José Maria de Brito Tavares Soares Franco

Licenciado em Engenharia Electrotécnica e de Computadores

Desenvolvimento de um *serious game* para a aprendizagem da leitura

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador: Yves Rybarczyk, Professor Doutor, FCT
Co-orientador: Tiago Cardoso, Professor Doutor, FCT

Júri:

Presidente:	Professora Doutora Anabela Monteiro Gonçalves Pronto FCT / UNL
Arguente:	Professor Doutor José António Barata de Oliveira – FCT / UNL
Vogal:	Professor Doutor Tiago Oliveira Machado de Figueiredo Cardoso – FCT / UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2015

Desenvolvimento de um *serious game* para a aprendizagem da leitura

Apraz-me efectuar este registo, dedicando esta dissertação a quem foi extremamente importante para mim, influenciando a minha vida através da criação de uma visão das coisas, que me permitiu valorizar a essência e a necessidade do estudo e que culmina com este trabalho, fechando-se um ciclo e abrindo-se outro.

Ao meu Avô Materno, André Manuel de Castro Lobo Pimentel de Brito Tavares, pela sua postura na vida e que me serviu sempre de exemplo na forma de estar e agir perante as vicissitudes com que me tenho deparado.

Ao meu Avô Paterno, José Fernandes Soares Franco, que me transmitiu a atitude positiva, que tão mais fácil é para viver bem, e que me ajudou a chegar até este ponto.

À minha Avó Paterna, Maria Rita Bráz Fernandes Reis Soares Franco, a quem devo muito do que sou e, provavelmente, do que não sou.

A todos estes queridos Avós a minha dedicatória deste trabalho e o meu muito obrigado por terem contribuído para ser quem sou.

Agradecimentos

Gostaria de agradecer ao meu orientador, Professor Yves Rybarczyk, e ao co-orientador, Professor Tiago Cardoso, por me terem dado a oportunidade de desenvolver esta dissertação, pela motivação e apoio ao longo das várias fases de desenvolvimento.

À Universidade Nova de Lisboa e à Faculdade de Ciências e Tecnologia em especial por me terem acolhido e permitido aprofundar os meus conhecimentos, disponibilizando os seus recursos para me prepararem nesta nova etapa. Ao corpo docente pelo apoio prestado e a todos os meus colegas que se cruzaram no meu caminho, em especial ao José Simão pela sua perseverança, ajuda e companheirismo. Ao centro Diferenças pela disponibilidade e interesse nesta dissertação, bem como todo o tempo dispensado na mesma. À Doutora Fátima Trindade e à Doutora Sandra Silva, pela ideia desenvolvida e todo o apoio demonstrado ao longo do desenvolvimento, permitindo que presenciasse o trabalho árduo efectuado no centro Diferenças, e ao arquitecto Luís Castro, por toda a componente gráfica desenvolvida para o jogo. Um sincero agradecimento a todos.

Ao meu pai, pelo seu constante desafio às minhas capacidades culminando na progressão dos meus estudos, por ter acreditado em mim e pelo seu apoio incondicional. À minha mãe, que nunca duvidou das minhas capacidades, que sempre me apoiou e me motivou em qualquer altura da minha vida, não só como mãe mas também como amiga. Aos dois, pelos princípios que me transmitiram. À minha família, que sempre me apoiou ao longo desta dissertação bem como da minha vida. Ao Gonçalo Costa, por me ter apoiado e ajudado ao longo do meu curso e por me ter incentivado à conclusão desta dissertação. À Catarina Elias de Sousa, pelo seu companheirismo e motivação. Ao Ricardo Mexia, por se ter transformado num amigo importante mantendo a sua maneira de ser e me ter mostrado um caminho diferente do meu, permitindo que pudesse aprender. Um muito obrigado sincero a todos os meus amigos que me acompanharam ao longo da minha vida.

Resumo

A introdução de videojogos como ferramenta auxiliar do processo de educação tem vindo a crescer lado a lado com a evolução tecnológica. O aumento da capacidade dos processadores e das placas gráficas permitiram um aumento na complexidade dos videojogos e, por consequência, aumentaram as experiências fornecidas pelos mesmos. Sob o prisma da educação, o uso de um videojogo não foi a primeira ferramenta utilizada. O ensino por correspondência foi o primeiro auxílio ao método de ensino, iniciando assim o primeiro conceito de *e-learning*. Este permite enriquecer quem procura mais conhecimento, quem não consegue ter acesso ao mesmo ou quem tem dificuldades no ensino tradicional.

A dificuldade presente em crianças com problemas do neurodesenvolvimento na aprendizagem da língua portuguesa é um problema na sociedade actual e para o combater são usados vários métodos manuais, sem recurso a tecnologia, passando normalmente por exercícios manuscritos e de observação. Desta forma, as pessoas que acompanham as crianças são obrigadas a estar presentes para recolher dados, em vez de se concentrarem nas outras actividades educativas. Este facto resulta numa possível falta de interesse por parte da criança no exercício apresentado, implicando uma perda de produtividade.

A introdução de tecnologias ao serviço de causas sociais é crucial, pois permite um melhor acompanhamento das crianças, auxiliando tanto as pessoas que precisam do acompanhamento como aquelas que as acompanham.

Por exemplo, um sistema automático que apresente os exercícios manuscritos num ecrã e, ao mesmo tempo, guarde os dados referentes ao seu uso seria útil para as pessoas que se encarregam de ajudar as crianças na aprendizagem da Língua Portuguesa.

Esta dissertação insere-se num projecto desenvolvido pelo DIFERENÇAS – Centro de Desenvolvimento Infantil, denominado por “No Reino dos Fonemas”. Este projecto baseia-se em apresentar a crianças diversas imagens com objectivos diferentes de forma a cobrir as cinco vogais e todas as consoantes, no âmbito da aprendizagem da Língua Portuguesa. Neste contexto, as crianças podem-se interessar por um videojogo e aprender ao mesmo tempo, enquanto as pessoas que as acompanham, através dos dados do videojogo, podem focar-se mais nas dificuldades apresentadas por cada criança. Desta forma, é possível uma melhor organização dos dados de cada criança e, por conseguinte, um melhor acompanhamento das suas dificuldades.

Palavras-chave: Unity®, Motor de Jogo, Jogo, E-Learning, Vogal, Jogador

Abstract

The introduction of video games as a tool on the educational process has been growing side by side with the advance if technology. The increase of processors and graphic cards capabilities have unlocked a new level of complexity of a video game and enabled the growing experience retainers by video games. In the educational area, the use of video games were not the first took used, the correspondence method was used at first on helping the education system to develop itself, initiating the e-learning concept. E-learning allows those who search for more knowledge, ones that cannot access other types of information or people that have difficulties on the traditional education to have another way to enrich their possibilities.

Nowadays there are some children with neurodevelopment problems, who have difficulties on learning the Portuguese language. In order to address this problems, several manual methods are used without any type of technology. These methods are based on written exercises and observation. With the above methods, the person in charged has to be always present taking notes from the exercises done. This procedure increases the chances of the child to lose focus and interest from the exercise, which implies the lost of productivity.

The introduction of technologies on social causes have been crucial as it allows a better monitoring of the hole process, helping both, the person in charged and the one that has difficulties.

An automatic system which can show on a screen the previous written exercises and saves the data, would be helpful to people who work with this type of the Portuguese language learning processes and the helped children as well.

This dissertation was part of a project developed by DIFERENÇAS - Centro de Desenvolvimento Infantil, named by "No Reino dos Fonemas". This project is based on showing several images to children with different objectives in order to cover all the five vowels and all the consonants present on the Portuguese language. On this context, children can learn at the same time as they keep interested in the video game. On the other hand, by using this video game, the person in charged can focus on the difficulties shown by the child. With this project is possible to better organise data from each children and keep focus on the children difficulties themselves.

Keywords: Unity®, Game Engine, Game, E-Learning, Vowel, Player

Índice

Lista de Figuras

Lista de Tabelas

Lista de Símbolos e Acrónimos

2D Duas Dimensões

3D Três Dimensões

NPC *Non Playable Character*

FPS *Firs-Person Shooter*

XML *eXtensible Markup Language*

UML *Unified Modeling Language*

OS X *Apple's Operating System*

IOS *Iphone Operating System*

MMORPG *Massive Multiplayer Online Role-Playing Game*

RPG *Role-Playing Game*

1

Introdução

As causas sociais têm uma importância extrema em qualquer sociedade. A integração de deficientes na sociedade é um problema que não é fácil e estas causas tentam facilitar a integração dos mesmos. Segundo os últimos censos realizados, existem 634 408 deficientes recenseados em 2001 [1] e cada uma destas pessoas pode sofrer de problemas de integração na sociedade.

É através de pessoas dedicadas, psicólogos, sociólogos, terapeutas da fala, etc., que existe ajuda neste processo. Desde que a integração na sociedade se considerou um problema e se começou a endereçá-lo, foi desenvolvido um conjunto de métodos e ferramentas manuais, maioritariamente no seio da organização onde trabalham ou por conta própria. Para crianças com perturbações no neurodesenvolvimento esta integração é particularmente difícil por não conseguirem manter a atenção durante muito tempo, por demorarem mais tempo a entender os conteúdos educativos, por ser necessário um acompanhamento continuado e por não o conseguirem obter, entre outros. Para as pessoas que se deparam com este problema existem algumas dificuldades, principalmente por não existirem sistemas de informação aplicados à educação que visem acompanhar esta integração. Estes sistemas precisam de um desenvolvimento acompanhado por quem define os planos da integração de forma a produzir um objectivo desejável.

O acesso a sistemas de informação aplicados à educação é difícil devido a um défice no mercado. Para se desenvolver este tipo de sistemas há duas hipóteses, pagar o seu desenvolvimento, ou proceder o seu desenvolvimento. Existem várias empresas com capacidade para o fazer, no entanto é necessário que se remunere este trabalho e nem sempre as organizações de

carácter social podem dispensar os fundos necessários, por este não ser uma prioridade ou por o sucesso na integração, ao utilizar este sistema, não compensar o valor a pagar para o seu desenvolvimento. As causas sociais não costumam integrar nas suas equipas pessoas que detenham o conhecimento técnico para realizar este desenvolvimento e, por esta razão, não conseguem desenvolver nenhum sistema que apoie qualquer área social em que intervenham.

O uso de tecnologia na aprendizagem é uma prática que tem vindo a evoluir, sendo este chamado de *e-learning*. Trata-se de uma ferramenta ou sistema educacional com base num computador ou dispositivo digital que permita a aprendizagem em qualquer local e hora. Hoje em dia está maioritariamente disponível através da internet, apesar de, no passado, ser disponibilizado através de métodos como o CR-ROM.

O *e-learning* oferece a possibilidade de partilhar material em vários formatos, tais como vídeos, documentos *word*, PDF e apresentações. Permite também seguir *webinars*[1] e comunicar com os professores via *chat* e mensagens em fóruns.

Existem diversas tecnologias que o *e-learning* usa, algumas foram especificamente desenvolvidas para o efeito, enquanto outras complementam convenientemente o processo de aprendizagem. Um exemplo dessas tecnologias são os videojogos.

Epignosis [2] refere que o uso de bases de dados e de CMS[2] estão associados ao *e-learning*, pois estes trabalham lado a lado de forma a gravar os resultados, o conteúdo do curso e os registos dos estudantes. Os dados são gravados na base de dados e o CMS disponibiliza uma interface para se adicionar, actualizar ou apagar dados. Um bom LMS[3] normalmente disponibiliza ferramentas para gerar e guardar relatórios do progresso.

O desenvolvimento de um videojogo para *e-learning* pode ser feito de duas formas: *gamification* e *Game Based Learning* ou GBL. Estes dois tipos de desenvolvimentos são similares mas têm objectivos diferentes. Enquanto o GBL pode ser usado como uma ferramenta para auxiliar um processo de aprendizagem, o *gamification* é o processo na íntegra, ou seja, não tem o intuito de auxiliar, tem o intuito de ser o processo completo de aprendizagem.

Um jogo é composto por vários módulos e estes são desenvolvidos separadamente, sendo que cada módulo pode ser desenvolvido por uma pessoa ou uma equipa. A figura 1.1. representa alguns dos módulos principais de um jogo.

No passado, o desenvolvimento de um jogo passava por constituir uma equipa de pessoas, em que cada uma era responsável por um módulo. Neste prisma, a equipa necessitava

[1] Aulas *online* ao vivo.

[2] Content Management System.

[3] Learning Management System.

de, pelo menos, cinco elementos: *designer* do jogo, artista gráfico, artista de som, programador e produtor.

O *designer* do jogo é responsável por definir a interacção entre os vários ecrãs do jogo, bem como a sua consistência, se esta for forte, o jogador interessa-se mais pelo jogo e este não entra em contradição. A história também é definida pelo *designer*.

O artista gráfico tem a responsabilidade de criar mapas, personagens, NPC e todo o tipo de objectos que podem ou não interagir com o jogador. Cardoso [3] afirma que é nele que recai a apresentação do jogo e, normalmente, está em contacto constante com o programador.

O artista de som possui uma das tarefas mais importantes. A decisão do som num jogo permite ao jogador imergir no mesmo e criar a sensação de que a realidade do jogador passou para o jogo, fazendo com que toda a atenção esteja nos acontecimentos do jogo. Da mesma forma como um filme com uma boa banda sonora nos permite abstrair do que nos rodeia, o mesmo se passa num jogo.

Figura 1.1: Módulos de um jogo [4]

O programador está encarregue de criar toda a mecânica do jogo. É ele que define as leis físicas presentes no jogo, sejam elas iguais à realidade ou não, bem como todas as acções que provocam alguma alteração ao mapa ou jogador. Este está sempre em contacto com o artista de som, o artista gráfico e o *designer* de forma a não se perder a consistência do jogo.

Por último, e mais importante, está o produtor. Este gere a equipa de forma a poder comprometer-se com datas e a estas serem cumpridas. É responsável por manter a equipa concentrada e motivada ao longo do desenvolvimento do jogo e também por preencher alguma lacuna que possa ser criada com algum membro da equipa [4].

Outro conceito de desenvolver um jogo é o *indie*. O *indie developer* não está associado a nenhuma distribuidora e para ele não é relevante o facto de o jogo estar aprazível comercialmente. Apenas se preocupa com a opinião da comunidade que tenha experimentado o jogo, e estes jogos costumam estar durante algum tempo em versão *beta*[4]. Estes *developers* podem ser pessoas individuais ou equipas, auxiliando-se em plataformas para levar o seu jogo ao consumidor.

Existem algumas plataformas que ajudam os *indie developers* a distribuir o seu jogo e cobrando habitualmente apenas direitos sobre o uso da mesma, percentualmente ao número de vendas efectuadas. A Steam, a itch.io, a GOG.com, a Green Man Gaming e a Humble são algumas plataformas que apoiam os *indie developers* e categorizam os seus jogos de *indie games*.

Alguns destes jogos são também distribuídos sob a forma de ajuda à caridade. A plataforma Humble costuma organizar vendas de conjuntos de jogos em que cada pessoa paga o que entende ser justo, com um mínimo de 1 dólar, e possui alguns patamares para desbloquear mais jogos. O comprador também pode decidir como quer o seu dinheiro distribuído, ou seja, qual a percentagem do que gastou que vai para caridade e quanto vai para os criadores dos jogos. Esta divisão pode ser total, isto significa que não existe um mínimo que tenha de ser para caridade ou para criadores dos jogos, como é explicado no *website* da Humble [5].

1.1 Objectivos

Esta dissertação tem como objectivos:

- Criar um videojogo que apresente exercícios de apoio na aprendizagem da leitura;
- Criar um videojogo que grave os dados de utilização dos jogadores;
- Desenvolver um sistema que se consiga executar em computadores com dez anos ou menos.

[4] Versão que não está pronta para venda, que contém alguns erros ou que não contém todos os requisitos propostos.

1.2 Motivação

Uma ferramenta que automaticamente apresente exercícios para ajudar a aprendizagem da leitura e recolha dados para crianças com perturbações do neurodesenvolvimento é uma ferramenta que pode ajudar instituições como o DIFERENÇAS – Centro de Desenvolvimento Infantil. Estas ferramentas capacitam as pessoas encarregues de acompanhar as crianças de organizar tanto as práticas terapêuticas como os documentos de dados das terapias de forma a aumentar a *performance* destas.

Sob o ponto de vista tecnológico, o desenvolvimento permitiu aprofundar os conhecimentos sobre um motor de jogo, os vários módulos existentes e o seu uso, bem como perceber melhor a capacidade da linguagem de programação C#.

A possibilidade de debater o desenvolvimento com pessoas que não têm conhecimento na área é uma oportunidade que, para além de considerar importante, é enriquecedora a nível profissional. A possibilidade de o conseguir fazer também foi determinante na escolha deste tema.

2

Estado de Arte

Neste capítulo irão ser apresentados os vários programas que se podem usar para desenvolver um videojogo, o seu uso na educação e todos os seus conceitos adjacentes.

São abordadas comparações entre conceitos e as suas definições.

2.1 Videojogos

Desde que os primeiros videojogos foram criados, o jogo “Pong” surgiu por volta de 1974, que foi necessário definir o que é um videojogo. Assim, entende-se que um videojogo é qualquer jogo interactivo que é jogado usando dispositivos electrónicos especializados, tais como um computador ou um dispositivo móvel e uma televisão ou outro ecrã juntamente com os meios para controlar imagens gráficas. Daqui em diante a palavra jogo refere-se a videojogo.

Game Invaders [6] descreve que os jogos são desenhados para captar a atenção dos jogadores, para os introduzir e forçá-los, de forma viciante, a seguir milhares de ciclos de interacção com graus limitados de variação na jogabilidade, entretenimento, desafios, etc. Um jogo é uma sucessão de momentos de intervenção que obrigam o jogador a fazer a próxima intervenção e assim sucessivamente, talvez centenas de vezes em cada minuto. Um óptimo jogo é aquele em que perseguir estes infindáveis objectivos, momentos de intervenção, mais ou menos repetidos, se torna tão compulsivo para o jogador que mais nada interessa. Um jogo é essencialmente uma repetição compulsiva.

Ao serem desenhados para atrair os jogadores podem criar um vício. Esse vício pode provocar uma confusão entre personalidades adoptadas num jogo e a realidade.

Outro problema que os jogos provocam é o aumento de dopamina produzida no corpo humano. O jornal Tech Times [7] dedicou um artigo a um dos aspectos negativos dos jogos e, baseando-se num estudo publicado no jornal Neurology Now, concluindo que jogar excessivamente, especialmente nos jogadores mais jovens, pode ter efeitos negativos. A aceleração de dopamina gerada por jogar jogos é tão grande que pode desligar certas áreas do cérebro, tal como o córtex pré-frontal, uma área que é responsável por controlar e ajudar a medir riscos e recompensas. Em jogadores jovens, cujo córtex pré-frontal não está completamente desenvolvido, pode originar que as necessidades mais básicas, como comer, dormir ou ir à casa-de-banho, sejam ignoradas e, em casos extremos, pode resultar em morte de jogadores que as ignorem durante um longo período de tempo.

Este excesso de dopamina também pode provocar uma avaliação errada de riscos e determinar um conjunto de decisões erradas, como por exemplo na condução. Os simuladores de corridas podem ajudar os jogadores numa situação na vida real, mas também pode acontecer o contrário. Os simuladores não conseguem modular todas as variáveis reais. Assim podem ser confundidas acções que transpondo para a vida real podem originar acidentes. A ausência de vento num simulador pode alterar a capacidade de avaliar situações com condições adversas na vida real e provocar uma decisão prejudicial para o condutor, o mesmo se passa com a resposta de um carro quando entra em aquaplanagem ou derrapa. Muitos dos simuladores exploram a capacidade de pôr um carro a derrapar e, muitas vezes, não descrevem o comportamento real de um carro, provocando a tomada de decisões nesta situação que podem não ser as melhores.

Tendo em atenção a definição de um videojogo e do seu propósito, é possível aplicar estes conceitos à educação, tornando um jogo com características educativas de tal forma viciante que quanto mais tempo estiver a jogar, mais o jogador aprende de forma passiva.

Distinguem-se dois conceitos que abordam a conversão de um conteúdo ou plano educativo num jogo, o *gamification* e o *Game Based Learning* ou GBL.

2.1.1 *Gamification vs Game Based Learning*

Adam e Mohamed [8] afirmam que o *gamification* assenta no conceito de usar mecânicas e *design* de jogos de forma a envolver e motivar digitalmente as pessoas a atingirem os seus objectivos:

- As mecânicas de jogo descrevem o uso de elementos tal como pontos, medalhas e quadros de liderança, que são muito comuns em jogos.
- O *design* descreve o caminho que os jogadores tomam com elementos tais como jogabilidade do jogo, espaço de jogo e a história.

- *Gamification* é um método para envolver digitalmente, em vez de envolver pessoalmente, isto significa que os jogadores interagem com computadores, dispositivos móveis e outros aparelhos electrónicos em vez de interagirem com uma pessoa.
- O objectivo da *gamification* é motivar as pessoas a mudar os seus comportamentos, a desenvolver habilidades ou impulsionar a inovação.
- *Gamification* foca-se em permitir aos jogadores atingir os seus objectivos. Quando os objectivos educacionais ou organizacionais são alinhados com os objectivos do jogo, estes são atingidos como consequência do jogador ter atingido os objectivos do jogo.

Existe um outro conceito semelhante ao *gamification*, o GBL.

O GBL, como Adam e Mohamed o descrevem, assenta no princípio de produzir um resultado específico baseado na aplicação de um jogo a um conteúdo educacional. Isto significa que um jogo não é um sistema pedagógico, mas apenas contém uma característica educativa.

Apesar dos conceitos de *gamification* e GBL serem usados no mesmo contexto, são dois caminhos para transformar situações sociais em experiências do tipo jogo.

Gamification transforma o processo inteiro de aprendizagem num jogo. Utiliza as mecânicas e os elementos de jogo e aplica-os a conteúdos e cursos de aprendizagem já existentes, de forma a motivar e a envolver melhor os alunos. Algumas das características implementadas em *gamification*, que são bastante comuns em jogos, são:

- Medalhas de objectivos concluídos;
- Pontuação;
- Quadros de liderança;
- Barras de progresso;
- Nível e/ou missões.

