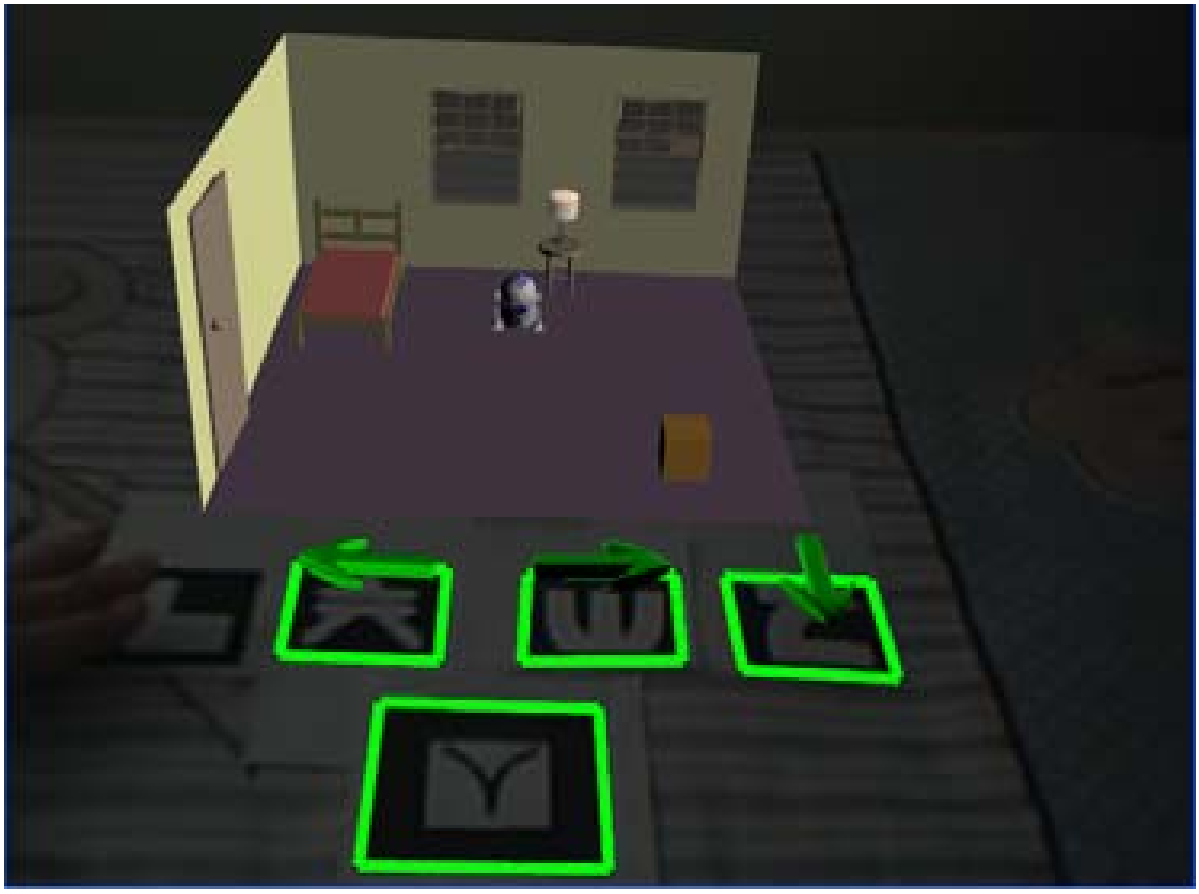
acessórios para definir a cena e sua interação.

Apesar da facilidade de uso, o sistema pode não ser ideal em alguns cenários. O caso mais evidente está em definir um comportamento complexo, a caixa lógica normalmente associa apenas dois atributos para definir o comportamento daquela interação. Para um comportamento complexo, a definição através das caixas lógica também seria complexa. Por exemplo, é difícil imaginar como definir que um objeto virtual realize uma sequencia de movimentos, apenas associando dois atributos.

3.2 ARInteraction - Biblioteca para Interação com Objetos Virtuais para Realidade Aumentada

O trabalho de Nunes et al. (2006) apresenta uma biblioteca integrada ao ARToolKit, para facilitar o desenvolvimento de interações em aplicações de realidade aumentada. A biblioteca funciona da seguinte forma: um arquivo é definido para associar um objeto virtual e um marcador, a biblioteca então detecta se o marcador está visível na cena, se não, significa que o usuário está interagindo com o marcador. Ou seja, é um método de interação baseado na oclusão dos marcadores. Desta forma, se um usuário cobrir a visão de um marcador pela câmera (colocando a mão sobre ele por exemplo), a ação de interação associada aquele marcador é executada, como mostrado na Figura 3.

Figura 3 – Cobrindo a visão do marcador para que sua ação seja executada, que neste caso é movimentar o robô virtual para cima



Fonte – Nunes et al. (2006)

O grande problema da biblioteca é que esta trabalha com apenas um método de interação possível (por oclusão), limitando bastante o domínio das aplicações suportadas. Além disso, essa técnica de interação possui um problema inerente: todos os marcadores devem estar visíveis para a câmera no caso em que não há nenhuma interação. Quando pensamos em uma câmera de um dispositivo móvel, que normalmente está em constante movimento, garantir isso pode ser um problema. Além disso, as associações de marcadores e objetos virtuais são definidos em um arquivo de configuração. Esta tarefa pode se tornar complexa, tanto para usuários programadores quanto para usuários que não tenham conhecimento em programação, pois é preciso definir hierarquias entre os objetos virtuais, os objetos virtuais que devem ser carregados e os objetos que possuem colisão e interação.

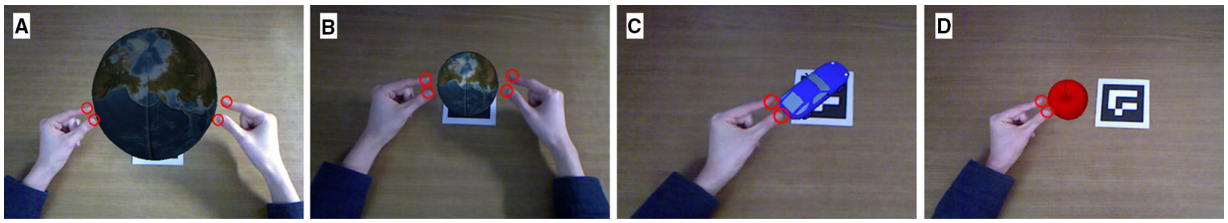
3.3 Gesture-based interactive augmented reality content authoring system using HMD

O artigo (SHIM et al., 2016) propõem um sistema em que usuários que não possuem conhecimento em programação consigam facilmente interagir com objetos virtuais em um ambiente de realidade aumentada. O sistema funciona com um display que fica acoplado a cabeça (*head-mounted display* ou HMD), uma câmera RGB-D, e o uso de marcadores. O usuário pode interagir de três formas: com a utilização de gestos próximos ao objeto virtual, utilizando uma interface virtual interativa ou através de interações tangíveis.

Com uso da câmera RGB-D o sistema faz a detecção das pontas dos dedos dos usuários, essa informação é utilizada na interação via gestos e via interface. Na interação via gestos o usuário executa ações realizando gestos diferentes com a ponta dos dedos. Quatro ações são disponíveis: aumentar a escala (Figura 4a), diminuir a escala (Figura 4b), rotacionar (Figura 4c) e mover (Figura 4d). Para utilizar a interface o usuário utiliza a ponta do dedo como um ponteiro de mouse para realizar um clique em um botão virtual. É possível tanto escolher o objeto que será relacionado ao marcador (Figura 5a,b) quanto executar as quatro operações descritas anteriormente (Figura 5c).

Por fim, também é possível realizar uma interação tangível, ou seja utilizar objetos físicos para realizar a interação. Nesse caso os objetos físicos são os marcadores. São duas formas de interação possíveis: aproximando os marcadores (Figura 6a,b) ou ocluindo (cobrindo a visão) um marcador (Figura 6c).

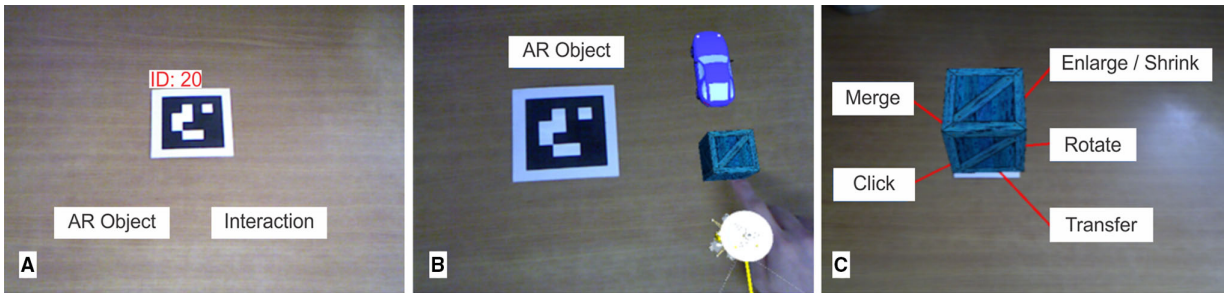
Figura 4 – Interação via gestos



Fonte – Shim et al. (2016)

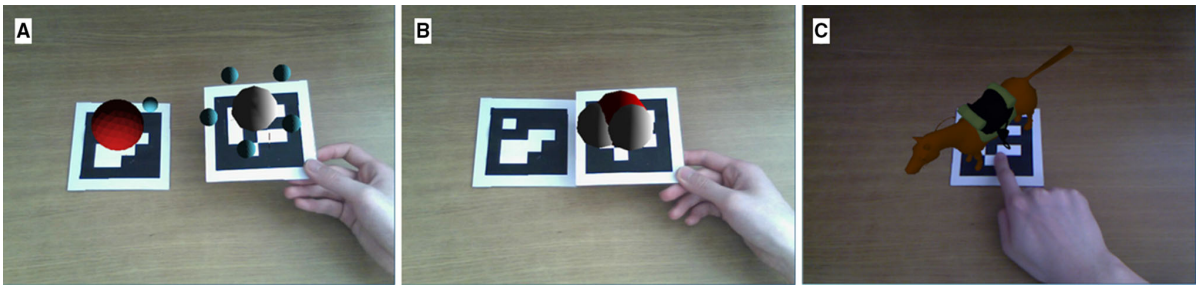
Apesar de possuir bons resultados, esse sistema apresenta algumas limitações, a principal talvez seja a necessidade de uma câmera RGB-D. Com essa câmera é possível obter não só uma matriz de pixels que formam a imagem mas também a profundidade de cada pixel. Porém dispositivos móveis e *notebooks* por exemplo, geralmente possuem apenas uma câmera capaz de capturar a matriz de pixels.

