Instituto Superior de Engenharia do Porto

Desenvolvimento de uma *framework* para a criação de videojogos

Ricardo Miranda Almeida Moreira

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Área de Especialização em Sistemas Gráficos e Multimédia

Orientador: Doutor João Paulo Jorge Pereira

Júri:

Presidente:

Doutora Maria de Fátima Coutinho Rodrigues

Vogais:

Doutor Paulo Alexandre Gandra de Sousa

Doutor João Paulo Jorge Pereira

Porto, Outubro de 2014

Família, Amigos e Namorada

Agradecimentos

Em primeiro lugar, quero agradecer ao meu orientador, Professor Doutor João Paulo Pereira, pela disponibilidade e por todo o apoio incondicional durante toda a realização desta dissertação, pois fez com que esta tenha sido uma excelente oportunidade para enriquecer as minhas competências pessoais e profissionais.

A todos os meus amigos que sempre estiveram por perto e que nunca deixaram de me apoiar, dando sempre incentivos para prosseguir este caminho.

Aos meus pais, por estarem sempre presentes e me apoiarem em todas as minhas decisões, pois, sem eles, nunca teria chegado onde cheguei. Ao meu irmão, pelo apoio e transmissão de conhecimentos que foram essenciais para a realização desta dissertação.

Por fim, à minha namorada, Ana Tojal, por me ter dado sempre força e apoio para nunca desistir. Graças a ti, melhorei em imensos aspetos e provaste ser uma das pessoas mais importantes da minha vida.

A todos vós e a todos que possa não ter referido, um enorme obrigado!

Resumo

Os videojogos são cada vez mais parte integrante da sociedade, sendo que a massificação dos vários dispositivos que se encontram atualmente veio ajudar os videojogos a estarem mais presentes no dia-a-dia das pessoas.

Mas a criação de jogos só é possível através de ferramentas bem específicas, as *frameworks* ou motores de videojogos. Com estas é possível criar os mais diferentes géneros de videojogos para os mais diferentes dispositivos. Contudo, nem todas essas ferramentas são gratuitas, e as que são encontram-se pouco documentadas ou limitadas em determinadas funcionalidades, o que poderá levar mais tempo no desenvolvimento de um videojogo.

O trabalho desenvolvido nesta dissertação visa a criação de uma *framework* capaz de suportar diferentes géneros de videojogos, mas também que facilmente possibilite a alteração ou substituição de diferentes partes internas da *framework* sem que esta deixe de funcionar.

Para isso, foi realizada uma análise ao estado atual do mercado dos videojogos, bem como das ferramentas que possibilitam a criação dos mesmos, passando também pelas interfaces gráficas existentes nos videojogos.

Como forma de demonstrar as funcionalidades implementadas na *framework*, foi desenvolvido um protótipo de um videojogo de luta, tirando-se, assim, partido de algumas das características dessa ferramenta.

Palavras-chave: Videojogos, Framework, Motor de videojogo, Interface

Abstract

Nowadays, videogames are a big part of society and the increasing number of devices that exist currently is helping the videogames to be more present in the day-to-day lives.

But creating games is only possible through very specific tools, like frameworks or game engines. With these ones it is possible to create the most different genres of videogames for the different devices. However, not all of these tools are free, and those that are, are poorly documented or limited in certain features, and so it may take longer to develop a videogame.

The work developed in this dissertation is about the creation of a framework capable of supporting different genres of videogames, but that also allows any change or replacement of the various internal parts of the framework without this one crashing.

For this to happen, current state of the videogames market was analyzed, as well as the tools that enable their creation, also passing through the existing graphical interfaces in videogames.

In order to demonstrate the functionalities implemented in the framework, a prototype of a fight videogame was developed, and therefore it was possible to take advantage of some of the features of this tool.

Keywords: Video Games, Framework, Game Engine, Interface

Índice

Índice de Figuras

Índice de Tabelas

Índice de Gráficos

Acrónimos

Lista de Acrónimos

2D	Duas dimensões
3D	Três dimensões
AI	*Artificial Intelligence*
API	*Application Programming Interface*
GDD	*Game Design Document*
GPU	*Graphics Processing Unit*
HUD	*Heads-up Display*
IDE	*Integrated Development Environment*
ISEP	Instituto Superior de Engenharia do Porto
MMO	*Massively Multiplayer Online*
SDK	Software Development Kit

1 Introdução

Neste primeiro capítulo irá ser apresentada uma breve abordagem ao tema desta dissertação, passando pelo enquadramento, a apresentação da dissertação e por fim, a estrutura em que se encontra organizado este documento.

1.1 Enquadramento

Desde o aparecimento dos primeiros videojogos (doravante designados também por jogos), estes fazem cada vez mais parte da sociedade, estando cada vez mais disseminados através de campanhas publicitárias, eventos de competição, fazendo movimentar imensas pessoas e ao mesmo tempo, quantias avultadas de dinheiro. Hoje em dia o ser humano sente a necessidade de se distrair do seu dia-a-dia e uma boa forma de o fazer será através de um videojogo, seja ele muito simples ou mais complexo.

Com esta rápida evolução dos videojogos, as ferramentas que possibilitam a sua criação também sofreram um grande progresso, fazendo com que atualmente existam inúmeras ferramentas, desde as mais simples às mais complexas. Este tipo de ferramentas nem sempre apresenta as melhores funcionalidades, sendo que muitas são pagas ou exigem que num determinado videojogo seja explícito que o mesmo foi desenvolvido por tal ferramenta.

O trabalho que aqui foi desenvolvido teve como objetivo o desenvolvimento de uma ferramenta própria capaz de suportar qualquer género de videojogo. O desenvolvimento dessa ferramenta foi efetuada pelos alunos Ricardo Moreira e Gil Canizes, sendo que cada aluno ficou responsável por diferentes partes bem definidas da mesma. Como forma de demonstrar o seu funcionamento, cada aluno criou um género diferente de videojogo para

exemplificar que a ferramenta é capaz de dar resposta a diferentes géneros, sem que exista qualquer tipo de modificação interna.

Para além disso, a criação desta ferramenta surgiu devido aos alunos em questão trabalharem numa empresa de desenvolvimento de videojogos, a *Insane Sheep* [1]. Esta empresa foi criada pelo aluno Gil Canizes, à qual o aluno Ricardo Moreira também pertence como programador e já conta com um videojogo no mercado dos dispositivos móveis, mais concretamente nos sistemas operativos *Android*.

Com esta oportunidade, o desenvolvimento desta ferramenta tem como objetivo a sua aplicação a um ambiente real, ou seja, na empresa em que ambos os alunos se encontram a trabalhar. Assim, a empresa poderá focar não só os dispositivos móveis, mas também os computadores.

Por fim, este documento visa apresentar todo o processo de desenvolvimento desta dissertação no âmbito da disciplina de Tese/Dissertação, do ano letivo 2013/2014, do curso de Mestrado em Engenharia Informática, na área de especialização em Sistemas Gráficos e Multimédia, do Instituto Superior de Engenharia do Porto (ISEP).

1.2 Apresentação da Tese

Esta dissertação pretende analisar e estudar as várias ferramentas existentes no mercado para que assim se possa perceber as vantagens e desvantagens de cada uma. De entre as várias ferramentas existentes no mercado, estas dividem-se entre *frameworks* e motores de videojogo, sendo que as *frameworks* são basicamente um conjunto de classes criadas por um programador sem que exista qualquer tipo de interface visual. Por outro lado, os motores de videojogo oferecem uma interface visual de forma a ser mais simples e rápida a criação de um videojogo. Para além da análise às ferramentas, será feito um estudo ao universo dos videojogos, para que se possa perceber o estado atual e as tendências desse mercado.

Com base nessa investigação e análise feita às várias ferramentas de desenvolvimento de videojogos, pretende-se desenvolver uma *framework* capaz de responder às necessidades básicas de qualquer género de videojogo. O desenvolvimento irá ser dividido em duas partes principais: o desenvolvimento de uma *framework* e a criação de um videojogo contendo todos os aspetos importantes que um jogo deve conter.

Resumidamente, os principais objetivos desta dissertação serão:

- Efetuar o levantamento do estado da arte dos videojogos, bem como das atuais tecnologias;
- Analisar as diferentes tecnologias existentes no mercado;
- Desenvolver uma ferramenta capaz de dar resposta às funcionalidades básicas de qualquer jogo;
- Desenvolver um videojogo com base na ferramenta criada.

1.3 Organização do relatório

Nesta secção é apresentada a estrutura da dissertação de modo a que se possa obter uma visão geral dos conteúdos que serão abordados no decorrer do seu desenvolvimento. Com isto, serão descritos sucintamente os vários capítulos que compõem esta dissertação.

No primeiro capítulo, Introdução, será introduzido o tema a ser abordado, bem como o problema que se propõe resolver, sendo no fim, apresentada a ideia para a sua resolução.

O Contexto desta dissertação encontra-se no segundo capítulo e é aqui que será feita uma abordagem ao tema a desenvolver, apresentando diferentes tecnologias existentes no mercado. Serão também descritas as ferramentas tecnológicas utilizadas no desenvolvimento da solução ao problema apresentado.

O levantamento de requisitos necessários para o desenvolvimento da *framework* será apresentado no terceiro capítulo - Análise e *Design*. Dentro deste capítulo também será apresentada a estrutura lógica da *framework* e a interface a ser implementada no videojogo a ser desenvolvido.

No quarto capítulo, Implementação, é descrito todo o modo de funcionamento da *framework* e do videojogo, onde serão apresentadas e detalhadas as suas principais funcionalidades.

Por fim, no último capítulo, irá ser apresentada a Conclusão, onde será feita uma visão geral de toda a dissertação e trabalho realizado, bem como as suas limitações e dificuldades surgidas no decorrer do seu desenvolvimento. São também apresentadas novas funcionalidades e melhoramentos a serem implementados no futuro.

2 Contexto

Neste capítulo será apresentada uma análise de todo o estado da arte do tema em estudo, passando por diversos componentes como, por exemplo, o que são videojogos e quais as empresas contribuidoras para a sua evolução, quais as atuais plataformas e as suas tendências futuras, o estado atual do mercado dos jogos e as suas propensões no futuro e uma análise a diferentes tecnologias que se encontram disponíveis para o desenvolvimento dos mesmos.

2.1 Estado da arte do negócio

O entretenimento é constituído por diversas indústrias, desde a cinematográfica, televisiva, literária, musical, sendo que este mercado é um dos que apresenta um maior crescimento a nível mundial. Isto deve-se à evolução tecnológica e à grande adesão aos computadores e à internet, o que faz com que o conteúdo destas indústrias tenha vindo a surgir de forma digital, fazendo com que as pessoas se interessem mais, visto ser um conteúdo de rápido e fácil acesso.

De entre as diversas indústrias que constituem o mercado de entretenimento, existe uma que tem vindo a evoluir largamente nos últimos anos e, atualmente, encontra-se praticamente em qualquer dispositivo – a indústria dos videojogos.

Graças ao aumento da adesão ao mundo dos videojogos, devido, em grande parte, à rápida evolução tecnológica e, tendo como consequência, o surgimento de novas plataformas, algumas das quais portáteis, faz com que haja uma maior aproximação entre as pessoas e este universo. Desta forma, a indústria dos videojogos tem vindo a crescer muito,

o que faz com que esta seja cada vez mais promissora e comecem a surgir novas empresas que se dedicam ao desenvolvimento de novos produtos para a mesma.

2.1.1 Videojogos

A indústria dos videojogos surgiu na época dos anos 70 com o lançamento do jogo *Pong* (Figura 1), pela empresa *Atari,* que fez um enorme sucesso, sendo considerado o primeiro caso comercialmente viável em videojogos [2].

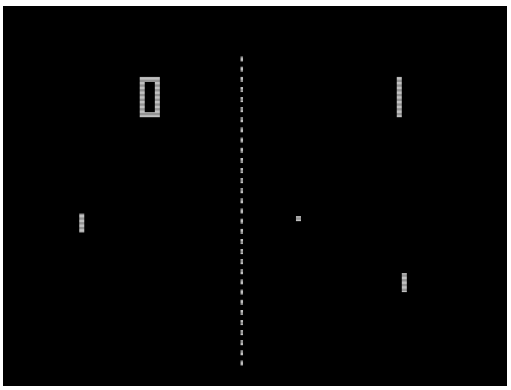


Figura 1 - Imagem do videojogo Pong

Após o lançamento deste primeiro jogo rapidamente começaram a surgir outros, despertando o interesse de numerosas pessoas nesta nova indústria.

Desde então foram disponibilizados ao público diversos videojogos dos mais diferentes géneros, dos quais se destacam os seguintes:

Ação: Exigem ao jogador uma maior coordenação entre o olhar e o tempo de reação perante os diferentes desafios encontrados ao longo do videojogo. Dentro deste género existem em largo número, como por exemplo, *Call Of Duty* (Figura 2), *Half-Life*, *Grand Theft Auto*, *Halo* ou *Metal Gear Solid*.

Figura 2 - Imagem do videojogo Call Of Duty

Aventura: Apresentam uma história interativa, em que a personagem principal é o jogador. Dentro do género destacam-se videojogos como o *Sam & Max* (Figura 3) e *The Walking Dead*.

Figura 3 - Imagem do videojogo Sam & Max

Corrida: Possibilitam ao jogador dirigir automóveis ou ciclomotores. São vários os videojogos que se apresentam neste registo, desde os menos realistas, como *Mario Kart,* a outros com maior realismo, como o *Gran Turismo* (Figura 4).

Figura 4 - Imagem do videojogo Gran Turismo

Desporto: Dedicam-se a vários tipos de desportos. No entanto, os de futebol são os que têm maior destaque, tais como o *Pro Evolution Soccer* e FIFA (Figura 5).

Figura 5 - Imagem do videojogo FIFA

Luta: Colocam o jogador num combate contra outro jogador ou então contra a inteligência artificial do computador. Os de maior destaque são o *Street Fighter* (Figura 6), *Mortal Kombat* e *Final Fight*.

Figura 6 - Imagem do videojogo Street Fighter

Estratégia: Normalmente exigem ao jogador que estabeleça uma estratégia de forma a conquistar os seus objetivos. Destaque para *StarCraft* (Figura 7) e *Civilization*.

Figura 7 - Imagem do videojogo StarCraft

Plataforma: O jogador terá de orientar a sua personagem de forma a poder saltar pelas diversas plataformas dos diferentes níveis do videojogo. Dentro do género, são vários os videojogos que fizeram sucesso, como por exemplo, o *Rayman* (Figura 8), *Donkey Kong*, *Sonic*, *Super Mario*.

Figura 8 - Imagem do videojogo Rayman

Alguns desses videojogos ficaram marcados na história pela sua inovação, pelos gráficos atrativos, pela jogabilidade, história e interatividade. Atualmente os videojogos apresentam uma qualidade gráfica cada vez mais realista e com uma infinidade de opções, dando, assim, uma maior liberdade ao jogador.

2.1.2 Empresas

Com a crescente evolução dos videojogos, foram muitas as empresas interessadas em entrar nesta nova indústria, a qual veio a evoluir, ano após ano, e a tornar-se numa das mais lucrativas do mundo.

São inúmeras as empresas que se dedicam ao desenvolvimento de videojogos, das quais se destacam a:

Konami: Surgiu em 1969 e encontra-se sediada no Japão. É responsável pelo desenvolvimento de vários videojogos, destacando-se, por exemplo, o *Pro Evolution Soccer* e o *Metal Gear Solid*.

Square Enix: Fundada em 1975 no Japão, é conhecida por desenvolver vários videojogos de grande sucesso, como a série *Final Fantasy*, *Hitman* e *Tomb Raider*.

Ubisoft Montreal: Estúdio criado em 1997, no Canadá, e é responsável pela série *Prince of Persia*, *Assassin's Creed*, *Tom Clancy's.*

Rockstar Games: Fundada em 1998, na Escócia, é mundialmente conhecida pelo desenvolvimento dos famosos videojogos da série *Grand Theft Auto*, *Max Payne*, *Red Dead Redemption*.

Infinity Ward: Empresa californiana criada em 2002, responsável pelo desenvolvimento da série *Call of Duty*.

DICE (Digital Illusions Creative Entertainment): Empresa sueca fundada em 1992, é conhecida por desenvolver os videojogos *Battlefield* e *Mirror's Edge*.

Rovio: Criada em 2003, na Finlândia, e é conhecida pelo desenvolvimento do videojogo *Angry Birds*.

Para além destas grandes empresas de desenvolvimento de videojogos, muito recentemente tem-se verificado o aparecimento de novos estúdios de pequena dimensão, que se dão pelo nome de "videojogos independentes" ou "*indie games".*

Os estúdios de videojogos independentes normalmente têm um capital financeiro reduzido e também uma equipa constituída por poucos elementos, o que poderá ser um benefício, visto que aumenta a participação individual de cada elemento. Desta forma, existe uma maior inovação e criatividade, o que se torna bastante vantajoso.

Estes estúdios aumentaram significativamente nos últimos anos devido ao aparecimento de novos métodos de distribuição *online*, como o *Steam,* para os videojogos para computadores, *Google Play,* para os dispositivos móveis com o sistema operativo *Android,* e *Apple Store,* para os dispositivos móveis com o sistema operativo *iOS*. Com estes novos métodos de distribuição tornou-se muito mais fácil para estes estúdios divulgarem os seus jogos e, assim, alcançar o maior número de pessoas, visto que estes serviços são disponibilizados em todo o mundo.

Para além da distribuição *online*, o aparecimento de ferramentas de desenvolvimento gratuitas também contribuíram para um processo de criação mais simples e rápido de videojogos, o que fez com que viesse reforçar ainda mais a posição que os videojogos independentes têm vindo a alcançar nesta indústria.

Alguns dos exemplos de videojogos criados por estes estúdios que obtiveram enorme sucesso a nível mundial são o *Minecraft*, *Braid* e *World of Goo*.

Em Portugal, a indústria de jogos começa a surgir com a criação de vários estúdios de videojogos independentes [3], mas ainda não apresentam um grande volume de produção e qualidade, o que faz com que não existam empresas a investir neste meio.

2.1.3 Plataformas

Desde o lançamento do primeiro videojogo comercialmente viável na década de 70, a evolução dos videojogos só foi possível graças ao desenvolvimento que as plataformas

tiveram, conseguindo estas ser cada vez mais pequenas, leves e mais poderosas a nível computacional.

Nintendo, *Atari* e *Sega* foram as principais empresas impulsionadoras no mundo das plataformas, desenvolvendo consolas domésticas como a *Super Nintendo*, *Atari 2600* ou *Mega Drive,* fazendo com que estas se tornassem um enorme sucesso de vendas [4].

Nos últimos anos, as empresas *Atari* e *Sega* deixaram de produzir consolas devido ao fracasso de vendas nos seus últimos lançamentos, estando apenas dedicadas ao desenvolvimento de videojogos. Em relação à empresa *Nintendo*, esta tem conseguido acompanhar as tendências e exigências dos jogadores, inovando na forma como as pessoas interagem com as suas novas consolas.

Para além da última empresa referida, existem as empresas *Microsoft* e *Sony* que também se têm dedicado ao desenvolvimento de novas consolas. Estas têm conseguido proporcionar novas experiências a cada nova geração de produtos que vão sendo lançados para o grande público, tais como as consolas da 7ª geração, a *Playstation* 3, *Xbox* 360 e *Nintendo Wii* (Figura 9).

Figura 9 - Consolas da 7ª geração: Playstation 3, Xbox 360 e Nintendo Wii

Relativamente à consola da empresa *Nintendo*, a *Nintendo Wii*, esta não apresenta uma qualidade gráfica como as suas rivais, mas destaca-se pela sua principal caraterística: o comando *Wii Remote,* capaz de captar todos os movimentos do jogador, inovando assim no campo da jogabilidade.

No caso da empresa *Microsoft*, a sua consola *Xbox 360* apresenta como principais caraterísticas o serviço *Xbox Live,* que permite aos jogadores competir através da internet, visualizar filmes, ouvir músicas e transferir versões experimentais de videojogos. Para além deste serviço, é disponibilizado o acessório *Kinect* que permite detetar os movimentos dos jogadores, possibilitando, assim, que os mesmos possam interagir com os videojogos sem necessitar de qualquer tipo de comando, usando apenas as suas próprias mãos.