GBL é usar um jogo como parte de um processo de aprendizagem. O alvo do GBL é ensinar uma capacidade concreta ou um resultado de aprendizagem específico em vez de ser um sistema pedagógico completo [8].

2.2 *E-Learning*

Horachek [9] indica que o *e-learning* se baseia no uso de tecnologia digital de modo a facilitar a aprendizagem. Isto pode incluir servidores na internet e *web browsers*. Pode incluir o uso de vídeos embebidos numa aplicação que permitem ao utilizador rever quando lhe apetecer ou lhe der jeito sob a forma de blocos de bits.

Afirma também que as razões pelas quais a *gamification* funciona em *e-learning* são várias e são suportadas tanto pela pedagogia tradicional como pela neurobiologia, listando algumas das razões mais atractivas:

- **Imersão** – Jogos que são imersíveis para o jogador, activam naturalmente mais percursos de aprendizagem significativos no cérebro. Isto deve-se ao facto de o cérebro guardar e consolidar diferentes tipos de informação em diferentes regiões do mesmo, baseado na sua relevância;
- **Aprendizagem espacial** – Vale a pena atribuir à aprendizagem espacial uma referência especial, apesar da sua semelhança à imersão como modalidade de aprendizagem. Há uma área específica do cérebro que guarda um mapa mental da localização dos objectos/locais que nos rodeiam. Jogos que têm uma componente de navegação espacial, vão naturalmente mobilizar esta parte do cérebro de forma a facilitar a aprendizagem;
- **Aprendizagem activa** – A instrução é passiva e a aprendizagem é activa. Jogar jogos que requerem um nível de pensamento acima da observação passiva é naturalmente mais útil para a aprendizagem e a retenção. Ao usar jogos que têm desafios e *puzzles*, força-se o jogador a participar num nível mais elevado, pensando enquanto controla o objectivo dos resultados de aprendizagem;
- **Afecto emocional** – Jogos que constroem uma ligação emocional nos seus jogadores são mais propícios a um jogo activo e a uma maior atenção por parte dos seus utilizadores. Isto resulta em maior retenção dos objectivos de aprendizagem;
- **Fluxo cognitivo** – Perder a noção do tempo é comum quando se está concentrado na realização de uma tarefa. Os psicólogos chamam a isto estado de fluxo, pois neste estado de ligação o cérebro está a trabalhar ao máximo e o potencial para aprender aumenta;
- **Ambiente seguro** – Jogos de vídeo e simulações em tempo real são bons métodos para treinar porque são inerentemente seguros. O jogador pode praticar uma capacidade dentro do jogo sem existir qualquer risco físico para o corpo. Ao repetir num ambiente virtual, isto permite que o jogador experimente a liberdade de repercussões físicas e encoraja a exploração e a aprendizagem activa.

Estas razões enquadram-se em algumas vantagens que o *e-learning* tem, nomeadamente a inexistência de fronteiras e restrições, o custo associado e a diversão que se obtém.

O facto de, na aprendizagem cara-a-cara, a localização limitar quem se pode deslocar para assistir, no caso do tempo, limita-se a quem tem essa disponibilidade temporal. No *e-learning* esses problemas são resolvidos por não haver um local específico nem uma hora exacta, cada pessoa escolhe o local onde vai assistir e a hora a que o vai fazer.

Ao desenhar um curso de uma maneira mais interactiva e divertida, através do uso de material multimédia ou de métodos de *gamification,* aumenta o envolvimento da pessoa e o tempo de vida relativo do curso.

Epignosis [2] refere que um dos problemas do sistema de aprendizagem convencional é a constante desactualização dos manuais escritos, seja por haver matéria nova, seja por uma reestruturação do programa educacional, pelo que o manual pode já não se enquadrar no programa. Para o *e-learning* esse problema não existe.

O uso de *e-learning* também pode resolver o problema de quem tem vontade de aprender e não tem disponibilidade financeira, maneira de se deslocar, tempo para dispensar ou até mesmo quem não se pode deslocar, por estar fisicamente incapacitado. O acesso a um computador e à internet nos dias de hoje já tem um custo bastante mais reduzido do que antigamente e é atingível pela maioria das pessoas, tornando o *e-learning* uma ferramenta muito útil à aprendizagem.

Apesar de possuir estas vantagens todas, existem algumas questões que o *e-learning* não consegue resolver. O facto de se poder partilhar o conhecimento de forma simples e rápida, por exemplo a montagem de uma mesa, não faz com que o aluno consiga montar à primeira tentativa, mesmo se estiver a seguir passo a passo o curso. A aprendizagem por experiência prática é necessária para algumas áreas, tal como a mecânica ou a cerâmica.

Outra desvantagem de usar o *e-learning* é o isolamento. O facto de a aprendizagem *online* ser uma acção principalmente solitária, pode dar uma sensação de solidão a quem está a aprender. À medida que a tecnologia avança, as pessoas podem-se envolver mais activamente com os professores ou outras pessoas através de ferramentas como a conferência de vídeo, as discussões em fóruns, entre outros.

O uso de *e-learning* está associado ao uso de um computador ou outros dispositivos electrónicos. Isto significa que pode provocar um cansaço da vista, uma má postura sentado e outros problemas físicos. Um constante aviso durante um curso pode diminuir os problemas de estar sentado várias horas em frente a um ecrã e assim minimizar os danos físicos provocados. Todas estas desvantagens são enunciadas em Epignosis [2].

Cabe ao aluno equacionar as vantagens e desvantagens no seu caso e decidir se beneficia mais do que se prejudica, tentando minimizar os riscos para si.

Clark e Mayer [10] dividem o *e-learning* em dois estilos distintos, o estilo síncrono e o estilo assíncrono e explicam o que consiste cada um:

- O estilo síncrono implica o acompanhamento de uma aula *online* na qual existe um professor ou orador a explicar o conteúdo. Isto permite que qualquer pessoa no mundo, com acesso à internet, assista à aula. As aulas são gravadas para futuras dúvidas e também para serem acedidas por quem as queira acompanhar ao seu ritmo. Um dos problemas deste estilo é a necessidade de a pessoa estar *online* e ter tempo para assistir à aula, ficando impossibilitada de sair, sob risco de perder a aula.
- O estilo assíncrono é mais livre, ou seja, a pessoa interessada em aprender ou aprofundar os seus conhecimentos decide qual o ritmo, a hora e o local onde está a aprender. Este estilo é largamente mais usado do que o síncrono, tanto por empresas como por particulares, por permitir que o trabalhador não perca horas de trabalho para assistir a uma aula, ou que um particular possa fazer os seus horários consoante a sua disponibilidade.

A aplicação de material de estudo sob a forma assíncrona é vasta, destacando-se os cursos *online*, disponíveis num servidor, aulas em vídeo, jogos *online* ou *offline*, tutoriais escritos, interactivos ou por vídeo, etc. O uso de jogos pode ser feito de duas formas: o jogo ser criado com o propósito de ser educativo e o jogo adaptar-se a uma aplicação educativa.

A melhor opção nem sempre passa por adaptar um jogo a uma aplicação educativa. De facto, há casos de jogos que foram adaptados mas que contêm alguns aspectos que não se inserem no pretendido. Um exemplo disso é o jogo Minecraft da Mojang.

O objectivo deste é matar um dragão que se encontra num plano de existência diferente chamado *the end* e para isso o jogador tem de conseguir alguns itens que caem quando determinado monstro morre. Após coleccionar doze itens, tem de encontrar um castelo especial e activar um portal com os itens que coleccionou e prossegue para o plano de existência do dragão. De seguida terá de matar o dragão.

Este jogo é largamente usado a nível educativo para impulsionar a criatividade através do seu sistema de blocos e de um mundo aberto. O jogador consegue construir vários tipos de edifícios, armadilhas, armas e muito mais, usufruindo de uma liberdade de criatividade enorme. No entanto, existem monstros com os quais o jogador tem de ter cuidado. Esta última parte sai

do contexto da criatividade e perde o objectivo da sua aplicação educacional. Na figura 2.1. é possível ver uma construção realizada no jogo Minecraft.

Figura 2.1: Construção feita no jogo Minecraft

No entanto, já existe uma versão do jogo específica para uso escolar que apenas se foca nos conteúdos educativos tais como o cálculo do perímetro, da área e do volume [11].

Um jogo que seja criado à partida para fins educativos vai produzir resultados muito positivos, uma vez que não contém desvios ao objectivo. Existem vários jogos na internet que foram desenvolvidos desde o início no conceito de *e-learning*. É o caso do jogo criado por Nick Russel da Benchmark-Learning, chamado crossword, que são palavras cruzadas em que o objectivo do jogador é escrever o nome dos animais que aparecem, completando os espaços. A figura 2.2. apresenta uma imagem do jogo com o nome de um animal já preenchido.

Como este jogo existem vários na internet, e também existem esqueletos de como criar um jogo de *e-learning*.

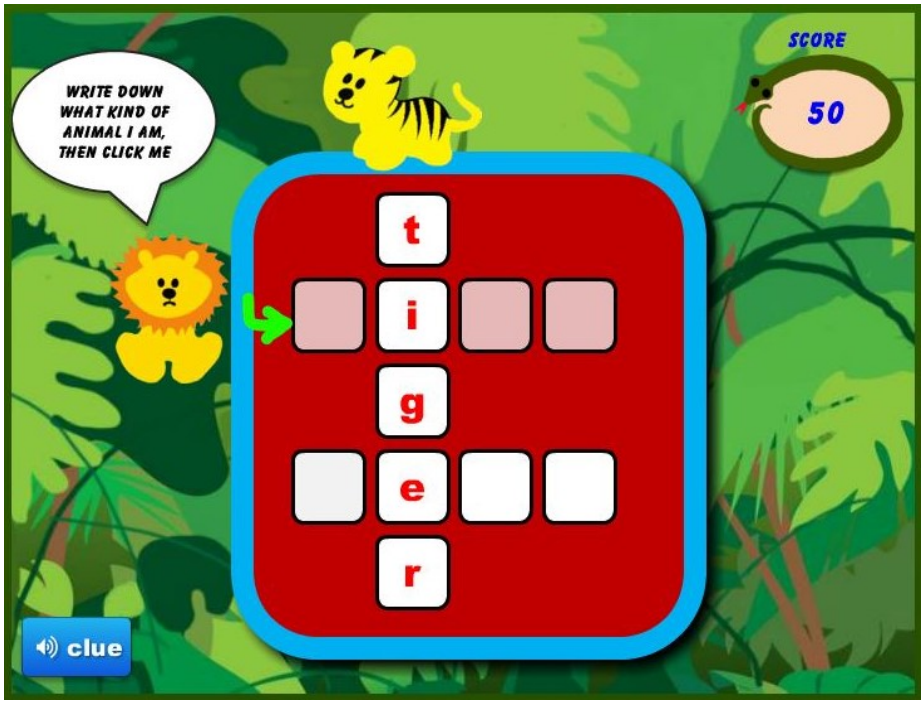


Figura 2.2: Jogo crossword

2.3 Jogos com características educativas

Existem vários jogos que contêm partes educativas, informativas ou da prática de habilidades. Um jogo pode desenvolver várias capacidades, nomeadamente a tomada de decisão, a destreza, a concentração, etc. Cada jogo pode ou não conter conteúdos de áreas pertinentes do ponto de vista educativo, tais como História, Geografia, Matemática, Línguas, etc.

Os jogos podem categorizar- se em vários tipos e estes podem-se inserir em mais do que um, criando uma possibilidade infindável de jogos distintos. Os jogos que serão abordados serão focados em conteúdos educativos.

2.3.1 Jogos de estratégia

Um jogo de estratégia é um jogo que permite desenvolver capacidades como a tomada de decisão, a concentração, percepção visual e pode enquadrar-se num cenário de guerra medieval, actual ou futurista, ou num cenário económico. O jogo Europa Universalis destaca-se por ser um jogo de domínio mundial, no qual o jogador controla uma nação e tem como objectivo desenvolvê-la a nível militar, económico, diplomático e colonial.

O Europa Universalis foi desenvolvido pela Paradox Development Studio em Março de 2000 e foi baseado num jogo de tabuleiro, com o mesmo nome, de Philippe Thibaut. O mapa de jogo está dividido em aproximadamente 1500 províncias e corre a um ritmo real pausado, passando-se entre os anos de 1444 a 1821. É possível escolher a data em que se quer começar a jogar e o jogo actualiza todo o mapa para coincidir com a história até essa data.

Actualmente, o jogo vai na quarta sequela e conta com 3003 províncias, sendo 1419 conquistáveis ou colonizáveis pelo jogador. Ganhou os prémios de melhor jogo de estratégia e de melhor jogo histórico nos prémios *Game Debate's* 2013, como descrevem os criadores do jogo no seu *website* [12].

Neste jogo é possível conhecer o nome das diversas províncias, regiões e países que fizeram parte do nosso passado, as suas alterações e principais acontecimentos marcantes na história da Europa, Ásia, África e América. Conta com nomes de exploradores, reis, generais e filósofos marcantes neste período. Destaca-se a nação Portuguesa e a Castelhana, por estas terem tido uma importância significativa na altura dos Descobrimentos ao estabelecer colónias e acordos comerciais pelos cinco continentes, começando com uma regência no reinado de D. Afonso V da casa real de Avis, reinado esse que acabou em 1481. Se se avançar para o ano de 1500, o rei de Portugal já é D. Manuel I e como exploradores já temos Vasco da Gama e Pedro Álvares Cabral. Portugal já tem presença na costa africana, em Cabo Verde e na Índia. Na figura 2.3. é possível observar o menu de escolha da nação e a data em que o jogo se inicia.

Figura 2.3: Painel de escolha da nação do Europa Universalis

No entanto, este não é o único jogo que explora a Geografia. O título Sid Meier's Civilization, desenvolvido pela Firaxis Games, também explora este conteúdo e aprofunda a História

antes da idade do bronze, 4000 a.C., até ao ano de 2100 D.C., podendo a história ser alterada consoante as decisões tomadas tanto pelo jogador como pela Inteligência Artificial, ou IA. São aqui apresentados todos os principais monumentos mundiais como edifícios especiais e os avanços científicos, culturais e militares que o mundo presenciou.

O título Rome: Total War também aborda a História e a Geografia. Desenvolvido pela Creative Assembly da Sega, há vários jogos na colecção Total War que tratam de diferentes momentos históricos, sendo que todos se centram na Europa com excepção de um título, o Shogun: Total War, passado no Japão medieval.

2.3.2 Simuladores

Os jogos de simulação são vários e das mais diversas áreas. Estes jogos possibilitam um aumento de conhecimento em áreas distintas, tais como economia, politica, gestão, entre outras.

O objectivo de um jogo de simulação é retratar a realidade o mais veridicamente possível. O uso de simuladores é vasto, principalmente no treino de pilotos de aviões militares, ou de pilotos de corridas de carros ou motas. Estes simuladores tentam igualar o jogo à realidade o melhor possível, permitindo que quem o jogue ganhe alguma experiência sem pôr em risco a vida. Isto permite ao jogador cometer erros que na vida real podiam resultar na sua morte, ou no seu condicionamento físico.

No *website* oficial do governo dos Estados Unidos da América é informado o início de um projecto, em parceria com a GameSim, para desenvolver um simulador de voo para os aviões F-35 com base num modelo 3D, tendo como objectivo comercial inicial a sua integração na força naval americana. Como objectivo seguinte pretende proceder-se à sua comercialização em todos os países que compraram o avião F-35 [13].

Existem no entanto outros simuladores que se destinam a todo o tipo de audiências. O RollerCoaster Tycoon é um simulador de um parque de diversões, que permite criar as próprias montanhas russas ou utilizar alguns modelos já existentes no jogo. O objectivo principal do jogo é criar um parque de diversões com sucesso e para isso o jogador deve pensar na disposição das diversões, ter em atenção as suas finanças, fazer publicidade e manter o parque num nível de manutenção e higiene aceitáveis para os clientes. Para isto precisa de contratar mecânicos e empregados de limpeza, e delimitar as áreas de acção destes de forma a cobrir o parque todo. Controla também o preço de cada diversão, sendo que um preço demasiado elevado, atendendo à elasticidade da procura/oferta, pode provocar um prejuízo directo, com consequências globais para o parque. Este jogo permite simular a capacidade de gestão que o jogador tem e aprender quais as principais necessidades de um negócio. Na figura 2.4. é possível ver um parque criado no RollerCoaster Tycoon 3.

Figura 2.4: Parque de diversões criado no jogo RollerCoaster Tycoon 3

O jogo Masters of the World é um simulador geopolítico do mundo actual. Os jogadores podem jogar como líderes de um estado ou de um governo de um país que escolhem quando iniciam o jogo. Podem tomar acções económicas, sociais, militares, políticas, tanto internas como externas, ambientais, culturais, entre outras. Todos os países do mundo são representados com as suas variáveis e modos de funcionamento. O jogo inclui fases para gestão económica, mercantilismo, guerras, construções, espionagem, simulações e manipulação política.

No seu *website* refere também que o Masters of the World é largamente utilizado devido ao seu simulador do mundo extremamente realista que ajuda os alunos a perceber como funcionam várias áreas, tais como a área económica e a política. O jogo possui acima de mil decisões que um líder pode tomar nos campos da política e da economia, mas também em termos financeiros, militares, de segurança, culturais entre outras áreas [14].

Na figura 2.5. pode-se observar uma imagem do jogo.

Figura 2.5: Imagem retirada do jogo Masters of the World

2.4 Programas de desenvolvimento de jogos

Antes de se estudar os programas usados para desenvolver um jogo, também chamados de motores de jogo, é preciso explicar o que é um motor de jogo.

Gregory [15] afirma que um motor de jogo é um programa que é extensível e que pode ser usado como fundação para construir vários jogos sem sofrer grandes modificações. Normalmente consiste numa ferramenta e num componente *runtime*. Tal como todos os sistemas, os motores de jogos são construídos em camadas, e normalmente as camadas superiores dependem das inferiores e não o contrário.

Também descreve algumas das camadas de motores de jogo como se pode verificar na figura 2.6.:

- ***Hardware* alvo** – A camada do *hardware* alvo representa o sistema de computador ou consola onde o jogo irá correr. As plataformas típicas incluem o Microsoft Windows e computadores com base em Linux, o iPhone e o Macintosh da Apple, consolas como a Xbox, a PlayStation e a Nintendo;
- ***Drivers* dos dispositivos** – Os *drivers* são programas de baixo nível que são facultados pelo vendedor do sistema operativo ou do fabricante. Os *drivers* gerem os recursos de *hardware* e protegem o sistema operativo e as camadas superiores

do motor de detalhes de comunicação com as várias variantes de dispositivos de *hardware* disponíveis;

- **Sistema operativo** – Num computador o sistema operativo está sempre a correr. Ele orquestra a execução de múltiplos programas num único computador, em que um deles é o jogo. Sistemas operativos como o Microsoft Windows empregam uma aproximação de partilha de tempo para partilhar o *hardware* com todos os programas que estão em execução, é conhecido como *pre-emptive multitasking*. Isto significa que um jogo não pode assumir que terá todo o controlo do *hardware*, terá de correr bem com todos os programas em execução no computador;
- ***Middleware* e SDK feitos por terceiros** – A maioria dos motores contém um número de programas desenvolvidos por terceiros, os *Software Development Kits* ou SDK, e *middlewares*. A funcionalidade ou a interface baseada em classes de um SDK é muitas vezes chamada de API, *Application Programing Interface*. Alguns SDK são mais conhecidos do que outros, mas todos têm a sua importância. Alguns exemplos são:
  - Algoritmos e estruturas de dados – STL e Loki são dois SDK cujo objectivo é manipular e estruturar dados;
  - Gráficos – OpenGL, DirectX e Glide são alguns exemplos de SDK para gráficos. Na maioria dos jogos estes SDK são construídos em cima de uma biblioteca de interface de *hardware*;
  - Colisões e físicas – PhysX, Havok e Open Dynamics Engine, ou ODE, são alguns exemplos. Estes são responsáveis por detectar colisões entre objectos e pela dinâmica de corpos rígidos;
  - Animação de personagens – Havok Animations, Edge e Granny são alguns exemplos que possibilitam a animação de personagens;
  - Inteligência artificial – Kynapse é um exemplo de um SDK que é responsável pela IA de um jogo;
  - Modelos biomecânicos de personagens – Endorphin and Euphoria é um SDK que permite criar movimentos de uma personagem com base em modelos biomecânicos avançados de movimento humano realista.

No entanto, há SDK que se misturam, possibilitando trabalhar em duas áreas. O Havok Animation tenta juntar a física com a animação de forma tradicional, usando um animador humano para fornecer a maioria dos movimentos através de ferramentas como o Maya e com físicas que aumentam esse movimento.

- **Camada independente da plataforma** – A maioria dos motores de jogos têm de ter a capacidade de correr em mais de uma plataforma. Existem algumas empresas que assim o fazem, mas há outras que preferem criar um motor de jogo para uma só plataforma. Por consequência, os motores de jogo são arquitectados com uma camada independente da plataforma. Esta camada situa-se em cima do *hardware*, dos *drivers*, do sistema operativo e de outros programas de terceiros, protegendo o resto do motor da maioria do conhecimento das plataformas em baixo. Ao substituir as funções mais usadas da biblioteca C estandardizada, chamadas ao sistema operativo e outras fundações de API, a camada de independência da plataforma garante um comportamento consistente ao longo das várias plataformas de *hardware*. (em duvida se ponho ou não)
- **Gestor de recursos** – Presente em todos os motores de jogo, o gestor de recursos fornece uma interface unificada para aceder a todos os *assets* do jogo e a dados de *input* de outro motor. Alguns motores de jogo fazem-no de forma centralizada e consistente, enquanto outros tomam uma aproximação de *ad hoc*, deixando arquivos comprimidos.
- **Motor de *rendering*** – Este motor é um dos maiores e mais complexos componentes de qualquer motor de jogo. Podem ser arquitectados de várias maneiras e não há uma correcta de o fazer, apesar de os mais recentes partilharem algumas filosofias de *design*, derivado em grande parte do *design* do *hardware* de gráficos 3D de que dependem.
- **Efeitos visuais** – os motores de jogo modernos suportam uma vasta colecção de efeitos especiais, incluindo:
    - Sistemas de partículas para fumo, fogo, água, etc;
    - Sistemas de decalque para buracos de balas, pegada, etc;
    - Iluminação do mapa e mapa do ambiente;
    - Sombras dinâmicas;
    - Pós-feitos em *full-screen*, aplicados após o *rendering* das cenas em 3D.

