FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Jogo Digital para o Ensino dos Fundamentos da Programação

Ricardo Emanuel Ferreira Gonçalves

Mestrado Integrado em Engenharia Informática e Computação

Orientador: António Fernando Coelho (PhD)

17 de Junho de 2011

Jogo Digital para o Ensino dos Fundamentos da Programação

Ricardo Emanuel Ferreira Gonçalves

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo júri:

Presidente: Jorge Alves Silva (PhD)

Vogal Externo: João Paulo Moura (PhD)

Orientador: António Fernando Coelho (PhD)

---

11 de Julho de 2011

Resumo

Os jogos digitais sérios têm um grande potencial imersivo que pode ser usado para motivar e ajudar estudantes na sua aprendizagem. No ensino dos fundamentos da programação, onde se requer uma grande dedicação de estudo, esta tecnologia poderá se tornar numa ferramenta de grande utilidade. Nesse sentido, para usar e usufruir de todos os benefícios que os jogos sérios trazem, esta dissertação visa o desenvolvimento de uma plataforma de criação de jogos sérios para o suporte ao ensino da programação.

De forma a conseguir desenvolver esta plataforma com sucesso, vários passos foram dados. O primeiro passo foi analisar o estado da arte actual, estudando os vários tipos de jogos sérios e focando alguns jogos relacionados com o ensino da programação, de forma a haver uma melhor compreensão e preparação para o planeamento e desenvolvimento da plataforma.

Com os novos conhecimentos obtidos no passo anterior foi definida a mecânica de jogo que irá reger a plataforma bem como o plano de como a implementar. Durante este passo foi decidido usar o motor de jogo Unity3D para criar o jogo e o DOMJudge para avaliar as soluções dos exercícios de programação submetidas.

Seguidamente, a plataforma foi implementada, seguindo as linhas definidas no passo anterior como guia. Sobre a plataforma foi gerado um protótipo que posteriormente foi alvo de testes por parte de vários alunos. Após os testes, um inquérito foi preenchido com o objectivo principal de procurar aspectos negativos a serem melhorados.

Os resultados dos testes e o feedback dos participantes foi bastante positivo e animador, no entanto há aspectos que ainda precisam de ser melhorados.

Este sistema apesar de já estar funcional e de ser capaz de suportar o ensino de qualquer linguagem de programação, ficou sem implementar uma parte importante - a página de controlo e interacção entre docente e alunos. Além da parte que faltou implementar, há ainda mais trabalho pela frente até que este sistema possa atingir a maturidade suficiente para ser usada activamente.

Abstract

Digital serious games have great immersive potential that can be used to motivate and help students in their learning. In the teaching of the fundamentals of programming, this technology could become a valuable tool, for it requires a great dedication to study. In this sense, to use and enjoy all the benefits serious games, this thesis aims to develop a platform for creating serious games to support the teaching of programming.

In order to develop this platform successfully, several steps were taken. The first step was to analyze the current state of the art, studying the various types of serious games and focusing on some games related to the teaching of programming. This allowed a better understanding and preparation for the planning and development of the platform

With the new knowledge gained in the previous step the game mechanics that will govern the platform were defined as well as the plan on how to implement it. During this step it was decided to use the game engine Unity3D to create the game DOMJudge and to evaluate the submitted programming exercises solutions. Thereafter, the platform was implemented, following the guidelines defined in the previous step. On top of the platform a prototype was developed that was later the subject of testing by several students. After the tests, a survey was filled with the main objective of seeking the aspects to be improved.

The test results and feedback from participants was very positive and encouraging, however there are aspects that still need to be improved.

Although this system is functional and is able to support the teaching of any programming language, an important part wasn't implemented - the page control and interaction between teacher and students. Besides the part that was not implemented, there is still more work ahead to be done so that this system can achieve the maturity it needs to be used actively.

Agradecimentos

Ao longo da nossa vida vamos nos deparando com inúmeros desafios. Muitos desses desafios, devido à sua dificuldade necessitam da ajuda, da cooperação, do apoio dos outros seres que coexistem connosco. Esta dissertação é um exemplo de tal desafio.

Agradecer é um acto que para mim tem que ser um exercício diário, estar tão consciente quanto possível da nossa realidade e da contribuição para esta pelo todo e todos.

Nesse espírito gostaria de começar por agradecer ao meu orientador Professor Doutor António Fernando Coelho pela sua ajuda, a sua disponibilidade, e a liberdade que me proporcionou no desenvolvimento desta dissertação.

Quero também agradecer ao Enrique Kato pela sua paciência e cooperação a criar a mecânica do jogo, nos vários brainstormings que fizemos ao longo do desenvolvimento e da ajuda com os modelos 3D e texturas. Estou também grato ao João Xavier pela sua longa paciência em explicar-me alguns aspectos que não entendi do DOMJudge, bem como permitir-me o uso dos seus servidores.

Ao André Cunha, agradeço o apoio constante e a ajuda nos testes e opiniões que foi dando à medida que desenvolvimento foi progredindo.

Aos meus velhos amigos Mondlane e Isidro, agradeço a vossa amizade e continuado apoio

Um agradecimento especial e eterno vai para os meus pais e para a irmã, obrigado por me terem e continuarem a acompanhar e a apoiar, sem a vossa ajuda sei que não estaria onde estou, nem seria quem sou.

Finalmente, quero agradecer à Irina a amizade e apoio incondicional, tens estado sempre do meu lado apesar das minhas falhas e da distância, não poderia desejar uma melhor amiga.

A todos um profundo e sentido: Obrigado!

Ricardo Gonçalves

Conteúdo

Lista de Figuras

LISTA DE FIGURAS

Lista de Tabelas

Abreviaturas e Símbolos

API	Interface de Programação de Aplicações
CAD	Desenho Assistido por Computador
ER	Entity Relationship
JVM	Java Virtual Machine
TS	Torque Script
UC	Unreal Script
UDK	Unreal Development Kit
UML	Unified Model Language
WAN	Wide Area Network

Capítulo 1

Introdução

O grau de concentração que conseguimos ter na execução de uma actividade depende de vários factores. Um desses factores é a nossa percepção, gerada através das várias experiências ao longo da nossa existência, sobre o quê, o porquê e o para quê. Essa percepção influencia o grau de envolvimento que nos permitimos ter em tudo aquilo que fazemos. Portanto, ao fazermos algo de que gostamos, há uma maior probabilidade de nos deixarmos envolver pela actividade o que implica ao nível psico-fisiológico uma tendência para o nosso estado mental se alterar para aquilo que em psicologia e hipnose se denomina de estado Beta. Este estado reflecte a frequência das ondas cerebrais produzidas pela actividade eléctrica no cérebro humano e que é associado a um estado de alerta e alta concentração, ideal para tarefas que exigem toda a atenção possível.

Por outro lado, quando identificamos uma actividade/tarefa como aborrecida, existirá uma maior dificuldade em nos conseguirmos manter motivados e concentrados, podendo em muitos casos implicar a divisão da atenção entre aquilo que se está a fazer e aquilo que se gostaria de estar a fazer e possivelmente gerar-se alguma frustração, comprometendo a capacidade de raciocínio e aprendizagem.

Aprender é uma tarefa crucial no nosso desenvolvimento pessoal e social, que no entanto em muitos casos é percebida como algo difícil e aborrecido, repelindo a mente. No entanto, essa percepção pode ser alterada para algo divertido de se fazer, modificando a tarefa de forma a que esta se transforme numa experiência lúdica, sem no entanto perder o aspecto sério. Adicionalmente, se o aluno se interessar verdadeiramente pelo jogo, há uma grande probabilidade deste procurar informação extra de fontes externas, como na internet, de forma a melhorar o seu entendimento do jogo e conseguir obter melhores resultados.

Nesta dissertação pretende-se tornar a aprendizagem dos fundamentos da programação numa actividade divertida, desenvolvendo nesse sentido uma plataforma para o

suporte à criação de jogos sérios de programação.

1.1 Descrição do Problema

A aprendizagem dos fundamentos da programação é algo que requer esforço e uma grande dedicação do estudante. É através da resolução de exercício após exercício, que alguém conseguirá atingir o nível de excelência necessário para produzir software de qualidade e ser capaz de ultrapassar os desafios que a vida tem reservado a este nível.

Todavia, este esforço e dedicação é passível de se tornar fastidioso [AJ] [Ton], principalmente se os exercícios forem desligados entre si e sem um objectivo imediato e significativo, correndo o risco de se tornar numa tarefa cansativa e não ser alvo de toda a atenção que lhe é devida.

O desafio desta dissertação é construir uma plataforma informática que proporcione aos estudantes motivação e imersão a fim de que estes sejam capazes de aprender de uma forma descontraída e simples. Ao mesmo tempo, deverá também permitir ao docente acompanhar o progresso da aprendizagem dos seus estudantes e definir a sequência de desafios que integram o plano de estudos, para melhor conseguir inferir as dificuldades experienciadas por estes e definir qual a melhor via para os ajudar a ultrapassá-las.

1.2 Objectivos

O objectivo principal deste projecto é conceber uma plataforma para o suporte à definição de jogos sérios cuja mecânica de jogo se baseie na resolução de exercícios de programação de diferentes níveis de complexidade. A plataforma deverá também possibilitar ao docente ter *feedback* sobre a performance dos estudantes e controlar a disponibilidade dos níveis do jogo.

No final desta dissertação a plataforma deverá ser constituida por:

- Um servidor central capaz de gerar e gerir todos os aspectos de um jogo;
- Disponiblizar uma interface que permita o docente interagir com os alunos e verificar o seu progresso;
- Disponibilizar um jogo que receba informações do servidor central e seja capaz de gerar um jogo baseado nessas informações.
- Ser capaz de disponibilizar exercícios de programação e avaliar as respectivas soluções propostas.
- Suportar o máximo de linguagens de programação possíveis.
- Disponibilizar um prototipo de um jogo gerado pela plataforma.

1.3 Resultados Esperados

No final desta dissertação pretende-se obter uma plataforma que permita gerar jogos sérios orientados para o ensino da programação, ferramentas Web que permitam ao docente configurar o jogo e acompanhar o desenvolvimento dos estudantes e um protótipo funcional de um jogo sério.

O jogo deverá estar acessível online e dividido em diversos níveis encadeados uns nos outros, seguindo uma história cativante, e a disponibilidade destes deverá ser configurada pelo docente de modo que estes acompanhem ao mesmo passo a que as aulas são dadas.

Os níveis deverão proporcionar motivação e um desafio ao jogador/estudante para obter os conhecimentos necessários para resolver o problema e avançar o nível e a história.

Finalmente, deverão existir um grupo de ferramentas que permitam gerir e alterar o comportamento do jogo e o conteúdo apresentado aos alunos, bem como visualizar um conjunto de estatísticas sobre a progressão da sua aprendizagem.

1.4 Cooperação entre projectos

Este projecto foi desenvolvido em cooperação com o Enrique Kato, uma vez que a sua dissertação é a implementação da mesma plataforma sobre o ponto de vista de design. Através desta cooperação, este dissertação beneficiou bastante da troca de ideias e da perspectiva de design que o Enrique proporcionou ao nível da mecânica de jogo, da história e das missões criadas para o protótipo. Por outro lado, a dissertação desenvolvida pelo Enrique beneficiou de poder ter as suas ideias implementadas.

1.5 Estrutura da Dissertação

Este documento está estruturado em seis capítulos, sendo o primeiro composto por esta introdução.

O capítulo 2 introduz o tema Edutainment, explora os tipos de jogos sérios e projectos desta área, aborda o processo de desenvolvimento de jogos sérios e as vantagens de usar um motor de jogo em vez de desenvolver um jogo totalmente de raiz, uma análise resumida de quatro motores de jogos actuais, finalizando com um breve sumário das principais ideias.

No capítulo 3 é descrita a mecânica de jogo adoptada, a especificação do planeamento para a implementação da plataforma e quais as ferramentas que irão ser usadas no seu desenvolvimento.

O capítulo 4 explora os detalhes mais relevantes na implementação da plataforma.

No capítulo 5 é avaliada a implementação de acordo com a especificação feita no capítulo 4 e analisados os testes feitos à plataforma e os resultados do inquérito feito

aos participantes nesses testes. É também explicada a natureza da cooperação com outra dissertação em curso e a historia criada para o protótipo.

Finalmente, o ultimo capítulo faz um apanhado de tudo o que foi feito, juntamente com as respectivas conclusões e considerações finais. Este capítulo inclui também possível trabalho futuro.

No final deste documento foi incluído um apêndice com a descrição de instalação da plataforma e protótipo.

Capítulo 2

Jogos Sérios para o Ensino

Neste capítulo será analisada a forma de ensino através de entretenimento, *edutainment*, seguida da análise de vários jogos e projectos relacionados com esta dissertação. Serão também analisados alguns motores de jogos e finalmente será feito um breve sumário com as ideias principais a reter do trabalho relacionado.

2.1 Edutainment

Edutainment é educação através do entretenimento, isto é, aprendizagem enquanto se faz uma actividade de um carácter mais lúdico, no entanto mantendo sempre presente que o objectivo é aprender. Na sua essência, esta forma de aprendizagem já existe há centenas de anos, através de contos e fábulas contadas aos mais novos no sentido de lhes transmitir ideias e conhecimentos. Na nossa história mais recentemente, este método evoluiu para outros formatos como programas de televisão e jogos digitais.

Os jogos digitais para a educação distinguem-se dos restantes por servirem um propósito específico para além de serem meros mecanismos de divertimento.

Por essa razão a elaboração do conceito do jogo, sendo este o factor mais importante de sucesso [WJ03], deverá concentrar-se mais no objectivo educacional do que no entretenimento que este deverá proporcionar. No entanto sempre que possível deverão ser feitos os possíveis esforços para se conseguir chegar a um equilíbrio que maximize as potencialidades de ambas as vertentes.

Outro factor importante que pode contribuir para o sucesso de um jogo sério, é os jogos terem a possibilidade de criarem ambientes virtuais onde podemos experimentar, tomar riscos e explorar várias vias para atingir objectivos e/ou resolver problemas [Fra].

Esses ambientes estão directamente dependentes do conceito e objectivos que cada jogo tem, dando origem aos seguintes tipos de jogos sérios:

- **Simulação** — Neste tipo de jogo sério o conceito principal é o de tentar reproduzir a realidade tão fielmente quanto possível de modo a proporcionar uma experiência realista que permita o jogador aprender pela prática. Um bom exemplo é o famoso Miscrosoft Flight Simulator, no qual é possível pilotar várias aeronaves reais por todo o planeta executando missões variadas, ou assumindo o papel de controlador de voo numa torre de controlo de um aeroporto;

- **Estratégia e exercício de raciocínio** — O xadrez talvez será o melhor exemplo de um jogo deste tipo, pois permite exercitar a nossa capacidade de antevisão e planeamento [Fer];

- **Treino físico** — Com a saída da Wii no mercado das consolas, um novo tipo de jogo apareceu, o Wii Fit. Este jogo dispõe de vários sensores que permitem ao jogo seguir os movimentos do jogador e aconselhar correcções em vários tipos de exercício físico, como Yoga, entre outros;

- **Estudo de áreas teóricas** — Este tipo de jogo foca áreas científicas mais teóricas, como história, biologia, política, etc. Exemplo deste tipo de jogo é o PeaceMaker que coloca o jogador no papel de governante do estado Israelita ou Palestiniano e através de decisões governamentais tem que conseguir a atingir a paz;

- **Estudo de áreas práticas** — Este tipo de jogo foca áreas mais práticas, como a matemática, física, programação, etc. O Colobot é um jogo deste tipo, baseado num cenário fictício em que o planeta terra sofre um cataclismo terrível que força a humanidade a procurar um planeta capaz de sustentar vida. Para ajudar o jogador na sua missão, este tem robôs à sua disposição, os quais pode programar numa linguagem muito semelhante à linguagem de programação Java para executar várias tarefas, como recolha de recursos, construção ou defesa;

- **Desenvolvimento de capacidades inter-pessoais** — Os jogos deste tipo têm por função ajudar os jogadores a melhorar as suas capacidades de relacionamento com outros indivíduos e com as actividades que realiza em grupo. O melhor exemplo é o jogo NoviCraft o qual suporta vários jogadores e tem o objectivo de ajudar os seus jogadores a desenvolver as suas capacidades de cooperação e liderança através da resolução de puzzles e/ou problemas em grupo;

- **Publicidade** — Actualmente, este é possivelmente o tipo mais comum dentro dos jogos sérios e é caracterizado pelo objectivo de publicitar produtos e/ou serviços. Este tipo de jogo está presente principalmente em banners de páginas-web;

- **Recrutamento** — Estes jogos servem fundamentalmente para recrutar indivíduos, como é o caso do Americas's Army [SL], um jogo de acção que coloca o jogador

na posição soldado do exército dos Estados Unidos da América e dar uma imagem do que esperar na vida de soldado.

