

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



FEUP

Jogo S3rio para Reabilita33o Neurocognitiva: Cidade Virtual

Maria Jo3o Pinto Lu3s Miranda

Mestrado Integrado em Engenharia Inform3tica e Computa33o

Orientador: Ant3nio Fernando Vasconcelos Cunha Castro Coelho (Doutor)

Co-orientadora: Artemisa Agostinha Monteiro da Rocha Dores (Doutora)

11 de Julho de 2012

Jogo Sérió para Reabilitação Neurocognitiva: Cidade Virtual

Maria João Pinto Luís Miranda

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Rosaldo José Fernandes Rossetti (Doutor)

Arguente: Leonel Caseiro Morgado (Doutor)

Vogal: António Fernando Vasconcelos Cunha Castro Coelho (Doutor)

11 de Julho de 2012

Resumo

O aparecimento das novas tecnologias tem provocado um grande impacto na sociedade, particularmente na área da saúde. Desde sempre foi evidente que as tecnologias poderiam ter um importante papel neste setor, pela contribuição positiva nos métodos clínicos e qualidade de vida dos pacientes. Sendo um dos principais problemas dos métodos clínicos tradicionais a dificuldade em motivar os pacientes, os jogos sérios vieram eliminar essa barreira.

A contribuição das novas tecnologias na reabilitação neurocognitiva vai mais longe. Nesta área, os jogos sérios são essenciais não só pela motivação, como também pelo realismo que é conseguido através de cenários tridimensionais e pelo recurso à realidade virtual. Desta forma, tornou-se possível aproximar os exercícios do contexto clínico às tarefas quotidianas dos pacientes.

No contexto deste projeto de dissertação, desenvolveu-se um jogo sério para a reabilitação neurocognitiva de pessoas com disfunção executiva e outras funções cognitivas subsidiárias, como a atenção e a memória. O principal objetivo do jogo é proporcionar aos pacientes diversos desafios que se assemelhem a tarefas reais do dia a dia. Nesse sentido, pretende-se diariamente educar o paciente a primeiro planear as suas tarefas e posteriormente executá-las com sucesso pela ordem inicialmente agendada. A primeira fase é composta por um conjunto de tarefas que devem ser corretamente ordenadas, de acordo com a regra estipulada para o nível. Seguidamente, o jogador deve percorrer a cidade, usando os transportes que dispõe, no sentido de alcançar com sucesso todas as tarefas dadas. O sucesso em cada nível só é conseguido caso o jogador avance com sucesso as duas fases, caso contrário, terá que o repetir até atingir o objetivo.

Este documento apresenta o estudo feito durante o projeto da dissertação. Nesse sentido, procurou-se inicialmente perceber quais as principais tecnologias a utilizar e estudar a melhor estratégia de resolução do problema levantado. Após análise foi desenhada a arquitetura do sistema e desenvolvido o protótipo do jogo.

Palavras-chave: Reabilitação Neurocognitiva; Jogos Sérios; Modelação 3D; Modelação Procedimental

Abstract

The appearance of new technologies has had a great impact on society, particularly in the area of health. These products' fundamental role in this sector, was easily predictable from the start essentially because of their positive contribution to improving clinical procedures and patient life quality. Since one of traditional clinical methods' main problems is how to motivate the patients, the serious games came to remove this barrier.

The new technologies' contribution in neurocognitive rehabilitation goes further. In this area, the serious games are essential not only for the motivation, but also because of the realism that is achieved with three-dimensional scenes and virtual reality. This way, it was possible to approach the clinic's exercises to the daily tasks to the patients.

In the context of this dissertation project, it was developed a serious game for a neurocognitive rehabilitation for executive dysfunctional people and others subsidiaries cognitive functions, like attention and memory. The main objective for the game is to provide to the patients different challenges that are similar to the real daily tasks. This way, it pretends to educate the patients in order to oblige them to, first of all, plan their tasks and then execute them with success, in the same order that was previously orderly, according to the level's rule. Then, the player should travel around the city, using the available transports, in order to reach with sucess all the tasks. The success in each level is only achieved if the player successfully advance the two phases, otherwise, he will repete the same level until he reaches the goal.

This document presents a study done during the dissertation project. In that sense, it is examined the main technologies to decide which should be used, followed by a description of the proposed architecture and the implementation methodology.

Keywords: Cognitive Rehabilitation; Serious Games; 3D Modeling; Procedural Modelling

Conteúdo

1	Introdução	1
1.1	Contexto e motivação	1
1.2	Objetivos	2
1.3	Descrição do Problema	3
1.4	Resultado Esperado	3
1.5	Estrutura do Documento	3
2	Jogos	5
2.1	Jogos Sérios	5
2.2	Jogos Sérios na Reabilitação Neurocognitiva	6
2.3	Jogos Online	7
2.4	Trabalhos Relacionados	7
2.4.1	<i>RehaCom</i>	8
2.4.2	<i>Virtual Life Skills</i>	9
2.5	Sumário	10
3	Ferramentas	11
3.1	Motores de Jogo	11
3.1.1	Análise geral das soluções	11
3.1.2	<i>NeoAxis</i>	12
3.1.3	<i>Unity3D</i>	13
3.2	Modelação 3D	15
3.2.1	Modelação Procedimental	15
3.2.2	<i>City Engine</i>	16
3.3	Dispositivos de Entrada 3D	17
3.3.1	Visão Geral	17
3.3.2	Apontadores	17
3.3.3	Rato 3D	18
3.4	Sumário	19
4	Cidade Virtual - Descrição do jogo	21
4.1	Conceito do Jogo	21
4.1.1	Cenário	21
4.1.2	Lotes	21
4.1.3	Transportes	22
4.2	Mecânica do Jogo	23
4.2.1	Sessão	23
4.2.2	Patamar	24

CONTEÚDO

4.2.3	Nível	25
4.3	<i>Feedback</i>	25
4.4	Aprendizagem	26
4.5	Fluxo de jogo	26
4.6	Interação	29
5	<i>Cidade Virtual - Implementação</i>	31
5.1	Metodologia	31
5.2	Requisitos	32
5.2.1	Requisitos funcionais	32
5.2.2	Requisitos não funcionais	33
5.3	Arquitetura	34
5.3.1	Bloco Interface	35
5.3.2	Bloco Base de Dados	35
5.3.3	Bloco Mecânica de Jogo	35
5.4	Modelação 3D	36
5.4.1	Importar mapa - <i>OpenStreetMap</i>	36
5.4.2	Modelar a cidade - <i>CityEngine</i>	37
5.4.3	Exportar cenário - <i>CityEngine</i>	40
5.5	Cena do jogo	41
5.5.1	Cidade	41
5.5.2	Luz	41
5.5.3	Câmara	41
5.5.4	Autocarros e paragens	42
5.6	Base de Dados	42
5.6.1	Gestão de Níveis	42
5.6.2	Gestão de Sessões	43
5.6.3	Gestão de Utilizadores	44
5.6.4	Gestão de Autocarros	44
5.7	Implementação da mecânica de jogo	45
5.7.1	Planeamento das tarefas	45
5.7.2	Executar as tarefas	47
5.7.3	Sistema de autocarros	48
5.7.4	Decisões de implementação orientadas à reabilitação	51
5.8	Proposta de avaliação da eficácia do programa	52
6	Conclusão	53
6.1	Trabalho Futuro	54
A	Diagrama de arquitetura de Base de Dados	57
B	Diagrama de classes	61
	Referências	63

Lista de Figuras

2.1	<i>RehaCom Custom Panel</i> [Has]	9
2.2	Supermercado [BNC99]	10
2.3	Café [BNC99]	10
2.4	Casa [BNC99]	10
3.1	Interface de utilização do <i>NeoAxis</i> [Ltd]	12
3.2	EMSAVE - Exemplo de jogo sério desenvolvido com <i>NeoAxis</i> [Ltd]	13
3.3	Interface de utilização do <i>Unity3D</i> [Tec12]	14
3.4	CliniSpace - Exemplo de jogo sério desenvolvido com <i>Unity3D</i> [Sol09]	15
3.5	Processo durante a modelação no <i>City Engine</i> [ESR12]	16
3.6	Controlador de movimento da Playstation [Son]	17
3.7	SpaceNavigator [3DCb]	18
3.8	SpaceExplorer [3DCa]	18
3.9	Spheric [Axs10]	18
3.10	Direções possíveis com rato 3D [3DC12]	19
4.1	Cenário de Gaia	22
4.2	Cenário do Porto	22
4.3	Correios	22
4.4	Jardim	22
4.5	Residência	22
4.6	Exemplo de transporte - Autocarro	23
4.7	Conceito de sessão, patamar e nível [DMC ⁺]	24
4.8	Funcionamento dos níveis num patamar [DMC ⁺]	25
4.9	Instrução para o nível	27
4.10	Ordenação das tarefas	27
4.11	Após seleção de um objeto do mapa	28
4.12	Erro na tarefa	28
4.13	Sucesso no nível	28
4.14	Vista ortogonal do mapa	29
4.15	Diagrama de atividades	30
5.1	Metodologia para desenvolvimento de jogos	32
5.2	Arquitetura do sistema <i>Virtual City</i>	34
5.3	Diagrama de pacotes	35
5.4	Metodologia para modelação tridimensional	36
5.5	Exportar mapa para ficheiro OSM XML	37
5.6	Planta do mapa bidimensional - <i>CityEngine</i>	38
5.7	Atribuição de regras a blocos	39

LISTA DE FIGURAS

5.8	Aspeto final do cenário	40
5.9	Estratégia de implementação da mecânica de Jogo - Unity3D	45
5.10	Evolução entre níveis e patamares	49
5.11	Deslocação do autocarro no cenário	50
A.1	Arquitetura da Base de Dados	58
A.2	Arquitetura da Base de Dados - Autocarros	59
B.1	Diagrama de classes	62

Lista de Tabelas

2.1	Diferença entre jogos de entretenimento e jogos sérios [Cor06]	6
2.2	Análise do Estado da Arte e comparação com o <i>Virtual City</i>	10
3.1	Análise de motores de jogo [Gar12] [Dev11b] [Tec12] [Sys] [Ese] [Ltd]	12
4.1	Correspondência entre lote e categoria	22
5.1	Requisitos funcionais - Interface do jogo	32
5.2	Requisitos funcionais - Lógica de jogo	33
5.3	Requisitos funcionais - Gestão de utilizadores	33

LISTA DE TABELAS

Abreviaturas e Símbolos

CARP-VR	Computer Assisted Rehabilitation Program - Virtual Reality
XML	Extensible Markup Language
OSM	OpenStreetMap
FBX	filmbox
UML	Unified Modeling Language
FPP	First-person perspective
TPP	Third-person perspective
.OBJ	3D object
PHP	Personal Home Page
URL	Uniform resource locator
MD5	Message- Digest algorithm 5
API	Application Programming Interface
HADS	Hospital Anxiety and Depression Scale
MMSE	Mini-mental State Examination
BADS	Behavioural Assessment of the Dysexecutive Syndrome
LCA	Lesão cerebral adquirida
GUI	Interface Gráfica do Utilizador
3D	Tridimensional
2D	Bidimensional

Capítulo 1

Introdução

Ao longo dos últimos anos, a tecnologia revolucionou os mais diversos setores, criando novas formas de integração e criação de valor. A área da saúde, que procura evoluir os seus métodos clínicos, não foi exceção. De facto, a melhoria da qualidade de vida dos pacientes e o aumento da esperança de vida foram possíveis devido à procura incessante de novos, melhores e mais avançados métodos clínicos. Neste contexto surge a aliança entre as áreas da saúde e as novas tecnologias de informação, numa cooperação cada vez mais importante e que tem abrangido a reabilitação física e a neuropsicológica.

Os jogos sérios ocupam um lugar importante na colaboração com a área da saúde, pela capacidade de orientarem os jogos para objetivos reais de aprendizagem e proporcionarem simultaneamente um tratamento aliciante para os pacientes. O propósito dos jogos sérios aplicados à reabilitação neurocognitiva é auxiliar na reabilitação das funções cognitivas afetadas, resultado de lesão ou doença, como : as visuoespaciais, a atenção, a memória e as funções executivas, visando o aumento da funcionalidade e da qualidade de vida. A realidade virtual começa a ter um importante papel nesta área, pela possibilidade de potenciar a aprendizagem e permitir treinar atividades orientadas à vida real.

1.1 Contexto e motivação

Uma das principais causas de morte e inaptidão em crianças e adultos é a lesão cerebral adquirida (LCA) [DMC⁺]. A LCA pode ter origem num acidente de trabalho ou rodoviário, o que leva a um traumatismo crânio-encefálico, ou ser resultado de uma vida sedentária, o que provoca distúrbios neurológicos. Uma das consequências da LCA é a disfunção executiva, ou seja, a incapacidade cognitiva de ter iniciativa, planejar, organizar e regular comportamentos para tarefas do dia a dia. A incapacidade de viver de forma independente é uma das principais barreiras para a re-inserção na vida social e profissional. A reabilitação neurocognitiva, pretende melhorar as capacidades dos pacientes na resolução de problemas diários [DMC⁺]. No entanto, o grande desafio,

durante a reabilitação, está em manter os pacientes motivados de forma a prevenir eventuais desistências. Em resposta à falta de motivação, surgiu a cooperação entre a reabilitação neurocognitiva e as novas soluções proporcionadas pelas tecnologias.

A interação entre realidade virtual e jogos sérios é cada vez mais usual na área de aplicações médicas, pela vantagem de se utilizar as novas tecnologias em prol de uma melhoria nos métodos clínicos. Ainda por permitir que os exercícios propostos devem ser ajustados às capacidades individuais de cada utilizador. A reabilitação das funções executivas, pela sua complexidade e por serem determinantes no funcionamento independente, exige esta realidade, no sentido de habilitar o paciente à resolução de problemas do dia a dia.

Esta dissertação insere-se no projeto de doutoramento “A Realidade Virtual na Reabilitação” [DCB⁺11], que resultou no desenvolvimento do *Computer-Assisted Rehabilitation Program - Virtual Reality (CARP-VR)*, que promove o auxílio aos profissionais de saúde na reabilitação de funções cognitivas de pessoas com LCA. Trata-se de um jogo sério que pretende, de uma forma lúdica e ao mesmo tempo didática, ajudar na reabilitação neurocognitiva, aproximando os desafios do jogo às atividades do dia a dia do utilizador.

Os requisitos do sistema final, *CARP-VR*, são definidos pelos membros da equipa multidisciplinar, através do desenho do programa pelos psicólogos, com o apoio dos programadores que estão ligados às diferentes fases de desenvolvimento do jogo. A primeira fase de desenvolvimento já foi concluída, o *Supermercado*, um ambiente virtual onde o utilizador deve fazer compras segundo uma lista fornecida, dirigir-se às secções corretas e cumprir requisitos especiais na resolução dos problemas propostos, como não ultrapassar o montante disponibilizado.

1.2 Objetivos

O objetivo geral desta dissertação é o desenvolvimento de um jogo sério de apoio à reabilitação neurocognitiva, utilizando uma cidade virtual adaptada à realidade portuguesa. Neste contexto os principais objetivos são:

- Desenvolver uma arquitetura do sistema que permita que pacientes e terapeutas possam aceder ao sistema em diferentes computadores;
- Escolher um motor de jogo capaz de integrar com periféricos (por exemplo: *joystick*, rato, teclado) e que permita disponibilizar a aplicação na web;
- Modelar o ambiente virtual de uma cidade portuguesa;
- Escolher um periférico que seja adaptado às necessidades dos utilizadores finais. Estes utilizadores apresentam por vezes dificuldades de manuseamento de um teclado e rato *standard*;
- Desenvolver uma mecânica de jogo que seja adequada aos procedimentos de reabilitação das funções executivas;
- Produzir um protótipo de jogo;

- Planear testes e validação do protótipo.

1.3 Descrição do Problema

O facto dos centros de reabilitação não oferecerem as melhores condições para a realização de exercícios orientados a tarefas do dia a dia pode limitar a generalização das competências adquiridas para contextos de vida real. Os métodos tradicionais, também conhecidos por *exercícios de papel e lápis* são por vezes inadaptados a este tipo de reabilitação. Mesmo que os doentes apresentem evolução nos exercícios das funções executivas, podem não estar aptos para o retorno à realidade, pela não generalização das competências adquiridas para este contexto.

Um outro problema identificado é a reduzida motivação para as sessões de reabilitação. Os métodos tradicionais, podem não ser suficientemente estimulantes para despertar a motivação dos pacientes. Aliado a este fator está também o *feedback*, o utilizador geralmente só recebe informação sobre a sua prestação através dos terapeutas. Esta dependência pode de alguma forma atrasar o processo de reabilitação, sendo desejável que seja fornecido imediatamente a seguir ao desempenho.

É importante que para cada paciente seja programado um processo de reabilitação adequado às suas necessidades na vida real, o que significa que deve ser feito um esforço para aproximar o ambiente do jogo sério a essa realidade. Contudo, ainda hoje os exercícios são na sua maioria tipificados e nem sempre personalizados, ajustando-se à evolução dos pacientes.

1.4 Resultado Esperado

Do trabalho final espera-se que contribua para a melhoria na reabilitação das pessoas com LCA. Desta forma, pretende-se obter um jogo sério de suporte a esse processo, desenhado de acordo com a realidade portuguesa, em que a mecânica do jogo seja definida segundo as técnicas de reabilitação das funções executivas.

Pretende-se que o produto final seja orientado aos utilizadores, que todo o ambiente e história do jogo sejam completamente adaptáveis às suas necessidades.

1.5 Estrutura do Documento

O presente relatório está dividido em seis capítulos: Introdução (capítulo 1), Jogos (capítulo 2), Ferramentas (capítulo 3), Mecânica de jogo (capítulo 4), Implementação (capítulo 5) e Conclusão (capítulo 6). O atual capítulo, *Introdução*, mostra a problemática da tese e o que era esperado no final do desenvolvimento. O segundo capítulo contextualiza os jogos sérios na reabilitação neurocognitiva e apresenta alguns trabalhos relacionados. O capítulo reservado para as ferramentas expõe algumas soluções para o problema levantado e termina com uma perspetiva de solução para cada uma dos pontos abordados: motor de jogo, modelação 3D e periféricos. O quarto capítulo destina-se à exposição da mecânica de jogo, explicação dos elementos do jogo e

Introdução

do seu funcionamento. A implementação do protótipo é detalhada no capítulo cinco. O último capítulo é reservado para as conclusões do projeto e proposta de trabalhos futuros.

Capítulo 2

Jogos

O capítulo dos jogos destina-se a esclarecer alguns conceitos mais teóricos relativamente aos jogos, jogos sérios e como são aplicáveis à área da saúde, mais concretamente à reabilitação neurocognitiva.

Cailois, em 1957, definia os jogos como atividades de divertimento, competição, sorte e simulação [Cai57]. Outros autores afirmam ser uma forma de arte em que os participantes terão que tomar decisões [Cos94], obedecendo a regras pré-estabelecidas [SZ03], para atingir um objetivo final concreto [Kev01].

Com o tempo os jogos foram ganhando novas dimensões e hoje em dia já não são associados a “geeks” [RMR], são uma ferramenta de entretenimento familiar, aprendizagem e de suporte no mundo do trabalho.

