Daniel de Oliveira Murteira

Licenciado em Engenharia Informática

Colaboração em Realidade Aumentada

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Orientador: Nuno Manuel Robalo Correia, Professor Catedrático, Universidade Nova de Lisboa

Co-orientadores: Fernando Pedro Reino da Silva Birra, Professor Auxiliar, Universidade Nova de Lisboa

João Carlos Gomes Moura Pires, Professor Auxiliar, Universidade Nova de Lisboa

Júri

Presidente: Prof. Doutor Pedro Manuel Corrêa Calvente de Barahona
Arguente: Profª Doutora Anabela Gonçalves Rodrigues Marto
Vogal: Prof. Doutor Nuno Manuel Robalo Correia

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Outubro, 2020

Colaboração em Realidade Aumentada

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Para os meus pais

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus orientadores por me terem introduzido e cativado para o desenvolvimento do tema desta dissertação. Em especial, gostaria de agradecer ao professor Nuno Correia e ao professor Fernando Birra pela paciência e pelo apoio contínuo ao longo do projeto.

Agradeço também à Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, e em especial ao Departamento de Informática, por me terem fornecido todas as condições necessárias para a elaboração deste documento e pelos anos de formação que lhe antecederam.

Agradeço à equipa da ArticaCC pelo acompanhamento, pela constante proposta de novas ideias e pela contribuição com os elementos gráficos do projeto. Um obrigado especial ao André Almeida, ao Filipe Barbosa, ao Tarquínio Mota e ao Ricardo Imperial, com os quais trabalhei mais de perto.

Agradeço aos meus amigos e a todas as pessoas que me acompanharam e ajudaram ao longo do curso.

Por último, quero agradecer à minha família, e em especial aos meus pais e ao meu irmão, pelo apoio incondicional e independente de qualquer contratempo. A finalização deste ciclo só assim se tornou possível.

Resumo

A contínua evolução das tecnologias móveis, combinada com os avanços no campo da visão computacional, está a gerar um crescimento consequente na área da Realidade Aumentada. O aparecimento de novas plataformas comerciais, de suporte à tecnologia, torna o desenvolvimento de novas aplicações mais acessível para os programadores, sendo que o vasto número de dispositivos móveis em circulação simplifica o processo de adoção e divulgação deste tipo de conteúdos. Ainda assim, e apesar do âmbito da Realidade Aumentada englobar diversas áreas aplicacionais, grande parte das soluções existentes atualmente estão direcionadas à utilização individualizada da tecnologia.

O objetivo desta dissertação é explorar os paradigmas de interação em grupo aplicados num contexto de Realidade Aumentada. Foi concebida e realizada uma infraestrutura de colaboração com base nas tecnologias, paradigmas e padrões já existentes na área, bem como duas aplicações de teste para analisar a solução proposta.

As aplicações foram inspiradas em dois padrões de jogos convencionais e o seu desenvolvimento foi feito com base na infraestrutura concebida nesta dissertação. Por fim, os resultados obtidos visam o estudo dos parâmetros de escalabilidade, modularidade, manutenção e extensibilidade da solução.

Palavras-chave: Realidade Aumentada, Colaboração, Dispositivos Móveis, Interação Pessoa-Máquina

Abstract

The continuous evolution of mobile technology combined with recent advances in the computer vision field is creating a consequential growth in the area of Augmented Reality. The introduction of new commercial support for the technology allows developers to create new apps more easily, whilst the huge number of handheld devices in circulation simplifies the adoption and propagation processes for this kind of content. Nevertheless, although Augmented Reality includes very diverse application areas, the majority of existing solutions today is aimed at single user usage of the technology.

The goal of this thesis is to explore groupware interaction paradigms applied in the scope of Augmented Reality. A collaboration framework was designed based on current technologies and patterns available in the field. In addition to the framework, two test applications were also developed to evaluate the proposed solution.

The applications were inspired by two conventional game patterns and their development was done using the framework developed in this thesis. Lastly, the obtained results studied the parameters of scalability, modularity, maintenance and extensibility of the proposed solution.

Keywords: Augmented Reality, Collaboration, Handheld Devices, Human-Computer Interaction

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O estudo das técnicas de interação entre o ser humano e a máquina é uma área de pesquisa relativamente abrangente e em constante evolução. A introdução dos *smartphones* no quotidiano das pessoas não só veio abrir novas possibilidades de estudo na área como está a mudar a forma como as pessoas interagem entre si e com o mundo que as rodeia.

Um problema específico das técnicas de interação pessoa-máquina é a dificuldade em estabelecer metáforas suficientemente expressivas entre objetos reais e as suas representações no mundo virtual. A Realidade Aumentada visa resolver este problema, fazendo uso do ambiente real e sobrepondo-lhe camadas de informação virtual adicionais.

A evolução de áreas tecnológicas relacionadas com a Realidade Aumentada tem proporcionado um crescimento consequente da tecnologia. As técnicas de processamento de imagem, a capacidade de processamento dos computadores, a precisão dos sensores, as tecnologias de rede e a eficiência energética são campos de estudo essenciais no escopo da Realidade Aumentada. Para além disso, os recentes progressos nos *kits* de desenvolvimento de *software* da Apple[1] e da Google[2] sugerem que estamos à beira da disponibilização generalizada desta tecnologia, nomeadamente para uso em aplicações para o grande público.

1.1 Motivação

O avanço tecnológico e a imersão proporcionada pelos ambientes de Realidade Aumentada criam novos desafios por resolver. Uma área ainda pouco explorada neste âmbito é a das experiências partilhadas [7] [13] [22], onde vários utilizadores partilham um mesmo mundo e podem interagir entre si através desse mundo.

[1] ARKit, https://developer.apple.com/arkit/
[2] ARCore, https://developers.google.com/ar/

O rápido crescimento da tecnologia, com as suas debilidades inerentes, e a existência de uma plataforma de fácil acesso para o grande público proporcionam a oportunidade ideal para aprofundar o estudo dos métodos de interação em tarefas colaborativas no campo da Realidade Aumentada.

Aplicações colaborativas podem ser desenvolvidas nas diversas áreas aplicacionais da Realidade Aumentada, conferindo um grau de relevância assinalável para o estudo proposto. Algumas das áreas aplicacionais mais relevantes do campo são as do entretenimento, do marketing, da arquitetura, da educação, da medicina e do turismo. Pegando em exemplos concretos, há o sucesso do Pokémon GO[3], na indústria do entretenimento, os ambientes de simulação cirúrgica [36], na área da medicina e da educação, ou as aplicações que sobrepõem conteúdo informativo sobre os pontos históricos de uma cidade, no turismo [26] [40].

Para além dos desafios impostos pelo desenvolvimento de métodos para experiências partilhadas, a implementação de aplicações de Realidade Aumentada em dispositivos móveis é também ela um desafio. Apesar dos avanços tecnológicos dos últimos anos, continua a ser preciso equilibrar as questões de desempenho e de consumo quando se faz uso de técnicas de computação mais exigentes, como os algoritmos de processamento de imagem e a sincronização de dados entre diferentes dispositivos.

1.2 Descrição e Contexto

Esta dissertação enquadra-se no âmbito do projeto "Interactive New Technologies Regarding Augmentation in a Computerized Taxonomy" (INTERACT), liderado por uma equipa multidisciplinar da empresa ArticaCC. Este projeto visa a investigação de conhecimentos emergentes em áreas de estudo como a Inteligência Artificial, a Realidade Aumentada, Virtual e as Interações Homem-Máquina para obter soluções tecno-artísticas não convencionais e interativas entre grupos de indivíduos.

O que se propõe nesta dissertação é o estudo de novos paradigmas de interatividade colaborativa através de dispositivos que façam uso de técnicas de Realidade Aumentada. Para tal, foram desenvolvidas duas aplicações móveis que serviram como casos de estudo para testar e analisar diferentes tipos de interação. O objetivo principal foi o de desenvolver uma infraestrutura de colaboração e interação modular, aproveitando os padrões das ferramentas de comunicação e de Realidade Aumentada já existentes.

A interação individual entre o homem e a máquina tem sido alvo de investigação extensiva, encontrando-se bem estabelecidas classificações e taxonomias que abrangem este nível de interação. Embora existam alguns estudos focados na interação com grupos de pessoas, a investigação realizada tem-se focado na exploração do conceito de inteligência coletiva, que se refere a atividades inteligentes realizadas em grupo. Nos últimos anos, um novo tipo de inteligência coletiva emergiu, onde grupos interconectados de

[3] https://www.pokemongo.com/pt-pt/

pessoas e computadores realizam, coletivamente, ações inteligentes [29]. O trabalho realizado nesta dissertação visa trazer um acrescento para a área, explorando novas formas de colaboração entre grupos de pessoas e computadores.

Com base na infraestrutura de colaboração desenvolvida, poderão futuramente ser efetuados estudos para classificar comportamentos de grupo comuns tendo em vista a sua integração em ambientes interativos multiutilizador. Outro objeto de estudo associado com o trabalho desenvolvido é o da exploração de hipóteses científicas relacionadas com as relações entre as pessoas (cognoscentes) e os objetos (cognoscíveis), associadas ao nível de teorias do conhecimento, no âmbito da Realidade Aumentada.

Os avanços tecnológicos explorados na totalidade do projeto INTERACT visam à definição de novos algoritmos de interação com o desenvolvimento de funcionalidades associadas à emoção, de modo a conferir um maior grau de simbiose interativa entre um grupo de pessoas e a máquina. Serve, para isso, consolidar o conhecimento de modelação e arquitetura sobre a aplicabilidade dos diferentes sistemas de interação para grupos num contexto mediado por interfaces computacionais, como é o caso dos sistemas de Realidade Aumentada. Ao nível de classificação e caracterização do mundo que rodeia o espaço interativo, os avanços visam atingir a criação de soluções mais imersivas e envolventes entre pessoas, com experiências de utilização com maior realismo e impacto social. A ideia base é interligar as pessoas com o mundo que as rodeia através de elementos virtuais partilhados e interativos.

1.3 Solução Proposta

A abordagem proposta para o desenho da infraestrutura a desenvolver passa por dividir o problema nas suas diversas áreas técnicas de aplicação. Nomeadamente, são exploradas as componentes de reconhecimento e rastreamento de conteúdos, de visualização e representação de objetos virtuais, de comunicação e partilha de informação entre vários utilizadores, de sincronização de grupos e de armazenamento do fluxo de dados gerado pela infraestrutura.

Cada subproblema é resolvido individualmente, como um módulo - ou uma camada - da infraestrutura, antes de ser feita a sua integração com o resto das componentes do sistema.

A solução final é apresentada na forma de uma interface de programação de aplicações modular e extensível, que é posteriormente usada na implementação das duas aplicações de teste concebidas no âmbito desta dissertação.

1.4 Contribuições

As principais contribuições científicas desta dissertação são:

- **Conceção de um modelo de colaboração em Realidade Aumentada**

  São conciliadas técnicas de sincronização de grupo com técnicas de Realidade Aumentada para conceber uma infraestrutura de colaboração dentro do campo da Realidade Aumentada.

- **Implementação do modelo**

  O modelo teórico é reproduzido numa arquitetura de *software* extensível e modularizável no contexto da Realidade Aumentada. Essa arquitetura é ainda testada e implementada tendo por base duas aplicações de teste concretas, desenvolvidas para suportar a análise e verificação do modelo apresentado.

- **Avaliação do modelo com utilizadores e testes de sistema**

  São aplicados métodos de avaliação quantitativos e qualitativos sobre o modelo considerado, de forma a retirar conclusões e direções a seguir em trabalhos futuros.

1.5 Organização do Documento

O presente documento seguirá a seguinte estrutura:

- **Introdução**

  Apresenta o contexto geral do problema e apresenta as motivações para a realização da dissertação. São ainda delineadas as contribuições esperadas como resultado final.

- **Realidade Aumentada**

  Introduz brevemente os conceitos basilares da Realidade Aumentada. Para além da sua definição e aplicabilidade em ambientes móveis, são apresentadas as diferentes tecnologias de visualização e técnicas de rastreamento existentes.

- **Trabalho Relacionado**

  São aglomerados os trabalhos já existentes dentro do escopo desta dissertação. É primeiro feita uma revisão de colaborações inseridas na área de Realidade Aumentada e porque é que a ligação é relevante. São depois apresentados os métodos de interação existentes na área e os sistemas desenvolvidos num âmbito semelhante ao desta dissertação. Por fim, são identificadas as metodologias para avaliar o trabalho a realizar.

- **Solução Proposta**

  É feito o enquadramento dos objetivos e requisitos que o desenho da solução deverá cumprir. É apresentada a arquitetura deste desenho, as interfaces de programação desenvolvidas e as respetivas funcionalidades. São também especificados os conceitos das aplicações a desenvolver com base na infraestrutura proposta.

- **Implementação**

  São descritos os detalhes de implementação da solução e das aplicações de teste. É feita a contextualização com as tecnologias de suporte utilizadas no processo. Para terminar, é ainda feita uma pequena análise crítica dos detalhes de implementação que poderiam ser melhorados.

- **Testes de Validação**

  São expostos os resultados dos testes realizados.

- **Conclusões**

  São apresentadas as conclusões do documento e são sugeridas algumas melhorias a considerar para trabalho futuro.

Capítulo 2

Realidade Aumentada

Este capítulo introduz os conceitos gerais de Realidade Aumentada (RA). Para além da definição do conceito, é apresentada a sua utilização em dispositivos móveis, assim como as técnicas de visualização e rastreamento mais referenciadas na literatura. Por fim, é delineado o processo necessário para registar um objeto virtual numa cena representativa do mundo real.

2.1 Definição

De acordo com Azuma [3], um sistema de Realidade Aumentada é definido como um conjunto de três características:

- Combina o mundo real e o virtual;
- É interativo em tempo real;
- É registado em 3D.

Estas três características definem também os requisitos técnicos de um sistema de RA. Nomeadamente, ter um ecrã que combine imagens reais com virtuais, possuir uma componente computacional capaz de gerar gráficos interativos em resposta às ações dos utilizadores e ser dotado dum sistema de rastreamento que permita interpretar o ponto de vista do utilizador de modo a sobrepor os objetos virtuais no mundo real [7].

Em 1994, Milgram e Kishino [31] introduziram os conceitos de Realidade Mista e do Continuum de Virtualidade (figura 2.1). Realidade Mista é a abstração que reúne elementos de Ambientes Virtuais (AV) com elementos do mundo real, no qual se inclui a Realidade Aumentada. Nos dias de hoje, o termo mais popularmente utilizado para Ambientes Virtuais é o de Realidade Virtual (RV). Ao contrário da RA, a RV imerge

completamente um utilizador dentro de um ambiente virtual, onde existem bastantes limitações para interagir com o mundo exterior. Por seu lado, a RA serve como suplemento à realidade, em vez de a substituir.

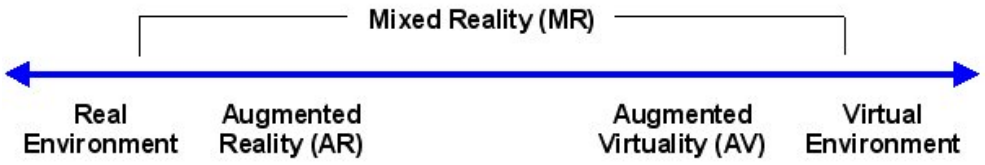

Figura 2.1: Continuum de Virtualidade de Milgram [31]

2.2 Realidade Aumentada Móvel

Os avanços da tecnologia móvel possibilitaram o uso de RA em dispositivos móveis. A melhoria nos recursos computacionais, das câmaras incorporadas, dos sensores utilizados e da computação em nuvem têm sido essenciais para o avanço da tecnologia. Melhorias nas interfaces de interação pessoa-máquina, na compreensão de cena, nas técnicas de visão computacional e nas técnicas de cache de rede e comunicação entre dispositivos alteraram a maneira como adquirimos, interagimos e mostramos informação do mundo em nosso redor. Segundo Chatzopoulos et al. [11], uma aplicação móvel pode ser considerada de RA se tiver as seguintes três características:

- **Canal de Entrada**: considera vários sensores do dispositivo (câmara, giroscópio, microfone, GPS), assim como qualquer eventual dispositivo emparelhado.
- **Canal de Processamento**: determina o tipo de informação a representar no ecrã. Para o fazer poderá ter que aceder a uma base de dados local (na memória do dispositivo) ou remota (na *Cloud*).
- **Canal de Saída**: projeta os objetos representados no ecrã do dispositivo móvel em conjunto com a vista do mundo real (isto é, aumenta a realidade do utilizador).

Um típico sistema de RA móvel inclui plataformas de computação móvel, infraestruturas de *software*, suporte de reconhecimento e rastreamento de imagem, o ecrã do dispositivo, comunicação sem fios e capacidade de gestão de dados. Um objeto virtual pode ser um simples texto, uma forma 2D, uma forma 3D ou até um vídeo. Cada objeto pode ser associado com um local, com outros objetos ou com utilizadores móveis. A potencial quantidade de objetos virtuais é enorme e isto torna impossível (e um desperdício) que um dispositivo móvel os guarde a todos em memória local. Nestes casos a aplicação é assistida por soluções DBaaS (*Database-as-a-Service*) e implementa algoritmos de cache e de pesquisa de modo a garantir que os objetos virtuais necessários estão guardados localmente.

Pela sua alta portabilidade, aceitação social e nível de intrusão diminuto, o telemóvel tornou-se a plataforma predominante para desenvolver sistemas de RA móvel. No entanto, apesar dos avanços computacionais dos telemóveis, o seu desempenho para aplicações em tempo real ainda é limitado [11]. A maioria dos telemóveis estão equipados com pequenas caches e acessos à memória relativamente lentos. As câmaras incorporadas têm problemas a lidar com campos de visualização encurtados. Os acelerómetros e os magnetómetros ainda são propensos a erros de imprecisão. No desenvolvimento de uma aplicação, ainda devem ser tomadas em especial consideração questões como a eficiência energética e o desempenho em tempo de execução.

2.3 Tecnologias de Visualização

Os ecrãs utilizados para técnicas de RA podem ser divididos em três grupos [2, 7, 11]: (1) ecrãs de vídeo, (2) ecrãs óticos transparentes, e (3) ecrãs de projeção. Os três casos são apresentados nas seguintes subsecções, na ordem enunciada.

2.3.1 RA Baseada em Vídeo

Sistemas de RA baseados em vídeo consistem na digitalização da vista do mundo real através de uma câmara de vídeo, de maneira a que as imagens do mundo real possam ser misturadas com as imagens virtuais. Normalmente a câmara de vídeo está incorporada na traseira do ecrã de um dispositivo móvel. Este tipo de ecrãs criam a ilusão de ver o mundo real através do ecrã e, portanto, são usualmente chamados de "ecrãs de vídeo transparentes". A figura 2.2 demonstra a estrutura do processo.

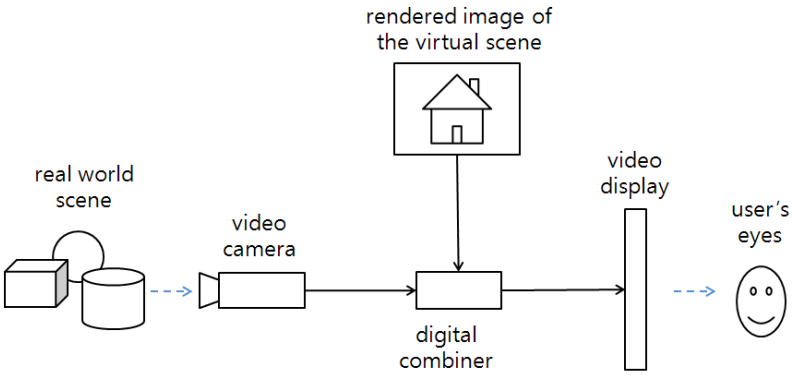

Figura 2.2: Estrutura de RA baseada em vídeo [7]

2.3.2 RA Baseada em Ecrãs Óticos Transparentes

Sistemas de RA baseados em ecrãs óticos transparentes utilizam misturadores óticos para combinar imagens virtuais com a vista do mundo real. Funcionam à base de lentes parcialmente transmissivas do mundo real, colocadas sobre a vista do utilizador. Estas

lentes são ligeiramente inclinadas para possibilitar a reflexão de imagens geradas a partir de um dispositivo de vídeo. A estrutura do processo está demonstrada na figura 2.3.

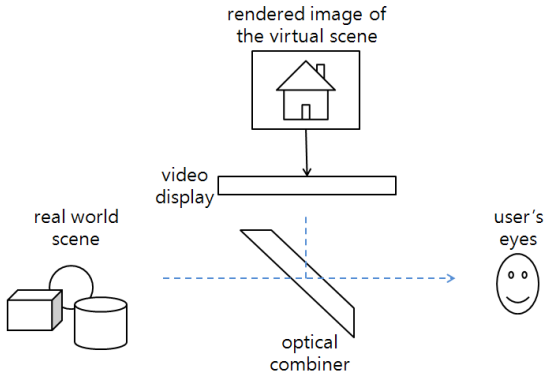

Figura 2.3: Estrutura de RA baseada em ecrãs óticos transparentes [7]

2.3.3 RA Baseada em Projeções

Sistemas de RA baseados em projeções sobrepõem imagens virtuais diretamente sobre uma superfície física real, seja esta superfície uma parede ou o modelo real do objeto virtual em consideração. Ao contrário dos outros dois sistemas apresentados, este não restringe o utilizador à utilização de um dispositivo que lhe permite receber e processar as imagens virtuais. Por outro lado, a visualização de objetos virtuais a três dimensões é mais limitada.

2.4 Técnicas de Rastreamento

Rastreamento é o mecanismo que permite identificar a posição e orientação de um objeto em relação ao ponto de vista da câmara. É o tópico de pesquisa mais popular do ISMAR (*International Symposium on Mixed and Augmented Reality*) por duas décadas consecutivas [22]. De acordo com Chatzopoulos et al. [11], as técnicas de rastreamento podem ser divididas em duas categorias: técnicas com sensores e técnicas com visão computacional. A primeira categoria utiliza sensores de orientação e localização para estimar a posição da câmara em relação ao mundo real. A segunda serve-se de algoritmos de processamento de imagem. Os métodos híbridos surgem da utilização em conjunto de técnicas das duas categorias.

Um conceito central ao rastreamento em RA são as formas como os objetos virtuais se podem mover pelo espaço. Existem no total seis graus de liberdade (6DOF, do inglês) no espaço tridimensional. Estes seis graus são divididos em movimentos de translação e movimentos de rotação ao longo dos três eixos do espaço. Mais especificamente, um objeto capaz de se movimentar com 6DOF pode movimentar-se nos eixos x, y e z, assim como pode mudar a sua orientação relativa as esses eixos através de movimentos de rotação (figura 2.4).

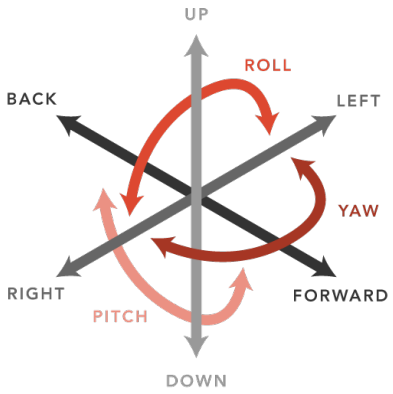

Figura 2.4: Seis graus de liberdade

As técnicas identificadas por Chatzopoulos são apresentadas nas seguintes subsecções.