  É normal para um motor de jogo conter um sistema para efeitos que gere a necessidade especializada de *rendering* para partículas, decalques e outros efeitos visuais.
- ***Front end*** – A maioria dos motores de jogo emprega um tipo de gráficos 2D por cima de uma cena 3D para vários objectivos, estes incluem:
    - O *head's up display*, ou HUD, do jogo;

  - Menus dentro do jogo, uma consola, e/ou outras ferramentas de desenvolvimento, que podem ou não ser incluídas no produto final;
  - Possivelmente uma *grafical user interface*, ou GUI, que possibilita ao jogador manipular o inventário da personagem, configurar unidades para batalhas ou realizar outras tarefas complexas dentro do jogo.

- **Colisões e físicas** – a detecção de colisões é importante para qualquer jogo. Sem ela, os objectos iriam transpor-se e iria ser impossível interagir com o mundo virtual de uma maneira razoável. Alguns jogos incluem uma simulação dinâmica realista ou semi-realista. Isto é chamado de sistema de físicas, apesar de o nome dinâmica de corpos rígidos ser mais apropriado, pois normalmente só a cinemática de corpos rígidos, as forças e o binário é que causam o movimento.
- ***Human Interface Devices*, ou HID** – Todos os jogos precisam de processar *inputs* do jogador obtidos de vários HID:
  - O teclado ou o rato;
  - Um volante ou um *joystick*;
  - Outros controladores de jogo especializados, como um volante, canas de pesca, tapete de dança, o *WiiMote*, etc.

  Às vezes este componente é chamado de *Input/Output*, ou I/O, do jogador porque também se pode fornecer *output* ao jogador através do HID, tal como a força contrária e vibração num volante, ou o som produzido pelo *WiiMote*.
- **Áudio** – O som é tão importante como os gráficos num motor de jogo. Infelizmente o som costuma ter menos atenção do que o *rendering*, as físicas, a animação, o IA e a jogabilidade. Uma das razões é o facto de alguns programadores de jogos não programarem com som e alguns nem colunas ou auscultadores têm. Contudo, nenhum grande jogo está completo sem um motor de som.
- **Sistema de fundação à jogabilidade** – O termo jogabilidade refere-se à acção que ocorre no jogo, às regras que governam o mundo virtual no qual o jogo ocorre, às habilidades da personagem do jogador, de outros personagens e objectos no mundo, aos objectivos e metas do jogador. A jogabilidade é tipicamente implementada tanto na linguagem nativa na qual o resto do motor de jogo é escrito, ou numa linguagem de *script* de alto nível, ou às vezes nos dois. Para ligar o código da jogabilidade com os sistemas de baixo nível do motor, a maioria dos motores

introduz uma camada que pode ser chamada de camada de fundação à jogabilidade. Esta camada faculta uma colecção de funcionalidades centrais, na qual a lógica de jogo específica pode ser implementada:

- Mundos do jogo e modelos de objectos – A camada de fundação à jogabilidade introduz a noção de mundo de jogo contendo elementos estáticos e dinâmicos. Os conteúdos do mundo são normalmente modelados de uma maneira orientada a objectos, normalmente mas nem sempre, usando uma linguagem orientada a objectos;
- Sistema de eventos – os objectos de jogo precisam invariavelmente de comunicar entre eles. Isto pode ser atingido de várias maneiras. Por exemplo, um objecto que envia uma mensagem pode simplesmente invocar uma função do objecto receptor. Uma arquitectura guiada por eventos, muito idêntica ao que se encontra numa típica GUI, é também uma aproximação à comunicação entre objectos. Num sistema guiado por eventos, o emissor cria uma estrutura de dados pequena que se chama de evento ou mensagem, contendo o tipo da mensagem e qualquer argumento necessário para ser enviada. O evento é passado para o objecto receptor chamando a sua função de controlo de eventos. Os eventos também podem ser guardados em espera para serem controlados numa outra altura;
- Sistema de *scripting* – muitos motores de jogos empregam uma linguagem de *scripting* de forma a desenvolver as regras de jogabilidade do jogo e conteúdo mais fácil e mais rápido. Sem uma linguagem de *scripting*, teria de se recompilar e voltar a ligar o executável do jogo sempre que uma mudança fosse feita à logica ou às estruturas de dados usadas no motor. Mas quando uma linguagem de *scripting* é integrada no motor, mudanças à lógica e aos dados podem ser feitas apenas modificando e recarregando o código do *script*. Alguns motores permitem que o *script* seja recarregado enquanto o jogo continua em execução. Outros motores requerem que o jogo seja encerrado antes da recompilação dos *scripts*. Mas em qualquer dos casos, o tempo despendido é muito inferior do que se tivesse que se recompilar e voltar a ligar o executável do jogo;
- Fundações da Inteligência Artificial – Tradicionalmente, a inteligência artificial caiu para o patamar dos programas específicos de jogos, não era normalmente considerada uma parte de um motor de jogo. Mais recentemente, as empresas de jogos reconheceram padrões que cresceram em quase todos os sistemas de IA e estas fundações estão lentamente a começar a ficar sob a alçada dos motores.

- **Subsistemas específicos do jogo** – em cima da camada da fundação da jogabilidade e de outros componentes de baixo nível do motor, os programadores de jogabilidade e *designers* colaboram para implementar as características do jogo. Os sistemas de jogabilidade são normalmente numerosos, altamente variáveis e específicos para o jogo em desenvolvimento. Estes sistemas incluem, mas não são limitados só à mecânica da personagem do jogador, vários sistemas de câmara dentro do jogo, IA para controlo de NPC, sistemas de armas, veículos e a lista continua. Se uma linha recta pudesse ser definida entre o motor e o jogo, esta teria de um lado os subsistemas específicos do jogo e do outro lado a camada de fundação da jogabilidade [15].

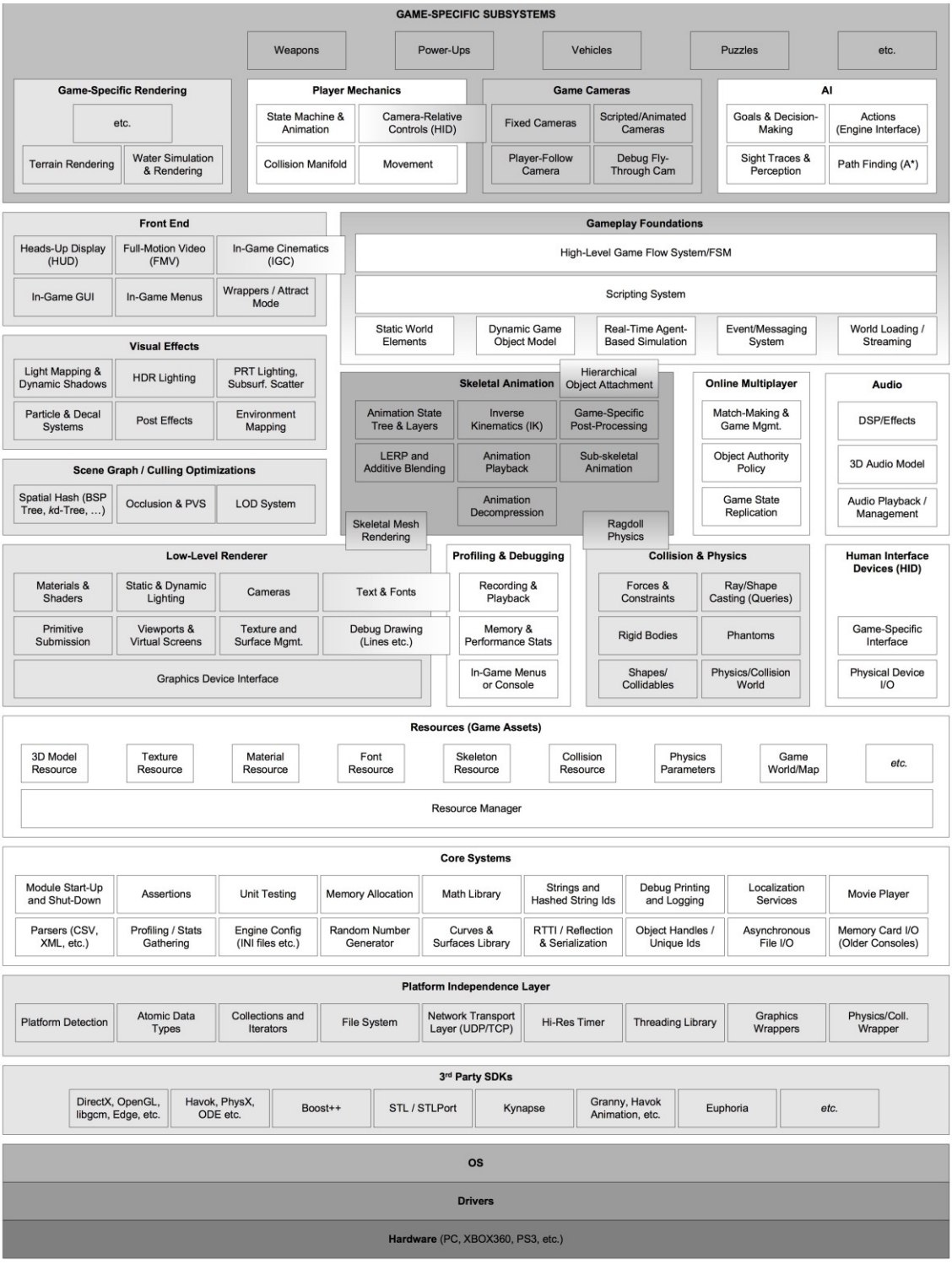


Figura 2.6: Arquitectura do *runtime* de um motor de jogo [15]

Existem vários motores de jogo: Unity®, Unreal Engine, CryEngine, GameMaker, Havok Vision Engine, etc. Este estudo vai recair nos primeiros quatro.

2.4.1 Unity®

O Unity® foi anunciado pela primeira vez na *Apple's Worlwide Developers Conference* em 2005 como sendo só para OS X. Desde então foi estendido para mais de vinte plataformas [16] e é o programa padrão usado para os jogos da consola Wii U afirmado por Helgason numa entrevista conduzida pela GamesBeat [17].

O *website* oficial descreve-o como uma ferramenta para desenvolver conteúdo interactivo e implantar em multiplataformas. Seja para criar jogos, orientações de arquitectura, simulações de treino *online* ou arte interactiva [16]. Permite a criação de diferentes cenários em 2D ou 3D através do seu motor de físicas de forma rápida e com resultados impressionantes. Na criação de um cenário é possível editar todos os objectos existentes através do seu sistema *drag-and-drop* e obter uma pré-visualização do resultado final, afirma Mota [3].

O Unity® suporta diversos formatos multimédia, tais como psd, jpg, png, bmp, mp3, wav, avi, mp4, html, xml, etc. Tem a possibilidade de importar modelos 3D criados por outros programas no seu formato nativo, tais como, modelos criados em Max, Maya, Blender, Cinema4D, Modo, Lightwave e Cheetah3D. Para processar os modelos, o Unity® usa três APIs, a Direct3D para *Windows* e *Xbox360*, a OpenGL para Mac e *Windows* e a OpenGL ES para *Android* e *IOS* [16].

O motor de *scripts* do Unity® é construído em cima do Mono, uma implementação *open-source* da *.NET Framework* da Microsoft e permite o uso de UnityScript[5], C# ou Boo[6]. Contém características de IA com avançado *path finding* automático e malhas de navegação [16].

Como motor de físicas, o motor PhysX é o usado pelo Unity®. O PhysX foi criado pela NovodeX, uma *spin-off* da ETH Zurich. Em 2004 foi comprado pela Ageia e em Fevereiro de 2008 a Ageia foi comprada pela Nvidia [18]. É um *multi-threaded* SDK de simulação de físicas disponível para Windows, Mac OS X, Linux, PlayStation 3, Xbox 360, Wii, etc. Suporta dinâmicas

[5] Linguagem por defeito com sintaxe inspirada em ECMAScript, referida por JavaScript pelo Unity®.

[6] Sintaxe inspirada em Python.

de *rigid body, soft body*, controladores de *ragdolls*[7] e personagens, dinâmica de veículos, partículas, simulação de volumes de fluidos, simulação de roupa, incluindo rasgar e roupa pressurizada são algumas das capacidades descritas pelo *website* oficial da Nvidia [19].

Com este motor o Unity® é capaz de simular todos os comportamentos físicos e de interacção entre objectos, seja por parte do jogador seja por parte de outro objecto. Estas físicas estão disponíveis para serem usadas em *scripts* conforme o programador desejar.

2.4.2 Unreal Engine

O Unreal Engine, actualmente na versão 4, é um conjunto completo de ferramentas de desenvolvimento de jogos feito por programadores de jogos para programadores de jogos. Foi criado pela Epic Games em 1998 apresentando o seu primeiro jogo, *Unreal*, um FPS. Apesar de se ter focado em FPS no início, tem vindo, com sucesso, a ser usado para outros tipos de jogos, nomeadamente, *stealth*, MMORPGs e outros RPG.

No seu *website* oficial indica que permite criar jogos para várias plataformas, tais como Windows, OS X, Android, IOS X, HTML5, Oculus Rift, Samsung Gear VR [20].

O Unreal Engine 4 está implementado em C++ e possibilita uma customização e/ou extensão das ferramentas do Unreal *Editor* bem como dos subsistemas do Unreal Engine, tornando-se mais flexível que a sua versão anterior, Unreal Engine 3.

O seu motor de *script* é o Blueprints Visual Scripting, tal como é referido no *website*. Através do uso de *blueprints* é possível prototipar, implementar ou modificar virtualmente qualquer elemento do jogo. A linguagem é facilmente traduzida para C++ e, caso não exista alguma funcionalidade em *blueprint,* pode-se criar uma classe em C++ e fazê-la aparecer em *blueprint*, como é descrito no Project NORS. [21] [20].

A Epic também informa que o Unreal Engine usa como motor de físicas o PhysX da Nvidia. Está capacitado para integrar mais de uma dúzia de *middlewares* líderes na indústria, tais como a Autodesk Gameware, permite criar luzes ao longo dos mapas, animação de personagens, baixo nível de *path finding* e alto nível de IA, a Enlighten, permite criar todo o tipo de luzes para os mais variados cenários, a Umbra, permite criar bases de dados optimizadas de todos os modelos 3D, a Oculus Rift VR, permite toda a integração de realidade aumentada do Oculus Rift e outras bibliotecas [20].

[7] Animação que é muitas vezes substituída à morte estática em jogos e filmes animados.

2.4.3 CryEngine

CryEngine é um motor de jogo desenvolvido pela empresa Crytek e é usado em todos os seus títulos desde o *Far Cry*. A Crytek afirma que actualmente é possível criar jogos para as seguintes plataformas: Windows, OS X, Linux, PlayStation 4, Wii U, Xbox One, IOS e Android [22]. Também é o único motor de jogo que apresenta vários prémios ganhos em características que possui, tais como gráficos, física realista, *visual scripting* intuitivo, som de alta-fidelidade, uma solução eficiente de 3D estereoscópica sobre várias plataformas.

No *website* oficial do CryEngine são descritas algumas das suas características como o *SandBox Editor*, o *Flow Graph*, *Track View Editor*, entre outros.

O *Sandbox Editor* é o editor de níveis do CryEngine e dá total controlo sobre as criações multiplataforma em tempo real. Uma das ferramentas contidas na *Sandbox* é o *Material Editor* que permite interagir e modificar materiais que tenham sido criados numa ferramenta externa. Possibilita a aplicação de várias texturas a vários tipos de objectos e terrenos. Os utilizadores também podem aplicar vários *shaders* e ajustar os parâmetros e as propriedades dos materiais dentro do editor.

Outra ferramenta bastante útil é o *Flow Graph*, este sistema dá aos *designers* uma interface intuitiva para criar e controlar eventos, *triggers*, lógica do jogo, efeitos e desenho de som. Isto permite construir níveis complexos sem necessitar de programar um único *script*. Se for necessária alguma funcionalidade de baixo nível, uma vez que não é suportada pelo *Flow Graph*, é feita através de *scripts* em C++ e Lua.

Existem mais alguns sistemas dentro da ferramenta *Sandbox* que permitem editar cenas de vídeo, criação de cenários do tipo floresta ou urbanos, criação do comportamento da vegetação de acordo com as leis da natureza, fácil criação de efeitos especiais complexos como o fogo, explosões ou o fumo, criação de estradas, escadas ou rios, criação de veículos e um sistema para simular a órbita do sol em torno da terra, simulando a passagem do dia no jogo.

Como motor de físicas usa um próprio e, como é descrito no seu *website,* não precisa de usar nenhum *middleware* para as físicas [22].

2.4.4 GameMaker

Originalmente intitulado de Animo, o programa foi lançado em 1999 e começou por ser um programa para criação de animações 2D. Foi criado por Mark Overmars em linguagem Delphi, como é descrito na plataforma Desura da Bad Juju Games [23].

Ford destaca o GameMaker por permitir aos utilizadores um desenvolvimento facilitado de um jogo sem terem de aprender uma linguagem de programação complexa através do seu sistema *drag-and-drop*.

Afirma também que corre jogos que usem 2D e permite o uso de 3D mas limitado. O GameMaker não tem maneira de escolher qual API de gráficos deve correr para renderizar em determinada plataforma e, portanto, usa o Direct3D desde a versão 6.0 em Windows e o OpenGL desde a versão 7.0 noutras plataformas que não sejam Windows. Também não suporta DirectX *mesh*, apenas suporta o *mesh* "d3d" nativo do programa [24].

No seu *website* faz-se grande referência ao motor de *scripts* que usa um nativo e único, o GML que significa Game Maker Language. Com este motor é possível controlar vários detalhes da criação do jogo tais como *path finding*, física, interacção com objectos, partículas, estruturas de dados, etc.

Afirmam também que consegue distribuir jogos ao longo de várias plataformas, de entre as quais se destacam a PlayStation, a Xbox, o Windows, o Mac, o IOS, o Android, entre outras. Com apenas um *click* e módulos adicionais consegue criar executáveis para as plataformas acima mencionadas, através do seu motor de *script* [25].

Tabela 2.1: Comparação dos diversos motores de jogo

	Unity3D	UnrealEngine	CryEngine	GameMaker
Scripts	C#, UnityScript, Boo	blueprints, UnrealScript	C++, Lua	GML
Motor de físicas	PhysX da Nvidia	PhysX da Nvidia	CryEngine physics	ExtremePhysics
Motor gráfico	DirectX 11	DirectX 11 & 12	CryEngine	Direct3D, OpenGL (limitado)
Plataformas suportadas	Windows, Mac, Linux/steam OS, iOS, Android, Windows phone, Playstation, Xbox, Wii U	Windows, Mac, iOS, Android, Playstation, Xbox, Web Browsers	Windows, Linux Playstation, Xbox, Wii U	Windows, Mac, Ubunto, iOS, Android, Playstation
Licenças	Versão gratuita limitada	$19 por mês	$9,90 por mês	Versão gratuita limitada

Dimensões suportadas	2D, 3D	2D, 3D	2D, 3D	2D, 3D (limitada)

2.5 Síntese

O uso de jogos para auxiliar um método educativo é uma das vertentes do *e-learning* e o sistema que vai ser proposto enquadra-se neste conceito, uma vez que visa ajudar na aprendizagem da língua portuguesa. Quanto ao seu desenvolvimento, tanto o trabalho desenvolvido pelo Centro Diferenças, como todas as decisões tomadas referentes ao jogo implicam a base de GBL, em que um jogo é desenvolvido em torno de um método educativo de forma a auxiliar um determinado objectivo educativo. O *gamification* define que se use mecânicas de jogo para ajudar o jogador a atingir objectivos, no entanto, este sistema desenvolvido pode não ser um jogo. Para se desenvolver um jogo, tanto no conceito de *gamification* como no de GBL, é necessário utilizar um motor de jogo, e o sistema proposto assenta no uso de um dos motores estudados. Um motor de jogo vai realizar todo o processamento necessário para que o jogo seja executado automaticamente, permitindo que o criador de jogo consiga desenvolver todas as componentes de um jogo sem necessidade de desenvolver um *middleware* ou uma API.

3

Proposta

Tendo como problema o défice no mercado de ferramentas que visam ajudar as causas sociais na aprendizagem da língua Portuguesa, propõem-se uma que não só ajude neste campo mas também possibilite a gravação do uso da mesma.

3.1 Identificação do problema

A falta de ferramentas nas práticas terapêuticas de causas sociais é um problema actual. Normalmente as ferramentas são adaptadas com o intuito de tentar resolver os problemas de aprendizagem e muitas vezes não só não o fazem como dificultam o processo. As dificuldades presentes na aprendizagem da língua Portuguesa em crianças com problemas do neurodesenvolvimento são particularmente complexos, uma vez que são necessários exercícios diferentes.

Os exercícios devem explorar as dificuldades que cada criança apresenta, tendo o cuidado de não provocar desatenção nem desmotivação. Com este intuito, é difícil de adaptar uma ferramenta que assente no ensino corrente uma vez que há determinadas características que esta não contém.

Para uma criança com problemas do neurodesenvolvimento a reprovação é uma causa de desmotivação e deve ser evitada ao máximo. Por outro lado, uma ferramenta visualmente atractiva pode provocar uma desatenção, desviando as atenções do objectivo. Para crianças sem problemas, a reprovação é um dos métodos usados para motivar e uma ferramenta que seja visualmente atractiva faz com que a criança tenha vontade de continuar a jogar.

