Francisco Ferreira Santiago

OS PÉS NÃO SERVEM SÓ PARA ANDAR

INTERACÇÃO COM OS PÉS EM REALIDADE VIRTUAL 360°

1 2 9 0 UNIVERSIDADE D COIMBRA

1 2 9 0

UNIVERSIDADE D
COIMBRA

Francisco Ferreira Santiago

OS PÉS NÃO SERVEM SÓ PARA ANDAR

INTERACÇÃO COM OS PÉS EM REALIDADE VIRTUAL 360°

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas de Informação, orientada pelo Professor Doutor Jorge C. S. Cardoso e apresentada ao Departamento de Engenharia Informática Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro de 2021

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Informática

Os Pés Não Servem Só Para Andar

Interacção com os Pés em Realidade Virtual 360°

Francisco Ferreira Santiago

Dissertação de Mestrado no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas de Informação orientada pelo Professor Doutor Jorge C. S. Cardoso e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Janeiro de 2021

1 2 9 0

UNIVERSIDADE D
COIMBRA

Esta página é deixada intencionalmente em branco.

Resumo

A interacção em sistemas de realidade virtual (RV) é, no geral, realizada por controladores *standard* (controladores de mão). No entanto, considerar outras formas de *input* torna-se importante para estudar soluções alternativas em situações nas quais os controladores *standard* não podem ser usados ou simplesmente porque se pretende oferecer uma experiência diferente ao utilizador, como por exemplo, o uso dos pés para interacção em RV.

Assim, procurou-se realizar um estudo sobre o pé, explorando os movimentos que este pode realizar e as várias interacções já desenvolvidas na literatura. Neste sentido, o presente trabalho pretende desenvolver e avaliar a implementação de uma biblioteca capaz de utilizar os pés como forma de interacção com RV.

Para atingir este fim, foi desenvolvido um projecto em *Unity* recorrendo ao *headset* e *tracker*, ambos HTC VIVE (Gen. 1), que tem como função a utilização do pé para navegação através de um sistema de menus.

Os testes de avaliação do sistema revelaram que este é de fácil implementação por parte de programadores. Ao nível das funcionalidades de utilização o sistema revelou-se ser *user-friendly*, não sobrecarregando em demasia os utilizadores.

Este trabalho torna possível a utilização do pé como principal modo de interacção para navegação em ambientes virtuais, libertando assim as mãos para outro tipo de interacções. Permite também a utilização dos pés como forma de interacção com diversos tipos de menus como, por exemplo, a selecção de comandos para reprodução de multimédia (*play, pause, stop, forward*, entre outros) ou a manipulação de objectos e das suas propriedades.

Palavras-Chave

"Interacção humano-computador", "Realidade Virtual", "Interacção com pés","Foot tracking", "Foot input"

Esta página é deixada intencionalmente em branco.

Abstract

The interaction in virtual reality (VR) systems is, in general, performed by standard controllers (hand-held controllers). However, considering other forms of input becomes important to study alternative solutions in situations where standard controllers cannot be used or simply because it is intended to offer a different experience to the user, such as using the feet as a mean for interaction in VR.

Thus, we sought to carry out a study on the foot, exploring the movements it can perform and the various interactions that are already developed in the extant literature. In this sense, this work intends to develop and evaluate the implementation of a library capable of using the feet as a form of interaction with VR.

To achieve this goal, a project was developed in Unity framework using the headset and tracker, both HTC VIVE (Generation. 1), which has the function of using the foot for navigation through a menu system.

System evaluation tests revealed that it is relatively easy to be implemented by other programmers. In terms of functionalities of use, the system proved to be user-friendly, not overloading users.

This work makes it possible to use the foot as the main mode of interaction for navigation in virtual environments, thus freeing the hands for other types of interactions. It also allows the use of feet as a way of interacting with different types of menus, such as selecting commands for media playback (play, pause, stop, forward, among others) or manipulating objects and their properties.

Keywords

"Human-Computer Interaction", "Virtual Reality", "Foot interaction","Foot tracking", "Foot input",

Esta página é deixada intencionalmente em branco.

Agradecimentos

Por muitas que sejam as dificuldades encontradas ao longo do caminho, há sempre um suporte ou um apoio que fazem com que a motivação para fazer melhor esteja presente. Ao reflectir acerca deste trabalho, não posso deixar de agradecer aos que me apoiaram na realização do mesmo e estiveram presentes no decorrer desta jornada.

Em primeiro lugar, ao Professor Doutor Jorge C. S. Cardoso, devo prestar o meu mais sincero agradecimento, por toda a orientação prestada, a constante disponibilidade e o encorajamento para fazer melhor.

À minha cidade, a minha Coimbra, por ser a melhor cidade para se estudar. Por toda a experiência académica, projectos em que me envolvi e amizades que construí. Levarei comigo para a vida.

A todos os meus amigos e colegas de faculdade, pelo companheirismo e solidariedade.

À minha família, Isabela, Ana e Marcelo, por todo o apoio demonstrado, constante disponibilidade e paciência.

A todos, muito obrigado!

Esta página é deixada intencionalmente em branco.

Conteúdo

Esta página é deixada intencionalmente em branco.

Acrónimos

ANN Artificial Neural Networks. 21, 22, 62

DEI Departamento de Engenharia Informática. 33, 77

FCTUC Faculdade de Ciências e Tecnologias da Universidade de Coimbra. 1

HMD Head-Mounted Display. 1, 14

HTC High-Tech Computer. 14

IHC Interação-Humano Computador. 6, 11

MEI Mestrado de Engenharia Informática. 1

RA Realidade Aumentada. 16

RV Realidade Virtual. 1, 2, 6, 8, 14–17, 35, 36, 47

Esta página é deixada intencionalmente em branco.

Lista de Figuras

Esta página é deixada intencionalmente em branco.

Lista de Tabelas

Esta página é deixada intencionalmente em branco.

Capítulo 1

Introdução

Esta dissertação foi desenvolvida no âmbito da disciplina "Dissertação/Estágio", no ano lectivo 2020/2021, inserido no Mestrado de Engenharia Informática (MEI), da Faculdade de Ciências e Tecnologias da Universidade de Coimbra (FCTUC).

O presente capítulo pretende introduzir o trabalho evidenciando o contexto e motivação para a realização do mesmo e os seus objectivos, bem como apresentar a estrutura da dissertação.

1.1 Contexto e Motivação

O conceito de Realidade Virtual (RV) foi formalmente definido pela primeira vez em 1965 por Ivan Sutherland [9], quando deu a conhecer o projecto *Ultimate Display*. A RV pode ser considerada como uma interface entre o utilizador e um sistema computacional, capaz de simular um ambiente virtual gerado computacionalmente, permitindo que o utilizador tenha interacção com os objectos virtuais. Esta interacção é normalmente conseguida através de controladores de mão, que são o padrão na utilização para controlo da experiência em RV. Contudo, é importante considerar outras formas de interacção a fim de encontrar soluções alternativas para situações em que os controladores de mão não podem ser usados ou, simplesmente, para fornecer uma diferente experiência ao utilizador. Por exemplo, a utilização de controladores de pé pode libertar as mãos para diferentes usos, tornando a experiência mais imersiva para o utilizador.

Actualmente, a interacção com Head-Mounted Display (HMD) depende fortemente do uso das mãos do utilizador. Apesar de outras parte do corpo humano já terem sido exploradas, ainda não houve uma avaliação sistemática quanto à interacção baseada nos pés para HMD. Neste sentido, o presente trabalho visa colmatar estas ausências de interacção com os pés através do desenvolvimento de uma biblioteca que reúna um conjunto de elementos de interacção para RV, de modo a serem aplicados em ambiente de visita virtual.

1.2 Objectivos

O objectivo desta dissertação recai no desenvolvimento de um *toolkit* que facilite, tanto o uso por utilizadores, como a implementação por parte de programadores, de conteúdos de RV no que diz respeito a interacções com os pés. Embora a detecção de pés possa ser realizada através de vários mecanismos, neste projecto o objectivo não é desenvolver nenhum mecanismo que detecte os pés. Procura-se desta forma desenvolver um conjunto de componentes que tenham como foco a interacção com os pés e que possam ser aplicadas em visitas virtuais.

É também objectivo desta dissertação a avaliação da componente desenvolvida, nomeadamente quanto à facilidade da sua implementação por parte de programadores. Mais ainda, pretende-se também avaliar a usabilidade da componente, bem como a experiência do utilizador em ambiente de visita virtual.

Por fim, a dissertação tem também como objectivo disponibilizar todo o sistema criado em *open-source*, dando a liberdade a qualquer programador para usar a *toolkit* nos seus projectos.

1.3 Estrutura da dissertação

Este documento está organizado em capítulos que retratam todas as fases do trabalho desenvolvido. O conteúdo do documento é descrito resumidamente de seguida:

1. **Introdução**: Descreve a motivação e os objectivos propostos para o projecto.
2. **Estado da Arte**: Revê a literatura acerca dos principais conceitos utilizados neste estudo, bem como as bases tecnológicas existentes.
3. **Metodologia e Planeamento**: Descreve a metodologia utilizada e o planeamento feio para cumprir os objectivos propostos.
4. **Implementação**: Descreve detalhadamente todo o processo de desenvolvimento da *toolkit*, incluindo a tentativa de detecção de gestos manual, tentativa de detecção através do uso de biblioteca de aprendizagem, uso de *colliders* para criação de um pedal e de uma componente *FootMenu* e a criação de uma demonstração de visita virtual 360º.
5. **Avaliação**: Descreve todo o procedimento da realização de testes, bem como a apresentação e discussão dos resultados obtidos.
6. **Conclusão**: Expõe as conclusões deste estudo e apresenta as limitações do mesmo, assim como indicações para trabalhos futuros.

Esta página é deixada intencionalmente em branco.

Capítulo 2

Estado da Arte

Neste capitulo é feito o levantamento de vários estudos realizados sobre o tema deste trabalho. Para tal, é importante perceber quais os movimentos que os membros inferiores conseguem desempenhar bem como as diversas interacções desenvolvidas. Sendo assim, este capitulo está organizado em diversas secções:

1. Fisiologia: recolha das características do membro inferior;
2. Interacções com os pés em ambiente virtual: diversas interacções implementadas por vários autores;
3. Categoria de gestos: categorização dos gestos dos pés;
4. Detecção do movimento dos pés: várias formas de detecção dos movimentos realizados pelos pés;
5. Background Tecnológico: caracterização das tecnologias existentes;

2.1 Fisiologia

O design de sistemas interactivos é geralmente optimizado para o movimento e capacidades das mãos. Portanto, é essencial perceber as qualidades e limitações dos membros inferiores, especialmente em comparação com os braços e mãos, a fim de definir interfaces que tenham em conta o alcance do movimento, peso e velocidade.

O pé é estruturalmente complexo sendo composto por 26 ossos, mais de 100 ligamentos e combina cooperação de músculos e tendões para manter o equilíbrio e impulsionar o corpo humano [10]. O movimento dos membros inferiores é realizado principalmente por três articulações em cada membro: o tornozelo, o joelho e o quadril (bacia). A Tabela 2.1 mostra a distribuição das amplitudes de movimento para cada uma destas articulações [6].

A articulação do tornozelo é capaz de três tipos de rotação, cada um em duas direcções: dorsiflexão/flexão plantar, abdução/adução e inversão/eversão. Dorsiflexão é o movimento que diminui o ângulo entre a parte superior do pé e a perna. Flexão plantar é o movimento que amplia o mesmo ângulo. Este tipo de movimentos podem ser usados para operar pedais [11].

Abdução é o movimento do pé que se afasta da linha central do corpo e adução é o movimento contrário em que o pé se aproxima da linha central do corpo. Como gesto, estes

Amplitude do Movimento (°)

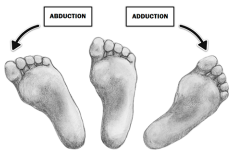
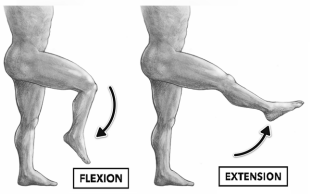
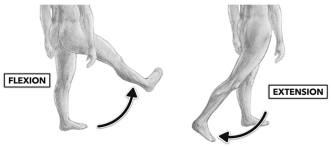
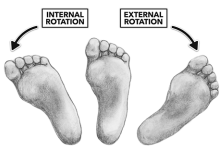
Articulação	Movimento	Média	Desvio padrão	
Tornozelo	Dorsiflexão	15.3	5.8	
	Flexão Planar	39.7	7.5	
	Abdução	20.1	2.4	
	Adução	41.3	5.3	
	Inversão	27.7	6.9	
	Eversão	27.6	4.6	
Joelho	Flexão	143.8	6.4	
	Extensão	1.6	2.8	
Quadril	Flexão	120.3	8.3	
	Extensão	9.4	5.3	
	Abdução	38.8	7.0	
	Adução	30.5	7.3	
	Rotação Externa	33.6	6.8	
	Rotação Interna	32.6	8.2	

Tabela 2.1: Alcance dos movimentos das articulações do tornozelo, joelho e quadril [6].

movimentos são interpretados como rotação do calcanhar (se girar em torno do calcanhar) ou como rotação sobre os dedos (se girar em torno dos dedos). Um exemplo de interface que é controlada por abdução ou adução é o FootMenu [3], em que o utilizador faz o pé pivotar ao redor do calcanhar para controlar o movimento horizontal do cursor.

Inversão é o movimento de torção para o interior, enquanto que eversão é o movimento contrário, ou seja, para o exterior. O alcance deste movimento ao longo de um eixo é bastante limitada. Estes movimentos normalmente são combinados com outras rotações em supinação (um movimento triplanar em que o pé se move para baixo e em direcção ao centro do corpo, combinando inversão, flexão planar e adução) e pronação (um movimento triplanar da articulação subtalar em que o pé se move para cima e se afasta do centro do corpo, combinando eversão, dorsiflexão e adbução). Supinação e pronação são os típicos movimentos usados para manipular horizontalmente o joystick de pé, pois permitem, com pouco movimento, uma mudança de peso do pé.

O joelho tem dois possíveis movimentos: rotação e flexão/extensão. Dado que a abdução/adução do pé é auxiliada pela rotação do joelho, estes são tratados em conjunto. A flexão do joelho é o movimento que diminui o ângulo entre a perna e o tornozelo, enquanto que a extensão do joelho é o movimento que amplia esse ângulo. A combinação destes movimentos resulta em "pontapé"[12].

O quadril tem rotação em três direcções: flexão/extensão, abdução/adução e rotação externa/interna. Devido ao facto das rotações do quadril envolverem toda a perna, geralmente torna-se cansativo a serem usados para Interação-Humano Computador (IHC), mas muitas vezes auxiliam o movimento de outras articulações. Por exemplo, o chuto pode ser produzido usando a força da perna [12]. O quadril pode ser usado para deslocar o centro de massa do corpo, o que é útil no uso das interfaces com sensores de pressão, como é o exemplo da Wii Balance Board [13, 14, 15]. Uma outra interface mecânica que usa directamente o movimento do quadril é o banco Swopper, onde este aproveita o movimento do quadril do utilizador para ter a mesma função que um joystick [16].

2.2 Interacção com os pés em ambiente virtual

Segundo Bowman e Hodges [17] podemos definir as interacções no ambiente virtual em três tipos gerais, que são apresentados nas secções seguintes: selecção, manipulação e locomoção.

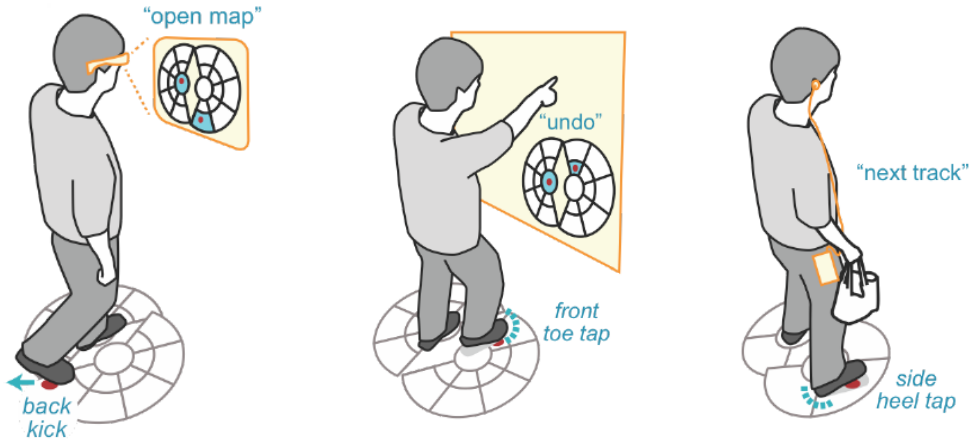
2.2.1 Selecção

Uma tarefa vulgar em Realidade Virtual (RV) é a selecção de um objecto, como um botão no menu. Normalmente é feito apontando, ou tocando, no objecto pretendido. Por exemplo, Muller et al. [1], no sistema Mind the Tap, faz uso das batidas do pé para seleccionar uma opção do menu que é apresentado ao utilizador. Muller et al. só considerou as interacções do pé em menus semi-circulares. Desta maneira, a batida do pé pode ser feita de forma directa ou indirecta (Figura 2.1). Na forma directa é apresentado, no chão há frente do utilizador, o conjunto de hipóteses, sendo estas seleccionadas através da batida do pé no quadrante desejado. A outra forma de interacção, difere um pouco da anterior na medida em que o conjunto de hipóteses fica a "flutuar" á frente dos olhos do utilizador.

Uma abordagem semelhante foi desempenhada por Saunders e Vogel [2] onde usaram a batida do pé e o movimento de chuto para a selecção . Neste caso, o menu de selecção é representado por uma circunferência em torno do pé. O pé é representado por um ponto vermelho, que está no centro da circunferência, e de seguida, usando a batida ou chuto, o utilizador selecciona a opção que deseja, como ilustra a Figura 2.2.

(a) Forma directa **(b)** Forma indirecta

Figura 2.1: Mind the Tap [1]: batida do pé na selecção



(a) Chuto de calcanhar **(b)** Batida frontal com a ponta do pé **(c)** Batida lateral com o calcanhar

Figura 2.2: Abordagem Saunders e Vogel [2]

Outra forma de selecção é fazer uso da rotação do calcanhar para escolher o objecto que se quer. FootMenu faz uso dessa técnica onde o calcanhar está fixo e o pé roda sobre esse eixo, seleccionando o objecto que se pretende, como se pode observar na Figura 2.3. Devido ao limite de rotação que o pé pode ter, quando numero de itens aumenta, o espaço para cada um diminui, o que influencia o desempenho [3].

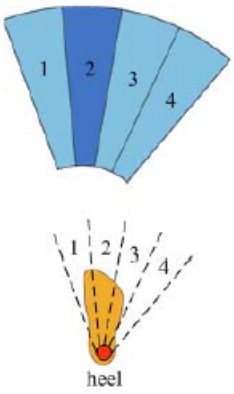
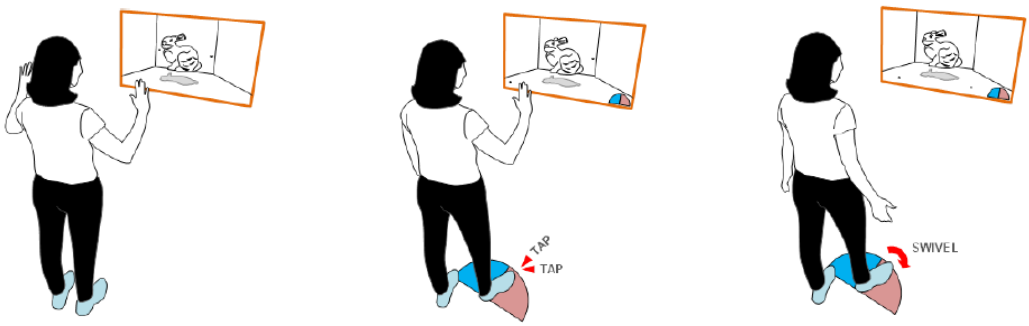


Figura 2.3: FootMenu [3]: exemplo com 4 itens

2.2.2 Manipulação

Uma outra forma de interacção em RV é a manipulação de objectos onde, de alguma maneira, se altera algum atributo desse objecto. Um exemplo seria alterar o tamanho ou a cor do item ou mover/rodar o item para uma nova posição no ambiente virtual. De notar que na maioria dos casos, a selecção tem de ser realizada antes da manipulação. FEETICHE [4] combina o pé dominante e os gestos das mãos para a manipulação de objectos 3D, como ilustra a Figura 2.4. O pé numa fase inicial é usado para seleccionar, através da rotação do calcanhar e de batida, a opção de translação, rotação ou deslocação. Depois da selecção estar feita, o utilizador usa as mãos para realizar a tarefa, ou no caso da rotação e deslocação, podem ser realizadas somente com o pé. Nestas duas operações o utilizador tem primeiro que seleccionar, com as mãos, o eixo que pretende usar e de seguida utiliza o pé para proceder com a manipulação, em que basta rodar o calcanhar para a opção de rotação e na deslocação basta posicionar o pé onde pretende ter o objecto. Um outro exemplo é o *augmented football game* onde o utilizador usa os movimentos do pé para movimentar a bola dentro do campo virtual [18].



(a) Manipulação de objectos 3D usando as mãos e pés

(b) Comandos de selecção

(c) Transformações geométricas

Figura 2.4: FEETICHE [4]

2.2.3 Locomoção

A acção mais comum no mundo virtual é a locomoção, em que o utilizador se move duma localização para outra no ambiente virtual. Como no mundo real, o utilizador move-se para explorar o ambiente em diferentes ângulos, ou para ter um ponto de vista diferente de um objecto próximo. A locomoção pode ser realizada por diferentes métodos. O método de locomoção *"Center of Pressure"* usa directamente a pressão exercida pela planta do pé para mapear directamente o movimento, como se fosse um *joystick* [14, 19, 20]. O utilizador ao querer mover-se, usa a inclinação, na orientação que deseja ir, e a pressão que é exercida pelo pé é mapeada directamente no movimento.

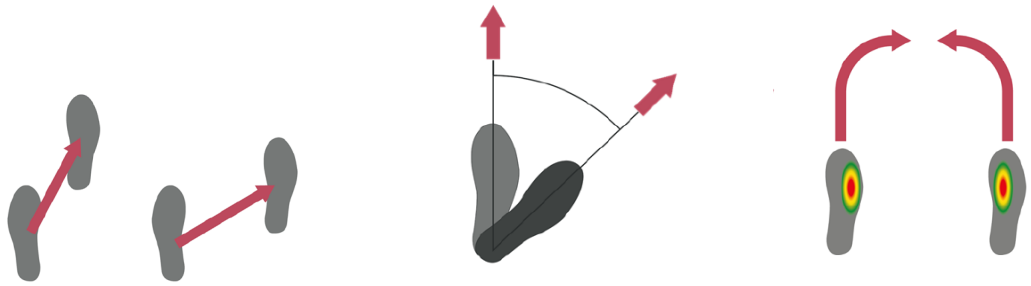
Uma outra alternativa é *"Walking in Place"* em que o utilizador imita o movimento de andar sem sair do lugar [21, 22]. Neste caso, o utilizador usa o movimento das pernas para imitar andar/correr mas sem sair do mesmo sítio. Uma abordagem mais natural é contrariar o movimento do utilizador prendendo-o ou movendo-o no lugar, deixando assim as mãos livres para puder usar noutras interacções [23, 24, 25]. Este método é normalmente usado por hardware especializado, como uma passadeira rolante, em que acção de mover é sempre contrariada, deste modo o utilizador faz o movimento de deslocação mas não sai do sítio. Contudo esta abordagem pode induzir náuseas [26, 27].

Para a locomoção, o uso de acções abstractas permite que o utilizador permaneça no lugar enquanto se move no mundo virtual. Isto pode ser realizado através da navegação baseada na direcção do olhar, técnica conhecida por GAZE. Esta técnica faz o uso combinado de pedais com os olhos, sendo os pedais de vários tipos: unidireccional, bidireccional ou multi-direccional. O utilizador para realizar esta técnica usa os olhos para fixar uma posição no mundo virtual e de seguida utiliza os pedais para realizar operações de deslocamento [28].

O utilizador pode ser também iludido em andar em círculos, mais conhecido por *redirected walking*. No geral, esta técnica aplica pequenas rotações ao ecrã do utilizador, sem que ele se aperceba, de modo a que este altere a sua rota de deslocação, ou seja, ao utilizador pode ter a percepção de que percorreu uma enorme distancia física ao mover-se no ambiente virtual mas na realidade no espaço físico esteve a mover-se em círculos [29]. Uma recente implementaçaõ faz uso de estimuladores eléctricos musculares, presos à perna, que manipulam a direcção do andar no mundo real [30]. Estes estimuladores eléctricos fazem com que a perna do utilizador rode consoante o andar do mesmo, ou seja, o utilizador tem a sensação que está a andar em linha recta no mundo real, embora esteja a andar em círculos sem que se aperceba. Contudo esta abordagem requer grandes espaços abertos visto que o trajecto do utilizador só pode ser ligeiramente alterado.

Por fim, a abordagem mais frequente para a locomoção é o tele-transporte, que permite o utilizador mover-se distancias virtuais arbitrárias sem se mover no mundo real. Wilich et al. notou que esta abordagem requer, do utilizador, a direcção e a distância para que possa ser realizada [31]. Para a direcção, existem três métodos diferentes:

- *Inter feet direction*: Para o primeiro método, o pé passivo age como ponto de referência. A direcção é definida pelo vector entre o pé passivo e o pé activo, como a Figura 2.5 a) demonstra.
- *Foot direction*: Neste método, a direcção é definida pelo apontar do pé. O calcanhar do pé activo é usado como pivô, podendo este rodar sobre esse eixo. Deste modo, a ponta do pé determina a direcção, como é representado na Figura 2.5 b).
- *Point and lean*: Este tem uma abordagem semelhante que o *foot direction*. A direcção para a qual o pé está apontado determina a direcção principal de tele-transporte, mas desta vez o utilizador pode ajustar a direcção, tanto para a esquerda como para a direita, deslocando o seu peso sobre o pé activo, como é visto na Figura 2.5 c). Esta mudança de peso resulta em pequenos ajustes na direcção correspondente.

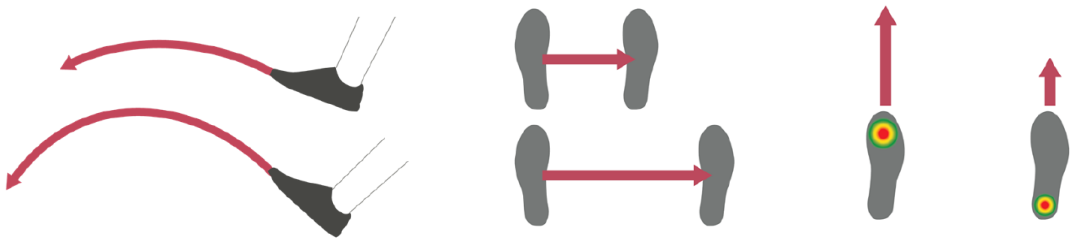


(a) *Inter feet direction* **(b)** *Foot direction* **(c)** *Point and lean*

Figura 2.5: Ilustração dos diferentes métodos de definir a direcção

Para a determinação da distância, existem também três modos diferentes:

- *Forefoot lift*: Esta abordagem faz uso de uma parábola, como é no caso do tele-transporte feito pelo controlador de mão. Ao levantar o antepé, o ângulo inicial da parábola é definido. O ponto de intersecção desta parábola com o chão é usado para estabelecer a distancia do tele-transporte. A trajectória da parábola é feita por uma simples função matemática. A Figura 2.6 a) mostra a representação desta abordagem.
- *Inter feet distance*: Aqui a distancia entre o pé activo e o pé passivo é determinado e escalado para designar a distancia de tele-transporte (Figura 2.6 b)). Para o utilizador definir distancias maiores de tele-transporte tem, há semelhança do andar para efectuar passos mais longos, que fazer os movimentos mais rápidos.
- *Intra foot pressure*: Para este método, a distancia é determinada pelo rácio da pressão exercida pelo pé activo. Se o utilizador aplicar mais pressão sobre a ponta do pé, a distancia de tele-transporte aumenta. Enquanto que se a pressão exercida for sobre o calcanhar, a distancia diminui. Figura 2.6 c) ilustra como esta técnica funciona.