2.4.1 Rastreamento com Sensores

O rastreamento com sensores divide-se em quatro subcategorias de sensores: inerciais, magnéticos, eletromagnéticos e ultrassons.

2.4.1.1 Inerciais

Sensores inerciais são sensíveis ao movimento e podem ser usados na maior parte das situações sem limitações de alcance. Os acelerómetros e os giroscópios são dois exemplos de sensores de inércia utilizados em dispositivos móveis. Os primeiros medem a aceleração não gravitacional do dispositivo, através das vibrações causadas pelas mudanças de velocidade do dispositivo. Os segundos servem para determinar a orientação do dispositivo.

2.4.1.2 Magnéticos

O rastreamento magnético utiliza o campo magnético da Terra para obter a orientação do dispositivo. É combinado com outros métodos de rastreamento para obter seis graus de liberdade nas três dimensões do espaço. Estes métodos têm problemas com interferências de campos magnéticos presentes no ambiente envolvente.

2.4.1.3 Eletromagnéticos

Métodos eletromagnéticos rastreiam a posição com base em parâmetros como o tempo de chegada, o indicador da força do sinal recebido ou a diferença de fases dos sinais eletromagnéticos. GPS, WiFi, RFID e Bluetooth são alguns exemplos da utilização destas técnicas.

2.4.1.4 Ultrassons

Um sensor ultrassónico emite sinais sonoros inaudíveis ao ser humano e é capaz de derivar a distância entre o dispositivo e um objeto (ou superfície) tendo em conta o tempo que levou a transmitir e a receber o sinal refletido. Não é um método muito utilizado em dispositivos portáteis pela sua sensibilidade a condições de oclusão ou de ruído do ambiente variáveis.

2.4.2 Rastreamento com Visão Computacional

O rastreamento com visão computacional divide-se em três subcategorias. Pode ser feito com a utilização de marcadores, com base nas características naturais da imagem ou utilizando estruturas 3D para identificar o espaço físico.

2.4.2.1 Com Marcadores

Métodos à base de marcadores utilizam marcadores - ou fiduciários - como pontos de referência para sobrepor os objetos virtuais gerados computacionalmente. A figura 2.5 mostra vários exemplos de marcadores que se podem utilizar. Para ser facilmente identificado, um marcador tem as suas propriedades geométricas predefinidas, tais como a forma, o tamanho ou o padrão de cores. Marcadores planares são os mais populares por registarem níveis de precisão superiores e maior capacidade para se adaptarem a condições de iluminação variáveis. Métodos à base de marcadores são mais indicados para aplicações num espaço fechado, uma vez que não é prático instalar e manter marcadores ao ar livre, em ambientes demasiado extensos ou desconhecidos. A oclusão é também um problema a ter em conta na utilização de marcadores.

Figura 2.5: Exemplos de RA com recurso a marcadores [7]

2.4.2.2 Através de Características Naturais da Imagem

Técnicas de rastreamento com base nas características naturais da imagem rastreiam pontos e regiões chave numa sequência de imagens para calcular as relações correspondentes e estimar a informação da pose do dispositivo. O rastreamento *frame* por *frame* é capaz de eliminar erros de tratamento de imagem para os quais a maior parte dos métodos à base de sensores têm dificuldades. O maior problema deste método é a carga de processamento

computacional que vem acrescentar, principalmente para o contexto de um dispositivo móvel. Algoritmos como SIFT, SURF, ORB, BRISK, AKAZE [44] ou a utilização de redes neuronais convolucionais são algumas das técnicas utilizadas para lidar com este tipo de problemas.

2.4.2.3 Com a Utilização de Modelos 3D

É também possível o rastreamento de objetos do mundo real com a utilização de uma estrutura 3D conhecida *à priori*, através de técnicas de reconhecimento baseadas nos contornos geométricos dos objetos [7].

SLAM (*Simultaneous Localization and Mapping*) [14] é uma das técnicas que permite simultaneamente criar e atualizar o modelo do ambiente real em conjunto com a localização do dispositivo quando inserido nesse ambiente. Outra técnica baseada em modelos é PTAM (*Parallel Tracking and Mapping*) [23], que surge com a separação das componentes de localização e mapeamento do SLAM e introduz melhorias de precisão e desempenho em relação ao seu antecessor. Ambos os métodos têm dificuldades a lidar com ambientes muito grandes e são, portanto, mais adequados para usar em pequenos espaços.

2.4.3 Métodos Híbridos

Cada método isoladamente tem as suas vantagens e limitações. Uma solução proposta é a de superar as limitações existentes de cada método através da utilização de vários em simultâneo. Rastreamento híbrido requer a fusão dos resultados de mais do que um método para obter a informação de pose do dispositivo. Métodos à base de sensores funcionam em "ciclo aberto". Isto é, os erros de rastreamento não podem ser avaliados e usados para correções adicionais. Métodos à base de visão computacional usam os resultados de rastreamento como *feedback* para corrigir os erros dinamicamente e, portanto, funcionam em "ciclo fechado". No entanto, a precisão destes métodos é inconsistente, uma vez que é sensível a oclusões ou a condições do ambiente demasiado variáveis.

Santos et al. [42] misturaram técnicas à base de sensores com técnicas à base de visão computacional para melhorar o tempo de processamento e a precisão nos processos de reconhecimento e rastreamento de uma imagem. Neste caso de estudo, verificou-se que o tempo médio de reconhecimento e rastreamento diminuiu consideravelmente com a utilização de sensores para estimar a posição do dispositivo em conjunto com algoritmos de processamento de imagem. A introdução dos sensores surge da incapacidade do algoritmo utilizado (AKAZE [44]) para processar todos os *frames* da câmara no contexto de uma aplicação móvel.

2.5 Registo

Azuma [3] definiu o processo de registo como um dos problemas com o potencial mais limitador para um sistema de RA. Este processo consiste no registo de objetos virtuais

na cena que está a capturar o mundo real. Tanto os objetos reais como os objetos virtuais precisam de estar precisamente alinhados em relação uns aos outros. Caso contrário, a ilusão que os dois mundos coexistem fica comprometida. Amin e Govilkar [1] descrevem este processo em três passos:

- **Reconhecimento**: deteção de uma imagem, objeto, corpo ou espaço no qual o objeto virtual será sobreposto.
- **Rastreamento**: rastreamento da localização do objeto virtual em tempo real.
- **Mistura (ou Representação)**: sobreposição do objeto virtual no mundo real.

Quando Azuma descreveu os problemas relacionados com este procedimento, foram evidenciados dois tipos de erros de registo: os estáticos e os dinâmicos. Os primeiros ocorrem mesmo quando o ponto de vista do utilizador e os objetos da cena permanecem imóveis. São sobretudo causados por erros de leitura dos sensores de rastreamento. Os erros dinâmicos, por outro lado, acontecem se houver movimento do ponto de vista ou dos objetos e são normalmente causados por atrasos de processamento computacional.

Capítulo 3

Trabalho Relacionado

Neste capítulo é feita a revisão bibliográfica nas áreas de colaboração e interação com RA. É apresentada a taxonomia de Brockmann [9], que fornece uma infraestrutura para classificar sistemas de RA colaborativa. São mencionadas as técnicas de interação existentes e as diretivas de desenho para o desenvolvimento de uma aplicação de RA. São ainda apresentados vários estudos da área nas vertentes de navegação e colaboração. Por fim, são referidas as técnicas de validação utilizadas para avaliar um sistema de RA.

3.1 Colaboração

Trabalho cooperativo suportado por computador (CSCW, do inglês) é o campo que lida com a maneira como atividades colaborativas e a sua respetiva coordenação podem ser suportadas por computadores. Este suporte pode ser obtido através de uma melhoria dos meios de comunicação, de possibilidades de monitorização melhoradas para os colaboradores ou pela redução da complexidade nas tarefas a realizar [10]. A colaboração visa uma liberdade de interação semelhante aquela que existe no mundo real, onde é possível partilhar objetos e informação sem quaisquer barreiras de discurso entre os intervenientes [5].

Em RA colaborativa, os utilizadores podem experienciar um espaço partilhado com objetos reais e virtuais à sua disposição. Para além disso, um meio de interação a três dimensões abre novas áreas de aplicação como a elaboração colaborativa de desenhos assistidos por computador [39]. Aplicações de RA colaborativas partilham tanto a vista do mundo como os dados inerentes à aplicação em questão [46]. A comunicação por rede é essencial para a sincronização dos elementos virtuais entre os diferentes dispositivos, ainda que a qualidade de serviço das tecnologias de rede seja tida como uma das grandes limitações dos sistemas móveis de RA [11].

Wang e Dunston [11] apresentaram um estudo comparativo entre a eficácia das tarefas de desenho colaborativo no papel e num sistema de RA, com configurações no mesmo local e em ambientes remotos. Neste estudo mostrou-se que o tempo de desempenho e o esforço mental em tarefas de desenho colaborativo pode ser significativamente melhorado em ambientes de RA.

Brockmann et al. [9], num estudo baseado em 237 artigos da área, definem uma infraestrutura (ou *framework*, em inglês) para a classificação de sistemas de RA colaborativos e das suas propriedades. A taxonomia definida é apresentada de seguida.

3.1.1 Taxonomia de Brockmann

A taxonomia proposta divide os sistemas de RA colaborativa em seis dimensões, com as diferentes manifestações de cada dimensão detalhadas em subcategorias. A primeira dimensão identificada foi a do **espaço**, que considera se as pessoas estão a interagir no mesmo espaço ou não. Esta dimensão divide-se em três atribuições: interação no mesmo local, interação remota e interação mista. A segunda dimensão identificada é **temporal**. Esta faz a distinção entre interações síncronas, assíncronas e mistas.

A terceira dimensão para representar aplicações de RA colaborativas é a da **mobilidade**, que distingue entre sistemas móveis, estacionários e mistos. Num sistema de RA móvel, um utilizador pode circular livremente enquanto utiliza a aplicação, ao passo que num sistema fixo ou estacionário, os utilizadores estão limitados a interagir num local específico. Num sistema misto, pelo menos um utilizador está limitado pelo local enquanto que os outros se podem movimentar livremente. Um exemplo deste caso é o de um sistema de RA remoto onde o utilizador remoto está a interagir com o ambiente através de um computador fixo. A quarta dimensão apontada pelos autores é a do **conteúdo virtual** representado. A manifestação mais comum desta dimensão é quando o conteúdo virtual é exclusivamente representado por objetos virtuais, que podem ser ou não a extensão de objetos existentes no mundo real. Em contrapartida, também é possível criar representações virtuais apenas tendo por base as pessoas presentes no meio ambiente, com a utilização de avatares. A combinação das duas abordagens também é possível e resulta na última subcategoria da dimensão.

A quinta dimensão tem a ver com a **diferenciação de papéis** atribuídos aos diferentes utilizadores da aplicação desenvolvida. Os utilizadores ou partilham a experiência como iguais ou são divididos em grupos com acesso a diferentes funcionalidades. A última dimensão endereça os **dispositivos de visualização** utilizados. Os dispositivos montados na cabeça (HMD, do inglês) na forma de óculos ou capacetes - que fazem uso de ecrãs óticos transparentes - são uma das opções. As outras opções consistem na utilização de dispositivos individuais de visualização (móveis ou estacionários) - através de RA baseada em vídeo - e na utilização de ecrãs públicos, à base de projetores ou de vídeo. O uso de mais do que um tipo destes dispositivos é a última opção, tendo já sido realizados estudos para este tipo de configuração.

Um dos requisitos no desenvolvimento da taxonomia foi que cada artigo referenciado como objeto de estudo se ajustasse concretamente a cada uma das subcategorias encontradas para as seis dimensões, de modo a evitar definições redundantes. É também de notar que não existem quaisquer dependências entre as seis dimensões e que o sistema desenvolvido permite a introdução de novas dimensões ou subcategorias na taxonomia. Uma limitação apontada pelas reflexões dos autores foi a falta de distinção entre os diferentes sistemas de *feedback* sensorial (háptico, olfatório, gustativo) que ainda podem ser explorados em sistemas de RA. No entanto, nenhuma aplicação de RA colaborativa que cobrisse tais requisitos foi identificada na altura.

Outra dimensão que não é referida e poderia ser acrescentada é a do **tamanho do grupo** a partilhar a experiência, uma vez que é um fator fundamental para a determinação da complexidade das soluções propostas. Esta dimensão seria dividida em interações de duas pessoas, de grupos pequenos e de maior dimensão. Em [45], é sugerido o número sete para distinguir entre grupos pequenos e grandes.

A figura 3.1 ilustra as seis dimensões da taxonomia descrita nesta secção, com o número de objetos de estudo referenciados em cada subcategoria.

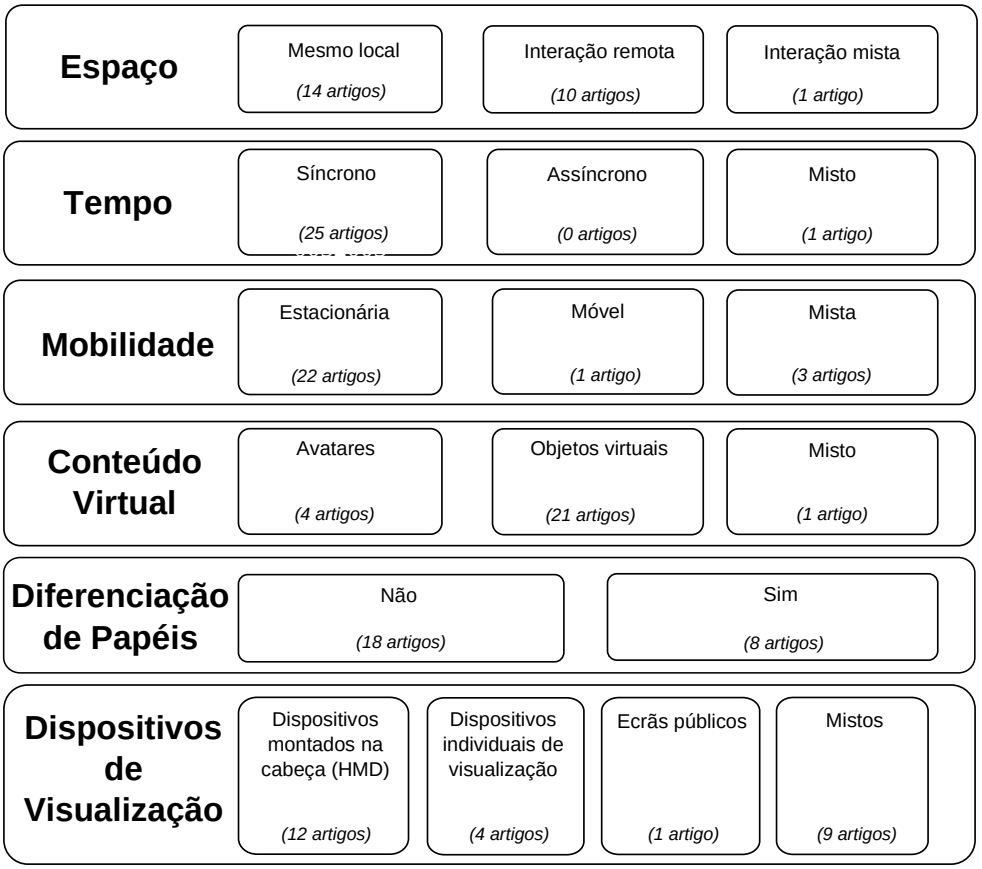

Figura 3.1: As seis dimensões na Taxonomia de Brockmann [9]

3.2 Métodos de Interação

Os métodos de interação consistem nas possíveis formas de comunicação entre os seres humanos e os sistemas de RA. O objetivo de qualquer um destes métodos visa facilitar a manipulação do conteúdo virtual, da forma mais intuitiva possível. Billinghurst et al. [7] dividem as interfaces de comunicação em cinco categorias diferentes, apresentadas de seguida.

- **Navegadores de Informação**

  Ecrãs de navegação de RA são considerados como uma janela de acesso a um espaço informativo. A tarefa principal do utilizador é a de manipular esta janela para pesquisar os conteúdos virtuais que mais lhe interessam. Por exemplo, com um *smartphone* o utilizador apenas precisa de se deslocar ou apontar o dispositivo em diferentes direções para manipular a janela de informação a que tem acesso. Outras maneiras de interagir com estes sistemas de RA passam por aplicar filtros sobre o conteúdo virtual disponível ou expandir os detalhes sobre um certo objeto virtual. Este tipo de interação normalmente é feito através de interfaces gráficas tradicionais, a duas dimensões, através de um ecrã táctil ou de um teclado.

  Este método de interação é um dos mais importantes e mais simples de aprender, uma vez que as pessoas já estão habituadas às interfaces de utilização móvel tradicionais. Por outro lado, é um método claramente limitado ao nível da manipulação dos objetos virtuais, cingindo-se apenas às tarefas mais básicas de visualização e de pesquisa.

- **Interfaces 3D**

  Uma possível forma de manipular objetos virtuais é através de técnicas tradicionais de interfaces 3D, como a seleção e a manipulação de objetos virtuais de três dimensões. Para tal, são usados dispositivos capazes de controlar estes objetos em qualquer eixo de rotação ou de translação (6DOF). Podem ser usados ratos 3D ou dispositivos apontadores de controlo (figura 3.2), por exemplo, da mesma maneira que estes são usados em ambientes puramente virtuais como os de RV.

  Ainda que este tipo de técnicas introduza a possibilidade de manipular objetos virtuais num sistema de RA, a diferenciação da interação dos utilizadores com os objetos reais e com os objetos virtuais é potencialmente problemática. Transportar um dispositivo de controlo apenas para utilizar em objetos virtuais poderá criar algumas barreiras aplicacionais no desenvolvimento de um sistema de RA.

Figura 3.2: Dispositivo apontador de controlo (HTC Vive)

- **Interfaces Tangíveis**

  Outra maneira de interagir com objetos virtuais é através do seu mapeamento em objetos físicos, do mundo real. O objeto físico é considerado uma interface tangível. Billinghurst et al. [6] definem as seguintes características para um sistema de RA que faça uso de interfaces tangíveis:

  - Todo o objeto virtual está registado num objeto físico;

  - A pessoa interage com os objetos virtuais ao manipular o objeto físico.

  A grande vantagem que as interfaces tangíveis vêm introduzir é a possibilidade de aplicar metáforas de interação entre os objetos do mundo físico e os objetos do mundo virtual.

- **Interfaces Corporais**

  Neste tipo de interfaces, o utilizador é capaz de controlar um objeto virtual através de movimentos naturais tais como a utilização de gestos. Ao contrário do que acontece com as interfaces 3D, neste caso a pessoa não tem a necessidade de recorrer a qualquer periférico de controlo para manipular os objetos virtuais. A interação entre a pessoa e o espaço virtual é reconhecida através de algoritmos de visão computacional.

- **Interfaces Multimodais**

  As interfaces multimodais são interfaces que visam aumentar a interatividade da experiência de RA ao combinar as diferentes modalidades de discurso disponíveis ao ser humano, como a voz e a linguagem gestual. Cohen et al. [12] demonstraram que a utilização da voz em conjunto com os gestos são complementares no campo da interação pessoa-máquina, uma vez que a voz pode agir como uma via de comunicação quantitativa enquanto que os gestos são ideais para uma interação qualitativa e mais minuciosa.

Para além dos diversos métodos de interação existentes, é ainda preciso tomar em consideração algumas diretivas de desenho e planeamento no desenvolvimento de aplicações de RA. Fatores como o esforço mental despendido ou a integração impercetível do conteúdo virtual no real são fundamentais para o êxito de um sistema de RA.

3.2.1 Diretrizes de Desenho

Billinghurst et al. [7] identificam três componentes fundamentais no desenho de aplicações de RA:

- Os objetos físicos;
- O conteúdo virtual;
- As metáforas de interação que juntam os elementos reais e virtuais.

O objetivo principal é criar uma ligação entre os comportamentos do utilizador no mundo real com aquilo que ele perceciona no mundo virtual.

Kruijff et al. [25] dividem os problemas de perceção com a utilização de um sistema de RA em três categorias: (1) distorções de cena, (2) distorções de profundidade e ordenação dos objetos, e (3) visibilidade. Tendo estes fatores em consideração, Rolim et al. [41] apresentam algumas soluções de desenho para criar instruções em RA. A primeira conclusão que retiram é a necessidade de demonstrar o movimento gerado por uma possível ação do utilizador, de modo a poder guiá-lo a proceder corretamente para obter o seu objetivo. A segunda diretiva sugere que se dê ênfase às partes de um objeto prestes a ser movido ou alterado. Isto é especialmente importante quando existe um elevado número de elementos em cena. Técnicas para destacar as fronteiras ou o grau de transparência dos objetos podem ser especialmente eficazes neste sentido. A terceira diretiva é uma das mais importantes para qualquer sistema de interação pessoa-máquina. Trata-se da capacidade do sistema de oferecer respostas instrutivas às ações dos utilizadores, designado por mecanismo de *feedback*. Através deste mecanismo, o utilizador poderá perceber se está a agir da maneira esperada ou se precisa agir de maneira diferente para alcançar um certo objetivo. Por fim, a gestão de oclusão e profundidade é especialmente importante para assegurar o realismo de uma aplicação de RA. No processo de desmontagem (separação de peças) a três dimensões de um objeto complexo, por exemplo, é importante adicionar sugestões visuais para informar o utilizador sobre as relações de profundidade, distância e oclusão entre as pequenas peças do objeto a ser desmontado.

Xu et al. [47] apresentam nove padrões de desenho aplicáveis a jogos de RA móvel. Estes padrões estão assentes sobre habilidades intrínsecas ao ser humano, como a consciência e conhecimento dos mesmos em relação aos próprios corpos, ao ambiente que os rodeia ou às normas de interação social em vigor [20]. Os padrões definidos neste estudo são resumidos de seguida:

- **Metáfora para o dispositivo**: metáforas de utilização com objetos do dia a dia ajudam os jogadores a perceberem que tipo de ações devem executar. Por exemplo, a interface de um *smartphone* pode ser comparada com uma câmara de filmar, um microscópio ou uma lente de um sistema de raios X. Assim, o utilizador pode transferir o conhecimento desses dispositivos para o tipo de ações que deve executar enquanto utiliza o *smartphone*.

- **Mapeamento das ações de controlo**: a maneira como os objetos virtuais são controlados através do dispositivo móvel deve ser analisada de antemão. Este controlo pode ser feito na interface do dispositivo móvel - com ações de seleção e manipulação no ecrã - ou pode ser feito através de interfaces tangíveis presentes no contexto espacial do jogo.
- **Desenho para omitir falhas**: um sistema de RA móvel está inevitavelmente limitado pela sua capacidade de processamento e pela precisão dos seus sensores. Um desenho aplicacional adequado passa por encobrir estas limitações, tomando medidas quando o sistema de rastreamento está perdido ou mesmo limitando o raio de ação dos utilizadores de maneira preemptiva.
- **Consistência do mundo virtual**: define se os elementos virtuais aderem às leis da física do mundo real ou não. No contexto de um jogo menos realista poderá ser interessante subverter essas leis para surpreender ou desafiar o utilizador.
- **Utilização de pontos de referência**: pontos de referência geográficos (como uma igreja ou um monumento) são ferramentas de orientação importantes num contexto de navegação. Misturar locais físicos de referência com informação digital ajuda os jogadores a situarem-se no espaço físico e virtual.
- **Presença**: o sentimento de presença dos jogadores no espaço virtual é importante para obter uma experiência mais imersiva.
- **Animações**: personagens virtuais independentes (como monstros ou animais) que reajam às ações externas dos utilizadores com comportamentos plausíveis têm a capacidade de tornar a experiência do utilizador mais imersiva.
- **Limitações de movimento**: as ações de um jogador podem restringir o leque de opções de outro jogador, da mesma maneira que um jogador de xadrez fica limitado pelas jogadas do oponente. Num sistema de RA, o posicionamento dos jogadores poderá ser especialmente relevante para este tipo de interações.
- **Informação escondida**: a informação escondida que pode ser parcialmente revelada consoante o comportamento dos jogadores tem potencial para alimentar estratégias de jogo sociais.

