TOOLKIT DE INTERAÇÃO PARA PROGRAMAÇÃO DE INTERFACES TANGÍVEIS BASEADAS EM MARCADORES EM REALIDADE VIRTUAL
João Diogo Lontro Saborano Mesquita

1 2 9 0

UNIVERSIDADE D
COIMBRA

João Diogo Lontro Saborano Mesquita

TOOLKIT DE INTERAÇÃO PARA PROGRAMAÇÃO DE INTERFACES TANGÍVEIS BASEADAS EM MARCADORES EM REALIDADE VIRTUAL

VOLUME 1

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas de Informação orientada pelo Professor Doutor Jorge Carlos dos Santos Cardoso e à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Junho de 2021

Faculdade de Ciências e Tecnologia
Departamento de Engenharia Informática

TOOLKIT DE INTERAÇÃO PARA PROGRAMAÇÃO DE INTERFACES TANGÍVEIS BASEADAS EM MARCADORES EM REALIDADE VIRTUAL

João Diogo Lontro Saborano Mesquita

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas de Informação orientada pelo Professor Doutor Jorge Carlos dos Santos Cardoso e apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Junho de 2021

UNIVERSIDADE D
COIMBRA

Resumo

Este projeto tem como foco o desenvolvimento de um toolkit de interação para interfaces tangíveis baseadas em marcadores para Realidade Virtual (RV). Pretende-se detetar objetos passivos através de marcadores visuais, e explorar as interações possíveis com os mesmos em ambientes RV. As possibilidades de interação deste paradigma de interação foram estudadas anteriormente, no entanto, a complexidade das interações foi limitada, devido à inexistência de abstrações programáticas de alto nível que facilitem a programação. Como tal, neste trabalho, foi desenvolvido um conjunto de abstrações que facilitam o desenvolvimento de projetos de RV, através de componentes da framework A-Frame na linguagem JavaScript. Os componentes desenvolvidos para as interações são baseadas na visibilidade dos seus respetivos marcadores, rotação e posicionamento no ambiente virtual. No total, a biblioteca criada tem 6 componentes: os componentes Angle Detector com um ou dois marcadores, que pretendem medir o ângulo entre um marcador e um eixo, ou entre os dois marcadores. O Button e o Swipe que permitem a interação através das mecânicas de clique e deslize. O Shake Detector que pretende detetar quando um objeto é abanado e o Noise Controller que permite reduzir o ruído associado à visibilidade dos marcadores. Os componentes podem trabalhar em conjunto em muitas situações para produzir uma maior variedade de interações. A biblioteca criada foi avaliada por um total de 8 programadores, através de testes de usabilidade. Os resultados indicam que apresenta indicadores de uma forma geral positivos de compreensibilidade, capacidade de aprendizagem, reutilização e abstração.

Palavras-Chave

Realidade virtual, interfaces tangíveis, interação, abstração programática, marcadores visuais, biblioteca de programação

Abstract

This project focused on developing an interaction toolkit for marker-based tangible interfaces for Virtual Reality (VR). It is intended to detect passive objects through visual markers, and explore possible interactions with them in VR environments. The interaction possibilities of this interaction paradigm have been studied previously, however, the complexity of the interactions has been limited due to the lack of high-level programmatic abstractions that facilitate programming. As such, in this work, a set of abstractions that facilitate the development of VR projects was developed, through components of the A-Frame framework in the JavaScript language. The components developed for interactions are based on the visibility of their respective markers, rotation and positioning in the virtual environment. In total, the library created has 6 components: the Angle Detector components with one or two markers, which intend to measure the angle between a marker and an axis, or between the two markers. The Button and Swipe that allow interaction through click and swipe mechanics. The Shake Detector which aims to detect when an object is shaken, and the Noise Controller which allows you to reduce the noise associated with the visibility of the markers. The components can in many situations work together to produce a greater variety of interactions. The library created was evaluated by a total of 8 programmers through usability tests. The results indicate that it has positive indicators for understandability, learnability, reusability, and abstraction.

Keywords

Virtual reality, tangible interfaces, interaction, programmatic abstraction, visual markers, programming library

Agradecimentos

Agradeço à minha família pelo esforço enorme que fizeram para me dar as condições de chegar onde cheguei. Agradeço também ao meu orientador, Jorge Carlos dos Santos Cardoso, pela constante disponibilidade e orientação no decorrer desta dissertação.

Índice

Capítulo 1 Introdução 1

1.1 Estrutura do documento 2

Capítulo 2 Background 1

2.1 Realidade Virtual e Aumentada 1

2.2 A-Frame 3

2.3 Marcadores 5

Capítulo 3 Objetivos e Metodologia 6

3.1 Plano de trabalho 6

3.2 Planeamento temporal 7

Capítulo 4 Estado da Arte 10

4.1 Projetos de interação em RV com marcadores 10

Interação em Realidade Virtual Através de Tangíveis Passivos 10

Opportunistic Controls: Leveraging Natural Affordances as Tangible User Interfaces for Augmented Reality 12

Tangible User Interaction Using Augmented Reality 14

Tangible VR: Diegetic Tangible Objects for Virtual Reality Narratives 15

Two-handed tangible interaction techniques for composing augmented blocks 16

In Your Hand Computing: Tangible Interfaces for Mixed Reality 17

Occlusion based Interaction Methods for Tangible Augmented Reality Environments 18

Conclusão 19

4.2 Toolkits de interação 20

XR Interaction Toolkit 20

VRTK 22

ARCore 23

ARKit 24

Zapbox 25

Capítulo 5 VR Tangible Marker Interaction Library 28

5.1 Estrutura 28

Arquitetura 28

Documentação 29

5.2 Shake Detector 30

5.3 Button 32

5.4 Swipe 34

5.5 Angle Detector Single Marker 36

5.6 Angle Detector Double Marker 39

5.7 Noise Controller 42

Capítulo 6 Testes de Usabilidade 44

6.1 Procedimento 44

Recrutamento 44

Tarefas 44

Questionário 46

6.2 Resultados 47
Dados demográficos 47
Tempos de realização 48
Compreensibilidade 48
Aprendizagem 49
Abstração 49
Reutilização 50
Testes presenciais 50
Comentários 51
Capítulo 7 Aplicações de Demonstração 54
7.1 Album Viewer Experience 54
7.2 Shake Book Experience 56
7.3 Conclusão 59
Capítulo 8 Conclusão 61
Referências 62
Apêndice A 66
Apêndice B 68

Acrónimos

RV – Realidade Virtual
RA- Realidade Aumentada
RM -Realidade Mista
XR- *Extended Reality*
HDM- *Head Mounted Display*
2D- 2 dimensões
3D- 3 dimensões
SDK – *Software Development Kit*
API- *Application Programming Interface*
UI- *User Interface*
IDE- *Integrated Development Environment*

Lista de Figuras

Figura 1- Contínuo rea-virtual 1
Figura 2- *The Sword of Damocles* 2
Figura 24- Conjunto de artefatos da Zapbox (https://www.zappar.com/zapbox/) 3
Figura 4- Exemplo de marcador 5
Figura 5- Planeamento temporal inicial 7
Figura 6- Planeamento temporal final 8
Figura 7- Protótipo para interação com rotação 10
Figura 8- Protótipo de torre com sino 10
Figura 9- Interação de clique 11
Figura 10- Interação de deslize 11
Figura 11- Interação de proximidade para entrada em portal 12
Figura 12- Rotação de objeto que permite a alteração de variável 12
Figura 13- Exemplo de aproveitamento de interface tangível 13
Figura 14- Menu com marcadores 14
Figura 15- Interação através da oclusão de marcadores 15
Figura 16- Interação com tangível 16
Figura 17- Método de aparafusamento 16
Figura 18- Método de *block-assembly* 17
Figura 19- *Magic Rings* e *Magic Bracelets* 18
Figura 20- Marcadores híbridos 19
Figura 21- Interação de clique com painel de marcadores 19
Figura 22- Interação de selecionar e agarrar objeto 21
Figura 23- Exemplo de locomoção com *feedback* visual 21
Figura 24- Interação de agarrar objeto com controladores 23
Figura 25- Kit de marcadores do Zapbox 25
Figura 26- Diagrama das tecnologias por detrás dos componentes 28
Figura 27- Exemplo da documentação do componente Button 29
Figura 28- Exemplo da segunda parte da documentação do componente Button 29
Figura 29- Funcionamento do algoritmo Shake Detector 30
Figura 30- Representação tangível de um sino 31
Figura 31- Funcionamento do algoritmo Button 32
Figura 32- Exemplo de utilização do Button 33

Figura 33- Funcionamento do algoritmo do Swipe 34
Figura 34- Exemplo de utilização do Swipe 35
Figura 35- Funcionamento do algoritmo Angle Detector Single Marker 37
Figura 36- Exemplo de utilização do Angle Detector Single Marker 37
Figura 37- Funcionamento do algoritmo do Angle Detector Double Marker 39
Figura 38- Exemplo de protótipo para utilização do Angle Detector Double Marker 39
Figura 39- Segundo tipo de ângulo 40
Figura 40- Funcionamento do Noise Controller 42
Figura 41- Exemplo de experiência virtual de uma tarefa 45
Figura 42- Tempos de realização das tarefas 48
Figura 43- Respostas no campo da compreensibilidade 49
Figura 44- Respostas no campo da aprendizagem 49
Figura 45- - Respostas no campo da capacidade de abstração 49
Figura 46- Respostas no campo da capacidade de reutilização 50
Figura 47- Experiência Album Viewer 55
Figura 48- Exemplo da experiência Shake Book (página 1) 56
Figura 49- Exemplo da experiência Shake Book (página 2) 56
Figura 50- Exemplo da experiência Shake Book (sino) 57

Lista de Tabelas

Tabela 1- Métodos do lifecycle da framework A-Frame 4
Tabela 2- Tipos de valores nas propriedades da framework A-Frame 4
Tabela 3- Métodos disponíveis para Interactors e Interactables 22
Tabela 4- Resumo das características dos toolkits estudados 26
Tabela 5- Parâmetros do Shake Detector 31
Tabela 6- Parâmetros do Button 33
Tabela 7- Parâmetros do Swipe 35
Tabela 8- Parâmetros do Angle Detector Single Marker 38
Tabela 9 - Parâmetros do Angle Detector Double Marker 41
Tabela 10- Parâmetros do Noise Controller 43
Tabela 11- Tarefas do teste de usabilidade 45
Tabela 12- Resumo dos dados dos participantes 47
Tabela 13- Dificuldades dos participantes 51

Capítulo 1
Introdução

A interação em Realidade Virtual (RV) é frequentemente atingida através de controladores como comandos, ou aparelhos com sensores incorporados. No entanto, considerar novas formas de input pode ser vantajoso, por exemplo, para situações em que os controladores não estão disponíveis ou não podem ser usados. Por outro lado, também podem servir simplesmente para fornecer uma experiência de uso mais divertida e apropriada ao ambiente virtual.

De modo a facilitar o desenvolvimento de experiências de RV é importante permitir que o utilizador final do conteúdo possa usufruir de forma fácil e com objetos de baixo custo e aquisição. Desta forma, este projeto foca-se em interações com interfaces tangíveis para interação em ambientes de RV com a utilização de marcadores visuais: um objeto que qualquer pessoa pode facilmente adquirir. A utilização de interfaces tangíveis possui algumas vantagens em ambientes virtuais, como a possibilidade de prestar feedback háptico ao utilizador, aumentado o realismo e imersividade da experiência. Além disso, permitem a criação de experiências mais intuitivas na medida em que o utilizador pode interagir diretamente com o ambiente virtual.

O uso de interfaces tangíveis para interação em RV não é novo, mas este projeto pretende apenas tratar de objetos passivos, detetáveis através de marcadores visuais. Um projeto anterior [1] explorou o espaço de design deste paradigma de interação e produziu vários exemplos de experiências que demonstram as possibilidades de interação através de marcadores visuais. Nesse trabalho, foram produzidos vários protótipos demonstrativos destas interações, utilizando um processo focado no feedback do utilizador final. No entanto, chegou-se à conclusão de que a complexidade destes protótipos está limitada pela dificuldade de programação deste tipo de interação, dada a inexistência de abstrações programáticas de alto-nível. Tendo como exemplo um designer que tenha conhecimentos em programação web: este não terá dificuldade em desenvolver experiências básicas, mas terá alguma dificuldade em criar experiências mais complexas, devido à complexidade programática que estas requerem. Como tal, as interações sobre um objeto tangível (como detetar o fechar ou abanar de um livro detetado por marcadores, por exemplo), teriam de ser totalmente programadas de raiz. Contudo, caso existissem bibliotecas para incluir interações nas experiências, o programador teria apenas de incluir o tipo de interação que pretendia, deixando que a ferramenta tratava do processamento de baixo nível. Consequentemente, reconheceu-se a falta de abstrações que permitissem que programadores de linguagens de alto-nível (ou outro tipo de programadores) e até mesmo entusiastas da área pudessem também desenvolver conteúdo na área da RV de forma mais fluida e abstrair do desenvolvimento deste tipo de interações.

Assim sendo, o objetivo é produzir uma família de componentes para programadores de RV, partindo das interações estudadas em outros projetos e outras possíveis interações que se mostrem úteis neste contexto.

1.1 Estrutura do documento

Capítulo 1- Introdução
Este é o capítulo onde exponho as motivações e o contributo que pretendo dar para a área em questão.

Capítulo 2- Background
Apresentação de conceitos relacionados com o tema e a plataforma de desenvolvimento do toolkit.

Capítulo 4- Objetivos e metodologia
Neste capítulo são descritos os objetivos e métodos utilizados para o desenvolvimento deste projeto, ao nível da planificação temporal e plano de trabalho.

Capítulo 4- Estado da Arte
Apresenta os resultados da pesquisa sobre as tecnologias e metodologias existentes nesta área e de que forma estes resultados me auxiliaram na escolha das ferramentas a utilizar nesta dissertação.

Capítulo 5- VR Tangible Marker Interaction Library
Um capítulo de apresentação do trabalho realizado, nomeadamente os componentes desenvolvidos e o seu funcionamento.

Capítulo 6- Testes de Usabilidade
Neste capítulo é apresentado de que forma decorreram os testes de usabilidade e as conclusões tiradas.

Capítulo 7- Aplicações de demonstração
Algumas experiências virtuais que foram desenvolvidas com auxílio dos componentes desenvolvidos.

Capítulo 8- Conclusão
Neste capítulo são explicadas as aprendizagens e conclusões tiradas com este projeto.

Capítulo 2 Background

2.1 Realidade Virtual e Aumentada

O conceito de RV remete-nos para algo que não é real. No entanto, apesar de virtual, estas realidades são também reais. São realidades alternativas, criadas artificialmente para serem percecionadas pelos nossos sistemas sensoriais da mesma forma que o mundo físico que nos rodeia. Desta forma, podem criar emoções, ensinar, entreter e responder às nossas interações com ou sem uma forma tangível. Atualmente, a tecnologia existente permite a criação de ambientes virtuais com alta definição, realismo e baixo custo. Porém, a área tem ainda uma enorme margem de evolução tanto em termos de otimização como de inovação.

Em 1994, um importante artigo [2] apresentou o que passou a ser conhecido por "Continuo real-virtual", apresentado na figura 1.

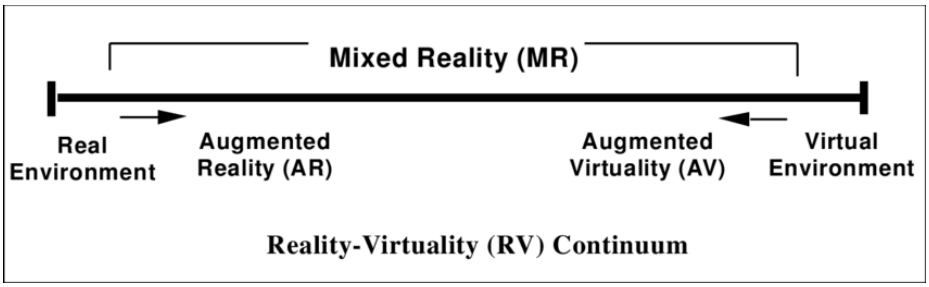

Figura 1- Contínuo real-virtual

No extremo direito encontra-se a RV e no direito o ambiente real. A RA, sendo obtida a partir da mistura do mundo real com elementos virtuais, encontra-se também mais próxima do mundo real. Por outro lado, é apresentado o termo Virtualidade Aumentada (VA) que ocorre quando o utilizador se encontra num mundo virtual, com a presença de elementos do mundo real. Um exemplo de VA é a utilização da animação e movimento de uma pessoa num elemento virtual em tempo real. Por último, a Realidade Mista (RM) acaba por ser o conjunto de todos estes tipos de realidades.

A RV e a RA tem já uma história longa, tendo os desenvolvimentos mais significativos sido o Sensorama [3] na década de 1950 para experiências cinematográficas (o equipamento de uso individual submetia os utilizadores a movimentos, sons, odores e visão estereoscópica) e a Sword of Damocles [4] (figura 2), desenvolvida por Ivan Sunderland nos anos 60 e considerado por muitos como o primeiro *Head Mounted Display* (HMD) .

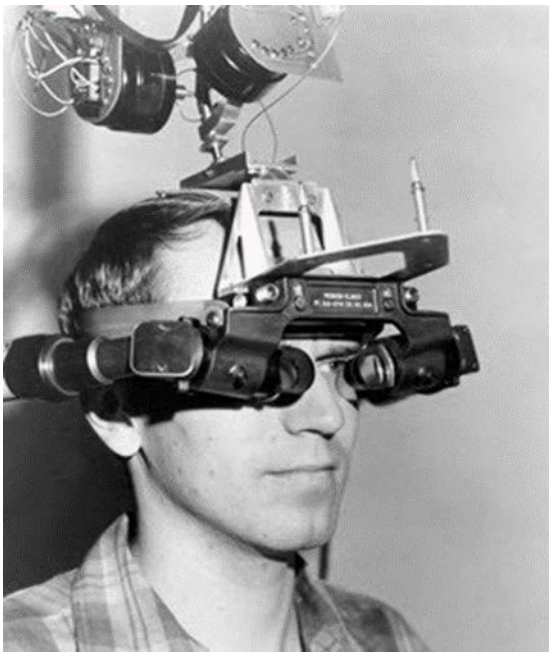

Figura 2- *The Sword of Damocles*

Nos anos 80 e 90 ocorreram também vários avanços significativos devido ao aumento da capacidade de armazenamento e de processamento dos dispositivos. Consequência destes avanços, em 1985 Jaron Lanier criou uma empresa para criar e investigar sistemas RV, chamada VPL Research. Esta empresa desenvolveu um aparelho em forma de luva [5] que permitiu captar os dados posicionais da mão do utilizador e reproduzi-los num ambiente virtual através da utilização de cabos de fibra ótica e sensores: a Data Glove. Mais tarde, em conjunto com a NASA, foi desenvolvido o Virtual Interface Environment Workstation [6]: um sistema que utilizava a Data Glove e permitia gerar a imagem da mão no ambiente virtual em tempo real. Com isto, era possível interagir com o ambiente através de gestos e movimentos do utilizador, movendo objetos virtuais.

