Instituto Superior de Engenharia do Porto

Criação de uma Framework Modular

Vocacionada para o Desenvolvimento de Videojogos

Gil Manuel Rabaça Pinto Canizes

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática, Área de Especialização em
Sistemas Gráficos e Multimédia

Orientador: Prof. Doutor João Paulo Pereira

Júri:

Presidente:

Prof. Doutora Maria De Fátima Coutinho Rodrigues, ISEP

Vogais:

Prof. Doutor João Paulo Jorge Pereira, ISEP

Prof. Doutor Paulo Alexandre Gandra De Sousa, ISEP

Porto, Outubro 2014

«A quem me apoiou»

Resumo

Os videojogos são cada vez mais uma das maiores áreas da indústria de entretenimento, tendo esta vindo a expandir-se de ano para ano. Para além disso, os videojogos estão cada vez mais presentes no nosso dia-a-dia, quer através dos dispositivos móveis ou das novas consolas. Com base nesta premissa, é seguro de afirmar que o investimento neste campo trará mais ganhos do que perdas.

Esta Dissertação tem como objetivo o estudo do estado da indústria dos videojogos, tendo como principal foco a conceção de um videojogo, a partir duma *Framework* Modular, desenvolvida também no âmbito desta Dissertação.

Para isso, é feito um estudo sobre o estado da arte tecnológico, onde várias ferramentas de criação de videojogos foram estudadas e analisadas, de forma a perceber as forças e fraquezas de cada uma, e um estudo sobre a arte do negócio, ficando assim com uma ideia mais concreta dos vários pontos necessários para a criação de um videojogo.

De seguida são discutidos os diferentes géneros de videojogos existentes e é conceptualizado um pequeno videojogo, tendo ainda em conta os diferentes tipos de interfaces que são mais utilizados na indústria dos videojogos, de forma a entender qual será a forma mais viável, conforme o género, e as diferentes mecânicas presentes no videojogo a criar.

A *Framework* Modular é desenvolvida tendo em conta toda a análise previamente realizada, e o videojogo conceptualizado. Esta tem como grande objetivo uma elevada personalização e manutenibilidade, sendo que todos os módulos implementados podem ser substituídos por outros sem criar conflitos entre si.

Finalmente, de forma a unir todos os temas analisados ao longo desta Dissertação, é ainda desenvolvido um Protótipo de forma a comprovar o bom funcionamento da *Framework*, aplicando todas as decisões previamente feitas.

Palavras-chave: *Framework* Modular, Videojogos, Interfaces para Videojogos, *Game Design*

Abstract

Video games are now one of the most profitable areas in the entertainment industry, and it's been growing every year. Aside from that, video games are more present now in our daily life, by playing on our mobile devices, or on our brand new consoles, than they were ever before. With that in mind, it's safe to say that any investment in this field will be more rewarding, than it will be unfruitful.

The main objective of this Dissertation is to study the state of the video game industry, having it's main focus in the implementation of a video game, supported by a Modular Framework, also developed for this Dissertation.

For this, a study is made on the technological status, where various tools used in the creation of video games were analysed, as a way to understand the multiple weaknesses and strengths in each one of them, and by studying the business status, as a way of developing a more concrete idea about the different themes involved in the process of creating a video game.

Then the different genres of video games are discussed, and a little video game is conceptualized. This video game is also made in accordance with all the different types of interfaces that are used in the video game industry, so that we can understand what is the best type of interface that we can apply to the video game, in conformity with its genre and mechanics.

The Modular Framework was developed in accordance with all the previous analysis and the video game that was conceptualized. This Framework must be highly customizable and easy to maintain, so that all the different implemented modules might be changed without creating any kind of conflicts.

Finally, a Prototype was developed as a way to connect all the previous topics presented in this Dissertation, as well to demonstrate the potential and performance of the Modular Framework.

Keywords: Modular Framework, Video Games, Video Game Interfaces, Game Design

Agradecimentos

Gostaria de começar por agradecer aos meus pais, por todo o apoio, e pela confiança que sempre depositaram em mim durante toda a minha vida.

A toda a minha família pela preocupação demonstrada ao longo de todo o tempo de desenvolvimento desta Dissertação.

Por fim, mas não menos importante, ao meu orientador, Professor Doutor João Paulo Pereira por toda a disponibilidade demonstrada e pelos valiosos conselhos e contributos no desenvolvimento desta Dissertação.

Índice

Índice de Figuras

Índice de Tabelas

Índice de Gráficos

Acrónimos e Símbolos

Lista de Acrónimos

API	*Application Programming Interface*
CD	*Compact Disc*
CLI	*Command-Line Interface*
CPU	*Central Processing Unit*
DRM	*Digital Rights Management*
DVD	*Digital Versatile Disc*
FPS	*First Person Shooter*
FPS	*Frames Per Second*
GDD	*Game Design Document*
GML	*Game Maker Language*
GUI	*Graphical User Interface*
HDRR	*High Dynamic Range Rendering*
IDE	*Integrated Development Environment*
MMORPG	*Massive Multiplayer Online Role Playing Game*
NPC	*Non-Playable Character*
OO	*Orientada a Objetos*
PC	*Personal Computer*
ROB	*Robotic Operating Buddy*
RPG	*Role Playing Game*
SO	*Sistema Operativo*
UDK	*Unreal Development Kit*
UI	*User Interface*
WYSIWYG	*What You See Is What You Get*

1 Introdução

Nos dias que correm os videojogos estão cada vez mais presentes no nosso dia-a-dia. Pode-se até dizer que, devido ao elevado número de videojogos que aparecem semanalmente no mercado, quer completos e prontos a jogar, quer uma ideia à espera de ser financiada pelo público, existem videojogos para todos os gostos e feitios. Estes, também devido ao elevado número de *software* de desenvolvimento disponibilizado no mercado, muitas das vezes suportam mais do que uma plataforma, de forma a acomodar um mercado alvo maior.

Para além de todos estes aspetos, os videojogos são hoje uma das maiores fontes de rendimento dentro da área do entretenimento, ultrapassando já há alguns anos a indústria dos filmes e da música, tendo ainda uma perspetiva de crescimento bastante elevada, de $67 mil milhões em 2013 para $82 mil milhões em 2017 [1].

Muitas das empresas conseguem obter receitas elevadas ao venderem videojogos devido à facilidade com que estes conseguem ser comercializados nos dias de hoje através de lojas *online* como Steam [2], Desura [3], entre outras, já com um elevado número de visitas diárias, e também utilizando as redes sociais, como o Facebook [4] ou o Twitter [5], de forma a aumentarem a visibilidade que o videojogo tem no mercado. Estas facilidades devem-se muito à evolução tecnológica com que nos deparamos nos dias de hoje, tornando assim possível colocar o nosso produto ao alcance de comunidades globais de jogadores de todas as idades e sexos.

1.1 Enquadramento da Dissertação

O trabalho aqui apresentado incide sobre as várias fases necessárias para a criação de um videojogo, pelo ponto de vista de um pequeno estúdio independente.

Para a criação de um videojogo, muitos aspetos têm de ser tidos em conta, tais como: o estilo da Arte e os tipos de Sons, a Música, a História, os Objetivos, entre outros, a serem utilizados nos videojogos, pois sem a ajuda, ou até mesmo a má escolha, destes elementos

torna-se complicado envolver o jogador no mundo criado para o videojogo, pois este pode questionar se certas ações fazem sentido tendo o seu ponto de vista do mundo real em conta [6], ou até mesmo passar uma mensagem, subliminar ou não, errada ao jogador [7]. Todos estes aspetos devem estar presentes num documento denominado GDD (*Game Design Document*). Este documento será abordado e explicado num capítulo posterior desta Dissertação.

Apesar de virem a ser referenciadas as várias áreas necessárias para a criação de um videojogo, tal como o género do mesmo, a escolha da interface, etc., esta dissertação terá como seu foco principal a criação duma *Framework* Modular, pois para tratar de todas as fases com a devida atenção, seria preciso muito mais tempo do que o disponível para a escrita desta dissertação.

Esta *Framework* Modular será feita com o género, a forma de jogar e as plataformas suportadas pelo videojogo selecionado em mente. Desta forma pretende-se conseguir obter o melhor produto final possível. Tentar-se-á evitar alguns dos problemas existentes na construção de videojogos a partir de *Frameworks* e/ou motores de videojogo existentes, tais como problemas no código generalista de alguns motores de videojogo e *Frameworks*, ou a falta de uma caraterística necessária para o videojogo. Também se pretende, desta forma, evitar a aquisição de licenças para o uso dos mesmos, coisa que para um pequeno estúdio independente pode fazer uma grande diferença.

Para além deste foco na construção da *Framework* Modular, será ainda elaborado um *Game Design Document* (GDD) para o videojogo (Protótipo) a ser criado, e serão ainda feitos alguns estudos sobre os diferentes tipos de Interfaces existentes, tentando assim, escolher a mais indicada para o videojogo.

1.2 Organização do Relatório

Esta dissertação encontra-se dividida em sete capítulos. O primeiro capítulo trata da Introdução. Pretende-se que o leitor perceba duma forma rápida e simples do que irá tratar a dissertação e que temas serão debatidos.

Numa segunda fase será analisado o contexto em que esta dissertação se insere, começando por falar do estado da arte do negócio, isto é, avaliando duma forma global os videojogos, as plataformas suportadas pelos videojogos e realizando uma breve análise do mercado dos videojogos. Posteriormente serão enunciados alguns aspetos existentes, tanto positivos como negativos, dos vários tipos de ferramentas para a criação de videojogos existentes no mercado. Segue-se uma análise mais profunda a uma ferramenta específica de cada tipo. Para terminar este capítulo serão ainda abordadas as tecnologias que foram utilizadas ao longo deste projeto.