2.2 Projectos na área de Edutainment

A área de edutainment e jogos digitais sérios é hoje em dia já bastante extensa e entre os inúmeros jogos existentes, há alguns projectos que sobressaiam devido a factores como a sua natureza multidisciplinar e os seus objectivos sociais para o melhoramento da espécie humana e do mundo em que vivemos. Nas próximas subsecções são apresentados alguns projectos que demonstram exactamente esses factores e uma empresa internacional que opera na área de jogos sérios, bem como alguns projectos na área do ensino da programação de bastante relevância para este projecto.

2.2.1 Robocode

O Robocode é um jogo de programação, cujo objectivo lúdico é programar um tanque de guerra para enfrentar outros tanques desenvolvidos da mesma forma. O objectivo sério deste jogo é facilitar a aprendizagem das linguagens Java e C#, bem como a aprendizagem de inteligência artificial [Rob].

Esta plataforma contém à partida integrados um editor de robôs e um compilador java, tendo apenas como requisito a existência da plataforma Java Virtual Machine (JVM) na máquina onde o robocode será instalado.

O mundo virtual deste jogo é representado numa vista top-down 2D, cuja qualidade gráfica pode ser considerada demasiado antiga e a qual influenciou negativamente a reacção de vários estudantes [KA].

2.2.2 Wu's Castle

O Wu's Castle é um jogo 2D desenvolvido no RPG Maker desenvolvido com o objectivo de ensinar a usar ciclos e matrizes de dados de forma interactiva e visual. Este jogo proporciona *feedback* imediato e representação visual da execução do código ajudando o utilizador a ter uma melhor noção do que está a acontecer.

Num estudo feito por Michael Eagle e Tiffany Barnes [MT] sobre os resultados do uso deste jogo na aprendizagem da programação, foram criados dois grupos de teste, sendo dado a um dos grupos um exercício comum de programação comum e ao outro foi dado o jogo para aprender através deste e depois resolveram o mesmo exercício dado ao primeiro grupo. Este teste demonstrou que o grupo que usou o jogo como fonte de aprendizagem obteve melhores resultados que o primeiro grupo e todos os alunos intervenientes demonstraram preferência pela aprendizagem com recurso de jogos sérios.

2.2.3 Alice

Alice é uma plataforma de programação 3D a qual permite criar animações para contar histórias e jogos. Isto é conseguido através do arraste de objectos gráficos para a cena, os quais correspondem a funcionalidades especificas, da mesma forma que objectos programados numa linguagem de programação orientada a objectos, como o C++, teriam. Os conceitos de programação orientada a objectos são assim transmitidos aos alunos, contudo sem nunca terem contacto directo com uma verdadeira linguagem de programação.

Através deste sistema de drag&drop os alunos podem obter resultados imediatos, permitindo uma maior rapidez de compreensão de como os objectos interagem entre si, evitando também erros de sintaxe comuns aos iniciantes da programação, agilizando a sua aprendizagem.

O mundo virtual é representado em 3D, fácil de usar e intuitivo o que é um factor fundamental para atrair a atenção e o envolvimento dos jogadores [BDS].

2.2.4 M.U.P.P.E.T.S.

Sistema Multi-Utilizador para o melhoramento do estudo tradicional da programação (em inglês Multi-User Programming Pedagogy for Enhancing Traditional Study) é um sistema desenvolvido no Rochster Institute of Technology, com o objectivo de substituir o uso do Robocode [KA].

O M.U.P.P.E.T.S. é um sistema multi-utilizador desenhado para permitir aos alunos desenvolver objectos em Java, bem como criar robôs da mesma forma que o Robocode permite, no entanto com um grau de customização superior. A representação visual do mundo virtual e dos objectos criados é feita em 3D e contem ferramentas para importação de texturas e modelos.

Este sistema é baseado em módulos, os quais podem ser facilmente activados ou desactivados, conforme as necessidades e os objectivos de aprendizagem.

2.2.5 C-Sheep

A C-Sheep é uma mini linguagem de programação baseada na linguagem ANSI C, cujo objectivo é ajudar na aprendizagem dos princípios da ciência computacional e também da linguagem C. Esta mini linguagem permite programar o comportamento de uma ovelha virtual, cujo habitat natural é um prado verde modelado em 3D denominado The Meadow [EL].

O mundo virtual, The Meadow, inclui uma máquina virtual a qual avalia o código fonte escrito em C-Sheep e executa-o. Os resultados da execução são visíveis através da observação do comportamento da ovelha no mundo virtual. Uma vez que se trata de um

jogo 3D moderno, espera-se obter uma maior atenção por parte dos alunos e levar estes a sentir um maior prazer em programar.

2.2.6 CiberRato

O CiberRato é uma modalidade integrada no concurso Micro-Rato [MR] da Universidade de Aveiro, a qual consiste no desenvolvimento de robôs ao nível algorítmico, sem a necessidade da sua construção física, que sejam capazes de executar tarefas sem colidir com outros objectos, como por exemplo, encontrar a saída de um labirinto. De modo a poder desenvolver e observar os robôs, os utilizadores têm ao seu dispor uma ferramenta que executa o codigo que controla os robôs, comunica com os seus sensores de modo a informá-los de que existe algum objecto na sua proximidade e dispõe de uma interface gráfica 2D na qual se pode seguir o robô a movimentar-se no seu ambiente.

Os robôs são programados usando a linguagem Java, que, do ponto de vista de edutainment é um ponto forte, pois além de permitir desenvolver competências ao nível algorítmico e de inteligência artificial, permite aprender uma linguagem de programação bastante popular.

2.2.7 DEI Academy

A DEI Academy é um projecto do Departamento de Engenharia Informática da Universidade de Coimbra, o qual foi feito para ajudar os estudantes do ensino secundário na sua decisão do rumo que irão tomar para o ensino superior ao nível da informática e da programação [Aca].

Este projecto consiste num portal web, o qual está aberto a todos que se queiram registar, e lá pode-se encontrar vários desafios de programação na linguagem Python, de dificuldade variada, tutoriais dessa mesma linguagem e um fórum para colocar dúvidas. Para resolver os desafios de programação, os utilizadores submetem os programas que serão processados por um sistema de avaliação automática de programas chamado Mooshak (desenvolvido na Faculdade de Ciências da Universidade do Porto).

É um projecto inovador em Portugal, com interface acessível que permite aos seus utilizadores aprender ou melhorar os seus conhecimentos de programação tanto ao nível algorítmico como ao nível da linguagem em si, bem como ajudar os mais novos a entrar em contacto com o mundo da programação de uma forma acompanhada e divertida.

2.2.8 Stop Disasters

Com o propósito de sensibilizar os mais novos para a prevenção de catástrofes naturais, as Nações Unidas desenvolveram um jogo digital ao qual chamaram Stop Disasters [fDR].

Este jogo foi desenvolvido em Flash e é suportado por uma página web que além do jogo contém informação adicional sobre os vários tipos de desastres naturais e uma página com o ranking dos jogadores mais bem sucedidos na prevenção de catástrofes.

O jogo em si, permite simular vários tipos de desastres naturais - tsunamis; cheias; terramotos; cheias e fogos florestais - com diferentes níveis de dificuldade e diferentes zonas do mapa terrestre. Para a prevenção dos desastres, o jogador tem um orçamento inicial que deverá usar para construir e desenvolver estruturas que o ajudem na sua missão, adicionando há já presente componente educacional a aprendizagem de gestão de recursos.

2.2.9 Playgen

A Playgen é um estúdio de desenvolvimento de jogos sérios [Plab], cuja contribuição para esta área tem sido bastante importante através dos vários jogos que desenvolveram, bem como força dinamizadora através da criação de eventos e workshops para formação de aprendizagem de como desenvolver jogos sérios.

Outro aspecto notável é terem disponibilizado vários dos jogos desenvolvidos na sua página web, direccionados para várias áreas de interesse. Além da possibilidade de jogar os vários jogos disponíveis na sua página web, contém também informação sobre jogos sérios, como estes se aplicam e sua utilidade, e também um mapa de eventos.

Esta empresa, é sem dúvida um exemplo a seguir, pelo seu dinamismo e contribuição para um futuro em que a aprendizagem se torne cada vez mais em algo divertido e fácil de absorver.

2.3 Desenvolvimento de Jogos Sérios

Há cerca de vinte anos atrás, o desenvolvimento de jogos era maioritariamente amador e feito por uma ou duas pessoas, no entanto à medida que as capacidades computacionais evoluíram e novas tecnologias surgiram, os jogos tornaram-se cada vez mais complexos ao nível de conceito, visual, áudio e conteúdo.

O que era antes algo relativamente simples, hoje requer plataformas bastante complexas, mais conhecidas por motores de jogo, e na maioria dos casos uma equipa extensa e profissional que seguem um processo de desenvolvimento, adaptado às suas necessidades, no entanto há alguns passos que são comuns no desenvolvimento de jogos sérios [Plaa]:

- **Estudo do tema sério** — Antes de começar o desenvolvimento, há que ter uma clara ideia do que se pretende transmitir, isto é, qual a componente séria do jogo;
- **Conceito** — Neste passo é onde a equipa gera ideias para as melhores formas de implementar a componente séria de uma forma divertida;

- **Design** — Após ter várias ideias escolhidas, há que verificar se estas são fazíveis e começar a estruturar o projecto;
- **Desenvolvimento** — Neste passo, com o conceito e desenho do projecto definidos, passa-se ao desenvolvimento do jogo;
- **Lançamento da versão Beta** — A esta altura, o jogo já deverá encontrar-se num estado quase acabado, faltando corrigir pequenas falhas que normalmente serão detectadas por equipas de pessoas chamadas de beta-testers;
- **Lançamento Final** — As falhas reportadas são corrigidas e o jogo final está completo e concluído.

É importante referir que, estes passos deverão ser considerados como guias. O processo de criação de qualquer aplicação informática deverá ser tão flexível quanto possível de modo a ser capaz de integrar alterações ao conceito que possam surgir como absolutamente necessárias em qualquer momento do desenvolvimento.

2.4 Motores de Jogos

Um motor de jogo é uma plataforma concebida para o desenvolvimento de jogos, composta por várias componentes como motor de renderização 2D e/ou 3D, motor de física, suporte de áudio, inteligência artificial, animação, suporte de rede, suporte de scripting e ferramentas de desenvolvimento. Estes sistemas tornam o desenvolvimento mais rápido e permitem reutilização em vários projectos diferentes.

Devido à variedade de plataformas de hardware disponíveis no mercado, de modo a que uma aplicação/jogo pudesse ser executada em qualquer uma dessas plataformas, era necessário passar por um processo de conversão custoso e que em muitos casos não era perfeito. Para ultrapassar esse problema, os motores começam agora a adaptar-se e a suportar diferentes sistemas operativos e plataformas de entretenimento como a Xbox e a Playstation, entre outras.

Na selecção de um motor de jogo para esta dissertação, o motor escolhido deverá permitir a implementação de todas as opções escolhidas no capítulo 3, ser tão completo quanto possível e permitir representação gráfica e sonora de alta qualidade.

Seguindo o estudo desenvolvido por Panagiotis Petridis e seus colegas relativamente a uma metodologia de selecção motores de jogos para jogos sérios [PDdFP10], a tabela 2.2 foi concebida para ajudar na escolha do motor mais adequado ao desenvolvimento desta dissertação. Nesse estudo foram focadas as áreas:

- **Fidelidade audiovisual e funcional** — Capacidade de transmitir ideias e imagens tão reais quanto possível;

- **Agregabilidade** — Capacidade de importar/exportar e modificar conteúdo;
- **Acessibilidade** — O quão simples é o uso do motor para desenvolvimento;
- **Rede** — Capacidade de suportar vários jogadores;
- **Heterogeneidade** — Capacidade de suportar diferentes plataformas.

Fidelidade audiovisual	Suporte Gráfico
	Animação
	Áudio
Fidelidade funcional	Scripting
	Inteligência Artificial
	Motor de física
Agregabilidade	Importação de conteúdo CAD
	Editor de ambiente integrado
Acessibilidade	Código Fonte
	Curva de aprendizagem
	Documentação e tutoriais
	Licença
Rede	Suporte de rede
Heterogeneidade	Integração em web browser
	Multi Plataformas

Tabela 2.1: Metodologia para comparação de motores de jogos

Através da tabela 2.2 pode-se verificar que a nível de capacidades todos os motores apresentados são bastante completos e semelhantes entre si, no entanto aquilo que os diferencia poderá ser fundamental para a implementação do jogo.

Para melhor poder decidir qual dos motores se adequa melhor para o desenvolvimento desta dissertação, há que compreender melhor a natureza das principais diferenças entre os motores:

- Inteligência artificial - Nos motores analisados o suporte para a inteligência artificial é feita sobre três formas, parcialmente ou totalmente:
  - Pesquisa de caminho (Pathfinding) é a pesquisa do caminho mais curto entre dois pontos;
  - Tomada de decisões é um processo através do qual se chega a um curso de acção, segundo objectivos e possíveis vias alternativas a seguir pré estabelecidas;
  - Através de scripts, modelar o comportamento dos objectos dentro do jogo recorrendo a pesquisa de caminhos e tomadas de decisão, caso estas estejam disponíveis, e à implementação de outros sistemas.

Componente	Unity3D	UDK	CryEngine 3	Torque 3D
Suporte Gráfico	3D	3D	3D	3D
Animação	Sim	Sim	Sim	Sim
Áudio	2D & 3D	2D & 3D	2D & 3D	2D & 3D
Scripting	Javascript; C#.NET	UC	LUA	TS
Inteligência Artificial	Scripts	Scripts; Pathfinding; Decision Making	Scripts; Pathfinding; Decision Making	Scripts
Motor de física	Detecção de Colisões; Rigid Body; Vehicle Physics	Detecção de Colisões; Rigid Body; Vehicle Physics	Detecção de Colisões; Rigid Body; Vehicle Physics	Detecção de Colisões; Rigid Body; Vehicle Physics
Importação de conteúdo CAD	3ds max; maya	3ds max; maya	3ds max; maya	3ds max; maya
Editor de ambiente integrado	Sim	Sim	Sim	Sim
Código Fonte	Não	Não	Sim	Sim
Curva de aprendizagem	Média	Elevada	Média	Média
Documentação e tutoriais	Sim	Sim	Sim	Sim
Lincença	Gratuita	Gratuita	Gratuita[a]	$99
Suporte de Rede	Cliente/Servidor	Sim	Sim	Sim
Integração em web browser	Sim	Não	Não	Sim
Plataformas	PC; Mac; XBox; iOS	PC; iOS	PC; PSX; XBox; Gamecube	PC; Mac

Tabela 2.2: Comparação entre dois motores actuais

[a] Licença Educacional apenas para Universidades que façam uma aplicação que seja aceite pela Crytek.

- Integração em web browser - Uma vez que um dos objectivos é poder controlar o jogo através de uma página web, esta característica poderá ser útil para simplificar o acesso e instalação do jogo;
- Código fonte - O acesso ao código fonte poderá ser bastante útil no caso de se verificar necessária a alteração do motor.

2.5 Sumário

Neste capítulo começou-se por perceber o fenómeno chamado de Edutainment para melhor compreender esta área onde a aprendizagem encontra o divertimento, analisando os vários tipos de jogos que foram surgindo de acordo com as necessidades emergentes.

Foram analisados também vários jogos e projectos que procuram facilitar a aprendizagem da programação e desta análise ficou claro que nos dias de hoje, com a tecnologia disponível, simples gráficos 2D já não são suficientes para cativar e manter o interesse

continuado dos alunos, sendo por isso o uso de gráficos 3D tão modernos quanto possíveis um factor que poderá ser de bastante importância. Verificou-se também que a maioria dos jogos foca-se em apenas uma ou duas linguagens de programação, ou até evitam totalmente a sintaxe de programação como é o caso da Alice. Este é um outro factor de peso no sucesso da plataforma que esta dissertação pretende implementar, isto é, que linguagens são suportadas - tantas quanto possível, todas seria o ideal. Uma vez que a representação 3D do mundo virtual é um factor quase que obrigatório, finalizou-se este capítulo com uma breve análise a vários motores de jogos actuais, no sentido de poder escolher um que melhor se adapte ao que é pretendido implementar nesta dissertação.

Capítulo 3

Mecânica do Jogo e Planeamento

Neste capítulo será definida a mecânica de jogo a implementar, bem como as estratégias de planeamento para uma implementação bem sucedida.

3.1 Mecânica de Jogo

A mecânica de um jogo pode ser definida como conjunto de regras que define todas as possíveis interacções com o jogo a que pertencem e como este se comporta perante elas. Estas regras estão presentes em todos os tipos de jogos, sejam eles jogos desportivos, de tabuleiro ou jogos de computador [Fab07].