2.1 Jogos Sérios

O desenvolvimento de jogos sérios surgiu em meados dos anos noventa com o objetivo de melhorar alguns aspetos no processo de aprendizagem [Der07]. Corti acredita que os jogos de computador captam e envolvem os utilizadores finais para um propósito específico [Cor06]. Estes jogos vão para além do mero entretenimento [SJB07] (Tabela 2.1), sendo uma forma de aprendizagem com regras, desafios e objetivos concretos. Os jogos sérios são geralmente desenvolvidos para exploração científica, engenharia [PSB⁺11], treino militar, serviços médicos, educação e cultura [SJB07]. O primeiro jogo sério a ser desenvolvido foi o *Army Battezone*, em 1980, para treino militar [Mac01]. Muitos outros jogos foram desde então desenhados, inclusive na área da saúde, para intervenção psicológica, treino cognitivo e/ou reabilitação física. As vantagens dos jogos sérios nestas áreas começam a tornar-se evidentes e o seu crescimento é prova disso mesmo. Através dos jogos torna-se mais fácil:

- assimilar a informação transmitida em imagem, vídeo ou som em detrimento de textos exaustivos;

- o jogador ficar motivado com os desafios propostos e pela forma como são expostos;
- promover competição e/ou colaboração, que podem atuar como fator motivador;
- prever resultados que por vezes não são possíveis de concretizar na vida real, por perigo, tempo ou despesa excessiva [SJB07];
- relacionar capacidades do utilizador e a complexidade entre diferentes níveis, tornando o jogo mais desafiante.

O mundo empresarial também recorre a estas tecnologias para recrutar novos funcionários, despertar novos talentos ou melhorar a comunicação dentro da empresa [Der07]. Assim, fica clara a popularidade dos jogos sérios nos mais diferentes contextos [SJB07].

Desde o momento em que se evidenciou o potencial de contribuição noutras áreas, que diariamente mais instituições de ensino, de saúde e militares adotam este tipo de ferramentas. Apesar disso, nem sempre foi óbvia a sua maior valia e na área da educação houve algumas reticências à sua utilização, pois algumas pessoas envolvidas receavam as consequências sociais e comportamentais resultantes da sua utilização pelos estudantes [Squ03]. Os jogos eram considerados como sinónimo de violência e agressividade para quem os utilizava, parecendo resultar daí o medo de serem más influências para crianças e adultos. Contudo, a facilidade, rapidez e motivação que podem trazer à aprendizagem foram argumentos que justificaram avançar com a sua utilização também na educação.

Tabela 2.1: Diferença entre jogos de entretenimento e jogos sérios [Cor06]

	Jogos sérios	Jogos de entretenimento
Tarefas vs Experiência	Focado na resolução de problemas	Focado na experiência
Objetivo	Aprendizagem	Divertimento
Comunicação	Comunicação natural	Comunicação perfeita

2.2 Jogos Sérios na Reabilitação Neurocognitiva

Os jogos sérios, tal como já foi demonstrado na secção 2.1, trouxeram grandes vantagens aos mais diversos contextos. A área da saúde não foi exceção e intervenção clínica e a reabilitação foram invadidas por novos métodos suportados por estes jogos [SJB07].

Apesar de se considerar que os jogos sérios desempenham um importante papel na reabilitação neurocognitiva e que podem permitir obter resultados eficazes e por vezes de forma mais rápida, não se deverá entender que estas novas ferramentas vieram substituir as intervenções tradicionais. Esta deve ser considerada de forma complementar à intervenção, potenciando-a. Uma das vantagens destes tipos de jogos é a capacidade que apresentam de promover a motivação, visto que por vezes se torna complicado manter o paciente interessado, após sucessivas repetições dos mesmos exercícios [RMR]. A possibilidade de contribuir para a generalização das aprendizagens para contextos de vida real é outra vantagem. Ao permitir ao utilizador uma maior autonomia,

resultante da possibilidade de praticar os exercícios individualmente e ajustados às suas reais necessidades, mesmo que seja apenas em moldes virtuais pode contribuir para este fim. Por outro lado, uma vez que o utilizador é desafiado, em alguns casos mesmo sem o apoio de nenhum especialista de saúde, é ainda mais importante que receba, sempre que se justificar, *feedback* da sua prestação. Através do *feedback*, o utilizador receberá informação sobre os seus erros, com objetivo de os tentar superar, e sobre o seu sucesso nas tarefas, como fator promotor de motivação e aprendizagem

2.3 Jogos Online

A evolução das novas tecnologias, para além de procurar a novidade, aposta também na melhoria e facilidade dos métodos já existentes. Em 1991, foi dado mais um grande passo no mundo dos jogos, esta forma de divertimento foi transferida para a Internet. Inicialmente a comunicação entre jogadores exigia grandes conhecimentos técnicos, o que limitava a gama de utilizadores finais. Por esta razão, a dificuldade de acesso e a lentidão da mesma foram das primeiras barreiras a serem eliminadas pelos peritos informáticos.

Os jogos online, sejam eles de carácter educativo ou lúdico, podem trazer várias vantagens, designadamente:

- ultrapassar as dificuldades associadas às tarefas de instalação e configuração, por vezes árduas [KCS11]
- permitir comunicação entre diferentes jogadores em tempo real;
- favorecer a interação entre utilizadores de diferentes locais, o que permite cooperação e competição [Yee06];
- serem acessíveis em qualquer computador, sem qualquer instalação prévia;
- o código final ficar mais facilmente disponível para futuras alterações;
- facilitar a inclusão de novas funcionalidades dinamicamente [KCS11].

O crescimento dos jogos online foi notório e espetável, após a demonstração de todas vantagens a que estão associados.

2.4 Trabalhos Relacionados

Atualmente, já existem vários jogos sérios com o objetivo de auxiliar os métodos tradicionais na reabilitação neurocognitiva. Alguns dos produtos foram desenvolvidos com o intuito de contribuírem para a avaliação neuropsicológica do utilizador, providenciando aos profissionais de saúde um *feedback* objetivo do estado do paciente. Outros jogos foram criados para treinar as diferentes funções cognitivas, sejam elas a atenção, a memória ou as funções executivas.

As seções seguintes apresentam exemplos reais de trabalhos já realizados nesta área: o *RehaCom* e o *Virtual Like Skills*.

2.4.1 *RehaCom*

O *RehaCom* [Reh] surgiu em 1986 pelo interesse da empresa Alemã, Hasomed GmbH, em desenvolver uma aplicação de suporte à reabilitação cognitiva de pacientes com lesão cerebral. Desde então que esta empresa, focada em desenvolver software para suporte médico, tem vindo a melhorar este produto, no sentido de o aproximar à vida quotidiana dos pacientes e de modelar o sistema às necessidades que estes têm vindo a apresentar.

Uma equipa constituída por neurologistas, psicólogos, engenheiros biomédicos e engenheiros informáticos trabalharam para conseguir o atual *RehaCom*. Este visa, não só a restituição das funções perdidas, como a possibilidade da sua substituição ou compensação. Os centros hospitalares optaram por este sistema de reabilitação não com o intuito de substituir os métodos anteriormente utilizados, mas sim como forma complementar de acelerar o processo, e de baixar consideravelmente os custos geralmente associados.

O *RehaCom* foi ainda desenvolvido para facilitar a intervenção dos terapeutas, portanto alguns fatores como a facilidade de utilização, a rapidez e a facilidade de configuração foram requisitos à partida estabelecidos. Na aquisição deste produto, os utilizadores (terapeutas e pacientes) não necessitam de nenhuma formação para se sentirem integrados com a aplicação. Por esta razão, tornou-se possível transportar a aplicação da mesa do contexto clínico para a casa do doente, visto que os terapeutas têm total acesso aos resultados obtidos pelos seus pacientes. Sempre que achar necessário, o terapeuta poderá observar a evolução, através da intranet do hospital ou do servidor *Hasomed RehaCom* (internet).

As lesões cerebrais em que este program visa intervir podem resultar de diferentes tipos de acidentes e podem ocorrer a qualquer pessoa, como traumatismo crânio-encefálico ou acidente vascular cerebral. Assumindo então este facto, a empresa Alemã viu-se forçada a tornar esta aplicação o mais ampla possível e ao mesmo tempo personalizável. Através do *RehaCom* os terapeutas têm a possibilidade de definir exercícios, como: atenção e concentração, memória, pensamento lógico e planeamento de ações, noção espacial, distúrbios visuais. Para além dos diferentes tipos de exercícios, há ainda a possibilidade de reajustar o seu nível de acordo com as capacidades e idade dos pacientes. Ainda no sentido de facilitar a utilização do *RehaCom*, a Hasomed GmbH criou um teclado especial, denominado de *RehaCom Custom Panel*, que conta apenas com as teclas essenciais em dimensões maiores do que é usual (Figura 2.1). Há também a possibilidade de utilização do teclado e rato de um computador normal, no entanto, tal pode ser impraticável, por exigir capacidades dos pacientes de que eles se encontram privados.

Num futuro próximo, a equipa envolvida no desenvolvimento do *RehaCom*, conta tornar a aplicação ainda mais realista, usufruindo das vantagens da multimédia, e providenciar uma forma de mostrar ao paciente a sua evolução cognitiva diária.



Figura 2.1: *RehaCom Custom Panel* [Has]

2.4.2 *Virtual Life Skills*

David Brown, professor na Universidade de Nottingham, tem estado desde 1990 envolvido no estudo e desenvolvimento de ambientes virtuais para educação de pessoas com necessidade especiais [BNC99]. Após alguns anos de investigação nesta área, propôs o projeto *Virtual Life Skills*, que pretende orientar pessoas com dificuldades de aprendizagem na resolução de problemas do dia a dia, tornando-as mais autónomas em ações tão simples como atravessar a rua [BKB98].

Virtual Life Skills tem como objetivo desenvolver a capacidade de viver de forma autónoma. O jogo proporciona experiências baseadas em atividades do mundo real de motivação e, encorajamento à utilização de transportes públicos e de sensibilização à poupança. Torna-se então claro que esta aplicação não pretende ser mais um método de avaliação neuropsicológica ou reabilitação cognitiva, mas sim uma forma de estimular as capacidades funcionais de cada utilizador.

Nos primeiros passos do projeto foram desenvolvidos independentemente três ambientes virtuais distintos: supermercado, café e casa virtual. O *Supermercado Virtual* (Figura 2.2), baseado num supermercado real localizado em Nottingham, tem como objetivo transmitir conhecimentos sobre como fazer compras e incutir algumas noções de gestão de dinheiro. No *Café Virtual* (Figura 2.3), baseado no *University of Nottingham's Art Center Cafe*, são exploradas capacidades de socialização e comportamentais (relação com o álcool e formas adequadas de se vestirem). No ambiente casa virtual (Figura 2.4) pretende-se transmitir o máximo de conhecimentos para tornar o utilizador mais confiante na sua relação com a casa, com quartos, cozinha, sala e casa-de-banho. Na casa o utilizador pode aprender a associar cada tarefa à sua respetiva divisão, e desenvolver capacidades de lidar com situações de emergência, explorar o vestuário adequado de acordo com as condições climáticas e a combater alguns dos seus medos, designadamente na utilização de objetos cortantes, como facas.

Jogos



Figura 2.2: Supermercado [BNC99] Figura 2.3: Café [BNC99] Figura 2.4: Casa [BNC99]

A aplicação transportes virtuais surgiu como elo de ligação entre os três ambientes anteriores desenvolvidos. Trata-se de uma cidade e autocarros modelados segundo a realidade de *Nottingham*. Através deste novo ambiente, o utilizador pode navegar entre casa, supermercado e café, sendo que para isso basta que aprenda as noções de viver de forma segura numa cidade (forma como deve e onde deve atravessar a rua) e de como usar os transportes públicos (entrar no autocarro correto, pagar e receber o bilhete da viagem).

2.5 Sumário

Os jogos sérios, apesar de não focarem os seus objetivos no entretenimento, têm um papel importante nesse campo. Na área da saúde, a componente de motivação e interesse que os jogos proporcionam são essenciais para o sucesso dos tratamentos. Os jogos sérios oferecem maior interação com o jogador, mais realismo em cenários tridimensionais e possibilitam ainda a passagem do divertimento para a Internet. Os jogos *online* permitem o controlo do desempenho dos jogadores à distância e são facilmente acessíveis a partir de qualquer computador.

Existem já alguns jogos sérios desenvolvidos na área da reabilitação neurocognitiva, tais como o *RehaCom* e *Virtual Life Skills*. No entanto, falham ainda pela ausência de uma evolução progressiva e automática entre níveis, por não permitem que os terapeutas possam consultar esporadicamente o desempenho dos seus pacientes e ainda porque os cenários e a história de jogo não estão orientados à realidade portuguesa (Tabela 2.2).

Tabela 2.2: Análise do Estado da Arte e comparação com o *Virtual City*

	<i>RehaCom</i>	<i>Virtual Life Skills</i>	<i>Virtual City</i>
Cenários 3D	✗	✓	✓
Orientado à vida real	✗	✓	✓
Periféricos ajudados às necessidades	✓	✓	✓
Acompanhamento à distância	✓	✗	✓
Progressão automática entre níveis	✗	✗	✓
Histórico de performance	✗	✗	✓
Orientada à realidade portuguesa	✗	✗	✓

Capítulo 3

Ferramentas

O presente capítulo destina-se à descrição e análise das ferramentas disponíveis para a resolução do problema levantado no primeiro capítulo, um jogo sério para a reabilitação neurocognitiva. Na primeira secção será feito um estudo sobre os motores de jogo, seguido das ferramentas para a modelação tridimensional e na terceira secção é analisada a oferta de dispositivos de entrada de suporte ao jogo. No final será feito um breve sumário com as conclusões do capítulo.

3.1 Motores de Jogo

Motor de jogo é um programa que permite desenvolver aplicações, incorporando os elementos básicos para o seu desenvolvimento: componente física, interface gráfica do utilizador (GUI), inteligência artificial, som, comunicação em rede, entre outros [FK04]. A renderização das cenas 2D ou 3D é feita através de motores gráficos. O motor físico põe em prática as leis da física nos modelos, jogando com componentes como a velocidade, resistência do ar e deteção de colisões. Desta forma, é possível afirmar que os motores de jogo são o coração dos jogos [FK04], um conjunto de módulos que permitem que os programadores se foquem mais no conteúdo do produto do que no conteúdo técnico [Gol04]. Em 1990 surgiu o conceito de motores de jogo, que abriu portas às linguagens de alto nível durante o desenvolvimento de aplicações com gráficos. Para além disso, os computadores apresentavam melhor suporte físico, com memórias mais aceitáveis e gráficos de alta resolução [BEW⁺98].

3.1.1 Análise geral das soluções

O mercado dos jogos caracteriza-se pelo seu dinamismo, e por consequência existem atualmente diversas soluções para motores de jogo. Greenwood [GSRC09] destacou o *Torque*, *Quest3D*, *Unity3D* e *Virtools*, por serem os de maior sucesso comercial. A tabela 3.1 resume algumas das características destas ferramentas, juntamente com o *NeoAxis Engine*, por ter sido o motor de jogo usado na primeira fase do *CARP-VR*.

Ferramentas

Tabela 3.1: Análise de motores de jogo [Gar12] [Dev11b] [Tec12] [Sys] [Ese] [Ltd]

	Plataforma			API		Preço (€)
	Windows	Linux	Mac OS X	Linguagem	Gráficos	
<i>Unity3D</i>	✓	✓	✓	C#, Boo, JavaScript	DirectX, OpenGL	Gratuito; 1137.4
<i>Torque</i>	✓	✓	✗	TorqueScript	OpenGL	135.7
<i>Quest3D</i>	✓	✗	✗	C/C++	DirectX	757.5
<i>Virtools</i>	✓	✗	✓	C/C++	DirectX, OpenGL	?
<i>NeoAxis</i>	✓	✓	✓	C/C++, C#	DirectX, OpenGL	Gratuito; 299.5

Não foi contemplado na tabela se as aplicações estariam preparadas para produção em ambiente web, nem se permitiam interação com dispositivos de entrada de dados, por todas elas apresentarem estas funcionalidades. Pela análise da tabela, é possível concluir que as ferramentas *Torque*, *Quest3D* e *Virtools* não são compatíveis com todos os sistemas operativos e não disponibilizam versões gratuitas. O *Unity3D* e o *NeoAxis* apresentam características idênticas, não considerando a linguagem de programação por não ser relevante para a escolha do motor de jogo.

Sendo a portabilidade um dos fatores mais relevantes para a decisão, as ferramentas que não são multiplataforma foram descartadas. Neste sentido, as opções de escolha ficam limitadas ao *Unity3D* e ao *NeoAxis Engine*.

3.1.2 NeoAxis

O NeoAxis [Ltd] é um motor de jogo reconhecido pelas diversas ferramentas que disponibiliza e por permitir com facilidade o desenvolvimento de jogos de entretenimento, jogos sérios e jogos educativos (Figura 3.1).

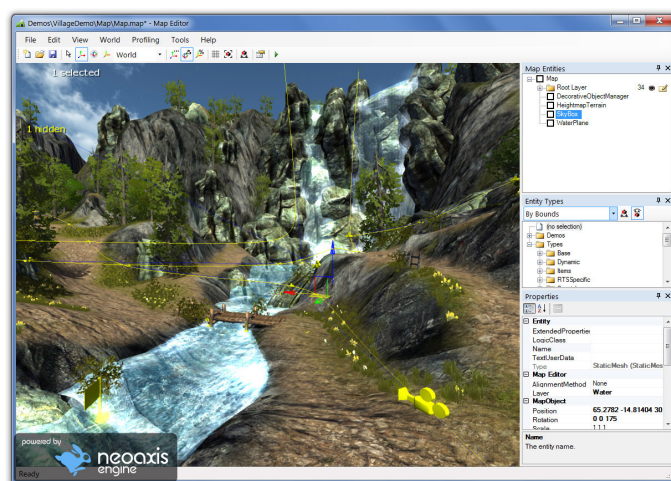


Figura 3.1: Interface de utilização do *NeoAxis* [Ltd]

3.1.2.1 Especificações Técnicas

O *NeoAxis* utiliza *C#*, mas é possível usar código de qualquer linguagem suportada por *.NET* e qualquer biblioteca *C/C++*. Esta ferramenta destaca-se pelas características:

- fácil integração com dispositivos de entrada de dados, como o *joystick* ou o *kinect*;
- capacidade em gerar *plugins* para produção web, através do *NeoAxis Web Player*;
- criação de modelos cliente-servidor para até sessenta e quatro jogadores;
- multiplataformas.

3.1.2.2 EMSAVE - Exemplo de um jogo sério desenvolvido com *NeoAxis*

EMSAVE, *Emergency Medical Services for the disabled*, é um sistema para treino de enfermeiras em situações de urgência médica. Todo o ambiente é tridimensional, incluindo todos os modelos representativos dos objetos de emergência (Figura 3.2).



Figura 3.2: EMSAVE - Exemplo de jogo sério desenvolvido com *NeoAxis* [Ltd]

A simulação inclui interação verbal entre os avatares e poderá haver colaboração com um segundo enfermeiro. Durante o tratamento do paciente, para além da demonstração do processo em ambiente virtual, a aplicação mostra também, de forma detalhada, a mesma situação em ambiente real, através de um vídeo.

3.1.3 *Unity3D*

Unity3D é um dos motores de jogos mais conceituado [WMZ⁺10], pela quantidade de funcionalidades que oferece, sem exigir grandes conhecimentos técnicos [CBM07]. É uma ferramenta distinta pela sua simplicidade e rapidez de desenvolvimento, sendo caracterizada como WYSIWYG, ou seja, *What you see is what you get* [LK11] (Figura 3.3).