A consola *PlayStation 3* foi desenvolvida pela empresa *Sony* e apresenta como principal caraterística o serviço *PlayStation Network,* que permite aos jogadores imensas funcionalidades, desde a competição *online* com diversos jogadores, a transferências de jogos em formato digital e visualização de vídeos. Para além deste serviço, esta consola disponibiliza o acessório *PlayStation Move* que permite ao jogador, utilizando este acessório na mão, captar todos os seus movimentos corporais de forma a interagir com os videojogos desenvolvidos para esse efeito.

Atualmente, as consolas encontram-se numa fase de transição das consolas da 7ª para a 8ª geração (Figura 10).

Figura 10 - Consolas da 8ª geração: Playstation 4, Xbox One e Nintendo Wii U

Na Tabela 1 é possível visualizar algumas das características técnicas das consolas da 7ª geração.

Tabela 1 – Resumo das especificidades técnicas das consolas da 7ª geração

	Playstation 3	**Xbox 360**	**Nintendo Wii**
Média	Blu-ray, DVD, CD	CD, DVD	Discos óticos Wii
CPU	Processador Cell PowerPC com 8 núcleos a 3.2GHz	Processador IBM PowerPC com 3 núcleos a 3.2 GHz	Processador IBM com 1 núcleo a 729 MHz
GPU	RSX a 550MHz	Xenos a 500 MHz	ATI a 243 MHz
Memória	256 MB	512 MB	88 MB

Já na Tabela 2, são apresentadas algumas das características técnicas das consolas da nova geração.

Tabela 2 – Resumo das especificidades técnicas das consolas da 8ª geração

	Playstation 4	**Xbox One**	**Nintendo Wii U**
Média	Blu-ray, DVD	Blu-ray, DVD	Discos óticos Wii e Wii U
CPU	Processador AMD "Jaguar" com 8 núcleos	Processador AMD com 8 núcleos a 1.75 GHz	Processador IBM Power PC 750 com 3 núcleos a 1.24 GHz
GPU	1.84 TFLOPS, motor gráfico AMD Radeon	1,31 TFLOPS, motor gráfico AMD Radeon	AMD Radeon HD GP a 550 MHz
Memória	8 GB GDDR5	8 GB DDR3	2 GB DDR3

Como se pode visualizar através das Tabelas 1 e 2, a evolução do *hardware* de uma geração para a outra é notória. Este tipo de evolução tecnológica irá proporcionar o desenvolvimento de videojogos cada vez mais realistas, inteligentes, dinâmicos e de interação cada vez mais inovadora.

Para além das consolas, a plataforma que também apresenta grande destaque são os computadores pessoais. Os computadores são equipamentos multitarefas usados para várias situações, como por exemplo, no trabalho, no desenvolvimento de videojogos e no entretenimento. Esta plataforma destaca-se por já se encontrar massificada e integrada na vida das pessoas e por apresentar um poder computacional bastante superior relativamente às outras plataformas.

Além destas, uma nova plataforma tem vindo a surgir no mercado, que são os *smartphones* e *tablets*. Estes *gadgets* são cada vez mais poderosos e capazes de proporcionar uma experiencia bastante próxima ao que acontece nas consolas domésticas, apresentando como principal vantagem em relação às consolas o fator portabilidade, podendo a pessoa estar em qualquer lugar e em qualquer momento jogar um videojogo. Para além da portabilidade, estes novos tipos de equipamentos apresentam um poder computacional bastante próximo das consolas domésticas da 7ª geração, sendo estes novos dispositivos um novo potencial mercado a ser explorado pela indústria dos videojogos.

Para além destas plataformas, existem outras, de entre as quais se destacam:

Consolas:

- *Playstation 2, 3 e 4*
- *Playstation Vita*
- *PSP*
- *Xbox 360 e One*
- *Nintendo Wii e Wii U*
- *Nintendo DS e 3DS*

Sistemas Operativos:

- *Windows e Windows Phone*
- *Linux*
- *Android*
- *Mac OS e iOS*

2.1.4 Análise do Mercado

Com a evolução dos videojogos e a sua presença nas mais diversas plataformas, esta nova forma de entretenimento ganha cada vez mais adesão por parte dos compradores, o que se irá refletir nas vendas que este mercado tem vindo a apresentar.

Com cerca de 70,4 mil milhões de dólares de faturação a nível mundial em 2013, este mercado não apresenta nenhum abrandamento nas vendas mas, muito pelo contrário, um forte crescimento, ano após ano, como se pode verificar no Gráfico 1 [5].

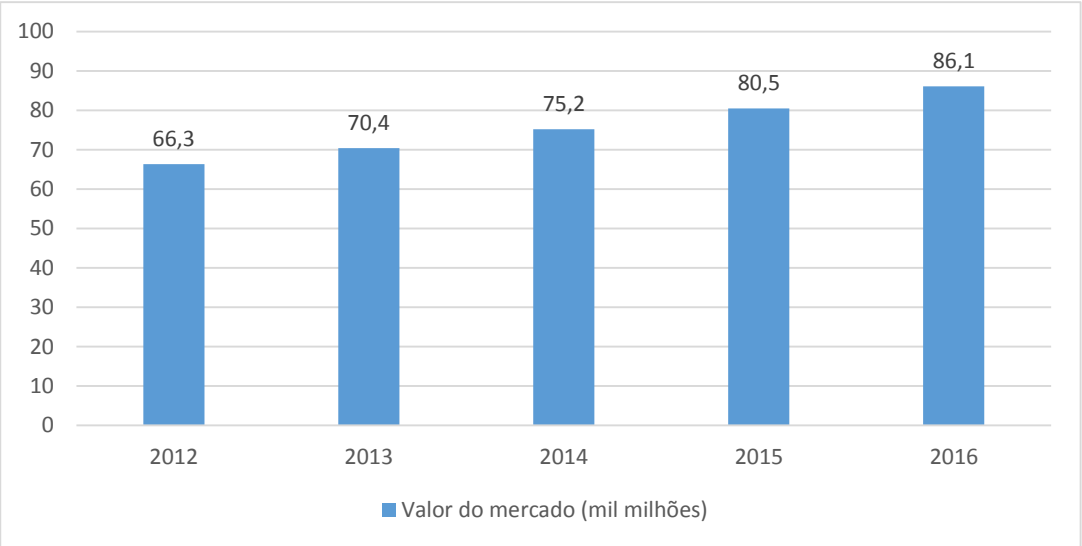


Gráfico 1 - Evolução do mercado dos videojogos a nível mundial

De acordo com os dados obtidos, prevê-se que este mercado, em 2016, tenha um valor na ordem dos 86,1 mil milhões de dólares, um crescimento de mais de 22% em relação ao valor obtido no ano de 2013.

Relativamente aos continentes que mais contribuem para o grande crescimento deste mercado, pode-se verificar no Gráfico 2 [5] que o continente asiático é o que mais contribui, com cerca de 25.1 mil milhões de dólares, logo a seguir a América do Norte, com cerca de 22.8 mil milhões de dólares, e os continentes europeu e africano, com cerca de 19.5 mil milhões de dólares. No final da lista aparece América do Sul, com uma faturação anual de 3 mil milhões de dólares.

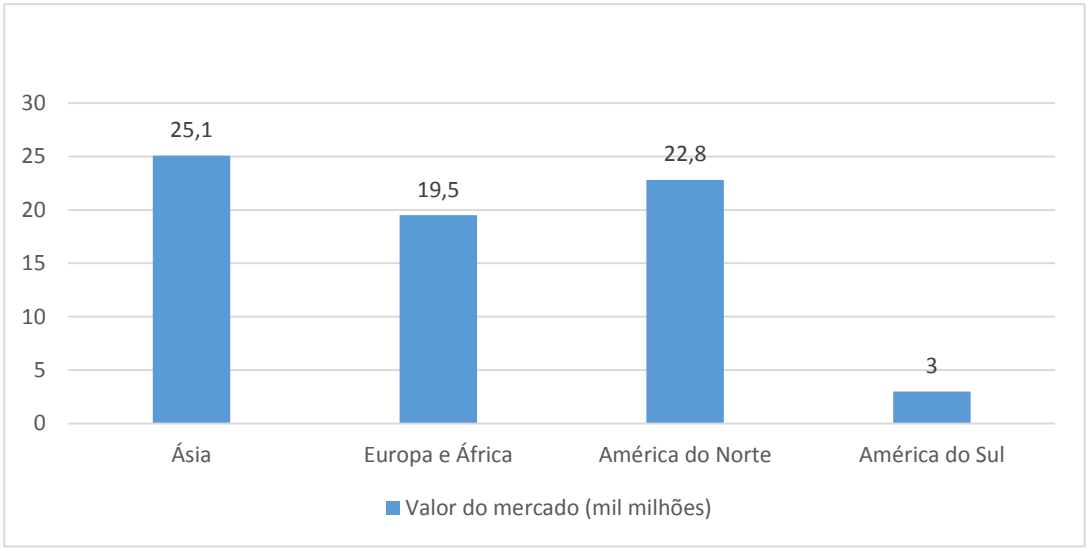


Gráfico 2 - Distribuição mundial do mercado dos videojogos no ano de 2013

Em relação às vendas das várias plataformas pode-se verificar que algumas estão a começar a ganhar terreno neste mundo de entretenimento e, por consequência, outras plataformas estão a ficar para trás, como se poderá ver no Gráfico 3 [5]. Exemplos disso são as consolas portáteis e o computador, para os quais se prevê que, em 2016 haja, em relação a 2013, uma descida de 3,4% e 2,8%, respetivamente. Já as plataformas que têm vindo a crescer neste mercado são os dispositivos móveis como *smartphones* e *tablets*, para as quais se prevê, no mesmo período de tempo, um crescimento na ordem dos 4,1% e 6,3%, respetivamente.

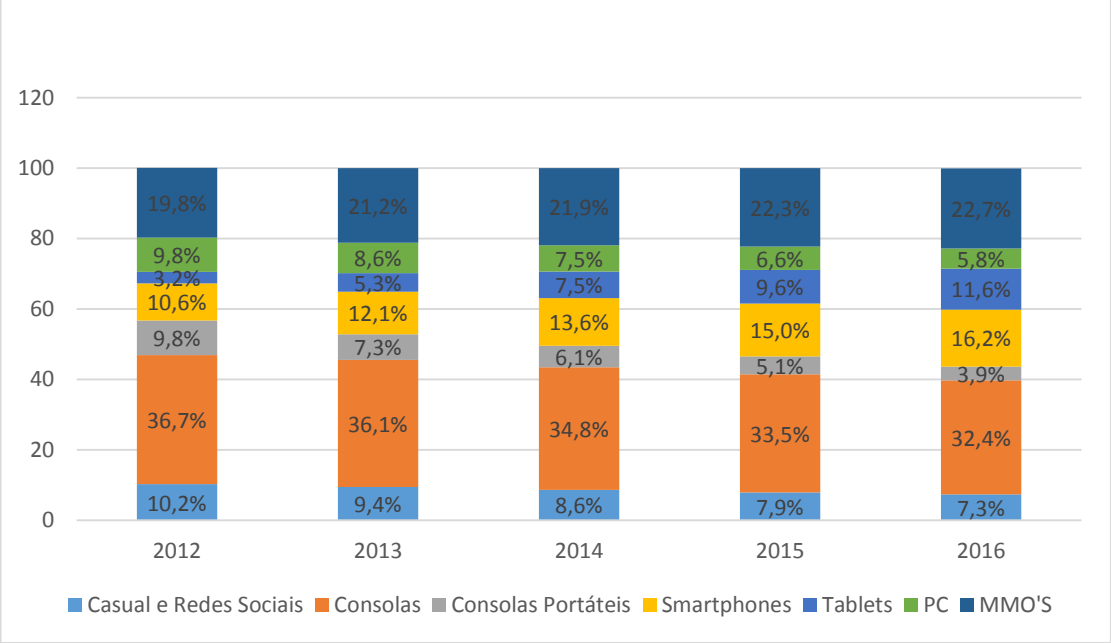


Gráfico 3 - Distribuição dos segmentos dos videojogos a nível mundial

Apesar do crescimento a nível dos dispositivos móveis, estes não são os que apresentam maior faturação no mercado dos videojogos. Através do gráfico anterior, é possível prever que as consolas apresentem um decréscimo de 3,4% em 2016, tendo por base os valores de 2013. Contudo, é a plataforma que mais lucro gera, com um valor de 30,6 mil milhões de dólares (Gráfico 4) [5]. Os Videojogos para PC apresentam um valor muito mais baixo em relação às consolas: apenas 6 mil milhões de dólares. No entanto, consegue ser compensado graças aos *MMO'S* (*Massively Multiplayer Online*), que conseguem gerar um valor na ordem dos 14,9 mil milhões de dólares. Relativamente aos dispositivos móveis (*smartphones* e *tablets*), estes conseguem arrecadar cerca de 12,3 mil milhões de dólares, que é aproximadamente o dobro do valor que as redes sociais e os jogos causais conseguem gerar.

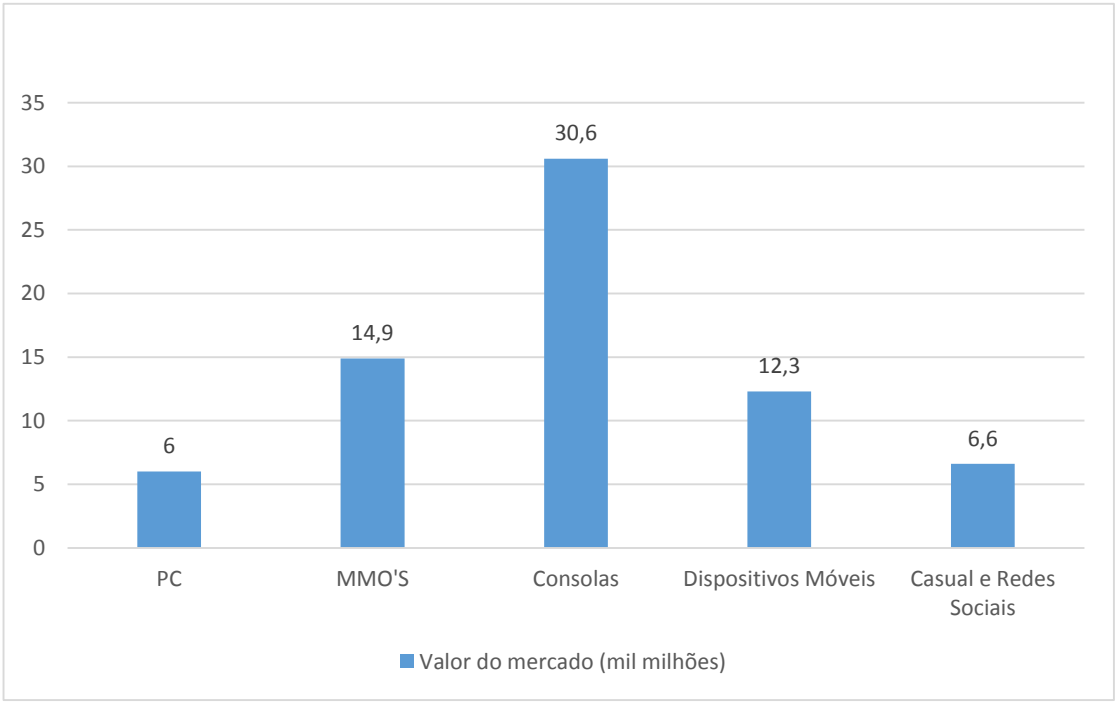


Gráfico 4 - Os segmentos que geram mais lucro no ano de 2013

Um outro aspeto bastante importante é das vendas digitais e físicas. Este novo método de venda digital tem vindo a crescer cada vez mais em relação ao formato físico, como se pode verificar no Gráfico 5 [6].

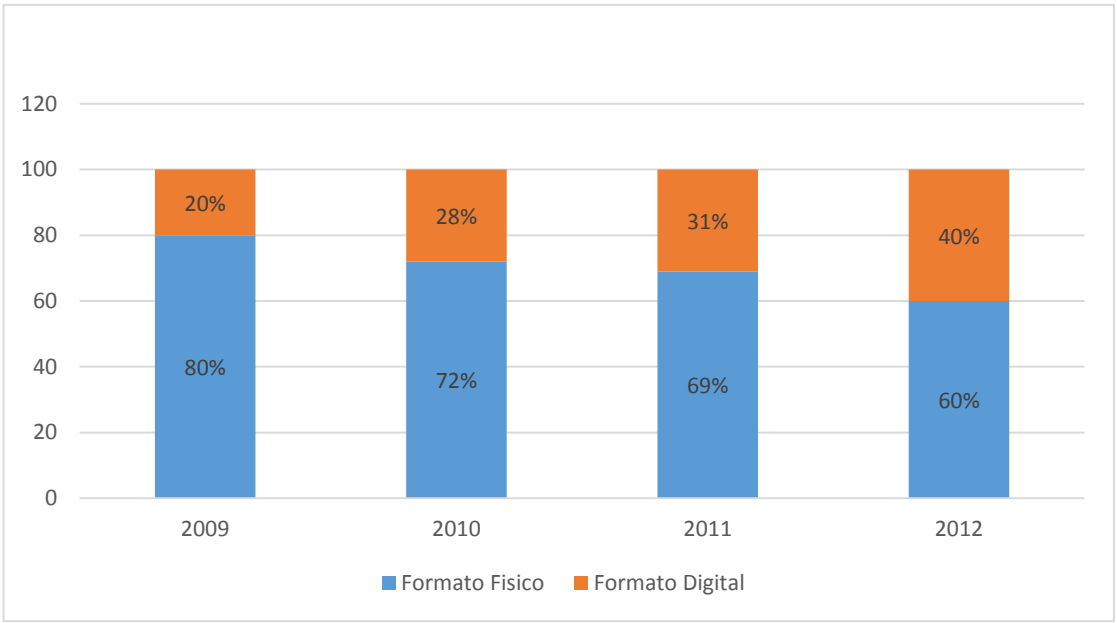


Gráfico 5 - Evolução das vendas digitais e física

Entre 2009 e 2012, podemos verificar que as vendas digitais tiveram um crescimento de 20% e que as vendas em formato físico tiveram um decréscimo de, também, 20%.

Os dispositivos móveis têm vindo a beneficiar do crescimento das vendas digitais. A prova é que, em 2013, esta plataforma conseguiu faturar 12,3 mil milhões de dólares.

Assim, podemos concluir que o mercado dos videojogos tem vindo a faturar cada vez mais, sendo uma das indústrias mais fortes da atualidade. Novas plataformas têm vindo a crescer no mercado e prevê-se que no futuro os dispositivos móveis irão ser aqueles que mais contribuirão para o crescimento dos videojogos a nível mundial.

2.2 Estado da arte tecnológica

A evolução dos videojogos só foi possível graças ao contínuo melhoramento das tecnologias e à criação de novas ferramentas para o apoio ao desenvolvimento de novos videojogos.

Na época em que começaram a surgir os primeiros videojogos, as ferramentas de desenvolvimento eram escassas e muitas delas bastante complicadas de usar e bastante limitadas. No entanto, com a evolução da tecnologia, foram-se criando novas ferramentas cada vez mais simples e cada vez mais capazes de proporcionar experiencias únicas nos videojogos.

Atualmente existem diversas ferramentas no apoio ao desenvolvimento de videojogos, desde as mais simples às mais completas e complexas, oferecendo ao programador diversas funcionalidades que podem ser aplicadas nos jogos.

As ferramentas que hoje em dia existem podem ser divididas em *frameworks* e motores de videojogo.

***Frameworks*:** Conjunto de classes fortemente interligadas que auxiliam o programador no desenvolvimento do videojogo. As *frameworks* permitem ao programador aceder a toda a sua estrutura, dando uma maior liberdade e precisão ao mesmo. Este tipo de ferramenta não apresenta qualquer tipo de interface, tendo o programador que escrever todo o código.

Motores de videojogo: Os motores de videojogo ou *game engines* são ferramentas mais completas e complexas do que as *frameworks*. Não permitem que o programador possa aceder à estrutura principal do motor, estando limitados neste aspeto. Como vantagem, possuem uma interface com o programador, podendo o mesmo construir o videojogo apenas com o sistema de arrasto ou, em inglês, *drag-and-drop* e, assim, evitar ter

que escrever qualquer linha de código para o videojogo funcionar, o que permite um desenvolvimento mais rápido do jogo.