A primeira coisa a ter em mente ao definir a mecânica deste jogo é que, uma vez que se trata de um jogo sério o ênfase deste terá que ser na contribuição educacional que este dará. Esta contribuição será feita através de exercícios de programação e a parte lúdica será tentar tornar a resolução destes exercícios tão divertida quanto possível.

Habitualmente os jogos contêm um ou mais mundos virtuais que contêm missões relacionadas com a história do jogo. No caso desta plataforma, essas missões serão exercícios de programação descritos de forma a que pareçam missões relacionadas com a história desenvolvida para o jogo, podendo cada jogo gerado ter a sua própria história. Estas missões serão acessíveis através de terminais dispostos pelo mundo virtual com os quais o jogador poderá interagir.

Com estas ideias presentes e de acordo com as conclusões do capítulo anterior, a representação do mundo virtual será feita em 3D com uma visão de trás do avatar e o seu movimento será feito através do teclado em conjunto com o rato, imitando o sistema de controlo usado no famoso jogo World of Warcraft. O movimento terá liberdade de 360º, podendo movimentar-se em qualquer direcção desde que tenha chão por baixo dos pés, ou saltando para onde este exista.

O jogador movimentar-se-á num mundo virtual 3D baseado numa matriz de células que podem estar vazias ou conter chão ou paredes. Esse mundo terá também rico em objectos que permitam ao jogador aceitar novos exercícios, conhecer a história e obter dicas sobre os exercícios. Para além desses objectos, deverão existir outros que potenciem a componente lúdica do jogo, como por exemplo campos de força ou teletransportadores. O mundo estará dividido em diversos mapas pequenos, tendo que existir portas que permitam a movimentação dos jogadores entre os vários mapas. Esta funcionalidade poderá permitir gerar múltiplas vias de progresso no jogo.

De forma a poder jogar, o jogador terá que se autenticar e posteriormente seleccionar um perfil, entre dez perfis, em que deseja jogar. Estes perfis são definitivos, isto é, não podem ser alterados. O objectivo destes perfis é dar a possibilidade de os jogadores poderem completar o jogo várias vezes, podendo até seguir diferentes vias de solução.

Relativamente aos exercícios, apenas se pode resolver um de cada vez e sempre que este é resolvido é atribuída uma pontuação ao jogador, tendo cada exercício uma pontuação específica. No caso de errar a resolução, é incrementado o número de tentativas e quando finalmente este exercício for resolvido, a pontuação recebe uma penalização baseada no número de submissões erradas.

O jogo poderá ser prolongado indefinidamente, a menos que seja adicionado um tipo de objecto especial cujo uso implique o fim do jogo.

3.1.1 História do Protótipo

A história é um elemento bastante importante em quase todos os tipos de jogos, pois dá sentido e profundidade a toda a experiência proporcionada pelos mesmos. Uma boa história poderá ser fundamental para manter e/ou aumentar o interesse por um determinado jogo, como quem lê um livro desejoso de conhecer todos os detalhes da trama.

Foi criada uma história base a partir da qual será possível expandir, para que cada jogo tenha a mesma base mas um desenvolvimento diferente, todavia ao gerar um jogo não é obrigatório o uso desta historia base já que a plataforma deverá por si mesma proporcionar liberdade total a esse nível.

A história base tem lugar algures num futuro no qual os seres humanos terrestres, após atingirem um grau de consciência mais elevado que lhes permitiu ultrapassar as ilusões que os mantinham em permanente conflito entre si e as outras espécies do planeta, foram contactados pela federação intergaláctica e convidados a integrar a mesma.

Os humanos da terra aceitam e são lhes atribuídas várias responsabilidades, sendo uma delas a de embaixadores espaciais. A missão destes embaixadores será ajudar outros planetas em desenvolvimento e estudar as suas civilizações.

O jogador toma o papel de um destes embaixadores e através do seu progresso nos vários mapas e do seu progresso nas várias missões vai obtendo informações sobre o planeta a que lhe foi atribuído.

3.2 Planeamento

Com a mecânica de jogo definida, o próximo passo será definir um plano de desenvolvimento, começando pela especificação da arquitectura que a plataforma tomará, seguida por uma análise de cada um dos seus componentes e sub-componentes.

3.2.1 Arquitectura

Logo à partida, há duas componentes que, em função da mecânica do jogo definida, são imediatas e absolutamente necessárias. Uma componente representa o cliente, com a qual o jogador irá interagir directamente, e a outra representa o servidor, a qual disponibiliza um ou mais serviços, com a qual a componente anterior interage remotamente através de uma rede local ou de rede alargada (WAN). Esta arquitectura segue a típica arquitectura cliente/servidor, no entanto, uma vez que parte fundamental da mecânica do jogo implica a resolução e compilação de exercícios de programação, surge a necessidade do uso de um sistema de Correcção Assistida por Computador ou CBA, neste caso o DOMJudge.

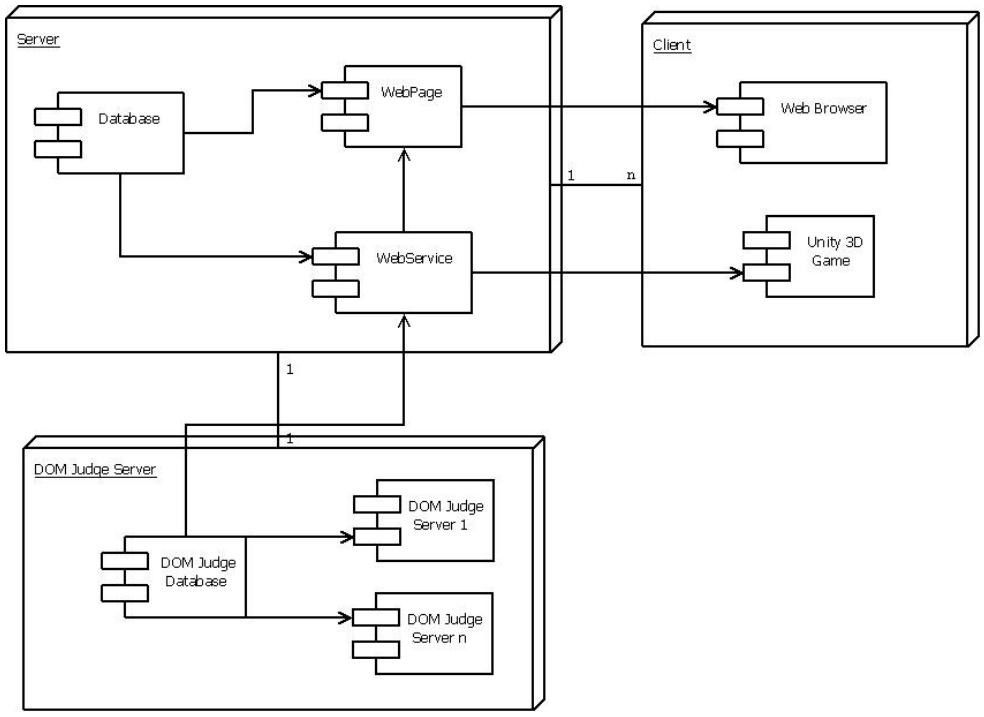

Figura 3.1: Arquitectura geral

Na imagem 3.1 podem-se observar três super componentes ligadas entre si. A componente de servidor, a qual será desenvolvida sobre a .NET 4.0 Framework e que controlará toda a plataforma. A componente cliente, a qual proporciona o jogo, desenvolvido com o motor Unity3D, o qual será embebido numa página web e acessível através de um browser. Finalmente o pacote do sistema DOMJudge, que através das suas capacidades de avaliação de código fonte permitirá à plataforma obter o resultado de soluções para as missões submetidas pelos jogadores.

3.2.1.1 Casos de Uso

Os seguintes casos de uso representam a forma como os actores intervenientes nesta plataforma (docente, aluno e serviço web) interagem com esta. A figura 3.2 mostra a interacção geral dos vários actores com a plataforma. A vista geral é expandida em quatro casos de usos, respectivamente, nas figuras 3.3, 3.4, 3.5 e 3.6.

A figura 3.3 demonstra as interacções que poderão ser feitas pelo docente ao nível da gestão da plataforma, podendo criar/editar/remover mapas, adicionar objectos aos mapas já criados, bem como editar as suas propriedades ou removê-los. Deverá poder criar/editar/remover missões e associa-las a objectos já existentes e a exercícios presentes no DOMJudge. Terá também a capacidade de adicionar novos utilizadores e automaticamente adicionar à base de dados do DOMJudge uma equipa associada ao utilizador. Finalmente, deverá poder definir variáveis de jogo, como por exemplo nome do jogo gerado.

Na figura 3.4 estão representadas as interacções do docente e dos utilizadores. O docente poderá verificar o progresso pessoal de cada um dos alunos, interagir com um fórum e observar as estatísticas de jogo relativas ao progresso de todos os jogadores. O utilizador irá ter acesso apenas ao seu progresso pessoal, no entanto poderá interagir com o fórum e visualizar as estatísticas.

A figura 3.5 demonstra as interacções que deverão ser possíveis no jogo pelo aluno. O aluno deverá poder autenticar-se, seleccionar perfil, aceitar missão, submeter soluções para as missões aceites, movimentar-se no mundo virtual e interagir com objectos.

A figura 3.6 permite observar a interacção do serviço web, que no fundo se traduzem nas suas funcionalidades. Isto é, obter os dados necessários da base de dados, como por exemplo as missões, os mapas, os utilizadores, etc.

3.2.2 Componente Servidor

Como mencionado anteriormente, esta componente irá disponibilizar serviços para os outros componentes da arquitectura e a implementação da plataforma deverá centralizar tanto quanto possível a gestão e processamento da mecânica do jogo nesta componente.

Desta forma, retirando a capacidade de decisão da componente do cliente, a arquitectura estará mais segura contra ataques e contra a manipulação de resultados.

Pode-se então descrever esta componente como o "coração", gerindo o tráfego de dados (informação) para as outras componentes vitais, e como o cérebro desta plataforma, pois a gestão e tomadas de decisão serão aqui centralizadas

O funcionamento correcto desta componente será assegurado por três sub-componentes:

- Base de dados - Toda a informação que seja necessária guardar será guardada numa base de dados, acessível apenas localmente.
- Página Web - Esta sub-componente proporcionará um painel de controlo para que o docente responsável pela administração da plataforma possa gerir todos os aspectos do jogo, através de ferramentas como editor de mapas e exercícios, e ao mesmo tempo será também uma interface que lhe permitirá seguir o progresso dos alunos, bem como permitirá aos próprios alunos seguir o seu progresso pessoal e receber feedback do docente.
- Serviço web - A maioria da lógica de jogo será processada por esta componente e funcionará como intermediário entre a componente cliente e a base de dados central.

3.2.3 Componente Cliente

Esta componente é em essência a interface que permitirá aos utilizadores desta plataforma interagir com a mesma. Essa interacção será conseguida através do uso de um web browser, tanto para aceder à página web, assim como o próprio o jogo, isto é, este estará integrado na página e acessível através desta.

3.2.3.1 Mundo Virtual

Através dos dados guardados na base de dados, transmitidos através do serviço web segundo a lógica de jogo definida, o jogo construirá um mundo virtual 3D baseado numa matriz de células, cada uma podendo conter muros nas quatro direcções dos pontos cardiais, (Norte, Este, Sul e Oeste). O mundo virtual conterá também objectos com os quais será possível interagir:

- Directamente - Através de um click do botão esquerdo do rato sobre o objecto ou pelo movimento, por exemplo clicar numa porta para mudar de mapa, ou passar por cima de um teletransportador (Nota: os objectos têm de estar activos para que se possa interagir com eles);
- Indirectamente - Ao concluir um exercício com sucesso que active ou desactive um objecto, ou grupo de objectos, por exemplo, um exercício que active uma porta que

permita a progressão do jogo para outro mapa com novos exercícios, ou desactivar um grupo de campos de força que impedem o acesso a uma determinada área do mapa.

Foram definidos sete tipos de objecto, no entanto poderão ser adicionados mais conforme as necessidades ditarem, com os quais é possível interagir:

- Terminal - Este objecto permite aceitar novos exercícios e aceder directamente ao menu de submissão do código fonte relativo ao exercício associado com cada terminal (mais sobre os menus à frente);
- Porta - A navegação entre os vários mapas é conseguido através deste objecto;
- Teletransportador - Através deste objecto é possível aceder a áreas do jogo que possam, por exemplo, estar inacessíveis directamente;
- DataPad - Este objecto permite dar informações ao jogador relevantes à história do jogo ou dicas importantes para os exercícios a resolver;
- Campo de Forças - Este objecto serve para impedir o acesso a determinadas áreas do mapa;
- Luzes - Iluminam a cena;
- Objecto fim de jogo - Assinala o fim do jogo e é o objectivo final.

Através do uso destes objectos, será possível criar mundos ricos em opções e exercícios, em que, como no exemplo dado na imagem 3.8, será possível ter vários caminhos no qual um é mais imediato, no entanto mais difícil e para o qual é possível que o jogador possa ainda não ter os conhecimentos necessários para conseguir resolver o exercício que abre a porta e para esse caso, existir uma via mais fácil que lhe permitirá adquirir esses conhecimentos de forma a progredir.

3.2.3.2 Menus

Os Menus são conjuntos de opções ou comandos dispostos graficamente representados através da combinação de elementos nativos da interface usada para os gerar como texto, imagens, botões, etc. Através destas opções ou comandos, o utilizador pode escolher várias vias de execução ou formas de execução ou simplesmente ter acesso a informação.

Os jogos irão dispor dos seguintes nove menus:

- Autenticação - Quando o jogo é executado, este menu é a primeira coisa que aparece no ecrã do jogo e sem uma autenticação bem sucedida, não será possível passar deste ponto;

- Perfis - Após o utilizador se ter autenticado com sucesso, navegará para este menu onde poderá escolher um perfil, entre os dez disponíveis;
- Detalhes de Perfil - Uma vez escolhido um perfil, o jogo passará para este menu onde serão visíveis detalhes deste perfil e as escolhas de voltar atrás, jogar ou reiniciar o perfil, isto é, apagar todo o progresso feito neste perfil e começar de novo;
- Lista de exercícios - Após ter sido escolhido um perfil e ter escolhido jogar, a qualquer momento do jogo, através do uso de uma tecla do teclado pré definida, o utilizador poderá aceder a este menu que lhe permitirá visualizar todos os exercícios, resolvidos e todos aqueles que já descobriu durante a exploração do jogo, mas ainda por resolver;
- Submissão de código - Existirão duas formas de aceder a este menu, através do terminal associado ao exercício em questão, ou através do menu Lista de exercícios. Este menu permitirá submeter o código fonte para resolver o exercício e deverá aguardar até que o resultado seja recebido e então proceder à lógica de jogo definida para cada possibilidade;
- Informação - Este menu é o mais simples de todos e serve apenas para dar alguma informação ao jogador, acessível através do uso do objecto DataPad;
- Ajudas - Este menu pode conter várias instâncias, cada um com um texto definido na base de dados com o intuito de dar informação extra ao jogador;
- Menu Introdutório - Este menu pode conter várias instâncias, cada uma tendo um texto e uma imagem que servem para dar informação introdutória dos jogos, por exemplo da história;
- Menu de Controlos - Este menu contém informação sobre os controlos usados nos jogos;

3.2.3.3 Comunicação com o servidor

Uma vez que o jogo irá comunicar com o servidor através de uma rede digital (como mencionado na secção 3.2.1), há que definir as funções remotas oferecidas pelo serviço web. Estas funções são chamadas remotamente pelo jogo e após a sua execução, consoante a função chamada, valores poderão ser retornados.

3.2.4 Componente CBA

A componente CBA (Computer Based Assessment) é um sistema que, através do uso de um computador, permite automatizar o processo de avaliação de exercícios, testes, exames, etc.

Função	Descrição
Autenticação	Recebe informação do utilizador e se for válida, retorna os perfis do utilizador.
Obter Exercícios	Recebe o identificador do perfil escolhido e retorna todos os exercícios aceites resolvidos e ainda por resolver.
Obter Mapa	Recebe o identificador do mapa e retorna a informação do mapa que permitirá construir o mundo virtual 3D, bem como lista de objectos presentes nesse mapa e exercícios associados a estes.
Aceitar exercício	Recebe o identificador do exercício e adiciona-o à lista de exercícios.
Submeter código fonte	Recebe o identificador do exercício e o código fonte e submete-o para análise, retornando um valor booleano que representa o sucesso da operação.
Obter estado do código	Não recebe argumentos de entrada, no entanto verifica o estado da análise da operação e retorna esse estado.
Salvar estado do jogador	Recebe o estado actual do jogador e actualiza o perfil.