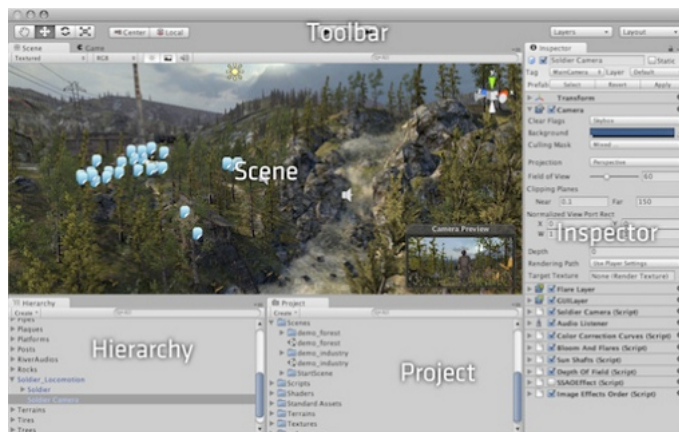


Figura 3.3: Interface de utilização do *Unity3D* [Tec12]

3.1.3.1 Especificações Técnicas

Unity3D [Tec12] foi escrito em *C#* e *C++*, mas suporta *JavaScript*, *C#*, *Boo* (inspirado em *Python*) e qualquer biblioteca *.NET*. Por serem usadas linguagens de *scripting*, o *Unity3D* proporciona um comportamento iterativo entre objetos [CBM07] e permite que sejam adicionadas novas funcionalidades a aplicações já terminadas. O *Unity3D* usa o *NVIDIA PhysX* [Cor12] de forma embutida para simular a componente física [CBM07] e o *Direct X* e o *OpenGL* para a renderização dos elementos [CBM07]. Para além disso:

- utiliza o paradigma *drag-and-drop* (arrastar e largar) [CBM07] [Tec12] [LK11];
- usa um código único para a produção web, em consola, computador ou até *smartphones* [Tec12];
- dispõe da funcionalidade *play and test*, ou seja, o programador poderá correr a sua aplicação no editor e alterar valores, sem parar a pré-visualização [CBM07] [Tec12];
- a componente física tem em conta a massa, resistência, elasticidade, limites e colisões [CBM07] [LK11];
- faz renderização de sombras e texturas [CBM07] [LK11];
- suporta integração com dispositivos de entrada de dados, como por exemplo, o *joystick*.

Para além das vantagens já expostas, o *Unity3D* dispõe de uma comunidade online bastante ativa e toda a API, *Application Programming Interface*, encontra-se bem documentada. Para o desenvolvimento de aplicações não comerciais, esta empresa propõe uma versão gratuita, que não está integrada com um sistema de controlo de versões [LK11], no entanto continua a ser multi-plataformas [CBM07] [Tec12].

3.1.3.2 CliniSpace - Exemplo de um jogo sério desenvolvido com *Unity3D*

O CliniSpace é resultado das aplicações já desenvolvidas usando *Unity3D*, um jogo sério para o treino de profissionais de saúde de forma independente ou em equipa. A aplicação pretende usar ambientes virtuais 3D, sendo fiel à realidade dos hospitais (Figura 3.4). Para além disso, a aplicação segue um conceito colaborativo e foi desenvolvido num ambiente online [Dev11a].



Figura 3.4: CliniSpace - Exemplo de jogo sério desenvolvido com *Unity3D* [Sol09]

A ferramenta é rica em funcionalidades e de alta qualidade gráfica. A simulação dispõe de um leque de ofertas de ações que normalmente seriam utilizadas na vida real. O jogador pode consultar os dados do paciente, em monitores típicos neste ambiente hospitalar. O estado do doente está sujeito às condições ambientes proporcionadas no cenário [Dev11a]. Ao longo da simulação, está sempre disponível a opção de pedir ajudar hospitalar, visto se tratar de uma treino.

3.2 Modelação 3D

A modelação 3D de filmes, jogos ou ambientes virtuais exige bastante dedicação e tempo. É um grande desafio transpor a complexidade e diversidade da realidade para ambientes virtuais [MM11]. Existem várias soluções que já permitem obter um resultado final semelhante ao real, não descurando a qualidade e detalhe das imagens. No entanto, estas ferramentas implicam elevado conhecimento e treino técnico para que o utilizador se sinta confortável e confiante durante a sua utilização. Para além disso, a criação destes ambientes obriga a modelação individual de cada componente, mesmo que se tratem de repetições. Desta forma, houve a necessidade de transformar esse tempo em trabalho útil, surgindo o conceito de modelação procedimental.

3.2.1 Modelação Procedimental

Como referido anteriormente, o desenvolvimento de ambientes virtuais exige uma grande dedicação, dado que a modelação individual de cada componente é um processo árduo e que exige alguma prática, tornando esta fase dispendiosa e demorada. A geração de modelos 3D de ambientes urbanos trouxe vários problemas para a computação gráfica, devido à quantidade e complexidade da informação a receber [CS03]. Por esta razão, surgiu a modelação procedimental, que tem como objetivo gerar de forma automática ou semiautomática modelos complexos, usando regras

e/ou algoritmos. Com este método foi possível eliminar o tempo gasto em repetições de modelos, dando lugar à criatividade e melhoramento do produto final [WMW08].

Por ser uma forma automática de geração de modelos, que permite poupar tempo e dinheiro, é uma estratégia cada vez mais usada no mundo do cinema [LWR⁺04]. Através da modelação procedimental, tornou-se fácil, rápido e de alta qualidade criar cenários para filmes. A simplicidade deste método atraiu também profissionais da área de planeamento urbano e da arquitetura, pela facilidade e baixo custo das simulações [LWR⁺04]. Como também seria de esperar, a modelação de forma automática traz vantagens no contexto dos jogos, pelas ferramentas que oferecem e que aceleram o processo de desenvolvimento [WMW08].

3.2.2 *City Engine*

City Engine [ESR12] é uma aplicação de geração de modelos 3D de forma automática, criada em 2008 pela empresa *Procedural Inc.* Esta ferramenta é considerada uma das mais maduras [WMW08] e mais utilizadas na modelação procedimental, pela facilidade e precisão em gerar modelos tridimensionais. O *City Engine* é especializado na modelação de ambientes urbanos, podendo copiar mapas de cidades reais importadas do *OpenStreetMap*.

O processo deve iniciar-se por construir a rede de ruas da cidade e a divisão dos lotes (topo da figura 3.5). Após a estruturação do mapa da cidade, é possível a modelação tridimensional da cidade, bem como a inserção da camada das texturas (meio e fundo da figura 3.5). A modelação funciona através da definição de regras específicas e a atribuição dessas mesmas regras a partes específicas do mapa.

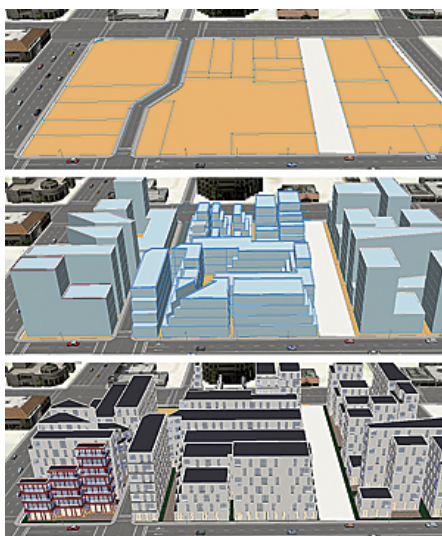


Figura 3.5: Processo durante a modelação no *City Engine* [ESR12]

3.3 Dispositivos de Entrada 3D

Apesar das vantagens resultantes da associação das novas tecnologias à reabilitação, os pacientes podem apresentar algumas dificuldades na utilização das ferramentas de interação. O problema resulta do facto das pessoas com lesão cerebral adquirida, para além das limitações cognitivas, geralmente apresentarem incapacidades físicas. Algumas experiências com potenciais utilizadores revelaram que regra geral estes pacientes não mostram destreza suficiente para o manuseamento do teclado e do rato *standard* [Reh]. Neste sentido, surgiu a necessidade de estudar algumas soluções para facilitar a sua interação com o programa.

3.3.1 Visão Geral

Os dispositivos 3D distinguem-se dos restantes por oferecerem mais do que dois graus de liberdade¹ e por orientarem as suas funcionalidades ao suporte de aplicações de contexto tridimensional. Quanto maior é o grau de liberdade dos dispositivos de entrada, mais intuitivo é o movimento durante a sua utilização. Para uma maior aceitação dos utilizadores, o grau de liberdade das rotações e translações, deve ser máximo, tornando os movimentos mais intuitivos [BBK⁺97].

3.3.2 Apontadores

Os controladores de movimento são periféricos que oferecem total liberdade de movimentos e que são frequentemente utilizados em jogos 3D, tal como mostra na Figura 3.6. Estes dispositivos foram desenhados para utilização aérea, ou seja, o controlo de movimentos exige que o braço e o comando em uso não estejam pousados numa superfície.

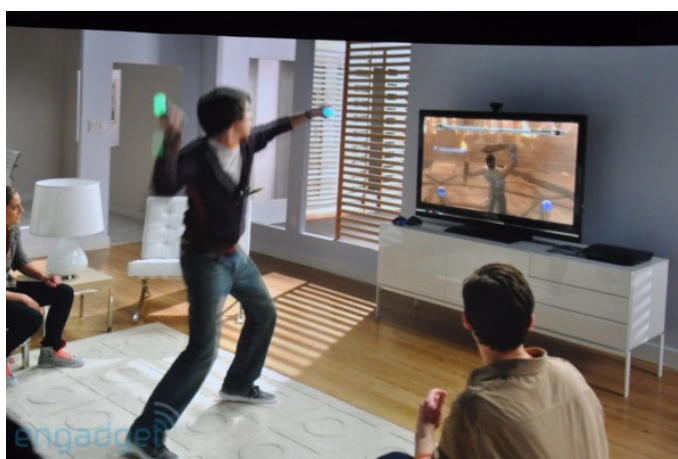


Figura 3.6: Controlador de movimento da Playstation [Son]

Devido à total liberdade dos movimentos, o uso destes dispositivos torna-se natural e evidente para os seus utilizadores. Para além da fácil utilização, os controladores de movimento permitem

¹Graus de liberdade: número de direções em que é capaz de se mover



Figura 3.7: SpaceNavigator [3DCb]



Figura 3.8: SpaceExplorer [3DCa]



Figura 3.9: Spheric [Axs10]

uma interação pessoa-computador rápida e eficaz [Zha98]. No entanto, apesar da liberdade de movimentos, existe limitação das operações possíveis, pela impossibilidade de realizar movimentos se estes forem no mesmo espaço físico. Uma das maiores desvantagens é a forma de manuseio, os ratos aéreos exigem que o utilizador mantenha o braço no ar durante a utilização, o que pode ser limitante para determinado tipo de população.

3.3.3 Rato 3D

Estes equipamentos podem ser utilizados em documentos de texto, jogos, reprodutores de música, programas de modelação e na navegação em ambientes 3D. Os dispositivos são desenhados no sentido de serem de fácil adaptação, utilização e aprendizagem. Dois casos de estudo revelam que os utilizadores confiam no equipamento, devido à simplicidade de controlo e precisão no posicionamento [3DCb] [3DCa].

A utilização destes dispositivos poderá exigir alguns conhecimentos técnicos, pela dificuldade de sentir controlo total dos movimentos e no caso de ser necessário configurar às aplicações em uso [Zha98]. Por outro lado, os ratos de secretária 3D revelaram enumeras vantagens para o utilizador, como:

- o movimento não ser feito através da deslocação física do rato [BBK⁺97];
- o contacto direto com uma superfície plana permitir uma maior precisão [BBK⁺97];
- ao contrário dos ratos aéreos, o braço poder estar apoiado numa superfície, evitando cansaço durante a utilização do equipamento [Zha98];
- os movimentos podem ser feitos de forma mais suave e estável [Zha98].

Os ratos representados nas figuras 3.7 e 3.8 apresentam limitações direcionais, por apenas abrangerem seis graus de liberdade² (Figura 3.10). As restrições direcionais impedem que o utilizador possa deslocar-se naturalmente num ambiente tridimensional³ e obrigam que essas direções não contempladas sejam realizadas com mais do que uma ação⁴.

²três rotações e três translações

³as diagonais não estão abrangidas

⁴no caso de querer deslocar-se na diagonal, o utilizador terá de: 1) primeiro rodar e depois andar para a frente; 2) primeiro deslocar-se para a esquerda e depois seguir em frente



Figura 3.10: Direções possíveis com rato 3D [3DC12]

O rato esférico, representado na Figura 3.9, permite rotações de 360° , no entanto não abrange translações no espaço.

3.4 Sumário

O mercado de motores de jogo oferece diversas soluções, tornando a escolha do programa um processo demorado e complexo. Após análise de algumas das ferramentas, o *Unity3D* e o *NeoAxis Engine* mostraram ser as melhores soluções para o problema em estudo. Estas duas ferramentas permitem desenvolver para web, suportam a comunicação com dispositivos de entrada de dados e são multiplataformas. No entanto, o *Unity3D* destaca-se pela facilidade de utilização e rápida produção. Para além deste motor usar linguagens de *scripting*, permitindo adicionar novas funcionalidades, em qualquer fase do desenvolvimento. Esta característica torna-se importante por o *CARP-VR* ser aplicado em contexto clínico, onde a rápida produção e a necessidade em adicionar novas funcionalidades de forma fácil justificam a decisão de escolher o *Unity3D*.

A modelação de uma cidade virtual é um processo árduo, devido à complexidade de cada elemento da cidade e à quantidade de elementos da mesma. Por esta razão, a opção será utilizar as técnicas de modelação procedimental, recorrendo a regras e/ou algoritmos para a modelação da cidade. Para este efeito será usado o *City Engine* (3.2.2), visto ser um dos *softwares* mais maduros para a modelação de ambientes urbanos.

Durante a primeira fase do *CARP-VR*, os utilizadores interagiam com o sistema através de um *joystick*. Após alguns meses de testes, o centro de reabilitação comprovou que os pacientes apresentavam algumas dificuldades durante a sua utilização. Os controladores de movimento, apesar de oferecem total liberdade, obrigam os utilizadores a manter o braço no ar durante toda a utilização. Tendo em conta as limitações dos utilizadores finais e o ambiente tridimensional do jogo, a solução aponta para um dispositivo com dois graus de liberdade.

Ferramentas

Capítulo 4

Cidade Virtual - Descrição do jogo

Neste capítulo é feita uma descrição detalhada do jogo, desde o esclarecimento de conceitos à exposição do seu funcionamento em geral. Nas primeiras seções do capítulo serão descritos o conceito e a mecânica do jogo. Nas duas secções seguintes será feita uma pequena exposição de como o jogador receberá informações sobre o seu desempenho e de que forma ele poderá aprender ao longo do jogo. O capítulo termina com a demonstração do jogo e explicação do funcionamento da interação entre o jogador e o jogo.

4.1 Conceito do Jogo

A presente secção descreve os conceitos do jogo, como o cenário, os lotes e os transportes. O cenário trata-se da representação do local onde o jogo decorre. Os lotes são o espaço bidimensional onde são alocados os edifícios. Por fim, os transportes são os veículos que se pretendem ter disponíveis na versão final do jogo.

4.1.1 Cenário

O jogo desenrola-se num ambiente citadino, através da representação tridimensional das cidades: Gaia e Porto (Figuras 4.1 e 4.2). O cenário é composto por múltiplos lotes (4.1.2), que se distinguem pela categoria em que se inserem. Para complexificar a tarefa ao jogador e ao mesmo tempo tornar mais realista o cenário, a cidade contém alguns meios de transporte de suporte às viagens entre pontos de interesse.

4.1.2 Lotes

O lote é uma área do cenário que não está ocupada por estradas, desde edifícios a jardins. Ao longo de cada uma das cidades o jogador poderá encontrar diferentes tipos de lotes. Os lotes são classificados de acordo com a função que desempenham na sociedade, essa relação, entre o tipo

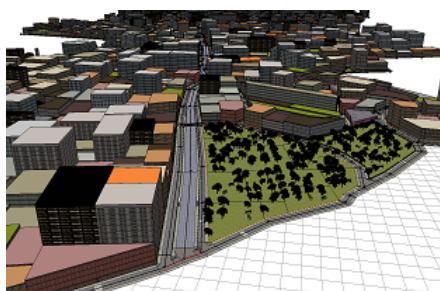


Figura 4.1: Cenário de Gaia

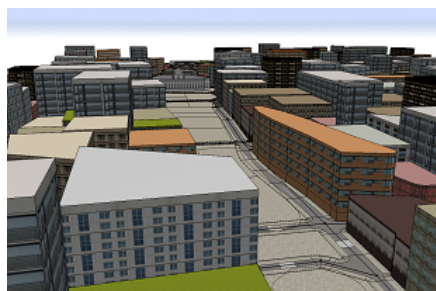


Figura 4.2: Cenário do Porto

de lote e a categoria a que pertencem, é mostrada na tabela 4.1. O objetivo não é só que o jogador saiba distinguir os diferentes lotes, mas também que saiba a que categoria pertencem, que tipos de serviços fazem para a sociedade.

Tabela 4.1: Correspondência entre lote e categoria

Categoria	Tipo de Lote
Comércio	Padaria
	Florista
	Mercearia
Serviços	Banco Correios (Figura 4.3)
Lazer	Jardim (Figura 4.4)
Social	Residência (Figura 4.5)
	Café

4.1.3 Transportes

O jogador terá disponível os seguintes transportes: carro pessoal, táxi, metro, comboio, autocarro (Figura 4.6) e a possibilidade de se deslocar a pé. Cada um destes transportes tem associado diferentes custos e tempo gasto até chegar ao destino pretendido. No caso de optar por ir de autocarro, metro ou comboio, a viagem será feita de estação em estação. Para usufruir desses transportes, o jogador deve deslocar-se à estação mais próxima, escolher o transporte correto e sair na paragem desejada. Para viajar de táxi existe um botão próprio, no lado direito do ecrã principal, que chamará um táxi ao sítio de onde foi realizada a chamada e levará o jogador até ao local



Figura 4.3: Correios



Figura 4.4: Jardim



Figura 4.5: Residência

pretendido. O carro pessoal pode transportar o utilizador desde a sua casa até qualquer parque de estacionamento dentro da cidade.



Figura 4.6: Exemplo de transporte - Autocarro

4.2 Mecânica do Jogo

“Game mechanics are rule based systems / simulations that facilitate and encourage a user to explore and learn the properties of their possibility space through the use of feedback mechanisms.” [Coo09]

Nesta secção é descrita a mecânica de jogo implementado, onde são explicados os conceitos de sessão, patamar e nível do jogo.

4.2.1 Sessão

Sempre que o paciente começa uma nova consulta, independentemente de estar acompanhado pelo terapeuta ou sozinho em casa, é iniciada uma nova sessão. Ao iniciar um jogo, o utilizador pode optar por criar uma nova sessão ou retomar uma anteriormente criada. Ambas as opções têm o mesmo significado para o paciente, visto que ele irá sempre recomeçar a partir do nível anteriormente terminado. No entanto, para o terapeuta cada uma das soluções tem um valor diferente, pela divergência de comportamentos e prestações que poderão estar associadas à sessão em que o paciente se encontra. Desta forma, uma sessão poderá incluir um nível, vários níveis ou até mesmo vários patamares.

Uma sessão começa sempre a partir do nível seguinte ao último que foi realizado com sucesso. Se o último nível em que o paciente esteve a jogar não foi completo com êxito, ele terá obrigatoriamente que o repetir e não avançará no jogo enquanto não conseguir, ainda que seja numa das sessões seguintes(Figura 4.7).

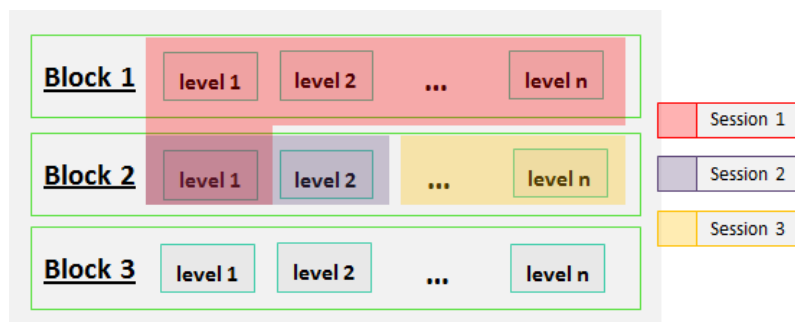


Figura 4.7: Conceito de sessão, patamar e nível [DMC⁺]

4.2.2 Patamar

Um patamar consiste num conjunto de níveis que partilham o mesmo objetivo no âmbito da reabilitação neurocognitiva. Corresponde a uma fase do jogo onde se espera que o jogador utilize a mesma estratégia de jogo. Os patamares estão dependentes entre si, ou seja, os objetivos são cumulativos e como tal o patamar atual, para além do seu objetivo principal, exige sempre conhecimentos adquiridos no patamar anterior.

