Licenciatura em Engenharia Informática

Turma Pós-Laboral

Videojogos como Ferramenta de Aprendizagem

João Pedro Mota da Silva de Marques Cardoso, Aluno n.º 1505

Coordenador: Professor Doutor Pedro Ramos dos Santos Brandão

Lisboa 21 de Junho de 2017

Ano Letivo 2016/2017

Agradecimentos

Ao Miguel Cintra por disponibilizar as musicas e efeitos áudio.

A Cigny Malvar por disponibilizar os gráficos e animações.

Resumo

As mais recentes metodologias de educação e ensino, quer de crianças, quer de adultos, apontam para os jogos e para a componente lúdica como uma das mais eficazes ferramentas de aprendizagem. Atualmente, os meios eletrónicos estão no centro do interesse das crianças e são uma das ocupações mais motivantes e cativantes, aliando a tecnologia e a diversão.

Tendo um potencial de motivação elevado, este contexto permite-nos proporcionar momentos de aprendizagem baseados na visualização e repetição, enquanto ferramentas tradicionais de apreensão de conceitos.

Partindo destes pressupostos – aprendizagem por visualização e repetição, em contexto lúdico e formato eletrónico – o objetivo deste Projeto Global é construir um jogo infantil, que permita a apreensão dos conceitos de contagem, numeração e distinção de cores.

Neste caso, dados os conceitos em causa, a amostra pretendida estará em idade pré-escolar, com crianças entre os 3 e os 6 anos.

A consulta bibliográfica incluiu três áreas diferentes – condução de estudos em ciências sociais, metodologias e técnicas de ensino e tecnologias para a construção de um jogo. Este projeto vai apoiar-se nestas três fontes, sendo a sua aplicação descrita nos diferentes momentos como justificação para as escolhas feitas.

As metodologias e técnicas de ensino vão dar suporte ao tipo de mecânica e lógica usada no jogo e temos a sua descrição no resumo teórico. As fontes referentes à condução de estudos em ciências sociais vão dar estrutura ao trabalho, sendo a sua aplicação mais frequente que as referências. E finalmente, as bibliografias técnicas vão dar orientação às escolhas de ferramentas e programação.

O objetivo é desenvolver um jogo que permita validar o interesse das crianças aplicado à aprendizagem, em contexto lúdico e tecnológico.

Palavras-chave: Jogos, videojogos, aprendizagem, jogabilidade

Abstract

The latest educational and learning methodologies, either for children, or for adults, point to games and the fun component as one of the most effective learning tool. Nowadays, electronics are an attention focus for kids and are one of the most motivating and captivating occupation, allying technology and fun.

Having an high motivation potential, this context allows us to offer learning moments based on visualization and repetition, as traditional tools for teaching concepts.

With these assumptions – learning by visualization and repetition, in a fun context and electronic format – the goal for this Global Project is to build a kid's game that allow the apprehension of counting, numbering and color distinction concepts.

In this case, given the concepts to explore, our intended sample will be in a pre-school age, with children of 3 to 6 years old.

The bibliographic research was made in three areas – conduction of studies in social sciences, educational methodologies and techniques, and technologies for the construction of an electronic game. This project will find support in these three sources, describing them along in different moments as a justification for the choices made.

Educational methodologies and techniques will serve as base for the mechanic and logical following used in the game and we'll have a description in the theoretical resume. The sources referring to conducting studies in social science will give structure to the project and its presentation, being its application more frequent than the needed references to its use. And finally, the technical bibliographies will guide the choices in terms of tools, software and programming.

The goal is to develop a game that will allow us to validate the interest of children in learning, in a fun and technological context.

Keywords: Games, Video Games, learning, Gameplay

Lista de Acrónimos

IDE	Integrated Development Environment
GUI	Graphical User Interface
2D	Bidimensional; ou relativo a duas dimensões
3D	Tridimensional; ou relativo a três dimensões
HUD	Heads-Up Display
VS	Visual Studio
MS	Microsoft
GIMP	GNU Image Manipulation Program
FSM	Finite State Machine
HSM	Hierarchical state Machine
YAML	YAML Ain't Markup Language
MSDN	Microsoft Developer Network
DGBL	digital game-based learning
COTS	commercial off-the-shelf
PBL	Problem-based learning

Conteúdo

Lista de Figuras

Lista de Tabelas

Introdução

Os jogos e as brincadeiras como ferramentas de aprendizagem (Trybus, 2014) são considerados pelos diferentes teóricos como as melhores abordagens do ponto de vista da sua eficácia e da capacidade de retenção e memória.

Assim, o objeto deste projeto é testar a eficácia dos jogos eletrónicos na transmissão de conceitos matemáticos. O projeto inclui a criação de um jogo para a faixa etária dos 3 aos 6 anos, que apresente o mesmo tipo de conceito que está a ser lecionado nas escolas pré-primárias – contagens, identificação dos números e identificação das cores. Depois tentarei convidar um grupo de crianças em idade pré-escolar para testar o jogo.

A metodologia escolhida foi desenvolver um jogo em Unity3d, que permita funções de movimento, *design* (MacKay, 2013) adequado à faixa etária, sons apelativos e um ciclo fechado que leve o jogador do início ao final do jogo.

O objetivo é um jogo eletrónico que chame a atenção pelo *design*, que crie um desafio com contagens e que vá mostrando os números de forma sequencial e repetitiva.

A estrutura do projeto implica todo o processo do jogo, desde a abertura com imagens claras e apelativas que permitam rapidamente perceber a mecânica do jogo; depois tem o desafio de conseguir acertar nas cores, implicando um algoritmo que dificulte um pouco o jogo e constitua um desafio; finalmente terá a contagem progressiva até ao final do jogo.

As imagens e os sons foram contributos valiosos, críticos para o projeto, de dois amigos que se disponibilizaram para o efeito – estes componentes são propriedade dos seus autores e não podem ser usados para nenhum outro fim.

1. Estado da Arte

1.1. Jogos e aprendizagem.

A palavra jogo é um termo do latim, "*Jocus*", que significa brincadeira, divertimento; a arqueologia registra a presença de competições e jogos desde 2600 A.C.

"Um jogo é uma atividade entre dois ou mais tomadores de decisão independentes que procuram atingir os seus objetivos em um contexto de limitação." (Abt, 1987)

É na República de Platão [427-384 a.C.], que um elo é estabelecido entre a educação ou cultura (*paideia*) e jogo (*paidiá*).

O objetivo da pedagogia (*paidagogía*), é encorajar a aprendizagem como uma forma de jogo (*paidiá*).

Platão também sugeriu que a aprendizagem das crianças devera ser feita através de jogo que simulassem as atividades dos adultos (brincar ao faz de conta)

A partir da idade média o uso de jogos como forma de aprendizagem perdeu-se, devido à interferência do cristianismo que impunha uma educação disciplinadora e condenava o uso de jogos como método educativo, ou mesmo de entretenimento, sendo considerado pecado.

É no século XVI com o Renascimento que os humanistas reconhecem o poder educativo dos jogos e incorporam-nos na educação de jovens e adultos.

Mais recentemente, em 2006, Richard Van Eck falou sobre o DGBL (Eck, Digital Game-Based Learning: It's Not Just the Digital Natives Who Are Restless.... , 2006), em que chama a atenção para o uso de jogos na aprendizagem, dando como exemplo o uso de jogos comerciais COTS com "Sim City","CSI", "Civilization" e ainda "Age of Empires", podendo ser usados para estudar história, justiça criminal, ou engenharia e planeamento urbano.

Mais tarde, em 2015, Richard Van Eck (Eck, digital-game-based-learning-still-restless-after-all-these-years, 2015), quase uma década depois, reviu a evolução do DGBL, onde constatou que existe informação suficiente para afirmar que DGBL melhoram a aprendizagem entre 7 e 40%, sendo o suficiente para um aluno não reprovar.

Ficou também claro que o DGL é a ferramenta certa para o PBL, que é uma das áreas mais difíceis na aprendizagem, sendo os jogos por definição uma forma de PBL.

1.2. Quantificar o mercado de vídeo jogos

Desde de sempre que os jogos são uma área de interesse, nas duas últimas décadas o mercado de jogos teve um crescimento rápido, tendo no ano de 2016 gerado 91 biliões de dólares (SUPERDATA, 2017).

O mercado de jogos em Portugal também tem tido um crescimento assinalável, o volume de negócios das empresas portuguesas de vídeo jogos no ano de 2016 é um valor entre 6 e 12 milhões (não foi possível recolher dados de todas as empresas) de Euros (IOL, 2017)

1.3. Motor de jogos

O motor de jogos é uma arquitetura que os programadores usam para executar o jogo (Game Designing, 2017).

Com a evolução dos jogos de computador e a crescente complexidade dos mesmos (3D, Realidade Virtual e Realidade Aumentada), o conhecimento técnico requerido aumentou drasticamente, sendo muito difícil conseguir dominar todas as técnicas, junta-se também o crescente número de plataformas (dispositivos móveis e consolas), que são compostas por "*hardware*" e sistemas operativos proprietários. Todas estas variáveis fazem com que o desenvolvimento de jogos sem o suporte de um motor de terceiros, seja uma solução pouco viável.

O motor permite ao programador focar-se no jogo, acelerando o processo de desenvolvimento e criativo do mesmo, que seria bastante mais complicado e demorado (Felicia, 2015).

Um motor de jogos por norma disponibiliza funcionalidades (Game Designing, 2017) que permitem ao programador implementar:

- Física (Physics)
- Entrada (Input)
- Renderização (Rendering)
- Script (Scripting)
- Deteção de colisões (Collision detection)
- Inteligência Artifical (artificial inteligence)

Os motores mais conhecidos (G2 Crowd, 2017) são (sem qualquer ordem especifica):

- Unreal Engine

- Cry engine
- Unity
- GameMaker

1.3.1. Unity

O motor foi criado em 2004 por David Helgason (CEO), Nicholas Francis (CCO), e Joachim Ante (CTO) em Copenhaga, Dinamarca, depois do primeiro jogo, "GooBall", não ter tido o sucesso esperado. Perceberam a utilidade de um motor de jogos como um conjunto de ferramentas e decidiram criar um motor que todos pudessem usar por um preço baixo (Wikipedia, 2017).

Encontra-se disponível para OSX, Windows e também para Linux (ainda em versão Beta), permite desenvolver e disponibilizar conteúdos para mais de 24 plataformas (Unity, 2017)

Dispositivos moveis

- IOS
- Android
- Windows Phone
- Tizen
- Fire OS

Realidade Virtual (VR) e Realidade Aumentada (AR)

- Oculus Rift
- Google Cardboard
- Steam VR
- Playstation VR
- Gear VR
- MicroSoft HoloLens
- Day Dream

Desktops

- Windows
- Windows Store Apps
- Mac

- Linux/Steam OS
- Facebook GameRoom

Consolas

- PS4
- PS Vita
- Xbox One
- Wii U
- Nintendo 3DS
- Nintendo Switch

Web

- Web GL

Smart TV's

- Android TV
- Samsung Smart TV
- Tv OS

O Unity permite o desenvolvimento de jogos de computador sem ser necessário ter conhecimento profundo de todas as tecnologias necessárias para a criação de jogos, por exemplo não é necessário saber como o motor comunica com a placa gráfica para renderizar os gráficos (Schrier, 2016).

Permite ainda um desenvolvimento rápido em que o programador se pode concentrar apenas na mecânica do jogo, graças ao conjunto de funcionalidades disponíveis.

O IDE (Integrated Development Environment) permite controlar todas as fases do desenvolvimento, criar, compilar o código e depuração durante a execução do jogo.

O GUI (Graphical User Interface) permite a criação de conteúdos através de "*drag & drop*" permitindo a visualização do mesmo em tempo real, sendo possível alterar os scripts e conteúdos sem ser preciso compilar (Felicia, 2015).

Para a programação está disponível C# e *JavaScript*, que permite aos programadores optar pela linguagem com que se tenha melhor produtividade

O Unity permite o desenvolvimento de jogos 2D e 3D, podendo misturar conteúdos 2D e 3D, criando jogos tão variados quanto os RPGs ou FPS.

Além de um completo manual "*online*", que é atualizado sempre que sai uma nova versão, estão disponíveis uma série de tutoriais, que envolvem todas as áreas de desenvolvimento e para todos os níveis de conhecimento.