Mais tarde, no ano de 1993 a empresa SEGA anunciou um novo produto de RV. O Headset SEGA VR disponibilizava som stereo, teclas para interação e rastreamento. Apesar de a empresa ter desenvolvido jogos para este aparelho, não passou da fase de protótipo. Igualmente a empresa Nintendo decidiu seguir a tendência da RV existente na altura e desenvolveu o Nintendo Virtual Boy, uma consola portátil que exibia conteúdo em 3D.

Na última década, ocorreu um forte crescimento na área com o lançamento do aparelho Oculus Rift desenvolvido pela empresa Oculus VR que representa a última vaga de aparelhos de RV e impulsionou o lançamento de vários dispositivos como o Google Cardboard lançado pela Google em 2014, um HDM de baixo custo (construído em cartão) para uso de smartphones. Mais tarde em 2016, foi lançado o jogo Pokémon GO que originou um aumento significativo no número de utilizadores da RA. Este jogo utilizava o GPS e a câmara dos dispositivos e permitia aos jogadores treinar, batalhar, e capturar criaturas virtuais, apresentados nas telas dos dispositivos como se estivessem integrados no mundo real.

Atualmente a RV e a RA encontram-se em fase de crescimento, tendo conhecido uma grande expansão nas áreas de jogos, entretenimento, educação,

engenharias e medicina. Além de estar acessível a dispositivos como HMD, com a crescente evolução da capacidade de processamento agora também é possível executar experiências de RV em smartphones o que dá uma maior acessibilidade à população. Com um potencial elevado, é expectável que a área continue a crescer nos próximos anos.

2.2 A-Frame

A-Frame [7] é uma *framework web based* desenvolvida pela equipa de RV da Mozilla para WebGL. Tem como base um sistema entidade-componente que permite aos programadores criar cenas de RV usando as bibliotecas Three.js e WebXR Device API na linguagem de programação Javascript.

A equipa por detrás da WebXR Device API combinou RV e RA e integrou as tecnologias XR nos nossos navegadores. Esta API é uma descendente da WebVR, criada para resolver os principais problemas da plataforma, focando-se nos problemas de construção dos programadores. As suas capacidades permitem trabalhar com hardware de RM (como auscultadores e óculos RV com características de RA integradas) e tanto possibilita a gestão do processo de renderização da experiência 3D, como a capacidade de sentir o movimento através de sensores. Por outro lado, Three.js é uma biblioteca JavaScript usada para criar e apresentar gráficos 3D animados para *web*.

As principais vantagens da plataforma A-Frame são a interoperabilidade na *web*, que significa que é compatível entre dispositivos, navegadores e sistemas operativos. Além disso, a acessibilidade é um ponto muito forte a seu favor, pois muitos programadores não têm familiaridade com linguagens de programação mais avançadas. A linguagem HTML fornece uma ferramenta familiar e acessível para programadores, e pode incluir código em JavaScript para experiências mais elaboradas. Tendo em conta as linguagens utilizadas, torna-se fácil de programar e de analisar o código em qualquer editor de texto. Assim, os programadores podem importar bibliotecas e desenvolver experiências, abstraindo-se da criação do conteúdo que estas oferecem. O seu conteúdo pode ser de RV, RA, áudio, vídeo, interações e sistemas de partículas e bibliotecas de reconhecimento de padrões em marcadores visuais. A criação é constante por parte dos utilizadores da framework sendo que é possível partilhar componentes de forma a criar entreajuda entre a comunidade de programadores.

A comunidade tem acesso a diversos meios de comunicação como Slack e Discord, Github e Stack Overflow. Além disso, a documentação e tutoriais disponibilizados pela equipa do A-Frame é muito completa. Por último, o facto de permitir que o conteúdo seja exibido em *browsers* facilita também a acessibilidade do utilizador final das experiências, que não precisam de adquirir aparelhos específicos para visualização de RV e RA e podem assim usar o seu *smartphone* pessoal.

No que diz respeito à programação em Javascript, a *framework* possui um *lifecycle* definido com vários métodos que são executados por uma ordem previamente definida (tabela 1).

Tabela 1- Métodos do lifecycle da framework A-Frame

Método	Descrição
Init	Chamado uma vez apenas, quando o componente é inicializado.
update	Chamado quando o componente é inicializado ou quando as propriedades são alteradas.
remove	Chamado quando o componente é removido da entidade ou da cena e remove todas as modificações feitas à entidade.
tick	Chamado no início de cada renderização da cena. É particularmente usado para fazer mudanças e verificações constantes.
tock	Chamado no final de cada renderização da cena. Usada para processamento que necessite de ser feito após a cena estar renderizada.
play	Chamado para iniciar um comportamento dinâmico na entidade. É tipicamente usada para começar ou recomeçar comportamentos e é sempre chamada pelo menos uma vez na inicialização da cena.
pause	Chamado quando a cena ou entidade pausa para remover comportamentos. Também é usado quando o componente é removido da cena.
updateSchema	Chamado quando as propriedades do componente são atualizadas.

As propriedades de um componente (tabela 2) podem ser definidas através da estrutura *schema*. Estas, funcionam como argumentos de uma função, e podem ser definidas diretamente no código HTML. Caso não seja definida, é aplicado um valor predefinido, aplicado no código.

Tabela 2- Tipos de valores nas propriedades da framework A-Frame

Propriedade	Descrição
Array	Lista de valores separados por vírgula.
Asset	URL's que apontam para *assets* como texturas ou imagens.
Áudio	Igual ao tipo Asset mas especificamente do tipo áudio.
Boolean	Valores booleanos.
Color	Cores.
Int	Inteiros.
map	Igual ao tipo Asset mas especificamente para texturas.
model	Igual ao tipo Asset mas especificamente para modelos.
number	Números com virgula flutuante.

selector	Uma entidade.
selectorAll	Todas as entidades.
String	Valores textuais.
Vec2	Um objeto com 2 propriedades: x e y.
Vec3	Um objeto com 3 propriedades: x, y e z.
Vec4	Um objeto com 4 propriedades: x, y, z e w.

2.3 Marcadores

A utilização de marcadores é geralmente associada a RA. Porém, a sua utilização não é exclusiva a esta área visto que são objetos de fácil acesso e consequentemente muito usados nas diversas áreas do "Continuo real-virtual". A sua utilização tem como objetivo facilitar o processamento dos algoritmos de visão computacional ajudando no rastreamento posicional. A sua utilização torna possível a utilização de hardware de mais baixo custo, como processadores e câmaras de menor qualidade.

Os marcadores são imagens com uma moldura e uma figura no seu interior (figura 4), que permite a distinção entre eles e o seu rastreamento posicional por parte de softwares próprios para o efeito. Podem ser impressos em papel, cartão, cartolina, ou qualquer outro objeto físico. As suas principais vantagens são a facilidade de aquisição, utilização. Por outro lado, a sua principal desvantagem é a sensibilidade relativa à iluminação que cria falhas no rastreamento do marcador.

Figura 4- Exemplo de marcador

Capítulo 3 Objetivos e Metodologia

O principal foco desta dissertação é o desenvolvimento de um toolkit programático focado no desenvolvimento de aplicações de RV com utilização de interfaces tangíveis detetados através de marcadores visuais. De forma a dar um contributo importante para o crescimento da área da RV, é importante que o toolkit se mostre util e consiga facilitar a programação de experiências de complexidade mais elevada. Após o desenvolvimento de componentes pretende-se criar experiências que demonstrem o potencial do toolkit utilizando as interações programadas. Para garantir a utilidade do toolkit e satisfação dos utilizadores é necessário avaliar a sua usabilidade através de testes e recolher o feedback dos programadores testados.

3.1 Plano de trabalho

O principal objetivo deste projeto é o desenvolvimento de um *toolkit* que crie abstrações a programadores de conteúdos de RV no que diz respeito a interações com interfaces tangíveis. Assim sendo, é essencial conhecer os trabalhos e estudos feitos na área, tal como as tecnologias existentes, de forma a identificar as limitações e potencialidades, tal como projetar realisticamente os conteúdos do toolkit.

O primeiro passo foi estudar o estado da arte. Isto permitiu enriquecer o conhecimento acerca da área da RV, desde os seus primeiros desenvolvimentos ao estado atual e aplicações futuras que poderá ter. O estudo dos *toolkits* de interação e dos projetos desenvolvidos relacionados com a área permitiu conhecer uma maior variedade de abordagens e técnicas de interação utilizadas por outros investigadores. Inicialmente foram procuradas formas de abstração possíveis no desenvolvimento destas interações com o objetivo de as incluir como funcionalidades do *toolkit*, saindo depois do âmbito da dissertação de Jorge Ribeiro [1] e procurando outras abstrações possíveis e abrindo horizontes a possíveis novas técnicas e interações estudadas do estado da arte.

De seguida, foram analisadas algumas plataformas de desenvolvimento de forma a decidir qual seria a mais vantajosa no sentido de cumprir os objetivos principais da dissertação e estudada a forma de desenvolver componentes na plataforma escolhida. A plataforma de desenvolvimento escolhida foi a *framework* A-Frame devido às suas características *web*. O fato de ser orientado a *web* aumenta o público de utilização, visto que torna possível a visualização de experiências em smartphones, tornando as experiências mais acessíveis ao público. Da mesma forma, também aumenta o número de programadores confortáveis com a complexidade programática.

Outras plataformas muito usadas na programação de RV, como Unity e Unreal Engine, já têm muitos componentes e conteúdos desenvolvidos por grandes empresas, em contraste com A-Frame que é uma *framework* recente que tem crescido principalmente devido a entusiastas.

Após terminar o estudo do estado da arte foi desenvolvida a arquitetura da biblioteca, tendo como referência inicial as interações estudadas. No primeiro semestre foi

desenvolvida um dos componentes do *toolkit* que deteta a interação de abanar de um tangível : o Shake Detector. No segundo semestre foram desenvolvidos os restantes componentes, os testes de usabilidade e o desenvolvimento de experiências demonstrativas das capacidades da ferramenta.

3.2 Planeamento temporal

O planeamento temporal inicial (figura 5) previsto para a realização desta dissertação, está apresentado na figura seguinte, de modo assegurar com antecedência a participação de terceiros e assegurar que o projeto é desenvolvido no tempo previsto. O planeamento está divido em 5 partes:

1. Relatório: Consiste no desenvolvimento dos relatórios intermédio e final. Foram previstos os tempos de desenvolvimento de cada uma das secções.
2. Desenvolvimento de componentes: O desenvolvimento foi dividido em várias partes, cada uma correspondente aos componentes do toolkit incluindo a documentação associada a cada um.
3. Experiências RV: Desenvolvimento de experiências de RV que demonstrem as potencialidades do dos componentes.
4. Testes de usabilidade: Sessões de testes para avaliar ao desenvolvimento de interações com os componentes.
5. Reuniões: As reuniões com o orientador para atualização do processo de desenvolvimento.

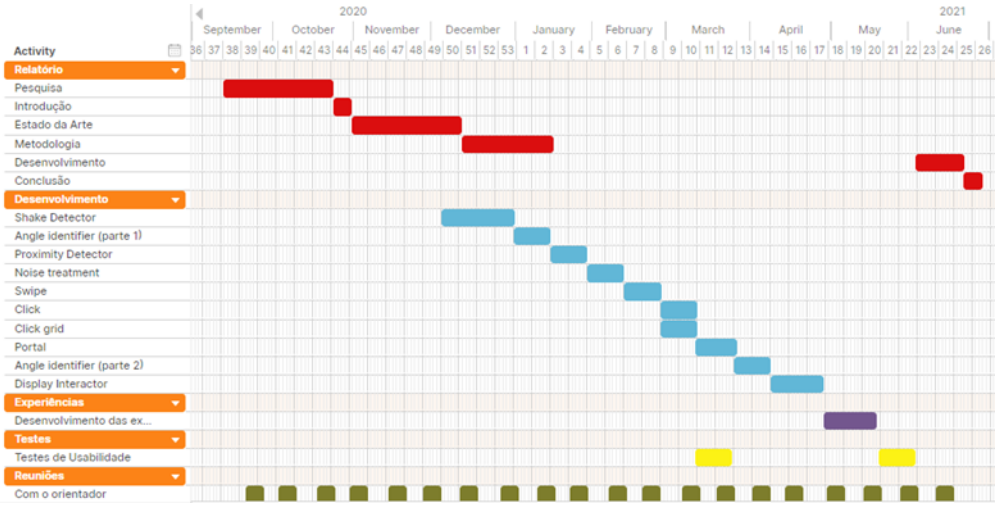

Figura 5- Planeamento temporal inicial

O planeamento temporal do projeto sofreu algumas alterações:

- O desenvolvimento do componente Proximity Detector foi cancelado pois já estava desenvolvido e o Click Grid (que pretendia detetar cliques numa matriz de marcadores) também devido às dificuldades encontradas na deteção e rastreamento de múltiplos marcadores de forma eficaz.

- O componente Portal (que pretendia permitir a personalização do elemento portal do A-Frame) não foi desenvolvido por sair demasiado do âmbito do projeto e não incluir qualquer tipo de interação direta.
- Partindo da conclusão de que o Display Interactor seria mais próximo de uma experiência virtual do que de um componente, o seu desenvolvimento foi também cancelado e foi desenvolvida no seu lugar uma experiência (Album Viewer) que utiliza as funcionalidades da biblioteca.

Quanto às sessões de testes de usabilidade, também sofreram desvios da data devido à dificuldade de realizar testes presenciais com os utilizadores devido às condições pandémicas existentes e consequentemente mudança do plano de testagem inicial. Devido a estas alterações, o planeamento temporal foi restruturado, tendo como resultado final o representado na figura 6.

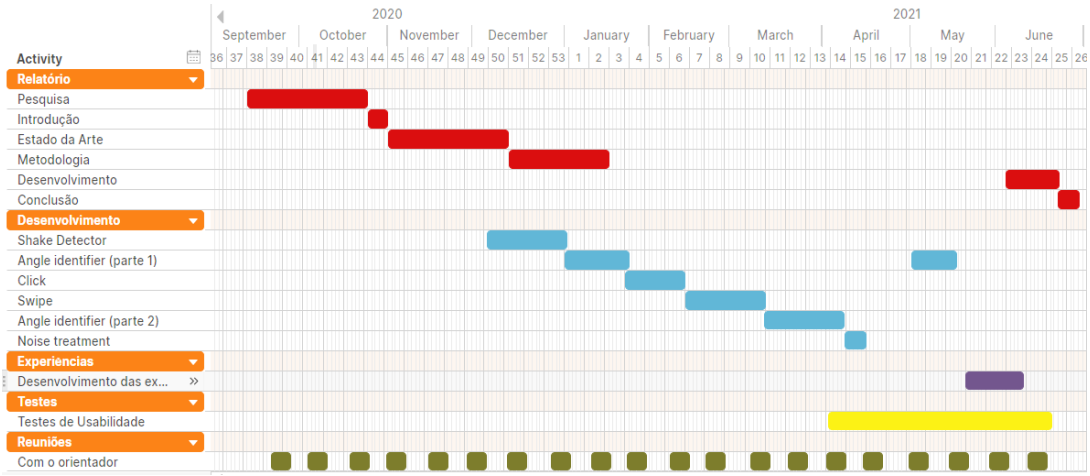

Figura 6- Planeamento temporal final

Capítulo 4 Estado da Arte

4.1 Projetos de interação em RV com marcadores

Interação em Realidade Virtual Através de Tangíveis Passivos

Neste projeto, Jorge Ribeiro [1] explora a utilização de interfaces tangíveis para interação em ambientes de RV, fazendo um rastreamento através de marcadores visuais. Sendo a presente dissertação inspirada nesta, ela é de bastante importância no sentido em que contêm muitos exemplos de interações com tangíveis e, porque foi a partir das dificuldades que surgiu a necessidade de criar abstrações programáticas para este tipo de interações. O principal objetivo foi criar diferentes experiências e formas de interação em ambientes virtuais através da *framework* A-Frame, de forma a explorar de que forma a interação com tangíveis passivos pode beneficiar a experiência com um sistema RV. Os primeiros protótipos desenvolvidos trataram a propriedade de rotação dos marcadores e consistiram num livro de três páginas (de forma a explorar a interação de mudar de página) e uma garrafa com um marcador posicionado no exterior, que permitia a rotação do objeto em ambiente virtual consoante a do mundo real (figura 7).

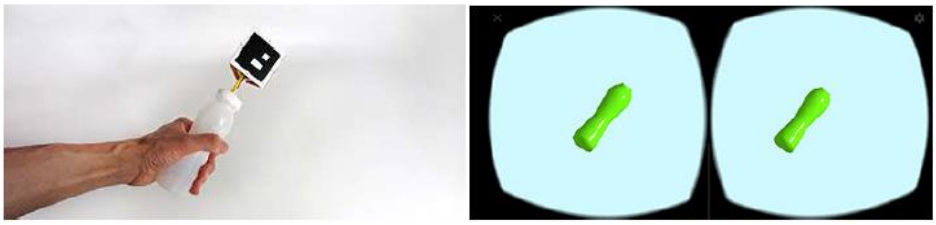

Figura 7- Protótipo para interação com rotação

Mais tarde, foram explorados a personalização dos objetos tangíveis, tendo o autor construído um protótipo de torre com sino, uma porta na qual explora a interação de abrir, e cubos com pesos associados de forma a explorar uma interação que proporciona realismo. A partir do protótipo da torre com sino (figura 8) revelou-se a primeira dificuldade programática, na deteção de abanar o sino e despoletar um som característico.

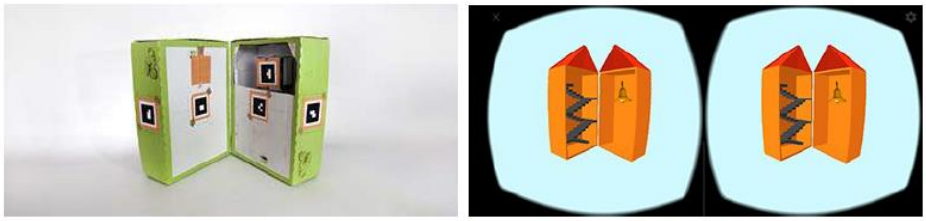

Figura 8- Protótipo de torre com sino

Também foi explorado o conceito de interface tangível com uma placa multimédia e de diversos tipos de interações mecânicas como um controlo deslizante e clique através da oclusão de marcadores. Outros métodos de interação foram desenvolvidos com base na rotação de marcadores associada ao controlo de variáveis como luminosidade do ambiente. Esta interação pode ser generalizada por um componente de deteção de ângulo do marcador, sendo que através dele podem ser aplicadas vários tipos de interação de controlo de variáveis dentro do ambiente.