A *Cidade Virtual* tem sete patamares com diferentes focos para a reabilitação, são eles:

- **Horário de Funcionamento** - o patamar oferece um conjunto de tarefas que devem ser organizadas de acordo com o horário de funcionamento dos locais onde as tarefas devem ser cumpridas.
- **Grau de Importância** - acrescenta o grau de importância de cada tarefa, aumentando assim a dificuldade no agendamento das mesmas. O jogador deve tentar conciliar o conceito de grau de importância de cada tarefa com o horário de funcionamento da mesma.
- **Caminho mais Curto** - treina a noção de distância. O utilizador deve agrupar as tarefas que são mais próximas, durante a fase de planeamento. Por outro lado, durante a fase de execução, o jogador deve realizar as tarefas utilizando o menos percurso.
- **Transportes** - o utilizador deve optar pelos transportes que lhe permitam chegar ao destino no menor período de tempo.
- **Orçamento Limitado** - introduz restrições monetárias, ou seja, o jogador deve escolher os transportes mais rápidos mas que tenham um custo inferior ao valor máximo fornecido.
- **Contra-relógio** - o desafio está na realização das tarefas propostas sem ultrapassar o tempo limite do nível. De notar que continuará a ser exigida a aplicação dos requisitos dos patamares anteriores, intervalo de tempo, importância de cada tarefa e proximidade entre tarefas.
- **Procura de Soluções** - apresenta tarefas aparentemente irresolúveis para que o jogador procure soluções alternativas, como por exemplo pedir que vá ao cinema e este encontra-se fechado, sendo esperado que o jogador realize outra tarefa de carácter lúdico.

4.2.3 Nível

Tal como foi explicado na seção anterior, um nível está integrado num patamar e como tal obedece aos critérios de avaliação por ele definidos. Nesse sentido, em cada nível o jogador é avaliado segundo esses critérios. O nível está dividido em duas fases: planeamento e execução.

No início de cada nível, na fase de planeamento, o jogador deve demonstrar que identifica o problema levantado para o nível e que consegue ordenar um conjunto de tarefas de acordo com o critério do patamar. O jogador recebe uma lista de tarefas comuns no dia a dia e deve reordená-las segundo o critério do patamar em que se encontra (Figura 4.10).

Durante a fase de execução irá treinar a aptidão do jogador em viver numa cidade, avaliando a sua eficácia na realização das tarefas. O jogador deve realizar as tarefas atendendo sempre à ordem definida na fase de planeamento.

O nível só é validado quando o jogador ultrapassa com sucesso as duas fases (Figura 4.8): planeamento e execução. O valor mínimo de sucesso é previamente definido na base de dados e deve ser ultrapassado nas duas fases, ou seja, o jogador deve ter um sucesso maior do que o valor mínimo na fase de planeamento e na fase de execução.

Quando todos os níveis de um patamar forem concluídos, o jogador avançará para o patamar seguinte. Caso o paciente consiga avançar três níveis consecutivos com sucesso será premiado e avançará de patamar, como será explicado mais à frente (ver no capítulo seguinte seção 5.7.2.2).

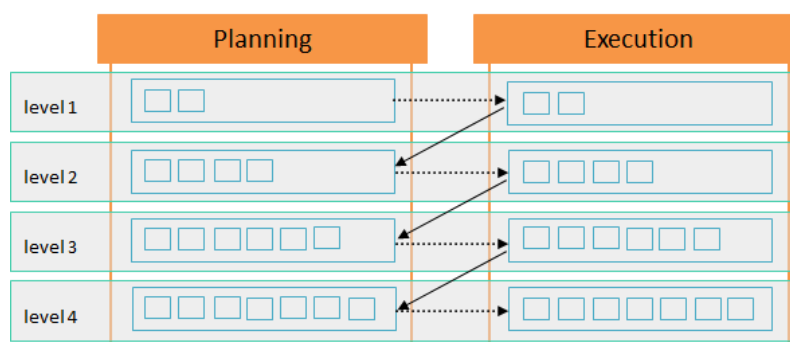


Figura 4.8: Funcionamento dos níveis num patamar [DMC⁺]

4.3 Feedback

O *feedback* serve para que o utilizador compreenda onde errou na tarefa ou durante o nível e assim saber como melhorar prestações seguintes. Idealmente, e de forma a abranger uma maior diversidade de pacientes, o *feedback* será sonoro e escrito.

Durante o jogo, o utilizador será consciencializado da sua prestação, recebendo frequentemente notificações sobre a sua performance. O sistema pode apresentar *feedback* em três condições:

- **instantâneo** - o sistema devolve uma notificação imediata e automática após um sucesso ou insucesso numa tarefa;
- **periódico** - no final de cada nível, patamar ou sessão, o jogo apresentará automaticamente um pequeno relatório sobre a performance do jogador, que poderá variar de acordo com o critério de avaliação em questão (ex: a diferença entre o caminho mínimo e caminho percorrido pelo jogador);
- **requisitado** - pacientes e terapeutas podem pedir para consultar informações adicionais, onde é possível analisar o progresso do jogador ao longo dos níveis, patamares e sessões.

4.4 Aprendizagem

O programa desenvolvido - *Cidade Virtual* - pretende desenvolver uma solução para a reabilitação neurocognitiva. Neste sentido, o jogo oferece formas de aprendizagem: apoio à decisão, *feedback* e adaptação do jogo ao nível do jogador.

O apoio à decisão é fundamental nas fases iniciais da reabilitação, para incutir e guiar progressivamente o jogador para um objetivo final. Na fase de planeamento, o jogador consegue, nos primeiros níveis, ter acesso à descrição do critério de ordenação, não obrigando a que o jogador memorize as regras estabelecidas. Durante a execução das tarefas, também unicamente no níveis iniciais, o jogador tem acesso ao mapa global com informações sobre a sua posição atual e caminho percorrido.

O *feedback*, tal como se explica na seção 4.3, varia entre instantâneo, periódico ou requisitado. São informações adicionais relevantes para a aprendizagem por “tentativa-erro” que implica que o jogador se torne mais independente e procure a solução sem apoios, mas ao mesmo tempo que desenvolva competências que lhe permitam melhorar as próximas tentativas. A aprendizagem por “tentativa-erro” não admite que o jogador avance de nível sem saber que errou, obrigando a que o repita em caso de falhanço.

Numa outra perspetiva, o sistema contribui para a evolução progressiva, continua e adaptada a cada paciente. Desde o primeiro patamar até ao último que é feito um esforço para que a aprendizagem seja progressiva, ou seja, que os conceitos aprendidos nos primeiros patamares, não sejam esquecidos durante a aquisição de novas competências.

A aprendizagem desempenha um papel fundamente durante o jogo. Após a interação e a cada erro cometidos, o jogador recebe *feedback* da sua performance e deve usar essa informação como forma de aprendizagem para nas interações seguintes o seu comportamento seja mais adequado ao problema [Coo09].

4.5 Fluxo de jogo

A presente secção descreve o fluxo de ações do jogo, representado na imagem 4.15. O esquema mostra a forma como o jogo se desenrola e como são interpretas as interações do jogador. A

presente seção apenas descreve os eventos possíveis no protótipo desenvolvido nesta dissertação, apesar do diagrama de atividades incluir a mecânica do jogo final.

No primeiro momento em que um paciente entra no jogo, são dadas instruções sobre como deve planejar as tarefas no corrente nível/patamar (Figura 4.9). Enquanto não selecionar a botão *Avançar*, o painel com a instrução do nível ficará visível para o jogador. Após confirmação, o jogador avança para a fase de planeamento e recebe um painel com um conjunto de tarefas desordenadas (Figura 4.10). Para a ordenação das tarefas, o jogador deve selecionar as setas da coluna *Ordenar*. De salientar que a ordem final irá sempre corresponder à ordem correta de ordenação, nem que para isso o jogador tenha que repetir o processo de planeamento mais do que uma vez.



Figura 4.9: Instrução para o nível



Figura 4.10: Ordenação das tarefas

Após a ordenação correta o jogador avançará para a fase de execução. Nesta fase as tarefas aparecerão num quadro secundário, posicionado no lado direito do ecrã de jogo (Figura 4.11). Na lista de tarefas, aquela que deve ser cumprida no presente momento de jogo, encontra-se destacada através do uso da cor verde e do aumento do tamanho da letra na descrição da tarefa.

Cidade Virtual - Descrição do jogo



Figura 4.11: Após seleção de um objeto do mapa

O jogador pode caminhar ao longo da cidade, usando as setas direcionais do teclado, rodando a câmara através do rato. Assim que o jogador se aproxima do local da tarefa, deve selecioná-lo, através do clique do rato. O clique dispara um novo painel (no canto superior esquerdo), com a informação sobre o objeto selecionado. Se o jogador pretender confirmar a seleção, deve pressionar do botão verde (Figura 4.11), caso contrário pode anular a operação através do botão vermelho. A validação da tarefa é feita através da comparação do local do objeto com o local da tarefa. Essa análise é visível para o utilizador, devido ao retorno de *feedback* do sistema através de mensagens de erro (Figuras 4.12 e 4.13).

O processo de seleção e aprovação das tarefas ocorre enquanto existirem tarefas por completar. No final da realização de todas as tarefas, o sistema avaliará se o jogador está preparado para avançar de nível ou não.

Durante a deslocação no cenário, o jogador pode usar o botão do mapa, representado pela imagem de um mapa com uma lupa, para ter uma perspetiva geral da cidade e ainda conseguir localizar-se no mapa 4.14. A posição atual do jogador no mapa está marcada por uma esfera amarela. Neste ponto, o jogador pode, através do *scroll* do rato, aproximar/afastar a câmara da cidade, e com as teclas direcionais deslocar a câmara vertical ou horizontalmente.



Figura 4.12: Erro na tarefa

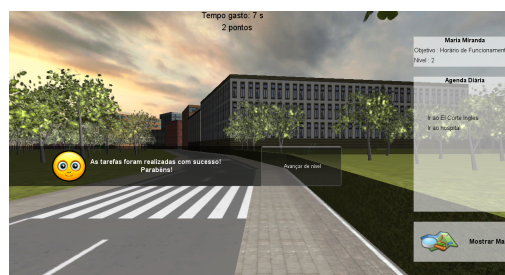


Figura 4.13: Sucesso no nível

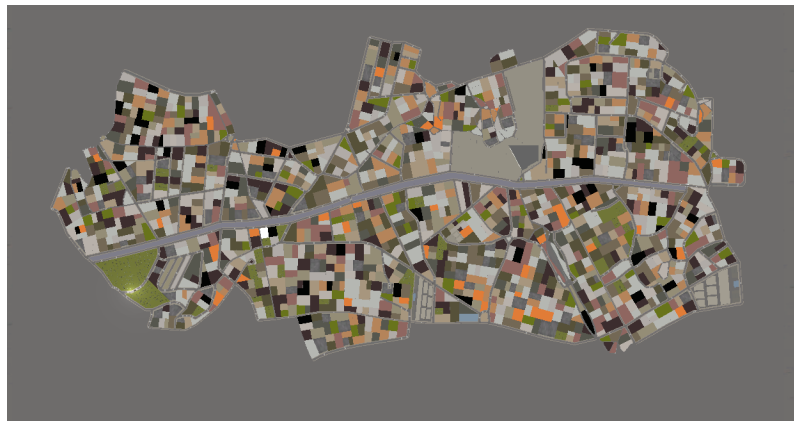


Figura 4.14: Vista ortogonal do mapa

4.6 Interação

A interação do utilizador com a personagem é feita através do joystick ou da combinação do rato com o teclado. Existem dois tipos de movimentos possíveis: translação e rotação. O sistema está preparado para translações com dois graus de liberdade¹ e rotações até 360°. As rotações são feitas através do movimento do rato e as translações com os botões direcionais do teclado. Para seleccionar objetos ou escolher opções dos menus, o jogador terá que clicar no botão esquerdo do rato.

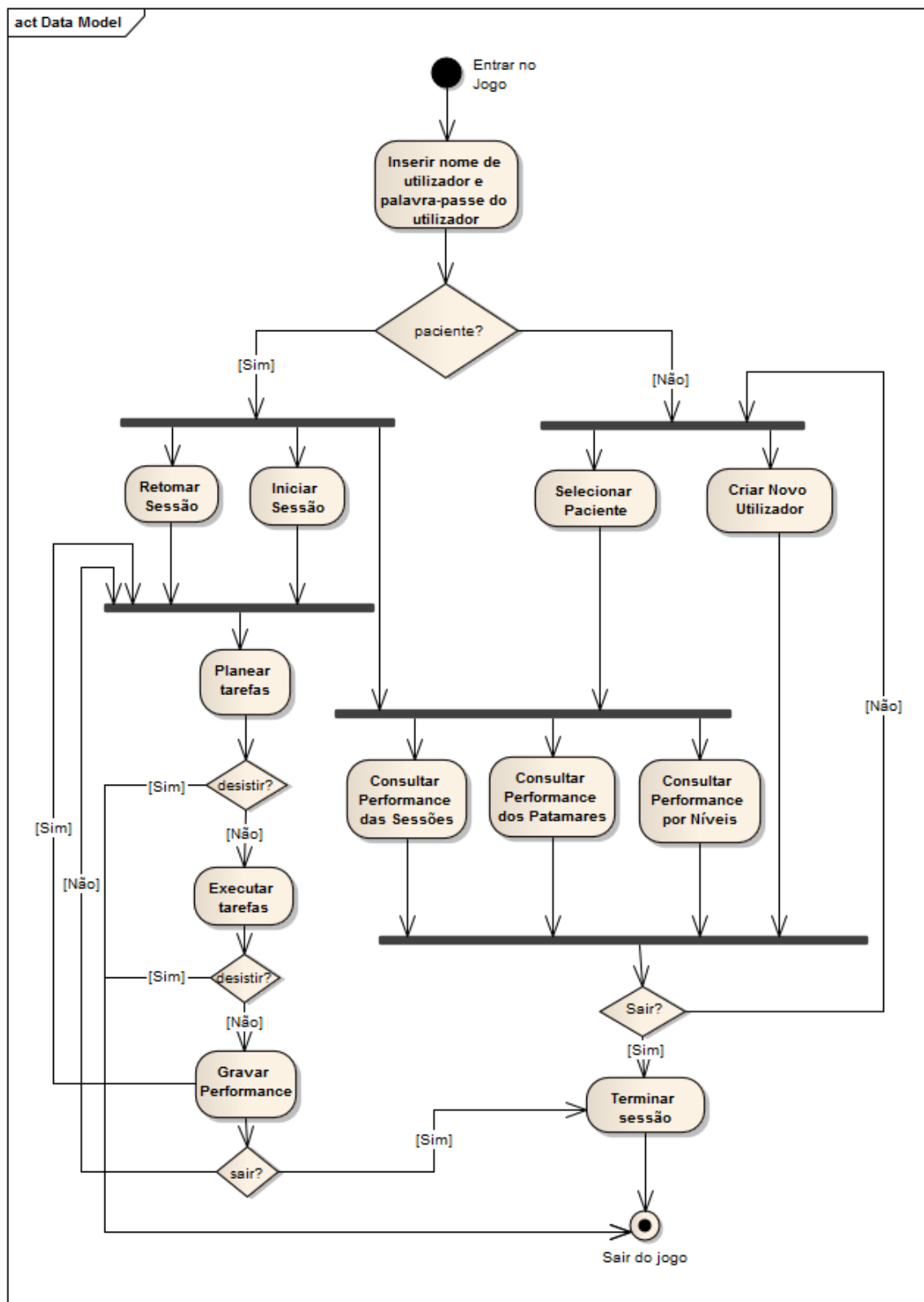


Figura 4.15: Diagrama de atividades

Capítulo 5

Cidade Virtual - Implementação

O presente capítulo descreve detalhadamente todas as fases de implementação do protótipo, explicando algumas das opções tomadas ao longo do desenvolvimento. Inicialmente é apresentada a metodologia usada durante a fase de implementação, seguindo-se da conceção do jogo (requisitos e arquitetura do sistema), explicação do processo de modelação, funcionamento da base de dados, detalhes de implementação e proposta para a avaliação da eficácia do programa.

5.1 Metodologia

Desde a fase de desenvolvimento que existe uma relação próxima com um consumidor final, no sentido de fazer levantamento de requisitos fiéis e orientar o desenvolvimento do produto às necessidades do utilizador. Uma vez que o consumidor final não possui conhecimento técnicos na área de informática, as entregas contínuas foram importantes para receber *feedback* sobre o estado atual do projeto, segundo a perspetiva de um potencial utilizador.

O desenvolvimento do projeto seguiu os parâmetros tradicionais passando pelas seguintes fases: conceção, desenvolvimento e planeia-se terminar com uma fase de testes aos potenciais utilizadores finais (Figura 5.1). A fase de conceção pode ser dividida em duas partes : levantamento dos requisitos e desenho da arquitetura. O levantamento dos requisitos foi realizado em parceria com a Dra. Artemisa Rocha Dore, responsável pelo projeto que trabalha diretamente com utilizadores finais, pessoas com lesão cerebral adquirida. O desenho da arquitetura do sistema foi estruturado após a análise dos requisitos e das tecnologias a usar. O desenvolvimento foi repartido em : modelação do cenário e implementação da mecânica do jogo. Tendo em conta que se pretendia modular um cenário facilmente adaptável a diferentes mapas, o processo de modelação no CityEngine foi mais longo do que o previsto inicialmente. Para a segunda parte de desenvolvimento foi feito um esforço para tornar a aplicação o mais modelável possível a objetos externos diferentes, ou seja, para que fosse fácil alterar apenas o cenário e todo o jogo estaria a funcionar corretamente.



Figura 5.1: Metodologia para desenvolvimento de jogos

Os testes e o *deployment* estão previstos para as fases seguintes do desenvolvimento.

Durante toda a fase de desenvolvimento do jogo, foi feito um esforço para que a interface fosse *user-friendly* e adequada às necessidades dos utilizadores finais, de forma a oferecer soluções adaptáveis.

5.2 Requisitos

Nesta secção será realizada uma descrição geral dos requisitos do sistema. Nesta descrição haverá espaço para apresentação dos requisitos funcionais (5.2.1) e não funcionais (5.2.2). Os requisitos funcionais são reflexo dos conteúdos e funcionalidades do sistema. Insere-se no grupo de requisitos não funcionais aqueles que estão associados à qualidade do *software*.

5.2.1 Requisitos funcionais

Os requisitos funcionais deste projeto podem ser divididos em três blocos: interface do jogo (5.2.1.1), lógica de jogo (5.2.1.2) e gestão de utilizadores (5.2.1.3). Os requisitos podem estar ligados entre si, ou seja, poderá existir dependência entre os requisitos nos diferentes blocos. Para além disso nem todos os requisitos têm o mesmo grau de importância, ou seja, depreende-se que para um bom funcionamento do jogo, seria suficiente que todos os requisitos classificados com “alta” prioridade estivessem implementados.

5.2.1.1 Interface do jogo

Os requisitos de interface são indispensáveis para o funcionamento adequado do jogo, no sentido de transmitir ao jogador uma aproximação realista entre a lógica de jogo e o mundo real (Tabela 5.1).

Tabela 5.1: Requisitos funcionais - Interface do jogo

Código	Descrição	Dependência	Classificação
IJ001	Mostrar Cidade Tridimensional	-	Alta
IJ002	Mostrar Cidade Bidimensional	-	Média
IJ003	Viajar dentro da cidade 3D a pé	IJ001	Alta
IJ004	Ter transportes	-	Alta
IJ005	Viajar dentro da cidade 3D, através dos transportes	IJ001	Média
IJ006	Menu inicial	-	Alta
IJ007	Iniciar jogo	-	Alta

5.2.1.2 Mecânica de jogo

Os requisitos de mecânica de jogo definem a dinâmica de jogo, as regras do jogo e o funcionamento entre níveis, patamares e sessões (Tabela 5.2).