Figura 5 – Interação via interface virtual



Fonte – Shim et al. (2016)

Figura 6 – Interação tangível



Fonte – Shim et al. (2016)

Outro ponto a se ressaltar é que essa é uma solução voltada a pessoas ordinárias, que não possuem conhecimento em programação. Isso faz com que o sistema tenha que limitar as operações realizadas ao detectar uma interação, como: aumentar ou diminuir a escala, rotacionar e mover o objeto virtual. Caso esse sistema fosse voltado para programadores, poderíamos ter algo como: o programador define uma operação, por exemplo, destruir o objeto, e então associa essa operação a um determinado gesto. Assim seria possível executar qualquer operação que fosse programaticamente possível ao ocorrer determinado gesto.

4 INTERATIVAAR

Chamamos a biblioteca desenvolvida neste trabalho de InterativaAR. A InterativaAR tem como função principal simplificar a implementação da interação em uma aplicação que utilizar a tecnologia de realidade aumentada. Assim, uma aplicação que já é capaz de projetar objetos virtuais, pode fazer uso da biblioteca para realizar a interação com os objetos virtuais de forma mais simples. Geralmente as aplicações já utilizam uma biblioteca para realizar a projeção dos objetos virtuais. Caso essas aplicações necessitem tratar a interação com esses objetos, elas podem utilizar a InterativaAR combinada com a biblioteca que realiza a projeção.

Para auxiliar o usuário, a biblioteca fornece métodos de interação com objetos virtuais. Chamamos de método de interação a maneira com que o usuário pode interagir com um objeto virtual. Para garantir que a biblioteca funcione corretamente é necessário cumprir alguns requisitos técnicos, que estão listados na Seção 4.2. Na Seção 4.3 apresentamos mais detalhadamente os métodos de interação que estão disponíveis para o usuário da biblioteca. O usuário pode fazer uso desses métodos através da API. A utilização da API está detalhada na Seção 4.4.

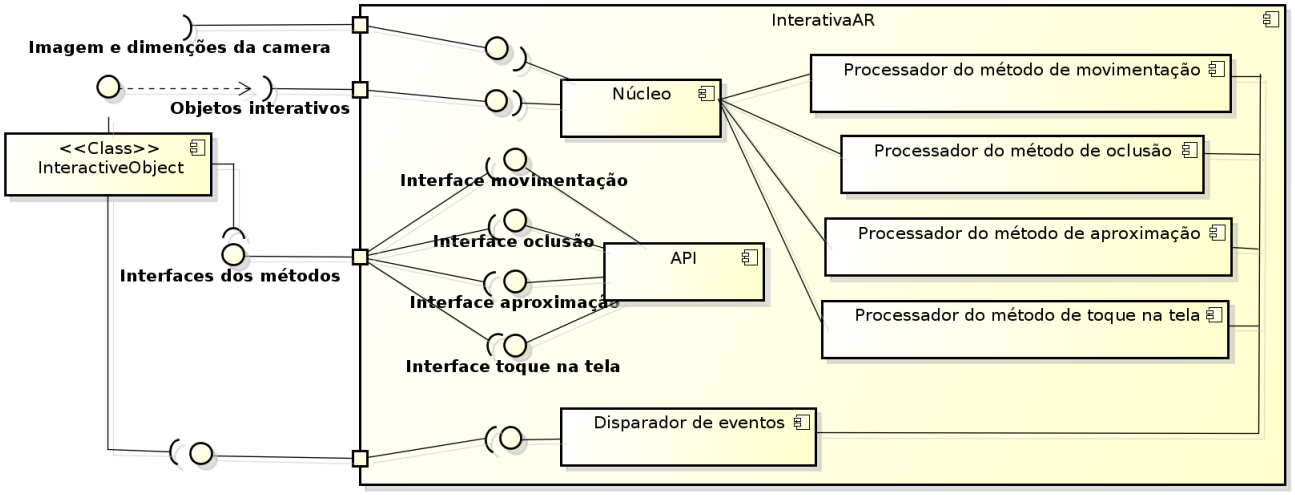
4.1 Visão geral

Para visualizar a arquitetura da biblioteca foi utilizado o diagrama de componentes, que é um diagrama UML (Unified Modeling Language). Um componente pode ser entendido como uma parte (lógica ou física) do software. Além de mostrar os componentes esse diagrama também mostra as interfaces, portas e relações. O diagrama de componentes da InterativaAR é mostrado na Figura 7.

O principal componente para o usuário da biblioteca é o InterativeObject, que nesse caso é uma classe que representa um objeto virtual com o qual se pode interagir utilizando um dos métodos de interação. Para que o objeto virtual se torne interativo é preciso que ele possua um ouvinte (ou *listener*) que implementa uma das interfaces fornecidas pela API.

O usuário da biblioteca deve fornecer os objetos virtuais, com seus respectivos ouvintes, além da imagem da câmera e as dimensões em que a imagem é mostrada ao usuário. Esses recursos são passados para o núcleo da biblioteca, que por sua vez aciona apenas os processadores dos métodos necessários. Por exemplo, se os objetos possuem apenas ouvintes que implementam a interface de oclusão, apenas o processador do método de oclusão será ativo.

Figura 7 – Arquitetura da InterativaAR



Fonte – Próprio autor

Os processadores ativos testam repetidamente se alguma interação ocorreu, caso algum detecte uma interação, ele aciona o disparador de eventos que por sua vez notifica ao ouvinte do objeto virtual sobre a interação ocorrida. O ouvinte então executa uma função definida pelo usuário.

Um ponto a ser ressaltado é que a biblioteca é independente da projeção dos objetos virtuais, pois os métodos de interação são calculados utilizando apenas as propriedades dos objetos interativos e a imagem da câmera. Isso abre a possibilidade desta biblioteca ser compatível com várias bibliotecas responsáveis por realizar a projeção de objetos virtuais, que podem ou não utilizar marcadores por exemplo.

Apesar da biblioteca desenvolvida neste trabalho não ser focada no uso de marcadores, é possível utilizá-los e criar interações tangíveis facilmente. Para definir um objeto virtual nesta biblioteca basta que se defina posição, rotação e escala. Então, se o usuário é capaz de obter a posição, rotação e escala de um marcador, basta que ele associe essas propriedades às respectivas propriedades do objeto virtual. Assim, o objeto virtual vai estar associado ao marcador, e alguns métodos de interação desta biblioteca irão se assemelhar à metáfora da interface tangível, ou seja será possível utilizar objetos físicos para realizar a interação com os objetos virtuais.

Uma aplicação capaz de projetar objetos virtuais sem fazer o uso de marcadores também consegue utilizar esta biblioteca. Entretanto as propriedades de posição, rotação e escala do objeto virtual são obtidos a partir do plano em que o objeto virtual será desenhado. Nesse

caso não se pode dizer que a interação irá se assemelhar a metáfora da interface tangível, pois não haverá nenhum objeto físico associado ao objeto virtual.

4.2 Requisitos técnicos

A seguir são listados os requisitos técnicos que as aplicações devem seguir para que a InterativaAR funcione corretamente.

1. O dispositivo móvel deve rodar o sistema operacional Android.
2. A aplicação deve possuir acesso a imagem da câmera usada na captura do ambiente e às dimensões que esse vídeo é mostrado ao usuário (altura e largura).
3. A aplicação deve ser capaz de ler a matriz de projeção utilizada para converter um ponto tridimensional em uma posição de um pixel na tela.
4. Os objetos virtuais criados devem ser representados pela classe *InteractiveObject*, uma classe própria da InterativaAR. As transformações também devem ser aplicadas utilizando os métodos disponibilizados pela classe.

Caso a aplicação já implemente a projeção dos objetos virtuais essas restrições são facilmente satisfeitas, a aplicação terá acesso a todos os recursos definidos na lista. Caso o usuário utilize alguma biblioteca para essa função, o cumprimento dessas restrições, principalmente os itens 2 e 3, vão depender se essa biblioteca fornece acesso aos recursos necessários. É esperado que seja possível acessar a câmera e a matriz de projeção pela maioria das bibliotecas de realidade aumentada. Criar objetos utilizando a classe *InteractiveObject* não deve ser um problema, pois mesmo que a biblioteca de projeção possua uma classe própria para os objetos basta criar um objeto do tipo *InteractiveObject* que possua os mesmos atributos de posição, rotação e escala do objeto usado pela biblioteca de projeção.

