



DANIEL ALEXANDRE PEDRAS FLAMINO

Licenciatura em Ciência e Engenharia Informática

GEOTILEQUERY

INTEGRAÇÃO, VISUALIZAÇÃO E ANÁLISE DE DADOS
GEORREFERENCIADOS NO UNITY

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Setembro, 2023

GEOTILEQUERY

INTEGRAÇÃO, VISUALIZAÇÃO E ANÁLISE DE DADOS GEORREFERENCIADOS NO
UNITY

DANIEL ALEXANDRE PEDRAS FLAMINO

Licenciatura em Ciência e Engenharia Informática

Orientador: Fernando Pedro Reino da Silva Birra

Professor Auxiliar, Universidade NOVA de Lisboa

Coorientador: João Carlos Gomes Moura Pires

Professor Associado, Universidade NOVA de Lisboa

Júri

Presidente: Fernanda Maria Barquinha Tavares Vieira Barbosa

Professora Auxiliar, Universidade NOVA de Lisboa

Arguente: Adriano Martins Lopes

Professor Auxiliar Convidado, ISCTE – Instituto Universitário de Lisboa

Orientador: Fernando Pedro Reino da Silva Birra

Professor Auxiliar, Universidade NOVA de Lisboa

GeoTileQuery

Integração, Visualização e Análise de Dados Georreferenciados no Unity

Copyright © Daniel Alexandre Pedras Flamino, Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa.

A Faculdade de Ciências e Tecnologia e a Universidade NOVA de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

To Diana. Keep looking up.

AGRADECIMENTOS

À Sofia, Diogo, Carolina e Rita, pelo apoio e companhia, nos momentos altos e baixos, ao longo destes longos anos.

RESUMO

Todos os dias, existem diversos tipos de desafios que afetam uma cidade grande como Lisboa, desde a recolha do lixo ao trânsito. Neste contexto, foi estabelecido o Centro Operacional Integrado de Lisboa (COI), onde é feita a gestão inteligente da cidade, através da monitorização, análise, previsão e coordenação operacional. Dentro deste existe a chamada sala de crise, onde são acompanhadas as situações mais complexas que podem ocorrer na cidade, como no caso de incêndios ou cheias.

Para permitir uma melhor compreensão dos dados e colaboração dos decisores presentes na sala de crise, ou até fora dela, o Laboratório de Dados Urbanos de Lisboa (LxDataLab) lançou um desafio para a criação de uma ferramenta colaborativa de suporte à decisão que recorra à realidade aumentada (RA). Dada a sua complexidade, este desafio pode ser dividido em quatro etapas: integração dos dados georreferenciados, criação de visualizações para estes, interação com o modelo virtual em RA e sincronização de múltiplos clientes.

Esta dissertação encontra-se focada na etapa de integração de dados. É então proposta uma *framework* que integre dados georreferenciados no motor de jogos Unity, de forma a permitir a criação de experiências 3D com base nesses dados, dado o suporte multiplataforma e capacidades gráficas do Unity. Assim, foi desenvolvido um protótipo de uma aplicação *Desktop* e *Mobile* que permite visualizar e analisar diversos conjuntos de dados num modelo 3D de Lisboa.

Esta aplicação foi também sujeita a uma avaliação relativamente à sua extensibilidade e adaptabilidade a diferentes requisitos futuros, o seu desempenho em diferentes condições e cargas de trabalho, assim como à sua usabilidade em testes com utilizadores. De um modo global, esta avaliação foi positiva, embora tenham sido notadas potenciais áreas de melhoria, como expectável.

Palavras-chave: Sistemas de Informação Geográfica, Computação Gráfica, Aplicação Móvel, Realidade Aumentada

ABSTRACT

Every day, there are different kinds of challenges that affect a big city like Lisbon, from garbage collection to traffic. In this context, the Centro Operacional Integrado de Lisboa (COI) was established, where smart city management is carried out, through monitoring, analysis, forecasting and operational coordination. Within it there is the situation room, where the most complex scenarios that may occur in the city are monitored, like fires or floods.

To enable a better understanding of the data and collaboration between the decision makers inside or outside the room, the Laboratório de Dados Urbanos de Lisboa (Lx-DataLab) launched a challenge to create a collaborative decision support tool that uses augmented reality (AR). Given its complexity, this challenge can be split into four stages: integration of the georeferenced data, creation of visualizations for it, interaction with the virtual model in AR and the synchronization of multiple clients.

This dissertation is itself focused on the data integration stage. A framework is then proposed to integrate georeferenced data into the Unity game engine, enabling the creation of 3D experiences based on this data, leveraging Unity's multiplatform support and graphic capabilities. Based on it, a Desktop and Mobile application prototype was developed which allows to view and analyze various datasets in a 3D model of Lisbon.

This application was also subject to an evaluation regarding its extensibility and adaptability to different future requirements, its performance under different conditions and workloads, as well as its usability in user tests. Overall, this evaluation was positive, although a few potential areas for improvement were identified, as expected.

Keywords: Geographic Information Systems, Computer Graphics, Mobile Application, Augmented Reality

ÍNDICE

Índice de Figuras	x
Índice de Tabelas	xii
Siglas	xiv
1 Introdução	1
1.1 Motivação e Contexto	1
1.2 Descrição do problema	2
1.3 Abordagem proposta	3
1.4 Contribuições previstas	4
1.5 Organização do documento	5
2 Estado da Arte	6
2.1 Sistemas de Informação Geográfica e dados georreferenciados	6
2.1.1 Sistemas de Informação Geográfica	6
2.1.2 Sistemas de Referência de Coordenadas (CRS)	7
2.1.3 Tipos de dados	8
2.1.4 <i>Tiles XYZ</i>	10
2.1.5 Formatos de dados	11
2.2 Realidade aumentada	14
2.2.1 Perspetiva histórica	15
2.2.2 Métodos de visualização	19
2.2.3 Técnicas de <i>tracking</i>	21
2.2.4 Técnicas de interação	25
2.3 Exemplos de Casos de Uso	28
2.4 Análise Comparativa de Soluções Existentes	29
2.4.1 ArcGIS Maps SDK for Unity	29
2.4.2 Mapbox Unity SDK	30
2.4.3 Cesium for Unity	31

2.4.4	Bing Maps SDK for Unity	31
2.4.5	Tabela sumária	32
2.5	Ferramentas	33
2.5.1	QGIS	33
2.5.2	ArcGIS CityEngine	33
2.5.3	geojson-tile-server	34
2.5.4	Unity	35
2.5.5	Nominatim	35
2.5.6	Open Source Routing Machine	36
2.5.7	ARCore	36
2.6	Resumo	37
3	Abordagem	38
3.1	Requisitos da aplicação	38
3.2	Abordagem proposta	39
3.3	Conjuntos de dados	40
3.4	Arquitetura da aplicação	42
3.4.1	Camada de Apresentação	42
3.4.2	Camada da Lógica	43
3.4.3	Camada dos Serviços de Dados	43
3.5	Resumo	44
4	Aplicação	45
4.1	Mapa	45
4.1.1	Modelo digital do terreno	46
4.1.2	Camadas Raster	47
4.1.3	Camadas Vetoriais	47
4.2	Interface	48
4.2.1	Controlos de orientação e posição da câmara	49
4.2.2	Localização	49
4.2.3	Camadas e filtros	50
4.2.4	Consulta de dados	51
4.2.5	Rota	52
4.2.6	Régua	52
4.2.7	Visualização de elevação do terreno	53
4.2.8	Modo de Realidade Aumentada	54
4.2.9	<i>Modals</i> de depuração e informação sobre a aplicação	55
4.3	Resumo	56
5	Implementação	57
5.1	Estrutura geral do sistema	57
5.2	Servidores de <i>tiles</i>	58

5.2.1	Servidores de <i>tiles</i> Raster	59
5.2.2	Servidor de <i>tiles</i> Vetoriais	59
5.3	Pré-processamento dos dados	59
5.3.1	Shapefile/CityGML para GeoJSON	61
5.3.2	GeoDatabase para Shapefile	61
5.3.3	GeoDatabase para FBX + GeoJSON	62
5.3.4	<i>Scripts</i> de normalização dos dados	62
5.4	Sistema de carregamento de Tiles	65
5.4.1	O mapa e a gestão das Tiles	65
5.4.2	Ciclo de vida das Tiles e TileLayers	66
5.5	Terreno	67
5.5.1	Modelação do terreno a partir das <i>tiles</i> Terrain-RGB da Mapbox	67
5.5.2	Desafios no uso de <i>tiles</i> para o terreno: Espaços nas fronteiras e dependências entre Tiles	70
5.6	Integração de dados georreferenciados	71
5.6.1	<i>Parser</i> de GeoJSON	71
5.6.2	Desafios de <i>floating-point precision</i> na integração de dados georreferenciados no Unity	72
5.7	Renderers	73
5.7.1	Tipos de Renderers e configuração das Camadas	73
5.7.2	Renderização dos dados	73
5.7.3	Renderização de Camadas Vetoriais	74
5.7.4	Adaptação e extensão de Renderers aos conjuntos de dados	76
5.8	Filtros	76
5.8.1	Camadas Filtráveis e suas propriedades	77
5.8.2	Configurar e aplicar os filtros	78
5.8.3	Estendendo filtros: Criação de novos tipos de filtro ou propriedades	79
5.9	Navegação	80
5.10	Consulta de dados	81
5.11	Rota	82
5.12	Régua	83
5.13	Modo de Realidade Aumentada	84
5.14	Resumo	86
6	Avaliação	87
6.1	Avaliação crítica da extensibilidade da aplicação	87
6.2	Avaliação de desempenho	88
6.2.1	Grupo L - Carregamento de área	89
6.2.2	Grupo M - Desempenho durante movimento	94
6.3	Testes de usabilidade com utilizadores	97
6.3.1	Metodologia	97

6.3.2	Resultados	104
6.3.3	Limitações da avaliação	120
6.4	Resumo	122
7	Conclusões e Trabalho Futuro	123
7.1	Conclusões	123
7.2	Trabalho futuro	124
	Bibliografia	125
	Apêndices	
A	Marcador do modo de Realidade Aumentada	134
B	Folha de rosto da sessão de testes	136
C	Formulário de Observações	138
	Anexos	
I	Questões do <i>System Usability Scale</i> (SUS) e <i>User Experience Questionnaire</i> (UEQ)	141

ÍNDICE DE FIGURAS

1.1	Dashboard com informação sobre os dispositivos móveis na cidade [63]. . .	2
2.1	A mesma localização num GCS e num PCS [37].	8
2.2	Dados vetoriais e <i>raster</i> como representações do mundo real (Adaptado de [73]).	8
2.3	Exemplo de modelo digital do terreno [23].	10
2.4	<i>Tiles</i> XYZ da área de Boston com diferentes níveis de <i>zoom</i> (Adaptado de [85]).	11
2.5	Tipos de objetos de geometria presentes no formato GeoJSON.	12
2.6	Diferentes níveis de detalhe de um edifício em CityGML [62].	13
2.7	Espetro realidade-virtualidade (Adaptado de [56]).	14
2.8	Comparação de tipos de interações [70].	15
2.9	Sword of Damocles, o sistema de realidade aumentada de Sutherland [81]. .	16
2.10	Virtual Interface Environment Workstation [84].	16
2.11	Touring Machine [26].	17
2.12	Aplicação de realidade aumentada para telemóvel [59].	17
2.13	Experiência aumentada de captura de um Pokémon [5].	18
2.14	<i>Render</i> de utilização do Vision Pro [44].	18
2.15	Sistemas de realidade aumentada projetada.	19
2.16	HoloDesk [41].	20
2.17	Aplicação de realidade aumentada para <i>smartphone</i> [6].	21
2.18	Realidade aumentada com marcadores do ARToolKit [47].	22
2.19	Correspondência de caraterísticas na imagem alvo e na capturada [53]. . . .	23
2.20	Navicam [70].	25
2.21	Augmented Chemistry [32].	27
2.22	Visualização de conjuntos de dados vetoriais no QGIS.	33
2.23	Dados dos edifícios no ArcGIS CityEngine.	34
2.24	Planos horizontais e verticais do ARCore.	37
3.1	Arquitetura da solução.	42
4.1	Diagrama dos componentes que constituem o mapa e as relações entre estes.	46

4.2	Vista principal da aplicação.	48
4.3	Detalhe da barra de ferramentas principal.	49
4.4	<i>Modal</i> para alteração da localização.	50
4.5	<i>Modal</i> das camadas com a vista de detalhe da camada selecionada aberta.	50
4.6	<i>Modal</i> com resultado da consulta de dados sobre um objeto.	51
4.7	Visualização de trajeto entre dois pontos.	52
4.8	Medição de perímetro com recurso à régua.	53
4.9	Visualização da elevação do terreno.	54
4.10	Modo de Realidade Aumentada.	54
4.11	Janela de depuração.	55
4.12	Janela de informação da aplicação.	56
5.1	Diagrama com relação entre os tipos de servidores de <i>tiles</i> e a aplicação.	58
5.2	Diagrama de transição de estados de uma <i>Tile</i>	66
5.3	Diagrama de transição de estados de uma <i>TileLayer</i>	67
5.4	Área coberta pela <i>tile</i> 15/15552/12557.	68
5.5	<i>Tile</i> criada com sistema de terreno do Unity.	69
5.6	Detalhe das fronteiras entre <i>tiles</i>	70
5.7	Resultado do <i>BuildingRenderer</i>	74
5.8	Exemplo de modelo carregado pelo <i>PrefabRenderer</i>	75
5.9	<i>Prefabs</i> instanciadas para marcar <i>Lamps</i> e <i>Trees</i>	76
5.10	Configuração do filtro de uma propriedade.	78
5.11	Efeito da ativação dos filtros.	79
5.12	<i>Dropdown</i> para seleção do valor a filtrar de variável categórica.	80
6.1	Tempos de carregamento em função do número de pedidos concorrentes.	90
6.2	Utilização de memória em função da área carregada.	91
6.3	Percursos do voo rasante e da rota e instantes de medição de métricas.	95
6.4	<i>Framerate</i> média ao longo dos dois testes.	96
6.5	Caracterização da população participante.	105
6.6	D1Q3 - Suavidade do movimento.	111
6.7	Erros nas medições com a régua.	112
6.8	Facilidade de navegação e interação.	114
6.9	E1Q3 - Facilidade e intuitividade do modo RA.	115
6.10	Interface preferida.	115
6.11	Resultados do questionário específico de usabilidade.	116
6.12	Resultados do questionário SUS.	117
6.13	Resultados do questionário UEQ [89].	118

ÍNDICE DE TABELAS

2.1	Comparação das soluções existentes.	32
3.1	Conjuntos de dados utilizados	40
3.2	Descrições dos conjuntos de dados	41
5.1	Pré-processamento dos dados vetoriais	60
6.1	Tempos médios das várias etapas (valores em ms).	93
6.2	Desempenho dos participantes no Grupo A.	106
6.3	Desempenho dos participantes no Grupo B.	106
6.4	Navegação na Tarefa B1.	107
6.5	Desempenho dos participantes no Grupo C.	108
6.6	Navegação na Tarefa C1.	108
6.7	Desempenho dos participantes no Grupo D.	110
6.8	Navegação na Tarefa D1.	110
6.9	Desempenho dos participantes no Grupo E.	112
6.10	Navegação nas Tarefas E1D1 e E1M1.	113
6.11	Controlo da câmara nas Tarefas E1D3 e E1M3.	113

ÍNDICE DE LISTAGENS

5.1	ConvertEnvironmentSensorsJsonToGeoJson.	63
5.2	FlattenTrafficRestrictionsGeoJson.	64
5.3	Exemplo de representação de modelo com ponto GeoJSON.	75
5.4	Extrato de resposta de pedido de <i>geocoding</i>	81
5.5	Extrato de resposta de pedido de <i>reverse geocoding</i>	82
5.6	Extrato de resposta de pedido de rota.	83

SIGLAS

CAD	Desenho Assistido por Computador ou <i>Computer-Aided Design</i> (p. 34)
COI	Centro Operacional Integrado de Lisboa (pp. 1, 2)
COM	<i>Concurrent Odometry and Mapping</i> (p. 36)
CRS	Sistema de Referência de Coordenadas ou <i>Coordinate Reference System</i> (pp. 7, 8, 11, 37, 61, 62)
DEM	modelo digital do terreno ou <i>digital elevation model</i> (p. 10)
GCS	Sistema de Coordenadas Geográficas ou <i>Geographic Coordinate System</i> (pp. 7, 8)
HMD	<i>head-mounted display</i> (pp. 16, 20)
HUD	<i>heads-up display</i> (p. 14)
IMU	unidade de medição inercial ou <i>inertial measurement unit</i> (pp. 24, 36)
PCS	Sistema de Coordenadas Projetadas ou <i>Projected Coordinate System</i> (pp. 7, 8)
PGIL	Plataforma de Gestão Inteligente de Lisboa (pp. 1, 2)
SDK	<i>software development kit</i> (pp. 17, 30–32, 36)
SIG	Sistemas de Informação Geográfica (pp. 6, 7, 12, 28, 29, 33, 37, 104, 118, 120)
SLAM	<i>simultaneous localization and mapping</i> (p. 36)
SUS	<i>System Usability Scale</i> (pp. ix, 103, 104, 116, 141, 142)
UEQ	<i>User Experience Questionnaire</i> (pp. ix, 103, 104, 117, 141, 142)

INTRODUÇÃO

Neste primeiro capítulo, começa-se por introduzir o contexto no qual se encontra inserida esta dissertação, tal como a motivação por detrás desta. Após isso, é descrito o problema identificado e apresentada a abordagem proposta para o solucionar. De seguida, são indicadas as principais contribuições previstas desta dissertação. Por fim, é delineada a estrutura em que está organizado o resto deste documento.

1.1 Motivação e Contexto

À semelhança de tantas outras grandes cidades, existem diversos tipos de problemas com que Lisboa é confrontada todos os dias, desde o trânsito ou a recolha eficiente do lixo, à resposta em situações de emergência como incêndios ou cheias. Dada a dimensão da cidade, a estes problemas acrescem também os desafios de processar as inúmeras fontes de informação provenientes de todas as partes da cidade e de coordenar as várias entidades responsáveis de modo a permitir a sua rápida e eficiente resolução. Para esse efeito, foi então estabelecido um centro de comando operacional do município, denominado [Centro Operacional Integrado de Lisboa \(COI\)](#), que utiliza a [Plataforma de Gestão Inteligente de Lisboa \(PGIL\)](#) como sua infraestrutura informática de suporte. A [PGIL](#) é uma solução tecnológica integradora de diversos sistemas, que consolida diversas fontes de dados do município e entidades parceiras, estáticos e em tempo real, com capacidade de monitorização, análise e gestão de todo o ecossistema urbano.

Dentro do [COI](#), existem, entre outras, uma sala de operações, uma sala para equipas de suporte e uma sala de crise e coordenação. A partir da sala de operações é feita constantemente uma monitorização em tempo real de toda a cidade e a coordenação de toda a atividade de gestão operacional bem como a respetiva comunicação. Na sala de equipas de suporte é realizado todo o trabalho técnico de previsão, análise de dados e de preparação para situações expectáveis. Por fim, na sala de crise e coordenação encontra-se o contexto deste trabalho. Como o seu nome indica, é na sala de crise do [COI](#) que são acompanhadas e coordenadas as situações de maior complexidade e urgência, como no caso de incêndios ou cheias, sendo que atualmente toda a informação necessária para este

processo é acedida através de um conjunto de *dashboards*.

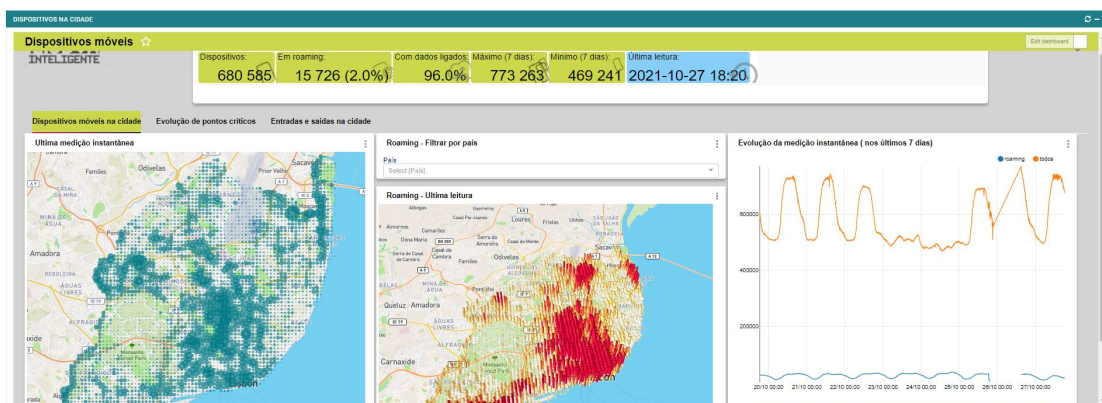


Figura 1.1: Dashboard com informação sobre os dispositivos móveis na cidade [63].

Estes *dashboards* permitem a visualização da informação presente na PGIL através de listas, gráficos, mapas, etc (Figura 1.1). No entanto, este tipo de visualizações poderão apresentar alguma dificuldade na sua compreensão devido à sua natureza técnica e bidimensionalidade, quando realmente se pretende agilizar a sua utilização quer seja por técnicos ou por chefias de topo da cidade. Assim, este trabalho surge de um desafio proposto pelo Laboratório de Dados Urbanos de Lisboa (LxDataLab) para o desenvolvimento de uma ferramenta de apoio à decisão na sala de crise do COI, que possa recorrer à realidade aumentada para ajudar na visualização dos dados.

1.2 Descrição do problema

O objetivo principal do desafio lançado é a criação de um ambiente aumentado, multiutilizador e interativo, que permita e facilite a cooperação dos diversos intervenientes na análise dos dados da cidade em situações de crise. Isto é, a aplicação deverá permitir a visualização de vários tipos de dados sobrepostos num modelo tridimensional da cidade, com o qual se pode interagir, com recurso à realidade aumentada. Como pretende-se que decisores na sala de crise ou fora dela possam reunir e efetuar decisões, será necessário também que haja múltiplos clientes que possam estar a observar a mesma visualização em simultâneo.

O desenvolvimento desta ferramenta de realidade aumentada pode ser dividido em quatro etapas, cada uma delas importante para a realização do objetivo final e com um certo grau de complexidade. Essas etapas são as seguintes:

- **Integração de dados georreferenciados** - Nesta primeira etapa, pretende-se criar uma maneira de integrar diversos tipos de dados georreferenciados que possam ser relevantes para a análise de situações de crise na cidade. Serão incluídos dados de edifícios, da rede viária e ferroviária, de infraestrutura elétrica e outros dados relevantes, em adição a cartografia base e imagens aéreas.

- **Criação de visualizações de dados** - Na segunda etapa, o objetivo será o de criar visualizações eficazes dos dados integrados. Para tal, é preciso efetuar uma escolha da melhor forma de representar os dados, desde a seleção de cores aos modelos 3D para identificar diferentes tipos de informações. Um exemplo poderia ser a representação do trânsito através da colocação de vários modelos de carros sobre a rede viária, consoante a intensidade deste. A criação de visualizações eficazes permitirá aos decisores na sala de crise uma compreensão clara e célere dos dados, tal como a sua relevância para a situação de crise.
- **Interação com o modelo virtual em Realidade Aumentada** - Na terceira etapa, é focada na interação com o modelo virtual da cidade recorrendo à realidade aumentada. Estas interações deverão permitir as operações base de navegação e exploração do mapa e também a visualização de informação associada aos objetos nele presente e a filtragem destes, entre outras, a fim de facilitar a análise dos dados.
- **Sincronização de múltiplos clientes** - Na quarta e última etapa, a visualização do modelo virtual da cidade e as interações com este serão sincronizadas entre múltiplos clientes. Deste modo, todos os decisores e participantes na gestão da situação, quer estejam fisicamente presentes na sala de crise com outro dispositivo ou fora dela por ligação remota, podem observar a mesma visualização em simultâneo e colaborar na tomada de decisões em tempo real. Numa primeira fase, pode haver um único cliente que serve de comando e interage com o modelo enquanto os restantes utilizadores veem uma versão sincronizada do modelo.

Dada a complexidade de cada uma destas etapas, este trabalho está então focado na primeira etapa, concretamente na criação de uma aplicação que possa integrar diversos tipos de dados georreferenciados. No entanto, também é feita uma pequena exploração relativamente às etapas seguintes.

1.3 Abordagem proposta

A solução proposta é uma aplicação multiplataforma que permite a visualização e análise de diversos tipos de dados georreferenciados. A aplicação foi desenvolvida utilizando o motor de jogos Unity, aproveitando o seu amplo suporte de compilação para diferentes dispositivos e sistemas operativos, como computadores Windows e *smartphones* Android, além de todas as funcionalidades típicas dos motores de jogos modernos que agilizaram a implementação do protótipo.

De um modo geral, a aplicação é um mapa que integra dados georreferenciados, separados em várias camadas, cada uma correspondendo a um conjunto de dados. No protótipo desenvolvido, existem dois tipos principais de camadas, as camadas Raster e Vetorial, ambas divididas em várias *tiles* por uma questão de eficiência e para evitar o carregamento de conjuntos de dados inteiros. No que diz respeito às camadas Vetoriais

em particular, o formato GeoJSON foi escolhido por ser capaz de representar a maior parte dos casos pretendidos e que, por ser baseado no formato JSON, permite que os dados sejam lidos com recurso a bibliotecas de *parsing* de JSON. Os dados das camadas Raster e Vetoriais são obtidos a partir de servidores de *tiles raster* e GeoJSON, sendo depois integrados no mapa presente na aplicação.

Podem ser realizados vários tipos de interações com este modelo virtual, desde a navegação básica e seleção de camadas a visualizar, à filtragem de objetos com base nas suas propriedades ou medições de distância, podendo o modelo também ser visualizado através de realidade aumentada.

1.4 Contribuições previstas

As principais contribuições que se esperam alcançar com a realização deste projeto são:

- **Criação de uma *framework* de código aberto para integração de dados georreferenciados no Unity:** Esta *framework* fornecerá uma solução flexível para incorporar dados georreferenciados em projetos do motor de jogos Unity. O objetivo desta é o de tornar mais fácil a criação de aplicações gráficas que usem dados georreferenciados, aproveitando ao mesmo tempo o amplo conjunto de funcionalidades fornecidas pelo Unity. Ao disponibilizar a *framework* como código aberto, qualquer pessoa que o pretenda fazer poderá utilizá-la na criação de aplicações e também contribuir quaisquer melhorias que faça, podendo até torná-la numa solução mais abrangente. Além disso, sendo a *framework* flexível, esta poderá ser adaptada às necessidades e requisitos específicos que existirem, o que resultará numa solução mais eficiente e eficaz para cada caso de uso.
- **Desenvolvimento de um protótipo,** utilizando a *framework* criada, no qual seja possível visualizar e analisar certos conjuntos de dados georreferenciados: O protótipo servirá para demonstrar as capacidades da *framework* de uma forma concreta e prática. Este permitirá que os utilizadores visualizem e analisem uma variedade de conjuntos de dados, *raster* e vetoriais. O objetivo deste protótipo é mostrar possíveis aplicações da *framework* e demonstrar como esta pode ser usada para dar suporte a vários casos de uso e requisitos. Além disso, o protótipo fornecerá uma oportunidade para testar e refinar a *framework*, ajudando a melhorá-la.
- **Avaliação de usabilidade e desempenho** do protótipo: Para validar a eficácia da *framework* e do protótipo desenvolvido, é fundamental realizar uma avaliação rigorosa da usabilidade e do desempenho. Isso envolve testar quão intuitiva é a interface da aplicação e como é interagida, e quão eficientemente o sistema lida com cargas de trabalho variadas. A avaliação permitirá identificar áreas que requerem aprimoramento e refinamento, contribuindo para a evolução e melhoria contínua da *framework*.

1.5 Organização do documento

Para além deste capítulo, o resto do documento está organizado com a seguinte estrutura:

- **Estado da Arte** - Para dar a entender o estado da arte e contextualizar o trabalho a desenvolver, o Capítulo 2 começa por introduzir conceitos base sobre Sistemas de Informação Geográfica e dados georreferenciados, assim como a Realidade Aumentada. De seguida, são apresentados alguns exemplos de casos de uso para os quais uma integração de dados georreferenciados pode ser útil para criar experiências 3D baseadas nestes. Após isto, é feita uma comparação entre algumas das soluções existentes, relativamente às suas capacidades e funcionalidades. Por fim, é feita uma breve descrição das várias ferramentas que foram utilizadas no desenvolvimento do protótipo ou no processamento dos dados a utilizar.
- **Abordagem** - No Capítulo 3 são apresentados os requisitos a serem cumpridos pela aplicação, seguidos pela abordagem proposta. São depois brevemente descritos os conjuntos de dados utilizados na aplicação. Após isto, é apresentada mais concretamente a arquitetura da aplicação.
- **Aplicação** - O protótipo desenvolvido é descrito no Capítulo 4, começando pelo mapa de conteúdos georreferenciados que constitui a parte central da aplicação e, de seguida, a interface desta que permite a interação com o mapa pelo utilizador.
- **Implementação** - Os diversos componentes da aplicação e os detalhes de implementação do sistema são descritos de modo mais detalhado no Capítulo 5, mais focado no processo de desenvolvimento. São também discutidos os problemas e desafios encontrados na implementação do sistema e como foram superados.
- **Avaliação** - O Capítulo 6 contém uma avaliação abrangente da aplicação dividida em três fases: crítica, desempenho e usabilidade. A avaliação crítica discute a adaptabilidade da aplicação a requisitos futuros. A avaliação de desempenho testa o carregamento de uma área e desempenho durante o movimento. A última fase aborda os testes de usabilidade, detalhando a sua metodologia, resultados e potenciais limitações.
- **Conclusões** - Por fim, no Capítulo 7 é feito um sumário do trabalho realizado e dos resultados obtidos, uma discussão destes tendo em conta os objetivos a cumprir e são apontadas sugestões para trabalho futuro.

ESTADO DA ARTE

Neste capítulo é feita uma descrição dos conceitos essenciais que contextualizam o trabalho a desenvolver e também um levantamento dos trabalhos realizados na área ou que sejam relevantes para o tema. Deste modo, começa-se por introduzir o que são os Sistemas de Informação Geográfica, alguns conceitos base, assim como tipos e formatos de dados georreferenciados. De seguida, são apresentadas definições do que é a realidade aumentada, uma perspetiva histórica desta, os diferentes métodos de visualização, técnicas de *tracking* e de interação. São depois indicados alguns exemplos de casos de uso para os quais uma integração de dados georreferenciados pode ser útil para criar experiências 3D a partir destes. Após isto, é feita uma análise comparativa entre algumas das soluções existentes, em termo das suas capacidades e funcionalidades. Por fim, é feita uma breve descrição das várias ferramentas que foram utilizadas no desenvolvimento do protótipo ou no processamento dos dados a utilizar.

2.1 Sistemas de Informação Geográfica e dados georreferenciados

Nesta secção, é feita uma exploração do que são os Sistemas de Informação Geográfica, alguns conceitos-chave como os Sistemas de Referência de Coordenadas e os dois principais modelos de dados georreferenciados: vetoriais e *raster*. Além disso, é também apresentado o padrão de *tiles* XYZ e alguns dos formatos de dados geográficos comuns em que se encontravam os conjuntos de dados utilizados neste projeto.

2.1.1 Sistemas de Informação Geográfica

Os **Sistemas de Informação Geográfica (SIG)**, como o seu nome indica, englobam *hardware* e *software* relacionados com a utilização de informação geográfica. Numa perspetiva focada no *software*, trata-se de ferramentas que permitem armazenar, processar, editar, visualizar e analisar informação geográfica e dados georreferenciados [76].

Estas tecnologias desempenham um papel crucial em várias áreas desde o planeamento e gestão urbana, a operações de logística ou até na tomada de decisões em cenários de crise

como desastres naturais, possibilitando a compreensão dos dados geográficos e, assim, auxiliando na compreensão do mundo ao nosso redor [20].

Existem diversos *softwares* SIG [76][20] comerciais proprietários como a *suite* ArcGIS¹ da Esri, que é uma das mais utilizadas na área, ou o MapInfo². Avanços recentes relativamente à Internet deram também lugar à criação de plataformas SIG baseadas na web como a Mapbox³ e a Carto⁴. Embora empresas como a Google, Microsoft ou Apple forneçam serviços de mapeamento e interfaces para navegação, estes não são geralmente considerados como sendo plataformas SIG compreensivas. Adicionalmente, existem também alternativas *open-source* como o QGIS⁵ e o GRASS⁶ disponibilizadas gratuitamente e mantidas pela comunidade.

Todos estes *softwares* SIG, independentemente do seu fornecedor, consistem em sistemas de gestão de dados capazes de processar e integrar dados espaciais e de atributos. Com dados espaciais refere-se a factos relativos à localização e posição de elementos ou fenómenos na superfície da Terra, e com atributos refere-se às características, propriedades e qualidades desses elementos ou fenómenos [76].

2.1.2 Sistemas de Referência de Coordenadas (CRS)

Quando estamos a tratar de informação geográfica, perceber onde é que algo está localizado na superfície da Terra é fundamental. Para esse efeito, existem os *Sistema de Referência de Coordenadas* ou *Coordinate Reference System (CRS)*, que fornecem uma estrutura para definir e representar posições no nosso planeta. Existem dois tipos principais de CRS [39]:

- *Sistema de Coordenadas Geográficas* ou *Geographic Coordinate System (GCS)* - definem as coordenadas num modelo da Terra (tipicamente um elipsoide ou esfera). A localização é definida por um par latitude (ângulo entre o equador e o ponto) e longitude (ângulo entre o meridiano primário e o ponto), cujas unidades são angulares, normalmente em graus. Os GCS contêm um *datum*, que é a parte que determina qual o modelo (esferoide) a utilizar para representar a superfície da Terra, existindo diversos *datums* desenhados para partes diferentes do mundo dada a irregularidade da superfície terrestre.
- *Sistema de Coordenadas Projetadas* ou *Projected Coordinate System (PCS)* - definem as coordenadas num plano. A localização é definida por um par de coordenadas cartesianas X (posição horizontal) e Y (posição vertical), cujas unidades são lineares, normalmente expressas em metros. Estes contêm um GCS e uma projeção que é utilizada para converter as coordenadas angulares nas cartesianas (Figura 2.1), daí o seu nome.

¹<http://www.esri.com/software/arcgis>

²<https://www.precisely.com/product/precisely-mapinfo/mapinfo-pro>

³<https://mapbox.com/>

⁴<https://carto.com/>

⁵<https://qgis.org/en/site/>

⁶<https://grass.osgeo.org/>

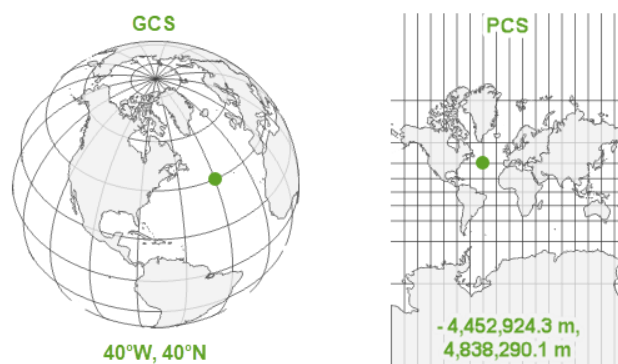


Figura 2.1: A mesma localização num GCS e num PCS [37].

Como a forma da Terra é irregular, cada **CRS** tem as suas aplicações mais adequadas, sendo necessária a escolha do que for mais apropriado para cada contexto. Existem **CRSs** específicos para certas regiões caso seja preciso diminuir a distorção a nível local e representar as informações geográficas de forma mais fidedigna. Alguns exemplos de **CRS** são o **PCS** ETRS89 / Portugal TM06 (EPSG:3763) utilizado em Portugal Continental que aplica uma projeção Transversa de Mercator ao **GCS** ETRS89 [66], o **GCS** WGS 84 (EPSG:4326) de âmbito global utilizado no sistema GPS [80] e o **PCS** Web Mercator (EPSG:3857) que é o *standard de facto* utilizado em aplicações de mapas para a Web. Utilizar um **CRS** apropriado é então importante para que seja feita uma representação, visualização e análise o mais exatas possível.

2.1.3 Tipos de dados

Os dois modelos de dados georreferenciados principais utilizados em Sistemas de Informação Geográfica para representar os elementos espaciais são dados vetoriais e dados raster (Figura 2.2).

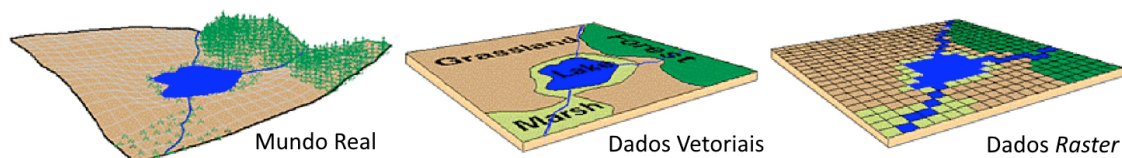


Figura 2.2: Dados vetoriais e *raster* como representações do mundo real (Adaptado de [73]).

Dados vetoriais

O modelo de dados vetoriais utiliza objetos geométricos discretos para representar os elementos espaciais presentes na superfície da Terra. Neste modelo, são utilizados os seguintes tipos vetoriais para representar os elementos espaciais [20]:

- **Pontos** - trata-se da primitiva base a partir da qual são construídas as restantes. Tem zero dimensões, sendo apenas composto por um par de coordenadas X e Y, que

indica a posição do elemento no espaço. Podem ser utilizados para marcar pontos de interesse, sinais, candeeiros, poços, etc.

- **Linhas** - são unidimensionais, passando a ter um comprimento para além da sua localização. Formada por dois pontos nas suas extremidades, podendo ter outros pontos intermédios para definir a sua forma. A forma pode, por sua vez, ser representada por segmentos de reta que ligam os vários pontos ou por curvas expressas por fórmulas matemáticas a partir dos pontos. São tipicamente utilizadas para representar estradas, cursos de água, cabos elétricos, canalizações, etc.
- **Polígonos** - são bidimensionais, tendo a definição da sua área para além do comprimento e localização. Formados por ciclos fechados de linhas que formam o seu perímetro, podem também conter outros ciclos fechados de linhas no seu interior que formam limites interiores de forma a poder representar buracos na sua área. São tipicamente utilizados para representar áreas urbanas, plantas de edifícios, lotes de terreno, corpos de água como mares e lagos, etc.

Por sua vez, estes tipos podem ser compostos para representar elementos espaciais mais complexos. Um exemplo seria a utilização de dois polígonos num mesmo elemento para representar uma propriedade composta por duas parcelas de terreno não contíguas.

Adicionalmente à sua geometria, os elementos geográficos presentes em modelos de dados vetoriais têm atributos associados a si. Estes atributos são dados adicionais relacionados com o próprio elemento e que fornecem informações extra do elemento que podem ser relevantes para a sua visualização e análise. Por exemplo, um polígono que represente um país pode ter atributos como o seu nome, população e PIB associados ou uma linha que represente uma estrada pode conter o seu nome, indicação de sentido, limite de velocidade e nível atual de trânsito.

Os dados vetoriais são então bastante versáteis e aptos para criar representações de elementos geográficos complexos e para tarefas de análise espacial.

Dados raster

Embora o modelo de dados vetoriais seja ideal para representar elementos geográficos discretos com formas e localizações bem definidas, o mesmo não se aplica a elementos contínuos como precipitação, elevação ou erosão do solo. Para esses casos, uma melhor opção para representar os dados é o modelo de dados *raster* [20].

O modelo de dados *raster* utiliza, como o nome indica, gráficos *raster* para a sua representação. São matrizes bidimensionais em que cada célula contém o valor associado ao ponto correspondente. Este modelo de dados é guardado sobre a forma de imagens *raster*, em que cada pixel corresponde a uma célula e o valor da mesma é representado pela cor do pixel. Ao estarem georreferenciados, como cada célula ou pixel tem o mesmo tamanho na grelha, cobrem parcelas de terreno de tamanho igual.

Como as células cobrem a mesma área de terreno, uma maior resolução dos dados *raster* permite que estes sejam mais detalhados e precisos, evitando que se recorra a métodos como interpolação para obter valores que correspondam a pontos intermédios entre duas células. Em contrapartida, o aumento da resolução dos dados tem a desvantagem de precisar de mais espaço de armazenamento. No entanto, como são baseados em imagens, podem beneficiar das estruturas de dados e os métodos de compressão utilizados para o armazenamento e visualização eficiente destas.

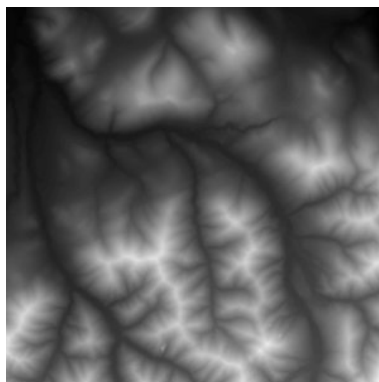


Figura 2.3: Exemplo de modelo digital do terreno [23].

Alguns casos comuns da utilização de dados *raster* são: [modelo digital do terreno ou digital elevation model \(DEM\)](#) que tipicamente representam a elevação do terreno através de imagens *grayscale* (Figura 2.3) em que tons mais escuros são usados para elevações mais baixas e tons mais claros são usados para altitudes mais elevadas; Imagens de satélite em que cada pixel corresponde a certos tipos de informação espectral; Ortofotomapas, que são imagens aéreas ou de satélite que foram corrigidas geometricamente para eliminar distorções causadas pela inclinação do terreno, a altitude do voo e outras distorções óticas; Cartografia em papel digitalizada.

2.1.4 Tiles XYZ

As *tiles* XYZ, também conhecidas como *slippy map tiles* ou *web map tiles*, são um método popular utilizado em aplicações web de mapas interativos. É um *standard de facto* para dividir dados geográficos de maneira que estes possam ser servidos através da Internet e visualizados de uma maneira mais eficiente [77]. Ao dividir o mapa em várias *tiles* de tamanho idêntico, permitem que apenas sejam carregadas as que estejam dentro da área que se pretende visualizar, diminuindo a quantidade de recursos utilizados para transmitir, processar e renderizar o conjunto de dados completo ao fazer isso apenas para um subconjunto deste.

As *tiles* são servidas a partir de servidores que habitualmente seguem o formato de URL Base/Layer/Zoom/X/Y.png, daí o nome das *tiles* XYZ. O URL tem os seguintes componentes [78]:

- *Base* - é a parte base do URL, indicando o servidor e o diretório em que se encontram as *tiles*;
- *Layer* - embora nem todos os servidores de *tiles* tenham este parâmetro, é por vezes utilizado para indicar a camada ou conjunto de dados a que as *tiles* pertencem;
- *Zoom* - o Z das *tiles* XYZ, é um número inteiro que indica o nível de ampliação do mapa. O nível 0 contém uma única *tile* que representa a totalidade da superfície terrestre. O nível 1 divide esta *tile* única em quatro outras *tiles* de tamanho idêntico entre si, sendo que cada uma destas é posteriormente divisível noutras quatro *tiles* à medida que é aumentado o nível de ampliação. Deste modo, as *tiles* estão organizadas numa estrutura piramidal semelhante a uma *quadtree* (Figura 2.4);
- X e Y - indicam a posição horizontal e vertical, respetivamente, da *tile* no mapa para o nível de ampliação anteriormente especificado. Ambos são números inteiros que identificam unicamente cada *tile* de um dado nível de ampliação, que se encontram numeradas de 0 a 2^{Zoom-1} . Na maior parte dos casos, a *tile* no canto superior esquerdo é a $Zoom/0/0$ e a do canto inferior direito é a $Zoom/2^{Zoom-1}/2^{Zoom-1}$.

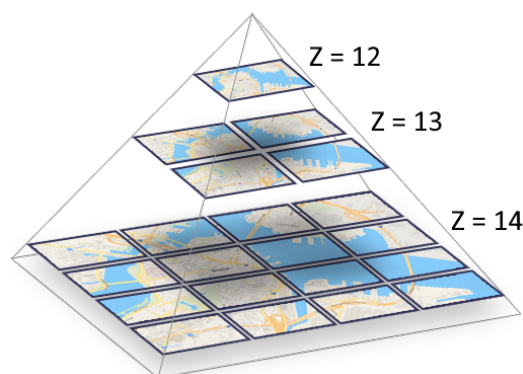


Figura 2.4: Tiles XYZ da área de Boston com diferentes níveis de zoom (Adaptado de [85]).

2.1.5 Formatos de dados

Existem diversos formatos de dados georreferenciados amplamente utilizados na indústria. Nesta secção são explorados alguns dos mais comuns (GeoJSON, Shapefile, CityGML e File GeoDatabase), nos quais se encontravam originalmente os conjuntos de dados utilizados.

GeoJSON

O GeoJSON é um formato de dados de *standard* aberto baseado no JSON, que define diversos tipos de objetos JSON e a maneira em que podem ser combinados para representar dados de elementos geográficos e as suas propriedades [14] e utiliza o CRS WGS 84 (EPSG:4326). Como é baseado no GeoJSON, a sua leitura é fácil tanto para humanos como máquinas.

Um objeto GeoJSON pode representar uma região do espaço (uma *Geometry*), uma entidade limitada espacialmente (uma *Feature*), ou um conjunto de múltiplas *Features* (uma *FeatureCollection*). O GeoJSON suporta os seguintes tipos de geometria (Figura 2.5): *Point*, *LineString*, *Polygon*, *MultiPoint*, *MultiLineString*, *MultiPolygon* e *GeometryCollection*. *Features* no GeoJSON podem conter um objeto de geometria e propriedades adicionais (que são extensíveis, podendo ser adicionados atributos customizados consoante a necessidade), sendo uma *FeatureCollection* uma lista de *Features*.

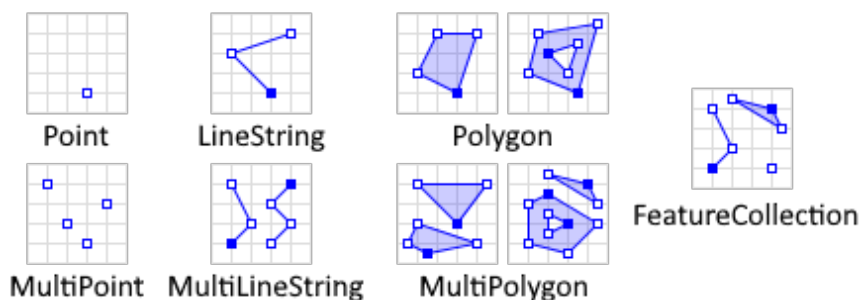


Figura 2.5: Tipos de objetos de geometria presentes no formato GeoJSON.

Desde a sua primeira publicação em 2008 [15], o formato GeoJSON tem vindo a ganhar gradualmente popularidade. É amplamente utilizado em bibliotecas JavaScript de mapeamento para a web, bases de dados de documentos baseadas em JSON e APIs.

Shapefile

O Shapefile é um dos formatos de dados vetoriais georreferenciados mais antigos e utilizado. Foi desenvolvido originalmente pela Esri, uma das principais empresas de *software SIG*, sendo compatível com os seus produtos como o ArcGIS, mas também com muitas ferramentas *SIG open source*.