É também disponibilizado um conjunto de serviços que complementam o desenvolvimento de jogos (Unity, 2017), sendo alguns deles gratuitos:

- Unity Ads – Plataforma para gerir anúncios, que representam receita adicional.
- Unity Analytics – Fornece um conjunto de métricas que ajudam a analisar o comportamento dos jogadores
- Unity Certification – Certificação que valida conhecimentos fundamentais no desenho de jogos e programação.
- Unity Could Build – Faz a compilação, instalaçãor, teste do jogo e disponibilização para várias pessoas automaticamente.
- Unity Collaborate – Permite que equipas em diferentes áreas geográficas, possam trabalhar no mesmo projeto em tempo real.
- Unity Everplay – Facilita aos utilizadores a gravação do seu jogo e partilha nas principais redes sociais.
- Unity IAP – Simplifica compras *in-app*, de várias lojas on-line
- Unity Multiplayer – Simplifica a criação de jogos multiutilizador.
- Unity Performance Reporting – Permite a recolha de erros em tempo real, sendo possível iniciar a depuração de imediato.

1.4. Ferramentas de trabalho em equipa

1.4.1. Porque são importantes

A necessidade de ferramentas para partilha de informação entre membros de equipas é crucial para o sucesso dos projetos, o correio eletrónico não se adapta à necessidade de ter vários elementos a contribuir para distintos assuntos, com partilhas de informação em tempo real (Dodd, 2017).

1.4.2. Confluence

É uma ferramenta de colaboração (DUFFY, 2017) que permite que as equipas sejam mais produtivas, facilitando a recolha e partilha de qualquer tipo de informação num único local, e pode ser usado como *SAAS* (*Software as a Service*) ou instalada localmente, usa um interface WEB acessível desde dispositivos fixos e móveis (Altassian, 2015), é agnóstica em termos de infraestrutura, podendo ser usada nos sistemas operativos mais comuns (Windows, OSX e Linux) (Thomas E. Murphy, 2017).

Aliado a um robusto sistema de relatórios e um poderoso motor de buscas, agiliza o acesso a qualquer tipo informação.

1.4.3. Jira

É uma ferramenta para gerir projetos usando uma metodologia AGILE, que faz a necessária integração com "*Confluence*" e pode ser usado como SAAS (*Software as a service*) ou instalada localmente, usa um interface WEB, acessível desde dispositivos fixos e móveis, é agnóstica em termos de infraestrutura, podendo ser usada nos sistemas operativos mais comuns (Windows, OSX e Linux) (itcentralstation, 2017).

A integração com *Confluence*, permite transformar os requisitos em histórias (*stories*), com o mínimo de esforço, ficando de imediato disponíveis no *backlog*.

A integração com Bamboo, faz a implementação de um ambiente de CI (*Continuous Integration*), sendo possível monitorizar o estado da "*build*".

A integração com BitBucket, garante que as tarefas (Stories/Tasks) são atualizadas quando o código é guardado (*Commited*).

1.5. Visual Studio

O Visual Studio é o IDE (Integrated Development Environment) da Microsoft (Microsoft, 2017), desde da sua primeira versão 97 (1997), passando pela versão Visual Studio .Net (2002), onde se introduziu a "framework" .Net , que adiciona linguagem de programação com gestão de memória automática, e com a versão mais atual 2017 (2017), têm um conjunto de funcionalidades que aceleram o desenvolvimento e permitem o desenvolvimento de aplicações mais poderosas. Entre as suas principais funcionalidades estão as seguintes:

- *Designer* – Para a criação de formulários (forms) graficamente
- *InteliSense* – Enquanto o programador escreve o código, fica de imediato disponível para escolha as classes, métodos ou atributos, que estejam dentro do contexto.

- *Incremental build* – Faz compilações rápidas, apenas compilando o código alterado
- *Debugger* – Depuração remota e alteração do código enquanto se depura, mesmo em execução
- *Extensions* – Permite adicionar funcionalidades de terceiros, totalmente integradas com o IDE

Atualmente o Visual Studio 2017 encontra-se disponível em diferentes versões:

- *Code* – Versão open-source que corre em múltiplas plataformas (Windows, OSX e Linux), compatível com dezenas de linguagens
- *Comunity* – Versão gratuita para a comunidade, para desenvolvimento de aplicações não empresariais
- *Professional* – Conjunto de ferramentas profissionais e subscrição do MSDN *Library*.
- *Enterprise* – Conjunto de ferramentas para desenvolvimento de aplicações empresariais que inclui *DevOps*, ferramentas de diagnóstico, testes de carga

1.6. GIT

É um sistema de controlo de versões, para controlar de alterações em ficheiros, permitindo que várias pessoas alterem os mesmos ficheiros e consigam sincronizar as alterações de diferentes utilizadores.

Criado por Linus Torvalds em 2005 para o desenvolvimento do Linux Kernel com outros programadores.

É um sistema distribuído, não tendo uma componente cliente e servidor, cada diretório GIT contem uma cópia completa do repositório (Wikipedia, 2017).

1.7. GIMP

GIMP (GNU Image Manipulation Program) é um editor de imagem open-source, a primeira versão foi desenvolvida por Spencer Kimbal e Peter Mattis em 1995, como projeto de semestre na Berkeley Universidade da Califórnia, a versão 0.54 foi a primeira versão publicada para o público em geral, ao longo dos anos têm tido melhorias substanciais, sendo indicado como um bom substituto do Photoshop para equipas que não têm capacidade financeira e não precisam de todas as funcionalidades do PhotoShop (Wikipedia, 2017).

Encontra-se disponível nas principais plataformas Windows, OSX e Linux.

1.8. Audacity

É uma aplicação de edição digital de áudio open-source (Wikipedia, 2017), foi inicialmente desenvolvida por Dominic Mazzoni e Roer Dannenber na Universidade Carnegie Mellon - a primeira versão 0.8 foi publicada em Maio de 2000.

Além de capturar áudio de diferentes fontes, permite a sua edição usando uma biblioteca de efeitos (Williams, 2017), é bastante popular devido à sua simplicidade e capacidade de importar e exportar múltiplos formatos, encontra-se disponível nas principais plataformas Windows, OSX e Linux.

Contextualização

O objetivo deste projeto é explorar a temática de aprendizagem com a ajuda de jogos, sendo utilizado para o efeito um jogo de computador.

Os equipamentos eletrónicos são uma parte integrante das nossas vidas, quase todas as nossas ações são efetuadas interagindo com um dispositivo eletrónico, mesmo a interação com pessoas é frequentemente efetuada com ajuda de dispositivos eletrónicos.

As novas gerações, nascem num mundo onde os dispositivos eletrónicos são uma parte integrante da sua existência, não olham para eles como uma opção, os dispositivos eletrónicos, estão para as recentes gerações como o relógio estava para as anteriores, um objeto imprescindível.

O facto de termos passado do objetivo de ter um computador em cada casa, para ter um dispositivo móvel por pessoa, ajudou ao crescimento e criação de mercados que anteriormente eram considerados nichos, entre eles o mercado de jogos de computador.

Este projeto foca-se na criação de um jogo de computador que permita às crianças do primeiro ciclo aprender a contar até 10 e aprender as cores básicas, enquanto jogam um jogo divertido.

2. Desenvolvimento

Este capitulo têm como objetivo descrever a metodologia usada para o desenvolvimento do projeto, e detalhar a estrutura da aplicação.

2.1. Conceito

O primeiro passo é definir qual o conceito do jogo, o objetivo primário e qual o público-alvo.

2.1.1. Objetivo

Aprendizagem dos números até 20, em conjunto com a identificação das cores primárias.

2.1.2. Público-alvo

Crianças entre os 4 até aos 6 anos, com competências instaladas de manuseamento de dispositivos eletrónicos e cujos programas curriculares incluam a aprendizagem da numeração e das cores.

2.2. Desenho

O desenho do jogo envolve várias etapas, sendo um processo iterativo em que se vai avançando, mas revendo tudo o que foi definido anteriormente.

2.2.1. Grafismo

O estilo gráfico do jogo tinha de ser apelativo para as crianças da faixa etária escolhida, foi escolhido um estilo com cores apelativas e linhas simples, para estar próximo do tipo de desenhos que as crianças conhecem, para que possam facilmente fazer os seus próprios desenhos.

A palete de cores é predominante azul (a cor que as crianças usualmente usam para descrever o oceano), sendo os personagens de cores constituídos por uma cor base, preferencialmente uma cor primária e com pequenos detalhes de outras cores.

2.2.2. Ambiente

O ambiente em que se desenrola a ação, é um ambiente marinho, a motivação da escolha foi o facto de as crianças gostarem deste tipo de temas e permitir uma movimentação mais simples de objetos e personagens.

Figura 1 Imagem de fundo "Background"

2.2.3. Personagens

Tal como um filme, um jogo, têm um ou mais atores principais, atores secundários e figurantes. Estes bonecos e animações foram criados por um animador que disponibilizou a sua arte para a construção do jogo.

2.2.3.1. Personagem Principal

O ator principal é uma foca, com olhos grandes para transmitir a mesma imagem que os desenhos animados atuais, com cores fortes e brilhantes.

Figura 2 Ator principal "Pixie"

A proporção das várias partes do corpo não é importante, a cabeça é grande em relação ao corpo (não obedecendo a fisionomia da foca), sendo o elemento principal os grandes olhos, que é um dos principais órgãos para a transmissão emoções, o objetivo é ser giro e cativante, captando atenção dos jogadores.

Foi-lhe dado o nome de "*Pixie*", que significa pequeno ser, geralmente uma pessoa imaginária, é um nome originário do Século XVII, que é frequentemente usada nas histórias para crianças, contribuiu o facto de ser um nome pequeno e fácil de pronunciar.

2.2.3.2. Atores secundários

Neste jogo existem quatro atores secundários (uma homenagem aos quatro fantasmas do jogo *Pac-man*) (Wikipedia, 2017). Mantendo o ambiente marítimo os atores secundários são peixes, tendo cada um como cor principal uma cor chamativa primária/secundária, para simplificar foi-lhes atribuído um nome que é a cor em Inglês – aumentando o leque de aprendizagem para o programa de inglês nas primeiras idades.

Figura 3 Blue

Figura 4 Red

Figura 5 Green

Figura 6 Yellow

2.2.4. Adereços

Os adereços são elementos que não têm qualquer tipo de interação com os atores, têm o propósito de enriquecer o jogo, foram criados alguns adereços marítimos estáticos, para ornamentar a área de jogo. Os adereços foram criados por uma artista – Cygny Malvar e disponibilizados para a criação do jogo, mantendo-se reservados todos os seus direitos.

Figura 7 Adereço 1

Figura 8 Adereço 2

Figura 9 Adereço 3

2.3. Anatomia de uma cena

Os elementos que constituem uma cena de um jogo de computador, são comuns à maioria de jogos, podendo haver pequenas diferenças consoante se trate de uma plataforma móvel ou estática.

2.3.1. Fundo (background)

Elemento que tal como indica o nome, fica no fundo da cena, ajuda a criar o ambiente desejado para a cena, geralmente o nível de detalhe é baixo, para que não se misture com os outros elementos.

2.3.2. Cenário (scenery)

Elementos que constituem a cena, podem ser apenas decorativos ou podem interagir com os atores, podem ser estáticos ou animados.