4.3 Métodos de interação

A seguir são detalhados os métodos de interação fornecidos pela biblioteca. Um método de interação é uma maneira que o usuário tem de realizar uma interação com um objeto virtual. Os métodos de interação escolhidos foram baseados em outros trabalhos que também propõem facilitar a interação com objetos virtuais.

Para que um objeto interativo possa tratar as interações, ele deve possuir um ouvinte para o método em que se pretende interagir com esse objeto. Quando a biblioteca detectar uma

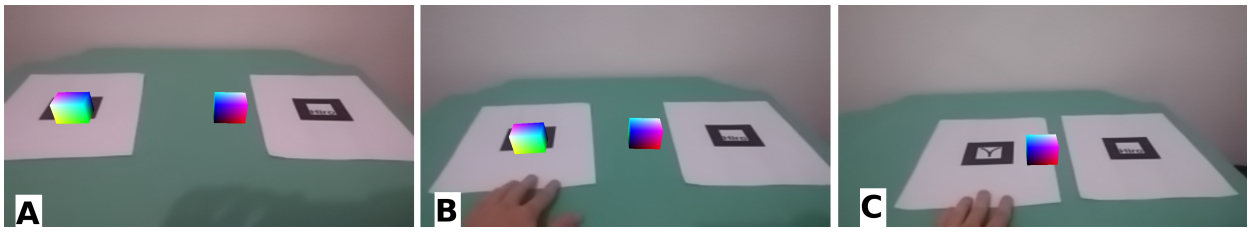
interação por determinado método, a ação definida no respectivo ouvinte do objeto é executada. Por exemplo, se um objeto interativo possui um ouvinte de aproximação, quando ocorrer uma interação por aproximação envolvendo o objeto, a ação definida no ouvinte será executada.

4.3.1 Aproximação

Esse método de interação consiste em checar se dois objetos virtuais estão próximos, caso estejam, alguma ação é executada. Vários artigos relacionados a interação em um ambiente de realidade aumentada utilizam essa abordagem (SHIM et al., 2016), (POUPYREV et al., 2002) e (REGENBRECHT; WAGNER; BARATOFF, 2002).

Para exemplificar, considere a Figura 8a, que mostra um cenário construído com auxílio da InterativaAR onde dois cubos estão relativamente distantes um do outro. O cubo da direta (na Figura 8a) possui um ouvinte para o método de interação de aproximação. Os cubos são aproximados continuamente (Figura 8b), até que atingem a distância mínima e a ação definida no ouvinte do cubo da direita é disparada, que neste caso é destruir o cubo da esquerda (Figura 8c). O usuário é responsável por definir a proximidade mínima que irá disparar a ação, que também é definida por ele.

Figura 8 – Método de interação por aproximação utilizando a InterativaAR



Fonte – Próprio autor

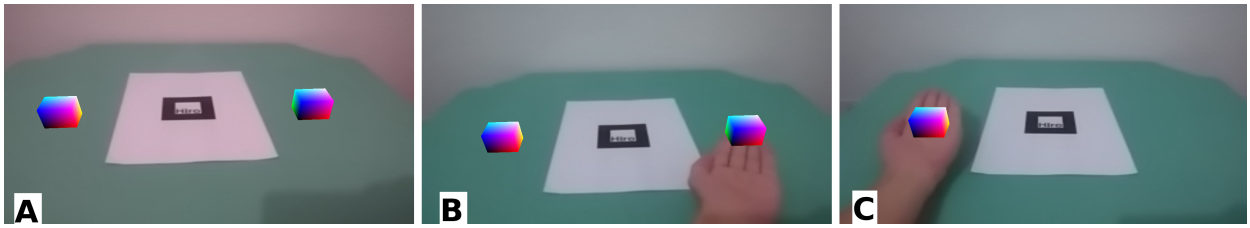
4.3.2 Oclusão

Nesse método de interação, uma ação, definida pelo usuário, é disparada quando um objeto virtual é ocluso (coberto) por algum corpo. Para exemplificar, considere a Figura 9a, que mostra um cenário construído com auxílio da InterativaAR onde temos dois cubos. O cubo da esquerda possui um ouvinte de oclusão, enquanto o cubo da direita não possui ouvinte algum. Ao ocluir (cobrir) o cubo da direita, como esperado, nada acontece (Figura 9b). Entretanto ao realizar a mesma ação no cubo da esquerda, a ação do ouvinte é disparada, que nesse caso é

destruir o cubo da direita (Figura 9c).

É importante ressaltar que para esse método de interação funcionar corretamente é necessário que o objeto que irá sofrer a oclusão seja um objeto estático e não sofra nenhuma transformação. O objetivo aqui é imitar o comportamento de um botão. Esse método é citado em vários artigos como: (NUNES et al., 2006), (SHIM et al., 2016) e (LEE; BILLINGHURST; KIM, 2004). Entretanto, nesses artigos ocorre a oclusão do marcador. No caso da biblioteca desenvolvida por esse trabalho é verificada a oclusão no objeto virtual.

Figura 9 – Método de interação por oclusão utilizando a InterativaAR



Fonte – Próprio autor

4.3.3 Movimentação

Esse método foi inspirado em uma interação por gestos, como a mostrada em (SHIM et al., 2016). Entretanto para conseguir um resultado semelhante a Shim et al. (2016), é necessário que o dispositivo possua uma câmera RGB-D. Algo que não está disponível para a maioria dos dispositivos móveis atualmente. Para superar essa restrição a biblioteca lida apenas com a direção do gesto feito. Assim, dada uma direção do movimento (direita, esquerda, cima, baixo) o usuário pode decidir que ação será tomada.

Para exemplificar, considere a Figura 10a, que mostra um cenário construído com auxílio da InterativaAR onde um cubo parado possui um ouvinte de movimentação. Ao executar um movimento para direita passando pelo cubo (Figura 10b) é executado a ação definida no ouvinte que nesse caso é movimentar o cubo para direita(Figura 10c), de maneira análoga, ao executar um movimento para esquerda (Figura 11a,b) o cubo se movimenta para esquerda, retornando ao seu lugar de origem (Figura 11c).

Para essa interação funcionar corretamente é necessário que a câmera se mantenha estática, ou com pouca movimentação, durante a realização do gesto. Para garantir isso é possível utilizar o giroscópio do dispositivo para verificar se ele está se movimentando, assim o gesto só é capturado quando o dispositivo (consequentemente a câmera) está estático.

Figura 10 – Movimentação para direita

Fonte – Próprio autor

Figura 11 – Movimentação para esquerda

Fonte – Próprio autor

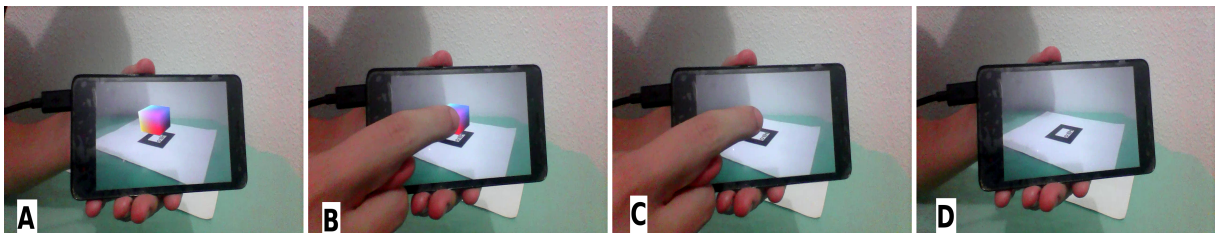
4.3.4 Toque na tela

Um método bem comum de interação é tocar no objeto virtual na tela em que ele está sendo projetado. Inclusive está disponível em algumas bibliotecas de realidade aumentada, como nas bibliotecas Vuforia e ARCore.

Como exemplo, considere a Figura 12 (a), que mostra um cenário construído com auxílio da InterativaAR onde um cubo que possui um ouvinte de toque na tela é projetado na tela de um celular. Ao ser tocado com o dedo (Figura 12b,c), a ação do ouvinte é executada, que nesse caso é destruir o cubo (Figura 12c,d).

Para que esse método funcione corretamente é necessário fazer com que a biblioteca seja capaz de receber e tratar os eventos de toque na tela que são gerados pelo Android. Esse assunto é abordado mais detalhadamente na Seção 4.5.7.

Figura 12 – Método de interação por toque na tela utilizando a InterativaAR



Fonte – Próprio autor

4.4 Utilização da API

A seguir são mostrados os códigos para criação de um ouvinte (ou *listener*) e o registro em um objeto, neste caso chamado de cubo, para cada um dos possíveis métodos de interação. O local onde o comentário "TODO" aparece, é o local onde deve ficar o código responsável pela ação que irá ocorrer após o disparo do evento (a resposta ao evento), que é definido pelo usuário da biblioteca.