Tabela 5.2: Requisitos funcionais - Lógica de jogo

Código	Descrição	Dependência	Classificação
LJ001	Níveis e transição entre níveis	-	Alta
LJ002	Patamares (conjunto de níveis) e transição entre patamares	LJ001	Alta
LJ003	Conjunto de tarefas por nível de acordo com o patamar em que se insere	LJ001, LJ002	Alta
LJ004	Avançar de nível se sucesso	LJ001	Alta
LJ005	Avançar de patamar se concluir todos os níveis	LJ001, LJ002	Alta
LJ006	Avançar de patamar se concluir três níveis consecutivos	LJ001, LJ002	Média
LJ007	Receber <i>feedback</i> se sucesso	-	Média
LJ008	Receber <i>feedback</i> se insucesso	-	Alta
LJ009	Organizar tarefas	LJ003	Alta
LJ010	Marcar como concluída uma tarefa terminada	LJ003	Baixa
LJ011	Marcar como concluída uma tarefa impossível, mas que o jogador encontrou uma alternativa	LJ003	Média
LJ012	Devolver o caminho mais curto	-	Média
LJ013	Por cada tarefa mostrar grau de importância, descrição e local	LJ003	Alta
LJ014	Iniciar nova sessão	-	Alta
LJ015	Retomar sessão anterior	-	Alta

5.2.1.3 Gestão de utilizadores

O bloco de gestão de utilizadores engloba os requisitos associados aos utilizadores (Tabela 5.3). Inserem-se neste grupo as funcionalidades CRUD¹, que gerem terapeutas e pacientes do sistema.

Tabela 5.3: Requisitos funcionais - Gestão de utilizadores

Código	Descrição	Dependência	Classificação
GU001	Conta de Paciente	-	Alta
GU002	Conta de Terapeuta	-	Alta
GU003	Criar conta	GU001, GU002	Média
GU004	Entrar no sistema através da conta	GU001, GU002	Alta
GU005	Sair do sistema	GU001, GU002	Alta
GU006	Editar informação pessoal	GU001, GU002	Baixa
GU007	Consultar informação pessoal	GU001, GU002	Média
GU008	Consultar histórico - níveis, patamares, sessão	-	Alta

5.2.2 Requisitos não funcionais

O principal requisito não funcional do sistema é a definição da interface e a garantia quanto à usabilidade. Considerando que o sistema é desenvolvido orientado para pacientes com lesão cerebral adquirida, a interface deve ser desenhada de acordo com o público-alvo. Desta forma,

¹CRUD - contém as funções básicas como criar, ler, atualizar e apagar (create, read, update and delete)

pretende-se que o jogo seja de fácil entendimento, prestando especial atenção aos menus em que há interação direta com os pacientes (i.e., selecionar objetos no cenário). Aliado à interface gráfica está a forma como o jogador deve interagir com o jogo. O dispositivo deve ser de fácil utilização e de fácil adaptação a todas as pessoas. As deslocções do jogador dentro do cenário devem também ser adaptadas às necessidades dos utilizadores finais. As translações e rotações não devem ser demasiado sensíveis e o dispositivos não deve permitir rotações verticais, com o objetivo de focar o jogador no cenário e não permitir que se desoriente no mapa.

Relativamente ao desempenho do jogo, espera-se que seja compatível com qualquer computador de uso corrente, de forma a que qualquer utilizador possa usufruir da aplicação web na sua casa. No entanto, deve ser garantida a confidencialidade e segurança dos dados dos pacientes e terapeutas, limitando o acesso à base de dados.

5.3 Arquitetura

Considerando os requisitos anteriormente definidos, a arquitetura do sistema, esquematizada na figura 5.2, é repartida em três blocos: interface com o utilizador, mecânica de jogo e base de dados. Os blocos representados a cinzento (Utilizador e Interface-Relatório) não estão ainda adaptados ao jogo, no entanto a mecânica de jogo e base de dados foram desenhados para essa integração.

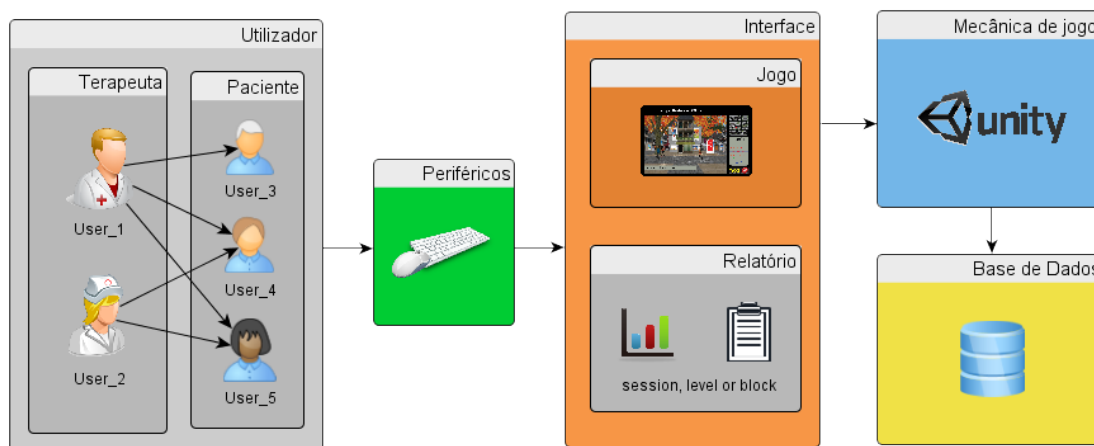


Figura 5.2: Arquitetura do sistema *Virtual City*

A arquitetura do sistema é sustentada por um conjunto de classes e ficheiros ligados entre si, esquematizada no diagrama UML B.1. As classes e ligações foram implementadas no jogo e estão agrupadas em pacotes tal como na figura 5.3. Cada pacote representa uma parte relevante no jogo, como por exemplo, num pacote é tratada a atribuição de autocarro e noutra a gestão de utilizadores. De forma geral, todos os pacotes estão ligados entre si, sendo a classe-Mãe, *Game*, a principal responsável pela comunicação entre todas as classes.

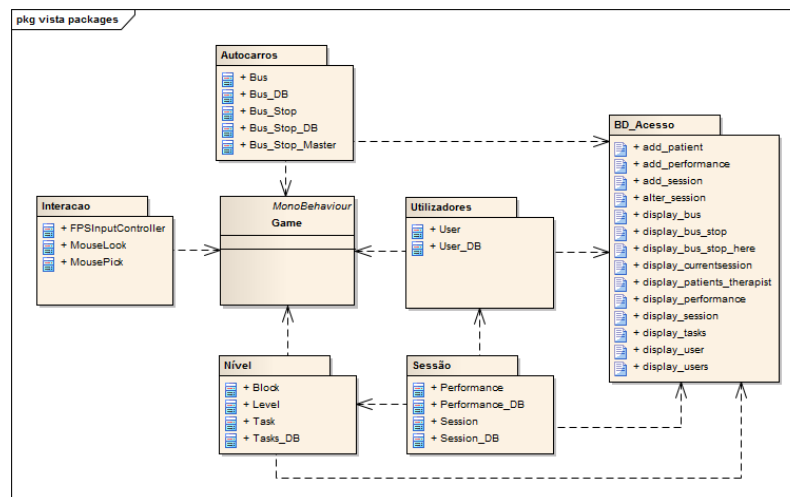


Figura 5.3: Diagrama de pacotes

5.3.1 Bloco Interface

Na seção da interface, representada no bloco laranja, incluem-se todas as funcionalidades resultantes da interação entre o utilizador e o jogo, através dos periféricos rato e teclado (bloco a verde).

Para a interação entre o utilizador e o cenário, o sistema verifica em cada *frame* (através da função *Update*) se existiu um evento do rato ou do teclado. As classes responsáveis por esta verificação estão incorporadas no pacote *Interacao* juntamente com a classe-Mãe (Figura 5.3), *Game*.

O utilizador interage com o jogo através da camada *GUI*, função definida em diferentes classes que tem como objetivo suportar a componente gráfica do jogo, por exemplo, quadro de tarefas ou instruções do nível.

5.3.2 Bloco Base de Dados

O bloco da base de dados, preenchido a amarelo no diagrama 5.2, reserva-se para a gestão de dados armazenados, tais como dados do utilizador ou de níveis. Pela análise da figura 5.3 ou da figura B.1 são perceptíveis múltiplas relações entre os ficheiros de comunicação com a base de dados com as restantes classes funcionais do sistema. Estas relações permitem a consulta de todo o percurso dos jogadores ao longos dos níveis e sessões e, assim, entender as áreas fortes dos pacientes.

5.3.3 Bloco Mecânica de Jogo

No esquema da figura 5.3, o bloco da mecânica de jogo representa-se pelos módulos níveis e sessões. Neste bloco, foram implementadas as regras do jogo, cruzando com a maioria das classes

do sistema (Figura B.1), pela necessidade de interligar componentes como os autocarros, o acesso à base de dado e a interação.

5.4 Modelação 3D

A fase de modelação é a primeira etapa do desenvolvimento. Esta secção explica todo o processo para a obtenção do cenário final. O processo de modelação da cidade divide-se em três fases: importação do mapa, modelação da cidade e exportação do resultado final (Figura 5.4).

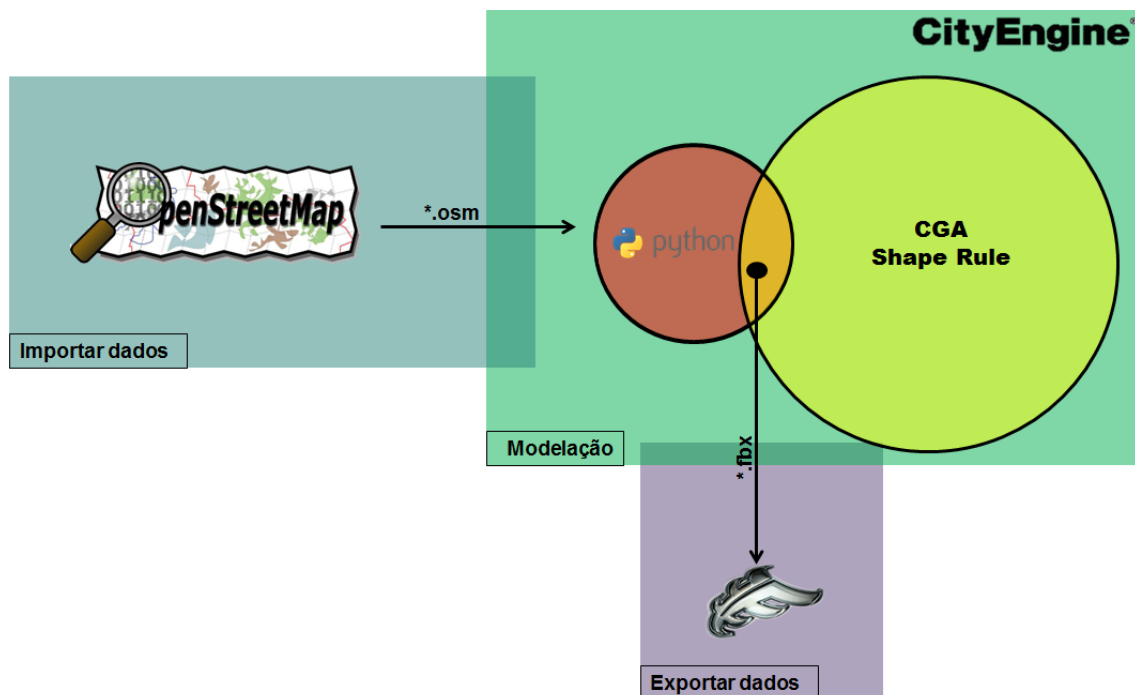


Figura 5.4: Metodologia para modelação tridimensional

5.4.1 Importar mapa - *OpenStreetMap*

No sentido de obter o maior realismo no cenário do jogo, foram exportados dados do mapa das cidades em causa. Para este efeito foi usado o *OpenStreetMap* (Figura 5.5), seleccionando a área pretendida e exportando o resultado. O *OpenStreetMap* gera um ficheiro de formato OSM XML com a informação dos nós, caminhos e relações entre eles [Ope12]. O exemplo de código em 5.1 mostra como é definido o caminho para o *El Corte Inglés*, em Gaia. O caminho é determinado pelo conjunto de nós, nome do local, categoria a que pertence (ex., superfície comercial, edifício público, farmácia, ...), entre outros atributos, que variam de acordo com os dados disponibilizados por utilizadores do sistema. Cada nó contém a informação das coordenadas latitude e longitude (ex., 5.2), o que permite que, durante a interpretação do ficheiro, sejam perceptíveis as posições dos

```

1 <way id="54369569" user="Micha Wiedenmann" uid="151490" visible="true" version="2"
  changeset="7232265" timestamp="2011-02-09T00:23:25Z">
2   <nd ref="685164706"/>
3   <nd ref="685164707"/>
4   <nd ref="685164709"/>
5   <nd ref="1143177793"/>
6   <nd ref="1143177698"/>
7   <nd ref="1143177747"/>
8   <nd ref="685164712"/>
9   <nd ref="1143177582"/>
10  <nd ref="685164706"/>
11  <tag k="landuse" v="commercial"/>
12  <tag k="name" v="El Corte Ingles"/>
13 </way>

```

Listing 5.1: Extrato de código OSM XML - caminho El Corte Inglés

nós e assim criar formas geométricas que definem as ruas e os blocos. Para a leitura do ficheiro OSM XML foi utilizado o *software* de modelação *CityEngine*.

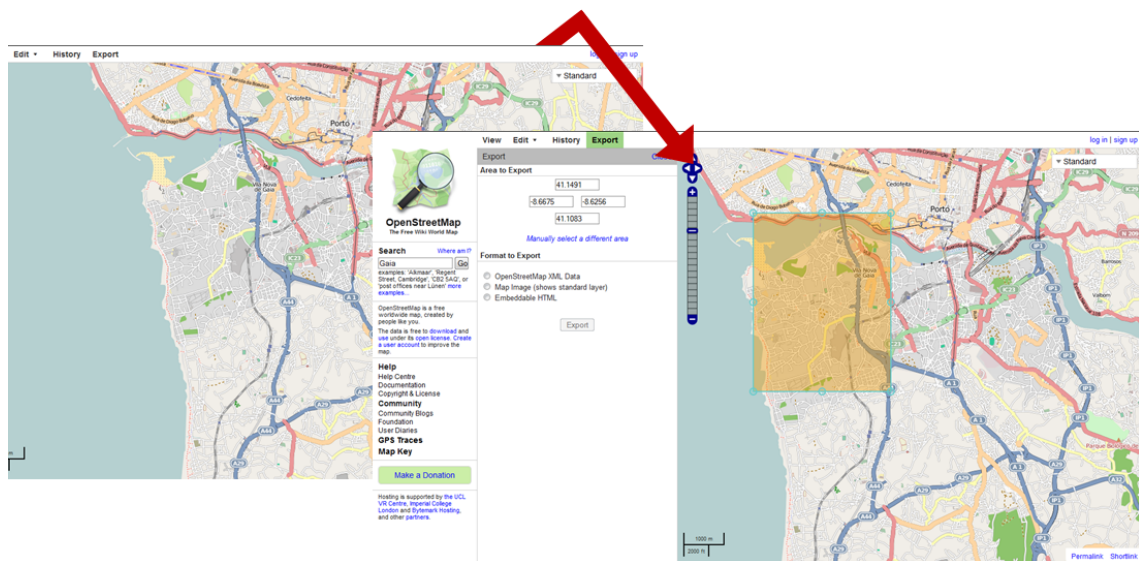


Figura 5.5: Exportar mapa para ficheiro OSM XML

5.4.2 Modelar a cidade - *CityEngine*

CityEngine é uma aplicação de modelação procedimental de objetos tridimensionais. Este tipo de modelação procura resolver a problemática da repetição de modelos, da perda de tempo e dinheiro, causadas pela complexidade durante a geração de modelos de três dimensões [WMW08]. O *CityEngine* suporta ficheiros OSM XML e interpreta os dados por ele transmitidos, resultando num mapa bidimensional (Figura 5.6) com os caminhos traçados no mapa real do *OpenStreetMap*.

```
1 <node id="685164706" lat="41.1260527" lon="-8.6044460" user="Micha Wiedenmann" uid=  
  "151490" visible="true" version="3" changeset="7232265" timestamp="2011-02-09  
  T00:23:23Z"/>  
2 <node id="685164707" lat="41.1251428" lon="-8.6044657" user="Micha Wiedenmann" uid=  
  "151490" visible="true" version="3" changeset="7232265" timestamp="2011-02-09  
  T00:23:25Z"/>
```

Listing 5.2: Extrato do código OSM XML - nós do caminho El Corte Inglés

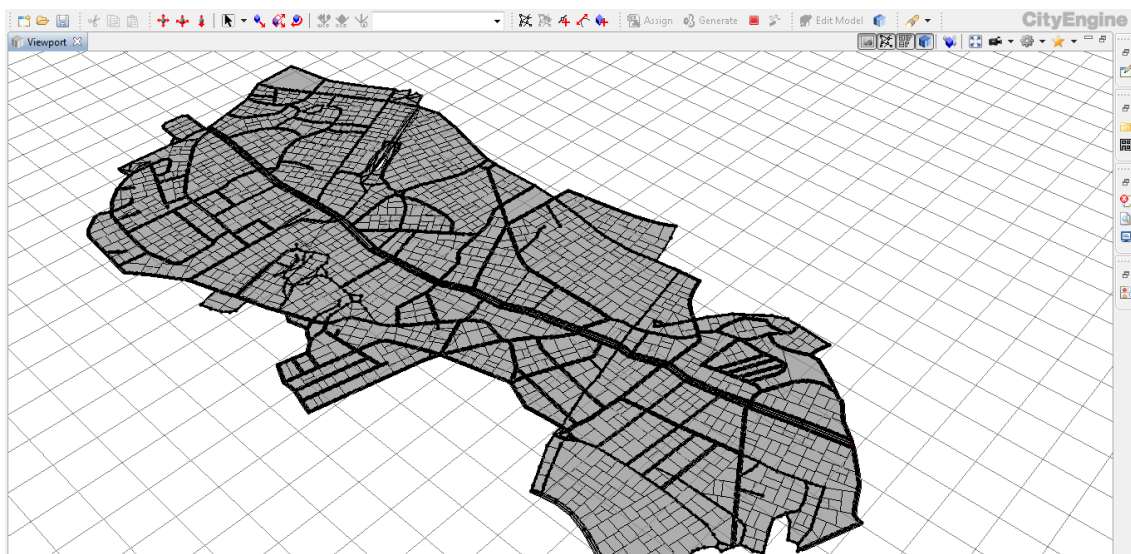


Figura 5.6: Planta do mapa bidimensional - CityEngine

Tal como se pode observar na figura 5.6, as linhas a preto representam estradas ou caminhos, os restantes blocos cinzentos estão reservados para alocar edifícios residências, comerciais ou qualquer outro espaço referente a uma cidade.

Ao contrário da maioria dos *softwares* de modelação 3D, o *CityEngine* cria objetos tridimensionais através da definição de um conjunto de regras. As regras são definidas usando *CGA shape grammar*, que através da iteratividade, consegue criar com elevado detalhe objetos 3D. O extrato de código 5.3 mostra parte das regras necessárias para criar um edifício. Existe associado às regras um conceito de aleatoriedade, pela necessidade de tornar a cidade diversa em texturas e dimensões. A aleatoriedade foi definida através de probabilidades para permitir que essa associação fosse automática, ou seja, que não fosse necessário atribuir individualmente uma regra diferente por bloco. Na figura 5.7 é possível verificar que dois blocos com a mesma regra (“Lot”), tomam texturas e dimensões incertas.