Xu et al. [47] sugerem ainda que um sistema de RA móvel deve: (1) suportar interações a uma mão, (2) certificar-se que pelo menos uma superfície rastreável está em cena, e (3) não encorajar movimentações demasiado rápidas da câmara.

Por fim, Ko et al. [24] estudaram a utilização de princípios de usabilidade para desenvolver e avaliar aplicações de RA móvel. Foram analisados vinte e dois princípios cujo foco principal foi o ponto de vista do utilizador. Entre eles, faz sentido realçar os princípios relacionados com a cognição e a interação dos utilizadores. Os princípios cognitivos visam minimizar o esforço cognitivo despendido pelo utilizador, enquanto que os

princípios interativos tencionam minimizar o número de interações entre o utilizador e o dispositivo para realizar uma tarefa. Os quatro princípios cognitivos são a **consistência**, a **curva de aprendizagem**, a **previsibilidade** e o **reconhecimento**. Uma interface consistente não deve atribuir definições diferentes para a mesma ação ou objeto, de maneira a evitar confundir o utilizador. A curva de aprendizagem deve ter o mínimo de exigência possível e o resultado das funcionalidades do sistema deve ser suficientemente previsível para o utilizador. Por fim, o sistema deve estar pensado de modo a que o utilizador não tenha que recorrer à sua memória de curta duração. Isto é, o utilizador deve ser capaz de reconhecer os elementos da interface sem ter que raciocinar ou guardar informação sobre eles. No que toca à interação, para o caso de um sistema de RA são especialmente importantes as diretivas do **esforço reduzido** do utilizador e da **capacidade de resposta** do sistema. Uma vez que a interação com um sistema de RA é geralmente em movimento, é preciso ter em atenção o cansaço (mental) que a interação contínua entre a pessoa e o ecrã poderá causar. Para além disso, o sistema deve estar preparado para responder rapidamente aos pedidos dos utilizadores, de modo a preservar o realismo do conteúdo virtual apresentado.

3.3 Casos de Estudo

Nesta secção são apresentados estudos de RA focados nas vertentes de colaboração e de navegação móvel. Segundo Dey et al. [13], a pesquisa no campo da RA colaborativa está sobretudo direcionada à comunicação remota, havendo poucos estudos com experiências de RA partilhada no mesmo local. O foco desta secção visa precisamente explorar os trabalhos existentes no âmbito da colaboração presencial. Será seguida uma ordem cronológica para a narrativa desta secção, com a exceção dos trabalhos realçados no final.

Em 1997, Feiner et al. [16] desenvolveram o primeiro protótipo de RA móvel para navegação ao ar livre. O sistema foi denominado de "Touring Machine" e consistiu num capacete com um ecrã ótico transparente suportado por um computador guardado numa mochila e um dispositivo de computação portátil que serviu de controlador. A ideia principal era a de expandir a informação sobre os edifícios da faculdade à medida que o utilizador navegava pelo campus.

Em 2000, Billinghurst et al. [5] criaram o projeto "*Shared Space*", onde é estudado como a RA pode ser usada para desenvolver técnicas de colaboração presencial mais eficientes. Foi desenvolvido um jogo de grupo com dezasseis cartas como interfaces tangíveis cuja principal mecânica de interação era a troca dessas cartas entre os jogadores. Os autores assinalam que a utilização de interfaces tangíveis com padrões de comportamento facilmente reconhecidos pelo ser humano são uma característica chave para o desenvolvimento de interfaces de RA colaborativa em trabalhos futuros.

Em 2001, Reitmayr e Schmalstieg [39] apresentam o primeiro protótipo de RA colaborativa móvel. O sistema de colaboração configurado é entre um utilizador móvel e um estacionário e a comunicação é feita por LAN (rede de área local). No ano seguinte,

Piekarski e Thomas [38] adaptaram o jogo de computador "Quake" a um ambiente de RA.

Em 2004, Reitmayr e Schmalstieg [40] desenvolveram um sistema de RA para uma aplicação de guia turístico na cidade de Viena. Esta serve de auxílio à navegação, utilizando setas para guiar o utilizador ao local desejado. Este trabalho propõe três formas de navegação colaborativa: seguir, guiar e encontrar. Um utilizador que escolha seguir outro utilizador terá o seu local desejado constantemente atualizado para coincidir com o ponto mais perto do utilizador que está a seguir. A ação de guiar permite simplesmente que um utilizador escolha o destino de outro e a ação de encontrar guia dois utilizadores para se encontrarem a meio caminho.

Em 2005, é criada a primeira aplicação de RA para vários utilizadores num dispositivo de mão portátil, chamada "*The Invisible Train*". Este jogo foi desenhado para quatro utilizadores (dois jogadores e dois espectadores), onde os jogadores controlavam o rumo dos comboios virtuais numa pista real delimitada por marcadores. O objetivo do jogo é que os jogadores trabalhem em conjunto para evitar colisões. Wagner et al. [46] referem a importância das tecnologias de rede para o trabalho elaborado. Durante o jogo, o estado da aplicação é constantemente sincronizado entre os diferentes dispositivos (com latências de 40 a 50 milissegundos). São utilizados *sockets* de rede para estabelecer a comunicação.

No mesmo ano, Henrysson et al. [19] apresentam o jogo "*AR tennis*", desenvolvido para telemóveis. A comunicação entre dispositivos foi feita por Bluetooth, onde um dispositivo agia como servidor e anunciava o jogo para que outro dispositivo se conectasse a esse canal. O campo de ténis foi criado através de marcadores. Foram feitos dois testes para mostrar a utilidade da RA e a importância do *feedback* multissensorial (sonoro e háptico) para melhorar a experiência de um jogo de telemóvel colaborativo. Os autores concluíram que o grau de satisfação não teve desvios significativos ao longo das diferentes condições experimentais, mas que a interface de RA permitiu com que os utilizadores estivessem mais conscientes da presença dos seus colaboradores.

Mais recentemente, em 2013, Baldauf e Fröhlich [4] estudaram o uso de RA colaborativa num ecrã gigante. O sistema foi dividido em duas componentes. O ecrã, com vários vídeos para os utilizadores escolherem, e os *smartphones* dos utilizadores, que serviram de controladores remotos. As técnicas de rastreamento da aplicação Android foram implementadas com a utilização do Vuforia[1] e os pedidos de rede entre as duas aplicações foram feitos utilizando *sockets* TCP através de WiFi ou 3G. Apesar da experiência partilhada requerer um sistema de gestão central (aplicação a correr no ecrã gigante), a utilização de tecnologias atuais (como o sistema Android e o *kit* Vuforia) introduz um ponto de referência mais significativo em relação aos trabalhos referidos anteriormente. Por fim, a separação entre as funções de rastreamento e as funções de comunicação também é uma abordagem relevante no âmbito desta dissertação. Em 2016, Khan et al. [21] propõem uma solução central e modular semelhante, para um caso de uso diferente. Neste caso, os

[1] https://developer.vuforia.com/

utilizadores podem partilhar um evento histórico dentro de uma sala, juntamente com personagens virtuais desse evento. A experiência foi criada com o motor de jogo Unity3D[2] e foi usado rastreamento eletromagnético com base no sistema Polhemus G4[3]. Foi usado um servidor central para receber a informação dos sensores dos dispositivos e adaptar essa informação a um sistema de coordenadas global.

Em 2018, Mackamul e Esteves [28] compararam as diferenças entre a utilização de dispositivos móveis e a utilização de projetores para a execução de tarefas colaborativas em RA. Foi testado o jogo "Match Pairs" para computador e para Android. A aplicação Android foi desenvolvida em Unity3D com o *kit* Vuforia para a RA. A sincronização de rede entre os dispositivos foi feita através da plataforma Photon Engine[4]. Concluiu-se com este estudo que a diferença de usabilidade não é relevante entre uma aplicação de RA colaborativa para dispositivos móveis ou com base num projetor. Os utilizadores consideraram a natureza das interfaces móveis mais intuitiva, apesar da comunicação não verbal não ser tão óbvia. Os autores sugerem duas soluções para contornar este problema: (1) permitir que os utilizadores partilhem temporariamente os seus pontos de vista e (2) realçar um elemento virtual para todos os utilizadores quando um utilizador o seleciona no seu ecrã.

Em 2019, Pereira et al. [37] criaram uma infraestrutura colaborativa de Realidade Mista onde sugerem diversas técnicas de interação de grupo dentro de um espaço virtual partilhado. São empregues avatares para identificar os utilizadores e são propostos vários mecanismos de manipulação e realce dos objetos partilhados. O sistema foi desenvolvido em Unity3D e a comunicação foi configurada com a solução de rede do Photon Engine.

Por fim, um comentário a fazer é que quase todas estas referências são baseadas em ambientes controlados e com um reduzido número de utilizadores de teste em simultâneo. De seguida são apresentados com maior detalhe trabalhos que se relacionam particularmente com o tópico desta dissertação.

3.3.1 ARTiFICe

Em 2012, Mossel et al. [33] criaram uma infraestrutura para aplicações de RA e RV colaborativas e distribuídas, assentando toda ela sobre o Unity3D como plataforma de desenvolvimento. As extensões da infraestrutura relativas à RA e RV são construídas em torno das funcionalidades oferecidas pelo Unity3D. A infraestrutura criada é multiplataforma e suporta colaboração tanto em configurações de RA/RV móveis (*smartphones*, *tablets*, óculos) como em configurações estáticas (computadores fixos, projetores). A solução proposta está esquematizada na figura 3.3.

Esta solução foi dividida em três camadas: os dispositivos de entrada (como câmaras de vídeo ou controladores como ratos 3D), o *middleware* e a camada aplicacional. A camada de *middleware* serve para converter a informação de posicionamento emitida pelos

[2] https://unity3d.com/
[3] https://polhemus.com/motion-tracking/all-trackers/g4
[4] https://www.photonengine.com/en/PUN

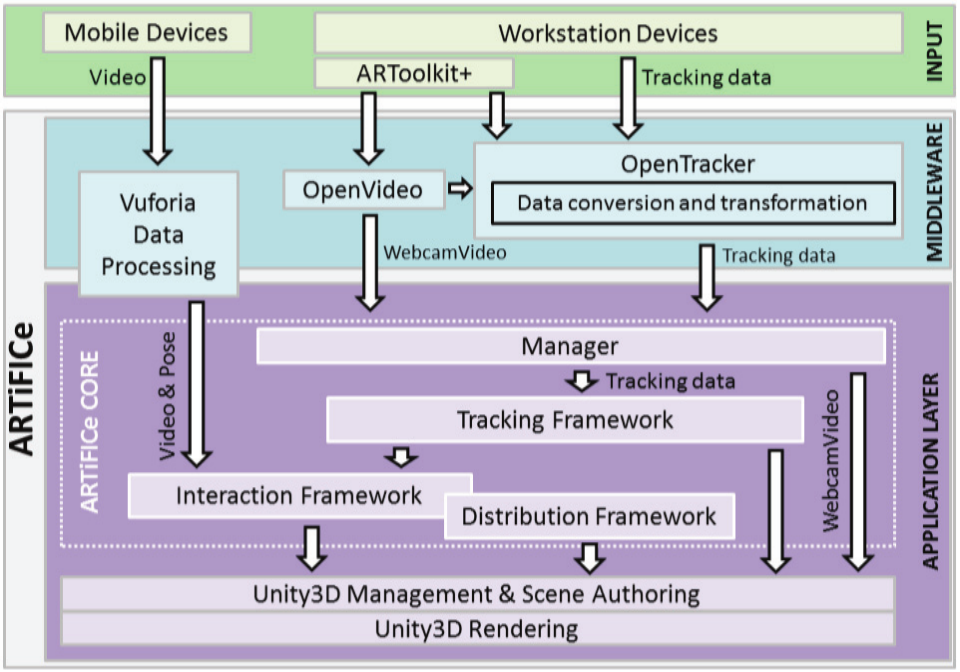

Figura 3.3: Infraestrutura proposta por Mossel [33]

dispositivos de entrada num formato comum para depois ser processado pela camada aplicacional. Para dispositivos móveis é utilizado o *kit* Vuforia. A camada aplicacional está dividida em três componentes: rastreamento, interação e colaboração. A componente de rastreamento serve para estimar a posição e a orientação dos objetos virtuais em relação ao dispositivo. A vertente de interação lida com as técnicas de seleção e manipulação dos objetos em cena. São aplicados vários padrões de interação da RV. Para a interação com telemóvel foi adaptada a técnica HOMER [8]. Esta técnica, num sistema de RV, permite que a mão virtual do utilizador se desloque até ao objeto, selecionado previamente por um feixe de laser. Num telemóvel, o toque do utilizador no ecrã táctil funciona como o feixe de laser e a mão real substitui a mão virtual. Por fim, a componente de colaboração serve para suportar a comunicação entre diferentes dispositivos, sejam eles de configuração móvel ou fixa. As funções de rede são baseadas na camada de rede disponibilizada pelo Unity3D, que usa o protocolo UDP. É utilizada uma arquitetura cliente-servidor com ligação direta entre o servidor e todos os clientes (topologia de estrela). Neste caso, o servidor é um dos dispositivos que age como servidor e cliente em simultâneo, para facilitar a implementação da arquitetura cliente-servidor. A troca de dados entre dispositivos é realizada através de chamadas de procedimento remotas (RPC, do inglês) e da sincronização de estados. A sincronização é feita por segmentos de informação para prevenir a perda de dados.

Para testar a aplicabilidade da infraestrutura criada, foram criadas quatro pequenas aplicações por 80 estudantes no âmbito da unidade curricular "Virtual Reality Lab Exercise" da universidade de Viena. Noutro programa da universidade, 17 estudantes desenvolveram uma aplicação móvel distribuída e colaborativa em 4 semanas.

3.3.2 CARS

Zhang et al. [48], mais recentemente, conceberam também uma infraestrutura de colaboração para RA, baseada na *Cloud*. Neste caso, os autores aproveitam a natureza social do ser humano para melhorar a qualidade da experiência nos sistemas de RA, especialmente a nível de latência. Para além das interações em tempo real entre dois utilizadores, a ideia de colaboração aqui também vai no sentido de melhorar o desempenho aplicacional e reutilizar recursos computacionais em tarefas cuja carga de processamento seja mais intensiva.

Os autores dividem o processo de reconhecimento (referido em 2.5) em duas fases: a deteção dos objetos em cena e a comparação desses objetos com modelos guardados numa base de dados local. A computação pode ser passada para a *Cloud* em qualquer uma destas fases. Fazê-lo na primeira fase permite com que os algoritmos de visão computacional possam ser executados remotamente. Por outro lado, o tamanho dos *frames* da câmara em bruto transmitidos por rede também introduz uma carga significativa a nível de comunicação. Na segunda fase referida, já com os algoritmos de visão computacional executados, o objetivo de fazer a comparação em *Cloud* é o de não sobrecarregar a memória do dispositivo com uma base de dados local demasiado extensa.

Um dos objetivos deste estudo é perceber se a interação direta entre dispositivos pode melhorar o desempenho aplicacional quando comparada com o auxílio exclusivo da *Cloud* por parte de cada dispositivo.

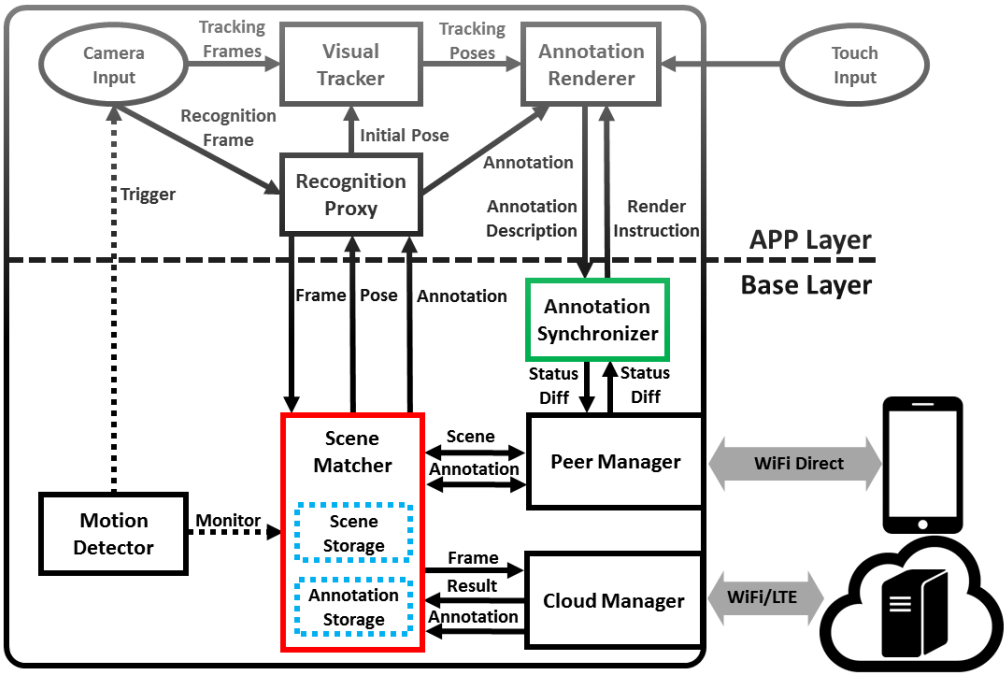

Figura 3.4: Infraestrutura proposta por Zhang [48]

A arquitetura do sistema proposto está dividida em duas camadas, representadas na figura 3.4. A camada aplicacional recebe as imagens da câmara para o reconhecimento

e rastreamento dos objetos. A componente de reconhecimento passa a imagem recebida pela câmara para a camada de base e recebe o posicionamento do objeto virtual a representar como resposta. A camada de base trata dos algoritmos de reconhecimento e do *download* e sincronização de objetos virtuais para passar para a camada aplicacional. O detetor de movimento desta camada indica se o dispositivo está preparado para processar o *frame* atual da câmara. O *frame* apenas é processado se o dispositivo estiver numa posição relativamente estável e na vertical. O sincronizador de anotações serve para sincronizar as interações entre os diferentes utilizadores e partilhar o estado dos objetos virtuais (posição, escala, orientação) quando estes são modificados. É usado um gestor de comunicação individual entre dispositivos e outro gestor para comunicações com a *Cloud*. Para o primeiro é usado *WiFi Direct* ou *Bluetooth Low Energy*. Esta componente verifica periodicamente por dispositivos nas redondezas que estejam a usar a aplicação. É mantida uma lista de dispositivos colaboradores aos quais se pode pedir informação antes de fazer um pedido por *Cloud*. Em relação ao gestor de *Cloud*, a comunicação é feita por WiFi ou 3G/4G. Por fim, o comparador de cenas usa algoritmos de visão computacional sobre os *frames* recebidos da camada aplicacional e compara os resultados com a sua base de dados local. Se não encontrar nenhum resultado, essa verificação pode ser feita por dispositivos colaboradores ou por *Cloud*.

Os autores testaram a infraestrutura apenas com dois *smartphones* Android. Os resultados experimentais mostram que a comunicação entre dispositivos pode reduzir a latência da fase de reconhecimento até 40 por cento, quando comparada com um sistema que faça apenas uso da comunicação por *Cloud*.

3.3.3 Tese de Mimaroğlu

Mimaroğlu [32] na sua dissertação propõe uma infraestrutura modularizada para comportar a componente de colaboração entre dispositivos. A infraestrutura é apresentada na figura 3.5.

São destacados quatro módulos principais de gestão no modelo apresentado: (1) módulo de reconhecimento e rastreamento, (2) módulo de representação, (3) módulo de interação, e (4) módulo de grupo. É também criado um módulo conector entre a aplicação e os restantes módulos, que serve de intermediário na comunicação entre os eventos recebidos pela interface da aplicação ou pelos diferentes módulos. O autor dá o nome de "fachada" a esta componente. De resto, vale a pena destacar a composição dos módulos de interação e de grupo. O primeiro serve para gerir as interações dos utilizadores com os objetos virtuais disponíveis na sessão de utilização. As alterações feitas sobre os objetos são depois comunicadas ao módulo de gestão de grupo. O módulo de grupo, por sua vez, está encarregue da gestão de sessões entre dispositivos que partilham a mesma experiência. É também o intermediário entre a camada de rede e a restante aplicação, sendo que é usada a biblioteca Alljoyn[5] para realizar a comunicação entre dispositivos.

[5] https://openconnectivity.org/developer/reference-implementation/alljoyn

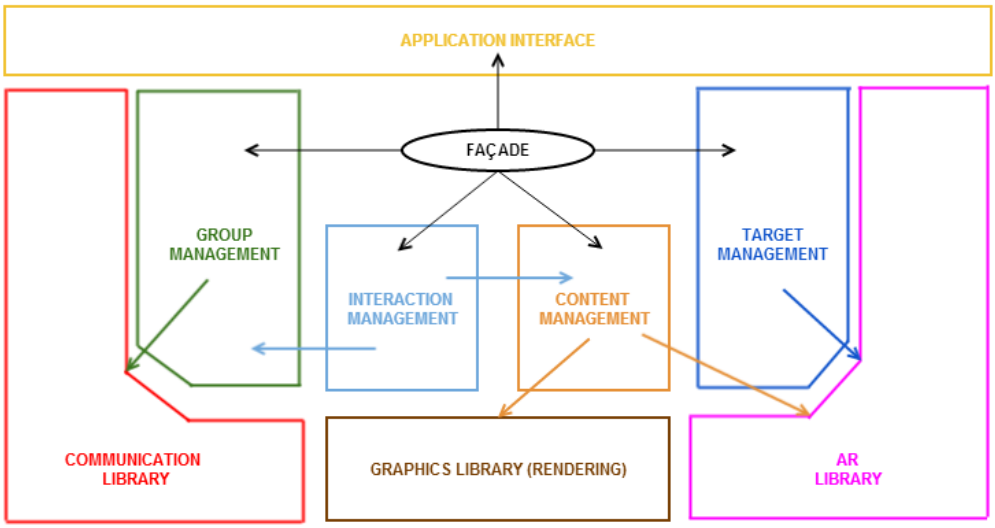

Figura 3.5: Infraestrutura proposta por Mimaroğlu [32]

3.4 Tecnologias de Suporte

Neste momento já existe alguma variedade no processo de decisão para escolher uma ferramenta de desenvolvimento de uma aplicação de RA. Nesta secção, são apresentadas algumas dessas tecnologias, especialmente aquelas que suportam a realização do trabalho proposto.

3.4.1 Unity3D

O Unity3D é um motor de jogo - com motores gráficos e de física - que suporta o desenvolvimento de aplicações 2D e 3D para diferentes plataformas, incluindo computadores, consolas e dispositivos móveis. O desenvolvimento é feito em C#, com uma documentação de apoio bem detalhada. A principal razão para escolher o Unity3D como plataforma de desenvolvimento é a sua compatibilidade com outras soluções de desenvolvimento. Nomeadamente as soluções de comunicação por rede entre vários dispositivos, como Photon Engine, Alljoyn ou Mirror Networking[6]. Foram apresentados exemplos que utilizam estas soluções em 3.3. Para além dos *kits* de comunicação, o Unity3D também é compatível com as extensões de RA mais utilizadas atualmente, como ARCore, ARKit, Vuforia, Wikitude[7] e Kudan[8].