O formato é composto por vários ficheiros que armazenam as informações sobre a geometria e atributos associados, servindo cada um o seu propósito, sendo os seguintes obrigatórios [24]:

- **.shp** (ficheiro principal) - Este ficheiro é o que contém os dados geométricos (como pontos, linhas ou polígonos) que representam os elementos geográficos. Contém múltiplos registos, em que cada um descreve a forma geométrica com uma lista dos seus vértices.
- **.shx** (ficheiro índice) - Neste ficheiro estão presentes os *offsets* de cada registo presente no ficheiro principal relativo ao início do ficheiro para permitir um acesso direto ao registo mais rápido.
- **.dbf** (tabela dBASE) - A tabela dBASE presente neste ficheiro contém os atributos dos elementos geográficos (como identificadores, nomes, entre outras propriedades), tendo um registo por elemento. A relação um para um entre a geometria e os atributos

está baseada no número do registo, sendo que os registos de atributos na dBASE devem estar na mesma ordem que os registos com a geometria no ficheiro principal.

CityGML

O CityGML é um modelo conceptual de dados abertos extensível e um formato de dados georreferenciados para modelos virtuais 3D de cidades. Atualmente apenas existe um *schema* para a Geography Markup Language 3 (padrão internacional para troca de dados espaciais do Open Geospatial Consortium) baseado em XML, estando também planeada a definição de especificações para a sua codificação com JSON [62].

O seu objetivo principal é estabelecer uma definição comum das entidades, atributos e relações base de modelos 3D de cidades, promovendo a consistência e reutilização de dados urbanos através de várias áreas de aplicação. Sem se limitar à representação gráfica de modelos citadinos, também se foca nas suas propriedades semânticas e temáticas, taxonomias e agregações, consistindo num modelo geométrico para definir as propriedades dos objetos espaciais e um modelo temático para utilizar o modelo geométrico em conjunção com vários campos temáticos como edifícios, modelos de terreno, vegetação, mobiliário urbano, entre outros.

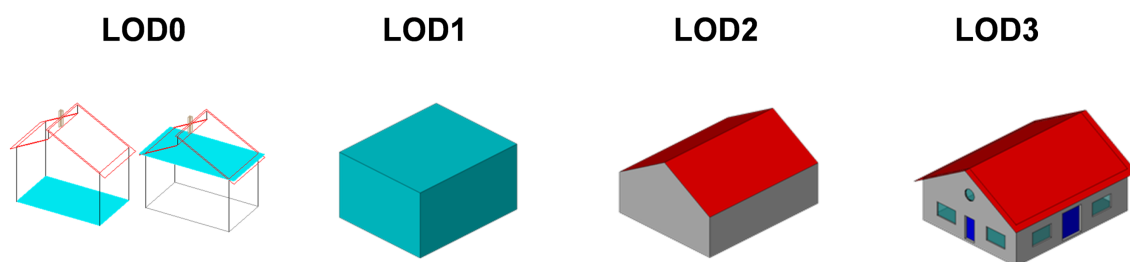


Figura 2.6: Diferentes níveis de detalhe de um edifício em CityGML [62].

Uma parte notável deste formato é a diferenciação em quatro níveis de detalhe (LOD), que permitem que os modelos sejam mais detalhados conforme a necessidade. Os ficheiros CityGML podem conter múltiplas representações dos objetos para diferentes níveis em simultâneo, como por exemplo um edifício retangular que para o nível LOD 0 é representado apenas pela sua planta, para o nível LOD 1 por um paralelepípedo, até ao nível LOD 3 em já tem detalhada as suas janelas e fachada (Figura 2.6).

Embora seja altamente eficaz para a criação de representações complexas de dados urbanos 3D, a sua modelação pode ser excessivamente detalhada para alguns casos quando comparado com formatos mais simples, como o GeoJSON ou o Shapefile.

File GeoDatabase

A File GeoDatabase é um tipo de base de dados geoespacial desenvolvida pela Esri. Trata-se de um formato proprietário utilizado dentro do ecossistema de produtos da Esri, como o ArcGIS, para armazenar, organizar e gerir múltiplos conjuntos de dados

georreferenciados através de uma estrutura baseada em ficheiros, que são contidos numa pasta com a extensão `.gdb` [87].

Ao se tratar de uma base de dados espacial permite análises mais rápidas e eficientes dos conjuntos de dados e das relações entre estes em comparação com formatos como os Shapefiles, tendo um melhor desempenho de forma geral para conjuntos de dados de maior dimensão [90]. A File GeoDatabase permite armazenar dados vetoriais, *raster* e até tabelas de dados não espaciais, sendo bastante versátil [28]. No entanto, ao se tratar de um formato proprietário implica que há um limitado suporte para interoperabilidade.

2.2 Realidade aumentada

Uma das definições mais consensuais do que é a realidade aumentada é-nos dada por Azuma, que descreve a realidade aumentada como tendo três propriedades [9]:

- combina conteúdo real e virtual;
- conteúdo interativo em tempo real;
- conteúdo registado em 3D.

Estas propriedades acabam assim por ser os requisitos principais de um sistema de realidade aumentada, que tem de ter um método de visualização capaz de combinar a imagem real com a virtual, de utilizar um ou vários métodos de *tracking* na determinação da pose do utilizador para registar o conteúdo no espaço tridimensional (este registo é o alinhamento do conteúdo virtual com o real que permite estabelecer a relação entre estes), e de utilizar técnicas de interação para que se possa interagir em tempo real com o conteúdo (seja num ecrã tátil ou com algum comando). Estes métodos e técnicas serão abordados nas próximas secções. É também de notar que estas propriedades não limitam o tipo de tecnologias utilizadas, pelo que podem ser aplicadas em experiências áudio ou hápticas, e não necessariamente visuais.

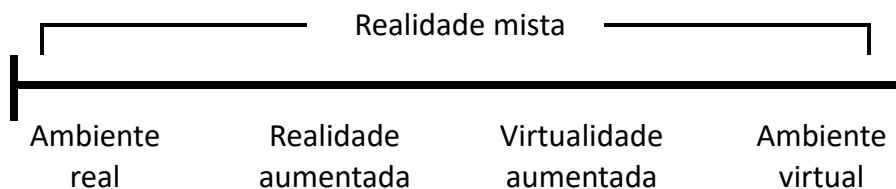


Figura 2.7: Espectro realidade-virtualidade (Adaptado de [56]).

Por sua vez, Milgram [56] define um espectro da realidade mista (Figura 2.7) que tem como extremos o ambiente real sem quaisquer modificações e um ambiente completamente virtual, sendo que os vários tipos de experiências de realidade aumentada encontram-se neste espectro ao invés de serem um ponto discreto, pois o modo em que a realidade é aumentada pode variar entre elas (como um *heads-up display* (HUD) que apenas mostre uma seta com a direção a seguir, em contraste com uma experiência em que

um avatar sobreposto de maneira imersiva sirva de guia ao utilizador como se estivesse presente fisicamente). Adicionalmente, define também um oposto à realidade aumentada, a virtualidade aumentada, em que se parte de um ambiente completamente virtual e é adicionada informação relativa ao mundo real.

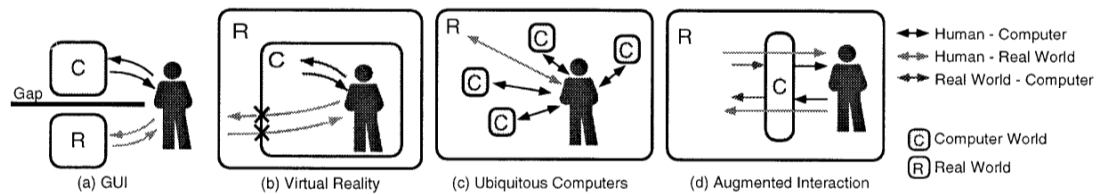


Figura 2.8: Comparação de tipos de interações [70].

Outra categorização é feita por Rekimoto [70] que foca-se nas interações com interfaces gráficas tradicionais e aquelas que tentam tornar-se invisíveis para o utilizador (Figura 2.8). Uma das categorias são os computadores ubíquos por serem integrados no mundo real e se esconderem do utilizador. Outra é, mais uma vez, a realidade virtual que, ao imergir o utilizador completamente no mundo virtual, faz com que este não note a existência da interface, embora isto seja alcançado recorrendo ao corte da ligação ao mundo real, sendo necessário um campo de visão bastante grande e gráficos realistas, podendo no entanto utilizar um método de *tracking* menos preciso pois o utilizador deixa de ver o mundo real. Mantendo a ligação ao mundo real surge a realidade aumentada, que contrasta com a realidade virtual ao não precisar de preencher completamente o campo de visão do utilizador e podendo utilizar gráficos mais simplistas, mas em contrapartida necessita de técnicas de *tracking* mais precisas para efetuar o registo pois é fácil para o utilizador notar discrepâncias entre o mundo real e os objetos virtuais, quebrando então a imersão na experiência aumentada.

A definição de Azuma por si só já fornece uma boa descrição das experiências de realidade aumentada, mas existem outras várias classificações e taxonomias para além das apresentadas e, embora não sejam todas sempre relevantes, é importante considerá-las para se ter uma ideia completa do que é a realidade aumentada.

2.2.1 Perspetiva histórica

Embora a realidade aumentada tenha visto a sua popularidade aumentar nos últimos anos, não se trata de uma área completamente nova, sendo que um dos primeiros sistemas de realidade aumentada foi desenvolvido em 1968 pelo "pai da computação gráfica", Ivan Sutherland [81]. O sistema consistia num HMD *optical see-through* ligado a um sistema de *tracking* mecânico que determinava a posição da cabeça do utilizador, que estava ligado a um computador com *hardware* próprio para processamento de gráficos (Figura 2.9). Através deste sistema era possível visualizar um modelo *wireframe* sobreposto sobre a vista da sala em que o utilizador se encontrava. Esta investigação de realidade aumentada era um passo inicial para a concretização da sua visão de um Ultimate Display [82] em que

um computador poderia controlar a matéria, dando o exemplo de se poder sentar numa cadeira criada por esse dispositivo.

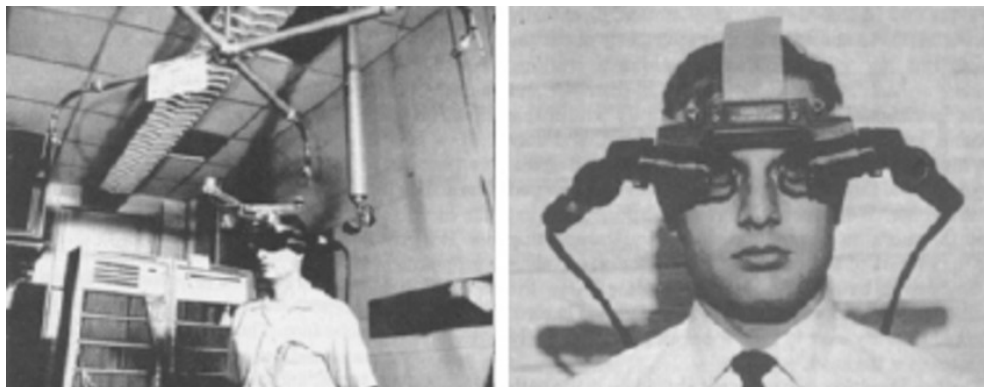


Figura 2.9: Sword of Damocles, o sistema de realidade aumentada de Sutherland [81].

Nas décadas seguintes, a maior parte do trabalho seria realizado, não em contextos de investigação académicos ou na indústria, mas sim em laboratórios governamentais e militares, como era o caso do programa SuperCockpit [34] em que foram estudadas novas maneiras de apresentar a informação necessária aos pilotos, com uma interface sobreposta sobre a imagem real durante o dia e a substituição total durante a noite quando não fosse possível ver nada. Outro exemplo é o Virtual Interface Environment Workstation (VIEW) da NASA (Figura 2.10) que foi utilizado para investigar possíveis interfaces para os astronautas controlarem remotamente robôs ou para visualização imersiva de dados [31].



Figura 2.10: Virtual Interface Environment Workstation [84].

No final da década de 80 e inícios da década de 90, estas tecnologias começam a estar ao alcance dos laboratórios de investigação académicos e começa-se a efetuar os primeiros estudos de várias técnicas e tecnologias como *head-mounted display (HMD) see-through* [71], registo e *tracking* [7, 8], realidade aumentada portátil (Figura 2.11), entre outras.

Devido à necessidade da realidade aumentada ter uma alta precisão na determinação da pose do utilizador, a maior parte destes sistemas iniciais utilizavam *tracking* magnético que era bastante dispendioso. Como alternativa surgiram desenvolvimentos como [69] e



Figura 2.11: Touring Machine [26].

posteriormente a biblioteca *open-source* ARToolKit [48, 47] que permitiam a utilização de marcadores impressos para determinar a localização. Outros desenvolvimentos relevantes incluem a NaviCam [70] e a primeira aplicação de realidade aumentada para telemóveis (Figura 2.12).

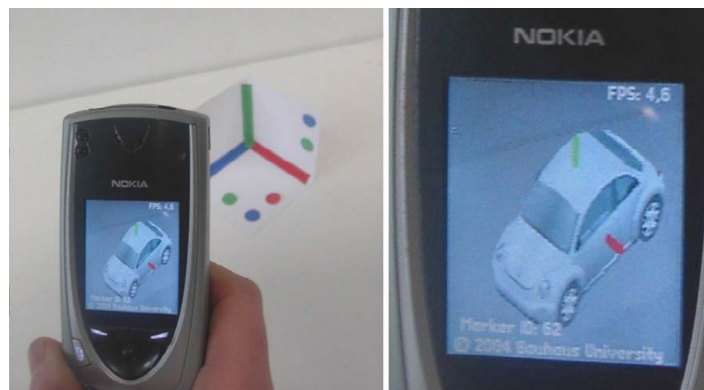


Figura 2.12: Aplicação de realidade aumentada para telemóvel [59].

Mesmo com estes desenvolvimentos, no que respeita à realidade aumentada móvel, ainda havia demasiadas limitações em termos de *hardware*. No entanto, isso mudou com o aparecimento e massificação dos *smartphones* e *tablets*, cada vez dotados com maior capacidade de processamento e múltiplos tipos de sensores, o que desencadeia o surgimento de *frameworks* como o Wikitude⁷ em 2008, Layar e Junaio (este último adquirido pela Apple em 2015).

Nos anos seguintes, a área de realidade aumentada começou a mover-se mais rapidamente. O lançamento dos *software development kit* (SDK)s Apple ARKit⁸ em 2017 e do Google ARCore⁹ em 2018 ajudou consideravelmente o desenvolvimento de experiências aumentadas móveis. Este foi um passo crucial para uma adoção mais generalizada da realidade aumentada, com o aparecimento de vários tipos de aplicações de RA nas respetivas *app stores*.

⁷<https://www.wikitude.com>

⁸<https://developer.apple.com/augmented-reality/>

⁹<https://developers.google.com/ar>

Uma das utilizações mais famosas foi o jogo Pokémon GO que permite que jogadores explorem o mundo real enquanto capturam criaturas Pokémon virtuais e batalham e interagem com outros jogadores. Enquanto se tenta capturar um Pokémon, é possível ter o modo de realidade aumentada ativo, que representa a criatura como estando presente no mundo físico em frente do jogador (Figura 2.13). Embora inicialmente utilizasse uma *framework* própria, foi adicionado suporte para o ARKit [5] e, posteriormente, para o ARCore.



Figura 2.13: Experiência aumentada de captura de um Pokémon [5].

O campo de óculos de realidade aumentada e mista também viu um desenvolvimento considerável desde a introdução dos Google Glass, com a HoloLens da Microsoft e os óculos Magic Leap. Estes dispositivos já contêm sensores mais avançados, assim como a sua ótica e poder de processamento. Mais recentemente a Apple anunciou o Vision Pro (Figura 2.14), óculos de realidade mista que pretendem que seja visto mais como um computador espacial que faz a junção do mundo físico ao virtual [44].

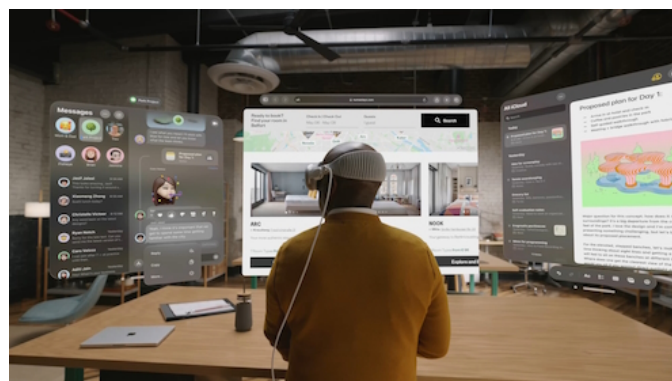


Figura 2.14: Render de utilização do Vision Pro [44].

Olhando para o futuro, o campo da realidade aumentada continuará a evoluir. Avanços na visão computacional, *machine learning* e conectividade 5G prometem melhorar as experiências de RA, tornando-as cada vez mais imersivas e transparentes.

2.2.2 Métodos de visualização

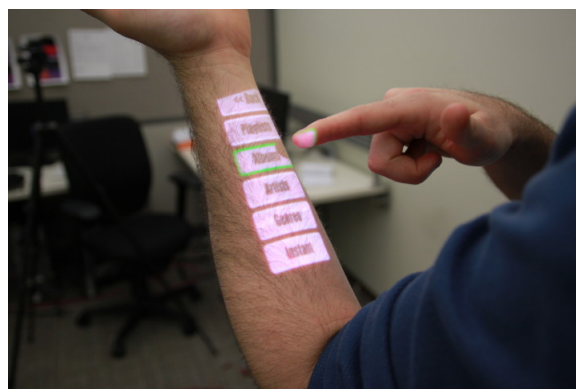
Como foi descrito, uma das propriedades que definem um sistema de realidade aumentada segundo Azuma é a de combinar no ambiente real os objetos virtuais com os objetos reais. Existem vários métodos, desde óticos a digitais, que permitem realizar essa combinação e visualizar o resultado aumentado, como a projeção, *optical* e *video see-through*.

Projeção

O método de projeção recorre, como o nome indica, à projeção do conteúdo virtual sobre o mundo real, aumentando-o desse modo. Estes sistemas são geralmente fixos, criando uma experiência que, por não requererem a colocação de qualquer dispositivo pelo utilizador, facilmente pode ser partilhada por múltiplas pessoas, sendo ideal para cenários como exposições em museus ou ambientes colaborativos para reuniões, podendo-se limitar a uma mera projeção de vídeo ou imagem interativa [68] ou então serem sistemas mais complexos que recorrem a lasers [79] ou ultrassons para mover partículas que ao serem iluminadas criam um imagem tridimensional com áudio posicional e feedback háptico [42] que se assemelham a um holograma (Figura 2.15(a)). No entanto, também existem equipamentos *wearable* como o SixthSense [57] e o OmniTouch [40] que são usualmente compostos por um projetor num capacete ou noutra parte do corpo que projeta a imagem aumentada sobre uma superfície à frente do utilizador ou até no próprio corpo deste (Figura 2.15(b)), sendo mais adequado para experiências mais pessoais e intimistas, embora também possam ser utilizados para experiências partilhadas. A principal desvantagem dos sistemas de realidade aumentada projetada é a sua suscetibilidade a interferências por parte de outras fontes luminosas mais intensas, como o Sol, o que torna inviável o seu uso no exterior, limitando-se assim a ambientes fechados e com iluminação mais controlada.



(a) Projeção 3D [42]



(b) OmniTouch [40]

Figura 2.15: Sistemas de realidade aumentada projetada.

Optical see-through

Os dispositivos de realidade aumentada *optical see-through* funcionam a partir da fusão ótica da imagem observável do mundo real com o mundo virtual num combinador ótico, sendo a solução utilizada em *smartglasses* como o Google Glass, Microsoft Hololens¹⁰ e Magic Leap¹¹. Para além dos *smartglasses*, existem também outros equipamentos como a HoloDesk [41] na qual são utilizados espelhos para combinar a imagem projetada e ao mesmo tempo permitir a visualização das mãos do utilizadores através do vidro (Figura 2.16). Os dispositivos *optical see-through* não são afetados do mesmo modo que os de projeção pois a combinação das imagens é feita no próprio dispositivo. Além disso, como permitem constantemente visualizar o mundo real de forma direta, é recomendada a sua utilização em contextos que necessitem dessa garantia de segurança em caso de falha como na medicina ou em aplicações militares. Contudo, sofrem no que respeita à oclusão de objetos físicos por objetos virtuais, pois estes últimos são semitransparentes, e na precisão do registo entre o virtual e o mundo real, havendo sempre algum atraso no *tracking* que pode causar algum desfasamento.

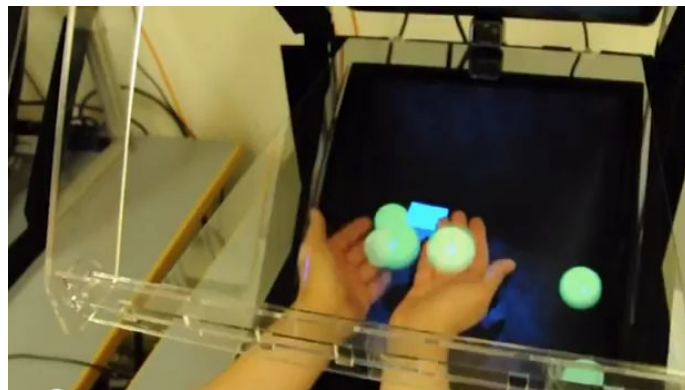


Figura 2.16: HoloDesk [41].

Video see-through

O método *video see-through* assemelha-se ao *optical see-through* no sentido em que a combinação do real e virtual é feita ao nível do dispositivo e que o utilizador observa o mundo real através do dispositivo como se fosse uma janela. Mas difere em que não há qualquer transparência, sendo o mundo real captado por uma câmara, combinado digitalmente com o mundo virtual e por fim mostrado num ecrã ao utilizador, passando a ser visualizado indiretamente.

Este é o método usualmente escolhido para as aplicações de realidade aumentada em *smartphones* (Figura 2.17) dada a existência no mesmo dispositivo de um ecrã, uma câmara de vídeo no lado oposto e, adicionalmente, outros sensores que facilitam o processo de *tracking*. Através deste método é também possível transformar um HMD de realidade

¹⁰<https://www.microsoft.com/hololens/>

¹¹<https://www.magicleap.com/>

virtual num de realidade aumentada ao adicionar uma câmara no exterior deste, o que mostra que as barreiras entre as duas áreas podem ser bastante ténues. Como a combinação é feita digitalmente, torna-se mais fácil a manipulação da imagem real para efetuar um registo mais preciso e também para uma oclusão correta e realista. No entanto, a inclusão da captura, processamento e visualização da imagem real introduz uma certa latência entre aquilo que o utilizador consegue ver no ecrã e o que ele pode ver a acontecer na realidade ao mesmo tempo podendo quebrar a ilusão que se pretende realizar se a latência for significativa. Além disso, todo este processamento adicional quando comparado com os outros métodos impõe requisitos maiores ao nível do *hardware*, o que limita o que se pode fazer em aplicações de realidade aumentada móveis onde já existem limitações dado o tamanho dos *smartphones*.



Figura 2.17: Aplicação de realidade aumentada para *smartphone* [6].

2.2.3 Técnicas de *tracking*

Para além do método para combinar o conteúdo virtual com o real, também é necessário que ocorra um registo adequado para que os objetos reais e virtuais correspondentes possam ser alinhados o mais corretamente possível, de modo a tornar a experiência aumentada mais imersiva. Para isso, existem diversas técnicas que recorrem aos vários tipos de sensores disponíveis (ou até a combinações destes) e que permitem determinar uma estimativa da pose do dispositivo para o alinhamento.

Visão Computacional

Com marcadores

Uma das utilizações mais antigas de visão computacional como técnica de *tracking* para aplicações de realidade aumentada é a deteção de marcadores de referência ou fiduciais colocados no ambiente físico, que servem de âncora ao mundo virtual e permitem efetuar um registo mais preciso no mundo real. Inicialmente, eram primariamente utilizados autocolantes coloridos monocromáticos que poderiam ser detetados a partir da análise das cores presentes na imagem, sendo esta ideia expandida de várias formas, como através da utilização de múltiplos círculos concêntricos de cores diferentes para permitir a deteção

dos marcadores a partir de distâncias diferentes [93]. Um dos exemplos mais populares é o ARToolKit [48], com marcadores formados por uma moldura preta quadrada para detetar o marcador e determinar a pose relativa do utilizador a este, e um padrão no interior que é correspondido numa base de dados e que serve como identificador para o conteúdo virtual a visualizar (Figura 2.18). Outros trabalhos posteriores melhoraram este sistema, como o ARTag [27] que diminui o número de falsos positivos, melhora a deteção do marcador quando está parcialmente tapado e também melhora a performance em condições desfavoráveis de iluminação. Como se tratam de referências artificiais cuja forma, aspeto e localização são previamente conhecidos, quando se pretende descobrir a pose do utilizador para efetuar o registo basta determinar a transformação geométrica a aplicar à posição do marcador de modo a que este corresponda ao seu aspeto na imagem captada pelo sistema de realidade aumentada.

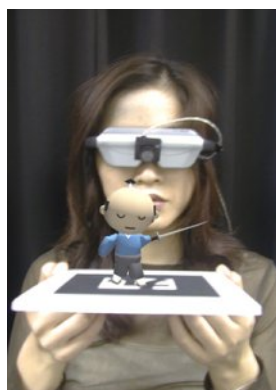


Figura 2.18: Realidade aumentada com marcadores do ARToolKit [47].

Sem marcadores

Os marcadores fiduciais utilizados pelo ARToolKit e ARTag permitiram superar várias limitações que existiam na utilização de marcadores coloridos simples, mas continuavam a necessitar que fossem efetuadas modificações no ambiente a aumentar o que nem sempre é desejável, como no caso de uma exposição de museu que teria de ser coberta de marcadores (cujo aspeto, de modo a facilitar a sua deteção pelo sistema, contrasta com o contexto em que está inserido, podendo ter um impacto negativo na experiência de um visitante do museu). Adicionalmente, a sua utilização no exterior seria bastante limitada, pois teriam de ser produzidos e colocados vários marcadores para cobrir uma vasta área, o que faria com que a solução não escalasse facilmente, podendo ter ainda outros problemas adicionais como a permissão para alterar os espaços com marcadores e a até a possível vandalização dos mesmos. No entanto, os progressos feitos ao nível do *hardware* tornaram viável a determinação da pose do utilizador, não com recurso a marcadores de referência físicos, mas através da deteção, em tempo real, de certas características naturais já presentes no ambiente. De uma forma geral este tipos de algoritmos, como o SIFT [53], SURF [10], BRIEF [18] e ORB [72], funcionam detetando características presentes na imagem capturada (como pontos, cantos, linhas retas, texturas contrastantes) e calculando descritores que

servem de identificador e descrição dessas características, podendo estes ser comparados com outros previamente calculados a partir de imagens conhecidas do objeto a encontrar (como no caso do quadro no museu) e após estabelecida uma correspondência (Figura 2.19) pode-se então determinar a pose do utilizador através de métodos semelhantes aos utilizados anteriormente para os marcadores.

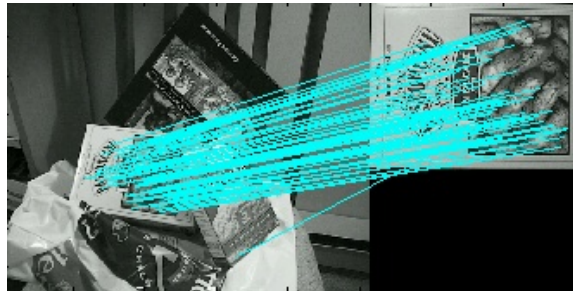


Figura 2.19: Correspondência de características na imagem alvo e na capturada [53].

Wireless

Uma técnica mais adequada para o uso no interior de edifícios é a de comunicação wireless, em que a posição do dispositivo é estimada a partir do *time of flight* de um sinal emitido por este até a um *access point*, que é utilizado para calcular a distância entre os dois equipamentos. Um exemplo concreto é o WiFi RTT, especificado na norma 802.11mc e suportado pelo Android desde a versão 9 (Android Pie) [92], que determina a posição do utilizador ao fazer a triangulação a partir do tempo de que um pacote demora a fazer uma round trip a vários *access points*, sem ser necessário que o dispositivo se conecte a cada AP desde que ambos os tipos de equipamentos suportem o protocolo. No entanto, esta técnica apenas consegue determinar a posição do utilizador, pelo que será necessária outra técnica que determine a orientação para que a pose possa ser obtida.

Magnetómetro

O tracking que recorre ao magnetómetro, como o nome do próprio sensor indica, funciona a partir de medições de propriedades dos campos magnéticos como a sua intensidade ou a sua direção. Podem-se utilizar medições do campo magnético terrestre para determinar a orientação do dispositivo e também é possível obter uma estimativa da sua localização através da colocação de âncoras magnéticas, isto é, dispositivos que emitem campos magnéticos distintos cujas propriedades são previamente conhecidas. Esta técnica tem a vantagem de que o sensor tem uma alta taxa de atualização, essencial para que a experiência de realidade aumentada possa ser imersiva ao executar em tempo real. Infelizmente, esta técnica é suscetível a interferências por parte de outros campos magnéticos e, além disso, a intensidade dos campos decresce rapidamente com o aumento da distância.

Inercial

Usualmente quando se refere ao tracking inercial, trata-se de uma **unidade de medição inercial** ou *inertial measurement unit (IMU)*, geralmente composta por acelerômetros e giroscópios, que permite calcular uma estimativa do movimento do dispositivo a partir de medições da sua aceleração e velocidade angular num processo denominado *dead reckoning*. Este processo é particularmente útil para complementar outra técnica que obtenha uma posição aproximada do utilizador, pois permite rapidamente atualizar a posição ao tomar essa medição mais exata como origem e ir adicionando a esse ponto a estimativa dos vários deslocamentos do utilizador, até que outra medição seja feita. A estimativa inercial é sujeita a discrepâncias com o decorrer do tempo e aumento da distância percorrida, pois os erros da estimativa da posição a partir da velocidade vão acumulando em cima do erro da medição inicial, daí ser mais adequada para complementar outro método durante o período de tempo entre as medições.

Sistemas de navegação por satélite

A técnica de *tracking* geralmente mais indicada para uso no exterior são os sistemas de navegação por satélite (como o GPS, GLONASS e Galileo, sendo que o primeiro é informalmente utilizado para descrever este tipo de sistemas) pois permitem determinar a posição de um utilizador com uma exatidão razoável em qualquer parte do mundo através da triangulação a partir de quatro ou mais satélites que transmitem o tempo do seu relógio e informação de posição. É possível que haja uma degradação ou até perda do sinal se na linha de visão entre o recetor e os satélites existirem obstáculos que afetem a qualidade do sinal, daí o GPS não ser adequado para o uso em espaços interiores como edifícios e túneis pois a estrutura estaria a obstruir os satélites. Para além das obstruções da linha de visão, o sinal também pode ter interferências causadas pela propagação multicaminho em que o próprio sinal é refletido por estruturas ou montanhas, havendo uma deterioração da qualidade, principalmente em centros urbanos densos. Como com qualquer outro sensor, a exatidão e precisão dependem também da qualidade deste, sendo que os recetores integrados em *smartphones* têm uma precisão menor que os equipamentos dedicados.

Fusão de sensores

Como já foi descrito anteriormente, existem variadas técnicas para determinar a posição e orientação de um dispositivo de realidade aumentada, todas com as suas vantagens e desvantagens. Para aproveitar o melhor de cada uma destas técnicas, surgiu então a fusão de sensores na qual estas técnicas são combinadas, podendo a combinação também ser feita de diversas maneiras. Um dos tipos de combinações passa por simplesmente trocar a técnica principal durante a execução consoante a sua adequação às necessidades (por exemplo, alternar entre a utilização de computação visual para obter uma posição com precisão alta e depois funcionar por *dead reckoning* para atualizar a posição mais

rapidamente mesmo que esta seja imprecisa de modo a consumir menos recursos). Outro tipo de combinação é a utilização de filtros que combinam os valores obtidos diretamente dos sensores.

2.2.4 Técnicas de interação

Por fim, falta mencionar a propriedade de um sistema de realidade aumentada ser interativo e executado em tempo real, mais concretamente alguns dos vários tipos de interfaces e métodos de input que têm sido desenvolvidos até hoje para variadas experiências aumentadas. Em [11] são separados nas seguintes categorias: navegadores de informação, interfaces 3D, interfaces tangíveis, interfaces naturais e interfaces multimodais.

Navegadores de informação

Nos navegadores de informação, a aplicação de realidade aumentada funciona como uma janela para um mundo de informação que os utilizadores podem manipular de modo a visualizar a informação que pretendem e requerem apenas duas tarefas de interação que são a visualização da cena aumentada e da informação fornecida. Como os conteúdos virtuais encontram-se registados no mundo real, o utilizador apenas precisa de se mover no seu espaço físico caso queira alterar o que está a ver. Um caso típico de navegadores de informação é a Navicam [70] que permite visualizar informação sobre objetos sobreposta na imagem da câmara ao detetar um marcador colorido que corresponda ao objeto, como a descrição de um quadro (Figura 2.20). No entanto, também é frequente adicionar elementos clássicos de interfaces gráficas 2D, sendo que estes elementos são interagidos através de inputs mais tradicionais em vez de métodos criados exclusivamente para o uso em realidade aumentada (por exemplo, clicar num botão para seleccionar a camada de informação a visualizar). Devido à sua relativa simplicidade e facilidade de aprendizagem e utilização, os navegadores de informação são bastante comuns em contextos de navegação, simulação e treino, embora estas interfaces sejam mais limitadas no que toca a interações diretas com objetos virtuais.



Figura 2.20: Navicam [70].

Interfaces 3D

Uma das soluções utilizadas para superar as dificuldades dos navegadores de informação em interagir diretamente com entidades virtuais é a de recorrer a métodos utilizados em realidade virtual e outras aplicações 3D para poder seleccionar e manipular entidades, sendo que a navegação não precisa de ser adaptada dada a sua simplicidade e intuitividade em realidade aumentada (basta o utilizador se mover). Muitas vezes são utilizados os mesmos tipos de dispositivos de input (como controladores tipo varinha, joysticks, spaceballs), como no caso do *SeamlessDesign* [50] em que o utilizador segura em dois comandos que determinam a posição das mãos e também possuem botões que permitem controlar e interagir com a aplicação. Para aumentar a imersão também é explorada a utilização de controladores com feedback háptico (em [4] é usado feedback háptico para que um professor possa guiar a mão do estudante e auxiliar o treino de uma cirurgia, em que ambos os utilizadores operam controladores diferentes). Como são utilizados os mesmos métodos da realidade virtual, podem ser aproveitados os estudos feitos nessa área de investigação e adaptar os resultados para realidade aumentada, embora estes métodos tenham limitações no que respeita à interação com objetos virtuais e físicos, pois o utilizador tem de largar o controlador para poder agarrar o objeto real.

Interfaces tangíveis

Para evitar o problema da desconexão entre o controlador e o conteúdo aumentado ao adaptar os métodos da realidade virtual surgiu a ideia de utilizar objetos como instrumento de input, as denominadas interfaces de utilizador tangíveis [46], podendo estes ser aumentados de modo a unir o objeto físico à entidade virtual, criando uma realidade aumentada tangível [47], em que o utilizador interage com os objetos virtuais ao manipular os objetos físicos.

Dependendo do tipo de ferramentas pretendido, estas podem ser classificadas segundo [11] como espacialmente ou temporalmente multiplexadas (ou um misto de ambas). Nas ferramentas espacialmente multiplexadas, cada objeto físico corresponde a uma única ferramenta, enquanto que nas temporalmente multiplexadas corresponde a múltiplas ferramentas, o que poupa em termos de objetos físicos necessários, mas tem a desvantagem de serem menos intuitivas.

Um exemplo de realidade aumentada tangível é a *Augmented Chemistry* [32], uma aplicação didática em que se podem combinar "átomos". O utilizador pode folhear um livro que contém marcadores que representam átomos. Quando encontra um átomo que pretende usar na combinação, pode pegar numa pá com um botão que, se for pressionado perto do átomo no livro, carrega a pá com o átomo e deposita-o na área de combinação (Figura 2.21(b)). Por fim, pode manipular um cubo que modifica como são combinados os átomos em moléculas. Todo este processo acaba por ser feito de um modo intuitivo devido às *affordances* dadas pelos objetos físicos que servem de input no sistema (Figura 2.21(a)). Infelizmente, esta fisicalidade que torna a experiência mais imersiva é também uma

limitação ao seu uso num contexto exterior ou móvel em que o utilizador precise de se afastar consideravelmente da área inicial, não sendo prático nem adequado o seu uso como método de interação nesses casos.

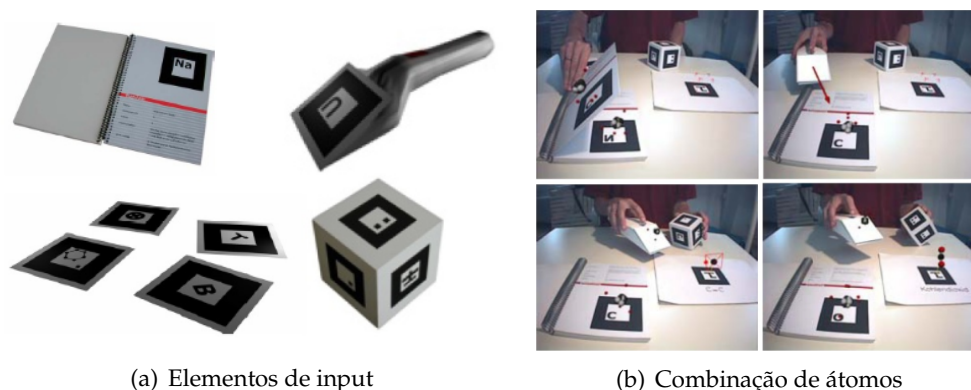


Figura 2.21: Augmented Chemistry [32].

Interfaces naturais

Um tipo de técnicas de interação ativamente investigadas são as interações naturais, em que são utilizados gestos com as mãos ou outras partes do corpo para interagir com o sistema, sendo particularmente úteis para sistemas de realidade aumentada móveis ou *wearable* (como os gestos manuais da HoloLens [43]). Desenvolvidos primariamente a pensar noutras áreas como a realidade virtual, os movimentos podem ser registados a partir de sensores que o utilizador tem equipado desde *datagloves* a fatos completos de captura de movimentos. Com os avanços e maior comercialização dos sensores de profundidade, estes também são alvo de investigação, como no caso do Microsoft HoloDesk [41] que recorria à câmara de profundidade Kinect para que os utilizadores pudessem interagir com o sistema. Outra área que também tem mostrado bastantes avanços é a da visão computacional, permitindo recorrer apenas a uma câmara para monitorizar os gestos feitos pelo utilizador e utilizá-los como input no sistema (o HandyAR [51] é um exemplo disto ao estimar a posição a partir da deteção das pontas dos dedos só com a câmara como sensor).

Interfaces multimodais

Semelhante ao que ocorre nas técnicas de *tracking* com a fusão de sensores, surge também nas técnicas de interação a ideia de interfaces multimodais, nas quais são combinados diferentes modos de input para o sistema, sendo bastante comum a combinação de input por comandos de voz para inserir informação quantitativa e a gestual para informação qualitativa, com [61] como exemplo concreto dessa combinação.

2.3 Exemplos de Casos de Uso

Existem diversos casos de uso para os quais uma integração de dados georreferenciados pode ser útil para criar experiências 3D que recorram a esses mesmos dados, sendo que nesta secção serão apresentados alguns exemplos relevantes.

O primeiro exemplo é a *Aqueduct AR* [64], em que é utilizada realidade aumentada para providenciar uma experiência mais rica de divulgação do património histórico e cultural associado ao Aqueduto das Águas Livres. Através da aplicação desenvolvida, os utilizadores que queiram saber mais sobre o aqueduto podem visualizar representações dos vários elementos deste património sobrepostos num mapa turístico de Lisboa, o seu traçado, descrições e fotografias sobre cada um dos seus pontos de interesse. Cada um destes elementos é sobreposto no mapa de papel de modo a corresponder com a sua posição real.

Já num contexto virado para a manutenção urbana, podemos encontrar o sistema *Vidente* [74] que permite visualizar através de realidade aumentada vários tipos de infraestrutura subterrânea, como condutas de água ou gás, cabos de eletricidade e telecomunicações, entre outras. Este sistema visa agilizar o trabalho de equipas no terreno que não têm acesso direto às ferramentas SIG que habitualmente contêm a localização destas infraestruturas, requerendo então a utilização de plantas em papel ou uma comunicação com uma equipa que possa dar indicações remotamente relativamente à sua localização. Assim, a equipa no local pode utilizar um dispositivo que, ao apontar na direção do chão, mostra uma imagem capturada pela sua câmara, à qual são sobrepostos os modelos 3D georreferenciados que correspondem à infraestrutura aí presente.

Outro caso de uso é o planeamento e simulação de projetos futuros, como o *Istanbul Solar Estimation* [16], no qual é utilizada uma representação virtual da edificação da cidade de Istanbul no motor de jogos Unity para poder simular a exposição solar de cada edifício, tendo em conta a sua localização, relevo do terreno e as sombras que outros edifícios ou estruturas podem cobrir, utilizando para isso o sistema de iluminação do Unity. O sistema apresentado em [67] também aproveita o sistema de áudio do Unity, neste caso para o planeamento da construção de turbinas eólicas, aplicando um modelo para simular o impacto em termos de ruído sonoro que uma dada localização das turbinas pode ter no ambiente em redor, de forma que este possa ser minimizado.

Mais relevante ao desafio inicialmente proposto, é o treino e gestão de cenários de crise. O projeto *EGCERSIS* [22] foi criado para permitir o treino de cenários de crise através de realidade virtual para locais em que a realização de simulacros de treino regularmente pode ser dispendiosa e limitada, como no caso de um incêndio numa plataforma de metro, que é um dos cenários estudados. Outro exemplo é o *GeospatialVR* [75], um sistema em que é possível observar um modelo de uma cidade em realidade virtual, sobre o qual são sobrepostos alertas sobre diversos cenários que podem acontecer na cidade, como incêndios, cheias ou acidentes viários, enquanto fornece a localização e estado de agentes envolvidos nestas operações (bombeiros, polícia, proteção civil, etc.), tratando-se de um

sistema de suporte à decisão. Em [36], é aplicado um modelo para simular a dispersão atmosférica de partículas radioativas em caso de acidente numa central nuclear, tendo sido implementado um protótipo de uma ferramenta de treino para equipas de emergência para que possam responder a este cenário.

Por fim, existem também os *Serious Games*, isto é, aplicações interativas que utilizam os mesmos princípios e mecânicas típicas de jogos, mas com um foco mais educativo e de treino, em vez de apenas entretenimento. Um exemplo é o Project Lily Pad [86], criado após o furacão Harvey ter fustigado Dickinson, Texas. Este projeto, criado em colaboração com sobreviventes desse desastre natural, pretende fornecer uma forma de educar e treinar os jogadores a lidar com cenários semelhantes, desde como ler um mapa, navegar um ambiente inundado, entre outras boas práticas de sobrevivência, tendo para isso desenvolvido um modelo da cidade de Dickinson a partir de dados reais para permitir uma experiência mais fidedigna e imersiva.

Resumindo, estes exemplos demonstram o potencial de integrar dados georreferenciados e tecnologias 3D numa variedade de áreas, desde o turismo e manutenção urbana, ao planeamento e gestão de cenários de crise. Estas aplicações têm assim a capacidade de melhorar a compreensão, eficácia e preparação em muitos aspetos da vida moderna.

2.4 Análise Comparativa de Soluções Existentes

Nesta secção, é feita uma exploração e comparação de diversas soluções disponíveis para integração de mapas e dados georreferenciados no motor de jogos Unity. Cada uma dessas soluções oferece um conjunto variado de funcionalidades e características, sendo esta comparação sumariada na tabela presente na subsecção 2.4.5.

2.4.1 ArcGIS Maps SDK for Unity

A Esri lançou em 2022 o ArcGIS Maps SDK for Unity¹² que permite aceder a mapas e conteúdos 3D do ArcGIS através de um *plugin* para o motor de jogos Unity. De todas as soluções que são comparadas nesta secção é a que apresenta o mais diverso leque de funcionalidades, em parte devido às suas décadas de experiência no desenvolvimento de *software* SIG.

À semelhança dos outros produtos ArcGIS, trata-se de uma solução proprietária e bastante integrada neste ecossistema. O mapa contém um *basemap*, as camadas de dados e elevação para representações 3D do terreno. Este mapa [54] tanto pode ser visualizado como um globo para contextos à escala mundial ou plano para contextos locais.

Para além de visualizar os diversos tipos de conjuntos de dados, desde camadas de imagens *raster* a conjuntos vetoriais e até modelos fotogramétricos, permite consultar os valores dos atributos que lhes estão associados e também analisar estes dados através da sua filtragem. Esta filtragem [29] pode ser feitas através dos seus atributos, modificando

¹²<https://developers.arcgis.com/unity/>

o aspeto das representações dos dados com base nos seus valores, ou espacialmente, definindo uma geometria em que apenas são visualizados os objetos dentro ou fora desta, consoante a opção.

Outras funcionalidades de destaque [49] são a possibilidade de efetuar medições geodésicas (seguindo a curvatura da Terra) de distâncias, a obtenção de rotas e também o modo *offline* que permite exportar e guardar um conjunto de *tiles* para uso local.

2.4.2 Mapbox Unity SDK

A Mapbox é uma empresa fornecedora de mapas para uso em vários tipos de aplicações, desde websites a aplicações móveis. Das suas diversas ferramentas a mais pertinente é o seu [SDK](#) para o motor de jogos Unity¹³, desenhado para criar jogos baseados no mundo real. O [SDK](#) é de código aberto, sendo distribuído com a licença MIT.

O mapa gerado encontra-se dividido em múltiplas *tiles* de modo a facilitar o carregamento dos dados que se localizem dentro da região de visualização. Estas *tiles* suportam a API de elevação da Mapbox para gerar representações 3D do terreno, camadas de imagens *raster* que servem de base para o mapa e também camadas vetoriais.

Quanto aos dados que as compõem, pode-se utilizar os conjuntos que a Mapbox fornece para todas as suas ferramentas como as imagens de cobertura aérea ou dados de estradas, edifícios e pontos de interesse provenientes de fontes como o OpenStreetMap ou dados próprios. A utilização de dados próprios ou de outras fontes, implica que primeiro estes sejam carregados na plataforma da Mapbox, sendo servidos a partir da sua API de *tiles*.

Como a Mapbox fornece APIs de localização e trânsito, o [SDK](#) também permite a obtenção de endereços ou coordenadas através de *geocoding* e *reverse geocoding*, respetivamente, assim como a obtenção de trajetos entre dois pontos com base no nível de trânsito.

Adicionalmente, o [SDK](#) também tem um componente que permite a criação de aplicações aumentadas para o interior e exterior. Nas aplicações para o interior, limita-se a usar o reconhecimento de superfícies fornecido pelas bibliotecas ARCore e ARKit para colocar o modelo nas superfícies horizontais detetadas. Quanto ao uso no exterior, combina as bibliotecas com a posição obtida por GPS para determinar a pose do dispositivo. No entanto, este componente não utiliza a *framework* AR Foundation para aceder a estas bibliotecas, mas sim a sua predecessora ARInterface [45] que foi descontinuada pelo Unity em 2018 quando anunciou a sua substituta [60].