Cada bloco do cenário tem atribuído uma regra, definida pela combinação dos atributos : *Rule File* e *Start Rule*. No entanto, prevendo a necessidade de saber, em fases seguintes, o que cada bloco representa, é necessário ter também definido o *Name* para todos eles. Para este efeito, e de forma a automatizar o processo, é usado um *script* que define como *Name* de cada objeto

```

1 Lot -->
2   25% : set(height, rand(height_p*2.5, height_p*4)) BigBuilding
3   37% : set(height, rand(height_p, height_p*2.5)) MediumBuilding
4   36% : set(height, rand(height_p, height_p*1.5)) SmallBuilding
5   else : LotInner
6
7
8 LotInner -->
9   Lot
10
11 # Big Model
12 BigBuilding -->
13   20% : extrude(height) Building(big_1, color_big_1, color_big_1, "", door_new_2)
14   20% : extrude(height) Building(big_2, color_big_2, color_big_2, "", door_new_2)
15   20% : extrude(height) Building(big_3, color_big_3, color_big_3, "", door_new_2)
16   10% : extrude(height) Building(big_4, color_big_4, color_big_4, "", door_new_2)
17   else : extrude(height) Building(big_5, color_big_5, color_big_5, "", door_new_2)
18
19 # building construct
20 Building(wall, wall_color, roof, roof_type, door) -->
21   comp(f) {
22     front : Frontfacade(wall, wall_color, door) |
23     side : Sidefacade(wall, wall_color) |
24     top : Roof(roof_type, roof) }

```

Listing 5.3: Extrato do código CGA Shape Grammar

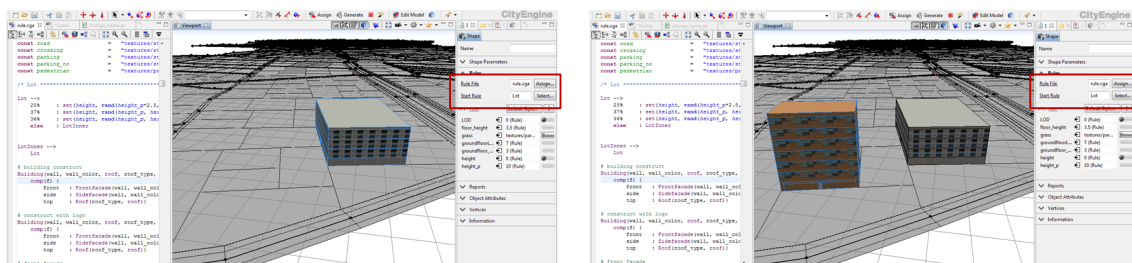


Figura 5.7: Atribuição de regras a blocos

o valor da *Start Rule*. Esta operação só é possível porque a nomeações das *Start Rule* foram escolhidas de forma a refletirem claramente a categoria específica do objeto. Um bloco ainda que use construtor, por exemplo, *Building*, foi definido com a *Start Rule* para detalhar exatamente o tipo de bloco em questão, por exemplo, *Hospital* ou *ElCorte*. O *script* foi desenvolvido usando *Python Scripting*, que altera os *Name* de todos os objetos selecionados automaticamente, através do código 5.4. Para além disso, o *Python Scripting* é usado para atribuir paralelamente regras a objetos que correspondem a edifícios relevantes no mapa real. Essa informação é retirada do ficheiro OSM XML e é guardada nos atributos de cada objeto, por exemplo, com a *tag shop*. As *tags* indicam ao *CityEngine* que, o bloco em causa, se trata de um edifício do tipo específico, por exemplo, comercial, e a estrutura do mesmo será gerada de acordo com essa informação.

```
1 from scripting import *
2
3 ce = CE()
4
5 selection = ce.selection()
6
7 for i in selection :
8     if ce.isShape(i) :
9         ce.setName(i, ce.getStartRule(i))
```

Listing 5.4: Extrato do código pyhton - Atribuição de nomes a cada objeto do cenário

No final da geração de todos os modelos, suportados pelas regras que são definidas aleatoriamente ou através dos *scripts* em *Python*, o cenário da cidade apresentar-se-á semelhante à figura 5.8.

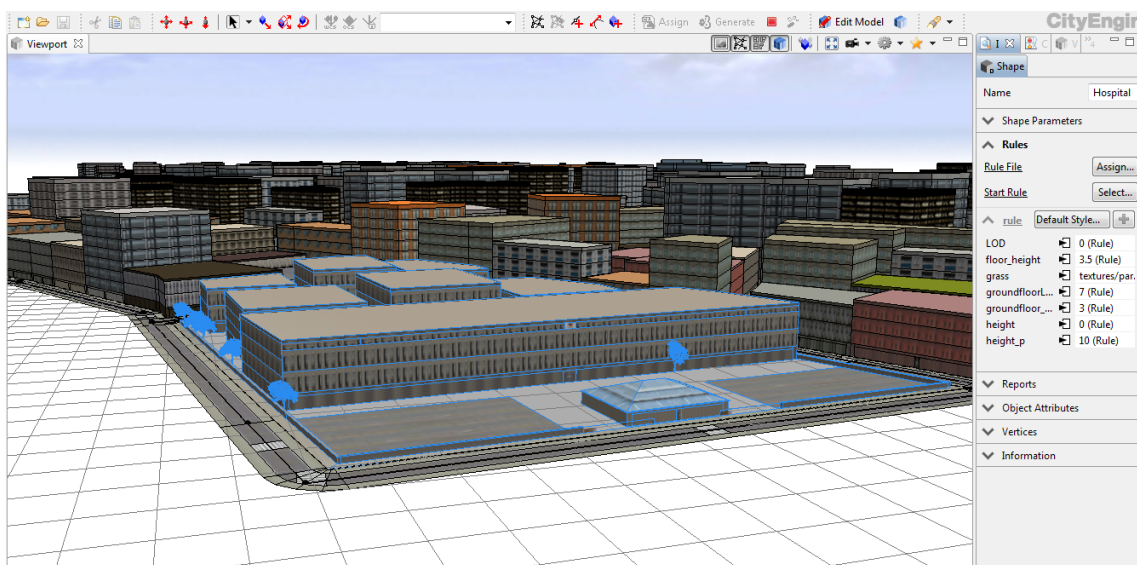


Figura 5.8: Aspeto final do cenário

5.4.3 Exportar cenário - CityEngine

O resultado final (Figura 5.8) está preparado para ser exportado e usado como cenário num motor de jogo. Antecipando a necessidade de se conseguir usar o resultado obtido na modulação da cidade no motor de jogo, foi garantido que o tipo de ficheiro exportado seria legível no motor de jogo. Avaliando os possíveis formatos de saída do *CityEngine* e comparando com os de entrada do *Unity3D*, o motor de jogo escolhido, a solução foi usar ficheiros de formato FBX.

5.5 Cena do jogo

Segundo o dicionário português, designa-se por cena o “conjunto de objetos que se oferecem à vista”, o que adaptado a este contexto se traduz pelo conjunto de elementos do jogo. Na maioria das vezes assume-se que cada cena corresponde a um nível, com o ambiente, os obstáculos e os objetos apropriados. No entanto, este projeto tem apenas uma cena que serve de base para todos os níveis. Esta decisão deve-se ao facto de se pretender evitar repetições de informação, uma vez que o cenário é inalterável durante todo o jogo. A gestão de níveis é feita através da codificação (Seção 5.7) e não pelos objetos da cena. A cena é constituída pela cidade, luzes, câmaras e paragens de autocarro.

5.5.1 Cidade

A cidade que serve de ambiente principal para jogo, foi importada para o *Unity3D* através da funcionalidade *Import Asset*, que recebe qualquer ficheiro *mesh*, texturas ou sons. Desta forma, o ficheiro FBX foi facilmente incorporado no projeto do jogo, sendo convertido para um *prefab*² do projeto. A partir deste momento, um simples arrastar do *prefab* para a *tab Scene* ou *Hierarchy* é suficiente para inserir o cenário dentro da cena do jogo.

5.5.2 Luz

Complementar ao ambiente, foi adicionada uma componente *light*, no sentido de simular a luz solar na cidade. Dentro das possibilidades para o tipo de *light*, a opção passou pela *Directional light*, por ser aquela que mais se aproxima do objetivo pretendido.

5.5.3 Câmara

O jogador não está fisicamente representado no cenário por se tratar de um FPP³, *First-person perspective*. A visão do jogador está alinhada pela perspetiva da câmara no jogo. O *Unity3D* oferece um *package*, denominado de *Character Controller*, constituído por dois *prefab*(câmaras FPP e TPP⁴, *Third-person perspective*) e *scripts* auxiliares para o movimento da câmara no cenário. Para este jogo foi usada a câmara FPS do *package*, com os *scripts* *FPSInputController*, para movimentar o jogador através do teclado e *MouseLook* para rodar a câmara através do movimento do rato. De forma a mover o jogador no cenário, basta associar à câmara aos *scripts* anteriormente referidos. O valor do *near clipping*⁵ da câmara toma o valor mais baixo possível para evitar que, na interseção com edifícios, o jogador os atravesse.

²prefab - conjunto de objetos com determinadas definições e funcionalidades. Funciona, tal como o nome indica, como um pré-fabricado ou pacote de objetos, permite que o conjunto de elementos sejam facilmente inseridos na cena, sem que seja necessário incluir individualmente os vários objetos, ajustar os seus atributos e determinar as suas funcionalidades.

³FPP - perspetiva da câmara em primeira pessoa, ou seja, o jogador não tem representação física no jogo.

⁴TPP - perspetiva da câmara na terceira pessoa, ou seja, o jogador é representado por um avatar no cenário.

⁵clipping-planes - garante que antes do plano *near clipping* e depois do *far clipping* nada será renderizado

Para além da câmara principal, o jogo disponibiliza uma segunda câmara para a vista de cima da cidade. Ao contrário da anterior, esta segunda câmara, tem uma projeção ortogonal de tamanho proporcional ao tamanho do mapa, para permitir uma visão completa do mapa (Figura 4.14). Não tem capacidade de rotação, mas pode diminuir e aumentar a distância ao chão da cidade e fazer translações no mapa.

5.5.4 Autocarros e paragens

As paragens de autocarro e os autocarros são ficheiro de formato .OBJ integrados no projeto. O *Unity3D* tem capacidade nativa para ler ficheiros desta extensão.

Prevendo que seria necessário inserir vários autocarros e paragens no cenário, foi criado um *prefab* que agrupava ambos os objetos. O *prefab* inclui o objeto da paragem, o autocarro e o caminho a fazer pelo autocarro. Cada autocarro tem incluída uma câmara interna, a ser ativada durante a viagem. O posicionamento da paragem é feito manualmente.

O uso de *prefab* traz vantagens no desenvolvimento do jogo por possibilitar a criação dinâmica de objetos e por encurtar o tempo na especificação dos elementos num grupo de objetos que são usados regularmente.

5.6 Base de Dados

A base de dados armazena os dados do jogo. Todas as funcionalidades são guardadas desde as regras à gestão dos utilizadores. Os dados do jogo foram armazenados na base de dados de forma a estarem acessíveis a qualquer pessoa e tornar possível a fácil alteração dos níveis ou critérios de avaliação, sem que para isso seja necessário aceder de forma exaustiva ao código do jogo. Sendo o principal objetivo a fácil alteração das regras de jogo, foi imposto que a arquitetura estivesse na terceira forma normal, conseguindo assim um acesso praticamente direto às tabelas que se pretendem alterar. Apesar das normalizações na terceira forma normal tornarem a pesquisa entre tabelas mais lenta, esta normalização retira redundâncias e aumenta a flexibilidade. A figura A.1 e a figura A.2 mostram como está organizada a base de dados e de que forma estão ligadas as tabelas entre si.

5.6.1 Gestão de Níveis

O conjunto de tabelas do bloco *Gestão de Níveis* agrupa todos os dados necessários para criar níveis e patamares. Esta informação nunca é manipulada durante o jogo, os dados só podem ser alterados diretamente na base de dados, para garantir a consistência dos mesmos. O bloco de *Gestão de Níveis* é formado pelas tabelas:

- **levels** - contém informações básicas sobre os níveis, tais como o bloco em que se insere ou a percentagem de sucesso mínima admissível. Esta é considerada a principal tabela do bloco, por ser aquela a que o jogo vai aceder para criar toda a lógica de jogo.

- **blocks** - representa os patamares do jogo, guardando assim apenas dados sobre o número total de níveis para o atual patamar.
- **blocks_ecriterias** - é de entidade fraca⁶ que serve como conector entre as tabelas *evaluation_criterias* e *blocks*, numa relação de muitos-para-muitos.
- **evaluation_criteria** - gere todos os dados relativos aos critérios de avaliação dos patamares, ou seja, os desafios do jogo a serem usados em cada patamar. A ligação entre esta tabela e os patamares é feita através da tabela anterior, *blocks_ecriterias*.
- **levels_tasks** - é de entidade fraca, só existe caso hajam níveis e tarefas. Os dados da tabela *levels_tasks* resultam de uma relação de muitos-para-muitos entre *levels* e *tasks*, ou seja, relaciona cada tarefa a um ou mais níveis.
- **tasks** - tal como *levels* e *blocks*, é uma das mais relevantes do bloco de *Gestão de níveis*. Nesta tabela são guardados os dados descritivos das tarefas, por exemplo, o local da tarefa ou a categoria a que pertencem.
- **taks_values** - trata-se de uma tabela reservada para uma enumeração do par (id, descrição) com a informação sobre o valor da tarefa. A tabela tem, até ao momento, os valores de “muito importante”, “importante” ou “pouco importante”.
- **categories** - à semelhança da tabela anterior, contém uma lista de pares (id, descrição) com dados de possíveis categorias de cada tarefa, por exemplo, saúde ou compras.
- **locals** - também é formado pelo par (id, descrição) com os locais possíveis para a realização das tarefas, por exemplo *ElCorteIngles*, caso se trate de compras.

5.6.2 Gestão de Sessões

O grupo de tabelas agrupadas na *Gestão de Sessões* são a base para o armazenamento de dados das sessões. Por cada sessão importa saber em que nível/patamar o jogador iniciou e terminou e qual o resultado de cada uma dessas relações entre os níveis e sessões. Na perspectiva dos terapeutas, este é o principal bloco, é neste contexto que se torna possível a análise de resultados das sessões, níveis e patamares e assim avaliar a progressão dos pacientes. Inserem-se neste bloco as seguintes tabelas:

- **sessions** - é a principal tabela do bloco. Esta tabela guarda todos os dados fornecidos durante o jogo relativamente ao nível de início da sessão, ao nível atual e ao nível quando o utilizador termina a sessão. A cada sessão está associada apenas um utilizador.
- **performance** - guarda, a cada sessão, um valor qualitativo da prestação do jogador aliado ao tempo gasto para o nível. Para além disso, deve futuramente, armazenar as posições percorridas durante o jogo.

⁶weak entity - entidade que só existe caso esteja relacionada com outras entidades

- **abandonment** - aponta as desistências do jogador por cada sessão. Interessa saber neste contexto, em que nível e em que data o jogador abandonou o jogo.

5.6.3 Gestão de Utilizadores

A *Gestão de Utilizadores* é constituída pelo conjunto de tabelas responsáveis por sustentar dados dos diferentes utilizadores, dividido entre pacientes e terapeutas. Optou-se por criar uma tabela de *users* genérica com a informação comum entre os dois tipos de utilizadores, evitando redundância de informação. Assim, este módulo gere os dados dos diferentes tipos de utilizadores, incluindo a atribuição do terapeuta ao paciente. As tabelas pertencentes a este módulo são:

- **users** - contém as informações comuns a todos os utilizadores, tais como o nome ou a idade. Com o intuito de poder aceder a informações específicas ao tipo de utilizador, foi criada uma relação de composição⁷ entre a tabela *users* e as tabelas *patients* e *therapists*.
- **patients** - é parte da composição da tabela anterior. No entanto, esta tabela está focada em guardar informação específica dos pacientes, tais como o género e a descrição de informações como a doença, o tratamento, entre outros.
- **therapists** - tal como a tabela anterior, está inserido na composição dos *users*. Neste caso, a tabela guarda valores, tais como a especialidade.
- **therapistHasPatients** - define a alocação de paciente a terapeutas.

5.6.4 Gestão de Autocarros

A atribuição de destinos a autocarros e paragens é feita de forma dinâmica. Os autocarros e as paragens são apenas objetos alocados no cenário que vão retirar informação sobre os destinos possíveis à base de dados. Para este efeito, o bloco de *Gestão de autocarros* é formado por tabelas que guardam um conjunto de valores de forma organizada. O bloco é constituído pelas tabelas:

- **bus** - tem um conjunto de possíveis destinos associados sempre ao mesmo número de autocarro.
- **bus_local** - armazena uma lista de possíveis locais de destino ou de paragem. Esta tabela está associada à anterior para a definição do destino de cada autocarro.
- **bus_stop_here** - atribui locais a autocarros e associa uma posição, resultando nas paragens para um determinado autocarro.
- **bus_stop** - determina o nome da paragem, obrigando a que esse nome pertença à lista de destinos dos autocarros.

⁷relação de composição - quando uma tabela contém os elementos da outra tabela

5.7 Implementação da mecânica de jogo

Os primeiros esforços para o desenvolvimento da mecânica do jogo foram na criação da cena e do cenário, seguida da alteração do cenário e implementação de regras (Figura 5.9). A mecânica de jogo foi desenvolvida no *Unity3D*, usando as linguagens *c#* e *javaScript*. A arquitetura usada para o desenvolvimento da mecânica de jogo está definida na seção 5.3.

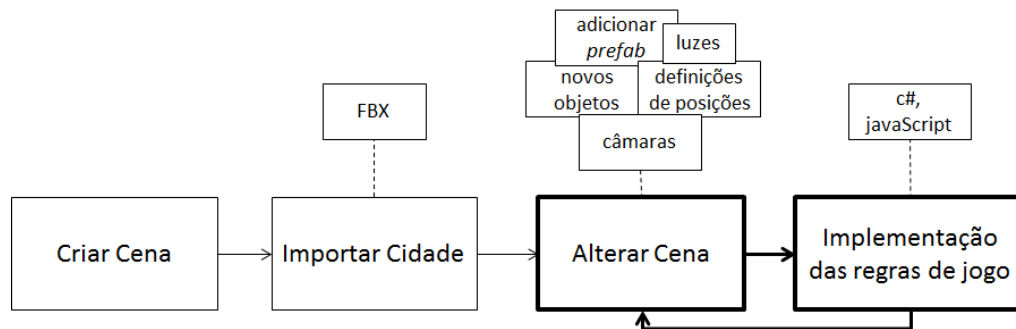


Figura 5.9: Estratégia de implementação da mecânica de Jogo - Unity3D

5.7.1 Planeamento das tarefas

Na primeira fase de cada nível, o jogador terá que planejar o conjunto de tarefas que recebe. Esta secção explica de que forma as tarefas são seleccionados de acordo com o utilizador.