No terceiro capítulo será feita a Análise e Design do videojogo a desenvolver. Para isso será inicialmente feito o desdobramento do videojogo, escolhendo o género do mesmo, a

plataforma que será suportada e ainda a arquitetura da aplicação. Neste capítulo ainda será feito o levantamento dos requisitos e a metodologia de desenvolvimento escolhida para o processo de desenvolvimento do videojogo.

O capítulo seguinte dedica-se inteiramente à Interface Gráfica. Esta deverá existir com o principal objetivo de possibilitar uma fácil comunicação entre o videojogo e o jogador. Aqui serão analisadas várias técnicas de construção de interfaces, percebendo quais os defeitos e qualidades de cada uma. Desta forma pretende-se conseguir chegar a uma interface para o videojogo com o menor número de defeitos possível.

No capítulo cinco serão expostos alguns dos pontos mais interessantes da implementação da solução pretendida, ou seja, da *Framework* criada para este projeto.

No penúltimo capítulo serão enumerados alguns dos pontos mais importantes da conceção do videojogo (protótipo), de modo a demonstrar as capacidades da *Framework* desenvolvida. Para além de demonstrar as capacidades da *Framework*, o videojogo também tem o papel de demonstrar que a *Framework* realmente funciona, e que os resultados pretendidos para este projeto foram alcançados.

Finalmente, no último capítulo serão apresentadas, de forma concisa, as conclusões sobre todo o trabalho realizado durante o tempo desta dissertação, de acordo com os objetivos iniciais. São ainda referidas algumas limitações e propostas de trabalho futuro.

2 Contexto

Neste capítulo serão abordadas todas as fases importantes para a conceção e desenvolvimento de um videojogo.

Inicialmente será feita uma apresentação do estado da arte do negócio, de forma a demonstrar o tipo de indústria em que este se insere. De seguida será feita uma abordagem a alguns dos temas principais sobre o estado da arte tecnológico. Finalmente, serão abordadas as tecnologias que foram utilizadas no decorrer do desenvolvimento deste trabalho.

2.1 Estado da arte do negócio

Como já abordado na Introdução, os videojogos são hoje uma das maiores fontes de rendimento na área do entretenimento. Para demonstrar bem a evolução comercial que os videojogos têm tido ao longo do tempo em relação as outras áreas de entretenimento, é apresentado abaixo o Gráfico 1 que demonstra o total arrecadado pelo filme "The Avengers", filme com o recorde de maior receita no primeiro fim-de-semana (três dias) [8], em comparação com as vendas do primeiro dia do videojogo "Grand Theft Auto V", o videojogo mais vendido globalmente [9].

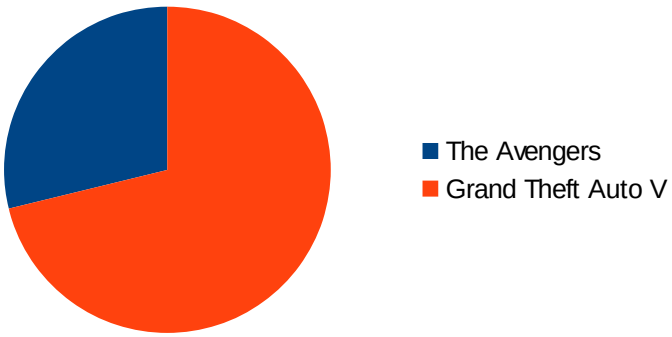


Gráfico 1 - Comparação de vendas entre dois produtos de entretenimento

A diferença entre os dois produtos é abismal. Com este exemplo pode-se perceber duma forma simples que a indústria dos videojogos está numa posição vantajosa em comparação com as outras indústrias da mesma área.

2.1.1 Videojogos

Os videojogos são uma das maiores peças que definem a indústria do entretenimento nos dias que correm. Estes podem ser divididos de duas formas, pelo género do videojogo, ou pelo dinheiro investido durante todo o processo de vida do mesmo.

Se se quiser agrupar os videojogos por género consegue-se encontrar imensas categorias e subcategorias de videojogos, sendo que algumas das mais conhecidas são [10]: Desporto, Simuladores, *First Person Shooter* (FPS), Aventura, Ação, Plataformas, *Role Playing Game* (RPG) e Estratégia.

Hoje em dia os videojogos tendem a juntar vários géneros diferentes, de forma a criar um produto final mais completo, ou até dão origem a um novo género de videojogo, tal como sucedeu com os videojogos do género *God Game* (Jogo Deus, traduzido à letra), onde o jogador tem o papel de um Deus. Com isto não se pode dizer que se conhece todos os géneros de videojogos, pois no futuro próximo poderá aparecer um novo género.

Se for pelo dinheiro investido durante toda a vida do videojogo pode-se encontrar dois grandes tipos: os videojogos AAA (Triple A), ou os videojogos Indie (ou independentes).

Os videojogos do tipo AAA [11] são normalmente os videojogos desenvolvidos por grandes estúdios onde o dinheiro investido, por parte de distribuidoras e/ou investidores, ultrapassa grande parte das vezes os milhões de dólares. Para estes videojogos são sempre esperadas receitas bastante maiores do que o dinheiro investido.

Os videojogos Indie [12] acabam por ser o oposto dos AAA, pois não têm qualquer suporte das distribuidoras e/ou investidores. Os videojogos deste tipo são muitas das vezes criados por uma só pessoa, ou por uma pequena equipa, e o investimento é muito menor do que num videojogo AAA, devido aos riscos que os criadores têm de tomar.

Apesar de todas estas diferenças, ao contrário do que se possa pensar, muitas das recentes inovações feitas neste campo provêm de videojogos Indie. Pois apesar de não terem o dinheiro, nem o número de colaboradores de um videojogo AAA, tentam sempre dar o seu melhor, sem terem qualquer das restrições impostas pelos investidores e/ou distribuidoras presentes nos videojogos AAA, e desta forma criar algo inovador com fim a ganharem algum reconhecimento e destaque no mercado.

2.1.2 Plataformas

Um videojogo, só por si, não pode ser jogado, pois este não é mais do que *software* e para que o jogador possa tirar proveito do mesmo terá de possuir uma componente de *hardware*, muitas vezes denominada por Plataforma.

Estas plataformas existem já há imensos anos, sendo que uma das primeiras plataformas domésticas de videojogos foi a Magnavox Odyssey [13], lançada em 1972 nos Estados Unidos da Améria. Desde essa altura que nunca mais pararam de sair novas plataformas de videojogos, tais como: a Atari, a Sega Genesis, o Gameboy, a PlayStation, a XBOX, entre outras.

Ainda hoje em dia são feitas novas plataformas, como no caso da OUYA [14], uma consola lançada em Junho de 2013. Ou novas iterações de consolas já existentes, caso da PlayStation 4 [15] lançada em Novembro de 2013, e que já conta com 5.3 Milhões de unidades vendidas[1], e da XBOX One [16] também lançada em Novembro de 2013. Pode-se ainda dizer que existe outro tipo de plataforma devido ao Computador Pessoal (PC), pois este, em vez de ter novas iterações, é melhorado conforme as necessidades de cada jogador, podendo estes optar por mudar todo o PC ou alterar parcialmente o mesmo, como por exemplo modificar unicamente a Placa Gráfica ou a *Unidade Central de Processamento*(CPU).

As plataformas suportadas pelos videojogos podem variar de uns para os outros, sendo que alguns destes suportam várias plataformas. Chama-se a este tipo de videojogos “Cross-Platform” [17] (ex.: Fallout 3 e Assassin's Creed). Noutros casos, os videojogos são dependentes da plataforma, denominados de “Platform-Exclusives” [18] (ex.: God of War 3 e Uncharted).

Para além de existirem imensas plataformas, como referido acima, para cada uma podem ainda existir diferentes Sistemas Operativos (SO), como por exemplo no caso do PC, onde o código nativo do videojogo terá de ser diferente de SO para SO; isto caso a linguagem usada para o desenvolvimento do mesmo seja suportada em todos os sistemas operativos. Caso contrário, o criador do videojogo terá de refazer todo o videojogo com uma linguagem suportada pelo SO para o qual pretende exportar o videojogo.

Pode-se com isto concluir que, com o elevado leque de plataformas existentes, não faltam locais onde os criadores de videojogos possam por as suas criações a funcionar.

2.1.3 Análise do Mercado de Distribuição

Depois de escolhida a plataforma/plataformas para o videojogo e realizados todos os processos envolventes no desenvolvimento do mesmo, o próximo passo é arranjar maneira de distribuir o videojogo pelos possíveis compradores. Pode-se dizer que a distribuição, ao lado do *marketing*, é uma das fases finais no longo percurso da criação de um videojogo.

1 Números publicados a: 18 e Fevereiro de 2014

Hoje em dia existem duas grandes formas de distribuição de um videojogo, sendo estas: *Retail* e Digital.

O processo de *Retail* [19] envolve pôr o videojogo num formato físico (*Compact Disc* (CD), *Digital Versatile Disc* (DVD), Blu-Ray, etc.), e pôr o mesmo à venda em lojas espalhadas pela área onde se pretende vender o videojogo. Neste caso, o cliente deverá dirigir-se a uma loja que venda o videojogo que pretende e comprar o mesmo, trazendo desta forma para casa uma cópia física do videojogo pretendido. Como é de esperar este processo pode envolver imensos custos para os criadores do videojogo, custos estes que para empresas mais pequenas, como as independentes, podem fazer a diferença entre iniciar o desenvolvimento ou não um videojogo. É muito graças a este problema, que muitos dos novos criadores de videojogos tendem a vender os seus produtos exclusivamente pelo processo de distribuição Digital.