Outro aspecto que as ferramentas já existentes não fornecem são dados de utilização. Estes são importantes para as pessoas que acompanham as crianças, pois permitem realizar estudos sobre os métodos usados e avaliar as dificuldades apresentadas.

Ao perceber-se as características especiais das crianças surge a necessidade de se esquematizar os requisitos da ferramenta proposta que, neste caso, será num jogo.

3.1.1 Requisitos do jogo

Wilson Filho [26] define que o valor de um programa está nas suas características e neste caso divide-as em duas:

- Funcionais, que representam os comportamentos que o programa deve apresentar diante de determinadas acções dos seus utilizadores;
- Não funcionais, que quantificam determinados aspectos do comportamento.

Os requisitos são características que definem os critérios de aceitação de um produto. O objectivo é colocar no produto as características que são requisitos.

Requisitos Funcionais

Para requisitos funcionais foram estabelecidos, com base na descrição do problema, os seguintes:

- Interface deve ser usada através do rato e teclado, maioritariamente rato;
- O sistema deve permitir um cancelamento prematuro dos níveis;
- O sistema deve exportar os dados num formato conhecido, de fácil manuseamento e de forma automática;
- O sistema não deve permitir a alteração dos seus componentes de forma a garantir o seu correcto funcionamento.
- O sistema deverá permitir a criação de vários utilizadores.

Requisitos não Funcionais

Para requisitos não funcionais foram enumerados, com base na descrição do problema e possuidores da maior compatibilidade possível para uma futura actualização e manutenção, os seguintes:

- O sistema deverá operar na plataforma Windows;
- A navegação pelo sistema deverá ser intuitiva e auto-explicativa;
- O sistema deverá executar-se de forma fluída, mesmo em computadores não recentes;
- A estrutura do código deverá ser esquematizada e desenvolvida de forma padrão, de modo a facilitar a manutenção e a actualização;
- O sistema deve permitir acrescentar mais letras sem a necessidade de alterar as mecânicas;
- Caso exista algum erro de funcionamento, o sistema deverá conseguir continuar a sua execução, não perdendo dados.

3.2 Descrição da solução

Tendo em atenção os requisitos explicados no ponto 3.1.1., o jogo terá de ter um conjunto de características essenciais.

O jogo proposto terá a capacidade de gerir vários utilizadores, gravando a utilização de cada um e esta não deverá requerer uma interacção do jogador.

O acesso às funcionalidades deverá ser efectuado de forma simples, não obrigando a uma sucessão de botões. Os objectivos pretendidos deverão ser explicados de forma clara e a navegação deverá ser intuitiva.

O acesso aos dados deverá ter em atenção o conhecimento das pessoas que os vão utilizar, permitindo usar ferramentas já conhecidas como, por exemplo, o Excel.

3.2.1 Estrutura do Jogo

O jogo, intitulado “No Reino dos Fonemas”, está assente na história desenvolvida pelo centro Diferenças e está estruturado em três zonas. Cada zona contém vários níveis e cada nível é constituído por diversas etapas. Na figura 3.1. é possível ver a estrutura do jogo.

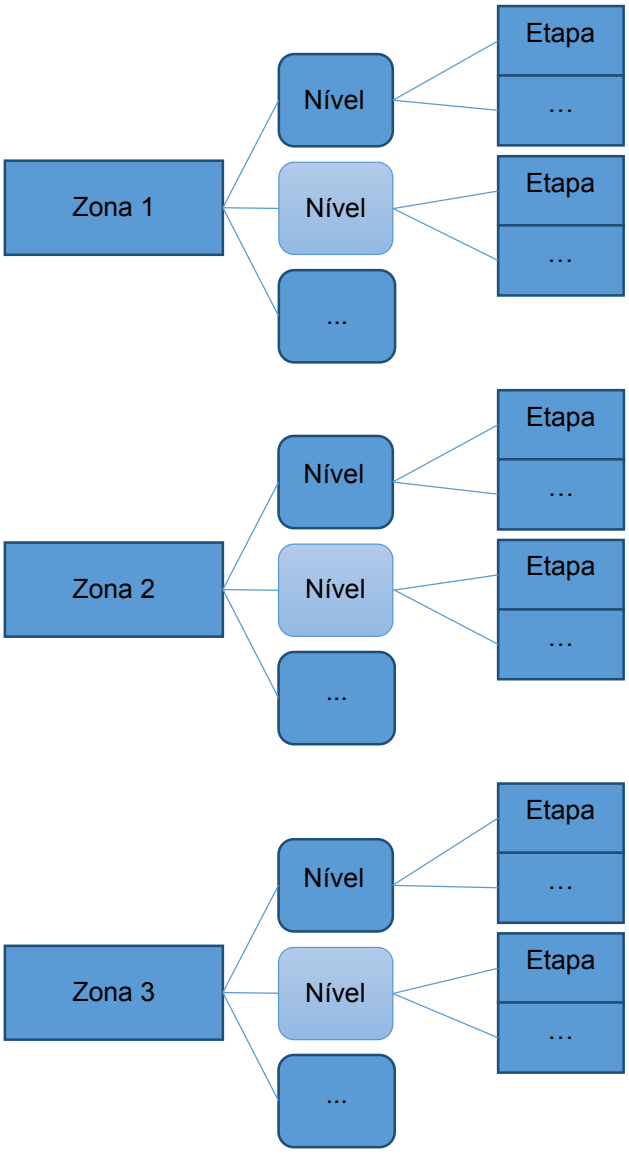


Figura 3.1: Estrutura do jogo

3.2.2 História

No Reino dos Fonemas, o Rei Fonos vivia calmamente e em paz no seu castelo com a sua mulher, a Rainha Xílaba, e as suas cinco filhas. Numa determinada noite o reino foi acordado por uma trovoada que surgiu de forma inesperada.

Preocupado, o Rei Fonos dirigiu-se à grande janela do salão, onde jantava com a sua família, para a fechar e evitar que batesse ou se partisse com o vento que também se levantara.

Neste momento, um grande clarão iluminou a noite e num raio de trovão desceu o Feiticeiro Errum, que entrou pela janela antes de esta se ter fechado e aterrou à frente da família real. O Feiticeiro, com um sorriso maléfico e agitando a sua varinha mágica no seu braço, lançou um poderoso feitiço sobre as princesas e fê-las desaparecer instantaneamente numa névoa de fumo espesso.

O Rei, espantado com o sucedido, perguntou ao Feiticeiro para onde ele tinha levado as suas filhas, respondendo o Feiticeiro que só lhe diria se o tesouro fosse usado como moeda de troca.

Perante o pranto da Rainha, que o abraçava, o Rei decidiu dar as riquezas do reino para salvar as suas filhas. No final dos acontecimentos, o Rei percebeu a direcção que o Feiticeiro tomava e apercebeu-se que este se dirigia para a Ilha Alfabeto, tendo o Feiticeiro deixado um mapa e a indicação de que ele encontraria as filhas se seguisse o mapa.

No dia seguinte, o Rei Fonos espalhou pelo reino um pedido de ajuda, em busca de jovens meninas e meninos destemidos que tivessem coragem e vontade de aprender e derrotar os feitiços do Feiticeiro e salvar as princesas.

Não tardaram a aparecer dois jovens irmãos, a Alfa e o Berto, prontos para enfrentar o desafio da sabedoria, vencer o Feiticeiro, resgatar o tesouro do reino e as princesas. O Rei Fonos entregou-lhes o mapa, apresentado na figura 3.2. e advertiu-os de que o caminho, tendo como percurso final a Torre Mágica, não ia ser fácil.

Figura 3.2: Mapa da Ilha Alfabeto

O jogador terá de concluir os desafios propostos na Ilha Vogalis e prosseguir para a Ilha Consoantis, concluir os desafios nesta apresentados e avançar para a etapa final do jogo, a Torre Mágica. O jogo está separado em duas componentes, a estática e a dinâmica.

A componente dinâmica foi implementada por Pedro Bueno Mota [3] e consiste numa mecânica de Super Mário com uns balões um pouco acima da personagem, tendo esta de saltar para apanhar uma letra que pode aparecer escrita à máquina ou à mão.

Por esta dissertação assentar nas vogais, a organização dos níveis vai incidir mais nas mesmas.

3.2.3 Estrutura dos níveis

Cada letra representa um nível e cada um tem várias etapas que, ao serem concluídas pelo jogador, o fazem avançar para o nível seguinte.

O nível varia dependendo da vogal em questão. A componente dinâmica mantém-se inalterável, tendo sempre três etapas gráficas iniciais e duas finais sonoras. A componente estática varia consoante a vogal que se está a jogar. Tem etapas variáveis na parte do som e tem três finais.

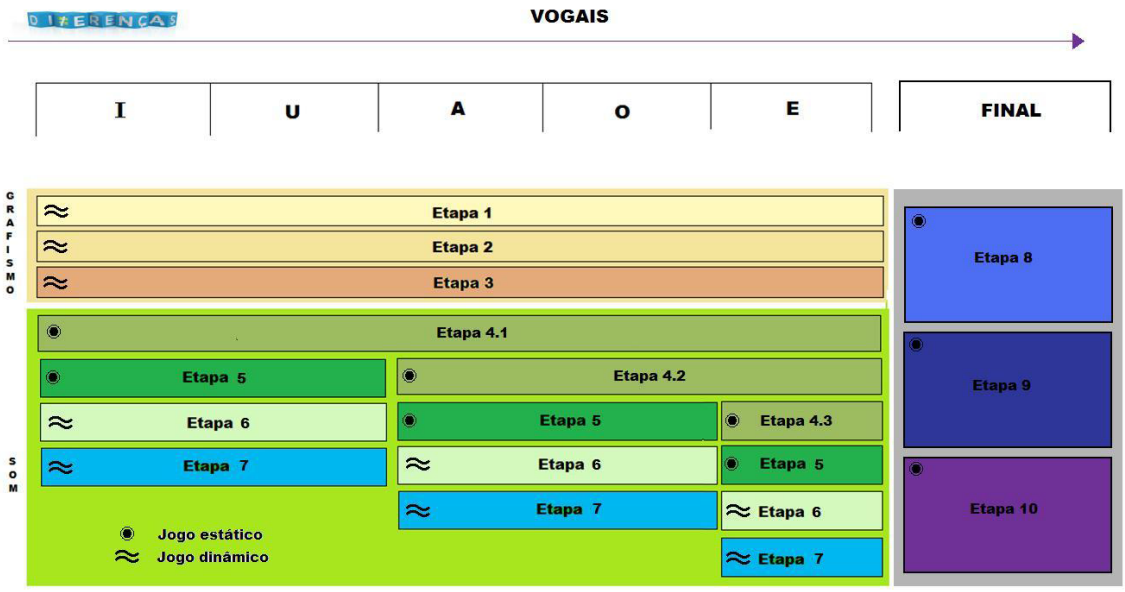


Figura 3.3: Estrutura dos níveis das vogais

Como se pode ver na figura 3.3., para a letra “i” existem duas etapas, já para a letra “e” existem quatro. Esta diferença é explicada por a componente estática estar associada ao som que cada vogal produz na língua portuguesa. Para a letra “i” apenas um som é produzido, enquanto que para a letra “e” existem três distintos e diferentes. No fim existe uma etapa que representa todos os sons possíveis pela vogal, daí a letra “i” conter duas etapas, apesar de ter apenas um som e de a letra “e” conter quatro etapas, apesar de conter três sons. Através de diálogos com o centro Diferenças foi decidido não implementar a última etapa de cada vogal, devido a ser impossível errar e, por conseguinte, não se perceber se o jogador conseguiu acertar por saber ou porque jogou de forma aleatória. Nesta última etapa as imagens apresentadas podem ser constituídas apenas por uma só vogal, como por exemplo “batata”, fazendo com que todas as sílabas possuam um som alusivo à letra “a”, produzindo apenas sílabas correctas e tornando impossível o erro.

As etapas finais, pertencentes à componente estática são o desafio final. Ao conclui-las o jogador pode aceder à Ilha Consoantis e concluir a Ilha Vogalis.

A interface que permitirá aceder aos níveis da Ilha Vogalis deve ter em atenção a vogal escolhida, de forma a iniciar o nível pretendido, e seguirá o esquema apresentado na figura 3.4.

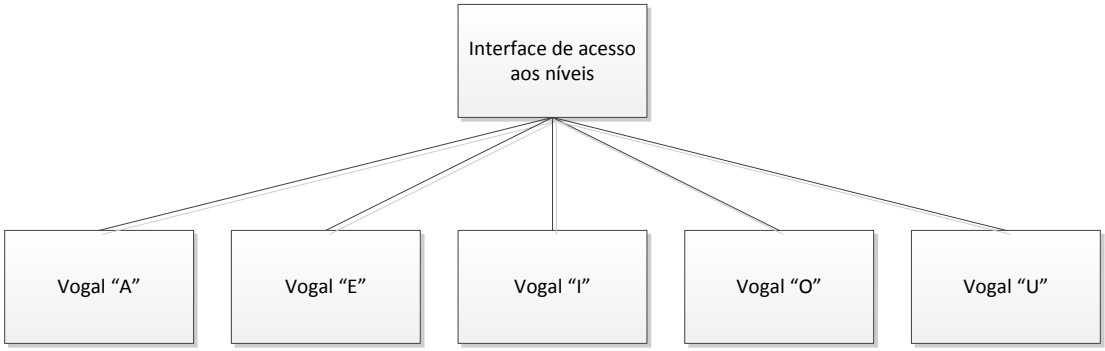


Figura 3.4: Esquema da interface de acesso aos níveis do jogo

Esta interface deverá permitir o acesso a qualquer vogal, iniciando assim o jogo para a vogal seleccionada.

Uma vez seleccionado o nível, iniciará o ciclo de jogo e, quando for terminado, voltará à interface de acesso aos níveis.

Etapa de um nível

Um nível está estruturado por várias etapas, sendo que cada uma caracteriza um som específico. Para se avançar entre etapas é necessário que o jogador acerte três vezes com sucesso e de seguida o jogo avança para a seguinte. No caso da letra "a", que contém duas etapas, serão mostradas seis imagens.

Na figura 3.5. pode-se observar o ciclo referente a cada etapa de uma letra. Este ciclo é repetido três vezes até prosseguir para a próxima etapa, concluindo-o e iniciando um novo.

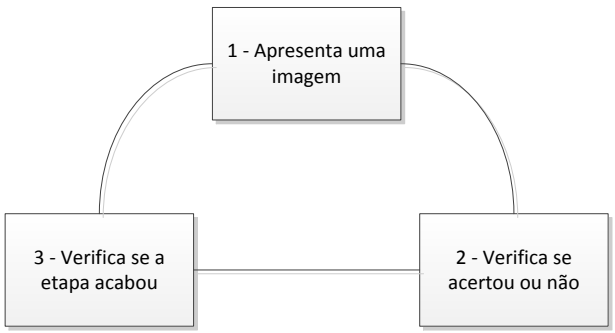


Figura 3.5: Ciclo de uma etapa

A resposta correcta será dada por alvos, tal como o centro Diferenças descreveu e conforme apresentado na figura 3.6. Cada alvo corresponde a uma sílaba da palavra que identifica a imagem que será apresentada.

As imagens podem ser apresentadas em mais do que uma vogal, como por exemplo a imagem “barrete” que se insere tanto na vogal “a” como na “e”.

Figura 3.6: Área de jogo com os alvos

3.2.4 Apresentação do Jogo

A apresentação do jogo é descrita pelo centro Diferenças e terá a funcionalidade referida no ponto 3.1.1., isto é, a capacidade de aceder a cada letra localizada no painel de fundo. Este painel irá variar dependendo se se está na Ilha Vogalis ou Consoantis.

O objectivo do painel de apresentação da Ilha Vogalis será de possibilitar o acesso às cinco vogais e iniciará o jogo para cada uma. A interface pode-se observar na figura 3.7. e possui como fundo o mapa da Ilha Vogalis.

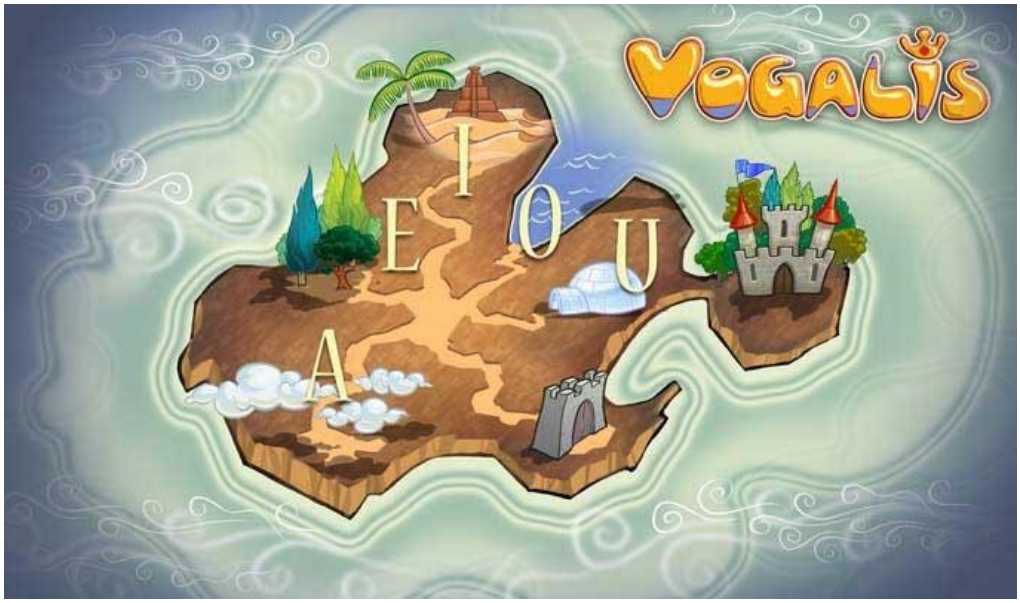


Figura 3.7: Interface com as vogais [27]

Cada vogal está posicionada num local em que se enquadre na história. A letra “e” está ao pé da floresta por a ajuda no jogo ser dada por uma égua.

A área de jogo vai ser diferente para cada vogal escolhida. Na figura 3.8. é possível ver o fundo para a vogal “i” e na figura 3.9. é possível ver o fundo para a vogal “u”.

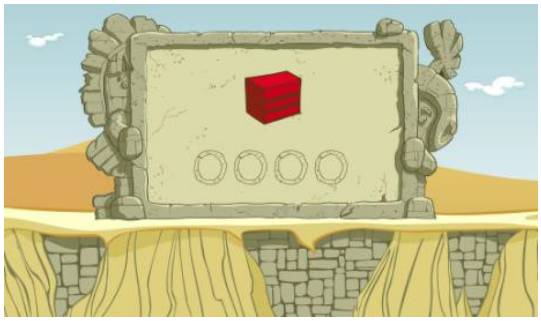


Figura 3.8: Fundo da vogal “i” [27]

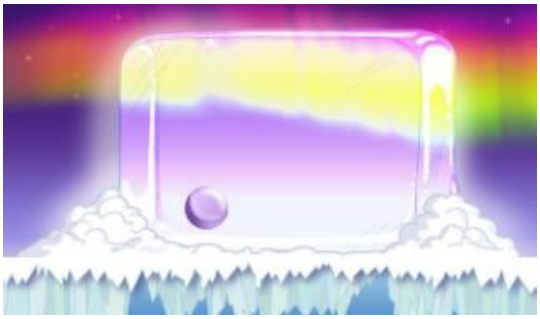


Figura 3.9: Fundo da vogal “u” [27]

A área de jogadores terá uma apresentação funcional em que sejam apresentadas todas as funcionalidades necessárias. As diversas possibilidades são ilustradas na figura 3.10.

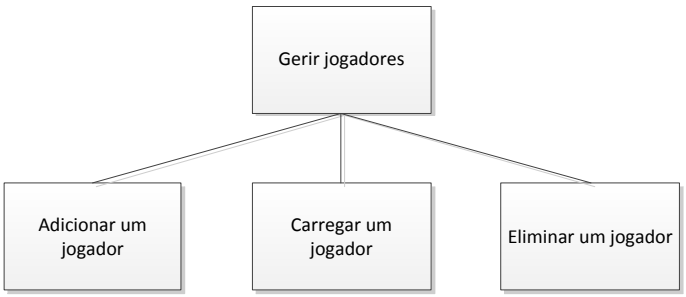


Figura 3.10: Possibilidades da área de jogadores

Estas funcionalidades devem permitir adicionar, carregar ou eliminar um jogador, sendo usada uma interface para escolher o que se pretende.

3.2.5 Ciclo do Jogo

O jogo deverá correr num ciclo constante de forma a percorrer todas as etapas necessárias para cada vogal, descritas no ponto 3.2.3., e no final da última etapa prossegue para o painel de apresentação. Este ciclo é descrito na figura 3.11., em que o jogador inicia o jogo, e após escolher uma vogal, o ciclo é iniciado.

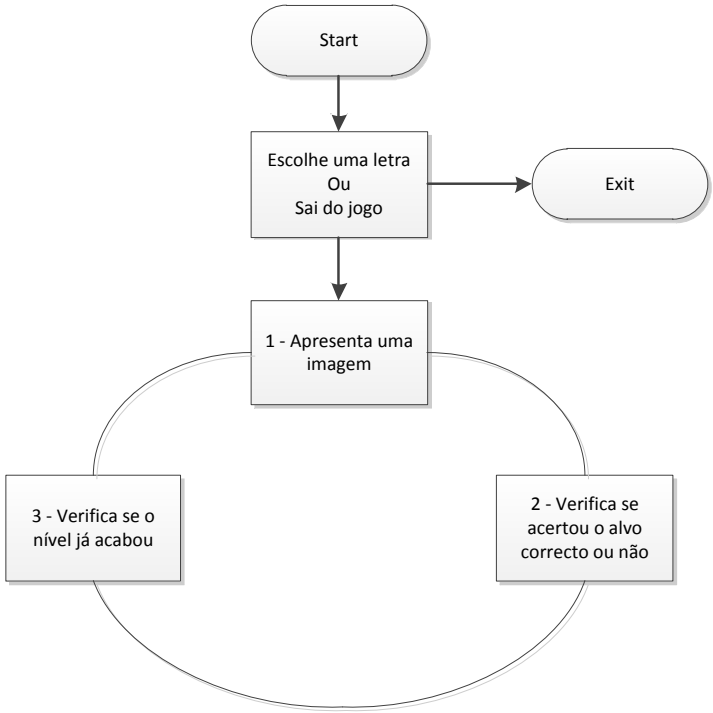


Figura 3.11: ciclo do jogo

Este ciclo é equivalente ao ciclo de uma etapa, embora mais completo. Deverá apresentar as várias vogais que se podem jogar ou sair do jogo. Uma vez iniciado o ciclo de uma etapa, o jogo deve proceder até ao fim do nível.