(a) *forefoot lift* **(b)** *Inter feet distance* **(c)** *Intra foot pressure*

Figura 2.6: Ilustração dos diferentes métodos de definir a distância

2.3 Categoria de gestos

Nesta secção, irá ser abordada a interacção entre os utilizadores e o sistema. Mais especificamente, as acções que os utilizadores executam com os pés para as interacções. Karam e Schraefel [32] definiram a taxonomia para o gestos das mãos e Velloso et al. [7] baseou-se nessa mesma definição para distinguir quatro categorias de acções dos pés em IHC: semafóricos (*"qualquer sistema de gestos que emprega um dicionário especializado de gestos estáticos ou dinâmicos"* [33]), dêiticos (gestos que envolvem apontar), manipuladores (*"cuja finalidade é controlar alguma entidade aplicando uma relação directa entre os movimentos reais da mão/braço com a entidade que está sendo manipulada"* [33]) e implícito (gestos que não são expressados de forma intencional).

2.3.1 Semafóricos

Acções semafóricas são consideradas como gestos específicas pertencentes a um dicionário. Nesta secção observamos como estes gestos e as suas combinações são usadas na interacção com o sistema. A Tabela 2.2 mostra uma abrangente lista de gestos que Velloso et al. explorou [7]. A batida do pé (*toe tapping*) é o gesto mais comum que se encontra em vários sistemas na literatura [1, 2, 34, 35, 36, 37]. Tal se verifica visto que este gesto é realizado com pouco esforço e é comparado ao movimento que fazemos com o dedo indicador para seleccionar, por exemplo, um *icon* num ecrã táctil. Uma variação deste gesto é a batida com o calcanhar (*heel tap*) [2, 38, 39]. Em relação ao gesto anterior, este requer mais esforço uma vez que o utilizador tem que erguer todo o peso da perna, o que pode ser cansativo.

Na rotação do calcanhar ou dos dedos (*heel and toe rotation*), o utilizador fixa o pé sobre o calcanhar ou sobre os dedos e de seguida gira-o exercendo abdução ou adução sobre o tornozelo [3, 4, 39, 40]. O deslize (*swipe*) do pé é um gesto em que o utilizador move todo o pé numa certa direcção, o que requer algum esforço da perna [39, 40]. Combinando vários deslizes nas diversas direcções é possível traçar formas geométricas básicas (*shape trace*) [4, 40].

Um gesto que está bem examinado por Han et al. [41] é o chuto (*kick*). Ele investigou o nível de controlo do chuto nos utilizadores para a direcção e velocidade na selecção. Em vários sistemas descritos na literatura, o gesto de chutar é usado maioritariamente para selecção. [1, 2, 4, 42].

O passo (*step*) é a acção mais natural para o utilizador, dado que é usado sempre que o utilizador se quer deslocar no mundo real. Este gesto é a principal componente da

técnica de locomoção *"Walking in place"*, tendo em conta que o controlo da velocidade e da direcção são cruciais [21, 22, 43].

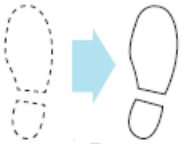
Gesture	Name	Description
	Toe Tap	User raises and lowers the toes
	Heel Tap	User raises and lowers the heel
	Toe Rotation	User pivots the foot around the toes
	Heel Rotation	User pivots the foot around the heel
	Swipe	User slides the foot in a certain direction
	Shape Trace	User draws the outline of a shape with the toes
	Kick	Vigorous movement of the foot in a certain direction
	Step	User puts one foot in front of the other as if walking

Tabela 2.2: Velloso [7]: Dicionário dos gestos semafórico dos pés.

2.3.2 Dêitico e Manipulador

As acções dêiticas são geralmente conhecidas por gestos de apontar. Quando o utilizador usa o pé para seleccionar um item específico, por exemplo, usar a batida para seleccionar uma tecla especifica do piano [18] ou seleccionar um objecto sobre o chão interactivo (*augmented floor*) [37], está a usar acções dêiticas. Acções manipuladoras fazem uso de elementos físicos do pé, como a posição e orientação, para alterar/transformar as propriedades de um objecto. Quando o utilizador arrasta o pé para rodar um objecto ou quando usa o posicionamento do pé para alterar a posição do objecto no ambiente virtual [4], está a ser usado acções manipuladoras. Embora estas duas acções tenham diferentes propósitos, são examinadas em conjunto porque ambas requerem que o pé passe de uma posição original para uma outra posição. Relativamente aos tempos de movimento e precisão, os estudos realizados por Chan et al. e Velloso et al. verificaram que existe pouca ou nenhuma influencia da dominância do pé na precisão e nos tempos de movimento em tarefas como batidas repetidas (*tapping*) e no apontar (*pointing*), mas observaram que os movimentos laterais são mais rápidos e fáceis que os movimentos para a frente e para trás [7, 34].

No que diz respeito aos tempos de reacção, Simonen et al. comparou os tempos de reacção das mãos e dos pés dominantes e encontrou que os esses tempos são praticamente os mesmos, com a mão a ser ligeiramente mais rápida que o pé [44].

2.3.3 Implícito

Os gestos dos pés podem ser usados para interacções explicitas ou implícitas. Schmidt [45] fez a distinção entre estes dois tipos onde nas interacções explicitas "o utilizador comunica ao computador, em um certo nível de abstracção (...) o que espera que o computador faça", enquanto que as interacções implícitas, o sistema reage as interacções do utilizador que não são direccionadas directamente há interacção do sistema, por outras palavras, o sistema interpreta as acções inconscientes que o utilizador gera como uma acção.

Vários sistemas conseguem extrair informações do comportamento do pé sem que o utilizador use explicitamente os gestos para interacção. Por exemplo, o uso de sapatos inteligentes é útil para melhorar patologias osteoarticulares da coluna, visto que consegue detectar possíveis problemas na coluna, através do andar da pessoa, pela pressão que a planta do pé exerce [46].

Multitoe [37] consegue distinguir os diferentes utilizadores através da monitorização da postura dos pés de cada indivíduo, bem como detectar objectos.

2.4 Detecção do movimento dos pés

Várias são as formas de detenção dos movimentos dos pés que o utilizador produz. Nesta secção é feito o levantamento das várias maneiras bem como a categorização das mesmas. Os movimentos do pé podem ser monitorizados directamente, através de sensores preso ao membro, ou indirectamente, através de dispositivos que ele controla. Desta maneira dividimos a captura dos movimentos em três categorias: detecção mediada, detecção intrínseca e detecção extrínseca.

A detecção mediada, como a palavra indica, acontece quando a captura dos movimentos do pé não é feita directamente, mas sim através do dispositivo que o pé está a operar. Esta

categoria tem maioritariamente instrumentos mecânicos, como os pedais ou ratos, que têm partes moveis que interpretam o movimento exercido. Começando pelo mais antigo e talvez o mais conhecido é o pedal. O pedal normalmente só tem uma direcção, pressiona-lo ou liberta-lo, como acontece nos carros, no entanto se aumentarmos o grau de liberdade que se pode ter, obtém-se um pedal bidireccional, que para além do pedal uni-direccional, este permite que a inclinação seja nos dois sentidos, para a frente e para trás, e multi-direccional, também conhecido como joystick de pé, onde o pé consegue movimentar-se livremente em todas as direcções [28, 47]. A bola de comando, mais conhecido por *trackball*, é um periférico de entrada muito semelhante a um rato de mão antigo. Ao contrário do rato se mover sobre uma superfície plana, na *trackball* o utilizador manipula uma esfera que está na parte superior, usado para controlo de cursor [48]. Semelhante a *trackball*, temos o rato de pé. Este dispositivo funciona da mesma maneira que o rato de mão, movendo o pé, só que desta vez os cliques do rato são simulados por um conjunto de botões a serem operados pelo outro pé. Por fim temos a Wii Balance Board que faz uso de quatro sensores de pressão que medem o centro de gravidade do utilizador [14, 20].

(a) *Trackball* **(b)** *Foot Mouse* **(c)** Wii Balance Board

Figura 2.7: Exemplos de detecção mediada

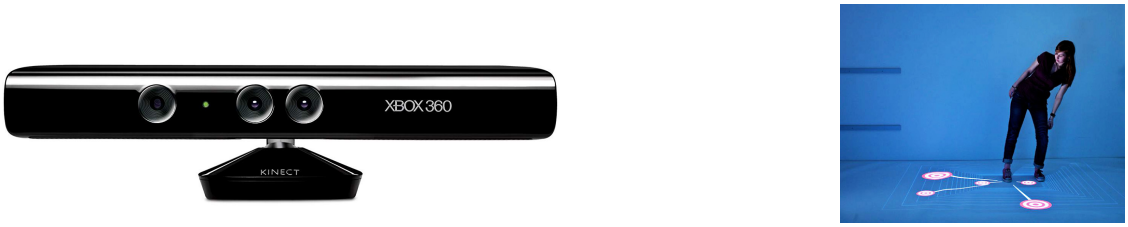
A detecção intrínseca refere aos equipamentos que monitorizam o pé através de sensores ligados directamente ao membro. Desta forma, os sensores de pressão, que se podem encontrar nos sapatos e palmilhas inteligentes, conseguem calcular a distribuição de peso do utilizador bem como a força exercida por cada membro inferior. Outros tipos de sensores, tais como os acelerômetros e giroscópios, conseguem extrair a informação do movimento, sendo assim possível estimar a posição e a orientação do pé. Este tipo de sensores podem ser encontrados nos smartphones, o que facilita o uso [39]. Uma outra forma que é muito utilizada em ambientes RV é o uso de *trackers* para a captura dos movimentos. Estes *trackers*, como é o exemplo do High-Tech Computer (HTC) Vive Tracker, usa o sistema de lasers do *Head-Mounted Display (HMD)* para obter a posição e orientação exacta do aparelho [40]. Uma alternativa ao *tracker* é fazer uso do próprio controlador do HMD, visto que o controlador foi desenvolvido para ser usado pelas mãos, coloca-lo no pé poderá tornar-se desconfortável.

(a) Palmilhas inteligentes **(b)** HTC Vive tracker

Figura 2.8: Exemplos de detecção intrínseca

Relativamente a detecção extrínseca, refere-se à detecção dos pés por sensores que estão colocados no meio ambiente. Existem vários tipos de câmaras que podem ser usadas para alcançar este meio. Um dos tipos são as câmaras de captura de imagem vídeo, capazes de extrair a posição e orientação dos pés, através da detecção do contorno do membro, contudo apresentam baixa precisão e a necessidade de um foco directo para o mesmo [18].

Outro tipo de câmara são as de profundidade, como a Microsoft Kinect e Intel RealSense. Estas conseguem monitorizar com precisão o pé nas três dimensões, fazendo uso de raios infra-vermelhos para determinar a posição dos membros [2, 4]. Os *tablets* são capazes de detectar os toques, arrastamentos e pressão exercida com as mãos. Uma técnica semelhante ao *tablet* para a detecção dos movimentos dos pés é o uso de um chão *multi-touch*, integrando *frustrated total internal reflection*. Este chão é capaz de determinar com precisão os diferentes tipos de toque com o pé e também consegue saber qual a pressão exercida por cada membro [37].



(a) Camara Kinect **(b)** Chão *multi-touch*

Figura 2.9: Vários tipos de detecção intrínseca

2.5 Background Tecnológico

Nesta secção, irão ser descritas as várias tecnologias existentes para concepção, desenvolvimento e implementação de experiências de RV, tanto de *hardware* como de *software*. Neste sentido, irão ser abordados vários *headsets*, *trackers* e *frameworks* que são comummente utilizados para o desenvolvimento deste tipo de experiências.

Ao nível dos *headsets*, podem ser destacados os seguintes:

- HTC VIVE (Gen. 1) [49]: foi introduzido no mercado em Abril de 2016 e o seu desenvolvimento contou com a parceria entre a empresa HTC e a empresa Valve, sendo o primeiro *headset* comercial. Este conta com uma tela de 90Hz com resolução de 2160x1200 pixeis e 37 sensores de captação espalhados na parte frontal do equipamento. O *tracking* é feito por um sistema *Lighthouse* que contem emissores de laser infravermelho que são emitidos através do espaço físico onde estão instalados. Esses raios são captados pelos sensores que estão no *headset* que, por sua vez, calcula a posição e a orientação exacta do equipamento. Contudo, este sistema não é aconselhado a ser usado ao pé de janelas ou em espaços onde a luz incida directamente, devido ao facto de poder haver interferências com o sistema infravermelho.
- VALVE INDEX [50]: foi introduzido no mercado a 28 de Junho de 2019, seu desenvolvido e criado por Valve. Este *headset* tem uma tela de resolução de 2880x1600 pixeis, com uma *refresh rate* até 120 Hz. À semelhança do HTC VIVE faz uso de um sistema de *tracking* através de *lighthouses*, no entanto, este tem mais precisão no *tracking* porque são usadas 3 *lighthouses* em vez de 2, aumentando assim a possibilidade de captura eliminando ângulos mortos. O controladores deste sistema são actualmente os mais avançados do mercado, integrando um sistema de 87 sensores,

monitorizando a posição, movimento e pressão, permitindo criar uma representação mais precisa das mãos do utilizador.
– OCULUS QUEST 2 [51]: foi desenvolvido pela Facebook e introduzido no mercado a 13 de Outubro 2020, tornando-se na opção mais popular do mercado. Apresenta uma tela de 1832x1920 pixeis por olho, com uma *refresh rate* de 90 Hz podendo ir até 120 Hz (ainda experimental). Este é um dispositivo *standalone*, não necessitando de um computador para a sua utilização, contudo é possível fazê-lo. Este *headset*, ao contrário dos anteriores, faz o tracking através de 4 câmaras existentes no exterior do dispositivo. Através destas, este *headset* consegue monitorizar os controladores, fazendo ainda o *tracking* das mãos de forma a substituir os controladores.

Quanto aos *trackers* destacam-se os segintes:

– HTC VIVE TRACKER [52]: à semelhança do *headset* este foi o primeiro dispositivo de *tracking* a ser comercializado. Este dispositivo é monitorizado pelo mesmo sistema de *lighthouses*, tendo no máximo um alcance de 10 metros. Contudo este requer um número de *dongles* igual ao número de dispositivos a serem utilizados. Recentemente, foi lançado uma nova versão (v3.0) que apresenta uma melhor monitorização, bem como uma maior duração de bateria.
– THUNDRA [53]: à semelhança do dispositivo anterior, este também utiliza o sistema de *lighthouses*. Contudo, este é mais barato, mais pequeno e podem ser utilizados vários *trackers* com apenas um *dongle*.
– SLIMEVR FBT [54]: este dispositivo é composto por um conjunto de 5 *trackers* e, ao contrário dos anteriores, estes conectam-se através de *wi-fi*. O seu sistema de sensores permite que este dispositivo faça o *tracking* sem recorrer a *lighthouses* ou outros aparatos externos.

No que diz respeito às *frameworks* para desenvolvimento de conteúdo RV destacam-se as seguintes:

– A-FRAME [55]: esta é uma *framework web* para a criação de experiências de RV, com suporte para a maioria dos *headsets*. Esta tem como base um sistema de componente e entidade (SCE) que permite criar os cenários de RV usando as bibliotecas *Tree.js* e *WebXR*, sobre a linguagem de programação HTML. Esta *framework* tem interoperabilidade na web, na qual é compatível entre os diversos dispositivos, navegadores e sistemas operativos, não necessitando de *software* externo. Porém, até ao momento, a *framwork* não contém suporte para a implementação de dispositivos como os *trackers*.
– UNITY [56]: é uma ferramenta de desenvolvimento de jogos 2D/3D e ao longo do tempo tem vindo a impulsionar o desenvolvimento em RV e Realidade Aumentada (RA). Esta *framework* tem como base a linguagem de programação C# (*C sharp*) e disponibiliza um conjunto de *pluggins* que torna desnecessária a configuração para diversas plataformas de *hardware*. Assim, os programadores podem focar-se somente na implementação dos recursos da experiência que pretendem e não gastar tempo em implementar repetidamente o mesmo recurso para cada uma das plataformas.
– UNREAL ENGINE [57]: à semelhança da *framework* anterior, esta é também uma ferramenta de desenvolvimento de jogos e recentemente tem vindo a impulsionar o desenvolvimento em RV e RA. No entanto existem diferenças entre ambas, sendo a mais preponderante a linguagem de programação que, neste caso, é C++, sendo esta *framework* a mais complexa de programar;

Capítulo 3

Metodologia e Planeamento

Como em qualquer projecto de desenvolvimento, é essencial definir um bom planeamento. Nesta secção é apresentada a metodologia de trabalho, as razões que levaram a esta decisão e o planeamento para o desenvolvimento do projecto.

3.1 Metodologia de desenvolvimento

O objectivo deste projecto é desenvolver uma biblioteca que sirva de base para a implementação de diversas interacções do pé em ambientes de RV.

Para atingir o objectivo do projecto, foi necessário fazer o levantamento da fisiologia dos membros inferiores, para saber que tipos de gestos/movimentos é possível realizar com os mesmos, bem como saber as várias formas de detecção desses movimentos e que tipos de interacções já foram usadas noutros projectos desenvolvidos. Para este fim, foi utilizada a seguinte tecnologia:

headset: HTC VIVE Virtual Reality System (Gen 1).
tracker: HTC VIVE tracker (Gen 1).
framework: UNITY 3D (v 2019.4.25f1)

Inicialmente, foi feita uma pequena prova de conceito recorrendo às tecnologias referidas anteriormente. Este consistiu em detectar o *tracker* e interagir com as propriedades de um objecto, mais concretamente, o objecto ao ser tocado pelo *tracker* alterava a sua cor. Esta prova de conceito mostrou que é possível levar a cabo o projecto nesta dissertação.

Após isto, foram estudadas as propriedades do *tracker* utilizado de forma a reconhecer os gestos. De seguida, foi adaptada uma biblioteca existente para o reconhecimento de movimentos. Então, foram usados *colliders* para a criação das componentes de pedal e do *FootMenu*, que culminou na criação de uma demonstração de visita virtual 360º, com recurso à componente.

Por fim, foram feitos dois tipos de avaliação. Uma que diz respeito à usabilidade do sistema e experiência do utilizador, enquanto que a outra diz respeito à complexidade da tarefa de programação.

Para tal, irá ser usada a metodologia ágil para gerir o projecto, mais concretamente a metodologia *Personal Kanban* [58]. Esta metodologia foi escolhida devido ao facto de permitir uma gestão flexível do projecto, sendo possíveis alterações, focando nas tarefas que mais importam, tendo melhor controlo sob o tempo de implementação. Personal

Kanban foca na ideia de ter um quadro dividido em três principais secções: *To-Do*, *Work in Progress* e *Done*, podendo estas secções serem divididas em outras sub-secções, como é exemplificado na Figura 3.1.

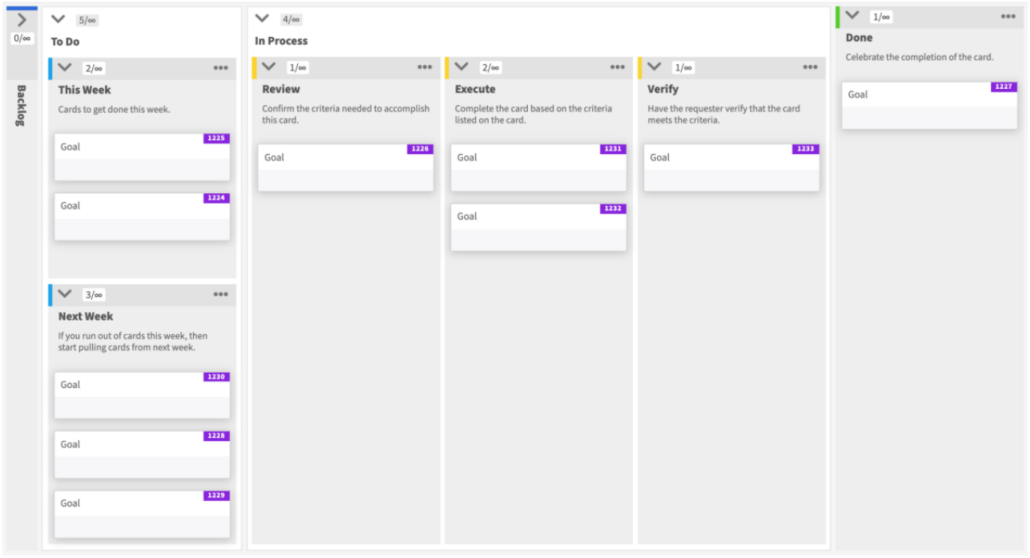


Figura 3.1: Exemplo de quadro do método *Personal Kanban*

As principais vantagens desta técnica são:

- Visualizar o trabalho de perto: Permite que em qualquer altura se tenha a noção da carga de trabalho a ser executada. Simplifica a visão do que é para realizar a seguir, atendendo as prioridades.
- Limitar o trabalho em progresso: Restringe o número de tarefas a serem realizadas ao mesmo tempo, o que facilita a monitorização do projecto e evita problemas de *multitasking*.

3.2 Escalonamento de tarefas

Para controlar o desenvolvimento do projecto foi elaborado um planeamento que contém as tarefas a realizar. Desta forma, é apresentado o plano de trabalhos seguido ao longo do ano lectivo.

A tabela 3.1 mostra o desenrolar das tarefas necessárias para iniciar este projecto, nomeadamente a revisão do estado da arte e a familiarização com a tecnologia existente.

	Semanas	
Tarefa	**Inicio**	**Fim**
Revisão da literatura e estado da arte	**20/set**	**29/nov**
Levatamento da fisiologia	20/set	04/out
Exploração de projectos que fazem uso de interações com os pés	04/out	08/nov
Levatamento das tecnologias para detecção do pé	08/nov	22/nov
Levantamento de várias frameworks de programação	22/nov	29/nov
Familiarização das tecnologias	**29/nov**	**13/dez**
Conhecimento da linguagem de programação	29/nov	29/nov
Experimentação do tracker	29/nov	06/dez
Elaboração de uma prova de conceito	06/dez	13/dez
Dissertação	**13/dez**	**31/jan**
Escrtia	13/dez	17/jan
Preparação para a apresentação	17/jan	31/jan

Tabela 3.1: Tabela de planeamento do 1º semestre

Na tabela 3.2 estão reflectidas as tarefas levadas a cabo para o desenvolvimento da biblioteca criada neste projecto, passando tanto pelas tentativas de reconhecimento de gestos, como pela implementação de uma componente de *FootMenu* e também pela sua avaliação por utilizadores e programadores.

	Semanas	
Tarefa	**Inicio**	**Fim**
Tentativa de detecção de gestos manual	**06/fev**	**06/mar**
Recolha e processamento de dados	06/fev	13/fev
Desenvolvimento do sistema	13/fev	06/mar
Testes	20/fev	06/mar
Tentativa de reconhecimento de gestos com uso a bibliotecas IA	**06/mar**	**17/abr**
Procura de bibliotecas	06/mar	13/mar
Gravação e interpretação de gestos	13/mar	20/mar
Desenvolvimento de sistema que use propriedades para reconhecimento	20/mar	27/mar
Desenvolvimento sistema Gaze	27/mar	17/abr
Testes	20/mar	10/abr
Pedal	**17/abr**	**08/mai**
Desenvolvimento de funcionalidade	17/abr	24/abr
Desenvolvimento da representação	24/abr	01/mai
Construção de um exemplo	01/mai	08/mai
Testes	24/abr	01/mai
Desenvolvimento FootMenu	**08/mai**	**19/jun**
Criação das propriedades	08/mai	15/mai
Criação do sistema de áreas	15/mai	05/jun
Desenvolvimento dos 2 tipos de perspectivas	05/jun	19/jun
Testes	15/mai	12/jun
Desenvolvimento de visita virtual	**19/jun**	**26/jun**
Recolha fotos 360º	19/jun	19/jun
Configuração do sistema de navegação	19/jun	26/jun
Testes	26/jun	26/jun
Avaliações	**26/jun**	**31/jul**
Testes usabilidade	26/jun	10/jul
Processamento dos dados	10/jul	17/jul
Testes de programação	17/jul	24/jul
Dissertação	**24/jul**	**14/set**
Escrita	24/jul	11/set
Preparação apresentação	11/jul	14/jul

Tabela 3.2: Tabela de planeamento do 2º semestre

No apêndice A encontram-se os diagramas de *Gantt* com as tarefas mencionadas anteriormente.

Capítulo 4

Implementação

Nesta secção é detalhada a implementação da biblioteca desenvolvida, que se apoia em cinco principais fases: tentativa de detecção de gestos manual, tentativa de detecção através do uso de biblioteca de aprendizagem, uso de *colliders* para criação de um pedal e de uma componente *FootMenu* e, por fim, a criação de uma demonstração de visita virtual 360°.

Esta biblioteca encontra-se disponível em `https://github.com/FranciscoSantiago15/Feet-Are-Not-Just-For-Walking`. Nesta directoria, encontra-se todo o projecto desenvolvido, como 2 *packages* e a documentação de utilização de uma componente.

4.1 Tentativa de reconhecimento de gestos de forma manual

O dispositivo que foi usado para a implementação (*tracker*) reúne um conjunto de propriedades que conseguem ser captadas pela *framework* de desenvolvimento *Unity*. Essas propriedades são [59]:

- *position* - define o vector (x,y,z) que representa a posição actual do dispositivo;
- *rotation* - define o *quartenion* que representa a rotação actual do dispositivo;
- *velocity* - define o vector que representa a taxa de variação da posição actual do dispositivo em relação ao tempo;
- *angularVelocity* - define o vector que representa a taxa com que a orientação do dispositivo muda;
- *acceleration* - define o vector que representa a taxa de variação de velocidade actual do dispositivo em relação ao tempo;
- *angularAcceleration* - define o vector que representa a taxa de variação da velocidade angular do dispositivo com o tempo;

Numa fase inicial, de modo a perceber como estas propriedades agem no desempenho de um gesto, foi efectuado o registo em ficheiro de vários movimentos, como o *toe-tap*, a rotação para a esquerda e direita e o chuto, em diversas orientações.

Após esse registo, verificou-se que a propriedade da rotação é a que tem mais influência sobre a forma como o gesto é capturado. A velocidade e a velocidade angular podem indicar se o pé está inerte ou em movimento.