Este problema com o componente de realidade aumentada é um dos vários que vão surgindo no [SDK](#) com o passar do tempo, visto que este não aparenta estar em desenvolvimento ativo. A sua última versão foi lançada em outubro de 2019 [21], o repositório não mostra *commits* relevantes feitas nos últimos anos e a página sobre o [SDK](#) já não se encontra mencionada na navegação da página web da Mapbox, tornando o [SDK](#) abandonado *de facto*.

¹³<https://www.mapbox.com/unity>

2.4.3 Cesium for Unity

A mais recente das soluções comparadas é o Cesium for Unity¹⁴ lançado pela Cesium em abril de 2023. É uma ferramenta de código aberto, sendo distribuída com a licença Apache.

Ao contrário das restantes, a representação do mapa no Unity é exclusivamente feita através de um globo terrestre. Este globo é gerado a partir de um conjunto de *tiles* 3D que formam uma estrutura otimizada para o efeito baseada numa *octree* [3], permitindo carregar uma representação completamente 3D da superfície terrestre. O *plugin* inclui uma câmara que ajusta a sua orientação consoante a sua posição no globo, de forma que o vetor que indica a direção para cima relativamente à sua posição local esteja sempre correto.

Esta técnica de representação implica, no entanto, que não é possível utilizar o motor de física incorporado no Unity para simular efeitos como a gravidade, pois este assume que a cena é sempre plana, sendo a gravidade aplicada como um vetor descendente em qualquer parte desta. Assim, um objeto no hemisfério sul seria então simulado como estando a "cair" em direção ao céu, em vez de ser corretamente aplicado um vetor que aponte para o centro do globo.

Para além do terreno, também permite importar conjuntos de dados *raster* que são aplicados como textura ao modelo do terreno e conjuntos que utilizem o formato de *tiles* 3D, como é o caso das *tiles* fotorrealistas que podem ser obtidas através da API da Google Maps Platform criada numa parceria entre as duas empresas [19], que disponibiliza os modelos de nuvem de pontos presentes no Google Earth e no modo 3D do Google Maps (curiosamente também a Google Maps teve um *SDK* para o Unity que foi deprecado em 2021 e encerrado em 2022 [35], embora este fosse limitado aos dados de edifícios que são utilizados para gerar a versão 2D).

Considerando que ainda é uma ferramenta relativamente recente, para além de visualizar estes dados, apenas se pode adicionar âncoras para georreferenciar objetos do Unity no globo, emulando o comportamento das cenas do Unity através de "subcenas" em que estes objetos apenas se encontram ativos caso a câmara dinâmica se encontra dentro da subcena.

2.4.4 Bing Maps SDK for Unity

O Bing Maps é o serviço de mapas da Microsoft, que tem um *SDK* para a sua utilização na criação de experiências de realidade mista no Unity¹⁵. Também é de código aberto, sendo distribuído com a licença MIT. Inclui um mapa que pode mostrar a elevação do terreno e, para um conjunto limitado de cidades, modelos fotogramétricos dos edifícios, estando limitado às camadas *raster* 2D do Maps aplicadas ao terreno para áreas sem cobertura.

¹⁴<https://cesium.com/platform/cesium-for-unity/>

¹⁵<https://github.com/microsoft/MapsSDK-Unity>

O **SDK** surge como um resultado de uma iniciativa interna, a Microsoft Garage, para desenvolvimento de experiências e protótipos pelos seus empregados que não se relaciona com as suas funções habituais. É primariamente desenhado para ser integrado como um controlo em aplicações de realidade mista para a HoloLens o que implica que, mesmo não estando restrito apenas para essa utilização, é a solução mais limitada das comparadas. Por este motivo, a área que permite visualizar é bastante restrita, não sendo possível cobrir todo o *viewport* com o mapa sem recorrer a uma aproximação excessiva da câmara.

2.4.5 Tabela sumária

A Tabela 2.1 contém um sumário das características presentes nas soluções existentes para que possam ser comparadas mais facilmente.

Tabela 2.1: Comparação das soluções existentes.

Característica	ArcGIS Maps	Mapbox	Cesium	Bing Maps
Código aberto	✗	✓	✓	✓
Desenvolvimento ativo	✓	✗	✓	✓
Terreno 3D	✓	✓	✓	✓
Mapa com área alargada	✓	✓	✓	✗
Visualização local	✓	✓	✗	✓
Visualização de globo	✓	✗	✓	✗
Dados <i>raster</i>	✓	✓	✓	✓
Dados vetoriais	✓	✓	✗	✗
Dados fotogramétricos	✓	✗	✓	✓
<i>Geocoding</i> e <i>Reverse geocoding</i>	✓	✓	✗	✓
Ver atributos dos dados	✓	✗	✗	✗
Filtragem de dados	✓	✗	✗	✗
Medições de distâncias	✓	✗	✗	✗
Obtenção de rota	✓	✓	✗	✗
Realidade aumentada	✗	⚠	✗	✓
Modo <i>offline</i>	✓	✗	✗	✗

2.5 Ferramentas

Nesta secção é feita uma breve descrição das ferramentas que foram utilizadas no desenvolvimento do protótipo ou que também foram úteis no processamento dos dados a utilizar.

2.5.1 QGIS

O QGIS¹⁶ é um *software* SIG gratuito e de código aberto (distribuído com a licença GNU GPL), criado em 2002 e disponível para sistemas Windows, macOS e Linux. Tratando-se de um SIG, permite visualizar, editar e analisar dados geoespaciais (Figura 2.22).

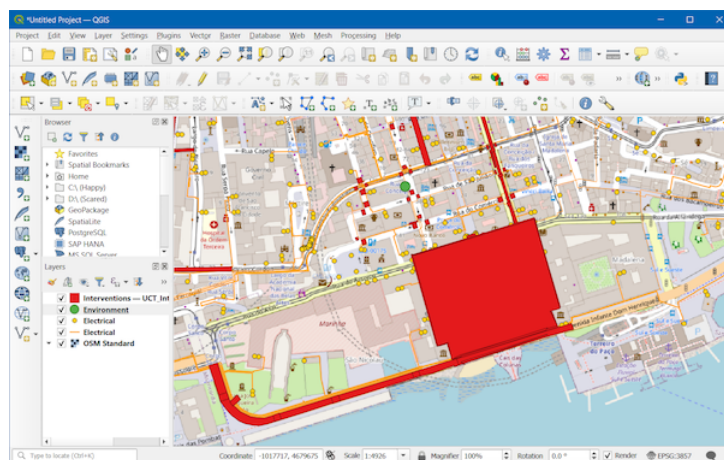


Figura 2.22: Visualização de conjuntos de dados vetoriais no QGIS.

Suporta um vasto conjunto de formatos de dados vetoriais, *raster* e também bases de dados e diferentes tipos de servidores de informação geográfica. Para além da interoperabilidade ao poder importar e exportar todos estes formatos, a possibilidade de editar os dados neles presentes ou até de criar camadas de dados de raiz, torna-o numa ferramenta poderosa e essencial para o processamento de conjuntos de dados.

As capacidades de visualização e análise dos dados geoespaciais também são úteis ao permitir compreender a informação que nos tinha sido disponibilizada de uma maneira mais completa, servindo também como fonte de inspiração para alguns elementos a implementar no protótipo.

Adicionalmente, a sua interface *user-friendly* e o forte ecossistema de *plugins* disponíveis foram uma mais valia na sua utilização.

2.5.2 ArcGIS CityEngine

O ArcGIS CityEngine¹⁷, como o seu nome indica, pertence à *software suite* ArcGIS desenvolvida pela Esri.

¹⁶<https://qgis.org/en/site/>

¹⁷<https://www.esri.com/en-us/arcgis/products/arcgis-cityengine/overview>

Trata-se de um programa comercial de modelação 3D especializado em ambientes urbanos, com foco na criação de representações em larga escala de cidades. A criação destas representações, ao contrário da modelação com recurso a técnicas típicas de *Desenho Assistido por Computador* ou *Computer-Aided Design (CAD)*, é feita com base num sistema de regras definidas pelos utilizadores que servem para gerar as formas desejadas.

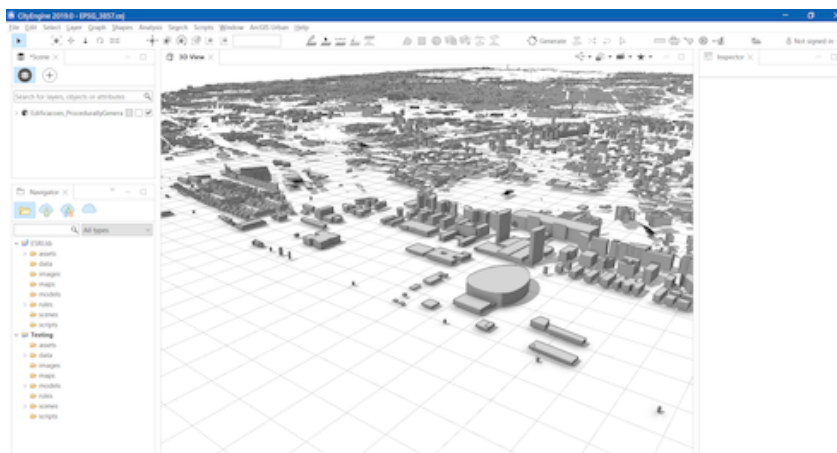


Figura 2.23: Dados dos edifícios no ArcGIS CityEngine.

Em parte, por se tratar de um produto comercial a sua interoperabilidade é mais reduzida, no entanto, é possível exportar os dados dos modelos 3D gerados para formatos como o FBX, DAE (COLLADA) ou DWG. Também é possível exportar as formas que servem de base para a sua geração através do formato Shapefile. Como os dados de edifícios fornecidos (Figura 2.23) foram gerados com esta ferramenta no formato Esri File GeoDatabase, o seu uso foi necessário para extrair os modelos destes.

2.5.3 geojson-tile-server

O geojson-tile-server é um programa de código aberto (distribuído com a licença MIT) que, como o seu nome indica, permite servir conjuntos de dados GeoJSON na forma de *tiles* XYZ. Devido a uns erros na sua interpretação de conjuntos GeoJSON (como retornar um objeto JSON vazio em vez de um objeto GeoJSON com uma *FeatureCollection* vazia para *tiles* que não tenham elementos) que foram detetados na sua utilização, foi feita uma *fork* do geojson-tile-server para essas alterações (disponível em <https://github.com/MagicalBlob/geojson-tile-server>).

Trata-se de um programa desenvolvido em Node.js que carrega os ficheiros *.geojson* que contêm a totalidade dos conjuntos de dados relativos a cada camada vetorial. Internamente, utiliza a biblioteca *geojson-vt*¹⁸ para recortar cada um dos conjuntos segundo o padrão de *tiles* XYZ. Quando um cliente faz então o pedido de uma *tile* à semelhança de um servidor de *tiles raster*, o servidor recorta a área correspondente, determinando quais os objetos que a intersectam e pertencem-lhe, gerando um novo objeto GeoJSON

¹⁸<https://github.com/mapbox/geojson-vt>

que contém apenas esses objetos dentro de uma `FeatureCollection` e que é retornado ao cliente.

Quanto aos objetos que se situem numa fronteira entre *tiles*, são divididos de forma que cada *tile* apenas contenha a parte do objeto que lhe pertence. No entanto, existe uma área de *buffer* na fronteira em que parte do objeto que fica localizado já na *tile* vizinha é incluído na mesma para evitar problemas com pequenos espaços vazios resultantes do corte dos objetos.

2.5.4 Unity

O Unity¹⁹ é um motor de jogos popular e poderoso, criado em 2005 pela Unity Technologies, que permite desenvolver aplicações para múltiplas plataformas, embora não tenham necessariamente de se tratar de jogos, sendo também utilizado por outras indústrias que pretendam criar aplicações 3D e 2D como no caso da arquitetura e engenharias [1].

Este motor de jogos suporta plataformas como Windows, macOS, Linux, Android, iOS, PlayStation, Xbox, assim como dispositivos de Realidade Virtual, Mista e Aumentada, como a Oculus, o Valve Index, a HoloLens ou a Magic Leap. Para estes últimos em particular, existe a *framework* AR Foundation que serve de interface entre o motor de jogo e as APIs nativas de cada dispositivo para realidade aumentada. A capacidade de poder compilar o mesmo projeto com apenas alguns ajustes para plataformas tão diversas é uma vantagem para um desenvolvimento e prototipagem mais eficiente.

Para o desenvolvimento das aplicações, os programadores têm ao seu dispor um conjunto robusto de ferramentas. Inclui um editor visual que simplifica o processo de criar e editar os *assets* e cenas a serem utilizados, suportando diversos tipos de modelos 3D, texturas, áudio entre outros tipos. A linguagem de desenvolvimento utilizada é o C#, o que o torna acessível a um largo conjunto de programadores, havendo também uma extensão documentação quer das APIs do próprio motor de jogos como do ecossistema .NET em que se insere o C#. Além disso, dada a sua popularidade existem bastantes recursos na comunidade que servem de ajuda desde a iniciantes a profissionais.

Entre estas e outras funcionalidades típicas de um motor de jogos modernos (como as suas capacidades gráficas, *physics engine* para simulações físicas e deteções de colisões ou o sistema de animações), relativa facilidade de uso e licença gratuita para pequenos projetos, tornou-se num dos motores de jogos mais utilizado globalmente.

2.5.5 Nominatim

O Nominatim²⁰ é um serviço gratuito e de código aberto baseado nos dados presentes no OpenStreetMap, permitindo a sua pesquisa. A sua API permite fazer o *geocoding* de um dado local, isto é converter a descrição de um local (como o seu nome ou endereço) num par de coordenadas latitude/longitude. Igualmente, permite fazer o inverso desta

¹⁹<https://unity.com/>

²⁰<https://nominatim.org/>

operação, o *reverse geocoding*, em que tenta obter o objeto que corresponda a um dado par de coordenadas e retorna a sua descrição (podendo servir para obter o seu endereço ou tipologia).

2.5.6 Open Source Routing Machine

À semelhança do Nominatim, a Open Source Routing Machine (OSRM)²¹ é um serviço gratuito e de código aberto, que também tem por base os dados contribuídos pelos utilizadores do OpenStreetMap.

Trata-se de um motor de pesquisa de rotas que tenta encontrar o caminho mais rápido entre os pontos dados, utilizando a rede presente no OpenStreetMap, podendo suportar carros, bicicletas e andar a pé como modo de deslocação.

2.5.7 ARCore

O ARCore²² é um *SDK*, lançado em 2018 pela Google, que permite o desenvolvimento de aplicações de realidade aumentada para dispositivos Android. Utiliza um método de *simultaneous localization and mapping* (SLAM), chamado *Concurrent Odometry and Mapping* (COM), em que através das imagens capturadas pela câmara do smartphone ou tablet vai criando uma nuvem de pontos a partir de descritores de características comuns detetadas nestas imagens [33]. Como é utilizada exclusivamente a câmara e não há qualquer auxílio de outros sensores neste mapeamento (ao contrário do Project Tango, o predecessor do ARCore, que necessitava também de um sensor de infravermelhos para deteção e medição de distâncias [83]), o ARCore tem dificuldade em detetar superfícies lisas e homogéneas (como uma parede branca) e também pode perder o *tracking* em condições de baixa ou alta intensidade de luz. A nuvem de pontos gerada é então utilizada para corrigir discrepâncias dos valores obtidos pela IMU e para comparar os pontos atualmente visíveis com os já guardados, de modo a localizar o dispositivo nesse mapa local. Assim, combina a alta taxa de atualização dos valores da IMU, necessário para a condição de tempo real da realidade aumentada, com a localização mais exata obtida a partir da visão computacional, necessária para um melhor registo do conteúdo virtual no mundo real.

Ainda relativo à utilização da visão computacional, o ARCore permite também detetar imagens 2D previamente carregadas numa base de dados e utilizá-las como âncora (como por exemplo um cartaz de publicidade ou um sinal de trânsito), suportando deste modo o *tracking* com recurso a marcadores, sendo que estes podem ter um aspeto mais apelativo em comparação com os tradicionais marcadores fiduciais.

Quanto aos pontos da nuvem, este são também utilizados para melhor compreender o mundo que o dispositivo vê, sendo agrupados aqueles que aparentam estar nas mesmas superfícies horizontais e verticais (como mesas e paredes), transformando estas superfícies

²¹<https://project-osrm.org/>

²²<https://developers.google.com/ar/>

em planos virtuais (Figura 2.24) que podem ser utilizados pela aplicação de realidade aumentada para colocar objetos em superfícies planas.



Figura 2.24: Planos horizontais e verticais do ARCore.

2.6 Resumo

Neste capítulo começou-se por introduzir o que são os **SIG**, conceitos base como os **CRS**, assim como tipos e formatos de dados georreferenciados. De seguida, definiu-se o que é a realidade aumentada, tendo sido também apresentada uma perspetiva histórica desta, os diferentes métodos de visualização, técnicas de *tracking* e de interação. Foram depois indicados alguns exemplos de casos de uso para os quais uma integração de dados georreferenciados pode ser útil para criar experiências 3D a partir destes. Após isto, foi feita uma comparação entre algumas das soluções existentes, em termo das suas capacidades e funcionalidades. Finalmente, foi feita uma breve descrição das várias ferramentas que foram utilizadas no desenvolvimento do protótipo ou no processamento dos dados a utilizar.

ABORDAGEM

Findo o levantamento do estado da arte, pode-se agora delinear os requisitos a serem cumpridos pela aplicação, seguidos pela abordagem proposta. De seguida, são brevemente descritos os conjuntos de dados utilizados. Após isto, é apresentada mais concretamente a arquitetura da aplicação.

3.1 Requisitos da aplicação

Para criar a solução proposta, foi necessário estabelecer alguns requisitos com base nos objetivos previamente mencionados nesta dissertação. O conjunto dos requisitos estabelecidos foi o seguinte:

- A aplicação deve suportar dispositivos *Desktop* e *Mobile*;
- A aplicação deve ser fácil e intuitiva de utilizar;
- A aplicação deve mostrar um mapa de Lisboa na forma de um modelo tridimensional;
- A aplicação deve ser expansível, de modo a poder ser utilizada com outros conjuntos de dados, para outras cidades ou até para outros contextos geográficos.
- A aplicação deve permitir navegar no mapa arrastando-o (*panning*);
- A aplicação deve permitir alterar o nível de ampliação do mapa;
- A aplicação deve permitir alterar a inclinação e orientação do mapa;
- A aplicação deve permitir alterar a localização atual ao indicar um par de coordenadas latitude/longitude;
- A aplicação deve permitir alterar a localização atual para um determinado ponto de interesse;
- A aplicação deve permitir alterar a localização atual através da pesquisa de um dado local;

- A aplicação deve permitir a representação de diversos conjuntos de dados georreferenciados, sejam *raster* ou vetoriais;
- A aplicação deve permitir selecionar quais os conjuntos de dados a ser visualizados;
- A aplicação deve mostrar informação sobre os conjuntos de dados (data de atualização, descrição, fonte dos dados);
- A aplicação deve permitir obter as coordenadas e a morada de um dado ponto do mapa;
- A aplicação deve permitir a visualização de atributos associados a objetos presentes nos conjuntos de dados;
- A aplicação deve permitir a filtragem dos dados a ser visualizados a partir dos seus atributos;
- A aplicação deve permitir a visualização da elevação do terreno através de uma representação tridimensional deste;
- A aplicação deve permitir a medição de distâncias entre pontos definidos pelo utilizador;
- A aplicação deve mostrar a rota entre dois pontos pedidos pelo utilizador;
- A aplicação deve permitir a utilização de realidade aumentada para visualizar o mapa.

3.2 Abordagem proposta

A solução elaborada trata-se de uma aplicação multiplataforma que permite visualizar e analisar diversos tipos de dados georreferenciados. Esta aplicação foi desenvolvida com recurso ao motor de jogo Unity, de modo a aproveitar o seu vasto suporte de compilação para diferentes dispositivos e sistemas operativos (por exemplo, computadores Windows e *smartphones* Android), o suporte de várias APIs nativas de Realidade Aumentada (através da AR Foundation) e também todas as funcionalidades típicas de motores de jogos modernos que facilitaram a implementação do protótipo.

De um modo geral, a aplicação é um mapa de dados georreferenciados, sendo estes separados em múltiplas camadas, cada uma correspondente a um conjunto de dados. No protótipo desenvolvido existem dois tipos principais de camadas, que são as camadas Raster e as camadas Vetoriais, sendo ambas divididas em várias *tiles* por uma questão de eficiência e para evitar carregar a totalidade dos conjuntos de dados. Relativamente às camadas vetoriais em particular, optou-se por utilizar o formato GeoJSON, dada a capacidade deste formato para representar objetos geográficos comuns em vários formatos,

o que torna-o num bom candidato alvo para converter outros formatos, e por ser baseado no formato JSON, o que permite a utilização de bibliotecas populares de *parsing* deste.

Com este modelo virtual, é então possível realizar vários tipos de interações, desde a navegação básica e seleção de camadas a visualizar, à filtragem dos objetos baseada nas suas propriedades ou medições de distâncias, sendo também possível visualizar o modelo através de realidade aumentada. Deste modo, para além de cumprir a primeira etapa do problema como definida na secção 1.2 ao fazer a integração dos dados georreferenciados, também pode ser feita uma exploração preliminar das etapas de criação de visualizações de dados e interação com o modelo virtual em Realidade Aumentada.

3.3 Conjuntos de dados

É agora feita uma breve descrição dos conjuntos de dados utilizados na aplicação. Na Tabela 3.1, apresentam-se os conjuntos de dados utilizados, indicando as fontes, as áreas de cobertura, as datas das últimas atualizações e os seus formatos originais. Em seguida, na Tabela 3.2, são apresentados os identificadores dos conjuntos de dados, seguidos por uma breve descrição dos mesmos.

Tabela 3.1: Conjuntos de dados utilizados

ID	Fonte	Cobertura	Atualização	Formato
OSMStandard	OSM	Mundial	<i>Desconhecido</i>	PNG
MapboxSatellite	Mapbox	Mundial	<i>Desconhecido</i>	JPEG
StamenWatercolor	Stamen	Mundial	<i>Desconhecido</i>	PNG
StamenToner	Stamen	Mundial	<i>Desconhecido</i>	PNG
StamenTerrain	Stamen	Mundial	<i>Desconhecido</i>	PNG
Bikepaths	CML	Lisboa	2022-11-18	GeoJSON
Buildings	CML	Lisboa	2020-07-28	Geodatabase
BuildingsLOD3	CML	Alta, Baixa	2020-07-28	Geodatabase
Electrical	E-Redes	Lisboa	2020-07-07	Shapefile
Environment	CML	Lisboa	2022-11-27	JSON
Interventions	CML	Lisboa	2022-11-10	GeoJSON
Lamps	CML	Lisboa	2022-10-28	GeoJSON
PedNet	U-Shift - IST	Alvalade, Arroios, Ave- nidas Novas, Belém, Ben- fica, Parque das Nações	2019-10-03	Shapefile
Rails	CML	Lisboa	2022-11-02	GeoJSON
Roads	CML	Lisboa	2020-07-28	CityGML

RoadsOld	CML	Lisboa	2012-10-18	GeoJSON
Sidewalks	CML	Lisboa	2020-07-28	CityGML
Signs	EMEL	Lisboa	2020-09-10	Shapefile
TrafficRestrictions	CML	Lisboa	2022-11-26	GeoJSON
Trees	CML	Lisboa	2022-11-02	GeoJSON

Tabela 3.2: Descrições dos conjuntos de dados

ID	Descrição
OSMStandard	Estilo padrão do OpenStreetMap
MapboxSatellite	Imagens aéreas e de satélite
StamenWatercolor	Um design em estilo de aguarela, remanescente de mapas desenhados à mão
StamenToner	Mapa de alto contraste para mashups de dados e exploração de meandros de rios e zonas costeiras
StamenTerrain	Mapas de terreno com hill shading e cor natural da vegetação
Bikepaths	Rede de ciclovias de Lisboa
Buildings	Edifícios de Lisboa com baixo detalhe (extrusão de planta)
BuildingsLOD3	Edifícios de Lisboa com alto detalhe geométrico
Electrical	Cabos da rede de distribuição elétrica, subestações, entre outras infraestruturas
Environment	Dados de sensores ambientais (Qualidade de Ar, Ruído, Meteorologia e Tráfego)
Interventions	Vários tipos de intervenções realizadas no espaço público
Lamps	Equipamentos de iluminação pública
PedNet	Rede parcial de passeios para bairros distintos de Lisboa
Rails	Rede ferroviária
Roads	Rede viária
RoadsOld	Rede viária
Sidewalks	Passeios
Signs	Sinalização vertical gerida pela EMEL
TrafficRestrictions	Restrições de trânsito
Trees	Diversos tipos de arvoredo existente em Lisboa

Alguns destes conjuntos de dados, como os edifícios, rede viária e passeios, foram fornecidos diretamente pela Câmara Municipal de Lisboa. Quanto aos restantes, foram obtidos a partir de repositórios de dados abertos ou APIs dedicadas.

Conforme se pode verificar na Tabela 3.1, os conjuntos de dados encontram-se em diversos formatos, sendo que podemos agrupá-los em duas categorias: dados *raster* e *vetoriais*.

Visto que os conjuntos de dados *raster* encontram-se todos no formato PNG ou JPEG, não é necessário realizar qualquer conversão destes, pois estes formatos já são suportados pelo Unity para serem utilizados como texturas.

No entanto, o mesmo não se aplica aos dados vetoriais, originalmente também em diversos formatos (GeoJSON, Shapefile, CityGML, JSON e ArcGIS Geodatabase), mas que não são suportados nativamente pelo Unity. Para evitar ter de suportar estes formatos individualmente foi feita a opção de focar no formato aberto GeoJSON, desenhado para representação de informação geográfica simples que, como o nome indica, trata-se de um formato baseado no formato JSON, o que por sua vez permite a utilização de bibliotecas de *parsing* de JSON. Com este foco escolhido, foi então feita a conversão para GeoJSON dos conjuntos de dados nos restantes formatos, sendo estes processos descritos com mais detalhe na secção 5.3.

3.4 Arquitetura da aplicação

A arquitetura da aplicação, representada na Figura 3.1, é baseada no modelo de três camadas [91], sendo estas as de Apresentação, da Lógica e dos Serviços de Dados.

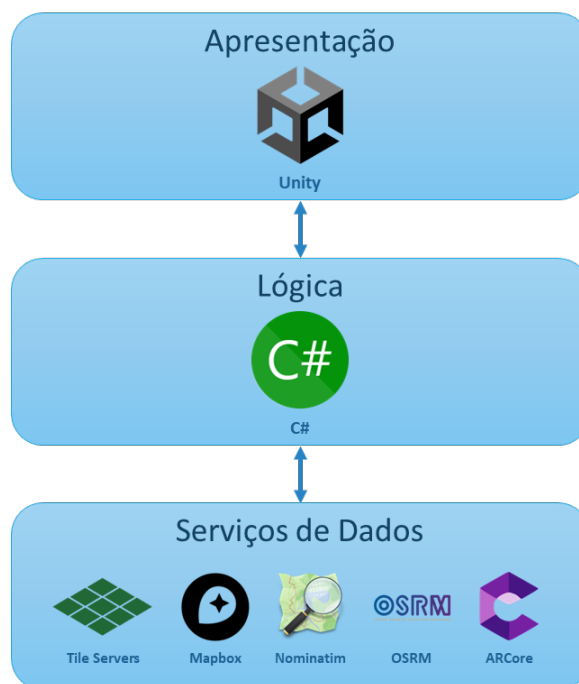


Figura 3.1: Arquitetura da solução.

3.4.1 Camada de Apresentação

A camada de Apresentação é responsável pela interface do utilizador, tal como a sua interação com esta. Para isto foi escolhido o motor de jogos Unity pelas suas ferramentas de renderização de gráficos 2D e 3D, que permitem criar interfaces dinâmicas, capturar

variados tipos de inputs como gestos e toques para plataformas móveis ou rato e teclado para computadores, entre outras funcionalidades (que também têm impacto nas outras camadas como é o caso da AR Foundation).

3.4.2 Camada da Lógica

A camada da Lógica desempenha o papel de ligação entre as camadas de Apresentação e dos Serviços de Dados, sendo que é nesta que é processada e integrada a informação obtida nas outras duas camadas. Deste modo, é a parte central da solução, onde estão implementadas as funcionalidades que fazem a aplicação funcionar. A lógica foi feita em C#, tendo a escolha sido condicionada em parte pela utilização do motor de jogos Unity e também por ter experiência prévia na utilização da linguagem. O C# é uma linguagem robusta e bastante utilizada, com um conjunto rico de funcionalidades que permite construir sistemas de software complexos.

3.4.3 Camada dos Serviços de Dados

A camada dos Serviços de Dados é onde constam os serviços e fontes de dados que fornecem a informação necessária para a aplicação. Os serviços utilizados são:

- **Servidores de *tiles*** - são estes que fornecem os conjuntos de dados que podem ser visualizados e analisados na aplicação já divididos em *tiles* XYZ, de forma a evitar o carregamento na sua totalidade e aumentar a eficiência da aplicação. Para os conjuntos GeoJSON é utilizado o `geojson-tile-server` que os recorta no mesmo formato de *tiles* XYZ usada para conjuntos *raster*.
- **Mapbox Terrain** - fornece o modelo digital do terreno que é utilizado para gerar a visualização tridimensional dos conjuntos de dados.
- **Nominatim**¹ - que fornece as capacidades de *geocoding* e *reverse geocoding*, isto é a conversão de endereços em coordenadas geográficas e vice-versa, para enriquecer os conjuntos de dados e também ajudar na navegação, exploração e análise do mapa. Utiliza o OpenStreetMap como fonte de dados.
- **OSRM**² - para obter os trajetos e informação do percurso que são essenciais para a funcionalidade da Rota, sendo que também utiliza o OpenStreetMap como a sua fonte de dados.
- **ARCore** - que permite criar experiências de realidade aumentada em dispositivos Android suportados. Este é utilizado através da *framework* de Realidade Aumentada multiplataforma do Unity, a AR Foundation, que trata da interação com as ferramentas nativas e que também serviu como motivo adicional para a opção de utilizar o Unity para desenvolver o protótipo.

¹<https://nominatim.org/>

²<https://project-osrm.org/>

3.5 Resumo

Neste capítulo, foram apresentados os requisitos da aplicação de visualização e análise de dados georreferenciados, delineando as funcionalidades que esta deve oferecer aos utilizadores, assim como a abordagem proposta para a sua criação. Além disso, uma visão geral dos conjuntos de dados utilizados na aplicação foi fornecida, abrangendo diversas fontes e tipos de dados georreferenciados, que serão a base para as análises e visualizações na aplicação. A arquitetura da aplicação foi então descrita, delineando a organização em três camadas: a camada de Apresentação, responsável pela interação com o utilizador; a camada da Lógica, onde as funcionalidades essenciais da aplicação são implementadas; e a camada dos Serviços de Dados, que consiste nas fontes de dados externas e serviços necessários para enriquecer a aplicação com informações geográficas precisas. O próximo capítulo irá apresentar a aplicação desenvolvida, a sua interface e detalhar as suas funcionalidades.

APLICAÇÃO

Neste capítulo é descrito o protótipo desenvolvido, começando pelo mapa de conteúdos georreferenciados que constitui a parte central da aplicação e, de seguida, a interface que permite a interação com este pelo utilizador.

A aplicação foi desenhada para dispositivos *Desktop* e *Mobile*, tendo para isso sido escolhida a utilização do motor de jogos Unity como base para o desenvolvimento do protótipo pelo seu suporte multiplataforma. Para além das funcionalidades disponíveis ao optar por um motor de jogos de propósito geral como o Unity e que facilitam em grande parte o desenvolvimento de aplicações gráficas 3D, a capacidade de compilar o mesmo código base para plataformas diferentes ao invés de soluções nativas para cada plataforma simplificou a escolha.

4.1 Mapa

A parte central da aplicação é o mapa pois trata-se do componente que pega nos dados georreferenciados e cria uma representação digital do mundo. Este contém várias camadas, as *Layers*, que correspondem cada uma a um conjunto de dados, sejam estes dados *raster* ou vetoriais no formato GeoJSON, existindo a possibilidade de estender para suportar outros formatos. É nestas que se encontram alguns metadados como o nome, descrição, fonte, última atualização, URL do servidor de *tiles* e também o *Renderer*, tratando-se este último da classe responsável por transformar os dados obtidos do servidor na sua representação gráfica no Unity.

Para tornar o carregamento dos dados mais eficiente, o mapa encontra-se dividido em múltiplas secções, as *Tiles*, não sendo assim necessário ter os conjuntos de dados carregados na sua totalidade, bastando apenas aqueles que compõem a área a visualizar. Além disso, permite também que todo o processo desde o pedido que obtém os dados à geração do modelo seja paralelizado, visto que a maior parte do trabalho a efetuar em cada *tile* é independente das suas vizinhas. Caso o utilizador queira visualizar uma área perto da atual, esta técnica permite manter carregadas as *Tiles* que ainda se encontrem dentro da nova área e carregar apenas as necessárias para preencher o resto da janela,

sendo eventualmente removidas todas as previamente carregadas que ficarem de fora.

As Tiles, por sua vez, estão divididas pelas diferentes camadas, formando as TileLayers, que são os objetos efetivamente responsáveis pela representação dos dados de cada camada que pertençam à Tile. Todas as TileLayers utilizam o mesmo padrão XYZ tipicamente utilizado para *tiles raster*, de modo a evitar problemas em que conjuntos de dados diferentes estivessem divididos com tamanhos diferentes, o que dificultaria o processo de gestão do carregamento das Tiles.

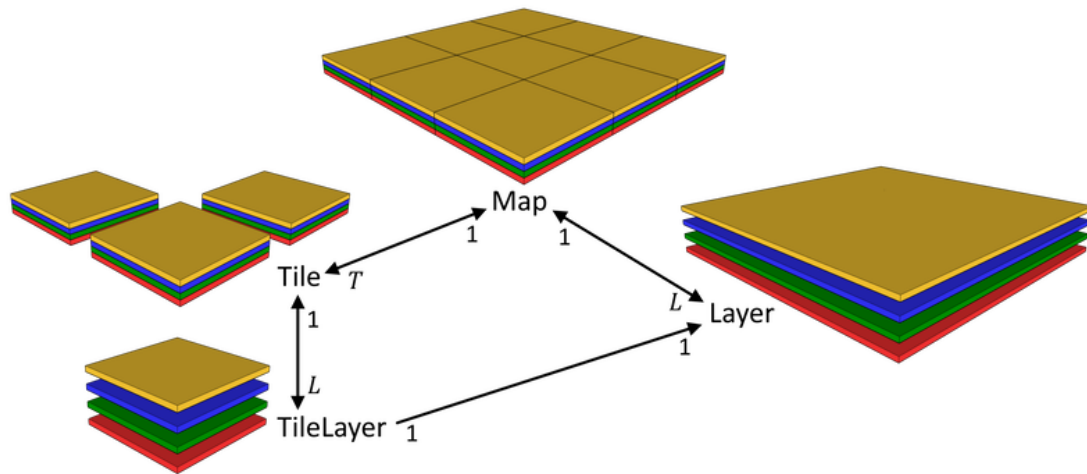


Figura 4.1: Diagrama dos componentes que constituem o mapa e as relações entre estes.

Na Figura 4.1 encontra-se um diagrama que mostra as relações entre estes diferentes objetos que constituem o mapa. Reiterando, o mapa contém L Layers e está dividido em T Tiles, que por sua vez estão divididas em L TileLayers, sendo que estas correspondem cada uma a uma única Layer. As TileLayers são responsáveis apenas pela gestão dos dados de uma camada que pertencem à área de uma Tile, enquanto que as Layers são responsáveis por gerir o que for transversal à camada inteira (como a configuração dos filtros que são aplicados a todas as *tiles*).

4.1.1 Modelo digital do terreno

Para os casos em que o utilizador queira saber a altura numa dada localização ou se tiver ativado a sua representação, foi incorporado um modelo digital do terreno através da API de elevação da Mapbox, cujos dados de elevação para o território europeu foram produzidos com recurso a dados do programa Copernicus da União Europeia [2] com resolução espacial de 25 metros e exatidão vertical $\pm 7m$ [25].

Como a API funciona com recurso a *tiles raster* que seguem a convenção XYZ, o mesmo padrão utilizado na aplicação para dividir os dados das várias camadas, e como as outras camadas podem necessitar de aceder aos dados de elevação, estes são os primeiros a serem pedidos cada vez que uma Tile do mapa é necessária. Só após a conclusão desse pedido é que são carregadas as várias TileLayers que compõem as diversas camadas.

4.1.2 Camadas Raster

Visto que as camadas *raster* são, como o nome indica, compostas por imagens *raster* que são fornecidas pelos servidores de *tiles*, não existe muito a fazer por parte da aplicação ao nível das *TileLayers* para integrar este tipo de dados, já que todo o processo relativo à divisão e gestão do carregamento das *tiles* é feito logo na classe *Tile*. Como as imagens já estão geradas, não sendo necessário um processamento adicional dos dados, e como encontram-se num formato suportado nativamente pelo Unity, o *Renderer* pré-definido apenas trata de as aplicar como a textura de um quadrado ou de um objeto de terreno (criado com recurso aos dados de elevação), consoante a opção do utilizador em ativar a visualização da elevação do terreno. No entanto, caso haja a necessidade de realizar algum processamento adicional na imagem, existe a possibilidade de criar um novo *renderer* que, desde que implemente a interface *IRasterRenderer*, pode aceder a imagem obtida do servidor e gerar quaisquer objetos que permitam a visualização desejada.

4.1.3 Camadas Vetoriais

Tal como já foi mencionado anteriormente, as camadas vetoriais são formadas a partir de conjuntos de dados que se encontram no formato GeoJSON, embora também neste caso seja possível adicionar suporte para outros formatos vetoriais (como as Mapbox Vector Tiles), desde que seja implementada a mesma interface *ILayer*.

Como o formato GeoJSON não é suportado nativamente pelo Unity e é baseado em JSON, foram implementados conversores para o *Json.NET*¹, uma biblioteca de *parsing* de JSON, para permitir a sua deserialização em classes que podem ser utilizadas pelo *Renderer*. Existem vários tipos de objetos geométricos que podem ser representados em GeoJSON, sendo que estes são depois reduzidos para primitivas *Node*, *Edge*, *Area* para que o *Renderer* possa ser mais agnóstico quanto ao formato.

Para além da sua geometria, os objetos GeoJSON também podem conter propriedades que fornecem informação adicional sobre si mesmos, podendo estas ser descritas por um qualquer objeto JSON. De modo a permitir operações sobre estas propriedades, as *GeoJsonLayer* implementam a interface *IFilterableLayer*, que por sua vez estende *ILayer*. A este tipo de camada filtrável são associadas propriedades dos objetos que, como o nome indica, podem ser utilizadas na aplicação de filtros para apenas serem visualizados os dados que correspondam.

No entanto, apenas são aceites pela aplicação como propriedades de objetos os membros que sejam Booleanos, Datas, Números, Strings ou Categorias (Strings que correspondem a um conjunto pré-definido de opções). Esta limitação foi o resultado de uma escolha para evitar a criação de interfaces complexas para a filtragem dos objetos, embora seja possível adicionar outros tipos se for implementada a interface *IFeatureProperty*.

¹<https://www.newtonsoft.com/json>

4.2 Interface

De modo a permitir a interação com o mapa e os dados presentes neste foi desenvolvida a interface representada na Figura 4.2 composta por uma barra de ferramentas sobreposta sobre o mapa. Nesta barra encontram-se, conforme se pode verificar com mais detalhe na Figura 4.3 e da esquerda para a direita, os botões que permitem:

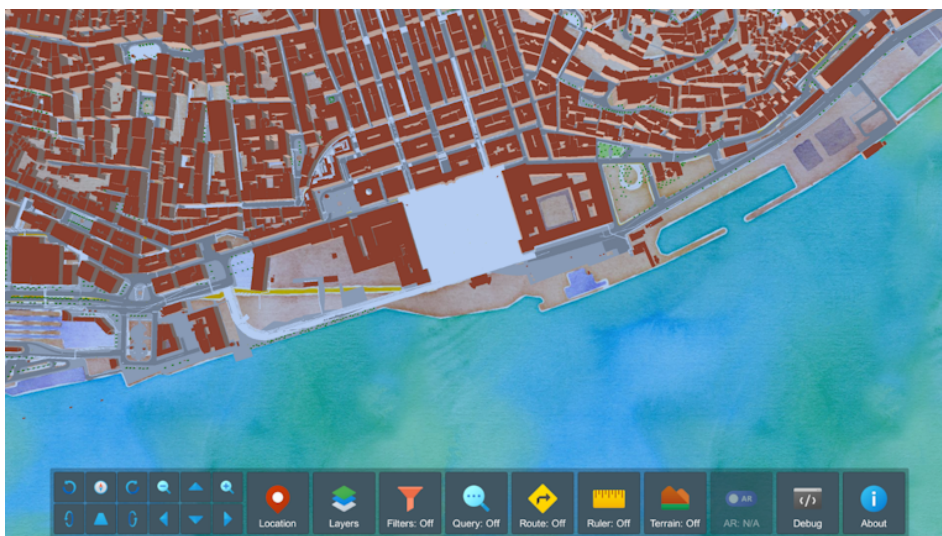


Figura 4.2: Vista principal da aplicação.

1. Controlo da direção do mapa para a esquerda e direita e *reset* relativo ao Norte;
2. Controlo da inclinação do mapa para baixo e cima e *reset* para alinhar perpendicularmente ao mapa;
3. Reduzir e aumentar a ampliação do mapa;
4. Navegar no mapa horizontalmente e verticalmente, relativo à direção atual deste;
5. Abrir o *modal* de localização;
6. Abrir o *modal* das camadas e filtros;
7. Ativar/desativar a utilização dos filtros;
8. Ativar/desativar o modo de consulta de dados;
9. Ativar/desativar a rota;
10. Ativar/desativar a régua;
11. Ativar/desativar a visualização da elevação do terreno;
12. Ativar/desativar o modo de realidade aumentada;

13. Abrir o *modal* de depuração;
14. Abrir o *modal* de informação sobre a aplicação.

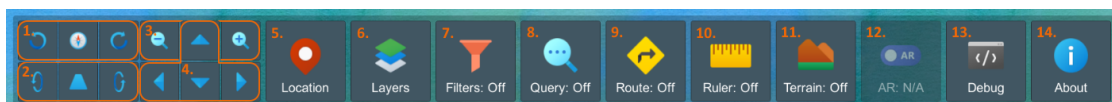


Figura 4.3: Detalhe da barra de ferramentas principal.

4.2.1 Controlos de orientação e posição da câmara

Os controlos de orientação permitem alterar a direção e inclinação da câmara de modo a permitir a visualização dos modelos no mapa a partir de outros ângulos. Os controlos de posição permitem alterar a ampliação e navegar no mapa para permitir a alteração da área a visualizar.

Como alternativa aos botões da barra de ferramentas, os controlos também podem ser usados através de gestos *touch* comuns a este tipo de aplicação se esta estiver a ser utilizada em Android. A navegação no mapa é feita com o arrastar de um dedo, a alteração de inclinação com o movimento vertical no mesmo sentido de dois dedos, a alteração da ampliação com o afastar/aproximar de dois dedos e a alteração da orientação com o movimento circular de dois dedos na direção pretendida.

Adicionalmente, para plataformas com rato e teclado, existe um mapeamento dos controlos com teclas e é também possível clicar e arrastar o rato para navegar no mapa ou usar a *scroll wheel* para alterar a ampliação deste.

4.2.2 Localização

Ao abrir o *modal* de localização, conforme se pode verificar na Figura 4.4, o utilizador depara-se com uma janela *modal* que se sobrepõe à restante aplicação que fica obscurecida, podendo ser fechada ao clicar no botão com uma cruz que se encontra no canto superior direito ou então em qualquer parte da área escura fora da janela. Este comportamento é comum a todos os *modals* presentes na aplicação.

Quanto à janela em questão, esta contém um campo de texto que permite introduzir o nome de um local para pesquisar as suas coordenadas e ir para lá. Existe também um menu *dropdown* onde se pode selecionar um de vários pontos de interesse pré-definidos para preencher automaticamente os campos da latitude e longitude, que se encontram por baixo do menu, com os valores correspondentes às coordenadas desse ponto. À direita destes campos, existe um botão que confirma a alteração da localização do mapa, passando a estar centrado no ponto com as coordenadas inseridas. Caso se pretenda ir para uma localização à qual não existe um ponto de interesse já definido ou se desconheça o nome a pesquisar, os campos das coordenadas podem ser editados para inserir o valor desejado.

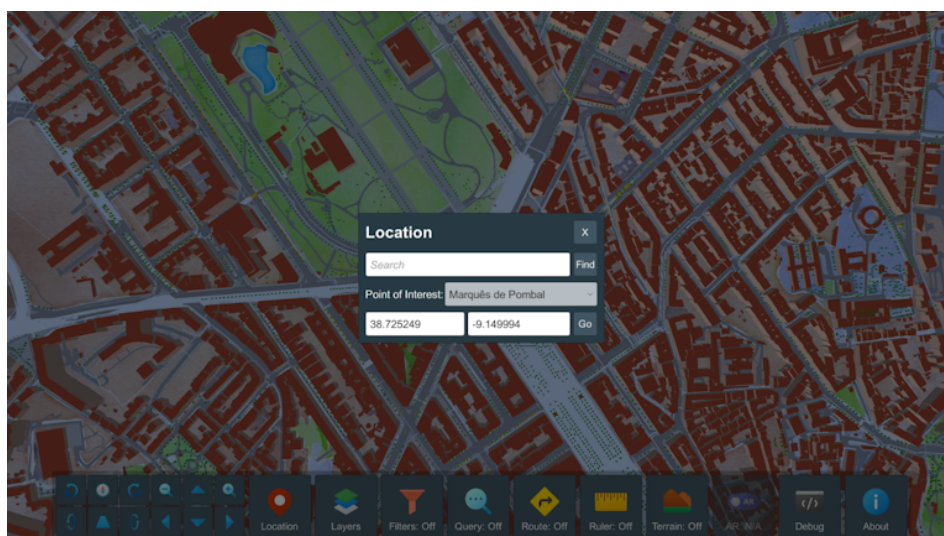


Figura 4.4: *Modal* para alteração da localização.

4.2.3 Camadas e filtros

O *modal* das camadas, presente na Figura 4.5, está dividido em duas áreas para a gestão das camadas e configuração dos filtros associados aos atributos dos objetos destas.