4.4.1 Aproximação

```
1  //definindo uma classe responsável por tratar a aproximação
       (ouvinte do método de interação por aproximação)
2
3  class OuvinteAproximacao implements ApproximationListener {
4
5      //função chamada quando outro objeto virtual está
           próximo. O parâmetro representa o objeto interativo
           que está próximo.
6      @Override
7      public void iObjectNearby(InteractiveObject
           interactiveObject){
8          //TODO
9      }
10 }
11
12 //instanciando objeto ouvinte
13 ApproximationListener ouvinteAprox = new OuvinteAproximacao
       ();
14
15 //registrando o ouvinte em um objeto (cubo)
16 cubo.setApproximationListener(ouvinteAprox);
```

4.4.2 Oclusão

```
1  //instanciando ouvinte e registrando-o de maneira mais
       pratica
2
3  cubo.setOcclusionListener(new OcclusionListener() {
4
5      //função chamada quando oclusão se inicia
6      @Override
7      public void occlusionEnter(){
8          //TODO
9      }
10
11     //função chamada enquanto oclusão acontece
12     @Override
13     public void occlusionPressed(){
14         //TODO
15     }
16
17     //função chamada quando oclusão termina
18     @Override
19     public void occlusionReleased(){
20         //TODO
21     }
22
23 });
```

4.4.3 Movimentação

```
1  cubo.setMovementListener(new MovementListener() {
2
```

```
3       //função chamada quando acontece algum movimento. O
            parâmetro direction contem a informação sobre a
            direção do movimento.
4
5       @Override
6       public void movement(Direction direction) {
7           //TODO
8       }
9
10  });
```

4.4.4 Toque na tela

```
1   cubo.setTouchListener(new TouchListener() {
2
3       //funcão chamada quando acontece um toque na tela na
            região onde está desenhado o objeto virtual
4       @Override
5       public void touched() {
6           //TODO
7       }
8
9   });
```

4.5 Discussão sobre a implementação

O objetivo dessa seção é discutir os detalhes da implementação da biblioteca e apresentar de forma mais detalhada seu funcionamento para o usuário. Como a biblioteca é desenvolvida para Android, foi escrita utilizando a linguagem Java, para o melhor entendimento dessa seção é recomendado um conhecimento básico de Java e orientação a objetos. A discussão foi dividida em 6 pontos essenciais: o núcleo da biblioteca, os objetos interativos, e os quatro

métodos de interação (aproximação, oclusão, movimentação e toque na tela).

4.5.1 O núcleo da biblioteca

O núcleo é o principal componente da biblioteca, responsável por armazenar os objetos virtuais e receber os recursos necessários para o funcionamento da biblioteca. Além disso é no núcleo que são ativados os componentes responsáveis pela detecção das interações.

Na prática o núcleo da biblioteca é uma classe chamada de *InterativaAR*. Essa classe possui duas formas de ser instanciada, sem nenhum parâmetro ou passando como parâmetros a matriz de projeção, a largura e altura da imagem da cena que será mostrada ao usuário. Esses parâmetros são necessários para que todos os métodos funcionem corretamente, entretanto também podem ser setados apos a instanciação, caso não seja possível passa-los logo na inicialização.

Uma vez instanciada a classe principal, que representa a própria biblioteca, o usuário deve registrar todo objeto interativo criado, na biblioteca, para que esse possa receber os eventos relativos as interações. Para isso, basta utilizar o método *addIobject(InteractiveObject Iobject)* da biblioteca instanciada.

Com os objetos registrados, o próximo passo é chamar o método *update(byte[] cameraImage)* a cada novo quadro capturado pela câmera. O parâmetro passado é justamente a imagem capturada. O tipo *byte[]* também é utilizado pela API do Android para representar uma imagem capturada pela câmera, assim basta que o usuário repasse a informação que já é fornecida pelo Android, sem necessidade de realizar nenhum tratamento. Também é possível passar essa imagem como um objeto do tipo *Mat*, uma classe da OpenCV que representa uma matriz de pixels. Ou ainda chamar o *update* sem passar nenhum parâmetro, entretanto desta forma a biblioteca só verificará se está ocorrendo uma interação de aproximação ou toque na tela, pois esses não necessitam da imagem da câmera. É no *update* que são chamados os componentes que verificam se está ocorrendo alguma interação. Assim como o núcleo da biblioteca, esses componentes nada mais são do que classes.

4.5.2 Os objetos interativos

Para representar um objeto virtual que pode sofrer interação a partir de um ou mais métodos de interação da biblioteca é preciso criar um objeto do tipo *InteractiveObject*. Chamamos esse objeto de objeto interativo. Esse é definido basicamente por 3 propriedades:

posição, rotação e escala. Essas 3 propriedades são representadas por uma matriz 4x4, também conhecida como matriz de modelo (*Model matrix*). Além disso cada objeto interativo pode possuir um ouvinte para cada método de interação possível (aproximação, oclusão, movimentação e toque na tela). O usuário pode ainda aplicar as operações de translação, escala e rotação.

Para ilustrar como utilizar um objeto interativo, considere o cenário onde o usuário está utilizando uma biblioteca baseada em marcadores para realizar as projeções. O usuário deseja colocar um cubo ao lado esquerdo de um marcador, além de fazer com que esse cubo seja capaz responder a uma interação por oclusão. Para isso, primeiro ele deve conseguir a matriz de modelo (*Model matrix*) do marcador, utilizando a biblioteca que realiza a projeção. Após isso o usuário deve instanciar um objeto interativo e setar sua matriz de modelo igual a matriz do marcador, chamado o método *setTransformationMatrix(float[][] transformationMatrix)*. Dessa forma o objeto interativo possui uma matriz de modelo idêntica ao do marcador, ou seja, caso um cubo seja desenhado com essa matriz, ficará em cima do marcador. Para traze-lo para esquerda basta utilizar o método *translate(float x, float y, float z)* do objeto interativo, utilizando como parâmetros: x = -10, y = 0, z = 0. Isso irá gerar outra matriz de modelo, que pode ser obtida com o método *getTransformationMatrix()*.

Caso o usuário queira que esse objeto passe a receber os eventos de oclusão ele deve utilizar o método *addIobject(InteractiveObject Iobject)* da biblioteca de interação instanciada, passando o objeto interativo. Para que o objeto interativo possa responder a eventos de oclusão o usuário deve setar um ouvinte de oclusão, com o método *setOcclusionListener(OcclusionListener ol)*. Por fim, basta utilizar a biblioteca que realiza a projeção para desenhar o cubo com a matriz de modelo do objeto interativo.

4.5.3 Métodos de interação

Antes de cobrir detalhadamente como cada método de interação funciona, apresentamos a seguir uma explicação genérica de como um evento de interação ocorre. Cada componente que realiza a verificação se uma interação ocorreu, percorre a lista de objetos interativos da biblioteca (que são adicionados com o método *addIobject(InteractiveObject Iobject)*). Para cada objeto dessa lista é verificado se ele possui o ouvinte referente ao método de interação que está sendo testado. Por exemplo, o componente responsável por verificar o método de interação por aproximação, verifica se objeto possui um ouvinte de aproximação. Se sim, o componente verifica se está ocorrendo a interação que ele é responsável. Confirmado que

alguma interação ocorreu, um evento é disparado. Na prática, um evento é realizar uma chamada ao método definido no ouvinte do objeto. Por exemplo, no caso do método de interação por aproximação é chamado o método *iObjectNearby(InteractiveObject interactiveObject)* do ouvinte do objeto que está próximo a outro.

Tanto o método de interação por oclusão quanto o por movimentação precisam realizar algum processamento sobre a imagem da câmera. Para auxiliar esse processamento esta biblioteca utiliza algumas funções e tipos da biblioteca OpenCV.

4.5.4 Interação por aproximação

Esse é o método de interação mais simples dos quatro oferecidos pela biblioteca. Para cada objeto interativo é calculada a distância para todos os outros objetos, e caso essa distância seja menor que a distância mínima é disparado o evento de aproximação para aquele objeto. É possível que o usuário defina a distância mínima utilizando o método *setApproximationMinDistance(double minDistance)* da biblioteca. Quanto menor o parâmetro *minDistance* mais próximos o objetos devem ficar para disparar o evento. Esse parâmetro pode assumir valores entre 0 e $1.7*10^{308}$ e o valor padrão é 100.

4.5.5 Interação por oclusão

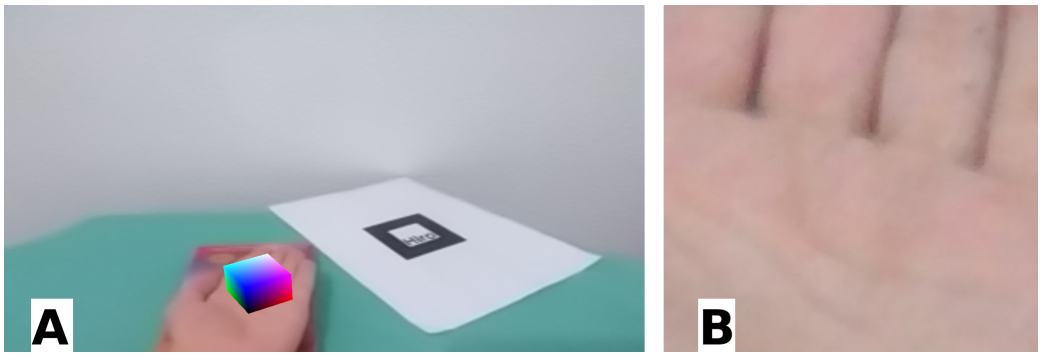
Para calcular se um objeto virtual está sofrendo oclusão, a parti da imagem da câmera é recordada a imagem da parte da superfície onde o objeto está sendo desenhado. Na primeira vez que isso é feito, a imagem é chamada imagem base, nas próximas vezes é chamada de imagem atual. A oclusão ocorre quando a imagem base é diferente da imagem atual. A Figura 13a mostra um cenário onde um objeto virtual está sendo desenhado sobre um livro. Assim a imagem base será uma parte da capa desse livro, como é mostrado na Figura 13b. Se esse cenário não for alterado, as próximas imagens (imagem atual) serão iguais a imagem base. Entretanto se um corpo se sobrepor a capa do livro, a imagem atual mudará e será diferente da imagem base. Esse cenário é ilustrado na Figura 14a, onde o objeto é coberto por uma mão. A Figura 14b mostra a imagem atual, que é diferente da imagem base (Figura 13b), isso indica que está acontecendo uma oclusão.