Com estas informações procedeu-se para a implementação de um sistema que consiga detectar certos gestos. Durante a implementação, verificou-se que a orientação do utilizador

tem interferência directa sobre a rotação do *tracker*, visto que o sistema de coordenadas usado no dispositivo apresenta-se fixo em relação à *scene* virtual, assim quando o utilizador muda a sua orientação, o eixo pelo qual o *tracker* está a ser usado também se altera. Assim sendo, os gestos têm que ter certas características de forma a que o sistema os consiga identificar. Se o utilizador efectuar uma mudança de orientação antes de efectuar um gesto, o sistema faz uma identificação errada. Por exemplo, o utilizador para realizar um *toe tap* desloca o pé para cima e para baixo, exercendo rotação no eixo do X, mas quando o utilizador roda 90° para a sua direita e realizar o mesmo gesto, a rotação é agora exercida no eixo dos Y, alterando a forma de como o sistema identifica o movimento. A figura 4.1 ilustra como a rotação interage com o sistema de coordenadas do dispositivo. Além disso, a componente não consegue distinguir o deslocar do utilizador com a intenção de realizar um gesto, fazendo com que o utilizador tenha que permanecer imóvel durante a sua utilização.

Desta forma, conseguir implementar um sistema que identifique certos gestos através somente das propriedades do *tracker* é difícil e requer imenso tempo, visto que é preciso criar um mecanismo que consiga detectar as mudanças de rotação internas do dispositivo, por outras palavras, um mecanismo de coordenadas internas do *tracker*.

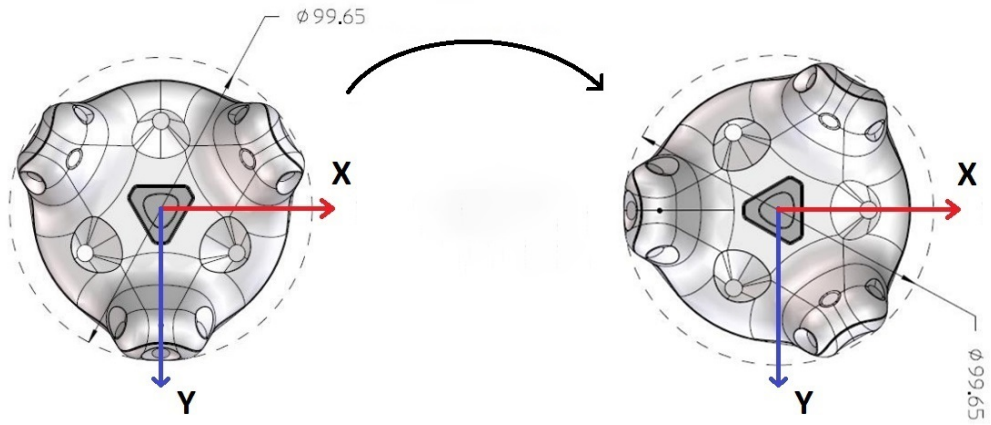


Figura 4.1: Sistema de coordenadas quando *tracker* sofre uma rotação de 90° para a direita [5]

4.2 Tentativa de reconhecimento de gestos com uso de biblioteca de aprendizagem

Após a fase anterior não ter dado o resultado esperado, devido às limitações do *tracker* e também à imensa complexidade que seria necessário para poder definir um movimento, optou-se então para outra abordagem de fazer uso de inteligência artificial para detectar gestos, mais propriamente fazer uso de *Artificial Neural Networks (ANN)*.

De uma forma geral, ANN [60] são modelos computacionais, inspirados pelo cérebro de um animal, com a particularidade de elaborar aprendizagem automática, bem como o reconhecimento de padrões. Tal como o cérebro, estas redes são baseadas num grupo conectado de *units* ou nós chamados de neurónios artificiais. Cada conexão, à semelhança das sinapses do cérebro, são capazes de transmitir um sinal (número real) a outros neurónios,

de forma a que o sinal recebido por um neurónio seja processado, através de uma função matemática, e de seguida transmitido para os restantes neurónios conectados a ele. Deste modo, através do processo de exemplos, as ANN conseguem realizar a aprendizagem/treino para a realização de reconhecimento de padrões.

Desta forma, o primeiro passo foi fazer um levantamento de bibliotecas e/ou *pluggins* existentes em *Unity* capazes de realizar a identificação de padrões, tendo sido encontrado a "MiVRy - 3D Gesture Recognition AI"[61]. Esta biblioteca, foi desenvolvida para reconhecer padrões produzidos pelo controlador de mão, de modo que, o utilizador ao premir o gatilho do controlador, dá início ao movimento de um gesto e, ssim que esse gesto estiver terminado, a biblioteca tenta reconhecer o que foi desempenhado, dando uma percentagem de acerto. Após a experimentação desta biblioteca, decorreu a implementação de uma componente capaz de fazer uso deste *pluggin* mas de forma a ser usado pelo *tracker* preso no pé. Na directoria da biblioteca *Assets/GestureRecognition* encontra-se todo o desenvolvimento desta componente.

De maneira a guardar todos os gestos que o utilizador pretende realizar, foi criado um *script* que detecte o dispositivo preso no pé e que recolha todas as posições por onde tenha passado, desde o inicio do gesto até ao fim do mesmo, e as guarde em ficheiros para que mais tarde possam ser usados pela biblioteca. Essa recolha é feita com o auxilio do controlador de mão em que o utilizador prime o gatilho para iniciar a captura e o solta para o terminar.

Após a criação da *script*, foram recolhidas diversas gravações de gestos, isto é, para cada gesto foram recolhidas 100 amostras de movimento, com o propósito de fornecer à biblioteca dados suficientes para que possa realizar a aprendizagem/treino. Os gestos gravados foram: *toe tap*, *double toe tap*, *rotation right*, *rotation left*, *swipe right*, *swipe left* e *kick*. Com os movimentos gravados, foi realizado a aprendizagem/treino. Depois da biblioteca ter treinado os dados, foi adicionado à *script* a possibilidade de identificação de gestos produzidos pelo utilizador, ainda com auxilio do controlador de mão para iniciar e terminar o movimento. Agora, sempre que o utilizador efectuar um gesto, a biblioteca consegue, com determinado grau de certeza, identificar o movimento realizado. Na figura 4.2 e 4.3 é possível verificar, respectivamente, o reconhecimento do gesto *toe tap* e a rotação à direita, pela componente. No apêndice B, estão representados os restantes reconhecimentos que a componente consegue realizar.

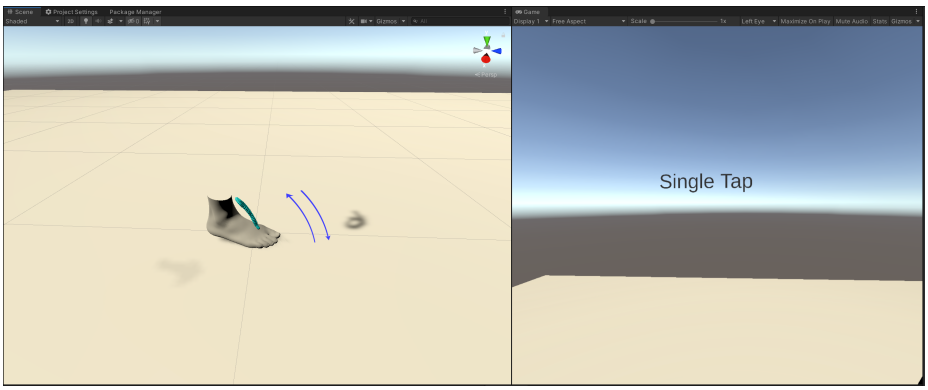


Figura 4.2: Reconhecimento do gesto *toe tap* pela componente desenvolvida

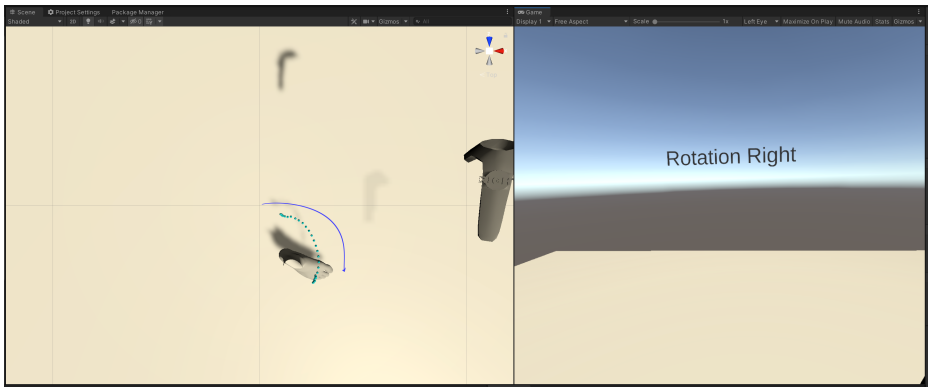


Figura 4.3: Reconhecimento do gesto de rotação à direita pela componente desenvolvida

Com a componente capaz de reconhecer padrões, a próxima etapa foi possibilitar a realização das operações sem o auxilio do comando. Para tal, houve experimentações em utilizar as propriedades do *tracker*, como a velocidade e a velocidade angular, para detectar a intenção do utilizador ao querer realizar um gesto. Verificou-se que usar estas propriedades tornam a componente limitativa, devido ao facto de ser sensível à movimentação do utilizador. O utilizador tem que, antes de realizar qualquer gesto, permanecer relativamente quieto. Para além disso, a componente não consegue distinguir a intenção do utilizador querer realizar um gesto como a intenção de se deslocar, assim sempre que o utilizador se movimenta no ambiente virtual ou altera a sua orientação, a componente detecta essa movimentação e começa a tentar reconhecer esse movimento, dando um *output* errado.

Tendo esta informação, foi necessário arranjar um mecanismo que consiga contornar estas limitações. Após alguma pesquisa, optou-se por fazer uso de uma técnica, que consta no capítulo 2 na secção da locomoção, conhecida por GAZE [28]. Foi implementado um sistema que faça uso desta técnica. O sistema desenvolvido contem uma retícula, com o propósito de simular o olhar do utilizador, e um mecanismo de *highligth*, que revela ao utilizador que o objecto que está a olhar pode ser seleccionado. Este sistema detecta a interacção da retícula com o objecto, alertando o utilizador que o objecto pode ser interagido, como ilustra a figura 4.4. No apêndice C está detalhado o funcionamento desta técnica.

(a) Objecto não seleccionado **(b)** Objecto seleccionado

Figura 4.4: Exemplo de funcionamento do sistema de GAZE

Com a técnica de GAZE desenvolvida, fez-se uma junção com a componente de reconhecimento de gestos. Esta junção permite que o utilizador consiga desempenhar gestos, sempre que um objecto específico seja seleccionado. Agora, quando o utilizador fixa um objecto, a componente aguarda que seja efectuado algum movimento para depois iniciar o reconhecimento. Deste modo, o utilizador já se pode deslocar livremente no ambiente

virtual sem que o sistema tente realizar o reconhecimento de forma indevida.

Contudo, esta componente apresenta algumas limitações. O utilizador não pode desviar o olhar do objecto quando está a executar um movimento, senão é interrompida a captação do movimento, fazendo com que o sistema dê um reconhecimento errado, bem como o utilizador não poder estar em movimento sempre que quiser iniciar um gesto, senão a componente começa a gravar o gesto mal o utilizador fixa o objecto. O utilizador também não pode cancelar um gesto sempre que o inicia e, por fim, sempre que é executado um gesto que não esteja definido pelo sistema, este irá tentar identificá-lo dando uma aproximação de um movimento que esteja definido.

4.3 Pedal

A implementação do pedal inspira-se no facto do pedal ser o objecto mais comum usado pelo pé. A figura 4.5 representa o modelo 3D do pedal criado.

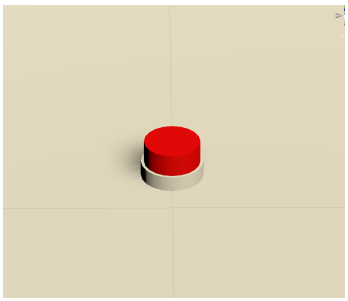


Figura 4.5: Modelo 3D do pedal

Esta componente divide-se em 3 partes: base, *pusher* e barra de intensidade. A base, como o nome indica, é a parte do pedal que serve de apoio. O *pusher* é a parte do pedal que o utilizador interage, exercendo "força", que é traduzida através da distância entre o topo do *pusher* e da base. É nesta parte que estão definidas as propriedades como *treshold*, *deadzone* e a percentagem de força usada para o pedal. Além disso, o pedal apresenta 3 estados de funcionamento, implementados nesta parte. O estado *"Pressed"*, indica que o pedal foi completamente pressionado, o estado *"Released"*, indica que o pedal foi totalmente solto, e o estado *"Between"*, estado intermédio que indica a quantidade de "força" que o utilizador está a exercer. Assim se o programador desactivar o estado *Between*, através da activação da opção *"Use Full Press And Release"*, transformando o pedal num botão, em que este pode ser configurado para exercer uma acção quando o botão é pressionado, estado *Pressed*, ou quando é libertado, estado *Released*.

Por fim, temos a barra de intensidade, que consiste num sistema onde é representada a "força"que o utilizador está a exercer sobre o pedal. Essa representação pode ser das seguintes formas:

- *OnObject* - a barra de intensidade surge por cima do pedal, como é ilustrado na figura 4.6;
- *OnHeadset* - a barra de intensidade surge à frente do utilizador, no ecrã do *headset*, como é ilustrado na figura 4.7;
- *OnController* - a barra de intensidade surge no controlador de mão, como é ilustrado na figura 4.8;
- *None* - aqui não existe barra de intensidade, em vez a cor do *pusher* altera de cor consoante a intensidade exercida, como é ilustrado na figura 4.9;

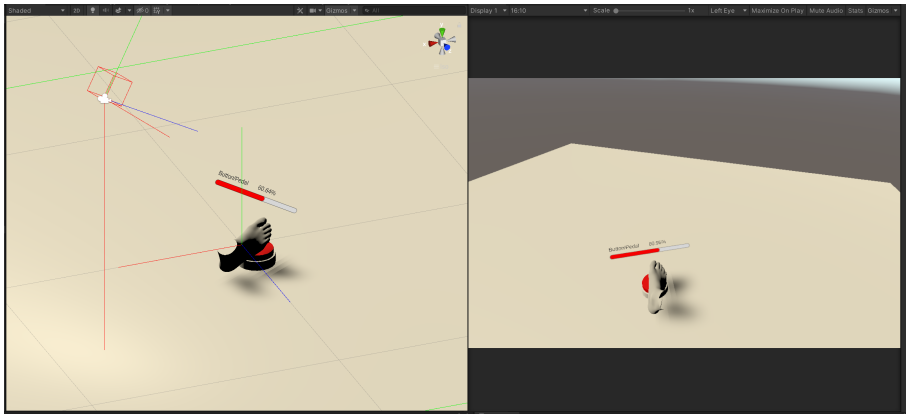


Figura 4.6: Exemplo da representação *OnObject*

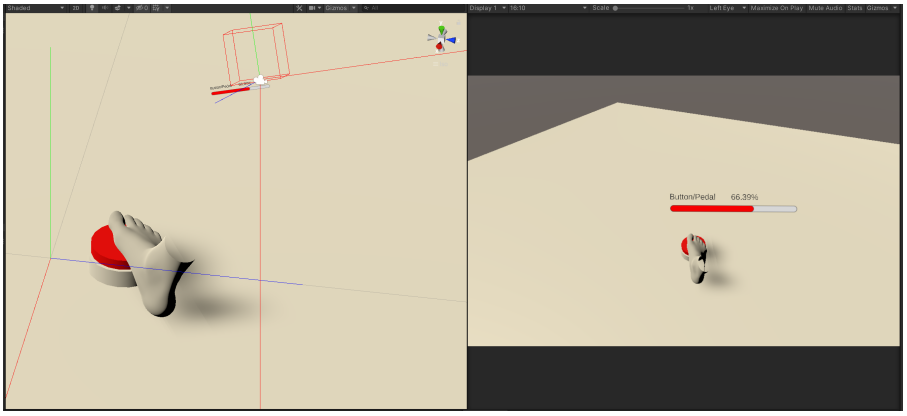


Figura 4.7: Exemplo da representação *OnHeadset*

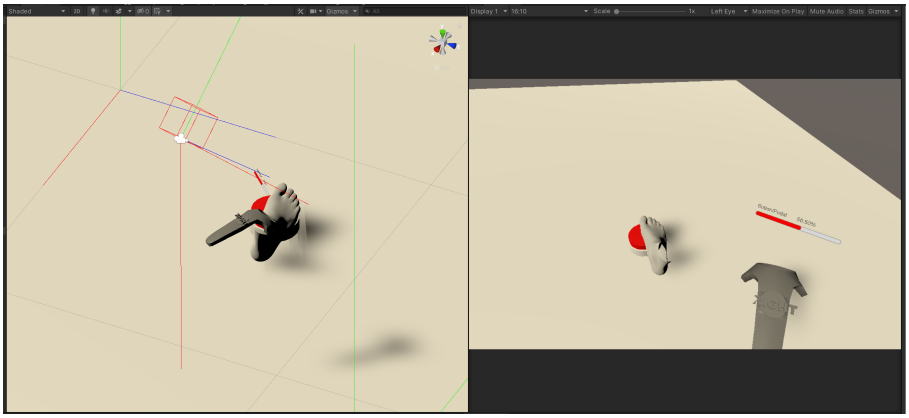


Figura 4.8: Exemplo da representação *OnController*

(a) Pedal não pressionado

(b) Pedal pressionado

Figura 4.9: Exemplo da representação *None*

Todo este desenvolvimento encontra-se no apêndice D

Com o pedal desenvolvido, foi criado um exemplo de interacção que use esta componente. A figura 4.10 demonstra a experiência desenvolvida, que consta na utilização de 4 pedais, cada pedal com um tipo de representação diferente, para o controlo de um *drone*, podendo manipulá-lo tanto na vertical como na horizontal, fazendo-o movimentar pelo ambiente virtual.

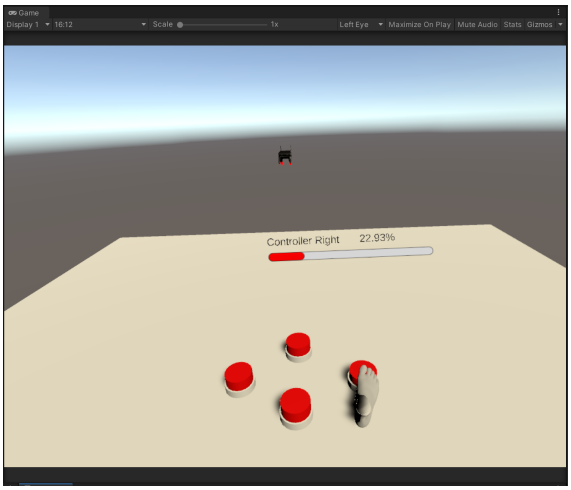


Figura 4.10: Exemplo de controlo de um *drone* com uso da componente pedal

Para o programador usar esta componente basta importar o modelo 3D do pedal, que se encontra em *Assets/Pedal/PedalVR.prefab*, para a sua *scene* virtual e configurá-lo consoante as suas necessidades e preferências.

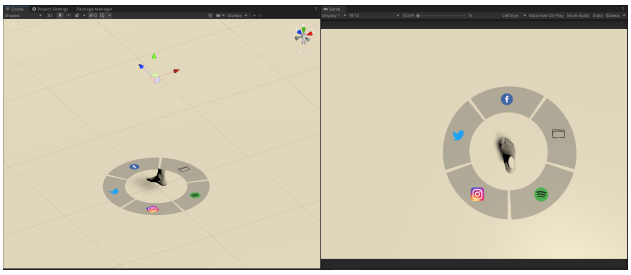
4.4 *FootMenu*

A criação do *FootMenu* foi motivado pelo sistema *Mind The Tap* [1], descrito no capítulo 2. O sistema desenvolvido permite que o utilizador use o pé para fazer a selecção de uma opção, de um conjunto de possibilidades existentes, num menu radial. Para além de o utilizador conseguir realizar uma selecção, a componente tem a particularidade de apresentar vários níveis, funcionando como um conjunto interligado de sub-menus. Isto é possível devido ao facto de os elementos do menu terem a capacidade de conter um outro menu. Deste modo, os utilizadores quando seleccionarem uma opção, podem accionar um

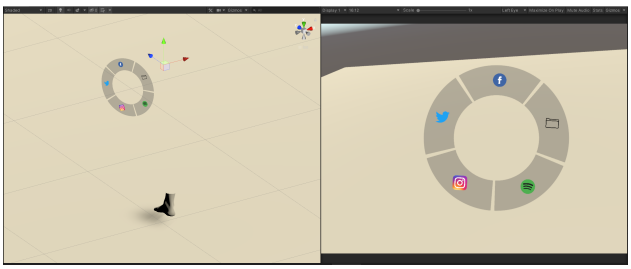
outro menu. Para além disso, o menu tem a possibilidade de ter uma representação 180° ou 360°, como é ilustrado na figura 4.11. O *FootMenu* consegue também fornecer 2 perspectivas ao utilizador, *Direct* e *Indirect*, alterando o modo de interacção com o utilizador, como demonstra a figura 4.12. Na perspectiva *Direct*, as hipóteses são apresentadas no chão em redor ao utilizador, enquanto que no *Indirect* essas hipóteses ficam a "flutuar"à frente dos olhos do utilizador. Dependendo da perspectiva usada, o menu poderá ser activado olhando para o chão, no caso se for *Direct*, ou através de um "chuto", no caso de *Indirect*, tendo também a opção de permanecer sempre activo. Por fim, a componente desenvolvida tem a possibilidade de acompanhar o utilizador quando este se desloca, tornando o menu estático num menu móvel. Para além destas característica, se o *FootMenu* for aplicado no âmbito de visitas virtuais, a componente actua como um sistema de navegação, onde o utilizador faz uma selecção da direcção para onde se deseja movimentar. Noutros casos, esta componente pode ser configurada para, por exemplo, servir como controlador de multimédia, em que o utilizador usa o pé para fazer diversas operações, como *play*, *pause*, *stop* ou *forward*, ou para alterar propriedades de um objecto, como por exemplo, alterar a cor e tamanho.

(a) Menu 180° **(b)** Menu 360°

Figura 4.11: Exemplo de representação dos modos do Menu



(a) *Direct*



(b) *Indirect*

Figura 4.12: Exemplo de representação das perspectivas de utilização

4.4.1 Funcionamento

Para a componente *FootMenu* foram desenvolvidas várias *scripts* com diversos propósitos. Essas *scripts*, encontram-se na directoria *"Assets/FootMenu/Scripts"*, e são compostas por:

- *Dummy.cs* - possui o funcionamento o modo de representação que o menu tem para o utilizador (*Direct* ou *Indirect*).
- *Menu.cs* - possui as propriedades dos menus.
- *MenuElement.cs* - possui as propriedades dos elementos dos menus.
- *MenuCakePiece.cs* - possui a representação gráfica dos elementos do menu.
- *MenuMB.cs* - possui todo o funcionamento de interacção do utilizador com o menu.
- *TriggerCollision.cs* - possui o mecanismo que auxilia pelo *MenuMB.cs* para detecção de selecção de um elemento.
- *HighlighterCollision.cs* - possui o mecanismo de *highlighter* dos elementos.

Atributos

Na fase inicial, foi necessário especificar as características que o sistema irá ter. Para tal, foram desenvolvidas 3 classes que abrangem essas propriedades. A classe *Menu.cs* contém a informação relativa a cada menu, isto é, esta classe identifica que tipo de representação, 180° ou 360°, o menu irá ter bem como a lista de elementos presentes nele. A classe *MenuElement.cs* contém a informação relativa a cada elemento do menu. Essa informação passa por caracterizar o nome, o *icon*, com a opção deste estar voltado para o utilizador, a acção, que o elemento despertará ao ser seleccionado, e a possibilidade de proceder para outro menu. Por fim, temos a classe *MenuCakePiece.cs* que contém a informação gráfica dos elementos, como a representação do *icon* e do quadrante.

A figura 4.13 ilustra estas classes. De notar que, tanto o *Menu.cs* como *MenuElement.cs* são classe do tipo *ScriptableObject*, com o propósito de facilitar a implementação pelos programadores. Desta forma o programador não terá de criar código, apenas terá que criar o objecto na directoria do projecto e, de seguida, configura-lo como é explicado mais abaixo na secção 4.4.2.

```
[CreateAssetMenu(fileName = "Menu 1", menuName = "FootMenu/Menu", order = 1)]
public class Menu : ScriptableObject
{
    public enum Degrees { _180, _360 }
    public Degrees degrees;
    public MenuElement[] Elements;

}
```

(a) Classe *Menu.cs*

```
[CreateAssetMenu(fileName = "MenuElement 1", menuName = "FootMenu/Element", order = 2)]
public class MenuElement : ScriptableObject
{
    public string Name;
    public Sprite Icon;

    public bool IconFacingMiddle = false;

    public UnityEvent Action = new UnityEvent();

    public Menu NextMenu;
}
```

(b) Classe *MenuElement.cs*

```
public class MenuCakePiece : MonoBehaviour
{
    public Image Icon;
    public Image CakePiece;
}
```

(c) Classe MenuCakePiece.cs

Figura 4.13: Classes pertencentes á componente *FootMenu*

Sistema Interno

Com as características do Menu definidas, realizou-se a implementação do funcionamento interno da componente. A classe *MenuMB.cs* abrange todo este funcionamento da componente *FootMenu*, e encontra-se dividido em várias partes com diferentes finalidades.

De forma a detectar a selecção de um elemento, foi desenvolvido um mecanismo de *colliders*, com o propósito de definir as áreas de interacção que cada elemento irá ter. Em *Unity*, os *colliders* [ref–coll] suportam detecção de colisão através das suas "caixas delimitadoras", sendo possivel desta maneira detectar quando o utilizador selecciona uma opção.

Foi desenvolvido o método ***CreateColliders()***, com o objectivo de definir, através de trigonometria, essas áreas de selecção, tendo em atenção o tipo de representação e o número de elementos presentes no menu. O método criado reparte os elementos, em partes iguais, ao longo de uma circunferência, no caso o menu ser do tipo 360°, ou semicircunferência, no caso de ser do tipo 180°. Essa divisão é feita através de um ângulo que irá ser repetido ao longo do menu. Esse ângulo (θ) é calculado através da divisão de π pelo número de elementos do menu, no caso ser do tipo 180°, ou pela divisão de 2π pelo número de elementos, no caso ser do tipo 360°. Com o ângulo (θ) definido, é calculado o comprimento que os *colliders* irão ter através da fórmula: $\sqrt[2]{(1-\cos(\theta))^2+\sin(\theta)^2}$. A figura 4.14 demonstra o conceito por detrás do funcionamento deste método para a repartição de 4 elementos num menu 180°.