3.4.2 Vuforia

Vuforia é um *kit* de desenvolvimento de *software* para aplicações de RA. É compatível com Android, iOS, Windows e Lumin OS. Apresenta uma capacidade de reconhecimento

[6] https://mirror-networking.com/
[7] https://www.wikitude.com/products/wikitude-sdk/
[8] https://www.xlsoft.com/en/products/kudan/index.html

robusta, tanto de múltiplas imagens 2D - marcadores - como de pequenos objetos 3D previamente modelados. Oferece a função de rastreamento prolongado dos marcadores reconhecidos, utilizando os sensores do dispositivo para calcular a posição do marcador mesmo quando este não aparece no *frame* da câmara. Implementa ainda a tecnologia de deteção de planos, com a qual se pode fazer o reconhecimento do espaço e das formas presentes no mundo real, possibilitando a construção de um modelo 3D à base de SLAM. Esta tecnologia é integrada com os *kits* ARCore e ARKit através da plataforma Vuforia Fusion. Ainda é possível referir outro tipo de funcionalidades secundárias, como a reprodução de vídeos em segundo plano e a interação com botões virtuais. Num cômputo geral, é uma das soluções mais robustas e completas no campo da RA.

3.4.3 Photon Engine

Photon Engine é uma ferramenta de comunicação em rede, com suporte de integração com o Unity3D e normalmente utilizada no desenvolvimento de jogos *online* para vários utilizadores. A plataforma do Photon Engine permite escolher entre configurar um servidor local e usar um dos servidores hospedados na Photon Cloud distribuída globalmente. Esta solução permite sincronizar a informação de vários objetos entre os diferentes utilizadores conectados, com a opção de serem usadas atualizações contínuas ou esporádicas. A arquitetura do sistema faz a utilização de um cliente mestre, que é um cliente ao qual são atribuídas funções especiais de gestão e sincronização que permitam suprimir a necessidade de configurar o mesmo tipo de lógica num servidor dedicado.

3.4.4 Experiências e Limitações

Num período inicial, foram feitos alguns testes para analisar o comportamento das ferramentas já mencionadas. Para aprender as funções básicas do Unity3D e do Vuforia, foi desenvolvida uma pequena prova de conceito de um projeto que fazia o reconhecimento da imagem de uma onda sonora e reproduzia o som que lhe estava associado, juntamente com a animação da onda a ser preenchida. Basicamente, era feito o reconhecimento da imagem (marcador) à qual era aplicada uma máscara que ia avançando ao longo da duração do som reproduzido. A figura 3.6 mostra três momentos da animação referida, por ordem cronológica.

Com o Vuforia foram também feitos testes à qualidade do reconhecimento de edifícios e de outros objetos de rua (ecopontos). Os resultados não foram totalmente satisfatórios, dependendo bastante dos ângulos e distâncias da câmara, assim como da ausência de sombras ditadas pela altura do dia e pelas condições de luminosidade.

Foi também testado o reconhecimento de objetos mais pequenos, onde a distância necessária para reconhecer esses objetos foi de cerca de dez vezes a largura dos mesmos. Um marcador numa folha A4 (vinte e um centímetros de largura), por exemplo, só era reconhecido por dispositivos a cerca de dois metros de distância. Ainda assim, as funcionalidades de rastreamento prolongado, com a mistura dos sensores, denotaram um nível

Figura 3.6: Aplicação experimental com ondas sonoras

de estabilidade assinalável.

Por fim, foram ainda feitos testes às *Cloud Anchors* do ARCore. É uma tecnologia ainda em fase de desenvolvimento e cuja utilização foi considerada, mas as dificuldades de persistência e de acesso aos dados limitam os cenários de utilização. O modelo 3D criado é razoavelmente estável, mas não totalmente robusto dado que esporadicamente o posicionamento da âncora regista pequenos saltos.

3.5 Técnicas de Validação

Nesta secção são revistas as metodologias de avaliação utilizadas em sistemas de RA. Em 2008, Duenser et al. [15] classificaram as técnicas de avaliação usadas nos estudos da área em cinco categorias. O estudo foi feito com base em 3309 artigos dos repositórios da *ACM*[9] e do *IEEE*[10]. As categorias encontradas foram as seguintes:

- **Medidas objetivas**: nesta categoria foram identificados estudos que fazem uma análise estatística e quantitativa de variáveis como as taxas de erro do sistema (precisão), os intervalos de tempo ou o número de ações necessárias para realizar uma tarefa.

- **Medidas subjetivas**: neste caso foram selecionadas publicações que recorreram a questionários, classificações subjetivas dos utilizadores ou opiniões. As escalas de Likert [27] e da NASA-TLX[11] são dois exemplos de modelos de avaliação subjetiva.

- **Análise qualitativa**: este tipo de análise é feito através de entrevistas formais com os utilizadores ou pela observação dos seus comportamentos durante a experiência. O tom de voz ou a movimentação corporal podem ser sugestivos do grau de satisfação da experiência, por exemplo. Num cenário de colaboração, medidas como o número de interrupções entre os intervenientes ou a duração dos seus discursos também podem ser aplicadas.

[9] https://dl.acm.org/
[10] https://ieeexplore.ieee.org/Xplore/home.jsp
[11] https://humansystems.arc.nasa.gov/groups/TLX/

- **Técnicas de avaliação da usabilidade**: estas técnicas são geralmente usadas no campo da interação pessoa-máquina. Exemplos destes métodos são os de pedir ao utilizador para "pensar em voz alta", de fazer uma avaliação heurística da usabilidade do sistema ou de fazer uma análise individual de cada uma das tarefas do sistema.

- **Avaliações informais**: estas avaliações comportam observações ou comentários informais dos utilizadores.

A combinação destes métodos pode ser utilizada para explorar diferentes aspetos da experiência do utilizador e formar uma base de análise mais completa. Gabbard et al. [17] criaram uma metodologia para combinar várias técnicas de validação no desenho de sistemas para Ambientes Virtuais. O processo envolve quatro etapas sequenciais de (1) análise de tarefas, (2) avaliação feita por especialistas, (3) avaliação focada no utilizador, e (4) avaliação comparativa. A primeira etapa consiste na identificação completa das tarefas e dos métodos necessários para os utilizadores usarem o sistema. A segunda etapa requer que um pequeno grupo de especialistas (entre três a cinco) faça avaliações heurísticas sobre a usabilidade do sistema, comparando o desenho do sistema com padrões de desenho bem estabelecidos na área da interação pessoa-máquina. Mais especificamente, os especialistas determinam as diretivas de desenho que o sistema cumpre. No caso das diretivas que o sistema não cumpre, cabe aos especialistas fazerem recomendações para melhorar o sistema. Na terceira etapa, baseada nos testes com utilizadores, são postas em prática as tarefas definidas na primeira etapa contextualizadas num cenário de utilização. Nesta etapa são aplicadas medidas objetivas, subjetivas e é feita uma análise qualitativa sobre as tarefas realizadas pelos utilizadores, de modo a perceber o que pode ser melhorado. Por fim, a avaliação comparativa faz comparações entre diferentes desenhos de interação desenvolvidos, para perceber qual é o que melhor se aplica ao sistema desenvolvido.

Capítulo 4

Solução Proposta

Neste capítulo são definidos os detalhes de desenho para a solução proposta. Esta solução comporta uma interface de programação de aplicações (API, em inglês) para o desenvolvimento de aplicações de RA para múltiplos utilizadores.

São primeiro apresentados os objetivos que se pretendem cumprir com o trabalho apresentado, com base nos conceitos já relatados na literatura e revistos no capítulo 3. De seguida, são expostos os requisitos e componentes necessários para a elaboração de uma solução focada em casos de uso com múltiplos utilizadores.

O desenho da solução está dividido em três partes. Primeiro é esclarecido o enquadramento e a arquitetura da solução e depois são apresentadas as interfaces definidas para a API e as suas funcionalidades adjacentes.

Por fim, são ainda descritos os conceitos das aplicações a desenvolver com base na API criada, assim como a integração destas com a Taxonomia de Brockmann, referida em 3.1.1.

4.1 Objetivos

Como já foi referido no capítulo 1, a presente dissertação tem como objetivo conjugar os paradigmas de comunicação em grupo com aqueles existentes no campo da RA. A solução apresentada visa agilizar o processo de desenvolvimento e integração de uma aplicação de RA pensada para um cenário onde os protocolos de comunicação entre múltiplos utilizadores sejam essenciais.

Serão tomados em consideração, no desenvolvimento da infraestrutura proposta nesta dissertação, os princípios de desenho expostos em 3.2.1 e os diferentes cenários cobertos pelas diversas combinações entre as seis dimensões da Taxonomia de Brockmann, apresentados em 3.1.1.

Na realização de qualquer objetivo proposto, será ainda importante ter em conta os pontos fortes e as limitações das tecnologias de suporte utilizadas, apresentados em 3.4.

Para abranger o maior número de casos de utilização possível, é fundamental que a solução encontrada não esteja limitada por demasiadas restrições de *hardware* ou por um processo de configuração (ou de reconfiguração) lento e complexo para o utilizador. É, portanto, importante que a solução apresentada seja capaz de lidar com a instabilidade de um ambiente móvel, onde os utilizadores podem perder a ligação por instantes.

Para além de uma natureza heterogénea a pensar em múltiplos dispositivos e múltiplas plataformas, outros parâmetros terão de ser garantidos pelo desenho da solução proposto. A infraestrutura criada deverá suportar múltiplos utilizadores capazes de interagir entre si, através do mundo real e virtual. Sendo que esta interação poderá ser de carácter competitivo ou colaborativo, dependendo do contexto de utilização. Para além disso, deverá ser possível definir diferentes níveis de acesso à informação para grupos de utilizadores distintos, de maneira a poder criar vistas individuais e partilhadas adaptadas para diferentes utilizadores. Esta diferenciação de acessos poderá ser útil para estudar o impacto na decisão de utilizadores em colaborar por voz ou de se limitarem a utilizar a interface que lhes é atribuída.

Um dos pontos importantes a considerar no desenvolvimento da API proposta é a necessidade de generalizar as suas funcionalidades a um cenário de utilização abrangente. Essas funcionalidades devem ainda ser simples o suficiente para garantir a extensibilidade da solução, e qualquer funcionalidade implementada deverá ser oferecida num contexto de fácil utilização e interpretação para quem a for usar.

Por fim, a solução desenvolvida deverá focar-se em estudar e suportar quatro dimensões fundamentais do problema, apresentadas de seguida.

- Gestão de utilizadores e da ligação entre eles;
- Gestão das permissões de grupo e dos diferentes níveis de acesso concedidos a cada utilizador;
- Controlo e manipulação dos objetos virtuais;
- Persistência da informação.

Cada um destes pontos será explorado com mais detalhe na secção seguinte.

4.2 Requisitos

Para conceber a solução proposta, alguns requisitos tiveram de ser estabelecidos, com base nos objetivos traçados para esta dissertação. Estes requisitos têm por base as necessidades gerais que uma aplicação de RA colaborativa precisa de cumprir para funcionar devidamente.

Tratando-se de uma infraestrutura com o objetivo de integrar técnicas de **RA**, dinâmicas de **grupo** e metodologias de **comunicação** por rede, é natural que os requisitos levantados espelhem sobretudo as necessidades básicas para servir este tipo de sistemas.

Levando isso em consideração, foi primeiro definido um conjunto de requisitos essenciais ao funcionamento de um sistema principalmente focado nas três categorias evidenciadas acima. Esse conjunto é apresentado de seguida.

- O sistema deve ser capaz de registar conteúdo virtual sobre a vista capturada pela câmara;
- O sistema deve permitir aos utilizadores interagirem com o conteúdo virtual;
- O sistema deve oferecer aos utilizadores uma maneira de se posicionarem no espaço, seja pela utilização do GPS ou pela integração e leitura de outros sensores;
- O sistema deve ser desenhado de maneira a comportar a possibilidade de ser utilizado em diferentes tipos de dispositivos de visualização;
- O sistema deve considerar ambientes para mais do que um utilizador;
- O sistema deve possibilitar a partilha de informação entre utilizadores, nomeadamente informação relativa ao mundo virtual partilhado;
- O sistema deve ser capaz de oferecer diferentes níveis de acesso à informação aos seus utilizadores;
- O sistema deve possibilitar a criação de objetos como entidades independentes dos seus utilizadores;
- O sistema deve permitir que dois objetos virtuais interajam entre si, sem precisar necessariamente da intervenção de um utilizador. Por exemplo, dois objetos poderão ser programados para lidar com interações de colisão ou de proximidade, sem que seja preciso que um utilizador intervenha;
- A infraestrutura de comunicação na qual o sistema está assente deve ser simplificada e de fácil configuração;
- A infraestrutura de comunicação deverá permitir definir se a informação é enviada com ou sem fiabilidade, isto é, se esta chega obrigatoriamente a todos os utilizadores ou não;
- A API da solução apresentada deve ser expansível e modularizável.

Para além dos requisitos básicos apresentados acima, no que toca à especificação da solução proposta, esta deverá ainda focar-se em quatro categorias em particular.

Como a infraestrutura proposta tem uma vertente muito ligada à comunicação em rede, será preciso fazer a gestão de utilizadores e a maneira como estes se ligam e organizam entre si.

Para lidar com a vertente colaborativa da solução, será também preciso definir certos mecanismos de grupo entre os utilizadores e estabelecer diferentes níveis de permissão e de acesso ao conteúdo virtual.

Para tratar da integração das técnicas de RA, nomeadamente do reconhecimento, da visualização e da interação com os objetos, será preciso especificar uma categoria dedicada ao controlo e manipulação de objetos virtuais.

Por fim, será ainda necessário tratar da persistência do conteúdo virtual e da possibilidade de o aceder de maneira assíncrona.

Estas quatro categorias são apresentadas de seguida, mais detalhadamente.

4.2.1 Gestão de Utilizadores

O desenho e levantamento de requisitos foi inspirado em grande parte por soluções e padrões de jogos multijogador já existentes. Assim, convém primeiro introduzir os conceitos de sala e de sessão de utilização para fazer um enquadramento daquilo que será esperado apresentar.

A sessão de utilização é a componente responsável por gerir a ligação do utilizador à rede de comunicação partilhada. Uma sala, por sua vez, é o espaço virtual no qual os diferentes utilizadores se juntam e organizam antes de começar um jogo em conjunto. Um utilizador tem duas maneiras de se juntar a uma sala: tendo acesso à identificação da sala ou através de métodos de procura que se baseiem em características específicas da sala. Um utilizador pode ter iniciado sessão sem ter entrado numa sala, mas não o contrário.

Tendo estes dois conceitos em consideração, é assim esperado que o sistema permita:

- Criar ou registar um utilizador;
- Iniciar ou terminar sessão;
- Criar salas;
- Entrar e sair de uma sala;
- Editar os atributos de um utilizador e de uma sala.

4.2.2 Mecanismos de Grupo

Esta categoria deverá ser responsável por lidar com a parte colaborativa e competitiva do problema, estabelecendo ligações especiais entre determinados grupos de utilizadores. Estas ligações permitem que se agrupem os jogadores em grupos, equipas ou papéis e, a partir daí, é possível conceder ou retirar acessos que um determinado grupo terá para

interagir com outro grupo ou para visualizar e manipular o conteúdo virtual introduzido na aplicação.

Assim, esta gestão de permissões e de grupos do sistema deve incluir:

- Criar um grupo;
- Adicionar ou remover um utilizador de um grupo;
- Enviar informação por rede para um grupo em específico;
- Mostrar ou ocultar conteúdo com base no(s) grupo(s) do utilizador.

4.2.3 Controlo de Objetos

O sistema deverá ser capaz de introduzir conteúdo virtual partilhado, em tempo real, pelos utilizadores.

Este conteúdo poderá ser estático e não interativo, estando apenas dependente de uma referência de reconhecimento para ser representado. Este é o processo já oferecido por ferramentas de desenvolvimento puramente de RA, como é caso do Vuforia, por exemplo.

No entanto, o conteúdo partilhado pela infraestrutura proposta também deverá ser interativo e, para além de interativo, as interações de um utilizador devem ser comunicadas e visualizadas pelos restantes. Para isto será preciso que o conteúdo seja colocado no mundo virtual com base em pontos de referência reais comuns a todos os utilizadores.

Deverá também ser possível distinguir entre objetos partilhados que pertencem a um utilizador e objetos partilhados cujo comportamento é independente das ações de um utilizador específico.

Portanto, a gestão de objetos partilhados do sistema deverá garantir os seguintes requisitos:

- Criar e destruir um objeto partilhado;
- Interagir com um objeto partilhado (tocar, selecionar, alterar dimensões, modificar atributos);
- Alterar o posicionamento de um objeto no espaço virtual partilhado;
- Atualizar a informação de um objeto de maneira periódica ou esporádica, dependendo do caso de uso.

4.2.4 Persistência de Informação

A infraestrutura proposta deverá também ser capaz de lidar com a persistência de informação gerada pelos utilizadores e pelas suas ações sobre o conteúdo virtual. Esta informação deverá ser guardada e carregada através de uma base de dados externa incorporada.

Esta componente da solução é particularmente indispensável para casos de utilização assíncrona, ou seja, quando a comunicação entre dois utilizadores não é feita em tempo real. Através desta componente será possível registar o historial da sessão e do conjunto de ações tomadas por cada utilizador, de maneira a que seja possível aceder a um conteúdo alterado por um utilizador que já não tenha sessão ativa.

4.3 Desenho da Solução

Nesta secção é apresentada e descrita a abordagem delineada para o desenho da solução proposta ao problema inicial. É apresentada a arquitetura do sistema e o seu enquadramento com o trabalho relacionado discutido no capítulo 3. A partir desta arquitetura é derivada uma API de colaboração em RA, dividida num conjunto de interfaces e funcionalidades propostas, de acordo com os requisitos levantados na secção anterior.

4.3.1 Arquitetura

A arquitetura da solução proposta será apresentada com dois níveis de detalhe.

Primeiro, será feito o seu enquadramento do ponto de vista do sistema, onde o objetivo é o de clarificar o nível de abstração da solução proposta para resolver o problema. Depois, será então apresentado e explicado com mais detalhe o desenho da infraestrutura e das componentes que a integram.

A implementação da solução proposta assume a possibilidade de utilizar bibliotecas complementares, desde que a utilização destas seja feita de forma modular para resolver subproblemas levantados pelas necessidades da infraestrutura. É na integração das diferentes bibliotecas com as outras componentes do sistema que está a contribuição do trabalho desenvolvido. Em 3.4 já foram referidas algumas tecnologias de suporte que se poderão utilizar.

Na figura 4.1 pode ser vista a arquitetura do sistema com um nível de abstração mais alto. Neste esquema, uma aplicação é desenvolvida com recurso às linguagens de programação utilizadas, que estão dependentes da plataforma que se for utilizar. Como um dos objetivos levantados na definição do problema foi o de criar uma solução compatível com várias plataformas, nem as linguagens de programação apresentadas nem os sistemas operativos são de cariz exclusivo. Neste caso, são apresentados exemplos sobre os quais a infraestrutura poderá assentar.

Seja como for, dentro da linguagem de programação escolhida, o criador da aplicação terá acesso a um conjunto de bibliotecas e de ferramentas de desenvolvimento específicas ao problema que este pretenda resolver. É aí que se enquadra a infraestrutura desenvolvida nesta dissertação. Infraestrutura que, por juntar mais do que uma área de estudo, também poderá integrar, em si mesma, um conjunto de bibliotecas de desenvolvimento úteis para resolver alguns dos subproblemas encontrados.

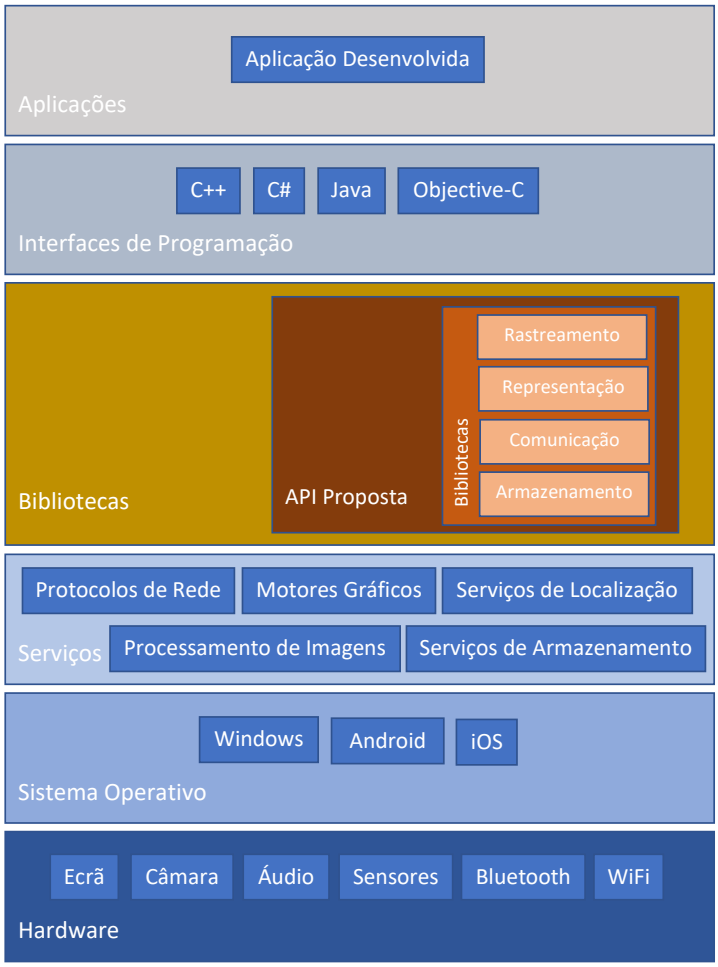

Figura 4.1: Enquadramento da infraestrutura proposta

Por fim, as funções implementadas pela infraestrutura oferecida fazem uso de serviços e componentes de *hardware* que possam ser relevantes para o problema, para ler ou enviar informação pertinente dessas mesmas componentes.

Na figura 4.2 a infraestrutura poderá ser vista e analisada com mais detalhe. O desenho da solução foi pensado de acordo com os objetivos e requisitos levantados anteriormente. Para isso, foram identificados quatro módulos essenciais para lidar com o problema: interação, registo, sincronização e persistência. Estes módulos interagem entre si e com outras bibliotecas externas de modo a fornecer às aplicações que usem a infraestrutura uma sessão de RA colaborativa.

O módulo de **interação** recebe a informação de entrada da aplicação e das componentes de *hardware* e faz o tratamento dos dados para os passar ao módulo de registo, para atualizar a sessão localmente, e ao módulo de sincronização, para atualizar a sessão de outros utilizadores.