Para comparar as duas imagens, base e atual, é feito uma comparação de histograma. Tanto a extração das imagens base e atual quanto a comparação de histograma são feitos com auxílio da biblioteca OpenCV. O usuário pode definir se essa comparação deve ser mais ou

Figura 13 – Extração da imagem base

Fonte – Próprio autor

Figura 14 – Imagem atual diferente da imagem base



Fonte – Próprio autor

menos rígida. Ou seja, se a imagem atual precisa ser muito diferente da base para acontecer a oclusão (comparação mais rígida), ou a oclusão deve acontecer mesmo com pequena diferença entre as imagens (menos rígida). Para isso, basta utilizar o método *setOcclusionRigor(double rigor)* da biblioteca, quanto maior o parâmetro *rigor* mais rígida é a comparação. Esse parâmetro pode assumir valores entre -1 e 1 e o valor padrão é 0.002.

4.5.6 Interação por movimentação

Para calcular a movimentação é utilizado o algoritmo de Gunnar Farneback (FARNEBÄCK, 2003), que é utilizado para calcular o fluxo ótico (*Optical Flow*). Esta biblioteca utiliza uma implementação do algoritmo disponível na biblioteca OpenCV. É importante ressaltar que para que o algoritmo funcione corretamente a câmera deve permanecer estática ou sofrer uma movimentação mínima. O algoritmo de Gunnar Farneback recebe duas imagens sequenciais, A e B, calcula a movimentação de cada pixel da imagem A para B. Esta biblioteca então verifica para que direção se movimentou a maioria dos pixels, essa direção que é representada por um vetor 2D é o resultado do cálculo da movimentação entre duas imagens.

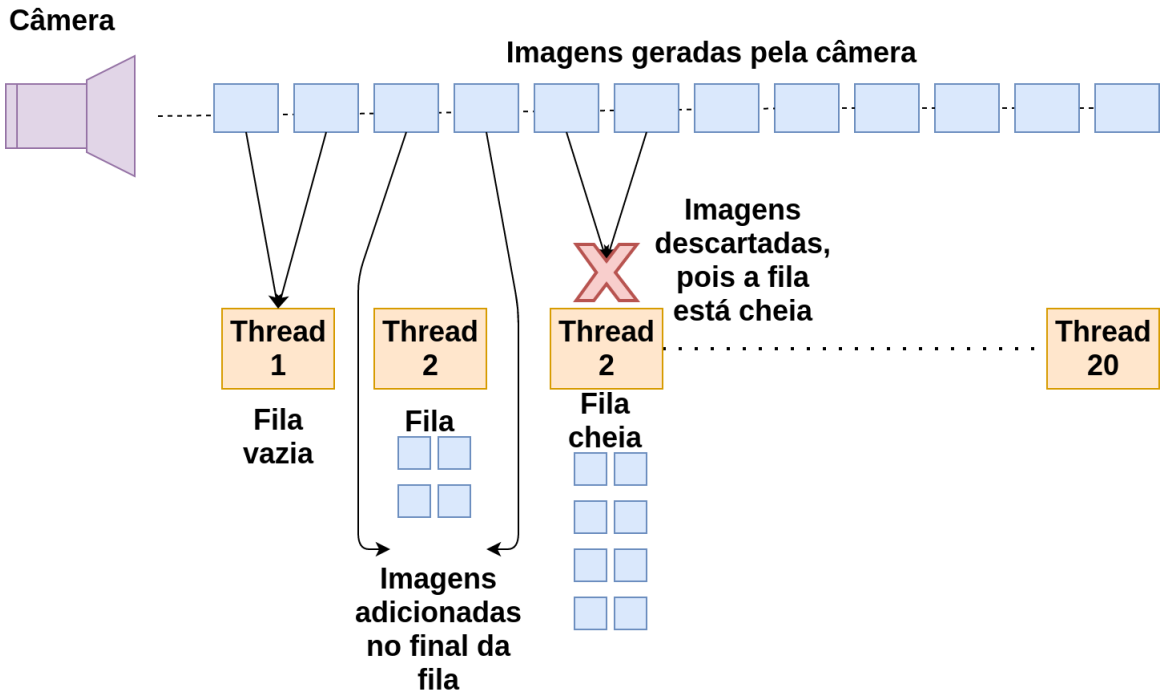
A fim de melhorar o desempenho, não é utilizada a imagem completa da câmera, mais apenas a região dessa imagem onde está o objeto virtual, chamada de região de interesse, assim como é feito na detecção da oclusão.

Algumas adaptações foram necessárias para garantir um bom desempenho. A primeira delas é na quantidade de cálculos realizados. Se uma câmera gera 4 imagens, nomeadas A, B, C, D. Para calcular a movimentação corretamente seria necessário calcular a movimentação de A para B, B para C e C para D. Entretanto para diminuir o número de operações, calculamos apenas de A para B e de C para D.

Outra mudança necessária foi a utilização de 20 *threads* para realizar o processamento. Cada *thread* é responsável por calcular a direção do movimento que aconteceu entre duas imagens. Elas estão dispostas em uma lista circular. De forma que o primeiro conjunto de duas imagens é passado para primeira, o segundo para a segunda, e assim segue. Como só existem 20 *threads*, o vigésimo primeiro conjunto de imagens é entregue a primeira *thread*. O próximo (vigésimo segundo) é entregue a segunda, e assim sequencialmente. Pode acontecer um cenário em que uma *thread* recebe um novo conjunto antes de terminar o cálculo anterior. Nesse caso o conjunto é adicionado a uma fila para ser calculado posteriormente. Caso essa fila já esteja cheia, o conjunto é descartado. Esses cenários podem ser visualizados mais facilmente com auxílio da Figura 15. Nessa Figura, a Thread 1 não possui nenhum conjunto de imagens na fila, por isso pode começar processar a movimentação no conjunto que chega. Já a Thread 2, possui alguns conjuntos na fila, porem ela não está cheia, por isso um novo conjunto que chega é adicionado ao final da fila. Por fim temos a Thread 3 que já possui a fila cheia, e enquanto a fila permanecer cheia os novos conjuntos que chegam são descartados.

Cada *thread* tem como resultado do cálculo da movimentação um vetor 2D, com atributos x e y, que representam da direção do movimento. O x do vetor representa a intensidade da movimentação no eixo x e o y no eixo y. Assim poderíamos ter como resultado, por exemplo, os vetores: (0,0),(2.1, -0.2),(0.2, -0.3),(3, 1),(2.3, -1.2),(0,0),(0,0),(0,0),(0,0),(0,0). Cada vetor é somado até o vetor nulo (0,0) ocorrer 5 vezes seguidas, ou seja até a movimentação parar. No exemplo anterior, a soma resulta no vetor: (7.6, -0.7). Então é verificado a tendência de direção do vetor resultante, calculando quais dos atributos, x e y, possui maior valor absoluto. No caso do vetor (7.6, -0.7) é o x, pois $|7.6| > |-0.7|$, ou seja aconteceu uma movimentação horizontal. Apos isso, é checado o sinal do atributo que possui o maior valor. No caso do x, o sinal positivo significa um movimento para direita, e o negativo para esquerda. Para o y, o sinal

Figura 15 – Cálculo da movimentação com multiplas *threads*



Fonte – Próprio autor

positivo significa um movimento para cima, e o negativo para baixo. No final, o vetor (7.6, -0.7) dispararia um evento de movimentação com a direção para direita.

Poderíamos ao invés de realizar a soma dos vetores, apenas disparar o evento de movimentação para cada vetor. Por exemplo, o conjunto de vetores: (2.1, -0.2),(0.2, -0.3),(3, 1),(2.3, -1.2), resultaria em 4 disparos de evento do movimentação. Entretanto foi observado empiricamente que o algoritmo para o cálculo da movimentação algumas vezes apresenta erros. Fazendo uma analogia com o exemplo anterior, é possível ver que 3 dos 4 vetores apresentam y negativo, mostrando uma tendência de movimento para baixo, apenas um apresenta y positivo, isso seria um erro do algoritmo. Quando somamos os vetores esses erros costumam ser minimizados.

Outra possibilidade seria passar o vetor resultante da soma, ao invés de uma direção na hora de disparar evento de movimentação. Por exemplo, o usuário receberia o vetor (7.6, -0.7) ao invés de uma direção (direita, esquerda, cima baixo). Desta forma, o usuário teria acesso a intensidade da direção dos eixos x e y. Entretanto novamente, o algoritmo apresenta erros, e algumas vezes uma movimentação mais lenta resulta em uma pequena intensidade. Às vezes uma movimentação rápida resulta em uma grande intensidade. Por isso, é melhor para o usuário receber apenas a direção do movimento, que possui uma precisão relativamente boa.