A interação de clique (figura 9) e deslize (figura 10) podem ser generalizadas em componentes reutilizáveis, baseados na oclusão de marcadores que permitam o desenvolvimento de várias interações dentro da experiência de forma mais simples. Estes são exemplos de tipos de interação que podem ser abstraídas pelo programador. Com eles, o tempo necessário para o desenvolvimento da experiência seria menor.

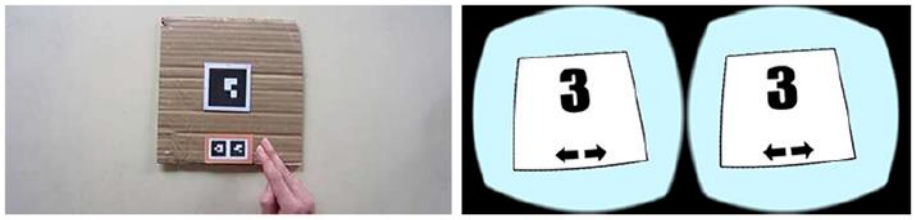

Figura 9- Interação de clique

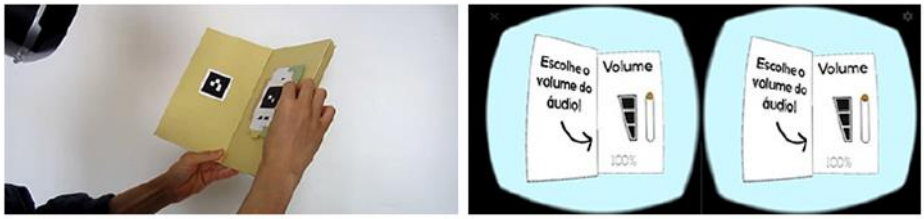

Figura 10- Interação de deslize

A prototipagem permitiu identificar diversos exemplos e foi desenvolvido um livro virtual que permite ao utilizador interagir com o sistema de várias formas e assim controlar os conteúdos multimédia apresentados no ambiente virtual, através de uma experiência de RV.

Numa segunda fase foram criados o projetos VR Book [8] nas versões Universidade de Coimbra e Mosteiro de Santa Cruz. Este, consistiu num livro com vários tipos de conteúdo virtual em cada página. As páginas são detetadas através de marcadores, tal como as opções dentro de cada página. Além de conteúdo estático como texto e imagens, vídeos e portais, o livro contém modelos 3D de alguns monumentos, que tornam este protótipo muito interessante do ponto de vista de interações em ambiente virtual. As interações aqui aplicadas, tais como cliques e deslizes para mudança de página, mostram que sendo uma interação muito útil, deve ser abstraída de forma a agilizar o processo de desenvolvimento. Também a interação de aproximação ao portal (figura 11) pode ser abstraída por uma componente de deteção de proximidade que permite ao programador implementar diretamente o comportamento que pretende com a aproximação.

Figura 11- Interação de proximidade para entrada em portal

Em conclusão, este projeto tem um vasto conjunto de interações que inspiram a criação de abstrações, tais como as interações de clique, deslize, a rotação em torno do objeto a controlar uma variável e até a interação de proximidade com portais.

Opportunistic Controls: Leveraging Natural Affordances as Tangible User Interfaces for Augmented Reality

Neste trabalho [9], Steven Henderson e Steven Feiner exploram um novo conceito de interação com interfaces tangíveis, aproveitando superfícies já existentes que contenham carcteristicas que permitam a interação, como paredes sem mobília e acessórios próximos. A partir destas superficies, foram feitas modificações no que toca ao design de forma a facilitar o reconhecimento gestual, táctil com o auxílio de marcadores visuais. A lógica do sistema utiliza três tipos de objetos para funções distintas. O primeiro tipo são objetos que se pareçam a botões, como parafusos e cavilhas procurando pequenos furos, covas ou a interseção de arestas. O segundo tipo de objeto são as superfícies lineares ou curvas, pois permitem a definição de valores em escala. Objetos deste tipo podem ser cantos de paredes, canos ou cordas sendo que o rastreamento destas interações requer um maior processamento e precisão. Por último, o terceiro tipo de objetos são objetos móveis que podem rodar, deslizam ou dobram como por exemplo anilhas giratórias e mangueiras (figura 12). Estes objetos permitem um controlo mais rico devido à flexibilidade da sua geometria.

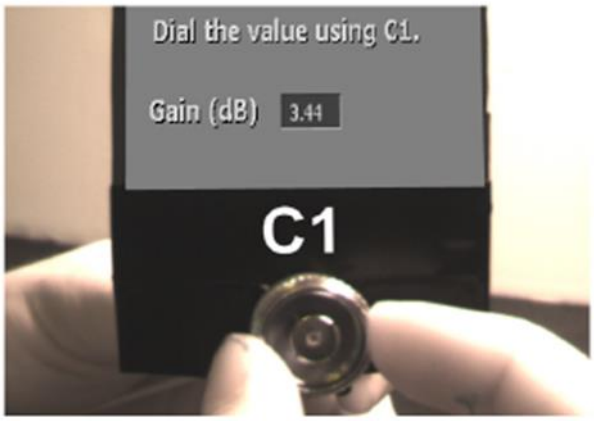

Figura 12- Rotação de objeto que permite a alteração de variável

O reconhecimento gestual é feito continuamente em três fases, redução de dados, correspondência gestual e análise gestual. Na primeira fase são acompanhados todos os movimentos da mão do utilizador, e de seguida comparados com os valores de cada movimento possível na área física do utilizador. Desta forma, certos tipos de movimentos só são possíveis em determinadas áreas, verificadas pelo algoritmo através do sistema de posicionamento espacial. Na figura 13, está exemplificada uma área desse tipo. No último passo, o gesto realizado pode ser ou não identificado através da correspondência por uma máquina de estados finita.

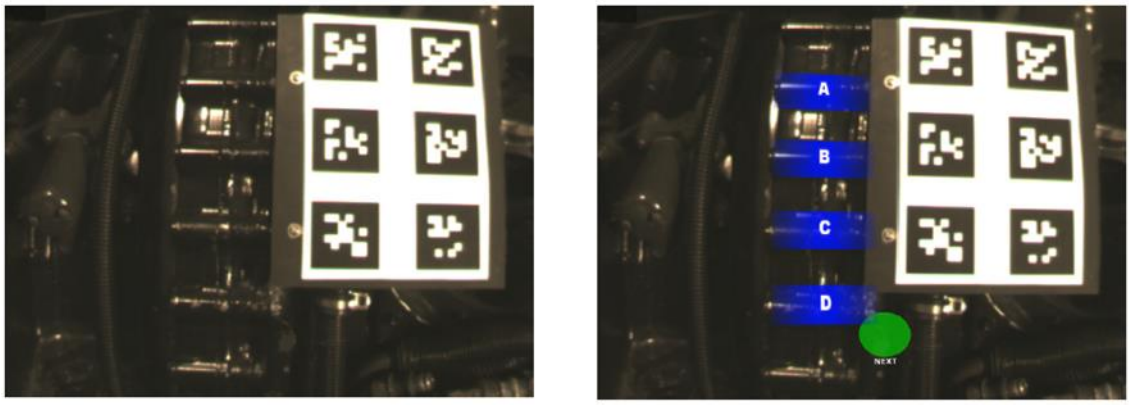

Figura 13- Exemplo de aproveitamento de interface tangível

De forma a testar este sistema, os autores simularam a operação de inspeção de um avião e os resultados mostraram que o sistema ajudou os participantes a concluir as tarefas designadas mais rápido e que a sua satisfação foi notória comparando ao método tradicional que usavam.

Tangible User Interaction Using Augmented Reality

Neste trabalho Slay, Thomas e Vernik [10] decidiram focar-se na criação de novos dispositivos de interação usando dispositivos criados através de marcadores visuais. Foi então desenvolvida uma aplicação de RA (ARVIS) que permite aos utilizadores visualizar e interagir com modelos tridimensionais para apresentar informação. A aplicação desenvolvida permite fazer um rastreamento em tempo real, das propriedades de posição e rotação dos objetos virtuais associados a marcadores. Segundo os autores, duas técnicas fundamentais de seleção e manipulação de objetos em ambientes virtuais são *ray-casting* e *arm-extension*. A primeira consiste em projetar um raio a partir da mão do utilizador e a segunda consiste numa virtualização dos braços, o que possibilita a extensão para chegar a objetos mais longínquos. No entanto, sendo que a investigação se foca na utilização de marcadores visuais, neste caso os objetos estão dentro do alcance do utilizador e podem ser rodados ou movidos fisicamente através do seu respetivo marcador.

Através da utilização única de marcadores foi desenvolvido um menu e a capacidade de alterar atributos dos objetos virtuais, tal como um mecanismo de interruptor. Neste menu, cada marcador corresponde a uma opção que está ativa desde que o marcador esteja visível. De seguida, foi implementada uma interação que simula o comportamento de interruptores, no qual os marcadores tinham um padrão em cada face. Desta forma, ao virar o marcador ao contrário seria ativada a opção oposta (figura 14).

Figura 14- Menu com marcadores

No que toca à interação de seleção de objetos virtuais, a primeira abordagem consistiu na utilização de um marcador que projetava um ponteiro a partir do centro. Após a tentativa de implementação de uma Magic Pen [11] esta foi descartada e foi continuado o desenvolvimento do primeiro protótipo de marcador projetor. Como tal a seleção de objetos virtuais foi feita através do aparecimento de marcadores em cena, cada um como um elemento de seleção definido (figura 15).

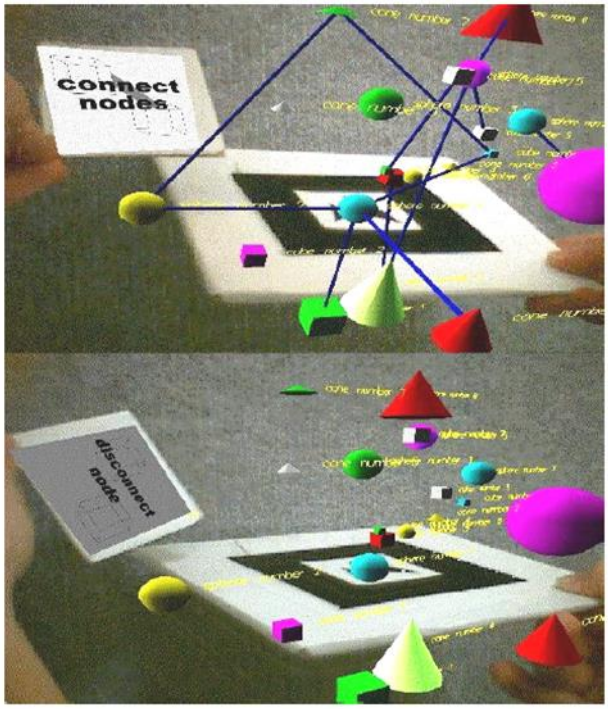

Figura 15- Interação através da oclusão de marcadores

Por último, ao nível das interações foi concebido um método de deteção de colisões que permitiu que foi mais tarde utilizado para permitir a passagem de elementos visuais entre marcadores.

Este trabalho mostra-nos uma abordagem diferente no que diz respeito aos métodos de interação baseados em oclusão. Em alternativa a marcadores sempre visíveis cuja oclusão é detetada, é implementado um menu cujas opções são selecionadas ao virar o marcador tornando-o visível. Desta forma, um marcador funcionaria como selecionador de opção, podendo ter um menu que não se fixava num pequeno espaço, mas que podia ser transportado por entre o espaço virtual. Por exemplo, ao ser mostrado o lado com um determinado padrão do marcador, o ambiente virtual teria um comportamento definido pelo programador.

Tangible VR: Diegetic Tangible Objects for Virtual Reality Narratives

Neste trabalho Harley, Tarun, Geriniario e Mazalek [12] constroem um sistema para objetos tangíveis em narrativas de RV. O sistema consiste numa unidade de sensores com *hardware* de baixo custo, que permite rastrear objetos em RV e suportar uma variedade de objetos tangíveis. Os quatro protótipos consistem num cubo, um peluche, uma arca do tesouro e um barco de madeira.

A capacidade de interagir com todas as seis faces do cubo (figura 16) e a capacidade de acariciar o peluche são interações personalizadas que beneficiam das qualidades tangíveis únicas desses objetos. As limitações do sistema desenvolvido sugerem um trabalho futuro e os autores propõem uma integração mais completa de tangibilidades em narrativas interativas. Segundo os mesmos, para isso, é necessário um rastreamento mais preciso do objeto. Por exemplo, a capacidade de pegar um objeto e colocá-lo para baixo novamente

requer que o sistema rastreie o objeto de forma consistente e precisa, mesmo quando ele não está a ser usado.

Figura 16- Interação com tangível

Além disso, também o rastreamento com marcadores sofre com a oclusão (maioritariamente pela mão) e dependendo da sua colocação e tamanho, os marcadores também podem interferir com as qualidades tangíveis dos objetos. Os autores sugerem então que, as limitações de rastreamento dos sistemas óticos possam ser aliviadas utilizando rastreadores eletromagnéticos de alta precisão que rastreiam objetos sem ser necessária uma linha de visão. Este trabalho mostra que a tangibilidade melhora as experiências RV no que toca á imersividade e realismo.

Two-handed tangible interaction techniques for composing augmented blocks

Neste estudo Lee, Billinghurst e Woo [13] propõem uma nova técnica de interação para RA tangível baseada num método de combinação de objetos natural e fácil de aprender, utilizando cubos tangíveis. A interface consiste em dois cubos tangíveis que são rastreados por marcadores e usando a interface, são possíveis dois tipos de interações baseadas em montagem de objetos reais. O primeiro, o método de aparafusamento (figura 17), reconhece os gestos de rotação do utilizador nos cubos e permite aparafusar objetos virtuais em conjunto. O segundo, o método *block-assembly* (figura 18*)*, adiciona objetos baseados na sua direção e posição em relação a estruturas predefinidas.

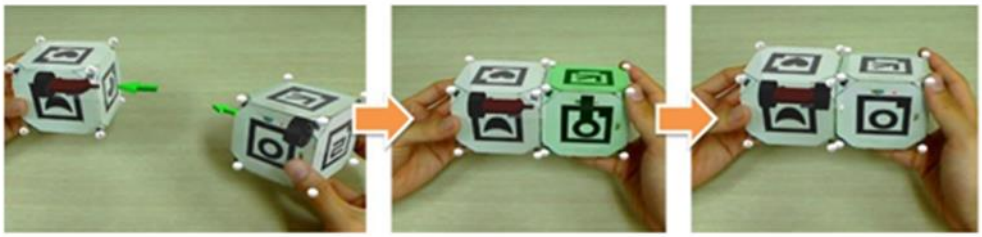

Figura 17- Método de aparafusamento

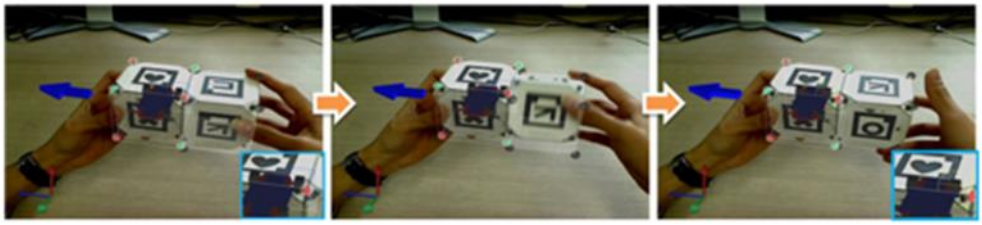

Figura 18- Método de *block-assembly*

Usando uma abordagem com vários marcadores, um marcador de rastreamento em cada face, um cubo fornece um rastreamento robusto com a rotação livre de blocos aumentados. Além disso, são usados vários botões sem fio que dão uma resposta mais rápida e estável do que o *input* baseado na oclusão. Foi então colocado um botão em cada canto do cubo para facilitar a usabilidade do sistema.

O *feedback* dado pelos utilizadores permitiu concluir que a utilização do ambiente era semelhante à montagem de blocos reais, por isso era familiar e fácil de usar. Também, os ímanes incorporados oferecem um *feedback* háptico realista em gestos de junção, separação e de rotação. De um modo geral, a maioria dos utilizadores ficaram satisfeitos com o sistema de *feedback* háptico e concordaram que é um fator que contribui muito para o realismo deste tipo de experiências.

Este foi um trabalho importante, pois explorou técnicas de interação tangível para desenvolver em RV ou RA, de uma diferente perspetiva. As interações desenvolvidas são muito versáteis, tendo um vasto conjunto apenas com dois objetos.

In Your Hand Computing: Tangible Interfaces for Mixed Reality

Neste projeto Dias, Santos e Nande [14] desenvolvem um sistema de controladores com interfaces tangíveis que permite fazer rastreamento posicional das mãos. Foram criados dois protótipos, Magic Bracelet e Magic Ring (figura 19), que consistem em marcadores visuais colocados nos pulsos e em dedos da mão de forma a rastrear o posicionamento destas partes do corpo e utilizá-las para interações. Este sistema permite usar o dedo para apanhar e largar qualquer objeto virtual colocado sobre um marcador ou, através da utilização dos anéis, realizar escalas e rotações através da variação da distância entre os dois. O anel também pode ser utilizado como cursor, projetando as coordenadas 3D do mesmo para coordenadas 2D na janela de visualização. Além disto, o sistema permite a identificação de gestos realizados pelas mãos, o que dá uma maior dimensão ao conjunto de interações possíveis com este mecanismo.

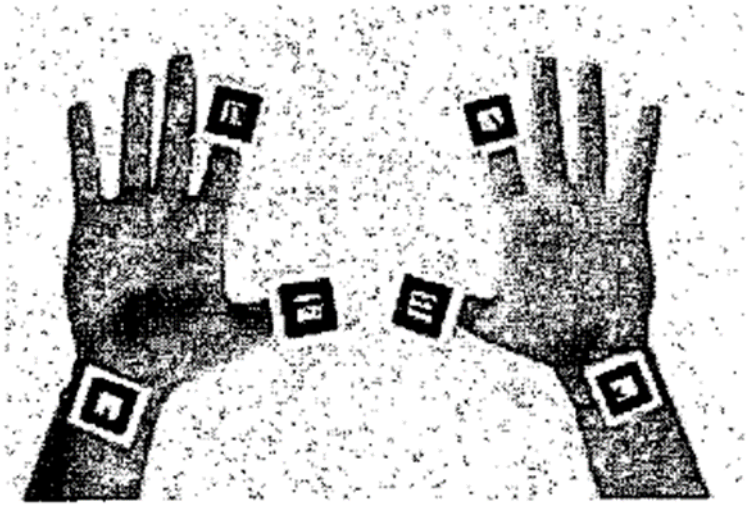

Figura 19- *Magic Rings* e *Magic Bracelets*

Apesar de simples, a utilização de marcadores no rastreamento das mãos é uma opção de difícil implementação devido á visibilidade reduzida resultado da constante movimentação e rotação. A ser implementada, esta interação teria de ser aliada a um componente de redução de ruído de visibilidade de marcadores. Desta forma, o componente guardaria a informação posicional dos marcadores relativa ao momento em que desaparecem de cena, alterando apenas as propriedades de posição e rotação apenas quando o marcador volta a ser identificado.