O módulo de **sincronização** faz a gestão da ligação a outros utilizadores e dos mecanismos de grupo existentes no sistema. Este recebe informação do módulo de interação e passa-a para o módulo de persistência e para os outros utilizadores, podendo utilizar

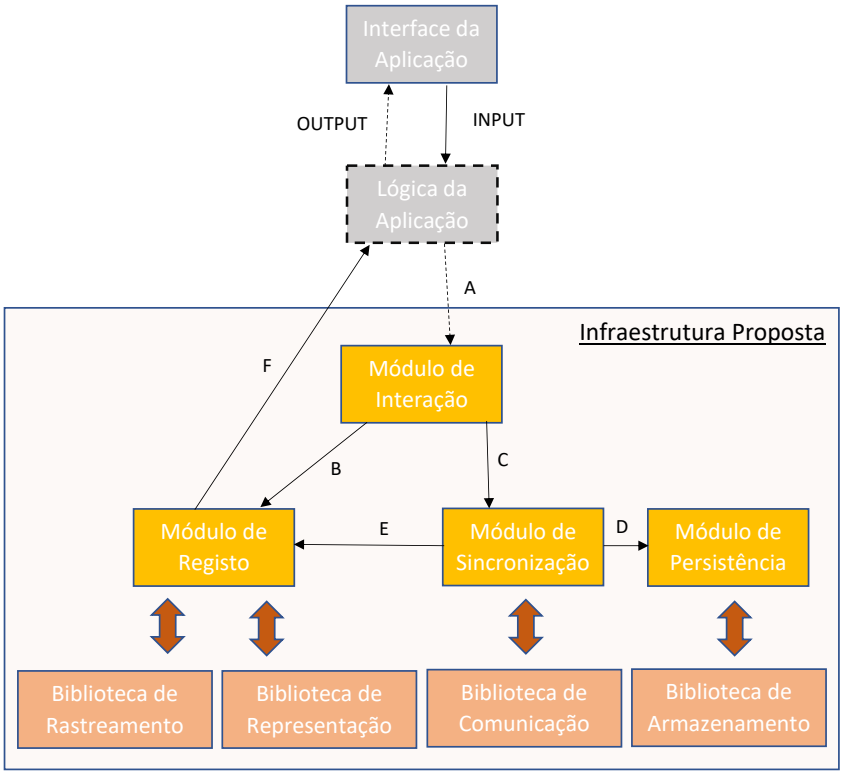

Figura 4.2: Infraestrutura proposta

uma biblioteca de comunicação externa para o efeito.

O módulo de **persistência** serve para gerir o processo de guardar e carregar os dados da sessão numa base de dados remota.

O módulo de **registo** é responsável pelo reconhecimento, rastreamento e representação dos objetos no contexto do mundo virtual representado. Faz o tratamento dos *frames* da câmara e da informação dos sensores para definir como e onde representar o conteúdo virtual. Pode também receber informação do módulo de sincronização para atualizar o estado ou o posicionamento relativos a um objeto virtual partilhado com outros utilizadores.

A lógica da aplicação está a tracejado para simbolizar a possibilidade de filtrar a informação recebida e enviada entre a infraestrutura e a interface da aplicação.

Existem dois fluxos de trabalho realizados pela infraestrutura para atualizar a instância de uma aplicação. Existe o fluxo local, no qual o conteúdo apresentado é alterado por meio de uma interação promovida pelo utilizador local e a ordem é: A -> B, C -> D, F. E existe o fluxo de rede, no qual o conteúdo apresentado é alterado por meio de uma interação promovida por um utilizador remoto. Neste caso, a ordem é: E -> F.

4.3.2 Interfaces

Em consonância com os objetivos, requisitos e a arquitetura desenhada anteriormente neste capítulo, foram definidas dez interfaces de programação como proposta para a

resolução do problema desta dissertação. Estas interfaces podem ser vistas na figura 4.3.

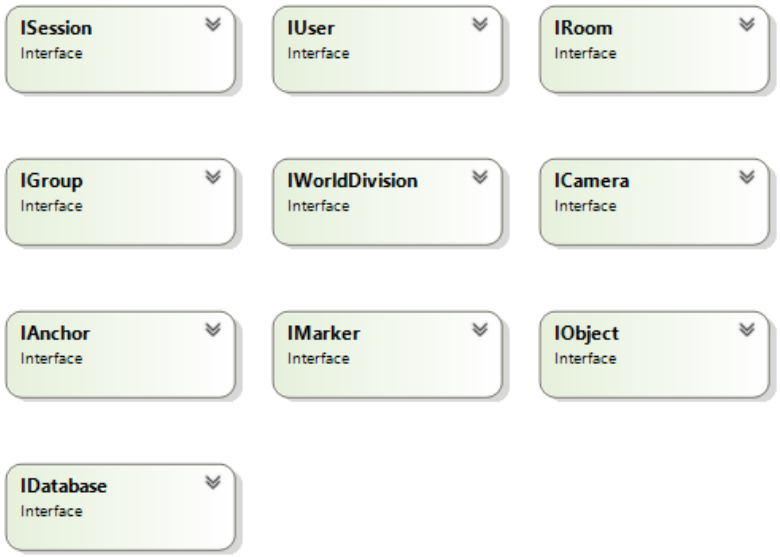

Figura 4.3: Interfaces da API

A definição, explicação e análise de cada uma das interfaces mostradas na figura são apresentadas de seguida.

- **ISession**: A interface que representa uma sessão individual de utilização. A sua utilização é indispensável para um programador que queira usar o resto da API, uma vez que é nesta componente que é feita a gestão de utilizadores, salas, grupos e divisões disponíveis no sistema. É também aqui que poderá ser feita a configuração de alguns parâmetros globais de rede e de registo relevantes para problemas de RA colaborativa;
- **IUser**: Representa um utilizador para registar ou entrar no sistema;
- **IRoom**: Interface de sala. Como já tinha sido referido anteriormente, uma sala representa o espaço fictício no qual os utilizadores se juntam e organizam antes de começar um jogo em conjunto. Só é possível aceder a uma sala a partir de uma sessão de utilização que já tenha um utilizador associado (com sessão iniciada);
- **IGroup**: Interface de grupo. Serve para agrupar os utilizadores do sistema em grupos, equipas ou papéis diferenciados que permitam gerir a filtragem do conteúdo virtual para um conjunto específico de utilizadores;
- **IWorldDivision**: Esta interface representa uma divisão no mundo virtual, isto é, o espaço virtual onde o conteúdo de RA é registado e partilhado. Cada divisão está encarregue por fazer a gestão do conteúdo que nela é registado, como os marcadores de RA, os objetos virtuais partilhados ou as âncoras espaciais. A possibilidade de um utilizador ir mudando entre divisões permite estruturar a aplicação por partes

e fazer uma gestão modularizada da mesma. Também permite dividir a capacidade de processamento exigida por uma aplicação, dividindo o conteúdo virtual por diferentes divisões;

- **ICamera**: Interface relativa às componentes e ao estado da câmara distinguida para fazer o processamento de imagens e o reconhecimento de pontos de referência relevantes dentro do contexto de utilização. É atribuída uma câmara por divisão;
- **IAnchor**: Interface que serve para representar as âncoras espaciais usadas em sistemas de RA, com base nas técnicas de SLAM referidas em 2.4.2.3. Uma âncora guarda um conjunto de pontos de referência para criar um modelo 3D representativo do mundo real. Neste caso, esta interface serve para gerir a criação desse tipo de conteúdo;
- **IMarker**: Interface que serve para fazer a gestão dos marcadores usados em sistemas de RA e referidos em 2.4.2.1;
- **IObject**: Interface que representa qualquer tipo de objeto ou conteúdo para ser partilhado entre os vários utilizadores do sistema. Este conteúdo pode ser um elemento com forma e posição no espaço virtual ou pode ser um elemento meramente informativo que possa servir de base à lógica da aplicação desenvolvida. O primeiro tipo referido tem ainda uma utilização especial, como avatar, quando atribuído a um utilizador;
- **IDatabase**: Interface que serve para fazer a gestão da persistência dos dados num sistema de armazenamento externo.

4.3.3 Funcionalidades

As funcionalidades oferecidas por cada uma das interfaces já descritas podem ser vistas na figura 4.4. Nos próximos parágrafos são evidenciadas as funcionalidades do sistema com maiores particularidades.

Na interface de sessão, assim como noutras que façam a gestão da integração com outras interfaces, são disponibilizados enumeradores. Neste caso, permitem oferecer a lista de utilizadores, salas, grupos e divisões presentes no sistema. Outra função a destacar desta interface é a maneira como se entra numa sala. Para além do método normal, por identificação da sala, é disponibilizada a chamada *JoinNearestRoom* para permitir que os utilizadores se juntem a uma sala com base na distância física entre o utilizador e o ponto geográfico atribuído à sala. Isto pode ser útil na criação de aplicações nas quais se pretenda limitar o acesso apenas a utilizadores co-localizados. Por fim, as chamadas *StartGame*, *EndGame* e *JoinWorldDivision* servem para fazer a gestão do estado de jogo e das divisões utilizadas. As duas primeiras chamadas fazem com que todos os utilizadores da sala entrem ou saiam da mesma divisão. A última serve para um utilizador entrar numa divisão diferente dos outros utilizadores, se o estado de jogo estiver ativo.

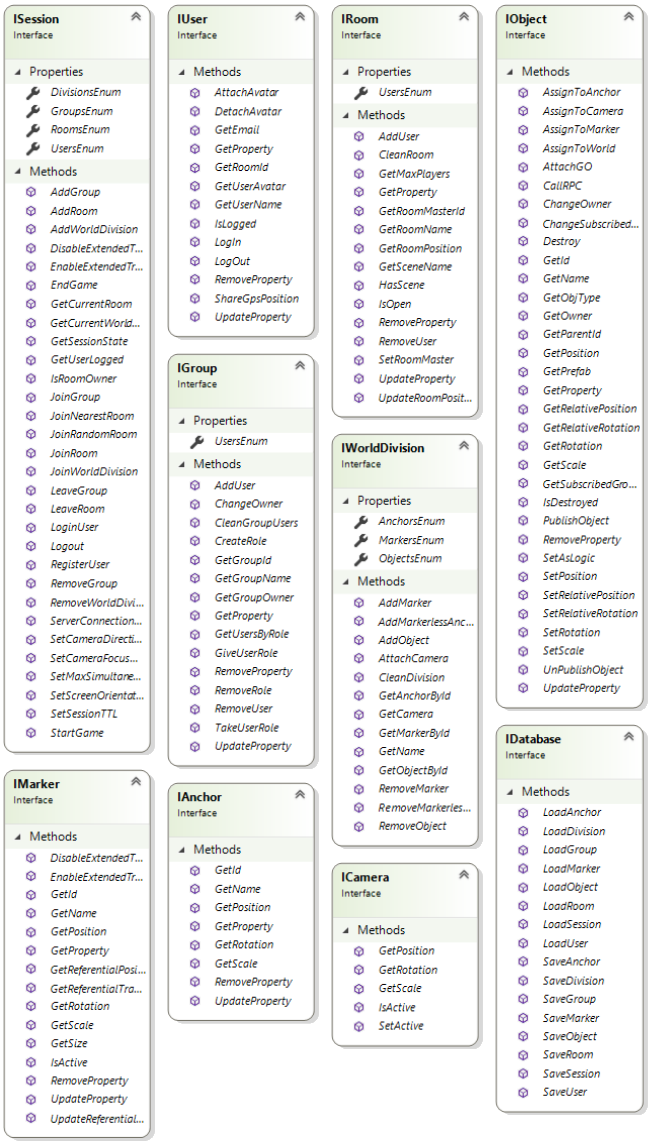

Figura 4.4: Funcionalidades da API

Na interface de utilizador, assim como na maioria das restantes, são oferecidos três métodos para definir propriedades específicas ao contexto da aplicação desenvolvida: *GetProperty*, *UpdateProperty* e *RemoveProperty*. É ainda possível anexar um objeto virtual à identificação de um utilizador ou receber a posição geográfica do mesmo, através das chamadas *AttachAvatar* e *ShareGpsPosition*, respetivamente.

A interface de sala inclui o conceito de mestre ou dono de sala, que é um utilizador com privilégios especiais e serve sobretudo como coordenador da comunicação entre pares dentro dessa sala. Se o mestre sair, é escolhido outro utilizador da sala para assumir o papel. Uma sala pode ou não ter uma interface visual associada. Essa componente é acessível através dos métodos *HasScene* e *GetSceneName*.

No que toca às interfaces do conteúdo virtual para ser disposto numa divisão, todas elas - câmara, âncora, marcador e objeto - fornecem o acesso às suas características

espaciais através dos métodos *GetPosition*, *GetRotation* e *GetScale*.

A localização de um marcador pode ser conhecida mesmo que este não esteja a ser visto pelo campo de visão da câmara, desde que a função *EnableExtendedTracking* esteja ativa. É também possível fazer o mapeamento da distância real entre dois marcadores em unidades virtuais, utilizando a função *GetReferentialTranslation*. Isto permite fazer a integração de um conjunto de marcadores num referencial comum e, assim, rastrear a posição de marcadores inativos no espaço virtual, desde que a posição de um outro marcador do referencial seja conhecida.

Um objeto virtual atualiza periodicamente as suas propriedades básicas entre os utilizadores que lhe têm acesso. Essas propriedades básicas incluem o posicionamento, o referencial associado e as propriedades parametrizáveis. No entanto, também é possível partilhar informação por rede esporadicamente, através da chamada *CallRPC*.

Para evitar sobrecarregar os canais de comunicação, são apenas propagadas por rede propriedades específicas do objeto em vez de se partilhar o objeto como um todo. Com base nessas propriedades alteradas, é replicada localmente, pelos utilizadores que estão a receber a informação, a transformação esperada do objeto.

A noção de posicionamento de um objeto está sempre dependente de uma outra referência virtual para ser interpretada. Esta referência pode ser a câmara, um marcador, uma âncora ou o espaço virtual existente. É possível alterar essa referência e a consequente posição relativa do objeto através do conjunto de métodos *Assign* e *SetRelative*.

4.4 Aplicações de Teste

No contexto desta dissertação, foram desenvolvidas duas aplicações para *smartphone* como base de suporte e de teste à infraestrutura proposta.

Estas aplicações foram concebidas em sincronia com o processo de desenvolvimento da API já apresentada neste capítulo. Isto é, primeiro foi feita uma proposta de desenho inicial da API, que depois foi sendo ligeiramente ajustada às necessidades de casos de utilização concretos, como é o caso das duas aplicações apresentadas nesta parcela do documento.

Serve esta secção para expor ao leitor os conceitos das duas aplicações, que foram fortemente inspiradas por dois padrões de jogos existentes e já bastante conhecidos.

4.4.1 Capture the Flag

Capture the Flag ou Capturar a Bandeira, em português, é um jogo tradicional jogado ao ar livre onde duas equipas têm uma bandeira para proteger num determinado local, ou base, e o objetivo é o de capturar a bandeira da outra equipa sem se ser interrompido pelos jogadores adversários que estão encarregues de proteger a sua base.[1]

[1] https://en.wikipedia.org/wiki/Capture_the_flag

Neste caso, a ideia é a de aproveitar o *campus* da faculdade como área de jogo, e combinar funções de RA com funções de localização para adaptar o conceito de jogo apresentado e testar e analisar a viabilidade da infraestrutura proposta dentro do contexto da aplicação criada.

A implementação deste jogo enquadra-se no estudo do trabalho proposto na medida em que permite avaliar as dinâmicas estratégicas exploradas pelos utilizadores num âmbito colaborativo e competitivo com recurso a tecnologias de RA.

No que toca às configurações e regras básicas desta versão adaptada, são as seguintes:

- O jogo começa com todos os jogadores na base da sua equipa;
- Cada base é representada por um marcador, que contém a bandeira da equipa;
- Ganha o jogo a equipa que fizer mais pontos dentro de um determinado período de tempo;
- Para ganhar um ponto, uma equipa tem de ir buscar a bandeira do adversário e trazê-la para a sua base. Se a bandeira da equipa não estiver na base, isto é, se o adversário também a tiver tirado, não é possível pontuar. Caso contrário, o ponto é registado e a bandeira do adversário volta à sua base;
- Cada jogador começa com dez pontos de vida e dez balas para interagir com os adversários;
- Depois de usar uma bala, será ativado um tempo de espera para o mesmo jogador poder voltar a utilizar essa função;
- Se um jogador perder todos os seus pontos de vida, este fica fora de jogo por um período de tempo e é obrigado a voltar à sua base (apontando a câmara para o marcador com a sua base);
- Os jogadores têm acesso a quatro ações: apanhar a bandeira, largar a bandeira (para pontuar), passar a bandeira a um colega de equipa e disparar sobre um adversário.

O desenho da aplicação foi pensado para suportar dois modos de jogo: RA e GPS. Os jogadores podem escolher alternar entre os dois modos a qualquer momento, ainda que no modo de GPS apenas seja permitido disparar sobre um adversário, das quatro ações referidas. Esta decisão assenta puramente na necessidade de testar o impacto das componentes de RA implementadas.

No modo de RA, os jogadores são identificados por marcadores individuais que carregam consigo. Cada marcador (dos jogadores e das bandeiras) terá botões de interação associados a uma das quatro ações referidas anteriormente.

No modo de GPS, os jogadores são identificados por pontos coloridos no mapa do *campus* da faculdade e apenas os adversários a menos de vinte metros de distância aparecem no mapa. Nestas condições, é possível disparar sobre o adversário. Por outro lado, um

jogador tem acesso à localização de todos os seus colegas de equipa. Para promover a colaboração não verbal, os jogadores têm ainda acesso de visualização sobre os adversários que estejam no campo de visão de algum colega de equipa.

4.4.2 King of the Hill

King of the Hill ou Rei da Colina, em português, é um jogo para crianças cujo objetivo é o de permanecer numa determinada área (como um monte, por exemplo) para ser considerado o "Rei da Colina". Alternativamente, também existe a adaptação do conceito a videojogos de tiro em primeira pessoa (FPS, em inglês). Neste caso, um jogador ou uma equipa tem que controlar a zona especificada durante o máximo período de tempo possível para ganhar o jogo.[2]

Neste caso, o conceito é ligeiramente diferente. A ideia é a de dividir duas equipas por duas salas físicas de iguais dimensões pré-configuradas da mesma maneira. Esta configuração passa por criar um referencial de marcadores de RA, dividido por zonas e a partir do qual os jogadores podem aferir o seu posicionamento no mundo virtual criado (desde que saibam a posição de pelo menos um dos marcadores). A disposição dos marcadores por sala pode ser vista na figura 4.5.

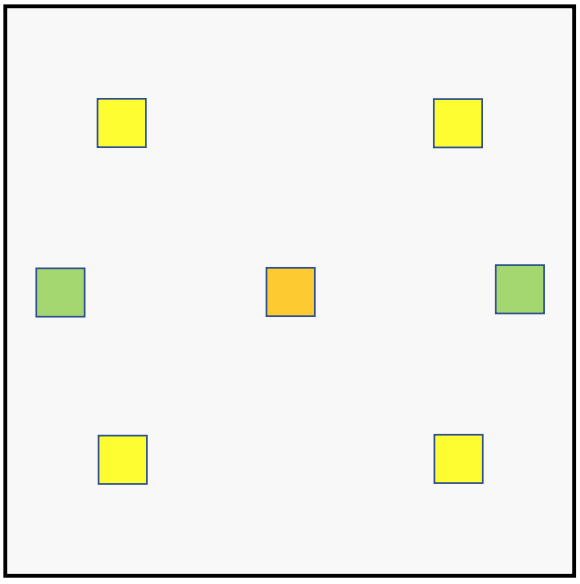

Figura 4.5: Referencial de marcadores para a aplicação *King of the Hill*

Esta configuração está dividida em três zonas: a zona amarela, que corresponde à zona de tiro; a zona verde, que corresponde à zona de bónus; e a zona laranja, que corresponde à zona da colina.

Tal como no primeiro jogo, os jogadores também têm dez pontos de vida e dez balas para interagir com os adversários e ganha a equipa que fizer mais pontos dentro de um período de tempo estipulado. São obtidos dez pontos por colina conquistada e um ponto

[2] https://en.wikipedia.org/wiki/King_of_the_Hill_(game)

por adversário eliminado. Cada vez que um jogador é eliminado, este tem de esperar alguns segundos até poder voltar a jogar.

Em relação às zonas antes referidas, na zona verde o jogador pode apanhar objetos virtuais que lhe aumentam os pontos de vida ou o número de balas. Na zona amarela, apenas se pode interagir com os jogadores adversários que se considerem na área do mesmo marcador.

Na zona laranja, está disposto um pilar com dois botões para conquistar a área. Um jogador que entre na zona laranja é visto por todos os adversários, independentemente da zona onde se encontrem. As equipas têm de arranjar maneira de colocar dois jogadores em simultâneo a carregar nos botões da zona durante alguns segundos, sem que os jogadores da equipa adversária sejam capazes de os interromper. Uma vez concluída a missão, a equipa conquista a colina, ganha dez pontos e fica imune na zona durante um período de tempo, até ser novamente possível voltar a conquistar a colina.

4.5 Integração com a Taxonomia de Brockmann

Com esta secção pretende-se avaliar o nível de abrangência da solução proposta quando integrada com a Taxonomia de Brockmann vista em 3.1.1. Para isso, será feita uma análise da amplitude de utilização da infraestrutura desta dissertação no contexto das seis dimensões propostas na taxonomia.

Posto isto, no que toca à dimensão de espaço, a solução proposta verifica as três possibilidades apresentadas. Temos o caso do primeiro jogo mencionado neste capítulo como um exemplo de um sistema de RA colaborativo executado no mesmo espaço (o *campus* da faculdade). Por outro lado, temos o exemplo do segundo jogo mencionado como um exemplo de interação mista entre os utilizadores, onde os jogadores estão localizados na mesma sala que os colegas de equipa, mas numa sala diferente da dos adversários.

Em relação à dimensão temporal, o módulo de sincronização da solução trata das interações síncronas, enquanto o módulo de persistência estará encarregue das interações assíncronas do sistema criado. No entanto, os jogos de teste concebidos apenas testam as interações em tempo real entre os utilizadores.

Quanto à vertente da mobilidade, nenhum dos cenários está limitado pela infraestrutura proposta. Será sobretudo aplicada para sistemas móveis, como é o caso denotado nas aplicações de teste, mas seria igualmente possível utilizar a infraestrutura num cenário em que a mobilidade de algum dos utilizadores do sistema tivesse de ser restringida.

O conteúdo virtual da infraestrutura é feito à base da noção de objetos da taxonomia. Também é possível gerar avatares através da infraestrutura, mas não são avatares sobrepostos nas pessoas com recurso a algoritmos de reconhecimento facial.

A dimensão da diferenciação de papéis é gerida pelo módulo de sincronização e, mais precisamente, pela interface de grupo. Sendo assim, ambas as categorias desta dimensão são possíveis de aplicar.

Por fim, a heterogeneidade dos dispositivos de visualização presentes na taxonomia (HMD, dispositivos móveis, projetores) também não fica comprometida pelo desenho proposto. Contudo, as aplicações de teste apenas exploram a utilização de dispositivos de visualização móveis e individuais.

Capítulo 5

Implementação

Neste capítulo são realçadas e analisadas as decisões tomadas no momento de implementação da infraestrutura e das aplicações de teste desenvolvidas. No final, são ainda feitas algumas considerações sobre os pontos de contenção a melhorar nas implementações apresentadas.

5.1 Infraestrutura

A implementação da infraestrutura leva em consideração os modelos de arquitetura apresentados em 4.3.1 e de que maneira as tecnologias estudadas em 3.4 podem ser integradas no desenho.

Assim, olhando para a figura 4.2, o desenvolvimento da solução foi feito ao associar as bibliotecas de rastreamento, representação e comunicação com as ferramentas de desenvolvimento do Vuforia, do Unity e do Photon Engine, respetivamente. Não foi utilizada nenhuma base de dados externa para fazer o armazenamento dos dados e, portanto, o módulo de persistência não foi implementado.