Devido ao aparecimento deste tipo de ferramentas, o desenvolvimento de videojogos tornou-se mais rápido e acessível.

2.2.1 *Frameworks*

Como foi descrito na página anterior, as *frameworks* são ferramentas que auxiliam o programador no desenvolvimento de um videojogo. São várias as que existem no mercado, sendo algumas mais completas do que outras e com diferentes caraterísticas.

De entre as várias *frameworks* existentes, foram selecionadas as mais conhecidas no desenvolvimento de videojogos em 2D, das quais se destacam:

- *Cocos2D* [7]*;*
- *LÖVE* [8]*;*
- *XNA* [9]*;*
- *Angel 2D* [10]*;*
- *Oxygine* [11].

Nesta seção irão ser analisadas as cinco *frameworks* selecionadas, sendo que a primeira *framework* será analisada do ponto de vista da sua arquitetura e funcionamento, enquanto que as restantes quatro serão brevemente analisadas relativamente às suas principais caraterísticas, dando assim uma melhor noção de como é constituída uma *framework* e de entender as principais diferenças entre as *frameworks* selecionadas.

2.2.1.1 Cocos2D

Cocos2D é uma *framework* de criação de videojogos 2D e que tem como principal foco as plataformas móveis.

Surgiu em 2010 e inicialmente foi desenvolvida para criação de jogos 2D para a plataforma *iOS*. Com o elevado sucesso que teve com a mesma, e com o grande aumento da comunidade, os seus criadores começaram a trabalhar noutras plataformas existentes no mercado e assim surgiu a *Cocos2D-X*.

A *Cocos2D-X* é uma *framework* bastante rápida, fácil de usar, leve, robusta, eficiente e, como principal caraterística, é capaz de exportar para as plataformas mais recentes, como *Android*, *Windows Phone* e *iOS*.

Atualmente muitos programadores, e até mesmo empresas, aderiram a esta *framework* para o desenvolvimento dos seus próprios videojogos, pois graças à sua arquitetura (Figura 11), esta permite a publicação do mesmo jogo em várias plataformas, sem assim ter a necessidade de refazer o jogo para cada plataforma em específico.

Alguns exemplos de videojogos que utilizam esta *framework* e que têm obtido sucesso são *Banana Kong*, *Hill Climb Racing*, *BADLAND*, *Contra: Evolution*.

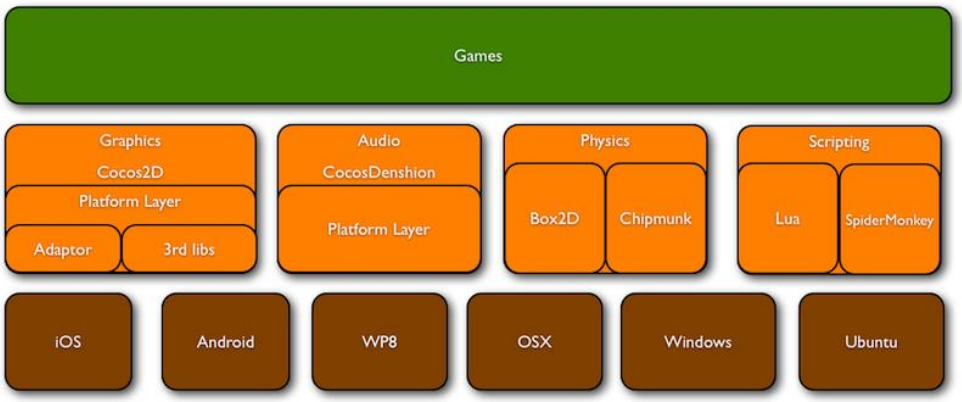


Figura 11 - Arquitetura da framework Cocos2d-X

Para além do suporte a várias plataformas, esta *framework* apresenta outros pontos bastante positivos, como a enorme comunidade que ajuda qualquer programador com dificuldades, uma documentação muito completa, inúmeros tutoriais para ajudar a quem está a começar a utilizar a *framework* e ferramentas de apoio ao desenvolvimento.

São também disponibilizadas outras versões:

- *Cocos2d-X*
  - Utiliza a linguagem de programação *C++* e destina-se às plataformas *Android*, *iOS*, *Windows Phone*, *Windows*, *Linux* e *Mac OS*.
- *Cocos2d-Iphone*
  - Utiliza a linguagem de programação *Objective-C* e destina-se à plataforma *iOS*.
- *Cocos2d-Html5*
  - Utiliza a linguagem de programação *JavaScript* e destina-se a qualquer plataforma, desde que possua um *browser*.
- *Cocos2d-XNA*
  - Utiliza a linguagem de programação *C#* e destina-se à plataforma *Windows Phone*.

Para além de tudo isto, a *framework* é de código-fonte aberto, ou seja, qualquer utilizador poderá modificar qualquer funcionalidade conforme as suas necessidades, sendo mais um ponto positivo para esta *framework*.

Cocos2D tem a licença *MIT License*, ou seja, é totalmente gratuita e o programador poderá utilizar a *framework* como desejar, para uso pessoal ou comercial (Tabela 3).

Tabela 3 – Principais características da framework Cocos-2D

Principais caraterísticas
Possibilidade de criar videojogos 2D
Linguagem de programação: *C++, Lua, Objective-C, Javascript* e *C#*
Uso da *API OpenGL*
Editor Visual *CocoStudio*
Código-fonte aberto
Comunidade composta por vários milhares de utilizadores
Licença gratuita para uso pessoal e comercial
Suporta as plataformas: *Windows 8, Linux, Mac OS, iOS, Android, BlackBerry, Tizen, Windows Phone 7, Windows Phone 8, HTML 5* e *Xbox 360*

2.2.1.2 LÖVE

LÖVE é uma *framework* para a criação de videojogos 2D para o computador. Foi criada em 2010 e tem como principal objetivo ajudar os programadores a desenvolver mais rapidamente os seus jogos.

A linguagem de programação usada para o seu desenvolvimento é o *Lua* [12]. Esta tem vindo a ser usada cada vez mais no mundo dos videojogos, visto ser uma linguagem rápida, leve e flexível.

LÖVE disponibiliza uma pequena comunidade com cerca de 10 mil utilizadores prontos a ajudar na resolução de problemas que o programador possa encontrar ao desenvolver o seu videojogo. Para além desta comunidade, também disponibiliza várias bibliotecas para tornar a *framework* mais robusta e completa.

A licença de utilização é totalmente gratuita e o código-fonte da *framework* encontra-se disponível para qualquer pessoa, dando assim a possibilidade ao programador de modificar ou acrescentar novas funcionalidades à mesma.

Esta *framework* suporta as plataformas *Windows*, *Linux* e *Mac OS* (Tabela 4).

Tabela 4 – Principais características da framework LÖVE

Principais caraterísticas
Possibilidade de criar jogos 2D
Linguagem de programação: *Lua*
Uso da *API OpenGL*
Código-fonte aberto
Comunidade com cerca de 10 mil utilizadores
Licença gratuita para uso pessoal e comercial
Suporta as plataformas: *Windows, Linux, Mac OS*

2.2.1.3 XNA

XNA é uma *framework* gratuita para o desenvolvimento de videojogos 2D ou 3D em *C#*.

Esta *framework* nasceu em 2004 e foi desenvolvida pela *Microsoft* para tornar a criação de jogos muito mais rápida e acessível. Nas primeiras versões só permitia desenvolver videojogos para *Windows* e *Xbox 360*, mas, com a sua evolução, mais funcionalidades eram disponibilizadas e mais plataformas eram abrangidas, como por exemplo *Windows Phone* e *Zune*.

As plataformas suportadas por esta *framework* pertencem exclusivamente à *Microsoft* (*Xbox 360*, *Zune*, *Windows* e *Windows Phone 7*), o que poderá ser um ponto negativo em relação às outras *frameworks* analisadas.

Em 2013 a *Microsoft* deu por descontinuado o desenvolvimento desta *framework*. Mesmo estando descontinuada, muitos utilizadores ainda continuam a usar para o estudo e desenvolvimento de videojogos (Tabela 5).

Tabela 5 – Principais características da framework XNA

Principais caraterísticas
Possibilidade de criar jogos 2D e 3D
Linguagem de programação: *C#*
Uso da *API DirectX*
Comunidade composta por vários milhares de utilizadores
Licença gratuita para uso pessoal
Suporta as plataformas: *Windows*, *Windows Phone 7*, *Xbox 360* e *Zune*

2.2.1.4 Angel 2D

Angel 2D é uma *framework* criada em 2009 por uma equipa de programadores que trabalhava para a *Electronic Arts* e que tem como principal foco a criação de jogos em 2D.

Apresenta-se como sendo rápida, leve e flexível, pois é baseada em motores de videojogo mais evoluídos, como *Unreal Egine* e *CryENGINE* e, assim, obtém as melhores técnicas que cada uma oferece.

A linguagem de programação utilizada é o *C++,* sendo mais um fator que faz com que esta *framework* tenha bons resultados a nível de desempenho.

Angel 2D suporta as plataformas *Windows*, *Mac OS*, *iOS* e *Linux* e o seu código-fonte encontra-se disponível para qualquer pessoa, sendo uma mais-valia para o programador, já que poderá modificar as classes constituintes da *framework* conforme as suas necessidades (Tabela 6).

Tabela 6 – Principais características da framework Angel 2D

Principais caraterísticas
Possibilidade de criar jogos 2D
Linguagem de programação: *C++* ou *Lua*
Uso da *API OpenGL*
Código-fonte aberto
Licença gratuita para uso pessoal e comercial
Suporta as plataformas: *Windows*, *Linux*, *Mac OS* e *iOS*

2.2.1.5 Oxygine

Oxygine é uma *framework* totalmente gratuita e tem como objetivo a criação de jogos 2D, com especial destaque para plataformas móveis.

Surgiu em 2013 e inicialmente foi desenvolvida para os dispositivos móveis *iOS* e *Android,* mas com o avanço da mesma, passou a ser possível abranger outras plataformas, como por exemplo *Windows*, *Linux* e *Mac OS*.

A linguagem de programação utilizada é o *C++,* tornando-a mais robusta no que diz respeito ao desempenho.

Esta *framework* tem o seu código-fonte totalmente disponível, para que, assim, qualquer utilizador possa modificar conforme as suas necessidades ou até mesmo contribuir para o avanço da mesma (Tabela 7).

Tabela 7 – Principais características da framework Oxygine

Principais caraterísticas
Possibilidade de criar jogos 2D
Linguagem de programação: *C++*
Uso da *API OpenGL*
Código-fonte aberto
Licença gratuita para uso pessoal e comercial
Suporta as plataformas: *Windows*, *Linux*, *Mac OS*, *iOS* e *Android*

2.2.2 Motores de videojogo

Para além das *frameworks*, os motores de videojogo também auxiliam o programador no desenvolvimento de um videojogo, sendo estas ferramentas mais completas e complexas em relação às *frameworks*.

De entre os vários motores de videojogo existentes no mercado, foram selecionados os mais conhecidos [13] [14] no desenvolvimento de videojogos:

- *Unity* [15]*;*
- *Unreal Engine* [16]*;*
- *CryEngine* [17]*;*
- *Source* [18]*;*
- *Id Tech* [19]*.*

Nesta seção irão ser analisados os cinco motores de videojogo selecionados, sendo estes avaliados à semelhança das *frameworks* previamente descritas. No primeiro motor de videojogo selecionado, o *Unity*, irá ser feita uma análise de forma aprofundada às suas funcionalidades, enquanto que os outros quatro motores irão ser brevemente analisados sobre as suas principais caraterísticas. Desta forma, é possível uma melhor compreensão acerca da constituição de um motor de videojogo, bem como das principais diferenças entre os motores selecionados.

2.2.2.1 Unity

Unity é um motor de videojogo que surgiu em 2005, criado pela *Unity Technologies* e é muito usado no desenvolvimento de videojogos 2D e 3D.

Este motor possui tecnologias que se conseguem equiparar a outros grandes motores que já existem há algum tempo no mercado, tais como a *Unreal Engine* e *CryENGINE*.

O *Unity* tem um imenso sucesso devido ao facto de ser gratuito e possuir um editor visual bastante acessível. Facilmente um programador consegue construir um videojogo básico com poucos cliques e sem ter que programar uma única linha de código.

É possível desenvolver videojogos para as mais variadas plataformas como por exemplo, *Xbox One, BlackBerry, Windows, Windows Phone 8, Mac OS, Linux, Android, iOS, Unity Web Player, Playstation 3, Xbox 360, Wii U* e *Wii*.

Como já foi referido, o *Unity* disponibiliza uma versão gratuita em que o programador poderá explorar livremente todas as suas funcionalidades. Infelizmente, nem todos podem desenvolver para *iOS, Android, Wii* ou outra plataforma, já que é necessário ter uma licença autorizada para se poder desenvolver nessas plataformas. Com a versão gratuita, apenas será possível desenvolver para as plataformas *Windows*, *Linux*, *Mac* e para os navegadores web que tenham instalado o *plugin UnityPlayer*.

Para além da parte técnica, a parte social também tem um grande ponto positivo, contando com uma excelente comunidade de apoio aos programadores para o auxílio de dúvidas ou a sugestão de novas funcionalidades a serem implementadas.

Editor

O editor do *Unity* é muito completo, bem construído, bastante fácil de utilizar e muito intuitivo. Tudo o que o programador necessita para desenvolver está presente neste editor, como por exemplo, posicionar objetos, editar *scripts* e importar ficheiros, como se pode verificar na Figura 12.

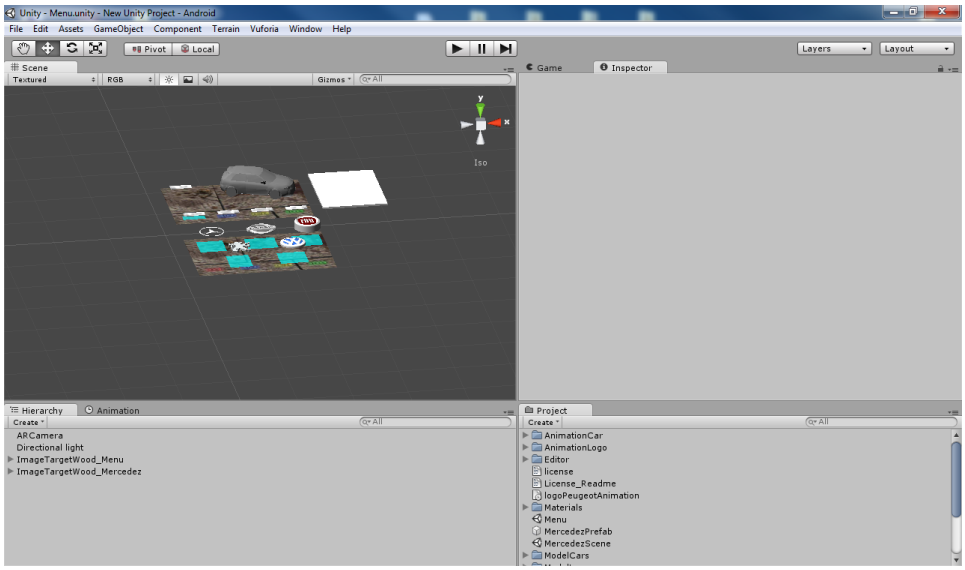


Figura 12 – Aspeto gráfico do Editor

O editor do *Unity* encontra-se bem estruturado, apresentando do lado esquerdo superior a cena atual onde se poderá deslocar, aumentar, diminuir e apagar os objetos da cena. Nesse mesmo lado, mas na parte inferior, é apresentado o nome de todos os objetos presentes na cena. Já no lado direito superior, o programador terá a possibilidade de visualizar o resultado final da cena, tendo assim o *feedback* de imediato. Nesse lado, mas na parte inferior, são apresentadas as pastas e ficheiros do projeto que se encontra a trabalhar.

Assets

Assets é um componente do *Unity* que faz com que seja possível carregar ficheiros nos mais diversos formatos. Alguns dos formatos que o *Unity* suporta são:

- Modelos 3D nos formatos *Autodesk*, *Maya*, *3D Max, Blender, XSI, Lightwave, Cinema4D*;
- Fontes *TTF* (*TrueType*);
- Formatos *Photoshop* e *Illustrator*, permitindo assim editar o ficheiro diretamente e através de camadas. Para além destes formatos, também suporta *jpg*, *png* e *bmp*.
- Formatos de áudio como *mp3*, *wav* e *ogg*.

Terrenos

Com o *Unity* é possível criar terrenos totalmente personalizáveis, desde montes altíssimos a buracos de grande profundidade, graças à ferramenta de criação de terreno disponível.

Como podemos visualizar na Figura 13, a ferramenta de criação de terrenos no *Unity* é muito poderosa, flexível e de fácil customização.

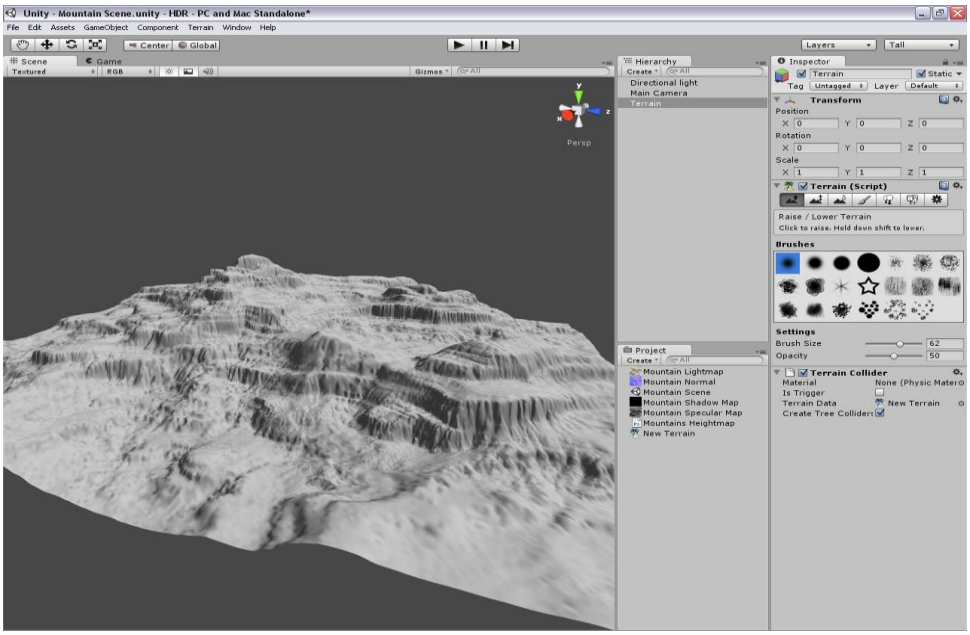


Figura 13 – Ilustração da criação de um terreno

Física

Para além destas funcionalidades disponíveis, o *Unity* utiliza um motor físico criado pela *nVidia*, o *PhysX*, que permite criar simulações físicas com extrema qualidade e realismo.

Alguns dos recursos disponíveis são:

- *Ragdolls* – permite o controlo de um "corpo". Por exemplo, seria possível a personagem principal ser um humano que sofreria com a ação das leis da física;
- Carros – simples de implementar e permite o controlo total de um veículo;
- *Joints* – com este recurso é possível ligar objetos entre si com o objetivo de criar, por exemplo, pontes e correntes.
- *Rigidbodies* – Objetos que sofrem com a ação das leis da física, como por exemplo, um cubo ou uma esfera.

Áudio & Vídeo

O *Unity* possui um excelente suporte em relação à reprodução de áudio e vídeo, oferecendo diversos recursos para a reprodução dos mesmos, tais como:

- Suporta reprodução de vídeos, tanto em textura 2D ou 3D;
- Suporta diversos formatos de áudio, como por exemplo, *mp3*, *ogg*, *wav* e *aac*;
- Capacidade de converter todos os ficheiros de áudio e vídeo para o formato *ogg* automaticamente. Este formato possui qualidade semelhante ao *mp3,* mas é totalmente gratuito.

Scripting

Para além da excelente ajuda que o editor visual disponibiliza ao programador, também é possível criar os próprios códigos onde irá conter toda a lógica e funcionamento de uma determinada ação, isto com a ajuda da ferramenta *Mono*.

As linguagens de programação disponíveis são o *C#, Javascript* e o *Boo* (dialeto da linguagem *Python*). Seja qual for a linguagem que deseja utilizar, a performance não irá ser afetada, ou seja, irá obter sempre o mesmo desempenho, seja qual for a linguagem de programação que esteja a utilizar (Tabela 8).