3.2.6 Gravação de Dados

Os dados deverão ser gravados de forma automática para que não seja bloqueada a execução do jogo. Estes podem ser estruturados de várias formas, dependendo da manipulação pretendida. Um exemplo de estrutura será a de um ficheiro XML, de forma a que se possa manipular os dados através do Excel ou de outro programa que permita efectuar cálculos.

A gravação deve acompanhar o ciclo de jogo e, quando algum resultado for definitivo, proceder-se à sua gravação. Para isso vai existir uma gravação em várias fases do ciclo, sendo descritas na figura 3.12. Estas fases são definidas na altura em que o jogo já tiver dados relevantes para gravar.

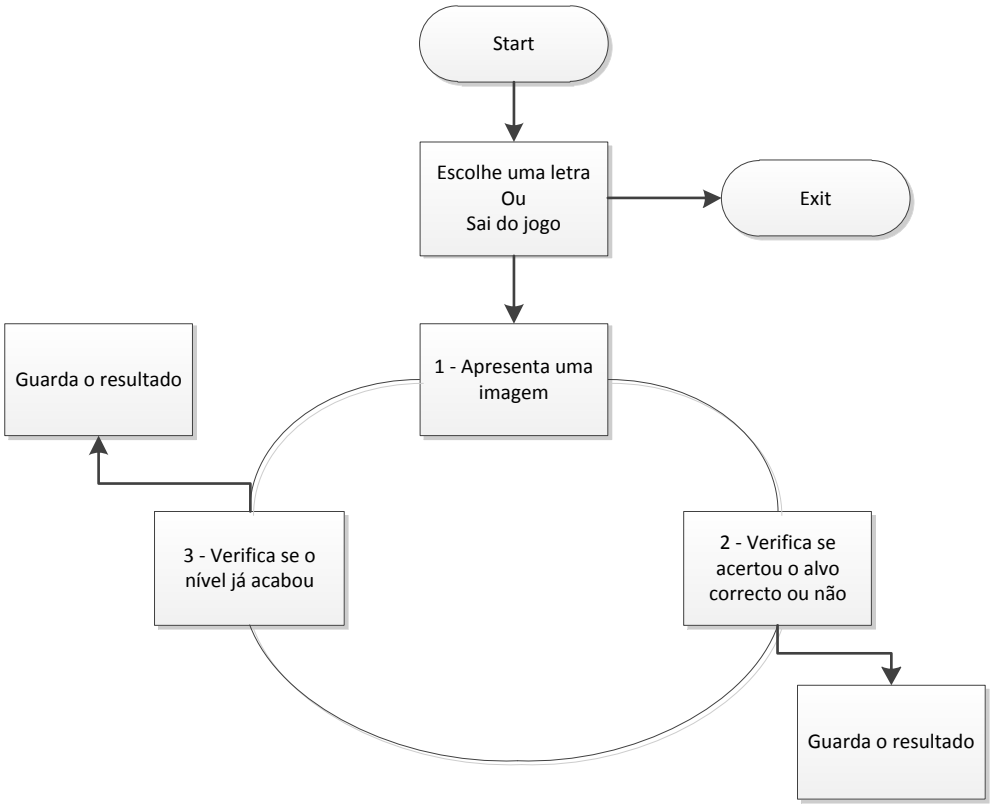


Figura 3.12: Fases de gravação no ciclo de jogo

As fases existirão depois de o jogador escolher um alvo e após o jogo avançar de etapa ou nível.

Esta gravação possibilitará o acesso a dados dos jogadores. Posteriormente, pode-se manipular os dados conforme o objectivo pretendido.

Os jogadores precisarão de ser gravados para que se possa continuar o jogo em várias sessões, não obrigando a que estes tenham de ser adicionados novamente. Ao gravar os jogadores, feito é possível um acompanhamento ao longo de todas as sessões que iniciar, ocorrendo esta gravação nas mesmas fases que as descritas na figura 3.11.

3.2.7 Estrutura do processo de jogo

É crucial que o modelo do processo de jogo que permita gerir os jogadores, jogar e produzir dados, alcance os objectivos descritos no ponto 3.1. Para isso terá uma ordem de execução de fases, descritas na figura 3.13, que permitirá o correcto funcionamento do jogo.

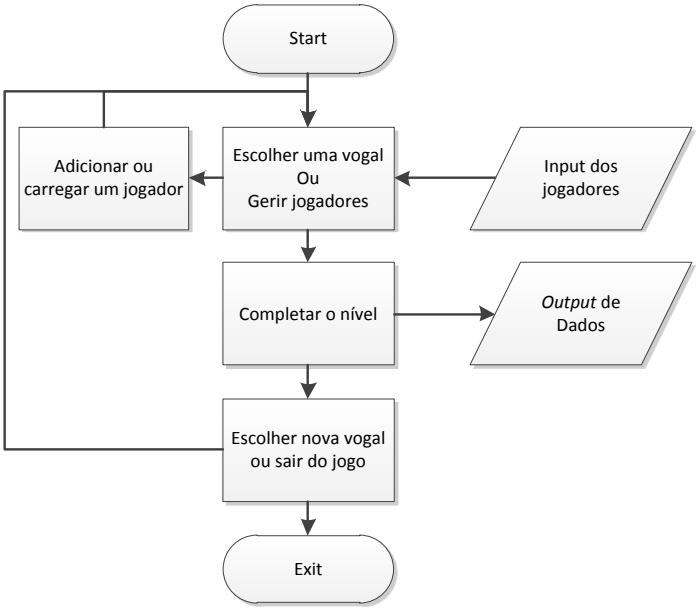


Figura 3.13: Processo do jogo

Este processo vai permitir que o jogo seja executado de forma cíclica cumprindo os requisitos funcionais. Na interface de apresentação deverá ser possível escolher uma vogal, proceder à gestão de jogadores e sair do jogo.

Os dados de *input* deverão ser carregados ao iniciar o jogo ou na interface de apresentação do mesmo e os dados de *output* deverão ser fornecidos após um nível ser completado. Ao completar um nível, o ciclo deverá iniciar, podendo-se sair, se assim se desejar, ou iniciar o jogo para uma vogal diferente.

4

Implementação e Validação

Neste capítulo serão apresentadas todas as metodologias para implementar o proposto no capítulo 3, sendo estas a organização do jogo, os painéis existentes e as suas funcionalidades, a organização dos dados de jogadores e de saída do jogo, os algoritmos usados para implementar as funcionalidades necessárias e o ciclo do jogo.

4.1 Ferramenta adoptada

O motor de jogo escolhido para implementar o jogo proposto é o Unity®. Uma vantagem que o Unity® apresenta, e esta é uma das razões para a sua escolha, é a linguagem de *script*, esta é em C# ou UnityScript, bastante parecido a JavaScript, ambas são largamente usadas fora do contexto do desenvolvimento de jogos e têm uma comunidade muito elevada. O Unreal Engine usa uma linguagem parecida com C++, e no UnrealScript a comunidade já é restrita para quem desenvolve jogos e, por conseguinte, inferior. O CryEngine apenas permite a implementação de funcionalidades de baixo nível em C++ ou Lua, o que não é ideal para os requisitos pretendidos. O GameMaker usa uma linguagem própria, o GML, e não favorece a implementação de funcionalidades fora do comum.

A existência de uma comunidade de grandes dimensões favorece a escolha tanto do Unity®, como do Unreal Engine ou do CryEngine, no entanto, a documentação do Unity® está mais organizada e alargada.

O sistema de *drag-and-drop* do Unity® ajuda bastante na construção das cenas face ao sistema do Unreal Engine ou da ferramenta *sandbox* do CryEngine. O GameMaker também possui um sistema de *drag-and-drop*.

O facto de o jogo não necessitar de físicas faz com que qualquer dos motores de jogo estudados no capítulo 2 se enquadre.

A implementação do jogo é em 2D e isso exclui o CryEngine, uma vez que este não possui motor de *rendering* para 2D, é possível criar o jogo em 3D e forçar o mapa a estar constantemente fixo em dois eixos, no entanto, são usados mais recursos por o mapa 3D precisar de ser renderizado. Tanto o Unity®, como o Unreal Engine ou o GameMaker possuem capacidade de criar jogos em 2D.

De modo a possibilitar uma futura integração da componente estática com a componente dinâmica, o Unity® foi a ferramenta escolhida, por esta ter sido utilizada no Game Wizard, implementando a componente dinâmica do mesmo.

Tendo em atenção as comparações mencionadas acima acrescidas do facto de esta componente seguir uma outra já implementada em Unity®, optou-se por escolher o Unity® para implementar o jogo.

Todos os algoritmos e estruturas de dados que não dependem de nenhum objecto de jogo foram implementados e testados em C# no Visual Studio, uma vez que o Unity® consegue correr *scripts* nesta linguagem.

4.1.1 Estruturas de dados

Os dados são gravados em várias estruturas de classes semelhantes, no entanto, existem alguns detalhes que impossibilitam o uso da mesma estrutura para todos. Foram definidos três tipos distintos de estruturas, uma para o ficheiro de *output*, uma para o ficheiro de *input* e outra para os jogadores. Os diagramas UML foram criados usando o Visual Studio por este construir o diagrama consoante as dependências de cada classe, apresentando os atributos, as relações e os métodos das mesmas.

Ficheiro de *Output*

O ficheiro de *output* é criado ou substituído, se não existir ou se já existir respectivamente, sempre que o jogo grava uma nova informação referente a qualquer jogador e os seus dados são apresentados num diagrama UML (figura 4.1.). Esta estrutura é específica para o ficheiro de *output* por ser necessária a introdução de *tags* nas classes de modo a que a gravação cumpra a

organização pretendida para o ficheiro XML. A estrutura do ficheiro de *output* é preenchida copiando os dados dos jogadores para si e de seguida executando a gravação do ficheiro XML. Este ficheiro tem o nome de Dados.

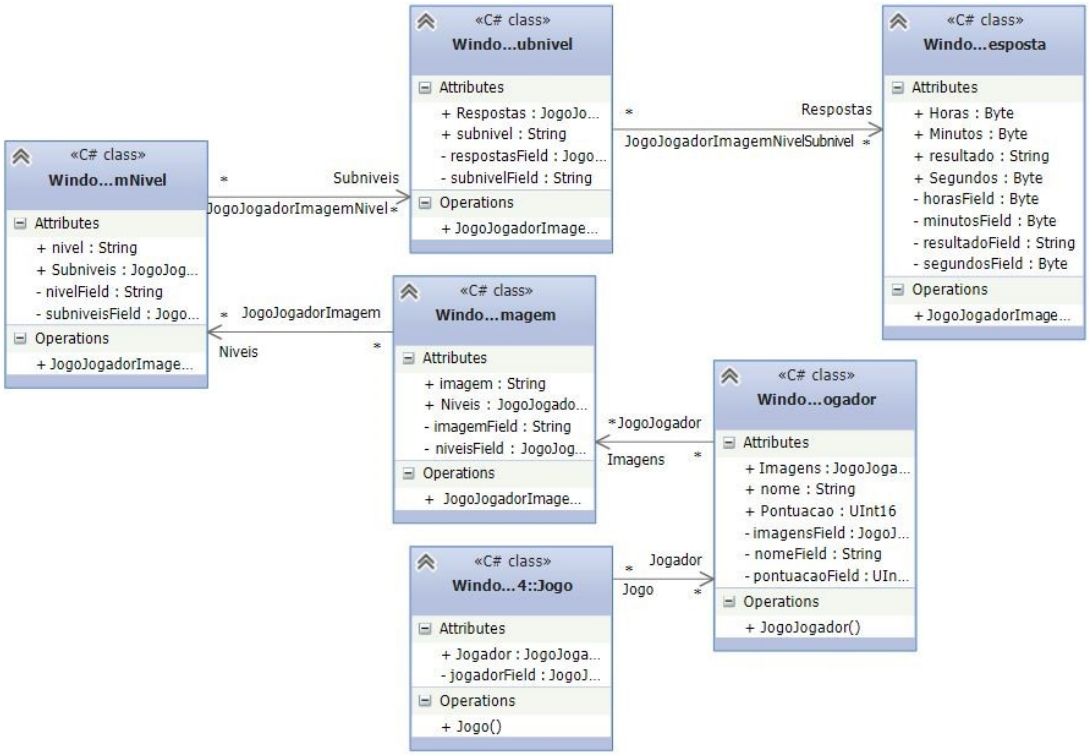


Figura 4.1: Diagrama UML das classes do XML de *output*

O ficheiro XML fica com a seguinte estrutura ilustrada na figura 4.2. Como se pode ver, são gravados vários dados sobre um jogador e neste caso só foi mostrada uma imagem, a “erva” no nível “e”.

```
<Jogador nome="José">
    <Pontuação>1770</Pontuação>
    <Imagens>
        <Imagem imagem="erva">
            <Níveis>
                <Nível nível="E">
                    <Subníveis>
                        <Subnível subnível="eeee">
                            <Respostas>
                                <Resposta resultado="Acertou">
                                <Horas>0</Horas>
                                <Minutos>0</Minutos>
                                <Segundos>3</Segundos>
                                </Resposta>
                            </Respostas>
                        </Subnível>
                    </Subníveis>
                </Nível>
            </Níveis>
        </Imagem>
    </Imagens>
</Jogador>
```

Figura 4.2: Ficheiro XML de *output*

Com esta estrutura é possível guardar o nome do jogador, as imagens apresentadas, em que níveis e etapas foram mostradas, o resultado do alvo escolhido e o tempo demorado desde que a imagem foi mostrada até um alvo ser escolhido.

Dados dos jogadores

A estrutura referente aos jogadores contem umas *tags* específicas para poder ser serializado com o objectivo de não se poder alterar manualmente o seu conteúdo. Esta estrutura é a usada para guardar os jogadores, bem como todos os respectivos dados, e é a necessária para copiar para o ficheiro de *output*. Na figura 4.3. pode-se observar o diagrama UML dos jogadores. A serialização é feita usando o *System Namespace* presente na linguagem C# produzindo um ficheiro com o nome de “SavedPlayers.rdf”.

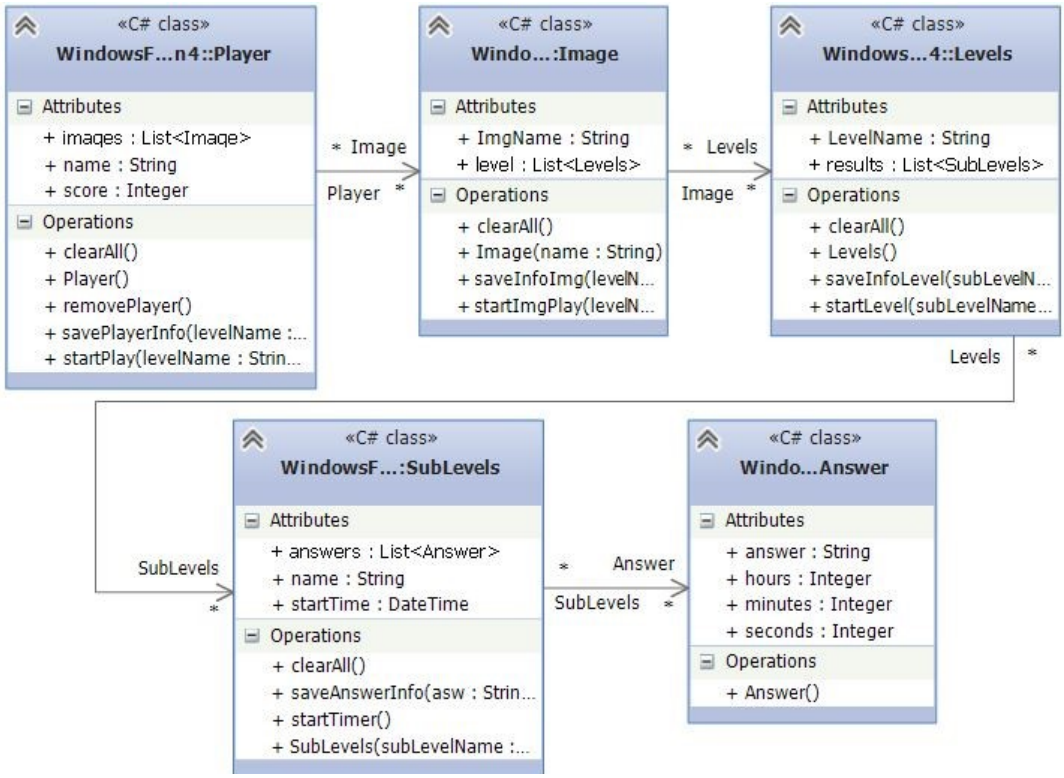


Figura 4.3: Diagrama UML das classes dos jogadores

Ficheiro de *input*

O ficheiro de *input* é responsável por descrever quais os nomes das imagens que serão apresentadas, em que nível podem sê-lo e em que etapa é que têm alvos correctos. Está estruturado no formato XML e é carregado no início do jogo tendo o nome de “Imagens”. Na figura 4.4., está representada a estrutura do mesmo e neste caso o nome da imagem é “espada”, que pode ser apresentada no nível “a” e no “e”. Neste caso, e para o nível “a”, a imagem pode ser mostrada nas duas etapas. Para o nível “e” pode ser apresentada em uma. O campo de “subnivel1” indica qual a etapa onde a primeira vogal “a” se insere, o campo “subnivel2” indica a segunda vogal e sucessivamente.

```
<img name = "espada">
    <niveis>
        <nivel name = "E">
            <subnivel1>3</subnivel1>
            <subnivel2>0</subnivel2>
            <subnivel3>0</subnivel3>
        </nivel>
        <nivel name = "A">
            <subnivel1>1</subnivel1>
            <subnivel2>2</subnivel2>
            <subnivel3>0</subnivel3>
        </nivel>
        <nivel name = "I">
            <subnivel1>0</subnivel1>
            <subnivel2>0</subnivel2>
            <subnivel3>0</subnivel3>
        </nivel>
    </niveis>
</img>
```

Figura 4.4: Ficheiro XML de uma palavra

Este ficheiro ao ser lido pelo jogo é carregado para uma estrutura de dados e desta forma evita que sejam carregadas as várias imagens existentes, poupando o consumo de recursos. O diagrama UML dos dados encontra-se ilustrado na figura 4.5.

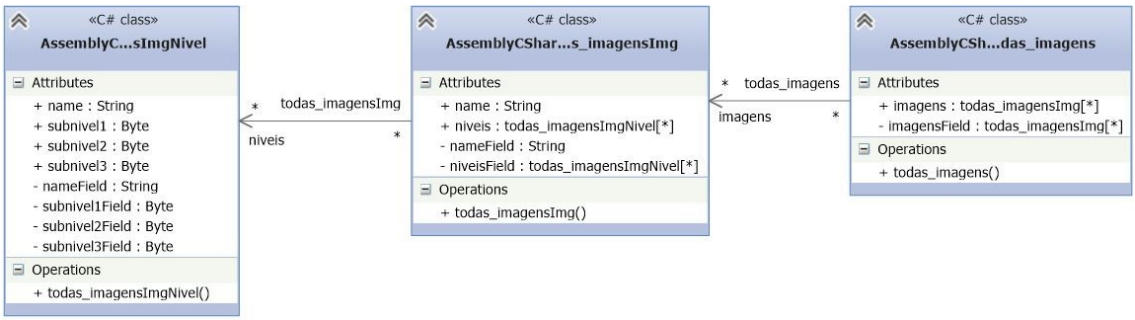


Figura 4.5: Diagrama UML das classes do XML de *input*

4.1.2 Painéis do Jogo

O jogo está estruturado em três painéis, o menu, a área de jogadores e a área de jogo, e é possível navegar por estas três como se desejar. Na figura 4.6. está ilustrada esta navegação.