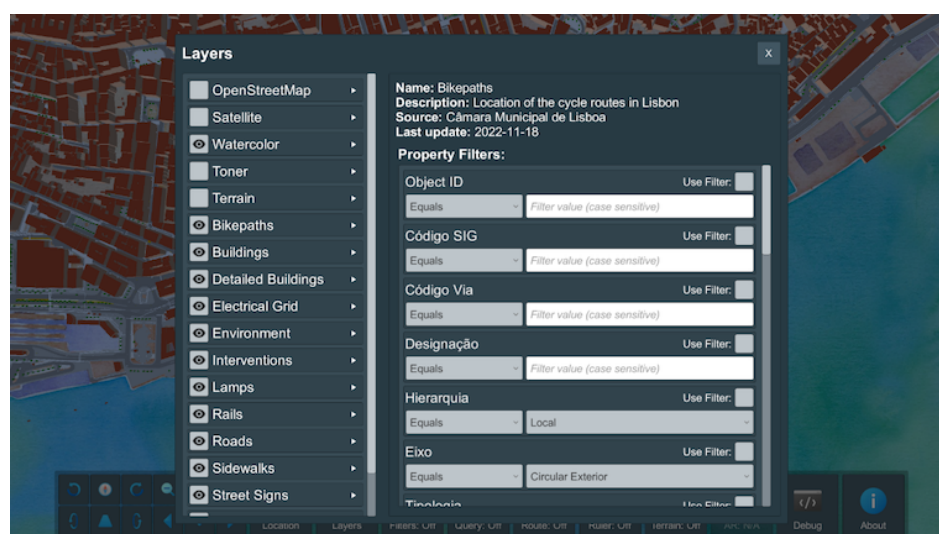


Figura 4.5: *Modal* das camadas com a vista de detalhe da camada selecionada aberta.

A área à esquerda é uma lista de todas as camadas que existem na aplicação, em que cada linha da lista corresponde a uma camada. Cada linha tem uma *checkbox* que controla a visibilidade da camada, aparecendo um ícone de um olho caso a camada esteja visível, e o nome da camada que também pode ser clicado para abrir a vista de detalhe da camada, que surge na área direita da janela.

Na área à direita da janela, caso tenha sido selecionada uma camada, surge a vista de detalhe desta. No seu topo encontra-se uma secção com alguns metadados da camada como o nome, descrição, fonte dos dados e a data da última atualização. Abaixo desta

secção existe uma outra, na qual são listadas todos os filtros de propriedades da camada. Cada entrada da lista contém o nome da propriedade, uma *checkbox* para optar se pretende utilizar o filtro da propriedade, um menu *dropdown* para escolher o filtro e, se aplicável, uma caixa de entrada (ou menu *dropdown* no caso de propriedades categóricas) para indicar o seu valor.

Após seleccionar as propriedades a filtrar, quais os filtros a aplicar e com que valores, o utilizador pode então fechar o *modal* das camadas e ativar ou desativar a filtragem através do botão presente na barra de ferramentas principal, que indica se os filtros seleccionados estão ativos.

4.2.4 Consulta de dados

Para além de visualizar os conjuntos de dados, o utilizador pode consultar mais informação ao ativar o modo de consulta de dados na barra de ferramentas principal. Com este modo ativo, quando se clica sobre um dado objeto, aparece o *modal* com o resultado da consulta de dados, ilustrado na Figura 4.6. Neste pode-se ler as coordenadas do ponto clicado (latitude, longitude e altitude), o seu endereço, o identificador do objeto seleccionado, o nome da camada a que pertence, o identificador da *tile* onde está localizado e também a listagem de todas as propriedades que lhe estão associadas tais como os seus valores.

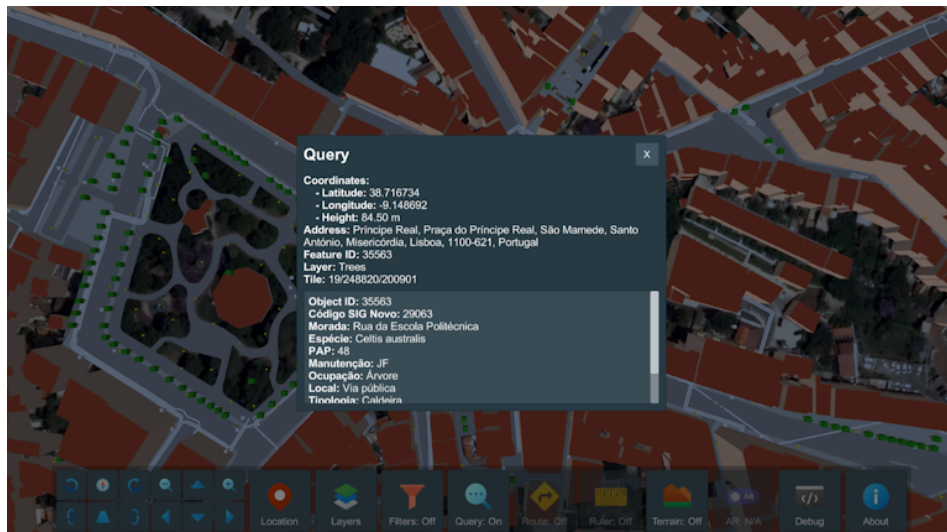


Figura 4.6: *Modal* com resultado da consulta de dados sobre um objeto.

Caso não se encontre nenhum objeto no ponto onde foi feito o clique, mas houver uma camada *raster* ativa, o resultado da consulta mostra apenas as coordenadas, o nome dessa camada e a *tile* localizada nesse ponto, visto que não existem outras propriedades a mostrar.

4.2.5 Rota

A funcionalidade de rota permite ao utilizador visualizar o trajeto entre dois pontos no mapa. Quando o modo de rota é ativado, surge um painel no qual o utilizador pode indicar um ponto de partida e um ponto de destino e pedir para encontrar uma rota entre estes. Se tal não for possível, o painel indica se não foi encontrada uma rota ou se um dos pontos não existe. Caso exista uma rota adequada, o seu resultado é exibido sobre o mapa, à semelhança do exemplo presente na Figura 4.7 que mostra o trajeto entre a Associação Humanitária de Bombeiros Voluntários Lisbonenses e o Parque Mayer.

O ponto de partida é assinalado com um marcador circular verde e o ponto de chegada é assinalado com um marcador circular vermelho. O trajeto entre esses dois pontos é representado por uma linha azul, com nós no local de cada manobra a efetuar. Adicionalmente, a visualização da rota inclui a indicação do modo de deslocação, a duração total e distância a ser percorrida.

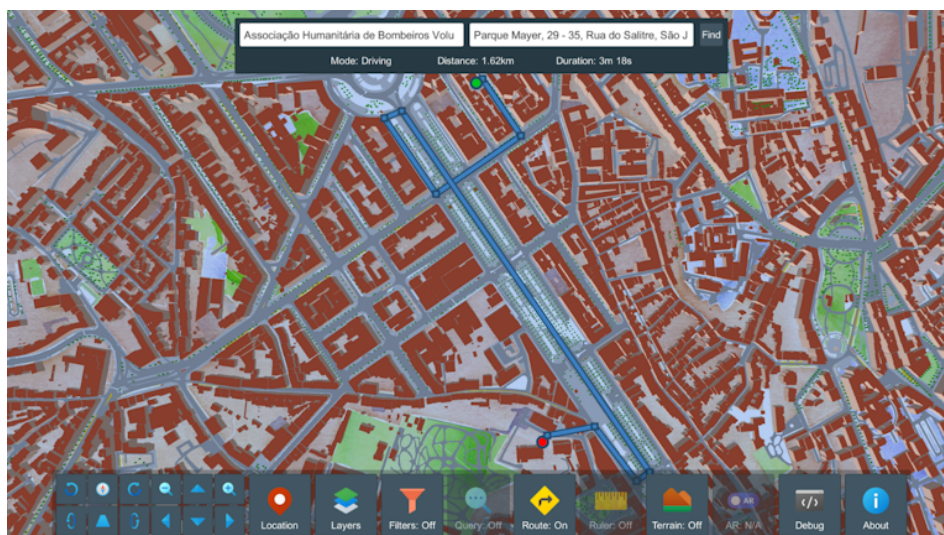


Figura 4.7: Visualização de trajeto entre dois pontos.

4.2.6 Régua

Uma outra ferramenta de análise presente na aplicação é a régua, que permite a medição de distâncias no mapa. Na Figura 4.8 encontra-se um exemplo do seu uso, neste caso a medição do perímetro do Parque Eduardo VII.

Ao ativar o modo da régua, a distância atual medida por esta é indicada num painel que surge no topo da aplicação, em conjugação com um botão para limpar essa medição e voltar ao seu estado inicial.

A régua em si é formada por uma cadeia de nós ligados por linhas retas, sendo que inicialmente não existe nenhum nó e a distância a medir é de 0m. Para adicionar um nó, o utilizador apenas tem de clicar/tocar num ponto no mapa, atualizando nesse instante a distância total medida. Se quiser mover o nó para corrigir a sua posição, tem de clicar/tocar

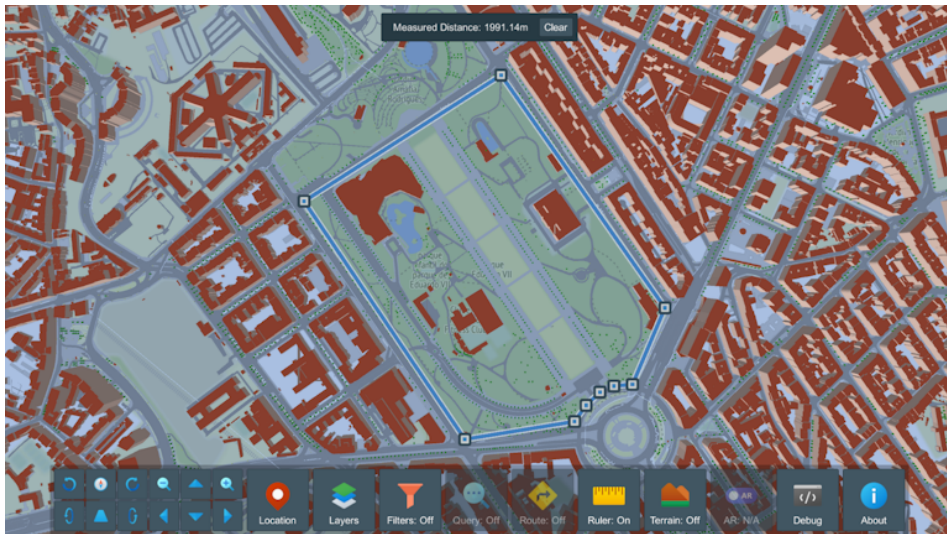


Figura 4.8: Medição de perímetro com recurso à régua.

no nó, manter pressionado e arrastá-lo para a posição pretendida (a distância medida e as arestas que ligam aos nós vizinhos são atualizadas durante o arrastamento, e não apenas no fim). Caso queira remover um nó, basta um simples clique/toque neste para o eliminar.

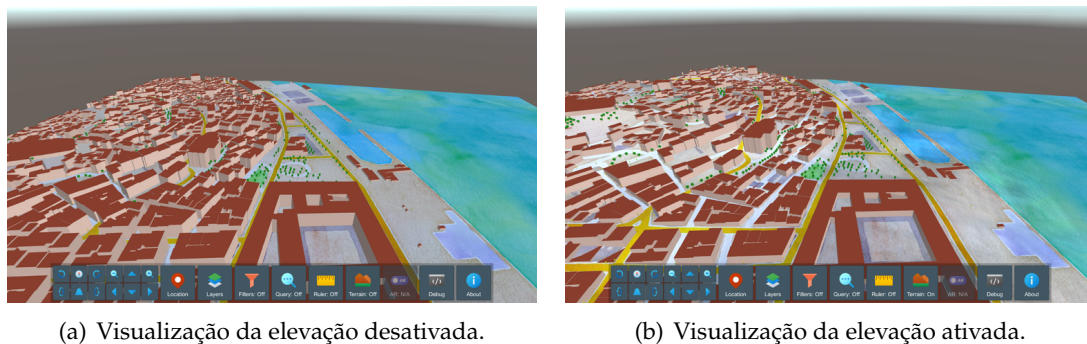
Adicionalmente, quando clicar/tocar numa aresta, é criado um nó nesse ponto entre os dois nós que a definiam, sendo a aresta original eliminada e criadas duas novas arestas para ligar os nós, permitindo uma edição mais simples da régua sem ter de eliminar todos os nós a partir do fim. Se neste último processo o utilizador mantiver pressionado no ponto que tocou da aresta original e arrastar para uma nova posição, o novo nó é logo movido, não sendo necessário um passo adicional após a criação do nó intermédio para o mover, pois toda a sequência faz parte da mesma interação.

Como este modo utiliza o clique/toque para poder criar e mover nós da régua, a navegação no mapa através do arrastar deste encontra-se indisponível enquanto o modo estiver ativo, evitando assim conflitos entre as operações a efetuar.

4.2.7 Visualização de elevação do terreno

Como foi mencionado anteriormente, é possível saber a elevação de um determinado ponto do mapa ou de um objeto que aí se encontre com o modo de consulta de dados, que indica o valor da altitude a partir da informação contida no modelo digital do terreno. No entanto, também é útil que esta informação possa ser transmitida de forma visual, existindo para isso a opção para ativar a visualização da elevação do terreno.

Quando esta visualização é ativada, as *tiles* planas (Figura 4.9(a)) são gradualmente substituídas por *tiles* cujo modelo de base para as camadas *raster* é constituído por um objeto de terreno do Unity gerado a partir dos dados de elevação (Figura 4.9(b)). As camadas vetoriais são também ajustadas de maneira que os objetos se mantenham por cima do terreno e se adaptem a este.



(a) Visualização da elevação desativada.

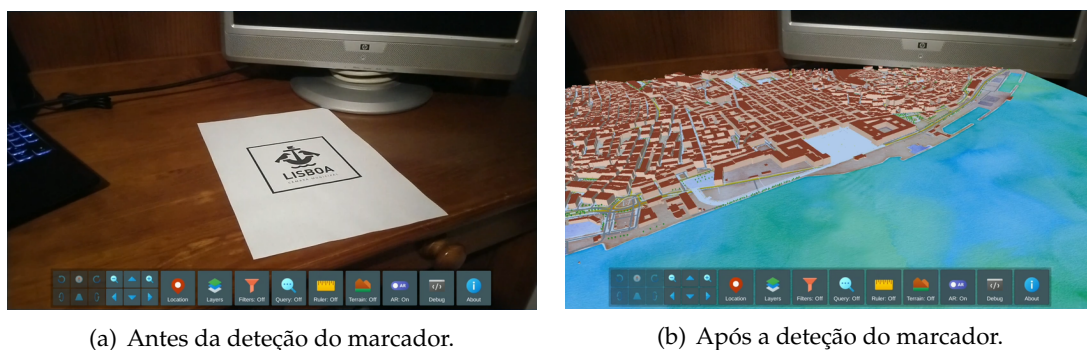
(b) Visualização da elevação ativada.

Figura 4.9: Visualização da elevação do terreno.

Esta visualização permite ter uma noção mais fidedigna dos dados em comparação à visualização plana, permitindo por exemplo a perceção do declive das estradas ou se a vista de um edifício está realmente ocluída por outro que se encontra mais abaixo numa colina. No entanto, devido ao uso de geometria mais complexa pelo terreno, tem a desvantagem de consumir mais recursos, sendo por isso opcional.

4.2.8 Modo de Realidade Aumentada

Em alternativa ao modo 3D tradicional, no qual o mapa "flutua" num vazio a uma distância relativamente constante da câmara, existe a possibilidade de ativar o modo de Realidade Aumentada que, como o seu nome indica, oferece a possibilidade de visualizar o mapa através de realidade aumentada. Para isso, é utilizado a *framework* AR Foundation do Unity, o qual permite a utilização de uma abstração de alto nível para interagir com as bibliotecas nativas de realidade aumentada de cada plataforma, como o ARCore, no caso de aplicações para Android.



(a) Antes da deteção do marcador.

(b) Após a deteção do marcador.

Figura 4.10: Modo de Realidade Aumentada.

Neste modo, a câmara do dispositivo é ativada e a imagem captada é mostrada no ecrã, sendo uma experiência *video see-through*. O utilizador tem então de enquadrar o marcador previamente definido dentro da imagem captada (Figura 4.10(a)). Assim que o marcador é detetado, é feita uma estimativa da sua pose, isto é, a sua posição e rotação relativa à

câmara. Ao obter a pose, torna-se possível sobrepor o mapa paralelamente ao marcador, de modo a simular a sua presença no espaço físico (Figura 4.10(b)).

É também importante de realçar que ao mover a câmara no espaço físico, naturalmente a sua orientação relativamente ao marcador, e por sua vez ao mapa digital que lhe é sobreposto, vai-se alterando. Por esse motivo, as opções relativas aos controlos da rotação e da inclinação do mapa são desativadas enquanto o modo de RA estiver ativo.

4.2.9 Modals de depuração e informação sobre a aplicação

Por fim, existem também as janelas *modal* de depuração e informação sobre a aplicação.

A janela de depuração, conforme se pode ver na Figura 4.11, encontra-se dividida em duas secções. A primeira contém informação sobre o dispositivo (como a memória, processador, ecrã, placa gráfica, sistema operativo e energia) e sobre a aplicação (como a versão, FPS, número de *tiles* carregadas, etc.). A segunda contém o *log* da aplicação, para permitir facilmente visualizar avisos e erros, tal como quaisquer outros tipos de informação relevante escritos na consola, durante a execução da aplicação. Estas mensagens contêm uma marca temporal para fornecer uma referência temporal do evento assim como diferentes categorias (por exemplo: Info, Warning, Error, Exception) que podem ser filtradas.

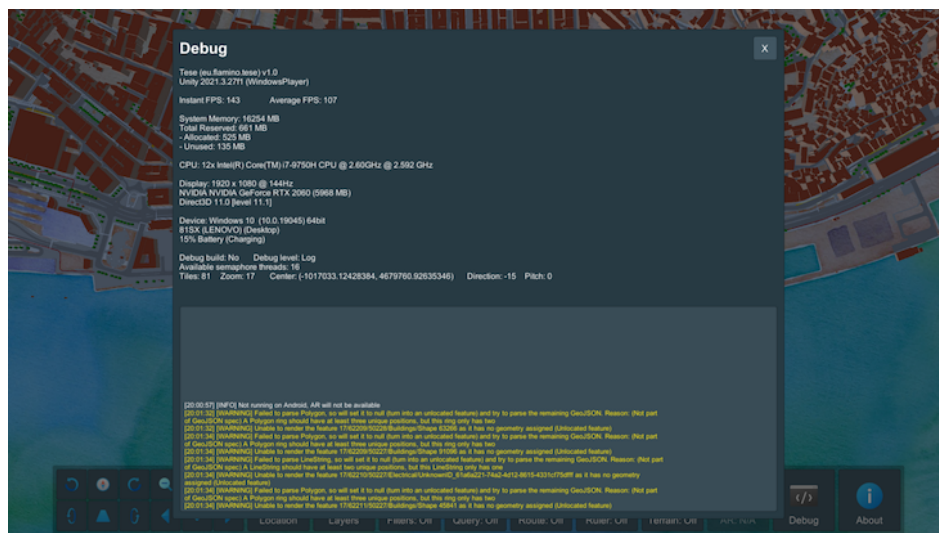


Figura 4.11: Janela de depuração.

Quanto à janela de informação sobre a aplicação (Figura 4.12), contém a ligação para o repositório de código e também as licenças de código aberto da própria aplicação, de código de bibliotecas terceiras, *assets* e conjuntos de dados.

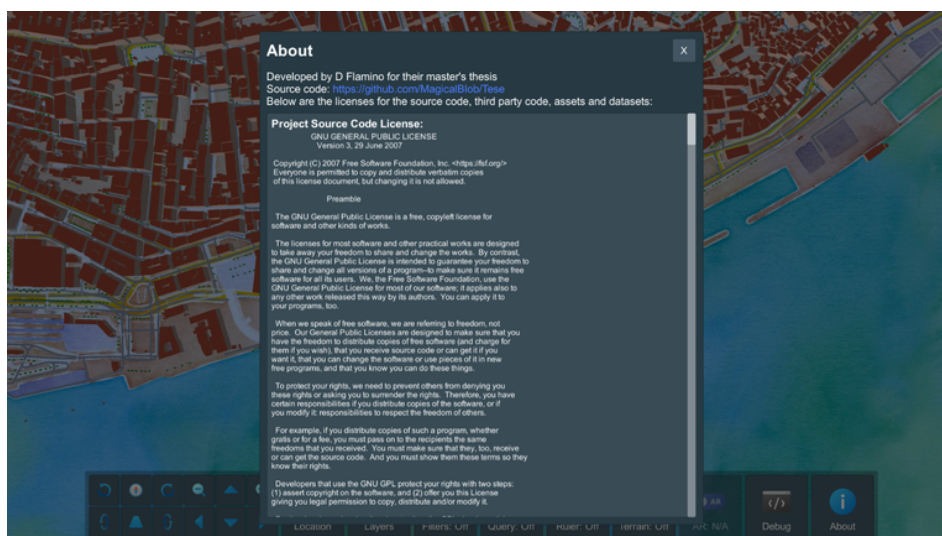


Figura 4.12: Janela de informação da aplicação.

4.3 Resumo

Neste capítulo foi apresentada a aplicação que foi desenvolvida, desenhada para dispositivos *Desktop* e *Mobile* com recurso ao motor de jogos Unity, que se trata de um mapa de conteúdos georreferenciados que podem ser visualizados e analisados. O mapa é a parte central da aplicação e é composto por várias camadas, cada uma correspondendo a um conjunto de dados, divididas em *tiles* para melhor eficiência de carregamento e processamento. O mapa também incorpora um modelo digital do terreno para exibir altitudes e elevações. O capítulo também detalha a interface da aplicação, incluindo os controlos do mapa, mudança de localização, gestão de camadas e configuração de filtros, consulta de dados, rota, régua e visualização de elevação do terreno, entre outros. Adicionalmente existe um modo de Realidade Aumentada *video see-through* com marcadores implementado através do AR Foundation que por sua vez utiliza o ARCore. Em resumo, a aplicação e as suas diversas funcionalidades visam proporcionar interatividade e compreensão dos dados georreferenciados ao permitir a sua visualização e análise.

IMPLEMENTAÇÃO

Neste capítulo são aprofundados os detalhes de implementação do sistema tais como os problemas e desafios encontrados, fornecendo uma visão geral da sua estrutura e vários componentes. Começa-se por relembrar a estrutura geral do sistema, estabelecendo as bases para as secções subsequentes. O capítulo passa então a explorar os servidores responsáveis por servir as *tiles raster* e vetoriais, destacando a sua importância dentro da arquitetura do sistema. De seguida, é detalhado o pré-processamento dos dados, sendo descritas as diferentes técnicas utilizadas na conversão dos conjuntos de dados. É então apresentado o sistema de carregamento de *tiles*, que desempenha um papel crítico no desempenho e eficiência da aplicação. A atenção depois é focada na modelação do terreno e os desafios associados ao uso de *tiles* na sua representação. O capítulo continua com uma exploração do *Parser* de GeoJSON que serve para interpretar as camadas vetoriais e dos desafios de *floating-point precision* na integração de dados georreferenciados no Unity, seguido pelos *renderers* (os diferentes tipos e configuração de camadas, o processo de renderização dos dados, e a adaptação de *renderers* para acomodar vários tipos de conjuntos de dados). Após isto, é feita a discussão dos filtros, explorado o conceito de camadas filtráveis e as suas propriedades, o processo de configuração e aplicação de filtros e as possibilidades de os estender através da criação de outros tipos ou propriedades. Por fim, são detalhados os componentes de Navegação, da Consulta de Dados, Rota, Régua e do Modo de Realidade Aumentada.

5.1 Estrutura geral do sistema

A solução elaborada é uma aplicação multiplataforma desenvolvida com o motor de jogo Unity, que permite visualizar e analisar dados georreferenciados, conforme foi referido na secção 3.2.

A aplicação consiste num mapa de dados georreferenciados, em que os vários conjuntos são representados por camadas, sendo divididos em duas categorias: as camadas Raster e as camadas Vetoriais. Para o caso particular das camadas Vetoriais, o formato GeoJSON foi escolhido devido à sua capacidade de representar objetos geográficos comuns em vários

formatos e por ser baseado em JSON, o que permite a utilização de bibliotecas populares de *parsing* deste.

De modo a garantir uma execução eficiente e evitar carregar todos os conjuntos de dados de uma só vez, o mapa está seccionado em *tiles* para apenas mostrar a área relevante. Então, a aplicação conecta-se a servidores de *tiles* Raster e Vetoriais para obter os conjuntos de dados necessários.

Com o modelo virtual gerado a partir dos conjuntos de dados, é possível realizar várias interações, como navegação básica, seleção de camadas a serem visualizadas, filtragem de objetos com base nas suas propriedades, medições de distâncias ou também visualizar o modelo através de realidade aumentada.

Tal como foi mencionado anteriormente na secção 3.4, a arquitetura desta aplicação é baseada no modelo de três camadas [91], sendo estas a camada de Apresentação, a da Lógica e a dos Serviços de Dados.

5.2 Servidores de *tiles*

Tanto a área de cobertura como a quantidade de informação de um certo conjunto de dados podem variar imenso, o que implica que ao tentar visualizar um conjunto de dados completo, os recursos necessários para o fazer possam ser elevados. Assim, são precisas técnicas que permitam o carregamento e visualização parcial de qualquer conjunto de dados, como as *tiles* XYZ referidas na subsecção 2.1.4.

Visto que esta convenção é amplamente utilizada, optou-se por adotá-la neste projeto. Deste modo, é possível integrar mais facilmente servidores de *tiles* que a utilizem, como é o caso dos servidores de *tiles raster*, sem requerer uma solução feita de raiz. Na Figura 5.1, encontra-se um diagrama com a relação entre os tipos de servidores de *tiles* e a aplicação, em que os conjuntos de dados em formatos Raster provêm de servidores de *tiles raster* e conjuntos em formatos Vetoriais provêm do *geojson-tile-server* (dada a opção de usar GeoJSON para a sua representação).

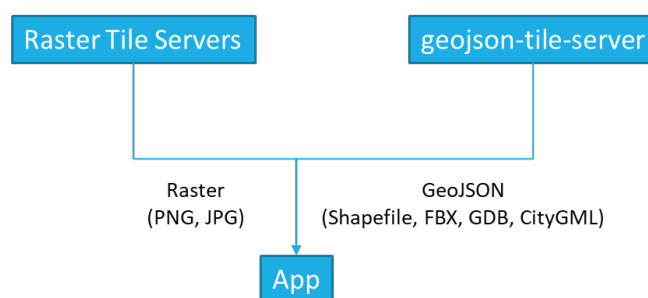


Figura 5.1: Diagrama com relação entre os tipos de servidores de *tiles* e a aplicação.

5.2.1 Servidores de *tiles* Raster

Os conjuntos de dados *raster* são simples de serem segmentados em *tiles*, visto que na sua base trata-se de grelhas de *pixels*, sendo apenas necessário determinar os limites de cada *tile* para poder recortar as imagens em conformidade.

Este procedimento é normalmente efetuado numa fase de pré-processamento dos dados, sendo os resultados da divisão guardados em ficheiros que podem ser servidos aos clientes de forma estática, evitando que se tenha de repetir o processo múltiplas vezes e também diminuindo o tempo de resposta do pedido.

Como se optou por utilizar a convenção de *tiles* XYZ neste projeto e como os formatos *raster* mais comuns são suportados nativamente pelo Unity, é então possível utilizar servidores de *tiles raster* já existentes como fonte para uma dada camada, como é o caso do OpenStreetMap.

5.2.2 Servidor de *tiles* Vetoriais

Quanto aos conjuntos de dados vetoriais, conforme já foi mencionado e ao contrário dos dados *raster*, não são suportados pelo Unity, tendo-se optado por focar em implementar o suporte para GeoJSON neste projeto, sendo os conjuntos que não se encontrem neste formato convertidos para este.

Mesmo com os conjuntos no formato GeoJSON, mantém-se o problema da sua segmentação. Para resolver essa questão, recorreu-se a uma *fork* do *geojson-tile-server*¹, que trata-se de um servidor Node.js que permite obter ficheiros GeoJSON na convenção de *tiles* XYZ.

Este servidor lê os ficheiros GeoJSON originais com os conjuntos completos e processa-os internamente. Quando um cliente pede uma dada *tile* ao servidor, este calcula quais os objetos que intersectam com essa área e gera um novo objeto GeoJSON que contém apenas esses objetos dentro de uma *FeatureCollection* e é retornado ao cliente.

Quanto aos objetos que se situem numa fronteira entre *tiles*, são divididos de forma que cada *tile* apenas contenha a parte do objeto que lhe pertence. No entanto, existe uma área de *buffer* na fronteira em que parte do objeto que fica localizado já na *tile* vizinha é incluído na mesma para evitar problemas com pequenos espaços vazios resultantes do corte dos objetos.

5.3 Pré-processamento dos dados

A utilização de dados georreferenciados requer frequentemente que sejam efetuadas conversões dos tipos de formato de dados. Tal como já foi mencionado na secção 3.3, os conjuntos encontravam-se originalmente em diversos formatos, sendo que podemos agrupá-los em duas categorias: dados *raster* e vetoriais.

¹Disponível em <https://github.com/MagicalBlob/geojson-tile-server>

Embora os dados *raster* não tenham necessitado um pré-processamento, ao serem já consumidos na forma de APIs de servidores de *tiles raster* com formatos suportados pelo Unity como texturas, o mesmo não se pode dizer dos dados vetoriais. Estes conjuntos encontravam-se também em diversos formatos (como GeoJSON, Shapefile, CityGML, JSON e Geodatabase), com a diferença de não serem suportados nativamente pelo Unity.

Isto implicou a conversão dos conjuntos que não estavam no formato GeoJSON, sendo este o único formato vetorial integrado. Na Tabela 5.1 estão indicados os conjuntos utilizados, o seu formato original e quais as conversões/operações aplicadas.

Tabela 5.1: Pré-processamento dos dados vetoriais

ID	Formato Original	Conversões/Operações aplicadas
Bikepaths	GeoJSON	-
Buildings	Geodatabase	GeoDatabase para Shapefile; Shapefile para GeoJSON
BuildingsLOD3	Geodatabase	GeoDatabase para FBX + GeoJSON
Electrical	Shapefile	Shapefile para GeoJSON; União de conjuntos GeoJSON
Environment	JSON	Script ConvertEnvironmentSensorsJsonToGeoJson
Interventions	GeoJSON	-
Lamps	GeoJSON	-
PedNet	Shapefile	Shapefile para GeoJSON; União de conjuntos GeoJSON
Rails	GeoJSON	-
Roads	CityGML	CityGML para GeoJSON
RoadsOld	GeoJSON	-
Sidewalks	CityGML	CityGML para GeoJSON
Signs	Shapefile	Shapefile para GeoJSON
TrafficRestrictions	GeoJSON	Script FlattenTrafficRestrictionsGeoJson
Trees	GeoJSON	-

Alguns destes conjuntos de dados, como o Electrical e PedNet, eram originalmente conjuntos de dados separados que, após convertidos para GeoJSON, foram unidos num único conjunto. No caso dos conjuntos que originaram o Electrical, eram conjuntos simples que apenas indicavam pontos do mesmo tipo (por exemplo, localização de subestações elétricas, pontos de transformação de cliente ou distribuição), sem quaisquer propriedades associadas. Por esse motivo, os conjuntos foram unidos e cada ponto passou a ter uma propriedade que indica o tipo de infraestrutura elétrica (que acaba por corresponder ao nome do seu conjunto original). Quanto aos conjuntos que foram unidos para criar o PedNet, esses tinham as mesmas propriedades, mas cobriam zonas diferentes. Por esse

motivo, a sua união foi mais trivial e foi feita apenas para consolidar num único conjunto com uma área de cobertura mais abrangente.

A partir da Tabela 5.1, podemos também observar duas categorias de transformações, conversões de formatos (Shapefile para GeoJSON, GeoDatabase para Shapefile e GeoDatabase para FBX + GeoJSON) e utilização de scripts para alterar a maneira como estão estruturadas as propriedades nos objetos GeoJSON, que serão descritos de seguida.

5.3.1 Shapefile/CityGML para GeoJSON

Dada a quantidade de conjuntos de dados originalmente nos formatos Shapefile e CityGML, a conversão de conjuntos nestes formatos para GeoJSON foi a mais frequentemente efetuada. Embora o tipo de objetos que o formato CityGML permite representar seja bastante mais complexo, os conjuntos de dados fornecidos apenas se referiam a polígonos das estradas e passeios. Deste modo, foi possível a conversão com recurso ao programa QGIS, sendo o processo igual para ambos os formatos Shapefile e CityGML.

Para converter um conjunto Shapefile ou CityGML para GeoJSON, tem de se começar por abrir um novo projeto no QGIS. Depois, clicar em "Add Vector Layer" e especificar a localização do ficheiro do conjunto de dados vetorial. Após adicionar a camada, esta tem de ser selecionada e, com um clique direito para abrir o menu de contexto, selecionar "Export > Save Features As...". Indicar GeoJSON como o formato de output, o nome a usar para a gravação do ficheiro, EPSG:4326 como [CRS](#) e que se pretende gerar a *bounding box* se ainda não existir. Finalmente, clicar em OK e aguardar que o QGIS efetue a geração do ficheiro GeoJSON.

5.3.2 GeoDatabase para Shapefile

O conjunto Buildings trata-se de uma representação tridimensional simplificada dos edifícios de Lisboa, no formato ESRI File GeoDatabase. Como as representações dos edifícios limitam-se a uma extrusão da sua planta, com o valor a extrudir já presente nas suas propriedades, a opção mais viável foi a sua conversão para Shapefile, por também se tratar de um formato da ESRI.

Para isso foi utilizado o Esri ArcGIS CityEngine do seguinte modo: começa-se por abrir uma cena em branco no CityEngine e selecionar EPSG:3857 como o seu [CRS](#). Depois, clica-se em "File > Import" e seleciona-se "File GDB Import". Especifica-se a localização da File GeoDatabase e aguarda-se pela sua leitura. A seguir, seleciona-se quais os conjuntos de dados a importar, clica-se em "Finish" e aguarda-se que os dados sejam importados. Após isso, clica-se em "File > Export" e seleciona-se "Export Selected Shapes". Escolhe-se a opção "Esri Shapefile", indica-se o nome do ficheiro a ser gravado e clica-se em "Finish". Finalmente, aguarda-se que o CityEngine conclua a geração do Shapefile.

5.3.3 GeoDatabase para FBX + GeoJSON

Embora o conjunto BuildingsLOD3, à semelhança do Buildings, esteja no formato ESRI File GeoDatabase, os modelos dos edifícios são bem mais detalhados, inviabilizando a estratégia de conversão para Shapefile. Em alternativa, os modelos foram convertidos para o formato FBX, que pode ser importado como mesh no Unity e criado um conjunto GeoJSON que indica a localização desses modelos.

Também recorrendo ao CityEngine, começa-se por abrir uma cena em branco e selecionar EPSG:3857 como o seu [CRS](#). Depois, clica-se em "File > Import" e seleciona-se "File GDB Import". Especifica-se a localização da File GeoDatabase e aguarda-se pela sua leitura. A seguir, seleciona-se quais os conjuntos de dados a importar, clica-se em "Finish" e aguarda-se pela importação dos dados.

Após isso, clica-se em "File > Export" e seleciona-se "Export Models of Selected Shapes and Terrain Layers". Escolhe-se a opção "Autodesk FBX", indica-se o nome do ficheiro a ser gravado, ativa-se a opção "Triangulate Meshes" e clica-se em "Center" na secção "Global Offset". Ainda na secção "Global Offset", altera-se "Y-Offset" para 0 e são copiados os valores dos offsets X e Z. Clica-se em "Finish" e aguarda-se que o CityEngine conclua a geração do modelo FBX.

De seguida, é preciso converter os valores offsets X e Z do [CRS](#) EPSG:3857 para EPSG:4326. Utiliza-se o offset X como a coordenada da longitude e inverte-se o seu sinal, e o offset Z como coordenada da latitude. Num ficheiro de texto, cria-se um objeto GeoJSON com uma FeatureCollection, em que a Feature usa o par longitude/latitude como coordenadas de um ponto (que representa o centro do objeto) e indica o nome do modelo FBX como propriedade adicional, guardando o ficheiro com a extensão `.geojson`.

É de notar que o processo descrito gera um único modelo, sendo necessário repetir para cada edifício e criar o GeoJSON final manualmente. Devido a problemas no conjunto de dados original utilizado, não foi possível separar os edifícios individualmente, tendo estes sido exportados como um único modelo. Visto que a cobertura do conjunto era limitada a apenas dois quarteirões (na Baixa e na Alta de Lisboa), este processo mostrou-se adequado, embora tenha limitado a hipótese de testar a utilização de níveis de detalhe, em que se faria uma combinação destes edifícios com os presentes no conjunto Buildings, para mudar a sua representação de acordo com a escala do mapa.

5.3.4 Scripts de normalização dos dados

Embora a maioria dos conjuntos de dados não exija um pré-processamento especial, existem alguns (como o Environment e o TrafficRestrictions) que necessitam de um tratamento extra para que as suas propriedades possam ser corretamente interpretadas pelo sistema.

ConvertEnvironmentSensorsJsonToGeoJson

O conjunto Environment tratava-se originalmente de um conjunto JSON, pelo que a sua transformação em GeoJSON é relativamente simples, visto que este formato é derivado do JSON. No entanto, o sistema atualmente não suporta que um objeto tenha uma propriedade do tipo Array, mesmo que tal seja possível para um objeto em ambos os formatos. Por esses motivos, foi utilizado o *script* presente na Listagem 5.1 para a sua conversão.

```

1  var json = JSON.parse(jsonText);
2  var geojson = {type: "FeatureCollection", features: []};
3  json.forEach(entry => {
4    var lng = entry.coordinates.lng;
5    var lat = entry.coordinates.lat;
6    delete entry.coordinates;
7    // Convert date
8    var date = entry.date;
9    entry.date = date.substr(0,4) + "-" + date.substr(4,2) + "-" + date.substr(6,2) +
    ↳ "T" + date.substr(8,2) + ":" + date.substr(10,2) + "Z";
10   // Flatten directions
11   if (entry.directions){
12     if (entry.directions["from-to"]){
13       entry.from_to_direction = entry.directions["from-to"].direction;
14       entry.from_to_P = entry.directions["from-to"].P;
15       entry.from_to_M = entry.directions["from-to"].M;
16       entry.from_to_L = entry.directions["from-to"].L;
17       entry.from_to_total = entry.directions["from-to"].total;
18     }
19     if (entry.directions["to-from"]){
20       entry.to_from_direction = entry.directions["to-from"].direction;
21       entry.to_from_P = entry.directions["to-from"].P;
22       entry.to_from_M = entry.directions["to-from"].M;
23       entry.to_from_L = entry.directions["to-from"].L;
24       entry.to_from_total = entry.directions["to-from"].total;
25     }
26     delete entry.directions;
27   }
28   var feature = {type: "Feature", id: entry.id, properties: entry, geometry: {type:
    ↳ "Point", coordinates: [lng, lat]}};
29   geojson.features.push(feature);
30 }
31 return JSON.stringify(geojson, null, 2);

```

Listagem 5.1: ConvertEnvironmentSensorsJsonToGeoJson.

A função começa por interpretar o texto de input JSON e inicializar um objeto vazio GeoJSON do tipo FeatureCollection. Depois, itera por cada objeto presente no JSON.

Para cada um, os valores da longitude e latitude são extraídos da propriedade coordinates. A propriedade date é convertida para o formato ISO 8601.

A seguir, a propriedade directions é *flattened* ao extrair os valores direction, P, M, L e total das direções "from-to" e "to-from", que são atribuídos a propriedades correspondentes na entrada, sendo a propriedade original directions apagada.

Após isso, é criado um objeto GeoJSON do tipo Feature, com um identificador obtido

a partir da propriedade `id` da entrada, as propriedades da entrada modificada e um objeto geométrico do tipo `Point` cujas coordenadas são o par longitude/latitude extraído.

Finalmente, a `Feature` é inserida na array da `FeatureCollection` e, quando todas as entradas foram processadas, a função retorna o objeto GeoJSON final.

FlattenTrafficRestrictionsGeoJson

Quanto ao conjunto `TrafficRestrictions`, este já se encontrava de origem no formato GeoJSON mas, à semelhança do `Environment`, também continha propriedades do tipo `Array`. Por isso, foi utilizado o *script* da Listagem 5.2 para alterar a sua estrutura.

```
1  var geojson = JSON.parse(geojsonText);
2  var flattenedGeojson = geojson;
3  flattenedGeojson.features.forEach(entry => {
4    if (entry.properties.periodos_condicionamentos){
5      if (entry.properties.periodos_condicionamentos.length > 0){
6        var firstRestriction = entry.properties.periodos_condicionamentos[0];
7        entry.properties.dtm_min = firstRestriction.date_min + "T" +
8          ↳ firstRestriction.hour_min + "Z";
9        entry.properties.dtm_max = firstRestriction.date_max + "T" +
10         ↳ firstRestriction.hour_max + "Z";
11      }
12      delete entry.properties.periodos_condicionamentos;
13    }
14  });
15  return JSON.stringify(flattenedGeojson, null, 2);
```

Listagem 5.2: FlattenTrafficRestrictionsGeoJson.

A função começa por interpretar o texto de input GeoJSON e cria uma cópia deste para ser alterada. Depois, itera por cada uma das `Features` presentes no GeoJSON.

Para cada `feature`, verifica se a propriedade `periodos_condicionamentos` existe e se tem um tamanho maior que 0. Se for o caso, a primeira restrição é extraída a partir da array `periodos_condicionamentos` e os valores das suas propriedades `date_min`, `hour_min`, `date_max` e `hour_max` são atribuídos às novas propriedades da `Feature` `dtm_min` e `dtm_max`, respetivamente. Os valores da data e hora são combinados para que o resultado esteja no formato ISO 8601.

Finalmente, a propriedade `periodos_condicionamentos` é apagada e finda a iteração é retornado o novo GeoJSON.

5.4 Sistema de carregamento de Tiles

Recapitulando brevemente o que foi dito na subsecção 4.1, o mapa é composto por várias camadas que representam diversos conjuntos de dados e está dividido em várias Tiles, que por sua vez contêm múltiplas TileLayers que correspondem a cada uma das camadas do mapa. No entanto, as TileLayers apenas gerem os dados da camada relativos à área da Tile, pois a gestão de tudo o que é transversal à camada (ex: configuração de filtros) é feita ao nível da Layer. Esta divisão em *tiles* permite manter carregadas na memória apenas as que são necessárias para cobrir a área de visualização, evitando assim o consumo excessivo de recursos do sistema. Nesta secção começa-se por descrever como é feita a gestão das Tiles ao nível do mapa e, de seguida, os ciclos de vida das Tiles e TileLayers.

5.4.1 O mapa e a gestão das Tiles

Três das propriedades do mapa relevantes para o carregamento das Tiles são as coordenadas do seu centro, isto é, as coordenadas do ponto que fica localizado na origem da cena do Unity, a distância de carregamento e o número da geração atual das Tiles. Quando o mapa é inicializado, é feito o primeiro carregamento de Tiles, que corresponde à primeira geração destas.

O carregamento começa por incrementar a geração atual e determinar as coordenadas segundo a convenção de *tiles XYZ* da Tile que fica localizada na origem do mapa a partir das coordenadas do centro deste. Então, para todas as Tiles que se encontrem a uma distância menor ou igual à distância de carregamento da Tile da origem, verifica-se se está presente na coleção de Tiles carregadas atualmente para evitar repetir pedidos desnecessariamente. Se não estiver presente, é criado o seu objeto e feito o carregamento. Caso já tivesse sido carregada, a sua geração é atualizada para corresponder à do mapa, pois continua dentro da área a visualizar.

Quanto ao descarregamento das Tiles antigas, é feita uma iteração pelas Tiles e, caso a sua geração seja inferior à do mapa, a Tile é escondida, inicia-se o processo de descarregamento e é removida da coleção de Tiles carregadas.

Para além do carregamento aquando da inicialização do mapa, as Tiles carregadas apenas são alteradas em duas situações, quando é feita uma translação da área a visualizar ou quando a ampliação do mapa é modificada. No caso da translação, começa-se por atualizar o centro do mapa e aplicar essa transformação a todas as Tiles, seguido pelo carregamento das Tiles que estejam presentes na nova área de visualização e, por fim, pelo descarregamento das que ficarem fora da área de visualização. A alteração da ampliação implica que nenhuma Tile se irá manter carregada, por isso é apenas feito o carregamento das Tiles com a nova ampliação e o descarregamento das que tinham o valor anterior.

5.4.2 Ciclo de vida das Tiles e TileLayers

Explicado o modo como é feita a gestão das Tiles ao nível do mapa, pode-se agora proceder à descrição do ciclo de vida destas unidades, que são relativamente independentes umas das outras, decorrendo em paralelo por uma questão de eficiência. A Figura 5.2 está o diagrama de transição de estados de uma Tile, com os estados em que esta se pode encontrar: *Initial*, *TerrainLoadFailed*, *TerrainLoaded*, *LayersLoadFailed*, *Loaded*, *Unloaded*.

A Tile tem o estado *Initial* aquando da sua criação. O processo de carregamento da Tile está dividido em duas fases, o carregamento do modelo digital do terreno e o das várias camadas de dados. Quando o carregamento do terreno é iniciado, a Tile passa para *TerrainLoaded* ou *TerrainLoadFailed* consoante se a operação foi sucedida ou não, para se poder tentar de novo. Após o carregamento com sucesso do terreno, tenta-se carregar em paralelo as várias camadas de dados, passando nesta fase para *Loaded* ou *LayersLoadFailed* também consoante o sucesso das tarefas, à semelhança do carregamento do terreno. Por fim, quando é requisitado o descarregamento da Tile com o método *Unload*, é feito o cancelamento de quaisquer pedidos pendentes ao servidor se ainda estiver nos estados *Initial* ou *TerrainLoaded* e, para qualquer estado, inicia-se o processo de descarregamento da Tile, das suas várias *TileLayers* e objetos associados.

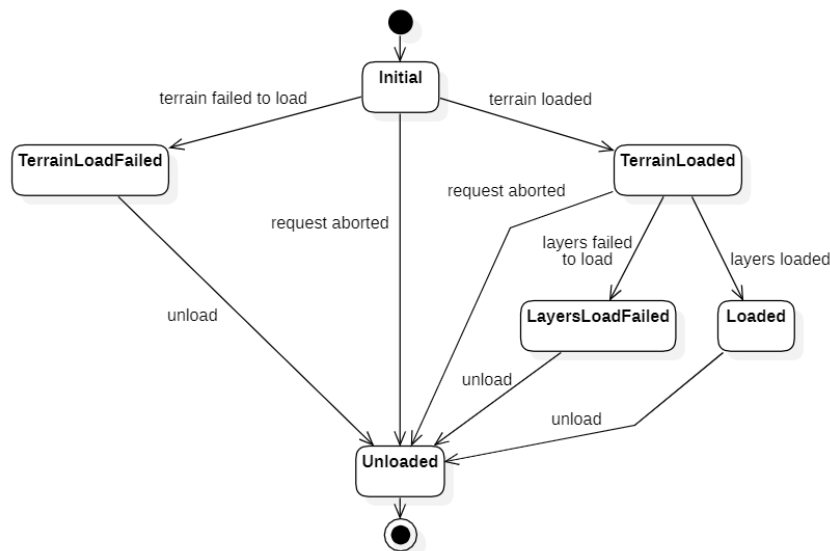


Figura 5.2: Diagrama de transição de estados de uma Tile.