Como já tinha sido referido em 4.3.1, a utilização de bibliotecas externas para resolver os subproblemas da solução proposta é opcional. Numa outra implementação, esses problemas poderiam ser resolvidos com soluções internas inteiramente adaptadas. No entanto, o período de tempo para o fazer teria que ser consideravelmente estendido e o objetivo da dissertação também não era o de desconsiderar e competir com tecnologias já existentes e amplamente testadas.

No que toca à implementação das interfaces propostas, o desenvolvimento foi feito em C#, com recurso às bibliotecas do Unity, do Vuforia e do Photon Engine. Foi preciso criar interfaces suplementares para lidar com a configuração de *callbacks* e com a lógica interna dos objetos partilhados (*IObject*). O sistema de *callbacks* será explicado mais à

frente, em 5.1.1 e 5.1.2. As três interfaces suplementares podem ser vistas na figura 5.1.

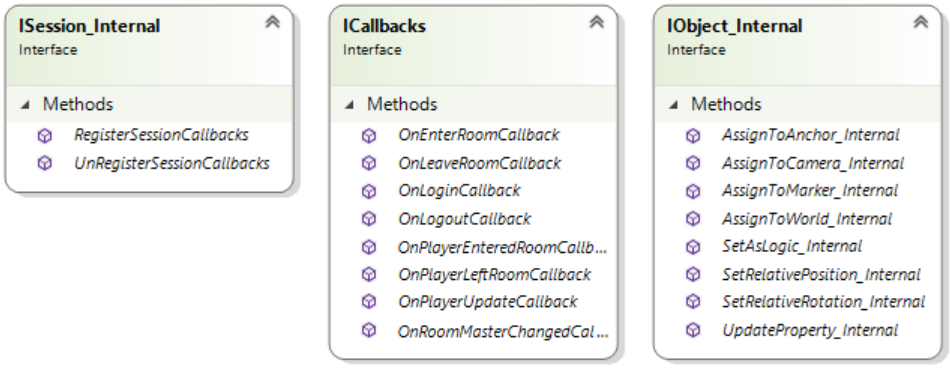

Figura 5.1: Interfaces internas

Em relação à estrutura da comunicação entre os utilizadores, esta é feita com base num servidor central de retransmissão e com um dos utilizadores a agir como cliente mestre, sendo-lhe atribuídas funções de organização de sala. Passar as mensagens por um servidor central permite gerir o fluxo de comunicação de uma maneira centralizada e consistente para todos os utilizadores.

Através desta estrutura, as mensagens de rede podem ser enviadas de três maneiras entre os utilizadores. Os três esquemas de mensagens estão ilustrados na figura 5.2[1], abstraindo a existência do servidor central. Podem ser enviadas em regime de um-para-todos ou de *broadcasting* (A), onde um utilizador envia informação para todos os membros da sala. Podem ser enviadas em regime de um-para-muitos ou de *multicasting* (B), através da utilização da interface de grupo da infraestrutura (*IGroup*). E podem ser enviadas em regime de um-para-um ou de *unicasting* (C), abstraindo um grupo ou um papel apenas a um utilizador individual.

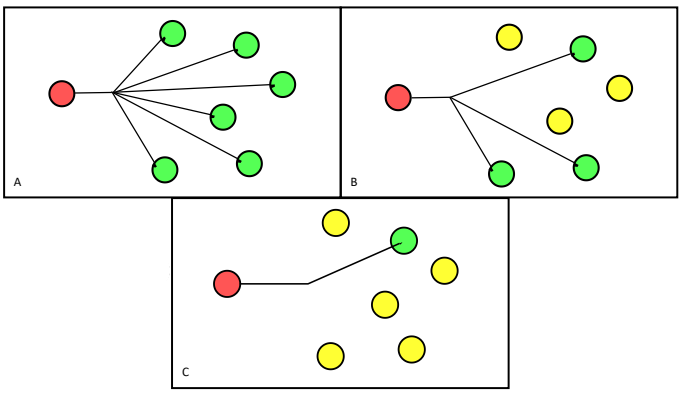

Figura 5.2: *Broadcasting*, *multicasting* e *unicasting*

[1] https://en.wikipedia.org/wiki/Broadcasting_(networking)

5.1.1 Modelos de Desenho Utilizados

Esta secção do documento serve para justificar as escolhas dos modelos de desenho selecionados durante a composição da solução apresentada.

Tendo sido a infraestrutura desenvolvida em C# e com o suporte das ferramentas de desenvolvimento do Unity e do Photon Engine, foram aproveitados os padrões de programação existentes e utilizados por estas duas soluções durante o processo de desenvolvimento da solução proposta.

Assim, de seguida serão apresentados os conceitos de:

- *Bridge*, *mediator* e injeção de dependências [34];
- *GameObject* e componentes associadas;
- *Singleton*;
- *Callback* e inversão de controlo;
- Objetos de rede;
- Chamada de procedimento remoto.

A API proposta em 4.3.2 foi criada com o paradigma de ***bridge*** - ou ponte - em consideração, isto é, dissocia a abstração (interfaces) da implementação (classes) de maneira a que as duas vertentes possam ser alteradas de forma independente. Assim, as interfaces propostas nesta dissertação não estão dependentes da maneira como são implementadas e posteriormente será possível estender ou alterar as implementações escolhidas e demonstradas neste capítulo.

Para além disso, a estrutura da solução foi desenhada com base no padrão de ***mediator*** - ou intermediário -, por meio da utilização de interfaces onde é feita a gestão de interfaces de outros tipos, como é o caso das interfaces de sessão (*ISession*) e de divisão (*IWorldDivision*) no sistema proposto. Esta decisão promove um mecanismo de acoplamento fraco entre as diferentes componentes da API, ou seja, a maioria das componentes do sistema têm pouco ou nenhum conhecimento das restantes. Um acoplamento fraco permite, portanto, a obtenção de uma solução mais modularizada.

Com o mesmo objetivo de modularizar a solução, é também utilizado o conceito de **injeção de dependências** entre as classes implementadas. O que isto quer dizer é que em nenhuma implementação é esperado que se instanciem objetos de uma classe externa. Dando um exemplo com base no sistema desenvolvido, repare-se no método *LoginUser* da interface de sessão, mostrado na listagem 5.1. Este método recebe um argumento *IUser* que vai afetar a lógica interna da classe de sessão e, assim, a dependência (de uma componente externa) é injetada. Sem a injeção de dependência, a solução alternativa passaria por instanciar um objeto que implementasse a interface *IUser* dentro do fluxo do método *LoginUser*. O que este padrão de desenho permite é que o utilizador tenha mais

controlo na instanciação e gestão de objetos, não lhe limitando o acesso a objetos que de outra maneira seriam instanciados internamente pela API.

Listagem 5.1: *LoginUser*

```
1  public void LoginUser(IUser user)
2  {
3      if (!User.IsNull(currentUser))
4          throw new Exception("UserAlreadyLoggedException");
5
6      currentUser = user;
7      if (User.IsNull(currentUser))
8          throw new Exception("UserDoesNotExistException");
9
10     currentUser.LogIn();
11     PhotonNetwork.LocalPlayer.NickName = user.GetUserName();
12     PhotonNetwork.ConnectUsingSettings();
13 }
```

Por fim, os construtores desenvolvidos na presente implementação contêm parâmetros opcionais para garantir cenários de utilização mais flexíveis da API desenvolvida.

Como a implementação da solução foi desenvolvida fazendo uso dos recursos do Unity, gerando dependência, convém esclarecer alguns dos conceitos e funcionalidades básicas utilizadas.

GameObject é a abstração de um contentor que aglomera diferentes componentes. Um *GameObject* pode representar os avatares de um jogo, um objeto de cena, as propriedades do jogo, um elemento de som, um elemento da interface ou, na verdade, qualquer elemento de uma aplicação desenvolvida em Unity. As componentes que são atribuídas ao *GameObject* ditam o tipo de objeto criado e fazem a gestão do seu estado e variáveis. Para além das componentes genéricas já oferecidas pela plataforma de desenvolvimento (como as de posicionamento ou de representação do objeto), há um tipo de componente especial que permite fazer a customização da aplicação por código: o *script*.

O *script* é uma classe que deriva da classe *MonoBehaviour* - implementada pelo Unity - para ser identificada como uma componente válida para ser anexada a um *GameObject*. A principal particularidade desta classe *MonoBehaviour* é a utilização dos métodos internos de *Start* e *Update*. O primeiro é executado assim que o *GameObject*, ao qual a componente do *script* está associada, é instanciado. O segundo é executado *frame* a *frame*, depois da instanciação do objeto.

A definição de *GameObject* é enquadrada no contexto da API desenvolvida através do padrão de *pooling* de objetos. *Pooling* é usado quando se pretende criar em cena várias instâncias do mesmo *GameObject*, através de código. No contexto do Unity, podem ser usados *GameObjects* pré-construídos (chamados *prefab*) como referência para serem clonados. Para o caso, é necessário associar um *prefab* à interface *IObject* aquando da criação de um objeto de RA partilhado, para que este venha a ser representado pela plataforma.

Singleton é o padrão de desenho que restringe a utilização de uma classe a apenas uma instância de si mesma. No caso do Unity, este padrão permite ainda a criação de um *GameObject* global, ativo em todas as cenas da aplicação desenvolvida. Como a solução proposta faz o mapeamento de uma mudança de divisão (*IWorldDivision*) numa mudança de cena no Unity, o mecanismo de *singleton* é necessário para não perder o estado da sessão de utilização (ISession) quando houver uma troca de cena. Para além disso, o desenho de uma sessão de utilização única por aplicação, limitando o uso da interface *ISession* a uma só instância, também acaba por facilitar o processo de utilização e interpretação da API proposta.

O sistema de ***callbacks*** funciona à base da inversão de controlo do fluxo normal de execução de uma aplicação, quando esta faz uso de uma infraestrutura de programação. Por fluxo normal, entenda-se que a aplicação chama processos ou métodos da infraestrutura. Ao inverter este fluxo, o que acontece é que as implementações da infraestrutura delegam e chamam métodos implementados pela aplicação. Estes métodos podem ser denominados de *callbacks*.

Este mecanismo é bastante útil para quando é necessário ficar à espera de uma resposta do servidor sem deixar o fluxo de execução da aplicação preso numa chamada da API. Por exemplo, no caso da solução apresentada, uma aplicação que use o método *LoginUser* não precisa de ficar bloqueada à espera da resposta do servidor. Nesta situação, a aplicação será notificada do sucesso da operação através da *callback OnLoginCallback*, que deverá ser implementada pela lógica da aplicação para retomar o fluxo de controlo, em vez de o perder para a infraestrutura.

O sistema de *callbacks* implementado pela solução desta dissertação é, maioritariamente, uma extensão ao sistema de *callbacks* já implementado pela infraestrutura do Photon Engine. É possível replicar e utilizar este mecanismo com uma tecnologia diferente do Photon Engine, ainda assim.

Um **objeto de rede**, simplesmente, é um *GameObject* criado e partilhado por todos os utilizadores do sistema, que lhe têm acesso. Assim, sempre que o objeto é atualizado por um dos utilizadores, esta alteração é sincronizada com os restantes atualizando as propriedades alteradas nos equivalentes duplicados remotos do objeto. Para um *GameObject* ser identificado como objeto de rede, este precisa de ter a si associada a componente *PhotonView*, disponibilizada pelo Photon Engine. O *GameObject* ou *prefab* associado à interface *IObject*, por exemplo, é um objeto de rede.

Para finalizar, a infraestrutura implementada faz também uso de um modelo de desenho à base de **chamadas de procedimento remoto** (RPC, em inglês). Uma RPC permite que um dispositivo seja capaz de invocar a execução de um método noutro dispositivo ligado por rede. Este mecanismo é útil para tratar das atualizações esporádicas dos objetos partilhados. Chamando o método *CallRPC* da interface *IObject*, o programador da aplicação pode definir o fluxo de execução de um método a ser chamado localmente por cada um dos utilizadores com acesso ao objeto partilhado.

Um ponto negativo levantado por este modelo de desenho é o problema da concorrência e a ambiguidade gerada no processo de tomada de decisão quando vários utilizadores fazem a mesma chamada *CallRPC* ao mesmo tempo, para ser executada localmente pelos outros utilizadores. Isto é o que acontece quando dois utilizadores fazem a mesma modificação sobre um objeto, no mesmo preciso momento.

Para lidar com este cenário, a solução implementada pela infraestrutura é a de resolver os conflitos com a utilização de um servidor autoritário. Para isso, um utilizador que chame uma RPC não executa localmente o método correspondente até receber o pedido do servidor retransmissor, como se viesse de um utilizador remoto. Assim, é possível fazer toda a gestão da ordem de chegada dos pedidos de RPC no servidor retransmissor. A ordem pela qual os métodos locais são executados por cada cliente é a de chegada ao servidor ou, em caso de empate, esta ordem é arbitrariamente decidida pelo servidor.

Desta forma, o programador que for utilizar a API sabe que a ordem correta das chamadas de RPC vai ser sempre respeitada. Assim, pode desenhar o código local chamado pelas RPC tendo em atenção este detalhe da ordenação. Se o objetivo de um método deste tipo for competitivo, onde só a primeira execução deve ser validada, o programador pode simplesmente utilizar uma *flag* que certifique se o método já foi executado anteriormente ou não.

Por exemplo, na aplicação *King of the Hill* desenvolvida nesta dissertação, a conquista da colina por uma das equipas faz a utilização de uma RPC que faz esta verificação com *flag* no método local, para garantir a coerência competitiva do jogo.

Para fazer uso desta solução com um servidor autoritário, foi utilizada a propriedade *RpcTarget.AllViaServer* disponibilizada pela API do Photon Engine.

5.1.2 Guia de Utilização

Esta secção serve para explicar os procedimentos necessários para criar uma aplicação de RA colaborativa, utilizando a API desenvolvida nesta dissertação dentro da plataforma de desenvolvimento do Unity.

Inicialmente, será preciso fornecer as chaves de identificação dos serviços do Vuforia e do Photon Engine. Para isso, é preciso criar uma conta de utilizador nas respetivas plataformas. Para a realização desta dissertação, foram utilizados os serviços de planeamento gratuito do Vuforia e do Photon Engine. Tendo acesso às duas chaves mencionadas, no projeto do Unity basta copiá-las para o editor de propriedades da API, ilustrado na figura 5.3.

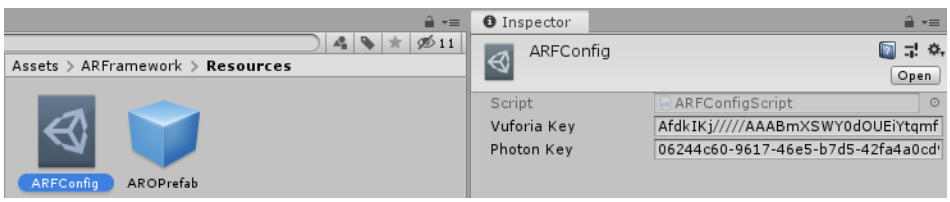

Figura 5.3: Chaves da API

De seguida, será preciso configurar a sessão de utilização individual a registar pela API. Para o fazer, basta arrastar o *singleton ARSession* para a primeira cena da aplicação que está a ser desenvolvida. Na figura 5.4 é mostrado este *singleton*, à esquerda, e a sua utilização na cena inicial das aplicações de teste desenvolvidas, à direita.

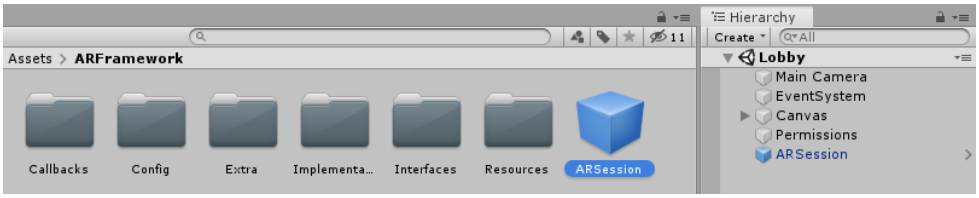

Figura 5.4: *Singleton* da sessão

Para aceder à interface *ISession* da instância guardada no *singleton*, basta utilizar a linha de código mostrada na listagem 5.2 dentro de outro *script* da aplicação que está a ser desenvolvida.

Listagem 5.2: Acesso ao *singleton* da sessão

```
1 ISession s = GameObject.Find("ARSession").GetComponent<ARSession>().mySession;
```

Caso o programador pretenda utilizar o sistema de *callbacks* oferecido pela API, terá que criar um *script* que herde a classe *ARCallbacks* em vez da classe base do Unity, uma vez que a classe *ARCallbacks*, para além de implementar as *callbacks* da API, também herda os comportamentos da classe *MonoBehaviour*.

No listagem 5.3 é mostrado um exemplo de como configurar um *script* da aplicação para ser notificado pelo sistema de *callbacks*. Neste exemplo, apenas são utilizadas as *callbacks OnLoginCallback* e *OnLogoutCallback*, mas as restantes podem ser aplicadas de igual maneira.

Listagem 5.3: Exemplo de configuração de *callbacks*

```
1  using ARFramework;
2  public class ExampleScript : ARCallbacks
3  {
4      void Start() { }
5
6      void Update() { }
7
8      public override void OnLoginCallback()
9      {
10         // your code
11     }
12
13     public override void OnLogoutCallback()
14     {
15         // your code
16     }
17 }
```

Por fim, passando à questão da instanciação de objetos partilhados (com a interface *IObject*), é oferecido um *prefab* exemplificativo a partir do qual o programador poderá criar os seus próprios objetos customizados. Este *prefab* pode ser visto na figura 5.5. Sucintamente, trata-se de um *GameObject* vazio com um *script AROPrefab* associado a uma componente de *PhotonView*. É esperado que todos os objetos de rede instanciados com a interface *IObject* tenham esta estrutura. No entanto, a partir deste exemplo, o *prefab* pode ser adaptado com as componentes adicionais que forem necessárias dentro do contexto da aplicação que estiver a ser desenvolvida.

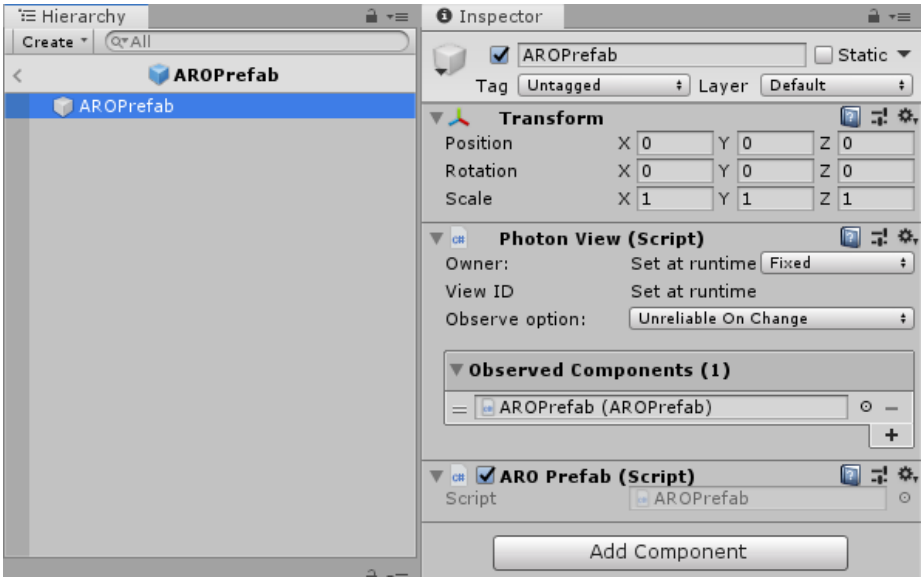

Figura 5.5: Objeto de rede de referência

O *script AROPrefab* faz a atualização periódica das variáveis principais do objeto de rede: posicionamento, rotação, escala, referencial e propriedades personalizadas. Por outro lado, para fazer atualizações esporádicas e utilizar o método *CallRPC*, o programador precisa de associar um *script* complementar ao objeto de rede, com a definição dos métodos locais a serem executados aquando da utilização da RPC. Estes métodos locais têm de ser definidos com a *tag PunRPC* e podem utilizar os argumentos que o programador considerar relevantes. Para dar um exemplo de implementação mais concreto, foi escrito o excerto de código mostrado na listagem 5.4.

Listagem 5.4: Método exemplo

```
1 [PunRPC]
2 void ExampleMethod(int n)
3 {
4     // local code
5 }
```

Ainda sobre os objetos de rede, uma questão que surgiu no âmbito desta dissertação foi o problema do mapeamento da posição de um dispositivo (por exemplo, a posição do *smartphone*) numa posição do mundo virtual partilhado.

A solução encontrada foi a de definir um referencial comum entre os utilizadores (um marcador ou uma âncora, por exemplo) e calcular a diferença posicional entre o dispositivo e o referencial. Neste caso, a posição do dispositivo é obtida com as coordenadas do *GameObject* da câmara (*ARCamera*). Para partilhar esta posição no mundo virtual sincronizado, um utilizador terá de criar um objeto de rede e atribuir o seu ponto de referência ao marcador ou à âncora que estiver a utilizar. Adicionalmente, também será preciso definir a posição relativa do objeto, entre este e o seu ponto de referência, com as coordenadas da câmara.

Para as distâncias baterem certo com unidades de medida reais (metros), os objetos e os marcadores devem ser modelados em cena assumindo que uma unidade virtual equivale a um metro. Assim, para criar um cubo de cinquenta centímetros, deve-lhe ser atribuída uma escala de 0.5 em x, y e z.

5.1.3 Requisitos e Limitações

A API desta dissertação foi desenvolvida com base nas seguintes versões das ferramentas de desenvolvimento complementares:

- Unity 2019.2.4f1
- Vuforia 8.6.10
- Photon Pun 2.14

Na tabela 5.1 são apresentados os requisitos mínimos que um dispositivo deve cumprir para poder executar uma aplicação desenvolvida com recurso à API desta dissertação. Estes requisitos têm por base as imposições necessárias para o correto funcionamento das funções de rastreamento fornecidas pelo Vuforia.

Sistema Operativo	Android[2]	iOS[3]	Windows[4]
Versão	6.0+	12+	10
IMU com Giroscópio	✓	✓	✓
Estabilização de Imagem	✓	✓	

Tabela 5.1: Requisitos mínimos dos dispositivos

Para além destes requisitos por dispositivo, o número máximo de utilizadores conectados em simultâneo à mesma aplicação é de vinte, utilizando o plano gratuito do Photon Engine. Seria possível aumentar este número para cem utilizadores, ainda com o plano gratuito, configurando um servidor local com a API do Photon Engine. Para um número maior de cem utilizadores concorrentes, já seria preciso outro tipo de plano, não gratuito.

[2] ARM 32 & 64-bit; x86 32-bit only
[3] 64-bit, Native and Bitcode
[4] 32 & 64-bit UWP 1809+; Intel 32 & 64-bit, ARM 64-bit

5.2 Aplicações

Nesta secção será explicado o processo de desenvolvimento das aplicações de teste apresentadas em 4.4, com base na estrutura da API de RA colaborativa exposta nesta dissertação. Serão relatados os passos necessários e as diferentes componentes lógicas utilizadas durante o processo.

5.2.1 Menu Inicial

Para agilizar o processo de desenvolvimento, e aproveitando as semelhanças entre os dois conceitos de aplicação propostos, foi desenhado um menu inicial comum às duas aplicações.