Tabela 8 – Principais características da game engine Unity

Principais caraterísticas
Possibilidade de criar jogos 2D e 3D
Linguagem de programação: *C#, Javascript* e *Boo*
Suporta as *API's OpenGL, WebGL* e *DirectX*
Comunidade composta por vários milhares de utilizadores
Suporta as plataformas: *Windows, Windows Phone 8, Linux, Mac OS, Playstation 3, Playstation 4, Xbox 360, Xbox One, Wii, Wii U, iOS, Android* e *BlackBerry*

2.2.2.2 Unreal Engine

Unreal Engine é um motor de videojogo desenvolvido pela empresa *Epic Games* e é considerado um dos mais poderosos e avançados do mundo.

Este motor de videojogo surgiu em 1998 com o lançamento do primeiro jogo da empresa *Epic Games* de seu nome *Unreal*. Desde então, foram vários os jogos lançados com este motor, destacando-se nomes como *Unreal Tournament, Deus Ex, Tom Clancy's Rainbow Six, BioShock e Batman*.

Como o seu motor de videojogo é desenvolvido em *C++*, faz com que permite a possibilidade de suportar as mais variadas plataformas como por exemplo *Windows, Linux, Mac, Android, iOS, Playstation 2, Playstation 3, Playstation 4, Xbox, Xbox 360, Xbox One* e *Wii*.

Atualmente este motor de videojogo encontra-se na sua quarta versão, sendo gratuito para uso pessoal, mas caso seja para entrar numa vertente comercial, terá de ser adquirida uma licença. Caso seja necessário ter acesso ao código-fonte do motor de videojogo, isso terá de ser comunicado à empresa *Epic Games* para solicitar uma licença (Tabela 9).

Tabela 9 – Principais características da game engine Unreal Engine

Principais caraterísticas
Grande potencial na criação de videojogos 3D
Linguagem de programação: *C++* ou *Unreal Script*
Suporta as *API's DirectX, OpenGL, Stage 3D* e *WebGL*
Uso de técnicas e tecnologias mais avançadas da atualidade
Comunidade composta por vários milhares de utilizadores
Suporta as plataformas: *Windows, Linux, Mac OS, iOS, Android, Playstation 2, Playstation 3, Playstation 4, Xbox, Xbox 360, Xbox One* e *Wii*

2.2.2.3 CryENGINE

CryENGINE é um motor de videojogo desenvolvido pela empresa *Crytek*, que surgiu em 2004 e inicialmente foi projetado como uma demo tecnológica para a empresa *Nvidia*. No entanto, quando a empresa viu o seu grande potencial, decidiu transformar e criar, assim, o motor de videojogo *CryENGINE*.

Com isto, ainda no mesmo ano, surgiu *Far Cry*, o primeiro jogo da empresa a utilizar este motor. Desde então, a sua evolução tem aumentado ano após ano, criando videojogos com um poder gráfico extraordinário. Alguns exemplos de jogos que marcaram o sucesso deste motor são *Far Cry*, a série *Crysis* e *WarFace*.

Este motor permite a utilização de funcionalidades bastante interessantes, como o suporte a *API DirectX 11*, sistema avançado de partículas de vidro, efeitos avançados na utilização de água, inteligência artificial avançada, editor de mapa em tempo real e desenvolvimento para várias plataformas. Como linguagem de programação, o programador poderá escolher entre *C++* ou *C#*.

A ferramenta é gratuita para uso pessoal, mas caso entre numa vertente comercial, terá de adquirir uma licença (Tabela 10).

Tabela 10 – Principais características da game engine CryEngine

Principais caraterísticas
Grande potencial na criação de jogos 3D
Linguagem de programação: *C++* ou *C#*
Suporta a *API DirectX*
Uso de técnicas e tecnologias mais avançadas da atualidade
Comunidade composta por vários milhares de utilizadores
Suporta as plataformas: *Windows, iOS, Android, Playstation 2, Playstation 3, Playstation 4, Xbox, Xbox 360, Xbox One, Wii* e *Wii U*

2.2.2.4 Source Engine

Source Engine é um motor de videojogo criado pela *Valve Corporation* e surgiu em 2004 com o lançamento do jogo *Counter-Strike Source*.

É conhecido por ter sido utilizado em jogos como *Half-Life 2, Counter-Strike, Left 4 Dead 2, Portal 2 e Team Fortress 2*.

A linguagem de programação utilizada é o *C++*, a mesma que foi utilizada para o desenvolvimento deste motor, o que faz com que o seu desempenho seja elevado em qualquer plataforma.

Este motor disponibiliza um *SDK* para os programadores o poderem explorar e desenvolverem novos videojogos com o mesmo. Infelizmente, esta ferramenta tem sido alvo de muitas críticas devido ao facto de ser muito difícil de usar e por se encontrar bastante desatualizada.

Para desenvolvimento pessoal é possível transferir uma versão totalmente gratuita. No entanto, caso seja para uso comercial, deverá ser adquirida uma licença para o efeito.

As plataformas que este motor suporta são *Windows, Linux, Mac OS, Playstation 3, Xbox, Xbox 360* e *Xbox One* (Tabela 11).

Tabela 11 – Principais características da game engine Source

Principais caraterísticas
Grande potencial na criação de jogos 3D
Linguagem de programação: *C++*
Suporta as *API's DirectX* e *OpenGL*
Uso de técnicas e tecnologias mais avançadas da atualidade
Comunidade composta por vários milhares de utilizadores
Suporta as plataformas: *Windows, Linux, Mac OS, Playstation 3, Xbox, Xbox 360* e *Xbox One*

2.2.2.5 Id Tech

Desenvolvido pela empresa *id Software,* surgiu em 1993 e foi utilizado no jogo *Doom,* lançado nesse mesmo ano.

Este motor tem evoluído imenso desde a sua primeira versão e atualmente encontra-se na quinta versão. Quando a empresa lança uma nova versão deste motor, a anterior normalmente passar a ser gratuita e de código-fonte aberto, o que é um aspeto bastante positivo, pois ajuda os programadores a explorarem as capacidades deste motor, estudar, melhorar e até mesmo criarem os seus videojogos com este motor.

Algumas das principais caraterísticas que a nova versão apresenta são a iluminação avançada, profundidade de campos, *motion blur* e, em particular, o *Mega Texture*. Esta última tecnologia consiste em suportar texturas de altas resoluções (até 128000x128000 pixéis), sendo este um dos pontos fortes desta nova versão do motor de videojogo.

Este motor suporta as plataformas *Windows, Mac, Plastation 3 e 4, Xbox 360* e *One* (Tabela 12).

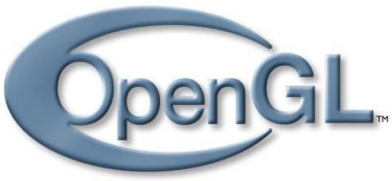
Tabela 12 – Principais características da game engine idTech

Principais caraterísticas
Grande potencial na criação de jogos 3D
Linguagem de programação: *C++*
Suporta a *API OpenGL*
Uso de técnicas e tecnologias mais avançadas da atualidade
Código-fonte do motor de jogo fica disponível ao fim de algum tempo
Comunidade composta por vários milhares de utilizadores
Suporta as plataformas: *Windows, Linux, Mac OS, Playstation 3, Xbox, Xbox 360* e *Xbox One*

2.3 Tecnologias e ferramentas utilizadas

Para dar início ao desenvolvimento da *framework* foi necessário efetuar uma pesquisa das várias tecnologias e ferramentas existentes no mercado, verificando aquelas que se enquadravam mais neste tema. Um fator importante a ter em conta na escolha das mesmas foi serem totalmente gratuitas e de código-fonte aberto, para a sua utilização ser livre e de fácil modificação, caso necessário.

2.3.1 *OpenGL*



OpenGL [20] é uma *API* desenvolvida pela *Silicon Graphics* com o objetivo de criar um padrão na criação de gráficos 3D e 2D. Isto porque nos anos 80 cada fabricante tinha o seu próprio conjunto de instruções para a criação de gráficos e, assim, surgiu a necessidade de criar um padrão que pudesse ser utilizado por qualquer fabricante.

Esta *API* apresenta como principais caraterísticas a possibilidade de gerar imagens de alta qualidade, ser totalmente independente de plataforma e ter o código-fonte aberto e gratuito.

Graças à criação desta *API* e do apoio de imensas empresas tecnológicas, como a *HP, Nvidia, AMD, IBM e Intel*, esta tem evoluído imenso, estando atualmente na versão 4.4.

2.3.2 *FMOD*

O *FMOD* [21] é uma *API* desenvolvida pela empresa *Firelight Technologies* e permite a reprodução e gestão de todo o áudio de uma determinada aplicação ou videojogo.

Pode ser usada de forma gratuita, desde que o lucro de um videojogo não ultrapasse os 100 mil euros. Caso contrário, será necessário adquirir uma licença para a comercialização do videojogo. Como características importantes destaca-se o facto de permitir a reprodução de som tridimensional de alta qualidade, a leitura de diversos formatos e a possibilidade de funcionar nas mais variadas plataformas.

2.3.3 *GLFW*

O *GLFW* [22] é uma biblioteca desenvolvia na linguagem *C* por várias pessoas e que tem como principal objetivo a criação e gestão de janelas em contexto com o *OpenGL*. Esta biblioteca é totalmente gratuita e de código-fonte aberto, podendo qualquer pessoa modificar ou contribuir para o melhoramento da mesma. Para além de tratar da gestão das janelas, esta biblioteca apresenta diversos recursos, como o suporte a vários monitores, capacidade de suportar diversas plataformas (*Windows*, *Linux* e *OS X)* e suporte a diferentes tipos de periféricos (rato, teclado e *joystick*).

2.3.4 *CMake*

CMake [23] é uma ferramenta gratuita e de código-fonte aberto desenvolvida pela *Kitware* e tem como principal objetivo a compilação de um determinado código e a possibilidade de exportar para diferentes plataformas.

Com a utilização desta ferramenta será possível desenvolver um videojogo apenas numa única linguagem e exportar para diferentes sistemas operativos, como por exemplo o *Windows*, *Linux* e *Mac OS*, poupando, assim, tempo e recursos.

2.3.5 *Vorbis*

Vorbis [24] é uma tecnologia gratuita de áudio digital desenvolvido pela fundação *Xiph*. Os ficheiros normalmente encontram-se no formato *ogg* ou *oga* e tem como principal característica apresentar um tamanho em disco muito reduzido e com uma qualidade de áudio igual ou superior a um formato muito conhecido, o *mp3*.

2.3.6 *Theora Playback Library*

Biblioteca desenvolvida na linguagem *C++* pelo programador Krešimir Špes e é utilizada para a reprodução de vídeos com áudio [25]. Esta biblioteca é totalmente gratuita e de código-fonte aberto, podendo qualquer pessoa modificar conforme as suas necessidades. Para além disso, tem a capacidade de suportar várias plataformas como *Windows*, *Linux*, *OS X*, *iOS*, *Android* e *Windows Phone 8*.

2.3.7 *FreeType*

É uma biblioteca desenvolvida na linguagem *C* por David Turner, Robert Wilhelm e Werner Lemberg, tendo como objetivo a possibilidade de carregar e apresentar diferentes estilos de letra numa aplicação ou videojogo [26]. Apresenta como principais caraterísticas o suporte a diferentes formatos como *TTF* e *OpenType* (*OTF*), capacidade de funcionamento nas mais diversas plataformas e de ser totalmente gratuita e de código-fonte aberto.

2.3.8 *Box* 2D

Box 2D é uma biblioteca desenvolvida por *Erin Catto* tendo como objetivo a possibilidade de adicionar física a corpos rígidos num plano 2D. Com a utilização desta biblioteca, será possível simular física nos mais diversos objetos como caixas, bolas, entre outros. Esta biblioteca é totalmente gratuita e de código-fonte aberto, dando assim uma maior liberdade.

2.3.9 *GLM*

OpenGL Mathematics (*GLM*) [27] é uma biblioteca desenvolvida pela *G-Truc Creation* tendo como principal objetivo o apoio ao cálculo de vetores e matrizes no âmbito de sistemas gráficos. Esta biblioteca apresenta diversos métodos como a translação, rotação, projeção ortogonal e perspetiva. *GLM* é uma biblioteca totalmente gratuita e de código-fonte aberto, podendo ser implementada nas mais diversas plataformas.

2.3.10 *GLEW*

OpenGL Extension Wrangler (*GLEW*) [28] é uma biblioteca desenvolvida em *C/C++* sendo totalmente gratuita e de código-fonte aberto, capaz de suportar várias plataformas e que simplifica o acesso aos métodos do *OpenGL*. Esta biblioteca consegue determinar em tempo de execução, quais os métodos que a placa gráfica disponibiliza, sabendo assim se um determinado computador suporta ou não a versão utilizada do *OpenGL*.

2.3.11 C++

C++ é uma linguagem de programação de alto nível, mas com facilidades para o uso em baixo nível. Esta linguagem foi desenvolvida por Bjarne Stroustrup como uma melhoria da linguagem de programação *C* e atualmente é uma das linguagens mais usadas em todo mundo devido à sua robustez e rapidez na interação com o *hardware* de uma determinada plataforma.

2.3.12 *CodeBlocks*

CodeBlocks [29] é um ambiente integrado de desenvolvimento (*IDE*) para as plataformas *Windows*, *Linux* e *Mac OS*, que permite o desenvolvimento de aplicações na linguagem de programação *C* ou *C++* e tem suporte a múltiplos compiladores, como por exemplo *GCC/MinGW*.

Para além deste *IDE* ser totalmente gratuito e de código-fonte aberto, tem a possibilidade de adicionar novas funcionalidades através do uso de *plugins*, sendo uma mais valia em relação aos outros *IDEs* existentes no mercado.

2.4 Sumário do capítulo

Através da análise efetuada neste capítulo, é notória a evolução que a indústria dos videojogos obteve nos últimos anos. Desde o aparecimento dos primeiros até à atualidade, obtiveram um claro progresso, permitindo uma jogabilidade mais diversificada, um maior realismo a nível gráfico, uma maior diversidade de plataformas e uma nova forma de distribuição digital. Com a evolução dos videojogos, o aparecimento de novas empresas e estúdios de desenvolvimento de videojogos é claro, visto ser uma indústria muito lucrativa e que prevê um crescimento ao longo dos próximos anos, como se pode verificar na análise do mercado efetuada neste capítulo. É, também, uma indústria muito competitiva.

A evolução dos videojogos só foi possível devido ao avanço tecnológico e ao aparecimento de cada vez mais ferramentas de desenvolvimento de videojogos. Algumas destas foram aqui analisadas, sendo que todas elas apresentam as suas vantagens e desvantagens, cabendo ao programador selecionar a que vá de encontro aos seus requisitos. Em alternativa, poderá desenvolver a sua própria ferramenta, tal como é o exemplo desenvolvido em conjunto com esta dissertação.

3 Análise e *Design*

Neste capítulo será apresentada a análise e *design* do tema em estudo, passando por diversos componentes, como por exemplo, o desdobramento e apresentação do protótipo de videojogo a ser desenvolvido, a arquitetura escolhida para o desenvolvimento da *framework*, a apresentação do levantamento de requisitos necessários para o seu funcionamento e a escolha da metodologia de desenvolvimento a seguir.

3.1 Desdobramento do projeto

Com o início da análise do projeto a ser desenvolvido, foi necessário dividir em duas partes principais, que serão apresentadas de seguida.

A primeira parte diz respeito ao desenvolvimento de uma *framework* capaz de responder às necessidades que qualquer videojogo necessita, tais como a apresentação de texturas, efeitos sonoros, interação através de diferentes dispositivos de entrada (teclado, rato e comando), suporte a diferentes sistemas operativos, sistema de colisão entre objetos da cena e o jogador.

A segunda parte será relativa ao desenvolvimento de um protótipo de um videojogo que irá tirar proveito das principais funcionalidades da *framework* criada, sendo que o género escolhido será de luta. Para se poder ter uma melhor perceção do videojogo a ser desenvolvido, recorreu-se ao uso de um documento muito utilizado por empresas e estúdios de desenvolvimento de videojogos, o *Game Design Document* (*GDD*).

Um *GDD* é um documento que contém todas as caraterísticas que constituem um videojogo, desde o nome, género, audiência a que se destina, mecânicas adotadas, história à jogabilidade. Com a criação deste documento, será possível ter uma melhor organização das ideias e conteúdos do videojogo.

Este documento poderá ser encontrado na secção de anexos desta dissertação.

3.2 Arquitetura

No decorrer do desenvolvimento desta dissertação foram analisadas várias *frameworks* e motores de videojogo existentes no mercado, tal como é apresentado na secção do estado da arte tecnológica deste documento.

Todas estas apresentam inúmeras funcionalidades e várias formas de interagir com as mesmas. No entanto, muitas pecam em alguns aspetos, como o pagamento de uma licença para a sua utilização comercial, inúmeras funcionalidades totalmente dispensáveis e que só irão fazer com que o videojogo necessite de mais poder de processamento e mais espaço em disco e manipulação totalmente restrita, no caso dos motores de videojogo. No caso das *frameworks*, apesar de algumas permitirem o acesso ao seu código-fonte, a falta de documentação ou a sua elevada complexidade em modificar partes importantes da mesma, fazem com que exista alguma dificuldade no desenvolvimento de um determinado jogo. Tendo em conta estas barreiras, surgiu a necessidade da criação de uma nova *framework* que possa ir de encontro aos requisitos básicos de que um videojogo necessita.

No entanto, existem várias vantagens e desvantagens na criação de uma *framework*:

Vantagens

1. Maior conhecimento de como é efetuado todo o processo de criação e de como se processa todo o seu funcionamento, o que será uma mais-valia no que toca ao enriquecimento do conhecimento;
2. Melhor conhecimento da estrutura da *framework* para eventuais deteções de falhas ou melhorias;
3. Facilidade de incorporação de novas funcionalidades;
4. Existência de inúmeras bibliotecas gratuitas para uma maior melhoria da *framework*;

5. Permite a criação de uma *framework* mais personalizada, introduzindo apenas as funcionalidades necessárias para o correto funcionamento de um videojogo, tornando-a mais limpa e leve, não contendo código desnecessário;
6. Possibilidade de desenvolver um videojogo para as plataformas desejadas;
7. Existência de fóruns de entreajuda para programadores.

Desvantagens:

1. Necessidade de estudo prévio para a criação e utilização das ferramentas necessárias para o desenvolvimento da *framework*;
2. Necessidade de mais tempo de desenvolvimento;
3. Criação de uma *framework* menos robusta em relação às existentes no mercado;
4. Não tirar partido das últimas novidades tecnológicas;
5. Maior risco de insucesso.

Após ponderação das vantagens e desvantagens do trabalho e a decisão de o concretizar, existe a necessidade de escolher a melhor arquitetura, sendo a arquitetura modular a melhor opção. Esta é a melhor opção, pois assim é possível desenvolver uma boa base e, conforme as necessidades de um determinado videojogo, facilmente se poderá adicionar novos módulos ou até mesmo atualizar os módulos já existentes, sem que a arquitetura deixe de funcionar, tornando a *framework* mais robusta e simples de usar.

Como se pode verificar na Figura 14, todo o processo de criação encontra-se dividido em três grandes módulos.

Figura 14 – Arquitetura modular

3.2.1 Módulo do Videojogo

Neste módulo encontram-se todas as classes que apenas dizem respeito ao videojogo, como por exemplo, a classe jogador, as classes para os diferentes menus e a classe de configurações. Para poder apresentar diferentes elementos no ecrã terá de comunicar com o módulo da *framework* para que este concretize o pedido feito pelo módulo videojogo. Com esta divisão por módulos, será possível desenvolver diferentes géneros de videojogos sem ter que reescrever todo o código, tendo apenas que trocar o módulo do videojogo.

3.2.2 Módulo da *Framework*

O módulo da *framework* é responsável por toda a gestão dos vários elementos que compõem a cena, como a apresentação de gráficos, som e textos. É constituído pelos módulos *Helpers, Libs, Math* e *Logic*, como se pode ver na Figura 15.