Occlusion based Interaction Methods for Tangible Augmented Reality Environments

Neste projeto Billinghurst, Lee e Kim [15] sugerem uma nova abordagem quanto aos métodos de interação convencionais em RA. Desta forma, sendo os métodos convencionais muitas vezes difíceis de implementar em RA tangível, eles optam por uma abordagem baseada na oclusão dos marcadores visuais. De forma a detetar a oclusão dos marcadores com eficiência, são utilizados marcadores de referência que suportam o rastreamento caso alguns dos marcadores estejam com problemas de visibilidade. Portanto, as falhas de rastreamento são aqui identificadas com dois casos: fora de vista e oclusão. De modo a detetar se o marcador está fora vista, são colocados alguns marcadores como delimitadores de um espaço (funcionando como marcadores híbridos) no qual sabemos á partida que estará o marcador analisado (figura 20).

Figura 20- Marcadores híbridos

Também é possível medir níveis intermédios entre marcadores, por exemplo numa situação em que dois marcadores consecutivos estão ocultos, assume-se que o ponteiro está entre os dois níveis dos respetivos marcadores. Este método é propício para a implementação de um *slider* tangível, por exemplo. Além disto, neste projeto é implementado uma forma de deteção do marcador selecionado num vasto conjunto de marcadores, assumindo que o selecionado é o marcador superior esquerdo ou direito (consoante o facto de o utilizador ser ou não destro) de entre os ocultos (figura 21).

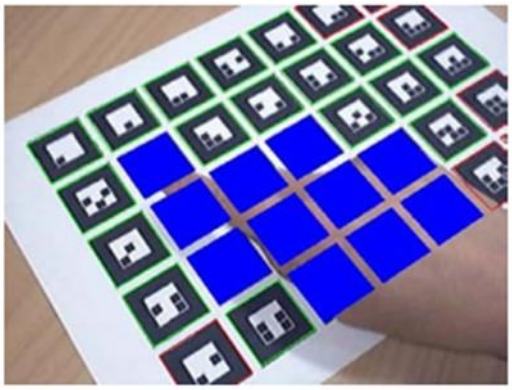

Figura 21- Interação de clique com painel de marcadores

As interações implementadas neste projeto são interessantes para o toolkit, na medida em que a sua abstração permite um vasto conjunto de interações diferentes. De forma a melhorar a eficiência das interações com oclusão, pode-se usar a primeira técnica de marcadores de referência para posicionamento. Por outro lado, a técnica de seleção de entre vários marcadores também é interessante para implementar como funcionalidade, na medida em que teria um difícil e demorado desenvolvimento e desta forma pode ser usado como uma simples funcionalidade.

Conclusão

A análise destes projetos permitiu obter algumas ideias de abstrações úteis que podem ser incluídas no toolkit. Em primeiro lugar, no projeto de Jorge Ribeiro [1], [8] as interações de clique, deslize e mudança de ambiente virtual através de portais são potenciais elementos da biblioteca. Por outro lado, o protótipo de marcadores que alteram

variáveis como luminosidade do ambiente através da oclusão pode ser abstraída para alterar o mesmo tipo de variável através da rotação do marcador em relação a um eixo.

O projeto de Slay, Thomas e Vernik [10] mostrou uma alternativa à interação de clique em que os marcadores despertam um comportamento pelo simples ato de aparecerem em cena. Podendo ser mantidos com o seu padrão virado para baixo, a interação de virar o marcador é simples e permite uma grande variedade de comportamentos, funcionando como um interruptor.

O trabalho de Lee, Billinghurst e Woo [15] apresentou uma forma de interação na qual é possível realizar diferentes comportamentos através de 2 ou mais objetos com vários marcadores. Além disso, apresenta uma forma de melhorar a implementação da interação de clique. O seu trabalho mostra uma forma de desenvolver esta abstração com vários marcadores, tal como uma solução para o problema de desaparecimento dos marcadores em cena poder ser associado a um clique (marcadores híbridos).

4.2 Toolkits de interação

Nesta secção vão ser apresentados os resultados de uma pesquisa por ferramentas ou toolkits de interação de RV. Esta pesquisa tem como principal objetivo a compreensão e descoberta das principais interações presentes neste tipo de ferramentas atualmente no mercado, tal como alguns aspetos técnicos das mesmas de forma a auxiliar na decisão do tipo de arquitetura e funcionalidades do *toolkit* a desenvolver. Como tal, a análise realizada para a utilização destas ferramentas será mais focada na utilização possível dos *toolkits* com objetos tangíveis.

Este tipo de ferramentas tem como objetivo auxiliar o programador de experiências/conteúdo RV e RA na criação do mesmo e são geralmente bibliotecas, Software Development Kits (SDK) ou API's que permitem um conjunto de funcionalidades, desde funções de criação a alteração do conteúdo em cena.

XR Interaction Toolkit

O XR Interaction Toolkit [16] é um sistema de interação de alto nível disponível na plataforma de desenvolvimento Unity. É baseado em componentes e a sua estrutura torna as interações disponíveis a partir de eventos de entrada. Estes eventos podem ser integrados através de botões e outros métodos de entrada dos controladores, mas também com o desenvolvimento de interações gestuais por parte dos programadores. O *toolkit* é composto por um conjunto de componentes que suportam um vasto conjunto de funcionalidades. No que diz respeito a interações mais básicas, é possível pairar, selecionar e agarrar objetos virtuais de forma simples para os programadores (figura 22).

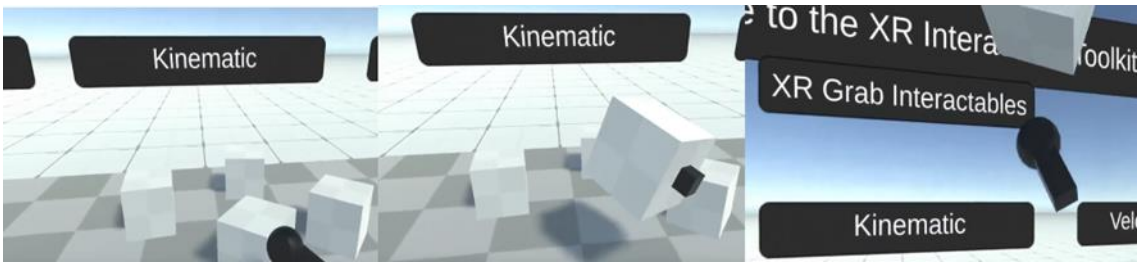

Figura 22- Interação de selecionar e agarrar objeto

O suporte para controladores também é uma funcionalidade importante, sendo que tem suporte para vários modelos, de diferentes plataformas permitindo *feedback* háptico (o controlador vibra, por exemplo) e interações básicas na interface de utilizador (UI). No que diz respeito às interações diretas no ambiente virtual, o toolkit permite o feedback visual de forma a indicar interações que estão ativas ou que podem ser feitas pelo utilizador (figura 23).

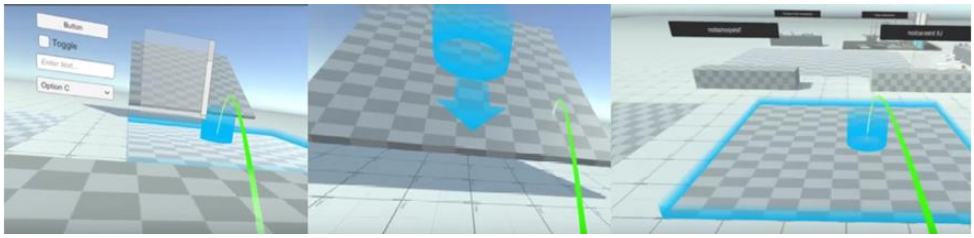

Figura 23- Exemplo de locomoção com *feedback* visual

No campo da RV, é possível interagir através de gestos, sendo que o toolkit proporciona abstrações neste aspeto, captando vários gestos automaticamente. Vários componentes permitem a colocação, seleção, locomoção, rotação e escala de objetos virtuais, tal como uma manipulação mais personalizada (juntando várias interações na mesma, por exemplo. O toolkit é composto por quatro componentes principais que trabalham em conjunto:

- Interactors: Permite a um objeto com este componente selecionar ou mover outro objeto
- Interactables: Permite ao utilizador interagir com um objeto com esta componente (agarrar, pressionar ou lançar, por exemplo)
- Interaction Manager: O componente que gere as interações entre os Interactors e Interactables em cena
- Controllers: O componente que torna eventos de entrada de controladores em eventos na cena (seleção por exemplo) e feedback háptico.

Os Interactors tratam das ações de movimentação e seleção de objetos no ambiente. Qualquer objeto com um componente Interactable está registado num Interaction Manager. Quanto aos métodos de entrada o toolkit possui a classe XRBaseInteractor que serve de base para as restantes classes de interação, como a XRSocketInteractor que tenta por defeito selecionar todos os objetos que estão a pairar. Por controladores, existe o script XRRayInteractor que permite interagir à distância através de técnicas de *Ray Casting*.

Por outro lado, os Interactables são objetos que um Interactor pode pairar, selecionar e ativar. Por defeito, todos os objetos com o componente Interactable podem ser alterados por todos os Interactores, podendo este comportamento ser alterado. O

Interactable define os comportamentos pairar, selecionar e ativar. O mesmo Interactor pode ser capaz de fazer apanhar e atirar uma bola, disparar uma arma, ou premir um botão 3D num teclado. Qualquer cena que utilize esta ferramenta, necessita um Interaction Manager (pelo menos) de forma a facilitar a interação entre os Interactors e os Interactables funcionando como intermediário. É possível, no entanto, ter vários Interaction Managers ativando e desativando cada um deles conforme a necessidade.

O componente Controller fornece uma forma de abstrair os dados de entrada dos controladores e traduzir em estados de interação. Os eventos de posição, rotação, seleção e ativação são ligados através de ações de entrada aos controlos de um dispositivo específico que os Interactores consultam.

No que diz respeito ao feedback háptico através dos controladores, é um pormenor interessante do ponto de vista de desenvolvimento com alguns objetos tangíveis. Para permitir o feedback háptico para um controlador, deve-se especificar uma ação no dispositivo. O Interactor pode então especificar intensidades e durações de feedback táctil para reproduzir nas interações escolhidas.

Em conclusão, esta ferramenta funciona com base em três componentes principais, que em conjunto conseguem proporcionar ao programador um vasto conjunto de funcionalidades de interação. Fazendo a ligação entre Interactors e Interactables, o Interactions Managers coordenam a execução dos métodos, proporcionando uma vasta gama de interações possíveis.

VRTK

VRTK [17] é uma coleção de *scripts* para ajudar a construir experiências RV de forma rápida e fácil em Unity3D. Abrange uma série de soluções comuns como movimentação dentro do espaço virtual como alguns estilos de teletransporte e saltos. Dentro do Unity, permite interações como tocar, agarrar, rodar, utilizar objetos e interagir com elementos da através de apontadores, por proximidade ou controlos 2D e 3D (botões, alavancas, portas…). Além disto, também suporta os tradicionais controladores, para os quais tem *scripts* de deteção de eventos para cliques, slide e rotação. Tal como o XR Interaction Toolkit (que também trabalha na plataforma Unity3D), este toolkit trabalha com Interactors e Interactables ao nível das interações. Todos estes *scripts* funcionam com base em colisões detetadas através das mecânicas do Unity, entre outros mecanismos.

Tabela 3- Métodos disponíveis para Interactors e Interactables

Interactors	**Interactables**
ControllerEvents	InteractableObject
InteractTouch	InteractableListener
InteractNearTouch	InteractHaptics
InteractGrab	ObjectAppearance
InteractUse	InteractObjectHighlighter
ControllerTrackedCollider	ObjectTouchAutoInteract
ControllerHighlighter	IgnoreInteractTouchColliders
ObjectAutoGrab	

No que toca aos Interactors, existe um script ControllerEvents cujo objetivo é auxiliar nas operações de interação através de controladores com diversas variáveis de controlo e eventos disponíveis para serem emitidos. No que toca a objetos virtuais, os *scripts* InteractTouch, InteractNearTouch, InteractGrab e InteractUse permitem ao programador desenvolver interações de toque e seleção dos objetos. Os *scripts* ControllerTracked e ControllerHighlighter servem para controlar colisões nos controladores e para colocar highlights no objeto virtual correspondente ao controlador, respetivamente. Por fim, o script ObjectAutoGrab serve para auxiliar nas interações de selecionar e "agarrar" automaticamente objetos que toquem no objeto controlador no ambiente virtual (figura 24), com uma aproximação definida pelo programador.

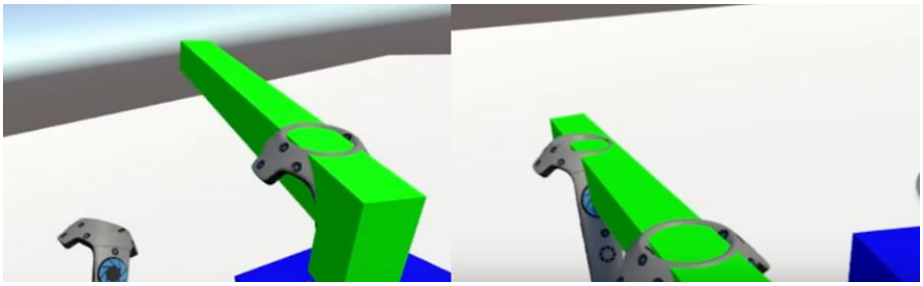

Figura 24- Interação de agarrar objeto com controladores

No que diz respeito aos Interactables, que são componentes sobre as quais se pode interagir, este toolkit disponibiliza um script InteractableListener que deve ser herdado por outros *scripts* de forma a escutar e interpretar eventos. InteractableObject serve para verificar o estado do objeto de forma a perceber se pode interagir. Por outro lado, IgnoreInteractTouchColliders permite fazer um objeto ignorar interações, ObjectTouchAutoInteract faz com que o objeto interaja automaticamente ao toque e InteractObjectHighlighter sinaliza o objeto dependendo do tipo de interação que este suporta, de modo a facilitar a experiência do utilizador. ObjectAppearance permite definir se o objeto está visível ou não consoante o tipo de interação ou mesmo no seu comportamento normal sem interação. Por fim, o VRTK possui o script InteractHaptics que permite uma série de mecanismos de interação com o utilizador que podem ser facilmente adaptados a tangíveis. Como o nome indica, este script refere interações hápticas e permite ao programador definir eventos através das interações de toque, agarrar e usar tanto tangíveis como objetos virtuais. É possível definir um som, fazer tremer um controlador com intensidade definida e ainda definir intervalos de uso e de interação.

ARCore

O ARCore [18] é um *kit* de desenvolvimento de *software* desenvolvido pela empresa Google, que permite a criação de aplicações em RA. Esta plataforma pode trabalhar em diferentes ambientes de desenvolvimento, como Unity3D, Android, iOS e Unreal Engine. O ARCore tem como as suas principais valências:

- **Rastreamento de movimento:** À medida que o dispositivo se movimenta, o ARCore utiliza um processo chamado SLAM (*simultaneous localization and mapping*), para compreender onde o dispositivo está relativamente ao mundo à sua volta. Também deteta características

visualmente distintas na imagem da câmara, chamadas pontos de característica e utiliza estes pontos para calcular a sua mudança de localização.

- **Conhecimento do ambiente:** A ARCore está constantemente a melhorar a sua compreensão do ambiente real através da deteção de pontos e planos. Desta forma, procurando aglomerados de pontos que parecem estar em superfícies horizontais ou verticais comuns, como mesas ou paredes, torna estas superfícies disponíveis para a sua utilização como planos. Essa informação pode ser utilizada para colocar objetos virtuais sobre superfícies planas.
- **Conhecimento de profundidade:** O ARCore pode criar mapas de profundidade, imagens que contêm dados sobre a distância entre superfícies a partir de um determinado ponto. Esta informação pode ser utilizada para permitir experiências imersivas e realistas do utilizador, como por exemplo fazer com que objetos virtuais colidam com precisão com superfícies observadas, ou fazê-los aparecer à frente ou atrás de objetos do mundo real.
- **Estimativa de luz:** Esta funcionalidade permite detetar informações acerca da iluminação do ambiente e fornecer-lhe a intensidade média e a correção de cor de uma determinada imagem da câmara. Esta informação permite-lhe iluminar os seus objetos virtuais nas mesmas condições que o ambiente que os rodeia, aumentando o realismo.
- **Interatividade:** Utilizando o toque para obter uma coordenada (x,y) correspondente ao ecrã do telefone, projeta um raio no mundo da câmara, devolvendo planos ou pontos que o raio intersecta, juntamente com a posição desse intersecção no espaço. Isto permite aos utilizadores selecionar e interagir com objetos no ambiente.
- **Pontos orientados:** Estas capacidades permitem colocar objetos virtuais em superfícies angulares. Quando se seleciona um ponto, são procurados os pontos que estão próximos e utilizados para estimar o ângulo da superfície no ponto em questão. Superfícies sem textura, como uma superfície branca, podem ter dificuldades em ser detetadas corretamente.
- **Imagens aumentadas:** Imagens Aumentadas são uma característica que permite responder a imagens 2D específicas, tais como embalagens de produtos ou cartazes de filmes. Os utilizadores podem desencadear experiências de RA quando apontam a câmara do seu telefone para imagens específicas. Por exemplo, podem apontar a câmara para um poster de um filme e um personagem desse filme aparecer.

ARKit

O ARKit [19] é a plataforma de desenvolvimento AR desenvolvida pela Apple para dispositivos iOS. Esta permite aos programadores de aplicações criar experiências de AR de maneira rápida e fácil. Tal como o ARCore, usa uma tecnologia chamada Odometria Inercial Visual para fazer rastreamento da área á volta do dispositivo, o que permite que o dispositivo iOS perceba a movimentação dentro de uma divisão e usa esses dados para detetar planos horizontais, como tabelas e pisos. As principais valências deste kit são muito semelhantes às do ARCore, com mais algumas características:

- **Âncoras de localização:** Permite ancorar as suas criações de RA em coordenadas específicas de latitude, longitude e altitude. Os utilizadores podem mover-se em torno de objetos virtuais e vê-los de diferentes perspetivas.
- **Rastreio facial:** Permite utilizar as funcionalidades de rastreio facial na câmara frontal do dispositivo e permite rastrear até três caras ao mesmo tempo.