Este menu serve sobretudo para juntar os utilizadores na mesma sala e dividi-los por equipas. Uma vez que o módulo de persistência da infraestrutura não foi implementado e dado que o contexto de teste das aplicações não o exige, o processo de entrada numa sala não requer que um utilizador se registe no sistema. Para entrar numa sala, neste caso, basta chamar as funções *LoginUser* e *JoinRoom* da interface *ISession*.

No ambiente do Unity, é utilizada uma cena *Lobby* para fazer a gestão dos gráficos e das transições empregues neste menu. As imagens da figura 5.6 mostram o menu aqui apresentado a ser executado num dispositivo. O jogo só começa quando todos os utilizadores assinalam que estão prontos. Apenas o cliente mestre da ligação, ou mestre da sala, tem permissão para dar inicio ao jogo.

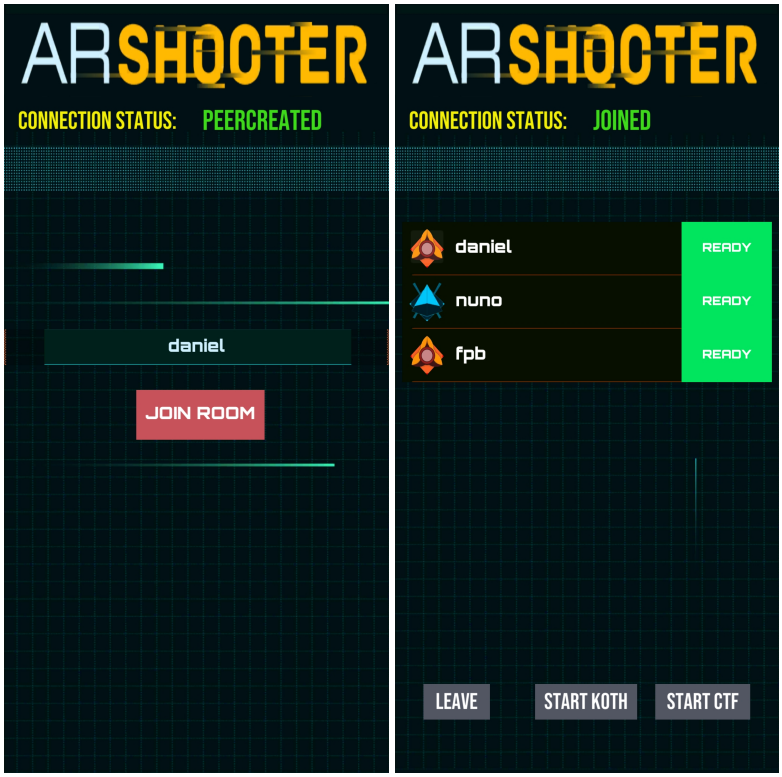

Figura 5.6: Menu inicial das aplicações

5.2.2 Capture the Flag

A primeira fase de implementação do jogo *Capture the Flag* passou por criar o conjunto de marcadores a utilizar, na plataforma do Vuforia, e importar esse conjunto para o projeto. Os marcadores das bandeiras foram modelados em cena, enquanto que os marcadores dos jogadores são processados e modelados dinamicamente (em tempo de execução), conforme o número de jogadores que entrar no jogo.

Depois de configurar o conjunto de marcadores, o desenvolvimento da aplicação pode ser explicado essencialmente através de quatro *scripts*: *CTFSetup*, *CTFLogic*, *GPSPanel* e *CTFPlayer*. O primeiro serve para fazer a configuração dos elementos de jogo com os elementos da API. Os *IObject* e *IMarker* da aplicação são criados aqui. É criado um objeto de rede especial - sem representação visual - que serve apenas para gerir a vertente lógica do jogo. O *script CTFLogic* é ligado a esse objeto e é onde os métodos locais das RPC desse objeto de rede são definidas. Por fim, os *scripts GPSPanel* e *CTFPlayer* estão ligados à lógica específica aplicada no modo de GPS.

Na figura 5.7, podem ser observados os modos de RA, à esquerda, e de GPS, à direita, num dispositivo com a aplicação em execução.

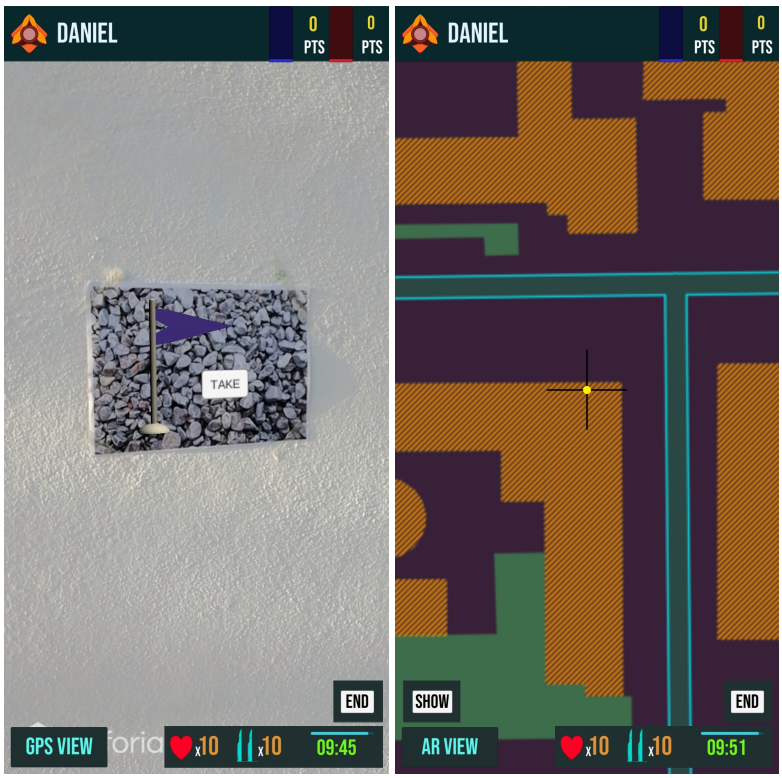

Figura 5.7: Aplicação *Capture the Flag* em execução

5.2.3 King of the Hill

O fluxo processual para implementar esta segunda aplicação não varia muito do que foi usado na primeira. Neste caso, os marcadores são usados para referenciar determinadas

zonas de um espaço fechado. Assim, quando são instanciados na API, é-lhes atribuído um valor de coordenadas correspondente ao seu posicionamento real na sala. Usando a figura 4.5 como referência, para o caso dos marcadores da zona amarela estarem a quatro metros de distância uns dos outros, ser-lhes-á atribuído, por exemplo, o conjunto de coordenadas {(0,0), (0,4), (4,0), (4,4)}.

A funcionalidade *EnableExtendedTracking* é acionada para permitir o rastreamento prolongado dos marcadores e permitir que os utilizadores não percam tão rapidamente a noção de posição dentro do espaço mapeado.

O desenvolvimento desta aplicação, tal como o da anterior, pode resumir-se a quatro *scripts*: *KOTHSetup*, *KOTHLogic*, *KOTHBonus* e *KOTHPlayer*. Os dois primeiros *scripts* têm o mesmo propósito dos *scripts* equivalentes na primeira aplicação. O *script KOTHBonus* está ligado aos objetos criados pela aplicação na zona de bónus. O último *script* está ligado aos objetos de rede criados por cada jogador, onde cada um destes objetos representa um avatar mapeado na posição do dispositivo do seu respetivo utilizador. O referencial associado a este objeto é o marcador que estiver mais próximo do utilizador.

É ainda usada a função *GetReferentialTranslation* para obter a posição de outros utilizadores com base no marcador que estiver associado ao avatar de um utilizador. Isto é, se um utilizador apenas tiver acesso à posição do marcador A e outro utilizador estiver na zona do marcador B, o que acontece é que a posição do segundo utilizador em relação ao marcador B só será aplicada depois de se fazer uma translação de A para B.

O cenário de execução da aplicação pode ser visto na figura 5.8.

Figura 5.8: Aplicação *King of the Hill* em execução

5.3 Considerações

Para terminar o capítulo, são trazidos para discussão alguns pontos das implementações relatadas, passíveis de ser melhorados, tanto da infraestrutura como das aplicações.

No caso da infraestrutura, o módulo de persistência ficou por implementar. Não é um módulo indispensável ao funcionamento do sistema, mas, sem este, não é possível criar cenários de utilização mais específicos, como é o caso de uma aplicação que pretenda utilizar um modelo de comunicação assíncrona entre os seus utilizadores.

Outro ponto que ficou por explorar foi a integração das *Cloud Anchors* da *Google* no sistema implementado. A interface *IAnchor* foi pensada com base na tecnologia de reconhecimento de planos do Vuforia (*Ground Planes*), mas numa versão seguinte o ideal seria utilizar uma mistura das duas tecnologias.

Em relação às aplicações, deve ser observado que estas não estão desenhadas nem otimizadas para um ambiente comercial. Algumas interações estão limitadas pela qualidade de processamento e de leitura dos dispositivos utilizados. Telemóveis mais antigos apresentam, inevitavelmente, desempenhos menos satisfatórios.

Os mecanismos de *feedback* (indicações visuais ou sonoras após a ação de um utilizador) oferecidos pelas aplicações também podiam ser, de um modo geral, melhores e mais refinados.

Por fim, a aplicação *Capture the Flag* podia, numa versão complementar, estender o seu modo de RA para incorporar indicações visuais de posicionamento dos seus elementos de jogo. Um exemplo deste tipo de indicações - ou pistas - seria o de incluir setas nos cantos do ecrã que estivesse na direção de uma bandeira. Outro exemplo seria mostrar elementos visuais que sugerissem a proximidade de um jogador adversário.

Capítulo 6

Testes e Resultados

Neste capítulo são descritos os testes elaborados para fazer a avaliação da solução proposta, assim como os resultados obtidos no processo. É ainda feita a análise de escalabilidade, de modularidade e de extensibilidade da infraestrutura, para complementar os resultados dos testes.

Uma nota inicial em relação ao protocolo de avaliação utilizado. Inicialmente, estava planeado avaliar as aplicações de teste com vários grupos de utilizadores para fazer uma análise qualitativa das aplicações desenvolvidas. No entanto, a atual situação de pandemia mundial da Covid-19 e as regras de distanciamento social em vigor trazem limitações aos cenários de teste pensados. Assim, e não sendo o estudo das aplicações o principal foco da dissertação, os testes por fim realizados visam aferir e validar individualmente o funcionamento e desempenho de algumas das funcionalidades oferecidas pela API concebida. Estes testes são feitos através do contexto de jogo da aplicação *King of the Hill*, onde é criado um cenário com vários utilizadores simulados num computador local. A análise dos resultados é assim toda ela feita à base de medidas de validação objetivas.

6.1 Testes Funcionais

Testes funcionais correspondem a um tipo de testes básicos utilizados para averiguar as funcionalidades do *software* desenvolvido, de acordo com a especificação e levantamento de requisitos. Neste caso, estes testes serão feitos para validar as funcionalidades da infraestrutura proposta, de acordo com os requisitos traçados em 4.2.

Foi elaborada uma sessão de teste com base na aplicação *King of the Hill*. Esta sessão foi executada por três utilizadores, todos eles com dispositivos Android. Cada utilizador realizou a sessão num local diferente e o acesso ao mesmo mundo virtual foi feito através duma configuração idêntica da disposição dos marcadores da aplicação. A comunicação

entre os utilizadores foi estabelecida por videochamada e a sessão durou cerca de uma hora.

O protocolo de ações realizadas durante a sessão é descrito de seguida.

6.1.1 Protocolo Utilizado

O guião concebido para a sessão de utilização preparada foi dividido em quinze tarefas que o grupo de utilizadores teve de executar. Estas tarefas são apresentadas na seguinte lista:

- (1) Definir nome de utilizador e carregar no botão para entrar na sala (*JOIN ROOM*);
- (2) Escolher equipa;
- (3) Mudar estado do utilizador para pronto (*READY*), dentro da sala;
- (4) Utilizador mestre da sala começa o jogo (*START KOTH*);
- (5) Encontrar um marcador para testar o reconhecimento (é exibido um cubo no local do marcador enquanto o rastreamento do mesmo estiver ativo);
- (6) Encontrar um adversário, identificado por um avatar no espaço virtual;
- (7) Disparar sobre um adversário, carregando no local do ecrã onde o avatar do adversário se encontra;
- (8) Eliminar um adversário;
- (9) Carregar num dos botões da colina a conquistar;
- (10) Dois utilizadores da mesma equipa trabalham em conjunto para conquistar a colina;
- (11) Dois utilizadores de cada equipa competem em simultâneo pela conquista da colina;
- (12) Entrar a meio do jogo;
- (13) Sair a meio do jogo (*EXIT*);
- (14) Utilizador mestre da sala acaba o jogo (*END*);
- (15) Utilizador sai da sala de jogo (*LEAVE*).

De notar que estas tarefas são de alto nível dentro do contexto da aplicação *King of the Hill*, para poderem ser realizadas por utilizadores sem quaisquer conhecimentos sobre o processo de desenvolvimento da aplicação. No entanto, convém referir que a lógica e os automatismos da aplicação desenvolvida também permitem validar outras

funcionalidades implementadas pela infraestrutura. Por exemplo, a criação dos objetos de rede - neste caso, correspondentes ao avatar de cada jogador - é tratada automaticamente pela lógica da aplicação, sem que seja preciso atribuir uma tarefa explicita aos utilizadores de teste. Assim como o posicionamento do avatar no mundo virtual também é atualizado de maneira contínua e passiva pela aplicação, dependendo do posicionamento em cena do *smartphone* do utilizador, mas não de uma tarefa explicita que necessite ser executada pelo mesmo.

Para as tarefas 11, 12 e 13, foi ainda utilizado um dispositivo complementar aos três referidos nas condições iniciais da sessão.

6.1.2 Validação de Requisitos

Para concluir o estudo funcional da infraestrutura, foi feita a validação de requisitos com base nos testes realizados. São analisados os requisitos de cada área apresentada na secção 4.2, validados e mapeados por um automatismo (A) da aplicação desenvolvida ou por alguma das tarefas apresentadas no protocolo de utilização, definido em 6.1.1. Os resultados são divididos em três possíveis atribuições, com a seguinte legenda: (T) requisito testado pela aplicação durante a sessão realizada; (I) requisito implementado pela infraestrutura que não foi diretamente testado pela aplicação; (N) requisito não implementado pela infraestrutura. Os resultados encontram-se apresentados na tabela 6.1.

O resultado N do primeiro requisito prende-se com o facto de não existir um processo de registo, uma vez que o módulo de persistência não foi implementado. Neste caso, um utilizador é criado, mas não registado.

6.2 Testes de Desempenho

Testes de desempenho são realizados para determinar como um sistema executa em termos de capacidade de resposta e de estabilidade quando sujeito a uma determinada carga de trabalho. No caso da infraestrutura criada, o objetivo é o de estudar as condições de escalabilidade do módulo de sincronização desenvolvido. Para isso, foram identificadas e analisadas as funções de comunicação mais relevantes do sistema desenvolvido.

Assim, foi criado um segundo cenário de teste para ajudar a clarificar as dúvidas relacionadas com a escalabilidade do sistema. Este cenário foi também testado com base na aplicação *King of the Hill*, mas neste caso foi apenas utilizado um computador local encarregue de executar múltiplas instâncias da aplicação, para simular os vários utilizadores ligados ao sistema.

De notar que, como a comunicação de cada utilizador é atribuída a um dos servidores hospedados na Photon Cloud (e não a um servidor local), torna-se indiferente fazer o teste com base em múltiplas instâncias no mesmo dispositivo ou com base em múltiplos dispositivos.

Requisito	**Resultado**	**Validação**
Gestão de Utilizadores		
Criar ou registar um utilizador	N	-
Iniciar ou terminar sessão	T	(1) (12) (13) (15)
Criar salas	T	A
Entrar e sair de uma sala	T	(1) (12) (13) (15)
Editar os atributos de um utilizador e de uma sala	T	(3)
Mecanismos de Grupo		
Criar um grupo	T	A
Adicionar ou remover um utilizador de um grupo	T	(2)
Enviar informação por rede para um grupo em específico	I	-
Mostrar ou ocultar conteúdo com base no grupo do utilizador	T	A
Controlo de Objetos		
Criar e destruir um objeto partilhado	T	A
Interagir com um objeto partilhado	T	(7) (8) (9) (10) (11)
Alterar o posicionamento de um objeto no espaço virtual partilhado	T	A
Atualizar a informação de um objeto de maneira periódica ou esporádica, dependendo do caso de uso	T	(7) (8) (9) (10) (11)
Persistência de Informação		
Guardar e carregar informação numa base de dados incorporada	N	-

Tabela 6.1: Validação de requisitos

O cenário descrito nesta secção foi dividido em duas partes. Uma parte com dois utilizadores e outra parte com vinte utilizadores. Cada parte foi executada cinco vezes e no final foi calculada a média do tempo de execução, para cada uma das funções de comunicação analisadas. A tabela 6.2 mostra os resultados da primeira parte do teste elaborado, com dois utilizadores envolvidos.

Em relação às funções de comunicação escolhidas, é preciso esclarecer alguns detalhes. O primeiro ponto é que os dados obtidos para o desfasamento consistem no tempo necessário para sincronizar os diferentes utilizadores na mesma sala. Neste caso, numa instância da aplicação anotou-se o tempo da *callback OnEnterRoomCallback* e na outra anotou-se o tempo da *callback OnPlayerEnteredRoomCallback*. A diferença destes dois tempos deu resultado ao desfasamento.

O segundo ponto tem a ver com a criação de objetos de rede. Neste caso, os tempos do objeto local e do objeto externo foram apontados sobre o mesmo objeto, variando apenas a instância da aplicação escolhida para registar os tempos. Sendo o objeto criado na instância A, por exemplo, este seria considerado local para A e externo para B.

Por fim, a mudança de controlador de um objeto refere-se simplesmente ao utilizador associado ao objeto. As três chamadas de procedimento remotas (RPC) escolhidas são apenas três exemplos arbitrários da funcionalidade, com base na aplicação desenvolvida.

Função						**Média**
Gestão de Sessão e de Sala						
Iniciar sessão	1401	1267	1249	1358	1264	1307.8
Entrar numa sala	343	790	706	679	758	655.2
Desfasamento	16	14	17	21	13	16.2
Terminar sessão	19	35	19	19	19	22.2
Atualização de sala	67	67	117	101	101	90.6
Atualização de utilizador	101	86	102	71	135	99
Criação de Objetos						
Objeto local	120	116	100	122	110	113.6
Objeto externo	527	491	475	658	473	524.8
Atualização de Objetos						
Mudança de controlador	265	625	546	203	218	371.4
Atualização periódica	202	188	160	180	137	173.4
HitPlayerRPC	119	150	104	92	126	118.2
PressMissionButtonRPC	163	101	90	117	100	114.2
ReleaseMissionButtonRPC	98	98	150	99	98	108.6

Tabela 6.2: Desempenho com 2 utilizadores (em milissegundos)

Para verificar o impacto da adição de utilizadores no sistema, foi então realizada a segunda parte do cenário de teste descrito nesta secção. Os resultados podem ser vistos na tabela 6.3.

Função						**Média**
Gestão de Sessão e de Sala						
Iniciar sessão	1150	899	1553	974	962	1107.6
Entrar numa sala	681	656	765	667	726	699
Desfasamento	13	14	8	16	26	15.4
Terminar sessão	14	15	8	11	14	12.4
Atualização de sala	103	137	141	169	125	135
Atualização de utilizador	115	102	87	132	94	106
Criação de Objetos						
Objeto local	365	621	487	420	473	473.2
Objeto externo	865	928	1102	816	1109	964
Atualização de Objetos						
Mudança de controlador	437	343	687	828	609	580.8
Atualização periódica	231	324	290	218	271	266.8
HitPlayerRPC	193	129	191	129	98	148
PressMissionButtonRPC	162	96	399	98	163	183.6
ReleaseMissionButtonRPC	115	90	140	130	130	121

Tabela 6.3: Desempenho com 20 utilizadores (em milissegundos)

6.2.1 Análise de Escalabilidade

Com base nos resultados obtidos nas tabelas 6.2 e 6.3, foi construída a tabela 6.4. Como seria de esperar, a maior parte das funções registou um atraso temporal adicional na condição em que o número de utilizadores é maior. Ainda assim, para uma carga de trabalho 10 vezes superior, as diferenças registadas não são, regra geral, significativas. Como referência, Gutwin [18] mostrou que valores de latência acima dos 240 milissegundos já são significativos o suficiente para causar erros de coordenação em tarefas de grupo em tempo real.

Função	**2U**	**20U**	**Diferença**
Gestão de Sessão e de Sala			
Iniciar sessão	1307.8	1107.6	-200.2
Entrar numa sala	655.2	699	+43.8
Desfasamento	16.2	15.4	-0.8
Terminar sessão	22.2	12.4	-9.8
Atualização de sala	90.6	135	+44.4
Atualização de utilizador	99	106	+7
Criação de Objetos			
Objeto local	113.6	473.2	+359.6
Objeto externo	524.8	964	+439.2
Atualização de Objetos			
Mudança de controlador	371.4	580.8	+209.4
Atualização periódica	173.4	266.8	+93.4
HitPlayerRPC	118.2	148	+29.8
PressMissionButtonRPC	114.2	183.6	+69.4
ReleaseMissionButtonRPC	108.6	121	+12.4

Tabela 6.4: Diferença de desempenho com 2 e com 20 utilizadores

Os resultados obtidos para iniciar e terminar sessão podem ser explicados à luz da independência destas funções em relação à variável de teste, que neste caso é o número de utilizadores. O que acontece é que um utilizador apenas precisa de comunicar com os servidores da Photon Cloud para executar estas funções e, portanto, está apenas dependente da latência registada entre o dispositivo e o servidor. No caso das outras funções, o servidor precisa adicionalmente de gerir e sincronizar a informação de sala referente aos restantes utilizadores e objetos.

As diferenças obtidas nestas funções, nomeadamente para iniciar sessão, podem também ser explicadas pela amostra reduzida de cinco provas experimentais para cada uma das duas condições de teste. Será de esperar que as diferenças se aproximem do 0 à medida que o tamanho da amostra aumente.

Os resultados relativos à mudança de controlador apresentam algumas discrepâncias entre as consecutivas provas experimentais apresentadas e, portanto, não é certo que a diferença apresentada entre os dois cenários de teste esteja diretamente relacionada com a variável de teste utilizada.

De resto, a única área que apresenta penalizações temporais estáveis - ao longo das provas experimentais consecutivas - e consideráveis é a da criação de objetos. No entanto, os valores observados ainda são suficientemente aceitáveis para o contexto do problema e para a condição de cariz ocasional relacionada com a criação de objetos.

Como este cenário de teste foi elaborado com base no contexto da aplicação *King of the Hill*, a única variável introduzida para aumentar a carga de trabalho foi a do número de utilizadores concorrentes. Contudo, será de esperar que um incremento no número de objetos de rede partilhados produzirá igualmente atrasos adicionais no processo de comunicação.

6.3 Análise de Software

Nesta secção, é analisado o código resultante das classes utilizadas para fazer a implementação da infraestrutura. É identificado um conjunto de métricas que servirá de base para avaliar a solução desenvolvida no que diz respeito a princípios de desenho como os da modularidade, da extensibilidade e da complexidade. Aproveitando as funcionalidades de análise disponibilizadas pelo Microsoft Visual Studio[1], destacam-se as seguintes cinco componentes para o estudo realizado nesta secção:

- IFM: Índice de Facilidade de Manutenção [35];
- CC: Complexidade Ciclomática [30];
- ACI: Acoplamento das Classes da Infraestrutura [43];
- NM: Número de Métodos;
- LC: Linhas de Código.