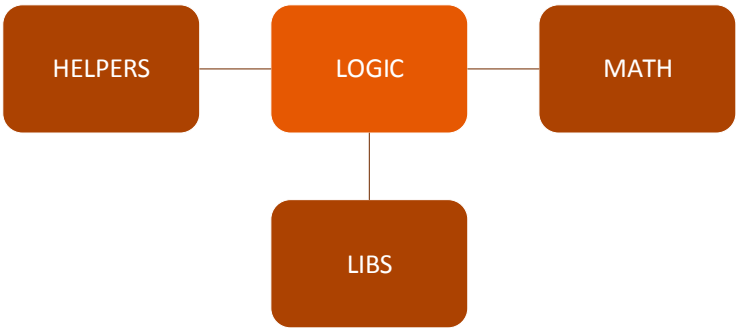


Figura 15 - Arquitetura do módulo da Framework

Cada um destes quatro módulos assume uma responsabilidade bastante importante na *framework*, tendo cada um a sua tarefa.

Módulo *Logic*

Este módulo trata de toda a lógica da *framework*, ou seja, tem como principal objetivo fazer a interligação dos vários módulos que a compõem, fazendo com que seja possível a comunicação entre todos os módulos. Assim, caso o módulo do videojogo necessite de alguma funcionalidade, como por exemplo, apresentar algum gráfico no ecrã, o módulo lógica irá tratar de comunicar com o módulo responsável por apresentar gráficos no ecrã.

Módulo *Helpers*

Tem como principal objetivo o apoio na criação de tarefas específicas e com utilização regular. Com a criação deste módulo, a reutilização e organização do código será muito mais eficaz e produtiva para o programador.

Módulo *Libs*

Agrega todas as bibliotecas importantes, como por exemplo, o *OpenGL,* que é responsável por apresentar gráficos no ecrã e o *FMOD*, que é responsável pela reprodução do áudio. Para além dessas bibliotecas, será possível atualizar ou adicionar outras a este módulo sem que a *framework* deixe de funcionar, pois a sua arquitetura é modular.

Módulo *Math*

O módulo *Math* é responsável por conter as principais classes que irão tratar de toda a parte matemática da *framework*. O cálculo de vetores, colisões entre objetos e outro tipo de operações, são tratadas por este módulo.

3.2.3 Módulo da Plataforma

Neste módulo encontram-se as plataformas a que se destina o videojogo, sendo, neste caso, o computador. Dentro do ramo dos computadores, são vários os sistemas operativos que existem no mercado, sendo que os mais utilizados são o *Windows*, *Linux* e *Mac OS*.

Com a utilização da ferramenta *CMake*, será possível compilar o mesmo código de forma nativa para os diferentes sistemas operativos. Com isto, o desenvolvimento de um videojogo será suportado por estes diferentes sistemas operativos para que, assim, mais utilizadores possam usufruir do videojogo.

3.3 Levantamento de requisitos

O levantamento de requisitos é um fator bastante importante a ter em conta no desenvolvimento do trabalho realizado no âmbito desta dissertação. Nesta secção será efetuado um levamento dos requisitos funcionais e não funcionais da *framework* e do videojogo a ser desenvolvido.

3.3.1 Requisitos funcionais

São vários os requisitos funcionais necessários para um correto funcionamento e para uma maior satisfação por parte do programador que irá tirar proveito das principais funcionalidades desta *framework*.

Framework

Suporte a vários sistemas operativos

Deverá ser capaz de suportar os atuais sistemas operativos *Windows*, *Linux* e *Mac OS*;

Diferentes dispositivos de entrada

Capacidade de interagir com vários dispositivos de entrada, tais como teclado, rato e comando, para que o jogador possa movimentar e interagir com o videojogo;

Áudio

Reprodução de músicas e efeitos sonoros para um maior envolvimento do jogador no videojogo;

Imagem

Apresentação dos vários objetos constituintes de uma cena de um videojogo, tais como visualização de vídeos, imagens, gráficos e animações;

Colisões

A *framework* deverá ser capaz de testar colisões entre o jogador e os objetos de uma cena de um videojogo;

Estilos de letra

Capacidade de carregar e apresentar diferentes estilos de letra através de ficheiros *TTF*;

Diferentes formatos

Capacidade de carregar os principais formatos de ficheiros tais como *PNG*, *TXT* e *OGG*;

Para além da *framework*, será necessário efetuar um levantamento dos requisitos funcionais do videojogo para, assim, ir de encontro às expectativas do jogador.

Vídeojogo

Menu interativo

Criação de um menu interativo onde o jogador poderá navegar entre as diferentes opções de menu, tais como jogar, selecionar uma personagem, configurar as teclas de videojogo e ativar ou descativar o som.

Diferentes personagens e cenários

O videojogo deverá apresentar pelo menos duas personagens diferentes para o jogador e escolher três cenários diferentes;

Pausa no videojogo

Deverá ser possível fazer pausa no videojogo e voltar ao mesmo estado quando voltar a jogar;

Sistema de combate

Criação de um sistema básico de combate entre duas personagens;

Interface

Capacidade de apresentar a energia do jogador e do seu adversário e o tempo de combate;

Multijogador

O videojogo deverá permitir que duas pessoas joguem entre si através de diferentes dispositivos de entrada (teclado e comando);

Vários Idiomas

Possibilidade do jogador alterar o idioma do videojogo, fazendo com que seja possível abranger um maior número de jogadores.

3.3.2 Requisitos não funcionais

O desenvolvimento de uma *framework* e um videojogo deverá satisfazer não só os requisitos funcionais, mas também os requisitos não funcionais, sendo necessário efetuar um levantamento dos mesmos em termos de desempenho, usabilidade, manutenção e compatibilidade.

Eficiência

A *framework* deverá ser capaz de apresentar um desempenho rápido e fluido, de modo a que a experiência no videojogo possa obter uma eficiência aceitável.

Fiabilidade

Dada a elevada existência de diferentes computadores e de sistemas operativos, poderá existir alguma probabilidade de ocorrência de erros. Contudo, a *framework* deverá ser capaz de evitar erros de maneira a garantir uma maior satisfação por parte do jogador.

Manutenção

Garantir que a *framework* e o videojogo permitem uma fácil manutenção e alteração das suas funcionalidades e componentes que as constituem, bem como a possibilidade de introdução de novas funcionalidades e mecanismos.

Usabilidade

O videojogo deverá apresentar uma interface simples, intuitiva e de fácil utilização por parte do jogador. Com isto será possível garantir uma boa comunicação entre o jogador e o videojogo, bem como a compreensão de tudo o que é apresentado ao jogador, para que o mesmo se sinta à vontade na utilização do videojogo.

3.4 Metodologia de desenvolvimento

Antes de ser iniciado o desenvolvimento da *framework* e respetivo videojogo, será necessário seguir uma metodologia de desenvolvimento, pois irá aumentar a eficiência e organização no decorrer do trabalho.

São várias as metodologias existentes no desenvolvimento de *software,* como a abordagem em cascata [30], em que esta geralmente é associada a um ciclo de desenvolvimento, que se inicia com a análise e irá percorrer as várias fases de forma sequencial até chegar à avaliação. Apesar de esta ser bastante utilizada no desenvolvimento de *software*, não será a melhor opção, visto não ser possível voltar à fase anterior, limitando bastante o processo de desenvolvimento.

Um outro exemplo de uma metodologia de desenvolvimento é em espiral [31], que colmata algumas das falhas existentes na metodologia em cascata. Contudo, este tipo de metodologia requer cada vez mais informação e detalhe a cada etapa, tornando o seu desenvolvimento mais exaustivo.

Uma outra opção que se adapta melhor a este tipo de desenvolvimento será a metodologia em estrela [32] (Figura 16). Esta metodologia foi desenvolvida por Hix e Hartson nos anos 90 e é adequada ao desenvolvimento de interação com o utilizador, porque se centra na avaliação contínua ao longo de todo o processo de desenvolvimento, o que faz com que no final de cada fase, os resultados sejam avaliados antes de se iniciar a fase seguinte. São várias as fases que esta metodologia apresenta, num total de seis.

A “Análise de funções/tarefas” é a fase responsável pelo estado atual do desenvolvimento e pelo levantamento das necessidades e oportunidades de melhoramento; os “Requisitos considerados” é a fase onde se define os problemas que devem ser resolvidos; o “Design” é a fase onde é concebida a interação entre o utilizador e o computador; a “Prototipagem” é a fase onde são desenvolvidas várias versões para serem devidamente avaliadas; “Implementação” é a fase onde é finalizado o desenvolvimento do sistema interativo e, por fim, a “Avaliação” é a fase onde são avaliados os resultados de cada fase, verificando se esses mesmos resultados vão de encontro às necessidades do utilizador.

Esta metodologia apresenta uma forma em estrela, sendo que cada uma das suas fases não se encontra ordenada ou ligada sequencialmente, o que faz com que o processo de desenvolvimento possa ser iniciado por qualquer uma das fases, sendo que no centro encontra-se o processo de avaliação, tornando assim esta metodologia muito interativa e

flexível. Isto torna-se numa vantagem em relação às metodologias sequenciais, pois estas últimas requerem que uma fase anterior esteja completamente concluída antes de passar para a próxima fase, o que nem sempre é possível.

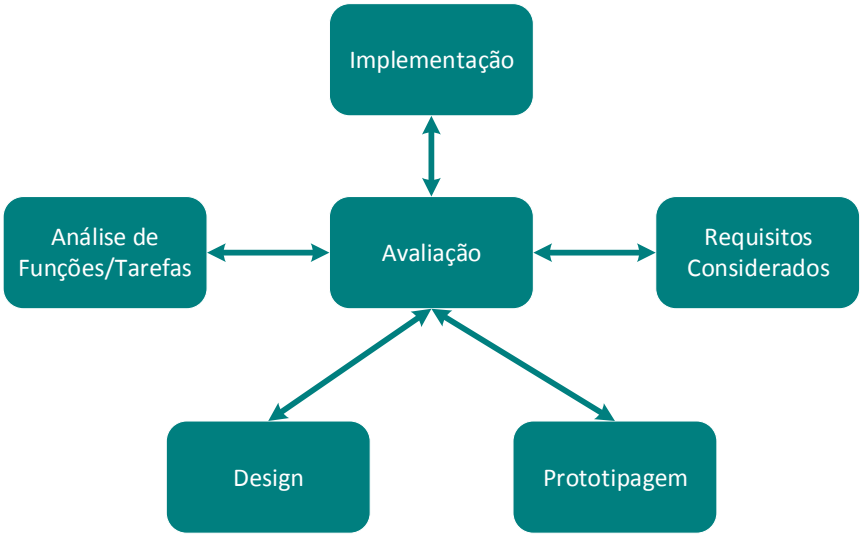


Figura 16 - Metodologia de desenvolvimento em estrela

A *framework* e respetivo videojogo foram desenvolvidos de acordo com a metodologia em estrela, tendo existido durante o processo de desenvolvimento diversas interações para correções e melhorias.

3.5 Interface

Os videojogos são constituídos por diversos elementos, como os objetos gráficos constituintes de uma cena, as animações, os efeitos visuais e sonoros, a vibração de um comando a uma determinada ação, mas também apresentam a forma como o jogador é convidado a interagir com o videojogo.

A interface é, então, o que permite ao jogador interagir com um jogo para que, assim, possa existir uma forma de estabelecer um meio de comunicação funcional e intuitivo entre o mundo físico e o virtual. A criação de uma boa interface é fundamental para o sucesso de um videojogo, pois uma interface confusa e sem qualquer contexto faz o jogador afastar-se e desistir de jogar o videojogo.

Esta divide-se em dois grupos distintos: interfaces físicas e gráficas.

3.5.1 Interface Física

Tem que ver com os dispositivos que são utilizados para interagir com um videojogo, seja através de dispositivos de entrada (como um comando de uma consola, o rato e teclado de um computador ou até através do toque num ecrã sensível de um dispositivo móvel), seja através de dispositivos de saída (como o ecrã de um computador ou um dispositivo móvel, o som das colunas ou então através da vibração de um comando).

3.5.2 Interface Gráfica

A interface gráfica tem que ver com a forma como são apresentados os elementos informativos de um determinado videojogo. Esta encontra-se dividida em quatro tipos [33]:

3.5.2.1 Não-Diegética

Este tipo de interface também é conhecida pelo termo *HUD*, sendo uma das mais comuns nos videojogos. Esta é responsável por apresentar as informações mais importantes para o jogador, como a vida, a força ou o dinheiro, conforme cada género de videojogo. Esta informação apenas é visível e audível para o jogador e pode apresentar-se sempre fixa num determinado ponto do ecrã, bem como ser apresentada numa posição dinâmica, mas nunca pertencente ao universo do videojogo.

O *World of Warcraft* (Figura 17) é um dos exemplos de jogos que apresentam este tipo de interface. A informação relativa aos poderes da personagem é apresentada de forma fixa na parte inferior do ecrã, enquanto que os nomes dos jogadores são apresentados em cima da personagem respetiva. Esta última informação é apresentada de forma dinâmica, conforme a posição da personagem.

Figura 17 – Interface não-diegética do videojogo World of Warcraft

Um outro exemplo deste tipo de interface é o videojogo *Street Fighter 2* (Figura 18), que apresenta toda a informação no ecrã de forma fixa.

Figura 18 – Interface não-diegética do videojogo Street Fighter 2

3.5.2.2 Espacial

Neste tipo de interface são apresentadas informações mais complexas do que na interface anterior, tais como sugestões de jogadas, precauções a ter em determinadas etapas ou informações de um determinado objeto em cena. Podem ser apresentadas dentro do próprio mundo do videojogo em 3D, estando ligadas ou não a esse mundo.

Atualmente, este tipo de interface tem vindo a ser cada vez mais utilizado, oferecendo assim um maior envolvimento no jogo ao jogador. Um exemplo de um videojogo que implementa este tipo de interface é o *Splinter Cell Conviction* (Figura 19), que apresenta informação adicional dentro do próprio mundo do videojogo conforme o avanço do jogador num determinado nível.

Figura 19 – Interface espacial do videojogo Splinter Cell Conviction

Um outro exemplo é o *Fable 3* (Figura 20), em que este apresenta uma luz dentro do mundo do videojogo a indicar o caminho que o jogador terá de seguir.

Figura 20 – Interface espacial do videojogo Fable 3

3.5.2.3 Meta

Pode conter elementos que existam no mundo do videojogo mas nem sempre estão visíveis para o jogador. O exemplo mais comum deste tipo de interface é a utilização de efeitos diretamente no ecrã, tais como manchas de sangue para alertar o jogador de ferimentos, como é o caso do jogo *Call Of Duty Modern Warfare 2* (Figura 21).

Figura 21 – Interface meta do videojogo Call Of Duty Modern Warfare 2

Um outro exemplo é o videojogo *Grand Theft Auto 4*, em que o jogador recebe uma chamada ou uma mensagem, a personagem retira o telemóvel do bolso e é apresentado o telemóvel no ecrã, tal como se pode visualizar na Figura 22.

Figura 22 – Interface meta do videojogo Grand Theft Auto 4

3.5.2.4 Diegética

Um novo tipo de interface que tem sido utilizada nos últimos anos é a interface diegética. Este tipo de interface é o mais envolvente para o jogador, pois toda a informação encontra-se dentro do mundo do videojogo, sendo que pode ser vista e ouvida por outras personagens do videojogo. Um exemplo de um videojogo que implementa esta interface é o *Dead Space* (Figura 23).

Figura 23 – Conjunto de várias imagens da interface do videojogo Dead Space

Todos os elementos informativos encontram-se dentro do próprio mundo do videojogo, como a energia da personagem, que é representada através de uma barra na coluna vertebral da personagem, as munições, que são apresentadas diretamente na arma

que estiver a utilizar, e para aceder ao seu inventário é apresentada uma espécie de holograma.

Um outro exemplo de interface diegética é o videojogo *Mirror's Edge*, em que o mecanismo se baseia no *parkour*, ou seja, o jogador terá de chegar o mais rápido possível a um determinado ponto. Com a inexistência de qualquer informação no ecrã, é indicado, através do uso de cores, como o jogador poderá ultrapassar um determinado desafio (Figura 24).

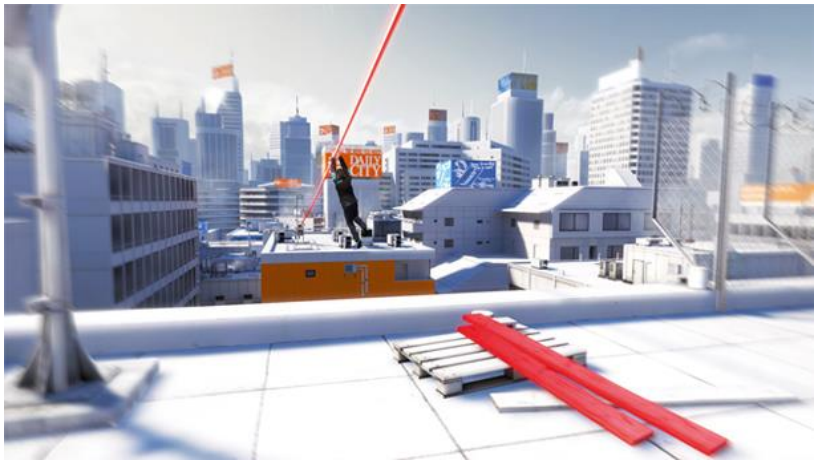


Figura 24 – Interface diegética do videojogo Mirror's Edge

3.5.3 Interface Final

Com base no estudo efetuado aos vários tipos de interface existentes no universo dos videojogos, concluiu-se que uma interface do tipo Não-Diegética será o mais apropriado para o protótipo de videojogo que se pretende desenvolver nesta dissertação. Chegou-se a esta conclusão após experimentar vários videojogos de luta existentes no mercado, como por exemplo, o *Street Fighter* e *Mortal Kombat*, mas também porque é a interface que se apresenta como sendo mais simples, intuitiva e não intrusiva para o jogador, por se encontrar na parte superior do ecrã. Assim, o combate encontra-se na parte inferior do ecrã, tal como se pode visualizar na Figura 25.

Figura 25 - Interface final do protótipo do videojogo

São utilizadas barras para ilustrar a energia da personagem, pois visualmente é mais fácil e rápido para o jogador se aperceber da energia que tem, do que se esta fosse apresentada em percentagem, por exemplo.

A posição central do tempo decrescente de combate, bem como a forma como diminui a energia de um jogador (da periferia do ecrã para o centro), prende-se com o facto de o jogador se encontrar com o seu olhar focado no combate e, para ter essas informações, apenas tem de olhar um pouco para cima e obter a informação do tempo de combate, a sua energia e a do adversário. Assim, torna-se mais rápido saber estas informações, visto que é um videojogo muito rápido.

A escolha da interface física, ou seja, os dispositivos com que o jogador poderá interagir com o videojogo, recaiu sobre o teclado e o comando. Com estes dispositivos torna-se mais simples e rápido o acesso aos diferentes menus do videojogo e ao combate em si.

3.6 Sumário do capítulo

Neste capítulo foi possível apresentar a estrutura que deverá ser implementada na *framework*, tendo sido escolhida uma arquitetura modular, para que se possa, a qualquer momento, modificar qualquer componente, sem que seja afetado o correto funcionamento da mesma. Para além da estrutura da ferramenta, foi necessário estabelecer os requisitos funcionais e não funcionais, tanto da *framework* como do videojogo, para que não se descarte qualquer funcionalidade base.

Além disso, surgiu a necessidade de seguir um tipo de metodologia que fosse capaz de ir de encontro às necessidades deste trabalho, sendo que a metodologia em estrela se revelou ser a mais indicada para este procedimento, o que irá fazer com que o processo de implementação seja mais rápido, seguro e organizado.

Por fim, o estudo das várias interfaces existentes no universo dos videojogos fez com que se seja expectável que tenha conseguido obter a melhor interface para o videojogo que se pretende desenvolver (interface não-diegética), criando, assim, numa melhor experiência para o jogador.

Com base nesta informação devidamente tratada e estudada, a implementação é a fase que se segue.

4 Framework

Após efetuada a análise da solução proposta e do estudo das tecnologias a serem utilizadas, neste capítulo pretende-se apresentar o processo de implementação da *framework*, bem como o protótipo do videojogo, passando pela explicação do funcionamento das principais classes de ambos. Como foi referido na Introdução desta dissertação, a *framework* foi desenvolvida por dois alunos, sendo que cada aluno ficou responsável por determinadas partes da *framework*. Nesta dissertação, irão ser apresentadas e explicadas apenas as partes que coube ao autor desta dissertação desenvolver.