Tabela 3.1: Descrição das funções remotas a implementar

Uma vez que a avaliação do código fonte submetido através do jogo requer uma avaliação rápida (de apenas alguns segundos), torna-se fundamental o uso de um sistema destes para conseguir atingir esse objectivo. Esta necessidade de rapidez de avaliação advém do facto de que, enquanto o código está a ser avaliado, o jogador está à espera para retomar o jogo e uma longa espera poderá ter um impacto negativo na motivação de continuar a jogar e de futuras tentativas.

Este sistema deverá possibilitar alguma forma de acesso ao serviço web de modo a que este possa enviar o código fonte para ser avaliado e posteriormente, após a avaliação ter sido concluída, conseguir obter o resultado.

3.3 Definição de ferramentas a utilizar

Com o planeamento definido, antes de passar à implementação, resta seleccionar quais as ferramentas a usar no desenvolvimento desta plataforma, começando pelo motor do jogo seguidamente do sistema de avaliação automático e finalmente da base de dados.

3.3.1 Motor de jogo

Após considerar os motores mencionados, o Unity3D de todos é o que melhor se adequa para conseguir atingir os objectivos traçados para esta dissertação, pela possibilidade de ser integrado numa página web e pela facilidade de comunicação com serviços web através da plataforma .NET Framework. Estas características que acabaram por determinar a escolha do motor serão exploradas em detalhe no próximo capítulo.

O Unity3D, conhecido pela sua flexibilidade a nível de criação de jogos, limitado apenas pela tecnologia existente actualmente e pela imaginação, tem por base alguns conceitos chave os quais são necessários compreender de modo a poder usufruir total partido de todo o potencial que esta ferramenta de desenvolvimento possibilita. Destes conceitos o mais importante e que de certa forma sustenta todos os outros é o conceito de GameObject. Através dos GameObjects, é possível hierarquizar objectos complexos em pequenas partes, tornando-os mais fáceis de gerir e reutilizar.

Suportados pela arquitectura proporcionada pelo conceito de GameObject, os outros conceitos importantes são:

- Cenas - Uma Cena é um *placeholder* para objectos que fazem parte de uma determinada parte de um jogo, sendo bastante útil para separar diferentes áreas de um jogo permitindo distribuir conteúdo mais eficientemente e testa-las individualmente;
- Recursos/Conteúdo - Texturas, modelos, sons e todos os recursos usados para construir e modelar o jogo;
- Componentes - Através de componentes, é possível adicionar e alterar o comportamento de GameObjects como por exemplo adicionar detecção de colisões ou adicionar elementos básicos como luzes;
- Scripts - Os scripts permitem expandir e adicionar funcionalidades e através destas implementar a lógica e mecânica do jogo a ser desenvolvido (na verdade o Unity3D classifica os scripts como componentes);
- Pré-fabricados - A reutilização de objectos é uma necessidade fundamental em qualquer jogo e através de pré-fabricados, objectos não instanciados são guardados e estão prontos a ser usados em qualquer ponto do jogo, sempre que necessário.

3.3.2 Sistema de avaliação automático

Após ler e analisar as conclusões obtidas por Pedro Pacheco na sua tese Computer-Based Assessment System for e-Learning [Pac10], o DOMJudge foi eleito como o sistema mais adequado para integrar a plataforma a ser desenvolvida nesta dissertação.

O factor que mais pesou na escolha do DOMJudge foi a possibilidade de este poder suportar qualquer linguagem de programação, atribuindo assim uma versatilidade ilimitada a este nível à plataforma que irá integrar. Além disso, é um sistema de código aberto baseado numa arquitectura distribuída que permite ter vários processos de avaliação a executar paralelamente, o que implica uma maior rapidez e escalabilidade para atender um maior número de pedidos.

A melhor forma encontrada para utilizar este sistema que o evite alterar, será aceder directamente à base de dados deste sistema e inserir directamente nesta as submissões de

soluções enviadas pelos jogadores. A avaliação de soluções poderá ser descrita em três passos simples:

- A solução para um exercício é submetida e guardada na base de dados associada a este sistema;
- O primeiro serviço de análise disponível analisa a solução e marca esta solução como sobre avaliação;
- Finalmente, após a análise ter sido concluída, o resultado é guardado na base de dados, juntamente com as saídas do compilador e da solução.

Após uma submissão, o jogo passará a um estado de espera no qual enviará um pedido ao serviço web periodicamente. O serviço web ao receber o pedido voltará a aceder à base de dados do DOMJudge para verificar se a solução submetida já foi ou não avaliada, retornando esse resultado para o jogo.

3.3.3 Base de dados

A selecção da base de dados recaiu sobre o MySql e foi influenciada pelo facto de o DOMJudge a usar nativamente, e caso outra base de dados fosse usada, implicaria a implementação de pelo menos duas formas de acesso à base de dados. Desta forma, a implementação, a este nível, ficará um pouco mais simplificada.

Por outro lado, esta base de dados é bastante poderosa e muito usada ao nível mundial, dispondo de licença gratuita e em virtude de ter uma política de código aberto, goza de uma comunidade dedicada bem como documentação altamente informativa, através das quais é relativamente fácil aprender e obter soluções para eventuais problemas.

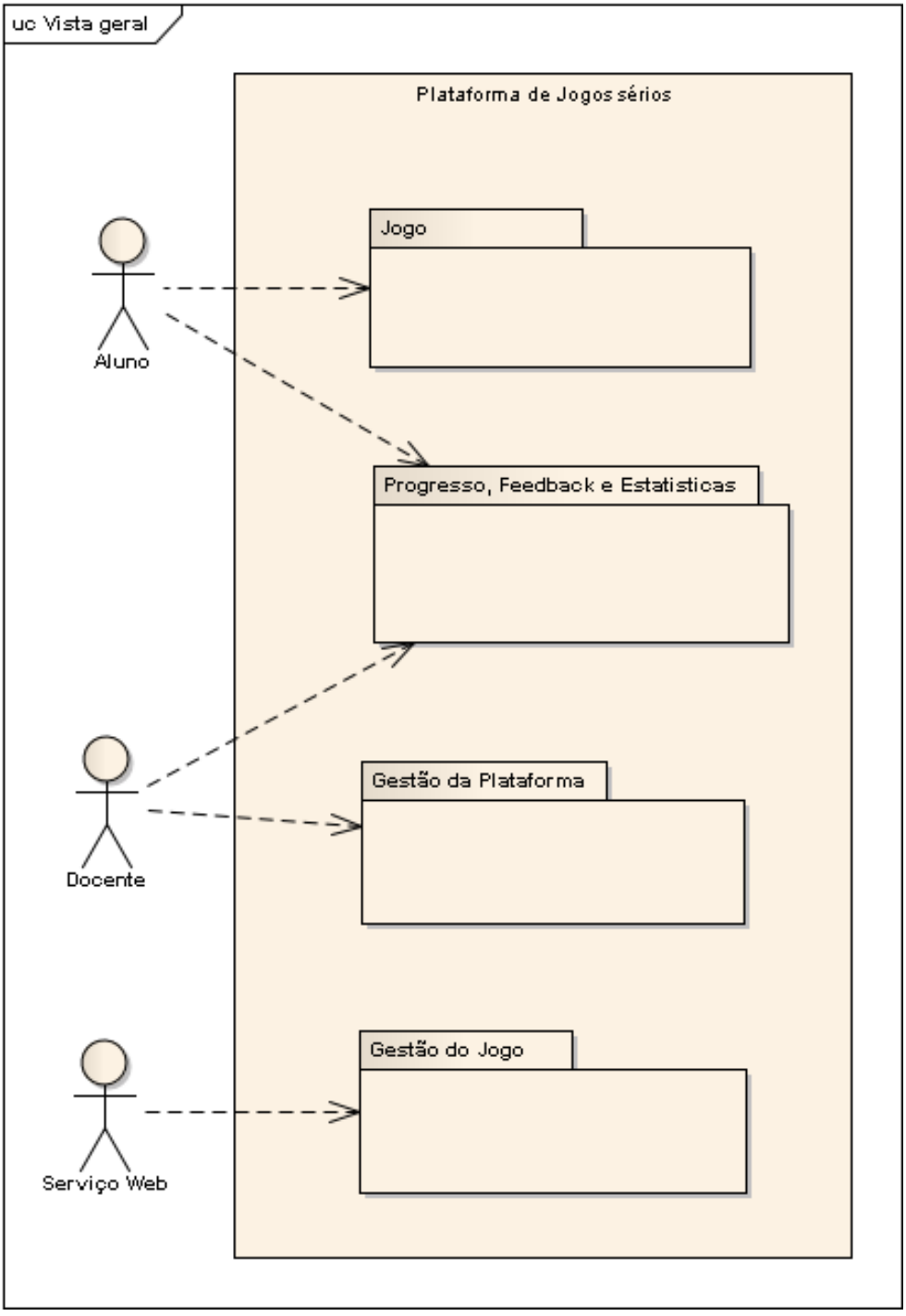

Figura 3.2: Caso de uso geral da plataforma

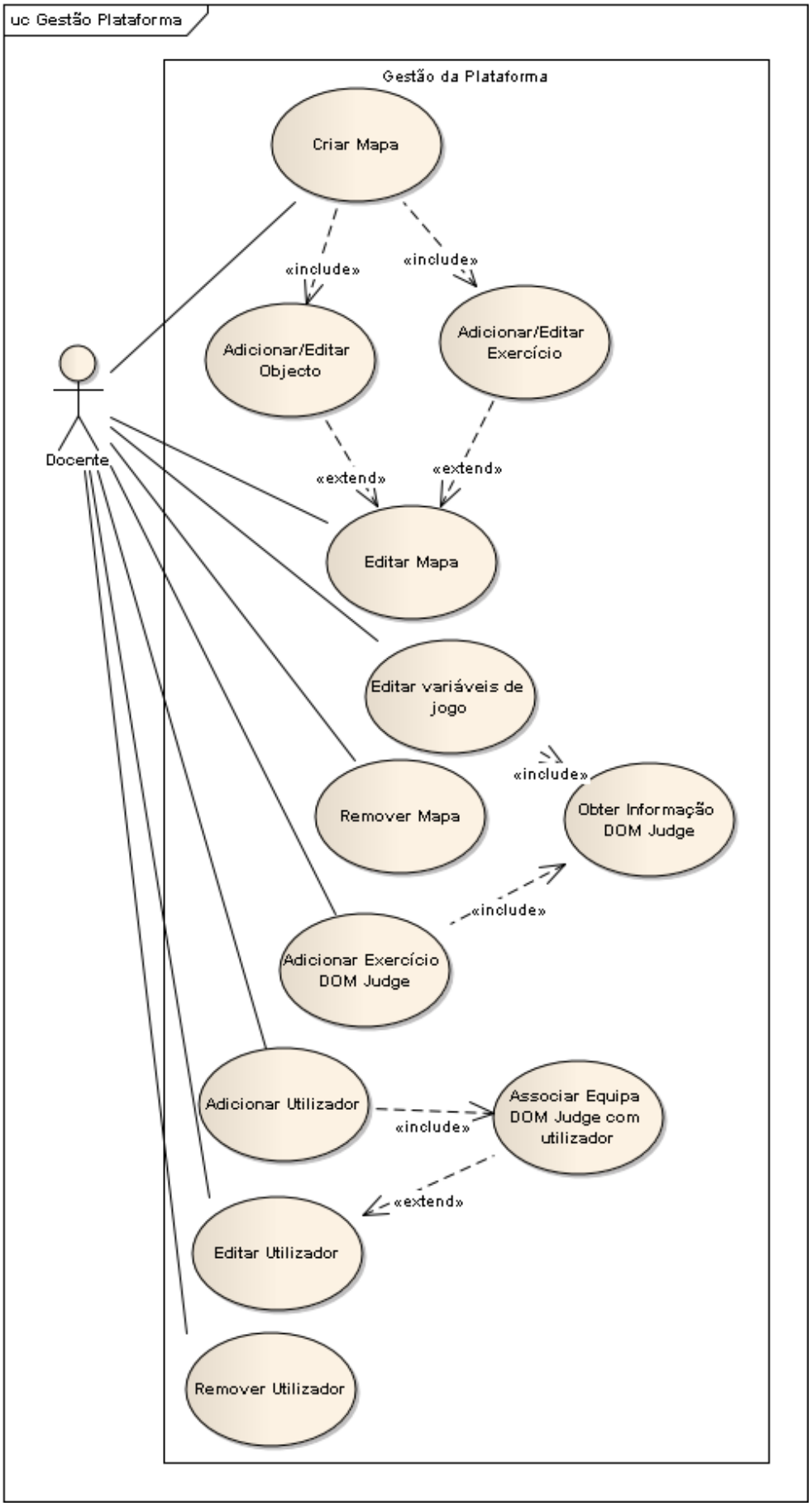

Figura 3.3: Caso de uso da gestão da plataforma

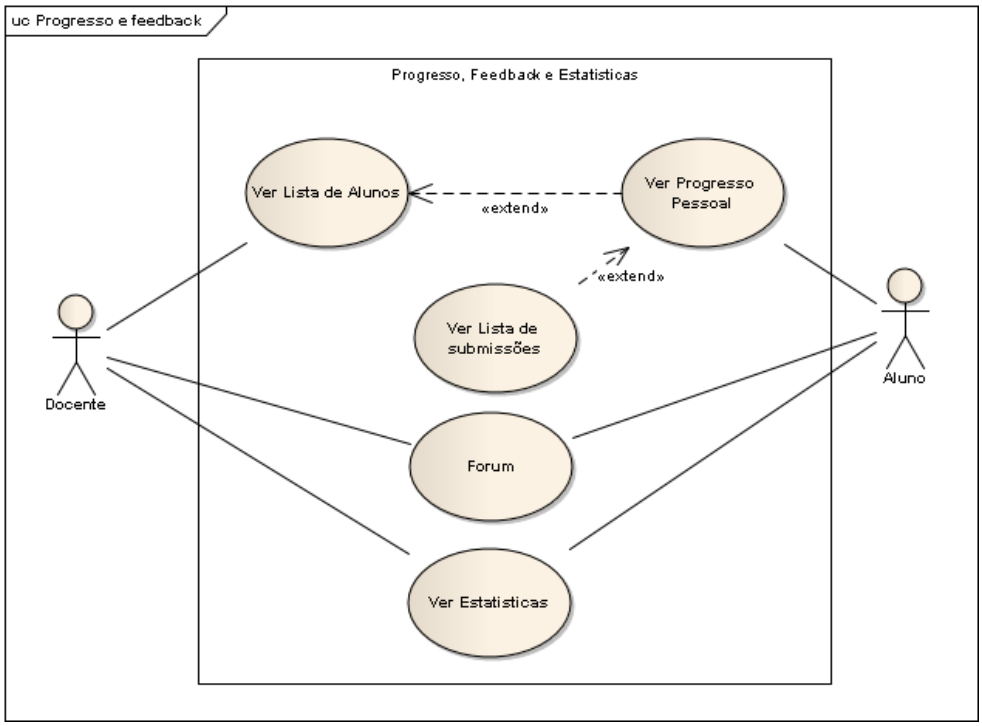

Figura 3.4: Caso de uso da visualização de progresso de alunos, feedback e estatísticas