2.3.3. Barra de estado (status bar/heads-up-display)

Fornece ao jogador informação sobre o estado do jogo (Wikipedia, 2017), são elementos gráficos que não fazem parte da ação principal, e encontram-se nos extremos da área de jogo estando sempre visíveis, mas interagem com o jogador fornecendo-lhe informação importante, os elementos mais comuns são:

1. Mini mapa ou direção a seguir – Mapa ou apontador que indica a direção a seguir
2. Número de vidas ou energia – Número de vidas ou energia disponível
3. Pontos – Quantidade de pontos
4. Tempo disponível – quando existe um tempo limite
5. Inventário – objetos que o personagem transporta consigo
6. Menu - Acesso ao menu de saída

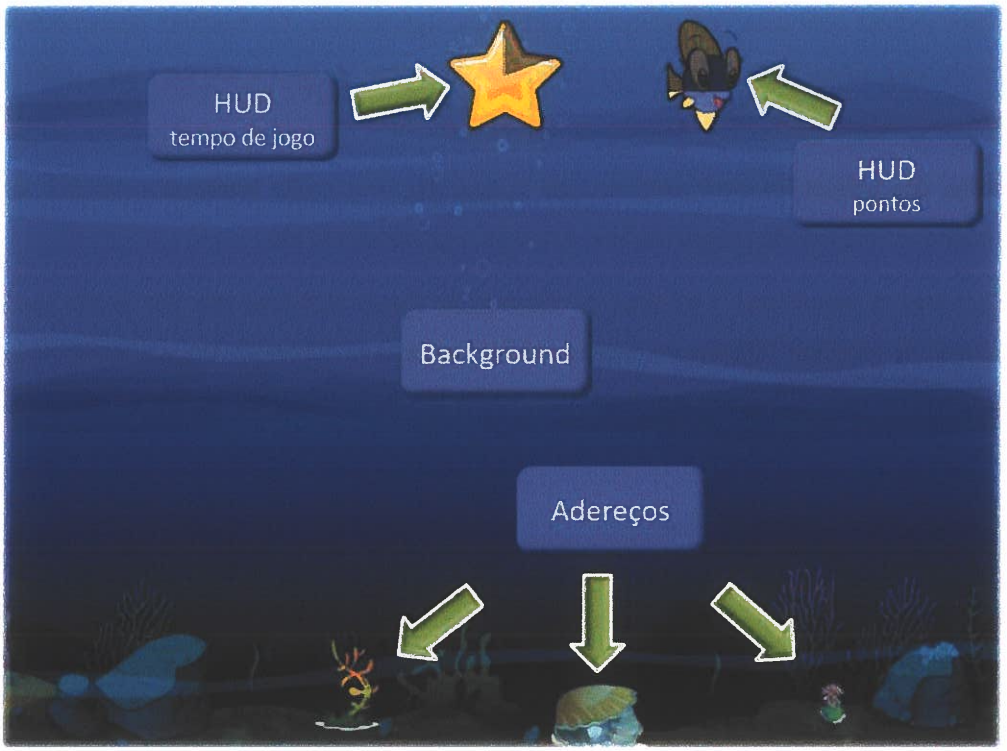

Figura 10 Anatomia de uma cena

2.4. Aplicações

O desenvolvimento do projeto foi suportado por uma seleção de aplicações, os critérios usados variaram para cada uma das aplicações, mas foi definido um conjunto de requisitos genéricos que foram os seguintes:

- Compatibilidade com MS Windows 10
- Gratuitas (a versão escolhida) ou baixo custo
- "Open-source"
- Curva de aprendizagem pequena

As aplicações dividiram-se em 4 grupos

- Programação e desenho
- Edição de imagem
- Edição de áudio
- Repositório e partilha de informação e controlo de tarefas

Sendo um projeto com uma componente multimédia significativa, foi bastante importante a escolha de aplicações para o tratamento de áudio e imagem, devido a ser uma área onde a experiência é reduzida.

A escolha das aplicações para programação e desenho teve como principais critérios os conhecimentos técnicos de programação e o tipo de jogo desenvolvido.

As aplicações de repositório e partilha de informação foram incluídas não por existir uma necessidade de partilhar informação, como seria imperativo se o projeto fosse desenvolvido por uma equipa com diferentes responsabilidades, mas num esforço de incluir as melhores práticas.

2.4.1. Programação e desenho

A programação e desenho são o principal enfoque deste projeto, foram os dois elementos onde se encontra a maior parte do esforço, as aplicações selecionadas são ambas gratuitas:

- Unity Personal edition
- MS VS 2015 Community edition

UNITY PERSONAL EDITION

É uma das plataformas mais popular para desenvolvimento de jogos, os fatores que levaram sua a escolha foram:

- Usar a linguagem de programação C# .Net
- Otimizado para o desenvolvimento de jogos 2D
- Permite a edição de código e configuração em tempo real
- Documentação on-line bastante completa

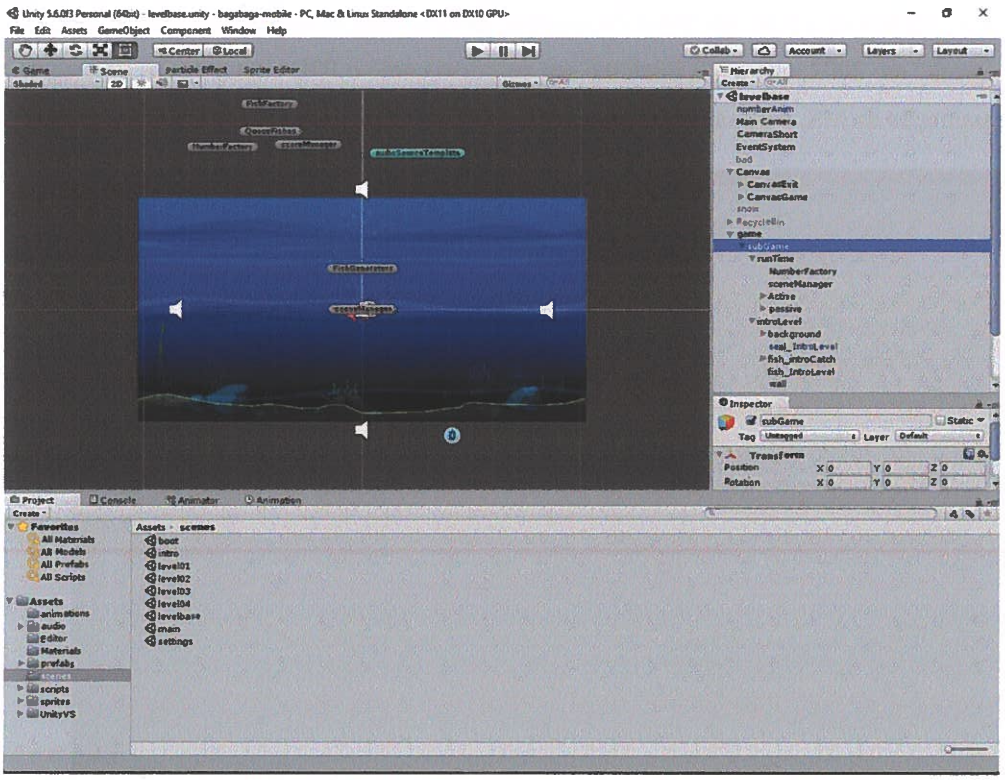

Figura 11 Unity IDE

MICROSOFT VISUAL STUDIO 2015 COMMUNITY

O "Visual Studio" é o IDE da Microsoft, sendo C# uma das linguagens de programação da MS, a escolha do VS é obvia, mas o VS está de acordo com os critérios de seleção, tendo também as seguintes características:

- Integração com Unity
- Permitir depurar a execução do Unity
- É o IDE que uso em todos os projetos em que uso C#.

"Visual Studio" é o IDE da MS, foi inicialmente desenvolvido para o MS VS, mas ao longo do tempo tem sido usado nos mais variados produtos da MS entre os quais "Microsoft SQL Management Studio" ou "SQL Server Data Tools", permitindo à MS um ambiente comum para diferentes plataformas de desenvolvimento.

A primeira versão do VS foi em 1997 (Wikipedia, 2017), mas foi com o lançamento da .Net "Framework" em 2002, que o VS iniciou o seu percurso como o IDE para todas as linguagens de programação da MS, a versão mais recente é a 2017.

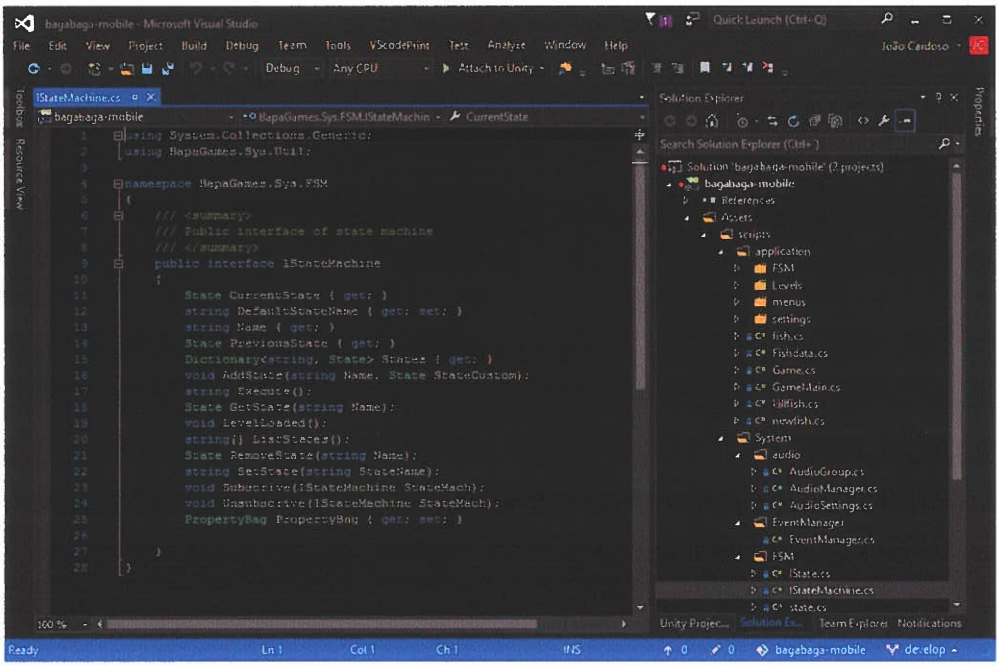

Figura 12 Visual Studio 2015 Community

2.4.2. Edição de imagem

Para a edição de imagem foi escolhido a GIMP, que disponibiliza as funcionalidades mais comuns para a edição de imagem, está de acordo com os requisitos de seleção e é a aplicação que uso para a edição gráfica, tendo por isso alguma familiaridade com a mesma.

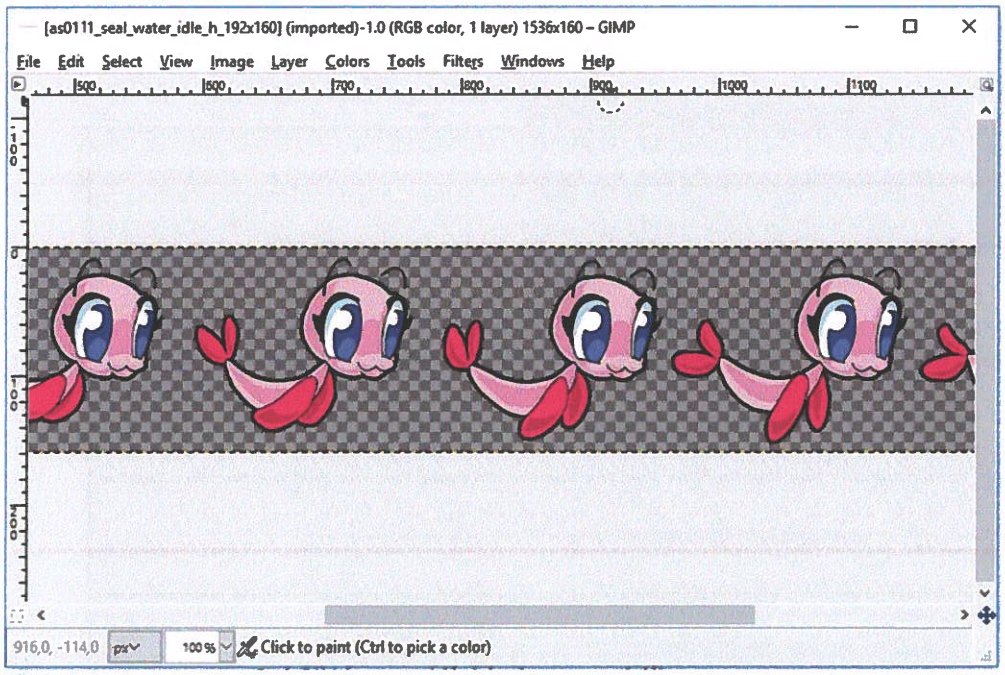

Figura 13 GIMP

2.4.3. Edição de áudio

Para a edição de áudio foi escolhido o "*Audacity*", que disponibiliza funcionalidades para a captura e edição de áudio, preenche os requisitos de seleção e revelou-se fácil de usar.