4.1 Esqueleto da *Framework*

Uma das principais preocupações na criação da *framework* é que a mesma seja capaz de suportar as bases de qualquer tipo de videojogo e que, ao mesmo tempo, seja simples e prática de utilizar, para que o desenvolvimento de um videojogo seja o mais rápido possível. Para isso, a *framework* encontra-se estruturada em quatro módulos principais, sendo que cada um é responsável por determinadas tarefas bem específicas da *framework*. A implementação das classes que pertencem a cada um dos módulos foi dividida de igual forma por ambos os alunos, sendo que o aluno Ricardo Moreira, autor desta dissertação, terá de desenvolver todas as classes que se encontram na Figura 26.

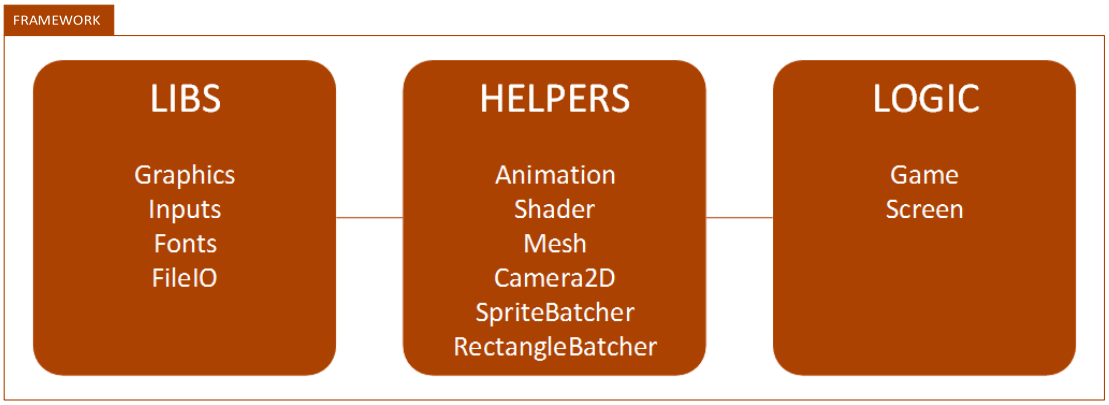


Figura 26 - Classes desenvolvidas pelo aluno Ricardo Moreira

Relativamente ao aluno Gil Canizes [34], este terá de desenvolver todas as classes que se encontram na Figura 27.

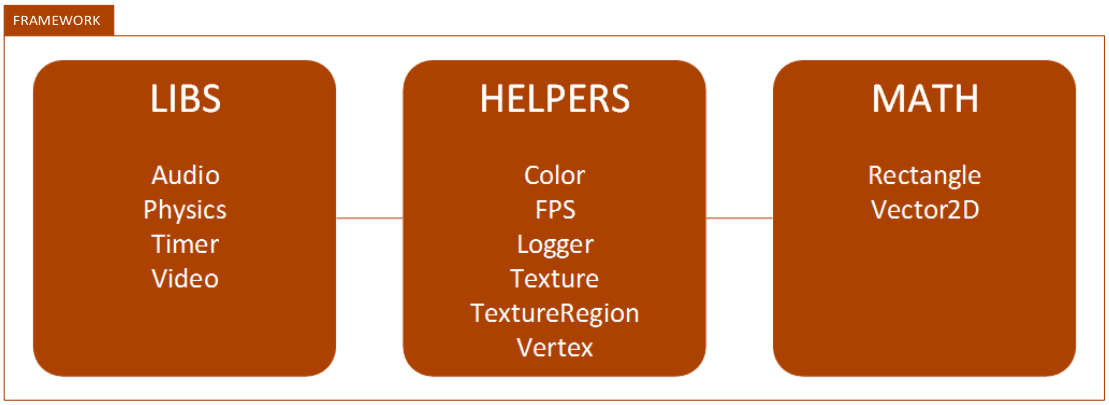


Figura 27 - Classes desenvolvidas pelo aluno Gil Canizes

4.2 Implementação

Durante o desenvolvimento da *framework*, todas as classes foram devidamente documentadas, sendo que no topo de uma classe é possível saber qual a sua funcionalidade, em que versão se encontra e em que data foi efetuada a última atualização. Relativamente aos métodos, é apresentada uma breve descrição acima da sua assinatura e explicada a função de cada parâmetro, caso o método assim o exija. Com esta preocupação, será mais fácil compreender o funcionamento de uma determinada classe ou método.

A *framework* encontra-se dividida em quatro principais módulos, sendo que cada um é constituído por diversos componentes.

Para além destes módulos, quando um videojogo é inicializado, este terá de começar por uma determinada classe que depois irá desencadear todo o resto do processo. A classe

Main é a responsável por essa tarefa de desencadear todo o processo de criação e inicialização de toda a *framework* e do videojogo.

4.2.1 Módulo *Logic*

Para que seja possível interligar os diferentes módulos e de forma correta, é necessária a criação de algumas classes responsáveis por essas tarefas. Com essa necessidade, foram criadas duas diferentes classes sendo que cada uma terá operações bem específicas.

4.2.1.1 Game

É a classe responsável por inicializar os objetos, como por exemplo gráficos, áudio, físicas e dispositivos de entrada.

4.2.1.2 Screen

É a classe responsável por apresentar o conteúdo de uma determinada cena no ecrã.

4.2.2 Módulo *Libs*

Este é o módulo mais importante da *framework*, sendo responsável pelo acesso a todas as bibliotecas existentes. A utilização de algumas dessas bibliotecas solicitou que fossem novamente compiladas, visto que estas apenas se encontravam compiladas para funcionar em sistemas operativos *Windows*, o que fazia com que não fosse possível executar esta *framework* e respetivo videojogo num sistema operativo *Linux* ou *Mac OS*. As ferramentas utilizadas para fazer esse processo de recompilação foram o *MinGW* e o *MSYS*, que têm como objetivo trazer para o sistema operativo *Windows* as principais funcionalidades de compilação do sistema operativo *Linux*.

São vários os componentes que constituem este módulo, sendo que nesta dissertação apenas irão ser apresentados os componentes de Gráficos, Dispositivos de entrada, Leitura e escrita de ficheiros e Estilos de letra.

4.2.2.1 Gráficos

A componente de gráficos é das componentes mais importantes da *framework,* pois sem ela não seria possível apresentar qualquer tipo de objeto no ecrã, quer fosse uma textura ou apenas uma forma geométrica. Esta componente encontra-se dividida em quatro classes: *Graphics*, *Graphics*_Image, Graphics_*Window* e *Graphics_OpenGL*, como se pode verificar na Figura 28, sendo que cada uma é responsável por tarefas bem definidas.

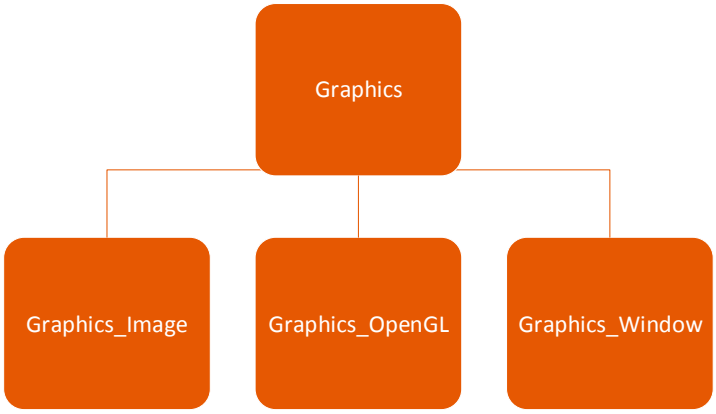


Figura 28 - Classes do componente gráfico

A classe *Graphics* é responsável por dar o acesso às restantes classes base, fazendo com que exista um melhor encapsulamento e uma menor dependência do código.

A classe *Graphics_OpenGL* é responsável pelo acesso aos métodos da *API* do *OpenGL*. A versão do *OpenGL* utilizada nesta *framework* será a 3.2, uma vez que, de acordo com os dados obtidos no *site* da *Steam* [35], mais de 96% dos utilizadores tem um computador que suporta uma versão igual ou superior à 3.2.

Assim, ao utilizar esta versão do *OpenGL* será possível tirar proveito das novas implementações adicionadas a esta *API* porque nas primeiras versões todo o processo de criação de métodos para iluminação, transformações de matrizes, texturização e colorização era desenvolvido pela própria *OpenGL*, o que trazia alguns problemas para os programadores e para os fabricantes de placas gráficas:

- Os fabricantes eram obrigados a desenvolver muito *software* para responder a pedidos por parte das empresas de videojogos;
- Se uma empresa de videojogos criasse um novo efeito visual e quisesse que os fabricantes o implementasse, a concorrência iria a ficar a saber;
- Os videojogos ficavam muito parecidos com os da concorrência devido às limitações das placas;

Para resolver este problema, foi desenvolvida uma solução que passa por programar a placa gráfica, ou seja, a introdução de uma *pipeline*[1] programável, o que faz com que os programadores adicionem os seus próprios programas e, assim, a responsabilidade passa a ser exclusivamente do programador da empresa de videojogos.

De seguida, vem a classe *Graphics_Image,* que é responsável por carregar e obter a informação de uma determinada textura num formato, como *PNG*, *JPEG* ou *BMP*. Esta classe é baseada na biblioteca gratuita *stb_imagem,* que foi criada para esse propósito.

Para finalizar a componente de Gráficos, a classe *Graphics_Window* é responsável pelo acesso à biblioteca *GLFW*, sendo que esta é responsável pela criação e comportamento da janela. Nesta classe, o programador terá acesso a um vasto leque de funcionalidades, como a dimensão da janela, a lista de resoluções que são suportadas pelo computador e definir se pretende que inicialize em modo de janela ou em ecrã cheio.

4.2.2.2 Inputs

A componente de *inputs* também é das componentes mais importantes da *framework,* visto que sem ela não seria possível interagir com os objetos no ecrã. Esta componente encontra-se dividida em quatro classes: *Inputs*, *Inputs_Mouse*, *Inputs_Keyboard* e *Inputs_Joystick*, como se pode verificar na Figura 29, sendo cada uma responsável por determinadas tarefas.

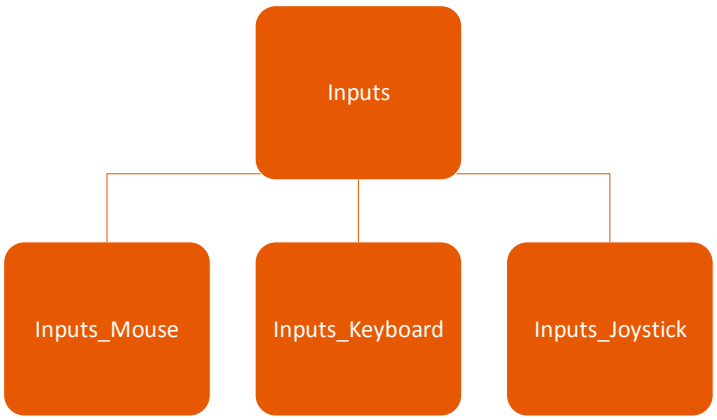


Figura 29 - Classes do componente inputs

[1] Conjunto de métodos criados pelo *OpenGL* para transformar as linhas de código em formas no ecrã

A classe principal *Inputs* é responsável por obter toda a informação sobre cada dispositivo, como rato, teclado e *joystick* através das classes que tratam de cada *input*. Isto é possível graças à biblioteca *GLFW* que traz suporte aos dispositivos de entrada, sendo que estas informações são obtidas através de *callbacks*[2].

A classe *Input_Mouse* é responsável por registar todas as ações do rato, seja a sua posição na janela, seja qual o botão clicado ou qual o estado de um determinado botão e de enviar uma lista com essa informação para a classe principal, a *Input*.

A classe *Input_Keyboard* é responsável por tratar de todos os eventos do teclado, como por exemplo a verificação de que tecla foi pressionada. Esta informação é depois enviada através de uma lista para a classe principal, a *Input*.

Por fim, a classe *Input_Joystick* é responsável por detetar e registar todo o comportamento de um *joystick,* como qual o botão que foi pressionado, verificar se existe algum *joystick* ligado ao computador e saber qual o estado de um botão. Depois de obtida toda a informação, esta será enviada através de uma lista para a classe principal, a *Input*.

4.2.2.3 FileIO

Para que a *framework* possa obter acesso a ficheiros de texto, será necessário recorrer a uma classe que possibilite a leitura e escrita de tais ficheiros. Esta classe fornece, então, um método que possibilita a leitura do conteúdo de um ficheiro para a memória e outro método em que escreve para um ficheiro em disco o conteúdo que é passado por parâmetro.

4.2.2.4 Fonts

A possibilidade de criar texto em tempo de execução e com os mais variados estilos de letra, como *OTF* ou *TTF,* só foi possível graças à utilização da biblioteca *FreeType*. Com esta classe, será possível carregar qualquer tipo de estilo de letra e apresentar qualquer tipo de texto, desde que esta suporte os carateres apresentados num texto. Caso contrário, não será possível apresentar corretamente o texto.

[2] *Callbacks* são rotinas que serão chamadas para tratar eventos

4.2.3 Módulo *Helpers*

Este módulo pretende auxiliar o programador em operações muito frequentes, fazendo com que o número de linhas de código utilizadas para a criação de uma determinada operação seja muito reduzido, facilitando assim a leitura do próprio código. Neste módulo poderão ser encontradas diversas classes, sendo que nesta dissertação apenas irão ser apresentadas as classes *Animation*, *Shader*, *Mesh*, *SpriteBatcher*, *RectangleBatcher* e *Camera2D.*

4.2.3.1 Animation

A classe *Animation* possibilita a animação de um conjunto de determinadas texturas no ecrã. Para que isso seja possível, esta classe utiliza o tempo em milissegundos gerado pela *framework* e calcula quando deverá apresentar a próxima textura de entre um conjunto de texturas. Assim, é possível a animação de qualquer textura em qualquer resolução. A classe também possibilita que a animação tenha um modo de ciclo infinito ou apenas funcione um único ciclo.

4.2.3.2 Shader e Mesh

Estas duas classes apresentam uma elevada importância para um correto funcionamento da *framework,* devido às mesmas serem responsáveis por criar e apresentar objetos na janela, fazendo uso da nova *pipeline* do *OpenGL* [36]. Esta nova *pipeline* programável veio retirar as limitações que existia nas versões anteriores do *OpenGL* e terá um aspeto semelhante ao que é apresentado na Figura 30.

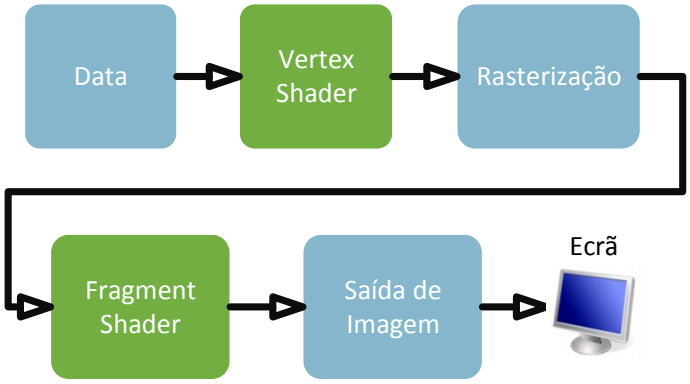


Figura 30 - Versão simplificada do pipeline do OpenGL

As áreas que se encontram em azul são do controlo do próprio *OpenGL*. Já as áreas que estão coloridas a verde, são as novas áreas introduzidas na *pipeline*, tendo agora que o

programador criar o código para elas. Para que isso seja possível, o programador terá de utilizar uma linguagem de programação específica, o *GLSL* (*OpenGL Shading Language*). Esta linguagem foi desenvolvida para que as placas gráficas possam interpretar o que é escrito nesse código, sendo que a sintaxe desta linguagem é muito idêntica à linguagem *C*. Com isto será possível desenvolver código para o *Vertex Shader* e para o *Fragmente Shader*.

Todo este processo é feito de forma paralela e cada uma destas áreas tem uma funcionalidade específica:

- *Data*: É nesta área que é recebida a informação para a criação de algo no ecrã, ou seja, será nesta fase que a classe *Mesh* será utilizada para a criação de toda a informação importante do objeto, tal como a posição, textura e cor;
- *Vertex Shader*: É responsável por converter a informação que é passada pela *Data* para uma forma em que o *GPU* (Unidade de Processamento Gráfico) perceba e, assim, nesta fase, possa criar os pontos conforme as coordenadas que recebeu. É nesta fase que irá ser interpretado o código que o programador criou para o *Vertex Shader*;
- Rasterização: Nesta fase o *OpenGL* irá ligar todos os pontos que foram criados anteriormente e preencher a forma que resultará da ligação dos pontos;
- *Fragment Shader*: É responsável por calcular a cor de cada *pixel* da forma que foi gerada anteriormente. É nesta fase que irá ser interpretado o código que o programador criou para o *Fragment Shader*;
- Saída de Imagem: Nesta última fase será feita a junção de todos os pixéis e criada a forma que será apresentada no ecrã.

Graças à utilização desta nova *pipeline* e às diferenças gráficas que existem entre as diversas empresas de desenvolvimento de videojogos, foi possível obter a qualidade gráfica dos jogos de hoje em dia.

4.2.3.3 SpriteBatcher

A classe *SpriteBatcher* foi criada para ser possível apresentar texturas em qualquer posição da janela de uma forma simples e rápida para o programador. Só foi possível criar esta classe graças à junção de outras classes, como por exemplo *Mesh*, *Shader*, *Texture* e *Color*, em que juntas fazem com que seja possível apresentar na janela várias texturas ao

mesmo tempo. Nesta classe também é utilizada a biblioteca *GLM* para o apoio ao cálculo para efetuar as translações, rotações e escala de uma determinada textura.

Com a criação desta classe, é apenas necessário uma linha de código para que seja possível apresentar algo no ecrã (Figura 31).

```
drawSprite(positionX, positionY, Width, Height, TextureRegion, Texture);
```

Figura 31 - Linha de código para desenhar algo no ecrã

4.2.3.4 RectangleBatcher

A classe *RectangleBatcher* é responsável por desenhar na janela um quadrado ou um retângulo conforme as dimensões que o programador assim o desejar. O seu funcionamento é muito idêntico à classe *SpriteBatcher,* excetuando o facto de que esta não apresenta texturas.

4.2.3.5 Camera2D

Para que se possa visualizar os objetos criados numa cena, é necessária a criação de uma câmara. Essa câmara poderá ser 2D, onde apenas se trabalha com o eixo das coordenadas X e Y, deixando o Z sempre com o valor 0, ou poderá ser 3D, trabalhando com todos os eixos X, Y e Z. O objetivo desta *framework* é a criação de videojogos em 2D, ou seja, a melhor opção será a criação de uma câmara 2D.

Assim, será necessário definir um tipo de projeção. Normalmente este tipo de jogos utiliza uma câmara com projeção paralela ortogonal, pois assim todos os objetos parecem ter o mesmo tamanho, independentemente de quão próximos ou afastados estejam da câmara (Figura 32), o que facilita os cálculos, pois não é necessário existir preocupação com a profundidade a que se encontram no eixo das coordenadas do Z.

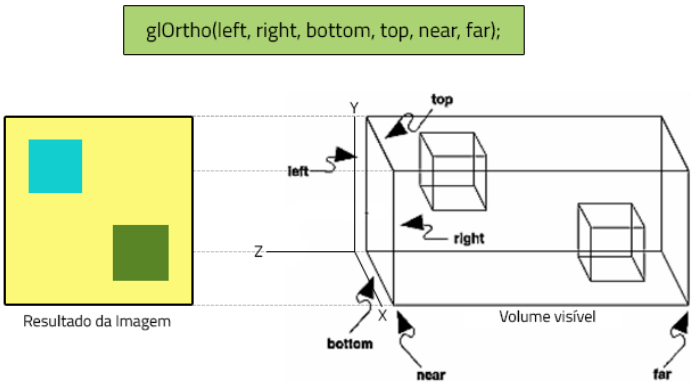


Figura 32 - Projeção Paralela Ortogonal

Para a criação desse tipo de projeção, foi utilizada a biblioteca *GLM* no apoio ao cálculo.