Quanto à *TileLayer*, o seu diagrama de transição de estados está representando na Figura 5.3 e pode ter os estados *Initial*, *LoadFailed*, *Loaded*, *Rendered*, *Unloaded*. É mais simples dado que a maior parte das complexidades são tratadas nas camadas de abstração mais acima.

Cada *TileLayer* também começa com o estado *Initial* aquando da sua criação, que ocorre quando a *Tile* parente está no estado *TerrainLoaded* e inicia o processo de carregamento das camadas de dados. A *TileLayer* passa para *Loaded* ou *LoadFailed* consoante o sucesso do pedido de carregamento dos dados da área da *Tile* para o conjunto

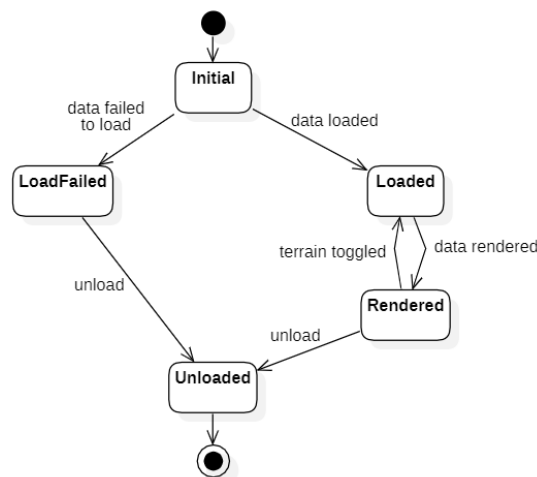


Figura 5.3: Diagrama de transição de estados de uma TileLayer.

de dados correspondente à camada. Com os dados carregados, estes são interpretados para ser feita a sua renderização, passando então para o estado Rendered. Caso tenha sido ativado ou desativado o modo de visualização do terreno, volta para o estado Loaded até a renderização completar outra vez. Por fim, tal como acontece com a Tile, passa para o estado Unloaded quando é feito o processo de descarregamento.

5.5 Terreno

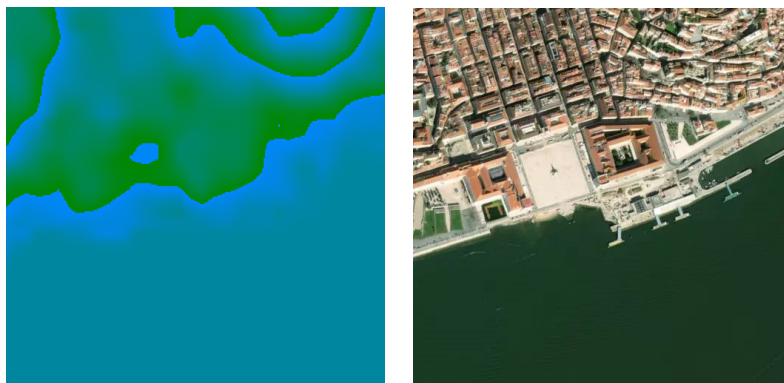
Como foi dito anteriormente, foi utilizada a API de elevação da Mapbox para a geração do modelo tridimensional do terreno. Os dados correspondentes ao território europeu foram produzidos a partir dos dados do programa Copernicus da União Europeia [2], que têm uma resolução espacial de 25 metros e exatidão vertical $\pm 7\text{m}$ [25]. É agora descrita como foi utilizada essa informação e problemas encontrados.

5.5.1 Modelação do terreno a partir das *tiles* Terrain-RGB da Mapbox

Embora a API REST de elevação da Mapbox permita determinar a altitude de um ponto dadas a sua latitude e longitude através de um pedido que retorna uma resposta JSON, esta não é uma solução eficiente em termos de utilização da rede para algo mais do que obter a altitude de um dado objeto. Como o que era pretendido para este trabalho era a construção de um modelo tridimensional do terreno, seriam necessários demasiados pedidos individuais para obter a informação que o permitisse fazer. Felizmente, existe outra API de elevação fornecida pela Mapbox mais adequada a este caso, a API de *tiles raster* Terrain-RGB.

As *tiles raster* Terrain-RGB são, como o nome indica, um conjunto de *tiles* XYZ nas quais os dados de elevação de terreno são codificados nas componentes de cores da imagem, utilizando cada um como uma posição num sistema de numeração de base 256,

o que permite ter 16777216 valores únicos, de -10000m a 1667721,5m, estando estes valores mapeados em incrementos de 0,1 metros [38].



(a) Resposta da API Terrain-RGB.

(b) Camada Satellite.

Figura 5.4: Área coberta pela *tile* 15/15552/12557.

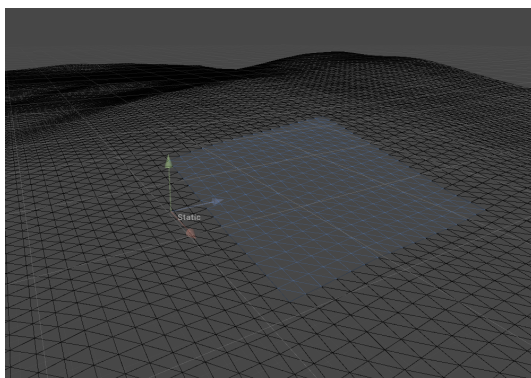
Ao utilizarem o mesmo padrão XYZ que as restantes *tiles* da aplicação, estas cobrem exatamente a mesma área (Figura 5.4) e permitem obter toda a informação relevante para cada *tile* com apenas um pedido. Assim, no processo de carregamento de uma *tile* na aplicação, conforme foi explicado na subsecção 5.4.2, é feito o pedido da *tile* Terrain-RGB correspondente, antes dos pedidos relativos às camadas de dados a serem sobrepostas, cuja visualização pode depender dos dados de elevação. Quando é obtida a resposta ao pedido, a informação das elevações é extraída da textura, pegando nas componentes vermelha (R), verde (G) e azul (B) de cada pixel da imagem e aplicando a seguinte fórmula para obter a sua altitude em metros:

$$h = -10000 + ((R \cdot 256 \cdot 256 + G \cdot 256 + B) \cdot 0.1)$$

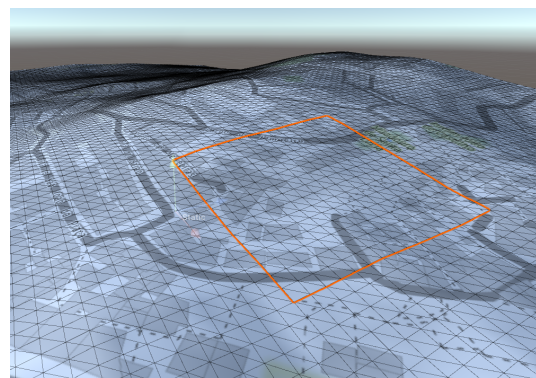
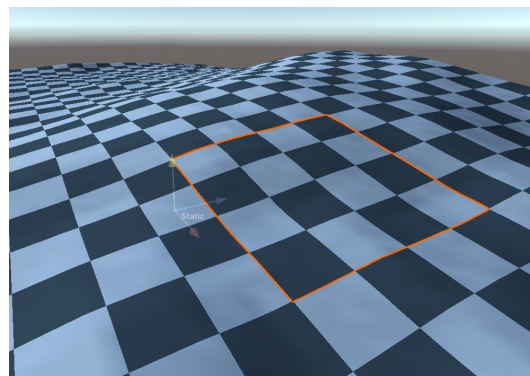
É nesta forma de codificação das cores que uma *tile* Terrain-RGB difere de um típico *heightmap*, pois este último trata-se de uma imagem *grayscale* em que cada pixel representa um nível de elevação do terreno, com tons claros a representar altitudes mais elevadas e escuros para mais baixas, sendo possível ter uma ideia do relevo do terreno apenas ao olhar para a imagem. Como as *tiles* Terrain-RGB codificam a elevação utilizando os três componentes de cor, não é possível ter uma ideia completa do relevo, visto que foram desenhadas para serem processadas por aplicações e não para serem visualizadas pelo utilizador final, enquanto aproveitam a mesma infraestrutura que serve as *tiles raster* clássicas (que utilizam os três componentes para formar as imagens a ser visualizadas).

Tendo a informação já convertida em metros, já pode ser utilizada por outras partes da aplicação, como no caso da consulta de dados para um certo ponto, que mostra a elevação em conjunto com a sua latitude e longitude, ou das camadas GeoJSON que podem utilizar a elevação para deslocar verticalmente as suas representações de dados para que estejam alinhadas com o terreno. As camadas *raster* são o caso de uso mais importante para estes dados, visto que ao ativar a visualização do terreno deixam de ser representadas por

um plano para ser utilizado o sistema de terreno do Unity. Nesse caso, as alturas são normalizadas numa escala de 0 a 1, em que 0 corresponde à elevação mínima que pode ser representada e 1 corresponde à elevação máxima, sendo estes valores configuráveis. A partir das alturas normalizadas é então criado o objeto de terreno do Unity (Figura 5.5) e é-lhe aplicada a imagem da *Tile raster* como a sua textura (Figura 5.5(b)). Caso não seja possível obter a imagem da *Tile*, quer por indisponibilidade do servidor ou por estar fora da área de cobertura do conjunto de dados, é aplicada uma textura por defeito (Figura 5.5(c)).



(a) Malha poligonal.

(b) Terreno com imagem da *tile* aplicada.(c) Terreno com textura *default*.Figura 5.5: *Tile* criada com sistema de terreno do Unity.

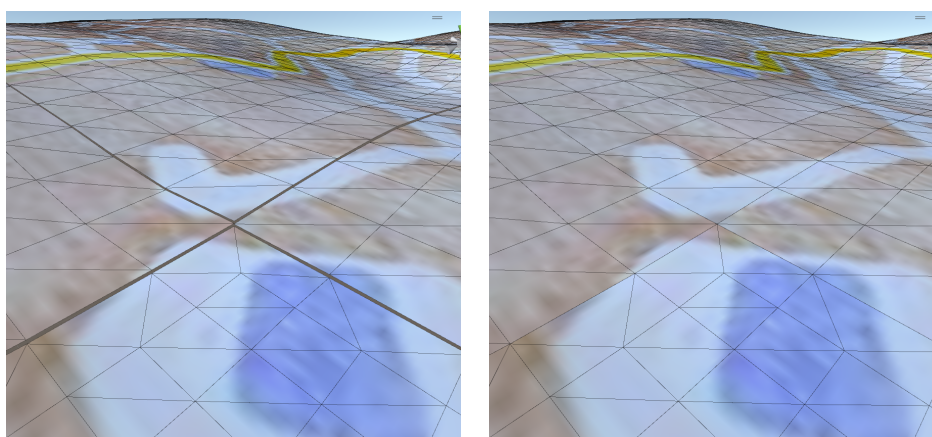
Originalmente, este projeto utilizava uma malha gerada a partir dos dados de elevação, mas essa solução acabou por ser abandonada em favor da utilização do sistema de terreno do Unity porque exigia demasiado processamento gráfico para renderizar grandes áreas, o que acabava por comprometer o desempenho da aplicação. Por outro lado, o sistema de terreno do Unity, para além de estar mais desenvolvido e integrado com o próprio Unity, utiliza uma técnica de nível de detalhe (LOD) que reduz a quantidade de detalhes para objetos distantes da câmara, mantendo assim um bom desempenho em grandes áreas.

5.5.2 Desafios no uso de *tiles* para o terreno: Espaços nas fronteiras e dependências entre Tiles

Um dos desafios principais encontrados ao aproveitar a mesma divisão em *tiles* para o modelo digital do terreno é de que as Tiles não são completamente independentes umas das outras. A geração do modelo tridimensional do terreno, quer na estratégia original de criar a malha, quer na utilização do sistema de terreno do Unity, tem um caso particular ao longo dos limites da *tile*.

O modelo digital do terreno utilizado, obtido pela API Terrain-RGB da Mapbox, contém amostras do valor da elevação em cada ponto correspondente a cada pixel da imagem, sendo efetuada uma interpolação desses dados para obter quaisquer valores intermédios que sejam relevantes. Como o sistema de terreno do Unity usa um *heightmap* para representar a altura, apenas é feita a normalização dos dados de elevação para uma escala de 0 a 1, visto que a resolução desse *heightmap* pode ser definida para corresponder com a resolução da *tile* Terrain-RGB, ficando o próprio sistema de terreno encarregue de realizar internamente as interpolações que para si forem necessárias.

Se tratarmos as *tiles* de um modo completamente independente umas das outras, pode ser inadvertidamente criado um espaço vazio entre estas, como podemos observar na Figura 5.6(a). Esta distância entre as arestas fronteiriças das *tiles* deve-se à diferença dos valores discretos da elevação nesses pontos fronteiriços. Para colmatar essa discrepância, sem recorrer a dados que não pertencessem à mesma *tile*, foi inicialmente utilizada uma extrapolação que determinava a altura a que ficaria o ponto pertencente à *tile* vizinha a partir dos declives dos três pontos anteriores o declive dos três pontos.



(a) Fronteiras com valores fixados.

(b) Fronteiras cozidas com as vizinhas.

Figura 5.6: Detalhe das fronteiras entre *tiles*.

Visto que se tratava de uma extrapolação, esta solução continuava a gerar artefactos, especialmente se o terreno tivesse variações irregulares da sua elevação. Optou-se então por consultar a *tile* vizinha para obter os seus dados de elevação reais, de maneira que o sistema de terreno consiga cozer as duas *tiles* corretamente (Figura 5.6(b)). No entanto, as

tiles continuam a ser carregadas paralelamente e independentemente umas das outras, o que implica que as suas vizinhas podem não ter dados disponíveis por ainda terem os seus respetivos pedidos pendentes ou até por estarem completamente fora da área de visualização e não serem carregadas de momento.

Se a *tile* vizinha estiver carregada, os dados da fronteira são logo copiados para poder gerar o terreno. Se não tiver já os dados carregados, pois pode ter um pedido pendente, são num primeiro passo fixados os valores que pertenceriam à *tile* vizinha para corresponder aos pontos limítrofes (*clamping*) para gerar um modelo temporário e, de seguida, agendada uma tarefa. Esta tarefa aguarda uma quantidade de tempo definida para que possam ser completados quaisquer pedidos pendentes e torna então a verificar os dados da *tile* vizinha.

Para os casos em que as *tiles* vizinhas estejam fora da área de visualização, quando esta é alterada e as *tiles* que se mantêm carregadas vêm a sua geração incrementada (conforme explicado na subsecção 5.4.1), é feita uma verificação se algumas delas ainda não têm dados das suas vizinhas e, se for o caso, é outra vez agendada a tarefa que aguarda pelo carregamento das vizinhas para obter os seus dados e proceder à costura final das *tiles*.

5.6 Integração de dados georreferenciados

A integração de dados georreferenciados é uma tarefa crucial nesta aplicação, embora não seja suportada nativamente pelo Unity. Nesta secção, são discutidos dois aspetos essenciais dessa integração: o *parser* de GeoJSON e os desafios relacionados com *floating-point precision* na integração de dados georreferenciados no Unity.

5.6.1 Parser de GeoJSON

O *parser* de GeoJSON é um componente que recebe um texto e o tenta converter, se possível, numa representação de objetos GeoJSON. Isto é feito com recurso à biblioteca Json.NET que trata da interpretação do texto, a sua divisão em *tokens* e conversão para os objetos correspondentes. Foram implementados sobre esta todos os tipos de objetos GeoJSON, desde as primitivas geométricas às *Features* e *FeatureCollections*, de acordo com a especificação do formato GeoJSON. Dentro destes objetos é feita a validação para confirmar que o texto fornecido é um objeto válido.

No entanto, o *parser* não é estrito, ao contrário do inicialmente planeado, pois existem vários conjuntos de dados mal formados, mas que podem ser parcialmente aceites, sendo esses erros registados no *log* na eventualidade de poder ser relevante para determinar problemas causados por isso. Com este componente é possível integrar dados GeoJSON no Unity, que apenas suporta nativamente pleno texto.

5.6.2 Desafios de *floating-point precision* na integração de dados georreferenciados no Unity

A integração de dados georreferenciados requer um nível significativo de exatidão e precisão para que o modelo virtual corresponda ao mundo real e o possa simular de uma forma mais fidedigna. No entanto, isto é ligeiramente complexo de alcançar pois o Unity utiliza *single-precision floating-point* (32 bits) no seu sistema de coordenadas, o que é normalmente suficiente para a maioria das aplicações desenvolvidas neste motor de jogos, mas a uma escala semelhante ao mundo real implica que erros de arredondamento possam causar a colocação de objetos nos locais errados em relação à realidade (por exemplo, uma árvore colocada no meio da estrada em vez do passeio). Estes erros também podem ser acumulados nas operações intermédias de transformação aplicadas às coordenadas desde a sua conversão de latitude/longitude ao referencial da cena do Unity.

Para reduzir estes erros intermédios, criou-se o tipo `Vector2D` para usar nestas operações. Trata-se de um tipo que replica a interface do `Vector2`, o tipo nativo do Unity para vetores bidimensionais, mas que em vez de usar `float` na sua representação interna, utiliza *double-precision floating-points* (64 bits) para um maior nível de precisão.

Um problema adicional da utilização de aritmética de ponto flutuante, é que o nível de precisão diminui à medida que a parte inteira aumenta ao afastar-se da origem, pois são precisos mais bits da mantissa para a representar à custa de perda de precisão da parte decimal. Para mitigar este efeito é utilizada uma origem relativa para o mapa, em que o centro da área a visualizar corresponde à origem da cena no Unity e são guardadas as suas coordenadas originais para servir como *offset* do mapa.

Considere-se que temos o mapa centrado no Cais das Colunas na Praça do Comércio cujo par latitude/longitude de coordenadas é (38.70671, -9.13612) (EPSG:4326) e que se converte para o par XY (-1017028.2262262426, 4679746.946450576) (EPSG:3857), sendo que este passa a ser o valor da origem relativa, isto é, o que consideramos como sendo as coordenadas que correspondem à origem (0, 0) no Unity. Assim, a estátua de D. José I que tem como coordenadas (-1017064.9616582043, 4679863.921738781) (EPSG:3857) será colocada na cena do Unity no ponto (-36.7354319617, 116.975288205), obtido pela subtração da origem relativa às coordenadas originais.

Deste modo há uma redução considerável da ordem de grandeza da parte inteira das coordenadas e um consequente aumento da precisão da parte decimal. Isto é importante porque, embora estes cálculos sejam feitos com o `Vector2D` para aproveitar a maior precisão dada pelo `double`, ocorre uma perda da precisão extra quando os valores são finalmente aplicados aos objetos do Unity, visto que a componente que indica a sua posição utiliza `Vector2` internamente. Se os objetos estivessem nas suas coordenadas originais estariam a milhões de unidades de distância da origem em vez de apenas dezenas.

5.7 Renderers

Todas as camadas de dados contêm na sua definição um `Renderer` que lhe está associado, que é uma parte importante do sistema pois são os `Renderers` que permitem a visualização dos dados do mapa. Os `Renderers` recebem os dados de uma camada e fazem a sua conversão em algo que possa ser exibido no mapa, para permitir a interpretação visual destes por parte do utilizador.

5.7.1 Tipos de Renderers e configuração das Camadas

Aquando da inicialização de uma camada, é necessário indicar um `Renderer`, isto é, uma classe que implementa a interface `ILayerRenderer`, o que implica que é capaz de renderizar os dados da camada. Tal como as camadas, os `Renderers` podem ser divididos em dois tipos principais, `Raster` e `Vetoriais`, que têm métodos específicos para lidar com esses tipos de dados. Para permitir esse uso, as camadas `Raster` requerem que o `Renderer` implemente a interface `IRasterRenderer` e as `Vetoriais` que implemente a interface `IGeoJsonRenderer`. Para cada um dos tipos de `Renderer` existe um `Renderer` que é usado por defeito, o `DefaultRasterRenderer` e o `DefaultGeoJsonRenderer`, sendo também possível optar por outro que se adapte melhor à visualização do conjunto de dados em questão.

5.7.2 Renderização dos dados

Conforme o descrito na subsecção 5.4.2, após o carregamento dos dados do terreno relativos a uma `Tile`, é feito o carregamento das várias camadas que a compõem. É no instante após os dados terem sido obtidos que os `Renderers` efetuam o processo de os interpretar e gerar quaisquer geometria e objetos necessários para a sua visualização.

No caso de camadas `Raster`, a interface `IRasterRenderer` tem um método `Render` que recebe o objeto da `Tile` a ser renderizada e a sua textura, sendo competência do `Renderer` gerar quaisquer objetos subalternos à `Tile` que são utilizados para visualizar os dados contidos na textura. No caso do `DefaultRasterRenderer`, o seu comportamento depende da opção de visualização do terreno. Se esta estiver desativada, é apenas criado um plano quadrado, cujo tamanho ocupa a área total da `Tile`, ao qual é aplicada a textura. Se a opção estiver ativada, é gerado um objeto de terreno do `Unity` baseado nos dados de altitude, conforme explicado na secção 5.5, sendo a textura aplicada na sua superfície.

Para as camadas `Vetoriais`, a interface `IGeoJsonRenderer` tem três métodos: `RenderNode`, `RenderEdge` e `RenderArea`. Cada um destes métodos recebe o objeto da `Tile` a ser renderizada, a `Feature` à qual pertencem e as suas coordenadas. Embora existam vários tipos de geometria que uma `Feature` `GeoJSON` possa ter, tratam-se todos de composições destes blocos base. Desse modo, ao tentar renderizar uma `Feature` que, por exemplo, tem como geometria uma `GeometryCollection` composta por um `Polygon` e `LineString`, são utilizados os métodos `RenderArea` e `RenderEdge`, respetivamente, para permitir a visualização deste objeto. Dada a variedade de representações possíveis para este tipo de dados, o

`DefaultGeoJsonRenderer` apenas cria representações simples para cada uma das três primitivas, embora também respeite a visualização do terreno, aplicando uma translação vertical às coordenadas correspondente à altitude nesse ponto se a opção estiver ativada.

5.7.3 Renderização de Camadas Vetoriais

Como as `Features` presentes nas camadas vetoriais podem ter, para além da sua geometria, propriedades associadas que indicam mais informação sobre si, é importante que os seus `Renderers` também tivessem acesso a estas para criar visualizações mais relevantes.

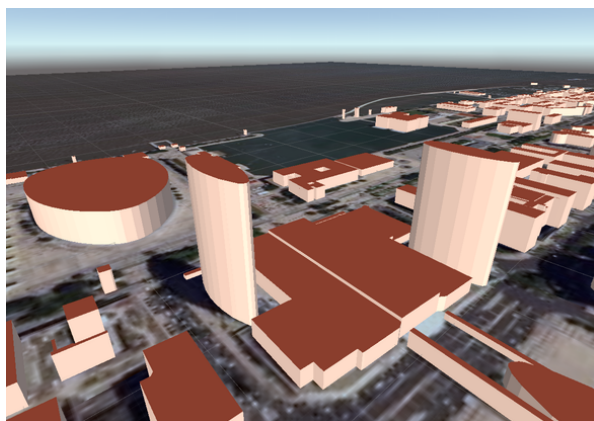


Figura 5.7: Resultado do `BuildingRenderer`.

O `BuildingRenderer` é um exemplo disso, pois faz uso de duas propriedades presentes no conjunto de dados: `EXTRUDE` e `Altitude`. No conjunto de dados `Buildings` apenas existem polígonos correspondentes à planta dos edifícios, por isso a propriedade `EXTRUDE` é usada, como o nome indica, para fazer a extrusão da planta, visto que o seu valor indica a altura do edifício. A extrusão é feita do seguinte modo: a partir das coordenadas do polígono da planta, é gerado um retângulo vertical para cada uma das arestas cuja altura é igual ao valor a extrudir, de forma a representar as paredes, e é gerado um polígono igual à planta, mas ao qual é aplicada uma translação vertical com o valor da extrusão para poder representar o telhado (Figura 5.7). Caso a opção de visualização do terreno esteja ativada, o valor da `Altitude` é adicionado às coordenadas, deslocando verticalmente o edifício para que este fique ao nível do terreno elevado.

Como o `Renderer` tem conhecimento do nível de ampliação a que corresponde a `Tile`, este pode ser utilizado para alterar a representação gerada consoante o seu valor, permitindo a criação de um sistema de nível de detalhe. No caso do `BuildingRenderer`, por exemplo, a sua geometria é simplificada quando o nível de ampliação é diminuído, sendo apenas gerados os polígonos do telhado e omitidas as paredes.

Quanto ao conjunto de dados `BuildingsLOD3`, já existe uma representação mais detalhada dos edifícios através de modelos 3D mais complexos. Para este conjunto foi optada a sua representação no formato GeoJSON apenas com `Points`, que correspondem ao centroide do modelo, e que têm uma propriedade `model` que indica qual o modelo 3D a ser

```
1 {  
2   "type": "FeatureCollection",  
3   "features": [  
4     {  
5       "type": "Feature",  
6       "id": "Baixa",  
7       "properties": {  
8         "model": "EPSG_3857_Baixa"  
9       },  
10      "geometry": {  
11        "type": "Point",  
12        "coordinates": [-9.1390461, 38.7084137]  
13      }  
14    },  
15    ...  
16  ]  
17 }
```

Listagem 5.3: Exemplo de representação de modelo com ponto GeoJSON.

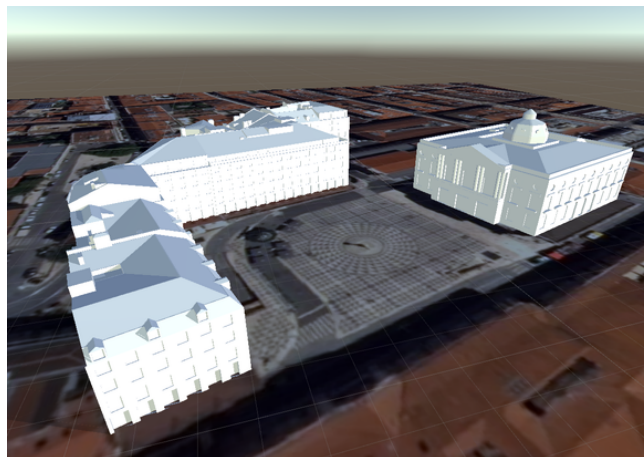


Figura 5.8: Exemplo de modelo carregado pelo PrefabRenderer.

carregado (Listagem 5.3). Como foi referido anteriormente, não foi feita a junção com o conjunto Buildings devido a uma má formação deste conjunto de dados, que não permitia separar os modelos 3D dos edifícios uns dos outros, para os poder depois corresponder com a sua planta presente no conjunto Buildings. Assim, como alternativa, foi escolhido o PrefabRenderer para representar esta camada, que carrega o modelo indicado na propriedade e instancia-o no local indicado pelo Point (Figura 5.8). Este Renderer também pode recorrer à leitura do valor da elevação do terreno (desta vez a partir do modelo digital do terreno presente na Tile) se estiver ativada a visualização do terreno.

Além do carregamento do modelo a partir das propriedades, o PrefabRenderer pode também apenas instanciar um único modelo para todos os pontos de um conjunto de dados, como por exemplo é o caso dos conjuntos Lamps e Trees (Figura 5.9), que servem como marcadores para estes objetos.

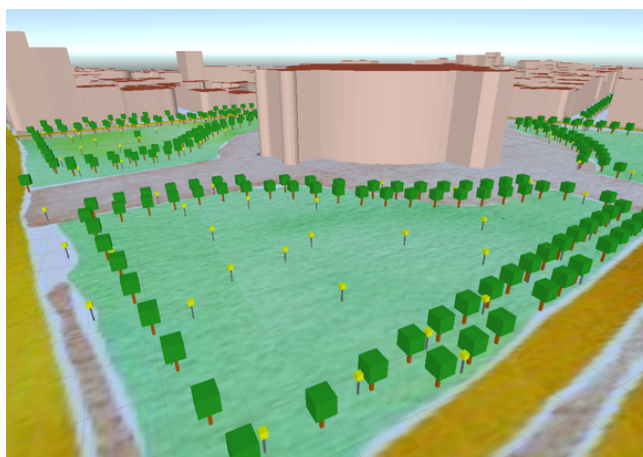


Figura 5.9: *Prefabs* instanciadas para marcar Lamps e Trees.

5.7.4 Adaptação e extensão de Renderers aos conjuntos de dados

Como diferentes conjuntos de dados requerem diferentes tipos de visualizações, é preciso que o sistema de Renderers seja flexível. Contribuem para isso a possibilidade de estes poderem obter a escala em que a *Tile* se encontra, permitindo a existência no mesmo *Renderer* de múltiplas estratégias de visualização. Por exemplo, um conjunto com muitos pontos poderia utilizar um modelo 3D detalhado se na escala atual apenas fossem visíveis uma dúzia destes e utilizar apenas um plano com a imagem de um marcador se a escala implicasse a representação de uma área com um vasto número de objetos.

O acesso às propriedades das *Features* também permite que as representações dos objetos possam ser mais dinâmicas, como já foi descrito. Por exemplo, um conjunto de dados de sinais de trânsito poderia utilizar um modelo adequado ao seu tipo de suporte, com uma textura representativa do sinal indicado e alinhar com a sua orientação, se essas propriedades estiverem disponíveis. Outro exemplo seria uma estação GIRA que poderia ter o seu modelo base com as docas e, consoante a propriedade da sua ocupação, mostrava também modelos de bicicletas nas docas ocupadas, podendo em alternativa usar apenas um ícone com a percentagem de preenchimento se a escala fosse reduzida.

Para poder criar este tipo de *Renderers* basta apenas que implementem as interfaces *IRasterRenderer* ou *IGeoJsonRenderer* e que sejam indicados como o *Renderer* a utilizar aquando da inicialização da camada. Se forem também criados outros tipos de camadas, terá então de ser criada uma interface similar a estas duas que, tal como elas, implemente *ILayerRenderer*.

5.8 Filtros

A filtragem de conjuntos de dados é uma ferramenta crucial para a análise destes, sendo relevante a discussão do seu funcionamento. Nesta secção começa-se por definir o que são as Camadas Filtráveis e as suas propriedades. De seguida, é explicado como são configurados e aplicados os filtros desde a interação do utilizador com a interface, o

processo de aplicação dos filtros até os seus efeitos serem visíveis. Por fim, é descrito o processo de extensão dos filtros, caso se pretenda criar outros tipos de filtros ou de propriedades de *Features*.

5.8.1 Camadas Filtráveis e suas propriedades

Ao contrário das camadas *Raster*, em que cada *Tile* só contém a imagem relativa à sua área, nas camadas Vetoriais (como é o caso das camadas GeoJSON) estão contidos múltiplos objetos que são denominados de *Features*. Independentemente da sua geometria, as *Features* podem ter associadas a si um conjunto de propriedades que servem para descrever com mais detalhe o objeto. Visto que estas propriedades são relevantes para compreender um certo conjunto de dados, torna-se necessária a existência de métodos que permitam efetuar uma análise dos dados a partir delas.

Estas camadas pertencem então a um subconjunto que são as Camadas Filtráveis. As Camadas Filtráveis são uma extensão das Camadas que adiciona a capacidade de conter um conjunto de propriedades que lhe são associadas e um booleano que indica se estão a ser aplicados os filtros com base nessas mesmas propriedades. Mais concretamente, existe a interface *IFilterableLayer* que estende a interface base *ILayer* que as Camadas Filtráveis têm de implementar, como é o caso das Camadas Vetoriais GeoJSON em que a *GeoJsonLayer* implementa a interface *IFilterableLayer* que por sua vez estende *ILayer*. Existe também a interface *IFilterableTileLayer* que estende a interface base *ITileLayer*, que adiciona um método para aplicar os filtros das propriedades às *Features* presentes numa dada *Tile*.

Quanto às propriedades, estas implementam a interface *IFeatureProperty*, que indica um identificador para a propriedade, um texto com que é mostrada ao utilizador final, um booleano que indica se o seu filtro está ativo e um método para verificar se uma dada *Feature* corresponde ao filtro. Existem cinco tipos de implementações de propriedades:

- *BooleanFeatureProperty* - para variáveis representadas por um booleano;
- *NumberFeatureProperty* - para variáveis representadas por um número;
- *StringFeatureProperty* - para variáveis representadas por texto;
- *CategoryFeatureProperty* - para variáveis categóricas;
- *DateFeatureProperty* - para variáveis temporais.

Estes tipos foram selecionados para serem implementados por serem os mais comuns que se encontravam nos conjuntos de dados utilizados, no entanto é possível criar outros tipos se os dados a utilizar assim o necessitarem, como é descrito na subsecção 5.8.3. A especificação do formato GeoJSON [14] indica que o membro das propriedades pode ser um valor nulo (caso não existam) ou um objeto JSON, o que implica que num conjunto GeoJSON as propriedades, para além dos tipos simples do JSON como *Number* ou *String*,

podem ser Array ou Object, sendo possível criar uma cadeia de objetos criança, tendo-se optado por simplificar os conjuntos de dados que utilizassem esses tipos na fase de pré-processamento dos dados, para evitar problemas no desenvolvimento da interface para os filtros dessas propriedades.

Cada uma destas implementações de `FeatureProperties` contém dois campos que são o `FilterValue` e `Filter`, que são utilizados para a filtragem. O `FilterValue` é o valor a filtrar ao qual se pretende comparar o valor que cada `Feature` tenha, enquanto o `Filter` é o tipo de operação a aplicar aquando dessa comparação (por exemplo, verificar se o valor da propriedade é menor que o do filtro ou se contém um certo texto).

5.8.2 Configurar e aplicar os filtros

Existem dois passos principais que um utilizador tem de efetuar para filtrar os dados de uma dada camada, a configuração dos filtros e a sua aplicação.



Figura 5.10: Configuração do filtro de uma propriedade.

Ao abrir a janela das camadas e selecionar a camada pretendida, é aberta a vista de detalhe da camada (Figura 5.10(a)) que mostra alguns metadados da camada e uma listagem das suas propriedades, se as tiver. Como já foi mencionado, cada uma dessas propriedades pode ser utilizada para filtrar o conjunto de dados da camada. Para isso, começa-se por selecionar o tipo de filtro a aplicar no menu *dropdown* (Figura 5.10(b)), indica-se o valor a utilizar na filtragem (se for aplicável ao tipo de filtro) e que se pretende usar o filtro associado à propriedade (Figura 5.10(c)). No exemplo dado pelas imagens, pretende-se usar o filtro associado a uma propriedade da camada dos Edifícios, para

apenas mostrar aqueles cuja altura seja maior que 42 metros. É neste passo que os campos `Filter` e `FilterValue` das propriedades são definidos.

Configurados os filtros, o utilizador pode fechar a janela das camadas e utilizar o botão presente na barra de ferramentas para alternar entre a vista normal (Figura 5.11(a)) ou a vista à qual os filtros estão aplicados (Figura 5.11(b)). Ao clicar para ligar os filtros, todas as camadas passam a ter ativos quaisquer filtros que tivessem sido configurados para ser utilizados, sendo todas as suas `TileLayers` percorridas para que sejam aplicados. Adicionalmente, se houver uma alteração da área de visualização que implique o carregamento de novas `Tiles`, estas serão logo filtradas se assim estiver definido ao nível da camada.

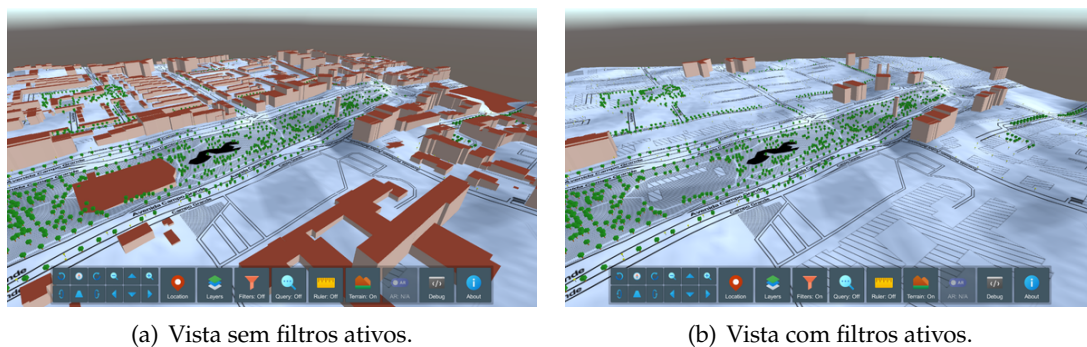


Figura 5.11: Efeito da ativação dos filtros.

Ao nível da `TileLayer`, são percorridas as `Features` que nela estejam contidas e, para cada propriedade cujo filtro está a ser utilizado, é testado o método que determina se a `Feature` é suposta ser filtrada. Utilizando o exemplo das imagens e supondo um edifício com a altura de 25 metros, como o valor configurado foi 42 metros e o filtro configurado foi que tem de ser maior que o valor, esse edifício não corresponde ao filtro e falha o teste. Todas as `Features` que falhem o teste não aparecem na vista filtrada, ficando apenas aquelas cujas propriedades cumpram as condições impostas pelos filtros.

5.8.3 Estendendo filtros: Criação de novos tipos de filtro ou propriedades

Ao longo do projeto foi feito um esforço para que, sempre que possível, os sistemas implementados fossem desenvolvidos por forma a permitir uma futura expansão das suas capacidades, permitindo deste modo adaptarem-se a diferentes casos de uso. O sistema de filtragem é um desses exemplos, ao permitir uma implementação relativamente simples de novos tipos de filtros para uma dada propriedade ou até a definição de outro tipo de propriedade.

Adicionar um tipo de filtro é o mais fácil dos dois, pois só é preciso modificar o enumerador `FilterOperator` e o método `MatchesFilter` da propriedade. Ao `FilterOperator` adiciona-se o identificador do filtro como membro do enumerador. No método `MatchesFilter`, adiciona-se mais um caso à estrutura `switch` para o novo tipo de filtro, sendo o seu corpo uma função booleana. Esta função recebe o valor da propriedade da `Feature`, que até pode

ou não ser um valor nulo, e tem de retornar `true` se, ao aplicar a operação do filtro, o valor corresponder ao definido para o filtro ou `false` caso contrário. Utilizando o operador `GreaterThan` da propriedade numérica como exemplo, este retorna `false` se o valor for nulo ou, caso contrário, retorna o resultado da operação `Value > FilterValue`.

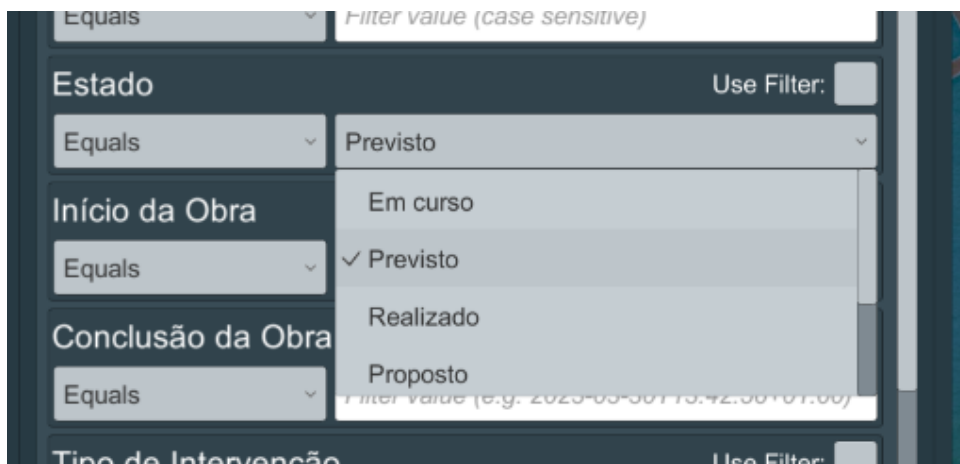


Figura 5.12: *Dropdown* para seleção do valor a filtrar de variável categórica.

Quanto à criação de um novo tipo de propriedade, primariamente trata-se de criar um struct que implemente a interface `IFeatureProperty` e que contenha os campos `T FilterValue` e `FilterOperator Filter`, definir o enumerador para os seus tipos de filtros e modificar a interface do detalhe da camada para suportar a definição de filtros para este tipo de propriedade. Um exemplo que surgiu durante este projeto foram as propriedades categóricas, que são uma espécie de subconjunto das propriedades de texto, com a diferença que existe um número limitado de opções que o seu valor pode ser. Este tipo de propriedade tem um campo extra que define as suas possíveis categorias e na interface que mostra a propriedade, é utilizado um segundo menu *dropdown* que restringe as opções que podem ser usadas para definir o valor do filtro (Figura 5.12).

5.9 Navegação

A navegação no mapa, que inclui a **translação**, mudança de **escala**, **rotação** e **inclinação**, engloba um conjunto de funcionalidades que permitem alterar a área do mapa a ser visualizada. Estes quatro mecanismos são os métodos de interação mais básicos que permitem a navegação do utilizador dentro do mapa. Enquanto a rotação e a inclinação apenas ajustam a posição da câmara em relação ao mapa, a translação e a mudança de escala implicam o carregamento de novas *tiles* que correspondam à nova área a ser visualizada conforme explicado na subsecção 5.4.1.

No entanto, estes métodos servem principalmente para explorar áreas próximas da atual, podendo ser mais morosos quando o utilizador pretende visualizar uma localização mais distante. Como alternativa, o utilizador pode inserir as coordenadas de **latitude** e **longitude** do local desejado para visitar, sendo a posição do mapa atualizada para que

o seu centro corresponda às coordenadas fornecidas e seja feito o carregamento das *tiles* necessárias.

Para os casos em que o utilizador não conheça as coordenadas do local, existem duas funcionalidades adicionais disponíveis, como a lista de **pontos de interesse** e a **pesquisa de local**. A lista de pontos de interesse, como o nome indica, apresenta uma coleção de locais que podem ser do interesse do utilizador, tendo sido previamente definidos na aplicação. Dessa forma, o utilizador pode simplesmente selecionar o ponto mais relevante da lista para mover o mapa para lá, estando no entanto limitado ao conjunto de pontos definidos.

Quanto à pesquisa de local, permite aos utilizadores inserir termos de pesquisa e obter as coordenadas do local mais relevante associado aos termos fornecidos. Esta funcionalidade utiliza a API de pesquisa do Nominatim, que utiliza os dados do OpenStreetMap para determinar as coordenadas com base num endereço ou nome de local, o chamado *geocoding*.

Por exemplo, quando o utilizador procura por "Campus de Campolide", é feito um pedido GET para o endereço <https://nominatim.openstreetmap.org/search?format=json&countrycodes=pt&q=Campus%20de%20Campolide>. A partir da sua resposta (Listagem 5.4), são obtidos o endereço completo do local que substitui o texto inserido no campo de pesquisa e as coordenadas do seu centroide (38.733047, -9.160744) e o mapa navega automaticamente para esse local.

```

1  [{
2    ...
3    "lat": "38.73304745",
4    "lon": "-9.160744712787931",
5    ...
6    "name": "NOVA University Lisbon - Campolide Campus",
7    "display_name": "NOVA University Lisbon - Campolide Campus, Avenida Miguel Torga,
8    ↵ Campolide, Lisbon, 1070-373, Portugal",
9    "boundingbox": [
10     "38.7312743",
11     "38.7346160",
12     "-9.1624869",
13     "-9.1590545"
14   ]
15 }]
```

Listagem 5.4: Extrato de resposta de pedido de *geocoding*.

5.10 Consulta de dados

A consulta de dados é uma funcionalidade que permite obter informações adicionais sobre um determinado ponto no mapa. Para qualquer ponto selecionado, as coordenadas da cena do Unity são convertidas no par de coordenadas latitude/longitude correspondente, sendo este mostrado ao utilizador, assim como a altitude correspondente com base nos

dados do modelo digital do terreno e o endereço ou nome do local usando a API de *reverse geocoding* do Nominatim. Esta API procura o objeto mais relevante e próximo das coordenadas fornecidas nos dados do OpenStreetMap e sintetiza um endereço com base nos seus metadados.

Por exemplo, se clicar no ponto que corresponde ao par de coordenadas (38.713751, -9.137952), é feito um pedido GET para o endereço <https://nominatim.openstreetmap.org/reverse?format=json&lat=38.713751&lon=-9.137952&addressdetails=0>. A partir da sua resposta (Listagem 5.5), é obtido o nome e endereço completo correspondente a esse ponto, sendo então indicado ao utilizador. Caso não seja possível determinar um endereço, é mostrado um aviso com essa mensagem.

```
1  {  
2    ...  
3    "class": "highway",  
4    "type": "pedestrian",  
5    ...  
6    "addresstype": "road",  
7    "name": "Praça da Figueira",  
8    "display_name": "Praça da Figueira, São Cristóvão e São Lourenço, Santa Maria Maior,  
    ↳ Lisbon, Portugal",  
9    ...  
10 }
```

Listagem 5.5: Extrato de resposta de pedido de *reverse geocoding*.

Adicionalmente, se no ponto indicado pelo utilizador esteja um objeto relacionado a uma camada vetorial (GeoJSON), é indicado o seu identificador, a camada à qual pertence e a *tile* correspondente. Caso existam propriedades associadas a esse objeto, elas também são listadas.

5.11 Rota

A funcionalidade da Rota permite visualizar um trajeto entre dois pontos no mapa. Quando é ativada, um painel é exibido onde é possível inserir os pontos de origem e destino, sendo a pesquisa do percurso efetuada quando o botão para encontrar é pressionado.

O painel tem também um componente de texto, que indica o perfil de navegação, a distância e a duração do percurso. Ao mesmo tempo, é sobreposto sobre o mapa um marcador dos pontos de partida (verde) e chegada (vermelho), assim como uma linha e nós que representam o trajeto e manobras a efetuar ao longo do percurso. Caso um dos pontos não tenha sido encontrado ou não seja possível obter um percurso entre estes, o mesmo é indicado no painel.

Todo este processo começa pela obtenção das coordenadas dos pontos de partida e chegada que são obtidas através de *geocoding*. Isto é feito recorrendo à API do Nominatim para obter as coordenadas de ambos, à semelhança do campo de pesquisa da localização

como descrito na secção 5.9. De seguida, essas coordenadas são então utilizadas para fazer um pedido à API da OSRM, que tentará então obter um percurso possível entre os dois pontos.

```

1  {
2    "routes": [
3      {
4        "legs": [
5          {
6            "steps": [
7              { "maneuver": { "location": [38.739981, -9.137164], ... }, ... },
8              ...
9              { "maneuver": { "location": [38.727751, -9.155088], ... }, ... }
10             ],
11            "duration": 452.3,
12            "distance": 1676.9,
13            ...
14          }
15        ],
16      }
17    ],
18    ...
19  }

```

Listagem 5.6: Extrato de resposta de pedido de rota.

Por exemplo, se tentar planear a rota entre a Praça de Londres e o Parque Eduardo VII, são primeiro obtidas as suas coordenadas, (38.740846, -9.137565) e (38.728621, -9.153409), através da API do Nominatim. Já com as coordenadas obtidas, é feito um pedido GET para o endereço <https://router.project-osrm.org/route/v1/driving/38.740846,-9.137565;38.728621,-9.153409?steps=true>. A partir da sua resposta (Listagem 5.6), são obtidas a distância (m) e duração (s) a converter e mostrar ao utilizador no painel, e a localização das manobras a efetuar para criar os nós da linha que representará o trajeto (para além dos marcadores dos pontos de partida e chegada).

5.12 Régua

A Régua é um outro componente de análise que permite determinar distâncias no mapa. Ao ser ativada, um painel indica a totalidade da distância por ela medida e fornece a opção de limpar a medição atual. É representada por um conjunto de nós definidos pelo utilizador, que são ligados entre si em linha reta.