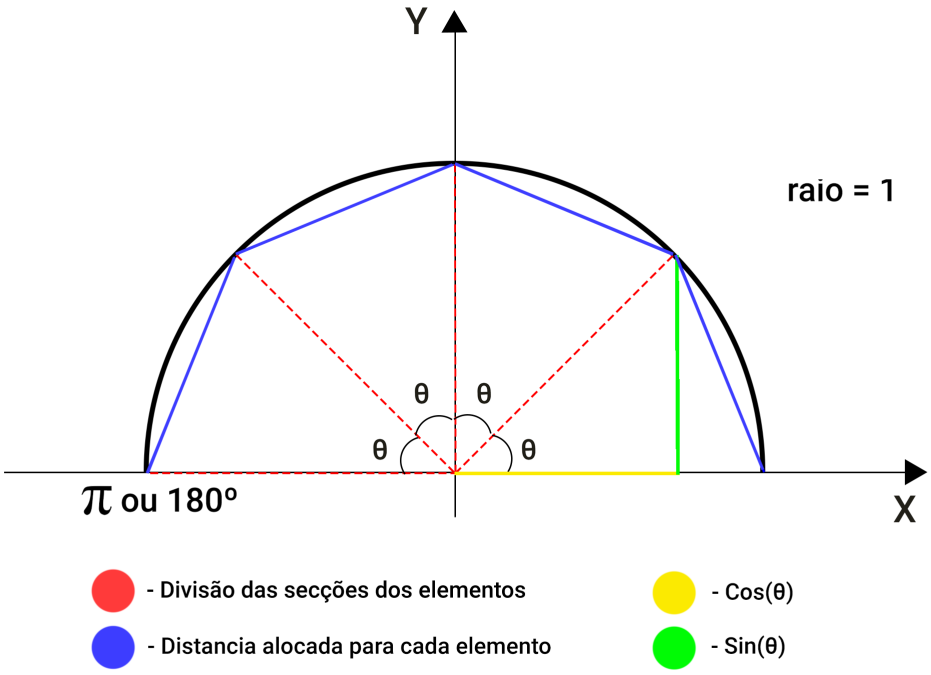
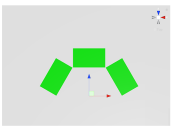
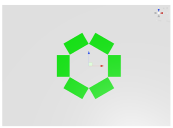
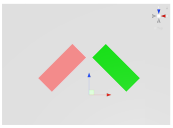
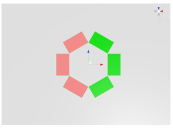
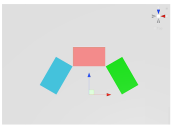
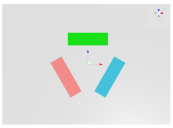
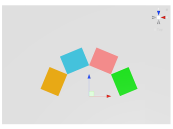
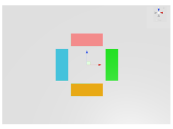
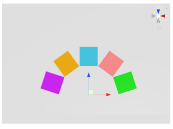
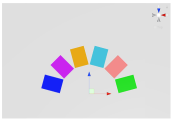
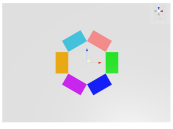


Figura 4.14: Exemplo de divisão da semicircunferência por 4 partes iguais

Com esta informação, o sistema de seguida vai iterativamente percorrer cada elemento, que consta no menu, e estabelecer essas áreas para a selecção. Para tal, a cada elemento, o sistema cria uma caixa para interacção, com o comprimento calculado anteriormente, e posiciona-o no lugar que lhe pertence, através da multiplicação do ângulo (θ), calculado

também anteriormente, pelo número da posição do elemento. Por exemplo, se o menu for 180° e conter 4 elementos, para o primeiro elemento o sistema irá criar uma caixa na posição X=cos(1π/4) e Y=sin(1π/4), enquanto que para o terceiro elemento o sistema irá criar a caixa na posição X=cos(3π/4) e Y=sin(3π/4).

Contudo há excepções que o método realiza. Se o menu for do tipo 180° e conter só 1 elemento, o sistema de *colliders* aborda como se fosse um menu de 3 elementos, só que desta vez as áreas definidas são todas relativas ao único elemento. Se o menu for do tipo 360° e conter só um elemento, o método cria 6 áreas de selecção só que todas estão relacionadas ao único elemento. Se o menu for do tipo 360° e conter 2 elementos, o sistema cria 6 áreas de selecção em que as 3 áreas da direita estão relacionadas com o primeiro elemento e as outras estão relacionados com o segundo elemento. E por fim, se o menu for 360° e tiver mais que 3 elementos, o sistema esquematiza as áreas de selecção consoante a paridade do número de elementos. Sendo par, o método realiza a esquematização começando na direita e preenchendo a circunferência no sentido anti-horário, enquanto que se for ímpar, o sistema esquematiza as áreas iniciando no topo da circunferência, preenchendo também no sentido anti-horário. A Tabela 4.1 ilustra como o sistema define as áreas consoante o numero de elementos existentes.

	Representação	
Num. Elementos	180º	360º
1		
2		
3		
4		
5		
6		
...	...	...

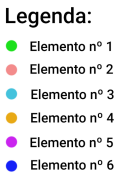


Tabela 4.1: Ilustração de como o sistema de *colliders* divide as secções

Todos os *colliders* criados irão ter associados a *script TriggerCollision.cs* e *HighlitterCollision.cs*. Ambas auxiliam o sistema de detecção de colisão, na medida que detectam a interacção do utilizador com as áreas definidas. Enquanto que *HighlitterCollision.cs* auxilia no realçar do elemento, mostrando ao utilizador que está prestes a seleccionar uma opção, a *TriggerCollision.cs* actua sobre o despertar da acção, executando o evento que está associado à respectiva opção.

Com o mecanismo de interacção desenvolvido, sucedeu-se o desenvolvimento da ilustração do menu, ou seja, representar visualmente a ilustração que o menu irá ter. Foi desenvolvido um método ***InstatiateMenuRepresentation()*** que utiliza a imagem do *icon* e do quadrante e as coloque na posição correta. Este método tem um funcionamento semelhante ao método anterior, na medida que verifica o tipo de menu (180° ou 360°) e o número de elementos presentes no mesmo para o posicionamento das imagens.

Com a interacção e a representação do menu definidas, desenvolveram-se 2 métodos de activação do menu. Estes métodos foram desenvolvidos para que o utilizador não tenha o menu sempre activo, assim o utilizador poderá realizar outras interacções no ambiente virtual sem ter que interagir indevidamente com o menu. Para o menu com o modo de exibição *Direct*, foi criado o método ***InvokeDirectMenuRep()***. Este método conciste em usar o olhar do utilizador para activar o menu. Sempre que o utilizador olhar para baixo, o menu fica activo, podendo ser interagido, caso contrário o menu fica inactivo. Para o menu com o modo de exibição *Indirect*, foi criado o método ***InvokeIndirectMenuRep()***. Este método verifica se o utilizador efectuou um "chuto" para activar o menu. O menu torna-se inactivo quando o utilizador alcança o fim do mesmo, isto é, quando o utilizador seleccionar uma opção e, nessa opção, não seguir outro menu, tendo o utilizador efectuar outra vez o "chuto" para o activar. Estes 2 métodos pode, ser desactivados, se o programador o quiser, tornado o menu sempre activo.

Em seguida, temos a função ***Update()***. Esta função está sempre em execução quando é iniciada execução da componente. Aqui, estão implementadas as operações de rastreamento do utilizador, bem como a verificação de selecção de um elemento. Estão implementadas as operações que permitem que o menu siga o utilizador quando se desloca. Se esta opção estiver activa, a função estará sempre a rastrear a posição do *headset* do utilizador, para depois aplicar essas mudanças sobre o posicionamento do menu. Desta forma, o menu irá deixar de ser estático passando a ser móvel, já que segue o utilizador para onde quer que ele se desloque. Para além disso, a função *Update()* realiza todo o funcionamento quando o elemento é seleccionado. Verifica se o utilizador seleccionou algum elemento e, caso tenha seleccionado, averigua se o presente elemento dá continuação a outro menu, verificando se está associado algum menu ao elemento. Se estiver associado um menu, a função instancia esse novo menu e elimina o antigo. Os *colliders* pertencentes ao antigo menu também são eliminados e são criados os novos com as novas propriedades referentes ao menu.

Por fim, foi desenvolvido, no ficheiro ***Dummy.cs***, as duas perspectivas existentes na componente. Para a *Indirect*, o método aplica o toda a representação gráfica à frente do olhar do utilizador. Nesta perspectiva, o utilizador terá um menu flutuante com a particularidade de este manter a orientação original, ou seja, quando o utilizador mudar a sua orientação, as opções do menu acompanham essa mudança. Para o *Direct*, o método projecta o menu no chão virtual, assim o utilizador conseguirá observar o menu quando olha para o chão.

Com o intuito de facilitar a programação desta componente, foram criados 3 *prefabs* que, como o nome indica, são objectos pré-fabricados que contém as componentes necessárias já implementadas para a utilização. Os *prefabs* criados foram *CakePiece*, que contém todo grafismo necessário do elemento, o *MenuPrefab*, que contém a informação do menu, a

representação dos elementos, neste caso o *prefab* anterior, bem como a opção de especificar o espaçamento que os elementos vão ter entre si, e para finalizar, o *MenuGO*, que contem o ficheiro *Dummy.cs*, onde se específica qual o menu a usar (*MenuPrefab*), que tipo de perspectiva a utilizar e opção de activar o método de acompanhar o utilizador.

4.4.2 Modo de utilização

A utilização desta componente por outro programador é relativamente simples. O programador não tem que escrever código para criar o sistema de menu, mas tem que ter atenção a 3 etapas importantes:

- Criação de Menu(s);
- Criação de Elemento(s) do Menu;
- Configuração do sistema;

Criação de Menu

Para criar um Menu o programador tem que, na directoria do projecto, carregar no botão direito do rato e seleccionar *"Create"*, de seguida *"FootMenu"* e por fim *"Menu"*. Após a conclusão do passo anterior, o programador irá reparar que um objecto foi criado. Com o objecto criado, o programador tem que o seleccionar e, a partir do inspector, especificar o tipo de menu, 180° ou 360°, bem como definir o número de elementos que este menu irá conter. Posteriormente, o programador tem que preencher a lista com os elementos que irão fazer parte do menu.

Para gerar vários menus, o programador tem que repetir os passos anteriores.

Criação de Elemento

Para criar um Elemento, o processo é muito semelhante há de criação de Menu. Primeiro, o programador tem que, na directoria do projecto, carregar no botão direito do rato e seleccionar *"Create"*, de seguida *"FootMenu"* e por fim *"Element"*. Com o passo anterior realizado, o programador irá reparar que um objecto foi criado. Com o objecto criado, o programador tem que o seleccionar e, a partir do inspector, atribuir o nome do elemento e adicionar o *icon*, com a opção de este estar voltado para o utilizador ou não. Além disso, é necessário também que o programador defina quais os eventos que este elemento irá executar e que especifique qual o menu que irá suceder após a selecção deste elemento, podendo estas propriedades estarem vazias.

Para gerar vários elementos, o programador tem que repetir os passos anteriores.

Configuração do Sistema

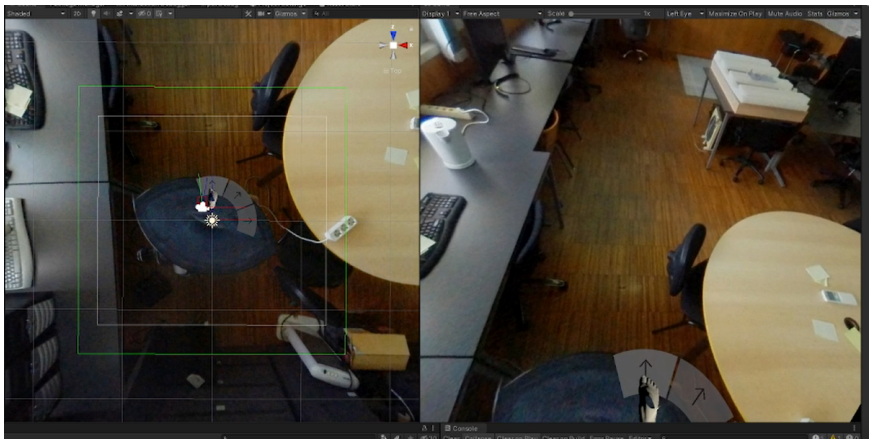
Com o Menu(s) e os Elementos criados, é necessário que o programador configure o sistema de forma a conseguir executar a componente. Para tal, o programador tem que aceder ao objecto *"Assets/FootMenu/Prefabs/MenuPrefab"*. Quando seleccionado, tem que definir, na secção *"Data"*, que se encontra no inspector no segmento *"Menu MB(Script)"*, qual irá ser o Menu principal que será apresentado na execução, tendo também a opção de indicar qual o espaçamento que os elementos vão ter entre si (*"Gap Width Degree"*). Após a definição do menu principal, é necessário aceder ao objecto *"Assets/FootMenu/Prefabs/MenuPrefab/MenuGO"* e "arrastá-lo" para a *scene* do projecto.

Por fim, o programador tem que seleccionar esse objecto e, no inspector, adicionar na secção *"Main Menu Prefab"* o objecto que foi configurado inicialmente (*"Assets/FootMenu/Prefabs/MenuPrefab"*). Após isto, o programador só necessita de definir qual o tipo de representação, (*"Direct"* ou *"Indirect"*), e indicar se o menu acompanha, ou não, o utilizador no ambiente virtual (*"Menu Following Player"*).

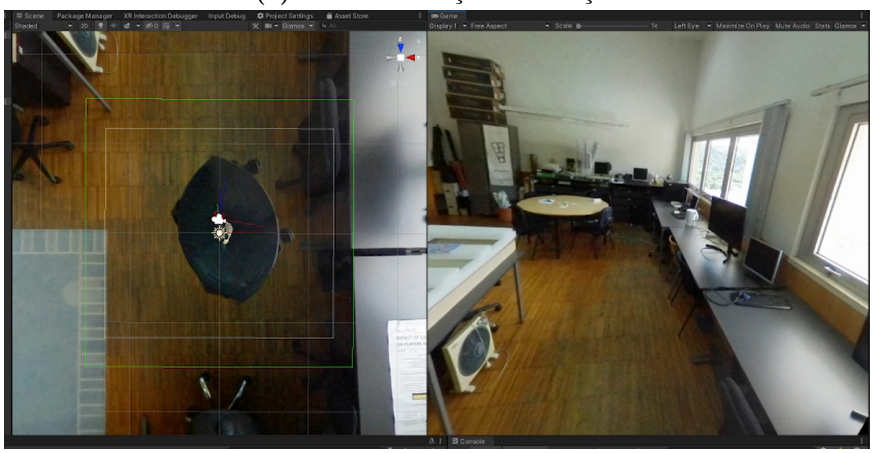
Com estes etapas realizadas, o programador irá ter a componente *FootMenu* preparada para utilização. No apêndice E está toda documentação que o programador tem acesso para a utilização da componente. Na documentação, está descrito as características do *FootMenu*, bem como as etapas para criação de cada objecto. Para além disso, também está ilustrado a criação de um menu com vários elementos.

4.4.3 Visita virtual 360°

Nesta secção, é demonstrado a criação de uma visita virtual 360° com interacção feita pelo pé. Para esta demonstração, foi realizado um cenário em que o utilizador utiliza apenas o pé para fazer a selecção, ao contrário do método tradicional de usar o controlador. Foi utilizado a componente *FootMenu*, criada anteriormente, para conceber um sistema de navegação. Esta visita decorre no Departamento de Engenharia Informática (DEI), mais precisamente desde o corredor do piso 6 da torre G até à sala G6.4, onde foi desenvolvido todo o projecto. A imagem 4.15 exemplifica uma passagem da visita criada.



(a) Antes da selecção da direcção



(b) Depois da selecção da direcção

Figura 4.15: Detalhe da visita virtual criada

Para a construção desta visita foi fotografado o espaço em questão, em 7 pontos distintos, recorrendo a fotografias 360°. As fotografias encontram-se no apêndice F. Foi montado um sistema de navegação pelo espaço, onde a cada um dos 7 pontos foi associado um menu cujas opções de selecção são representadas pelas direcções que o utilizador pode tomar, como se pode ver na figura 4.16. De notar que na figura, mencionada anteriormente, o menu 8 é um menu para a selecção de objectos 3D. Deste modo, o utilizador ao seleccionar uma opção vai despoletar uma acção, seja ela de movimento ou de interacção com o espaço. Todo este sistema de menus é fixo em relação ao ambiente virtual, isto é, não acompanha o utilizador quando este se desloca. Isto deve-se ao facto de serem usadas fotografias 360°, não podendo o utilizador deslocar-se pelo ambiente virtual.

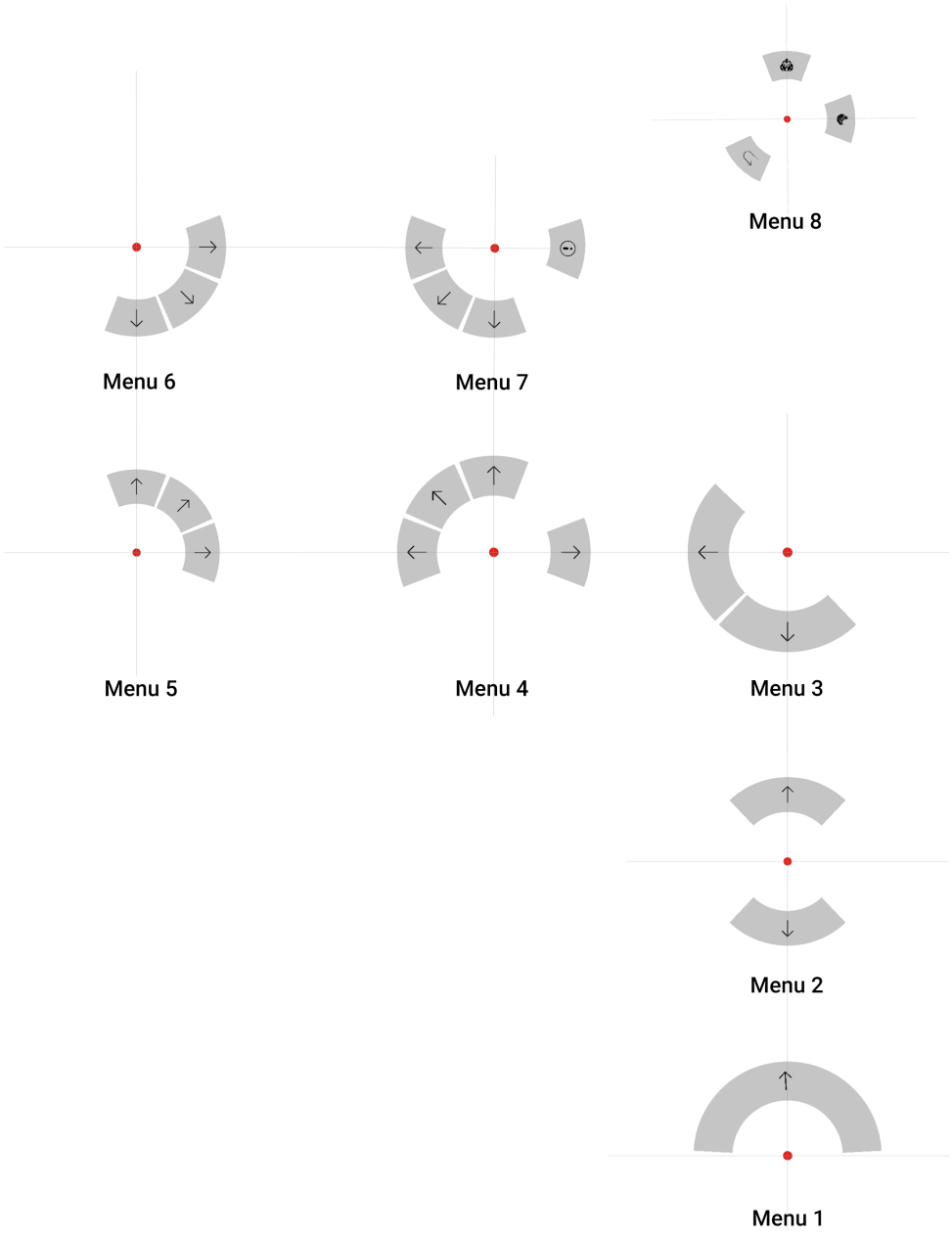


Figura 4.16: Esquema do sistema de navegação da visita

Foram criados um total de 8 menus e 22 elementos, sendo que estes últimos têm as acções associadas a si. Acções essas, que se tratam da substituição do *skybox* da cena virtual por uma das fotografias tiradas, com excepção do menu 8 onde são adicionados 2 objectos virtuais, mais propriamente o *heaset* e *tracker* usados no projecto. A figura 4.17 ilustra todos estes materiais.

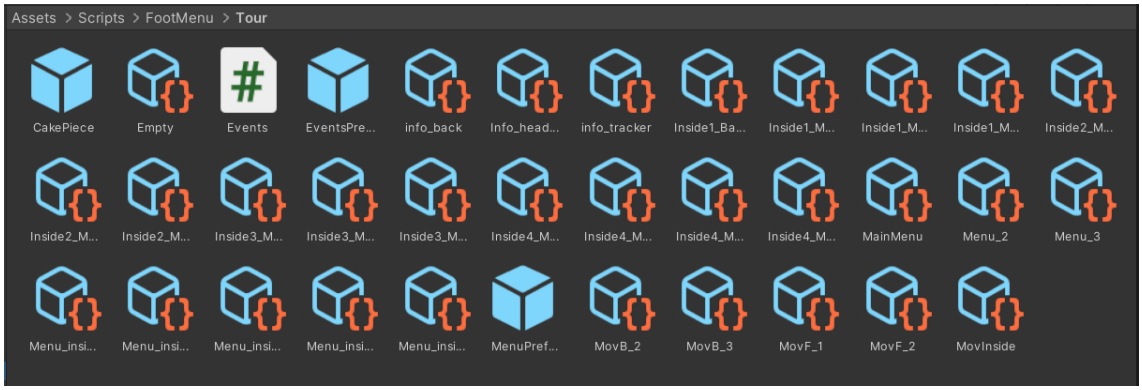


Figura 4.17: Objectos criados para a criação da visita virtual 360°

Com isto torna-se possível criar um percurso de uma visita interactiva em RV recorrendo somente à utilização do pé para controlo da mesma. Desta forma, o utilizador fica com as mãos livres para outro tipo de interacção com o espaço, onde o uso de controladores manuais pode permitir a manipulação de objectos.

Capítulo 5

Avaliação

O presente capítulo enuncia as avaliações realizadas sobre o que foi desenvolvido. Foi avaliada a usabilidade e experiência do utilizador sobre a visita virtual, como também a usabilidade da API *FootMenu*. O primeiro teste teve como finalidade avaliar as funcionalidades implementadas de um ponto de vista do utilizador, enquanto que o outro teste foi destinado a avaliar a componente de um ponto de vista da programação.

5.1 Avaliação de usabilidade da visita virtual 360°

O conjunto de testes realizados teve como objectivo avaliar as funcionalidades implementadas da componente do ponto de vista da usabilidade e da experiência do utilizador. Foram recrutados um total de 10 participantes, dos quais 6 eram do sexo feminino e 4 do sexo masculino, com idades compreendidas entre os 20 e os 25 anos. Deste conjunto de participantes, 4 já tinham tido contacto prévio com RV, enquanto que os restantes 6 não. O recrutamento foi realizado por contacto directo e pela divulgação via email, em que os interessados preenchiam um *doodle* de participação consoante a sua disponibilidade.

A experiência consistiu em utilizar o cenário referido na secção 4.4.3, onde os participantes realizaram um conjunto de tarefas nos 2 modos de utilização. Os participantes ímpares testaram em primeiro o modo *Direct* e, em seguida, o modo *Indirect*, enquanto que os participantes pares testaram o primeiro o modo *Indirect* e depois o *Direct*.

Procedimento

Ao chegar ao laboratório, os participantes foram informados acerca do conceito e propósito da experiência em questão. Neste ponto, foi pedido a cada participante que assinasse o documento de consentimento informado (apêndice G), que clarifica o uso dos dados recolhidos e os seus direitos enquanto participante deste estudo. Foi também explicado que o participante iria executar 2 testes, nos 2 modos existentes, e que para cada um destes iria realizar um conjunto de tarefas. Antes dos testes propriamente ditos, foi dado um breve período de experimentação para que o participante se acostumasse à componente, onde o participante "anda" livremente no cenário de teste.

Após este período, deu-se início à fase de testes, onde o participante recebia as instruções para realizar as tarefas. Durante este período, foi recolhido vídeo para avaliação da interacção do participante com o cenário de teste. No fim de cada teste, cada participante preencheu um questionário.

Este processo foi repetido para cada um dos modos de utilização.

Tarefas

Após a experimentação do cenário, é pedido aos participantes que realizem um conjunto de tarefas. As tarefas consistem em seleccionar a opção correcta do conjunto de hipóteses existentes no menu. As tarefas que foram pedidas aos participantes encontram-se descritas na tabela abaixo (tabela 5.1).

Tarefa 1	*Andar em frente, em direcção à porta exterior do corredor*
Tarefa 2	*Andar em frente, em direcção à porta exterior do corredor*
Tarefa 3	*Entrar na sala*
Tarefa 4	*Ir para o ponto atrás da mesa redonda*
Tarefa 5	*Ir para o ponto junto à janela*
Tarefa 6	*Ir para o espaço VR*
Tarefa 7	*Visualizar o objecto VIVE e Tracker*
Tarefa 8	*Estando virado para a porta da sala, ir para o ponto junto à janela*
Tarefa 9	*Estando virado para o espaço VR, ir para o ponto atrás da mesa redonda*
Tarefa 10	*Estando virado para o espaço VR, sair da sala*
Tarefa 11	*Estando virado para a porta da sala, ir para o meio do corredor*
Tarefa 12	*Estando virado para a porta com um letreiro, ir para o inicio do corredor*
Tarefa 13	*Estando virado para o WC das mulheres, ir para o meio do corredor*
Tarefa 14	*Estando virado para o inicio do corredor, ir para trás*
Tarefa 15	*Estando de costas para a porta do laboratório, entrar na sala*
Tarefa 16	*Estando de frente para a porta, ir para trás*

Tabela 5.1: Tabela de tarefas a realizar no teste da componente

O participante só realiza a tarefa seguinte quando der por terminado a tarefa anterior. As presentes tarefas foram definidas de forma a que o participante realize um conjunto de movimento com diferentes dificuldades. Desta forma, até à tarefa 7, o participante pode realizar a selecção como achar mais confortável, mas, a partir da tarefa 8, são exigidos as seguintes restrições ao movimento:

- da tarefa 8 à 10, é esperado que o participante seleccione a opção que se encontra à sua direita, sem rodar o corpo.
- da tarefa 11 à 13, é esperado que o participante seleccione a opção à sua esquerda, sem rodar o corpo.
- da tarefa 14 à 16 é esperado que o participante seleccione a opção à sua retaguarda, sem rodar o corpo.

Para efeitos de avaliação da usabilidade, foram registados a quantidade de erros cometidos e o tempo de execução de cada tarefa. Considerou-se como erro toda a acção que não estava prevista para na execução de cada uma das tarefas, por exemplo, seleccionar duas opções ao mesmo tempo, ultrapassar o limite do elemento fazendo com que este seja seleccionado duas vezes, entre outros.

Questionário

Efectuada as tarefas, foi pedido ao participante que preenchesse um formulário *online*, que se divide em 4 partes. A primeira parte corresponde ao questionário *Simulator Sickness* [8]. Este questionário abrange perguntas sobre o desconforto que a pessoa experienciou durante a execução do teste. A segunda parte corresponde ao questionário *System Usability Scale (SUS)* [62], que conciste num questionário com 10 perguntas, na qual o participante tem 5 opções de resposta, desde *"Strongly agree"* até *"Strongly disagree"*. A terceira parte está reservada para o *NASA TLX (Task Load Index)* [63]. Este questionário mede, numa escala de 7 pontos, o esforçodo participante na realização das tarefas. A última parte corresponde ao *Product Reaction Cards* em que o participante escolhe 5 palavras que associa à experiência que acabou de ter..

No apêndice H, está todo o questionário que os participantes tiveram que preencher.

Resultados

Os resultados obtidos são divididos em duas partes: a parte de utilização do modo *Direct* e a outra do modo *Indirect*. Os resultados são apresentados de forma separada para cada um dos modos, começando com o *Direct* e de seguida o *Indirect*.