O usuário pode controlar o rigor para detectar movimentação. Um rigor baixo, significa detectar pequenas movimentações, e um rigor alto detecta apenas movimentações bem aparentes. Para controlar esse parâmetro basta usar o método *setMovementRigor(int rigor)* da biblioteca. O parâmetro *rigor* assume valores entre 1 e o tamanho da largura da tela, o valor padrão é 32. Como o algoritmo necessita que a câmera não se movimente ou se movimente minimamente, o usuário pode utilizar giroscópio para garantir essa restrição. Se o giroscópio indica que o dispositivo está se movendo, o cálculo da movimentação é interrompido até que o dispositivo pare. O usuário pode ainda setar o erro do giroscópio com o método *setGyroError(float gyroError)* da biblioteca. Quanto maior o parâmetro *gyroError* mais o usuário poderá movimentar o dispositivo sem interromper o cálculo da movimentação. Quanto menor, mais estático a câmera deverá ficar para que o cálculo da movimentação seja realizado. Esse parâmetro assume valores entre 0 e 1, e o valor padrão é 0.06.

4.5.7 Toque na tela

A interação por toque na tela assim como a de aproximação é bem simples. Primeiramente é calculada a região na tela do dispositivo onde está desenhado o objeto virtual. Então, quando usuário toca na tela, é verificado se o toque aconteceu na região onde está sendo desenhado o objeto virtual, se sim o evento de toque na tela é disparado.

Como é necessário obter a posição que está ocorrendo o toque na tela, é preciso que o objeto da interface que está mostrando a imagem da câmera possa tratar os toques na tela. Para isso basta chamar o método *setOnTouchListener(OnClickListener l)* do objeto de interface, passando como parâmetro o retorno do método *getViewTouchListener()* da biblioteca. O código final se assemelha a algo como: *objetoInterfaceAndroid.setOnTouchListener(interativaAR.getViewTouchListener());*

5 RESULTADOS E DISCUSSÃO

Nesse capitulo são apresentados e discutidos os resultados a respeito da biblioteca. Na Seção 5.1 são mostrados os resultados referentes ao desempenho dos métodos de interação da biblioteca. Na Seção 5.2 são mostrados os resultados referentes a precisão dos métodos. Por fim, na Seção 5.3 são apresentadas aplicações construídas com o uso da biblioteca.

Todos os testes e aplicações foram executados em um mesmo ambiente, sem grandes variações na iluminação. Para realizar a projeção dos objetos virtuais nas aplicações foi utilizado a biblioteca ARToolKit. As aplicações e testes rodaram em uma resolução de 320 x 240. Foi utilizado unicamente o dispositivo Xiaomi Redmi 2 Pro, com as seguintes especificações:

- Sistema operacional Android 4.4.4 MIUI v6 KitKat
- Processador 1.2 GHz Quad Core;
- GPU Adreno 306;
- 2 GB de memória RAM;

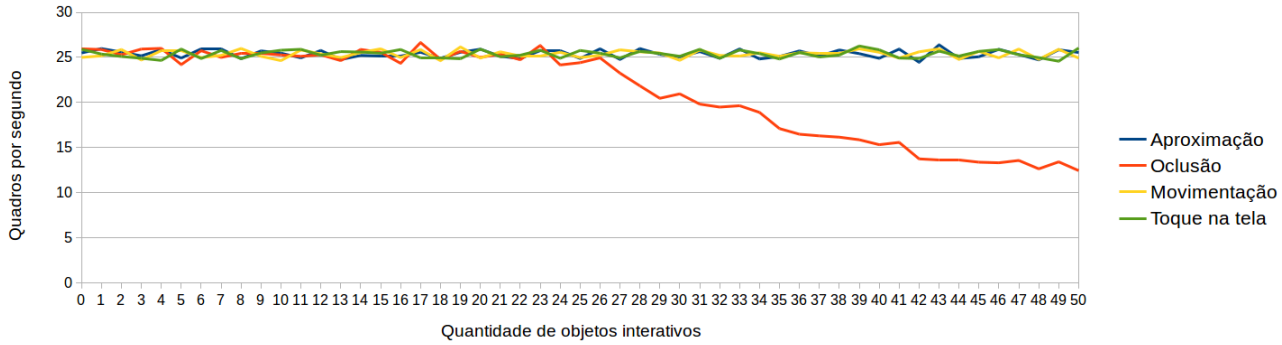
5.1 Teste de desempenho

Para medir o desempenho dos métodos de interação é utilizado como medida a quantidade de quadros mostrados pela aplicação a cada segundo (ou *Frames Per Second, FPS*). Como o processamento é realizado a cada quadro, a câmera precisa esperar que o processamento termine para que possa capturar outro quadro. Assim caso, a câmera tenha que esperar muito tempo, ela irá gerar menos quadros por segundos, consequentemente a aplicação irá mostrar menos quadros por segundos, e a percepção de movimento irá diminuir. Então é importante que os métodos de interação não causem grandes percas de quadros por segundo.

Quanto mais objetos interativos, mais processamento a biblioteca deve realizar. Portanto, nos testes de desempenho a quantidade de objetos interativos com um mesmo ouvinte foi aumentada gradativamente, até chegar a 50 objetos, e a taxa de quadros por segundo foi medida para cada aumento. Os resultados são apresentados na Figura 16.

Na Figura 16 é possível ver que o único método que apresentou uma queda considerável na taxa de quadros por segundo foi o método de oclusão, todos os outros se mantiveram estáveis ao aumento de objetos interativos. Esses resultados fazem sentido pois o método de interação por aproximação apenas realiza o cálculo da distância, mesmo com muitos objetos esse cálculo é realizado rapidamente. O método de interação por toque na tela só executa

Figura 16 – Quantidade de objetos interativos x quadros por segundo



Fonte – Próprio autor

algum processamento, quando o usuário realiza uma ação de toque na tela. Além disso, o processamento realizado também é executado rapidamente.

O método de interação por movimentação faz uso de um algoritmo que demanda bastante recurso, se comparado com os outros. Entretanto, como esse algoritmo é executado por outras *threads*, diferentes da *thread* principal, a câmera não espera o algoritmo finalizar para capturar outro quadro. Portanto não há interferência na taxa de captura de quadros. O ponto negativo é que a detecção da interação não é feita em tempo real e quanto mais objetos interativos mais demorada será a detecção. Já as interações por oclusão conseguem ser verificas em tempo real, mas para isso é necessário que a câmera espere a verificação acabar para capturar um novo quadro, o que gera uma diminuição da taxa de quadros por segundo, principalmente para uma grande quantidade de objetos interativos.

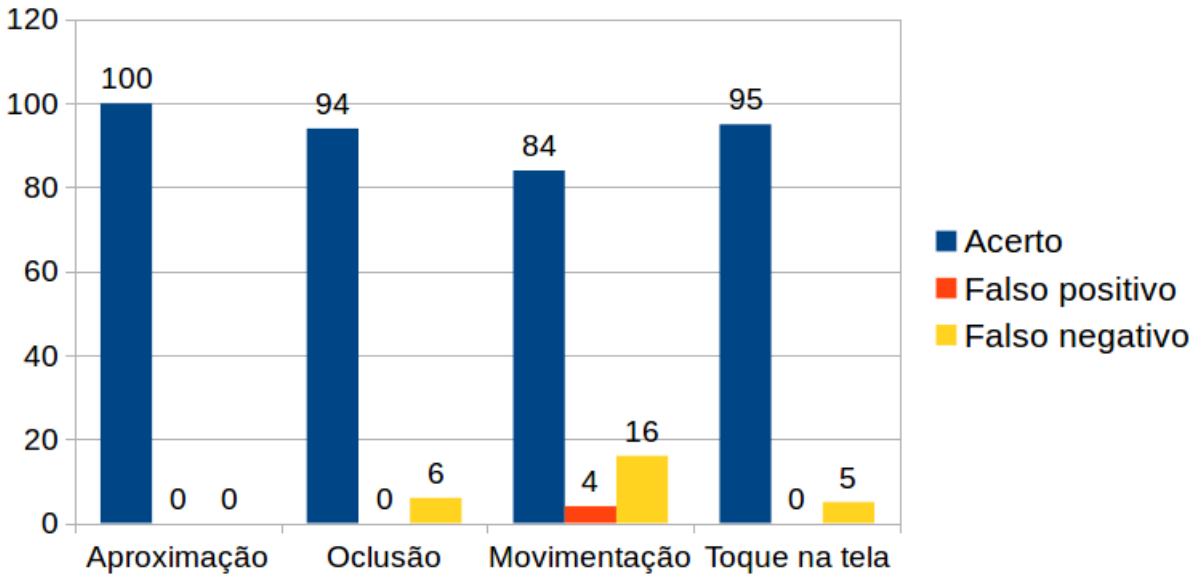
5.2 Teste de precisão

Para medir a precisão dos métodos de interação, foi criado para cada método uma aplicação com um objeto interativo que possuía o ouvinte do método. Então em cada aplicação foram realizadas 100 interações com o objeto. Os resultados são mostrados na Figura 17.

Na Figura 17, um resultado falso positivo significa que apesar de nenhuma interação ser realizada pelo usuário a biblioteca detectou alguma interação. Um resultado falso negativo significa que apesar de a interação ser realizada pelo usuário a biblioteca não foi capaz de detectá-la.