É possível adicionar novos nós no fim da régua ao clicar num ponto livre do mapa ou um nó intermédio entre dois já existentes ao clicar em cima da aresta que os liga. Ao arrastar um nó existente pode-se atualizar a sua posição. Se tentar clicar e arrastar um ponto pertencente a uma aresta, é criado um nó intermédio nesse local que assumirá a nova posição enquanto continuar a arrastar. Por fim, também se pode clicar num nó para o remover e assim diminuir a medição.

Essencialmente, a régua trata-se de uma lista duplamente ligada de nós, em que cada nó contém as suas próprias coordenadas e a distância medida pelo segmento de reta que o liga ao nó anterior (naturalmente, no caso do primeiro nó é sempre 0). A distância é calculada no momento de criação do nó e atualizada quando o nó é movido ou o seu predecessor imediato é movido, isto é, para os nós intermédios a sua medição e do nó seguinte são atualizadas, e no caso do último nó apenas a sua medição é atualizada. O valor total é controlado ao nível da régua, aumentando com cada novo nó, diminuindo com cada remoção e, quando um é movido, é subtraída as medições antigas dos segmentos a alterar e de seguida adicionada as suas novas medições.

Os cálculos das distâncias de cada segmento de reta que compõe a régua são feitos através da fórmula de haversine [17], frequentemente utilizada para calcular as distâncias entre pontos na superfície de uma esfera, tendo em conta a sua curvatura, sendo assim uma medição geodésica e não planar. Embora a Terra não seja uma esfera perfeita, trata-se de uma aproximação suficiente para este contexto. A fórmula de haversine é expressa da seguinte maneira:

$$a = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat_1) \cdot \cos(lat_2) \cdot \sin^2\left(\frac{\Delta lon}{2}\right)$$

em que Δlat é a diferença de latitude entre os dois pontos, Δlon é a diferença de longitude entre os dois pontos, e lat_1 e lat_2 são as latitudes dos dois pontos.

A distância d entre os dois pontos é então calculada como:

$$d = 2 \cdot R \cdot \arcsin(\sqrt{a})$$

onde R é o raio médio da Terra, que é aproximadamente 6371km.

No entanto, como os cálculos são feitos em radianos, é preciso primeiro converter a latitude/longitude de graus para radianos, multiplicando o seu valor por $\frac{\pi}{180}$.

5.13 Modo de Realidade Aumentada

Tal como o seu nome indica, o modo de Realidade Aumentada permite visualizar o mapa através de realidade aumentada. Quando este modo é ativado, a vista do mapa é substituída pela *stream* de vídeo que é captado pela câmara traseira do dispositivo móvel. A partir desse instante, a aplicação tenta encontrar nas imagens capturadas o marcador de referência (o Apêndice A contém o marcador utilizado), sobre o qual o mapa é depois posicionado. Trata-se então de Realidade Aumentada cujo método de visualização é *video see-through* e técnica de *tracking* é visão computacional com marcadores.

Como o mapa mantém a sua posição e rotação constantes relativamente ao marcador, tanto os controlos *touch* e os botões para alterar a direção e inclinação do mapa são desativados, visto que o mesmo efeito pode ser obtido, de forma potencialmente mais intuitiva, ao deslocar o dispositivo móvel no próprio espaço físico ou até movendo e/ou rodando fisicamente a folha que contém o marcador.

O funcionamento deste modo é feito com recurso à *framework* AR Foundation do Unity, que é criada especificamente para a produção de programas de realidade aumentada multiplataforma [88], sendo também um dos motivos pela opção por este motor de jogos para o desenvolvimento do protótipo. O AR Foundation funciona como uma camada de abstração, o que permite suportar diversos tipos de dispositivos através de subsistemas que implementam as suas APIs nativas, como o Android através do ARCore, iOS através do ARKit, HoloLens ou Magic Leap. Como o foco do desenvolvimento foi para Windows e Android, o modo de RA apenas se encontra disponível para Android, ao qual o AR Foundation recorre por sua vez ao ARCore para fazer a ligação nativa com o dispositivo.

A aplicação ao ser iniciada, antes da inicialização do mapa e da interface, verifica qual o sistema operativo em que se encontra. Caso se trate de um computador Windows, o botão para ativar o modo encontra-se desativado e com a sua *label* a indicar a indisponibilidade. Se for um dispositivo Android, verifica se este suporta o ARCore e se tem instalado o Google Play Services for AR que permite a sua utilização, tentando instalar se possível e, caso não o seja, também fica com o botão desativado como no Windows.

Se o dispositivo Android suportar o ARCore e tiver instalado o Google Play Services for AR, o utilizador pode então ativar o modo de realidade aumentada através do botão para esse efeito. Como foi dito, a *stream* de vídeo da câmara é mostrada ao utilizador que deve tentar enquadrar o marcador de forma a permitir a sua deteção.

Quando o ARCore deteta o marcador, faz uma estimação da sua pose, isto é da sua posição e rotação relativamente à câmara e comunica-a ao AR Foundation. Por sua vez, este cria um objeto chamado ImageTracker na cena do Unity que é constantemente posicionado e alinhado de tal forma a corresponder à pose do marcador.

Por fim, o objeto que representa o mapa é definido como sendo um objeto filho do ImageTracker na hierarquia da cena, assumindo a mesma posição e rotação, e deixa de estar escondido. Assim, é criado um efeito de sobreposição do mapa no marcador físico, assemelhando-se a uma maquete.

Quando o modo é desativado, é desativado o *tracking* do marcador, a *stream* de vídeo é desligada, o mapa deixa de ser um objeto filho do ImageTracker (sendo que este é removido após a desativação do *tracking*) e volta a ser posicionado na origem da cena do Unity.

Embora o modo de realidade aumentada neste protótipo tenha sido um componente mais exploratório, visto que o foco primário era na integração dos dados georreferenciados, optou-se pela utilização de um marcador de referência em vez da deteção do plano da mesa em que este se encontrava. A ideia é que, em trabalho futuro, o modo de realidade aumentada seja usado numa maneira colaborativa em que múltiplas pessoas que estejam presentes na mesma sala possam estar a ver o mesmo modelo no mesmo local. Para esse efeito, a utilização de um marcador comum permitiria que a deteção por dispositivos diferentes indicasse a mesma posição para colocar o mapa. Caso se tivesse optado pela deteção de planos, obrigaria que todos os utilizadores tivessem de primeiro detetar completamente a superfície da mesa para evitar discrepâncias, o que poderia ser mais

restritivo. No entanto, poderia também ser algo que devesse ser testado, comparando a utilização dos dois tipos de *tracking* e a preferência dos utilizadores, tal como a eficiência na sua utilização.

5.14 Resumo

Neste capítulo, foi explorada em detalhe a implementação do sistema, fornecendo uma visão geral da sua estrutura e dos vários componentes. Começou-se por examinar a estrutura geral do sistema, estabelecendo as bases para as secções subsequentes. Depois, foram explorados os servidores responsáveis pela disponibilização de *tiles raster* e vetoriais. Foi também discutido o pré-processamento dos dados, sendo descritas as diferentes técnicas de conversão e normalização de dados utilizadas. Em seguida, o sistema de carregamento de *tiles* foi introduzido, que desempenha um papel fundamental na gestão e renderização das *tiles*, sendo descritos os ciclos de vida das *Tiles* e *TileLayers*. Focou-se na modelação do terreno a partir das *tiles Terrain-RGB* da Mapbox, abordando também os desafios associados à representação do terreno baseada em *tiles*, como os espaços nas fronteiras e dependências entre *tiles*. O capítulo prosseguiu com uma exploração do Parser de GeoJSON para interpretar camadas vetoriais e os desafios de *floating-point precision* na integração de dados georreferenciados no Unity, seguidos pelos *renderers* (sendo discutidos os diferentes tipos existentes, a configuração de camadas, o processo de renderização para dados *raster* e vetoriais e, além disso, a possível adaptação e extensão dos *renderers* para acomodar diversos conjuntos de dados). Após isto, aprofundou-se o tema dos filtros, o conceito de camadas filtráveis e as suas propriedades, o processo de configuração e aplicação de filtros, e também foram discutidas as possibilidades de estender os filtros através da criação de outros tipos ou propriedades. Por fim, foram detalhados os componentes da Navegação, Consulta de Dados, Rota, Régua e o Modo de Realidade Aumentada.

AVALIAÇÃO

Neste capítulo é feita uma análise abrangente da aplicação, com três fases de avaliação. Começa com uma avaliação crítica da extensibilidade da aplicação, uma breve discussão da capacidade para adaptar a aplicação a requisitos futuros e possíveis alterações. De seguida, é feita a avaliação de desempenho, dividida em dois grupos distintos relativos ao carregamento de uma área e ao desempenho da aplicação durante o movimento no mapa. Por fim, a fase final de avaliação consiste nos testes de usabilidade com utilizadores, sendo apresentada a metodologia utilizada nas sessões de teste, os resultados obtidos nesta, assim como potenciais limitações desta avaliação.

6.1 Avaliação crítica da extensibilidade da aplicação

Nesta secção, é feita uma breve discussão da versatilidade e extensibilidade da aplicação e das suas capacidades, abordando questões como a sua adaptabilidade a outros contextos, compatibilidade com outros conjuntos de dados, implementação e extensão de *renderers*, filtros ou propriedades, assim como o suporte para formatos de dados adicionais.

1. É possível utilizar a aplicação noutras cidades ou contextos não urbanos?

A aplicação pode ser utilizada noutras cidades ou até em contextos não urbanos, estando primariamente limitada pela área de cobertura dos conjuntos de dados. Adicionalmente, a lista de pontos de interesse utilizada no componente de navegação pode ser modificada, sendo indicado o par de coordenadas e o nome do ponto a mostrar ao utilizador. De resto, a aplicação é agnóstica quanto ao contexto de utilização, podendo visualizar ou analisar dados de qualquer parte do mundo.

2. É possível utilizar outros conjuntos de dados na aplicação?

A aplicação no seu estado atual encontra-se totalmente capaz de utilizar outros conjuntos de dados diferentes dos que já estão presentes nesta. Para tal, basta declarar a inicialização da camada que lhe irá corresponder juntamente com as restantes na inicialização do mapa. Isto consiste em indicar os metadados associados à camada como o identificador a usar, nome a mostrar aos utilizadores, data de

atualização dos dados, descrição, assim como o URL do servidor que fornece os dados. Tanto para as camadas *raster*, como para as vetoriais, existem *renderers* simples que podem ser usados para a sua representação sem qualquer trabalho adicional, tendo apenas de declarar a sua utilização. No caso específico das camadas vetoriais, podem também ser indicadas as suas propriedades de modo que seja possível filtrar os seus atributos.

3. É possível utilizar outros *renderers* para visualizar os dados?

Como é expectável, diferentes conjuntos de dados requerem diferentes tipos de visualizações. Para esse efeito, é possível criar *renderers* que se adaptem melhor às visualizações pretendidas, desde que implementem as interfaces para o efeito conforme mencionado na subsecção 5.7.4. Deste modo, podem aceder aos dados que são recebidos do servidor e transformar na representação pretendida, especialmente no caso das camadas vetoriais, que podem utilizar os valores dos atributos associados a cada objeto presente no conjunto para determinar características da sua representação (como o modelo a carregar, a sua altura, texturas, entre outros).

4. É possível implementar outros filtros ou propriedades?

Embora já exista um conjunto diverso de tipos de propriedades e filtros destas suportados pela aplicação no seu estado atual, poderá ser interessante para um dado contexto suportar outros tipos. Assim, e à semelhança dos *renderers*, é possível criar outros tipos de filtros e de propriedades, bastando para isso seguir o processo descrito na subsecção 5.8.3. Um exemplo de um tipo de propriedade que surgiu durante o desenvolvimento deste projeto foram as propriedades categóricas, que são uma espécie de subconjunto das propriedades de texto, com a diferença que existe um número limitado de opções que o seu valor pode ser.

5. É possível suportar outros formatos de dados?

Dada a variedade de formatos de dados georreferenciados, é possível que se pretenda que a aplicação suporte outros formatos de dados. De todos os tópicos discutidos nesta secção, este é o que implica um maior trabalho. Será necessário dividir os conjuntos de dados presentes nesse formato de forma a usar o padrão de *tiles XYZ*, implementar um *parser* que possa interpretar os dados recebidos no pedido de uma *tile* e, por fim, um novo tipo de camada que implemente a interface *ILayer*. Caso se trate de um conjunto de dados vetoriais, poderá ser possível implementar a interface *IFilterableLayer*, que por sua vez estende *ILayer*, usada nas camadas GeoJSON para suportar as capacidades de filtragem baseadas nos atributos dos objetos.

6.2 Avaliação de desempenho

Visto que este trabalho envolveu o desenvolvimento de um protótipo funcional de uma aplicação, é possível testar a mesma de forma a medir o seu desempenho de forma

quantitativa e objetiva.

Esses testes foram divididos nos **Grupos L** (Load), relativo ao carregamento de uma área, e **M** (Movement), relativo ao desempenho durante o movimento no mapa.

Os testes foram efetuados num computador portátil Lenovo 81SX que tinha simultaneamente aberto o programa `geojson-tile-server` para servir as *tiles* vetoriais e a aplicação a ser testada. Esta máquina tem as seguintes especificações:

- CPU: Intel Core i7-9750H @ 2.60GHz
- GPU: NVIDIA GeForce RTX 2060 (Mobile)
- RAM: 16 GB
- Sistema Operativo: Windows 10 Education (22H2)

6.2.1 Grupo L - Carregamento de área

O objetivo principal deste grupo de testes é avaliar o carregamento de uma dada área com diferentes condições. Todos os testes deste grupo envolveram o carregamento de uma área centrada no Largo do Carmo com o nível de ampliação 18.

Foram definidas as seguintes métricas a medir durante os testes deste grupo:

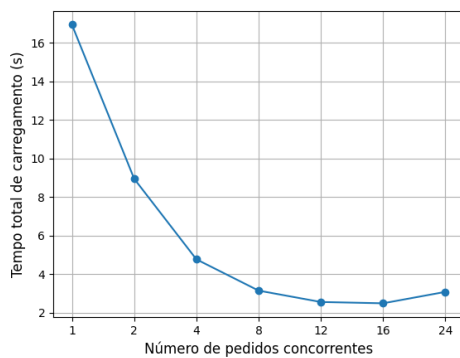
- **Tempo para carregar completamente a área:** O tempo que demora a carregar completamente a área desde que esta é pedida até à última *tile* que lhe pertença terminar de carregar.
- **First Contentful Paint:** O tempo que demora para qualquer conteúdo aparecer no ecrã desde o início do pedido de carregamento.
- **Tempo de pedidos do terreno:** O tempo gasto por cada *tile* à espera do pedido de dados do modelo digital do terreno.
- **Tempo de processamento do terreno:** O tempo que a aplicação, por cada *tile*, demora a processar os dados obtidos nos pedidos do terreno.
- **Tempo de pedidos das camadas:** O tempo gasto à espera do pedido de dados por cada camada das *tiles*.
- **Tempo de processamento das camadas:** O tempo que a aplicação demora a processar os dados obtidos nos pedidos das camadas (apenas se aplica às camadas vetoriais que têm de fazer *parse* do GeoJSON) de cada *tile*.
- **Tempo de renderização das camadas:** O tempo que a aplicação demora a gerar a visualização dos dados por cada camada das *tiles*.
- **Memória reservada:** A quantidade total de memória reservada pelo Unity para a aplicação.

- **Memória alocada:** A quantidade de memória alocada pelo Unity para a aplicação.
- **Memória reservada não utilizada:** A quantidade de memória reservada pelo Unity para a aplicação que não está alocada.

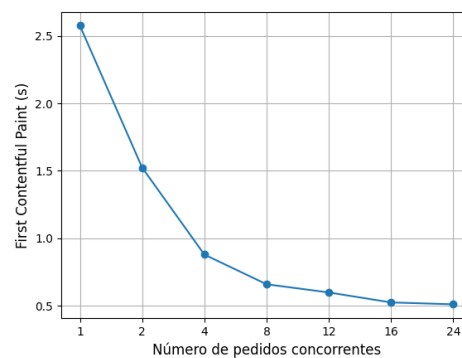
Teste L1 | Impacto do número de pedidos concorrentes

O objetivo deste teste é o de avaliar a influência do número de pedidos concorrentes (que as *tiles* podem fazer para obter dados) no tempo de carregamento da área. Essencialmente, pretende-se testar se a paralelização desta tarefa é eficaz.

- **Parâmetro a variar:** Número de pedidos concorrentes (1, 2, 4, 8, 12, 16 e 24 pedidos)
- **Parâmetro constante:** Número de *tiles* a carregar (área de 9x9 *tiles*, num total de 81)



(a) Tempo de carregamento total.



(b) First Contentful Paint.

Figura 6.1: Tempos de carregamento em função do número de pedidos concorrentes.

Na Figura 6.1 estão presentes os tempos de carregamento em função do número de pedidos concorrentes, tratando-se de tempos médios resultantes de três tentativas diferentes.

No caso do Tempo de carregamento total (Figura 6.1(a)), é possível observar que o tempo diminui à medida que o número de pedidos concorrentes aumenta, até estes atingirem 12 pedidos, passando então de cerca de 17s para 2.5s. Nesse ponto, mantém-se estável até aos 16 pedidos, a partir dos quais começa a aumentar ligeiramente para valores um pouco acima de 3s com 24 pedidos.

Este resultado está dentro do expectável para a paralelização efetuada, tendo em conta que o tempo total diminui aproximadamente para metade do seu valor à medida que é duplicado o número de pedidos concorrentes. Como cada uma das *tiles* têm de ficar a aguardar pela resposta aos seus pedidos de dados, é possível aproveitar esse período de inatividade ao processar e renderizar as *tiles* cujos dados já tenham sido recebidos.

Uma causa provável para o ligeiro aumento dos tempos a partir dos 16 pedidos será o facto do processador ter 12 *threads* (6-core). Isto implica que a partir desse ponto o

overhead relacionado com a mudança de contexto das tarefas poderá ter um impacto mais significativo, visto que estas superam o número de recursos físicos existentes.

Quanto à *First Contentful Paint* (Figura 6.1(b)), também é possível observar uma curva semelhante à do tempo de carregamento total, embora as melhorias sejam menos acentuadas. Como a *First Contentful Paint* representa o tempo que demora até surgir qualquer conteúdo no ecrã, trata-se do tempo é gasto pela primeira *tile* que conclua o processo de pedir e processar o terreno, assim como pedir, processar e renderizar uma das suas camadas. Como o pedido do terreno é a primeira etapa ser efetuada, a paralelização não tem um impacto significativo neste ponto, visto não haver outras tarefas que podem ser executadas enquanto se aguarda a resposta sem ser pedidos de terreno de outras *tiles*, o que ultrapassaria os recursos disponíveis. Deste modo, acaba por haver um limite mínimo que diminui as melhorias alcançáveis, mas mesmo assim estas existem ao descer de cerca de 2.6s com 1 pedido para cerca de 0.6s com 12 pedidos, sendo uma redução de 2 segundos.

Embora se trate de uma medição tipicamente utilizada no contexto de aplicações *web* (sendo que para essas é considerado bom se o seu valor for inferior a 1.8s [30]), achou-se que era relevante a sua observação pois serve como *feedback* ao utilizador de que algo está a acontecer e que o mapa está efetivamente a ser carregado.

Teste L2 | Impacto do número de *tiles*

O objetivo deste teste é o de avaliar a influência que o número de *tiles* e, por sua vez, o tamanho da área a carregar tem na memória utilizada.

- **Parâmetro a variar:** Número de *tiles* a carregar (áreas de 3x3, 5x5, 7x7, 9x9 e 11x11 *tiles*, num total de 9, 25, 49, 81 e 121 respetivamente)
- **Parâmetro constante:** Número de pedidos concorrentes (12 pedidos)

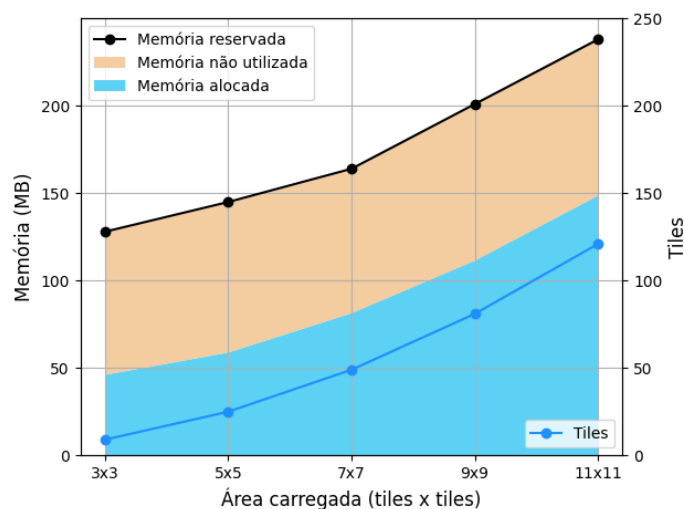


Figura 6.2: Utilização de memória em função da área carregada.

Na Figura 6.2 está presente a utilização de memória pela aplicação em função do tamanho da área carregada, dada pelo número de *tiles*, tratando-se dos valores médios resultantes de três tentativas diferentes.

A quantidade de memória alocada segue uma curva cujo aumento aparenta corresponder em proporção semelhante ao aumento da área carregada, o que era o expectável ao se tratar do único parâmetro a variar. Como a quantidade de memória relacionada com outras partes da aplicação será relativamente a mesma independentemente do tamanho da área a carregar, ao variar o tamanho desta era expectável que a memória alocada aumentasse na mesma proporção. Ao haver mais *tiles* que precisam de ser carregadas, a memória necessária para guardar a informação dos seus dados e das representações geradas terá de ser maior.

Quanto à memória reservada e a não utilizada, também os seus valores estão dentro do expectável. Como a quantidade de memória reservada é gerida pelo Unity, este tenta reservar uma quantidade adicional para garantir um desempenho mais estável, permitindo que essa memória extra possa ser alocada mais rapidamente se for necessária. Por este motivo, o valor da memória não utilizada depois do carregamento da área inicial é relativamente constante, independentemente do tamanho da área. Em consequência, os valores medidos para a quantidade total de memória reservada também aumentam em proporção com o tamanho da área carregada, correspondendo à soma da memória alocada com a não utilizada.

Como uma área de 9x9 *tiles* (perfazendo um total de 81) permite cobrir todo o *viewport*, deixando apenas uma ligeira margem fora deste, foi o valor escolhido para ser utilizado. Assim, permite mostrar conteúdo quando se desloca o mapa enquanto se espera que novas *tiles* carreguem permitindo uma experiência mais suave, sem ter de carregar gastar uma quantidade maior de recursos.

Teste L3 | Impacto das etapas de carregamento

O objetivo deste teste é o de avaliar a influência que cada etapa do processo de carregamento de uma *tile* tem no carregamento da área.

- **Parâmetros constantes:** Número de *tiles* a carregar (área de 9x9 *tiles*, num total de 81) e número de pedidos concorrentes (12 pedidos)

Na Tabela 6.1 estão presentes os tempos médios que foram gastos pelas *tiles* nas várias etapas do seu carregamento. Foram feitas três tentativas diferentes (T1, T2 e T3) de carregamento da área, sendo que para cada uma foi calculado o valor médio que todas as 81 *tiles* gastaram em cada uma das etapas. Dada a enorme quantidade de dados, optou-se por apenas mostrar os tempos relativos ao terreno, a uma camada *raster* (Satellite) e três camadas vetoriais com diferente complexidade geométrica (Buildings, Sidewalks e Signs), cujos dados georreferenciados apresentam uma distribuição espacial pelas várias *tiles* do mapa relativamente uniforme.

Tabela 6.1: Tempos médios das várias etapas (valores em ms).

Etapa	T1	T2	T3	Valor Médio
Pedidos do terreno	76.76	73.39	68.32	72.82
Processamento do terreno	2.05	2.07	2.45	2.19
Satellite - Pedidos de camada	45.68	47.78	44.15	45.87
Satellite - Renderização de camada	0.11	0.16	0.11	0.13
Buildings - Pedidos de camada	19.75	18.90	20.23	19.62
Buildings - Processamento de camada	8.23	8.09	7.83	8.05
Buildings - Renderização de camada	5.79	5.91	5.91	5.87
Sidewalks - Pedidos de camada	23.92	22.08	22.42	22.81
Sidewalks - Processamento de camada	2.03	2.05	1.88	1.99
Sidewalks - Renderização de camada	0.71	0.67	0.85	0.74
Signs - Pedidos de camada	23.15	20.15	20.13	21.14
Signs - Processamento de camada	0.44	0.71	0.53	0.56
Signs - Renderização de camada	0.45	0.34	0.34	0.37

Como é possível observar, as etapas que demoram uma fatia considerável do tempo são as relativas aos pedidos de dados, seja do terreno ou das várias camadas. Como a paralelização do carregamento da área foi feita para aproveitar os tempos de inatividade enquanto a aplicação aguarda pela resposta destes pedidos de dados, estes valores corroboram o que foi determinado no teste L1, em que ao ter um maior número de tarefas concorrentes permite que as *tiles* que já tenham recebido a resposta dos seus pedidos possam passar para as etapas seguintes, enquanto as outras se mantêm inativas a aguardar.

Também é notável a diferença entre a duração dos pedidos para as camadas vetoriais em comparação com as *raster* e do terreno, visto que estes são feitos localmente para o *geojson-tile-server*, enquanto os restantes implicam uma ligação através da Internet. Adicionalmente, os pedidos de dados do terreno também são mais morosos em comparação às camadas *raster*, o que também ajuda a explicar o que foi observado no teste L1 relativamente ao *First Contentful Paint*, pois antes que qualquer *tile* consiga mostrar algo, tem de ter carregado e processado os dados do terreno.

Quanto ao processamento e renderização das camadas vetoriais os valores estão dentro do esperado. A camada Buildings, dada a maior quantidade de dados presentes nela e a geometria gerada a partir destes (modelos extrudidos), demora mais tempo que a camada Sidewalks, representada apenas por polígonos. Por sua vez, esta demora mais tempo que a camada Signs que apenas tem marcadores nos pontos correspondentes aos sinais, que também são menos frequentes que edifícios e passeios.

Como estes valores são médias relativas a todas as *tiles*, focar na diminuição das etapas de cada *tile* cujo valores são mais elevados tem um maior potencial para melhorias ao nível do carregamento total da área. Visto que as etapas com maior impacto são as relativas aos pedidos, seria neste ponto que potenciais melhorias deveriam ser focadas. Duas ideias

que poderiam ser exploradas em trabalho futuro seria fazer *caching* e *bundling* dos pedidos de dados.

Ao guardar numa *cache* os pedidos recentes, permitiria uma diminuição da quantidade de tempo gasta em pedidos repetidos, limitando-se a utilizar os resultados dos pedidos anteriores. Teria, no entanto, de se determinar qual o tamanho apropriado para esta *cache*, para que os seus benefícios em termos de eficiência temporal não sejam anulados pela sua utilização de memória extra, visto que a eficiência espacial em termos de carregamento de dados foi o motivo da divisão do mapa em *tiles*.

A outra ideia, de *bundling* dos dados, consiste em fazer um único pedido para obter todos os conjuntos de dados relativos a uma única *tile*, em vez de fazer um pedido individual para cada camada de dados de cada *tile*. Esta diminuição na quantidade total de pedidos poderia implicar uma redução no tempo total gasto pela aplicação à espera de dados, especialmente ao eliminar o *overhead* associado a fazer cada pedido individualmente. No entanto, isto significaria que a *tile* não pode ir mostrando camadas de dados à medida que elas surgem, pois estão todas presentes no mesmo pedido, e só podem ser processadas após a conclusão deste. Adicionalmente, já envolveria uma modificação profunda do *geojson-tile-server* ou a criação de raiz de um servidor para o efeito, visto que não é algo suportado por este.

6.2.2 Grupo M - Desempenho durante movimento

O objetivo principal deste grupo de testes é avaliar o desempenho da aplicação durante dois tipos de movimentos ao longo do mapa com diferentes níveis de ampliação. Pretende-se simular a execução da aplicação quando um utilizador se tenta deslocar em linha reta com um nível de ampliação baixo (cobrindo uma área maior) e quando segue uma rota com um nível de ampliação alto (cobrindo uma área menor, mas com mais detalhe). Para todos os testes deste grupo, o número de *tiles* a carregar correspondeu a uma área de 9x9 *tiles* (num total de 81) e o número de pedidos concorrentes correspondeu a 12 pedidos.

Teste M1 | Voo rasante

O objetivo deste teste é o de avaliar o impacto do carregamento de uma vasta área durante o movimento, simulando o comportamento de um utilizador que pretenda deslocar em linha reta de um ponto a outro com um nível de ampliação reduzido.

Foi definido um voo rasante em linha reta entre a Praça do Comércio e a Praça do Chile com o nível de ampliação 16, tendo sido efetuada a medição da *framerate* em 7 instantes diferentes, conforme mostra a Figura 6.3.



Figura 6.3: Percursos do voo rasante e da rota e instantes de medição de métricas.

Teste M2 | Percurso de rota

O objetivo deste teste é o de avaliar o impacto do carregamento de uma área menor durante o movimento embora com uma certa inclinação da câmara, simulando o comportamento de um utilizador que tenha obtido uma rota e pretenda deslocar o mapa de forma a segui-la, com um nível de ampliação mais alto, enquanto observa os modelos presentes no mapa.

Foi definido um percurso entre a Praça de Londres e o Parque Eduardo VII com o nível de ampliação 19, tendo sido efetuada a medição da *framerate* em 8 instantes diferentes, conforme mostra a Figura 6.3.

Resultados do Grupo M

Na Figura 6.4 está presente a *framerate* medida em diversos instantes dos testes M1 e M2, tratando-se dos valores médios resultantes de três tentativas diferentes.

No caso do voo rasante (Figura 6.4(a)), o gráfico mostra uma descida acentuada, em que cerca de 88 FPS no instante 1 passa para cerca de 59 FPS no instante 2, seguido por cerca de 43 FPS no instante 3. A partir desse instante a descida suaviza para cerca de 42 FPS no instante 4 e 40 FPS no instante 5, oscilando ligeiramente em redor deste valor nos instantes seguintes.

Quanto ao percurso de rota (Figura 6.4(b)), os valores são bastante mais estáveis oscilando em torno de 144 FPS, com máximo de 144.03 FPS no instante 2 e mínimo de 143.92 FPS no instante 5.

Embora sejam feitas algumas simplificações nas representações geradas para o nível de ampliação 16 (como mostrar apenas os telhados dos edifícios, sem gerar as paredes) para melhorar o desempenho, ainda é visível um contraste significativo com o nível 19. No

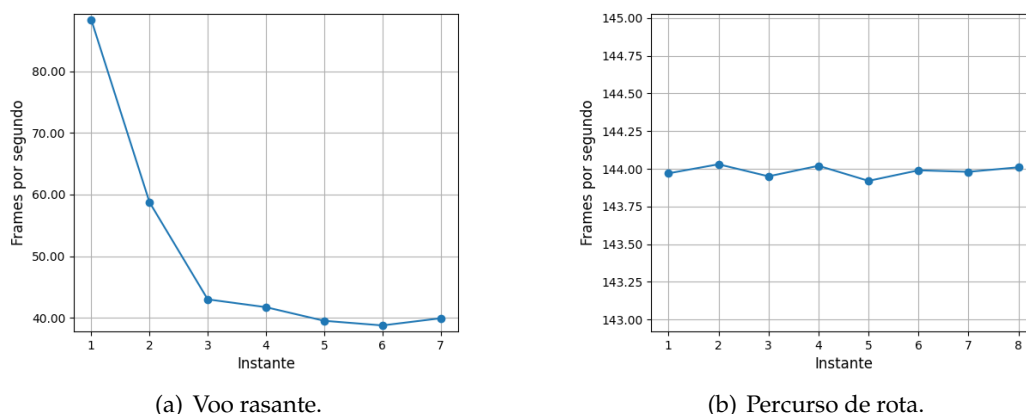


Figura 6.4: *Framerate* média ao longo dos dois testes.

entanto, é importante lembrar que as *tiles* XYZ funcionam como uma espécie de *quadtree*, o que implica que por cada diminuição do nível de ampliação implica um aumento da área coberta num fator de 4x. Desta forma, a área coberta pelas 81 *tiles* do nível de ampliação 16 utilizada no teste M1, corresponde a uma área que seria coberta por 5184 *tiles* do nível 19, pois ao reduzir o nível três vezes implica um fator de aumento da área de 64 vezes.

Mesmo tendo em conta a diferença da escala na área coberta, a *framerate* durante o percurso mantém-se constante, enquanto no voo há uma descida considerável e só depois é que estabiliza. Esta diferença pode ser explicada observando a localização dos instantes em que a descida ocorre no mapa, ou seja, onde se situam os pontos correspondentes aos instantes 1, 2 e 3.

O teste M1 começa na Praça do Comércio (instante 1), atravessa a Baixa Pombalina (instante 2) em direção à Praça Martim Moniz (instante 3), subindo ao longo da Avenida Almirante Reis (instantes 4 a 6) até à Praça do Chile (instante 7). Ao começar junto ao rio Tejo, uma parte significativa da área inicialmente coberta pelo teste encontra-se precisamente neste, não havendo então dados e, consequentemente, representações a gerar para essas *tiles*. À medida que o mapa é deslocado para o interior de Lisboa e se afasta do rio, a quantidade de edifícios e outros objetos a serem representados é bastante maior, o que explica que a maioria da descida do desempenho ocorra apenas até à Praça Martim Moniz, sendo mais estável a partir desse instante.

No caso do teste M2, todo o percurso entre a Praça de Londres e o Parque Eduardo VII é afastado do rio, não havendo enormes variações na quantidade de objetos a representar. Isto em combinação com a reduzida área de cobertura permite que a *framerate* se mantenha estável com 144 FPS, valor que corresponde à taxa de atualização do monitor utilizado.

De qualquer modo, este teste mostrou que as representações para níveis de ampliação mais reduzidos, que por sua vez cobrem uma área maior, ainda deveriam ser alvo de simplificação ou utilizar outras técnicas, de forma a melhorar o desempenho da aplicação nestes casos.

6.3 Testes de usabilidade com utilizadores

A fase final da avaliação compreendeu os testes de usabilidade da aplicação realizados com utilizadores. Estes testes foram essenciais para verificar o desempenho de uma forma mais qualitativa e a facilidade de utilização da aplicação em situações reais, bem como para recolher opiniões por parte dos utilizadores sobre a mesma.

Nesta secção, começa-se por apresentar a metodologia utilizada para as sessões de teste, seguida pelos resultados obtidos nessas sessões.

6.3.1 Metodologia

Cada sessão de teste envolveu apenas duas pessoas, o participante e o investigador. Estas sessões foram desenhadas com o intuito de ter um progressivo nível de dificuldade nas tarefas pedidas (tentando manter tarefas relacionadas no mesmo grupo) e evitar ultrapassar noventa minutos por sessão, de modo a encorajar o participante a continuar para tarefas mais complexas e evitar causar-lhe fadiga [58][65].

Durante a sessão, é disponibilizado ao participante um formulário criado com recurso ao Google Forms¹ onde está presente o guião da sessão, com as tarefas e questionários a realizar durante o teste. Esta ferramenta foi a escolhida para consolidar o guião e questionários, de modo a simplificar o processo de recolha das respostas dos participantes.

Para além do formulário fornecido aos utilizadores, foi também criado o formulário presente no Apêndice C para o investigador registar observações relativas às tarefas e questões e foi também efetuada a captura do ecrã da aplicação para posterior análise.

A sessão começa com um curto *briefing* do contexto do estudo, seguida pelas instruções e estrutura da sessão e, por fim, a declaração de consentimento informado, sendo que o teste apenas é realizado caso o participante concorde e declare o seu consentimento. O texto integral destes encontra-se no Apêndice B.

O restante da sessão de teste tem a seguinte estrutura:

- **Questionário de Caracterização** - para obter dados demográficos e de experiência relevante dos participantes.
- **Grupos A, B, C e D** - séries de tarefas e questões relativas à aplicação agrupadas por localização e tipos de tarefas, sendo pedido aos participantes para pensarem em voz alta durante este processo, descreverem as suas ações e raciocínios.
- **Grupo E** - comparação entre a utilização da aplicação em dispositivos *Desktop* e *Mobile*.
- **Questionário Específico de Usabilidade** - questões gerais sobre funcionalidades específicas da aplicação.

¹Ferramenta disponível em <https://www.google.com/forms/about/>

- **System Usability Scale e User Experience Questionnaire** - questionários *standard* de usabilidade e experiência de utilizador.
- **Comentários Finais** - partilha de *feedback* geral.

Questionário de Caracterização

O primeiro questionário que é apresentado ao participante é sobre si próprio. O seu objetivo é recolher informações demográficas e sobre experiência relevante dos participantes, fundamentais para contextualizar os resultados do teste de forma significativa.

São pedidos os seguintes dados:

- **Idade** - A idade do participante;
- **Género** - Questão de escolha múltipla com *Feminino*, *Masculino*, *Não Binário* e *Prefiro não dizer* como opções;
- **Nível de Escolaridade** - Questão de escolha múltipla com *Ensino Básico*, *Ensino Secundário*, *Ensino Superior - Licenciatura*, *Ensino Superior - Mestrado* e *Ensino Superior - Doutoramento* como opções;
- **Experiência com Sistemas de Informação Geográfica (SIG)** - Questão de escolha múltipla com *Nunca usei*, *Uso Casual*, *Uso em Contexto Académico* e *Uso em Contexto Profissional* como opções;
- **Experiência com Realidade Aumentada** - Questão de escolha múltipla com *Nunca usei*, *Uso Casual: Já usei algumas aplicações de realidade aumentada ocasionalmente* e *Uso Frequente: Uso aplicações de realidade aumentada de forma regular* como opções.

Grupo A - Praça do Comércio

Neste ponto da sessão de testes, a aplicação é aberta (no mesmo computador usado nos testes de desempenho) e o participante é convidado a partilhar em voz alta quaisquer pensamentos que tenha durante o processo, a descrever as suas ações, raciocínios e dúvidas. Não são dadas mais indicações ao participante para além das que constam no guião.

Este primeiro grupo de tarefas está centrado na Praça do Comércio e serve para o participante se familiarizar com as interações bases da aplicação, gestão de camadas e filtragem.

O grupo é composto pelas seguintes 3 tarefas e 1 questão:

- **Tarefa A1** | Mude da camada "Watercolor" para a camada "Satellite"
- **Tarefa A2** | Configure o filtro "Extrude" da camada "Buildings" como Igual a 31.17 e ative-o

- **Questão A2Q1** | Quantos edifícios restam?
 - Campo de resposta livre (Resposta certa: 1).
- **Tarefa A3** | Mude da camada "Satellite" para a camada "OpenStreetMap"

Grupo B - Praça da Alegria

Este grupo está centrado na Praça da Alegria e testa a combinação do modo de consulta com a filtragem de dados e a mudança de localização.

O grupo é composto pelas seguintes 4 tarefas e 3 questões:

- **Tarefa B1** | Vá para a Praça da Alegria
- **Tarefa B2** | Ative o Modo de Consulta
- **Questão B2Q1** | Qual é a "Tipologia" das árvores no Jardim Alfredo Keil (Jardim da Praça da Alegria)?
 - Escolha múltipla com as opções *Caldeira*, *Canteiro* (Resposta certa), *Não identificada* e *Sem caldeira*.
- **Tarefa B3** | Configure o filtro "Espécie" da camada "Trees" como Contém "celtis a" e ative-o
- **Questão B3Q1** | Quantas árvores correspondentes ao filtro existem na Praça da Alegria (dentro e fora do Jardim) e qual é o nome da espécie?
 - Escolha múltipla com as opções *11 árvores da espécie Celtis africana*, *11 árvores da espécie Celtis australis* (Resposta certa), *15 árvores da espécie Celtis africana* e *15 árvores da espécie Celtis australis*.
- **Tarefa B4** | Alterar o valor do filtro "Espécie" para "celtis o"
- **Questão B4Q1** | Quantas árvores correspondentes ao filtro existem nas 4 secções mais próximas da Avenida da Liberdade e qual é o nome da espécie?
 - Escolha múltipla com as opções *3 árvores da espécie Celtis occidentalis*, *3 árvores da espécie Celtis orthocanthos*, *5 árvores da espécie Celtis occidentalis* (Resposta certa) e *5 árvores da espécie Celtis orthocanthos*.

Grupo C - Praça de Londres

Este grupo está centrado na Praça de Londres e testa a utilização de filtros de propriedades categóricas, a utilização simultânea de múltiplos filtros e os controlos da câmara.

O grupo é composto pelas seguintes 5 tarefas e 2 questões:

- **Tarefa C1** | Vá para a Praça de Londres
- **Tarefa C2** | Remova a inclinação da câmara e alinhe-a em direção a Norte

- **Tarefa C3** | Ative o filtro "Tipologia" da camada "Bikepaths"
- **Questão C3Q1** | Qual destas tipologias não existe na área envolvente à Praça de Londres?
 - Escolha múltipla com as opções *30+Bici*, *Contrassentido*, *Pista Ciclável Bidirecional* e *Zona de Coexistência* (Resposta certa).
- **Tarefa C4** | Desative o filtro anterior e ative os filtros "Comprimento" e "Zonamento"
- **Questão C4Q1** | Quantos troços de ciclovias com menos de 50 metros de comprimento e zoneados como "Roma-Areeiro" existem na área envolvente à Praça de Londres?
 - Campo de resposta livre (Resposta certa: 3).
- **Tarefa C5** | Desative *todos* os filtros e o modo de Consulta

Grupo D - Parque Eduardo VII

Este grupo está centrado no Parque Eduardo VII e testa a deslocação no mapa e a utilização das ferramentas da Rota e da Régua.

O grupo é composto pelas seguintes 5 tarefas e 5 questões:

- **Tarefa D1** | Obtenha a Rota da "Praça de Londres" até ao "Parque Eduardo VII" e siga-a
- **Questão D1Q1** | Qual é a distância?
 - Campo de resposta livre (Resposta certa: 3.54km).
- **Questão D1Q2** | Qual é a duração?
 - Campo de resposta livre (Resposta certa: 7min 32s).
- **Questão D1Q3** | Concorda com a afirmação "Enquanto segui a rota, achei o movimento suave"?
 - Escala *Likert* com 5 níveis, em que 1 corresponde a *Discordo totalmente* e 5 a *Concordo totalmente*.
- **Tarefa D2** | Desative a rota
- **Tarefa D3** | Mude da camada "OpenStreetMap" para a camada "Satellite" e desative a camada "Buildings"
- **Tarefa D4** | Ligue a Régua
- **Questão D4Q1** | Qual é o comprimento da área verde central do Parque Eduardo VII (aproximadamente)?
 - Campo de resposta livre (Resposta certa: Aprox. 530m).

- **Questão D4Q2** | Qual é o perímetro da área verde central do Parque Eduardo VII (aproximadamente)?
 - Campo de resposta livre (Resposta certa: Aprox. 1170m).
- **Tarefa D5** | Limpe a régua e desligue-a

Grupo E - Castelo de S. Jorge

Este último grupo de tarefas está centrado no Castelo de S. Jorge e serve para fazer uma comparação entre a utilização da aplicação em dispositivos *Desktop* e *Mobile*, assim como o modo de Realidade Aumentada e testar a visualização tridimensional do terreno.

As tarefas deste grupo encontram-se divididas em dois subgrupos, correspondentes aos dispositivos (*Desktop* e *Mobile*) que estão a ser testados. Os testes *Desktop* foram efetuados no mesmo computador utilizado durante os testes de desempenho e os testes *Mobile* foram efetuados num *smartphone* Nokia 6.1 (2018) com as seguintes especificações:

- Chipset: Qualcomm SDM630 Snapdragon 630
- CPU: Octa-core 2.2 GHz Cortex-A53
- GPU: Adreno 508
- RAM: 3 GB
- Sistema Operativo: Android 10 (API 29)

Em ambos os casos, o computador tinha aberto o `geojson-tile-server` para servir os dados vetoriais. O grupo é composto por 7 tarefas e 5 questões no total.

Subgrupo E1D | Desktop

As seguintes 3 tarefas são relativas à utilização da aplicação em dispositivos *Desktop*. Neste ponto da sessão de testes, a aplicação *Desktop* é reiniciada de modo que o ponto de partida para a comparação seja igual para os dois dispositivos.

- **Tarefa E1D1** | Vá para o Castelo de São Jorge
- **Tarefa E1D2** | Ative a visualização de Terreno
- **Tarefa E1D3** | Use os controlos da câmara para observar o terreno

Subgrupo E1M | Mobile

As seguintes 4 tarefas são relativas à utilização da aplicação em dispositivos *Mobile*. Neste ponto da sessão de testes, a aplicação *Mobile* é aberta e o marcador de Realidade Aumentada é colocado na mesa.

- **Tarefa E1M1** | Vá para o Castelo de São Jorge

- **Tarefa E1M2** | Ative a visualização de Terreno
- **Tarefa E1M3** | Use os controlos da câmara para observar o terreno
- **Tarefa E1M4** | Ative o modo de Realidade Aumentada e detete o marcador para observar o terreno

Questões

- **Questão E1Q1** | Concorda com a afirmação "Achei que a navegação e interação na versão Desktop foi fácil"?
- Escala *Likert* com 5 níveis, em que 1 corresponde a *Discordo totalmente* e 5 a *Concordo totalmente*.
- **Questão E1Q2** | Concorda com a afirmação "Achei que a navegação e interação na versão Mobile foi fácil"?
- Escala *Likert* com 5 níveis, em que 1 corresponde a *Discordo totalmente* e 5 a *Concordo totalmente*.
- **Questão E1Q3** | Concorda com a afirmação "Achei que o modo de Realidade Aumentada foi fácil e intuitivo de usar"?
- Escala *Likert* com 5 níveis, em que 1 corresponde a *Discordo totalmente* e 5 a *Concordo totalmente*.
- **Questão E1Q4** | Qual das três interfaces é que achou mais adequada para visualizar o Terreno?
- Grelha para ordenar as opções *Desktop*, *Mobile* e *Realidade Aumentada* como 1^a (Mais adequada), 2^a e 3^a (Menos adequada).
- **Questão E1Q5** | Qual das três interfaces é que preferiu de um modo geral?
- Grelha para ordenar as opções *Desktop*, *Mobile* e *Realidade Aumentada* como 1^a preferência, 2^a preferência e 3^a preferência.

Questionário Específico de Usabilidade

Findas as tarefas que o participante tem de realizar, neste questionário pretende-se obter a opinião relativamente à usabilidade de funcionalidades específicas da aplicação. É composta pelas seguintes 7 afirmações às quais o participante pode dar a sua opinião através de escalas *Likert* com 5 níveis, em que 1 corresponde a *Discordo totalmente* e 5 a *Concordo totalmente*:

- **Questão U1Q1** | Achei a navegação e controlo da câmara fácil e intuitivo
- **Questão U1Q2** | Achei a mudança de localização fácil e intuitiva

- **Questão U1Q3** | Achei a gestão de camadas fácil e intuitiva
- **Questão U1Q4** | Achei a configuração e utilização de filtros fácil e intuitiva
- **Questão U1Q5** | Achei o modo de consulta fácil e intuitivo
- **Questão U1Q6** | Achei a ferramenta da Rota fácil e intuitiva
- **Questão U1Q7** | Achei a Régua fácil e intuitiva

System Usability Scale e User Experience Questionnaire

Nesta secção da sessão de testes, os participantes preenchem os questionários relativos ao [SUS](#) [12] e o [UEQ](#) [89], replicados no Anexo I. Trata-se de ferramentas amplamente utilizadas na área de *User Experience*, sendo desenhadas para obter *insights* sobre as perceções e avaliações dos utilizadores relativamente à usabilidade e experiência geral de um produto.