A tabela 5.2 mostra os tempos, em segundos, de execução de cada tarefa por cada participante no modo ***Direct***. Reportou-se uma média de 61.6 segundos para conclusão todas as tarefas, sendo que a tarefa 2 foi a que se realizou mais rapidamente (média de 2.3 segundos) e a tarefa 7 a que demorou mais tempo (média de 5.7 segundos).

	Tarefa 1	*Tarefa 2*	*Tarefa 3*	*Tarefa 4*	*Tarefa 5*	*Tarefa 6*	*Tarefa 7*	*Tarefa 8*	*Tarefa 9*	*Tarefa 10*	*Tarefa 11*	*Tarefa 12*	*Tarefa 13*	*Tarefa 14*	*Tarefa 15*	*Tarefa 16*	*Total*
Participante nº1	4	2	2	4	2	2	5	2	3	4	10	14	11	4	7	3	79
Participante nº2	8	2	5	3	4	12	8	6	4	6	4	7	4	3	3	2	81
Participante nº3	2	2	4	3	3	3	7	4	2	4	3	5	3	4	2	3	54
Participante nº4	2	2	3	8	4	6	6	5	4	9	4	4	3	3	6	4	73
Participante nº5	2	2	3	3	4	2	3	4	5	4	2	4	2	3	2	3	48
Participante nº6	2	4	2	3	3	1	5	6	2	5	3	3	5	3	3	4	54
Participante nº7	2	2	5	2	5	4	6	6	2	5	3	3	2	3	2	2	54
Participante nº8	2	3	1	3	2	10	7	13	10	5	3	2	2	8	9	3	83
Participante nº9	2	2	3	3	2	3	5	4	3	4	2	3	2	2	2	2	44
Participante nº10	2	2	3	3	3	3	5	4	3	4	3	3	2	2	2	2	46
Média	2.8	2.3	3.1	3.5	3.2	4.6	5.7	5.4	3.8	5	3.7	4.8	3.6	3.5	3.8	2.8	61.6

Tabela 5.2: Tempo de realização das tarefas (em segundos) para modo *Direct*

A tabela 5.3 ilustra os erros cometidos pelos participantes para as várias tarefas. Das 16 tarefas, apenas 3 foram concluídas sem qualquer erro por parte dos participantes. Foi registado um total de 27 erros, sendo os mais frequentes o falhar em activar o menu (não olhar para baixo) e seleccionar duas opções (passa o limite da zona de selecção e ao recolher o pé selecciona novamente).

	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5	Tarefa 6	Tarefa 7	Tarefa 8	Tarefa 9	Tarefa 10	Tarefa 11	Tarefa 12	Tarefa 13	Tarefa 14	Tarefa 15	Tarefa 16	Total erros
Participante nº 1																	4
Participante nº 2																	7
Participante nº 3																	3
Participante nº 4																	3
Participante nº 5																	0
Participante nº 6																	2
Participante nº 7																	1
Participante nº 8																	7
Participante nº 9																	0
Participante nº 10																	0
Total erros	2	1	3	3	1	3	0	2	1	2	0	3	2	1	3	0	27

Tabela 5.3: Quantidade de erros cometidos no modo *Direct*

Relativamente aos resultados obtidos com o questionário, verificou-se que os *scores* totais do *Simulator Sickness* variaram entre 0 e 44.88, com média de 15.334 Os *scores* totais foram calculados de acordo com a fórmula apresentada na tabela 5.4. A tabela 5.5 mostra os valores obtidos para todas as sub-dimensões. No apêndice I encontram-se as respostas dos participantes ao questionário *Simulator Sickness*.

Computation of SSQ Scores

	Weight		
SSQ Symptom[a]	*N*	*O*	*D*
General discomfort	1	1	
Fatigue		1	
Headache		1	
Eyestrain		1	
Difficulty focusing		1	1
Increased salivation	1		
Sweating	1		
Nausea	1		1
Difficulty concentrating	1	1	
Fullness of head			1
Blurred vision		1	1
Dizzy (eyes open)			1
Dizzy (eyes closed)			1
Vertigo			1
Stomach awareness	1		
Burping	1		
Total[b]	[1]	[2]	[3]

Score
N = [1] × 9.54
O = [2] × 7.58
D = [3] × 13.92
TS[c] = [1] + [2] + [3] × 3.74

[a]Scored 0, 1, 2, 3. [b]Sum obtained by adding symptom scores. Omitted scores are zero. [c]Total Score.

Tabela 5.4: Fórmula para o cálculo do Simulator Sickness [8]

	Nausea	Oculomotor	Disorientation	Total
Participante nº1	9.54	19.08	41.76	26.18
Participante nº2	19.08	9.54	55.68	44.88
Participante nº3	9.54	9.54	27.84	14.96
Participante nº4	0	0	27.84	14.96
Participante nº5	0	0	0	3.74
Participante nº6	0	0	0	0
Participante nº7	0	0	13.92	11.22
Participante nº8	9.54	38.16	13.92	11.22
Participante nº9	9.54	9.54	27.84	14.96
Participante nº10	9.54	9.54	13.92	11.22

Tabela 5.5: Resultados obtidos para Simulator Sickness no modo *Direct*

De acordo com o questionário de usabilidade, verificou-se uma média de *scores* de usabilidade de 83.5, encontrando-se as pontuações entre 70 e 95. Os resultados foram calculados de acordo com Lewis [62]. A tabela 5.6 ilustra os resultados individuais do questionário de usabilidade.

	I think that I would like to use this system frequently	I found the system unnecessarily complex	I thought the system was easy to use	I think that I would need the support of a technical person to be able to use this system	I found the various functions in this system were well integrated	I thought there was too much inconsistency in this system	I would imagine that most people would learn to use this system very quickly	I found the system very cumbersome to use	I felt very confident using the system	I needed to learn a lot of things before I could get going with this system	Scores Totais
Participante nº1	4	1	4	3	5	1	5	3	5	2	82.5
Participante nº2	4	1	4	3	4	1	4	2	4	5	70
Participante nº3	4	2	5	4	5	1	5	1	4	2	82.5
Participante nº4	4	1	5	2	5	1	4	1	4	1	90
Participante nº5	3	1	4	2	4	1	5	1	4	1	85
Participante nº6	4	1	5	3	4	1	4	1	4	2	82.5
Participante nº7	5	1	5	3	5	1	5	1	5	1	95
Participante nº8	3	1	5	1	5	3	5	2	4	1	85
Participante nº9	4	1	4	4	5	1	3	1	4	1	80
Participante nº10	4	1	4	4	5	1	4	1	4	1	82.5
											83.5

Tabela 5.6: *Scores* de usabilidade no modo *Direct*

Segundo o questionário NASA TLX, constatou-se um *workload* ponderado de 23, tendo sido obtidos valores compreendidos entre 18 e 65, como se pode verificar na tabela 5.7. Os valores totais ponderados da carga de trabalho foram calculados de acordo com o método proposto por Hart e Staveland [63], tendo sido usado como pesos os seguintes valores:

- *Mental Demand*: 5
- *Physical Demand*: 2
- *Temporal Demand*: 1
- *Performance*: 3
- *Effort*: 0
- *Frustration*: 4

	Mental Demand	Physical Demand	Temporal Demand	Performance	Effort	Frustration	Total
Participante nº1	7	7	3	3	5	1	65
Participante nº2	4	2	2	3	4	2	43
Participante nº3	4	1	1	2	2	1	33
Participante nº4	1	1	1	2	2	1	18
Participante nº5	2	2	3	2	3	3	35
Participante nº6	2	2	2	3	3	1	29
Participante nº7	2	2	1	2	2	1	25
Participante nº8	3	4	2	3	3	4	50
Participante nº9	2	1	1	3	2	1	26
Participante nº10	2	1	1	2	2	1	23
							23

Tabela 5.7: *Workload* ponderado para o modo *Direct*

Quanto às palavras utilizadas para descrever a componente, aquela que foi mais frequente foi *"Innovative"*. O gráfico 5.1 mostra a distribuição das palavras seleccionadas pelos participantes.

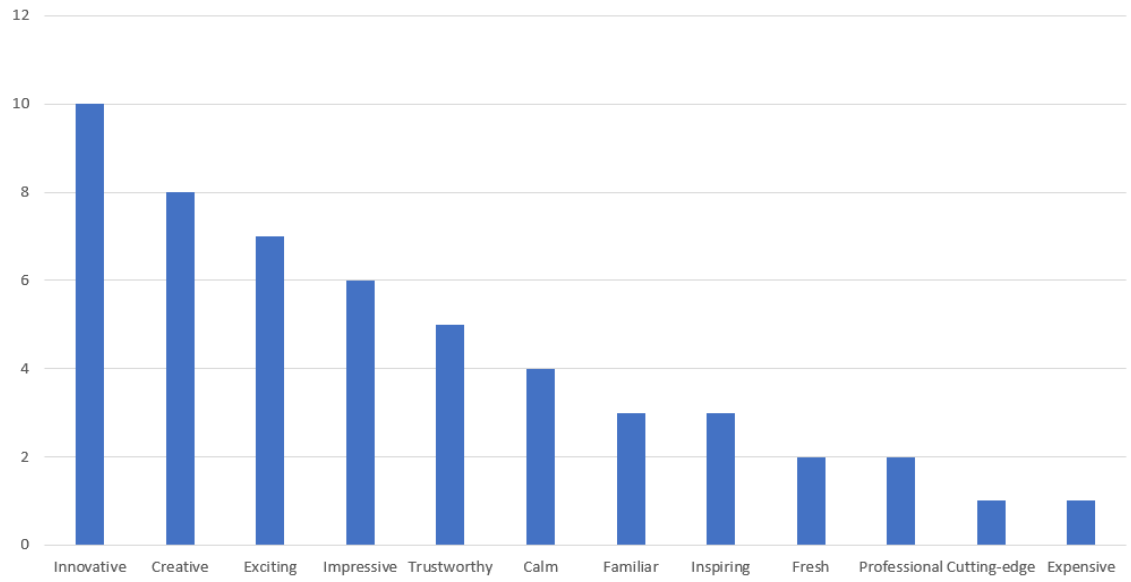


Figura 5.1: Gráfico de frequências das palavras escolhidas pelos participantes para o modo *Direct*

Tendo apresentado os resultados do modo *Direct*, segue-se a exposição dos resultados do modo ***Indirect***.

A tabela 5.8 mostra os tempos, em segundos, de execução de cada tarefa por cada participante no modo ***Indirect***. Reportou-se uma média de 57.6 segundos para conclusão todas as tarefas, sendo que a tarefa 2 foi a que se realizou mais rapidamente (média de 1.4 segundos) e a tarefa 7 a que demorou mais tempo (média de 6.7 segundos).

	Tarefa 1	*Tarefa 2*	*Tarefa 3*	*Tarefa 4*	*Tarefa 5*	*Tarefa 6*	*Tarefa 7*	*Tarefa 8*	*Tarefa 9*	*Tarefa 10*	*Tarefa 11*	*Tarefa 12*	*Tarefa 13*	*Tarefa 14*	*Tarefa 15*	*Tarefa 16*	*Total*
Participante nº1	7	2	5	2	12	10	3	5	1	4	2	2	3	2	2	1	63
Participante nº2	3	1	3	2	3	11	7	2	16	10	4	3	1	2	2	1	71
Participante nº3	3	1	2	2	8	3	5	6	5	8	2	3	10	2	3	3	66
Participante nº4	4	3	7	3	5	13	4	5	2	4	4	2	4	3	2	2	67
Participante nº5	3	1	1	1	1	2	6	7	1	4	2	2	1	2	1	2	37
Participante nº6	2	2	4	2	2	9	3	4	2	5	3	3	3	2	1	3	50
Participante nº7	3	1	3	2	2	5	9	5	2	5	2	2	1	2	2	2	48
Participante nº8	3	1	5	3	3	6	6	17	4	5	2	2	3	5	1	3	69
Participante nº9	2	1	5	2	3	2	18	10	2	6	2	2	2	2	3	3	65
Participante nº10	3	1	1	1	1	2	6	4	2	4	7	2	1	2	1	2	40
Média	3.3	1.4	3.6	2	4	6.3	6.7	6.5	3.7	5.5	3	2.3	2.9	2.4	1.8	2.2	57.6

Tabela 5.8: Tempo de realização das tarefas (em segundos) para modo *Indirect*

A tabela 5.9 ilustra os erros cometidos pelos participantes para as várias tarefas. Das 16 tarefas, 5 foram concluídas sem qualquer erro por parte dos participantes. Foi registado um total de 29 erros, sendo as mais frequentes a selecção de duas opções (passa o limite da zona de selecção e ao recolher o pé selecciona novamente) e fazer *swipe* sobre as opções ("varrer" com pé as várias opções, seleccionando todas por onde passa). De realçar que a tarefa 3 apresenta mais erros que as restantes.

	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5	Tarefa 6	Tarefa 7	Tarefa 8	Tarefa 9	Tarefa 10	Tarefa 11	Tarefa 12	Tarefa 13	Tarefa 14	Tarefa 15	Tarefa 16	Total erros
Participante nº 1																	4
Participante nº 2																	4
Participante nº 3																	4
Participante nº 4																	3
Participante nº 5																	1
Participante nº 6																	4
Participante nº 7																	3
Participante nº 8																	2
Participante nº 9																	4
Participante nº 10																	0
Total erros	2	0	7	0	2	4	2	2	3	2	1	0	3	1	0	0	29

Tabela 5.9: Quantidade de erros cometidos no modo *Indirect*

Relativamente aos resultados obtidos com o questionário, verificou-se que os *scores* totais do *Simulator Sickness* variaram entre 0 e 59.84, com média de 17.578. A tabela 5.10 mostra os valores obtidos para todas as sub-dimensões.

	Nausea	Oculomotor	Disorientation	Total
Participante nº1	19.08	22.74	41.76	29.92
Participante nº2	9.54	7.58	13.92	11.22
Participante nº3	9.54	0	13.92	7.48
Participante nº4	0	15.16	27.84	14.96
Participante nº5	0	0	0	0
Participante nº6	0	0	0	0
Participante nº7	0	22.74	27.84	18.7
Participante nº8	38.16	53.06	69.6	59.84
Participante nº9	9.54	15.16	41.76	22.44
Participante nº10	9.54	7.58	13.92	11.22

Tabela 5.10: Resultados obtidos para Simulator Sickness

De acordo com o questionário de usabilidade, verificou-se uma média de *scores* de usabilidade de 83.5, encontrando-se as pontuações entre 70 e 95. Os resultados foram calculados de acordo com Lewis [62]. A tabela 5.11 ilustra os resultados individuais do questionário de usabilidade.

	I think that I would like to use this system frequently	I found the system unnecessarily complex	I thought the system was easy to use	I think that I would need the support of a technical person to be able to use this system	I found the various functions in this system were well integrated	I thought there was too much inconsistency in this system	I would imagine that most people would learn to use this system very quickly	I found the system very cumbersome to use	I felt very confident using the system	I needed to learn a lot of things before I could get going with this system	Scores Totais
Participante nº1	5	2	4	3	4	3	5	2	5	1	80
Participante nº2	4	1	4	4	4	1	3	2	5	2	75
Participante nº3	4	2	4	4	3	1	2	1	4	2	67.5
Participante nº4	5	1	4	1	5	1	4	1	4	1	92.5
Participante nº5	3	2	3	1	5	1	5	1	3	1	82.5
Participante nº6	4	3	3	3	4	3	3	3	3	2	57.5
Participante nº7	5	1	5	3	5	1	4	2	4	1	87.5
Participante nº8	2	2	3	3	4	3	2	4	2	3	45
Participante nº9	2	3	3	4	5	1	2	4	3	3	50
Participante nº10	4	1	4	4	5	1	4	1	4	1	82.5
											72

Tabela 5.11: *Scores* de usabilidade no modo *Indirect*

Segundo o questionário NASA TLX, constatou-se um *workload* ponderado de 38, tendo sido obtidos valores compreendidos entre 24 e 89, como se pode verificar na tabela 5.12. Os valores totais ponderados da carga de trabalho foram calculados de acordo com o método proposto por Hart e Staveland [63], tendo sido usado como pesos os seguintes valores:

- *Mental Demand*: 4
- *Physical Demand*: 2.5
- *Temporal Demand*: 0
- *Performance*: 5
- *Effort*: 1
- *Frustration*: 2.5

	Mental Demand	Physical Demand	Temporal Demand	Performance	Effort	Frustration	Total
Participante nº1	6	7	3	3	6	1	65
Participante nº2	3	3	3	6	2	2	56.5
Participante nº3	4	3	2	8	2	2	70.5
Participante nº4	2	1	1	8	2	2	57.5
Participante nº5	3	5	2	8	4	2	73.5
Participante nº6	3	3	3	4	5	4	54.5
Participante nº7	2	1	1	2	1	1	24
Participante nº8	7	7	4	5	6	5	89
Participante nº9	3	2	1	5	3	4	55
Participante nº10	2	1	1	2	2	1	25
							38

Tabela 5.12: *Workload* ponderado para o modo *Indirect*

Quanto às palavras utilizadas para descrever a componente, aquela que foi mais frequente foi *"Creative"*. O gráfico 5.2 mostra a distribuição das palavras seleccionadas pelos participantes.

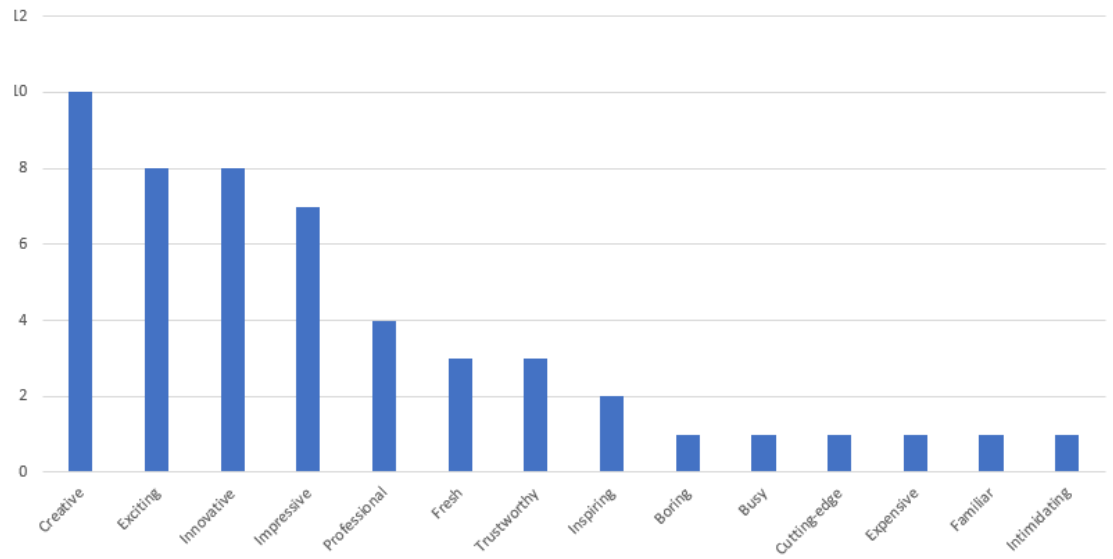


Figura 5.2: Gráfico de frequências das palavras escolhidas pelos participantes para o modo *Indirect*

Discussão

No que diz respeito aos tempos de execução das tarefas, verificou-se que o modo *Direct* tem, em média, um tempo de execução superior ao *Indirect*. Isto pode dever-se ao facto de, no modo *Direct*, se ter que realizar uma acção extra, isto é, olhar para o chão saber quais as opções que pode escolher, enquanto que no modo *Indirect* estas aparecem à sua frente. Em ambos os modos, verificou-se que as tarefas 7 e 8 são as que tem um tempo médio de execução mais elevado. Isto pode dever-se a que na tarefa 7 são exigidas 3 acções (seleccionar a opção de informação, seleccionar o *tracker* e seleccionar o *headset*), enquanto que noutras tarefas apenas é exigido que se seleccione a direcção a tomar. Já na tarefa 8 é introduzido algum grau de desconforto nas acções dos participantes, o que pode fazer com que estes demorem mais algum tempo a assimilar a instrução.

Quanto aos erros cometidos pelos participantes, verificou-se que o modo *Indirect* apresenta maior número de erros, ainda que muito próximo do modo *Direct*. No *Direct* cada tarefa teve no máximo 3 erros, enquanto que no outro modo uma das tarefas teve 7 erros dos 10 participantes. Isto ocorreu na tarefa 3 em que era pedido que o participante entrasse na sala. Uma vez que no modo *Indirect* não existia o chão como referência, os participantes tinham que rodar e fazer a selecção, o que resultava numa maior amplitude do movimento e, consequentemente, ultrapassavam o limite do elemento, activando a opção duas vezes. Isto está de acordo com os tipos de erros mais frequentes, nomeadamente o de seleccionar duas opções por ultrapassar os limites do elemento. Estes resultados podem indicar a necessidade de uma melhoria ao nível da área de selecção dos vários elementos.

Passando à discussão dos dados obtidos nos questionários, começando pelo *Simulator Sickness*, convém realçar que não existe existem limiares definidos na literatura para uma experiência mais, ou menos, desconfortável, sendo que os valores obtidos por Kennedy et al. [8] referem-se especificamente a aviadores, não se aplicando directamente ao actual estudo. No entanto, os *Total Scores* obtidos não estão muito distantes dos observados por Kennedy et al. [8]. Apesar da média do *Total Score* do modo *Indirect* ser superior à do modo *Direct*, a mediana de ambos é igual (mediana = 13.09). Em relação às 3 sub dimensões, estas também apresentaram pontuações superiores no modo *Indirect*. No entanto, verificou-se que as diferenças entre ambos os modos são superiores, em termos absolutos e relativos, nos efeitos oculomotores, sendo que ao nível da desorientação a diferença não é muito elevada entre os 2 modos.

Relativamente ao questionário sobre a usabilidade, verificou-se que os *scores* totais foram superiores no modo *Direct* face ao *Indirect*. Isto pode ser devido a que no *Direct* a selecção das opções seja feita de forma mais precisa, o que pode levar a um entendimento de melhor usabilidade. De acordo com a tabela de Sauro-Lewis CGS [62], o modo *Direct* encontra-se na categoria "A", enquanto que o modo *Indirect* encontra-se na categoria "C+".

Quanto aos resultados do questionário sobre o *workload*, verificou-se uma maior sobrecarga no modo *Indirect* face ao *Direct*. Contudo, os valores obtidos de sobrecarga em ambos os modos encontram-se no primeiro quartil da distribuição de frequências de outros estudos realizados em tarefas de comando e controlo [64]. Mais ainda verificou-se que no modo *Direct* as dimensões mais relevantes foram a *Mental Demand* e *Effort*, enquanto que no *Indirect* a *Performance* e *Mental Demand* foram as mais preponderantes. O facto da *Performance* ter um peso tão importante no modo *Indirect* pode estar relacionado com a percepção dos erros cometidos neste modo.

Por fim, no que diz respeito ao questionário correspondente à selecção de palavras para descrever o sistema, verificou-se que as quatro palavras mais frequentes são as mesmas para ambos os modos (*"Creative"*, *"Innovative"*, *"Exciting"*, *"Impressive"*).

5.2 Avaliação de usabilidade da API da componente *Foot-Menu*

Os testes de avaliação da usabilidade da API tiveram como objectivo averiguar a facilidade de implementação da componente *FootMenu*. Para isso foram recrutados um total de 6 participantes, todos do sexo masculino, com idades compreendidas entre os 23 e 27 anos, com experiência de 1 a 2 anos de programação em *Unity*. O recrutamento foi realizado por contacto directo onde se definiu o dia e o local da realização dos testes. Os testes consistiram em o participante realizar um conjunto de 5 tarefas de programação com auxilio da documentação da componente.

Procedimento

O participante quando chega ao local do teste, é explicado o conceito e o propósito da componente onde irá executar as tarefas. Após essa breve explicação, é dado a documentação da componente (que se encontra no apêndice E) ao participante, que tem alguns minutos para a estudar. Após este momento inicial e assim que este se sentir preparado dá-se início ao teste. Na *framework* de programação, é criado um novo projecto e é dada a biblioteca a integrar ao participante, bem como a lista de tarefas que este irá realizar. Cada tarefa só é dada como concluída após o participante demonstrar que a programação que fez de facto funciona como esperado, isto é, o participante terá que experimentar a acção que desenvolveu. Durante a realização das tarefas são anotado os erros/impasses/dificuldades que os participantes experienciam. No final da realização de todas as tarefas é pedido ao utilizador que comente a componente e a sua experiência com a programação.

Tarefas

Os participantes tiveram que realizar 5 tarefas que estão apresentadas na tabela 5.13.

Estas tarefas tinham como propósito manipular as várias propriedades dos menus, nomeadamente a tipo de interacção (*Direct* ou *Indirect*), o formato do menu (180° ou 360°), a criação de sub-menus e definição de *icons* para a representação do elemento.

Resultados e Discussão

Uma vez terminados estes testes, verificou-se que todos os participantes foram capazes de concluir todas as tarefas. Desta forma, parece que a configuração da componente é algo exequível para alguém com mínimo de experiência de programação. Para além disto, os participantes demoraram, em média, 53 minutos para completar todo o procedimento. Este tempo médio reflecte a inexperiência dos participantes com este sistema em particular, visto que todo o procedimento foi realizado sem um contacto prévio com o mesmo.

As dificuldades mais sentidas entre os participantes foram as seguintes:

– Falha em fazer a associação entre os menus e os elementos.
– Início da ordem dos elementos de um menu 360° ímpar na posição incorrecta.

A primeira dificuldade é, das duas, a mais frequente estando associada à inexperiência com o sistema. Uma vez que os participantes compreendem o que é um menu e um elemento e como os interligar, estas dificuldades diminuem significativamente. Já a última dificuldade poderá estar relacionada com o esquecimento de seguir o padrão de disposição dos elementos num menu 360°.

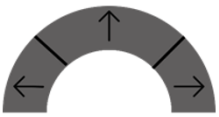
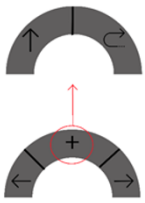
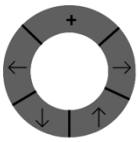
Tarefa 1	Imagine que está num espaço 3D e que quer deslocar-se para a direita, em frente e para a esquerda. Construa um menu Directo e 180º com essas 3 opções e confira na opçao de "andar em frente" a possibilidade de fazer print na consola de que foi selecionada.	
Tarefa 2	Com o menu criado anteriormente, mude a prespectiva de utilização de Directo para Indirecto.	-
Tarefa 3	Com o menu criado na Tarefa 1, altere-o de forma a que na opçõa "andar em frente", entre agora num submenu, ou seja, que nessa posição passe a ser um "+" e que depois de selecionado que entre noutro menu com a opção de andar em frente e retroceder para o menu anterior.	
Tarefa 4	Imagine que tem um drone à sua frente e que o quer manipular de maneira a que se mova no ambiente virtual. Construa um menu 360º com 5 opções na qual faça com que o drone se mova para cima e para baixo, para a esquerda e para a direita e ao carregar em "+" faça print das coordenadas.	
Tarefa 5	Crie um menu com 4 opções (a prespectiva e tipo de menu fica ao critério do programador) e que no lugar do icon insira texto do género: Elem 1, Elem2, ...	-

Tabela 5.13: Tarefas a realizar pelos programadores

Capítulo 6

Conclusão

Este estudo teve como objectivo o desenvolvimento de uma *toolkit* que facilite a implementação por parte de programadores, de conteúdos de RV no que diz respeito a interacções com os pés e que possa ser aplicado em visitas virtuais. Para além deste, era também objectivo a avaliação da componente desenvolvida, tanto por parte de programadores como de utilizadores ao nível da usabilidade e experiência de utilização.