Para esse teste as interações foram feitas com a câmera do dispositivo estática. No teste de movimentação cada interação (movimento) durou aproximadamente 5 segundos. Já no teste de oclusão cada interação durou cerca de 3 segundos. Foram utilizados os parâmetros

Figura 17 – Resultado do teste de precisão



Fonte – Próprio autor

padrões da biblioteca.

O gráfico da Figura 17 mostra que todos os métodos apresentaram um bom desempenho, para o ambiente onde foram testados. Os maiores erros ocorreram nos métodos de interação por oclusão e movimentação. Nesses dois métodos há uma dependência do tempo de duração da interação. Uma interação mais rápida é mais difícil de ser detectada, já uma interação mais lenta pode ser detectada mais facilmente, mas pode se tornar inconveniente para a aplicação. Como mostrado no gráfico, interações de duração aproximada de 5 segundos para a movimentação e 3 segundos para oclusão, apresentam bons resultados e não são muito lentas.

O erro presente no método de interação de toque na tela é causado quando um objeto interativo fica na borda da tela e é desenhado apenas parcialmente, ou seja o restante do objeto está fora da tela. Nesse caso não é possível achar a região na tela onde o objeto está sendo desenhado pois a biblioteca não consegue pegar um ponto tridimensional que não está sendo mostrado e converte-lo em um ponto na tela. Assim, nesse caso a interação não é detectada.

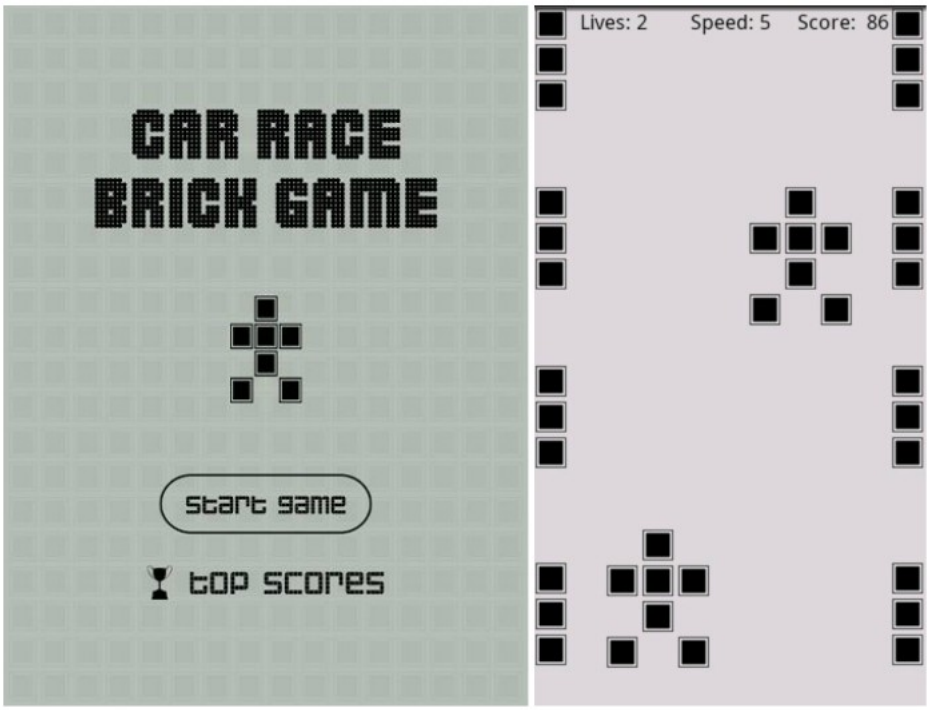
5.3 Aplicações

Essa seção apresenta um conjunto de aplicações desenvolvidas com o auxílio da InterativaAR. O objetivo é dar uma amostra do que é possível desenvolver utilizando a biblioteca.

5.3.1 Jogo de corrida

Utilizando como inspiração um antigo jogo de corrida do minigame Brick Game, mostrado na Figura 18. Foi construído uma versão do mesmo jogo em realidade aumentada. O objetivo do jogo é desviar dos carros que estão a frente, evitando a colisão, para isso no minigame o jogador utiliza os botões de esquerda e direita. Na versão em realidade aumentada desenvolvida por esse trabalho, o carro é movimentado de acordo com a direção da interação por movimentação. Além disso a interação por aproximação também é utilizada para checar se o carro colidiu com outro.

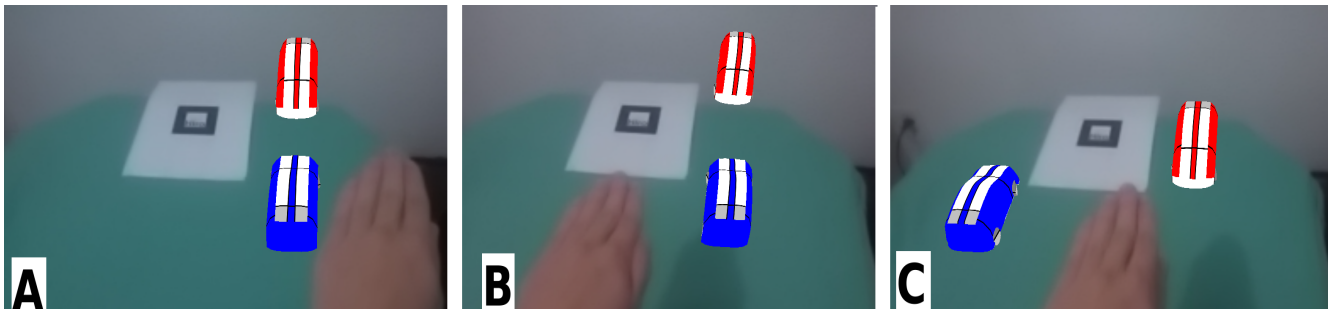
Figura 18 – Jogo do minigame Brick Game



Fonte – Disponível em: https://pipocacombacon.wordpress.com/2016/09/18/games-jogos-de-corrida/ Acesso em: 6 dez. 2017.

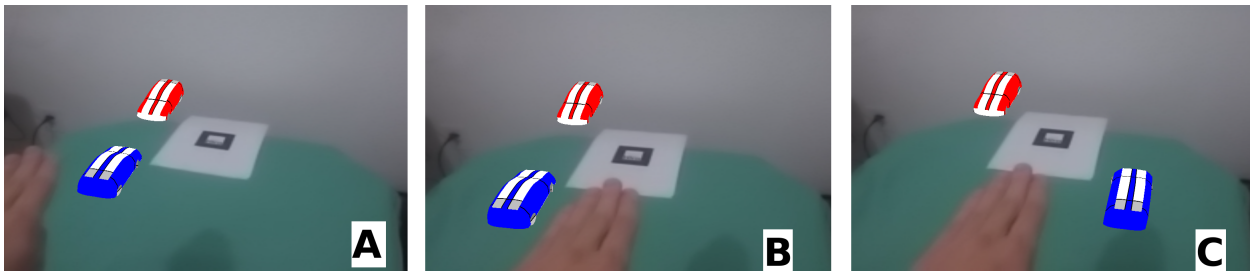
No jogo existem duas faixas em que o carro pode andar, a direita ou a esquerda. Na Figura 19 é mostrado a movimentação da faixa da direita a esquerda. E na Figura 20 é mostrado a movimentação da faixa da esquerda para a direita. Nessas duas imagens o carro azul é o carro do jogador e os carros vermelhos são os carros que ele deve desviar.

Figura 19 – Movimentação do carro para esquerda



Fonte – Próprio autor

Figura 20 – Movimentação do carro para direita



Fonte – Próprio autor

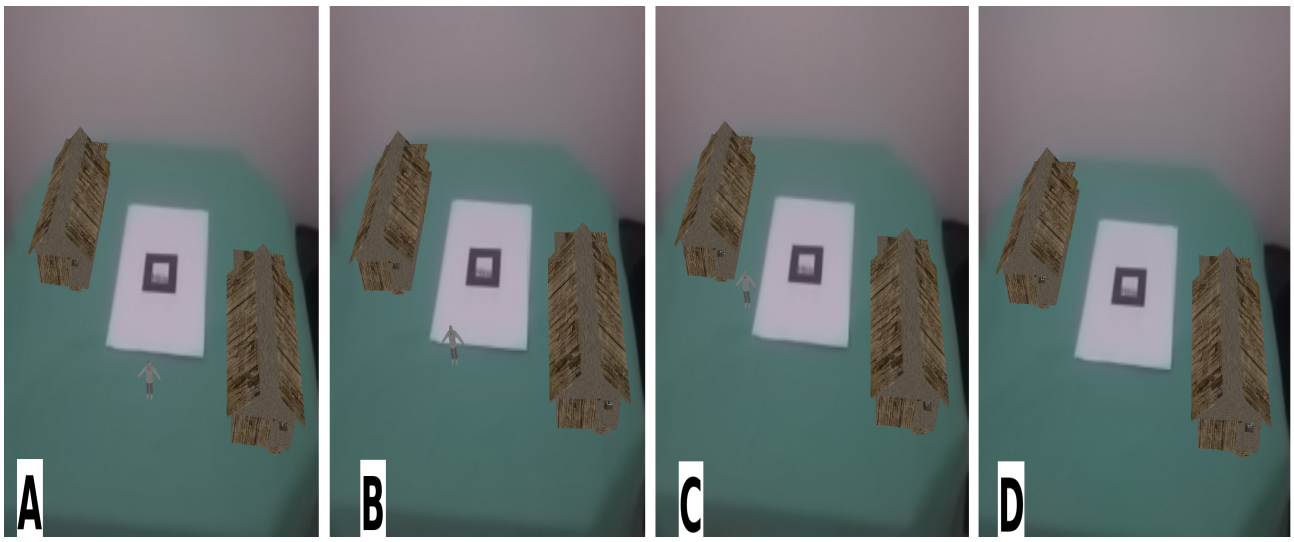
5.3.2 Cenário interativo

O objetivo dessa aplicação é demonstrar como o método de interação por toque na tela pode ser utilizado. A aplicação é constituída por 3 objetos interativos: duas casas e um personagem. Quando o usuário toca uma casa pela tela do dispositivo o personagem se movimenta em direção a casa que foi tocada. A Figura 21 mostra a movimentação do personagem quando a casa da esquerda é tocada. Utilizando o método de interação por aproximação no personagem, é verificado se ele está próximo a casa, se sim ele não é mais desenhado, como mostrado na Figura 21d, simulando a entrada do personagem na casa.