O processo Digital [20] começou a tornar-se importante há poucos anos graças ao desenvolvimento tecnológico, sendo a velocidade da internet uma das maiores causas. De certa maneira, este processo veio combater alguns dos problemas, um deles já tendo sido mencionado no parágrafo anterior, que se encontravam no processo de distribuição de *Retail*. Este processo não só trouxe regalias para os criadores de videojogos, mas também para os compradores, como por exemplo: acabar com a falta de *stocks* nas lojas, preços mais competitivos, acesso ao videojogo a partir de qualquer lugar, entre outros.

O que acontece é que neste processo o videojogo em vez de ser posto num formato físico, encontra-se num formato digital alojado num (ou vários) servidor. O cliente, quando compra o videojogo, só terá de o descarregar e instalá-lo para começar a tirar proveito da experiência que este tem para oferecer[2].

Apesar de todos estes pontos a favor da distribuição Digital, esta também possui diversas falhas. Sendo o tempo para descarregar o videojogo uma delas, com os videojogos cada vez a ficarem maiores, isto torna-se um problema para jogadores com internet mais lenta (desde já o jogador é obrigado a possuir uma ligação à internet). Com esta forma de distribuição deixa também de existir uma opção mais barata para os jogadores que estão habituados a comprar videojogos usados na distribuição digital. E um dos maiores problemas é o facto de o jogador estar dependente da loja *online* onde compra os seus videojogos, pois, ao contrário do processo *Retail*, onde este possui sempre a cópia física do videojogo, no processo Digital, se a loja/*website* onde o jogador comprou o videojogo fecha, este poderá nunca mais conseguir obter uma cópia do seu videojogo.

Com esta breve análise pode-se concluir que, apesar do mercado Digital trazer imensas oportunidades tanto para os criadores de videojogos como para os jogadores, esta forma de distribuição não é perfeita e ainda existe um longo caminho a percorrer para que esta se possa tornar no processo de distribuição mais comum [21].

2 Nesta dissertação não serão abordados os vários processos de *Digital Rights Management* (DRM) aplicados pelas diferentes empresas, com vista a evitar cópias ilegais do videojogo, pois isso iria muito mais além do que a ideia que se pretende transmitir sobre a distribuição Digital

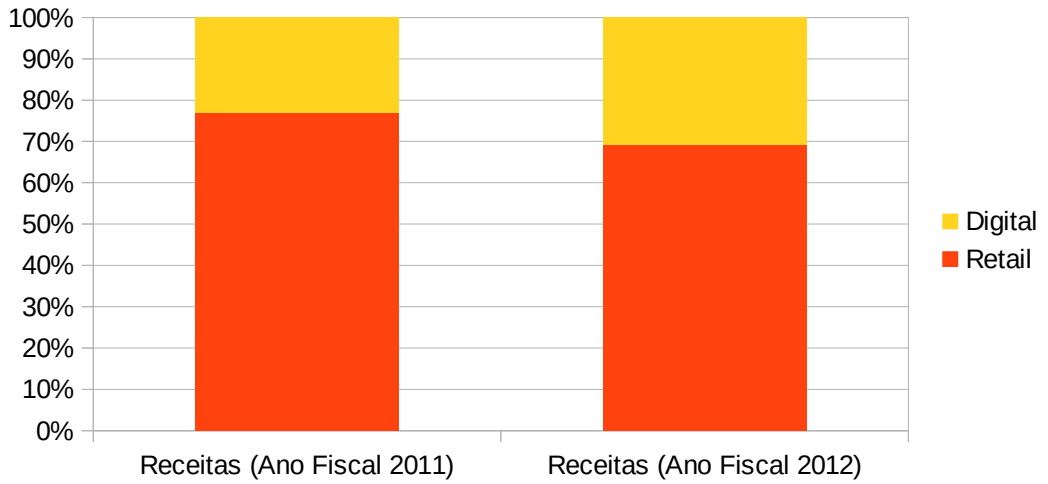


Gráfico 2 - Comparação das receitas nos diferentes tipos de distribuição da empresa Eletronic Arts.

Contudo, como se pode ver no Gráfico 2 [22], este processo de distribuição tem vindo a crescer aos poucos de ano para ano, roubando mercado ao processo de *Retail*, sendo que os *smartphones* (Android e iOS), as plataformas de distribuição como a Steam e o Desura e os criadores independentes de videojogos (por exemplo: a Mojang, criadores do videojogo Minecraft inicialmente vendido exclusivamente no formato Digital) têm ajudado imenso esta recente forma de distribuição.

2.2 Estado da arte tecnológico

Entrando agora na vertente mais tecnológica desta dissertação, esta secção servirá para apresentar alguns conceitos-chave que serão extremamente úteis nos capítulos subsequentes.

Neste subcapítulo será feita uma abordagem aos diferentes tipos de controladores para videojogos existentes no mercado, pois o tipo de controlador escolhido para este projeto irá afetar não só o estilo do videojogo, mas também a interface do mesmo e a forma de interação que o jogador tem com o videojogo.

Note-se ainda que, nesta dissertação, não serão tomadas em conta soluções que envolvam as tecnologias de *motion controllers*, como o Kinect [23] ou até mesmo o PSMove [24], nem qualquer tipo de dispositivos de realidade aumentada, como o Oculus Rift [25]. Apesar da utilização destas tecnologias ter subido exponencialmente nos últimos anos [26], para que este projeto suportasse as mesmas devidamente, iria ser necessária uma análise muito mais profunda do que é pretendido no que toca à escolha dos controladores a serem utilizados e na criação de todos os aspetos do videojogo.

Nesta secção também serão abordadas as várias formas de desenvolver um videojogo, do ponto de vista da programação, não sendo dado muita relevância a criação dos *Assets* para o videojogo. Tendo isto em conta pode-se dizer que existem quatro grandes formas de implementar um videojogo:

a) Recorrer à utilização de uma *Framework*. Sobre este tema será feita uma breve descrição do que se entende por *Framework* relativamente à implementação de um videojogo, e de forma a reforçar esta definição será feita uma breve análise à *Framework* "XNA Framework" [27].

b) Utilizar um Motor de Jogo. Este tema seguirá as mesmas linhas de análise do tema anterior sobre *Frameworks.* O Motor de Jogo analisado será o "*Unreal Development Kit* (UDK) - Unreal Engine" [28].

c) Empregar o uso de um *Game Maker*. Para este tema será, mais uma vez, também feita uma análise como nos temas anteriores, mas com foco nos *Game Makers* e o *Game Maker* a analisar será o "GameMaker: Studio" [29]

d) Finalmente, pode-se ainda implementar todo o código necessário para o videojogo, sem recorrer a nenhuma das tecnologias anteriormente mencionadas, usando unicamente bibliotecas existentes para o desenvolvimento de videojogos. A biblioteca a ser analisada será a "Bullet Physics Library" [30].

2.2.1 Controladores de videojogos

Hoje em dia existem tantos controladores, que quase que se pode dizer que existem controladores para todos os gostos. Não só existem diferentes tipos de controladores, como também um número abundante de variáveis dos mesmos.

Alguns destes controladores conseguem ser muito inovadores na altura em que são criados, e assim vingar no mercado, como é o caso de *Gamepad* [31] que, apesar já existir em 1977 [32], ainda hoje, apesar de modificado, existe no mercado como uma das formas mais comuns de jogar videojogos. Noutros casos, os controladores passam despercebidos pelos olhos do público em geral e são esquecidos com o tempo, como no caso do *Robotic Operating Buddy* (ROB) [33].

Os controladores mais utilizados[3], para jogar videojogos atualmente, não tendo em conta o género do videojogo, são: o *Gamepad,* Teclado e Rato e *Touch Screen* (Figura 1), sendo que este último só recentemente começou a ter um uso mais regular muito devido aos *Smartphones* e às consolas portáteis da Nintendo. Estes três controladores serão tidos em conta quando chegar o momento de escolher quais serão suportados neste projeto, mas para que o leitor se sinta a vontade quando estes forem discutidos, será feito em seguida um pequeno resumo de cada um.

Figura 1 - Controladores (*Gamepad*; Teclado e Rato; *Touch Screen*)

- *Gamepad* – Este controlador, também conhecido como *Joypad*, é um tipo de controlador onde o jogador deve segurá-lo com as duas mãos, onde os dedos são usados para premir os botões de forma a fornecer *input*. Este é o controlador padrão de grande parte das consolas existentes no mercado e muitas vezes utilizado no PC.

- Teclado e Rato – Estes são os controladores tipicamente usados em qualquer computador. Neste caso o jogador deverá usar as mãos para clicar no Rato e premir teclas do Teclado de forma a gerar *input.* É de notar que não é obrigatório usar-se estes dois controladores, pois são independentes um do outro, mas grande parte das vezes os videojogos utilizam estes em conjunto. Esta combinação de controladores é hoje em dia a mais usada nos videojogos para PC.

- *Touch Screen* – Neste caso, o ecrã onde é visualizado o videojogo é também o controlador. Desta forma, tocando no ecrã com o(s) dedo(s), ou com um estilete especial, fornece-se *input* ao videojogo. Este controlador é o mais utilizado nos *Smartphones* e em consolas portáteis, como a Nintendo DS.

Numa nota final, nesta dissertação os controladores são analisados pelos padrões, e não é sequer dada qualquer importância às variáveis extremas desses controladores, apesar de estas existirem no mercado [34].

3 Baseado nos controladores utilizados pelos vários criadores de videojogos, tais como: Sony, Nintendo, Microsoft, Halfbrick, Square Enix, Blizzard, entre outros.

2.2.2 Framework vs Motor de Jogo vs Game Maker vs Game Development Library

Uma das maiores questões que se deve fazer quando se está a desenvolver um videojogo é: Que ferramenta usar para o desenvolvimento do videojogo?

Esta questão não tem uma resposta certa mas sim muitas, pois dependendo do estilo/género do videojogo, do tempo, dos recursos financeiros, do número de pessoal e do nível de competências que temos disponíveis para o desenvolvimento do videojogo, a solução pretendida pode variar.