5.7.1.1 Entrar na sessão

Assim que o jogador entra na aplicação e se autentica, o sistema armazena pontualmente os seus dados. Caso exista já uma sessão iniciada, o jogador pode optar por retomá-la ou iniciar uma nova. Uma sessão é suspensa se a aplicação for desligada sem terminar corretamente, ou seja, sem premir a tecla *ESC*. São consideradas sessões suspensas se o valor *end_time* da tabela *sessoes* for nulo, isto é, se for *0000-00-00 00:00*. A cada entrada numa nova sessão, é adicionada imediatamente o *id* do corrente utilizador, o nível, o patamar e é definida automaticamente a data de início da sessão, que equivale à data no momento em que o jogo é iniciado. Quando a sessão é terminada corretamente ou quando não é retomada, o valor da *end_date* toma a data do momento da operação. Estas operações permitem que haja um melhor controlo sobre qual o nível e o patamar que se segue para cada utilizador, visto o objetivo ser a progressão de níveis e não a sua repetição. Com esta gestão de sessões, garante-se que o nível será sempre o seguinte ao último executado com sucesso 5.10

5.7.1.2 Acesso às tarefas

Após a autenticação, é possível saber qual o nível e o patamar atual do jogador. Essa informação é usada para se iniciar o jogo no nível e no patamar correto. Segue-se a fase de planeamento

```

1
2 public string getTasksUrl = "http://paginas.fe.up.pt/~ei07153/city/display_tasks.
   php?";
3
4 public IEnumerator GetTasks(int block, int level)
5 {
6     string b = block + "";
7     string l = level + "";
8     Md5 md5 = new Md5();
9     string hash = md5.Md5Sum(l + secretKey);
10    string get_url = getTasksUrl + "level=" + WWW.EscapeURL(l) + "&block=" + b + "&
       hash="+hash;
11
12    WWW hs_get = new WWW(get_url);
13
14    yield return hs_get;
15
16    if (hs_get.error != null)
17        print("Erro no levantamento de tarefas: " + hs_get.error);
18    else if (!valid) {
19        string tasks = hs_get.text;
20
21        string[] tasks_split = tasks.Split('\n'); //array de tarefas
22
23        if (tasks_split.Length > 0) {
24            for (int i = 0; i < tasks_split.Length - 1; i++)
25                saveTask(tasks_split[i]);
26            valid = true;
27        }
28    }
29 }

```

Listing 5.5: Extrato de código c# - comunicação com a base de dados

(ver 4.2.3), onde o jogador recebe um conjunto de tarefas que devem ser ordenadas de acordo com o critério de avaliação do respetivo patamar. Para este efeito as tarefas são retiradas da base de dados, segundo o nível e o patamar em questão.

A comunicação entre o *Unity3D*, representado pela linguagem *c#*, e a base de dados é conseguida através de *php*, por precauções a nível da segurança de dados. Do lado do *client*, no *Unity3D*, é feita uma chamada à página *php*, alocada num servidor. O extrato de código 5.5 mostra a primeira fase do processo de comunicação. Através do *url*, a aplicação acede à página *php*, passando como argumento do *url* os dados necessários para se obter o resultado pretendido. A segurança dos dados é conseguida através do *MD5*⁸, que compara as duas *hash* (na base de dados e no lado do cliente) para assegurar a comunicação.

⁸MD5 - algoritmo de encriptação unidirecional

```

1 private void mousepicking() {
2
3     //se pressionar botao do lado esquerdo do rato
4     if (Input.GetButton ("Fire1") && mouse_picking.active) {
5         Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
6
7         //detetar objeto encontrado
8         if(Physics.Raycast(ray, out hit)) {
9             if(hit.transform.name == <inserir_caso_especial>)
10                 inserir_funcao_caso_especial;
11             else{
12                 selected_name = hit.transform.parent.parent.name;
13                 turnOnWindow("");
14             }
15         }
16     }
17 }

```

Listing 5.6: Extrato de código c# - selecionar objetos do cenário

5.7.1.3 Seleção das tarefas

A seleção das tarefas para o nível é feita de forma aleatória. No caso de um jogador precisar de repetir o mesmo nível mais do que uma vez, o importante é que adquira competências e que não memorize a solução. O sistema escolhe de forma aleatória n tarefas, de acordo com o que está definido na base de dados para o nível em questão, dentro das tarefas todas que existem no corrente nível.

5.7.2 Executar as tarefas

A segunda fase de cada nível é a execução das tarefas anteriormente planeadas. A presente secção descreve o funcionamento no sistema de como as tarefas são selecionadas e como poderão ter sucesso no nível.

5.7.2.1 Selecionar objetos

Assim que as tarefas estão ordenadas corretamente, o jogador deve procurar a solução das mesmas dentro da cidade. O objetivo das tarefas é atingir os locais adequados para as realizar, ou seja, no caso da tarefa ser “Comprar pão”, o jogador deve deslocar-se a uma padaria e selecionar o edifício correspondente. Essa ação lança um evento (extrato de código 5.6) no sistema que procura se o edifício não se trata de um caso especial (e.g, autocarro) e, caso não seja, devolve ao jogador uma janela de confirmação de seleção do objeto. A possibilidade de *Desselecionar* o objeto não traz diferenças para o jogo. O *Selecionar* valida a seleção do objeto, positiva ou negativamente. Ao selecionar o objeto, o sistema irá comparar o local do objeto selecionado com o local da tarefa em causa e dará pontuação em conformidade.

5.7.2.2 Sucesso

A cada sucesso numa tarefa, o jogo incrementa um ponto à pontuação adquirida até aquele momento. O contrário já não acontece, no caso de o jogador falhar uma tarefa, não lhe será retirado nenhum ponto. Porém, a tentativa falhada o sistema obrigará a resolver a tarefa seguinte sem oportunidade de repetir a anterior.

O sucesso do nível não é estático. O valor atribuído como pontuação mínima para a progressão no nível resulta do produto da percentagem do sucesso esperado com o número total de tarefas do nível. O sucesso esperado é um valor pré-definido na base de dados e está atribuído um sucesso para cada nível de cada patamar. Caso o valor de sucesso seja atingido, o jogador evoluirá para o nível seguinte, caso contrário será obrigado a repetir o mesmo exercício. No entanto, é de salientar que se o nível atual for o último do seu patamar, o jogador avançará de patamar, para um novo critério de avaliação.

No sentido de premiar os pacientes com melhores prestações, foi criada uma lógica de jogo que admite que um jogador com vários sucessos consecutivos, poderá avançar imediatamente para o patamar seguinte. Em situações em que o paciente avança com sucesso três níveis consecutivos do mesmo patamar sem interrupções, admite-se que o paciente está preparado para evoluir para um novo patamar (tal como esquematizado na figura 5.10).

Se os níveis em jogo tiverem como critério de avaliação “Tempo despendido” ou “Caminho mais curto”, para além da fórmula anterior, o jogador será também avaliado numa destas componentes. Para os níveis cujo o critério de avaliação é o “Tempo despendido”, o tempo fornecido é proporcional ao número de tarefas, dando cerca de um minuto por tarefa para efeitos de teste. Após esse tempo estipulado ter sido ultrapassado, o jogador receberá uma mensagem de erro e terá que repetir o atual nível. Para o nível de “Caminho mais curto”, o sistema usará a formula descrita em 5.1. A incógnita *result* é o resultado final da operação, que corresponde ao valor da distância percorrida pelo jogador durante todo o jogo. *NumPathObject* representa o número de objetos criados durante o percurso e que devem ser analisado para o cálculo final. Os valores *diffX* e *diffY* equivalem, respetivamente em x e y, à diferença entre a posição do objeto em análise com o objeto anteriormente analisado.

$$result = \sum_{i=0}^{NumPathObject} \sqrt{diffX^2 + diffY^2} \quad (5.1)$$

5.7.3 Sistema de autocarros

As seguintes secções expõe a mecânica dos autocarros, desde a forma como são atribuídos os destinos e as estações, à forma como eles se deslocam no cenário.

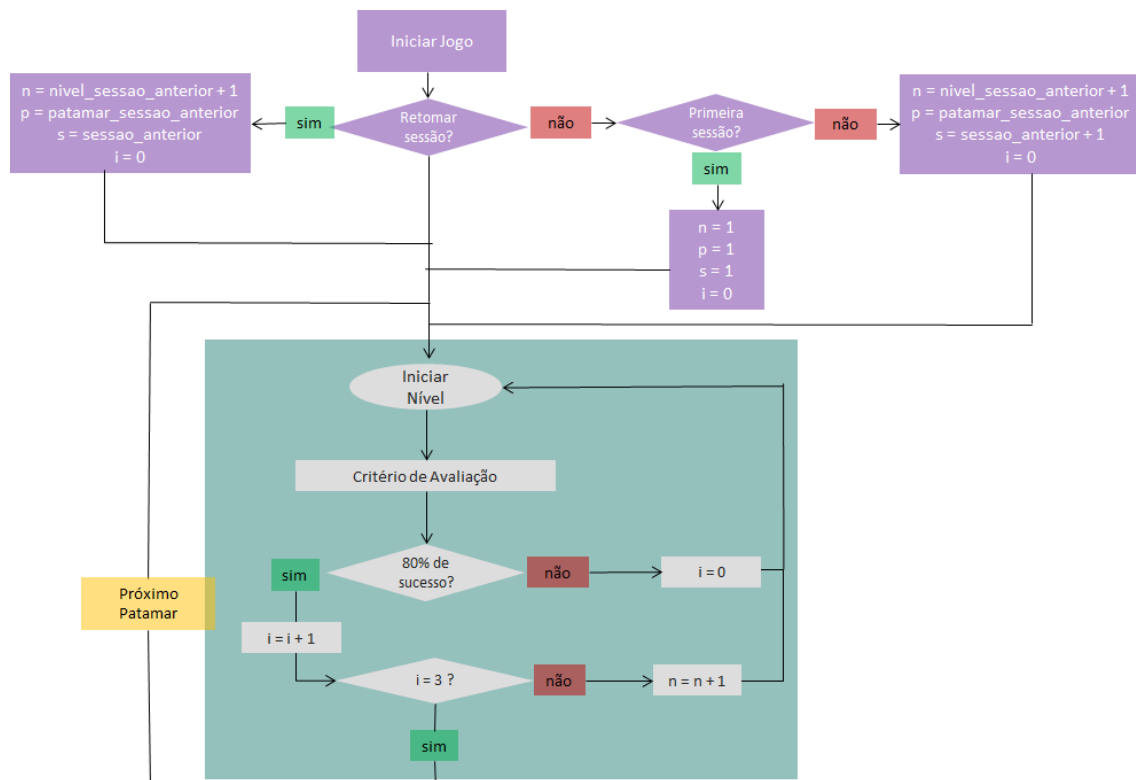


Figura 5.10: Evolução entre níveis e patamares

5.7.3.1 Atribuição de autocarros e paragens

Existe na base de dados (Figura A.2) do sistema um conjunto de destinos de autocarros, com os respetivos pontos de paragem. Extrair essa informação da base de dados, o sistema alocará aleatoriamente a combinação (número do autocarro, destino) a todos os autocarros do cenário. Esse processo é feito de forma automática, devido ao uso da função dinâmica 5.7. As paragens por onde passam cada autocarro são atribuída de acordo com os dados da base de dados.

Cada estação de autocarros tem associado um nome, que representa o local da paragem, e nomes de autocarros, que corresponde aos autocarros que têm como paragem a estação em causa. O nome da paragem é atribuída, tal como o destino dos autocarros, através de geração aleatória, dentro da lista de locais possíveis. Por outro lado, a listagem de autocarros que passam na paragem corresponde ao conjunto de autocarros que têm como paragem ou como destino o nome da paragem.

A cada novo jogo, este processo será repetido, no entanto a relação entre o número do autocarro

```
1 GameObject.FindGameObjectsWithTag("Bus");
```

Listing 5.7: Encontrar todos os autocarros do cenário

e o destino, bem como as paragens associadas a esse autocarro são sempre preservadas, através do uso da base de dados.

5.7.3.2 Deslocação dos autocarros

Um autocarro e uma paragem constituem um *prefab*, o que significa que sempre que uma estação é inserida no cenário, existirá também um autocarro associado. O autocarro aparecerá em cena esporadicamente e inicia a sua rota num ponto distante da estação, de forma a transmitir a sensação de movimento contínuo.

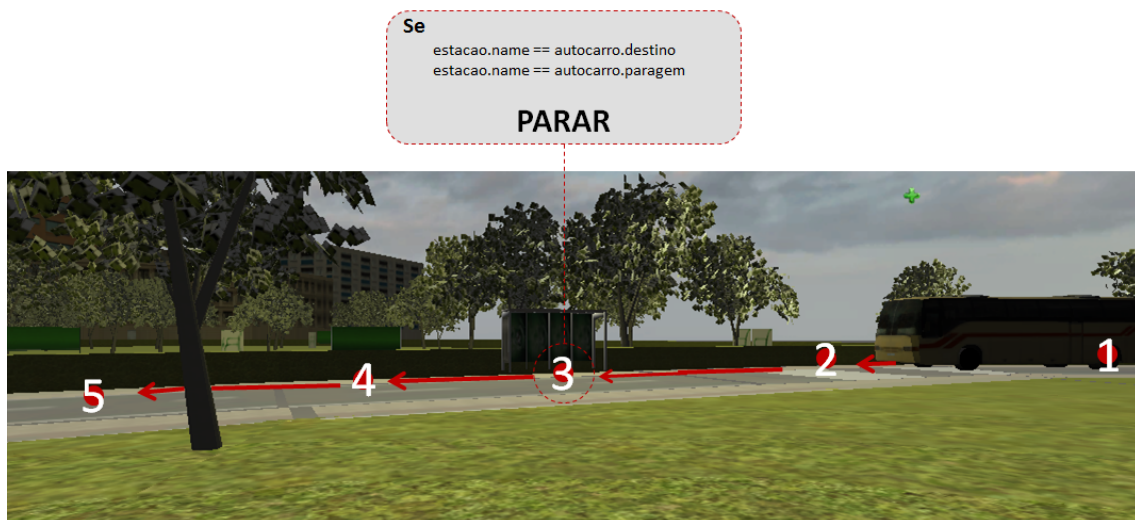


Figura 5.11: Deslocação do autocarro no cenário

A partir do momento em que o jogador entra na fase de execução das tarefas, o jogo inicia uma contagem do tempo decorrido. A cada vinte segundos, os autocarros iniciam uma nova marcha, passando forçosamente pelos pontos assinalados na figura 5.11. O autocarro inicia a sua rota no ponto 1 e termina no 5. Nas extremidades do caminho, o autocarro desaparece ou aparece no cenário, dependendo se está a terminar ou iniciar a marcha. No ponto 3, que corresponde ao ponto médio do percurso, o sistema analisa se o autocarro deve ou não para na estação. Um autocarro deve parar na estação sempre que o seu destino corresponda ao nome da estação ou sempre que uma das paragens previstas seja a estação.

5.7.3.3 Entrar no autocarro

O autocarro só pára na estação, no ponto 3 da figura 5.11, caso a estação em causa esteja incluída na lista de possíveis paragens ou destinos do autocarro em causa. Nessas situações, o jogador tem a opção de seleccionar o autocarro, entrar e iniciar viagem. Neste momento, a câmara principal é alterada para uma câmara interna do autocarro. Durante a viagem, o jogador recebe notificação sobre a próxima paragem, podendo pressionar o botão de *STOP*, se pretender sair na paragem seguinte do autocarro onde se encontra.

5.7.4 Decisões de implementação orientadas à reabilitação

Durante a fase de desenvolvimento, houve cuidado em adaptar o jogo às capacidades do público-alvo e em orientar a componente gráfica ao objetivo final do jogo.

No início de todos os níveis o sistema apresenta um quadro informativo com o objetivo principal do nível e com o critério de ordenação das tarefas. Adicionalmente, nos primeiros níveis de cada patamar (50% dos níveis), esse mesmo critério de ordenação é relembrado durante a fase de planeamento das mesmas. Isto deve-se ao facto de não ser relevante até ao momento exigir que o jogador memorize, focando a aprendizagem no saber ordenar e no saber distinguir o que é ou não importante.

Ainda no sentido de orientar o jogador, o sistema disponibiliza um mapa bidimensional com vista de topo, onde tem sinalizada a posição atual do jogador, a posição do objetivo final e ainda todo o caminho percorrido até ao momento pelo jogador. Porém, o mapa só está completamente disponível nos primeiros 20% dos níveis num patamar. Nos níveis seguintes, a consulta do mapa tem custos associados à pontuação do jogador, exigindo um melhor desempenho da sua parte na realização das tarefas.

Durante o jogo, existirá um cronómetro, que contabiliza o tempo gasto pelo jogador no nível. Para o nível cujo o critério de avaliação é ter um tempo-limite para executar todas as tarefas, o cronómetro contabiliza o tempo e quando estiver próximo do fim, o cronómetro alterará a cor das letras, para realçar esse facto e apressar o jogador. Durante a fase de execução, a próxima tarefa a ser realizada é realçada (através da cor amarela e de uma aumento do tamanho de letra), atraindo a atenção do jogador para o que deve ser cumprido nesse determinado momento.

A pontuação do jogador deve ser encarada não só como uma forma de avaliar o seu desempenho, mas também um meio de estimular e motivar o jogador. As mensagens de erro ou sucesso, dadas após cada tarefa ou nível, são a forma de desafiar o jogador na procura de melhorar o seu desempenho no nível seguinte. Os níveis de um patamar contam, sempre que se aplique, com um novo critério de avaliação adicionado ao critério do patamar anterior. O sistema pretende que o jogador pratique uma aprendizagem contínua mas sem descuidar as aprendizagens anteriores que continuam a ser treinadas.

No sentido de adaptar os conhecimentos e o grau de aprendizagem do utilizador, *Cidade Virtual* tem uma estratégia de progressão entre patamares. Quando o jogador consegue avançar com sucesso três níveis consecutivos, que pertençam ao mesmo patamar, ele avançará para o patamar seguinte, mesmo que não tenham completo todos os níveis desse patamar.

O cenário de jogo foi modelado de acordo com a realidade portuguesa, mais especificamente tendo em conta o mapa de duas cidades do Grande Porto, aproximando assim o jogo à vida real do paciente. Para viajar dentro da cidade, o sistema está preparado apenas para rotações de 360º horizontais. Isto deve-se ao facto dos utilizadores terem, em experiências anteriores deste projeto, demonstrado dificuldades em navegar num jogo que permitisse rodar a câmara no sentido vertical.

Relativamente à mecânica dos autocarros, a relação número de autocarro e destino é preservada. No entanto o nome do local de cada paragem, vai-se alterando. Este facto deve-se à ne-

cessidade de obrigar o paciente a entender o funcionamento habitual dos autocarros, sem decorar o procedimento. Dese saber consultar o quadro com a listagem de autocarros que passam nessa paragem .

5.8 Proposta de avaliação da eficácia do programa

No sentido de validar este projeto, são propostos dois tipos de avaliação: neuropsicológicas e usabilidade.

Dada a variedade de testes na área da neuropsicologia e a inexperiência da equipa de desenvolvimento, foram propostos alguns dos considerados mais adequados, propostos pela psicóloga inserida na equipa do projeto-mãe, *CARP-VR*. As propostas foram várias, tais como: *Hospital Anxiety and Depression Scale* (HADS), *Mini-mental State Examination* (MMSE), *Behavioural Assessment of the Dysexecutive Syndrome* (BADS), entre outras. Após a análise das sugestões, destaca-se para este projeto o teste BADS. Os princípios de avaliação destes testes vão de acordo com os que se pretendem avaliar no final deste projeto. A avaliação consiste num conjunto de seis testes e dois questionários. Os testes pretendem avaliar a capacidade de planeamento, iniciação, monitorização e capacidade de ajustar o comportamento às tarefas[WAB⁺03].

Por outro lado, seria interessante avaliar, do ponto de vista técnico, de que forma o jogador vai evoluindo entre os níveis. Uma vez que o jogador tem que repetir os níveis que não são terminados com sucesso, é relevante para o estudo, perceber se o jogador, com o avançar dos níveis, tende a atingir o sucesso mais rápido do que nos níveis anteriores do mesmo patamar.

Para os testes de usabilidade recomenda-se que se faça em média 10 exames a utilizadores finais [Gra98]. Nos testes, o paciente deve:

- ser capaz de trabalhar sem auxílio dos terapeutas;
- conseguir entrar no jogo com a sua conta pessoal;
- saber entrar no jogo, iniciar nova sessão e reiniciar;
- entender a funcionalidade de todos os botões do jogo;
- perceber o funcionamento do cenário.

Os testes de usabilidade apesar de não serem relevantes do ponto de vista da aprendizagem direta ou do desenvolvimento de competências, são essenciais para perceber se da perspectiva do utilizador o *design* da aplicação é adequado e o programa é de fácil utilização.