5.3.3 Piano virtual

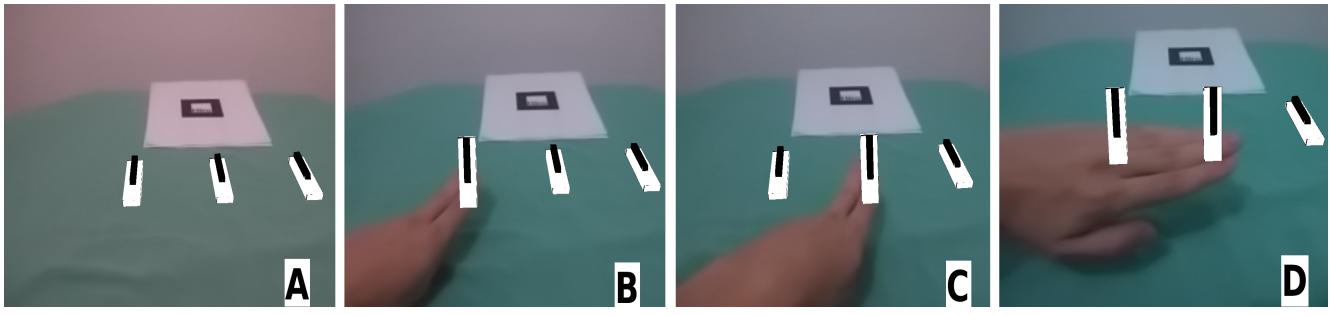
Para demonstrar como o método de interação por oclusão pode ser utilizado, foi construído uma aplicação que simula um piano em realidade aumentada. Cada tecla do piano é um objeto interativo que possui um ouvinte de oclusão. Ao ser ocluído (coberto) o modelo tridimensional da tecla sofre uma rotação, afim de simular o pressionamento da tecla. Além disso um som associado a tecla é tocado. A aplicação é mostrada na Figura 22.

Figura 21 – Movimentação do personagem pelo método de interação por toque na tela



Fonte – Próprio autor

Figura 22 – Piano em realidade aumentada



Fonte – Próprio autor

6 CONSIDERAÇÕES FINAIS

Este trabalho apresentou uma biblioteca capaz de facilitar a criação de aplicações de dispositivos móveis que utilizam a tecnologia de realidade aumentada. Para realizar essa tarefa foram definidos quais métodos de interação a biblioteca iria fornecer ao usuário. Esses, foram baseados em trabalhos anteriores que também visavam tornar mais fácil a construção de aplicações de realidade aumentada interativas.

Inicialmente a biblioteca havia sido pensada para funcionar apenas com uso de marcadores. Posteriormente optou-se por desenvolvê-la sem essa restrição. Assim os métodos de interação também precisaram ser repensados. Foram escolhidos os métodos de: aproximação, oclusão, movimentação e toque na tela. Todos eles poderiam ser implementados sem a necessidade de utilização de marcadores.

Quanto a implementação dos métodos de interação, houve certa dificuldade para os métodos de oclusão e movimentação. Esses são justamente os que utilizam a imagem da câmera. Assim, os algoritmos inicialmente pensados para detecção das interações ficavam sensíveis à mudança de ambiente ou da iluminação do ambiente. Ou ainda eram bastante lentos. Foi preciso realizar algumas adaptações para melhorar o desempenho e precisão nesses dois métodos.

Toda a complexidade das tarefas executadas pela biblioteca fica totalmente escondida do usuário devido a API implementada. A ideia de utilizar um sistema de eventos na API surgiu a partir dos eventos das interfaces gráficas do Java. Assim alguém que já estava familiarizado com esse sistema teria maior facilidade de aprendizagem. O desenvolvimento da interação nas aplicações na Seção 5.3 por exemplo, ocorreu de forma muito rápida. As maiores dificuldades encontradas nesse caso ocorreram na utilização da biblioteca responsável pela projeção. A construção da interação, com auxílio da InterativaAR, ocorreu sem grandes dificuldades.

Os métodos de interação se mostraram eficientes de forma geral, principalmente se poucos objetos interativos forem utilizados. Os métodos de aproximação e toque na tela, obtiveram resultados, principalmente se for considerado o tempo de resposta. Mesmo a interação por oclusão, que é calculada em tempo real, por muitas vezes demorou alguns segundos para ser detectada. Como a interação por movimentação não ocorre em tempo real o tempo de resposta por vezes era mais alto do que o desejável. Além disso a movimentação da câmera, mesmo pequena, causou erros no cálculo da direção.

Um possível trabalho futuro é melhorar o desempenho e precisão dos métodos de interação de oclusão e movimentação. Essa é uma tarefa complexa pois muitas vezes para

melhorar o desempenho um pouco de precisão é perdida e vice-versa. Além disso, seria interessante criar versões da biblioteca para diferentes plataformas.

REFERÊNCIAS

AZUMA, R. T. A survey of augmented reality. **Presence: Teleoperators and virtual environments**, MIT Press, v. 6, n. 4, p. 355–385, 1997.

BILLINGHURST, M.; CLARK, A.; LEE, G. et al. A survey of augmented reality. **Foundations and Trends® in Human–Computer Interaction**, Now Publishers, Inc., v. 8, n. 2-3, p. 73–272, 2015.

BILLINGHURST, M.; KATO, H.; POUPYREV, I. The magicbook-moving seamlessly between reality and virtuality. **IEEE Computer Graphics and applications**, IEEE, v. 21, n. 3, p. 6–8, 2001.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV**: Computer vision with the opencv library. [S.l.]: "O'Reilly Media, Inc.", 2008.

ELISEU, S. R. T. D. N. **O mundo como uma CAVE**. Tese (Doutorado) — FBAUP - Faculdade de Belas Artes, 2017.

FARNEBÄCK, G. Two-frame motion estimation based on polynomial expansion. **Image analysis**, Springer, p. 363–370, 2003.

LEE, G. A.; BILLINGHURST, M.; KIM, G. J. Occlusion based interaction methods for tangible augmented reality environments. In: ACM. **Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry**. [S.l.], 2004. p. 419–426.

LEE, G. A.; NELLES, C.; BILLINGHURST, M.; KIM, G. J. Immersive authoring of tangible augmented reality applications. In: IEEE COMPUTER SOCIETY. **Proceedings of the 3rd IEEE/ACM international Symposium on Mixed and Augmented Reality**. [S.l.], 2004. p. 172–181.

NUNES, C.; FONSECA, D. C. da; LEITE, D. S.; NUNES, E. P. dos S.; PONCE, F. dos S.; FARIA, L. C. G. de; SISCOUTO, R. A. Arinteraction-biblioteca para interação com objetos virtuais para realidade aumentada. **VIII Symposium on Virtual and Augmented Reality, Belém, PA.**, 2006.

PIZETTA, D. C. **Biblioteca, API e IDE para o desenvolvimento de projetos de metodologias de Ressonância Magnética**. Dissertação (Mestrado) — Universidade de São Paulo, 2014.

POUPYREV, I.; TAN, D. S.; BILLINGHURST, M.; KATO, H.; REGENBRECHT, H.; TETSUTANI, N. Developing a generic augmented-reality interface. **Computer**, IEEE, v. 35, n. 3, p. 44–50, 2002.

REGENBRECHT, H. T.; WAGNER, M.; BARATOFF, G. Magicmeeting: A collaborative tangible augmented reality system. **Virtual Reality**, Springer, v. 6, n. 3, p. 151–166, 2002.

SHIM, J.; YANG, Y.; KANG, N.; SEO, J.; HAN, T.-D. Gesture-based interactive augmented reality content authoring system using hmd. **Virtual Reality**, Springer, v. 20, n. 1, p. 57–69, 2016.

WANG, X.; DUNSTON, P. S. Groupware concepts for augmented reality mediated human-to-human collaboration. In: **Proceedings of the 23rd Joint International Conference on Computing and Decision Making in Civil and Building Engineering**. [S.l.: s.n.], 2006. p. 1836–1842.

ZHOU, F.; DUH, H. B.-L.; BILLINGHURST, M. Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In: IEEE COMPUTER SOCIETY. **Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality**. [S.l.], 2008. p. 193–202.