- **Geometria de Cena:** Cria um mapa da sua divisão com etiquetas identificando pisos, paredes, tetos, janelas, portas... Este conhecimento do mundo real desbloqueia a oclusão de objetos e a física do mundo real para objetos virtuais.
- **Conhecimento de profundidade:** Utiliza informação de profundidade por pixel sobre o ambiente. Quando combinada com os dados de malha 3D gerados pela Geometria de Cena, esta informação de profundidade torna a oclusão de objetos virtuais ainda mais realista, permitindo a colocação instantânea de objetos virtuais e misturando-os sem problemas com o seu ambiente físico.
- **Captura de movimento:** Ao compreender a posição e o movimento do corpo como uma série de articulações e ossos, pode usar o movimento como um *input* à experiência AR
- **Deteção de objetos melhorada:** Deteta até 100 imagens de cada vez para obter uma estimativa automática do tamanho físico do objeto nas imagens. A deteção de objetos 3D é mais robusta, uma vez que são mais bem reconhecidos em ambientes complexos e a aprendizagem mecânica é utilizada para detetar planos no ambiente ainda mais rapidamente.

Zapbox

O Zapbox [20] é um kit de RM que inclui componentes físicos e software (figura 25). O utilizador pode experienciar conteúdo através de um *headset* para smartphone e controladores de cartão baseados em marcadores visuais. Os marcadores visuais disponibilizados que fazem parte dos controladores foram desenhados para serem detetados mais rápido na aplicação da Zapbox ainda que pequenos ou em grande número. Com o auxílio destes marcadores otimizados, a aplicação ZapBox é capaz de construir um mapa de onde estão todos os códigos de pontos da sua divisão, e depois usar esse mapa para calcular a posição exata do utilizador e direção de visualização usando os códigos de pontos que pode ver em qualquer altura.

Figura 25- Kit de marcadores do Zapbox

Tabela 4- Resumo das características dos toolkits estudados

	Plataforma	**Open source**	**Permite a utilização de marcadores visuais / tangíveis**
XR Interaction Toolkit	Unity	Sim	Não diretamente, mas pode ser integrado no Unity.
VRTK	Unity	Sim	Não diretamente, mas pode ser integrado no Unity.
ARCore	Android	Sim	Sim
ARKit	iOS	Sim	Sim
Zapbox	ZapWorks Studio	Sim, mas trabalha com o kit de tangíveis próprio.	Sim

Capítulo 5 VR Tangible Marker Interaction Library

A VR Tangible Marker Interaction Library [21] é uma biblioteca de componentes para RV em A-Frame. É composta por seis *scripts* da linguagem Javascript: Shake Detector, Button, Swipe, Angle Detectors (single e double marker) e Noise Controller. Um dos principais objetivos no seu desenvolvimento foi produzir interações simples e úteis em diferentes aspetos da experiência virtual. Interações discretas e contínuas produzem uma variada lista de comportamentos possíveis. Estes componentes emitem eventos com informação detalhada quando detetam interações do utilizador.

Nos componentes que assim o exigem, é disponibilizado um parâmetro que permite personalizar os objetos virtuais alvo destes eventos, tal como um parâmetro que permite ativar e desativar a depuração na consola.

5.1 Estrutura

Arquitetura

As experiências em A-Frame são desenvolvidas em HMTL e os componentes na linguagem JavaScript e seguem a arquitetura geral de A-Frame com os métodos e *lifecycle* já existentes. De forma a recolher dados e modificar o comportamento e aspeto dos objetos virtuais é utilizada a estrutura das bibliotecas Three.js [22] e WebXR [23]. Sendo uma framework baseado na three.js, o A-Frame tem acesso total á API desta.

De forma a fazer o rastreamento dos marcadores, foi utilizada a framework de RA AR.js [24], que permite reconhecimento e rastreamento de imagens em tempo real.

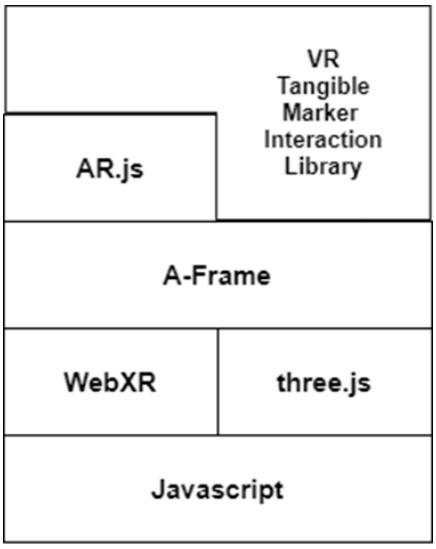

Figura 26- Diagrama das tecnologias por detrás dos componentes

Documentação

A documentação dos componentes [21] foi escrita de forma a ajudar os programadores a compreender e utilizar os mesmos. Esta, foi disponibilizada na plataforma GitHub e começa por apresentar uma explicação relativa ao componente e ao que pretende alcançar.

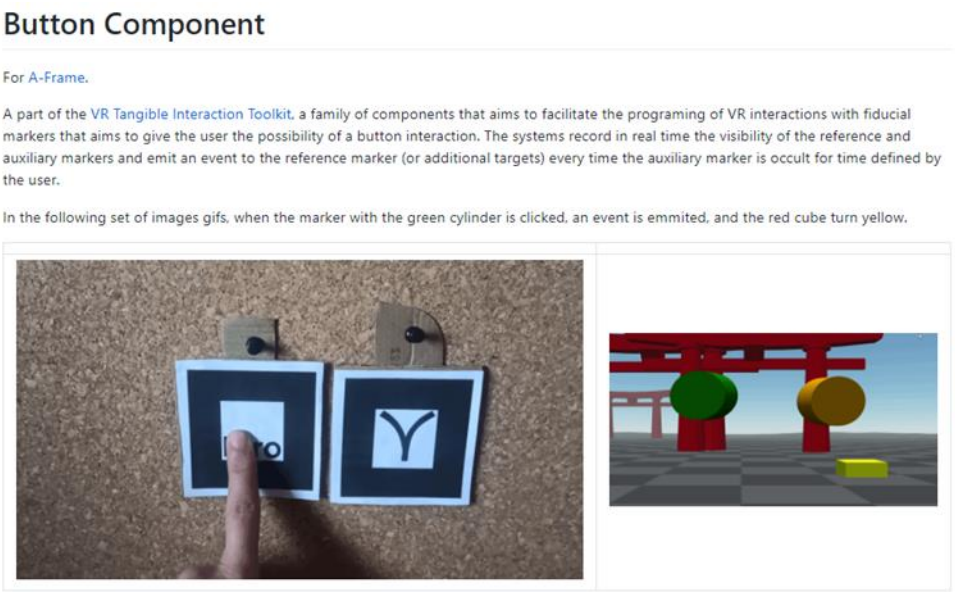

Figura 27- Exemplo da documentação do componente Button

De seguida, são explicados os parâmetros e um exemplo de utilização no código. Por último, também sobre forma de exemplo, são apresentados os eventos de cada componente de que forma podem ser utilizados. A documentação contém ainda ligações para exemplos mais completos.

Figura 28- Exemplo da segunda parte da documentação do componente Button

5.2 Shake Detector

O componente Shake Detector destina-se a detetar o movimento de abanar de um objeto tangível. Para detetar quando o marcador associado ao tangível está a ser abando, são detetados o número de mudanças de direção relativas ao movimento em cada um dos eixos definidos, a distância percorrida e o tempo entre estas mudanças de direção.

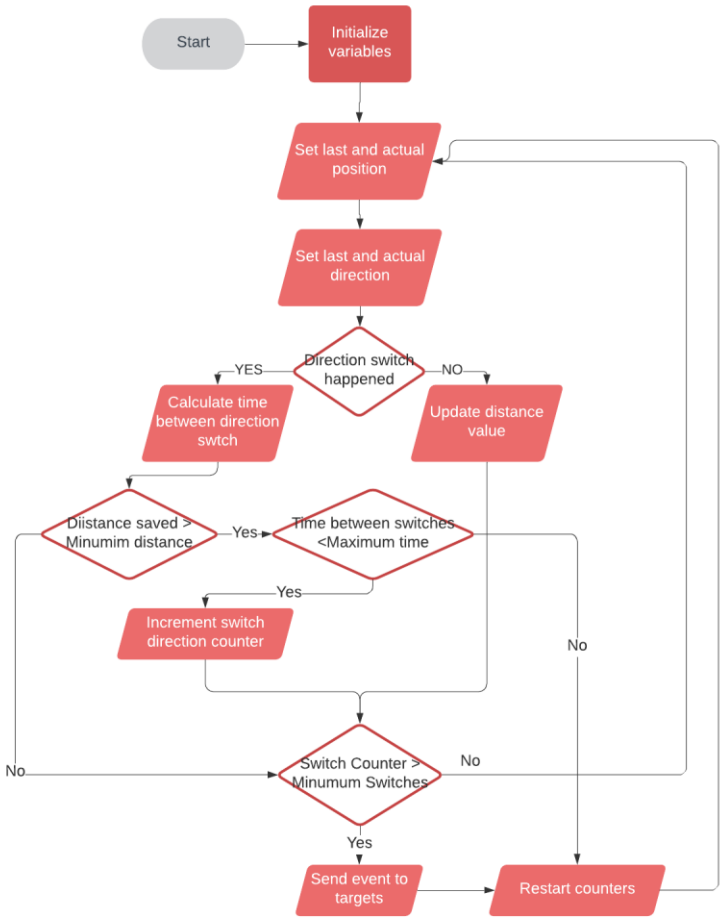

Figura 29- Funcionamento do algoritmo Shake Detector

Os programadores podem personalizar a distância (minimumDistance), o intervalo de tempo máximo entre as mudanças de direção (switchInterval), bem como o número mínimo de vezes que estas ocorrem (minimumSwitchTimes). Também podem ser definidos que eixos (axis) nos quais se pretende detetar: x, y ou z.

Tabela 5- Parâmetros do Shake Detector

Propriedade	Descrição	Tipo	Valor default
switchInterval	Tempo máximo (em ms) entre duas mudanças de direção.	int	1000
minimumSwitchTimes	Número de mudanças de direção.	int	3
minimumDistance	Distância virtual mínima para considerar o movimento válido.	float	0.5
eventTargets	Alvos opcionais para os eventos.	selectorAll	
axis	Eixos nos quais se pretende detetar a interação.	array	['x', 'y', 'z']
debug	Parâmetro optional para depuração.	boolean	false

Este componente pretende ajudar na criação de interações onde os utilizadores interagem com o ambiente virtual abanando objetos. Por exemplo, um utilizador que manipula uma representação tangível de um modelo arquitetónico de uma torre de igreja pode abanar o tangível e desencadear o som da campainha da torre.

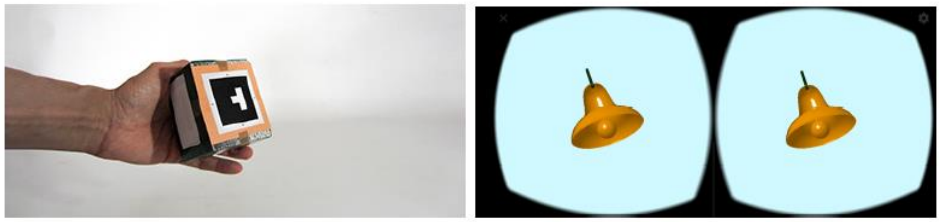

Figura 30- Representação tangível de um sino

No código o componente pode ser adicionado a um objeto que represente um marcador de forma simples, da seguinte forma:

```
<a-marker type="barcode" value="2"
          mt-shake-detector= "switchInterval:500; minimumSwitchTimes:3;
minimumDistance: 0.3; eventTargets: #id; axis:y; debug:true;">
</a-marker>
```

No que diz respeito aos eventos emitidos quando a interação é detetada, estes disponibilizam informação detalhada, nomeadamente o eixo e o tempo em que ocorreu o movimento tal como um identificador do objeto.

```
const event_shake = new CustomEvent('event_shake', {
    detail: {
      time: time,
      axis: elem,
      object: this.el
    }
})
```

5.3 Button

A implementação do componente Button inspira-se num trabalho já realizado que estuda métodos de oclusão de marcadores [15] e pretende utilizar um marcador como botão que pode ser clicado. O componente implica a utilização de dois marcadores: um serve como referência e deve ser visível para que a interação ocorra e o outro serve como botão no qual os utilizadores podem clicar tapando. A razão para a inclusão do marcador de referência é evitar a ambiguidade entre a oclusão do marcador de botão por sair de cena ou por ser propositadamente tapado pelo utilizador.

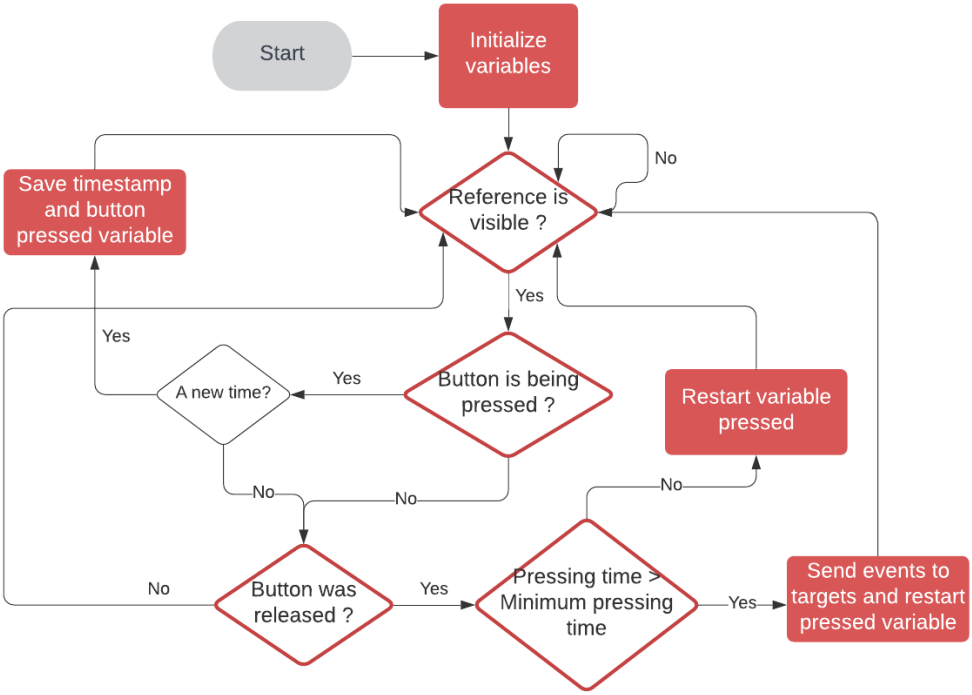

Figura 31- Funcionamento do algoritmo Button

Tabela 6- Parâmetros do Button

Propriedade	Descrição	Tipo	Valor predefinido
referenceMarker	Referência para outro marcador que precisa de estar visível para ocorrer a interação.	selector	
eventTargets	Alvos opcionais para os eventos.	selectorAll	
minimumTime	Tempo mínimo de oclusão.	array	['x', 'y', 'z']
debug	Parâmetro optional para depuração.	boolean	false

Numa experiência RV, esta interação pode ser facilmente incluída, atuando como um interruptor ou qualquer funcionalidade executada com um botão produzindo o respetivo comportamento, ou alteração de outras interações.

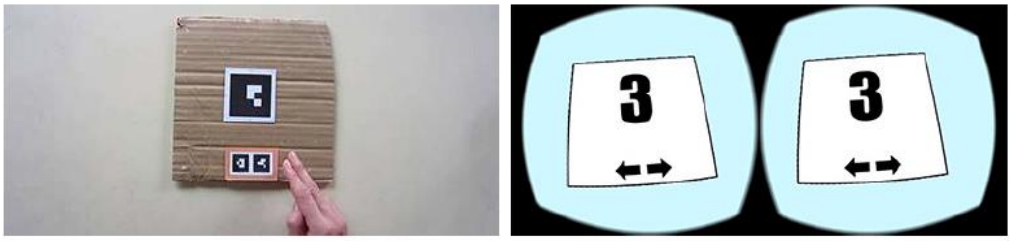

Figura 32- Exemplo de utilização do Button

No código, o componente pode ser associado a um objeto marcador, necessitando sempre de um novo objeto deste tipo para servir de referência. Os eventos emitidos quando a interação é detetada disponibilizam informação tal como um identificador do objeto clicado e um registo temporal.

```
<a-marker id = 'ref' type="barcode" value="11" a-marker>

<a-marker type="barcode" value="3" mt-button="referenceMarker: #ref; eventTargets:
#box; minimumTime: 100; debug: true;">
</a-marker>
```

Quando o marcador de botão é oculto durante um período de tempo predefinido configurável pelo programador, é emitido um evento característico deste componente.

```
const event_button_pressed = new CustomEvent('event_button_pressed', {
    detail: {
      time: time,
      object: this.el
    }
});
```

5.4 Swipe

O componente Swipe deteta interações de deslize numa sequência de marcadores. O algoritmo permite detetar o deslize através da verificação de sequências numa estrutura de dados ordenada (por tempo) que inclui as oclusões que aconteceram juntamente com o seu registo temporal. Esta estrutura é frequentemente verificada no sentido de remover as instâncias que ocorreram á mais tempo que o tempo máximo definido pelo utilizador. Após esta verificação e possível remoção de instâncias é feita a procura por ocorrência de sequências que pode ou não resultar na emissão de um evento.

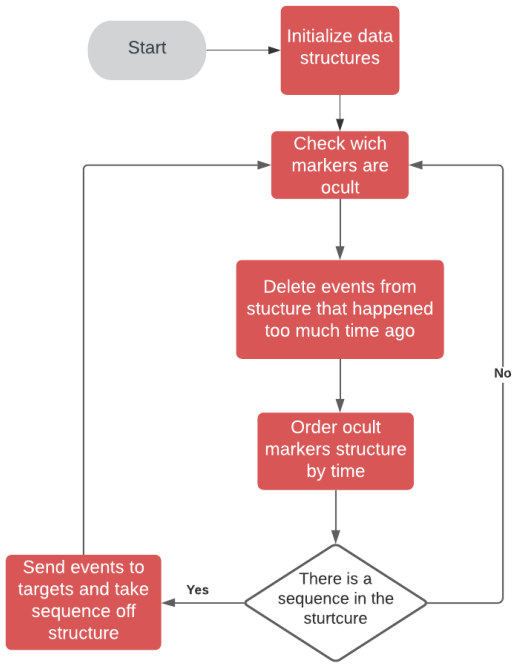

Figura 33- Funcionamento do algoritmo do Swipe

O programador define o conjunto de marcadores a serem utilizados, bem como as sequências a detetar, geralmente uma sequência para a esquerda e outra para a direita. Se os marcadores forem tocados numa ordem definida e dentro do tempo máximo definido, será emitido o evento.

Tabela 7- Parâmetros do Swipe

Propriedade	Descrição	Tipo	Valor predefinido
markers	Referência para outro marcador que precisa de estar visível para ocorrer a interação.	selectorAll	
sequences	As sequências a ser detetadas.	string	
eventTargets	Alvos opcionais para os eventos.	selectorAll	
maximumTime	Tempo máximo para a oclusão dos marcadores da sequência.	int	1000
debug	Parâmetro optional para depuração.	boolean	false

Uma das vantagens de utilizar este tipo de interação é a possibilidade de personalizar o comportamento tendo em conta a sequência detetada permitindo diferentes comportamentos nas mesmas interfaces tangíveis (deslizar para a esquerda, direita, para cima, para baixo). A figura 38 mostra e exemplo da utilização do componente Swipe onde três marcadores são utilizados para deslizar para a esquerda e para a direita para aumentar ou diminuir o valor da variável apresentada.