Capítulo 6

Conclusão

Sendo este projeto ambicioso, o resultado final desta dissertação não pretendia abranger funcionalmente todos os requisitos especificados para o produto final. Contudo, as primeiras fases de desenvolvimento, levantamento de requisitos e desenho da arquitetura, foram-no detalhadamente, considerando a totalidade das funcionalidades do produto final. O mesmo se aplica à fase de implementação e desenho da base de dados. Existiu, durante o desenvolvimento, um especial cuidado para deixar uma primeira fase do *Cidade Virtual* com uma base sólida, de modo a facilitar trabalhos futuros, independentemente da equipa que retome o projeto.

A primeira fase deste projeto consistiu na análise do problema da dissertação, fazendo um estudo sobre o estado da arte. Após esse estudo, foi possível identificar soluções para a problemática da reabilitação neurocongitiva com recurso às novas tecnologias, passando pela oferta de ferramentas e perspectivas de implementação. Na sequência da análise das soluções, foi desenhada a arquitetura do jogo e finalmente a implementação do protótipo.

De acordo com o problema da presente dissertação e após o estudo de diferentes soluções, propõe-se que a aplicação final esteja disponível em versão *online*, permitindo que seja acessível de qualquer local, desde que haja acesso à *Internet*. A desvantagem de optar pela versão *online* é o facto de não existir uma forma direta de ligar o atual projeto aos anteriormente desenvolvidos para o mesmo projeto-mãe. Em contrapartida, as vantagens de uma plataforma *online* compensam esta limitação, tais como a monitorização a longas distâncias.

Do ponto de vista das ferramentas a usar, e após a análise de algumas soluções, resultou como *software* de modelação tridimensional o *CityEngine* e como motor de jogo o *Unity3D*. Usando o *CityEngine* foi modelada a cidade de Gaia e do Porto. Após a experiência, conclui-se que o *CityEngine* traz muitas vantagens do ponto de vista da modelação procedimental, pela facilidade de geração automática de modelos para uma cidade de grande dimensão. No entanto, o *CityEngine* exige bases técnicas, na área da programação e dificulta o trabalho quando se trata de modelação detalhada de objetos. Neste projeto, existia a necessidade de modelar uma cidade de dimensão média de forma rápida, por outro lado, havia interesse em que alguns dos edifícios fossem modelados ao pormenor, por se tratarem de elementos relevantes da cidade e que poderiam trazer mais valias ao projeto. Contrabalançando os prós e contras da ferramenta, considera-se que a solução foi ade-

quada aos objetivos pretendidos para esta fase do projeto. Mais tarde, a exportação dos dados do *CityEngine* e a importação para o *Unity3D* revelou-se um processo complexo. Apesar de ambas as aplicações suportarem ficheiros FBX, o *Unity3D* mostrou ter algumas dificuldades em interagir com ficheiros de grande dimensão. Apesar disso, o *Unity3D* revelou ser um motor de jogo de fácil utilização e com capacidade para suportar a grande maioria das funcionalidades necessárias para o desenvolvimento de um jogo. A mecânica de jogo foi maioritariamente desenvolvida em *c#*.

Durante a primeira fase e também ao longo da dissertação, um dos grandes desafios esteve na articulação entre a área de informática e da neuropsicologia. Foi interessante conciliar diferentes conhecimentos e soluções para um objetivo final comum. Outro desafio foi desenvolver um projeto cujo produto final foi completamente centrado no utilizador final, sem que houvesse contato direto regular com mesmos utilizadores.

6.1 Trabalho Futuro

O trabalho futuro é dividido em curto e médio prazo, incluindo tarefas já definidas nos requisitos do projeto e novas sugestões.

Numa perspetiva a curto prazo, seria interessante e relevante terminar a gestão de utilizadores e providenciar graficamente informação da performance dos pacientes. Para além disso, seria importante realizar uma avaliação neuropsicológica e de usabilidade da aplicação, de acordo com o apresentado nesta dissertação. Os testes devem ser realizados nesta fase intermédia do projeto para que sejam detetadas eventuais limitações e se oriente o trabalho futuro de acordo com a experiência obtida nessas avaliações.

Para uma implementação a médio prazo as sugestões menos complexas são:

- integrar o jogo aqui apresentado com as aplicações já desenvolvidas (ex, supermercado);
- inserir sons e avatares para simular uma cidade real;
- viabilizar a possibilidade de jogar em TPS;
- guardar o caminho realizado pelo jogador em cada nível na base de dados;

De um ponto de vista mais ambicioso, traria grandes vantagens e inovação ao projeto se as posições do jogador correspondessem a posições reais no mapa em tempo real. Esta informação seria fornecida pelo *OpenStreetMap* ou *GoogleMaps* sob o conceito de *geocoding*. Com o *geocoding*, a avaliação do caminho mais curto tomava uma proporção mais real e consequentemente uma avaliação do paciente mais acertada. Para além disso, deste modo, conseguir-se-ia facilmente um jogo adaptável a qualquer mapa exportado pelo *OpenStreetMap*. A fácil e automática adaptação do jogo a novos mapas seria um outro ponto a melhorar. Apesar deste conceito ter sido tido em consideração, o conceito falha ainda na atribuição de estações do parâmetros e avaliação de caminho mais curto.

Conclusão

Em suma, esta dissertação resultou num protótipo funcional, do que virá a ser a *Virtual City*, um jogo sério ambicioso e com objetivos bem delineados.

Conclusão

Anexo A

Diagrama de arquitetura de Base de Dados

Diagrama de arquitetura da base de dados dividido em pacotes. Cada pacote pretende agrupar tabelas da mesma área lógica.

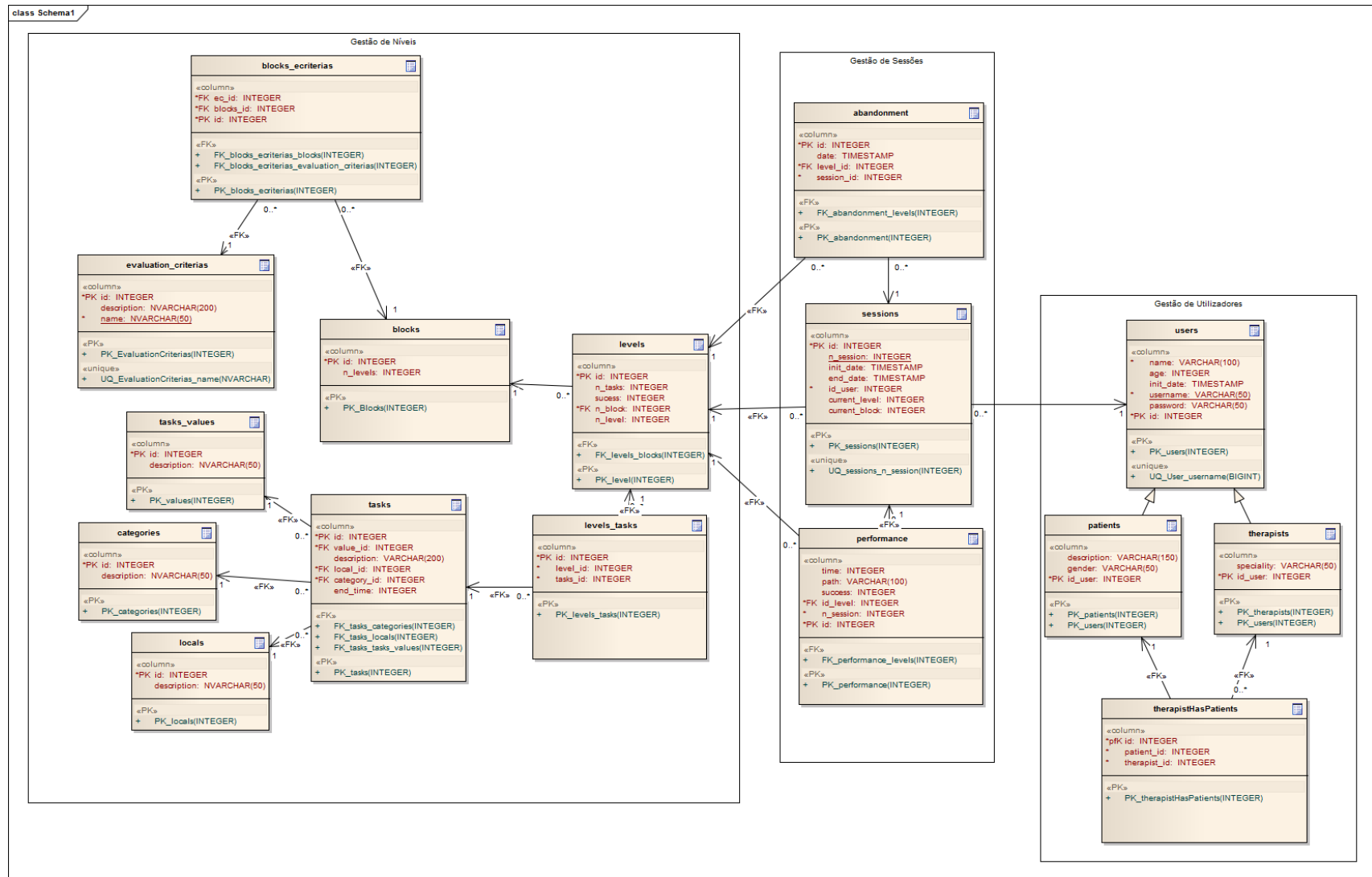


Figura A.1: Arquitetura da Base de Dados

Diagrama de arquitetura de Base de Dados

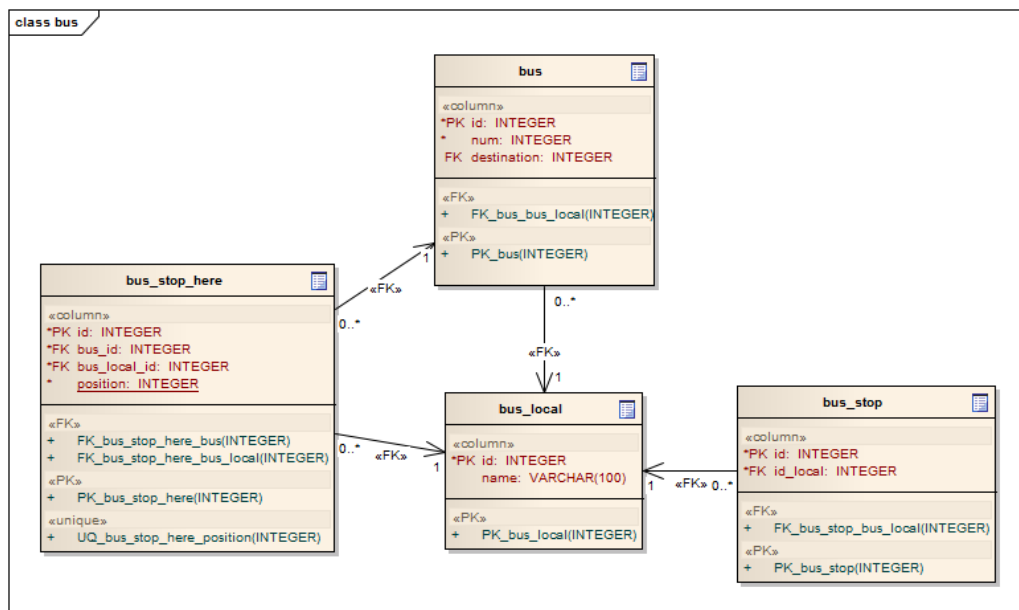


Figura A.2: Arquitetura da Base de Dados - Autocarros

Diagrama de arquitetura de Base de Dados

Anexo B

Diagrama de classes

Diagrama de arquitetura do sistema, incluindo todas as classes e respectivos atributos. Este sistema defini o funcionamento interno do mecânica de jogo

Referências

- [3DCa] 3DConnexion. Customer story Celebrity Custom Homes builds Denver dream homes smarter and faster with Autodesk Revit and 3Dconnexion 3D mice. pages 2–3.
- [3DCb] 3DConnexion. Take it to the max. pages 3–4.
- [3DC12] 3DConnexion. What is a 3D Mouse? <http://www.3dconnexion.com/products/what-is-a-3d-mouse.html>, 2012.
- [Axs10] Axsotic. AXSOTIC. <http://www.axsotic.com/index.html>, 2010.
- [BBK⁺97] Ravin Balakrishnan, Thomas Baudel, Gordon Kurtenbach, George Fitzmaurice e Richmond Street East. The Rockin ’ Mouse : Integral 3D Manipulation on a Plane Conference on Human Factors in Com-. *Human Factors*, pages 311–318, 1997.
- [BEW⁺98] Lars Bishop, Dave Eberly, Turner Whitted, Mark Finch e Michael Shantz. Designing a PC. *Ieee Computer Graphics And Applications*, (February):46–53, 1998.
- [BKB98] David J Brown, Steveno J Kerr e Victor Bayon. The development of the Virtual City : A user centred approach. *Virtual Reality*, pages 11–15, 1998.
- [BNC99] DJ Brown, Helen Neale e SVG Cobb. Development and evaluation of the virtual city. *Journal of Virtual*, pages 27–38, 1999.
- [Cai57] Roger Caillois. *Les jeux et les hommes*. 1957.
- [CBM07] Jeff Craighead, Jennifer Burke e Robin Murphy. Using the unity game engine to develop sarge: a case study. *COMPUTER*, 4552:366, 2007.
- [Coo09] Daniel Cook. What are game mechanics? http://www.lostgarden.com/2006_10_01_archive.html, 2009.
- [Cor06] K. Corti. Games-based Learning; a serious business application. *Informe de Pixel-Learning*, pages 1–20, 2006.
- [Cor12] NVIDIA Corporation. GeForce. <http://www.geforce.com/Hardware/Technologies/physx>, 2012.
- [Cos94] Greg Costikyan. I Have No Words & I Must Design. 1994.
- [CS03] A Coelho e AA Sousa. 3D Modelling of large urban scenes from diverse sources of information. *Proceedings of the 7th ICCG*, pages 279–288, 2003.
- [DCB⁺11] A.R. Does, I.P. Carvalho, F. Barbosa, I. Almeida, S. Guerreiro, M. Leitão, L. Sousa e A. Castro-Caldas. Serious games: Are they part of the solution in the domain of cognitive rehabilitation. pages 95–105, 2011.

REFERÊNCIAS

- [Der07] Anne Derryberry. Serious games: online games for learning. *Adobe Whitepaper*, November, 2007.
- [Dev11a] Parvati Dev. Healthcare Training with Simulation. <http://www.slideshare.net/fullscreen/parvatidev/healthcare-training-with-simulation/1>, 2011.
- [Dev11b] Devmaster. DevMaster. devmaster.net/devb/engines, 2011.
- [DMC⁺] A.R. Does, M.J. Miranda, A.F. Coelho, F. Barbosa, A.C. Caldas, I.P. Carvalho, L. Sousa e L. Mendes. Virtual City - Neurocognitive Rehabilitation of Acquired Brain Injury. pages 1–4.
- [Ese] Esenthel. Esenthel. <http://www.esenthel.com/?id=compare>.
- [ESR12] ESRI. Esri - City Engine. <http://www.esri.com/software/cityengine/index.html>, 2012.
- [FK04] Dieter Fritsch e Martin Kada. Visualisation using game engines. *Archiwum ISPRS*, 2004.
- [Gar12] GarageGames. Torque 3D. <http://www.garagegames.com/products/torque-3d>, 2012.
- [Gol04] J Gold. Object-oriented game development. 2004.
- [Gra98] Trevor Grayling. Fear and Loathing of the Help Menu : A Usability Test of Online Help. (November 1997), 1998.
- [GSRC09] Philip Greenwood, Jesse Sago, Sam Richmond e Vivi Chau. Using game engine technology to create real-time interactive environments to assist in planning and visual assessment for infrastructure. *Proceedings MODSIM. Cairns*, (July):2229–2235, 2009.
- [Has] Hasomed. Hasomed RehaCom. <http://www.hasomed.de/en/products/rehacom-cognitive-therapy/faq.html>.
- [KCS11] Neil Katz, Tom Cook e Robert Smart. Extending Web Browsers with a Unity 3D-Based Virtual Worlds Viewer. *IEEE Internet Computing*, 15(5):15–21, September 2011.
- [Kev01] Maroney Kevin. My Entire Life. *The Games Journal*, 2001.
- [LK11] M. Labschütz e K. Krösl. Content Creation for a 3D Game with Maya and Unity 3D. *cg.tuwien.ac.at*, 2011.
- [Ltd] NeoAxis Group Ltd. NeoAxis - EMSAVE. http://www.neoaxis.com/neoaxis/clients_and_titles/non_game_projects/emsave.
- [LWR⁺04] T. Lechner, B. Watson, P. Ren, U. Wilensky, S. Tisue e M. Felsen. Procedural modeling of land use in cities. *Cities*, 2004.
- [Mac01] Michael Macedonia. Games, simulation, and the military education dilemma. *Internet and the University*, pages 157–167, 2001.

REFERÊNCIAS

- [MM11] Paul Merrell e Dinesh Manocha. Model synthesis: A general procedural modeling algorithm. *Visualization and Computer Graphics, IEEE Transactions on*, (99):1–1, August 2011.
- [Ope12] OpenStreetMap. Wiki - OSM XML. http://wiki.openstreetmap.org/wiki/OSM_XML, 2012.
- [PSB⁺11] G C Peron, L I B Santos, L M Brasil, R C Silva, F Bombonato, P S De Lima, Universidade De Brasília Fga, Terapia Ocupacional, Universidade De Brasília Fce, Pós Graduação, Desenvolvimento De Jogos, Tecnólogo Programação e Senai Brasília. Serious Games na Reabilitação Cognitiva Serious Games in Cognitive Rehabilitation. *Pan American Health*, pages 94–95, 2011.
- [Reh] RehaCom. Literature RehaCom. <http://rehacom.co.uk>.
- [RMR] Paula Rego, Pedro Miguel Moreira e Luis Paulo Reis. Serious Games for Rehabilitation A Survey and a Classification Towards a Taxonomy.
- [SJB07] Tarja Susi, Mikael Johannesson e Per Backlund. Serious games—An overview. *Skövde: University of Skövde (Technical Report HS-IKI-TR-07-001)*, 2007.
- [Sol09] Indusgeeks Solutions. Indusgeeks. <http://indusgeeks.com/services-serious-games.html>, 2009.
- [Son] Sony. SonyInsider. <http://www.sonyinsider.com/author/admin/page/34/>.
- [Squ03] K.D. Squire. Video games in education. *Int. J. Intell. Games & Simulation*, 2(1):49–62, 2003.
- [Sys] Dassault Systemes. 3DVia Virtools. <http://www.3ds.com/products/3dvia/3dvia-virtools/>.
- [SZ03] Katie Salen e Eric Zimmerman. *Rules of Play: Games Design Fundamentals*. 2003.
- [Tec12] Unity Technologies. Unity 3D. <http://unity3d.com/>, 2012.
- [WAB⁺03] Barbara A Wilson, Nick Alderman, Paul W Burgess, Hazel Emslie e Jonathan J Evans. Behavioural Assessment of the Dysexecutive Syndrome (BADS). 5(2), 2003.
- [WMW08] Benjamin Watson, P Muller e P Wonka. Procedural urban modeling in practice. *Computer Graphics*, (June):18–26, 2008.
- [WMZ⁺10] Sa Wang, Zhengli Mao, Changhai Zeng, Huili Gong, Shanshan Li e Beibei Chen. A new method of virtual reality based on Unity3D. In *Geoinformatics, 2010 18th International Conference on*, pages 1–5. IEEE, 2010.
- [Yee06] Nick Yee. Motivations for play in online games. *Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society*, 9(6):772–5, December 2006.
- [Zha98] Shumin Zhai. User performance in relation to 3D input device design. *ACM Siggraph Computer Graphics*, 32(4):50–54, November 1998.