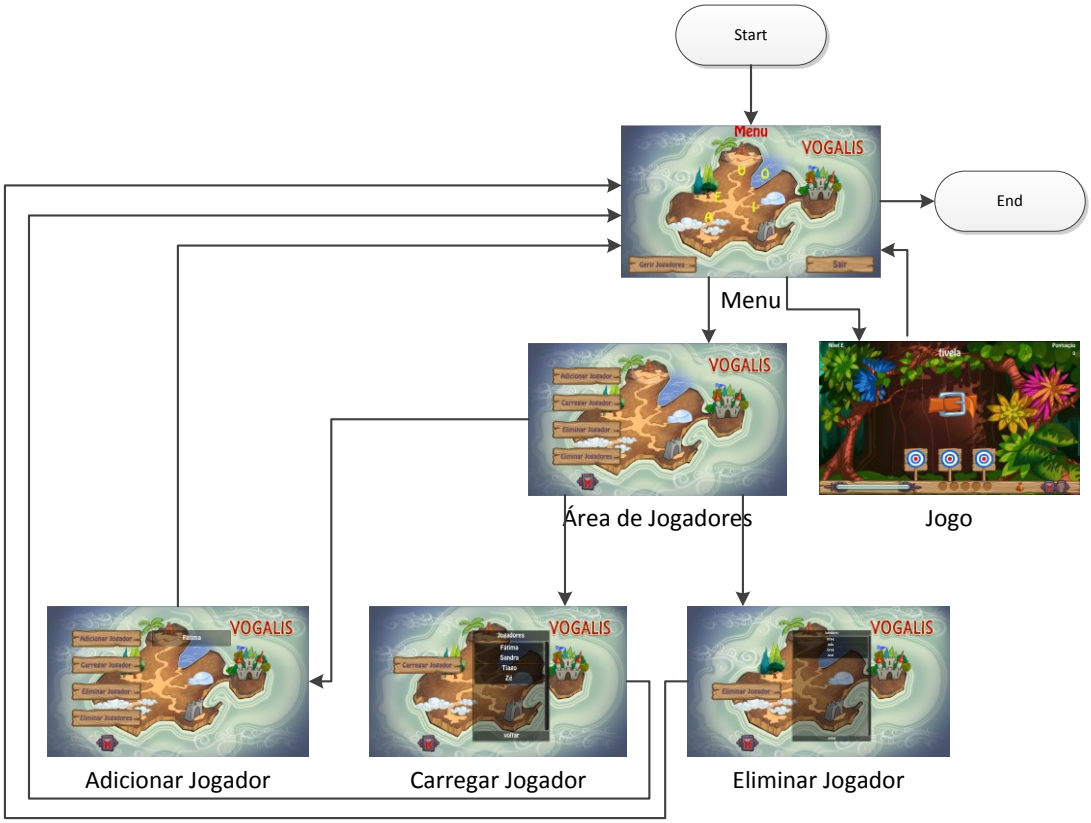


Figura 4.6: Fluxograma de navegação do jogo

Menu

Conforme o proposto no ponto 3.2.4., cada vogal está posicionada no mesmo local, existindo dois botões adicionais, o da navegação para a área de jogadores e o de sair do jogo. É no menu que o jogo inicia e pode-se escolher uma vogal para se jogar, proceder ao registo ou carregamento de um jogador através da navegação à área de jogadores ou sair do jogo. A figura 4.7. apresenta o menu implementado.

São apresentadas cinco letras a amarelo e, quando se passa o rato por cima, a letra muda para encarnado.

Figura 4.7: Menu principal do jogo

É no menu que todas as condições iniciais são reiniciadas, de forma a qualquer vogal estar apta a jogar, sendo também carregado na primeira vez que corre o ficheiro de jogadores, bem como um ficheiro que contém todos os nomes das imagens que podem ser apresentadas em qualquer das vogais.

Área de Jogadores

A área de jogadores permite que um jogador seja gravado no jogo e que este consiga gravar os dados referentes à sua utilização. É possível adicionar, carregar e eliminar jogadores, conforme se pode verificar na figura 4.8.

Para se adicionar um jogador carrega-se no botão de adicionar e uma caixa a preto aparecerá na qual se escreve o nome do jogador novo pressionando a tecla “Enter” para concluir a operação. Se o jogador não existir, será introduzido nos dados do jogo e carregado como jogador activo. De seguida o jogo prossegue para o menu, apresentando o nome do jogador no canto superior direito. Se o jogador já existir a caixa desaparece e o jogo não prossegue para o menu, não carregando nenhum jogador.

Figura 4.8: Área de jogadores do jogo

O carregamento e a eliminação de um jogador apresentam uma estrutura igual, apesar de desempenharem funções distintas, ou seja, quando se prime no botão de carregar jogador ou de eliminar jogador, é exibida uma lista de todos os jogadores existentes no jogo, como é mostrado na figura 4.9. Através dessa lista e carregando em cima do nome pretendido é desempenhada a sua funcionalidade. De seguida, o jogo prossegue para o menu exibindo no canto superior direito o nome do jogador activo se este foi carregado ou nenhum caso tenha sido eliminado o activo. Se se quiser eliminar um jogador, este é retirado da lista e a vista em lista não desaparece para o caso de se querer retirar outro.

Existe mais um botão nesta lista, o de voltar. Este botão permite ao jogador voltar à opção anterior, ou seja, se se carregou em eliminar um jogador e não era essa a opção pretendida, através do botão voltar o jogo volta a apresentar os quatro botões da área de jogadores.

O último botão é o de eliminar todos os jogadores. Este é bastante forte pois não pede que se confirme a opção de eliminar todos os jogadores. Uma vez carregado, elimina tudo e o jogo prossegue para o menu.

Em todas as opções possíveis dentro da área de jogadores existe um botão em forma de castelo que representa a ida para o menu. Se se carregar neste botão, é tudo cancelado e o jogo prossegue para o menu.

Figura 4.9: Lista de jogadores existentes

Área de Jogo

Esta cena é responsável por apresentar o jogo em si, ou seja, é onde as imagens são apresentadas, os sons ouvidos e a pontuação mostrada. Existem alguns botões presentes, sendo o do castelo o que prossegue para o menu, cancelando o estado do jogo.

Os cinco círculos, a bolsa e o sinal de interrogação estão presentes para corroborar o já implementado na componente dinâmica do jogo, mas não produzem qualquer acção por não se enquadrarem na implementação da parte estática.

No canto superior direito é apresentada a pontuação total do jogador que é alterada à medida que o jogador acerta em mais imagens, sendo a pontuação atribuída escrita no ponto 3.3.

Ao entrar na área de jogo, é explicado através de um ficheiro de som o objectivo da etapa referente à vogal escolhida, e de seguida é apresentada uma imagem escolhida de forma aleatória entre as escolhidas pelo centro Diferenças e uns alvos posicionados abaixo da imagem. O número de alvos é dependente da palavra que caracteriza a imagem mostrada, sendo o seu número igual ao número de sílabas da palavra, por exemplo, se for mostrada a palavra “caneca”, tendo a separação silábica de “ca-ne-ca”, irão aparecer três alvos, um para cada sílaba. O jogador terá de acertar na sílaba que contém o som que foi pedido no início da etapa e caso acerte é produzido um som de aprovação, se falhar é produzido um som de reprovação. De notar que o som de reprovação não tem nenhuma conotação negativa, mas incentiva a continuar a tentar.

Figura 4.10: Área de Jogo

Como é possível observar na figura 4.10., a imagem mostrada é “elmo”, que possui a separação silábica de “el-mo” e portanto são apresentados dois alvos. Neste caso o som pedido é o som do “e” aberto, o “é”. Isto pode ser confirmado por no canto superior esquerdo se apresentar o nível escolhido e neste caso é o nível “e”. A barra posicionada no canto inferior esquerdo vai crescer à medida que se avança nas etapas, crescendo da esquerda para a direita, por cada imagem mostrada e ultrapassada com sucesso, crescendo numa proporção igual ao número mínimo de imagens necessárias para ultrapassar a etapa. Neste caso o nível “e” tem três etapas e cada etapa mostra três imagens o que totaliza nove imagens mostradas, portanto, a barra vai crescer $\frac{1}{9}$ sempre que uma imagem for respondida com sucesso. Como neste caso a barra se encontra vazia, conclui-se que o som pedido é o “é”, tornando o primeiro alvo correcto e o segundo errado. Na figura 4.11. pode-se observar a sequência de um nível.

Neste caso o nível é o “e”, mas esta sequência é igual para qualquer nível que se jogue, alterando as imagens que são mostradas, os alvos correctos e o texto no canto superior esquerdo.

Quando o jogador acerta num alvo correcto é-lhe atribuída uma pontuação, que irá ser somada à pontuação total já acumulada. Cada resposta correcta atribui 100 pontos e por cada resposta errada são retirados tantos pontos como o número total de imagens existentes na etapa em jogo. Por exemplo, se houver 20 imagens para mostrar e o jogador falhar na primeira, já só irão ser mostradas 19 imagens e a pontuação atribuída por acertar na imagem seguinte será de $\frac{19}{20} \times 100$, arredondado às unidades, neste caso ganhará 95 pontos. Quantas mais vezes seguidas o jogador falhar menos pontos vai receber, se acertar uma resposta a seguinte já irá dar 100 pontos. No caso de o jogador falhar em todas as imagens, o jogo irá mostrar novamente as

mesmas imagens de forma aleatória e serão atribuídos 100 pontos novamente à primeira imagem.

Uma vez que complete o total de etapas descritas no ponto 3.2.2. para a vogal escolhida, o jogo avança para o menu e completa assim uma vogal.

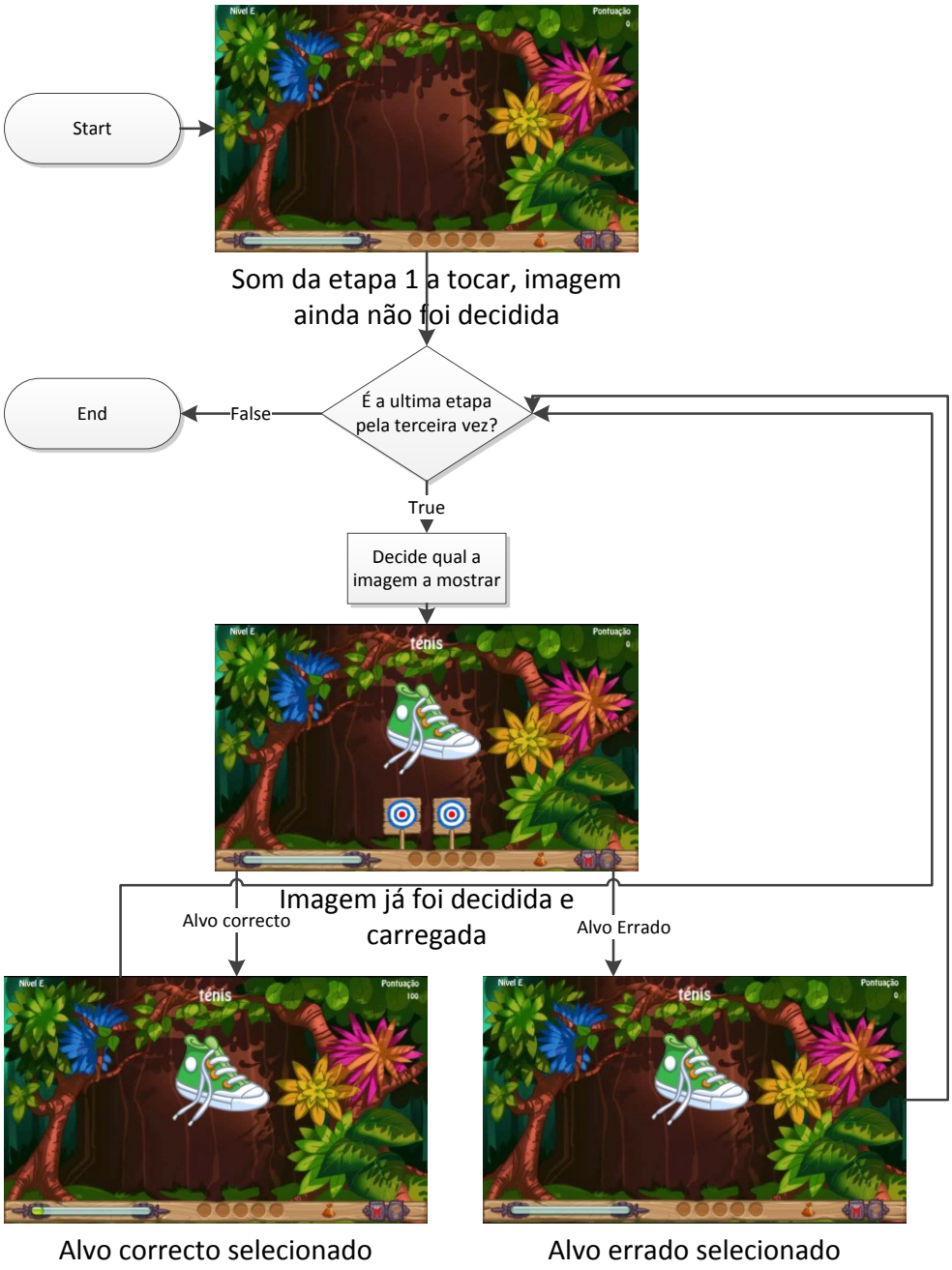


Figura 4.11: Fluxograma da sequência de um nível

4.2 Algoritmos e Processos

Para que o jogo consiga criar o número de alvos necessários e que estes sejam correctos ou errados foram implementados dois algoritmos. Estes capacitam o jogo de separar silabicamente qualquer imagem que seja apresentada criando assim o número de alvos necessários.

A gravação de dados é feita em fases deferentes do jogo de modo a não comprometer a sua execução.

4.2.1 Separação silábica

A separação silábica é feita através de um algoritmo que se encontra em [28] e foi alterado de forma a produzir a melhor separação pedida. O algoritmo inicial conseguia avaliar a palavra e separava correctamente as consoantes de acordo com a língua portuguesa, mas não conseguia separar as vogais e portanto não produzia uma separação silábica correcta. Quando foi alterado impôs-se como condição que o algoritmo conseguisse separar silabicamente todas as palavras, incluindo algumas excepções presentes na língua portuguesa. É capaz de separar correctamente qualquer palavra da língua portuguesa com uma condição, não pode conter mais de 15 sílabas.

O algoritmo assenta na base de avaliar conjuntos de letras que podem ou não estar juntos. No caso da palavra “chuva” o algoritmo permite que as consoantes “c” e “h” estejam juntas e vai separar a palavra em “chu-va”. Para o caso da palavra “alcunha” o algoritmo não vai permitir que a consoante “l” e “c” estejam juntas, já para as consoantes “n” e “h” vai permitir que estejam juntas, resultando a separação de “al-cu-nha”. Na figura 4.12. pode-se observar o seu fluxograma simplificado.

As vogais são separadas sempre que houver duas iguais seguidas, como no caso de “compreender”, que vai resultar em “com-pre-en-der” e, se não forem iguais, vai avaliar se estas podem estar juntas como no caso da palavra “água” de que resulta a separação “á-gua”.

Esta detecção é feita ao percorrer letra a letra da palavra, parando quando uma vogal é encontrada e comparando com as letras que estão até duas posições atrás e uma à frente, contando e gravando quantas letras correspondem a uma sílaba. No Anexo 1 encontra-se o fluxograma completo do algoritmo.

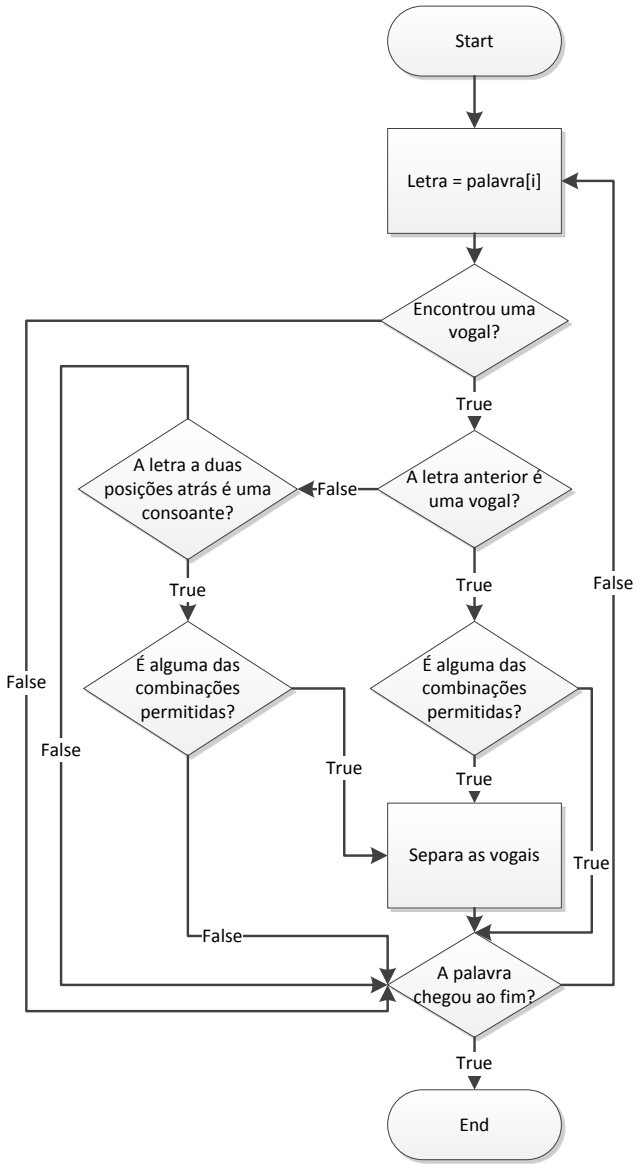


Figura 4.12: Fluxograma simplificado da separação silábica

Testando a detecção de sílabas executando o algoritmo num projecto realizado em Visual Studio e programado em separado do Unity®, é possível realizar a separação silábica de qualquer palavra. Na figura 4.13. é apresentada a *Graphic User Interface*, ou GUI, desenhada para o efeito, e na figura 4.14. é possível ver o resultado obtido ao separar silabicamente a palavra "otorrinolaringologista".

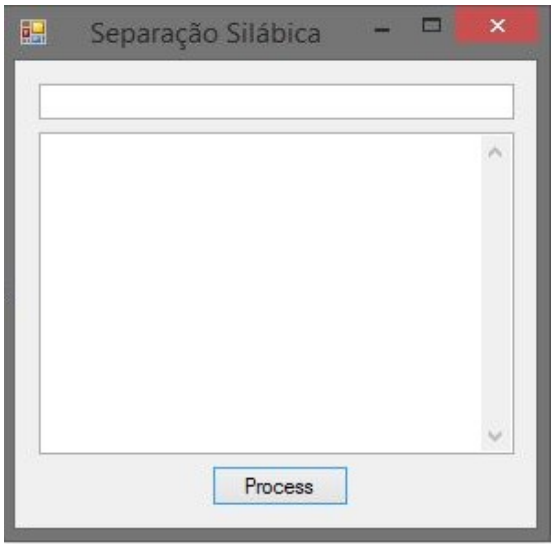


Figura 4.13: GUI da separação silábica

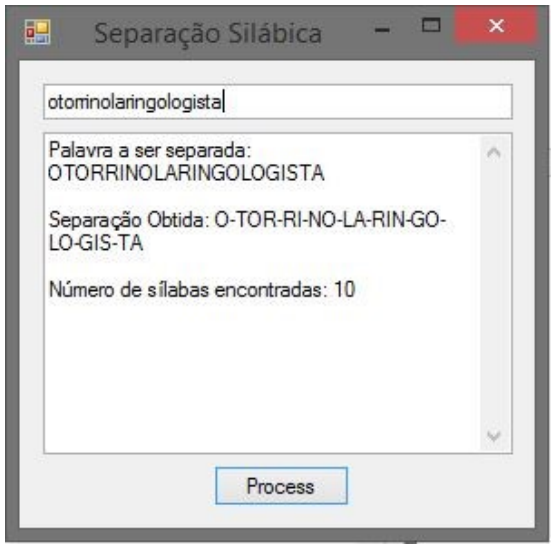


Figura 4.14: Resultado da separação silábica

No primeiro rectângulo posicionado na parte superior da GUI escreve-se a palavra que se quer separar e de seguida carrega-se no botão “Process” ou simplesmente na tecla “Enter” e no rectângulo em baixo aparece o resultado bem como algumas informações adicionais.

A separação silábica aplicada na figura 4.4. resulta em 10 sílabas, separando a palavra “otorrinolaringologista” em “o-tor-ri-no-la-rin-go-lo-gis-ta”. Este resultado foi comparado com o obtido em dois dicionários *online*, o da Porto Editora [29] e o do iLteC [30], Instituto de Linguística Teórica e Computacional, que confirmaram a correcta separação silábica.

Este algoritmo é o mesmo presente no jogo e verifica-se que consegue efectuar separações silábicas correctas, fornecendo à criação dos alvos tudo o que esta necessita para criar o mesmo número.

Todas as palavras contidas no jogo foram testadas separadamente nesta GUI de forma a garantir a correcta separação antes de este ser implementado no Unity®.

4.2.2 Criação e detecção de alvos

A criação dos alvos depende do número de sílabas presentes na palavra que representa a imagem mostrada, tal como é referido no ponto 3.4.3. Tomando como exemplo a palavra “fivela”, apresentada na imagem 4.15., e a palavra “exército”, apresentada na figura 4.16., pode--se ver que o número de alvos foi o correcto e coincidente com o número de sílabas.

Figura 4.15: Alvos criados para a palavra “fivela”

Figura 4.16: Alvos criados para a palavra “exército”

A separação silábica da palavra “fivela” resulta em “fi-ve-la”, que conta com três sílabas e como se pode verificar na figura 4.15, existem três alvos. Para a palavra “exército” a separação silábica é “e-xér-ci-to”, o que produz quatro sílabas, consequentemente existem quatro alvos na figura 4.16.

O posicionamento dos alvos é feito de forma automática e depende do número de alvos a colocar. Isto significa que nas palavras com um número de sílabas ímpar o posicionamento difere do das que possuem um número de sílabas par.

Para um número par é aplicada a equação $X = \frac{número\ de\ sílabas}{2}$ em que X é o número de alvos para cada lado. De seguida são posicionados por ordem crescente e da esquerda para a direita, ficando todos à mesma distância relativa entre eles.

Para um número impar é aplicada a equação $X = \frac{número\ de\ sílabas-1}{2}$ em que X tem o mesmo significado que para um número par. De seguida os alvos são posicionados da mesma forma que para um número par, excepto que existe um alvo central, ficando todos à mesma posição relativa entre eles. Na figura 4.17. pode-se observar um fluxograma que descreve a criação dos alvos. No Anexo 2 encontra-se o fluxograma completo da criação de alvos.