O [SUS](#) é um questionário desenvolvido por John Brooke [12], que consiste em 10 afirmações, classificáveis numa escala *Likert* com 5 níveis, desde 1 para *Discordo totalmente* a 5 para *Concordo totalmente*. Estas respostas são então utilizadas para determinar uma pontuação entre 0 e 100 que, embora não seja uma percentagem, indica uma melhor usabilidade quão maior for o resultado. Segundo o autor John Brooke numa publicação posterior [13], uma investigação de mais de 3500 resultados [SUS](#) mostrou que uma pontuação de 68 é o valor médio.

Por sua vez, o [UEQ](#) consiste em 26 pares de opostos relativos às propriedades que a aplicação possa ter. O participante classifica cada um destes itens numa escala de diferencial semântico com 7 níveis, sendo ambos os atributos as extremidades desta escala (por exemplo, para o par Atraente-Feia, uma classificação 2 implicaria que acha a aplicação mais Atraente do que Feia). Ao contrário da pontuação única do [SUS](#), os resultados do [UEQ](#) são divididos nas seguintes seis categorias distintas [89]:

- **Atratividade:** Impressão geral da aplicação. Os utilizadores gostam ou detestam?
- **Transparência:** É fácil familiarizar-se com a aplicação e aprender como utilizá-la?
- **Eficiência:** Os utilizadores conseguem resolver as suas tarefas sem esforço desnecessário? A aplicação reage rapidamente?
- **Controlo:** O utilizador sente-se em controlo da interação? É segura e previsível?
- **Estimulação:** É emocionante e motivante usar a aplicação? É divertida de se usar?
- **Inovação:** O design da aplicação é criativo? Atrai o interesse dos utilizadores?

Comentários Finais

Por fim, o participante é convidado a partilhar quaisquer tipos de feedback geral que tenha, para além dos comentários que pode fazer ao longo da sessão. Esta secção final é composta por três campos de resposta aberta para indicar Aspectos negativos, Aspectos positivos e Sugestões.

6.3.2 Resultados

Introduzida a metodologia utilizada nos testes de usabilidade, são agora apresentados os resultados, observações e *feedback* obtidos durante a sua realização. Começa-se pela caracterização da população testada, seguido pelos vários grupos de tarefas, o questionário específico de usabilidade, os questionários *standard* SUS e UEQ, concluindo com os comentários finais deixados pelos participantes.

Caracterização da população testada

Na Figura 6.5 estão presentes os gráficos com a caracterização da população que participou nos testes de usabilidade, composta por 12 participantes.

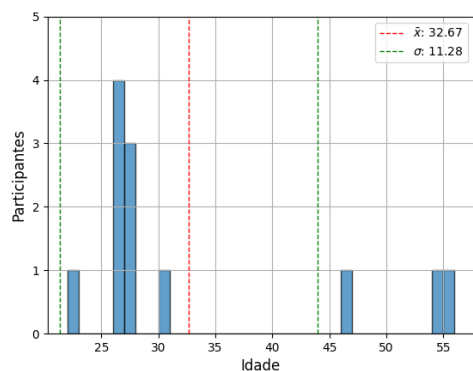
A idade da população (Figura 6.5(a)) está distribuída entre os 22 e 55 anos, com $\bar{x} = 32.67$ e $\sigma = 11.28$, sendo que a maioria está contida entre os 22 e 30 anos, tendo uma moda de 26 anos.

Em termos de género (Figura 6.5(b)) a população participante é bastante paritária, com 6 participantes do género feminino e 6 do género masculino.

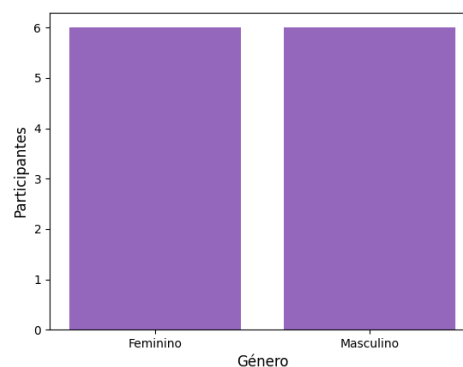
Quanto ao nível de escolaridade (Figura 6.5(c)), praticamente toda a população concluiu um grau de ensino superior, 7 participantes têm pelo menos uma licenciatura e 4 um mestrado, existindo apenas 1 participante com o grau de ensino secundário.

No que toca à experiência com SIG (Figura 6.5(d)), aproximadamente dois terços dos participantes não têm experiência prévia ou apenas casual (7 sem nenhuma e 1 casual), 3 participantes têm experiência de utilização num contexto académico e 1 participante tem experiência de utilização em ambiente profissional (cerca de uma década de experiência). Embora uma parte significativa possa não ter experiência relevante, uma aplicação que para seja fácil de usar para estes e que tenha boa usabilidade, potencialmente também o será para utilizadores com experiência.

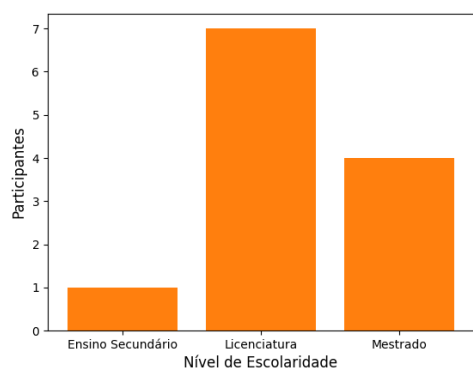
Por fim, relativamente à experiência com aplicações de realidade aumentada (Figura 6.5(e)), 4 participantes indicaram não ter qualquer experiência prévia e 8 indicaram ter utilizado ocasionalmente aplicações de RA, sendo a aplicação Pokémon Go o exemplo mais comum dado por estes. Interessantemente, foi detetada uma divisão da população em termos etários em relação à sua experiência com realidade aumentada (Figura 6.5(f)), com todos os participantes menores que 30 anos a reportarem a sua utilização ocasional e os 4 participantes com idade igual ou superior a 30 anos a reportarem não ter qualquer experiência prévia.



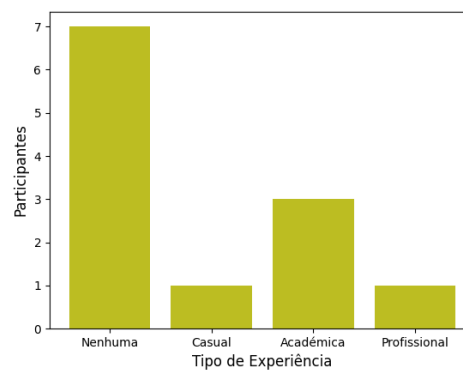
(a) Idade.



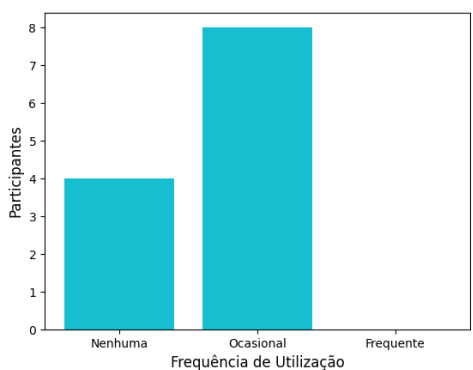
(b) Gênero.



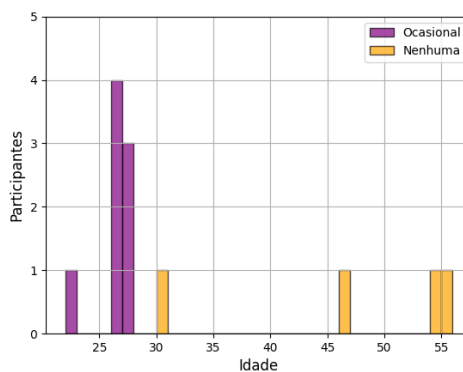
(c) Nível de escolaridade.



(d) Experiência com SIG.



(e) Experiência com RA.



(f) Relação Experiência com RA e Idade.

Figura 6.5: Caracterização da população participante.

Grupo A - Praça do Comércio

Na Tabela 6.2 é indicado o desempenho dos participantes nas tarefas e questão do Grupo A, centrado na Praça do Comércio, que serve para os participantes se familiarizarem com as interações bases da aplicação, gestão de camadas e filtragem.

Tabela 6.2: Desempenho dos participantes no Grupo A.

Tarefa/Questão	Completo	Completo, com ajuda	Não completo
Tarefa A1	12	0	0
Tarefa A2	10	2	0
Questão A2Q1	12	0	0
Tarefa A3	12	0	0

A única tarefa que gerou dificuldades para os participantes foi a Tarefa A2, em que é pedido para configurar o filtro "Extrude" da camada "Buildings" como Igual a 31.17 e ativá-lo. Três participantes clicaram primeiro no botão da barra de ferramentas que apenas ativa a filtragem, na expectativa de que abrisse uma janela de configuração dos filtros, tendo depois localizado que a configuração destes se encontra na vista de detalhe das camadas. Todos os 12 participantes configuraram o filtro com sucesso, mas 4 tiveram uma ligeira dificuldade a ativá-lo pois pensavam que apenas precisam de selecionar a opção Use Filter na configuração do filtro. Destes 4, 2 acabaram por clicar no botão da barra de ferramentas para ativar a filtragem autonomamente, enquanto os outros 2 necessitaram que lhes fosse questionado se existia alguma opção por ativar (sendo essa ajuda assinalada).

Grupo B - Praça da Alegria

Na Tabela 6.3 é indicado o desempenho dos participantes nas tarefas e questões do Grupo B, centrado na Praça da Alegria, que testa a mudança de localização e a combinação do modo de consulta com a filtragem de dados.

Tabela 6.3: Desempenho dos participantes no Grupo B.

Tarefa/Questão	Completo	Completo, com ajuda	Não completo
Tarefa B1	12	0	0
Tarefa B2	12	0	0
Questão B2Q1	11	1	0
Tarefa B3	12	0	0
Questão B3Q1	12	0	0
Tarefa B4	12	0	0
Questão B4Q1	12	0	0

A Tarefa B1, em que é pedido para ir para a Praça da Alegria, foi a que mais suscitou dúvidas neste grupo, embora todos os participantes a tenham completado com sucesso

e sem ajuda. Três participantes tentaram primeiro ativar o modo de consulta, devido ao ícone se tratar de uma lupa, pensando que Query seria para procurar a localização, tendo apenas aberto a janela da localização após isto. Na janela de localização, quatro participantes tentaram utilizar a pesquisa, assim como a navegação por ponto de interesse. Quando questionados posteriormente sobre este assunto, indicaram que pretendiam usar a pesquisa, mas acharam que tinham de selecionar as opções todas dada a sua proximidade entre si. Na Tabela 6.4, estão indicados os métodos preferidos pelos vários participantes para navegar para a Praça da Alegria, havendo uma forte preferência pela utilização da pesquisa com 10 participantes, sendo que apenas 1 participante utilizou os Pontos de Interesse e 1 deslocou o mapa ao arrastá-lo com o rato desde a Praça do Comércio.

Tabela 6.4: Navegação na Tarefa B1.

Método preferido	Participantes
Pesquisa	10
Pontos de Interesse	1
Rato (<i>panning</i>)	1
Teclado (<i>panning</i>)	0
UI (<i>panning</i>)	0

Na Questão B2Q1, em que é perguntada qual a "Tipologia" das árvores no Jardim Alfredo Keil (Jardim da Praça da Alegria) após terem ativado o modo de consulta na Tarefa B2, houve 1 participante que tentou obter essa informação primeiro ao observar o modelo das árvores (na expectativa de que existisse uma representação diferente para cada tipologia) e, após concluir que não era possível determinar desse modo, ponderou utilizar o filtro correspondente, tendo em vez disso optado passar com o rato por cima da representação da árvore (pensando que poderia surgir uma *tooltip* com os dados dos atributos). Só após ter sido sugerido que não evitasse clicar em elementos da aplicação é que tentou clicar numa árvore para abrir a janela de consulta com os atributos desta, tendo essa ajuda sido assinalada. É também relevante mencionar que, embora tenham completado a questão com sucesso e sem ajuda, houve 3 participantes que não se aperceberam do que fazia o modo de consulta, tendo assumido que podiam clicar para ver os atributos de um objeto em qualquer instante e só compreenderam isto já numa fase posterior quando questionaram sobre o seu propósito. Houve também 1 outro participante que sugeriu a mudança do cursor quando o modo de consulta está ativo (por exemplo ter um ponto de interrogação) para mostrar esta mudança de modo mais eficazmente.

Nas Tarefas e Questões B3, B3Q1, B4 e B4Q1 não houve nada de relevante a apontar, tendo todos os participantes conseguido responder corretamente às questões, demonstrando a capacidade de combinar a utilização dos filtros correspondentes com o modo de consulta.

Grupo C - Praça de Londres

Na Tabela 6.5 é indicado o desempenho dos participantes nas tarefas e questões do Grupo C, centrado na Praça de Londres, que testa os controlos da câmara, a utilização de filtros de propriedades categóricas e a utilização simultânea de múltiplos filtros.

Tabela 6.5: Desempenho dos participantes no Grupo C.

Tarefa/Questão	Completo	Completo, com ajuda	Não completou
Tarefa C1	12	0	0
Tarefa C2	10	2	0
Tarefa C3	12	0	0
Questão C3Q1	12	0	0
Tarefa C4	12	0	0
Questão C4Q1	12	0	0
Tarefa C5	12	0	0

Para a Tarefa C1, que pede para ir para a Praça de Londres, a totalidade dos participantes optou pela pesquisa como método de navegação como mostra a Tabela 6.6. Ao contrário da Tarefa B1, a proximidade da pesquisa à lista de pontos de interesse já não suscitou confusão, mas 1 participante sugeriu que deveriam estar situadas em janelas diferentes ou pelo menos a pesquisa poderia ser um campo permanentemente no topo do mapa à semelhança do Google Maps.

Tabela 6.6: Navegação na Tarefa C1.

Método preferido	Participantes
Pesquisa	12
Pontos de Interesse	0
Rato (<i>panning</i>)	0
Teclado (<i>panning</i>)	0
UI (<i>panning</i>)	0

Neste grupo a maioria das dificuldades concentraram-se na Tarefa C2, em que é pedido para remover a inclinação da câmara e alinhá-la em direção a Norte. Para remover a inclinação da câmara, 3 participantes começaram por o fazer clicando múltiplas vezes no botão para decrementar a inclinação até experimentarem o que a remove por completo. Para alinhar a Norte, houve 6 participantes que não tinham a certeza se o mapa estava alinhado ou não, mesmo após terem clicado no botão para o efeito, pois embora tenham notado que a orientação tinha sido alterada, não havia maneira de confirmar qual era a atual, tendo alguns sugerido que o ícone da bússola rodasse consoante a orientação atual do mapa. Ainda relativo à orientação, houve 1 participante que tentou rodar a orientação manualmente até perceber que não fazia ideia de qual era a sua orientação inicial nem

a atual, apenas tendo clicado no botão da bússola quando questionado se poderia haver um botão que ajudasse a alinhar (tendo essa ajuda sido assinalada), ao que mencionou não o ter visto, assim como tinha tido dificuldade em reconhecer o ícone do botão para remover a inclinação (tendo também sido um dos 3 participantes que tentou primeiro remover a inclinação manualmente). Finalmente, houve 1 participante que não reconheceu de todos os ícones dos controlos associados à câmara, em parte dada a ausência de uma legenda, tendo sido necessário apontar o grupo de botões para este efeito, mas sem indicar quais as suas funções específicas para além da sua associação ao controlo da câmara (tendo esta ajuda sido apontada). Dadas as dificuldades, é de notar a sugestão de 3 outros participantes para que fossem adicionadas *tooltips* aos controlos da câmara, visto que são os únicos botões da barra de ferramentas que não possuem legenda devido ao seu tamanho.

Na Questão C3Q1 não houve nada de relevante a apontar, tendo todos os participantes conseguido responder corretamente à questão, demonstrando a capacidade de utilizar os filtros de propriedades categóricas.

Na Tarefa C4, em que é pedido para desativar o filtro "Tipologia" (ativado na Tarefa C3) e ativar os filtros "Comprimento" e "Zonamento", todos os participantes completaram a tarefa com sucesso. No entanto, de forma a tornar mais fácil de encontrar as propriedades pretendidas, 2 participantes sugeriram que estas fossem ordenadas por ordem alfabética (a ordem utilizada era a presente nos conjuntos de dados) e outros 2 sugeriram que existisse um campo na vista de detalhe da camada que permitisse procurar a propriedade pretendida inserindo o seu nome.

Na Questão C4Q1, em que é perguntando quantos troços de ciclovias com menos de 50 metros de comprimento e zoneados como "Roma-Areeiro" existem na área envolvente à Praça de Londres, houve 2 participantes que levantaram dúvidas na utilização do filtro associado ao comprimento, mesmo tendo corretamente configurado para filtrar as ciclovias que não tivessem menos de 50 metros. Como não existe nenhuma indicação se a unidade utilizada no filtro é metros, não tinham a certeza se o valor a indicar seria 50 ou, por exemplo, se teriam de indicar 0.05 caso fosse quilómetros. No entanto, todos completaram corretamente a questão, que envolve configurar múltiplos filtros.

Quanto à Tarefa C5, em que é pedido para desativar todos os filtros e o modo de consulta, 3 participantes começaram por ir a cada filtro configurado e remover a opção para o usar, até se lembrarem da existência do botão para desativar a filtragem. Destes 3 participantes, 1 revelou não ter entendido o que ativar/desativar o modo de consulta alterou na aplicação (tendo sido um dos três que só compreenderam o seu propósito mais tarde, como mencionado na discussão da Questão B2Q1).

Grupo D - Parque Eduardo VII

Na Tabela 6.7 é indicado o desempenho dos participantes nas tarefas e questões do Grupo D, centrado no Parque Eduardo VII, que testa a deslocação no mapa e a utilização das

ferramentas da Rota e da Régua.

Tabela 6.7: Desempenho dos participantes no Grupo D.

Tarefa/Questão	Completo	Completo, com ajuda	Não completo
Tarefa D1	12	0	0
Questão D1Q1	12	0	0
Questão D1Q2	12	0	0
Tarefa D2	12	0	0
Tarefa D3	12	0	0
Tarefa D4	12	0	0
Questão D4Q1	12	0	0
Questão D4Q2	11	1	0
Tarefa D5	12	0	0

Na Tarefa D1 é pedido aos participantes que obtenham a rota da "Praça de Londres" até ao "Parque Eduardo VII" e que sigam o seu trajeto do início ao fim, permitindo testar a obtenção de uma rota e informações sobre esta (Questões D1Q1 e D1Q2) assim como o movimento no mapa. A Tabela 6.8 contém o método preferido pelos participantes para deslocar o mapa ao longo do percurso da rota, com a totalidade a preferir utilizar o arrastamento com o rato em vez das setas do teclado ou os botões de controlo da câmara presentes na barra de ferramentas.

Tabela 6.8: Navegação na Tarefa D1.

Método preferido	Participantes
Rato (<i>panning</i>)	12
Teclado (<i>panning</i>)	0
UI (<i>panning</i>)	0

Também relativo à deslocação ao longo da rota, é perguntado na Questão D1Q3 se concordam com a afirmação "Enquanto segui a rota, achei o movimento suave", sendo a resposta dada numa escala Likert de 5 níveis, com 1 para Discordo Totalmente e 5 para Concordo Totalmente. As respostas a esta questão encontram-se na Figura 6.6, em que 7 participantes concordam com a afirmação e 5 concordam totalmente, com $\bar{x} = 4.42$ e $\sigma = 0.49$, mostrando que a totalidade dos participantes classificou a suavidade do movimento como sendo positiva.

Na Tarefa D4 é pedido para ativarem a régua, de forma a poderem responder às questões D4Q1 e D4Q2, em que é perguntado quais as medições aproximadas do comprimento e do perímetro, respetivamente, da área verde central do Parque Eduardo VII. Estas duas questões foram assim definidas para testar o *edge case* da medição de um perímetro, que implica fechar o ciclo de medições feitas pela régua. Relembrando o seu comportamento

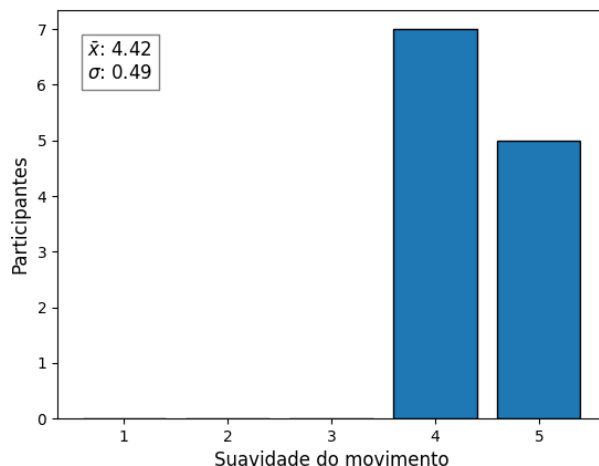


Figura 6.6: D1Q3 - Suavidade do movimento.

brevemente, para efetuar as medições pode-se clicar no mapa para introduzir nós na régua, sendo a soma das distâncias entre estes o valor medido pela régua. Para mover um nó, clica-se em cima dele e arrasta-se para o local pretendido. Para remover, basta apenas clicar em cima dele. Isto implica que para medir um perímetro na implementação atual, é necessário criar um último nó perto do inicial e arrastá-lo para cima deste, de forma a obter a medição correta.

Na Questão D4Q1 em específico, a maioria dos participantes não teve dificuldades em efetuar a medição do comprimento, à exceção de 1 participante. Este clicou no ponto em que iniciou a medição e arrastou o rato para o ponto em que pretendia terminar, sendo que este comportamento fez com que a régua tenha criado o nó inicial e apenas movido para o ponto final, medindo sempre 0m. Após isto, limpou a medição da régua e já tentou clicar nos dois pontos para gerar os nós extremos da reta, completando a questão com sucesso.

Quanto à questão D4Q2, esta já gerou mais dificuldades como era esperado, visto ter sido esse o motivo da sua escolha para o guião de testes. Apenas metade dos participantes (6) é que imediatamente após terem eliminado o nó inicial quando pretendiam completar o ciclo, criaram um nó para arrastar para cima deste. Da restante metade dos participantes, 3 tentaram primeiro encontrar se havia uma combinação com uma tecla que permitisse fechar o ciclo ao impedir a remoção do nó ou fazendo *snap* a este, com as teclas Shift e Control as opções comuns. 1 participante repetiu o processo de clicar no nó inicial, removendo-o não intencionalmente ao longo de cerca de cinco tentativas até finalmente tentar criar um nó e movê-lo para fechar o ciclo. 1 participante tentou compreender melhor como funcionava a régua, limpando completamente a medição e experimentando o que acontecia quando criava um nó, quando o arrastava, até finalmente fechar o ciclo. Por fim, 1 participante conseguiu completar o ciclo por acaso sem ter compreendido como, mas quando foi pedido que repetisse a tarefa não conseguiu reproduzir o que tinha feito,

tendo sido necessário sugerir que criasse um ponto fora da zona a medir e perguntar o que poderia fazer com esse ponto (tendo essa ajuda sido anotada).

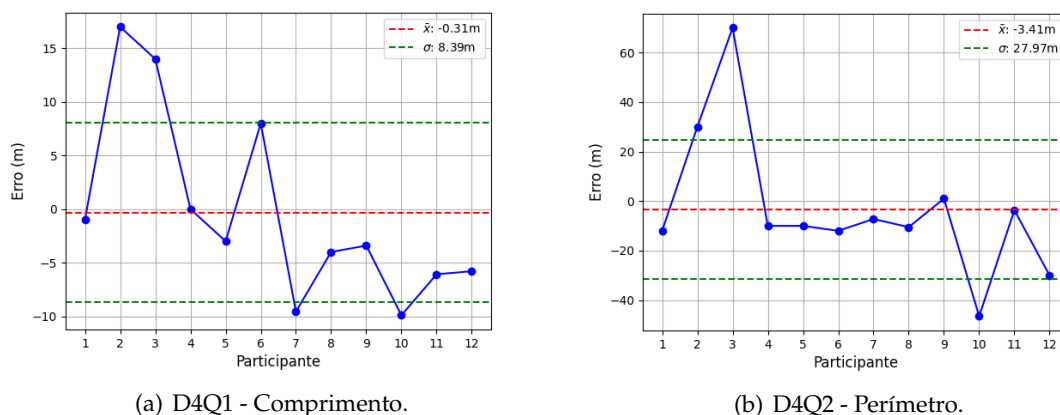


Figura 6.7: Erros nas medições com a régua.

No que diz respeito aos valores medidos pelos participantes, os erros destas medições relativos aos valores definidos como sendo a resposta certa na metodologia, 530m no comprimento e 1170m no perímetro, estão presentes na Figura 6.7. É possível observar que o erro é maior na medição do perímetro (Figura 6.7(b)) do que no comprimento (Figura 6.7(a)), tendo $\bar{x} = -3.41m$ e $\sigma = 27.97m$ em comparação a $\bar{x} = -0.31m$ e $\sigma = 8.39m$, o que é esperado visto que a medição do perímetro implica uma acumulação dos erros ao efetuar quatro medições ao longo do retângulo em comparação com a medição única do comprimento. Adicionalmente, por ter sido pedida a medição aproximada, alguns participantes inseriram um valor arredondado do que foi medido efetivamente pela régua e outros incluíram também parte de passeios adjacentes à área verde, não tendo sido tão precisos nas suas medições.

Grupo E - Castelo de S. Jorge

Na Tabela 6.9 é indicado o desempenho dos participantes nas tarefas do Grupo E, centrado no Castelo de S. Jorge, que serve para fazer uma pequena comparação entre a utilização da aplicação em dispositivos *Desktop* e *Mobile*, assim como o modo de Realidade Aumentada e testar a visualização tridimensional do terreno.

Tabela 6.9: Desempenho dos participantes no Grupo E.

Tarefa/Questão	Completo	Completo, com ajuda	Não completo
Tarefa E1D1	12	0	0
Tarefa E1D2	12	0	0
Tarefa E1D3	12	0	0
Tarefa E1M1	12	0	0

Tarefa E1M2	12	0	0
Tarefa E1M3	12	0	0
Tarefa E1M4	9	3	0

A Tabela 6.10 indica o método preferido pelos participantes para deslocar o mapa da Praça do Comércio para o Castelo de S. Jorge, conforme pedido nas Tarefas E1D1 e E1M1. Embora em ambas plataformas a pesquisa continue a ser o método de preferência da maioria dos participantes (10 em *Desktop* e 8 em *Mobile*), houve um ligeiro aumento na opção pela utilização dos Pontos de Interesse e de *panning* com gestos *touch* para deslocar o mapa. Quando questionados sobre a mudança de preferência, os 2 participantes indicaram não ter optado pela pesquisa por não quererem ter de escrever o endereço no *smartphone*, sendo as outras opções mais simples neste contexto.

Tabela 6.10: Navegação nas Tarefas E1D1 e E1M1.

Método preferido	E1D1	E1M1
Pesquisa	10	8
Pontos de Interesse	1	2
Rato/ <i>touch</i> (<i>panning</i>)	1	2
Teclado (<i>panning</i>)	0	-
UI (<i>panning</i>)	0	0

Nas Tarefas E1D3 e E1M3 é pedido que se utilize os controlos da câmara para observar a visualização 3D do terreno (ativada nas Tarefas E1D2 e E1M2), estando o método preferido pelos participantes indicado na Tabela 6.11. Embora exista a opção de controlar a câmara com o teclado em dispositivos *Desktop*, a totalidade dos participantes optou por usar os botões da interface. Esta opção poderá ser explicada pelo facto de os botões estarem constantemente no ecrã, enquanto não existe indicações das teclas a utilizar se optar pelo teclado, impactando negativamente a sua *discoverability*. Já no *Mobile*, existe uma enorme diferença com a maioria (8) a preferir a utilização de gestos *touch* para controlar a inclinação e orientação do mapa, contrastando com apenas 4 que preferiram continuar a usar os botões da interface. Quando questionados sobre a alteração da sua preferência pelos gestos em vez dos botões, a maioria dos participantes indicou que eram mais intuitivos e o comportamento que esperavam de uma aplicação móvel.

Tabela 6.11: Controlo da câmara nas Tarefas E1D3 e E1M3.

Método preferido	E1D3	E1M3
Gestos <i>touch</i>	-	8
Teclado	0	-
UI	12	4

Neste grupo, a tarefa que suscitou mais dificuldades foi a Tarefa E1M4, em que é pedido para ativar o modo de realidade aumentada e detetar o marcador para observar o terreno, sendo todas relativas à deteção do marcador. Embora tenham todos conseguido efetuar a tarefa, 3 participantes necessitaram de ajuda apenas para posicionar o *smartphone* de modo que este pudesse detetar o marcador de referência que serve de âncora ao mapa, tendo deixado de ser necessária após ter sido detetado. É de realçar que estes 3 participantes tinham indicado não ter qualquer experiência prévia com realidade aumentada, o que poderá explicar a dificuldade na deteção do marcador. Quanto aos restantes, embora tenham conseguido posicionar corretamente o dispositivo, este perdeu o *tracking* algumas vezes com 4 participantes e era notável a diminuição no desempenho a partir do instante em que a câmara era ativada, o que poderá ser devido às limitações do próprio dispositivo, tratando-se de um *smartphone* de gama média com 5 anos, lançado no mesmo ano que o ARCore e no limiar dos requisitos mínimos deste. Vários participantes comentaram que o modo de realidade aumentada era "interessante" e "bastante divertido", mas lamentaram que a experiência tenha sido impactada pelo dispositivo, tendo 1 participante comentado que achava que "a experiência aumentada poderia ser melhor num tablet com mais recursos".

Embora esta comparação seja um bocado limitada, após as tarefas foi averiguada a opinião dos participantes sobre a facilidade de navegação e interação na aplicação *Desktop* e *Mobile* nas Questões E1Q1 e E1Q2, cujos resultados estão presentes na Figura 6.8. As respostas a estas perguntas foram dadas numa escala Likert de 5 níveis, com 1 a indicar que discordavam totalmente de que fosse fácil e 5 a indicar que concordavam totalmente, tendo a aplicação *Desktop* obtido $\bar{x} = 4.92$ e $\sigma = 0.28$ (Figura 6.8(a)) e a *Mobile* obtido $\bar{x} = 4.67$ e $\sigma = 0.47$ (Figura 6.8(b)), havendo uma certa preferência pela primeira.

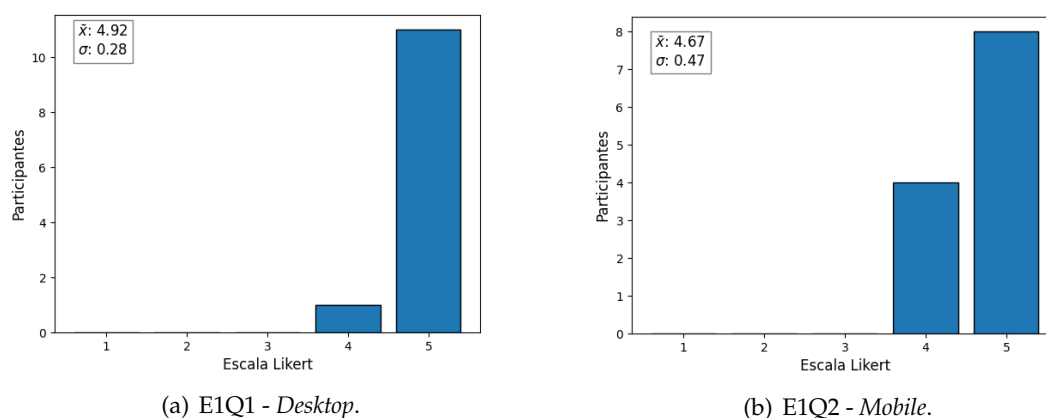


Figura 6.8: Facilidade de navegação e interação.

Na Questão E1Q3 foi perguntado se concordavam que o modo de realidade aumentada era fácil e intuitivo, sendo a resposta também dada numa escala Likert com 5 níveis, estando as respostas presentes na Figura 6.9 com $\bar{x} = 4.67$ e $\sigma = 0.47$. Como já foi mencionado

nos comentários feitos pelos participantes, embora tenham achado o modo intuitivo, o desempenho do dispositivo impactou negativamente a experiência, especialmente na detecção do marcador, o que poderá ter afetado negativamente a sua percepção.

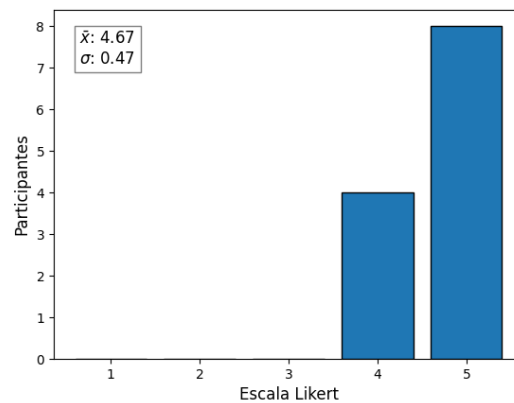
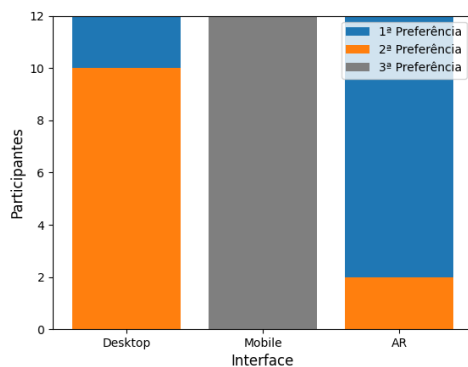
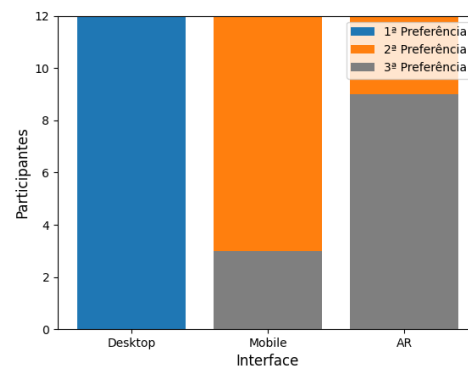


Figura 6.9: E1Q3 - Facilidade e intuitividade do modo RA.



(a) E1Q4 - Visualização do terreno.



(b) E1Q5 - Geral.

Figura 6.10: Interface preferida.

Por fim, nas Questões E1Q4 e E1Q5 foi pedido para que os participantes ordenassem as interfaces pela sua preferência para a visualização do terreno 3D e de um modo geral, respetivamente. Para a visualização do terreno (Figura 6.10(a)), há uma clara preferência pelo modo de realidade aumentada com 10 participantes a colocá-lo como 1ª opção e 2 como 2ª, seguido do *Desktop* com 2 a colocar como 1ª opção e 10 a colocar como 2ª opção, ficando o *Mobile* como 3ª opção de todos os participantes dando como motivos principais o tamanho do ecrã e desempenho do dispositivo para uma experiência similar ao *Desktop*. Já para uma utilização geral da aplicação (Figura 6.10(b)) há uma mudança significativa, passando o *Desktop* a liderar as preferências com a totalidade dos participantes a colocar como 1ª opção, seguido do *Mobile* com 7 a colocar como 2ª opção e 3 como 3ª opção, ficando o modo de RA para último com apenas 3 a colocar como 2ª opção e os restantes 7 como 3ª opção. O seguinte comentário de um participante poderá ajudar a explicar

este contraste: "Não acho que o modo de realidade aumentada é adequado para filtrar os dados. Demasiado complexo para fazer apenas com uma mão enquanto a outra agarra o telemóvel."

Questionário Específico de Usabilidade

Na Figura 6.11 são indicadas as respostas dos participantes ao questionário específico de usabilidade, no qual foram apresentadas 7 afirmações sobre diversas funcionalidades da aplicação serem fáceis e intuitivas de usar. Os participantes podiam responder a cada uma através de uma escala Likert com 5 níveis em que 1 indica que discordam totalmente que seja fácil e intuitiva e 5 indica que concordam totalmente. A partir da soma destes valores foram calculadas as pontuações para cada afirmação, sendo que a pontuação máxima possível é 60 para 12 participantes.

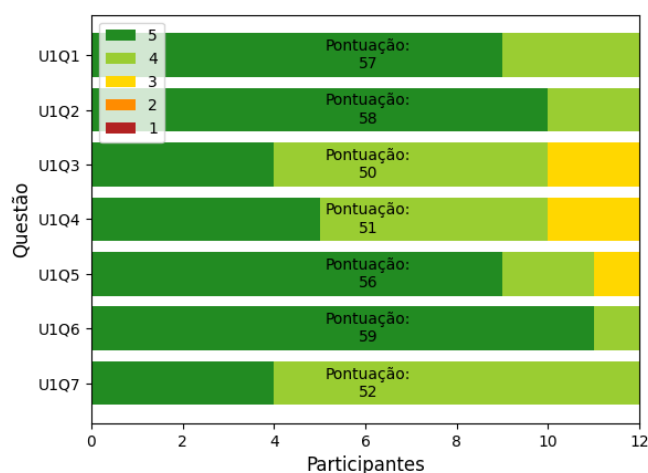


Figura 6.11: Resultados do questionário específico de usabilidade.

Globalmente, as afirmações têm uma classificação positiva, mostrando que de um modo geral a utilização da aplicação poderá ser fácil e intuitiva. Destas afirmações, as que têm uma menor pontuação são as das questões U1Q3 (50 pontos), U1Q4 (51 pontos) e U1Q7 (52 pontos), que correspondem respetivamente à gestão de camadas, aos filtros e à régua. Visto o que foi observado ao longo dos testes, estes valores estão dentro do esperado com a gestão das camadas e configuração de filtros a precisar de ser melhorados para simplificar certos elementos e a régua a necessitar de permitir uma medição mais simples de perímetros.

System Usability Scale

Na Figura 6.12 estão presentes os resultados do questionário SUS para cada um dos participantes, ordenados crescentemente pela sua pontuação desde 72.5 a 100, com $\bar{x} = 84.79$ e $\sigma = 7.60$. Este é um resultado bastante positivo, tendo em conta que uma pontuação

acima de 68 fica acima da média [13] e pode ser considerada como tendo uma boa usabilidade, como já tinha sido mencionado na subsecção 6.3.1 da Metodologia. Este valor médio está representado no gráfico como "SUS médio" e, conforme se pode comprovar, nenhuma das pontuações individuais ficou abaixo deste valor.

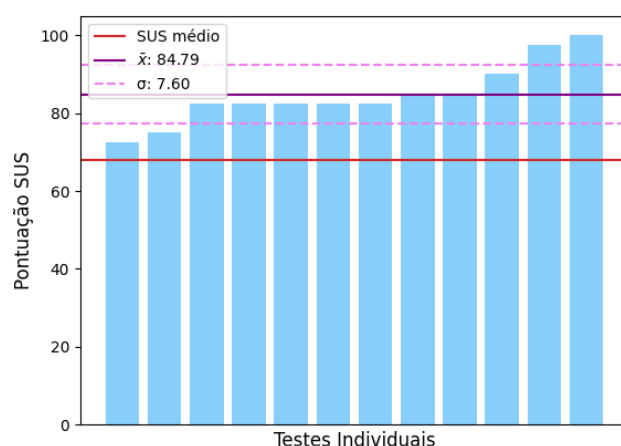


Figura 6.12: Resultados do questionário SUS.

Para ajudar na interpretação de resultados SUS, o autor do questionário, em [55], normalizou as pontuações de modo a poder obter uma classificação de A a F. Assim, estabeleceu-se que seria necessário obter uma pontuação SUS acima de 80.3 para alcançar a classificação A, que representa os 10% melhores resultados e que é também o ponto em que os utilizadores são mais propensos a recomendar o produto testado a alguém. Com isto em mente e considerando que tanto o valor médio desta avaliação como os valores individuais de 10 dos 12 participantes se encontram acima de 80.3, podemos concluir que a aplicação apresenta uma boa usabilidade.

User Experience Questionnaire

Na Figura 6.13 estão representados os resultados do questionário UEQ obtidos através da sua ferramenta oficial de análise de dados [89], que a partir das respostas dadas pelos participantes aos 26 pares de opostos presentes no formulário calcula uma pontuação para as 6 categorias: Atratividade, Transparência, Eficiência, Controlo, Estimulação e Inovação.

As pontuações destas categorias são depois classificadas de acordo com um *benchmark*, criado a partir de um conjunto de dados relativo a 21175 pessoas de 468 estudos de produtos diferentes, para permitir uma interpretação mais informada. No caso de serem classificadas como "Excelente", indica que se encontram no percentil 90, estando dentro dos 10% melhores resultados. Tendo sido este o caso para as seis categorias nesta avaliação, mostra que foi um resultado bastante positivo.

As categorias com maior pontuação foram o Controlo com 2.23 e a Eficiência com 2.19, demonstrando que os participantes acharam que, de um modo geral, a aplicação se

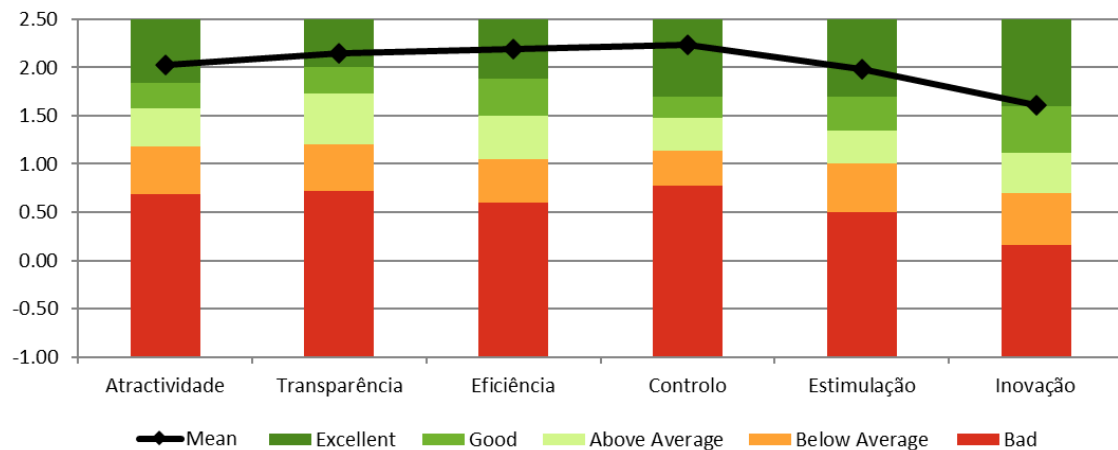


Figura 6.13: Resultados do questionário UEQ [89].

comportava como previsto sem obstruir o que tentavam fazer e que é prática, eficiente e rápida no seu desempenho. Por outro lado, a categoria com menor pontuação, estando mesmo assim no limiar da classificação como "Excelente", foi a Inovação com 1.60, o que está dentro do esperado tendo em conta que foi inspirada em *software SIG* existente e que a experiência de realidade aumentada, que seria o componente que mais poderia contribuir para esta categoria, foi reduzida.

Comentários Finais

No momento final das sessões de teste, os participantes foram convidados a deixar *feedback* geral em adição aos comentários que puderam fazer ao longo da sessão, com 3 participantes (25%) a deixar comentários sobre Aspectos negativos, 8 (67%) sobre Aspectos positivos e 7 (59%) com Sugestões. Estas respostas são aqui reproduzidas:

Aspectos negativos

- "Um pouco difícil de perceber quais os filtros que estão ativos em cada instante."
- "A interface poderia estar ligeiramente melhor organizada quanto aos filtros, e a nível visual a UI necessita alguns melhoramentos."
- "Botão da bússola está sempre alinhado para Norte."

Aspectos positivos

- "A visualização do terreno com RA foi interessante, parecia que estava a ver a vista de um drone. Gostei do design da aplicação e dos modelos em geral."
- "Apresenta bastantes opções/funcionalidades que usualmente não estão presentes ou então é claro que existem."

- "É muito inovadora e tem muita flexibilidade de opções que permitem explorar bem Lisboa. Usaria com frequência."
- "Tem potencial para ser um instrumento de grande utilidade para vários serviços autárquicos (urbanismo, rodoviário, manutenção urbana, etc..)"
- "O modo de consulta tem uma quantidade significativa de informação disponível, e é bastante impressionante. A régua também é muito interessante de usar"
- "Funcionalidades de pesquisa, filtragem de camadas e queries, régua e rota muito intuitivas; Zoom in/out, inclinação e demais idem, usabilidade simples e intuitiva; Render dos objetos rápido; Interface muito clean e intuitiva"
- "A aplicação é de rápido entendimento"

Sugestões

- "Ter um indicador dos filtros ativos. Ter uma explicação no ecrã sobre como detetar o marcador de RA. Podia haver uma maior separação entre a pesquisa de locais e a lista dos pontos de interesse."
- "Régua poderia colocar marcadores sobrepostos"
- "Senti que quando o cursor passa pelos comandos do controlo da câmara, deveria dar ao utilizador a informação para que serve cada comando."
- "Deveria ter um tooltip nos botões da câmara, tornar mais óbvio que se pode clicar no modo query com o cursor, navegação nos filtros por ordem alfabética"
- "Podia ter uma janelinha com ajuda"
- "Opção de poder seleccionar o idioma a utilizar na aplicação (limitado ao inglês)"
- "Um 'joystick' para controlar a câmara como no Google Earth."

De uma forma geral, o *feedback* sobre a aplicação é positivo, sendo, no entanto, também perceptíveis as dificuldades que foram sentidas e discutidas nos grupos respetivos. Do lado negativo, o destaque vai para a organização da interface relativamente às camadas e filtros. No entanto, existem vários aspetos positivos destacados pelos participantes, como o modo de consulta, a régua (mesmo tendo em conta as dificuldades que encontraram) ou a visualização do terreno em realidade aumentada.

Também o design geral da aplicação foi bem recebido, com os participantes a elogiar a sua inovação, flexibilidade e potencial utilidade para vários serviços municipais. Embora tenham sido feitas algumas sugestões de melhoria, como indicadores para filtros ativos, *tooltips* nos controlos da câmara ou o cursor no modo de consulta, em geral, expressaram entusiasmo e vontade em usar a aplicação com frequência, enfatizando a sua natureza intuitiva e exploratória.

Por fim, é destacado o seguinte comentário da participante com experiência profissional na utilização de *software SIG*, podendo ser considerada como *expert*:

"Muito, muito, muito bom! A aplicação tem detalhes que o próprio utilizador pode pesquisar e explorar de modo mais detalhado, mais pormenorizado. Senti a aplicação muito intuitiva, ficaria o dia a explorar esta aplicação. É possível qualquer pessoa pesquisar e explorar esta aplicação. Sou utilizadora diária de SIG desde 2014, que são ferramentas cruciais para o meu trabalho como arquiteta afeta ao Departamento de Urbanismo/Divisão de Gestão Urbanística, onde utilizo ferramentas como esta para analisar e realizar apreciações técnicas das várias operações urbanísticas apresentadas na autarquia, sem as quais nem os utilizadores dentro e fora das autarquias podem trabalhar ou simplesmente obter informação."