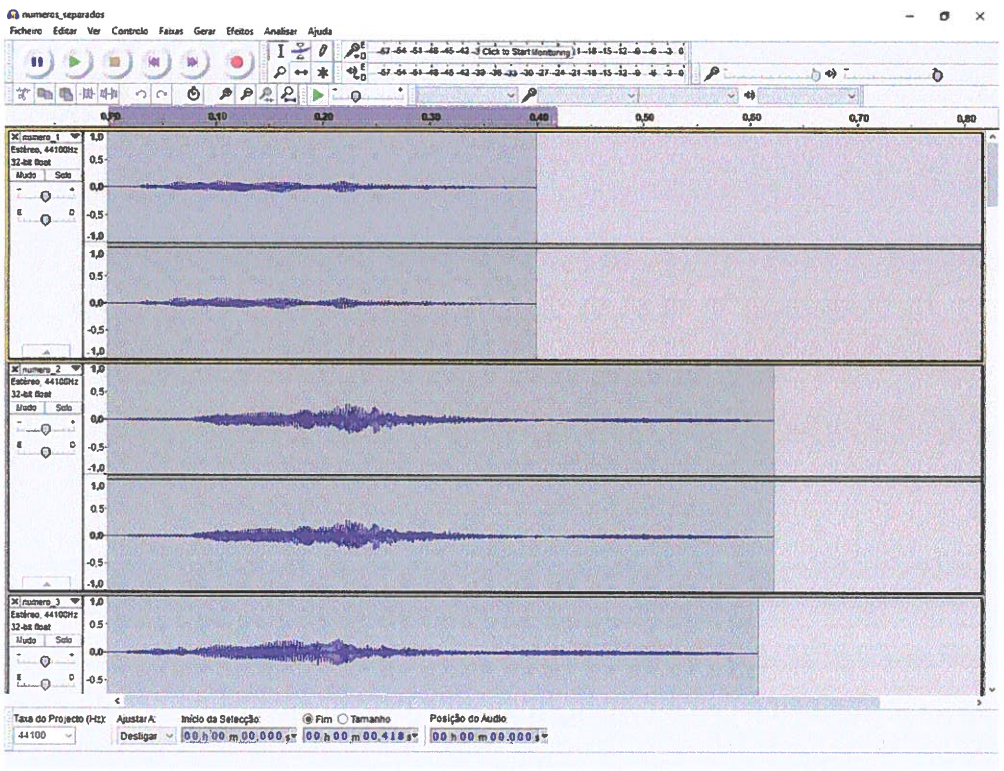

Figura 14 Audacity

2.4.4. Repositório de documentação e controle de tarefas

Para esta área foram selecionadas duas aplicações que são a exceção relativamente ao custo, estão disponíveis para teste por um período de 30 dias, mas essa hipótese foi colocada de lado dado queo projeto teve uma duração superior.

Foi adquirida uma licença até 10 utilizadores, com instalação em servidor local, tendo esta opção um custo de $10 por cada uma das aplicações, incluindo a tarefa de instalação local e a sua respetiva configuraçãoe a instalação da BD PostgreSQL

Ambas as aplicações são da companhia Atlassian

- Confluence – Repositório de documentação
- Jira Software – Controle de tarefas

Confluence

Mesmo tendo apenas um recurso, a utilização do "Confluence" permitiu documentar as decisões e ideias relativas ao desenvolvimento.

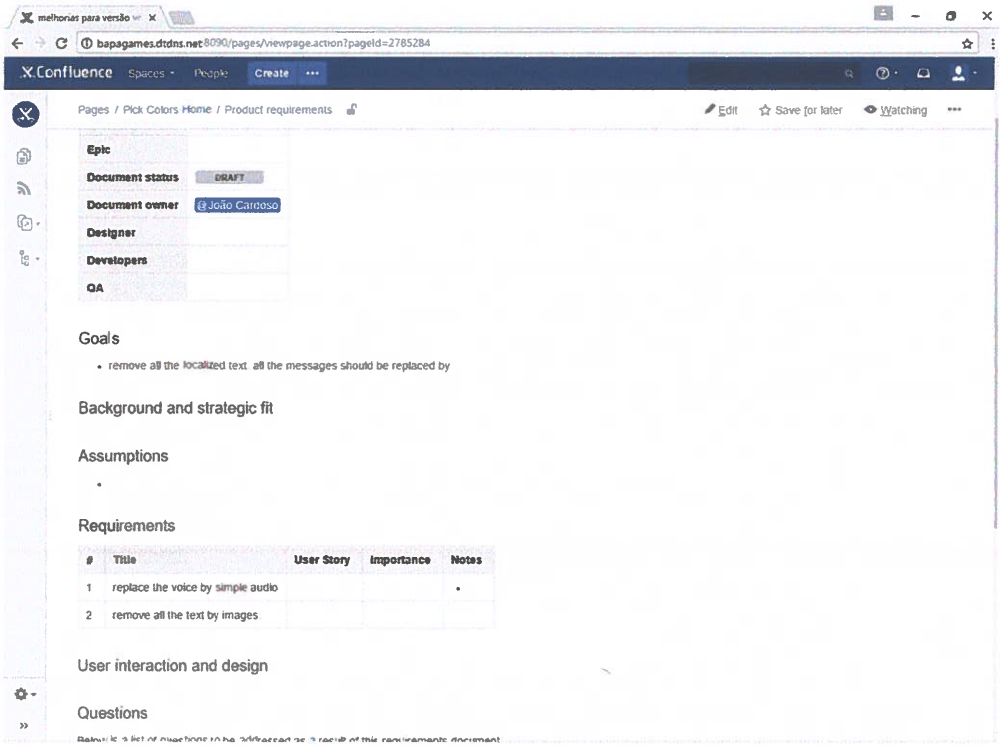

Figura 15 Confluence

Jira Software

O controlo de tarefas foi menos eficaz devido a natureza do projeto e a imprevisibilidade da disponibilidade do recurso, não tendo sido usado como inicialmente decidido.

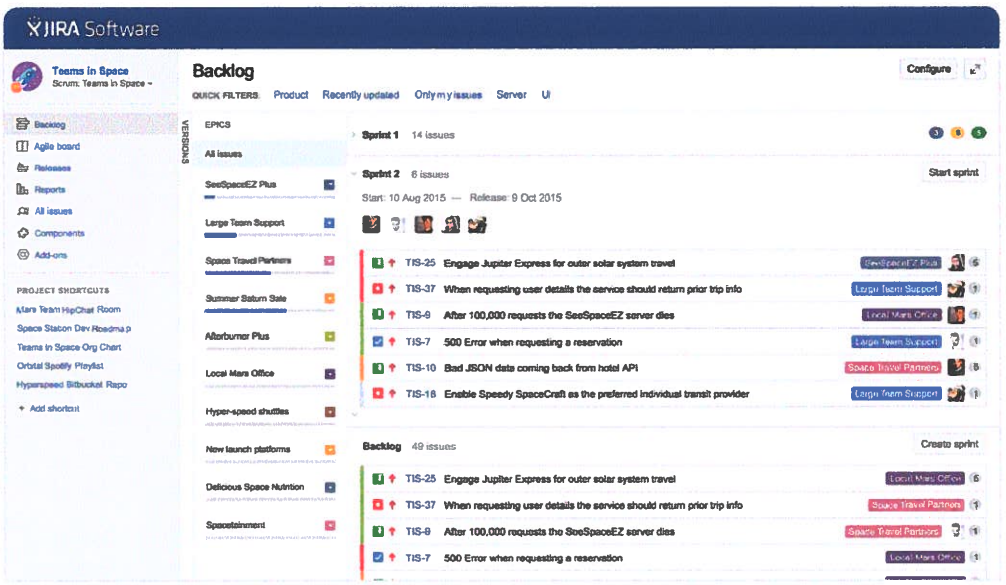

Figura 16 Jira Software

2.5. Fluxo do jogo

Este capítulo vai detalhar como está organizado o fluxo do jogo,

2.5.1. Diagrama de fluxo nível 0

Diagrama de nível 0, detalha as interações entre os principais blocos, o objetivo é dar uma visão simplificada do fluxo.

- Iniciar o jogo
- Menu principal onde se encontram as principais opções do jogo
- Configurar o áudio
- Jogar o nível
- Caso tenha passado o nível, passa para o próximo nível
- Caso não tenha passado o nível vai para o menu principal

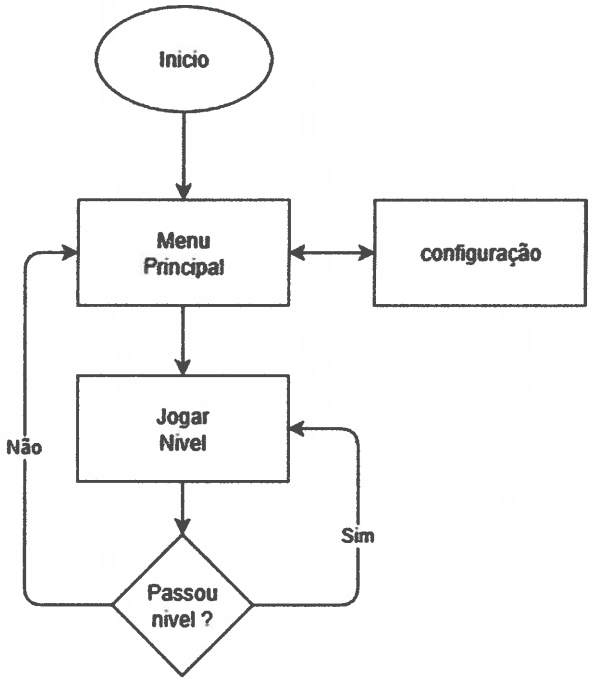

Figura 17 Diagrama de fluxo nível 0

2.5.2. Diagrama de fluxo nível 1

Diagrama de nível 1, detalha as interações com mais detalhe, permitindo uma melhor compreensão do fluxo.

- Iniciar o jogo
- Menu principal onde se encontram as principais opções do jogo
- Configurar o áudio, permite configurar o volume da música de fundo e efeitos especiais
- O jogador pode sair do jogo
- Carregar o próximo nível, configura o aspeto gráfico e define qual o peixe a apanhar e quantos, assim como o tempo disponível para atingir o objetivo
- Jogar o nível
- Durante o processo "jogar o nível", o jogador pode sair em qualquer altura, voltando ao menu principal
- Caso não tenha passado o nível vai para o menu principal
- Caso tenha passado o nível, verifica se é o ultimo nível, caso não seja passa para o próximo nível, se for vai para o menu principal

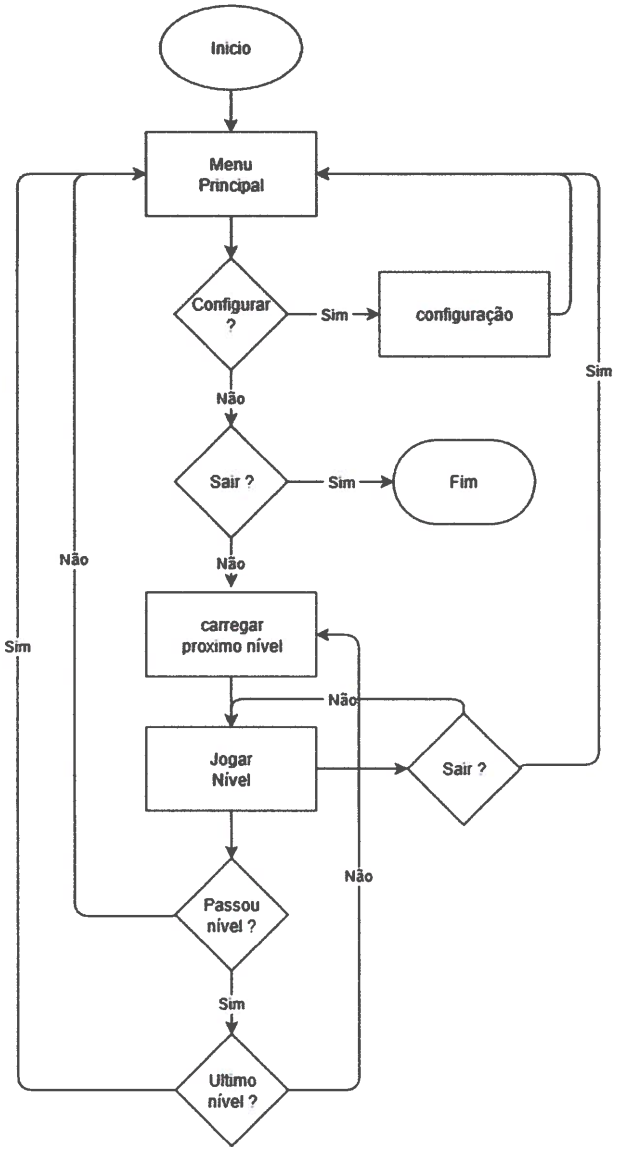

Figura 18 Diagrama de fluxo nível 1

2.5.3. Diagrama de fluxo nível 1 com imagens

O mesmo diagrama de nível 1, mas com imagens retiradas do jogo, permitindo uma visualização dos vários processos.