Antes de serem feitas definições para cada uma das diferentes ferramentas de desenvolvimento deve-se dizer que nem toda a gente acredita que exista uma diferença entre *Framework* e Motor de jogo [35]. Mas para esta dissertação estes dois serão diferenciados, de forma a facilitar a discussão entre os mesmos e para que se perceba bem os objetivos pretendidos para este projeto.

As definições abaixo baseiam-se principalmente nestas referências [36,37].

Framework: Uma *Framework* não é mais que um "esqueleto", formado por interfaces ou classes abstratas, para que o programador saiba o que é necessário fazer para o desenvolvimento do videojogo. Os componentes deste esqueleto são grande parte das vezes "colados" por pequenos fragmentos de código de forma a unir todos estes componentes. Para além disso, uma *Framework* também pode conter pequenos métodos que facilitam a implementação de partes da *Framework* que serão normalmente necessárias para o desenvolvimento do videojogo facilitando a vida do programador. Pode-se dizer que, neste caso, o programador, apesar de ter algumas barreiras, estas acabam por não influenciar muito o desenvolvimento do videojogo, pois o programador acaba por ter uma grande liberdade para fazer o que pensa ser melhor para o seu videojogo. Note-se ainda que algumas vezes as F*rameworks* já trazem alguns componentes implementados e não costumam possuir qualquer interface gráfica com o utilizador.

Motor de Jogo: Um Motor de jogo acaba também por ter uma *Framework* integrada, mas em vez de deixar o programador criar todos os métodos necessários para o bom funcionamento do videojogo, estes já vêm implementados de raiz. Com isto não se quer dizer que um Motor de Jogo tenha tudo que é necessário para a criação de qualquer videojogo, pois os criadores do Motor de Jogo é que escolhem que partes este deve possuir (como por exemplo: processamento de mensagens por Rede, simuladores de Física, etc.). Apesar disso, muitos dos Motores de Jogo existentes conseguem suportar a maior parte dos videojogos existentes, quer estes sejam 2D, 3D, *offline* ou *online*. Devido a toda esta implementação já realizada por parte dos criadores do Motor de Jogo, este muitas vezes pode ser muito complexo para o videojogo que se pretende criar, fazendo com que os programadores percam mais tempo a perceber como o Motor de Jogo funciona do que a implementar o seu videojogo. Estes normalmente possuem uma interface gráfica que faz interface com o

utilizador, para que alguns dos processos a realizar pelos criadores do videojogo sejam mais fáceis e intuitivos.

Game Maker: Um Game Maker é a escolha acertada para o caso do criador do videojogo não ser um programador, ou este não possuir qualquer ideia sobre o processo de criação de um videojogo. Apesar destes não possuírem um poder tão forte como um Motor de Jogo/*Framework*, quer-se com isto dizer que a grande maioria dos videojogos criados com estas ferramentas possuem restrições impostas pelo Game Maker, são muito fáceis de utilizar e requerem pouco, ou até mesmo nenhum conhecimento da área da Programação. Pode-se dizer que um Game Maker acaba por fazer todo o trabalho "pesado" na criação de um videojogo, como o processamento das texturas, processamento do *input* por parte do jogador, físicas do videojogo, entre outras, dando a possibilidade ao criador de se focar no videojogo em si, abstraindo-se dos outros componentes necessários. Alguns dos *Game Makers* existentes ainda possuem um editor de *scripts*, para que de alguma forma consigam combater algumas das restrições impostas pelo *software,* quando usado por um programador experiente. Para rematar, muitas das vezes, estes *Game Makers* só suportam um género de videojogos, como no caso do rpgmaker [38].

Game Development Library: Finalmente, uma *Game Development Library* não é mais que um conjunto de métodos que facilitam o programador no processo da criação de um videojogo. Existem *Game Development Libraries* para diversas funções, como por exemplo: processamentos gráficos, processamento de som, processamento de forças físicas, etc. Desta forma, o programador pode reunir as bibliotecas que acha necessárias para a implementação do seu videojogo e assim ter um maior controlo sobre tudo que se passa. Neste caso também não existe qualquer tipo de interface gráfica.

Como se pode ver pelas definições acima, não existe uma ferramenta certa para o desenvolvimento de videojogos, mas sim várias. Agora que o leitor já se encontra familiarizado com as definições, serão feitas análises a algumas das ferramentas disponíveis no mercado, para que no fim se possa tirar algumas conclusões, discutindo de uma forma mais clara as forças e fraquezas de cada um destes componentes.

2.2.2.1 XNA Framework

A XNA *Framework* pertence à Microsoft e consiste num conjunto de ferramentas de programação [39]. Esta *Framework* consiste num conjunto de Bibliotecas e *Application Programming Interfaces* (API) prontas a serem utilizadas pelos programadores, de forma a reduzir o código necessário para correr rotinas comuns num videojogo.

Os grandes objetivos desta *Framework* são [40]:

- Fazer com que o foco dos utilizadores desta *Framework*, quer programadores experientes, quer utilizadores amadores, seja voltado para o conteúdo do seu videojogo, sem ter de perder um período significativo de tempo a codificar as rotinas básicas de um videojogo.

- Possibilitar a portabilidade do videojogo para mais que uma plataforma, isto é, fazer com que qualquer videojogo criado nesta F*ramework* possa ser jogado quer em Windows, quer numa Xbox 360, sem que o criador tenha de fazer praticamente alguma alteração no código.

- Fazer com que um criador de videojogos que utilize esta *Framework* consiga começar a escrever o seu videojogo em menos de 5 minutos. Isto é, o programador não terá de se preocupar logo no início com imensas coisas vitais para o funcionamento de qualquer videojogo (ex: *graphic adapters, events*), visto estas já estarem a ser processadas no *background*.

De forma a demonstrar como esta *Framework* se comportou, tendo em conta os objetivos a que se propôs, no caso de um programador desejar criar a janela onde será apresentado o seu videojogo, nesta *Framework* ele só necessitaria de criar um novo projeto no IDE e escolher a opção Windows Game (2.0), demonstrada na Figura 2 (o 2.0 representa a versão da XNA Framework instalada). Depois do projeto criado teria unicamente que correr o mesmo e este iria mostrar uma janela onde está a ser feito o *render* de um sólido azul. Desta forma, o programador não precisou de implementar qualquer tipo de rotinas para que o videojogo funcionasse, pois essa parte já vem implementada na própria *Framework*.

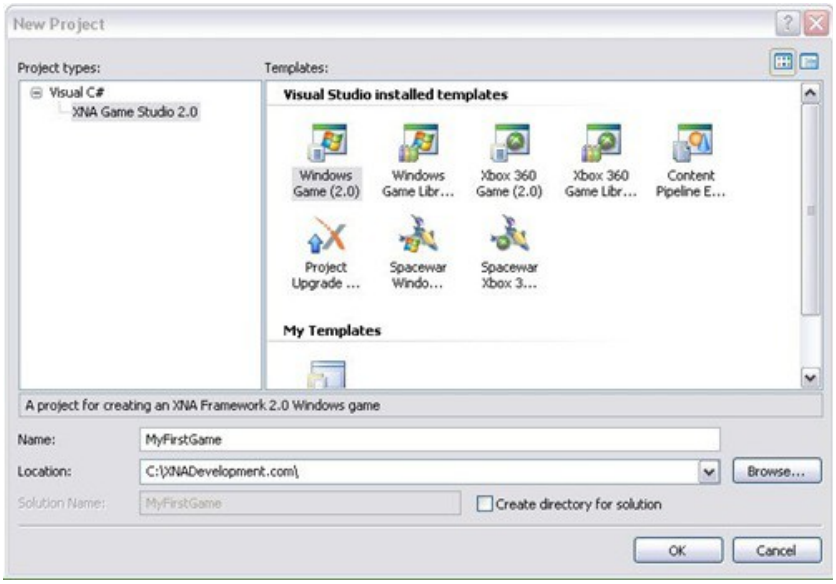


Figura 2 - Janela para criação de um novo projeto no Visual Studio.

Agora o programador só teria de começar a apresentar objetos/texturas para o videojogo, e implementar a lógica do mesmo, para conseguir ter algo muito simples mas funcional com um número muito pequeno de linhas de código. Pode-se dizer que a "regra dos 5 minutos" foi realizada com sucesso.

Deve-se ainda perceber que esta *Framework* só possibilita este tipo de processos graças a uma camada denominada de *Extended Framework*, pois normalmente uma *Framework* "normal" não faria isto. O que acontece neste caso é que, para além da *Core Framework,* esta *Framework* possui uma camada que tem como objetivo tornar o desenvolvimento do

videojogo mais fácil, aplicando vários processos que ajudam o programador na criação do mesmo.

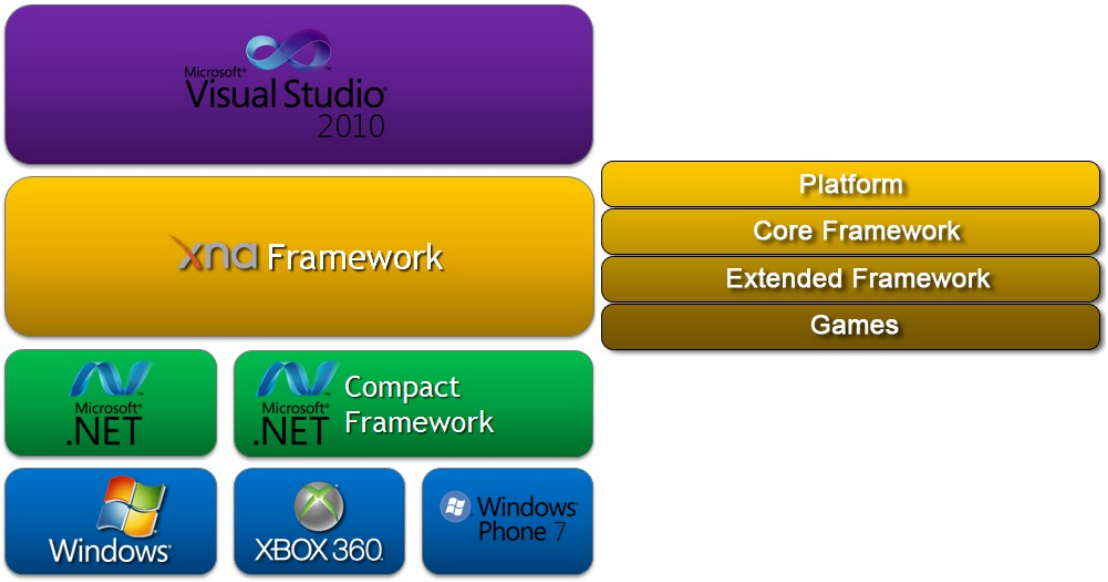


Figura 3 - Representação das diferentes camadas da XNA Framework.