6.3.3 Limitações da avaliação

São agora discutidas algumas das limitações no processo de avaliação de usabilidade, esclarecendo as restrições e fatores que podem ter influenciado os resultados obtidos, assim como quaisquer conclusões derivadas destes.

1. Número limitado de participantes:

Uma das limitações principais da avaliação é o número relativamente pequeno de participantes, apenas 12 pessoas no total. Esta limitação poderá afetar a capacidade de generalizar as conclusões obtidas a partir dos resultados, uma vez que há maior probabilidade de se tirarem conclusões erróneas devido ao número limitado de resultados, o que aumenta o viés em relação a cada resposta individual. Portanto, uma maior amostra de participantes poderia contribuir para evitar este problema, assim como revelar *insights* adicionais.

2. *Background* dos participantes:

Outra limitação surge do *background* dos participantes. A maioria dos participantes são de um *background* de Informática, o que poderá influenciar a sua forma de interagir com o sistema. Adicionalmente, a maioria não tinha experiência com *software SIG*, com uma porção cuja experiência é apenas em contexto académico. Embora tenha sido possível realizar a avaliação com 1 participante cujo *background* não é relacionado com a área de Informática, estando inserida no contexto alvo dentro de uma autarquia e com experiência profissional de vários anos com *software SIG*, este número fica muito aquém do ideal, o que também torna a avaliação sujeita a *overfitting*. No entanto, também houve 2 participantes que não vêm de um contexto informático, nem têm experiência com *SIG*, o que implica que puderam servir como uma pequena amostra para o comportamento de utilizadores sem qualquer

experiência relevante, sendo que uma aplicação que para estes tenha boa usabilidade, potencialmente também o será para utilizadores com experiência.

3. Sessão única de avaliação:

A avaliação envolveu sessões únicas em que os participantes interagiram com a aplicação sem conhecimento ou instruções prévias relativas ao seu funcionamento. Enquanto isto é útil para testar a usabilidade da aplicação e quão intuitiva é, não permite captar mudanças no comportamento e preferências dos participantes. Seria interessante realizar uma segunda sessão *follow-up* com os mesmos participantes, após terem alguma familiaridade com a aplicação na primeira sessão (que poderia ser primariamente para exploração), para observar se os problemas de usabilidade encontrados são persistentes ou se estão primariamente relacionados com a exploração inicial, podendo até ser encontrados outros que não tenham sido detetados. Esta segunda sessão também permitiria que fossem implementadas algumas das melhorias sugeridas e medir a sua eficácia, num processo de desenvolvimento mais iterativo.

4. Comparação *Desktop* vs *Mobile*:

A utilização de um *smartphone* de gama média de 2018 teve um impacto notável na avaliação, especialmente no que diz respeito ao modo de realidade aumentada, dificultando a tarefa dos participantes detetarem o marcador de referência. Como está no limite dos requisitos mínimos do ARCore, o seu desempenho durante a deteção do marcador fica limitado, tendo perdido o *tracking* algumas vezes para vários participantes, mesmo com estes a posicionar corretamente o dispositivo. Além disso, parcialmente pelas limitações do *smartphone* e também para evitar uma duração demasiado longa e desconfortável da sessão de teste, as tarefas realizadas no Grupo E em que é comparada a utilização em *Desktop* e *Mobile* foram reduzidas, o que coloca a aplicação móvel numa clara desvantagem neste teste. É de notar que 1 participante comentou que teria sido mais interessante se a comparação tivesse sido feita com um *tablet* com mais recursos, para aproveitar o seu desempenho e um maior tamanho de ecrã.

Embora a avaliação efetuada possa ter dado *insights* valiosos relativamente à usabilidade da aplicação, é importante reconhecer estas limitações. Abordar estas limitações em avaliações futuras, como aumentar o número e diversidade da população participante, realizar sessões *follow-up*, tal como uma melhor comparação entre *Desktop* e *Mobile*, poderão produzir uma compreensão mais abrangente da experiência de utilizador.

6.4 Resumo

Neste capítulo foi feita uma análise abrangente da aplicação, com três fases de avaliação. Começou com uma avaliação crítica da extensibilidade da aplicação, uma breve discussão da capacidade para adaptar a aplicação a requisitos futuros e possíveis alterações. De seguida, foi feita a avaliação de desempenho, dividida em dois grupos distintos relativos ao carregamento de uma área e ao desempenho da aplicação durante o movimento no mapa. Os resultados destes mostraram que a aplicação aparenta ser eficiente e tem um bom desempenho, embora tenham sido encontrados pontos de melhoria, especialmente para níveis de ampliação do mapa reduzidos dada a quantidade de objetos presentes. Finalmente, a fase final de avaliação consistiu nos testes de usabilidade com utilizadores, tendo sido apresentada a metodologia utilizada nas sessões de teste, os resultados obtidos nesta que foram bastante positivos e indicaram que poderá ter uma boa usabilidade (embora também com pontos de melhoria detetados como na gestão de camadas e filtros e na régua), assim como potenciais limitações desta avaliação. Resumindo, esta avaliação forneceu uma visão abrangente do estado atual da aplicação, destacando tanto os seus pontos fortes como os que ainda carecem de melhorias, sendo que os resultados obtidos permitem ter uma base sólida para as melhorias a efetuar no futuro.

CONCLUSÕES E TRABALHO FUTURO

Neste capítulo são apresentadas as conclusões do trabalho efetuado nesta dissertação, com uma síntese do que foi feito e a sua relação aos objetivos a cumprir e contribuições esperadas. Adicionalmente, são descritas algumas sugestões e ideias a considerar num trabalho futuro.

7.1 Conclusões

Esta dissertação teve como propósito principal a integração de dados georreferenciados no motor de jogos Unity. Esta etapa inicial representa o ponto de partida no desenvolvimento de uma ferramenta colaborativa de realidade aumentada de suporte à decisão, ferramenta esta que parte de um desafio lançado pelo Laboratório de Dados Urbanos de Lisboa (LxDataLab). As etapas seguintes incluem a criação de visualizações de dados, a interação com o modelo virtual em Realidade Aumentada e a sincronização de múltiplos clientes, tendo sido feita uma breve exploração das duas primeiras.

Deste modo foram definidas três contribuições principais que se pretendiam alcançar nesta etapa com a realização desta dissertação. Estas são a criação de uma *framework* de código aberto que permita a integração de dados georreferenciados no Unity, o desenvolvimento de um **protótipo** a partir desta que permita visualizar e analisar certos conjuntos de dados e a **avaliação** de usabilidade e desempenho deste.

De um modo geral, estas três contribuições foram alcançadas com sucesso. A *framework* de código aberto desenvolvida encontra-se disponível em <https://github.com/MagicalBlob/GeoTileQuery>, sendo distribuída com a licença GNU GPL. Esta permite a integração de dados georreferenciados no Unity, o que é demonstrável através do protótipo desenvolvido que permite visualizar e analisar diversos conjuntos de dados.

A avaliação efetuada ao protótipo, para além de o validar, por sua vez também valida a *framework* com que foi desenvolvido. A avaliação crítica discutiu a extensibilidade e adaptabilidade a diferentes requisitos futuros. No que toca à avaliação de desempenho, esta mostrou que a aplicação é eficiente e tem um bom desempenho, embora tenham sido identificados possíveis pontos de melhoria, como na representação de níveis de ampliação

reduzidos que podem conter um enorme número de objetos. Por fim, a avaliação de usabilidade demonstrou resultados positivos nos testes com utilizadores, apontando globalmente para que seja intuitiva e que tenha boa usabilidade, embora também tenham sido anotados áreas de melhoria como na gestão de camadas e filtros, assim como foram apontadas algumas potenciais limitações da própria avaliação de usabilidade.

Resumindo, pode-se considerar que o objetivo e contribuições esperadas que foram definidos para esta dissertação foram concluídos com sucesso, embora continue a haver espaço para melhorias.

7.2 Trabalho futuro

Por fim, são agora apresentadas algumas sugestões de trabalho futuro, de modo a melhorar ou complementar o que foi realizado.

A primeira diz respeito aos resultados dos testes de usabilidade, nos quais foram encontrados alguns pontos que podem ser melhorados, como a gestão de camadas e filtros que poderia ser unida e simplificada, melhoria no controlo da régua, *tooltips* nos controlos da câmara, suporte para outros idiomas e clarificação do modo de consulta. Idealmente, a realização de mais sessões de testes de usabilidade poderá ajudar nisto ao permitir que seja feito um processo de desenvolvimento mais iterativo.

Continuando com resultados provenientes da avaliação, embora da de desempenho, será uma melhor gestão do nível de detalhe, de forma a simplificar e diminuir a quantidade de objetos representados para níveis de ampliação reduzidos que cobrem uma maior área. Também seria interessante testar formas de melhorar o carregamento dos dados, como o *caching* dos pedidos tendo a utilização de memória em consideração ou aglomerar os dados relativos a todas as camadas num único *bundle* para diminuir o número de pedidos por *tile* e, potencialmente, o *overhead* associado a estes. No entanto, esta estratégia requiere também o desenvolvimento de um servidor próprio para o efeito, não podendo ser utilizadas as mesmas ferramentas.

Outra sugestão que também envolveria o desenvolvimento de um servidor próprio seria o suporte para integrar dados dinâmicos, como por exemplo o trânsito ou a posição de veículos municipais em tempo real. Atualmente isto pode ser parcialmente alcançado se os dados forem alterados, o servidor reiniciado e a área recarregada mas, como a aplicação assume que são estáticos, este processo não é automático e implicaria duplicação de pedidos de dados que se manteriam constantes, em vez de apenas serem transmitidos os *deltas* decorrentes da atualização dos dados que diminuiria também a *bandwidth* desperdiçada.

Uma última sugestão seria a criação e edição de camadas de dados, que poderia ser útil na etapa de sincronização de múltiplos utilizadores para permitir a sua colaboração no planeamento de certos cenários. Um exemplo seria num cenário em que se pretendesse determinar quais as estradas onde limitar o trânsito, onde colocar os operacionais no terreno, entre outros.

BIBLIOGRAFIA

- [1] *3D Software for Architecture, Engineering & Construction* | Unity. URL: <https://unity.com/solutions/architecture-engineering-construction> (acedido em 2023-09-10) (ver p. 35).
- [2] *About Mapbox Maps* | Mapbox. URL: <https://www.mapbox.com/about/maps/#faq-elevation-api> (acedido em 2023-01-20) (ver pp. 46, 67).
- [3] *Adaptive Subdivision of 3D Tiles* – Cesium. URL: <https://cesium.com/blog/2017/08/11/adaptive-subdivision-of-3d-tiles/> (acedido em 2023-09-11) (ver p. 31).
- [4] M. Adcock, M. Hutchins e C. Gunn. «Haptic collaboration with augmented reality». Em: *ACM SIGGRAPH 2004 Posters on - SIGGRAPH '04*. New York, New York, USA: ACM Press, 2004, p. 41. ISBN: 1581138962. DOI: [10.1145/1186415.1186463](https://doi.org/10.1145/1186415.1186463). URL: <http://portal.acm.org/citation.cfm?doid=1186415.1186463> (ver p. 26).
- [5] *AR+ is now available in Pokémon GO!* URL: <https://pokemongolive.com/post/arplus/> (acedido em 2023-09-09) (ver p. 18).
- [6] *Augmented Reality - Apple*. URL: <https://www.apple.com/augmented-reality/> (acedido em 2023-01-18) (ver p. 21).
- [7] R. Azuma. «Tracking requirements for augmented reality». Em: *Communications of the ACM* 36.7 (1993-07), pp. 50–51. ISSN: 00010782. DOI: [10.1145/159544.159581](https://doi.org/10.1145/159544.159581). URL: <http://portal.acm.org/citation.cfm?doid=159544.159581> (ver p. 16).
- [8] R. Azuma e G. Bishop. «Improving static and dynamic registration in an optical see-through HMD». Em: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*. New York, New York, USA: ACM Press, 1994, pp. 197–204. ISBN: 0897916670. DOI: [10.1145/192161.192199](https://doi.org/10.1145/192161.192199). URL: <http://portal.acm.org/citation.cfm?doid=192161.192199> (ver p. 16).
- [9] R. Azuma et al. «Recent advances in augmented reality». Em: *IEEE Computer Graphics and Applications* 21.6 (2001), pp. 34–47. ISSN: 02721716. DOI: [10.1109/38.963459](https://doi.org/10.1109/38.963459). URL: <http://ieeexplore.ieee.org/document/963459/> (ver p. 14).

- [10] H. Bay, T. Tuytelaars e L. Van Gool. «SURF: Speeded Up Robust Features». Em: *Computer Vision – ECCV 2006*. Berlin, Heidelberg: Springer, 2006, pp. 404–417. DOI: [10.1007/11744023_32](https://doi.org/10.1007/11744023_32). URL: http://link.springer.com/10.1007/11744023_32 (ver p. 22).
- [11] M. Billinghurst, A. Clarke e G. Lee. «A Survey of Augmented Reality». Em: *Foundations and Trends® in Human–Computer Interaction* 8.2-3 (2015), pp. 73–272. ISSN: 1551-3955. DOI: [10.1561/11000000049](https://doi.org/10.1561/11000000049). URL: <http://dx.doi.org/10.1561/11000000049> (ver pp. 25, 26).
- [12] J. Brooke. «SUS: A quick and dirty usability scale». Em: *Usability Eval. Ind.* 189 (1995-11) (ver pp. 103, 141).
- [13] J. Brooke. «SUS: A Retrospective». Em: *Journal of Usability Studies* 8.2 (2013-02), pp. 29–40 (ver pp. 103, 117).
- [14] H. Butler et al. *The GeoJSON Format*. RFC 7946. 2016-08. DOI: [10.17487/RFC7946](https://doi.org/10.17487/RFC7946). URL: <https://datatracker.ietf.org/doc/html/rfc7946> (ver pp. 11, 77).
- [15] H. Butler et al. *The GeoJSON Format Specification*. Rel. téc. 2008-06. URL: <https://geojson.org/geojson-spec> (ver p. 12).
- [16] I. Buyuksalih et al. «3D Modelling and Visualization based on the Unity game engine – Advantages and Challenges». Em: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-4/W4 (2017), pp. 161–166. DOI: [10.5194/isprs-annals-IV-4-W4-161-2017](https://doi.org/10.5194/isprs-annals-IV-4-W4-161-2017). URL: <https://isprs-annals.copernicus.org/articles/IV-4-W4/161/2017/> (ver p. 28).
- [17] *Calculate distance and bearing between two Latitude/Longitude points using haversine formula*. URL: <https://www.movable-type.co.uk/scripts/latlong.html> (acedido em 2023-09-04) (ver p. 84).
- [18] M. Calonder et al. «BRIEF: Binary Robust Independent Elementary Features». Em: *Computer Vision – ECCV 2010*. Berlin, Heidelberg: Springer, 2010, pp. 778–792. DOI: [10.1007/978-3-642-15561-1_56](https://doi.org/10.1007/978-3-642-15561-1_56). URL: http://link.springer.com/10.1007/978-3-642-15561-1_56 (ver p. 22).
- [19] *Cesium for Unity: Photorealistic 3D Tiles from Google Maps Platform – Cesium*. URL: <https://cesium.com/learn/unity/unity-photorealistic-3d-tiles/> (acedido em 2023-09-11) (ver p. 31).
- [20] K.-T. Chang. *Introduction to Geographic Information Systems*. Ninth Edition. McGraw-Hill Education, 2019. ISBN: 978-1-259-92964-9 (ver pp. 7–9).
- [21] *CHANGELOG | Mapbox SDK for Unity*. URL: <https://docs.mapbox.com/mapbox-unity-sdk/docs/05-changelog.html> (acedido em 2023-09-11) (ver p. 30).
- [22] A. Congès et al. «Crisis Management Exercises in Virtual Reality». Em: *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, 2020, pp. 87–92. DOI: [10.1109/VRW50115.2020.00022](https://doi.org/10.1109/VRW50115.2020.00022) (ver p. 28).

- [23] *Elevation – Polar Geospatial Center*. URL: <https://www.pgc.umn.edu/data/elevation/> (acedido em 2023-09-07) (ver p. 10).
- [24] *ESRI Shapefile Technical Description - An ESRI White Paper*. Rel. téc. ESRI, 1998-07. URL: <http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf> (acedido em 2023-09-08) (ver p. 12).
- [25] *EU-DEM v1.1 — Copernicus Land Monitoring Service*. URL: <https://land.copernicus.eu/imagery-in-situ/eu-dem/eu-dem-v1.1?tab=metadata> (acedido em 2023-01-20) (ver pp. 46, 67).
- [26] S. Feiner et al. «A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment». Em: *Personal Technologies 1.4* (1997-12), pp. 208–217. ISSN: 0949-2054. DOI: [10.1007/BF01682023](https://doi.org/10.1007/BF01682023). URL: <http://link.springer.com/10.1007/BF01682023> (ver p. 17).
- [27] M. Fiala. «Comparing ARTag and ARToolkit plus fiducial marker systems». Em: *IREE International Workshop on Haptic Audio Visual Environments and their Applications, 2005*. Ottawa, Ontario, Canada: IEEE, 2005, pp. 147–152. ISBN: 0-7803-9376-7. DOI: [10.1109/HAVE.2005.1545669](https://doi.org/10.1109/HAVE.2005.1545669). URL: <http://ieeexplore.ieee.org/document/1545669/> (ver p. 22).
- [28] *File geodatabases — ArcGIS Pro | Documentation*. URL: <https://pro.arcgis.com/en/pro-app/latest/help/data/geodatabases/manage-file-gdb/file-geodatabases.htm> (acedido em 2023-09-08) (ver p. 14).
- [29] *Filtering and modification | ArcGIS Maps SDK for Unity | ArcGIS Developers*. URL: <https://developers.arcgis.com/unity/filtering-and-modification/> (acedido em 2023-09-11) (ver p. 29).
- [30] *First Contentful Paint (FCP) | web.dev*. URL: <https://web.dev/fcp/> (acedido em 2023-09-15) (ver p. 91).
- [31] S. S. Fisher et al. «Virtual environment display system». Em: *Proceedings of the 1986 workshop on Interactive 3D graphics - SI3D '86*. New York, New York, USA: ACM Press, 1987, pp. 77–87. ISBN: 0897912284. DOI: [10.1145/319120.319127](https://doi.org/10.1145/319120.319127). URL: <http://portal.acm.org/citation.cfm?doid=319120.319127> (ver p. 16).
- [32] M. Fjeld e B. M. Voegtli. «Augmented Chemistry: An Interactive Educational Workbench». Em: *Proceedings of the 1st International Symposium on Mixed and Augmented Reality (ISMAR '02)*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 259–. ISBN: 0769517811. URL: <http://dl.acm.org/citation.cfm?id=850976.854940> (ver pp. 26, 27).
- [33] *Fundamental concepts | ARCore | Google for Developers*. URL: <https://developers.google.com/ar/develop/fundamentals> (acedido em 2023-09-10) (ver p. 36).

- [34] T. A. Furness. «The Super Cockpit and its Human Factors Challenges». Em: *Proceedings of the Human Factors Society Annual Meeting* 30.1 (1986-09), pp. 48–52. ISSN: 0163-5182. DOI: [10.1177/154193128603000112](https://doi.org/10.1177/154193128603000112). URL: <http://journals.sagepub.com/doi/10.1177/154193128603000112> (ver p. 16).
- [35] *Gaming Services Transition Guide | Gaming Maps Platform Gaming Services | Google for Developers*. URL: <https://developers.google.com/maps/documentation/gaming/transition-guide> (acedido em 2023-09-11) (ver p. 31).
- [36] P. D. Gelautz. «Development and demonstration of a high-performance Gaussian puff model for nuclear emergency training scenarios in Unity game engine». University of Ontario Institute of Technology, 2021-08. URL: <https://hdl.handle.net/10155/1341> (ver p. 29).
- [37] *Geographic vs Projected Coordinate Systems*. URL: https://www.esri.com/arcgis-blog/products/arcgis-pro/mapping/gcs_vs_pcs/ (acedido em 2023-09-06) (ver p. 8).
- [38] *Global elevation data | Mapbox*. URL: <https://www.mapbox.com/blog/global-elevation-data> (acedido em 2023-03-12) (ver p. 68).
- [39] *GSP 270: Introduction to Coordinate Systems*. URL: <https://gsp.humboldt.edu/olm/Lessons/GIS/03%20Projections/IntroductionToCoordinateSystems1.html> (acedido em 2023-09-06) (ver p. 7).
- [40] C. Harrison, H. Benko e A. D. Wilson. «OmniTouch». Em: *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. New York, New York, USA: ACM Press, 2011, p. 441. ISBN: 9781450307161. DOI: [10.1145/2047196.2047255](https://doi.org/10.1145/2047196.2047255). URL: <http://dl.acm.org/citation.cfm?doid=2047196.2047255> (ver p. 19).
- [41] O. Hilliges et al. «HoloDesk». Em: *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*. New York, New York, USA: ACM Press, 2012, p. 2421. ISBN: 9781450310154. DOI: [10.1145/2207676.2208405](https://doi.org/10.1145/2207676.2208405). URL: <http://dl.acm.org/citation.cfm?doid=2207676.2208405> (ver pp. 20, 27).
- [42] R. Hirayama et al. «A volumetric display for visual, tactile and audio presentation using acoustic trapping». Em: *Nature* 575.7782 (2019-11), pp. 320–323. ISSN: 0028-0836. DOI: [10.1038/s41586-019-1739-5](https://doi.org/10.1038/s41586-019-1739-5). URL: <http://www.nature.com/articles/s41586-019-1739-5> (ver p. 19).
- [43] *Instinctual interactions - Mixed Reality | Microsoft Learn*. URL: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/interaction-fundamentals> (acedido em 2023-07-11) (ver p. 27).
- [44] *Introducing Apple Vision Pro: Apple's first spatial computer - Apple*. URL: <https://www.apple.com/newsroom/2023/06/introducing-apple-vision-pro/> (acedido em 2023-09-09) (ver p. 18).

- [45] *Is Mapbox planning to upgrade to ARFoundation?* URL: <https://github.com/mapbox/mapbox-unity-sdk/issues/1747#issuecomment-791716026> (acedido em 2023-09-11) (ver p. 30).
- [46] H. Ishii e B. Ullmer. «Tangible bits: towards seamless interfaces between people, bits and atoms». Em: *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*. March. New York, New York, USA: ACM Press, 1997, pp. 234–241. ISBN: 0897918029. DOI: [10.1145/258549.258715](https://doi.org/10.1145/258549.258715). URL: <http://portal.acm.org/citation.cfm?doid=258549.258715> (ver p. 26).
- [47] H. Kato et al. «Virtual object manipulation on a table-top AR environment». Em: *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*. IEEE, 2000, pp. 111–119. ISBN: 0-7695-0846-4. DOI: [10.1109/ISAR.2000.880934](https://doi.org/10.1109/ISAR.2000.880934). URL: <http://ieeexplore.ieee.org/document/880934/> (ver pp. 17, 22, 26).
- [48] H. Kato e M. Billinghurst. «Marker tracking and HMD calibration for a video-based augmented reality conferencing system». Em: *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*. IEEE Comput. Soc, 1999, pp. 85–94. ISBN: 0-7695-0359-4. DOI: [10.1109/IWAR.1999.803809](https://doi.org/10.1109/IWAR.1999.803809). URL: <http://ieeexplore.ieee.org/document/803809/> (ver pp. 17, 22).
- [49] *Key features | ArcGIS Maps SDK for Unity | ArcGIS Developers*. URL: <https://developers.arcgis.com/unity/key-features/> (acedido em 2023-09-11) (ver p. 30).
- [50] K. Kiyokawa, H. Takemura e N. Yokoya. «SeamlessDesign for 3D object creation». Em: *IEEE Multimedia* 7.1 (2000), pp. 22–33. ISSN: 1070986X. DOI: [10.1109/93.839308](https://doi.org/10.1109/93.839308). URL: <http://ieeexplore.ieee.org/document/839308/> (ver p. 26).
- [51] T. Lee e T. Hollerer. «Handy AR: Markerless Inspection of Augmented Reality Objects Using Fingertip Tracking». Em: *2007 11th IEEE International Symposium on Wearable Computers*. IEEE, 2007-10, pp. 1–8. ISBN: 978-1-4244-1452-9. DOI: [10.1109/ISWC.2007.4373785](https://doi.org/10.1109/ISWC.2007.4373785). URL: <http://ieeexplore.ieee.org/document/4373785/> (ver p. 27).
- [52] J. M. Lourenço. *The NOVAthesis L^AT_EX Template User's Manual*. NOVA University Lisbon. 2021. URL: <https://github.com/joaomlourenco/novathesis/raw/main/template.pdf> (ver p. i).
- [53] D. Lowe. «Object recognition from local scale-invariant features». Em: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. IEEE, 1999, 1150–1157 vol.2. ISBN: 0-7695-0164-8. DOI: [10.1109/ICCV.1999.790410](https://doi.org/10.1109/ICCV.1999.790410). URL: <http://ieeexplore.ieee.org/document/790410/> (ver pp. 22, 23).
- [54] *Maps | ArcGIS Maps SDK for Unity | ArcGIS Developers*. URL: <https://developers.arcgis.com/unity/maps/> (acedido em 2023-09-11) (ver p. 29).

- [55] *Measuring Usability with the System Usability Scale (SUS) – MeasuringU*. URL: <https://measuringu.com/sus/> (acedido em 2023-09-21) (ver p. 117).
- [56] P. Milgram et al. «Augmented reality: a class of displays on the reality-virtuality continuum». Em: *Proc. SPIE 2351, Telemanipulator and Telepresence Technologies*. Ed. por H. Das. 1995-12, pp. 282–292. DOI: [10.1117/12.197321](https://doi.org/10.1117/12.197321). URL: <http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=981543> (ver p. 14).
- [57] P. Mistry e P. Maes. «SixthSense». Em: *ACM SIGGRAPH ASIA 2009 Sketches on - SIGGRAPH ASIA '09*. New York, New York, USA: ACM Press, 2009, p. 1. DOI: [10.1145/1667146.1667160](https://doi.org/10.1145/1667146.1667160). URL: <http://portal.acm.org/citation.cfm?doid=1667146.1667160> (ver p. 19).
- [58] MIT | *User Interface Design & Implementation - User Testing*. URL: <https://web.mit.edu/6.813/www/sp18/classes/12-user-testing/> (acedido em 2023-08-07) (ver p. 97).
- [59] M. Mohring, C. Lessig e O. Bimber. «Video See-Through AR on Consumer Cell-Phones». Em: *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, 2004, pp. 252–253. ISBN: 0-7695-2191-6. DOI: [10.1109/ISMAR.2004.63](https://doi.org/10.1109/ISMAR.2004.63). URL: <http://ieeexplore.ieee.org/document/1383062/> (ver p. 17).
- [60] *Multi-platform Handheld AR in 2018 | Unity Blog*. URL: <https://blog.unity.com/technology/multi-platform-handheld-ar-in-2018-part-1> (acedido em 2023-09-11) (ver p. 30).
- [61] A. Olwal, H. Benko e S. Feiner. «SenseShapes: using statistical geometry for object selection in a multimodal augmented reality». Em: *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings*. IEEE Comput. Soc, 2006, pp. 300–301. ISBN: 0-7695-2006-5. DOI: [10.1109/ISMAR.2003.1240730](https://doi.org/10.1109/ISMAR.2003.1240730). URL: <http://ieeexplore.ieee.org/document/1240730/> (ver p. 27).
- [62] Open Geospatial Consortium. *OGC City Geography Markup Language (CityGML) 3.0 Conceptual Model Users Guide*. Ed. por C. Heazel. Versão 1.0. OGC® Document Reference: 20-066. 2021-09. URL: <http://www.opengis.net/doc/UG/CityGML-user-guide/3.0> (acedido em 2023-09-08) (ver p. 13).
- [63] PGIL: *Agregar dados e informação para servir a cidade de Lisboa*. URL: <https://smart-cities.pt/smn/nec-plataforma-lisboa-2601/> (acedido em 2023-02-08) (ver p. 2).
- [64] A. Pires et al. «Combinando mapas de papel e smartphones na exploração do património cultural». Em: *Proceedings of 26º Encontro Português de Computação Gráfica e Interação (EPCGI 2016)*. Universidade da Beira Interior, 2016, pp. 155–159. ISBN: 978-1-5090-5386-5. URL: <http://hdl.handle.net/10362/21564> (ver p. 28).

- [65] *Planning a Usability Test* | Usability.gov. URL: <https://www.usability.gov/how-to-and-tools/methods/planning-usability-testing.html> (acedido em 2023-08-07) (ver p. 97).
- [66] *PT-TM06/ETRS89* | Direção-Geral do Território. URL: <https://www.dgterritorio.gov.pt/geodesia/sistemas-referencia/portugal-continental/PT-TM06-ETRS89> (acedido em 2023-09-06) (ver p. 8).
- [67] A. Rafiee et al. «Developing a wind turbine planning platform: Integration of “sound propagation model–GIS-game engine” triplet». Em: *Environmental Modelling & Software* 95 (2017), pp. 326–343. ISSN: 1364-8152. DOI: [10.1016/j.envsoft.2017.06.019](https://doi.org/10.1016/j.envsoft.2017.06.019). URL: <https://www.sciencedirect.com/science/article/pii/S1364815216309720> (ver p. 28).
- [68] R. Raskar et al. «Shader Lamps: Animating Real Objects With Image-Based Illumination». Em: *Rendering Techniques*. Vienna: Springer, 2001, pp. 89–102. DOI: [10.1007/978-3-7091-6242-2_9](https://doi.org/10.1007/978-3-7091-6242-2_9). URL: http://link.springer.com/10.1007/978-3-7091-6242-2_9 (ver p. 19).
- [69] J. Rekimoto e Y. Ayatsuka. «CyberCode: designing augmented reality environments with visual tags». Em: *Proceedings of DARE 2000 on Designing augmented reality environments - DARE '00*. New York, New York, USA: ACM Press, 2000, pp. 1–10. DOI: [10.1145/354666.354667](https://doi.org/10.1145/354666.354667). URL: <http://portal.acm.org/citation.cfm?doid=354666.354667> (ver p. 16).
- [70] J. Rekimoto e K. Nagao. «The world through the computer». Em: *Proceedings of the 8th annual ACM symposium on User interface and software technology - UIST '95*. New York, New York, USA: ACM Press, 1995, pp. 29–36. ISBN: 089791709X. DOI: [10.1145/215585.215639](https://doi.org/10.1145/215585.215639). URL: <http://portal.acm.org/citation.cfm?doid=215585.215639> (ver pp. 15, 17, 25).
- [71] J. P. Rolland, R. L. Holloway e H. Fuchs. «Comparison of optical and video see-through, head-mounted displays». Em: *Telemanipulator and Telepresence Technologies*. Ed. por H. Das. Vol. 2351. SPIE, 1995-12, pp. 293–307. ISBN: 081941686X. DOI: [10.1117/12.197322](https://doi.org/10.1117/12.197322). URL: <http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=981546> (ver p. 16).
- [72] E. Rublee et al. «ORB: An efficient alternative to SIFT or SURF». Em: *2011 International Conference on Computer Vision*. Barcelona, Spain: IEEE, 2011-11, pp. 2564–2571. ISBN: 978-1-4577-1102-2. DOI: [10.1109/ICCV.2011.6126544](https://doi.org/10.1109/ICCV.2011.6126544). URL: <http://ieeexplore.ieee.org/document/6126544/> (ver p. 22).
- [73] D. Saab. «Conceptualizing Space: Mapping Schemas as Meaningful Representations». Tese de doutoramento. 2003-05. DOI: [10.13140/2.1.3030.1767](https://doi.org/10.13140/2.1.3030.1767) (ver p. 8).
- [74] G. Schall, D. Schmalstieg e S. Junghanns. «VIDENTE - 3D Visualization of Underground Infrastructure using Handheld Augmented Reality». Em: *GeoHydroinformatics: integrating GIS and water engineering* (2010), pp. 207–219 (ver p. 28).

- [75] Y. Sermet e I. Demir. «An Immersive Decision Support System for Disaster Response». Em: *Proceedings of the 26th ACM Symposium on Virtual Reality Software and Technology*. VRST '20. Virtual Event, Canada: Association for Computing Machinery, 2020. ISBN: 9781450376198. DOI: [10.1145/3385956.3422087](https://doi.org/10.1145/3385956.3422087). URL: <https://dl.acm.org/doi/10.1145/3385956.3422087> (ver p. 28).
- [76] M. E. Shin, J. Campbell e N. Burkhart. *Essentials of Geographic Information Systems*. Version 2.1. Boston, MA, United States: Boston Academic Publishing/Flatworld, 2018. ISBN: 9781453390801 (ver pp. 6, 7).
- [77] *Slippy map - OpenStreetMap Wiki*. URL: https://wiki.openstreetmap.org/wiki/Slippy_map (acedido em 2023-09-07) (ver p. 10).
- [78] *Slippy map tilenames - OpenStreetMap Wiki*. URL: https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames (acedido em 2023-09-07) (ver p. 10).
- [79] D. E. Smalley et al. «A photophoretic-trap volumetric display». Em: *Nature* 553.7689 (2018-01), pp. 486–490. ISSN: 0028-0836. DOI: [10.1038/nature25176](https://doi.org/10.1038/nature25176). URL: <http://www.nature.com/articles/nature25176> (ver p. 19).
- [80] J. S. Subirana, M. Hernández-Pajares e J. J. Zornoza. *GNSS Data Processing*. Vol. I: Fundamentals and Algorithms. ESA Communications, 2013-05. ISBN: 978-92-9221-886-7 (ver p. 8).
- [81] I. E. Sutherland. «A head-mounted three dimensional display». Em: *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)*. New York, New York, USA: ACM Press, 1968, p. 757. DOI: [10.1145/1476589.1476686](https://doi.org/10.1145/1476589.1476686). URL: <http://portal.acm.org/citation.cfm?doid=1476589.1476686> (ver pp. 15, 16).
- [82] I. E. Sutherland. «The Ultimate Display». Em: *Proceedings of the IFIP Congress*. 1965, pp. 506–508 (ver p. 15).
- [83] *Tango down: Google's first AR project officially shuts down March 1 | Ars Technica*. URL: <https://arstechnica.com/gadgets/2017/12/tango-down-googles-first-ar-project-officially-shuts-down-march-1/> (acedido em 2023-09-10) (ver p. 36).
- [84] *The Virtual Interface Environment Workstation (VIEW), 1990 | NASA*. URL: https://www.nasa.gov/ames/spinoff/new%7B%5C_%7Dcontinent%7B%5C_%7Dof%7B%5C_%7Dideas/ (acedido em 2023-01-13) (ver p. 16).
- [85] *Tiles à la Google Maps: Coordinates, Tile Bounds and Projection | MapTiler*. URL: <https://www.maptiler.com/google-maps-coordinates-tile-bounds-projection/> (acedido em 2023-09-07) (ver p. 11).
- [86] B. Tomaszewski et al. «Supporting Disaster Resilience Spatial Thinking with Serious GeoGames: Project Lily Pad». Em: *ISPRS International Journal of Geo-Information* 9.6 (2020). ISSN: 2220-9964. DOI: [10.3390/ijgi9060405](https://doi.org/10.3390/ijgi9060405). URL: <https://www.mdpi.com/2220-9964/9/6/405> (ver p. 29).

-
- [87] *Types of geodatabases — ArcGIS Pro | Documentation*. URL: <https://pro.arcgis.com/en/pro-app/latest/help/data/geodatabases/overview/types-of-geodatabases.htm> (acedido em 2023-09-08) (ver p. 14).
- [88] *Unity's AR Foundation Framework | Cross platform augmented reality development software | Unity*. URL: <https://unity.com/unity/features/arfoundation> (acedido em 2023-01-28) (ver p. 85).
- [89] *User Experience Questionnaire (UEQ)*. URL: <https://www.ueq-online.org/> (acedido em 2023-08-12) (ver pp. 103, 117, 118).
- [90] *What is a geodatabase? — ArcGIS Pro | Documentation*. URL: <https://pro.arcgis.com/en/pro-app/latest/help/data/geodatabases/overview/what-is-a-geodatabase-.htm> (acedido em 2023-09-08) (ver p. 14).
- [91] *What is Three-Tier Architecture | IBM*. URL: <https://www.ibm.com/topics/three-tier-architecture> (acedido em 2023-06-24) (ver pp. 42, 58).
- [92] *Wi-Fi location: ranging with RTT | Android Developers*. URL: <https://developer.android.com/guide/topics/connectivity/wifi-rtt> (acedido em 2023-07-02) (ver p. 23).
- [93] Youngkwan Cho e U. Neumann. «Multi-ring color fiducial systems for scalable fiducial tracking augmented reality». Em: *Proceedings. IEEE 1998 Virtual Reality Annual International Symposium (Cat. No.98CB36180)*. IEEE Computer Society, 1998, p. 212. ISBN: 0-8186-8362-7. DOI: [10.1109/VRAIS.1998.658494](https://doi.org/10.1109/VRAIS.1998.658494). URL: <http://ieeexplore.ieee.org/document/658494/> (ver p. 22).

MARCADOR DO MODO DE REALIDADE AUMENTADA

Este Apêndice mostra o marcador utilizado no modo de Realidade Aumentada para detetar o plano em que é sobreposto o mapa digital. Ao ser detetado pelo ARCore, é feita uma estimacão da pose do dispositivo relativo ao marcador, isto é a combinacão da sua posicão e orientacão, que é utilizada para colocar corretamente os conteúdos digitais de forma que possam parecer estar presentes no mundo real.



FOLHA DE ROSTO DA SESSÃO DE TESTES

Este Apêndice mostra o conteúdo da folha de rosto do guião da sessão de testes de usabilidade. Contém as instruções da sessão, a sua estrutura e a declaração de consentimento informado. Apenas se os participantes concordarem com esta última e declararem o seu consentimento é que pode ser realizada a sessão de testes.

Instruções:

Durante esta sessão de teste, será pedido que realize tarefas específicas usando a aplicação fornecida e que responda a certas questões. Enquanto realiza essas tarefas e questões, por favor, tente verbalizar as suas ações e decisões em voz alta. Expresse os seus pensamentos, sentimentos e ações à medida que navega pela interface. Esteja à vontade para partilhar as suas opiniões, críticas construtivas e sugestões, pois isso ajudará a compreender a sua experiência e como melhorar a interface. Se sentir dificuldades ou confusão, partilhe. Pode dizer coisas como:

- "Estou a clicar neste botão porque espero que me leve para o local que pretendo ir."
- "Não tenho a certeza do que fazer a seguir."
- "Este texto é difícil de ler."
- "Esperava que esta opção estivesse num sítio diferente."

Queremos que se sinta confortável e também enfatizar que estamos a testar o sistema, não a avaliar a sua capacidade. Quaisquer dificuldades que enfrente são atribuíveis ao sistema e não a si. Precisamos da sua ajuda para detetar e corrigir esses problemas.

Estrutura da Sessão:

A sessão de teste tem a seguinte estrutura:

- Questionário de Caracterização: para obter dados demográficos e de experiência relevante dos participantes

-
- Grupos A, B, C e D: séries de tarefas e questões relativas à aplicação. Lembre-se de pensar em voz alta durante este processo, descrevendo as suas ações e raciocínios.
 - Grupo E: comparação entre a aplicação em dispositivos Desktop e Mobile
 - Questionário Específico de Usabilidade: questões gerais sobre funcionalidades específicas da aplicação
 - System Usability Scale e User Experience Questionnaire: questionários standard de usabilidade e experiência de utilizador
 - Comentários Finais: partilha de feedback geral

Tratamento de Dados, Privacidade e Confidencialidade:

Apenas pode participar nesta sessão se, após ler esta informação, consentir em participar neste estudo.

Toda a informação recolhida será usada exclusivamente no contexto desta dissertação. Os dados serão anonimizados antes de serem processados e armazenados de forma segura. Os dados originais serão retidos apenas durante o período de tempo necessário ao seu processamento, sendo posteriormente eliminados. Nenhuma informação pessoalmente identificável será publicada nem partilhada com terceiros alheios ao estudo. As informações recolhidas serão as introduzidas neste formulário, observações anotadas pelo investigador, comunicação verbal durante a sessão e captura do ecrã da aplicação.

A sua participação é voluntária e em qualquer momento pode desistir de participar no estudo, seja durante a sessão diretamente ao investigador ou após a sessão de teste através de contacto para d.flamino@campus.fct.unl.pt, sendo toda a informação recolhida durante a sessão devidamente eliminada.

Pode fazer perguntas em qualquer momento durante a sessão.

Obrigada!

Ao escolher "Aceito" abaixo, aceita participar voluntariamente neste estudo académico, indica que leu e compreendeu a informação acima apresentada e aceita que a sua informação seja recolhida e processada nos termos definidos.

FORMULÁRIO DE OBSERVAÇÕES

Este Apêndice mostra o formulário utilizado para o registo de observações pelo investigador durante uma sessão de testes de usabilidade.

Session ID		Data	
------------	--	------	--

Grupo A	Completo			Observações
	Sim	Com Ajuda	Não	
Tarefa A1				
Tarefa A2				
Questão A2Q1				
Tarefa A3				

Grupo B	Completo			Observações
	Sim	Com Ajuda	Não	
Tarefa B1				[POI/Search/Drag/Keys/UI]
Tarefa B2				
Questão B2Q1				
Tarefa B3				
Questão B3Q1				
Tarefa B4				
Questão B4Q1				

Grupo C	Completo			Observações
	Sim	Com Ajuda	Não	
Tarefa C1				[POI/Search/Drag/Keys/UI]
Tarefa C2				
Tarefa C3				
Questão C3Q1				
Tarefa C4				
Questão C4Q1				
Tarefa C5				

APÊNDICE C. FORMULÁRIO DE OBSERVAÇÕES

Grupo D	Completo			Observações
	Sim	Com Ajuda	Não	
Tarefa D1				[Drag/Keys/UI]
Questão D1Q1				
Questão D1Q2				
Questão D1Q3	---	---	---	
Tarefa D2				
Tarefa D3				
Tarefa D4				
Questão D4Q1				
Questão D4Q2				
Tarefa D5				

Grupo E	Completo			Observações
	Sim	Com Ajuda	Não	
Tarefa E1D1				[POI/Search/Drag/Keys/UI]
Tarefa E1D2				
Tarefa E1D3				[Keys/UI]
Tarefa E1M1				[POI/Search/Drag/UI]
Tarefa E1M2				
Tarefa E1M3				[Drag/UI]
Tarefa E1M4				
Questão E1Q1	---	---	---	
Questão E1Q2	---	---	---	
Questão E1Q3	---	---	---	
Questão E1Q4	---	---	---	
Questão E1Q5	---	---	---	

QUESTÕES DO SUS E UEQ

Neste anexo estão presentes as questões do questionário [SUS](#) e o [UEQ](#).

SUS

Desenvolvido por John Brook [12] como uma ferramenta rápida e eficaz para medir a usabilidade. Consiste num questionário de 10 afirmações sobre as quais o participante indica a sua opinião numa escala *Likert* com 5 níveis, desde 1 para *Discordo totalmente* até 5 para *Concordo totalmente*. Essas afirmações são as seguintes:

- Eu acho que gostaria de usar esta aplicação frequentemente
- Eu achei a aplicação desnecessariamente complexa
- Eu achei a aplicação fácil de usar
- Eu acho que precisaria do apoio de uma pessoa técnica para poder usar esta aplicação
- Eu achei que as várias funções desta aplicação estavam bem integradas
- Eu achei que havia demasiada inconsistência nesta aplicação
- Eu imagino que a maioria das pessoas aprenderia a usar esta aplicação rapidamente
- Eu achei a aplicação muito complicada de usar
- Eu senti-me confiante ao usar a aplicação
- Eu precisei de aprender muitas coisas antes de poder usar esta aplicação

UEQ

Por favor dê-nos a sua opinião.

A fim de avaliar o produto, por favor preencha o seguinte questionário. É constituído por pares de opostos relativos às propriedades que o produto possa ter. As graduações entre os opostos são representadas por círculos. Ao marcar um dos círculos, você pode expressar sua opinião sobre um conceito.

Exemplo:

Atraente	☐	☒	☐	☐	☐	☐	☐	Feio
----------	-----------------------	----------------------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	------

Esta resposta significa que avalia o produto mais **atraente** do que **feio**.

Marque a sua resposta da forma mais espontânea possível. É importante que não pense demasiado na resposta porque a sua avaliação imediata é que é importante.

Por favor, assinale sempre uma resposta, mesmo que não tenha certezas sobre um par de termos ou que os termos não se enquadrem com o produto.

Não há respostas "certas" ou respostas "erradas". A sua opinião pessoal é que conta!

Por favor, dê-nos a sua avaliação atual do produto em causa.

Por favor, marque apenas um círculo por linha.

	1	2	3	4	5	6	7	
Desagradável	☐	☐	☐	☐	☐	☐	☐	Agradável
Incompreensível	☐	☐	☐	☐	☐	☐	☐	Compreensível
Criativo	☐	☐	☐	☐	☐	☐	☐	Sem criatividade
De Fácil aprendizagem	☐	☐	☐	☐	☐	☐	☐	De difícil aprendizagem
Valioso	☐	☐	☐	☐	☐	☐	☐	Sem valor
Aborrecido	☐	☐	☐	☐	☐	☐	☐	Excitante
Desinteressante	☐	☐	☐	☐	☐	☐	☐	Interessante
Imprevisível	☐	☐	☐	☐	☐	☐	☐	Previsível
Rápido	☐	☐	☐	☐	☐	☐	☐	Lento
Original	☐	☐	☐	☐	☐	☐	☐	Convencional
Obstrutivo	☐	☐	☐	☐	☐	☐	☐	Condutor
Bom	☐	☐	☐	☐	☐	☐	☐	Mau
Complicado	☐	☐	☐	☐	☐	☐	☐	Fácil
Desinteressante	☐	☐	☐	☐	☐	☐	☐	Atrativo
Comum	☐	☐	☐	☐	☐	☐	☐	Vanguardista
Incómodo	☐	☐	☐	☐	☐	☐	☐	Cómodo
Seguro	☐	☐	☐	☐	☐	☐	☐	Inseguro
Motivante	☐	☐	☐	☐	☐	☐	☐	Desmotivante
Atende as expectativas	☐	☐	☐	☐	☐	☐	☐	Não atende as expectativas
Ineficiente	☐	☐	☐	☐	☐	☐	☐	Eficiente
Evidente	☐	☐	☐	☐	☐	☐	☐	Confuso
Impraticável	☐	☐	☐	☐	☐	☐	☐	Prático
Organizado	☐	☐	☐	☐	☐	☐	☐	Desorganizado
Atraente	☐	☐	☐	☐	☐	☐	☐	Feio
Simpático	☐	☐	☐	☐	☐	☐	☐	Antipático
Conservador	☐	☐	☐	☐	☐	☐	☐	Inovador



2023 TileQuery: Integrating Spatial Query, Visualization, and Analysis

University of Florida, Gainesville, FL, USA

NOVA SCHOOL OF SCIENCE & TECHNOLOGY