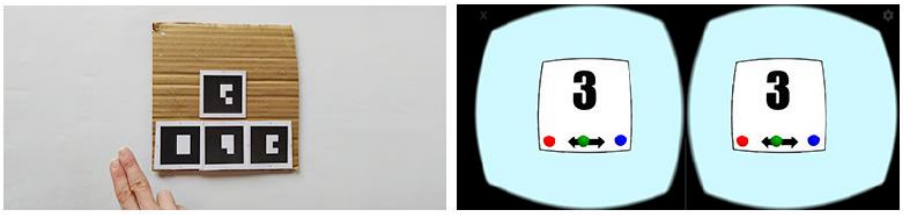

Figura 34- Exemplo de utilização do Swipe

No código, o componente deve ser associado ao marcador de referência que deve estar visível para a interação ser detetada à semelhança do componente Button. Na definição dos parâmetros deve atribuir-se a referência dos marcadores para a interação e das respetivas sequências a detetar e tempo máximo do movimento.

```
<a-marker type="barcode" value="1" mt-swipe = "markers: #my1, #my2, #my3;
eventTargets: #box1, #box2 ; sequences:1 2 3,3 2 1; maximumTime: 3000;  debug:
True;">
</a-marker>

<a-marker id="my1" type="barcode" value="1"></a-marker>

<a-marker id="my2" type="barcode" value="2" ></a-marker>

<a-marker id="my3" type="barcode" value="3" ></a-marker>
```

Os eventos emitidos contêm informação acerca de qual o marcador de referência, a sequência detetada e o seu índice na definição das sequências, juntamente com um timestamp.

```
const event_swipe = new CustomEvent('event_swipe', {
    detail: {
        time: time,
        object: this.el ,
        sequence: sequence,
        sequenceIndex: sequenceIndex,
    }
})
```

5.5 Angle Detector Single Marker

O componente Angle Detector Single Marker deteta a rotação de um marcador em relação a um ou mais eixos definidos. Este componente pode ser utilizado para alterar variáveis contínuas no espaço virtual, tais como luminosidade ou volume. O seu funcionamento passa por definir duas estruturas que guardam o angulo atual e o anterior. De seguida é calculado o angulo que ocorreu entre as duas iterações e este incrementa a uma última estrutura que guarda o angulo feito no conjunto das iterações. Caso esse angulo passe o valor do limite definido é reiniciada a estrutura e um evento é emitido.

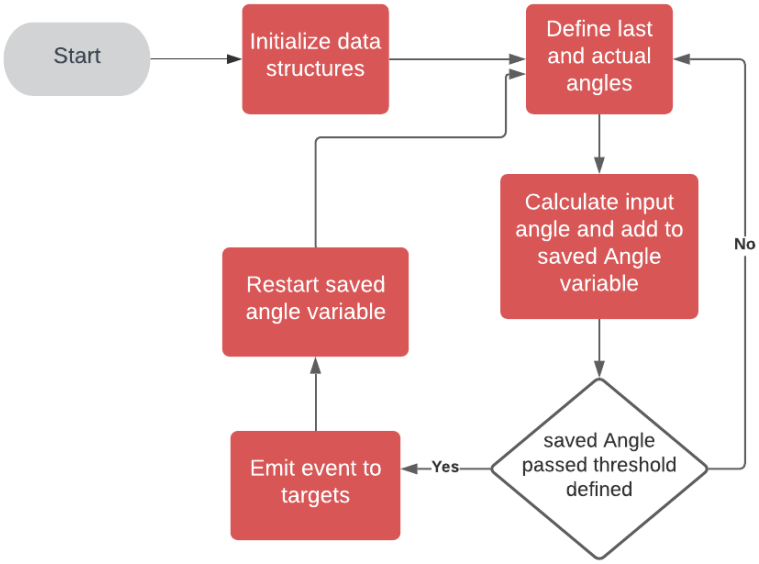

Figura 35- Funcionamento do algoritmo Angle Detector Single Marker

O utilizador pode definir como parâmetros um limite em graus e um conjunto de alvos para onde o evento deve ser emitido. Além disso, os eixos para deteção da rotação e o habitual parâmetro opcional para depuração.

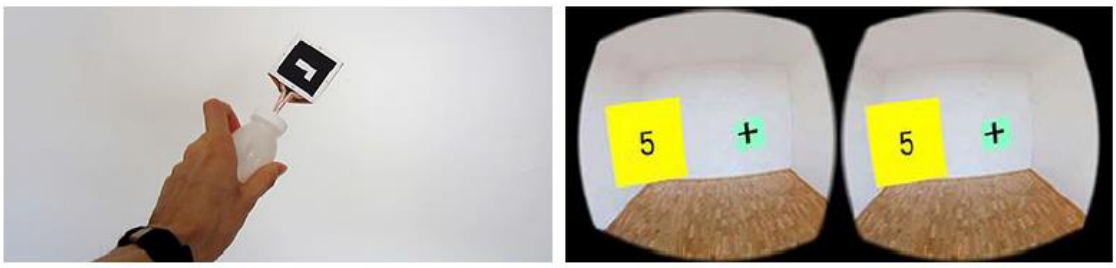

Figura 36- Exemplo de utilização do Angle Detector Single Marker

Tabela 8- Parâmetros do Angle Detector Single Marker

Propriedade	Descrição	Tipo	Valor predefinido
threshold	O limite a ser atingido para a interação ser detetada.	float	20
eventTargets	Alvos opcionais para os eventos.	selectorAll	
axis	Eixos nos quais se pretende detetar a interação.	array	['x', 'y', 'z']
debug	Parâmetro optional para depuração.	boolean	false

No código, pode ser incluído de forma simples associando a um objeto marcador da seguinte forma:

```
<a-marker mt-angle-detector-sm="threshold:45; eventTargets: #box1,#box2; axis:y;
debug: True" ></a-marker>
```

O evento contém informações sobre o eixo, direção da rotação, limite e o objeto onde ocorreu, juntamente com um registo temporal.

```
const event_rotation = new CustomEvent('event_rotation', {
    detail: {
      time: time,
      axis: axis,
      direction: direction,
      threshold: this.threshold,
      object: this.el
    },
});
```

5.6 Angle Detector Double Marker

O componente Angle Detector Double Marker trata da medição do ângulo entre dois marcadores. O funcionamento do algoritmo baseia-se na construção de um plano virtual nos lugares dos dois marcadores, seguido de da medição do angulo entre os dois através do método *angleTo()* [22] do *three.js*. Quando este angulo passa o limite definido, um evento é emitido.

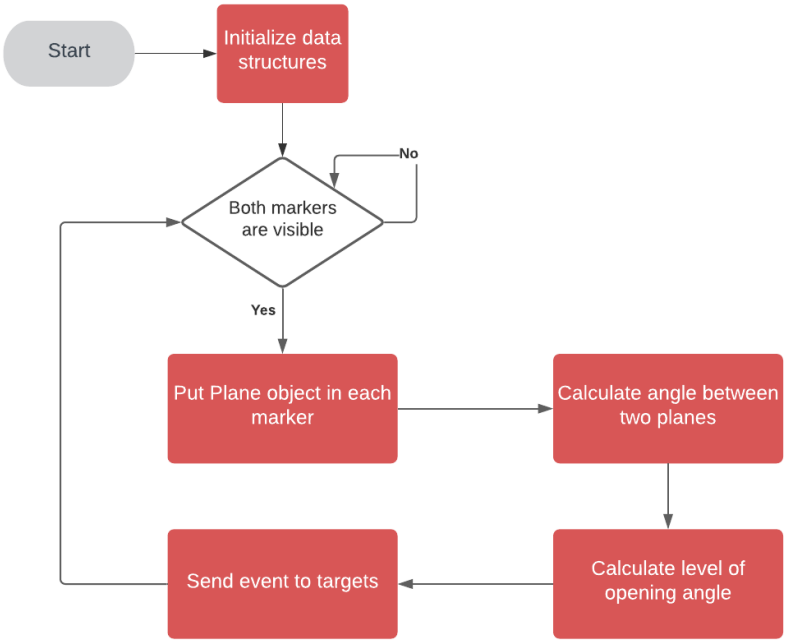

Figura 37- Funcionamento do algoritmo do Angle Detector Double Marker

É possível o programador personalizar o alcance e o tipo de ângulo que pretende detetar. Existem dois tipos de ângulos. O primeiro pretende detetar o ângulo entre dois marcadores que simulam a abertura de um livro, ou movimentos semelhantes. O segundo tipo de ângulo coloca os marcadores no mesmo plano, e o ângulo detetado é o que é feito entre um lado de cada marcador.

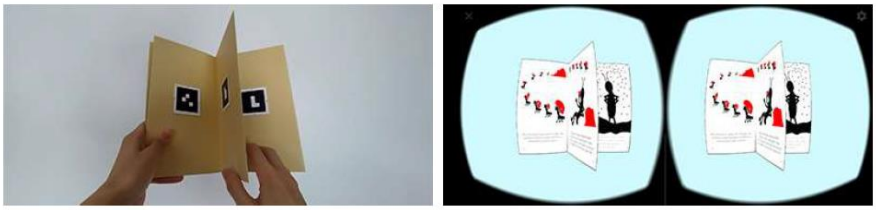

Figura 38- Exemplo de protótipo para utilização do Angle Detector Double Marker

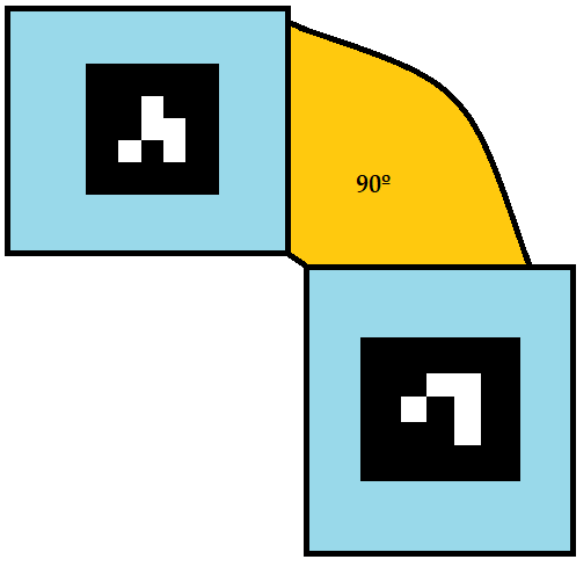

Figura 39- Segundo tipo de ângulo

Dado um limite introduzido pelo programador, serão emitidos eventos para o objeto associado ao componente e para alvos adicionais cada vez que este valor for atingido, permitindo que as entidades sejam atualizadas relativamente ao ângulo. Este componente pode ajudar em experiências, por exemplo, no movimento de abertura de um livro ou páginas de, produzindo uma animação em função do ângulo de abertura entre as duas páginas.

Tabela 9 - Parâmetros do Angle Detector Double Marker

Propriedade	Descrição	Tipo	Valor predefinido
threshold	O angulo a ser atingido para a interação ser detetada.	float	20
secondMarker	Uma referência ao segundo marcador.	selector	
eventTargets	Alvos opcionais para os eventos.	selectorAll	
axis	Eixos nos quais se pretende detetar a interação.	array	['x', 'y', 'z']
debug	Parâmetro optional para depuração.	boolean	false

No código, o componente exige a existência de dois objetos marcador distintos, sendo que um deles é dado como referência a aquele onde o componente estará associado.

```
<a-marker id="my1" type="barcode" value="3" ></a-marker>

<a-marker id="my2" type="barcode" value="4" mt-angle-detector-dm= "threshold: 10; movement: 1; secondMarker: #my1; eventTargets: #box2, #box1; debug:true;" ></a-marker>
```

O evento emitido contém a identificação dos dois marcadores envolvidos na interação, juntamente com o valor do angulo e um registo temporal.

```
const event_rotation_double = new CustomEvent ('event_rotation_dm', {
  detail: {
    time: time,
    object1: this.el,
    object2: this.secondMarker,
    angle: level
  },
});
```

5.7 Noise Controller

Este último componente não representa uma interação com um objeto tangível, mas sim uma tentativa de atenuar um dos maiores problemas da RA: o ruído associado á utilização de marcadores. Este ruido pode ser causado por diversos fatores: fraca luminosidade, fraco poder computacional do dispositivo ou pouca estabilidade na camara.

O algoritmo trabalha de modo que, quando o marcador deixar de ser reconhecido, o conteúdo virtual associado fica em cena por algum tempo na última posição onde ele esteve. Assim, quando voltar a ser reconhecido dentro do tempo estabelecido pelo programador, o seu conteúdo virtual associado volta a aparecer na posição do marcador. Por outro lado, caso o marcador não volte a ser encontrado, o conteúdo desaparece ao fim do tempo estabelecido.

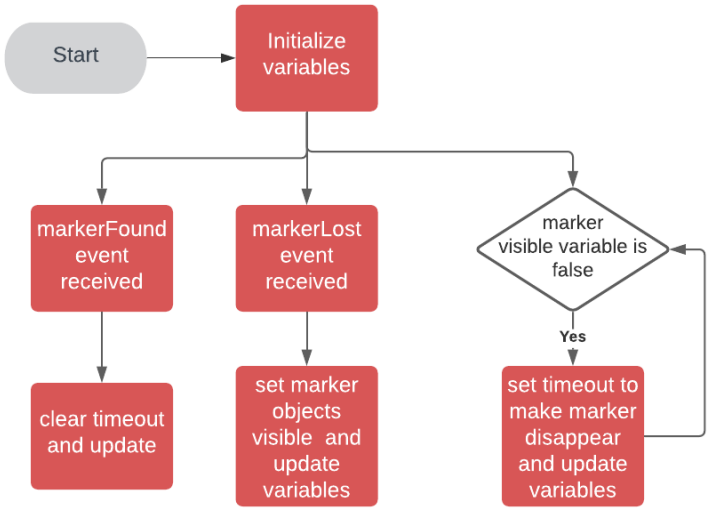

Figura 40- Funcionamento do Noise Controller

O componente pode ser associado a um objeto marcador, no entanto não é recomendável a sua utilização com outros componentes que sejam baseados em métodos de oclusão dos mesmos, pois interfere no funcionamento destes.

```
<a-marker preset="hiro" mt-noise-controller = "maximum_time: 2000; debug:
true;"></a-marker>
```

Os parâmetros pedidos ao utilizador são apenas o tempo máximo e o parâmetro opcional de depuração.

Tabela 10- Parâmetros do Noise Controller

Propriedade	Descrição	Tipo	Valor predefinido
maximumTime	Tempo máximo (em milissegundos) de exibição do conteúdo virtual depois de ele desaparecer.	int	1000
debug	Parâmetro optional para depuração.	boolean	false

Capítulo 6 Testes de Usabilidade

6.1 Procedimento

De forma a avaliar esta família de componentes foram realizados testes de usabilidade para avaliar a compreensibilidade, capacidade de abstração, aprendizagem e reutilização. Para isso, foi seguido um estudo de avaliação de usabilidade de API's [25] .

O objetivo de recrutamento de participantes passou por incluir tanto pessoas com experiência na framework (no desenvolvimento de experiências e de componentes), como pessoas sem qualquer experiência na área. Para estas, foi preparado um tutorial de A-Frame com os pontos essenciais para a programação das tarefas do teste. Este tutorial explica como programar uma experiência em A-Frame utilizando algumas das principais primitivas e propriedades dos elementos no código HMTL, tal como criar um componente de raiz.

A primeira fase do teste de usabilidade foi realizada de forma assíncrona e contou com 5 participantes. Os participantes receberam o conjunto de tarefas e a biblioteca de componentes e foram solicitados a resolvê-las ao seu próprio ritmo e a enviar as suas respostas quando terminadas.

De forma a recolher informação mais detalhada acerca das dificuldades e problemas dos programadores, foi conduzida uma segunda sessão presencial de testes com 3 participantes onde foi possível acompanhar de perto as suas dificuldades e duvidas e recolher notas mais especificas.

Recrutamento

O recrutamento foi feito por contacto direto a programadores (presencialmente e via *email*) e online por divulgação nas plataformas sociais Facebook e Reddit em grupos relacionados com RV e com as frameworks A-Frame, Three.js, WebXR.

Tarefas

Após a realização do tutorial, os participantes eram desafiados a realizar um conjunto de tarefas (tabela 8). Para os componentes incluídas nestes testes (Shake Detector, Angle Detetctor Single Marker, Button e Swipe) pedimos aos programadores com experiência que resolvam duas tarefas. No entanto, caso o participante não tivesse experiência foi aceite apenas a primeira tarefa de cada componente.

No que diz respeito às tarefas, foi fornecido aos participantes código de experiências básicas que incluíam vídeos pré-gravados com tangíveis de vários tipos de movimentos relativos. Este código era fornecido através da plataforma Glitch onde podiam programar diretamente, no entanto podiam fazê-lo no seu próprio ambiente, caso assim preferissem. Assim, os participantes podiam alterar diretamente o tipo de movimento no decorrer da experiência (figura X). Este código permitiu aos participantes concentrar na solução e não na criação dos elementos da cena.

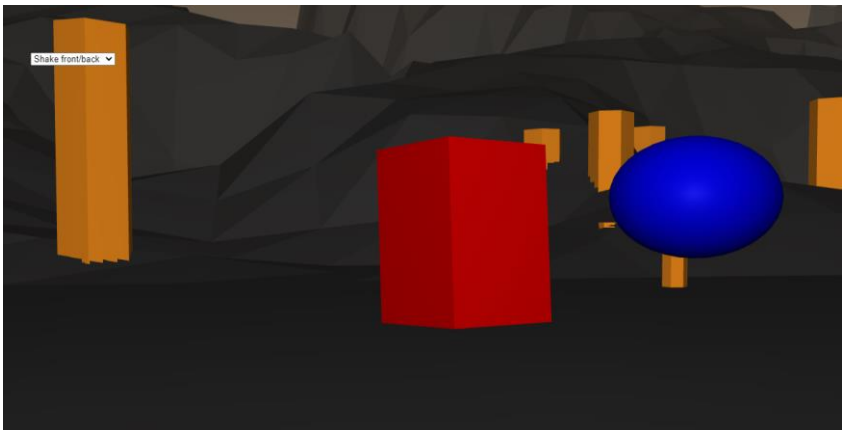

Figura 41- Exemplo de experiência virtual de uma tarefa

A primeira tarefa para cada componente poderia ser implementada apenas detetando a interação com os parâmetros certos no componente. A segunda tarefa, sendo mais complexa só pode ser resolvida através do tratamento da informação presente nos eventos que são emitidos ao detetar a interação.