Para além das duas camadas acima mencionadas, esta *Framework* ainda contém a camada *Platform*, sendo esta a mais baixa e que contém APIs como o *XACT e Xinpu,* e a camada *Games,* que consiste no código gerado pelo programador do videojogo.

De forma que se perceba facilmente as vantagens e desvantagens encontradas no uso desta *Framework*, estas serão listadas a seguir:

Vantagens:

- A linguagem utilizada é .NET. Não é uma linguagem difícil de utilizar e existem bastantes recursos disponíveis *online*.
- Tem um bom suporte quer para videojogos 2D, quer para videojogos 3D.
- Realmente funciona. Isto está provado devido a imensos videojogos existentes na plataforma "Xbox Live Arcade", como por exemplo: Schizoid e Dishwasher.
- Rápida e simples para criação de pequenos videojogos.
- Os videojogos criados podem ser facilmente postos a correr numa Xbox 360.

Desvantagens:

- A linguagem utilizada é .NET. Também pode ser uma desvantagem, pois é uma linguagem da Microsoft, o que impossibilita a portabilidade do videojogo para Linux ou Mac OS.

- É necessário ter um ambiente Windows para desenvolver videojogos nesta *Framework*.

- Já não é suportada pela Microsoft, o que pode levar ao aparecimento de diversas incompatibilidades.

Pode-se concluir que a XNA Framework, apesar de uma *Framework* robusta, não será a melhor opção em termos de *Frameworks* que existem no mercado, muito devido ao facto de não suportar vários SO, o que acaba por cortar uma grande fatia de mercado para quem deseja desenvolver um videojogo nos dias que correm. Apesar disso, também não se pode dizer que seja má, pois algumas das suas características, como a facilidade com que se cria um projeto, são realmente preciosas para utilizadores mais novatos.

É ainda de referir novamente que esta *Framework* deixou de ser atualizada por parte da Microsoft [41], apesar dos programadores ainda a poderem utilizar para a criação de videojogos para as plataformas Xbox 360, Windows e Windows Phone.

2.2.2.2 UDK - Unreal Engine

O Unreal Engine é um Motor de Jogo desenvolvido pela Epic Games, com elevado foco em videojogos FPS, apesar de já ter sido usado num elevado número de diferentes géneros de videojogos com sucesso. Este motor é escrito em C++ e tem como principal objetivo a elevada portabilidade dos videojogos criados, suportando as Plataformas e SOs mais comuns.

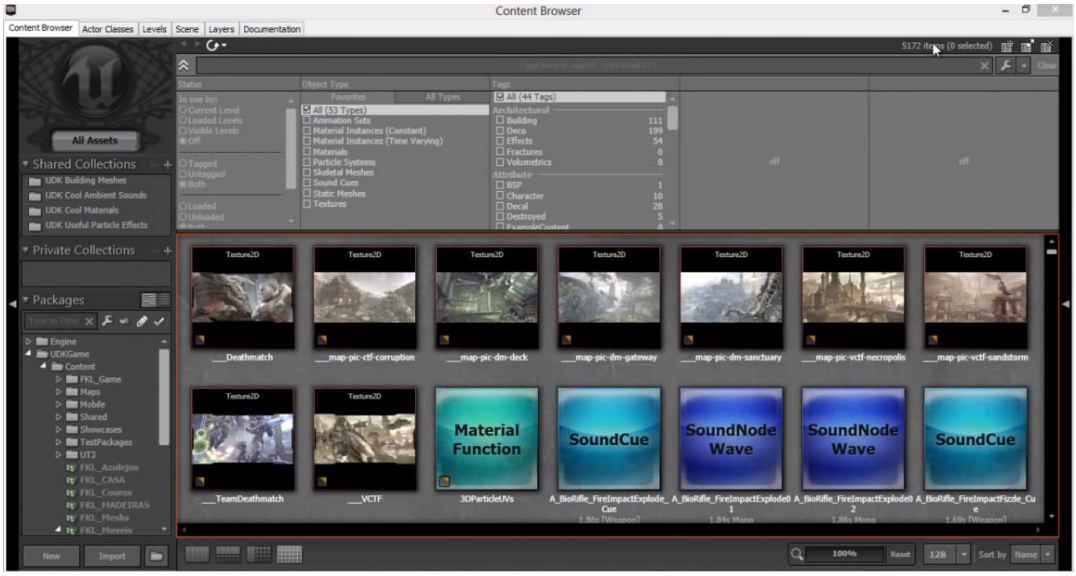


Figura 4 - Interface Gráfica do UDK.

Este Motor de Jogo é um dos mais utilizados pela indústria AAA dos videojogos, devido não só a sua portabilidade, mas também pela elevada potência e desempenho que possui. Para além disso ainda disponibiliza numerosos módulos que se encontram logo à partida dentro

do UDK para que os utilizadores possam criar os seus videojogos com os melhores *standards* existentes no mercado. Alguns destes módulos são: o módulo de simulação de forças físicas (*Unreal Physics*), o sistema de partículas (*Unreal Cascade*), o criador de *Cinematics* (*Unreal Matinee*), etc.

De forma a ficar com uma ideia do poder e facilidades proporcionadas por estes módulos, será demonstrado o módulo *Unreal Kismet,* que não é mais que um sistema de s*cripts* gráfico.

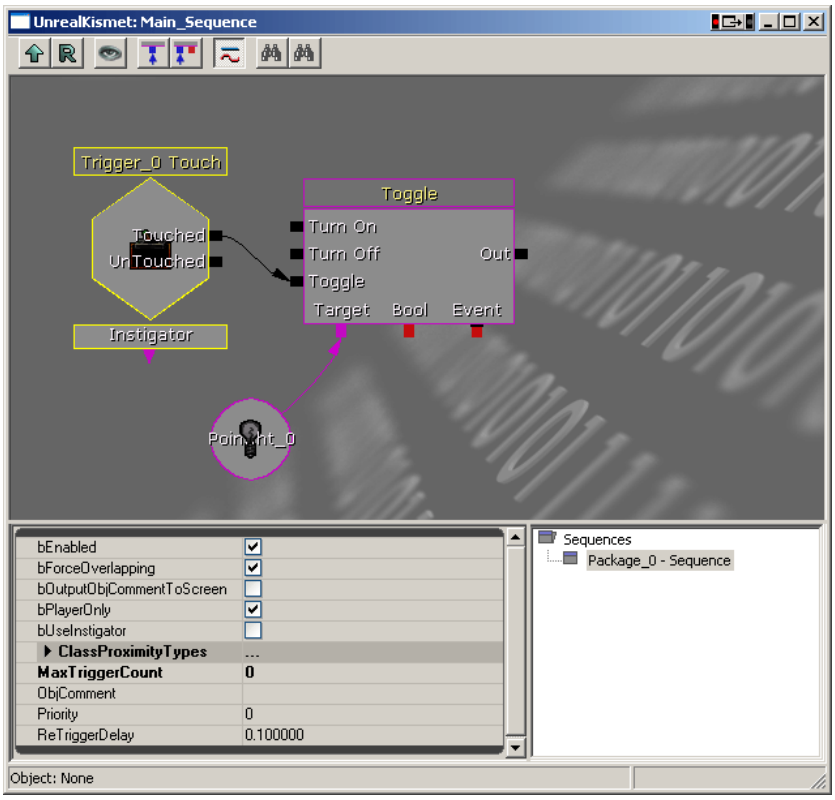


Figura 5 - Demonstração de um *Script* realizado no Unreal Kismet.

O exemplo acima é um *script* que serve para ligar/desligar uma luz dentro da cena criada. Este *script* é composto pelos seguintes elementos:

- Um *trigger* (Trigger_0): Este *trigger* é atribuído a um interruptor presente na cena. Este *trigger* pode ser acionado quando está a ser tocado ou não.

- Um evento (Toggle): Este evento serve para alterar uma ação que existe no videojogo.

- Um objeto (PointLight_0): Este componente serve para representar o objeto, neste caso uma luz, presente na cena.

O que acontece aqui é que o utilizador escolhe ligar o conector *touched* do *trigger* ao *toggle* do Toggle, que por sua vez está ligado ao objeto 'PointLight_0'. Isto faz com que sempre que um ator/jogador clique no interruptor, este *script* será desencadeado e ligará/desligará a luz.

Como se pode observar, este pequeno módulo possibilita que um utilizador sem qualquer formação no campo da programação possa realizar *scripts* para o seu videojogo, sem ter de olhar para uma linha de código.

Tal como na XNA Framework, serão listadas abaixo as vantagens e desvantagens encontradas no uso deste Motor de jogo:

Vantagens:

- Possui um desempenho bastante elevado, mesmo com o uso de objetos com muitos polígonos.
- Facilidade em tornar uma cena "normal". Numa cena com um aspeto muito mais profissional graças a certas componentes existentes no Motor de Jogo, nomeadamente a de Luz (*Unreal Lightmass*) e o suporte a *High Dynamic Range Rendering* (HDRR).
- Facilidade na criação das ações mais comuns num videojogo através dos vários módulos: *Scripts* (*Unreal Kismet*), *Cinematics* (*Unreal Matinee*) e Animações (*Unreal Animation System*), entre outros.
- Módulos gráficos para praticamente todas as ações, o que pode ser vantajoso para criadores inexperientes na área da programação.