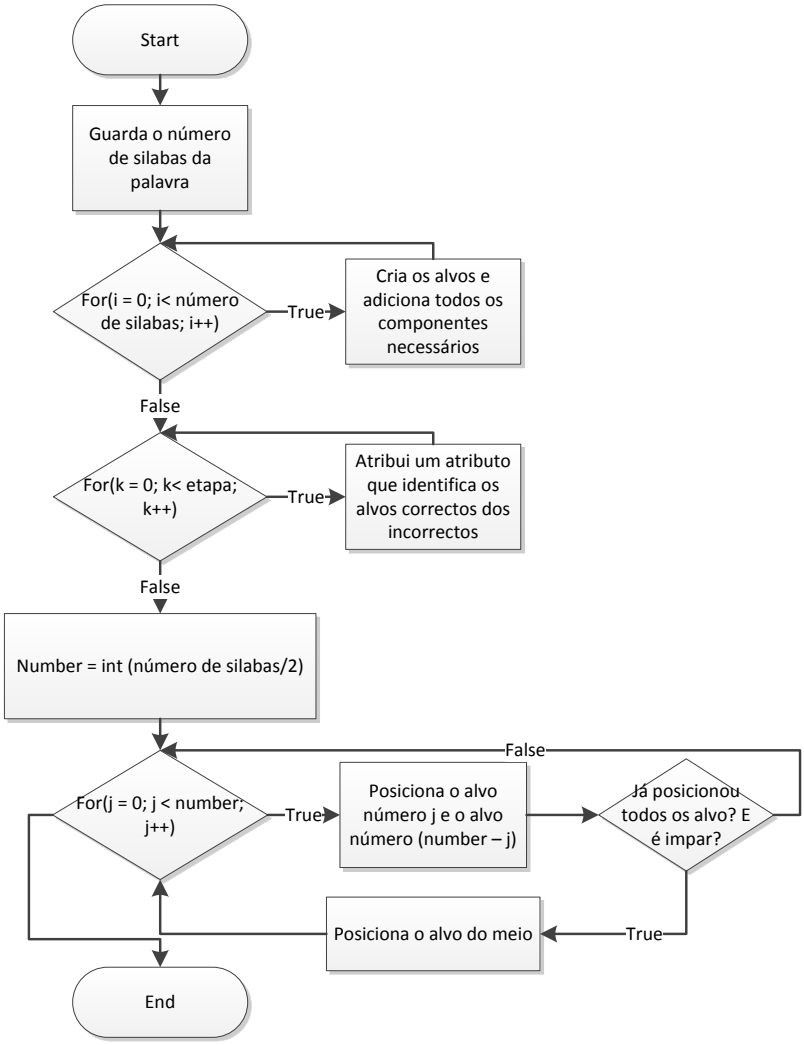


Figura 4.17: Fluxograma da criação dos alvos

4.2.3 Gravação de dados

A gravação de dados é processada em várias fases:

- Quando o jogador é criado é introduzido o seu nome para os dados dos jogadores, ficando apenas com esse campo preenchido e com a pontuação a 0;
- Quando o jogador escolhe uma letra, iniciando um nível e uma etapa, a gravação ocorre quando este escolhe um alvo. Entretanto, e quando a imagem é apresentada, é iniciado um relógio para contabilizar o tempo que este demorou a escolher um alvo;
- Após o alvo ser escolhido são gravados vários dados, a pontuação, o nível, a etapa, a imagem apresentada, o resultado do alvo e o tempo que demorou a escolher o alvo, ficando toda a informação nos dados do jogador;
- Quando uma etapa é concluída, o que implica que três imagens sejam concluídas com sucesso, os dados dos jogadores são copiados para os dados do XML de *output*, substituindo, se já existir, ou criando, se não existir, o ficheiro.

  Na figura 4.2. é possível observar a organização dos dados no ficheiro XML.

De forma a testar a capacidade do jogo foram criados quatro jogadores e iniciou-se o jogo em níveis iguais, de forma a verificar se este consegue diferenciar os mesmos e a seguir, de forma independente, a utilização de cada um.

Ao abrir o ficheiro de “Dados.xml” no Excel foi possível esquematizar os dados. Na tabela 4.1. são apresentados os dados obtidos e, como se pode ver, são guardados os descritos no ponto 4.1.1.

Estes dados podem ser usados para dois objectivos, o uso da pessoa que acompanha a criança, a fim de perceber que dificuldades este tem, ou por alguém que pretenda avaliar o desempenho do jogo, a fim de o actualizar, ou calibrar de acordo com o necessário para os jogadores.

Tabela 4.1: Resultados obtidos pelo jogador "José"

Nome	Pontuação	Imagem	Nível	Etapa	Resultado	Horas	Minutos	Segundos
José	838	régua	E	eeee	Acertou	0	0	3
José	838	anel	E	eeee	Falhou	0	0	6
José	838	boné	E	eeee	Acertou	0	0	5
José	838	ténis	E	eeee	Falhou	0	0	4
José	838	neve	E	eeee	Acertou	0	0	4
José	838	amuleto	E	ee	Falhou	0	0	5
José	838	camelo	E	ee	Acertou	0	0	3
José	838	gaveta	E	ee	Falhou	0	0	3
José	838	pêra	E	ee	Acertou	0	0	5
José	838	peso	E	ee	Falhou	0	0	4
José	838	ema	E	ee	Acertou	0	0	3
José	838	pote	E	e	Falhou	0	0	5
José	838	tecido	E	e	Acertou	0	0	2
José	838	esquilo	E	e	Falhou	0	0	2
José	838	nenúfar	E	e	Acertou	0	0	2
José	838	leque	E	e	Falhou	0	0	3
José	838	alce	E	e	Acertou	0	0	2

Na coluna da etapa observam-se três tipos diferentes, o "eeee", o "ee", o "e", cada um sendo equivalente a uma etapa. A primeira etapa que corresponde ao som "é", é equivalente ao "eeee", a segunda etapa que corresponde ao som "ê", é equivalente ao "ee" e, por fim, a última etapa que corresponde ao som "e", sendo o som diferente de "i", é equivalente ao "e". Observa-se também o tempo demorado em cada imagem bem como o seu resultado. Exemplificando com a segunda linha, a imagem que foi mostrada foi "anel" para o nível "e" na primeira etapa, som "é", no qual o jogador falhou no alvo correcto e demorou 6 segundos a escolher um alvo. Finalizou o nível com 838 pontos, derivados de um rácio de sucesso de 52,9%, acertando nove imagens e falhando 8. A média de tempo demorado foi de 3,6 segundos ao longo de todas as imagens apresentadas.

Com estes resultados é possível realizar vários estudos e análises estatísticas, bem como a detecção das maiores dificuldades dos jogadores.

4.3 Validação

De forma a perceber se o jogo implementado cumpre os objectivos pretendidos, é necessário validar o seu uso através dos dados que este produz. É possível realizar todo o tipo de operações estatísticas aos dados, com a finalidade de perceber tendências, dificuldades e padrões. Estas operações podem ser aplicadas quer ao jogador quer ao jogo em si. É possível verificar se uma imagem é apresentada de forma clara e se o objectivo é devidamente explicado ao analisar os dados.

Construindo alguns gráficos com base no ficheiro de *output,* consegue verificar-se algumas características dos jogadores.

Um gráfico do rácio de sucesso geral consegue-nos caracterizar a dificuldade geral do jogo e se este for muito baixo pode haver um problema com a dificuldade do jogo em si e não dos jogadores. Já um rácio elevado pode demonstrar uma dificuldade muito baixa por parte do jogo.

O tempo de resposta por cada imagem de um jogador consegue diferenciar os que jogaram de forma aleatória e os que se aplicaram no jogo. Se esta aleatoriedade for alta para todos os jogadores pode implicar uma falta de capacidade de motivação e interesse demonstrado pelo jogo. Se for baixa pode significar que um jogador específico não se interessa ou não se motiva pelo jogo.

Uma imagem pode aparecer repetida várias vezes no jogo, embora seja a jogadores diferentes, num nível diferente, numa etapa diferente ou se o jogador jogar o mesmo nível duas vezes. Ao se verificar, para um jogador, quantas vezes cada imagem lhe foi apresentada é possível verificar a sua evolução, detectar uma dificuldade encontrada e agir de forma adequada. Também se pode verificar a aleatoriedade das imagens ao se construir um gráfico com quantas vezes as imagens foram apresentadas ao longo de um nível. Isto permite-nos avaliar a correcta aleatoriedade do jogo e, caso não esteja como foi definido, corrigir.

Através da tabela 4.1. e de mais três tabelas, correspondentes aos três jogadores em falta, o “Filipe, o “João” e o “Jorge” foram realizados os gráficos de rácio de sucesso geral, um gráfico de tempos em cada resposta para o jogador “José” e um gráfico de palavras apresentadas mais do que uma vez.

O gráfico do rácio de sucesso pode ser visualizado na figura 4.18. Verifica-se que o rácio é positivo, tendo 56,25% de respostas acertadas contra 43,75% de falhadas.

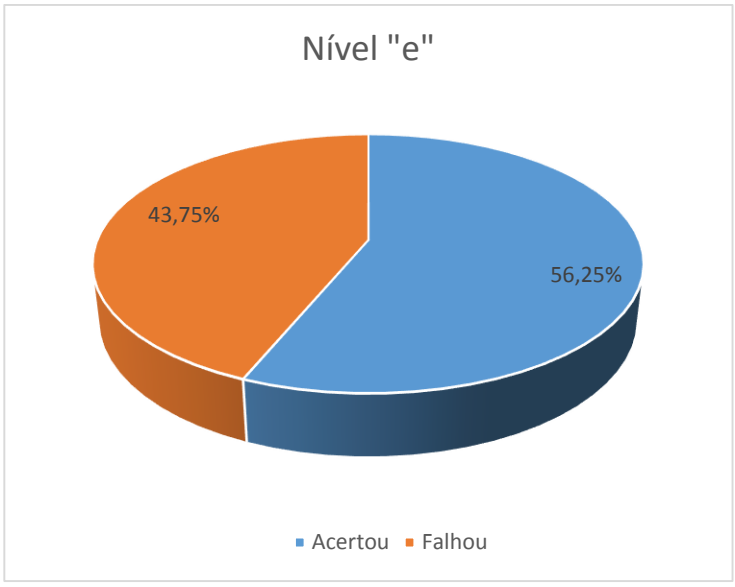


Figura 4.18: Rácio de sucesso dos quatro jogadores

Com este resultado verifica-se que a dificuldade do jogo se adequa aos quatro jogadores, não sendo nem muito baixa nem muito alta.

O tempo demorado em cada resposta permite-nos analisar se o jogador respondeu de forma aleatória, e um tempo muito rápido e constante em cada resposta normalmente indicam a aleatoriedade e a falta de interesse demonstrada pelo jogador. No gráfico apresentado na figura 4.19. é realçável a aleatoriedade por parte do jogador. O tempo demorado é constante, havendo 5 imagens no qual demorou cinco ou mais segundos a responder. De notar que três segundos para uma resposta não é o expectável para uma criança que tem dificuldades.

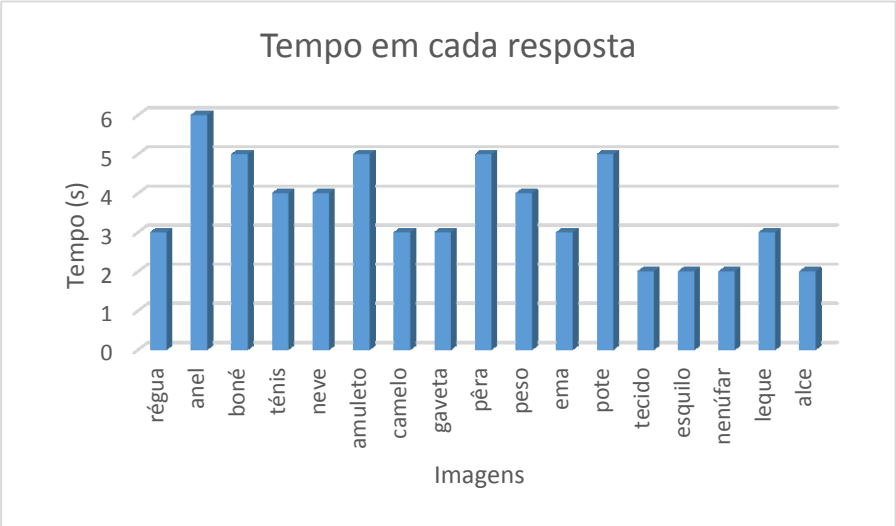


Figura 4.19: Tempo demorado em cada resposta do jogador "José"

Essa aleatoriedade pode ser devida à falta de interesse do jogador ou à falta de capacidade de motivação por parte do jogo. Com este resultado é possível agir de forma adequada junto do jogador a fim de perceber qual pode ser a razão para estes resultados.

Na figura 4.20. é apresentado o gráfico com as imagens que foram apresentadas mais do que uma vez. No total foram apresentadas 64 imagens, das quais 30 foram repetidas, menos de metade. De salientar que para o nível "e" existem 117 imagens que podem ser apresentadas ao longo das três etapas, o que significa que para os quatro jogadores foram apresentadas 29,06% das imagens possíveis. Na primeira etapa e para a primeira imagem cada uma tem 2,78% de probabilidade de ser apresentada, para a segunda imagem, por a primeira já não poder ser escolhida, cada uma tem 2,86% de probabilidade e para a terceira imagem, cada uma tem 2,94% de probabilidade. A probabilidade continua a subir quantas menos imagens houver para mostrar e diminui quando o jogador avança de etapa, reinicia o nível ou se falhar todas as imagens de uma etapa.

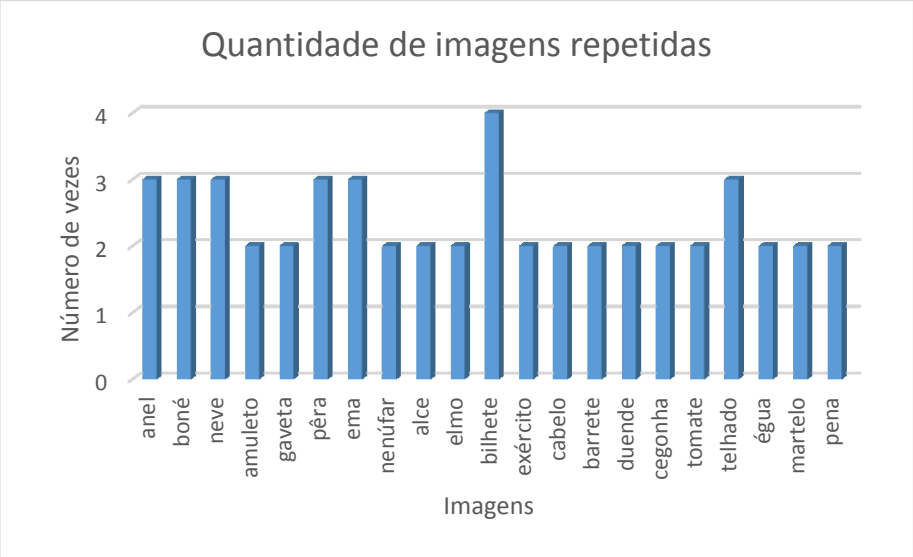


Figura 4.20: Quantidade de imagens repetidas apresentadas no geral

Com este gráfico consegue verificar-se que apenas uma imagem foi apresentada mais do que três vezes, o "bilhete", que seis foram apresentadas três vezes e que a grande maioria foi apresentada duas vezes, cumprindo a sua aleatoriedade.

5

Conclusão

5.1 Conclusão

Ao desenvolver este trabalho foi possível verificar a inexistência de dados de utilização por parte dos jogos. Estes apenas se interessam por cativar os jogadores e, quando guardam algum tipo de dados, é normalmente no contexto do jogo apenas. O desenvolvimento de jogos no contexto educativo já é desenvolvido há alguns anos e tem vindo a crescer muito. No entanto, este desenvolvimento é direccionado para jogos *online* por permitirem o acesso a qualquer dispositivo com acesso à internet.

O jogo “No Reino dos Fonemas” tem um contributo positivo para o aumento do conjunto restrito de ferramentas na intervenção e reeducação da consciência fonológica em crianças com perturbações do neurodesenvolvimento. Este jogo permite uma centralização da informação, facilitando a detecção das principais dificuldades, tendências, evoluções, etc. fornecendo dados mais esquematizados e de fácil análise estatística.

Uma das características que este trabalho me proporcionou foi o diálogo com pessoas, neste caso duas Doutoras, que não têm conhecimentos no desenvolvimento de jogos. Este diálogo foi essencial para completar a implementação ao receber opiniões e ideias à medida que uma nova etapa era apresentada e permitiu-me perceber como é feita a negociação entre os objectivos pretendidos e a exequibilidade dos mesmos.

Os objectivos delineados no capítulo 3 foram conseguidos com alguma dificuldade. A maneira de os dados serem gravados foi algo demorada para estes poderem ser utilizados por pessoas sem conhecimento na área e de forma a que se pudesse realizar algum estudo sobre os mesmos.

Um dos desafios presentes no desenvolvimento foi o de diminuir o sentimento de frustração quando as crianças fossem confrontadas com a dificuldade e o erro. Este sentimento foi diminuído através de uma redução das acções com conotação negativa que o jogo tem quando a criança não atinge o objectivo, incentivando-a a tentar outra vez.

A separação silábica foi a tarefa que se tornou mais desafiante, e ao mesmo tempo mais difícil, por ter de separar todas as palavras da língua portuguesa e revelou-se a mais crucial para o jogo, uma vez que, na língua portuguesa existem algumas excepções quanto à separação silábica, podendo esta ser diferente com as letras à volta de determinados ditongos.

A criação dos alvos também se demonstrou complicada por estes dependerem do número de sílabas e, consequentemente, a sua posição variar. Esta variação da posição tornou-se complexa por ter sido estabelecido que o posicionamento teria de acompanhar a definição do ecrã, ou seja, independentemente do ecrã utilizado, o jogo calcula todos os tamanhos necessários para que se consiga ver todos os componentes do jogo.

5.2 Contribuições

Gostaria de destacar todo o contributo dado pelo Centro Diferenças pelo trabalho realizado ao desenvolver a ideia do jogo “No Reino dos Fonemas”, bem como por todas as ideias fornecidas aquando do desenvolvimento da implementação.

Esta dissertação está acompanhada por um artigo científico intitulado “No Reino dos Fonemas” – jogo sério para a aprendizagem da leitura [31], apresentado na conferência SciTecIN’15 [32].

5.3 Trabalhos Futuros

Tal como referido no ponto 5.1., um melhoramento do jogo é possível a vários níveis e com diferentes objectivos.

A introdução de um utilizador administrador do jogo com a capacidade de alterar e/ou adicionar imagens e sons seria uma mais-valia para o objectivo do jogo em si, já que quem aplica o jogo seria capaz de focar para uma utilização específica consoante o jogador.

Em termos de consumo de recursos, e apesar de o jogo ter sido implementado com esse requisito não funcional, é possível diminuir este consumo e tornar o jogo mais leve para o computador.

A apresentação do jogo pode melhorar bastante no menu principal bem como na área dos jogadores. Uma mudança de forma a aumentar a interactividade do mesmo com o jogador é benéfica para que o jogo seja recebido com interesse, sem aumentar a capacidade de distrair os jogadores na área de jogo.

Este jogo representa metade do jogo estático apresentado no capítulo 3 e, por conseguinte, um dos trabalhos futuros seria a implementação da parte das consoantes, concluindo assim o jogo estático do “No Reino dos Fonemas”. As mecânicas são iguais, mas pode ser necessário uma adaptação das mesmas de forma a funcionar com as consoantes.

Na área de jogo há três componentes que podem ser melhoradas, as três informações em texto que se apresentam, o nível, a pontuação e a palavra. Estas podem sofrer uma mudança para se enquadrarem no contexto do jogo, sofrendo uma alteração de *design*.

O jogo pode ser implementado para outras plataformas, uma das capacidades do Unity® que traria uma mais-valia para as crianças. Esta possibilidade evitaria a restrição de ter de se usar um computador para o efeito, podendo o acompanhamento à criança ser realizado em qualquer lugar.

Bibliografia

[1] INE, “CENSOS 2001 Análie de População com Deficiência,” Instituto Nacional de Estatística de Portugal, Lisboa, 2001.

[2] E. L. E-Learning, Concepts, Trends, Applications, San Francisco, California: Epignosis LLC, 2014.

[3] P. B. Mota, *Game Wizard,* Lisboa: Faculdade de Ciências e Tecnologia, UNL, 2014.

[4] T. Cardoso, *Tecnologia de Jogos Digitais,* 2014.

[5] Humble, “Humble Bundle,” Humble Bundle Inc., 2015. [Online]. Available: https://www.humblebundle.com/.

[6] C. Fencott, M. Lockyer, J. Clay e P. Massey, Game Invaders - The Theory and Understanding of Computer Games, John Wiley & Sons, 2012.

[7] C. Koch, “Video Games are as bad as heroin, according to UK newspapper,” *Tech Times,* 2013.

[8] T. Adam e D. L. Mohamed, *Management of Technologies in Education,* 2015.

[9] D. Horachek, Creating E-Learning Games with Unity, Packt Publishing, 2014.

[10] R. C. Clark e R. E. Mayer, E-Learning and the Science of Instruction, San Francisco, California: Pfeiffer, 2008.

[11] Mojang, “education minecraft,” Microsoft, 2015. [Online]. Available: http://education.minecraft.net/.

[12] Paradox Interactive, “Europa Universalis 4,” 2012. [Online]. Available: http://www.europauniversalis4.com/.

[13] G. d. U. “SBIR STTR,” SBA, 2013. [Online]. Available: https://www.sbir.gov/sbirsearch/detail/415241.

[14] Eversim, “Masters of the World,” Eversim, 2015. [Online]. Available: http://masters-of-the-world.com/presentation.php?langue=en.

[15] J. Gregory, Game Engine Architecture, Taylor and Francis Group, LLC, 2009.

Bibliografia

[16] T. Unity, “Unity Technologies,” 2015. [Online]. Available: http://unity3d.com/unity.

[17] D. Takahashi e D. Helgason, Interviewees, *Game Developers, start your Unity 3D engines (interview).* [Entrevista]. 2 Novembro 2012.

[18] Nvidia, “Nvidia Corporation,” 13 Fevereiro 2008. [Online]. Available: http://www.nvidia.com/object/io_1202895129984.html.

[19] Nvidia, “PhysX SDK, Nvidia Corporation,” 18 Agosto 2015. [Online]. Available: https://developer.nvidia.com/physx-sdk.

[20] Epic, “Unreal Engine,” 2015. [Online]. Available: https://docs.unrealengine.com/latest/INT/index.html.

[21] J. Ness, A. Olsen, M. Røland e C. A. Sand, “Project NORS,” 2015.