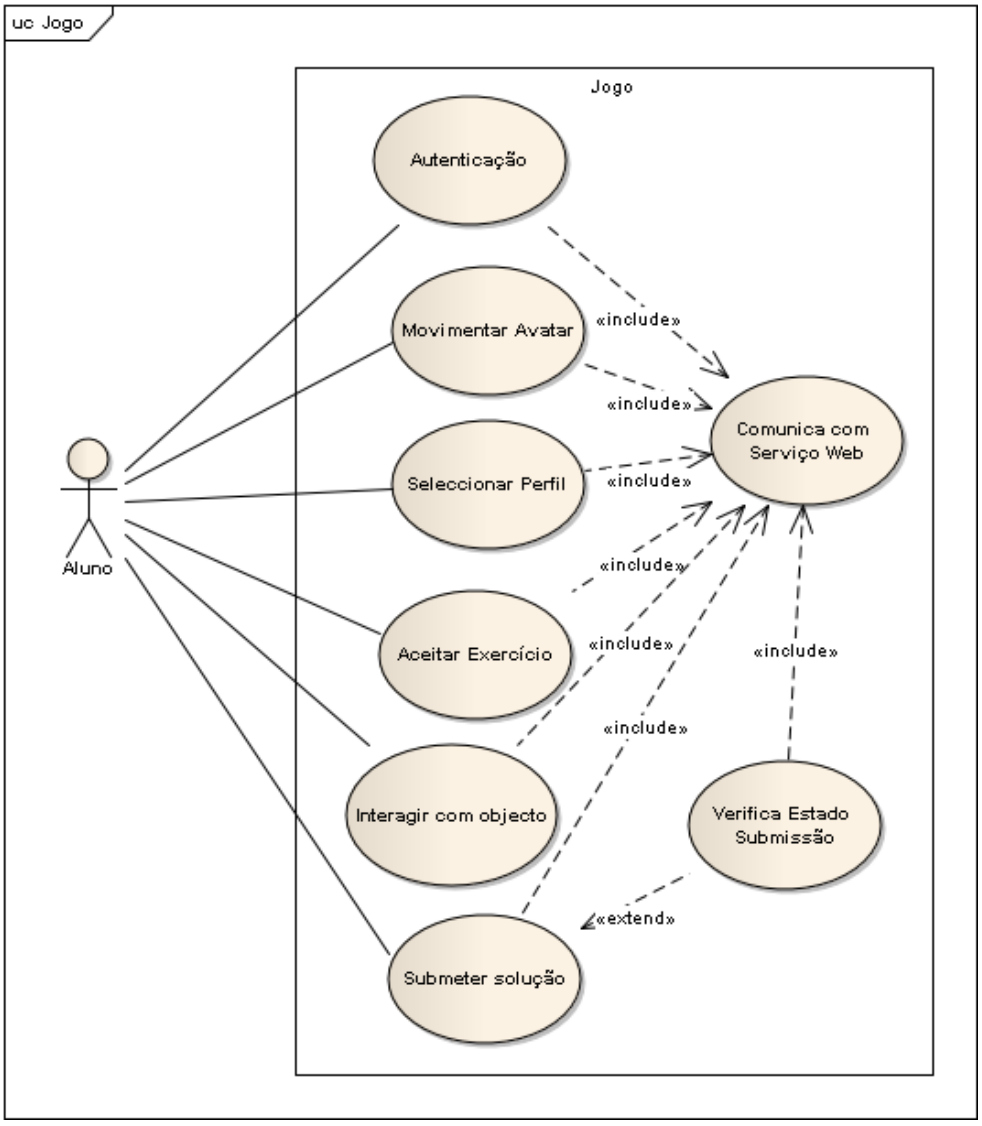

Figura 3.5: Caso de uso do jogo

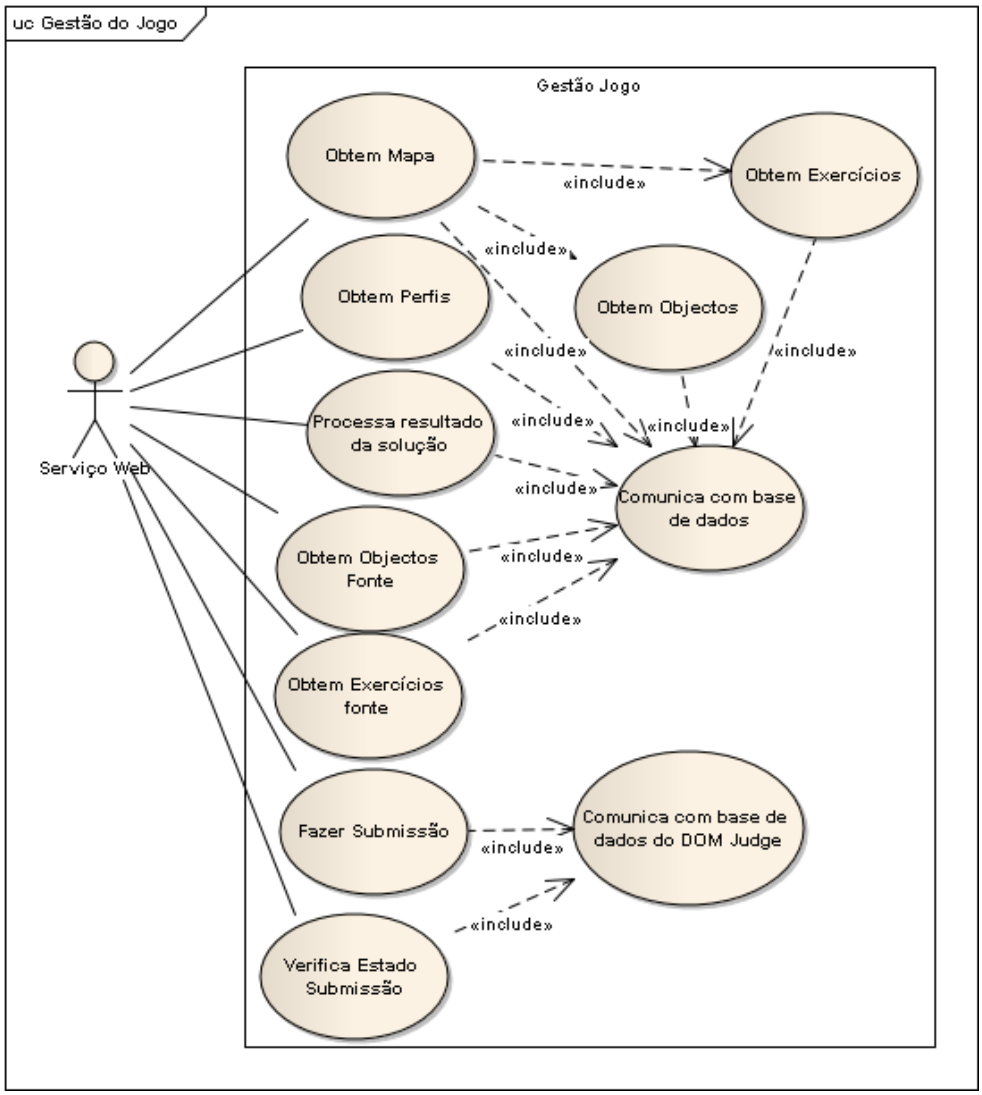

Figura 3.6: Caso de uso da gestão do jogo

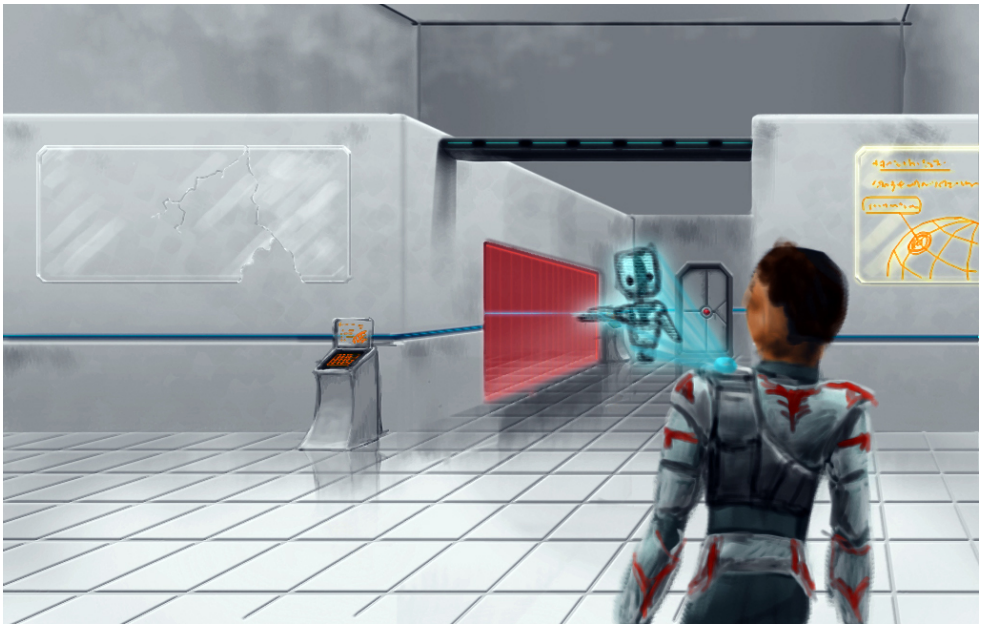

Figura 3.7: Exemplo conceptual de um mapa (desenhado por Enrique Kato)

Figura 3.8: Exemplo conceptual de progresso de mapas (desenhado por Enrique Kato)

Capítulo 4

Implementação

Usando as especificações definidas no capítulo anterior como guia, passou-se à implementação da plataforma e, posteriormente, à elaboração de um protótipo de teste baseado num nível/mapa de jogo com acesso a todos os objectos implementados. Este protótipo serve como prova de conceito e será a base para o próximo capítulo.

O serviço web foi implementado sobre .NET Framework e a linguagem C# foi usada no seu desenvolvimento, bem como nos scripts usados no jogo. Ambas as componentes partilham de algumas das classes implementadas..

Neste capítulo serão explorados em detalhe várias aspectos da implementação da plataforma, nomeadamente a estrutura de dados adoptada, vários detalhes de funcionamento do serviço web e do jogo e opções de implementação escolhidas.

4.1 Modelo de dados

O primeiro passo na implementação foi definir um modelo de dados para a base de dados, uma vez que a forma como os dados são estruturados e definidos é potencialmente um elemento chave para uma boa implementação da solução visada. Seguindo as especificações definidas no planeamento, a base de dados foi estruturada de forma a assegurar, a este nível, a correcta implementação das regras e funcionamento da plataforma. A figura 4.1 representa o relacionamento entre cada uma das tabelas e a tabela 4.1 descreve cada uma delas.

Os dados contidos na base de dados poderão ser considerados brutos, pelo que cabe ao serviço web e ao jogo tratar e processar estes dados. Assim, cada uma destas componentes tem o seu modelo de dados individual diferente um do outro, mas que no entanto têm algumas classes em comum, como é visível nas tabelas B.1 e B.2 em anexo. Como seria

Tabela	Descrição
User	Guarda os dados que permitem autenticar e conceder acesso aos utilizadores
Profile	Guarda todos os perfis associados a um utilizador
ProfileQuest	Guarda todas os exercícios aceites pelo utilizador num determinado perfil e o estado de resolução
ProfileObject	Guarda o estado dos objectos modificados num determinado perfil
Room	Guarda a matriz de células e outros dados que permitem ao jogo construir o mundo virtual
SourceQuest	Guarda todos os exercícios disponíveis para resolver
SourceObject	Guarda todos os objectos presentes no jogo
QuestSubmission	Guarda cada submissão de cada solução feita pelo utilizador e respectivo resultado
ObjectType	Guarda a descrição dos vários tipos de objectos
Group	Guarda identificadores que servem para agrupar objectos em grupos especificos
Settings	Guarda várias definições necessárias para o funcionamento da plataforma
Texture	Guarda o URL que permite ao jogo ler e usar as texturas escolhidas na tabela Room

Tabela 4.1: Descrição das tabelas presentes no modelo de dados ER

de esperar, estes diagramas são semelhantes ao diagrama da base de dados ao nível da definição classe - tabela, relações e atributos.

As tabelas B.1 e B.2, em anexo, descrevem as várias classes criadas, das quais duas se destacam devido ao seu papel fundamental no funcionamento de cada uma das componentes. Essas classes são a GameHandler no serviço web e a GameInfo no jogo.

4.2 Funcionamento geral

Quando um utilizador executa o jogo, o menu de login é a primeira coisa a aparecer. Após se autenticar correctamente, o jogo muda para o menu de perfis, onde os dez perfis disponíveis lhe são apresentados numa lista. Clicando nos elementos da lista o jogador irá seleccionar um perfil e avançar para o menu seguinte - o menu de detalhes de perfil. Uma vez nesse menu, poderá começar a jogar ou reinicializar o perfil. Ao seleccionar a opção de começar a jogar, o mapa identificado pelo perfil seleccionado é carregado do serviço web e construído em 3D. A partir desse momento o jogador, usando as teclas do teclado e o rato, pode movimentar o seu avatar pelo mapa e interagir com os vários objectos disponíveis.

Os processos externos visíveis na figura A.1 (processos dentro dos quadrados), em anexo, estão detalhados na figura A.2. Estes processos correspondem às funções remotas

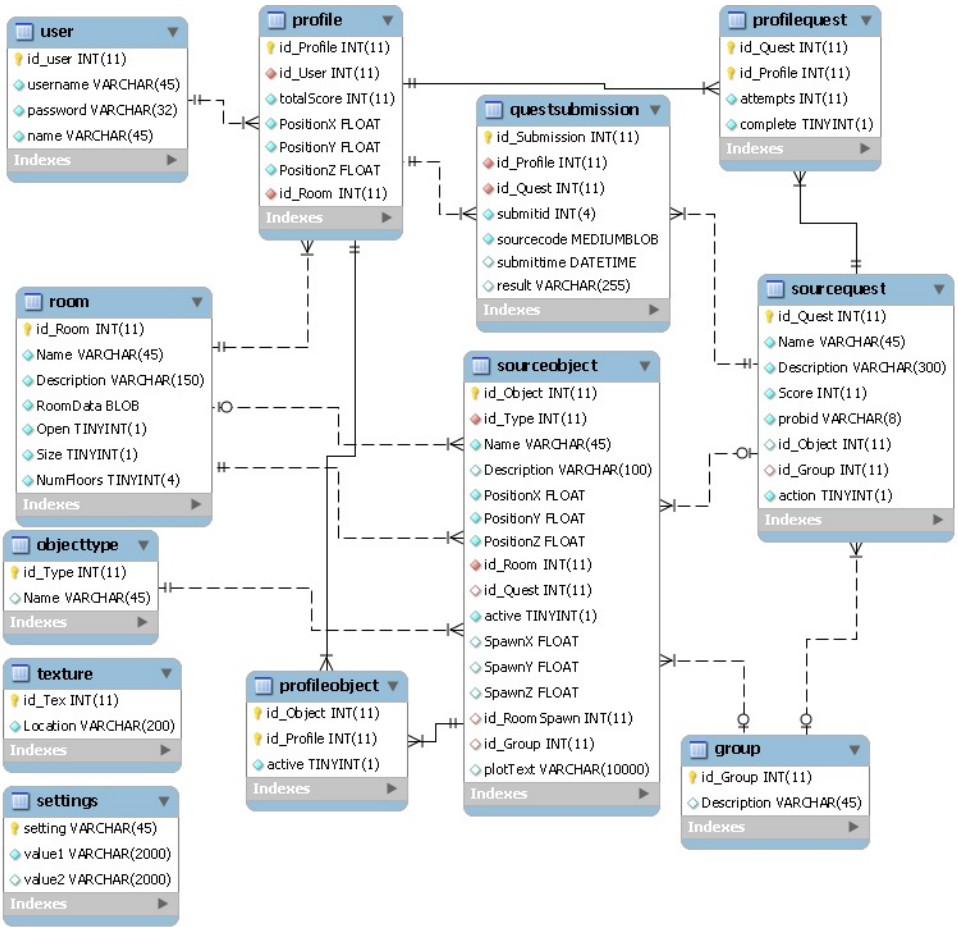

Figura 4.1: Caso de uso da gestão do jogo

do serviço web, descritos na secção 3.2.3.3 com a adição da função Reinicializar Serviço e Modo de Edição - mais informações na tabela A.1 em anexo.

De forma a evitar que um utilizador pudesse ter várias instâncias do jogo a executar, foi criado um sistema de sessões que atribui uma chave de sessão a cada utilizador que liga ao servidor. No caso de um utilizador abrir uma nova instância do jogo, ao fazer novo login, a chave de sessão é renovada e a chave anterior perde a validade e uma vez que nenhuma operação pode ser feita sem uma chave válida, instâncias anteriores, deixarão de funcionar.

As sessões são geridas pela classe *ServiceHandler*, especificada na secção anterior, que além de guardar a chave de sessão, guarda também a informação dos perfis e os mapas carregados, evitando assim constante acesso à base de dados, ou seja, fazendo cache da informação.

Logo após o início da execução do serviço web, a classe *ServiceHandler* gera uma cache estática das tabelas sourceQuest e sourceObject. Estes dados são guardados numa cache estática pois raramente serão alterados, no entanto caso ocorram alterações a este,

ou a outras definições da tabela settings, é possível gerar novas caches e definições, usando para isso a função de reinicialização.

Sempre que um jogador faz login, é verificado se há alguma submissão pendente e no caso de se confirmar, verifica-se de imediato se já foi avaliada. Se já foi avaliada, procede-se à lógica executada quando um exercício é resolvido com sucesso e continua o funcionamento normal. No caso de ainda estar à espera de ser avaliada, o jogo é notificado e ao entrar no jogo, este é imediatamente levado para o menu de espera. Este mecanismo foi implementado para atender à eventualidade de ocorrer uma interrupção inesperada do processo de submissão.

Para quando surgir a necessidade de editar os valores do jogo, como por exemplo adicionar uma mapa novo, ou alterar um já existente ou os objectos nele contidos, foi criada uma função que coloca o serviço em modo de edição. Enquanto esse modo estiver activo, nenhum jogo conseguirá passar do menu de login e todas as chaves de sessão são invalidadas, impedindo o acesso ao serviço por qualquer jogo que esteja em execução.