Desvantagens:

- Não suporta alguns dos formatos mais comuns nos dias de hoje, como por exemplo o facto de não reter a transparência existente num ficheiro .PNG.
- Pouca documentação *online*, visto que muitas vezes não é fácil de encontrar um tutorial para um certo componente, em comparação com outros motores, como no caso do Unity3D.
- Muitas das vezes é muito complicado fazer *debug* do videojogo, caso se tenha pouca experiência no uso deste Motor de Jogo.
- Não existe uma pasta para o projeto criado. Isto ao início pode ser um bocado estranho pois os ficheiros do utilizador estão na mesma pasta dos ficheiros das *samples* fornecidas com o UDK.

Apesar deste Motor de Jogo ser extremamente robusto e capaz de suportar quase todas as ideias, até mesmo as mais estranhas, que os utilizadores tenham, não se pode dizer que seja uma ferramenta de uso fácil, pois mesmo com todos os módulos gráficos que o UDK possui para praticamente todas as tarefas existentes na criação de um videojogo, tornando estas em ações *What You See Is What You Get* (WYSIWYG), estas também muito devido ao seu elevado

número de opções podem assustar um utilizador que pretende realizar um videojogo simples, e sem conhecimento prévio de criação de videojogos.

Com isto pode-se dizer que, para usar um Motor de Jogo como o UDK, o utilizador já deve possuir alguns conhecimentos prévios na área da criação de videojogos, pois este pode escalar de uma forma acentuada, no que toca à dificuldade de uso. É ainda de referir que este Motor de Jogo tem uma licença paga de 99$, e que para projetos com mais de 50,000$ de lucro todos os meses devem ser dados 25% dos lucros das vendas do videojogo à Epic Games.

2.2.2.3 GameMaker: Studio

O GameMaker: Studio é um Game Maker desenvolvido pela YoYo Games. A ideia principal que este Game Maker pretende transmitir é a de que é simples criar um protótipo para um videojogo, precisando apenas de duas ou três horas para que este esteja pronto, e que o videojogo final a partir desse Protótipo não demorará mais do que um par de semanas. Para realizar estes feitos, os utilizadores não precisam de saber nada sobre programação, nem nada dentro das áreas mais complexas na criação de videojogos, tornado assim este um processo de fácil acesso a qualquer pessoa que deseje criar o seu próprio videojogo.

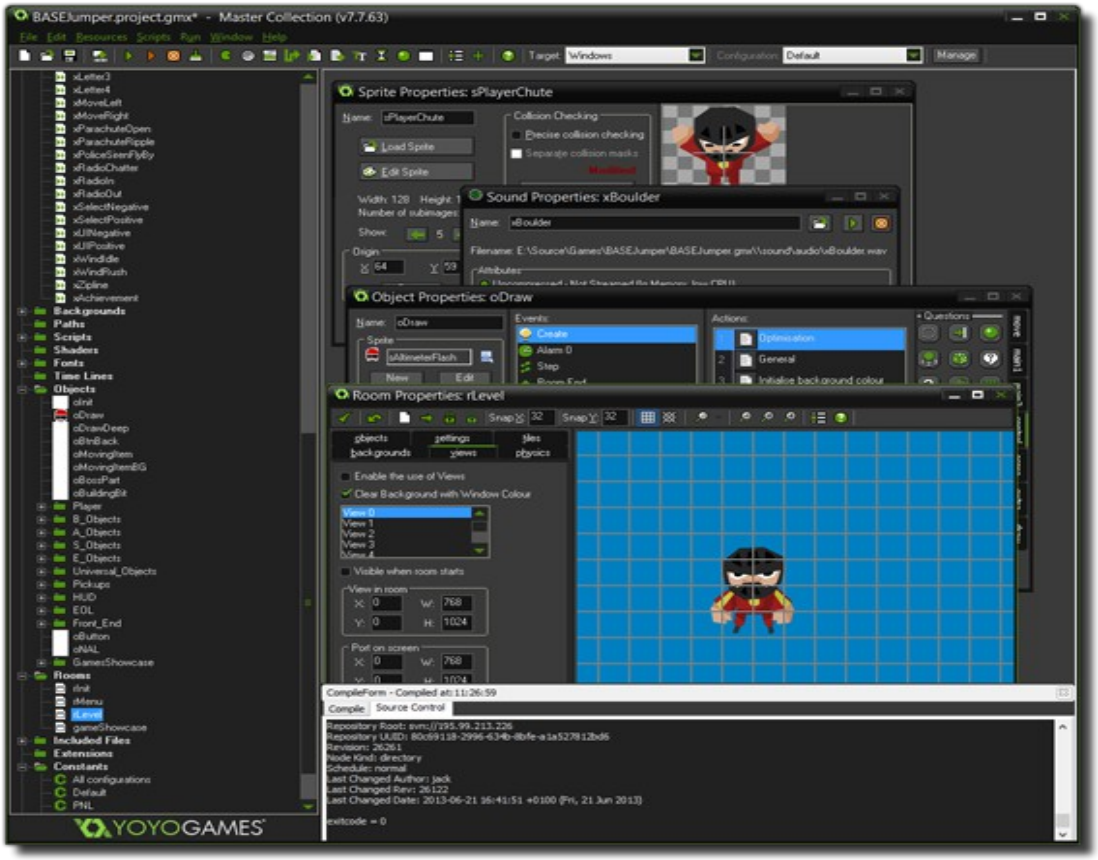


Figura 6 - Interface gráfica do GameMaker: Studio

O GameMaker: Studio tem uma interface gráfica bastante simples, e qualquer utilizador que dedique um par de horas à sua exploração, não terá muitos problemas na construção do seu videojogo. Para além disso, praticamente todas as ações necessárias para a criação de um videojogo são suportadas por um sistema de *Drag-and-Drop* simplificando ainda mais os processos já simples deste Game Maker.

Apesar de tudo parecer tão simples e eficaz, é de notar que o GameMaker: Studio apesar de suportar videojogos 3D, não está pensado para tal, sendo que o seu grande foco pertence à categoria dos videojogos 2D. Para que possamos perceber de que realmente é capaz o GameMaker: Studio, e quais os seus problemas, serão agora enumeradas algumas das suas vantagens e desvantagens.

Vantagens:

- Muito fácil de utilizar. Mesmo que o utilizador não saiba bem o que deve fazer, o GameMaker: Studio é bastante intuitivo.
- Própria linguagem de programação *Game Maker Language* (GML); esta é bastante simples e direta, com imensos métodos já criados.
- Exporta os videojogos para várias Plataformas e SO.
- Editores gráficos para praticamente tudo o que é necessário fazer.
- Bastante *user-friendly.*

Desvantagens:

- É uma ferramenta cara, pois se o utilizador pretender ter acesso ao pacote total do GameMaker: Studio, este poderá ficar por 800$ (é de notar que apesar de existir uma versão gratuita, esta é muito limitada).
- Bastante limitado na área dos videojogos 3D. Não suporta modelos padrão 3D (só suporta modelos .D3D).
- Se o utilizador quiser criar portefólio na área de desenvolvimento de videojogos, este pode não ser o melhor caminho, pois são muitos os conceitos que são aqui omitidos para facilitar o processo.
- Fazer *debug* de um videojogo não é relativamente fácil.
- Videojogos mais complexos acabam por encontrar restrições/*bugs* no sistema.
- Pode criar conflitos entre alguns ficheiros, e com isto perdemos algumas horas de trabalho, sem que o utilizador se aperceba.
- O utilizador é muitas das vezes obrigado a utilizar os editores do próprio GameMaker: Studio para realizar ações ou efeitos desejáveis, mas estes editores são

bastante fracos em comparação com muitos dos programas *standard* que existem no mercado.

Pelos pontos acima, consegue-se perceber que os *Game Makers* são muito focados nos utilizadores com pouca/nenhuma experiência na criação de videojogos e que pretendem começar o seu projeto o mais rápido possível, sem terem de perder imenso tempo a estudar conceitos como *Threads*, alocações de memória, etc.

Como também se pode observar, existem algumas parecenças entre os Motores de Jogo e os *Game Makers*, devido muito ao seu foco nos sistemas WYSIWYG, mas os *Game Makers*, muito devido ao fator de serem mais simples de usar, acabam por ter também muitas mais restrições no desenvolvimento de um videojogo, apesar de muitas destas restrições só se aplicarem em grandes projetos e não em pequenos videojogos 2D. Um criador que pretenda fazer algo simples nunca deve descartar um Game Maker só por este não ser tão poderoso como um Motor de Jogo, pois em grande parte dos casos o criador do videojogo nunca iria tirar o verdadeiro partido de um Motor de Jogo como o que o UDK – Unreal Engine tem para oferecer na criação de um pequeno videojogo 2D.

2.2.2.4 Bullet Physics Library

A Bullet Physics Library é uma biblioteca *open-source* e multi-plataforma que tem como principal objetivo tratar das simulações físicas em tempo real que existam num videojogo, isto é, serve para processar todos os tipos de forças, como a gravidade, colisões entre objetos e outras restrições físicas presentes no videojogo. Esta biblioteca já se afirmou há algum tempo no mercado, sendo já utilizada na produção de diversos videojogos e filmes, tais como o Cars 2, desenvolvido pela Walt Disney.

A particularidade das bibliotecas é que, ao contrário de todas as outras ferramentas anteriormente analisadas, estas quase sempre só tratam de um único campo, sendo que o campo desta biblioteca se situa nas simulações físicas.