Além da projeção, é necessário especificar a área da janela na qual se pretende apresentar os objetos de uma cena. Para isso, será utilizado o método existente no *OpenGL*, o *glViewPort*. Com a utilização deste método é possível definir exatamente em que posição da janela queremos que seja desenhada a cena e qual o seu tamanho, fazendo com que seja possível ocupar toda a sua janela, independentemente do tamanho da mesma. Um pormenor a ter em conta quando se utiliza este método é que a imagem projetada poderá deformar um pouco o seu conteúdo. Para colmatar esse problema é necessário calcular a razão de aspeto da imagem para que os objetos de uma cena não fiquem deformados (Figura 33).

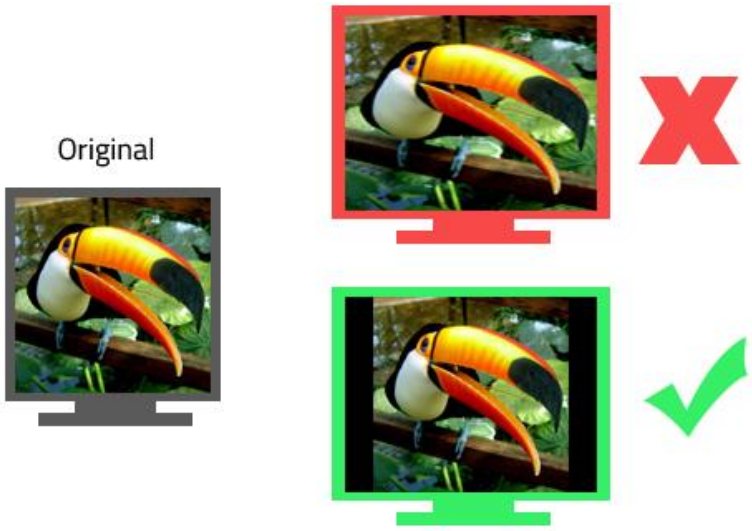


Figura 33 - Razão de aspeto da imagem

4.3 Sumário do capítulo

Neste capítulo foi apresentada a implementação da *framework*, onde foi descrita a arquitetura que deverá ser implementada e quais as classes que cada aluno deverá desenvolver para que exista uma divisão correta. Após essa separação, segue-se a descrição do funcionamento de cada classe que é implementada, passando assim algumas noções de como é o funcionamento de uma *framework*.

5 Protótipo do Videojogo

Após a implementação da *framework* concluída, será necessário demonstrar as funcionalidades da mesma. Assim, para demonstrar que a *framework* suporta diferentes géneros de videojogos, foram desenvolvidos dois géneros diferentes[3]. Nesta dissertação será descrita a forma como o protótipo de um videojogo de luta foi criado.

5.1 Implementação

No seguimento do levantamento de requisitos, este tipo de jogo apresenta diversas funcionalidades, como por exemplo a colisão entre duas personagens, contabilização da duração de um combate, a possibilidade de defrontar o computador ou um outro jogador e a possibilidade de interagir com diferentes dispositivos de entrada. Tudo isto torna a implementação destas funcionalidades um bom conjunto de testes ao correto funcionamento da *framework*.

Com o começo da implementação do videojogo, surgiu a necessidade de criar uma separação das várias partes constituintes do videojogo. Concluiu-se que este deveria ficar separado em dois módulos (Figura 34).

[3] cf. Criação de uma Framework Modular Vocacionada para o desenvolvimento de videojogos

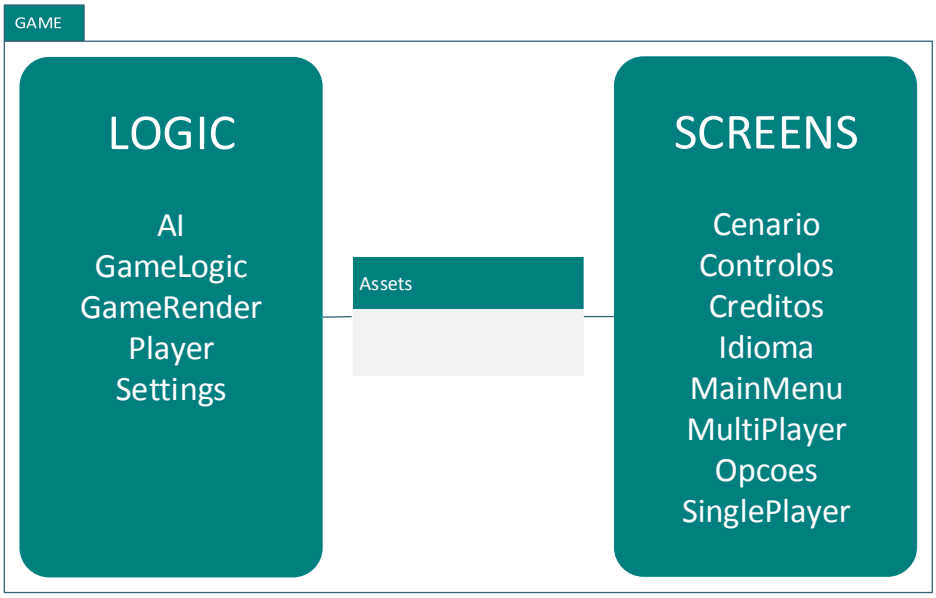


Figura 34 - Classes do videojogo

Esta separação visa trazer uma melhor organização na estrutura do videojogo, sendo que o módulo *Logic* irá conter a parte de lógica e funcionamento do videojogo, enquanto que o módulo *Screens* irá conter a parte dos menus, sendo que ambos acedem à classe *Assets* onde esta será responsável por conter toda a informação sobre texturas, sons, músicas e estilo de letra.

5.1.1 *Assets*

Como foi referido anteriormente, esta classe tem como principal objetivo carregar toda arte referente ao videojogo. Exemplo disso é a textura de uma das personagens como se pode visualizar na Figura 35.

Figura 35 - Sequência de 4 frames em modo de repouso da personagem

Neste caso, será criada uma animação em modo de repouso da personagem, sendo que para isso será necessário obter as coordenadas de cada *frame* da textura. Ao obter essa informação e com a utilização da classe *Animation*, será possível apresentar no ecrã esta animação.

5.1.2 Módulo *Logic*

A cena de combate entre duas personagens controladas através do computador, ou através de jogadores, apresenta grande importância neste videojogo. Neste módulo, serão apresentadas todas as classes referentes à lógica do videojogo:

- *AI*: Classe responsável pela inteligência artificial de uma personagem controlada pelo computador. O algoritmo desenvolvido é muito simples, sendo que o objetivo é apenas a exemplificação.
- *GameLogic*: Esta classe faz toda a gestão do combate, verificando sempre a posição das personagens na cena, a sua energia e quais as ações determinadas pelo jogador.
- *GameRender*: Faz toda a gestão visual do combate, como por exemplo, onde deverão ser apresentadas as personagens na janela conforme as ações do jogador, o tamanho da barra de energia e apresentação do tempo atual do combate em decrescendo.
- *Player*: Classe responsável por registar toda a informação a respeito do jogador, como por exemplo, a sua energia.
- *Settings*: Classe responsável por gerir toda a informação a respeito do videojogo, como a resolução, tempo e estado da música. Esta informação encontra-se armazenada num ficheiro de texto para que seja possível apresentar as últimas configurações de cada vez que o jogador entra no jogo.

5.1.3 Módulo *Screens*

Para que se possa interagir com um videojogo e configurar determinadas ações do mesmo, a existência de menus é um ponto crucial. Neste módulo serão apresentadas todas as classes referentes a todos os menus presentes no videojogo, sendo nove o número total de menus:

- Idioma: Classe responsável por apresentar na janela as diversas línguas que o videojogo suporta. Cada idioma é apresentado através da bandeira representativa do país do idioma em causa.
- *MainMenu*: Classe principal do videojogo, sendo esta responsável por apresentar as opções para os diferentes menus.

- História: Esta classe terá como responsabilidade apresentar as várias imagens referente à pequena história existente neste videojogo, sendo que cada imagem é apresentada durante 5 segundos e automaticamente irá transitar para a próxima imagem.
- *SinglePlayer*: Responsável por apresentar na janela uma matriz de 3 por 3 onde serão colocadas as personagens disponíveis, sendo que neste videojogo apenas irão ser apresentadas duas personagens. Ao mesmo tempo que o jogador seleciona uma personagem, do lado esquerdo irá ser apresentada uma imagem de corpo inteiro da personagem.
- *MultiPlayer*: Responsável por dar a possibilidade de dois jogadores poderem combater entre si, sendo que nesta classe irão ser apresentadas ao jogador as diversas personagens disponíveis. Cada jogador terá de escolher uma, sendo que é possível que ambos os jogadores possam ter a mesma personagem. Como foi referido na classe *SinglePlayer*, esta também apresenta uma matriz de 3 por 3 onde serão colocadas as personagens. No lado esquerdo será apresentada a personagem do jogador 1 e no lado direito será apresentada a personagem do jogador 2.
- Créditos: Nesta classe será apresentado um pequeno texto sobre o autor deste videojogo.
- Opções: Classe responsável por apresentar diversas opções existentes no videojogo, tais como ligar ou desligar sons ou a música, alterar a resolução e modificar o tempo de um combate. Ao sair deste menu, irão ser automaticamente guardadas as configurações que o jogador efetuou.
- Controlos: Esta classe é responsável por apresentar na janela os controlos disponíveis para um jogador ou dois jogadores. Neste menu poderá configurar qualquer controlo através do teclado ou de um comando, sendo que ao sair do menu irão automaticamente ficar registadas todas essas alterações.
- Cenário: Esta classe é responsável por apresentar o menu cenário, onde o jogador poderá selecionar um cenário dentro de três disponíveis.

5.2 Sumário do capítulo

Neste capítulo foi apresentada uma visão geral da implementação do protótipo do videojogo, passando pela apresentação das classes que o constituem e descrito o funcionamento de cada uma delas.

6 Conclusão

Neste capítulo será apresentada a conclusão do tema em estudo, passando por diversos componentes, como os objetivos concretizados, as limitações que apresenta a *framework* e videojogo, qual o futuro desta *framework* e a apreciação final do trabalho realizado.

6.1 Objetivos realizados

Este trabalho teve como ponto de partida o levantamento do estado da arte dos videojogos, passando pelo processo de recolha de vários dados informativos e estatísticos, o que permitiu obter um maior conhecimento sobre o estado atual desse universo.

Para além do levantamento do estado da arte dos videojogos, era necessário perceber quais as ferramentas que eram utilizadas para o processo de criação de um videojogo. Para isso, foi também essencial efetuar o levantamento do estado da arte tecnológico, dando a conhecer as mais diversas ferramentas, bem como as principais características de cada uma delas, fornecendo assim uma melhor perceção do que cada uma era capaz de realizar. Com base em algumas dessas características fez-se um levantamento de requisitos da *framework* que se pretende desenvolver.

A parte principal desta dissertação foi o desenvolvimento de uma *framework* capaz de suportar as funcionalidades básicas de qualquer género de videojogo, dando assim uma maior liberdade no género de videojogo a ser criado. Este processo de desenvolvimento da *framework* passou por um longo estudo de várias tecnologias que poderiam ser

implementadas na ferramenta e de vários testes para verificar se as mesmas eram capaz de se interligar e de realizar o que se era pretendido.

Já na parte final desta dissertação, foi desenvolvido um protótipo de um videojogo com base no estudo realizado durante este trabalho, para que se possa apresentar um videojogo com uma boa interface para o jogador e, também, demonstrar as principais funcionalidades da ferramenta que foi desenvolvida. Assim, foi possível testar e avaliar a capacidade de resposta da *framework*.

6.2 Limitações e trabalho futuro

Apesar dos objetivos principais terem sido alcançados com sucesso, existem algumas limitações que devem ser colmatadas num trabalho futuro:

- A inexistência de algumas opções gráficas para que o jogador possa facilmente reduzir ou melhorar a qualidade gráfica de um videojogo e, assim, fazer com que o mesmo possa apresentar um desempenho melhor ou pior;
- A necessidade do desenvolvimento de um novo videojogo ter de passar pela criação manual de todas as classes e respetivo código, o que poderia ser substituído por uma interface simples e amigável para tornar o processo de criação mais simples e rápido para o programador;
- O facto da *framework* apenas suportar três plataformas torna-se numa limitação nos dias de hoje, visto que existe um vasto leque de plataformas que a mesma poderia suportar, como por exemplo, as consolas ou os dispositivos móveis.

6.3 Apreciação final

Graças ao levantamento do estado da arte, obteve-se uma melhor noção de como se encontra o mercado atualmente e quais as suas tendências futuras.

O estudo e desenvolvimento desta *framework* foi muito enriquecedor a nível pessoal, pois o conhecimento e experiencia técnica nesta área dos videojogo era muito reduzida. Assim, foram adquiridos novos conhecimentos e uma melhor compreensão sobre o funcionamento de uma *framework* e os aspetos importantes a ter em conta durante o seu processo de criação, nomeadamente os cuidados a ter quando se trata de uma ferramenta que poderá vir a ser utilizada por outras pessoas.

Para terminar, é esperado que esta *framework* possa vir a melhorar em vários aspetos técnicos e que consiga um dia ser equiparável a grandes *frameworks* existentes no mercado.

Referências

[1] Insane Sheep, “Insane Sheep – Game Studio”, http://insanesheep.com/, [Último acesso: 20 de Outubro de 2014]

[2] Michael Miller, “A History of Home Video Game Consoles”, http://www.informit.com/articles/article.aspx?p=378141, [Último acesso: 20 de Outubro de 2014]

[3] Ivan Barroso, “Portugal tem capacidade para videojogos?”, http://p3.publico.pt/vicios/hightech/12947/portugal-tem-capacidade-para-videojogos, [Último acesso: 20 de Outubro de 2014]

[4] Nuno Folhadela, “História dos videojogos para consolas de Nuno Folhadela”, http://pt.slideshare.net/realidadesvirtuais/histria-dos-videojogos-para-consolas-de-nuno-folhadela, [Último acesso: 20 de Outubro de 2014]

[5] Newzoo, “Global Games Market Report Infographics”, http://www.newzoo.com/infographics/global-games-market-report-infographics/, [Último acesso: 20 de Outubro de 2014]

[6] esa, “Essential Facts About The Computer And Video Game Industry”, http://www.theesa.com/facts/pdfs/esa_ef_2013.pdf, [Último acesso: 20 de Outubro de 2014]

[7] Cocos2D-X, “Cocos2d-x: World’s #1 Open Source Game Development Platform”, http://www.cocos2d-x.org/, [Último acesso: 3 de Junho de 2014]

[8]LÖVE, “LÖVE – Free 2D Game Engine”, https://love2d.org/, [Último acesso: 3 de Junho de 2014]

[9] Microsoft, “XNA Game Studio 4.0 Refresh”, http://msdn.microsoft.com/en-us/library/bb200104.aspx, [Último acesso: 3 de Junho de 2014]

[10] Angel2D, “Angel2D”, http://angel2d.com/, [Último acesso: 3 de Junho de 2014]

[11] Oxygine, “Oxygine | Oxygine is C++ engine and framework for 2D games on iOS, Android, Windows, Linux and Mac”, http://oxygine.org/, [Último acesso: 3 de Junho de 2014]

[12] LabLua, “The Programming Language Lua”, http://www.lua.org/about.html, [Último acesso: 20 de Outubro de 2014]

[13] Dan Scharff, “Gamers, Start Your Engines! 6 Top Gaming Engines Face Off”, http://www.maximumpc.com/article/features/gamers_start_your_engines_6_top_gaming_engines_face, [Último acesso: 20 de Outubro de 2014]

[14] Mark Masters, “Choosing the Right Game Engine | Unity, UDK, Unreal Engine 4 or CryENGINE”, http://blog.digitaltutors.com/unity-udk-cryengine-game-engine-choose/, [Último acesso: 20 de Outubro de 2014]

[15] Unity Technologies, “Unity – Game engine, tools and multiplatform”, http://unity3d.com/unity, [Último acesso: 3 de Junho de 2014]

[16] Epic Games, “Unreal Engine Technology | Home”, https://www.unrealengine.com/, [Último acesso: 3 de Junho de 2014]

[17] CryEngine, “CRYENGINE | Crytek”, http://cryengine.com/features, [Último acesso: 3 de Junho de 2014]

[18] Valve, “Source SDK 2013 – Valve Developer Community”, https://developer.valvesoftware.com/wiki/Source_SDK_2013, [Último acesso: 20 de Outubro de 2014]

[19] id Software, “id Software”, http://www.idsoftware.com/, [Último acesso: 3 de Junho de 2014]

[20] Khronos Group, “OpenGL Overview”, http://www.opengl.org/about/, [Último acesso: 3 de Junho de 2014]

[21] Firelight Technologies, “FMOD”, http://www.fmod.org/, [Último acesso: 3 de Junho de 2014]

[22] GLFW, “GLFW – An OpenGL library”, http://www.glfw.org/, [Último acesso: 3 de Junho de 2014]

[23] Kitware, “CMake – Cross Platform Make”, http://www.cmake.org/, [Último acesso: 3 de Junho de 2014]

[24] Xiph.Org Foundation, “Xiph.org”, http://xiph.org/vorbis/, [Último acesso: 3 de Junho de 2014]

[25] Krešimir Špes, “Theora Playback Library”, http://libtheoraplayer.cateia.com/wiki/index.php/Main_Page, [Último acesso: 20 de Outubro de 2014]

[26] FreeType, “FreeType Overview”, http://www.freetype.org/freetype2/index.html, [Último acesso: 20 de Outubro de 2014]

[27] G-Truc Creation, “OpenGL Mathematics”, http://glm.g-truc.net/0.9.5/index.html, [Último acesso: 20 de Outubro de 2014]

[28] Milan Ikits and Marcelo Magallon, “GLEW: The OpenGL Extension Wrangler Library”, http://glew.sourceforge.net/, [Último acesso: 20 de Outubro de 2014]

[29] Code::Blocks, “Code::Blocks”, http://www.codeblocks.org/, [Último acesso: 3 de Junho de 2014]

[30] Dr. Winston W. Royce, “Managing The Development Of Large Software Systems”, 1970

[31] Barry W. Boehm, “A Spiral Model of Software Development and Enhancement”, IEEE Computer, vol. 21, no. 5, pp. 61-72, Março 1988

[32] Deborah Hix and H. Rex Hartson, “Developing User Interfaces: Ensuring Usability Through Product & Process”, 1993

[33] Marcus Andrews, “Gamasutra – Game UI Discoveries: What Players Want”, http://www.gamasutra.com/view/feature/4286/game_ui_discoveries_what_players_.php, [Último acesso: 20 de Outubro de 2014]

[34] Gil Canizes, “Criação de uma Framework Modular Vocacionada para o desenvolvimento de videojogos”, 2014

[35] Valve Corporation, “Inquérito Steam de Hardware & Software”, http://store.steampowered.com/hwsurvey/videocard/, [Último acesso: 20 de Outubro de 2014]

[36] Chua Hock-Chuan, “3D Graphics with OpenGL – The Basic Theory”, http://www.ntu.edu.sg/home/ehchua/programming/opengl/cg_basicstheory.html, [Último acesso: 20 de Outubro de 2014]

Anexo 1

Game Design Document do videojogo

Game Design Document

FFF – Fight For Freedom

Autor: Ricardo Miranda Almeida Moreira

Data: 20/10/2014

Versão 1.0

Índice

Índice de Figuras

1 Introdução

Este documento visa apresentar os principais aspetos técnicos e narrativos do videojogo "FFF – Fight For Freedom", tais como a história, o seu objetivo, mecânicas utilizadas e aspetos da jogabilidade. Relativamente a aspetos artísticos, como a arte do videojogo e músicas/sons utilizados no mesmo, foram utilizados recursos gratuitos e que se encontram referenciados no capítulo das Referências, sendo que esta parte não irá ser abordada.

1.1 Género do videojogo

É um videojogo do género de luta, em que o jogador terá de combater contra um adversário. Este videojogo apresenta algumas semelhanças com videojogos já existentes no mercado, como o *Street Fighter* (Figura 1), que é produzido pela empresa *Capcom* e *Mortal Kombat* (Figura 2), que é produzido pela empresa *Midway Games*.