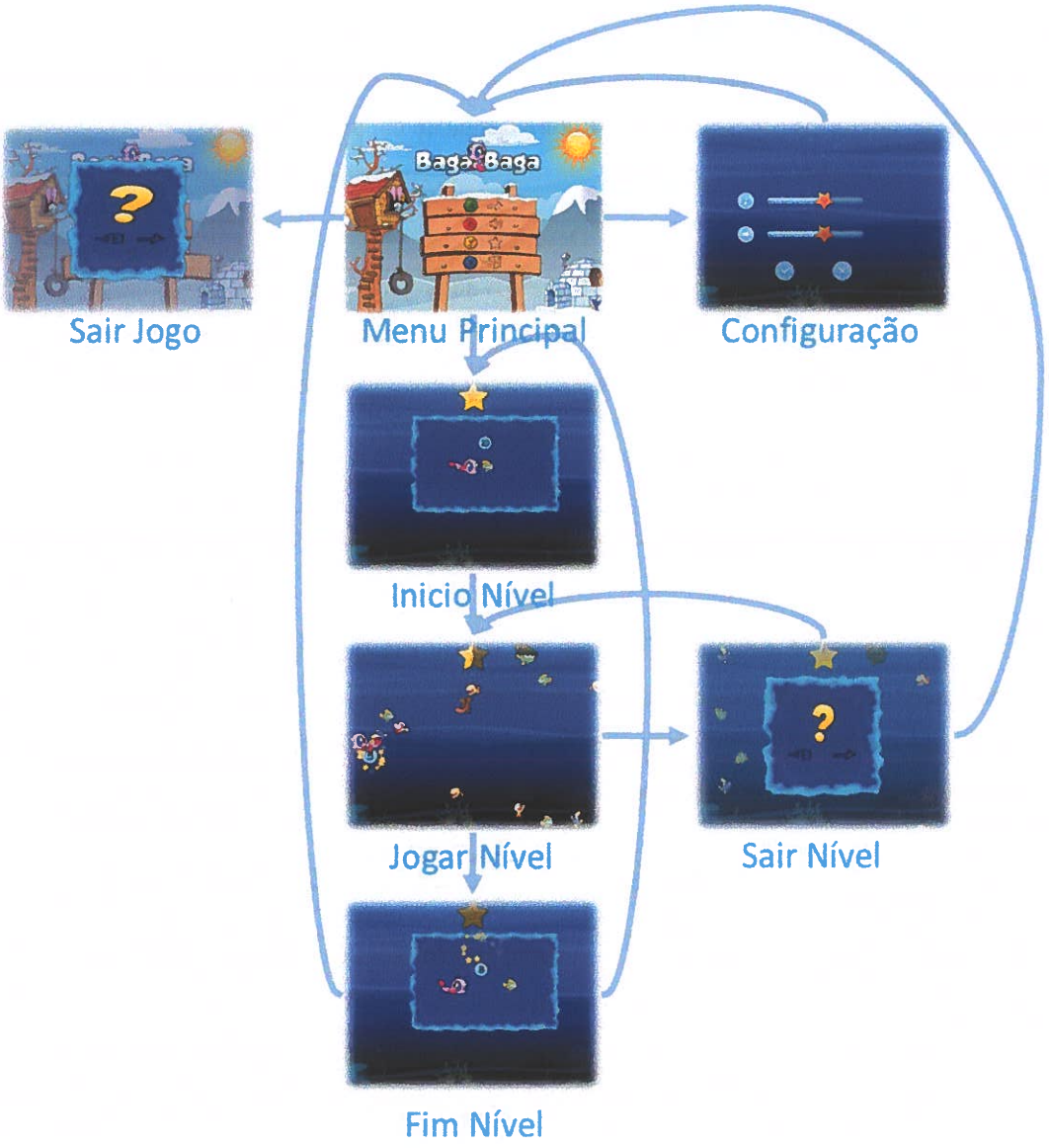

Figura 19 diagrama de fluxo nível 1 com imagens

2.6. Estrutura do projeto

Os projetos em UNITY têm uma estrutura de diretórios obrigatória, podendo o utilizador adicionar os diretórios necessários. A estrutura básica de um projeto de UNITY é a seguinte:

Tabela 1 Directórios pincipais de um projeto Unity

Diretório	Descrição
<projeto>	Ficheiros de projeto para VS
assets	Todos os ficheiros específicos do projeto
Library	Bibliotecas necessárias para o funcionamento do IDE do UNITY
projectSettings	Configurações do projeto

2.6.1. Assets

O diretório "Assets" é a raiz dos ficheiros que constituem o projeto, o Unity permite que o programador defina a sua própria estrutura de diretórios, mas existe um standard que por norma é usado, no entanto diferentes "plug-ins" podem requerer diferentes diretórios. Este projeto usou a seguinte estrutura de diretórios:

Tabela 2 Estrutura de diretórios do jogo

Diretório	Descrição
animations	Animações definidas usando a funcionalidade "animator" do UNITY
audio	Ficheiros de áudio, música e efeitos especiais
Editor	Scripts para estender as funcionalidades do editor UNITY
Materials	Matérias pré-definidos
Prefabs	Objetos com configuração pré-definida (UNITY, 2012)
Scenes	Definição dos diferentes ecrãs do jogo
Scripts	Scripts com a lógica do jogo
Sprites	Imagens usadas no jogo, incluindo animações
UnityVS	Scripts necessários para integração com MS VS

2.7. Programação

2.7.1. Arquitetura do Unity

O Unity usa uma arquitetura de objetos e componentes, todos os objetos num jogo são um "GameObject" (GameObject, 2017), quer isto dizer que tudo o que existe no jogo têm que ser um "GameObject", mas, um "GameObjecto" não faz nada sozinho, é necessário adicionar-lhe propriedades para que possa ser um ator, um som.

O "GameObject" deve ser visto como um contentor, ao adicionarmos funcionalidades passa a ser um ator, um som, ou qualquer outra coisa. Cada uma dessas funcionalidades chama-se **componente**.

Dependendo do tipo de objeto que queremos criar, combinamos diferentes componentes. Unity têm bastantes componentes de raiz, podendo desenvolver-se qualquer tipo de componente usando as linguagens de "script" disponíveis em Unity (C# e Java).

Um componente desenvolvido em C# é uma classe, com herança da classe "MonoBehaviour", que contêm a API de "scripting" do Unity, através da class "MonoBehaviour" é possível integrar os componentes proprietários com os restantes componentes do Unity.

A figura mostra um exemplo de um componente desenvolvido para o jogo, neste caso implementa dois eventos, "awake" e "update".

```
using UnityEngine;
using System.Collections.Generic;
using BapaGames.Sys;

namespace BapaGames.Appl
{
    /// <summary>
    /// Setup the game object
    /// Main entry point of default State Machine
    /// </summary>
    public class GameMain : MonoBehaviour
    {
        public Game game;
        /// <summary>
        /// Constructor
        /// </summary>
        void Awake()
        {
            game = Game.Instance();
            game.GameStatus = GameStatus.booting;
            game.Setup();
            game.GameStatus = GameStatus.Running;
        }

        void Update()
        {
            game.Update();
        }
    }
}
```

Figura 20 Exemplo de componente Unity

Durante a execução do jogo, Unity comunica com os diferentes componentes através de eventos, o seguinte diagrama descreve esses eventos e a ordem de execução dos mesmos, os eventos com o fundo a amarelo são os mais usados neste projeto.

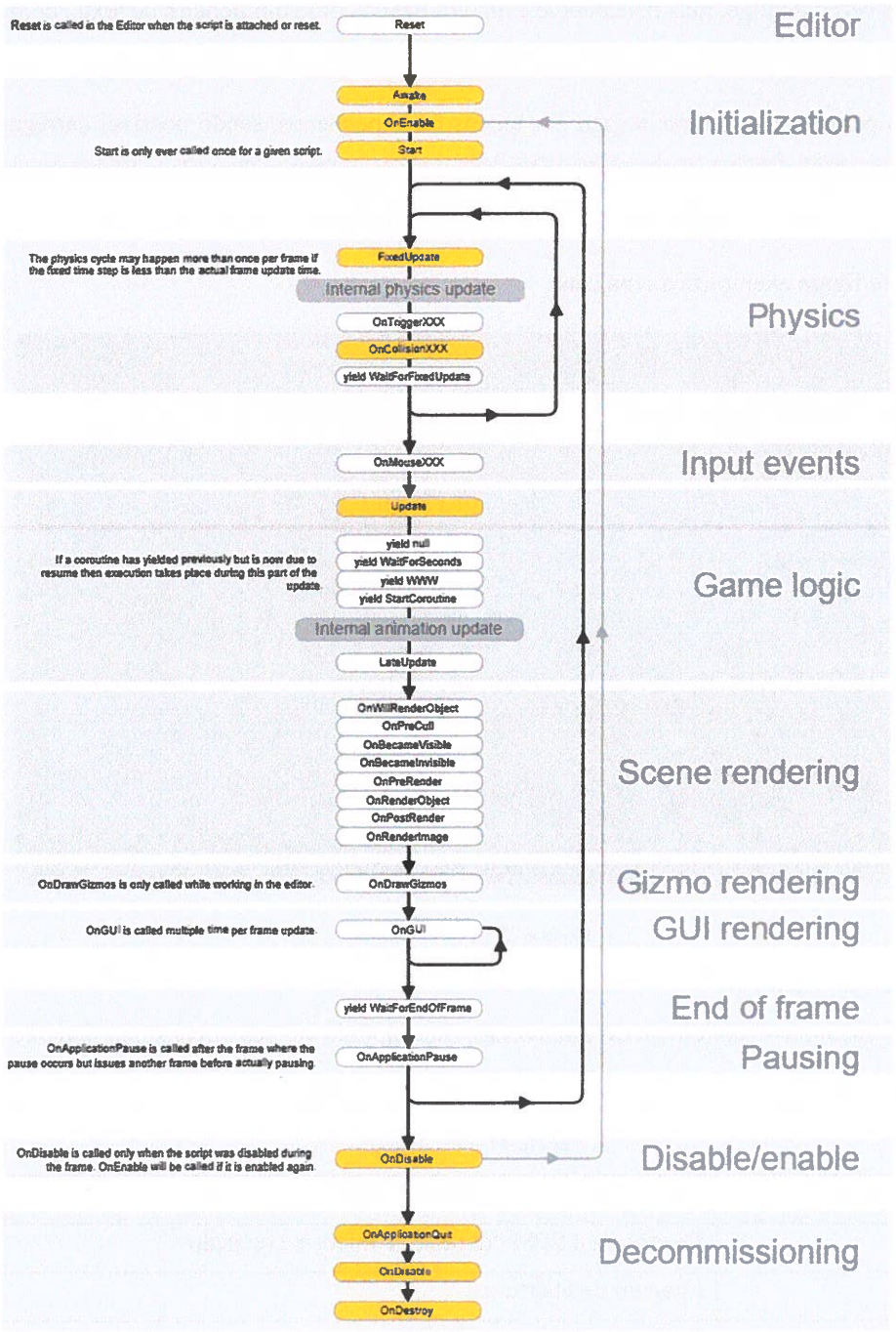

Figura 21 ordem execução eventos em Unity

2.7.2. Cenário

O cenário que passaremos a chamar de cena (Unity, 2017), é um repositório de "GameObjects", que juntos constituem uma parte do jogo, podendo ser apenas a imagem de abertura, Menus e

diferentes cenas do jogo, mas fisicamente é materializado como um ficheiro de texto, com uma estrutura que usa YAML.

As cenas permitem estruturar o jogo em blocos mais pequenos, sendo possível carregar em memória apenas os elementos que são necessários, esta arquitetura faz toda a diferença quando o volume de gráficos e áudio é grande, mas não é utilizada em simultâneo.

A seguinte figura exemplifica uma cena.