Por exemplo, se os programadores duma empresa quisessem utilizar o Bullet para o seu videojogo, não poderiam esperar que a integração desta biblioteca chegasse para fazer o seu videojogo. O que costuma acontecer é que estas bibliotecas costumam ser usadas em conjunto com outras que fornecem os restantes processos necessários para o bom funcionamento de um videojogo, conseguindo assim obter um melhor desempenho do que muitas *Frameworks* ou Motores de Jogo que tentam fazer tudo.

No que toca à integração, a Bullet Physics Library é extremamente simples de usar. Para isso só é necessário fazer um simples *setup* de 6 linhas:

```
btBroadphaseInterface* broadphase = new btDbvtBroadphase();

btDefaultCollisionConfiguration* collisionConfiguration = new
btDefaultCollisionConfiguration();

btCollisionDispatcher* dispatcher = new
btCollisionDispatcher(collisionConfiguration);

btSequentialImpulseConstraintSolver* solver = new
btSequentialImpulseConstraintSolver;

btDiscreteDynamicsWorld* dynamicsWorld = new
btDiscreteDynamicsWorld(dispatcher,broadphase,solver,collisionConfiguration);

dynamicsWorld->setGravity(btVector3(0,-9.81f,0));
```

Nestas 6 linhas são definidos alguns elementos necessários para a utilização da biblioteca, como: a *broadphase,* a configuração para as colisões, o *dispatcher*, o *physics solver,* e finalmente a definição do "mundo" do videojogo e a força gravitacional de -9.81ms^{-2} no eixo dos Y.

Depois destas linhas, vem o código onde o programador pode definir as formas de colisão e corpos rígidos que tem no seu mundo. E começar as simulações com os objetos criados previamente.

Vantagens:

- Pode-se importar os módulos necessários e não toda a biblioteca.
- Fácil de incorporar com outras bibliotecas.
- Só processa as simulações físicas do videojogo. Acaba por ser uma biblioteca bastante robusta pois só se preocupa com este aspeto.
- Um desempenho bastante melhor que outras bibliotecas concorrentes, como a Open Dynamics Engine.
- Suporta *Multi-Threading.*

Desvantagens:

- Complexa de usar para criadores de videojogos que não percebem alguns conceitos necessários de programação.
- Só processa as simulações físicas do videojogo; muitos criadores de videojogos preferem quando podem obter tudo que precisam de um só lugar.
- Documentação *online* acaba muitas vezes por estar desatualizada devido às diversas atualizações.

Depois desta pequena análise, percebe-se que as bibliotecas para desenvolvimento de videojogos acabam por ser as que mais controlo dão aos programadores, mas ao mesmo tempo também acabam por ser uma das formas mais trabalhosas de se fazer um videojogo, pois requerem bastante conhecimento prévio na área de programação para que sejam utilizadas da melhor forma.

As bibliotecas, de um modo geral, acabam por ser bastante confusas para alguns programadores, pois normalmente um videojogo não se faz com uma biblioteca, mas com várias. Por exemplo, no caso de um programador utilizar a Bullet Physics Library para simulações físicas, depois terá de escolher bibliotecas também para: Gráficos, Som, Inteligência Artificial, *Inputs*, etc, pois a Bullet Physics Library só se preocupa com as forças físicas do videojogo.

Um dos melhores aspetos da utilização de bibliotecas é que, ao contrário das outras ferramentas, se o programador não estiver completamente satisfeito com a biblioteca que usa num certo aspeto, pode sempre procurar outra que melhor se ajuste ao que pretende fazer e utilizá-la alterando unicamente as partes do seu código que fazem referências a essas mesmas bibliotecas. Note-se que, no caso dos *Game Makers*, isto torna-se muitas das vezes impossível e o programador acaba por ter de adaptar-se ao que a ferramenta oferece, mesmo que não seja a maneira ideal para o seu videojogo.

2.2.2.5 Conclusão

Pelas análises anteriores, consegue-se perceber que existem várias diferenças entre estes quatro tipos de ferramentas que se podem utilizar para a construção de um videojogo. Para finalizar esta pequena análise, serão agora comparados todos os quatro tipos numa perspetiva global, mas tendo como referência as análises previamente feitas. Desta forma as vantagens/desvantagens que uns têm sobre os outros serão mais simples de se perceber.

Tabela 1 - Comparação entre os diferentes tipos de Ferramentas

	Framework	Motor de Jogo	*Game Maker*	*Libraries*
Facilidade de Utilização	C+	A	B	C-
Necessário possuir conhecimentos de Programação	A	B (nalguns casos)	C	A+
Interface Gráfica	C	A	A	C
Suporta Videojogos 2D	A	A	A	A
Suporta Videojogos 3D	A	A	B	A
Multi-Plataforma	B	B+	B	B+
Desempenho	B+	B+	C	A-
Restrições sobre o que criador pode fazer	D	C-	B	D

A – Muito Fácil / Sim / Muito Elevada

B – Fácil / Alguns / Elevada

C – Normal / Não / Normal

D – Difícil / - / Baixa

Como se pode ver na Tabela 1, existem diversas razões que levam à escolha de uma ferramenta para determinado utilizador. Este pode querer ter o seu videojogo feito o mais rapidamente possível, e ai os Motores de Jogo e *Game Makers* são algumas das melhores opções, ou, se preferir ter maior controlo sobre tudo que acontece dentro das rotinas do seu videojogo poderá sempre optar por uma biblioteca, ou *Framework*.

Desta forma, pode-se concluir que não existe uma ferramenta perfeita, mas sim várias boas ferramentas que ajudam os utilizadores a criar os seus videojogos.

Os criadores devem sempre olhar para o videojogo que pretendem criar e tentar perceber qual a ferramenta que melhor se encaixa, conforme os seus requisitos.

2.3 Tecnologias e Ferramentas utilizadas

Nesta última secção do capítulo de Contexto serão mencionadas de uma forma bastante resumida todas as ferramentas, linguagens e tecnologias utilizadas no desenvolvimento do protótipo, para que o leitor consiga duma forma rápida ter uma ideia do que são as mesmas.

2.3.1 C++

O C++ [42] é uma das mais populares linguagens de programação Orientada a Objetos (OO), e esta tem como principal objetivo obter elevado desempenho. Algumas das vantagens desta linguagem são: ser uma linguagem *ISO-standardized*, compilar o código diretamente para código nativo da máquina tornando-a numa das linguagens mais rápidas, e possuir imensas bibliotecas de suporte.

Esta linguagem ainda hoje é muito utilizada no desenvolvimento de videojogos, como por exemplo no caso do Guild Wars 2 [43] e por alguns dos Motores de Jogo mais usados na indústria como no caso do Unreal Engine 4 [44].

Foi feita a escolha de se utilizar esta linguagem pois, como já mencionado acima, é uma linguagem muito usada no desenvolvimento de videojogos nos dias de hoje, devido a todas as suas vantagens, e porque é compatível com vários SO.

2.3.2 OpenGL

O OpenGL [45] (*Open Graphics Library*) é uma biblioteca desenvolvida pela *Khronos Group*. Esta biblioteca é um *standard* utilizado pela indústria da informática, para criação de aplicações com interfaces gráficas 2D e/ou 3D.

Esta biblioteca foi escolhida para este projeto pois, para além de ser um *standard*, é uma biblioteca muito estável e suporta diversos SO.

2.3.3 FMOD

O FMOD [46] é uma biblioteca de áudio desenvolvida pela Firelight Technologies. É uma das bibliotecas mais utilizadas, nos dias de hoje, pelas empresas que criam videojogos, pois é muito estável, suporta vários SO e vários tipos de ficheiros de som (.ogg, .mp3, etc.), e está constantemente a ser atualizada/melhorada pela Firelight Technologies.

Apesar de ser uma biblioteca paga, a nova versão da mesma é gratuita para criadores independentes, o que a torna numa das melhores escolhas disponíveis no mercado para estes criadores.

2.3.4 GLFW

GLFW [47] é uma biblioteca para ser usada com o OpenGL. Esta biblioteca desenvolvida pela The GLFW Development Team, facilita todo o processo de criação/manutenção de janelas e contextos OpenGL.

Ainda consegue tratar *inputs* recebidos através de *gamepads*, teclados e ratos, facilitando também o processamento destes.

2.3.5 Box2D

Box2D [48] é a biblioteca utilizada neste projeto para a simulação de forças físicas e colisões num espaço de duas dimensões.

Desenvolvida por Erin Catto, é gratuita e escrita em C++ e, graças a isso, suporta vários SO.

2.3.6 FreeType

FreeType [49] é uma biblioteca de renderização de tipos de caracteres, que suporta vários tipos tais como: *TrueType*, *OpenType*, entre outros.

Esta biblioteca para além de permitir a renderização de caracteres simples, suporta alguns efeitos, tais como, sombras, sublinhados e *strokes*.

2.3.7 Theora Playback Library

A Theora Playback Library [50] é uma biblioteca criada pela empresa Cateia Games. Tem como principal objetivo a reprodução de vídeos do tipo .ogg (*codec* gratuito criado pela Xiph.Org Foundation).

2.3.8 Code::Blocks

O ambiente de desenvolvimento escolhido para o protótipo desta Dissertação foi o Code::Blocks [51]. Este Ambiente Integrado de Desenvolvimento (IDE), desenvolvido pela The Code::Blocks team em C++, é gratuito e orientado à criação de *software* em C/C++. Esta ferramenta contém um vasto leque de caraterísticas necessárias para o desenvolvimento de *software*, tais como: compiladores, editor de código, *Debugger*, entre outras, facilitando assim, todo o processo de criação de *software*.

2.3.9 LibreOffice Writer

Para a escrita desta dissertação ainda foi utilizada a ferramenta LibreOffice Writer [52]. Esta ferramenta é um processador de texto gratuito (*Open-Source*), desenvolvido pela The Document Foundation, e suporta diversas plataformas, como por exemplo Linux, Microsoft Windows e Mac OS X.