Face ao estudo realizado até ao momento, é possível verificar que o pé consegue realizar um conjunto bastante abrangente de gestos. Estes gestos podem ser usados para diversas interacções em RV mas para tal é necessário ter um sistema de captação adequado à função a desempenhar.

O desenvolvimento da *toolkit* resultou de um processo de tentativa e erro, onde foram testadas várias hipóteses, tendo culminado num *package* capaz de utilizar os pés como forma de interacção com a RV. Das diversas avaliações efectuadas é possível concluir que o sistema é *user-friendly*, não apenas para os utilizadores, mas também para os programadores que o queiram integrar noutros projectos. Apesar desta biblioteca ser robusta, algumas melhorias poderão ser tomadas em consideração.

Assim, como em qualquer trabalho, é importante reconhecer as limitações deste estudo. No que diz respeito às tentativas de reconhecimento de gestos, a forma manual está limitada pelo facto de o sistema de coordenadas usadas pelo *tracker* ser fixo em relação ao espaço virtual e não à posição relativa do utilizador. Isto limita o reconhecimento de determinado gestos previamente definidos num sistema de coordenadas fixo, pois qualquer rotação do utilizador iria influenciar a forma como o gesto seria detectado. Uma possibilidade de contornar este problema seria o desenvolvimento de um sistema de coordenadas que fosse dinâmico em relação ao dispositivo. Já o reconhecimento de gestos com recurso a biblioteca de reconhecimento de padrões tem uma importante limitação no que concerne à detecção da intenção, ou não, de executar determinado gesto, ou seja, mesmo que o utilizador execute um gesto involuntariamente o sistema irá tentar interpretar esse gesto e devolver um *output* indesejado. Para resolver este problema seria necessário implementar um mecanismo de activação do reconhecimento de gestos, podendo este passar pela conjugação com *smart soles* cuja detecção de pressão poderia activar/desactivar este mecanismo.

Quanto às limitações da utilização conjunta da técnica de GAZE com o sistema de reconhecimento de gestos, destaca-se a necessidade de manter a retícula sobre o objecto durante toda a execução do gesto, a impossibilidade de cancelar/interromper o gesto uma vez iniciado e a obrigação do utilizador estar imóvel antes de executar o gesto. Estas podem ser entendidas como limitações do utilizador, na medida em que poderiam ser

ultrapassadas recorrendo a um sistema que permita alguma margem de erro, como uma extensão de limiares, isto é, ter um *threshold* para a deslocação da retícula ou para o movimento do utilizador antes da execução dos gestos.

Relativamente à componente *FootMenu*, verificou-se que muitos dos utilizadores ultrapassavam as áreas de selecção ao escolher uma das opções. Trabalho futuro pode considerar as alturas de cada utilizador, nomeadamente no que diz respeito às distâncias necessárias para que a amplitude do movimento da perna coincida com a dimensão das áreas de selecção para cada elemento do menu. Por outro lado, no modo *Indirect*, verificou-se alguma desorientação no momento da selecção das opções do menu. Poderá ser considerado no futuro a implementação de uma representação do pé no menu flutuante, desta forma o utilizador poderá manter a percepção das distâncias do pé às opções.

Referências

[1] F. Müller, J. McManus, S. Günther, M. Schmitz, M. Mühlhäuser, and M. Funk, "Mind the tap: Assessing foot-taps for interacting with head-mounted displays," *Conference on Human Factors in Computing Systems - Proceedings*, pp. 1–13, 2019.

[2] W. Saunders and D. Vogel, "The performance of indirect foot pointing using discrete taps and kicks while standing," *Proceedings - Graphics Interface*, vol. 2015-June, pp. 265–272, 2015.

[3] K. Zhong, F. Tian, and H. Wang, "Foot menu: Using heel rotation information for menu selection," *Proceedings - International Symposium on Wearable Computers, ISWC*, no. State 1, pp. 115–116, 2011.

[4] D. S. Lopes, F. Relvas, S. Paulo, Y. Rekik, L. Grisoni, and J. Joaquim, "FEETICHE: FEET Input for contactless hand gEsture interaction," *Proceedings - VRCAI 2019: 17th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*, no. October, 2019.

[5] "HTC VIVE Tracker (2018) Developer Guidelines Ver. 1.0 Version Control," 2018.

[6] A. Roaas and G. B. Andersson, "Normal range of motion of the hip, knee and ankle joints in Male subjects, 30-40 years of age," *Acta Orthopaedica*, vol. 53, no. 2, pp. 205–208, 1982.

[7] E. Vellosso, D. Schmidt, J. Alexander, H. Gellersen, and A. Bulling, "The Feet in HCI: A Survey of Foot-Based Interaction," *ACM Comput. Surv 9*, vol. 9, no. 4, 2015.

[8] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal, "Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness," *The International Journal of Aviation Psychology*, vol. 3, no. 3, pp. 203–220, 1993.

[9] I. E. Sutherland, "The ultimate display," pp. 506–508, 1965.

[10] E. J. Dawe and J. Davis, "(vi) Anatomy and biomechanics of the foot and ankle," *Orthopaedics and Trauma 25*, pp. 279–286, 2011.

[11] K. H. E. Kroemer, "Foot operation of controls," *Ergonomics 14*, pp. 333–370, 1971.

[12] V. Paelke, C. Reimann, and D. Stichling, "Foot-based mobile interaction with games," *ACM International Conference Proceeding Series*, vol. 74, pp. 321–324, 2004.

[13] E. X. De Lima Filho, M. B. Nunes, J. Comba, and L. Nedel, "Why not with the foot?," *Brazilian Symposium on Games and Digital Entertainment, SBGAMES*, pp. 270–281, 2011.

[14] J. Wang and R. W. Lindeman, "Comparing isometric and elastic surfboard interfaces for leaning-based travel in 3D virtual environments," *IEEE Symposium on 3D User Interfaces 2012, 3DUI 2012 - Proceedings*, pp. 31–38, 2012.

[15] D. Valkov, F. Steinicke, G. Bruder, and K. Hinrichs, "A multi-touch enabled human-transporter metaphor for virtual 3D traveling," in *2010 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 79–82, IEEE, mar 2010.

[16] S. Beckhaus, K. J. Blom, and M. Haringer, "Intuitive, Hands-free Travel Interfaces for Virtual Environments," *IEEE VR Workshop: New Directions in 3D User Interfaces*, pp. 57–60, 2005.

[17] D. A. Bowman and L. F. Hodges, "Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments," *Journal of Visual Languages and Computing*, vol. 10, no. 1, pp. 37–53, 1999.

[18] Z. Lv, S. Feng, M. S. Lal Khan, S. Ur Réhman, and H. Li, "Foot motion sensing: Augmented game interface based on foot interaction for smartphone," *Conference on Human Factors in Computing Systems - Proceedings*, pp. 293–296, 2014.

[19] A. Hilsendeger, S. Brandauer, J. Tolksdorf, and C. Fröhlich, "Navigation in Virtual Reality with the Wii Balance Board," *6th Workshop on Virtual and Augmented Reality*, pp. 269–280, 2009.

[20] D. Valkov, F. Steinicke, G. Bruder, and K. Hinrichs, "A multi-touch enabled human-transporter metaphor for virtual 3D traveling," *3DUI 2010 - IEEE Symposium on 3D User Interfaces 2010, Proceedings*, pp. 79–82, 2010.

[21] S. Tregillus and E. Folmer, "VR-STEP: Walking-in-place using inertial sensing for hands free navigation in mobile VR environments," *Conference on Human Factors in Computing Systems - Proceedings*, pp. 1250–1255, 2016.

[22] J. Bhandari, S. Tregillus, and E. Folmer, "Legomotion: Scalable walking-based virtual locomotion," *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST*, vol. Part F1319, 2017.

[23] V. Metsis, K. S. Smith, and D. Gobert, "Integration of virtual reality with an omnidirectional treadmill system for multi-directional balance skills intervention," *2017 International Symposium on Wearable Robotics and Rehabilitation, WeRob 2017*, pp. 1–2, 2018.

[24] T. Cakmak and H. Hager, "Cyberith virtualizer - A locomotion device for virtual reality," *ACM SIGGRAPH 2014 Emerging Technologies, SIGGRAPH 2014*, p. 4503, 2014.

[25] N. A. Skopp, D. J. Smolenski, M. J. Metzger-Abamukong, A. A. Rizzo, and G. M. Reger, "A Pilot Study of the VirtuSphere as a Virtual Reality Enhancement," *International Journal of Human-Computer Interaction*, vol. 30, no. 1, pp. 24–31, 2014.

[26] K. Drotner, V. Dziekan, R. Parry, and K. C. Schrøder, "Environments," *The Routledge Handbook of Museums, Media and Communication*, vol. 32, no. 1, pp. 97–99, 2018.

[27] J. R. Lackner, "Simulator sickness," *The Journal of the Acoustical Society of America*, vol. 92, no. 4, pp. 2458–2458, 1992.

[28] K. Klamka, A. Siegel, S. Vogt, F. Göbel, S. Stellmach, and R. Dachselt, "Look & Pedal," pp. 123–130, 2015.

[29] J. C. S. Cardoso and A. Perrotta, "A survey of real locomotion techniques for immersive virtual reality applications on head-mounted displays," *Computers & Graphics*, vol. 85, pp. 55–73, 2019.

[30] J. Auda, M. Pascher, and S. Schneegass, "Around the (virtual) world infinite walking in virtual reality using electrical muscle stimulation," *Conference on Human Factors in Computing Systems - Proceedings*, pp. 1–8, 2019.

[31] J. Von Willich, M. Schmitz, F. Müller, D. Schmitt, and M. Mühlhäuser, "Podoportation: Foot-Based Locomotion in Virtual Reality," *Conference on Human Factors in Computing Systems - Proceedings*, 2020.

[32] M. Karam and m. c. Schraefel, "A Taxonomy of Gestures in Human Computer Interactions," *Technical Report, Eletronics and Computer Science.*, no. January, pp. 1–45, 2005.

[33] F. Quek, R. Bryll, C. Kirbas, D. McNeill, K. E. McCullough, X. F. Ma, R. Ansari, and S. Duncan, "Multimodal Human Discourse," *ACM Transactions on Computer-Human Interaction*, vol. 9, no. 3, pp. 171–193, 2002.

[34] A. O. Chan, A. H. Chan, A. W. Ng, and B. L. Luk, "A preliminary analysis of movement times and subjective evaluations for a visually-controlled foot-tapping task on touch pad device," *Proceedings of the International MultiConference of Engineers and Computer Scientists 2010, IMECS 2010*, no. March 2010, pp. 1968–1970, 2010.

[35] A. Crossan, S. Brewster, and A. Ng, "Foot tapping for mobile interaction," *Proceedings of the 2010 British Computer Society Conference on Human-Computer Interaction, BCS-HCI 2010*, pp. 418–422, 2010.

[36] S. Papetti, M. Civolani, and F. Fontana, "Rhythm'n'Shoes: a wearable foot tapping interface with audio-tactile feedback," *International Conference on New Interfaces for Musical Expression (NIME)*, no. January 2017, 2011.

[37] T. Augsten, K. Kaefer, R. Meusel, C. Fetzer, D. Kanitz, T. Stoff, T. Becker, C. Holz, and P. Baudisch, "Multitoe," p. 209, 2010.

[38] J. La Viola, D. A. Feliz, D. F. Keefe, and R. C. Zeleznik, "Hands-free multi-scale navigation in virtual environments," *Proceedings of the Symposium on Interactive 3D Graphics*, pp. 9–15, 2001.

[39] J. Scott, D. Dearman, K. Yatani, and K. N. Truong, "Sensing foot gestures from the pocket," *UIST 2010 - 23rd ACM Symposium on User Interface Software and Technology*, pp. 199–208, 2010.

[40] J. Von Willich, M. Schmitz, F. Müller, D. Schmitt, and M. Mühlhäuser, "Podoportation: Foot-Based Locomotion in Virtual Reality," *Conference on Human Factors in Computing Systems - Proceedings*, 2020.

[41] T. Han, J. Alexander, A. Karnik, P. Irani, and S. Subramanian, "Kick: Investigating the use of kick gestures for mobile interactions," *Mobile HCI 2011 - 13th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pp. 29–32, 2011.

[42] R. Jota, P. Lopes, D. Wigdor, and J. Jorge, "Let's kick it: How to stop wasting the bottom third of your large scale display," *Conference on Human Factors in Computing Systems - Proceedings*, no. 1, pp. 1411–1414, 2014.

[43] P. T. Wilson, W. Kalescky, A. MacLaughlin, and B. Williams, "VR locomotion," pp. 243–249, 2016.

[44] R. L. Simonen, M. C. Battie, T. Videman, and L. E. Gibbons, "Comparison of Foot and Hand Reaction Times Among," *Perceptual and Motor Skills*, vol. 80, no. 3c, pp. 1243–1249, 1995.

[45] A. Schmidt, "Implicit human computer interaction through context," *Personal and Ubiquitous Computing*, vol. 4, no. 2-3, pp. 191–199, 2000.

[46] B. Abinaya, V. Latha, and M. Suchetha, "An advanced gait monitoring system based on air pressure sensor embedded in a shoe," *Procedia Engineering*, vol. 38, no. 3, pp. 1634–1643, 2012.

[47] J. Hernandez, W. Amanhoud, A. Haget, H. Bleuler, A. Billard, and M. Bouri, "Four-Arm manipulation via feet interfaces," *arXiv*, 2019.

[48] T. Pakkanen and R. Raisamo, "Appropriateness of foot interaction for non-accurate spatial tasks," *Conference on Human Factors in Computing Systems - Proceedings*, pp. 1123–1126, 2004.

[49] "VIVE United States | Next-level VR Headsets and Apps." https://www.vive.com/us/.

[50] "Valve Index - Melhora a tua experiência - Valve Corporation." https://www.valvesoftware.com/pt/index.

[51] "Oculus Quest 2: Os nossos mais recentes óculos de RV tudo-em-um mais avançados | Oculus." https://www.oculus.com/quest-2/?locale=pt_PT.

[52] "VIVE Tracker | VIVE United States." https://www.vive.com/us/accessory/vive-tracker/.

[53] "Tundra Tracker Pricing Revealed, How it Compares to Vive Tracker." https://www.roadtovr.com/tundra-tracker-pricing-comparison-vive-tracker/.

[54] "SlimeVR Full-Body Tracker | Crowd Supply." https://www.crowdsupply.com/slimevr/slimevr-full-body-tracker.

[55] "Introduction – A-Frame." https://aframe.io/docs/1.2.0/introduction/.

[56] "Unity - Manual: XR." https://docs.unity3d.com/Manual/XR.html.

[57] "Unreal Engine for extended reality (XR): AR, VR & MR - Unreal Engine." https://www.unrealengine.com/en-US/xr.

[58] "Personal Kanban - How to Focus and Achieve Your Goals." https://kanbanzone.com/resources/kanban/personal-kanban/.

[59] "Unity - Scripting API:." https://docs.unity3d.com/ScriptReference/XR.XRNodeState.html.

[60] "Rede neural artificial – Wikipédia, a enciclopédia livre." https://pt.wikipedia.org/wiki/Rede_neural_artificial.

[61] "MiVRy - 3D Gesture Recognition | Systems | Unity Asset Store." https://assetstore.unity.com/packages/templates/systems/mivry-3d-gesture-recognition-143176.

[62] J. R. Lewis, "The System Usability Scale: Past, Present, and Future," *International Journal of Human–Computer Interaction*, vol. 34, pp. 577–590, jul 2018.

[63] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research," in *Advances in Psychology*, vol. 52, pp. 139–183, jan 1988.

[64] R. A. Grier, "How high is high? A meta-analysis of NASA-TLX global workload scores," in *Proceedings of the Human Factors and Ergonomics Society*, vol. 2015-Janua, pp. 1727–1731, sep 2015.

Apêndices

Esta página é deixada intencionalmente em branco.

Apêndice A

Planeamento

Neste apêndice são apresentadas as análises gráficas do método de Gannt, que devido às dimensões não exista possibilidade de enquadrar no Capítulo 4.

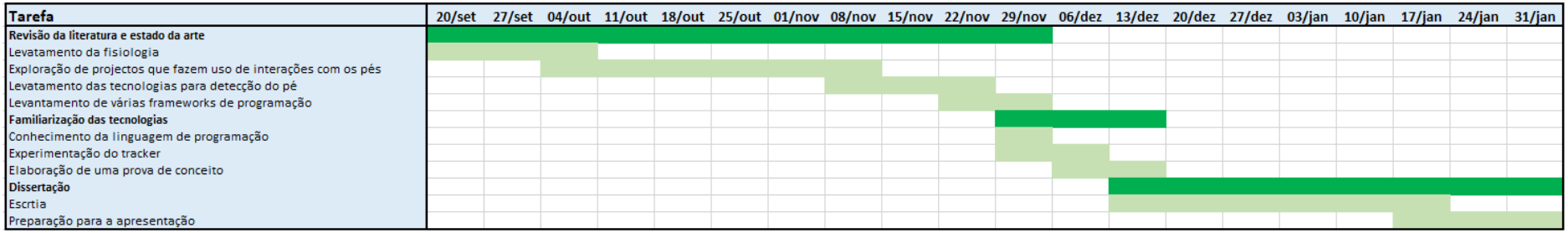


Figura A.1: Planeamento 1º semestre

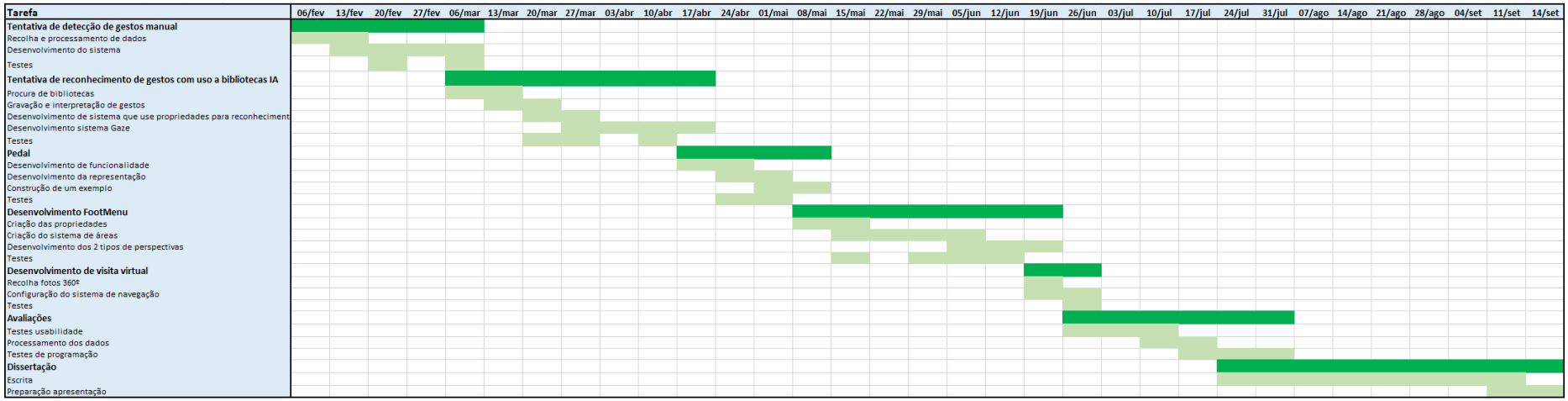


Figura A.2: Planeamento 2º semestre

Esta página é deixada intencionalmente em branco.

Apêndice B

Reconhecimento de Gestos

Apresenta-se de seguida a lista de figuras dos restantes gestos que a componente desenvolvida consegue realizar o reconhecimento.

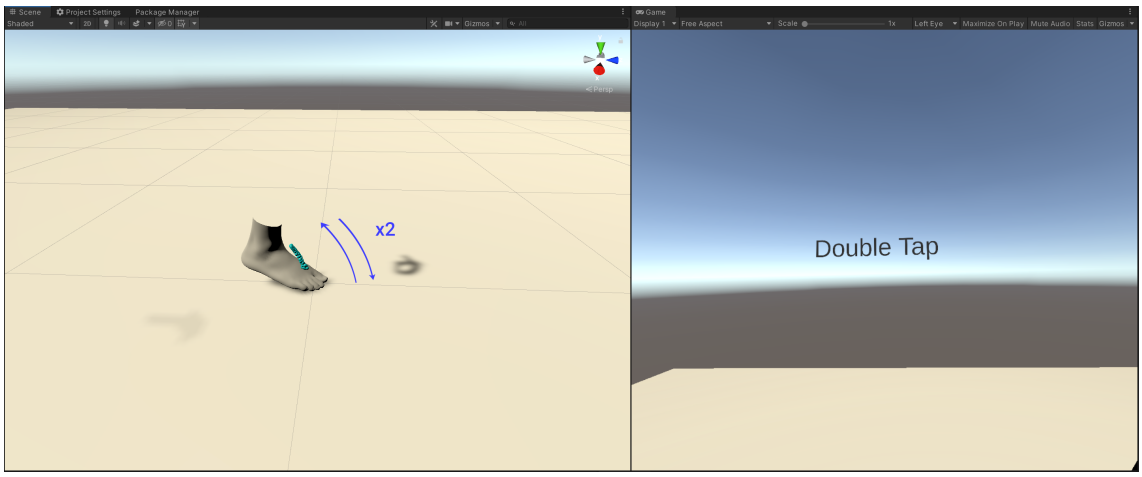


Figura B.1: Reconhecimento do gesto *double toe tap* pela componente desenvolvida

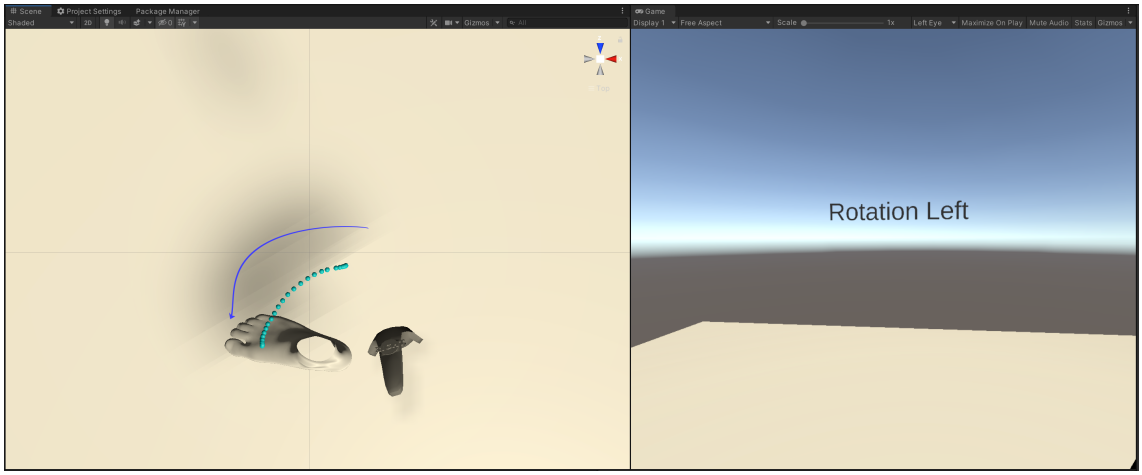


Figura B.2: Reconhecimento do gesto de rotação à esquerda pela componente desenvolvida

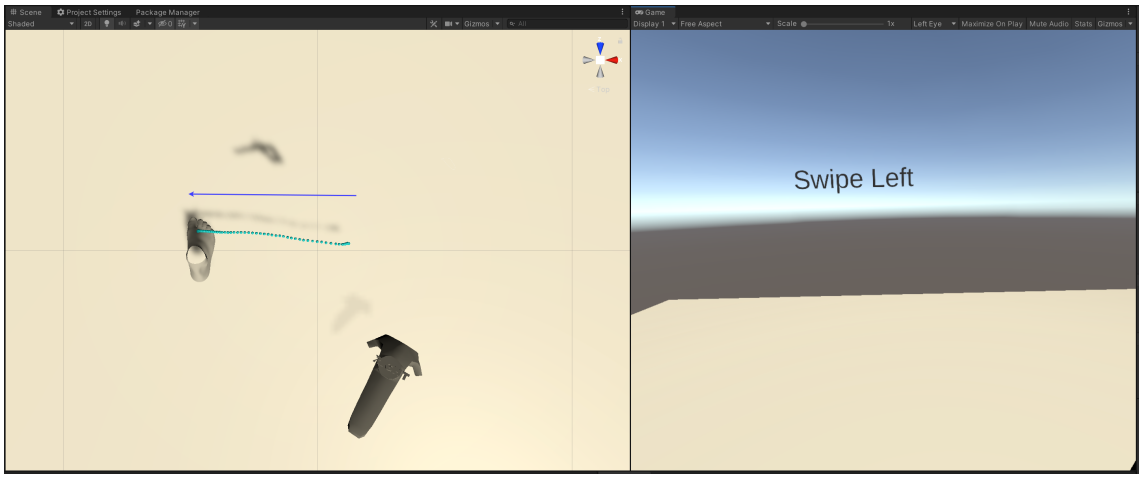


Figura B.3: Reconhecimento do gesto *swipe* à esquerda pela componente desenvolvida

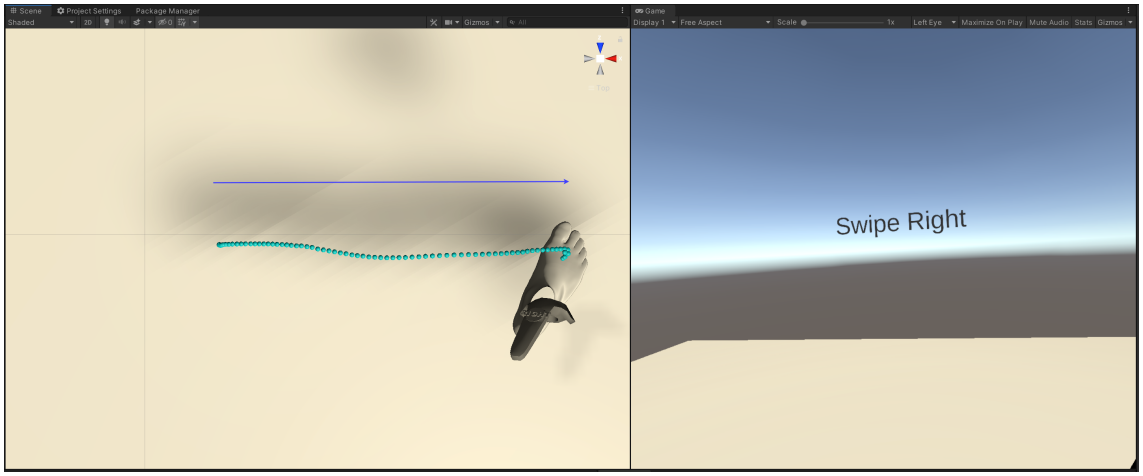


Figura B.4: Reconhecimento do gesto *swipe* à direita pela componente desenvolvida

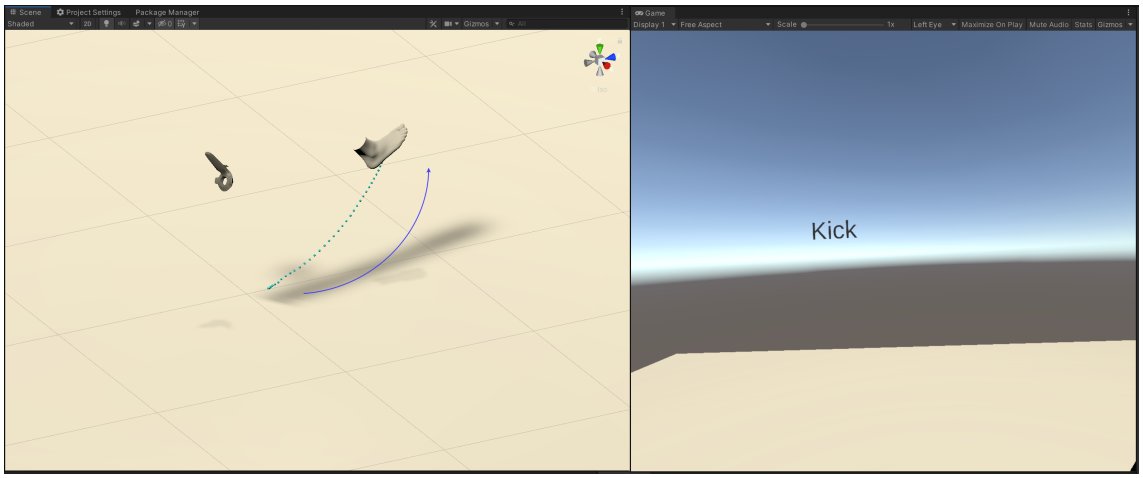


Figura B.5: Reconhecimento do gesto de chuto pela componente desenvolvida

Esta página é deixada intencionalmente em branco.