[22] Crytek, “CryEngine,” 2015. [Online]. Available: http://cryengine.com/features.

[23] Bad Juju Games, “Desura,” 2015. [Online]. Available: http://www.desura.com/engines/game-maker.

[24] J. L. Ford, Getting Started With Game Maker, Boston: Cengage Learning, 2010.

[25] YoYo Games, “Game Maker: Studio,” 2015. [Online]. Available: http://www.yoyogames.com/.

[26] W. d. P. P. Filho, Engenharia de Software: fundamentos, métodos e padrões, 2000.

[27] Diferenças, *Documentação do jogo "No Reino dos Fonemas".*

[28] A. T. Velloso, “Separando sílabas em C#,” MSDN, [Online]. Available: http://www.microsoft.com/brasil/msdn/Tecnologias/visualc/SeparandoSilabas.mspx.

[29] P. E. “infopedia,” Porto Editora, 2015. [Online]. Available: http://www.infopedia.pt/dicionarios/lingua-portuguesa-aao/otorrinolaringologista.

[30] iLteC, “portaldalinguaportuguesa,” Instituto de Linguística Teórica e Computacional, 2015. [Online]. Available: http://www.portaldalinguaportuguesa.org/about.html?action=syllables&act=list&search=otorrinolaringologista.

[31] J. M. Soares Franco, T. Cardoso, F. Trindade, S. Silva e M. Palha, *"No Reino dos Fonemas" - jogo sério para a aprendizagem da leitura,* 2015.

[32] SciTecIN, “SciTecIN,” 2015. [Online]. Available: http://scitecin.isr.uc.pt/index.php/pt/programa/sessoes.

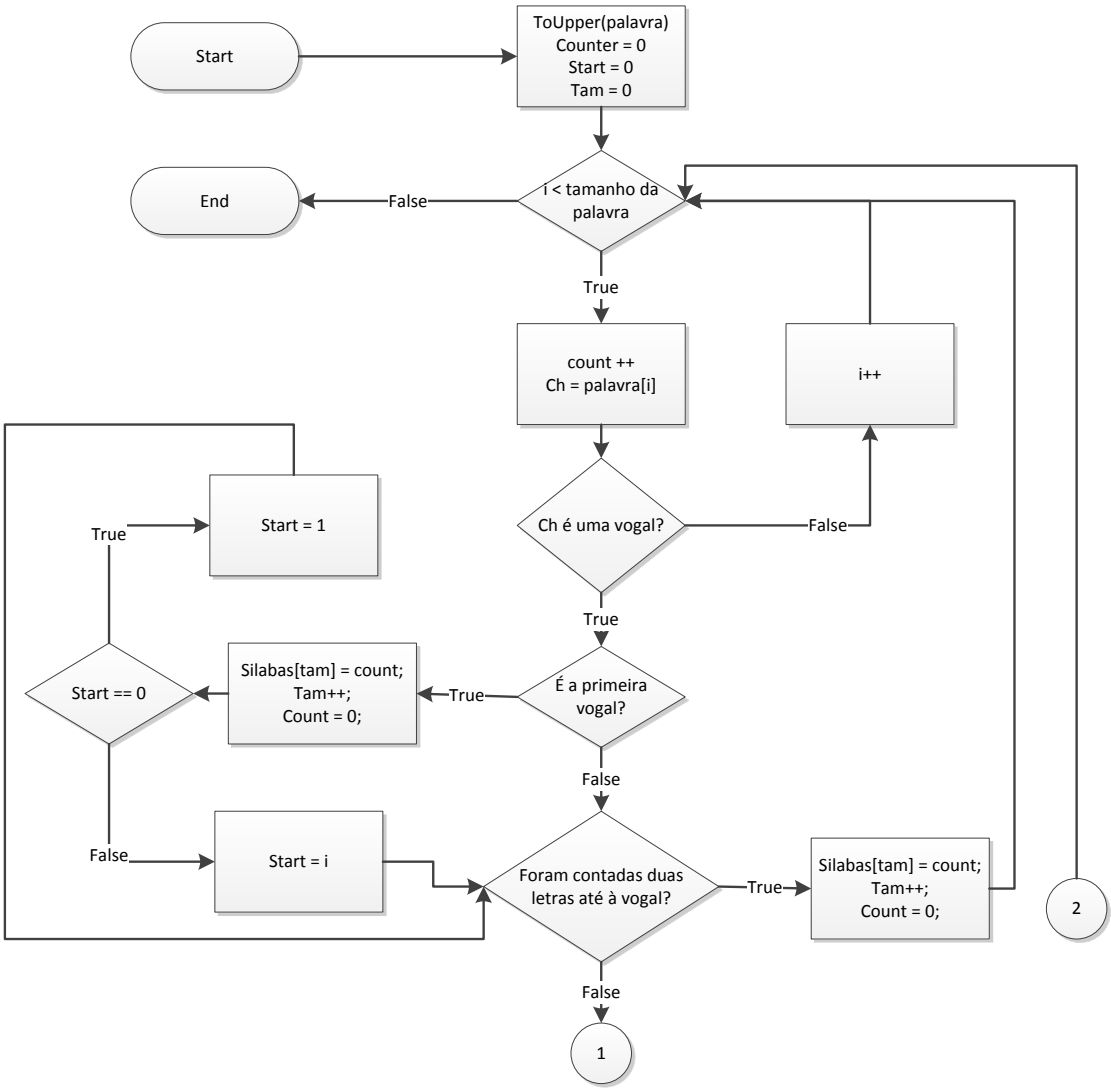
[33] B. Nilson e M. Söderberg, *Game Engine Architecture,* 2007.

[34] Epic, “Unreal Engine,” 2015. [Online]. Available: https://www.unrealengine.com/unreal-engine-4.

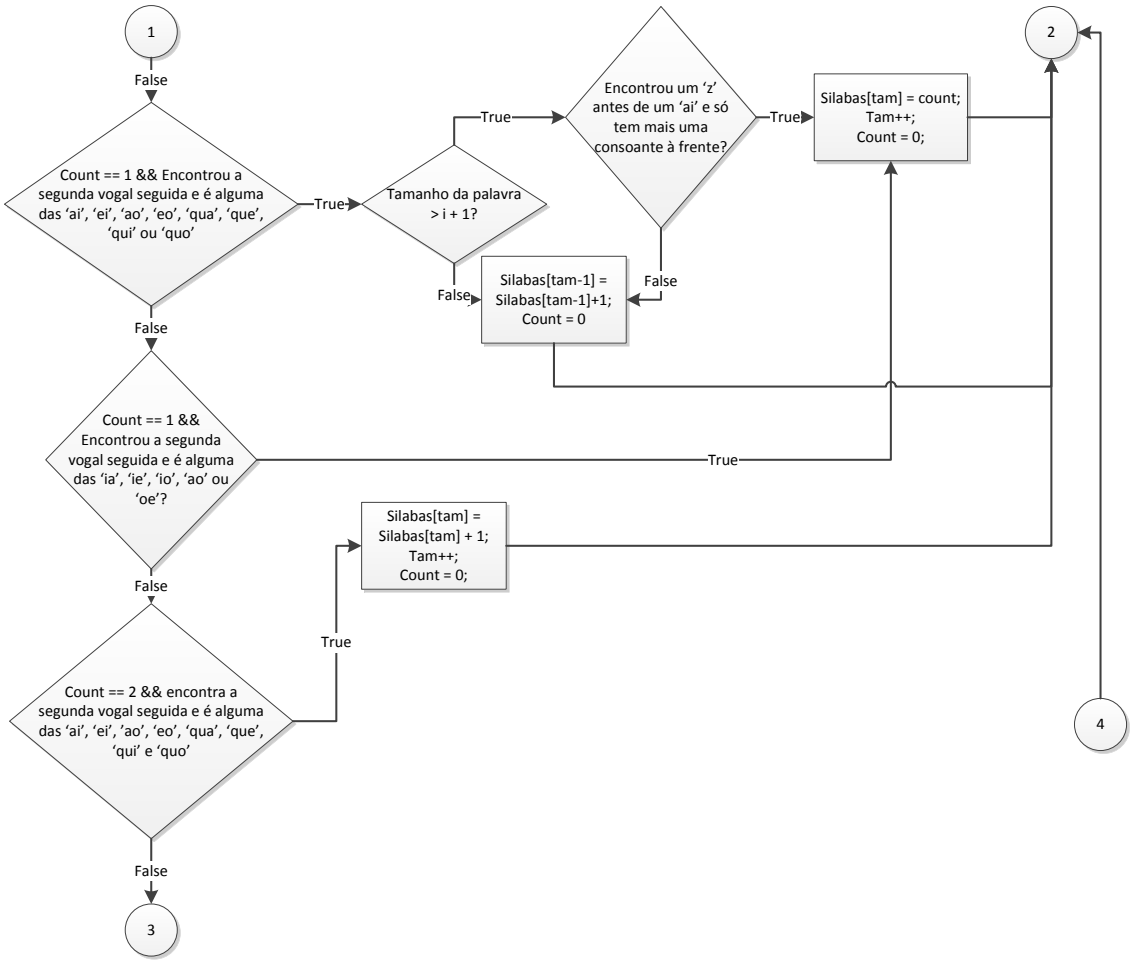
Bibliografia

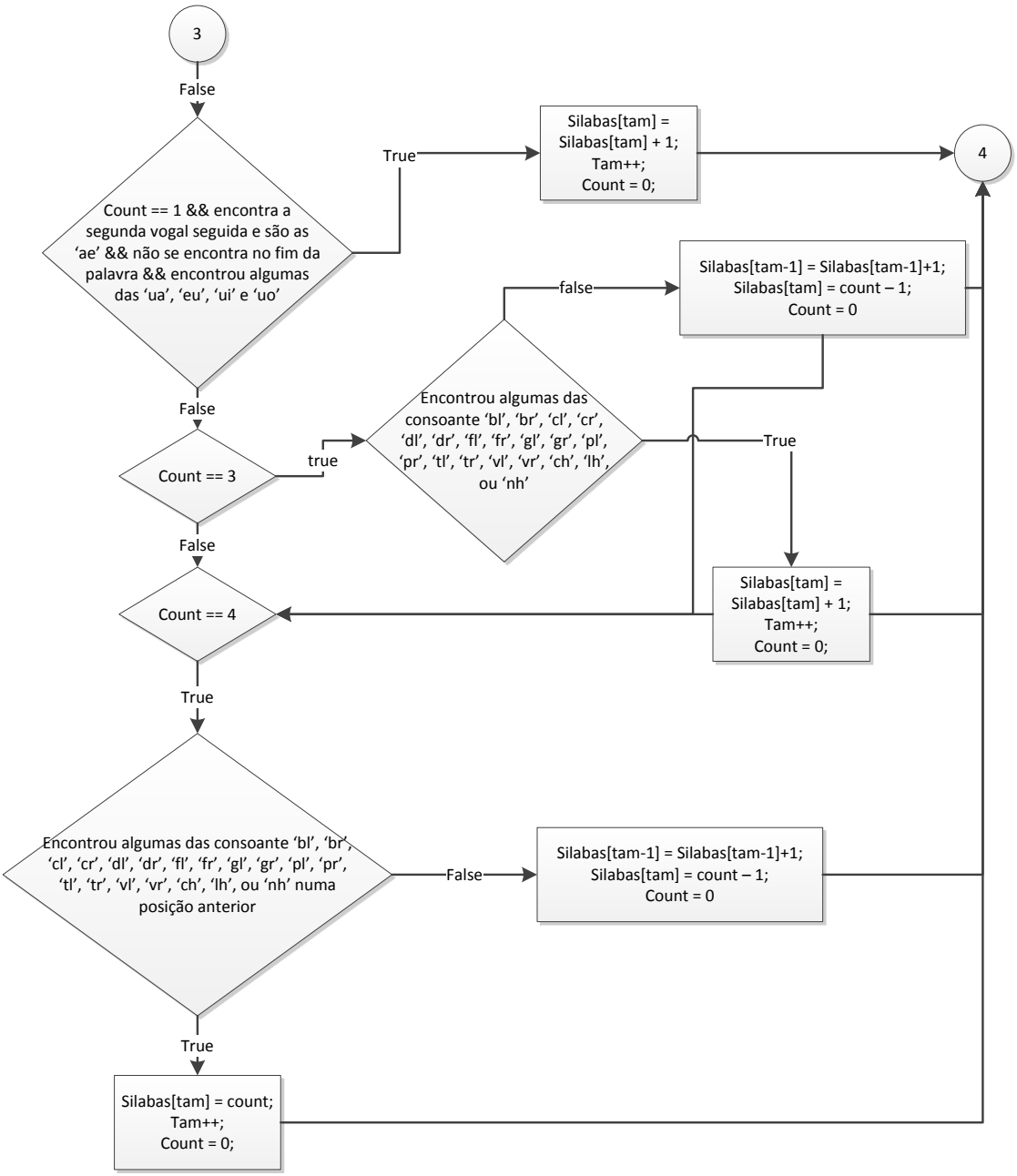
Anexos

Anexo 1

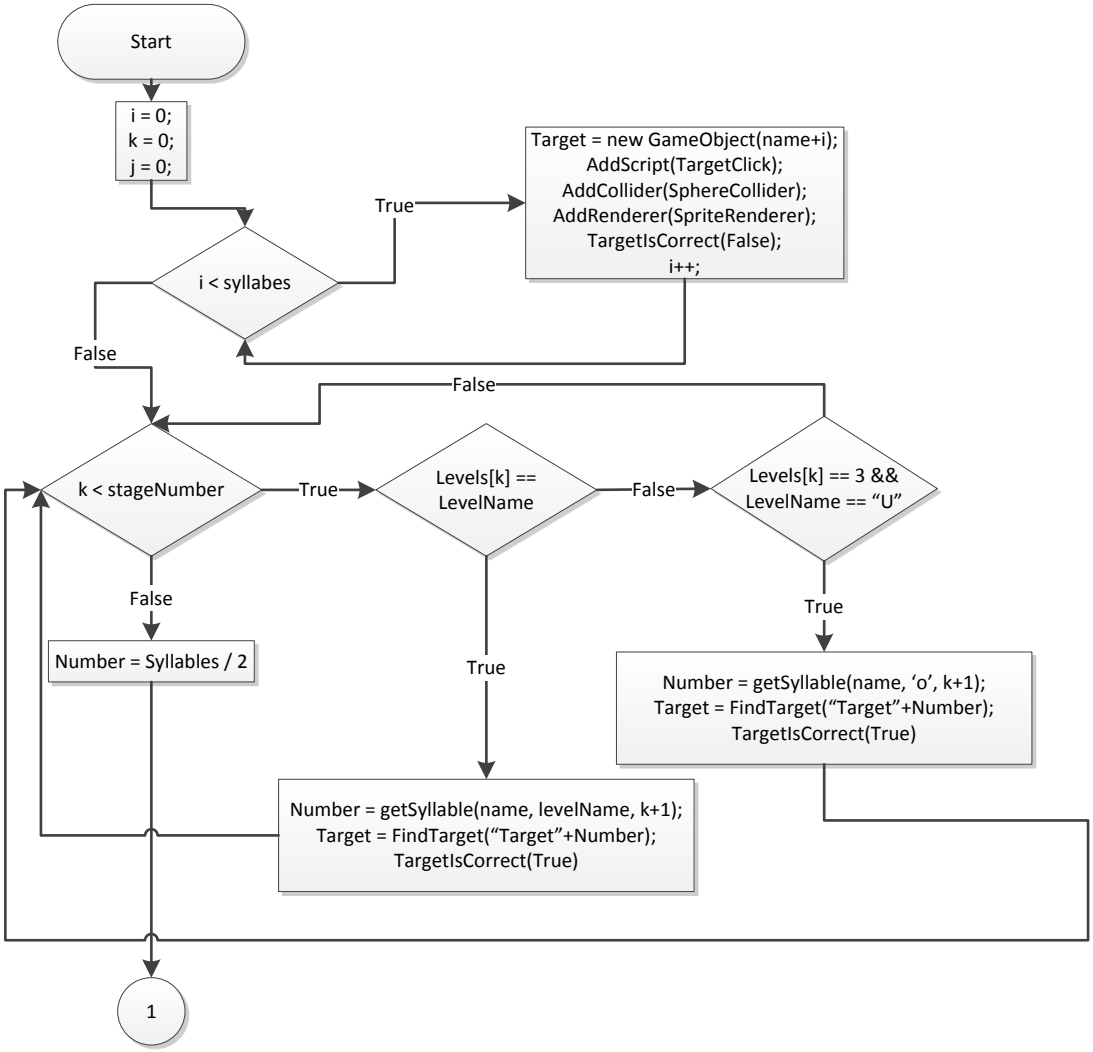


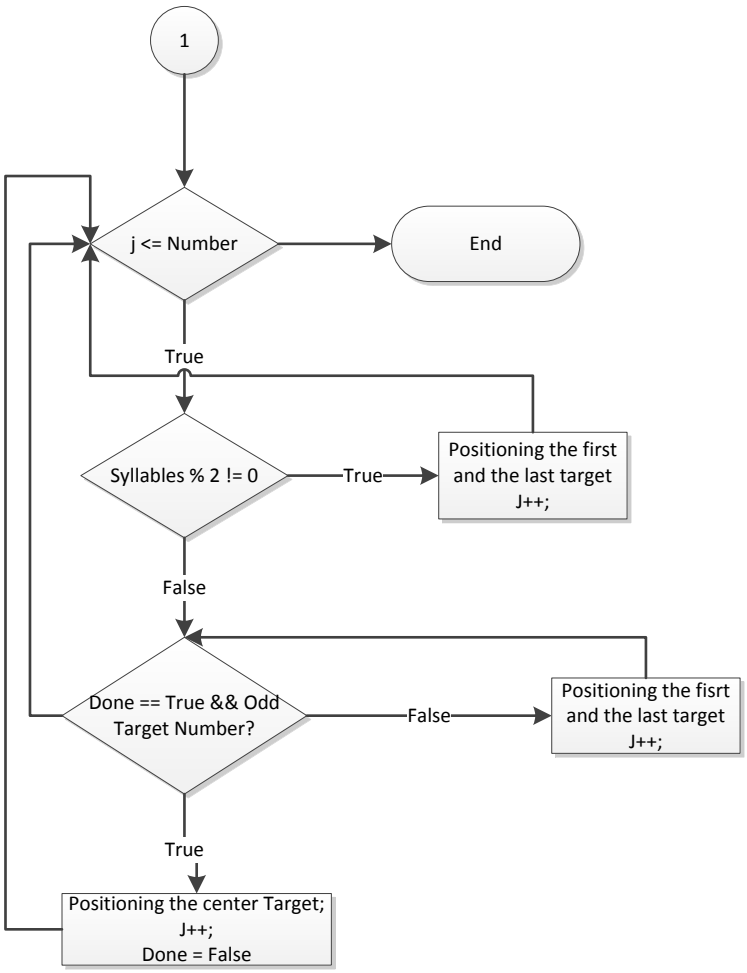
Anexos


1
False
Count == 1 && Encontrou a segunda vogal seguida e é alguma das ‘ai’, ‘ei’, ‘ao’, ‘eo’, ‘qua’, ‘que’, ‘qui’ ou ‘quo’
True
Tamanho da palavra > i + 1?
True
Encontrou um ‘z’ antes de um ‘ai’ e só tem mais uma consoante à frente?
True
Silabas[tam] = count;
Tam++;
Count = 0;
2
False
Silabas[tam-1] = Silabas[tam-1]+1;
Count = 0
False
False
Count == 1 && Encontrou a segunda vogal seguida e é alguma das ‘ia’, ‘ie’, ‘io’, ‘ao’ ou ‘oe’?
True
Silabas[tam] = Silabas[tam] + 1;
Tam++;
Count = 0;
False
True
Count == 2 && encontra a segunda vogal seguida e é alguma das ‘ai’, ‘ei’, ‘ao’, ‘eo’, ‘qua’, ‘que’, ‘qui’ e ‘quo’
False
3
4


3
False
Count == 1 && encontra a segunda vogal seguida e são as 'ae' && não se encontra no fim da palavra && encontrou algumas das 'ua', 'eu', 'ui' e 'uo'
True
Silabas[tam] = Silabas[tam] + 1; Tam++; Count = 0;
4
false
Silabas[tam-1] = Silabas[tam-1]+1; Silabas[tam] = count – 1; Count = 0
False
Count == 3
true
Encontrou algumas das consoante 'bl', 'br', 'cl', 'cr', 'dl', 'dr', 'fl', 'fr', 'gl', 'gr', 'pl', 'pr', 'tl', 'tr', 'vl', 'vr', 'ch', 'lh', ou 'nh'
True
False
Count == 4
Silabas[tam] = Silabas[tam] + 1; Tam++; Count = 0;
True
Encontrou algumas das consoante 'bl', 'br', 'cl', 'cr', 'dl', 'dr', 'fl', 'fr', 'gl', 'gr', 'pl', 'pr', 'tl', 'tr', 'vl', 'vr', 'ch', 'lh', ou 'nh' numa posição anterior
False
Silabas[tam-1] = Silabas[tam-1]+1; Silabas[tam] = count – 1; Count = 0
True
Silabas[tam] = count; Tam++; Count = 0;

Anexo 2


Start
i = 0;
k = 0;
j = 0;
i < syllabes
True
Target = new GameObject(name+i);
AddScript(TargetClick);
AddCollider(SphereCollider);
AddRenderer(SpriteRenderer);
TargetIsCorrect(False);
i++;
False
False
k < stageNumber
True
Levels[k] == LevelName
False
Levels[k] == 3 && LevelName == "U"
False
Number = Syllables / 2
True
Number = getSyllable(name, levelName, k+1);
Target = FindTarget("Target"+Number);
TargetIsCorrect(True)
True
Number = getSyllable(name, 'o', k+1);
Target = FindTarget("Target"+Number);
TargetIsCorrect(True)
1


1
j <= Number
End
True
Syllables % 2 != 0
True
Positioning the first and the last target
J++;
False
Done == True && Odd Target Number?
False
Positioning the fisrt and the last target
J++;
True
Positioning the center Target;
J++;
Done = False