Da mesma forma que o serviço web tem uma classe (*ServiceHandler*) que o gere, o jogo também tem uma classe responsável de gerir o jogo - *GameHandler*. Esta classe é particularmente útil pois esta assegura que nenhuma informação fundamental é perdida, uma vez que o Unity3D cada vez que muda de cena destrói todos os objectos instanciados nela, perdendo assim toda a informação nela contida. Esta informação, cuja preservação é vital, corresponde aos dados da sessão do jogador e respectivos perfis, mapas obtidos até ao momento (evitando estar sempre a fazer um pedido remoto quando se muda de mapa), todos os objectos e exercícios relacionados com cada mapa e várias outras variáveis de estado que ajudam o jogo a identificar o processo de execução correcto.

4.3 Submissão de exercícios e interacção com o DOM Judge

Quando um exercício é submetido pelo jogo, o primeiro passo é verificar se já existe alguma submissão pendente, isto é, um exercício submetido anteriormente ainda esteja a ser avaliado pelo DOM Judge. Nesse caso, o processo de submissão termina e é retornado o valor booleano de falso, identificador de que a submissão falhou. Caso não haja nenhuma submissão pendente, uma nova entrada é adicionada à tabela submission da base de dados associada ao DOM Judge.

Imediatamente a seguir é inserida uma nova entrada na tabela *questsubmission*, registando a submissão associada com um perfil. Finalmente, esta submissão é definida como pendente e é retornado o valor booleano verdadeiro, identificando o sucesso da operação. O jogo passa para o modo de espera, chamando periodicamente a função remota "obter estado da solução"até que seja retornado um estado de solução correcta ou incorrecta. Esta função irá aceder a tabela *judging* e procurar se existe alguma entrada associada com a submissão a ser verificada. Caso não exista, significa que ainda está em espera para ser

avaliada e é retornado o estado. No caso de já existir, significa que já a avaliação já teve início e o do valor do campo *result* identificará se a solução ainda está a ser avaliada, no caso de estar vazio, ou se já foi concluída, contendo o resultado da avaliação.

Se a solução estiver ainda a ser avaliada, é retornado esse estado, caso contrário, se a solução estiver correcta a pontuação associada ao exercício, menos penalizações no caso de mais do que uma tentativa, é adicionada à pontuação total do perfil, o exercício é actualizado como completo e todos os objectos associados são actualizados. Caso a solução esteja errada, apenas é incrementado o número de tentativas. Independentemente do resultado da solução é enviado esse resultado, permitindo ao jogo sair do modo de espera.

4.4 Requisitos da Plataforma

Uma vez que existem três componentes individuais na arquitectura, existem igualmente três definições de requisitos, definidos nas tabelas 4.2, 4.3 e 4.4. A tabela de requisitos do servidor baseia-se nos requisitos necessários para executar a plataforma *.NET Framework 4.0*, uma vez que é esta que suporta esta componente. No caso da tabela de requisitos da componente do cliente, usou-se como base os requisitos defindos pelos autores do Unity3D para executar este motor de jogos. Finalmente, a tabela de requisitos do DOMJudge baseia-se nos requisitos estabelecidos pelos autores desse sistema.

Hardware	**Mínimo**	**Recomendado**
Processador	1GHz	2 GHz
Ram	512MB	1024MB
Espaço em Disco	850MB (32bit) ou 2 GB (64bit)	
Software		
Sistema Operativo	Windows XP; Windows Vista; Windows 7	
Outros	Windows Installer 3.1; .NET 4.0 Framework; MySQL >= 5.5; MySQL Connector	

Tabela 4.2: Requisitos da componente servidor

Hardware	**Mínimo**	**Recomendado**
Processador	1GHz	2 GHz
Ram	512MB	1024MB
Memória Gráfica	64MB	256MB
Espaço em Disco	> 1GB	
Software		
Sistema Operativo	Windows XP; Windows Vista; Windows 7; Mac OS X	
Web Browser	Internet Explorer; Firefox; Chrome; Safari; Opera; Camino	

Tabela 4.3: Requisitos da componente cliente

Hardware	**Mínimo**	**Recomendado**
Processador	1GHz	2 GHz
Ram	512MB	1024MB
Espaço em Disco	> 1GB	
Software		
Sistema Operativo	Unix; Linux	
Outros	Apache web server; PHP 5; MySQL >= 5.5; Compiladores que se deseja suportar	

Tabela 4.4: Requisitos da componente DOMJudge

Capítulo 5

Resultados

Neste capítulo é avaliada a implementação da plataforma de acordo com a especificação feita no capítulo 3. São também avaliados os resultados da sessão de testes ao protótipo desenvolvido, bem como os resultados do inquérito preenchidos pelos alunos que participaram.

5.1 Resultados da Implementação

Da especificação idealizada para esta plataforma foram implementadas todas as componentes com sucesso, todavia devido aos limites de tempo que regem esta dissertação não foi possivel implementar a página web da componente servidor.

Como consequência da não implementação da página web, elemento de controlo e sistema de interacção docente - aluno, a plataforma ainda se encontra incompleta e sem este elemento fundamental não é possível testar a plataforma na sua totalidade.

Todavia foi elaborado um protótipo para testar as funcionalidades implementadas o qual foi testado por um grupo de alunos da FEUP. A resposta ao protótipo foi bastante positiva, no entanto por não estarem familiarizados houve alguma dificuldade inicial em usar os controlos do jogo (teclado + rato).

Relativamente à conectividade com o serviço web, o tempo de resposta aos pedidos ao servidor eram quase instantâneos e o mesmo se verificou ao carregar as várias texturas usadas no jogo. Da mesma forma, a conectividade com a base de dados do DOMJudge ocorreu sem atrasos.

O protótipo gerado consiste em dois mapas, cada um com duas missões (exercícios). No primeiro mapa o jogador tem que desactivar um campo de forças que o impede de aceder a um quarto, como se pode observar na figura 5.2 do lado esquerdo, tendo que aceder ao terminal exterior. Dentro do quarto fechado encontra-se um teleportador que

transporta o jogador para imediatamente acima, no entanto, este objecto encontra-se inicialmente inactivo, sendo necessário resolver mais uma missão para o activar. Para ajudar na resolução deste exercício, perto do terminal interior, existe um data pad com informações adicionais. Uma vez resolvido o segundo exercício e ter sido usado o teleportador, o jogador terá que saltar vários pilares para conseguir chegar a uma plataforma com uma porta que o leva ao segundo mapa.

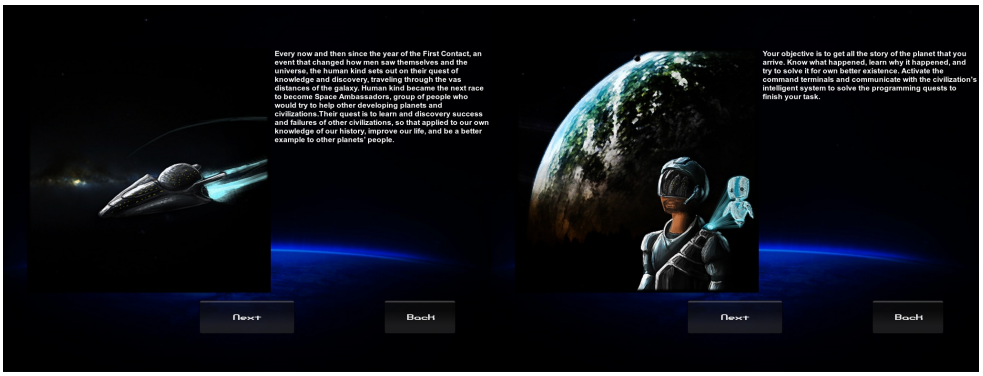

Figura 5.1: Menus de introdução à história do jogo (Imagens criadas por Enrique Kato)

Uma vez no segundo mapa, a primeira coisa que se nota é que o mapa está às escuras e o ultimo terminal está também inactivo. Imediatamente à frente da porta de onde o jogador veio existe um terminal com um novo exercício. O objectivo do terceiro exercício é activar as luzes do mapa e o terminal final. Uma vez chegado ao último terminal, é dada a missão final, no entanto não existe muita informação para conseguir concluir este exercício a menos que o jogador encontre um data pad com as informações necessárias.

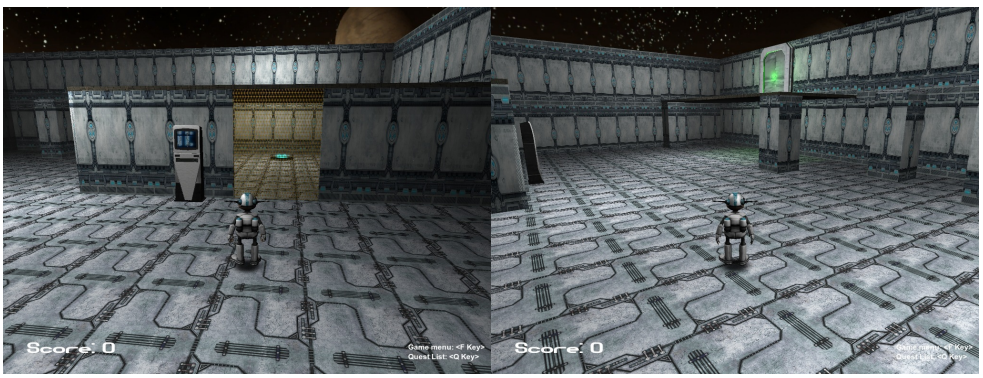

Figura 5.2: Imagens do primeiro mapa do protótipo

Finalmente, quando a última missão é concluída com sucesso, a porta de saída do jogo é activada e ao interagir com esta, um menu de fim de jogo congratula o jogador por ter conseguido concluir o jogo.

A parte lúdica presente neste protótipo, além de resolver os problemas consistiu também numa mecânica semelhante aos jogos de plataformas, ao ter que saltar os vários pilares no primeiro mapa e na exploração do mundo virtual, no segundo mapa, procurando pela informação necessária para resolver a última missão.

Para criar os mapas a serem usados no protótipo teve que ser elaborado um editor de mapas simples, limitado a criar mapas de tamanho fixo de 20x2x20.

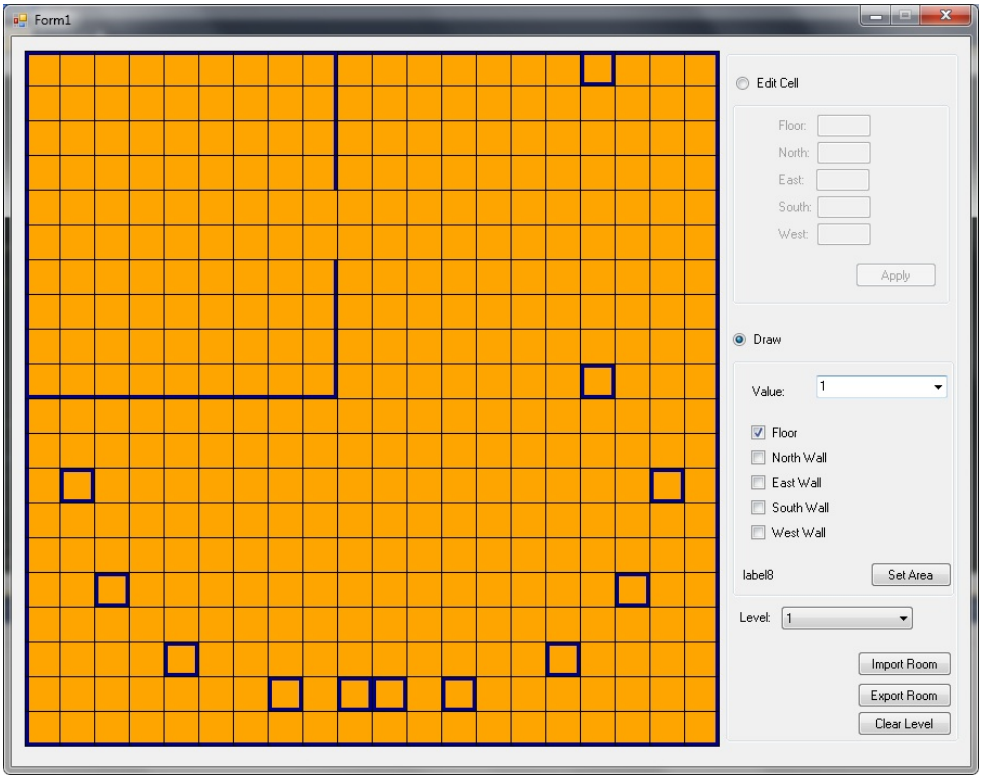

Figura 5.3: Editor de mapas temporário

No final do teste, os intervenientes foram convidados a preencher um inquérito sobre o protótipo do jogo.

5.2 Inquérito

O inquérito preenchido pelos participantes foi elaborado com os objectivos de ajudar a avaliar o protótipo e determinar possíveis aspectos a melhorar. Com esses objectivos em mente, construiu-se um inquérito de dez frases, tabela 5.1, às quais pode ser associado individualmente um nível de concordância. As figuras 5.4, 5.5, 5.6, 5.7 e 5.8 demonstram os resultados em percentagem das respostas dadas para cada nível de concordância.

Os principais pontos que este inquérito procurou determinar foram:

#	**Pergunta**
1	O tempo de resposta das submissões de solução é aceitável.
2	Este jogo é um bom complemento à aprendizagem de programação.
3	O jogo é intuitivo e divertido suficiente para a motivar a aprendizagem.
4	O número e tipo de objectos com que é possível interagir são suficientes.
5	As interfaces são adequadas e fáceis de usar.
6	Efeitos sonoros ajudariam a melhorar a experiência.
7	A dificuldade dos exercícios na demonstração é adequada para quem está a aprender.
8	Efeitos visuais como sombras fazem falta no jogo (luzes deixariam de passar através dos objectos/paredes).
9	Os resultados errados das submissões deveriam ter mais informação para além do tipo de resultado.
10	Fazer a a submissão de soluções devia ser possível apenas através do terminal a qual o exercício está associado, ao invés de poder também usar o menu de quests de qualquer parte do jogo.

Tabela 5.1: Perguntas do inquérito

- Se o tempo de espera, desde o momento que se submete uma solução até receber o resultado, é aceitável ou demasiado longo;
- Se esse resultado seria mais útil se tivesse informação adicional sobre erros ou se deve fazer parte do desafio do jogo determinar qual o erro;
- Se o jogo é interessante o suficiente para motivar alunos a aprender programação;
- Se a adição de mais objectos com o quais seja possível interagir e efeitos sonoros, expandindo a mecânica de jogo, poderia contribuir para o melhoramento da plataforma;
- Se a dificuldade dos exercícios/missões presentes no protótipo é acessível e adequada.

5.3 Resultados do Inquérito

Dos resultados obtidos através deste inquérito, pode-se concluir que o tempo de espera entre a submissão e o retorno de resultado é aceitável. O resultado retornado em si, para alguns é suficiente, no entanto outros acharam que seria útil haver mais informação relativamente ao erro.

A grande maioria achou que esta plataforma pode ser um bom complemento à aprendizagem da programação, suficientemente divertido para cativar o interesse dos aluno e

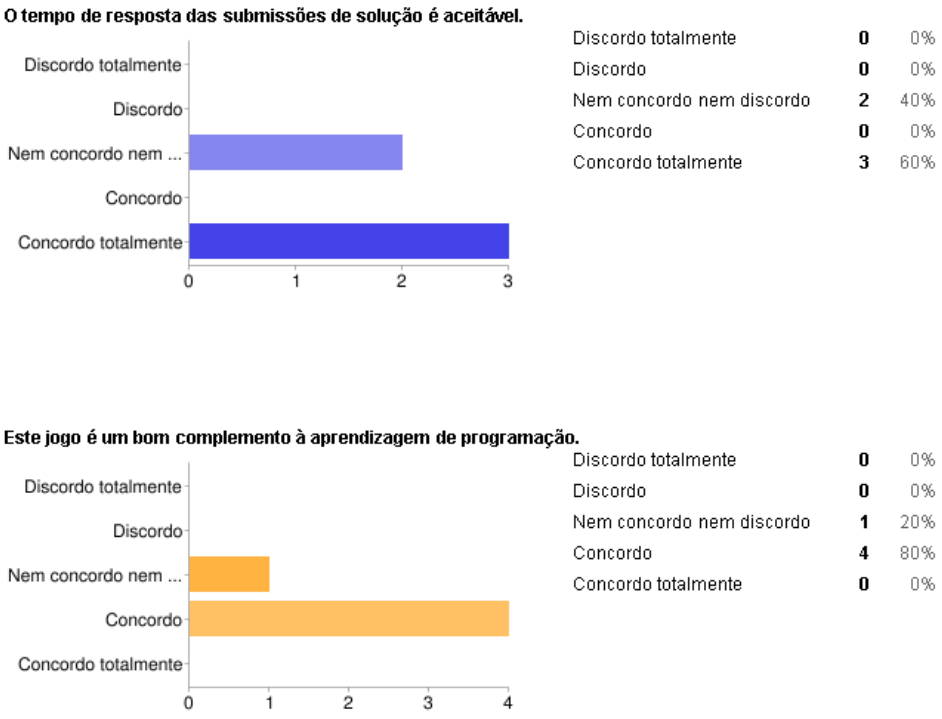

Figura 5.4: Resultados das perguntas do inquérito 1 e 2

cujas interfaces são intuitivas e fáceis de usar. A adição de efeitos sonoros, melhoramento do design dos mapas e de mais tipos de objectos com os quais seja possível interagir deverão ser aspectos a melhorar.