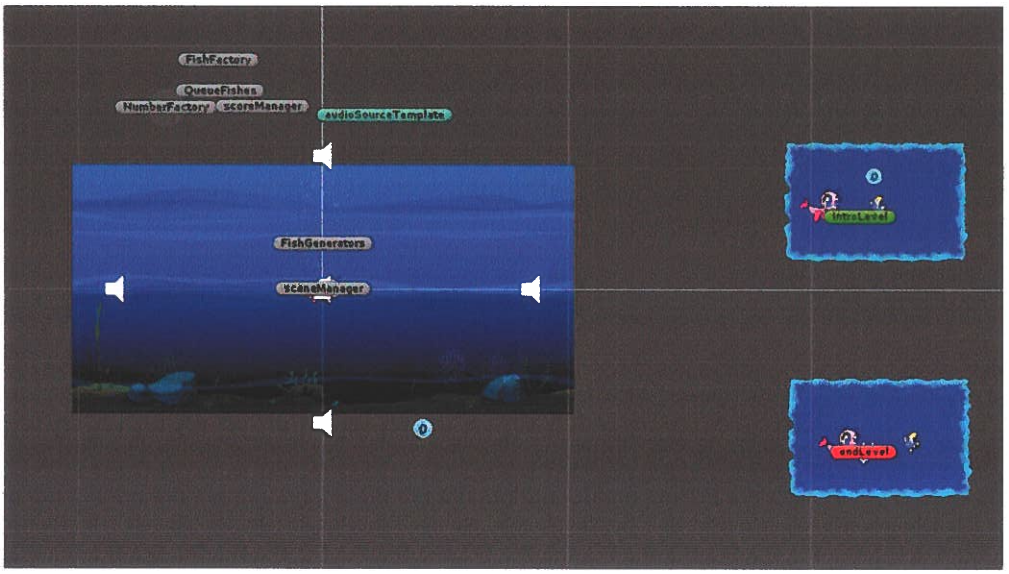

Figura 22 Cena do Unity

2.7.3. Cenários

O projeto encontra-se divido em diversas cenas, este capítulo vai descrever sumariamente cada uma das cenas.

Tabela 3 Cenas do jogo

Nome da cena	Descrição
Boot	Configura a SFM "default" e inicia a execução
Intro	Imagem de abertura
Main	Ecrã principal com os menus
Settings	Configuração do áudio
Levelbase	Configuração do nível base
Level01	Configuração do nível 1
Level02	Configuração do nível 2

Level03	Configuração do nível 3
Level04	Configuração do nível 4

Boot

Esta Cena é só para efetuar o arranque da aplicação, não contem elementos gráficos ou áudio.

É configurada a classe principal ("game") do jogo, que inclui a FSM "default", é iniciado o principal fluxo do jogo.

O "GameObject" "BootInit" aplica o componente "retain" que força o "GameObject" a ficar residente em memória, não sendo removido quando for descarregada a cena.

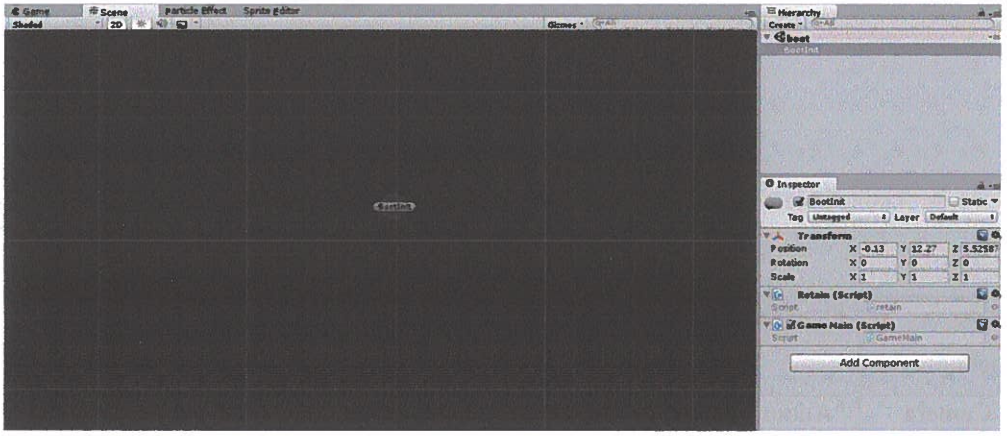

Figura 23 Cena "Boot"

Tabela 4 Objetos da cena "boot"

"GameObject"	Componentes
BootIni	"retain" – Marcar "GameObject" para não ser descarregado da memória "GameMain" – Configuração do SFM "default" e fluxo principal do jogo

Intro

É apresentado o ecrã de início também conhecido por "Splash-Screen", que tem como propósito a apresentação do nome do estúdio que desenvolve o jogo.

É efetuado um fade-in do logotipo enquanto se ouvem crianças a rir, terminando com as crianças a dizer "Bapa Games".

Figura 24 Cena "intro"

O número de "GameObject" é reduzido, a seguinte tabela detalha os objetos.

Tabela 5 Objetos da cena "intro"

Nome do "GameObject"	Funcionalidade/componentes
Main Camera	"Audio Listener" – Captura a áudio da cena "Camera" – Captura da imagem da cena
Bapa_logo	"fade" – componente para fazer o "fade-in" da imagem "Audio Source" – componente que emite som "Play Sound" – tocar o audio

Main

É a cena principal, onde o jogador pode iniciar o jogo, configurar o áudio e sair do jogo.

É constituído por um fundo estático, com bastante detalhe gráfico, os menus são animados, o jogador pode selecionar as opções de menu usando o teclado, o rato ou os dedos no caso de dispositivos "*touch*", enquanto ouve uma música de fundo.

Figura 25 Cena "Main"

A cena não é complexa, mas o número de objetos já é superior a 30, a seguinte tabela vai detalhar os principais blocos.

Tabela 6 Objetos da cena "main"

Nome do "GameObject"	Funcionalidade/componentes
Groupmenu	Controla animação dos menus e escolha dos mesmos
CanvasUI	Quando o jogador escolhe opção de sair do jogo, é apresentado um ecrã de saída.
Players	Controlo o movimento do cursor
Worldlimits	Define a área em que o cursor se pode movimentar

LevelBase

É a cena central do jogo, é nesta cena que se passa a ação do jogo, além da cena principal, contem mais duas cenas que são apresentadas aquando o início e o fim do nível.

Quando é iniciado o nível, é apresentado uma explicação onde se mostra o peixe a apanhar e a quantidade pretendida.

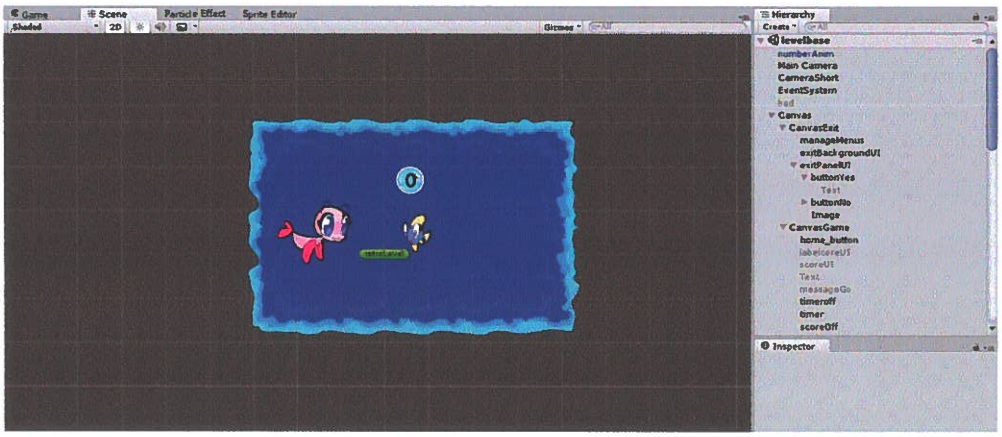

Figura 26 Cena "LevelBase" Inicio de nível

Cena principal onde decorre a acão do jogo.

Figura 27 Cena "LevelBase" área de jogo

Quando termina o nível, é efetuada a contagem do número de peixes da cor indicada efetivamente apanhados, e um efeito sonoro que indica se passou de nível ou não.

Sendo a principal cena do jogo o número de "GameObjects" ultrapassa os 200, sendo demasiado exaustivo a sua descrição, apenas estão detalhados os principais grupos.

Tabela 7 Objetos da cena "levelbase"

Nome do	Funcionalidade/componentes

"GameObject"	
Canvas\CanvasExit	Quando o jogador escolhe opção de sair do jogo, é apresentado um ecrã de saída.
Canvas\CanvasGame	Objetos de UI, usados durante ação do nível
Game\subgame\ introlevel	Lógica da cena de introdução do nível
Game\subgame\ Endlevel	Lógica da cena de fim do nível
Game\subgame\ Runtime\active	Lógica de todos os elementos ativos do jogo, todos os atores
Game\subgame\ Runtime\passive	Lógica de todos os elementos passivos do jogo, limites da área de jogo.

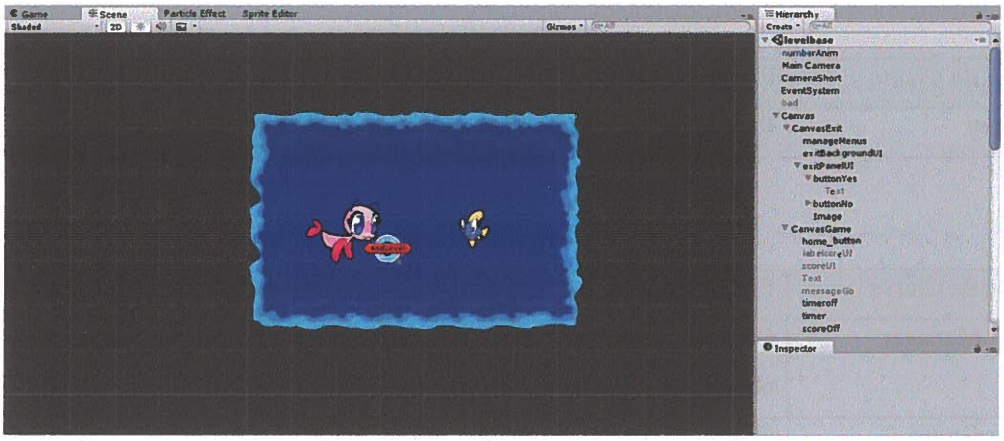

Figura 28 Cena "LevelBase" fim de nível

2.7.4. Estrutura do código

Todo o código do projeto encontra-se no diretório "scripts", é este diretório que é a raiz do projeto C#. O projeto está organizado de maneira a que o código possa ser reutilizado noutros projetos.

O primeiro nível de diretórios divide-se em 3 grupos:

- **System** – código que é genérico e pode ser usado noutros projetos
- **Application** – Código específico para este projeto

- **Util** – Utilitários que podem ser específicos do projeto ou de sistema

System

Este diretório contém os componentes de sistema e encontra-se subdivido em 3 áreas:

- **Audio** – Gestão da configuração de áudio e efeitos sonoros
- **FSM** – implementação do modelo de programação "finit-state-machine"
- **Game** – componentes que gerem o jogo

Principais classes

Este capítulo vai detalhar as várias classes e a sua funcionalidade

O diretório de "áudio" contém as seguintes classes

Tabela 8 classes do diretório "system"

Nome da Classe	Funcionalidade
Factory	Repositório de "GameObjects" mais usados

Tabela 9 classes do diretório "audio"

Nome da Classe	Funcionalidade
AudioManager	Controla todos os "GameObject" de áudio, permite agrupar os objetos de áudio em diferentes categorias.
AudioGroup	Grupo de objetos de áudio com as mesmas características ex. Música de fundo.
audioEffectManager	Toca pequenos efeitos de áudio que devem ser destruídos uma vez que terminem (ex. explosões)
AudioSettings	Armazena o volume da música e efeitos especiais

Tabela 10 classes do diretório "FSM"

Nome da Classe	Funcionalidade
StateMachine	Implementação da "Finit State Machine"
State	Implementação abstrata do estado, para ser usado para implementar os estados das FSM

IStateMachine	Implementação do Interface da FSM
IState	Implementação da interface do estado
State_Idle	Implementação do estado "idle"

Tabela 11 classes do diretório "game"

Nome da Classe	Funcionalidade
GameBase	Implementação abstrata do jogo, para ser usado para implementar a classe do jogo
GameStatus	Diferentes estados do jogo
IStateMachine	Implementação do Interface da FSM
LevelsManager	Gestor dos diferentes níveis do jogo
Level	Implementação abstrata da definição de um nível, para ser usado na definição dos níveis

Application

Este diretório contém os componentes específicos do jogo e encontra-se subdivido em 5 áreas:

- **FSM** – Configuração da FSM e implementação dos diferentes estados
- **Levels** – Lógica do nível base, incluindo FSM, configuração dos níveis
- **Controller** – Lógica do ator principal, interface com o jogador
- **Menus** – Definição e lógica dos menus
- **Settings** – Configuração do áudio

Tabela 12 classes do diretório "applications"

Nome da Classe	Funcionalidade
FishCode	Código numérico para cada um dos atores secundários (peixes)
Fishdata	Referência entre o código e um identificador
Game	Implementação da classe abstrata "GameBase"
GameMain	Inicialização do jogo e controlo de execução

killfish	Destruir um "GameObject" do tipo peixe
HomeButton	Controla saída durante a execução no nível

Tabela 13 classes do diretório "controller\palyer"

Nome da Classe	Funcionalidade
Guide	Mover o ator principal com o rato ou toque (dispositivos "touch")
Keyb	Mover o ator principal com o teclado
Ghost	"GameObject" que gere a física do ator principal

Tabela 14 classes do diretório "FSM\states"

Nome da Classe	Funcionalidade
State_GameBegin	Configura o jogo
State_GameEnd	Liberta a memória ocupada pelo nível personalizado
State_GameLevelHaltedMenus	Durante a execução do nível, gere a saída para o menu principal
State_GameIntro	Abertura do jogo quando iniciado
State_GameLevelBegin	Inicia um jogo
State_GameLevelEnd	Termina um nível, decide se segue para o próximo nível ou termina o jogo
State_GameLevelLoad	Carrega em memória o nível base
State_GameLevelLoadCustom	Carrega em memória o nível personalizado
State_GameLevelRunning	Verifica se o nível chegou ao fim ou se o jogador pediu para sair
State_GameLevelUnloadCustom	Liberta a memória ocupada pelo nível personalizado
State_GameMain	Gere o menu principal
State_GameMainExit	Quando o jogador escolhe sair do menu principal, pede uma confirmação
State_GameSettings	Configuração do áudio

Tabela 15 classes do diretório "levels\data"

Nome da Classe	Funcionalidade

LevelData	Dados de configuração do nível personalizado

Tabela 16 classes do diretório "levels\levelbase"

Nome da Classe	Funcionalidade
LevelBase	Dados de configuração do nível base
managerSceneLevel	Configura nível base com dados de personalização
introSceneManager	Configura a explicação do nível
instructionsKillFish	Controle dos atores da explicação no nível
scoreSceneManager	Configura visualização do resultado do nível
SetFishCatcher	Configura qual o peixe apanhar na explicação do nível

Tabela 17 classes do diretório "levels\levebase\states"

Nome da Classe	Funcionalidade
State_LevelBegin	Inicializa o nível
State_LevelEnd	Fim do nível
state_LevelIntro	Explica ao jogador qual o peixe apanhar e a quantidade
State_LevelRunning	Configura a execução do nível
State_LevelShowScore	Mostra o resultado do nível

Tabela 18 classes do diretório "menus"

Nome da Classe	Funcionalidade
CursorLimit	Controla os limites do cursor no menu principal
followmouse	Substitui o curso por um "GameObject" do tipo "sprite"
manageMenus	Controla os menus de Sim/Não

Tabela 19 classes do diretório "menus\main"

Nome da Classe	Funcionalidade
MenuAnim	Anima o menu selecionado

MenuAction	Define qual a ação do menu quando escolhido
groupmenus	Gestor de menus

Tabela 20 classes do diretório "settings"

Nome da Classe	Funcionalidade
SettingsActions	Lógica da configuração do áudio
settingsPlayEffect	Tocar um efeito de áudio durante a configuração do áudio

Tabela 21 classes do diretório "util"

Nome da Classe	Funcionalidade
audioKillEffect	Toca e destrói efeito de áudio
audioRegister	Regista componente de áudio no gestor de áudio
CacheAndDisable	Regista o "GameObject" na cache e desabilita-o
CacheObjects	Gestor de cache de "GameObjects"
Layers	Código numérico das principais camadas ("layers") do jogo
Fade	Efetua o "fade" de entrada e saída de uma imagem, com valores programados
fadeManag	Efetua o "fade" de entra e saída de uma imagem
FadingAudioSource	Efetua o "fade" de entrada e saída de audio
Extensions	Utilitários para a gestão da cache de "GameObjects"
fixedPosition	Fixa um "GameObject" em um determina posição
Kill	Destrói um "GameObject"
killtimer	Destrói um "GameObject" depois de número de segundos configurado
Math	Funções matematicas
PlaySound	Toca audio depois de um número de segundos configurado
PropertyBag	Gestor de propriedades
propBag<T>	Gestor de propriedades de um tipo de dado específico

Property<T>	Propriedade de um determinado tipo de dado específico
Retain	Configura o "GameObject" para nunca ser retirado de memória
screenratio	Configura a resolução do jogo
TimerAction	Efetua uma ação predefinida depois de um numero de segundos configurado

2.7.5. State Finit Machine

A implementação da FSM foi customizada para as necessidades do jogo e adaptada ao Unity.

Para ultrapassar as limitações de ter múltiplas FSM, que aumenta a complexidade das transições entre estados, foi implementado um modelo baseado na HSM, que permite que uma transição afete várias FSM, reduzindo o número de transições necessárias.Para o efeito foi usado um mecanismo de registo, em que cada uma das FSM se regista com as outras FSM.

Existem duas FSM no projeto, a primeira "default" controla o fluxo principal e a segunda "level_manager" controla a execução do nível.

FSM "*default*"

É responsável pelo fluxo principal da aplicação, controla todos os estados desde do seu arranque até a paragem, no total são 13 estados.

- *game_intro* – Carregamento do écran de abertura
- *game_main_menus* – Menu principal, permite ao jogador escolher entre as varias opções disponíveis
- *game_main_exit* – Quando o jogador escolher sair, é pedido uma confirmação antes de sair, permitindo não sair
- *game_settings* – Configurar o volume do áudio
- *game_begin* – Inicia um novo jogo
- *level_load* – Carrega o nível básico
- *level_load_custom* – Carrega o próximo nível e efetua a configuração do mesmo
- *level_begin* – Inicia o nível
- *level_run_long* – Monitorizar a execução do nível

- *level_halted_menus* – Quando o jogar escolhe sair durante a execução do nível, é pedido uma confirmação
- *level_end* – Termina o nível
- *level_unload_custom* – Descarrega a configuração nível
- game_*end* – Termina o jogo

O seguinte diagrama mostra os estados e as transições entre eles.

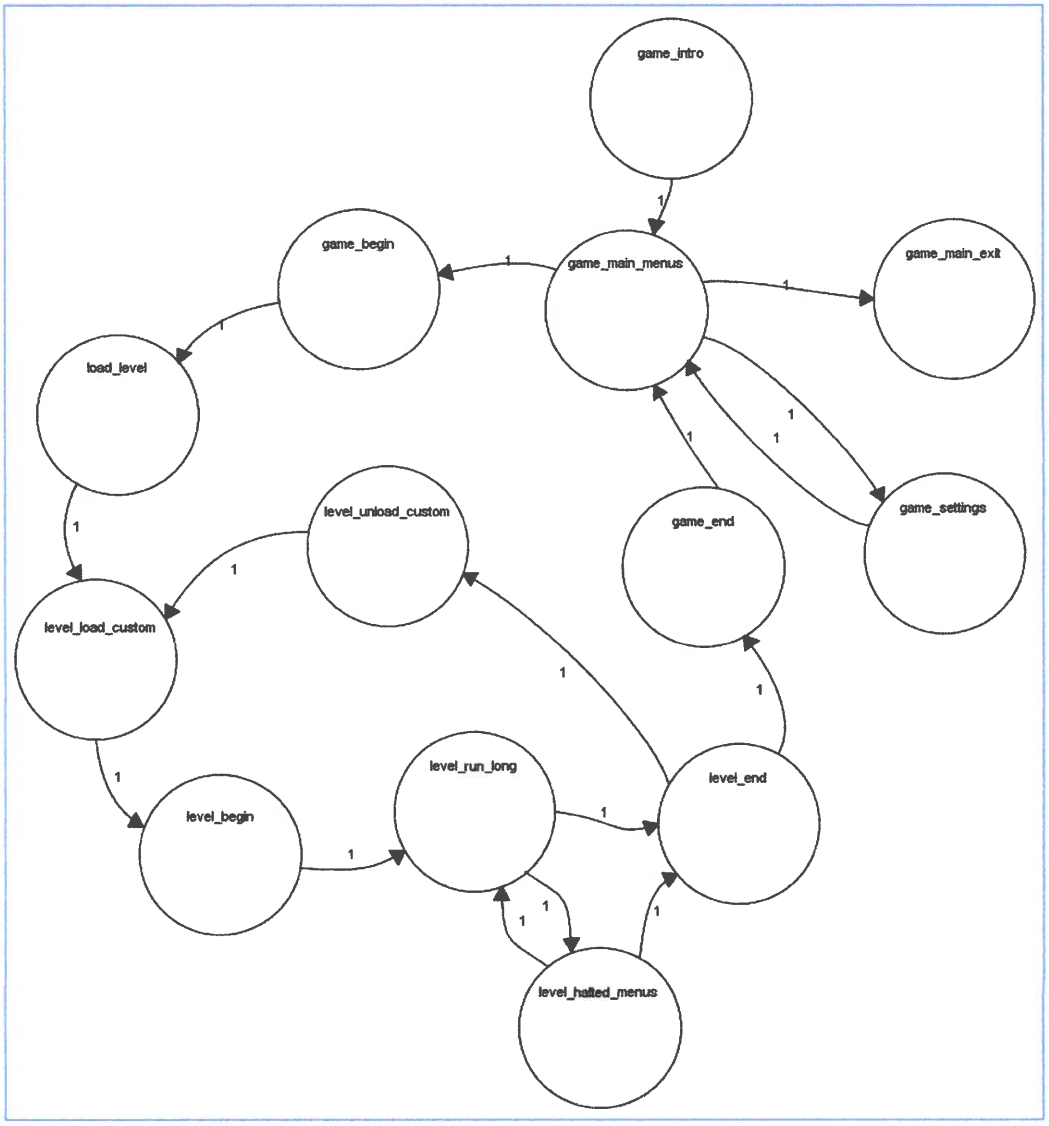

Figura 29 default FSM

A seguinte tabela mostra o mapeamento entre os estados e as classes de C#.

Tabela 22 mapeamento entre estados e classes, FSM "default"

Estado	Nome da Classe
game_intro	State_GameIntro
game_main_menus	State_GameMain
game_main_exit	State_GameMainExit
game_settings	State_GameSettings
game_begin	State_GameBegin
level_load	State_GameLevelLoad
level_load_custom	State_GameLevelLoadCustom
level_begin	State_GameLevelBegin
level_run_long	State_GameLevelRunning
level_halted_menus	State_GameLevelHaltedMenus
level_end	State_GameLevelEnd
level_unload_custom	State_GameLevelUnloadCustom
game_end	State_GameEnd

FSM "*level_manager*"

Esta FSM é ativada quando se inicia um nível do jogo, e é responsável pela sua gestão desde que se inicia até que termina, têm um total de 6 estados.

- *idle* – Em espera, não executa qualquer tipo de operação
- *level_begin* – Iniciar o nível
- *level_intro* – Explica qual o objetivo do nível
- *level_run* – Gere a execução do nível
- *level_score* – Mostra o resultado do nível
- *level_end* – Termina o nível

O seguinte diagrama mostra os estados e as transições entre eles.

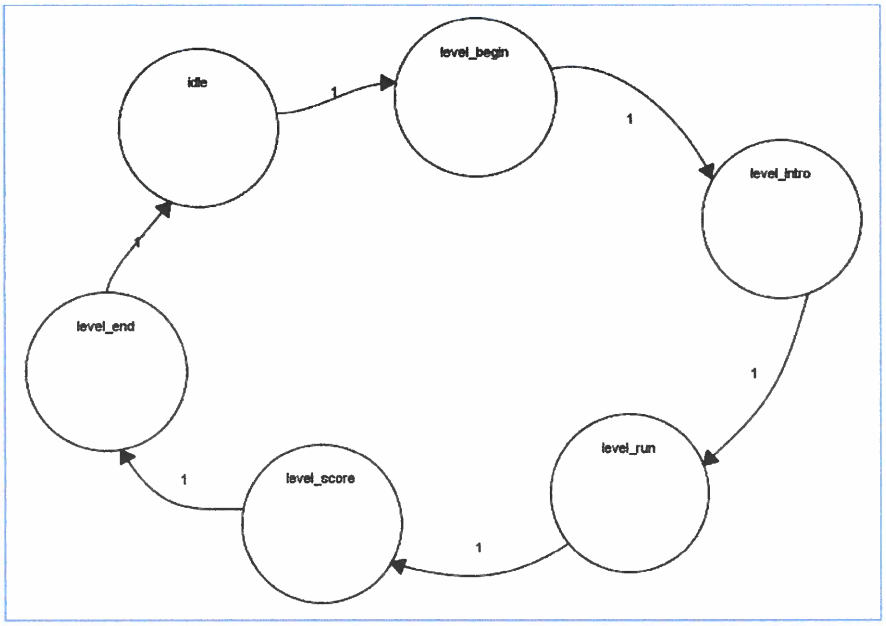

Figura 30 Level Manager FSM

A seguinte tabela mostra o mapeamento entre os estados e as classes de C#.