Tabela 11- Tarefas do teste de usabilidade

Shake Detector– Tarefa 1	*Use the Shake Detector component to detect the marker/box shaking front/back and turn the element with the sphere into green.*
Shake Detector – Tarefa 2	*Move the sphere 1 meters to the right each time the box is shaken horizontally (left/right) and move the sphere 1 meters up each time the box is shaken vertically (up/down).*
Button – Tarefa 1	*Use Button component to detect the click and use the event to change the color of the object 'box' into purple.*
Button – Tarefa 2	*Use Button component to detect the click and use the event to change the environment to 'egypt' preset.*
Swipe – Tarefa 1	*Use the Swipe without reference marker to detect the sequence (my1, my2, my3) and change the color of the sphere to red.*
Swipe – Tarefa 2	*Use the Swipe with reference marker (marker with preset "hiro") to detect the sequence (my3, my2, my1) and change the position of te sphere in 1 unit in X axis.*
Angle Detector – Tarefa 1	*Use Angle Detector component to detect a rotation of 20º in Y axis and use it to change the color of the sphere to blue.*
Angle Detector – Tarefa 2	*Add the rotation detection on the X axis with the same angle but this time, this detection must change the color of the sphere to red.*

Questionário

Finalizando as tarefas, os participantes responderam a um questionário com perguntas demográficas como a idade, país, área de trabalho, experiência de programação, e experiência específica de programação. Além disso, foi pedida autorização para utilização dos seus dados para fins de investigação e que submetessem o código e tempo de desenvolvimento das experiências. Por último, o questionário incluía uma secção de perguntas adaptadas (tabela 12) para avaliar a nossa biblioteca em termos de compreensibilidade, capacidade de abstração, aprendizagem e reutilização.

Tabela 12- Perguntas do questionário do teste de usabilidade

Compreensibilidade	*Do you find that the API types map to the domain concepts in the way you expected?*
	Do you feel you had to keep track of information not represented by the API to solve the tasks?
	Does the code required to solve the tasks match your expectations?
Abstração	*Once you performed the first two tasks, was it easier to perform the remaining tasks?*
	Do you feel you had to learn many classes and dependencies to solve the tasks?
Abstração	*Do you find the abstraction level on the interactions programed appropriate to the tasks?*
	Did you need to adapt the API (like overriding default behaviors in Javacript code) to meet your needs?
	Do you feel you had to understand the underlying implementation of the components to be able to use the API?
Reutilização	*Does the amount of code required for each task seem about right, too much, or too little for you?*
	How easy was it to evaluate your own progress (intermediate results) while solving the tasks?
	Do you feel you had to choose one way (out of many) to solve a task in the scenario?
	Do you feel you would have to change much in your code to change the interaction behaviour?

6.2 Resultados

Dados demográficos

Da totalidade dos participantes, todos são fluentes na lingua inglesa sendo que metade tem entre 18-25 anos de idade e os restantes 25-35. Um dos participantes é residente no Reino Unido e os restantes em Portugal. Metade programa entre 1-5 anos e os restantes entre 1-10 anos. No que diz respeito á experiência com A-Frame, 2 participantes já programaram componentes (um deles entre 1-2 e o ultimo mais de 3 componentes) e 4 já desenvolveram experiências virtuais com A-Frame. Os restantes 4 não tem qualquer experiência com a framework e nenhum dos participantes tinha experiêcia prévia em AR.js. Quanto á participação no tutorial, apenas 2 participantes não o fizeram.

Tabela 13- Resumo dos dados dos participantes

Idade	**Ocupação**	**Anos de programação**	**Fez tutorial**	**Experiência nas framework**	**Quantos componentes programou**
18-24	PhD	5-10	Não	Experiências e componentes A-Frame	1-2
24-35	Full stack developer	5-10	Não	Experiências A-Frame	0
24-35	Student of computer science	1-5	Sim	Nenhuma	0
24-35	Computer Scientist	5-10	Sim	Experiências A-Frame	0
24-35	Cybersecurity engineer	5-10	Sim	Experiências e componentes A-Frame	Mais de 3
18-24	Electrotechnical eng. student	1-5	Sim	Nenhuma	0
18-24	Mechanical eng. student	1-5	Sim	Nenhuma	0
18-24	Student	1-5	Sim	Nenhuma	0

Tempos de realização

O tempo de relização das tarefas varia muito devido á experiência do programador na framework, linguagem e até mesmo ambiente de desenvolvimento. Muitas vezes este tempo era maioritariamente gasto a tentar resolver um problema proveninente dessa mesma falta de experiencia e adaptação ao ambiente de desenvolvimento. A primeira tarefa, sendo o primeiro contacto, mostra-se mais demorada. Além disso, esta tarefa inclui um *outlier* com o valor de 3 horas, que após um esclarecimento com o participante, verificou-se que se tratou de um problema com a plataforma. Além desta, a segunda tarefa do componente Button demorou em média mais tempo, pois utiliza uma biblioteca não conhecida e implica alguma pesquisa. As restantes tarefas os tempos de realização mostram-se estáveis.

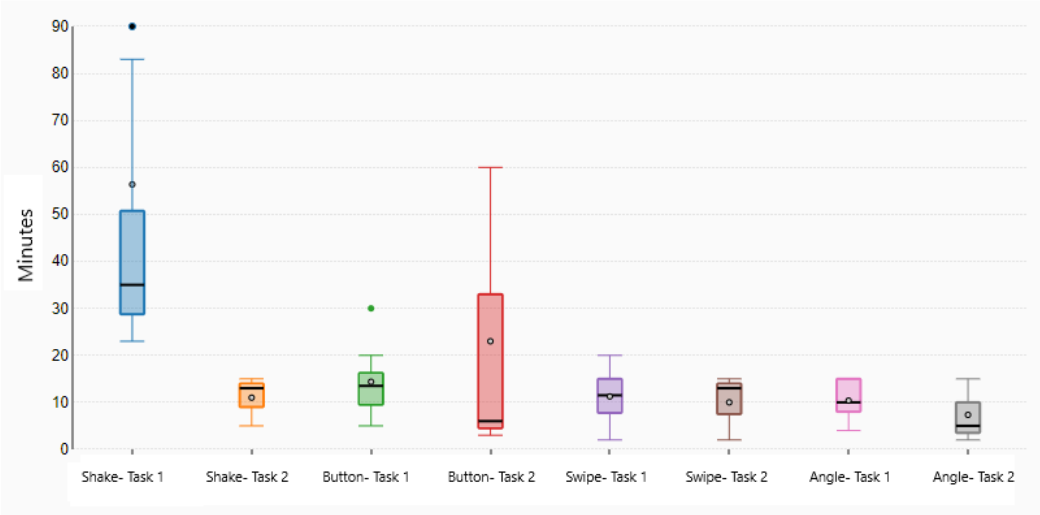

Figura 42- Tempos de realização das tarefas

Compreensibilidade

Os resultados no campo da compreensibilidade são positivos, existindo no entanto alguma discrepancia nos valores da pergunta 2. Os testados sentiram que o conceito dos componentes vão de encontro ao objetivo proposto e o código necessário para incluir as interações corresponde ás suas espectativas. No entanto, 4 utilizadores responderam de forma neutra e 1 de forma negativa quando o assunto era tratar da informação disponibilizada pelos componentes. Estas respostas neutras podem significar que a pergunta é ambigua, no entanto a resposta negativa diz-nos que existe algo a melhorar. Neste caso, uma solução para melhorar a biblioteca neste sentido seria aumentar a informação na consola quando o campo *debug* está ativo, aumentando comsequentemente a quantidade de informação que é passada para o utilizador.

Understandability	1. Do you find that the API types map to the domain concepts in the way you expected?	0	0	1	3	4
	2. Do you feel you had to keep track of information not represented by the API to solve the tasks?	1	0	4	1	2
	3. Does the code required to solve the tasks match your expectations?	0	0	1	5	2

Figura 43- Respostas no campo da compreensibilidade

Aprendizagem

No campo da facilidade de aprendizagem os resultados confirmam o que era apresentado pelos dados dos tempos relativos a cada tarefa. A utilização dos componentes torna-se cada vez mais fácil com a experiência. Os utilizadores sentiram que não tiveram de aprender muito para conseguir resolver as tarefas indicadas e utilizar os componentes. Como tal, todas as respostas foram positivas.

Learnability	4. Once you performed the first two tasks, was it easier to perform the remaining tasks?	0	0	0	1	7
	5. Do you feel you had to learn many classes and dependencies to solve the tasks?	0	0	0	4	4

Figura 44- Respostas no campo da aprendizagem

Abstração

Neste campo as respostas são também de uma forma geral positivas. Sete participantes acham o nivel de abstração indicado para as interações e 1 respondeu de forma neutra. Um dos participantes precisou de reescrever o código base para ficar satisfeito na realização das tarefas. Também um testado achou que sentiu necessidade de entender de que forma a implementação era feita para conseguir usar.

Abstraction	6. Do you find the abstraction level on the interactions programed appropriate to the tasks?	0	0	1	3	4
	7. Did you need to adapt the API (like overriding default behaviors in Javacript code) to meet your needs?	0	1	0	1	6
	8. Do you feel you had to understand the underlying implementation of the components to be able to use the API?	0	1	1	3	3

Figura 45- - Respostas no campo da capacidade de abstração

Reutilização

Neste campo os resultados também se mostram positivos, visto que não há respostas negativas. As respostas recolhidas mostram que o código necessário era minimo e que o progresso na programação era facilmente avaliado sem ter que alterar muito código. Também foi da geral opinião dos testados que a implementação não era ambiguas, tendo entendido como as fazer.

Reusability	9. Does the amount of code required for each task seem about right, too much, or too little for you?	0	0	0	4	4
	10. How easy was it to evaluate your own progress (intermediate results) while solving the tasks?	0	0	3	4	1
	11. Do you feel you had to choose one way (out of many) to solve a task in the scenario?	0	0	2	2	4
	12. Do you feel you would have to change much in your code to change the interaction behaviour?	0	0	1	3	4

Figura 46- Respostas no campo da capacidade de reutilização

Testes presenciais

Na sessão de testes presenciais que contou com 3 participantes, foram recolhidas algumas notas acerca das principais dificuldades dos programadores. As resolução das tarefas teve 3 partes: incluir o script do componente, utilizar um componente no objeto correto, produzir um comportamento através da captura do evento. Sobre estas partes, foram recolhidos diretamente acerca das dificuldades encontradas (tabela 13).

Tabela 14- Dificuldades dos participantes

	Participante 6	**Participante 7**	**Participante 8**
Incluir o script do componente	-Alguma dificuldade (esquecimento)	-Sem dificuldade	-Sem dificuldade
Utilizar o componente no objeto	-Algumas duvidas relativamente ao objeto no qual os componentes deviam ser associados no Swipe	-Sem dificuldade	-Alguma confusão relativamente aos eixos de cada movimento no Shake Detector e Angle Detector - Algumas duvidas relativamente ao objeto no qual os componentes deviam ser associados no Swipe
Produzir um comportamento através da captura do evento	-Dificuldade em captar o evento certo no Shake Detector (primeira tarefa)	-Sem dificuldade	-Sem dificuldade

As dificuldades encontradas nesta sessão foram principalmente duvidas relativamente aos objetos onde associar o componente necessário. Apenas um dos participantes teve dificuldade na captação do evento.

Comentários

Na secção do questionário dedicada a comentários foram recolhidos algumas opiniões dos testados relativamente ao funcionamento do teste e dos componentes. Quando questionados acerca de possiveis mudanças no componente Angle Detector o participante 1 afirmou que teve dificuldades em perceber qual o objeto onde deveria associar o componente. Para deixar esrte aspeto mais claro foram feitas alterações na docuemntação de todos os componentes de forma a ajudar na compreensão da utilização dos mesmos, apresentando vários exemplos na prórpia documentação e exemplos completos no repositório. O participante 4 afirmou que o componente deveria poder detetar rotações em vários eixos simultaneamente e o participante 7 queixou-se de problemas na deteção dos marcadores. Quanto ao problema do participante 4, entende-se que foi causado por alguma falta de atenção, tendo em conta que o componente já permite este comportamento desde a versão inicial. Quando ao problema da deteção de marcadores, não existe informação suficiente para perceber de onde vem, visto que o a deteção é feita com video que passa repetidamente na experiência virtual.

No que diz respeito ao Button Component o participante 2 propôs uma mudança na implementação do componente. Na sua opinião deveria existir um modo no qual vários eventos eram emitidos de forma continua, desde que o marcador estivesse tapado e não

apenas quando este deixa de ser tapado. Esta alteração não avançou pois, após alguns testes percebeu-se que assim que o marcador associado ao botão era tapado (mesmo que acidentalmente por questões de ruído) eventos eram emitidos, produzindo comportamentos inesperados.

Concluindo, os testes de usabilidade permitiram concluir que a biblioteca cumpre o objetivo principal do projeto: permite a programadores com ou sem experiência na framework, importar interações para interfaces tangíveis em experiências de RV. As observações na sessão persencial e os comentários das sessões assíncronas permitiram perceber que dificuldades existiam na programação com os componentes. Consequentemente, foram alterado o mecanismo de depuração de todos os componentes tal como a sua documentação, de forma a facilitar a compreensão dos utilizadores.

Capítulo 7 Aplicações de Demonstração

De forma a demonstrar o potencial desta família de componentes e de que forma estes podem trabalhar em conjunto para satisfazer diferentes interações foram desenvolvidas duas experiências de RV que utilizaram alguns dos componentes criados.

7.1 Album Viewer Experience

A primeira experiência a ser desenvolvida consiste num leitor de música, feito apenas com 6 marcadores distintos. Este leitor contém um elemento marcador que no qual é incluída uma imagem e serve de referência. Dos restantes marcadores, dois funcionam como botão de reprodução e pausa, através do componente Button e três fazem parte do conjunto de marcadores para a interação Swipe, que permite avançar ou retroceder na música. O código HMTL usado para a experiência é o seguinte:

```
<a-marker id="reference" preset='hiro' mt-swipe= "markers:#swipe1,#swipe2,#swipe3;
sequences:1 2 3,3 2 1; maximumTime:3000; hasReference: true;debug: true;"></a-
marker>

<a-marker id="swipe1" type="barcode" value="3"></a-marker>

<a-marker id="swipe2" type="barcode" value="4"></a-marker>

<a-marker id="swipe3" type="barcode" value="5"></a-marker>

<a-marker id="play" type="barcode" value="1" mt-button="referenceMarker:
#reference; minimumTime: 1000; debug: true;"></a-marker>

<a-marker id="pause" type="barcode" value="2" mt-button="referenceMarker:
#reference; minimumTime: 1000; debug: true;"></a-marker>
```

De forma a captar os eventos e tratar do comportamento a realizar, foi desenvolvido o seguinte código na linguagem Javacript:

```
let songs = [new Audio('pf-soycd.mp3'), new Audio('pf-cn.mp3'), new Audio('pf-
abitw.mp3')];
let index = 0;
var play = document.getElementById("play");
var pause = document.getElementById("pause");

songs[index].play();
document.getElementById("reference").addEventListener('event_swipe', e=>{
    songs[index].pause();
    songs[index].currentTime = 0;
    if(e.detail.sequenceIndex == 1){
        if(index + 1 < songs.length ){
            index += 1; }
        else{
            index = 0;
        }
    }
    else if(e.detail.sequenceIndex == 2){
        if(index - 1 >= 0  ){
            index -= 1;
        }
        else{
            index = 0;
        }
    }
    songs[index].play();
});

pause.addEventListener('event_button_pressed', e=> { songs[index].pause();} );
play.addEventListener('event_button_pressed', e=> { songs[index].play();} );
```

Figura 47- Experiência Album Viewer

7.2 Shake Book Experience

Esta experiência virtual baseia-se na utilização de um livro tangível, com um marcador distinto em cada uma das páginas e capa. Ao abrir uma página do livro, o ambiente virtual é alterado para um novo ambiente, consoante a página com o auxílio do componente Angle Detector Double Marker. Este angulo entre as duas páginas também altera a luminosidade da cena, aumentando quanto maior for o mesmo. Este comportamento é conseguido através da captação do evento relativo á abertura de 45 graus entre os dois marcadores.

Figura 48- Exemplo da experiência Shake Book (página 1)

Figura 49- Exemplo da experiência Shake Book (página 2)

Além disso, o utilizador pode alterar o conteúdo para um ambiente aleatório, abanando um objeto tangível representativo de um sino de igreja. Este sino, utiliza também o componente Noise Controller.

Figura 50- Exemplo da experiência Shake Book (sino)

O código HMTL usado para a experiência é o seguinte:

```
<a-marker type="pattern" url= "letterA.patt" mt-noise-controller>
    <a-entity obj-model="obj: #torreOBJ; mtl: #torreMTL"></a-entity>
</a-marker>

<a-marker preset = "hiro"  mt-shake-detector = "switchInterval: 1000 ;
minimumSwitchTimes: 3 ; minimumDistance: 0.2 ; eventTargets:#env; axis:z ; debug:
true;">
    <a-entity obj-model="obj: #sinoOBJ; mtl: #sinoMTL" ></a-entity>
</a-marker>

<a-marker type="barcode" value="0" >
    <a-image src="pink_floyd.png" rotation="-90 0 0"></a-image>
</a-marker>

<a-marker id="page1" type="barcode" value="1" mt-angle-detector-dm= "threshold: 10
; movement : 1; secondMarker: #my2;  debug:false;" >
    <a-image src="jungle.jpg" rotation="-90 0 0"></a-image>
</a-marker>

<a-marker id="page2" type="barcode" value="2"  >
    <a-image src="jungle2.jpg" height="1.5" width="1.5" position = "0 0 -2"
rotation="-90 0 0"></a-image>
</a-marker>

<a-marker id="my3" type="barcode" value="3"  mt-angle-detector-dm= "threshold: 10;
movement : 1; secondMarker: #my4; debug:false;" >
    <a-image src="sea.jpg" rotation="-90 0 0"></a-image>
</a-marker>

<a-marker id="my4" type="barcode" value="4" >
    <a-image src="sea2.jpg" rotation="-90 0 0"></a-image>
</a-marker>

<a-marker id="my5" type="barcode" value="5"  mt-angle-detector-dm= "threshold: 10;
movement : 1; secondMarker: #my6;  debug:false;" >
    <a-image src="japan.jpg" rotation="-90 0 0"></a-image>
</a-marker>

<a-marker id="my6" type="barcode" value="6"  >
```

```
    <a-image src="japan2.jpg" rotation="-90 0 0"></a-image>
</a-marker>
```

De forma a captar os eventos e tratar do comportamento a realizar, foi desenvolvido o seguinte código na linguagem Javacript:

```
let env = document.getElementById('env')
env.addEventListener('event_shake', e=>{
	let env = document.getElementById('env')
	let envs = ['contact', 'egypt', 'checkerboard', 'forest', 'goaland',
	'yavapai', 'goldmine', 'threetowers', 'poison', 'arches', 'tron', 'japan',
	'dream', 'volcano', 'starry', 'osiris']
	let random_element = envs[Math.floor(Math.random() * envs.length)];
	env.setAttribute('environment', 'preset: ' + random_element );
});

document.getElementById('my1').addEventListener('event_rotation_dm', e=> {
	if(e.detail.angle >= 90){
		let env = document.getElementById('env')
		if(env.getAttribute('environment').preset != "forest"){
			env.setAttribute('environment', 'preset: forest');
		}
	}
	let light = document.getElementById('ambientLight');
	light.setAttribute('intensity', e.detail.angle / 180);

});
document.getElementById('my3').addEventListener('event_rotation_dm', e=> {
	let env = document.getElementById('env')
	if(e.detail.angle >= 90){
		let env = document.getElementById('env')
		if(env.getAttribute('environment').preset != "dream"){
			env.setAttribute('environment', 'preset: dream');
		}
	}
	let light = document.getElementById('ambientLight');
	light.setAttribute('intensity', e.detail.angle / 180);

});

document.getElementById('my5').addEventListener('event_rotation_dm', e=> {
	let env = document.getElementById('env')
	if(e.detail.angle >= 90){
		let env = document.getElementById('env')
		if(env.getAttribute('environment').preset != "japan"){
			env.setAttribute('environment', 'preset: japan');
		}
	}
	let light = document.getElementById('ambientLight');
	light.setAttribute('intensity', e.detail.angle / 90);

});
```

7.3 Conclusão

Como está demonstrado neste capítulo, é possivel utilizar os componentes criados em conjunto de forma a criar experiências virtuais uteis e simples em termos de complexidade de programação.