Figura 1 - Imagem do videojogo Street Fighter

Figura 2 – Imagem do videojogo Mortal Kombat

1.2 Audiência

Fight For Freedom é um videojogo para todos os amantes de videojogos de luta e para todos aqueles que gostam de desafios. Este videojogo destina-se a todos os jogadores, mas com uma faixa etária superior a 13 anos, visto ser um videojogo que contém alguma violência.

1.3 Plataforma

O videojogo será desenvolvido para o computador, suportando os principais sistemas operativos *Windows*, *Linux* e *Mac OS*.

2 Visão do videojogo

Neste capítulo será apresentada a história do jogo, as personagens e os cenários que o constituem.

2.1 História

Chefe Kab é um terrível comandante que governa a cidade de Xukipa, roubando dinheiro e maltratando os cidadãos da sua cidade.

Devido à sua fraca popularidade, Chefe Kab desafia qualquer civil da sua cidade a um combate e quem o vencer ficará com o seu lugar. Foram vários os cidadãos que concorreram, mas, infelizmente, nenhum deles o conseguiu derrotar. Com uma enorme vontade de vencer, Mukiko é um jovem que pretende libertar os cidadãos das garras do Chefe Kab. Para isso, Mukiko irá inscrever-se no combate e terá de vencer para que finalmente seja restaurada a paz na cidade.

2.2 Personagens

Neste videojogo encontram-se disponíveis duas personagens, o Chefe Kab (Figura 3) e o Mukiko (Figura 4).

Figura 3 - Personagem Chefe Kab

Figura 4 - Personagem Mukiko

2.3 Cenários

O jogador poderá realizar combates em diferentes cenários, sendo que neste videojogo se encontram disponíveis três, como se pode visualizar pelas Figuras 5, 6 e 7.

Figura 5 – Cenário 1

Figura 6 - Cenário 2

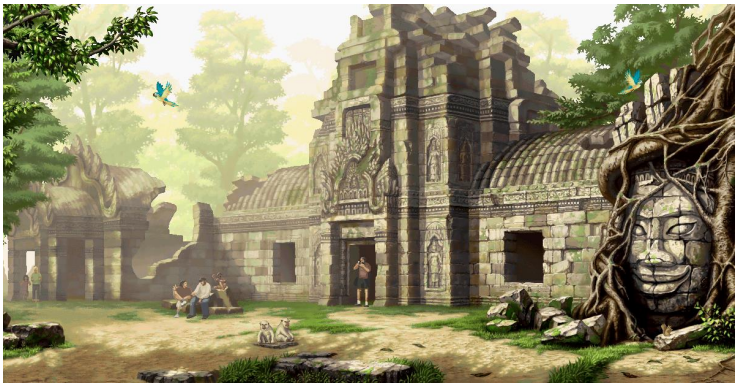


Figura 7 - Cenário 3

3 Jogabilidade

Neste capítulo serão descritos os controlos que o jogador terá de utilizar para interagir com o videojogo como as mecânicas implementadas no mesmo.

3.1 Controlos

O videojogo terá suporte a dois tipos de dispositivos de entrada, o teclado e o comando. Através de um desses dispositivos de entrada, o jogador poderá jogar ou navegar entre os diversos menus e opções.

3.1.1 Teclado

Com o uso das teclas existentes no teclado, o jogador poderá fazer diversas ações que irão ser apresentadas de seguida.

Navegação nos menus:

Figura 8 - Imagem ilustrativa das teclas de navegação nos menus

- A tecla “Esc” permite voltar ao menu anterior;
- As teclas direcionais permitem selecionar uma opção do menu;
- A tecla “Enter” permite acionar a opção selecionada do menu.

Navegação num combate:

Figura 9 - Imagem ilustrativa das teclas de navegação do jogador

- As teclas direcionais permitem deslocar o jogador para o lado esquerdo ou lado direito, bem como saltar ou ficar de cócoras;
- A tecla “X” permite ao jogador dar socos;
- A tecla “Z” permite ao jogador dar pontapés;
- A tecla “Esc” permite fazer pausa no combate.

Estas serão as teclas configuradas por omissão, mas será possível o jogador definir as suas próprias teclas, para assim tirar maior proveito do teclado.

3.1.2 Comando

Para além de ser possível o jogador utilizar o teclado como forma de interação com o videojogo, também é possível que utilizar um comando de computador para interagir com o videojogo. O jogador terá disponíveis diversas ações que irão ser apresentadas de seguida.

Navegação nos menus:

Figura 10 - Imagem ilustrativa dos botões de navegação nos menus

- O botão “Back” permite voltar ao menu anterior.
- Os botões direcionais permitem selecionar uma opção do menu.
- O botão “B” permite acionar a opção selecionada.

Navegação num combate:

Figura 11 - Imagem ilustrativa dos botões de navegação do jogador

- O botão “Back” permite fazer pausa no combate;
- Os botões direcionais permitem movimentar o jogador para o lado esquerdo ou lado direito, bem como saltar ou ficar de cócoras;

- O botão “B” permite que o jogador dê socos;
- O botão “A” permite que o jogador dê pontapés.

Como acontece no teclado, estes são os botões configurados por omissão, mas será possível o jogador definir os seus próprios comandos.

3.2 Câmara

A câmara terá uma vista frontal em 2D e irá seguir a personagem de forma automática para qualquer posição. O sistema de coordenadas (X,Y) adotado será (0,0), no canto inferior esquerdo, e (1280,800), no canto superior direito, sendo que o eixo do X cresce para a direita e o eixo do Y cresce para cima.

3.3 Mecânicas

O videojogo apresenta uma visão frontal em 2D (Figura 12) em que o jogador poderá movimentar a sua personagem para a direita e esquerda, sendo que ao mesmo tempo o jogador poderá utilizar várias teclas de combate, como socos e pontapés, para derrotar o seu adversário.

Figura 12 – Imagem da visão do videojogo

Sempre que o jogador consiga acertar no seu adversário através de um soco, o mesmo irá perder 10% da sua energia e, caso seja através de um pontapé, irá perder 5%.

Por omissão, a duração de um cada combate é cerca de 60 segundos, sendo que este valor poderá ser alterado, se o jogador pretender.

O jogador poderá terminar um combate em três estados possíveis:

Vitória

Para o jogador conseguir arrecadar a vitória, terá de derrotar o seu adversário, fazendo com que a sua energia chegue a 0 ou, quando chegar o fim do tempo, ter mais energia do que o seu adversário.

Derrota

O jogador perde o combate quando a sua energia chega a 0 ou quando chega ao fim do tempo e o adversário tem mais energia do que a sua.

Empate

Chegando ao fim do tempo e a energia de ambos é igual.

3.4 Multijogador

Com a funcionalidade Multijogador, será possível duas pessoas jogarem uma contra a outra, partilhando como dispositivo de entrada o teclado. Poderá também existir a possibilidade de um dos jogadores utilizar como dispositivo de entrada um comando e o outro jogador ficará com o teclado.

4 Interface

Neste capítulo serão apresentadas as interfaces dos vários menus, bem como a parte de *gameplay* que compõe o videojogo.

4.1 Interface dos menus

4.1.1 Idioma

Neste menu o jogador poderá escolher entre duas línguas (Português e Inglês), como se pode verificar na Figura 13. Este será o primeiro menu que o jogador irá visualizar assim ao iniciar o videojogo.

Figura 13 - Imagem do Menu Idioma

4.1.2 Menu Principal

Relativamente ao menu principal, este é composto por várias opções (Figura 14), que serão explicadas de seguida.

Na opção "História**"**, o jogador será colocado na pele de uma personagem que terá de combater um adversário; a opção "Jogar" permite iniciar um combate contra o computador; de seguida aparece a opção "Multijogador**",** que possibilita dois jogadores combaterem entre si utilizando dispositivos de entrada; "Opções" permite ao jogador modificar diversas preferências, como a resolução do videojogo, configurar os controlos, ativar ou desativar a música e sons do videojogo; "Créditos**"** apresenta a pessoa envolvida no desenvolvimento deste videojogo e, por fim, a opção "Sair" permite ao jogador encerrar o videojogo.

Figura 14 - Imagem do Menu Principal

4.1.3 História

O jogador irá ser colocado num pequeno enredo em que terá de combater contra o comandante Chefe Kab. Antes do início do combate, serão apresentadas diversas imagens para que o jogador possa compreender melhor a história, como se pode verificar na Figura 15.

Figura 15 - Imagem referente à história do videojogo

No final do combate, caso o jogador perca, terá a possibilidade de repetir o combate.

4.1.4 Jogar

É apresentado ao jogador um conjunto de personagens, em que o mesmo terá de escolher uma para o combate (Figura 16). Nesta versão do videojogo, apenas duas personagens estão disponíveis.

Figura 16 - Imagem do Menu Jogar

4.1.5 Cenário

Neste menu, é apresentado ao jogador três diferentes cenários de combate, em que terá de selecionar apenas um para iniciar o combate, como se pode verificar na Figura 17.

Figura 17 - Imagem do Menu Cenário

4.1.6 Pausa

Durante um combate, o jogador poderá fazer pausa em qualquer momento, fazendo com que apareça o menu pausa. Neste menu o jogador poderá continuar o combate ou simplesmente abandonar o mesmo (Figura 18).

Figura 18 - Imagem do Menu Pausa

4.1.7 Multijogador

Para além do menu Jogar, que permite ao jogador disputar um combate contra o computador, o menu Multijogador faz com que seja possível dois jogadores combaterem entre si através de um teclado e um comando. Neste menu é apresentado aos jogadores um conjunto de personagens onde os mesmos terão de escolher a sua personagem, como se pode verificar na Figura 19. Os jogadores poderão escolher diferentes personagens ou a mesma. Nesta versão do videojogo, apenas duas personagens estão disponíveis.

Figura 19 - Imagem do Menu Multijogador

4.1.8 Opções

No menu Opções são apresentas diferentes configurações do videojogo, como se pode verificar na Figura 20.

No lado esquerdo é apresentada a opção **Resolução,** que permite modificar a resolução do videojogo de entre várias resoluções disponíveis. A opção **Tempo** permite definir a duração de um combate e a opção **Modo** permite ao jogador escolher entre duas opções, janela ou ecrã cheio. Já no lado direito é apresentada a opção **Música,** onde o jogador poderá ligar ou desligar a música do videojogo, a opção **Sons,** onde o jogador poderá ligar ou desligar todos os sons do videojogo e, por fim, a opção **Configurar Controlos,** onde é permitido ao jogador modificar as teclas ou botões do teclado ou comando, respetivamente.

Figura 20 - Imagem do Menu Opções

4.1.9 Configurar Controlos

Neste menu o jogador poderá configurar todos os controlos existentes no videojogo. Como se pode verificar na Figura 21, do lado esquerdo, poderá configurar as teclas do teclado, enquanto que, do lado direito, poderá configurar todos os botões do comando, para assim tirar um maior proveito do dispositivo em questão.

Figura 21 - Imagem do Menu Configurar Controlos

4.1.10 Créditos

No menu Créditos é apresentada a pessoa envolvida no desenvolvimento deste videojogo (Figura 22).

Figura 22 - Imagem do Menu Créditos

4.2 Interface de *gameplay*

Nos videojogos normalmente existe informação que é importante o jogador saber, como por exemplo, a sua energia, o tempo e a energia dos adversários. O método como é apresentada essa informação designa-se por interface de *gameplay* ou HUD (*Heads-Up Display*). Essa informação, geralmente, encontra-se no topo ou nas extremidades do ecrã, conforme o tipo de videojogo. Neste caso essa informação encontra-se centrada no topo do ecrã (Figura 23), de forma minimalista e simples, para que seja fácil a sua visualização e compreensão, mas também de forma a não incomodar o jogador.

Figura 23 - HUD do videojogo

Existem algumas informações que são apresentadas ao jogador. Na Figura 24, do lado esquerdo, é apresentada a imagem da personagem escolhida para o combate, na parte superior é apresentado o nome do jogador e, por fim, na parte inferior é apresentada a energia que o jogador tem no combate. Esta informação é a mesma para o adversário.

Figura 24 – Informação do Jogador

É apresentado o tempo em segundos do combate, tal como demonstrado na Figura 25, sendo que o mesmo vai decrescendo até chegar ao valor de zero. Chegando a esse valor, o combate é terminado e vence o jogador que tiver mais energia, ou empata se ambos os jogadores tiverem a mesma.

Figura 25 - Tempo do videojogo

No caso de o jogador vencer o combate, uma mensagem irá ser apresentada, como se pode visualizar na Figura 26.

Figura 26 - Imagem de vitória num combate

Caso o jogador perca o combate, irá ser mostrada uma mensagem, tal como se pode verificar na Figura 27.

Figura 27 - Imagem de derrota num combate

Por fim, caso o jogador empate o combate, irá ser exposta a mensagem representada na Figura 28.

Figura 28 - Imagem de empate num combate

5 Fluxo do Videojogo

O fluxo do videojogo (Figura 29) segue as seguintes etapas:

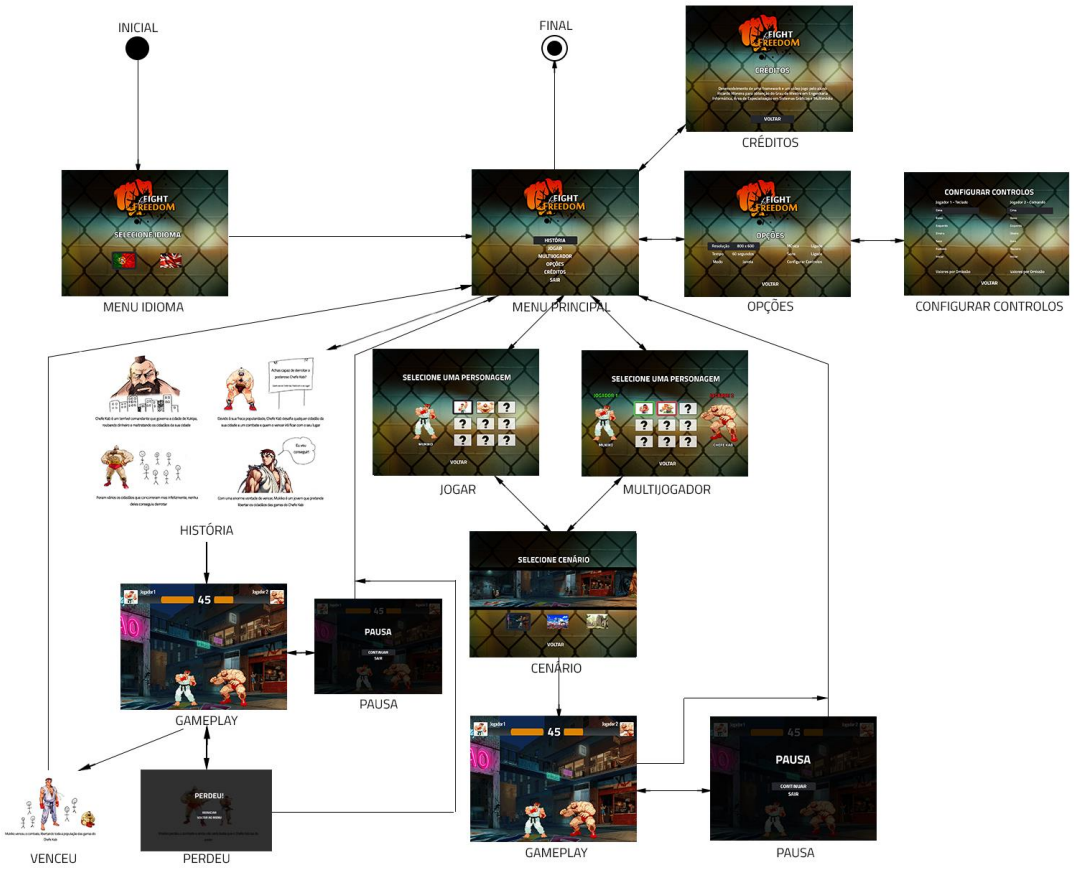


Figura 29 - Fluxo do videojogo

Ao iniciar o videojogo irá ser apresentado o menu Idioma, onde o jogador poderá escolher uma de duas línguas disponíveis. De seguida é apresentado o menu principal, em

que poderá entrar em diferentes menus, como o menu História, Jogar, Multijogador, Opções e Créditos.

Na escolha do menu História, serão apresentadas através de várias imagens a narrativa do videojogo, não podendo o utilizador passar esta parte. No fim desta etapa, o jogador terá de combater contra o seu adversário, podendo, a qualquer altura do combate, fazer pausa no jogo e retomar novamente. No final do combate, caso o jogador o tenha perdido, terá a possibilidade de o repetir ou, então, de voltar ao menu principal; caso tenha vencido, será redirecionado para o menu principal.

Caso o jogador escolha o menu Jogar, irão ser apresentadas duas personagens para selecionar uma e, de seguida, ser-lhe-ão apresentados três cenários diferentes, em que terá de escolher apenas um para o combate. Feitas estas escolhas, irá ser iniciado um combate entre o jogador e o computador. Durante o combate, o jogador poderá aceder ao menu Pausa a qualquer momento e conforme a opção que selecionou irá continuar o combate ou regressar ao menu principal. No final do combate o jogador irá regressar ao menu principal.

Caso o jogador escolha o menu Multijogador, o processo será muito idêntico a quando o jogador seleciona a opção Jogar, apenas com a diferença de que neste modo o jogador irá defrontar, não o computador, mas sim outro jogador.

Caso o jogador escolha o menu Opções, poderá interagir com diversas preferências do videojogo, como a resolução, tempo, música, sons e configurar controlos. Ao aceder a esta última opção, será apresentado ao jogador os controlos por omissão, podendo o mesmo personalizar todos os controlos para aqueles que mais lhe interessarem.

Caso o jogador escolha o menu Créditos, irá ser apresentada a pessoa envolvida no desenvolvimento deste videojogo e terá de regressar ao menu principal.

Todos os menus apresentam a opção de regressar ao menu anterior, exceto quando o jogador se encontra num combate. Nesse caso terá de regressar ao menu principal do videojogo.

Referências

[Street Fighter, 2014]
Street Fighter, "Street Fighter", http://www.streetfighter.com, [Último acesso: 19 de Setembro de 2014]

[Mortal Kombat, 2014]
Mortal Kombat, "Mortal Kombat", http://www.mortalkombat.com, [Último acesso: 19 de Setembro de 2014]

[Música do Videojogo, 2011]
Street Fighter II Ken Theme Original, https://www.youtube.com/watch?v=-14W5XTqL5U, [Último acesso: 30 de Setembro de 2014]

[Som Voltar, 2008]
sf3-sfx-menu-back.wav, https://www.freesound.org/people/broumbroum/sounds/50557/, [Último acesso: 30 de Setembro de 2014]

[Som Selecionar, 2008]
sf3-sfx-menu-select.wav, https://www.freesound.org/people/broumbroum/sounds/50561/, [Último acesso: 30 de Setembro de 2014]

[Som Validar, 2008]
sf3-sfx-menu-validate.wav, https://www.freesound.org/people/broumbroum/sounds/50565/, [Último acesso: 30 de Setembro de 2014]

[Personagem Chefe Kab, 2009]
Zangief, http://spritedatabase.net/file/203, [Último acesso: 30 de Setembro de 2014]

[Personagem Mukiko, 2009]
Ryu, http://spritedatabase.net/file/202, [Último acesso: 30 de Setembro de 2014]

[Cenários, 2013]
50 Animated Gifs of Fighting Game Backgrounds, http://twistedsifter.com/2013/05/animated-gifs-of-fighting-game-backgrounds/, [Último acesso: 30 de Setembro de 2014]

[Estilo de letra – Titillium Web, 2009]
Google Fonts, https://www.google.com/fonts/specimen/Titillium+Web, [Último acesso: 30 de Setembro de 2014]

[Bandeira Portuguesa, 2012]
Portugal-bandeira, http://santotirsoonline.com/parabens-portugal/portugal-bandeira/, [Último acesso: 30 de Setembro de 2014]

[Bandeira Inglesa, 2014]
Wallpaper londrês Bandeira para Desktop, http://wallpaper-imagens.org/2014/09/wallpaper-londres-bandeira-para-desktop/, [Último acesso: 30 de Setembro de 2014]

[Imagens utilizadas na história]
http://goo.gl/C1OW9y, http://goo.gl/7Gbkp0, http://goo.gl/wu4sci, http://goo.gl/VopGkB, http://goo.gl/GgZzMX, http://goo.gl/nug2rO , [Último acesso: 19 de Outubro de 2014]