Tabela 23 mapeamento entre estados e classes, FSM "levelManager"

Estado	Nome da Classe
Idle	State_Idle
level_begin	State_LevelBegin
level_intro	State_LevelIntro
level_run	State_LevelRunning
level_score	State_LevelShowScore
level_end	State_LevelEnd

Os estados comuns que permitem a interação entre as duas FSM, têm sempre uma FSM que atua como emissor e as restantes FSM como recetores, a seguinte tabela explica quais os estados partilhados identificando o emissor e recetor.

Tabela 24 estados comuns entre FSM "default" e "levelManager"

Estado	FSM Emissor	FSM Recetor
level_begin	Default	level_manager
level_end	default, level_manager	default, level_manager

3. Conclusões

3.1. Objetivo

O objetivo do projeto era o desenvolvimento de um vídeo jogo, com uma componente didática, o grande desafio foi conseguir que a componente didática fosse eficaz, mas que não fosse o foco principal, este equilíbrio é bastante difícil, e a falta de jogos que sejam educativos e cativantes, levou à escolha deste tópico para o projeto.

O facto da maior parte dos jogos educativos serem pouco cativantes, devido à pouca importância que dão à experiência do jogo, faz com que rapidamente os jogadores procurem outro tipo de jogos mais apelativos, e com frequência menos educativos.

Um jogo onde o jogador está focado no jogo, mas têm um conjunto de objetivos que o obrigam a tomar atenção ao objetivo (apanhar um determinado número de peixes de uma cor específica), e a parte didática está incorporada na ação principal do jogo, foi a estratégia definida.

3.2. Implementação

O primeiro passo foi a definição da componente didática e lúdica, e o equilíbrio entre ambas, garantindo que a parte lúdica era dominante.

O prazo para a elaboração do projeto teve peso no desenho do jogo, pois sendo um projeto de um só recurso, objetivos demasiado ambiciosos podiam prolongar o tempo de execução para lá do tempo disponível.

Uma vez decidido o tipo de jogo, a mecânica do mesmo foi o desafio seguinte: criar um interface que fosse compatível com diferente tipos de dispositivos - PC usando o teclado, PC usando comando, PC usando rato, e dispositivos de toque (*Touch*), - e usando o motor de física disponível no Unity, foi possível criar uma mecânica de jogo que se aproximou dos movimentos aquáticos, aumentado a envolvência e a jogabilidade.

A componente gráfica é um pilar dos vídeojogos e porque estava fora do contexto do projeto a criação de gráficos, foi pedido a terceiros para participarem criando alguns gráficos específicos para este projeto, sendo também feito o "*download*" de gráficos mais simples de sites públicos, autorizando os criadores dos mesmos a sua utilização em qualquer projeto pessoal ou comercial.

A componente áudio é também muito importante, mesmo não requerendo escuta ativa, mas contribuindo muito para a envolvência do jogo; mais uma vez, sendo uma área muito específica, onde não tenho qualquer conhecimento, foi pedido a terceiros que criassem, as músicas e alguns efeitos de som, sendo também feito o "*download*" de alguns efeitos sonoros de sites públicos, sendo a sua utilização em projetos pessoais ou comerciais aprovada pelos criadores detentores dos mesmos.

O desenhar do jogo e a programação é o projeto em si, a linguagem de programação C# é algo que utilizo em projetos profissionais, o motor Unity, já tinha experimentado, mas nunca para implementar um jogo completo, houve toda uma aprendizagem usando distintas aproximações, leitura de livros, leitura de documentação "*on-line*", pesquisa em fóruns, e testes.

3.3. Dificuldades

O desafio foi juntar todos os elementos - Imagem, áudio e programação, usando o Unity. O modelo de programação do Unity não é difícil, mas é preciso algum tempo para se conseguir ser eficiente, foi uma aprendizagem que durou todo o projeto, ficando algumas áreas por explorar.

Houve alguma dificuldade em perceber como Unity organiza um "*GameObjecte*" e os seus componentes, e como distintos "*GameObjects*" comunicam entre si. Existem diversas opiniões, acabei usando um misto de opções seguindo o modelo que melhor se adaptava.

3.4. Conhecimentos adquiridos

Aprender a trabalhar com um dos mais reconhecidos motores de desenvolvimento de vídeojogos, é muito gratificante, foi mais um passo na direção de uma das minhas paixões: a produção de vídeojogos.

A investigação efetuada para o "Estado da Arte", foi interessante e permitiu aumentar o meu conhecimento da aplicação dos vídeojogos; a aprendizagem ao estudar as opiniões de especialistas, ajudou-me a ter uma perspetiva mais abrangente e detalhada, confirmando que os benefícios existem e estão comprovados.

3.5. Resultados

O resultado final foi muito recompensador e satisfatório, na medida em que os objetivos foram atingidos: o jogo têm uma componente lúdica forte, conseguindo manter a sua função educativa.

O jogo pode ser jogado em PC e dispositivos móveis, nesta primeira fase da plataforma MS, mas podendo ser disponibilizado em diferentes plataformas tais como Android e IOS entre outras.

3.6. Próximos passos

Continuar o desenvolvimento do jogo, adicionando mais níveis, diversificando o conteúdo dos níveis, criando diferentes níveis de dificuldade, adicionando áudio em diferentes línguas.

Investir na formação de Unity com o objetivo de fazer a certificação oficial da Unity, para o desenvolvimento de vídeojogos.

Aprofundar o conhecimento sobre DGBL, com leitura de livros e discussões em fóruns e assistir a workshops do tema.

Criar uma equipa para dar continuidade ao projeto, convidando pessoas das diferentes áreas, tais como gráficos, animação, música, com o objetivo de comercializar o jogo nas principais lojas "*on-line*".

Bibliografia

Abt, C. C. (1987). Serious Games. Em C. C. Abt, *Serious Games.* University Press of America.

Altassian. (2015). *Atlassian enables state-of-the-art computer science education at TUM.* Obtido de https://www.atlassian.com: https://www.atlassian.com/customers/TUM

Dodd, R. (04 de 04 de 2017). *why-collaboration-tools-matter*. Obtido de https://about.gitlab.com: https://about.gitlab.com/2017/04/04/why-collaboration-tools-matter/

DUFFY, J. (05 de 01 de 2017). *The Best Online Collaboration Software of 2017*. Obtido de http://www.pcmag.com: http://www.pcmag.com/article2/0,2817,2489110,00.asp

Eck, R. V. (01 de 05 de 2006). Digital Game-Based Learning: It's Not Just the Digital Natives Who Are Restless..... *EDUCAUSE Review, 42*, p. 16.

Eck, R. V. (12 de 10 de 2015). *digital-game-based-learning-still-restless-after-all-these-years*. Obtido de http://er.educause.edu: http://er.educause.edu/articles/2015/10/digital-game-based-learning-still-restless-after-all-these-years

Felicia, P. (2015). Unity 5 From Zero to Proficiency (Foundations). Em P. Felicia, *Unity 5 From Zero to Proficiency (Foundations)* (pp. 346-409). Patrick Felicia.

G2 Crowd. (2017). *Best Game Engine Software*. Obtido em 10 de 05 de 2017, de www.g2crowd.com: https://www.g2crowd.com/categories/game-engine?segment=all

Game Designing. (12 de 04 de 2017). *The 3 Best Video Game Engines*. Obtido em 10 de 05 de 2017, de www.gamedesigning.org: https://www.gamedesigning.org/career/video-game-engines/ga

GameObject. (02 de 06 de 2017). Obtido de https://docs.unity3d.com: https://docs.unity3d.com/Manual/GameObjects.html

IOL. (07 de 01 de 2017). *videojogos-valem-entre-6-a-12-milhoes-de-euros-em-portugal*. Obtido de http://www.tvi24.iol.p:

http://www.tvi24.iol.pt/economia/portugueses/videojogos-valem-entre-6-a-12-milhoes-de-euros-em-portugal

itcentralstation. (25 de 05 de 2017). *Jira*. Obtido em 25 de 05 de 2017, de https://www.itcentralstation.com: https://www.itcentralstation.com/products/jira

MacKay, R. F. (01 de 03 de 2013). *Playing to learn: Panelists at Stanford discussion say using games as an educational tool provides opportunities for deeper learning*. Obtido de http://news.stanford.edu: http://news.stanford.edu/2013/03/01/games-education-tool-030113/

Microsoft. (25 de 05 de 2017). *Visual Studio*. Obtido em 25 de 05 de 2017, de https://www.visualstudio.com/: https://www.visualstudio.com/

Schrier, K. (04 de 10 de 2016). *Learning, Education and Games. Volume Two: Bringing Games into Educational Contexts.* Obtido de http://press.etc.cmu.edu: http://press.etc.cmu.edu/content/learning-and-education-games-volume-two-bringing-games-educational-contexts

SUPERDATA. (2017). *market-brief-year-in-review/*. Obtido em 25 de 05 de 2017, de https://www.superdataresearch.com: https://www.superdataresearch.com/market-data/market-brief-year-in-review/

Thomas E. Murphy, M. W. (27 de 04 de 2017). *Magic Quadrant for Enterprise Agile Planning Tools*. Obtido de https://www.gartner.com: https://www.gartner.com/doc/reprints?id=1-3YWBN1Q&ct=170427&st=sb%255Bgartner.com

Trybus, J. (2014). *Game-Based Learning: What it is, Why it Works, and Where it's Going.* Obtido de http://www.newmedia.org: http://www.newmedia.org/game-based-learning--what-it-is-why-it-works-and-where-its-going.html

UNITY. (20 de 01 de 2012). *Prefabs*. Obtido de https://docs.unity3d.com: https://docs.unity3d.com/355/Documentation/Manual/Prefabs.html

Unity. (2017). *Multiplatform*. Obtido em 10 de 05 de 2017, de unity3d.com: https://unity3d.com/unity/multiplatform

Unity. (02 de 06 de 2017). *Scenes*. Obtido de https://docs.unity3d.com: https://docs.unity3d.com/Manual/CreatingScenes.html

Unity. (2017). *Services*. Obtido em 15 de 05 de 2017, de https://unity3d.com: https://unity3d.com/services

Wikipedia. (14 de 06 de 2017). *Audacity_(audio_editor)*. (2017, Editor, 06, Produtor, & 14) Obtido de https://en.wikipedia.org: https://en.wikipedia.org/wiki/Audacity_(audio_editor)

Wikipedia. (17 de Abril de 2017). *GIMP*. Obtido de https://en.wikipedia.org: https://en.wikipedia.org/wiki/GIMP

Wikipedia. (22 de 04 de 2017). *Git*. Obtido de https://en.wikipedia.org: https://en.wikipedia.org/wiki/Git

Wikipedia. (24 de 04 de 2017). *HUD (Video gaming)*. Obtido de https://en.wikipedia.org: https://en.wikipedia.org/wiki/HUD_(video_gaming)

Wikipedia. (10 de Abril de 2017). *Microsoft Visual Studio*. Obtido de https://en.wikipedia.org: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

Wikipedia. (11 de 06 de 2017). *Pac-Man*. Obtido de https://en.wikipedia.org: https://en.wikipedia.org/wiki/Pac-Man

Wikipedia. (25 de 04 de 2017). *Unity Technologies*. Obtido de en.wikipedia.org: https://en.wikipedia.org/wiki/Unity_Technologies

Williams, M. (05 de 04 de 2017). *The best free audio editor 2017*. Obtido de www.techradar.com: http://www.techradar.com/news/the-best-free-audio-editor

Apêndices

CodigoFonte.pdf