Relativamente à dificuldade dos exercícios propostos no protótipo, os resultados demonstraram ser adequados para quem está a aprender a programar.

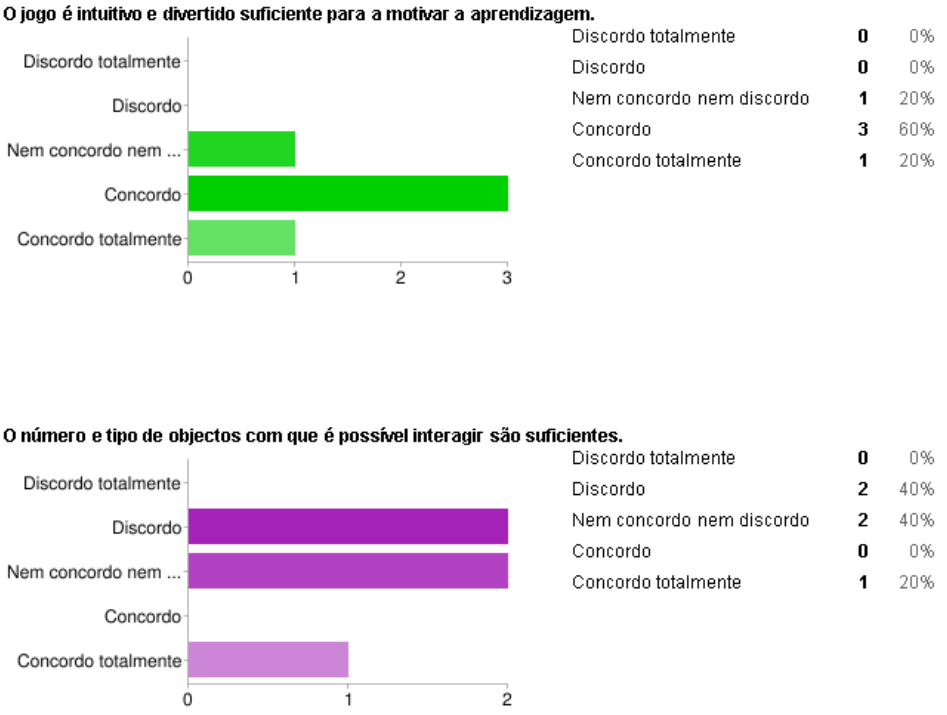

Figura 5.5: Resultados das perguntas do inquérito 3 e 4

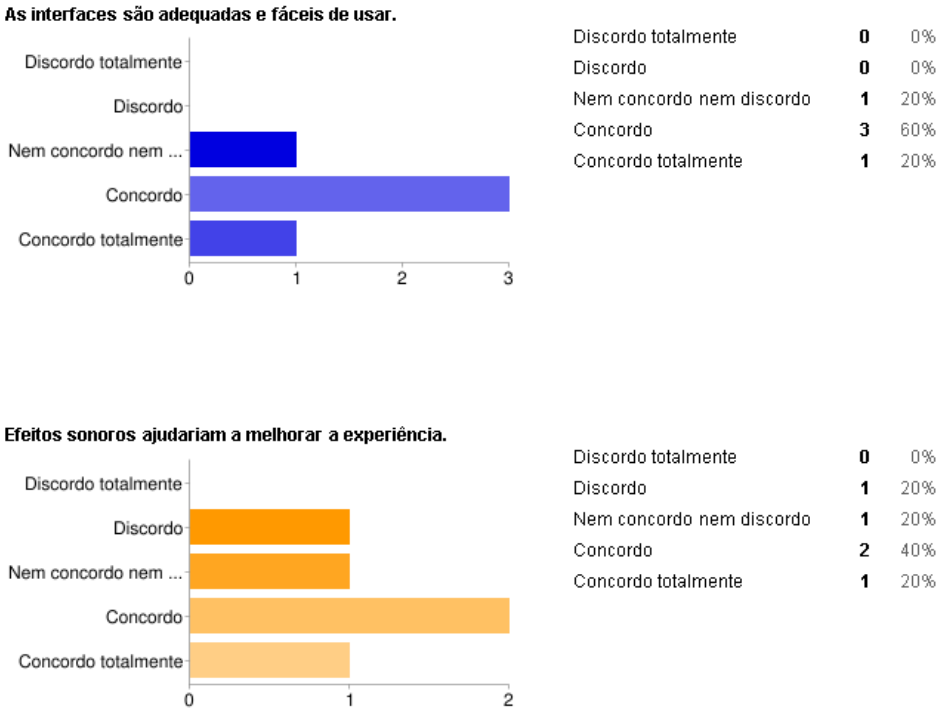

Figura 5.6: Resultados das perguntas do inquérito 5 e 6

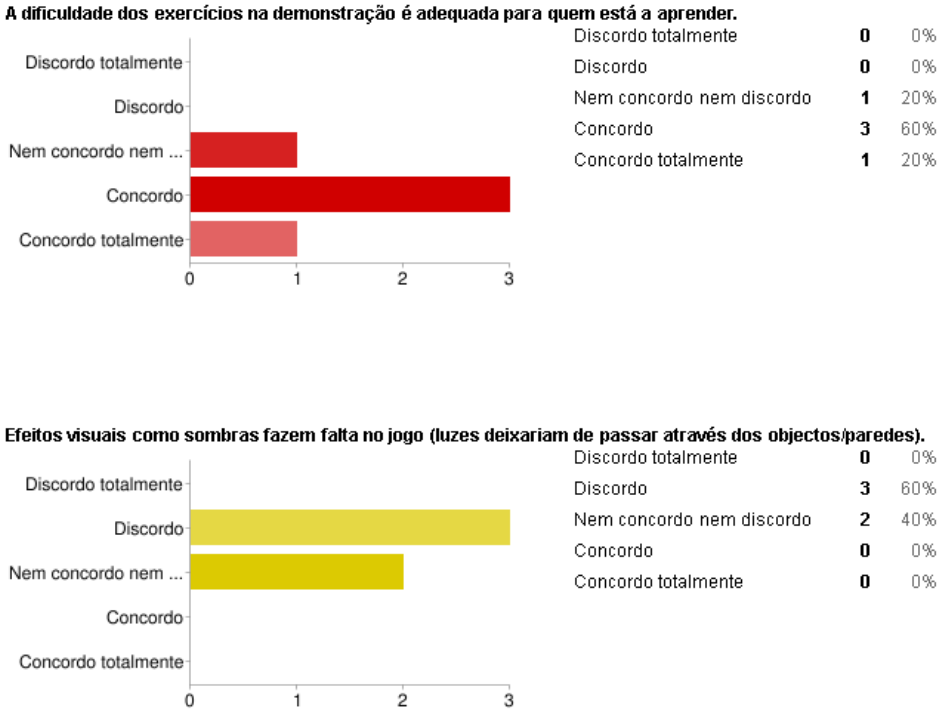

Figura 5.7: Resultados das perguntas do inquérito 7 e 8

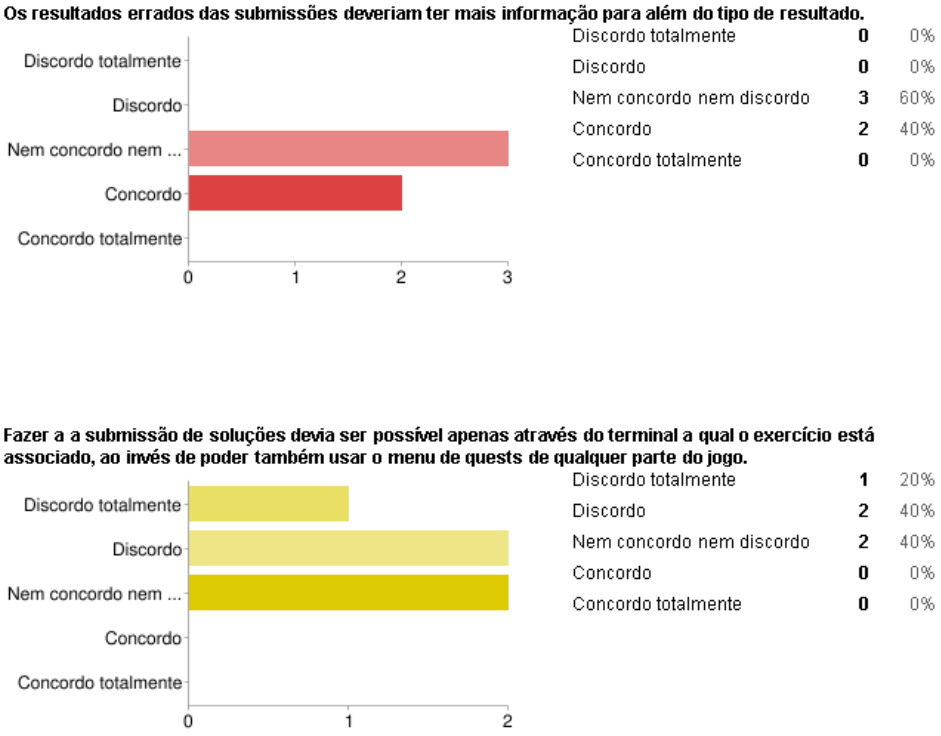

Figura 5.8: Resultados das perguntas do inquérito 9 e 10

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Conclusões

A plataforma cujo desenvolvimento teve início nesta dissertação poderá dar suporte ao desenvolvimento de jogos sérios para a aprendizagem dos alunos dos fundamentos da programação. Com isso em mente vários passos foram dados:

- Começou-se por estudar o que são jogos sérios e de que forma estes estão inseridos na nossa sociedade. Foram analisados os vários tipos de jogos sérios existentes e alguns projectos relativamente relacionados com a programação. Foram também analisados alguns motores de jogos que poderiam assistir no desenvolvimento. Através deste estudo a tarefa de idealizar uma mecânica de jogo simplificou-se e os objectivos ficaram melhor delineados.

- A idealização da mecânica de jogo e especificação da plataforma foram o passo seguinte. Todos os aspectos da plataforma foram definidos, isto é, tipo de jogo, interacção dos utilizadores com a plataforma, comunicação entre componentes e ferramentas a serem usadas.

- Com a mecânica e ferramentas a usar definidas, seguiu-se a implementação do sistema projectado. Foram desenvolvidas as componentes serviço web e o jogo e a integração do sistema de avaliação automático DOM Judge.

- Finalmente foi gerado um protótipo, bem como um editor de mapas para permitir criar os mapas a serem usados. Seguidamente o protótipo foi testado por vários alunos e um inquérito preenchido para ajudar a ter uma melhor ideia da opinião geral sobre este e do que foi feito com sucesso e do que é preciso melhorar.

Apesar de todo o esforço dedicado a este projecto, não foi possível implementar a página web, deixando assim a plataforma incompleta. No entanto todo este processo foi apenas o início, a criação dos fundamentos e alicerces que poderão vir a sustentar uma plataforma robusta de aprendizagem dos fundamentos da programação e até mesmo ser estendida para aprendizagens mais avançadas e complexas.

A mecânica de jogo é bastante rudimentar neste ponto, no entanto as bases estão estabelecidas. A plataforma em si tem potencial para oferecer um jogo sério rico, divertido, original e, fundamentalmente, capaz de transmitir conhecimentos.

6.2 Trabalho Futuro

Ao nível da implementação da plataforma, o desenvolvimento da página web de controlo e interface com o utilizador é fundamental para que, com a plataforma completa, se possam realizar mais testes e um novo inquérito, se possível num ambiente de ensino composto por um grupo de alunos maior preferencialmente do primeiro ano.

Com os testes efectuados e o novo inquérito preenchido, o passo seguinte será melhorar aspectos que os resultados dos testes evidenciem como negativos e enriquecer a experiência que o jogo proporciona, expandindo a mecânica de jogo, os objectos com os quais o jogador pode interagir (fruto dos novos conceitos estabelecidos na mecânica), uma maior qualidade gráfica a nível de menus, modelos e texturas. O som é também um elemento fundamental em qualquer jogo e deverá ser também equacionado no desenvolvimento futuro.

Relativamente aos exercícios de programação, poderá também vir a ser útil expandir as funcionalidades do serviço web para substituir o DOMJudge, no sentido de optimizar o tempo de resposta em uso mais intensivo.

Referências

[Aca] DEI Academy. http://academy.dei.uc.pt/.

[AJ] Gomes Anabela e Mendes A. J. Learning to program - difficulties and solutions.

[BDS] Moskal Barb, Lurie Deborah e Cooper Stephen. Evaluating the effectiveness of a new instructional approach.

[EL] Anderson Eike e McLoughlin Leigh. Critters in the classroom: A 3d computer-game-like tool for teaching programming to computer animation students.

[Fab07] Carlo Fabricatore. Gameplay and game mechanics design: A key to quality in videogames. 2007. http://www.oecd.org/dataoecd/44/17/39414829.pdf.

[fDR] International Strategy for Disaster Reduction. A disaster simulation game from the un/isdr. http://www.stopdisastersgame.org/.

[Fer] Robert C. Ferguson. Teacher's guide: Research and devefits of chess. http://www.quadcitychess.com/benefits_of_chess.html.

[Fra] Gonzalo Frasca. Playing with fire: Serious hype, serious opportunities. http://seriousgamessource.com/features/feature_092906_hype.php.

[KA] Bierre Kevin e Phelps Andrew. The use of muppets in an introductory java programming course.

[MR] Micro-Rato. http://microrato.ua.pt/.

[MT] Eagle Michae e Barnes Tiffany. Experimental evaluation of an educational game for improved learning in introductory computing.

[Pac10] Pedro Pacheco. Computer-based assessment system for e-learning applied to programming education. 2010.

[PDdFP10] Panagiotis Petridis, Ian Dunwell, Sara de Freitas e David Panzoli. An engine selection methodology for high fidelity serious games. *Second International Conference on Games and Virtual Worlds for Serious Applications*, 2010.

[Plaa] PlayGen. Cook 'em up, serious games development process game. http://playgen.com/sg2-viral/.

[Plab] PlayGen. Playgen. http://playgen.com/company_index/.

[Rob] Robocode. http://robocode.sourceforge.net/docs/ReadMe.html#what-is-robocode.

[SL] Richard Smithies e Sion Lenton. Serious games – a developer's perspective. http://www.defencemanagement.com/article.asp?id=211&content_name=Education+and+Training&article=5515.

[Ton] Jenkin Tony. The motivation of students of programming. http://www.psy.gla.ac.uk/~steve/localed/jenkins.html.

[WJ03] Charlotte Wiberg e Kalle Jegers. Satisfaction and learnability in edutainment: A usability study of the knowledge game 'laser challenge' at the nobel e-museum, 2003. http://www8.informatik.umu.se/~colsson/cwkjhci03.pdf.