Apêndice C

Técnica de GAZE

Apresenta-se de seguida o funcionamento do sistema GAZE desenvolvido para auxiliar o reconhecimento de gestos através de ANN.

No início, foi desenvolvido um sistema de retícula para a interacção com um objecto. Este sistema detecta essa interacção sempre que a retícula sobrepõe o objecto específico. Sempre que se dá a interacção, a retícula altera de forma e o objecto fica com uma iluminação á sua volta, alertando o utilizador que o objecto em questão pode ser interagido. Tanto a retícula como a iluminação podem ser personalizados para as preferências do programador. Na figura C.1 são ilustradas as propriedades do sistema desenvolvido.

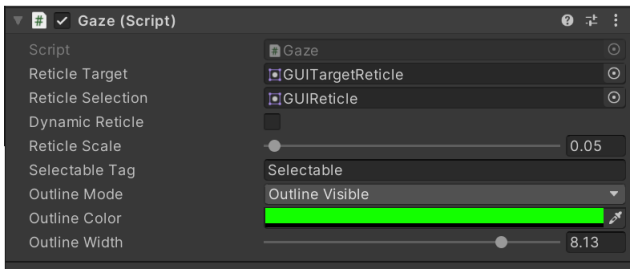
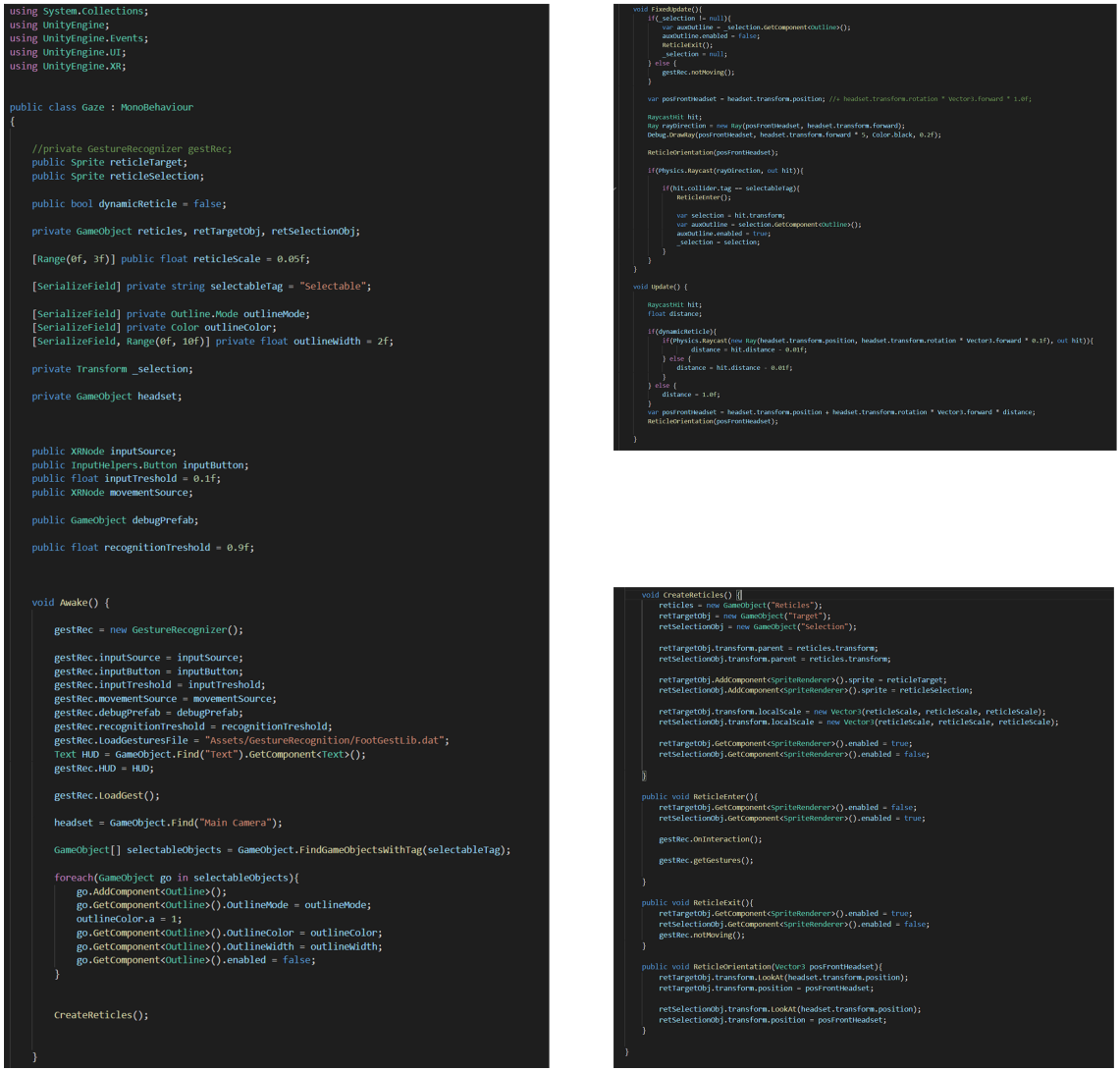


Figura C.1: Propriedades do sistema GAZE desenvolvido

De forma de conseguir detectar a sobreposição da retícula sobre o objecto, foi usado a funcionalidade *RaycastHit*, que faz uso de um raio para detectar todas as intersecções. Este raio tem origem na retícula e pode intersectar objectos que tenham a *tag* como *Selectable*. Assim, sempre que a retícula se sobrepuser sobre determinado objecto, é accionado a componente de detecção de gestos. Na imagem seguinte é apresentado todo o código desenvolvido para o sistema GAZE.



```
using System.Collections;
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.UI;
using UnityEngine.XR;


public class Gaze : MonoBehaviour
{

    //private GestureRecognizer gestRec;
    public Sprite reticleTarget;
    public Sprite reticleSelection;

    public bool dynamicReticle = false;

    private GameObject reticles, retTargetObj, retSelectionObj;

    [Range(0f, 3f)] public float reticleScale = 0.05f;

    [SerializeField] private string selectableTag = "Selectable";

    [SerializeField] private Outline.Mode outlineMode;
    [SerializeField] private Color outlineColor;
    [SerializeField, Range(0f, 10f)] private float outlineWidth = 2f;

    private Transform _selection;

    private GameObject headset;



    public XRNode inputSource;
    public InputHelpers.Button inputButton;
    public float inputTreshold = 0.1f;
    public XRNode movementSource;

    public GameObject debugPrefab;

    public float recognitionTreshold = 0.9f;



    void Awake() {

        gestRec = new GestureRecognizer();

        gestRec.inputSource = inputSource;
        gestRec.inputButton = inputButton;
        gestRec.inputTreshold = inputTreshold;
        gestRec.movementSource = movementSource;
        gestRec.debugPrefab = debugPrefab;
        gestRec.recognitionTreshold = recognitionTreshold;
        gestRec.LoadGesturesFile = "Assets/GestureRecognition/FootGestLib.dat";
        Text HUD = GameObject.Find("Text").GetComponent<Text>();
        gestRec.HUD = HUD;

        gestRec.LoadGest();

        headset = GameObject.Find("Main Camera");

        GameObject[] selectableObjects = GameObject.FindGameObjectsWithTag(selectableTag);

        foreach(GameObject go in selectableObjects){
            go.AddComponent<Outline>();
            go.GetComponent<Outline>().OutlineMode = outlineMode;
            outlineColor.a = 1;
            go.GetComponent<Outline>().OutlineColor = outlineColor;
            go.GetComponent<Outline>().OutlineWidth = outlineWidth;
            go.GetComponent<Outline>().enabled = false;
        }


        CreateReticles();


    }
```

```
void FixedUpdate(){
    if(_selection != null){
        var auxOutline = _selection.GetComponent<Outline>();
        auxOutline.enabled = false;
        ReticleExit();
        _selection = null;
    } else {
        gestRec.notMoving();
    }

    var posFrontHeadset = headset.transform.position; //+ headset.transform.rotation * Vector3.forward * 1.0f;

    RaycastHit hit;
    Ray rayDirection = new Ray(posFrontHeadset, headset.transform.forward);
    Debug.DrawRay(posFrontHeadset, headset.transform.forward * 5, Color.black, 0.2f);

    ReticleOrientation(posFrontHeadset);

    if(Physics.Raycast(rayDirection, out hit)){

        if(hit.collider.tag == selectableTag){
            ReticleEnter();

            var selection = hit.transform;
            var auxOutline = selection.GetComponent<Outline>();
            auxOutline.enabled = true;
            _selection = selection;
        }
    }
}

void Update() {

    RaycastHit hit;
    float distance;

    if(dynamicReticle){
        if(Physics.Raycast(new Ray(headset.transform.position, headset.transform.rotation * Vector3.forward * 0.1f), out hit)){
                distance = hit.distance - 0.01f;
        } else {
            distance = hit.distance - 0.01f;
        }
    } else {
        distance = 1.0f;
    }
    var posFrontHeadset = headset.transform.position + headset.transform.rotation * Vector3.forward * distance;
    ReticleOrientation(posFrontHeadset);

}
```

```
void CreateReticles() {
    reticles = new GameObject("Reticles");
    retTargetObj = new GameObject("Target");
    retSelectionObj = new GameObject("Selection");

    retTargetObj.transform.parent = reticles.transform;
    retSelectionObj.transform.parent = reticles.transform;

    retTargetObj.AddComponent<SpriteRenderer>().sprite = reticleTarget;
    retSelectionObj.AddComponent<SpriteRenderer>().sprite = reticleSelection;

    retTargetObj.transform.localScale = new Vector3(reticleScale, reticleScale, reticleScale);
    retSelectionObj.transform.localScale = new Vector3(reticleScale, reticleScale, reticleScale);

    retTargetObj.GetComponent<SpriteRenderer>().enabled = true;
    retSelectionObj.GetComponent<SpriteRenderer>().enabled = false;

}

public void ReticleEnter(){
    retTargetObj.GetComponent<SpriteRenderer>().enabled = false;
    retSelectionObj.GetComponent<SpriteRenderer>().enabled = true;

    gestRec.OnInteraction();

    gestRec.getGestures();

}

public void ReticleExit(){
    retTargetObj.GetComponent<SpriteRenderer>().enabled = true;
    retSelectionObj.GetComponent<SpriteRenderer>().enabled = false;
    gestRec.notMoving();
}

public void ReticleOrientation(Vector3 posFrontHeadset){
    retTargetObj.transform.LookAt(headset.transform.position);
    retTargetObj.transform.position = posFrontHeadset;

    retSelectionObj.transform.LookAt(headset.transform.position);
    retSelectionObj.transform.position = posFrontHeadset;
}
}
```

Figura C.2: Código da implementação da técnica de GAZE

Esta página é deixada intencionalmente em branco.

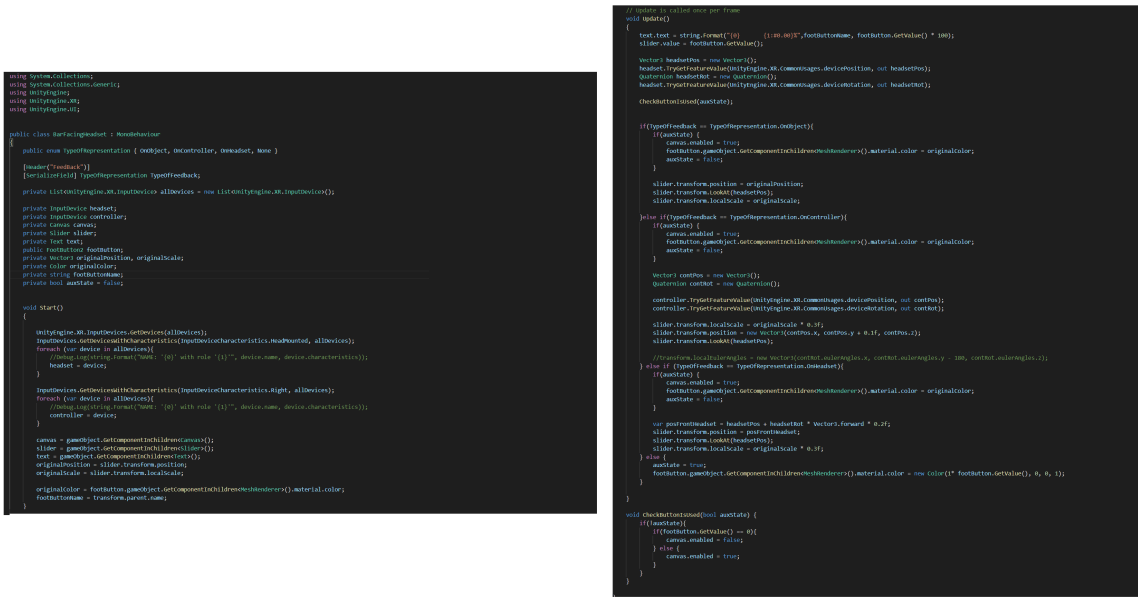
Apêndice D

Implementação do Pedal

Apresenta-se de seguida toda a implementação realizada para o desenvolvimento da componente pedal. Na figura D.1 é ilustrado o funcionamento interno do pedal, isto é, como o pedal calcula a "força"que está a ser exercida, bem como os 3 estados presentes na componente.

Figura D.1: Implementação do funcionamento do pedal

Na figura D.2 é ilustrado o desenvolvimento da barra de intensidade sobre os 4 modos disponíveis. A barra de intensidade consiste numa faixa que é preenchida consoante o esforço exercido sobre o pedal. Para o modo *OnObject*, o sistema vai buscar a posição do objecto e coloca a barra por cima dele. Para o modo *OnController*, o sistema está constantemente a rastrear o controlador para que a barra de intensidade acompanhe esse movimento. Para o modo *OnHeadset*, à semelhança do anterior, o sistema está constantemente a rastrear o headset de modo a colocar a barra de intensidade à frente do olhar do utilizador. Por fim, para o método *None*, o sistema usa os valores do esforço exercido no pedal e aplica-o para calcular a sua cor.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR;
using UnityEngine.UI;


public class BarFacingHeadset : MonoBehaviour
{
    public enum TypeOfRepresentation { OnObject, OnController, OnHeadset, None }

    [Header("FeedBack")]
    [SerializeField] TypeOfRepresentation TypeOfFeedback;

    private List<UnityEngine.XR.InputDevice> allDevices = new List<UnityEngine.XR.InputDevice>();

    private InputDevice headset;
    private InputDevice controller;
    private Canvas canvas;
    private Slider slider;
    private Text text;
    public FootButton FootButton;
    private Vector3 originalPosition, originalScale;
    private Color originalColor;
    private string FootButtonName;
    private bool auxState = false;


    void Start()
    {

        UnityEngine.XR.InputDevices.GetDevices(allDevices);
        InputDevices.GetDevicesWithCharacteristics(InputDeviceCharacteristics.HeadMounted, allDevices);
        foreach (var device in allDevices){
            //Debug.Log(string.Format("DATA: '{0}' with role '{1}'", device.name, device.characteristics));
            headset = device;
        }

        InputDevices.GetDevicesWithCharacteristics(InputDeviceCharacteristics.Right, allDevices);
        foreach (var device in allDevices){
            //Debug.Log(string.Format("NAME: '{0}' with role '{1}'", device.name, device.characteristics));
            controller = device;
        }

        canvas = gameObject.GetComponentInChildren<Canvas>();
        slider = gameObject.GetComponentInChildren<Slider>();
        text = gameObject.GetComponentInChildren<Text>();
        originalPosition = slider.transform.position;
        originalScale = slider.transform.localScale;

        originalColor = footButton.gameObject.GetComponentInChildren<MeshRenderer>().material.color;
        FootButtonName = transform.parent.name;
    }
```

```
// Update is called once per frame
void Update()
{
    text.text = string.Format("[0]        [1:#0.00]E",footButtonName, footButton.GetValue() * 100);
    slider.value = footButton.GetValue();

    Vector3 headsetPos = new Vector3();
    headset.TryGetFeatureValue(UnityEngine.XR.CommonUsages.devicePosition, out headsetPos);
    Quaternion headsetRot = new Quaternion();
    headset.TryGetFeatureValue(UnityEngine.XR.CommonUsages.deviceRotation, out headsetRot);

    CheckButtonIssued(auxState);


    if(TypeOfFeedback == TypeOfRepresentation.OnObject){
        if(auxState) {
            canvas.enabled = true;
            footButton.gameObject.GetComponentInChildren<MeshRenderer>().material.color = originalColor;
            auxState = false;
        }

        slider.transform.position = originalPosition;
        slider.transform.LookAt(headsetPos);
        slider.transform.localScale = originalScale;

    }else if(TypeOfFeedback == TypeOfRepresentation.OnController){
        if(auxState) {
            canvas.enabled = true;
            footButton.gameObject.GetComponentInChildren<MeshRenderer>().material.color = originalColor;
            auxState = false;
        }

        Vector3 contPos = new Vector3();
        Quaternion contRot = new Quaternion();

        controller.TryGetFeatureValue(UnityEngine.XR.CommonUsages.devicePosition, out contPos);
        controller.TryGetFeatureValue(UnityEngine.XR.CommonUsages.deviceRotation, out contRot);

        slider.transform.localScale = originalScale * 0.3f;
        slider.transform.position = new Vector3(contPos.x, contPos.y + 0.1f, contPos.z);
        slider.transform.LookAt(headsetPos);

        //transform.localEulerAngles = new Vector3(contRot.eulerAngles.x, contRot.eulerAngles.y - 180, contRot.eulerAngles.z);
    } else if (TypeOfFeedback == TypeOfRepresentation.OnHeadset){
        if(auxState) {
            canvas.enabled = true;
            footButton.gameObject.GetComponentInChildren<MeshRenderer>().material.color = originalColor;
            auxState = false;
        }

        var postFrontHeadset = headsetPos + headsetRot * Vector3.forward * 0.2f;
        slider.transform.position = postFrontHeadset;
        slider.transform.LookAt(headsetPos);
        slider.transform.localScale = originalScale * 0.3f;
    } else {
        auxState = true;
        footButton.gameObject.GetComponentInChildren<MeshRenderer>().material.color = new Color(1* footButton.GetValue(), 0, 0, 1);
    }
}