O IFM representa um valor de índice entre 0 e 100 que retrata a relativa facilidade de manutenção do código. Quanto maior for o valor, mais fácil é o processo de manutenção. Uma classificação entre 20 e 100 indica que o código desenvolvido é de fácil manutenção. A CC mede a complexidade do *software* desenvolvido através do número de caminhos possíveis no fluxo do programa. O ideal é que o valor da complexidade seja o mais baixo possível. ACI é a medida que verifica o número de ligações de acoplamento entre as classes utilizadas para implementar a infraestrutura. É contabilizada uma ligação por cada variável global de uma classe referente a outro objeto da infraestrutura. O NM engloba as implementações dos métodos da API (apresentados em 4.3.3), assim como os construtores e os métodos privados utilizados para implementar as respetivas classes. As LC não excluem os comentários e os segmentos de código não executáveis.

Os resultados da aplicação destas métricas à solução desenvolvida podem ser observados na tabela 6.5.

[1] https://docs.microsoft.com/en-us/visualstudio/code-quality/code-metrics-values?view=vs-2019

Classe	IFM	CC	ACI	NM	LC
Session	64	176	3	51	860
User	73	32	1	15	199
Room	73	38	0	17	208
Group	77	34	1	17	172
WorldDivision	69	51	1	17	174
Camera	77	14	0	6	83
Anchor	78	16	0	9	101
Marker	72	38	0	17	209
Object	69	101	5	43	540

Tabela 6.5: Métricas aplicadas ao código resultante da implementação da infraestrutura

A modularidade de um sistema pode ser inspecionada através do grau de interdependência das diferentes componentes do mesmo. Um sistema modular permite esconder a complexidade das partes e recombiná-las ou alterá-las se assim for necessário. Neste caso, podemos ver que o ACI da solução apresenta resultados de fraco acoplamento para todas as classes menos as de *Session* e de *Object*. Estas duas classes são também as que apresentam um maior grau de complexidade. No entanto, a complexidade apresentada não é suficientemente comprometedora para que seja necessário reestruturar o código desenvolvido. Esta conclusão pode ser corroborada pela interpretação dos valores obtidos com o IFM.

Por fim, a extensibilidade pode ser descrita como a capacidade que um sistema tem para crescer, com o mínimo de esforço empregue para adicionar novas componentes. A API proposta nesta dissertação é inteiramente extensível. O custo necessário para o fazer estará dependente de ser feita ou não a reutilização das tecnologias de suporte utilizadas na presente implementação. É ainda utilizada a noção de atributos extensíveis em grande parte das interfaces da API. Através da funcionalidade *UpdateProperty*, é possível criar e atualizar os atributos padronizados de uma classe em tempo de execução.

6.4 Problemas Relatados

Através da realização dos testes funcionais, descritos em 6.1, foram identificados alguns problemas. O primeiro foi que a infraestrutura não produziu comportamentos de rastreamento homogéneos nos diferentes dispositivos de teste. Nomeadamente, a estabilidade do rastreamento prolongado mostrou-se bastante dependente da qualidade do *hardware* utilizado. Apesar de ser um problema relacionado com a biblioteca de rastreamento incorporada (Vuforia), é algo que precisa de ser mencionado e considerado pelo impacto que pode causar na experiência de utilização.

Um problema especialmente significativo dentro do contexto da infraestrutura desenvolvida é o das atualizações contínuas dos objetos de rede. Neste caso, o problema relatado tem a ver com as atualizações de posicionamento dos avatares, que registaram

"saltos" de um lugar para outro em vez de uma movimentação contínua no espaço. O problema está relacionado com a natureza do protocolo utilizado para enviar as mensagens de rede (UDP), que não garante a entrega de todas as mensagens no recetor. Esta complicação pode ser contornada fazendo a interpolação linear entre a posição mais recente recebida por rede e a última posição do objeto registada localmente. Este método faz a aproximação das duas posições duma forma mais realista em tempo real. Ainda assim, o procedimento pode aumentar o *jitter* - ou instabilidade - do posicionamento do objeto registado em cena. Para obter resultados otimizados (entre a versão com os "saltos" e a versão com um posicionamento instável), seria preciso elaborar uma sessão de testes complementar, onde as variáveis analisadas seriam o valor do interpolante (que varia de 0 a 1) e o rácio de pacotes enviados por segundo, por cada objeto de rede. Se o número de pacotes enviados por segundo aumentar, as atualizações serão enviadas mais rapidamente e, à partida, não será preciso fazer um ajuste de posição tão acentuado através da interpolação linear.

Em relação aos mecanismos de *feedback* da aplicação, foi sugerido que se utilizassem sombras para identificar a posição dos outros utilizadores quando se aponta para o chão. Isto poderia ser facilmente alcançado sem ser preciso alterar a presente implementação da infraestrutura. Para isso, bastaria editar o modelo 3D do *prefab* utilizado para representar os avatares dos utilizadores. No entanto, também poderia ser interessante desenvolver uma funcionalidade da API que permitisse mostrar ou esconder as sombras dos objetos de rede da infraestrutura.

Outra sugestão de *feedback* foi a de limitar o realismo das oclusões. Neste caso, seria relevante tornar a colina do jogo parcialmente transparente para poder ver os avatares dos utilizadores do outro lado da colina.

Capítulo 7

Conclusões e Trabalho Futuro

Neste capítulo final são apresentadas as conclusões e reflexões sobre o trabalho realizado e são também feitas algumas recomendações para dar continuidade à investigação realizada nesta dissertação.

7.1 Conclusões

Esta dissertação descreve o processo de desenvolvimento de uma infraestrutura baseada num modelo de interação colaborativa em RA. O objetivo do trabalho realizado foi o de integrar as diversas áreas de investigação e tecnologias disponíveis para formular uma solução única que permita simplificar o processo de desenvolvimento de aplicações dentro do contexto proposto.

Foi idealizada uma solução que identificou e fez a ligação entre as diversas áreas do estudo desenvolvido. Nomeadamente, as áreas do reconhecimento e rastreamento de conteúdos, da visualização e representação dos objetos virtuais, da comunicação e sincronização da informação e do armazenamento dos dados. Desta ligação e abordagem modular surgiu uma interface de programação de aplicações de RA para múltiplos utilizadores. O modelo criado não está dependente de nenhuma tecnologia em particular, ainda que a implementação do mesmo - levada a cabo nesta dissertação - esteja.

Assim, para validar o modelo, foi feita a sua implementação com base em tecnologias já existentes. Esta implementação foi posteriormente testada com o desenvolvimento de duas provas de conceito que fizeram a utilização das funcionalidades disponibilizadas pela API implementada. Os resultados obtidos no capítulo 6 serviram sobretudo para avaliar e comprovar a adequabilidade da infraestrutura desenvolvida em relação aos princípios de escalabilidade, modularidade, manutenção e extensibilidade que um sistema desta natureza deve respeitar.

A abordagem escolhida nesta dissertação possibilita a realização de trabalho futuro complementar com base na infraestrutura proposta. Isto pode acontecer de duas maneiras. Através do acrescento - ou redefinição - de funcionalidades à solução implementada. Ou através de uma nova implementação da API com base no desenho proposto no capítulo 4, caso se verifique vantajoso alterar alguma das bibliotecas de suporte escolhidas para fazer a implementação descrita no capítulo 5. Assim, independentemente da solução implementada e das tecnologias que lhe estão subjacentes, o desenho da infraestrutura dividido por módulos garante que a mesma será flexível o suficiente para que possa ser refinada e adaptada a novos avanços tecnológicos ligados às áreas de estudo desta dissertação.

7.2 Trabalho Futuro

Para finalizar o documento, são feitas algumas recomendações de melhorias e direções de estudo a considerar com base no trabalho desenvolvido.

A primeira e mais significativa consideração a apontar é a do desenvolvimento de um módulo de persistência consistente com a solução implementada. Para isso é proposto que se utilize um sistema de armazenamento de dados externo e que se limite a sua utilização às operações indispensáveis do fluxo de execução e ao registo de atividades da infraestrutura (*log* de eventos). As operações em tempo real devem continuar a ser geridas sobretudo pelo módulo de sincronização e biblioteca de comunicação adjacente. O desenvolvimento do módulo de persistência pode ainda abrir caminho à exploração de novas áreas de investigação relacionadas com o tema, como é o caso da utilização de algoritmos de acesso à *Cloud* para fazer o carregamento dinâmico do conteúdo virtual guardado remotamente.

Seria também interessante explorar como poderiam ser integrados no contexto da infraestrutura os avanços tecnológicos do Vuforia com a atualização 9.0 e a introdução de *Area Targets* para fazer o mapeamento e integração do espaço real com o virtual. Os únicos inconvenientes da tecnologia seriam a necessidade de passar por um processo de pré-configuração complexo para fazer a leitura do espaço, assim como a obrigatoriedade de adquirir *hardware* especial para fazer essa configuração.

Por fim, as considerações feitas em 5.3 e os problemas relatados em 6.4 também devem ser tomados em conta para o desenvolvimento de qualquer trabalho complementar a esta dissertação.

De referir ainda que a iminente introdução da tecnologia 5G em larga escala - com a capacidade de enviar por rede maiores volumes de dados num período de tempo mais reduzido - poderá também servir para impulsionar o crescimento na área da RA colaborativa e assim dar origem a novas possibilidades de investigação que até hoje não eram possíveis.

Bibliografia

[1] D. Amin e S. Govilkar. "Comparative Study of Augmented Reality Sdk's". Em: *International Journal on Computational Science & Applications* 5 (fev. de 2015), pp. 11–26. DOI: 10.5121/ijcsa.2015.5102.

[2] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier e B. MacIntyre. "Recent Advances in Augmented Reality". Em: *IEEE Comput. Graph. Appl.* 21.6 (nov. de 2001), pp. 34–47. ISSN: 0272-1716. DOI: 10.1109/38.963459. URL: https://doi.org/10.1109/38.963459.

[3] R. T. Azuma. "A Survey of Augmented Reality". Em: *Presence: Teleoper. Virtual Environ.* 6.4 (ago. de 1997), pp. 355–385. ISSN: 1054-7460. DOI: 10.1162/pres.1997.6.4.355. URL: http://dx.doi.org/10.1162/pres.1997.6.4.355.

[4] M. Baldauf e P. Fröhlich. "The Augmented Video Wall: Multi-user AR Interaction with Public Displays". Em: *CHI '13 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '13. Paris, France: ACM, 2013, pp. 3015–3018. ISBN: 978-1-4503-1952-2. DOI: 10.1145/2468356.2479599. URL: http://doi.acm.org/10.1145/2468356.2479599.

[5] M. Billinghurst, I. Poupyrev, H. Kato e R. May. "Mixing realities in Shared Space: an augmented reality interface for collaborative computing". Em: *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)*. Vol. 3. Jul. de 2000, 1641–1644 vol.3. DOI: 10.1109/ICME.2000.871085.

[6] M. Billinghurst, R. Grasset e J. Looser. "Designing augmented reality interfaces". Em: *COMG*. 2005.

[7] M. Billinghurst, A. Clark e G. Lee. "A Survey of Augmented Reality". Em: *Found. Trends Hum.-Comput. Interact.* 8.2-3 (mar. de 2015), pp. 73–272. ISSN: 1551-3955. DOI: 10.1561/1100000049. URL: http://dx.doi.org/10.1561/1100000049.

[8] D. A. Bowman e L. F. Hodges. "An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments". Em: *Proceedings of the 1997 Symposium on Interactive 3D Graphics*. I3D '97. Providence, Rhode Island, USA: ACM, 1997, 35–ff. ISBN: 0-89791-884-3. DOI: 10.1145/253284.253301. URL: http://doi.acm.org/10.1145/253284.253301.

[9] T. Brockmann, N. Krüger, S. Stieglitz e I. Bohlsen. "A Framework for Collaborative Augmented Reality Applications". Em: *19th Americas Conference on Information Systems, AMCIS 2013 - Hyperconnected World: Anything, Anywhere, Anytime* 1 (ago. de 2013).

[10] P. H. Carstensen e K. Schmidt. "Computer Supported Cooperative Work: New Challenges to Systems Design". Em: *In K. Itoh (Ed.), Handbook of Human Factors*. 1999, pp. 619–636.

[11] D. Chatzopoulos, C. Bermejo, Z. Huang e P. Hui. "Mobile Augmented Reality Survey: From Where We Are to Where We Go". Em: *IEEE Access* 5 (2017), pp. 6917–6950. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2698164.

[12] P. R. Cohen, M. Dalrymple, D. B. Moran, F. C. Pereira e J. W. Sullivan. "Synergistic Use of Direct Manipulation and Natural Language". Em: *SIGCHI Bull.* 20.SI (mar. de 1989), pp. 227–233. ISSN: 0736-6906. DOI: 10.1145/67450.67494. URL: http://doi.acm.org/10.1145/67450.67494.

[13] A. Dey, M. Billinghurst, R. W. Lindeman e J. E. Swan. "A Systematic Review of 10 Years of Augmented Reality Usability Studies: 2005 to 2014". Em: *Frontiers in Robotics and AI* 5 (2018), p. 37. ISSN: 2296-9144. DOI: 10.3389/frobt.2018.00037. URL: https://www.frontiersin.org/article/10.3389/frobt.2018.00037.

[14] M. W.M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte e M. Csorba. "A solution to the simultaneous localization and map building (SLAM) problem". Em: *IEEE Transactions on Robotics and Automation* 17.3 (jun. de 2001), pp. 229–241. ISSN: 1042-296X. DOI: 10.1109/70.938381.

[15] A. Dünser, R. Grasset e M. Billinghurst. "A Survey of Evaluation Techniques Used in Augmented Reality Studies". Em: *ACM SIGGRAPH ASIA 2008* (jan. de 2008). DOI: 10.1145/1508044.1508049.

[16] S. Feiner, B. MacIntyre, T. Höllerer e A. Webster. "A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment". Em: *Personal Technologies* 1.4 (dez. de 1997), pp. 208–217. ISSN: 1617-4917. DOI: 10.1007/BF01682023. URL: https://doi.org/10.1007/BF01682023.

[17] J. L. Gabbard, D. Hix e J. E. Swan. "User-Centered Design and Evaluation of Virtual Environments". Em: *IEEE Comput. Graph. Appl.* 19.6 (nov. de 1999), pp. 51–59. ISSN: 0272-1716. DOI: 10.1109/38.799740. URL: https://doi.org/10.1109/38.799740.

[18] C. Gutwin. "The Effects of Network Delays on Group Work in Real-Time Groupware". Em: ECSCW'01. Bonn, Germany: Kluwer Academic Publishers, jan. de 2001, 299–318. ISBN: 0792371623. DOI: 10.1007/0-306-48019-0_16.

[19] A. Henrysson, M. Billinghurst e M. Ollila. "Face to Face Collaborative AR on Mobile Phones". Em: *Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality*. ISMAR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 80–89. ISBN: 0-7695-2459-1. DOI: 10.1109/ISMAR.2005.32. URL: https://doi.org/10.1109/ISMAR.2005.32.

[20] R. J. Jacob, A. Girouard, L. M. Hirshfield, M. S. Horn, O. Shaer, E. T. Solovey e J. Zigelbaum. "Reality-based Interaction: A Framework for post-WIMP Interfaces". Em: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Florence, Italy: ACM, 2008, pp. 201–210. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357089. URL: http://doi.acm.org/10.1145/1357054.1357089.

[21] N. M. Khan, X. Nan, N. Dong, Y. He, M. J. Kyan, J. James, L. Guan e C. Davis. "Towards a shared large-area mixed reality system". Em: *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* (2016), pp. 1–6.

[22] K. Kim, M. Billinghurst, G. Bruder, H. Duh e G. F. Welch. "Revisiting Trends in Augmented Reality Research: A Review of the 2nd Decade of ISMAR (2008–2017)". Em: *IEEE Transactions on Visualization and Computer Graphics* PP (set. de 2018), pp. 1–1. DOI: 10.1109/TVCG.2018.2868591.

[23] G. Klein e D. Murray. "Parallel Tracking and Mapping for Small AR Workspaces". Em: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. Nov. de 2007, pp. 225–234. DOI: 10.1109/ISMAR.2007.4538852.

[24] S. M. Ko, W. S. Chang e Y. G. Ji. "Usability Principles for Augmented Reality Applications in a Smartphone Environment". Em: *International Journal of Human–Computer Interaction* 29.8 (2013), pp. 501–515. DOI: 10.1080/10447318.2012.722466. eprint: https://doi.org/10.1080/10447318.2012.722466. URL: https://doi.org/10.1080/10447318.2012.722466.

[25] E. Kruijff, J. E. Swan e S. Feiner. "Perceptual issues in augmented reality revisited". Em: *2010 IEEE International Symposium on Mixed and Augmented Reality*. Out. de 2010, pp. 3–12. DOI: 10.1109/ISMAR.2010.5643530.

[26] G. A. Lee, A. Dünser, Seungwon Kim e M. Billinghurst. "CityViewAR: A mobile outdoor AR application for city visualization". Em: *2012 IEEE International Symposium on Mixed and Augmented Reality - Arts, Media, and Humanities (ISMAR-AMH)*. 2012, pp. 57–64.

[27] R. Likert. "A Technique for Measurement of Attitudes". Em: *Archives of Psychology*. Vol. 22. United States, jan. de 1932, pp. 1–55.

[28] E. B. Mackamul e A. Esteves. "A Look at the Effects of Handheld and Projected Augmented-reality on a Collaborative Task". Em: *Proceedings of the Symposium on Spatial User Interaction*. SUI '18. Berlin, Germany: ACM, 2018, pp. 74–78. ISBN: 978-1-4503-5708-1. DOI: 10.1145/3267782.3267793. URL: http://doi.acm.org/10.1145/3267782.3267793.

[29] T. Malone e M. Bernstein. *Handbook of Collective Intelligence*. HANDBOOK OF COLLECTIVE INTELLI. MIT Press, 2015. ISBN: 9780262029810. URL: https://books.google.pt/books?id=bENlrgEACAAJ.

[30] T. J. McCabe. "A Complexity Measure". Em: *IEEE Trans. Softw. Eng.* 2.4 (jul. de 1976), 308–320. ISSN: 0098-5589. DOI: 10.1109/TSE.1976.233837. URL: https://doi.org/10.1109/TSE.1976.233837.

[31] P. Milgram e F. Kishino. "A Taxonomy of Mixed Reality Visual Displays". Em: *IEICE Trans. Information Systems* vol. E77-D, no. 12 (dez. de 1994), pp. 1321–1329.

[32] O. Mimaroğlu. "Collaborative Augmented Reality". Tese de mestrado. National University of Ireland, Maynooth, 2014. URL: https://pdfs.semanticscholar.org/4c61/a9f343d46aa2949a3631051e9263191f130a.pdf.

[33] A. Mossel, C. Schönauer, G. Gerstweiler e H. Kaufmann. "ARTiFICe - Augmented Reality Framework for Distributed Collaboration". Em: *The International Journal of Virtual Reality* 11 (jan. de 2012). DOI: 10.20870/IJVR.2012.11.3.2845.

[34] *.NET Design Patterns*. Mar. de 2020. URL: https://www.dofactory.com/net/design-patterns.

[35] P. Oman e J. Hagemeister. "Metrics for assessing a software system's maintainability". Em: *Proceedings Conference on Software Maintenance 1992*. 1992, pp. 337–344. DOI: 10.1109/ICSM.1992.242525.

[36] P. Pantelidis, A. Chorti, I. Papagiouvanni, G. Paparoidamis, C. Drosos, T. Panagiotakopoulos, G. Lales e M. Sideris. "Virtual and Augmented Reality in Medical Education". Em: *Medical and Surgical Education*. Ed. por G. Tsoulfas. Rijeka: IntechOpen, 2018. Cap. 5. DOI: 10.5772/intechopen.71963. URL: https://doi.org/10.5772/intechopen.71963.

[37] V. Pereira, T. Matos, R. Rodrigues, R. Nóbrega e J. Jacob. "Extended Reality Framework for Remote Collaborative Interactions in Virtual Environments". Em: *2019 International Conference on Graphics and Interaction (ICGI)*. 2019, pp. 17–24.

[38] W. Piekarski e B. H. Thomas. "ARQuake: the outdoor augmented reality gaming system". Em: *Commun. ACM* 45 (2002), pp. 36–38.

[39] G. Reitmayr e D. Schmalstieg. "Mobile collaborative augmented reality". Em: *Proceedings IEEE and ACM International Symposium on Augmented Reality*. Out. de 2001, pp. 114–123. DOI: 10.1109/ISAR.2001.970521.

[40] G. Reitmayr e D. Schmalstieg. "Collaborative augmented reality for outdoor navigation and information browsing". Em: *Proceedings of the Symposium on Location Based Services and TeleCartography* (jan. de 2004).

[41] C. Rolim, D. Schmalstieg, D. Kalkofen e V. Teichrieb. "[POSTER] Design Guidelines for Generating Augmented Reality Instructions". Em: *2015 IEEE International Symposium on Mixed and Augmented Reality*. Set. de 2015, pp. 120–123. DOI: 10.1109/ISMAR.2015.36.

[42] C. Santos, T. Araujo, J. Morais e B. Meiguins. "Hybrid Approach Using Sensors, GPS and Vision Based Tracking to Improve the Registration in Mobile Augmented Reality Applications". Em: *International Journal of Multimedia and Ubiquitous Engineering* 12 (abr. de 2017), pp. 117–130. DOI: 10.14257/ijmue.2017.12.4.10.

[43] W. P. Stevens, G. J. Myers e L. L. Constantine. "Structured Design". Em: *IBM Syst. J.* 13.2 (jun. de 1974), 115–139. ISSN: 0018-8670. DOI: 10.1147/sj.132.0115. URL: https://doi.org/10.1147/sj.132.0115.

[44] S. A. K. Tareen e Z. Saleem. "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK". Em: *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)* (mar. de 2018), pp. 1–10. DOI: 10.1109/ICOMET.2018.8346440.

[45] A. Turner, M. Zeller, E. Cowley e B. Bray. *Shared experiences in mixed reality*. Mar. de 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/shared-experiences-in-mixed-reality.

[46] D. Wagner, T. Pintaric, F. Ledermann e D. Schmalstieg. "Towards Massively Multi-user Augmented Reality on Handheld Devices". Em: *Proceedings of the Third International Conference on Pervasive Computing*. PERVASIVE'05. Munich, Germany: Springer-Verlag, 2005, pp. 208–219. ISBN: 3-540-26008-0, 978-3-540-26008-0. DOI: 10.1007/11428572_13. URL: http://dx.doi.org/10.1007/11428572_13.

[47] Y. Xu, E. Barba, I. Radu, M. Gandy, R. Shemaka, B. Schrank, B. MacIntyre e T. Tseng. "Pre-patterns for designing embodied interactions in handheld augmented reality games". Em: *2011 IEEE International Symposium on Mixed and Augmented Reality - Arts, Media, and Humanities*. Out. de 2011, pp. 19–28. DOI: 10.1109/ISMAR-AMH.2011.6093652.

[48] W. Zhang, B. Han, P. Hui, V. Gopalakrishnan, E. Zavesky e F. Qian. "CARS: Collaborative Augmented Reality for Socialization". Em: *HotMobile '18*. United States: ASSOC COMPUTING MACHINERY, 2018, pp. 25–30. ISBN: 978-1-4503-5630-5. DOI: 10.1145/3177102.3177107.