Anexo A

Fluxograma de jogo e diagrama de comunicação

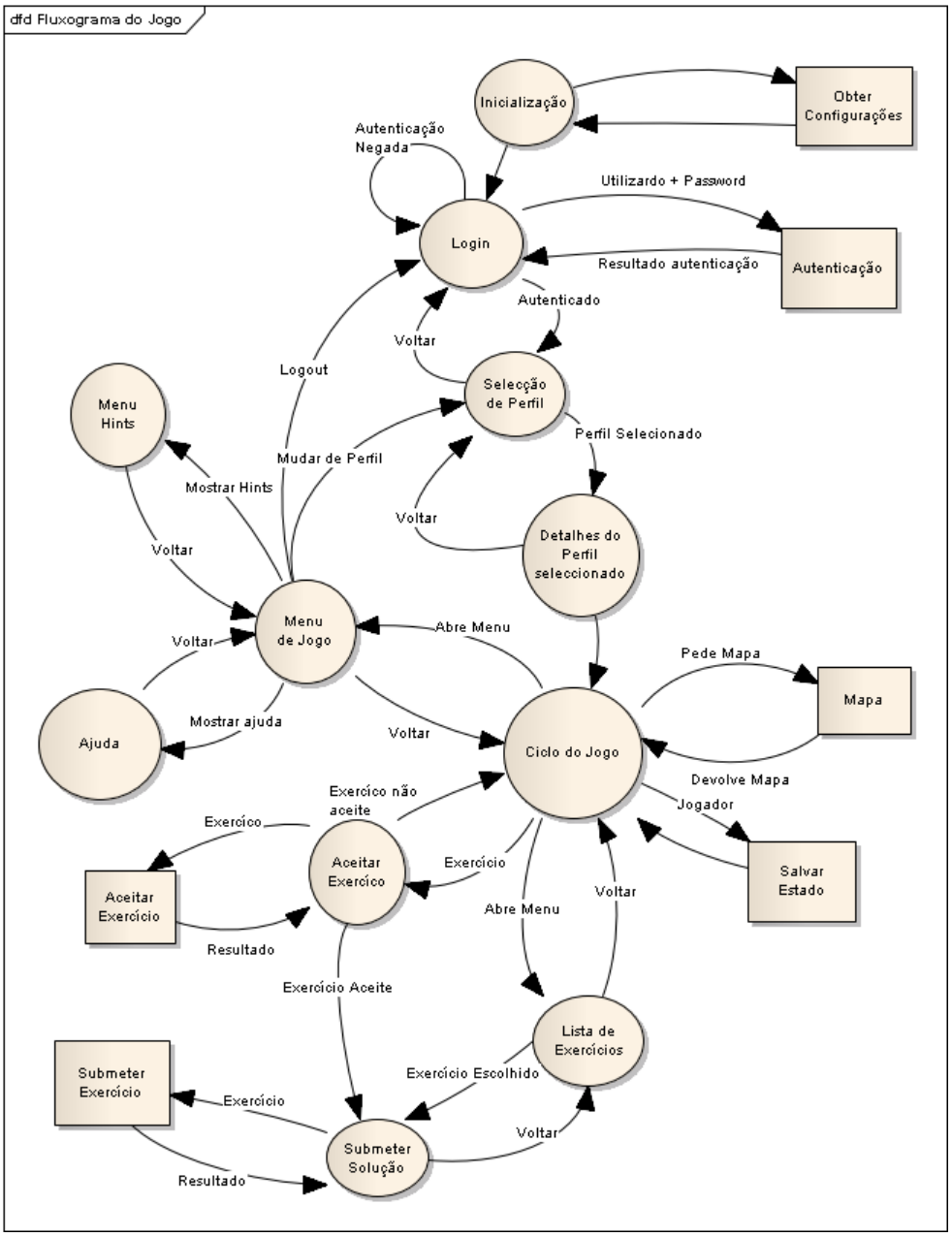

Figura A.1: Fluxograma do jogo

Função	**Descrição**
Obter configurações	Obtém configurações como o nome do jogo, imagens e texto das introduções, texturas usadas pelos mapas (as imagens são imediatamente carregadas para o cliente)
Autenticação	Recebe informação do utilizador e se for válida, retorna os perfis do utilizador. É também criada uma nova sessão (ou renovada caso o utilizador já tivesse uma sessão activa anterior).
Obter Exercícios	Recebe o identificador do perfil escolhido e retorna todos os exercícios aceites resolvidos e ainda por resolver, e define esse perfil por defeito na sessão - a partir deste momento todas as operações seguinte passam a usar este perfil
Obter Mapa	Recebe o identificador do mapa e retorna a informação do mapa que permitirá construir o mundo virtual 3D, bem como lista de objectos presentes nesse mapa e exercícios associados a estes
Aceitar exercício	Recebe o identificador do exercício e adiciona-o à lista de exercícios aceites e é inserida uma entrada na tabela profileQuest
Submeter código fonte	Recebe o identificador do exercício e o código fonte e submete-o para análise, retornando um valor booleano que representa o sucesso da operação. Se o exercício for aceite, além de adicionar uma nova entrada na base de dados do DOM Judge, é também adicionada na base de dados uma entrada na tabela de submissões
Obter estado do código	Não recebe argumentos de entrada, no entanto verifica o estado da análise da operação e retorna esse estado. O serviço web incrementa o número de tentativas quer a solução esteja correcta ou não e caso esteja correcta, actualiza o estado da quest e dos objectos que activa ou desactiva. O mesmo procedimento acontece no jogo
Salvar estado do jogador	Recebe o estado actual do jogador (posição) e actualiza o perfil actual
Reinicializar serviço web	Reinicializa as definições do serviço web e cache estática de exercícios e objectos
Salvar estado do jogador	Permite colocar o serviço em modo edição, impedindo novas ligações e cancelando as existentes

Tabela A.1: Descrição actualizada das funções remotas do serviço web

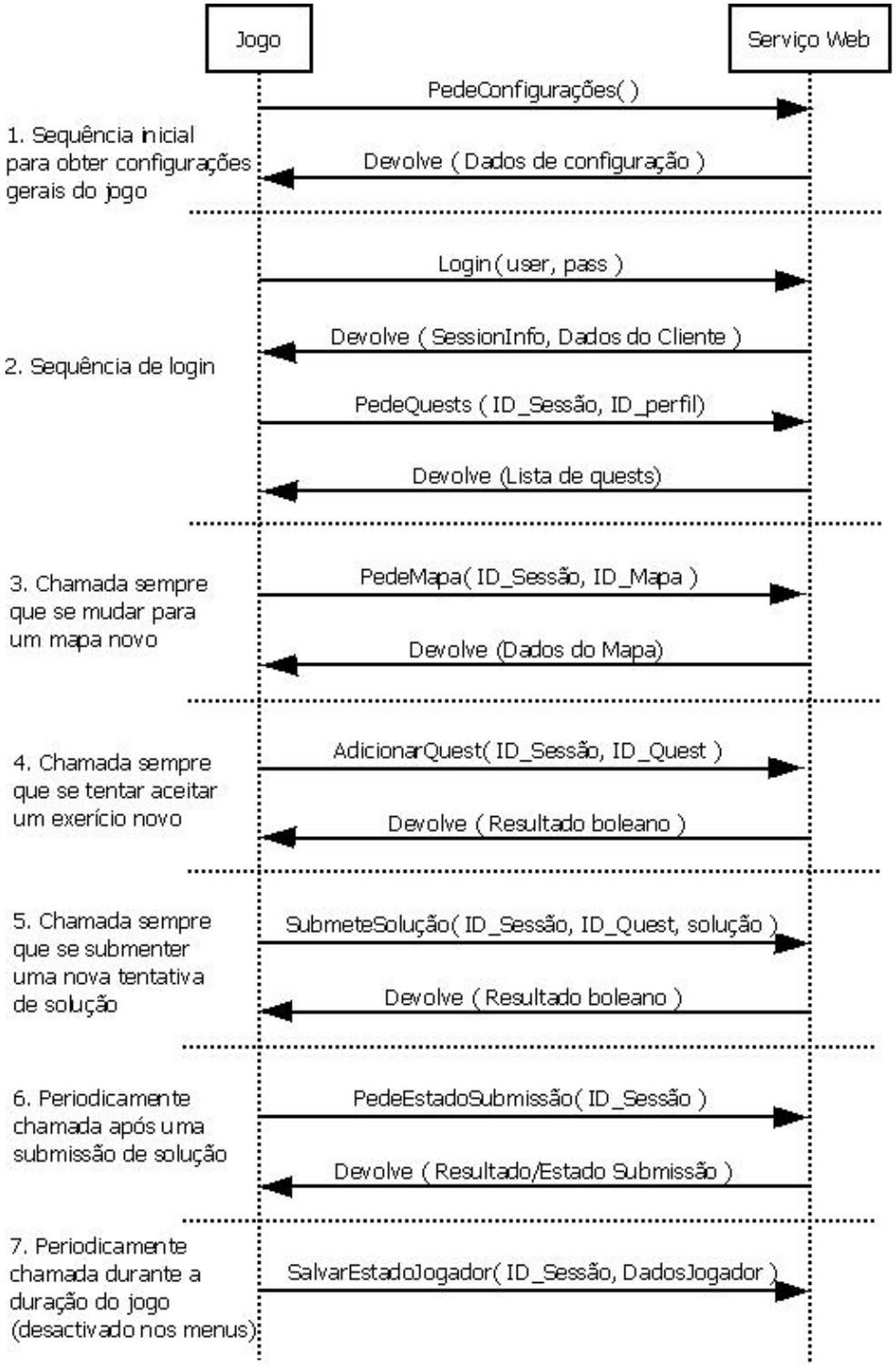

Figura A.2: Diagrama de comunicação entre o Jogo e o Serviço Web

Anexo B

Diagramas UML de dados

Classe	Descrição
ServiceHandler	Esta classe é do tipo singleton e é uma espécie de super classe responsável por gerir todos os aspectos do jogo
SessionInfo	A sessão de jogo é guardada aqui e serve primariamente para evitar que um utilizador tenha duas instâncias do jogo em execução
PlayerInfo	Contém a informação referente ao utilizador da sessão
ProfileItem	A informação relativa a um perfil do utilizador é guardada nesta classe
RoomInfo	Os dados que constituem um mapa são guardados aqui
ObjectInfo	Contém os dados relativos a um objecto
QuestInfo	Guarda a informação de um exercício
Localization	Guarda um grupo de coordenadas espaciais
Submission	Guarda a informação da ultima submissão, ainda sem resultado, de um utilizador
DataBase	Classe abstracta que define funções de comunicação com a base de dados
DBGame	Estende a funcionalidade da classe DataBase para interagir directamente com a base de dados do jogo
DBDomJudge	Estende a funcionalidade da classe DataBase para interagir directamente com a base de dados do DOMJudge

Tabela B.1: Descrição das principais classes presentes no diagrama de classes do serviço web

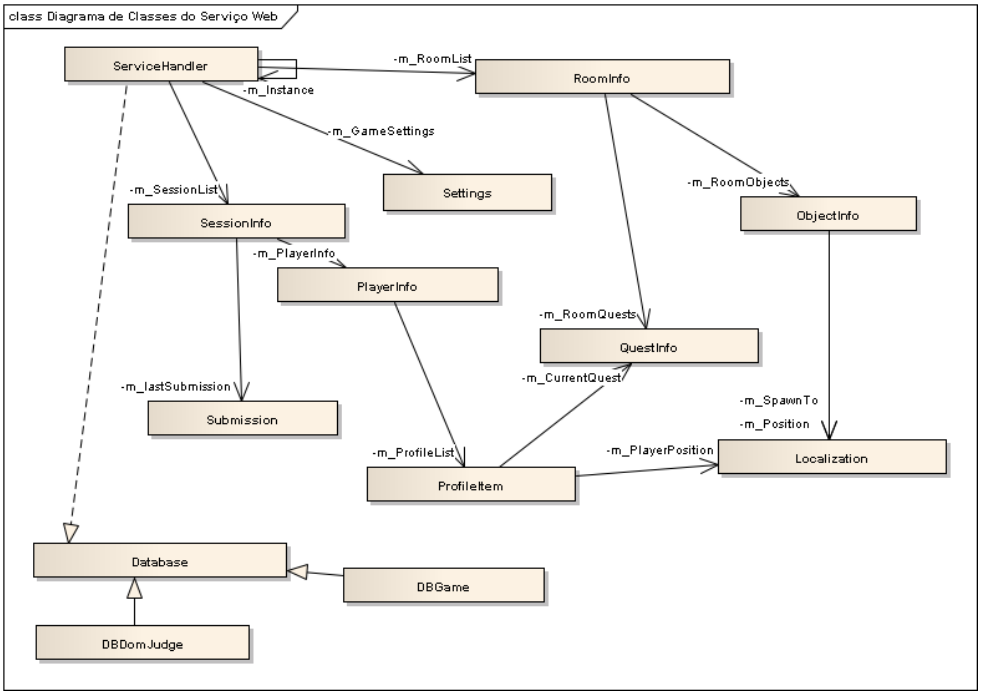

Figura B.1: Diagrama UML da estrutura de dados usada no serviço web

Classe	**Descrição**
GameHandler	Classe do tipo singleton responsável por guardar todas as informações importantes ao jogo e de gerir a lógica local de jogo
SessionInfo	Guarda a sessão estabelecida com o serviço web
PlayerInfo	Contém a informação referente ao utilizador da sessão
ProfileItem	A informação relativa a um perfil do utilizador é guardada nesta classe
RoomInfo	Os dados que constituem um mapa são guardados aqui
ObjectInfo	Contém os dados relativos a um objecto
QuestInfo	Guarda a informação de um exercício
WebServiceConnecter	Gere a comunicação com o serviço web
MenuGUI	Gere todos os menus do jogo
MenuInfo	Guarda informação referente ao menu activo
RoomGameObject	Gere a construção do mundo virtual
GameGUI	Gere a interface do jogo
InteractiveObject	Classe abstracta da qual derivam todos os objectos com os quais o jogador pode interagir
Textures	Guarda as texturas usadas nos mapas

Tabela B.2: Descrição das principais classes presentes no diagrama de classes do jogo

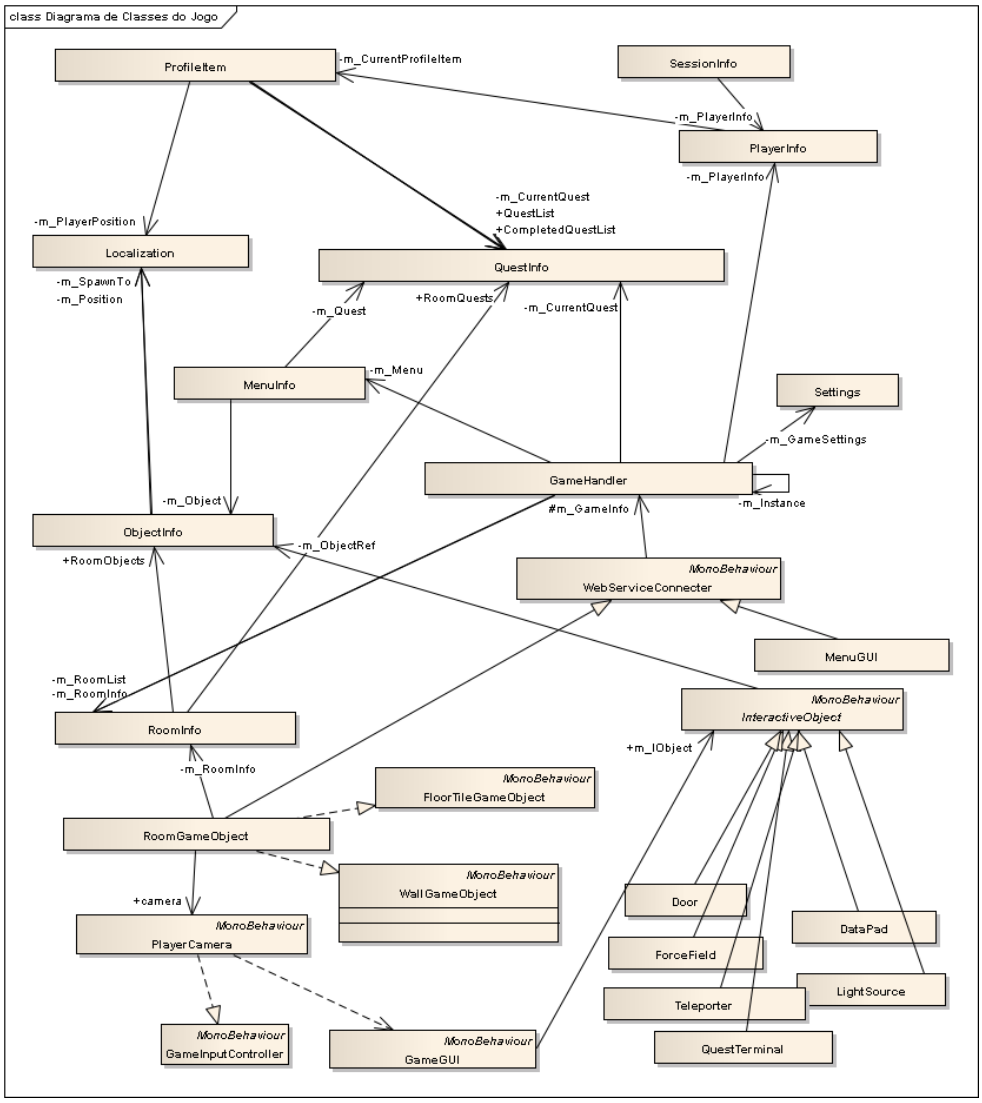

Figura B.2: Diagrama UML da estrutura de dados usada no jogo