void CheckButtonIssued(bool auxState) {
    if(!auxState){
        if(footButton.GetValue() == 0){
            canvas.enabled = false;
        } else {
            canvas.enabled = true;
        }
    }
}
```

Figura D.2: Implementação do funcionamento do sistema de barra de intensidade

Esta página é deixada intencionalmente em branco.

Apêndice E

Documentação FootMenu

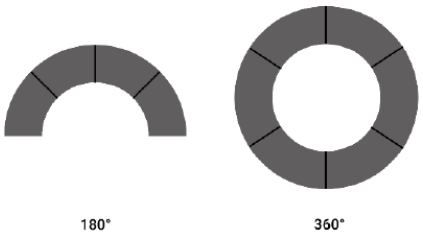
Apresenta-se de seguida a documentação que os programadores têm acesso para a utilização da componente *FootMenu*.

Conceitos

A componente *FootMenu* permite ao utilizador usar o pé para fazer uma selecção num sistema como um menu. Este sistema tem diversas propriedades, sendo elas:

- Apresentar ao utilizador 2 tipos de perspectivas:
  i) **Direct** (a) – em que o menu é representado no chão, sendo ativado sempre que o utilizador olha para baixo;
  ii) **Indirect** (b) – em que o menu é representado em frente do utilizador, sendo ativado quando o utilizador levanta a perna para a frente;

- Apresentar o menu com 2 tipos de representação:
  1. **180°** - nesta representação os elementos são preenchidos em sentido anti-horário começando na direita;
  2. **360°** - nesta representação, os elementos são preenchidos em sentido anti-horário mas começando na direita, caso o numero de elementos seja par, ou começando no topo, caso seja ímpar;



- Ter a possibilidade de acompanhar o utilizador quando este se moviementa;

Ficheiros incluidos na componente FootMenu:

- Assets/Scripts/Menu.cs -> ficheiro que contém os atributos relativos à criação do Menu (180 ou 360 e os elementos a usar);
- Assets/Scripts/MenuElement.cs -> ficheiro que contém os atributos relativos a cada Elemento a ser usado no Menu (Nome, Icon, orientação do icon, o evento ao selecionar e o proximo menu a aparecer);
- Assets/Scripts/MenuCakePiece.cs -> ficheiro que contém a representação gráfica do quadrante do Menu (icon e representação do quadrante);
- Assets/Scripts/Dummy.cs -> ficheiro usado para representar os vários tipo de Menu ao utilizador (Directo ou Indirecto);
- Assets/Scripts/MenuMB.cs -> ficheiro que contém as funções internas de selecção e representação do Menu;
- Assets/Scripts/HighliterCollider.cs -> ficheiro usado para fazer "highlith" do quadrante do Menu a selecionar;
- Assets/Scripts/TriggerCollision.cs -> ficheiro usado para detectar selecção do quadrante do Menu;

- Menu MB.prefab -> Objecto que contem a informação relativa ao Menu e sua representação
- CakePiece.prefab -> Objecto usado para representar graficamente os quadrantes do Menu;

Modo de utilização:

Para a utilização desta componente para criar um menu é necessário importar a package *FootMenu.unitypackage* que se encontra no repositório:

https://github.com/FranciscoSantiago15/Feet-Are-Not-Just-For-Walking

Após a importação da package para o projecto é necessário ter atenção a 3 importantes etapas:

- Criar Menu;
- Criar Elementos do Menu;
- Configuração no projeto;

Criação de um Menu

1. Carregar com o botão direito do rato, selecionar "Create", de seguida "FootMenu" e por fim "Menu". Após este passo irá aparecer o objecto criado;
2. Selecionar o objecto criado e, no inspector, escolher o tipo de menu (180° ou 360°) e especificar o numero de elementos que o menu vai ter.
3. Após o passo anterior, adicionar a lista de elementos os elementos para cada quadrante.

Criação de um Elemento

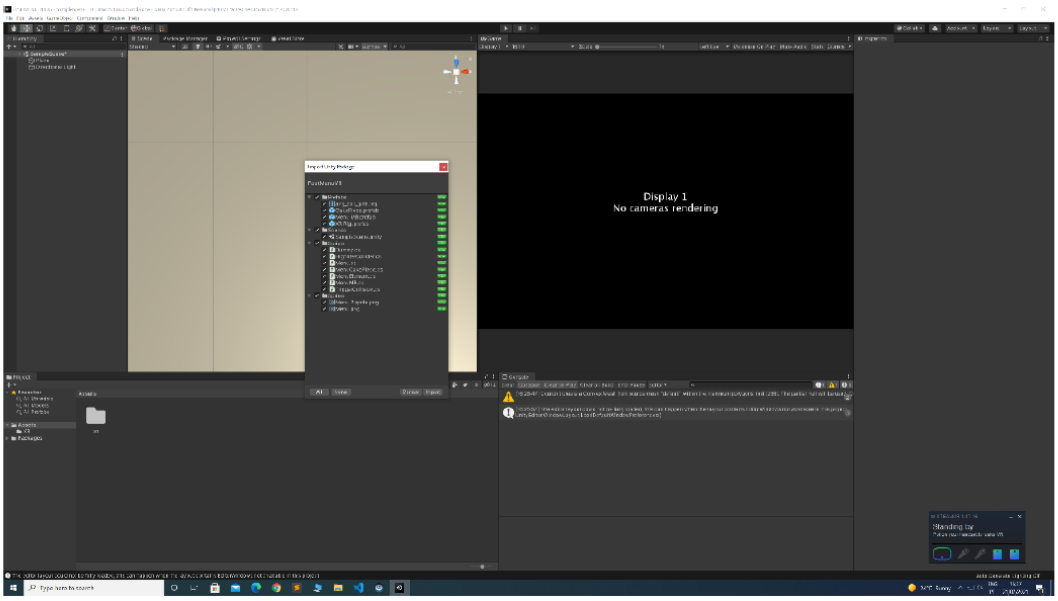
1. Carregar com o botão direito do rato, selecionar "Create", de seguida "FootMenu" e por fim "Element".
   Após este passo irá aparecer o objecto criado;
2. Selecionar o objecto criado e, no inspector, inseirir o nome para o elemento, adicionar o icon (sprite), tendo a opção de por o icon voltado para o centro, adicionar evento(s) a esse elemento e por fim indicar qual o menu que irá suceder-se após a selecção, ou não.

Configuração no projeto

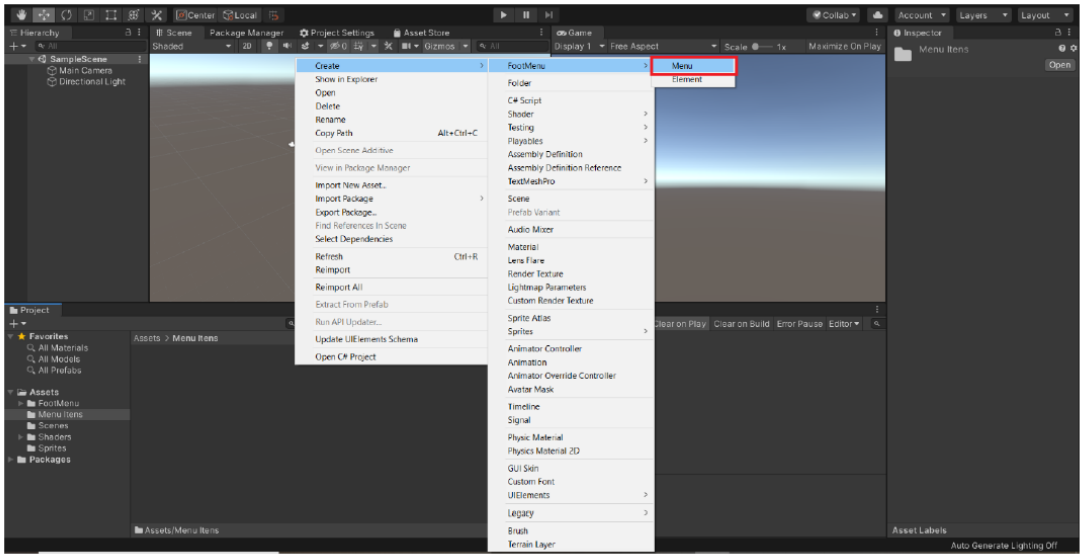
1. Aceder ao ficheiro "Assets/FootMenu/Prefabs/MenuPrefab" e, no inspector, na secção "Data" especificar qual o menu principal a ser executado, tendo tambem a opção de definir o espaçamento que os quadrantes irão ter ("Gap Width Degree").
2. Aceder ao ficheiro "Assets/FootMenu/Prefabs/MenuPrefab/ " e coloca-lo na hirearquia do projecto o objecto "MenuGO".
3. Selecionar o objecto anterior e, no inspector, na secção "Main Menu Prefab" adicionar o "MenuPrefab", definido no passo 1. Além disso, especificar o tipo de perspectiva que o utilizador irá experimentar (Direct / Indirect) e activar, caso desejar, a opção "Menu Following Player" para que o menu acompanhe o utilizador.

Tutorial (via imagem)

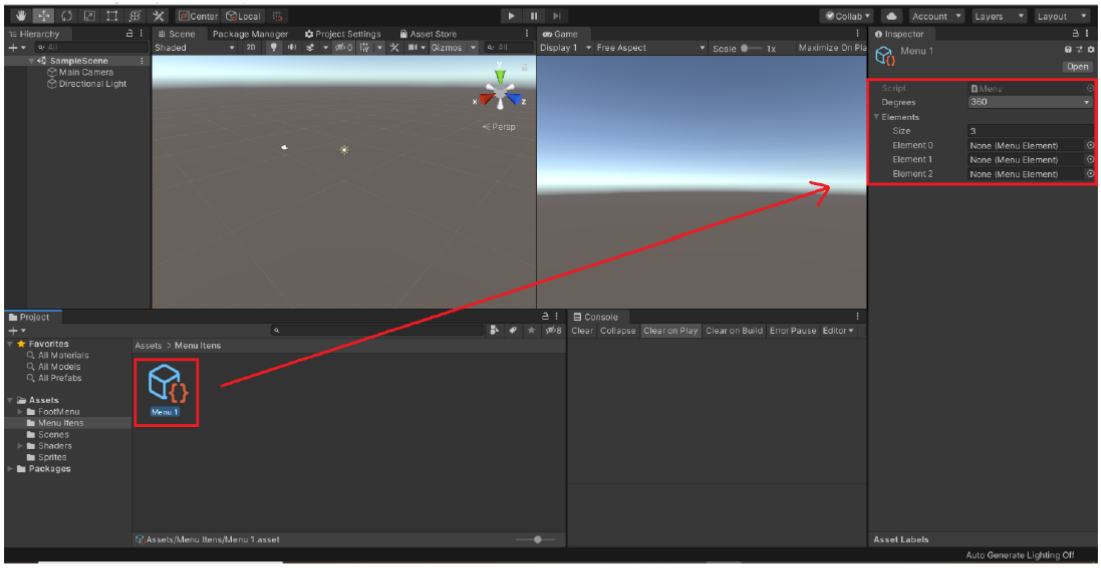
1. Importar a package para o projeto



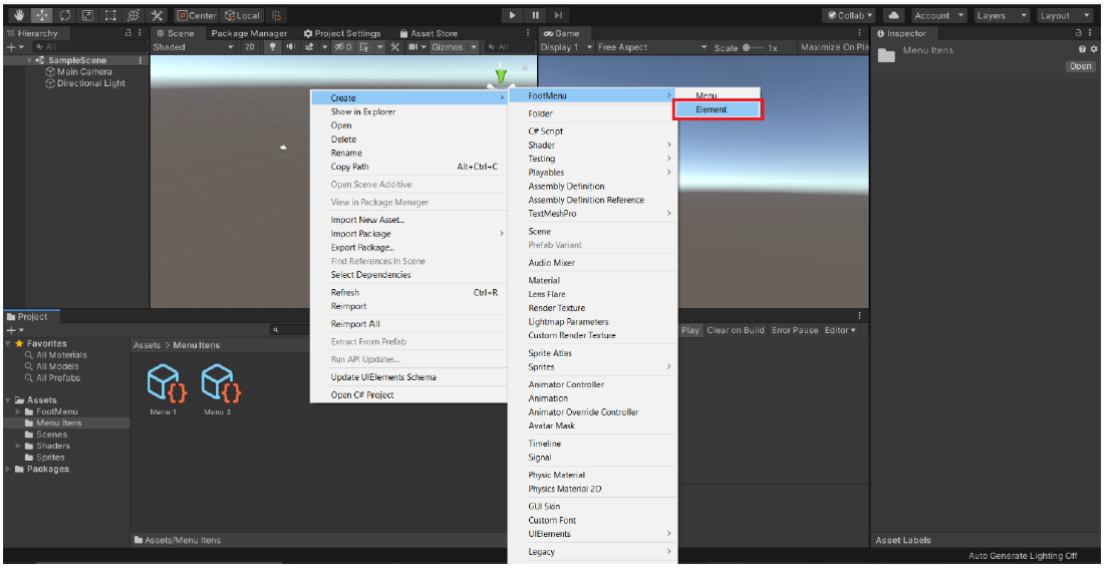
2. Com o botão direito do rato, selecionar “Create”, de seguida “FootMenu” e por fim “Menu”. Ao realizar este passo, irá ser criado uma asset.



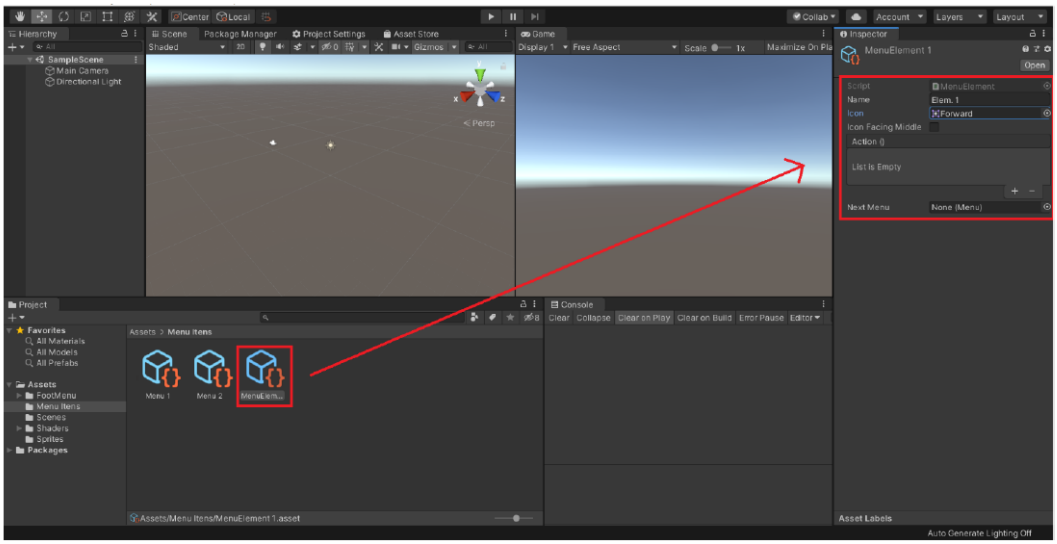
3. De seguida, selecionar a asset criada anteriormente e especificar as suas propriedades (tipo de menu [180 ou 360] e o numero de elementos que este menu irá ter). De notar que a secção dos elementos tem que ser preenchida depois da asset dos elementos estar criadada. (Nota: para criar vários menus basta repetir os passos 2 e 3)



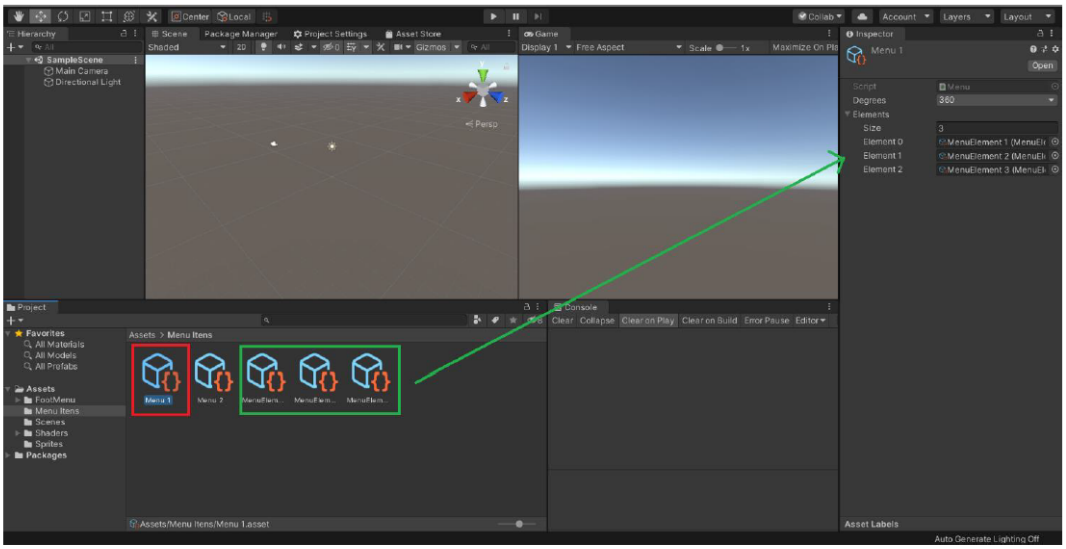
4. Agora passamos para a criação de um elemento. Para isso, há semelhança do menu, com o botão direito do rato, selecionar "Create", de seguida "FootMenu" e por fim "Element".



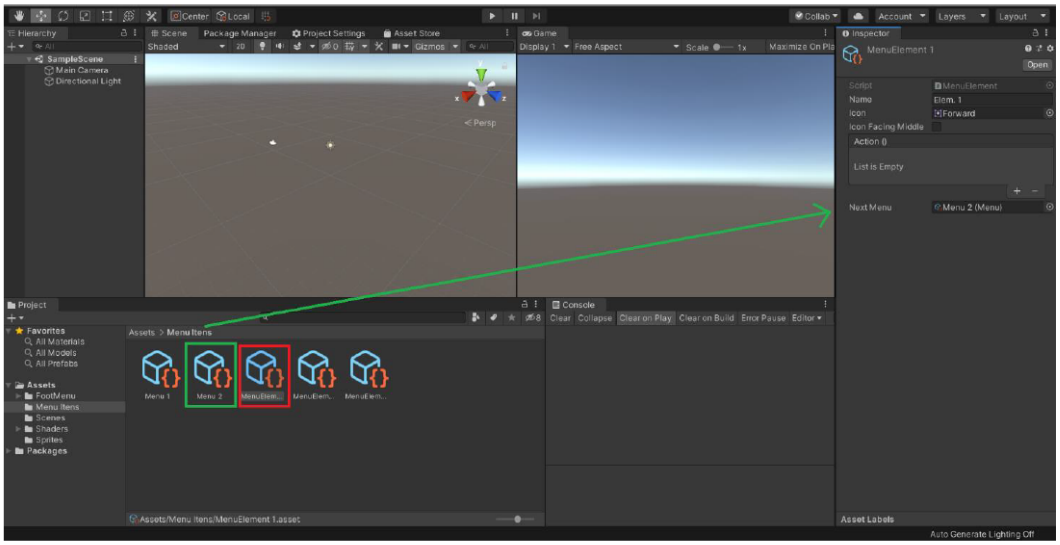
5. Depois da asset estar criada tem que se configurar. Para tal, seleciona-se a asset e no inspector poderá configurar as suas propriedades (nome, icon, evento e próximo menu a aparecer). (Nota: para criar vários elementos basta repetir os passos 4 e 5)



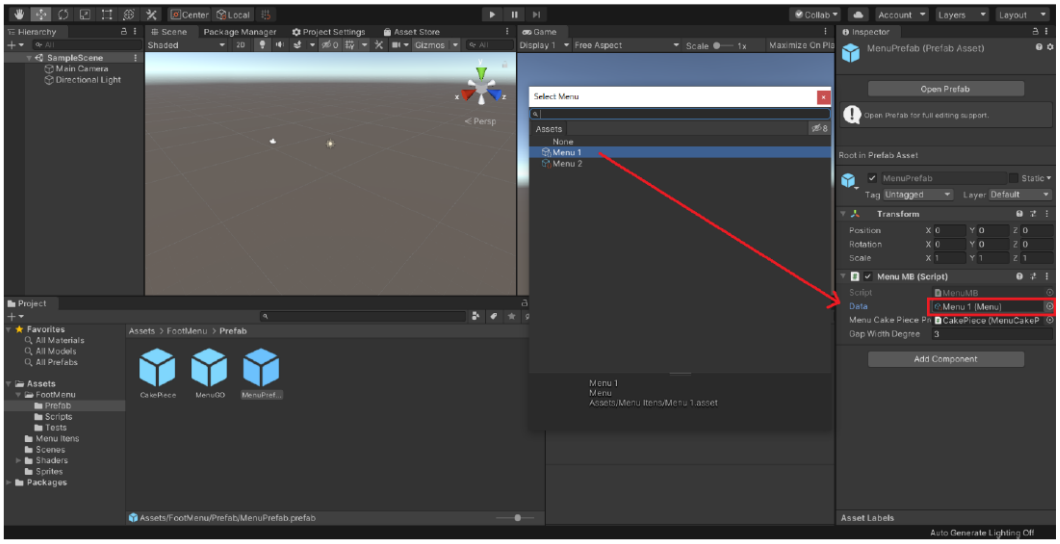
6. Após a realização dos passos anteriores, passamos então a associar os elementos ao menu. Desta forma, seleciona-se o menu que se quer adicionar os determinados elementos e, no inspector, adiciona-se os elementos desejados.



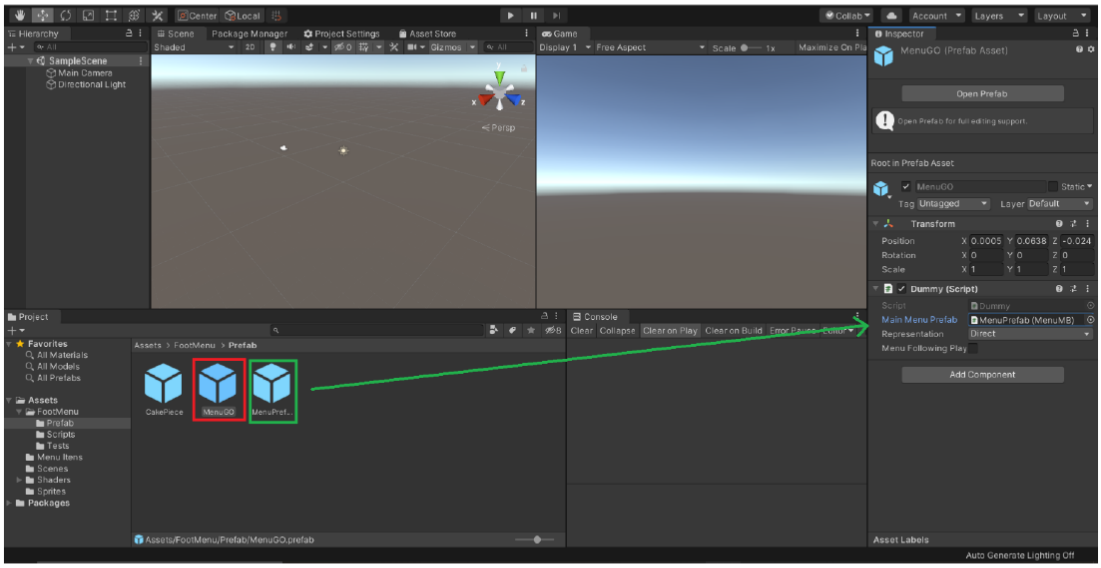
Bem como o menu que se sucede depois do determinado elemento ser selecionado.



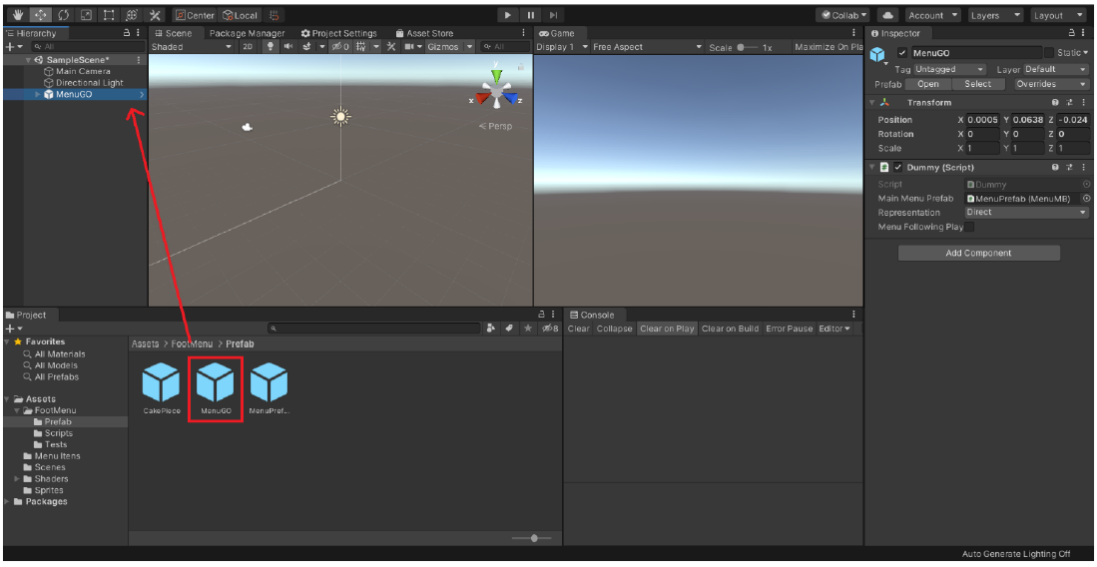
7. Após ter ligado os elementos ao menu e vice-versa, é necessário seleccionar o ficheiro Assets/FootMenu/Prefabs/MenuPrefab e indicar na secção "Data" qual o primeiro menu, isto é o Main Menu/Menu Principal, a ser executado, podendo tambem definir o espaçamento entre os quadrantes de cada elemento (Gap Width Degree).



8. De seguida, seleccionar o ficheiro Assets/FootMenu/Prefabs/MenuGO e especificar, no inspector, na seccção "Main Menu Prefab", o objecto configurado no passo 7 (MenuPrefab) que irá conter toda a informação necessária para executar o sistema. Para além disso, especificar que tipo de prespectiva é para ser utilizada (Direct ou Indeirect), bem como a opção do menu acompanhar o utilizador quando se desloca.



9. Por fim, basta arrastar o objecto "MenuGO" para a hieraquia do projecto para que se consiga executar o sistema de menus.



Esta página é deixada intencionalmente em branco.

Apêndice F

Fotos 360º retiradas para montagem da visita virtual

Neste apêndice é apresentado todas as fotografias 360º retiradas ao DEI para a montagem da visita virtual.

Figura F.1: Fotografia 360º utilizada para a 1º posição da visita virtual

Figura F.2: Fotografia 360º utilizada para a 2º posição da visita virtual

Figura F.3: Fotografia 360º utilizada para a 3º posição da visita virtual

Figura F.4: Fotografia 360º utilizada para a 4º posição da visita virtual

Figura F.5: Fotografia 360º utilizada para a 5º posição da visita virtual

Figura F.6: Fotografia 360º utilizada para a 6º posição da visita virtual

Figura F.7: Fotografia 360º utilizada para a 7º posição da visita virtual

Esta página é deixada intencionalmente em branco.

Apêndice G

Statment of Informed Consent

Neste apêndice é apresentado a declaração de consentimento que os participantes preencheram antes de realizarem os testes.

Statement of informed consent

PURPOSE OF THIS STUDY

The purpose of this study is to understand how people use feet to interact with VR. Your participation in this study will help me make the component easier to use.

☐ The researcher has explained the purpose of the research to me.

☐ I have had an opportunity to ask questions about the study.

FREEDOM TO WITHDRAW

Your participation in this study is voluntary.

- You can refuse to take part at any time.
- You can take a break at any time.
- You can ask questions at any time.

☐ I understand that I can leave at any time without giving a reason.

INFORMATION WE WILL COLLECT

We will ask you to show us how you use the product. We will watch how you do various tasks and we will ask you some questions. We will also record the session and we will take notes to record your comments and actions.

☐ I understand that people on the design team may be observing me during the research.

☐ I understand that my voice, my face and the computer screen will be video recorded.

PRIVACY AND CONFIDENTIALITY

We may watch the recording of your session so we can improve the product. No-one else will see the recordings. We may publish research reports that include your comments. The data used in these reports will be anonymous. This means you will not be identifiable and your comments will be confidential.

☐ I understand that people on the design team may view the recording in the future.

☐ I understand that my comments are confidential.

YOUR AGREEMENT

To take part in the research, please sign this form showing that you consent to us collecting these data.

Your name: Date:

Signature:

Figura G.1: Declaração de consentimento

Esta página é deixada intencionalmente em branco.

Apêndice H

Questionário de Usabilidade

Neste apêndice é apresentado os questionários que os participantes preencheram após a realização de cada experiência.

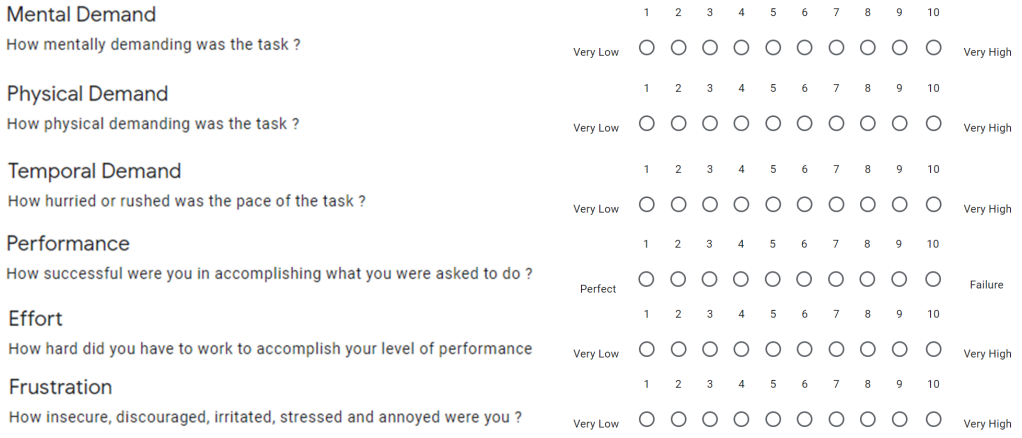
SIMULATOR SICKNESS QUESTIONS

- General discomfort
  - None
  - Slight
  - Moderate
  - Severe
- Fatigue
  - None
  - Slight
  - Moderate
  - Severe
- Headache
  - None
  - Slight
  - Moderate
  - Severe
- Eye strain
  - None
  - Slight
  - Moderate
  - Severe
- Difficulty focusing
  - None
  - Slight
  - Moderate
  - Severe
- Sweating
  - None
  - Slight
  - Moderate
  - Severe
- Nausea
  - None
  - Slight
  - Moderate
  - Severe
- Difficulty concentrating
  - None
  - Slight
  - Moderate
  - Severe
- « Fullness of the Head »
  - None
  - Slight
  - Moderate
  - Severe
- Blurred vision
  - None
  - Slight
  - Moderate
  - Severe
- Dizziness with eyes open
  - None
  - Slight
  - Moderate
  - Severe
- Dizziness with eyes closed
  - None
  - Slight
  - Moderate
  - Severe
- Vertigo
  - None
  - Slight
  - Moderate
  - Severe
- Stomach awareness
  - None
  - Slight
  - Moderate
  - Severe
- Burping
  - None
  - Slight
  - Moderate
  - Severe

SYSTEM USABILITY SCALE

		Strongly disagree				Strongly agree
1.	I think that I would like to use this system frequently	1	2	3	4	5
2.	I found the system unnecessarily complex	1	2	3	4	5
3.	I thought the system was easy to use	1	2	3	4	5
4.	I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5.	I found the various functions in this system were well integrated	1	2	3	4	5
6.	I thought there was too much inconsistency in this system	1	2	3	4	5
7.	I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8.	I found the system very cumbersome to use	1	2	3	4	5
9.	I felt very confident using the system	1	2	3	4	5
10.	I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

NASA TLX



PRODUCT REACTION CARDS

- Boring
- Busy
- Calm
- Cheap
- Creative
- Cutting-edge
- Exciting
- Expensive
- Familiar
- Fresh
- Impressive
- Innovative
- Inspiring
- Intimidating
- Old
- Professional
- Trustworthy
- Unprofessional

Esta página é deixada intencionalmente em branco.

Apêndice I

Resultados do Questionário SSQ

Neste apêndice é apresentado as respostas dos participantes ao questionário *Simulator Sickness*

	General discomfort	Fatigue	Headache	Eye strain	Difficulty focusing	Sweating	Nausea	Difficulty concentrating	« Fullness of the Head »	Blurred vision	Dizziness with eyes open	Dizziness with eyes closed	Vertigo	Stomach awareness	Burping
Participante nº1	Slight	None	None	None	Slight	None	None	None	None	Slight	None	None	Slight	Slight	None
Participante nº2	Slight	None	None	None	None	None	None	None	None	None	None	Slight	None	None	None
Participante nº3	None	None	None	None	None	None	None	None	None	None	None	None	Slight	Slight	None
Participante nº4	None	None	None	None	Slight	None	None	None	None	Slight	None	None	None	None	None
Participante nº5	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
Participante nº6	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
Participante nº7	None	None	None	Slight	Slight	None	None	None	None	Slight	None	None	None	None	None
Participante nº8	Slight	Slight	None	Slight	Moderate	None	Slight	Slight	Slight	Slight	None	None	None	Slight	None
Participante nº9	Slight	None	None	None	Slight	None	None	None	None	None	None	None	Moderate	None	None
Participante nº10	None	None	None	None	None	Slight	None	None	None	Slight	None	None	None	None	None

Figura I.1: Respostas obtidas do questionário SSQ para o modo *Direct*

	General discomfort	Fatigue	Headache	Eye strain	Difficulty focusing	Sweating	Nausea	Difficulty concentrating	« Fullness of the Head »	Blurred vision	Dizziness with eyes open	Dizziness with eyes closed	Vertigo	Stomach awareness	Burping
Participante nº1	Slight	None	None	None	Slight	None	None	None	None	Slight	None	None	Slight	Slight	None
Participante nº2	Slight	None	None	None	None	None	None	None	None	None	None	Slight	None	None	None
Participante nº3	None	None	None	None	None	None	None	None	None	None	None	None	Slight	Slight	None
Participante nº4	None	None	None	None	Slight	None	None	None	None	Slight	None	None	None	None	None
Participante nº5	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
Participante nº6	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
Participante nº7	None	None	None	Slight	Slight	None	None	None	None	Slight	None	None	None	None	None
Participante nº8	Slight	Slight	None	Slight	Moderate	None	Slight	Slight	Slight	Slight	None	None	None	Slight	None
Participante nº9	Slight	None	None	None	Slight	None	None	None	None	None	None	None	Moderate	None	None
Participante nº10	None	None	None	None	None	Slight	None	None	None	Slight	None	None	None	None	None

Figura I.2: Respostas obtidas do questionário SSQ para o modo *Indirect*