Capítulo 8 Conclusão

Com a realização deste projeto permitiu perceber de que forma a RV e RA podem ser utilizadas como forma de interação e visualização de conteúdo em ambiente virtual. É possível para qualquer pessoa entrar num ambiente virtual que pode controlar de várias formas e isso traz potenciais vantagens em diferentes áreas como a medicina, entretenimento e mesmo na indústria. O fator da interação traz ainda mais vantagens, permitindo a alteração dos conteúdos virtuais em tempo real e de forma imediata. Esta interação pode ser feita de várias formas, recorrendo a movimentos físicos do utilizador ou manipulação de controladores e interfaces tangíveis presentes no mundo real. Além de permitirem a interação, as interfaces tangíveis trazem dinamismo às experiências aumentando o realismo das mesmas sendo algo que está presente tanto no mundo real como no virtual. Através de marcadores visuais, tipicamente utilizados em RA, a sua associação a objetos físicos permitiu transformar estes objetos, virtualizando e adicionando utilidade aos mesmos. Estes objetos, visto que podem ser geometricamente idênticos aos elementos do espaço virtual, tornam possível uma experiência háptica para o utilizador, criando uma forma de interação que permite aos utilizadores manipularem objetos reais com consequências de interação no mundo virtual. Esta utilização implica a utilização da tecnologia de processamento de imagem, utilizando a camara de um dispositivo que oferece uma alternativa de baixo custo aos controladores genéricos.

No entanto a programação deste tipo de experiências ainda está numa fase inicial, não existindo ferramentas que a facilitem e que permitam aos programadores a abstração que existe noutro tipo de áreas. Esta biblioteca de componentes vem então complementar o material disponível para os programadores desenvolverem experiências que recorrem a métodos de interação com tangíveis de forma mais fluida e rápida. Os resultados do processo de desenvolvimento desta biblioteca resultaram num conjunto de componentes que podem funcionar em conjunto para interagir de forma simples com o ambiente virtual onde estão inseridos. O processo de desenvolvimento feito ao longo da dissertação foi essencial para a programação dos componentes, visto que os projetos analisados no estado da arte permitiram perceber de que forma os objetos tangíveis estão a ser usados e os testes de usabilidade permitiram conhecer melhor de que forma os componentes deveriam ser apresentados ao utilizador para uma melhor eficiência e utilidade. Este feedback foi muito importante para o desenvolvimento de um produto que será usado por programadores. Com a aprendizagem adquirida nesta dissertação, posso concluir que existe uma necessidade de conteúdo para abstrair os programadores de RM da interação com interfaces tangíveis e espero que esta libraria venha ajudar a colmatar essa falta.

Referências

[1] J. Ribeiro, "Interação em Realidade Virtual Através De Tangíveis Passivos," Universidade de Coimbra - Faculdade de Ciências e Tecnologia, 2020.

[2] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, "Augmented reality: A class of displays on the reality-virtuality continuum," in *Telemanipulator and Telepresence Technologies*, Dec. 1995, vol. 2351, pp. 282–292, doi: 10.1117/12.197321.

[3] M. Heilig, "The Cinema of the future," 1955.

[4] I. E. Sutherland, "The Ultimate Display," *Proc. IFIP Congr.*, vol. 2, pp. 506--508, 1965, Accessed: Jan. 07, 2021. [Online]. Available: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.136.3720.

[5] T. G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill, "HAND GESTURE INTERFACE DEVICE.," in *Proceedings - Graphics Interface*, 1987, pp. 189–192, doi: 10.1145/30851.275628.

[6] L. Rosson, "The Virtual Interface Environment Workstation (VIEW), 1990," 2014, Accessed: Jan. 07, 2021. [Online]. Available: http://www.nasa.gov/ames/spinoff/new_continent_of_ideas.

[7] "A-Frame." https://aframe.io/.

[8] J. C. S. Cardoso and J. M. Ribeiro, "VR Book : A Tangible Interface for Smartphone-based Virtual Reality," *Proc. 17th EAI Int. Conf. Mob. Ubiquitous Syst. Comput. Netw. Serv. (Mobiquitous 2020)*, 2020.

[9] S. J. Henderson and S. Feiner, "Opportunistic Controls: Leveraging natural affordances as tangible user interfaces for augmented reality," in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST*, 2008, pp. 211–218, doi: 10.1145/1450579.1450625.

[10] H. Slay, B. H. Thomas, and R. Vernik, "Tangible User Interaction Using Augmented Reality BT - Third Australasian User Interface Conference (AUIC2002)," vol. 7, no. January 2014, pp. 13–20, 2002, doi: 10.1145/563997.563988.

[11] H. T. Regenbrecht, G. Baratoff, I. Poupyrev, and M. Billinghurst, "A Cable-less Interaction Device for AR and VR Environments."

[12] D. Harley, A. P. Tarun, D. Germinario, and A. Mazalek, "Tangible VR: Diegetic tangible objects for virtual reality narratives," in *DIS 2017 - Proceedings of the 2017 ACM Conference on Designing Interactive Systems*, Jun. 2017, pp. 1253–1263, doi: 10.1145/3064663.3064680.

[13] H. Lee, M. Billinghurst, and W. Woo, "Two-handed tangible interaction techniques for composing augmented blocks," *Virtual Real.*, vol. 15, no. 2–3, pp. 133–146, Jun. 2011, doi: 10.1007/s10055-010-0163-9.

[14] J. M. S. Dias, N. Barata, P. Santos, A. Correia, P. Nande, and R. Bastos, "In your hand computing: Tangible interfaces for mixed reality," in *ART 2003 - IEEE International Augmented Reality Toolkit Workshop*, 2003, pp. 29–31, doi: 10.1109/ART.2003.1320422.

[15] G. A. Lee, M. Billinghurst, and G. J. Kim, "Occlusion based interaction methods for tangible augmented reality environments," in *Proceedings VRCAI 2004 - ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications*

in Industry, 2004, pp. 419–426, doi: 10.1145/1044588.1044680.
[16] "XR Interaction Toolkit." https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@1.0/manual/index.html.
[17] "VRTK." https://vrtoolkit.readme.io/.
[18] "ARCore." https://developers.google.com/ar.
[19] "ARKit." https://developer.apple.com/augmented-reality/arkit/.
[20] "Zapbox." https://www.zappar.com/zapbox/.
[21] "VR Tangible Marker Interaction Library," [Online]. Available: https://github.com/JoaoDiogoMesquita/VR-Tangible-Interaction-Toolkit.
[22] "Three.js."
[23] "WebXR Device API," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API.
[24] "AR.js." https://github.com/AR-js-org.
[25] M. Piccioni, C. A. Furia, and B. Meyer, "An empirical study of API usability," *Int. Symp. Empir. Softw. Eng. Meas.*, pp. 5–14, 2013, doi: 10.1109/ESEM.2013.14.

Apêndices

Apêndice A

Guide for the usability Test

Introduction

The main goal of this test is to evaluate the usability of four A-Frame components that are part of a programmatic toolkit that aims to assist the development of marker-based tangible interaction in virtual reality. The goal is to make improvements to these components to make them easier to use by programmers.
The A-Frame components that are being evaluated are documented at https://github.com/JoaoDiogoMesquita/VR-Tangible-Interaction-Toolkit.
The test consists of a number of programming tasks that you should solve using the A-Frame components that are mentioned in each task. For each task, an HTML file has been prepared with the basic A-Frame scene. In addition, the scene is already configured with a simulation of the interaction: it has been set up to play videos of a user interacting with a marker-based object, so you just have to program the experience to detect the interactions being made in the video.
While you are solving the tasks, we ask you to record three things:
the time spent on each task (rougly, you can just write the time of start and finish in the code in order to keep track)
how many different attempts at each task (how many different approaches you tried before getting it right)
any difficulties encountered (even if you solved the task, report anything that caused you to revise your code - misspelled a parameter name, misunderstood the meaning of a parameter, forgot including <script>, etc.)
We ask that you leave comments in the code indicating the three points mentioned above (time, attempts and difficulties), or anything else you feel could be useful to improve the A-Frame components.
Each task should be solved in its own HTML file, as provided.
If you don't have an IDE or simply prefer to use our programing environment, there are projects on Glitch that you just have to remix to edit and run code on yout browser:

- Task Set 1
- Task Set 2
- Task Set 3
- Task Set 4

If you have no experience with the A-Frame framework, I would first recommend that you go through the tutorial that has been prepared with the basics.
If you don't have experience in writing custom A-Frame components, you may wish/it is expected for you to solve only the first task in each task set below.
When you finish implementing the following tasks, please fill in the questionnaire available at this link .

Task Set 1 (Shake Detector Component (mt-shake_detector))

Task 1:

Use the Shake Detector component to detect the marker/box shaking front/back and turn the element with the sphere into green.

Task 2:

Detect the marker/box shaking and:
move the sphere 1 meters to the right each time the box is shaken horizontally (left/right)
move the sphere 1 meters up each time the box is shaken vertically (up/down).

Code: Task Set 1

Task Set 2 (Button Component (mt-button))

Task 1:

Use Button component to detect the click and use the event to change the color of the object 'box' into purple.

Task 2:

Use Button component to detect the click and use the event to change the environment to 'egypt' preset.

Code: Task Set 2

Task Set 3 (Swipe Detector Component (mt-swipe))

Task 1:

Use the Swipe without reference marker to detect the sequence (my1, my2, my3) and change the color of the sphere to red.

Task 2:

Use the Swipe with reference marker (marker with preset "hiro") to detect the sequence (my3, my2, my1) and change the position of te sphere in 1 unit in X axis.

Code: Task Set 3

Task Set 4 (Angle Detector Component (mt-angle_detector))

Task 1:

Use Angle Detector component to detect a rotation of 20º in Y axis and use it to change the color of the sphere to blue.

Task 2:

Add the rotation detection on the X axis with the same angle but this time, this detection must change the color of the sphere to red.

Code: Task Set 4

Final task

Fill in the questionnaire available at this link.

Apêndice B

A-Frame Tutorial and Component and Usability Testing

Introduction

A-Frame is an open-source web framework for building virtual reality (VR) experiences. A-Frame is an entity component system framework for Three.js where developers can create 3D and WebVR scenes using HTML. HTML provides a familiar authoring tool for web developers and designers while incorporating a popular game development pattern used by engines such as Unity.

Getting Started

A-Frame can be developed from a plain HTML file without having to install anything. To get started, create an `.html` file and include the A-Frame script in the `<head>` tag:

```
<html>
  <head>
    <script src="https://aframe.io/releases/1.1.0/aframe.min.js"></script>
  </head>

<body>
    <a-scene>

        <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9"></a-box>

        <a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E"></a-sphere>

        <a-cylinder
                position="1 0.75 -3"
                radius="0.5"
                height="1.5"
                color="#FFC65D">
        </a-cylinder>

        <a-plane
                position="0 0 -4"
                rotation="-90 0 0"
                width="4"
                height="4"
                color="#7BC8A4">

        </a-plane>
        <a-sky color="#ECECEC"></a-sky>
    </a-scene>
  </body>
</html>
```

Tutorial

All the a-frame action happens within the `a-scene` entity. In the example above, we have the scene tag with a handful of a-frame entities. These entities represented as HTML elements are called ***primitives***.

Primitives
Entities or primitives can be described as a 3D object which can take different geometric shapes, such as a box, sphere, or a cylinder. To see every primitive that A-Frame provides, refer to the A-Frame documentation, at the bottom of the documentation navigation sidebar.

For this tutorial, we will be focusing on the most basic primitives, detailed in the table below.

Primitive	Description
`<a-box>`	The box primitive creates shapes such as boxes, cube, or walls
`<a-cylinder>`	The cylinder primitive is used to create tubes and curved surfaces
`<a-sphere>`	The sphere primitive creates a spherical or polyhedron shapes
`<a-plane>`	The plane primitive is used to create flat surfaces
`<a-sky>`	The sky primitive adds a background color or 360° image to a scene

Components
If we use the primitives as presented in the table above, the created primitives appear in their default form. To change its attributes, we associate ***components*** with these entities such as **position**, **rotation**, **color**, and **scale**. In A-Frame, components have the same syntax as an HTML attribute:

```
<a-box position="0 0 -4" color="#FF0000"
```

The code above creates a red box in the position given by the (0, 0, -4) coordinates. A-Frame uses a right-handed coordinate system where the negative Z axis extends into the screen.

Position
The position component places entities at certain spots in 3D space. Position takes a coordinate value as three space-delimited numbers.

```
<a-sphere position="0 1 -1"></a-sphere>
```

A-Frame uses a right-handed coordinate system where the negative Z axis extends into the screen.

Rotation
The rotation component defines the orientation of an entity in degrees. It takes x, y, and z, as three space-delimited numbers indicating degrees of rotation.

```
<a-box rotation="45 90 180"></a-box>
```

Scale

The scale component defines a shrinking, stretching, or skewing transformation of an entity. It takes three scaling factors for the X, Y, and Z axes.

```
<a-plane scale="0.5 1 2"></a-plane>
```

Size

The unit of measurement used in A-Frame is the meter. To define the size in A-Frame we have different attributes depending on the object's geometry. The box primitive takes the **width**, **height**, and **depth** attributes to control the size whereas the sphere and circle take the **radius** attribute. A cylinder uses both the **height** and the **radius** attribute:

```
<a-box position="2 0 -5" width="1" height="2" depth="1" color="#FFAA00"></a-box>

<a-sphere position="-2 0 -5" radius="0.75" color="red"></a-sphere>

<a-cylinder
  position="0 0 -5"
  radius="1"
  height="1.5"
  color="#212121"
></a-cylinder>
```

Animations

The animation component lets us animate and tween values including:

- Component values such as position, visibility, scale, and rotation.
- Component property values such as light, intensity, etc.

As example, below we have a translating box:

```
<a-box
  position="-1 1.6 -5"
  animation="property: position; to: 1 8 -10; dur: 2000; easing: linear; loop:
true"
  color="tomato"
></a-box>
```

See more about the animation component in the A-Frame documentation.

Entities

So far we have mentioned entities numerous times during this brief introduction to A-Frame, so what is the definition of an entity? In A-Frame entities are placeholders objects to which we plug in components to provide them appearance, behavior, and functionality.
We can attach components to it to make it render something or do something. To give it shape and appearance, we can attach the geometry and material components. The geometry component provides a basic shape fo an entity. The `primitive` property defines the general shape. The material component gives appearance to an entity. We can define properties such as color, opacity, or texture. Entities are inherently attached with the **position**, **rotation**, and **scale** components. Consider the following example:

```
<a-entity
  geometry="primitive: box"
  material="color: red"
  position="2 2 -10"
  scale="2 2 1"
>
</a-entity>
```

Refer to the A-Frame documentation to see further more information.

Creating and register a component

Components of A-Frame's are JavaScript modules that can be mixed, matched, and composed onto entities to build appearance, behavior, and functionality. We can register a new component in JavaScript and use it declaratively from the DOM. Components are configurable, reusable, and shareable. Most code in an A-Frame application should live within components [1].
To use a component, we first must define it before the `<a-scene>` tag, as example:

```
<html>
  <head>
    <!-- Import external component -->
    <script src="foo-component.js"></script>
  </head>
  <body>
    <script>
      // Or inline before the <a-scene>.
      AFRAME.registerComponent("bar", {
        // ...
      });
    </script>

    <a-scene> </a-scene>
  </body>
</html>
```

Let's have a first look to a basic component to get the general idea. This component will log a simple message once when the component's entity is attached using the `.init()` handler. But first, we need to **register** the component. Components are registered with `AFRAME.registerComponent()`. The first argument is the name of the component, which will be used as the HTML attribute name, and for this example it

will be `hello-world`. The second is a JavaScript object of methods and properties. In the next example, we have our `init()` handler.

```
AFRAME.registerComponent('hello-world', {
  init: function () {
    console.log('Hello, World!');
  }
});
```

Then we can use our `hello-world` component declaratively as an HTML attribute. Do not forget to include the JavaScript file or declare inline before the `<a-scene>` tag.

```
<html>
  <head>
    <script src="https://aframe.io/releases/1.1.0/aframe.min.js"></script>
    <!-- Import the hello-world component -->
    <script src="hello-world.js"></script>
  </head>
  <body>
    <!-- Attach the component to the scene -->
    <a-scene hello-world> </a-scene>
  </body>
</html>
```

To set a component, rather than via static HTML, is to set it programmatically with `.setAttribute()`. In the example bellow, we set the `hello-world`component on the scene programmatically:

```
document.querySelector('a-scene').setAttribute('hello-world', '');
```

Some components have methods available programmatically, and it is not possible to access them directly through HTML. For example, the Vibrotactile component has three internal methods only accessible through JavaScript. A component's methods can be accessed through the entity from the `.components` object. Consider this example:

```
AFRAME.registerComponent('foo', {
  init: function () {
    this.bar = 'baz';
  },

  qux: function () {
    // ...
  }
});
```

To access the `qux`method:

```
var fooComponent = document.querySelector('[foo]').components.foo;
fooComponent.qux();
```

We can query for elements containing a component with the attribute selector (i.e., [COMPONENT_NAME]), as used in the example above.

Refer to the A-Frame documentation for more information on creating and register a component.

```
<html>
  <head>
    <script src="https://aframe.io/releases/1.1.0/aframe.min.js"></script>
    <!-- Import the Shake Book Experience Detector component -->
    <script src="shake_detector.js"></script>
  </head>
  <body>
    <a-scene>
      <a-box shake-detector = "switchInterval: 500 ; minimumSwitchTimes: 3 ;
minimumDistance: 0.3 ; eventTargets:#myBox; axis:y ; debug: true;""></a-box>
    </a-scene>
  </body>
</html>
```