2.3.10 LibreOffice Draw

Este *software* [53] foi utilizado para realizar todos os esquemas necessários para a análise do videojogo. Esta aplicação faz parte do mesmo pacote da aplicação acima mencionada (LibreOffice Writer), sendo desenvolvida pela mesma organização.

2.3.11 GIMP

O GIMP [54] (GNU Image Manipulation Program) é um editor de imagem completamente Open-Source. Este *software* foi originalmente criado por Spencer Kimball e Peter Mattis, em Janeiro de 1996.

Esta aplicação foi utilizada para a criação das diversas texturas utilizadas no videojogo (protótipo) desenvolvido.

2.4 Conclusão

Como se pode ver neste capítulo, os videojogos são cada vez mais uma das fontes de rendimento com maior potencialidade na área do entretenimento. Para além disso, estes estão cada vez mais detalhados e profissionais, muito devido às ferramentas existentes no mercado. Pois, como pode ser visto, existem ferramentas para todo os tipos de criadores de videojogos, quer os que percebem de programação, quer os que não percebem. Criando assim a possibilidade, de um maior número de pessoas poderem criar o seu próprio videojogo.

Para terminar, foram ainda apresentadas todas as ferramentas e linguagens que irão ser mais tarde utilizadas durante o desenvolvimento do protótipo.

3 Análise e Design

Neste capítulo serão abordados e definidos todos os elementos necessários para a implementação do videojogo.

Inicialmente será feito o desdobramento do videojogo a ser realizado. Note-se que este não passará de um protótipo e que algumas das áreas do mesmo não serão sequer abordadas, como o tipo de som, tipo de arte, entre outros, por não pertencerem concretamente ao estudo que se pretende realizar nesta dissertação. Ainda durante este desdobramento serão analisadas as potenciais plataformas suportadas pelo videojogo.

De forma a representar toda esta informação duma forma simples, será criado um *Game Design Document* para o videojogo. Um GDD [55] não é mais do que uma *blueprint* que expressa todos os detalhes de construção de um videojogo, desde a forma como o jogador interage com o videojogo, até ao conteúdo de cada nível existente. Com isto quer-se dizer que todos os elementos que estarão presentes no videojogo devem também estar presentes no GDD. Tal como referido acima, nem todos os pontos do GDD serão abordados, o que não significa que estes não sejam de máxima importância na criação de um videojogo.

Este videojogo (protótipo) terá como principal função a aprovação do bom funcionamento e também a demonstração da potencialidade da implementação escolhida, implementação essa que será discutida no subcapítulo posterior ao desdobramento e usada para a construção do videojogo e dos diferentes tipos de interfaces (ver Capítulo 4) a poderem ser aplicados no videojogo.

Depois de todos estes pontos decididos, irá ser feito o levantamento dos requisitos, sendo esta uma das tarefas mais importantes na fase de análise, pois caso estes não estejam bem definidos podem levar a que exista uma carga de trabalho adicional, ou atrasos no decorrer do desenvolvimento do protótipo.

Posteriormente, será discutida a metodologia de trabalho a ser empregue durante o desenvolvimento de todas as partes necessárias para a construção do protótipo.

3.1 Desdobramento do Videojogo

Neste subcapítulo será descrito o videojogo escolhido para este projeto, falando do seu género e abordando também alguns dos aspetos fundamentas do mesmo, os controlos necessários para o bom funcionamento do videojogo, as plataformas que serão suportadas e os seus SO.

Desta forma, separando todos os pontos, consegue-se perceber muito mais facilmente os requisitos que deverão existir para a implementação do videojogo duma forma muita mais simples e concisa, evitando erros futuros.

3.1.1 Videojogo escolhido

O videojogo (protótipo) a ser desenvolvido para esta dissertação será do género: Plataforma. Alguns dos videojogos do género mais conhecidos são: "*DuckTales*" [56], "*Super Meat Boy*" [57], "*Rayman Legends*" [58] e "*Super Mario Bros.*" [59].

Neste género de videojogo o jogador deve controlar o seu personagem de forma a chegar de um ponto A até um ponto B, ou seja, ao final de um nível. Durante a viagem entre estes dois pontos, o jogador deverá ser capaz de conseguir controlar, com precisão, o seu personagem de forma a conseguir ultrapassar todos os obstáculos que serão encontrados ao longo do nível, tais como: inimigos, armadilhas, buracos, etc.

Este género de videojogos tem a mecânica de salto do personagem, como um elevado elemento do videojogo, sendo que faz parte natural de quase todos os personagens presentes neste tipo de videojogos. Este salto pode ainda ser transformado/repensado, de forma a tornar o videojogo mais único, tal como acontece no videojogo "*Ristar* " [60] em que o personagem estende os braços enquanto salta para poder balançar.

Para este projeto, pretende-se criar um videojogo de Plataforma, que assente nas verdadeiras bases do género. Este será um videojogo em que o jogador poderá navegar e saltar com o personagem. Será composto por níveis, de forma a que o jogador tenha de ultrapassar o nível atual para poder aceder ao seguinte.

Os níveis serão constituídos por plataformas, plataformas estas que o jogador deverá utilizar par progredir no nível, obstáculos, que existirão de forma a complicar a vida ao jogador, e inimigos que pretendem destruir o jogador. Note-se ainda que tanto as plataformas como os obstáculos poderão tanto ser estáticos como dinâmicos, movimentando-se de forma a requererem ações mais precisas por parte do jogador. É ainda de acrescentar que estes níveis deverão ser desafiantes para o utilizador, conforme este progride, de forma a não se tornarem aborrecidos.

Este género foi escolhido pois apesar de na *8-bit/16-bit Era* ser um dos géneros mais comercializados, tem vindo a decair desde então, não sendo tão utilizado como um género de videojogo, mas sim como base de outros géneros, como acontece com o género dos

Running Platformers, que tem vindo a ser um dos géneros mais explorados no mercado *mobile*, mesmo pelas grandes companhias. Exemplo disso é o videojogo "Rayman Fiesta Run" [61], desenvolvido pela Ubisoft Casablanca.

Para que seja mais simples de perceber o que se pretende fazer neste videojogo, foi ainda criado um GDD (Anexo A), onde se pode consultar toda a informação previamente explicada num formato mais indicado para a mesma. É também de salientar que, para além da informação discutida neste capítulo, é também apresentada no GDD toda a informação que será ainda discutida nos capítulos/subcapítulos seguintes.

3.1.2 Público alvo do Videojogo

Como qualquer produto, este videojogo também se destinará a um grupo de pessoas bem definido, pois tentar criar algo que satisfizesse todo o tipo de pessoas seria uma tarefa praticamente impossível.

Tendo isto em conta, este videojogo será realizado com o objetivo de apelar aos jogadores entre os 15-45 anos, tanto do sexo masculino como do feminino, com computadores que estejam entre a qualidade média-alta.

Estes jogadores deverão ser consumidores de videojogos através de plataformas de venda de videojogos em formato digital, e com um tempo de jogo médio superior a 30 minutos. Para além de tudo isto, devem ainda ser fãs do estilo 2D, principalmente videojogos com elevado foco no modo de jogo *single-player* e gostar de um bom desafio.

3.1.3 Controlos do Videojogo

Os controladores a serem utilizadores para este videojogo serão: o *Gamepad* e o Teclado, mas o videojogo será otimizado para o *Gamepad,* tendo este como controlador principal/desejado. Este ponto acaba por ser muito importante devido ao facto dos controladores suportados irem alterar a forma como a interface do videojogo deve ser criada, de forma a criar o menor número de problemas ao jogador.

A escolha do *Gamepad* como controlador principal justifica-se, pois os *Gamepads* conseguem ser muito mais precisos do que os teclados; ao contrário destes últimos, consegue-se obter leituras muito mais precisas, como por exemplo, quando um jogar utiliza os Analógicos do *Gamepad*, estes podem fazer com que o personagem se movimente a 30% de velocidade, enquanto que no PC uma tecla está premida, ou não. Logo não existe esta sensibilidade que é muitas das vezes importante nos videojogos.

Os controlos do videojogo podem ser encontrados com maior detalhe no GDD (Anexo A).

Apesar de serem estas as teclas por omissão para o videojogo, o jogador deverá ter a possibilidade de configurar as ações necessárias para jogar o videojogo com as teclas que este achar mais convenientes para si.

3.1.4 Plataformas Suportadas

A plataforma escolhida para este projeto foi o PC, suportando os sistemas operativos mais comuns: Microsoft Windows, Linux, MacOS X.

Decidiu-se escolher esta plataforma pois:

- É uma das plataformas mais simples para se publicar um videojogo. Devido ao número elevado de serviços de distribuição digital, ou até mesmo pela facilidade de distribuição pelo próprio criador do videojogo.
- Tem uma das maiores comunidades mais entusiastas/apoiantes de videojogos por parte de criadores independentes.
- O estilo do videojogo a ser criado é bastante compatível com a plataforma escolhida.
- Um número de bibliotecas e tutoriais gigantescos para desenvolvimento de videojogos nesta plataforma.

Plataformas como: Playstation 4, Xbox One e Wii U, ficaram de fora muito por causa do processo que é preciso ter em conta para se poder publicar um videojogo nestes sistemas, e também porque muitas das vezes não é fácil ter acesso a um SDK para o sistema, ou até mesmo por este ser caro.

Os sistemas *Mobile* também acabaram por ser descartados, muito devido às mecânicas e controlos que este videojogo implementa, pois estas não são de todo ótimas para videojogos que recorram a um *touchscreen* como forma de *input*.

3.2 Arquitetura escolhida

Depois da análise realizada acima (Capítulo 2) e ainda com a ajuda da análise feita por Ricardo Moreira [62], chegou-se ao consenso de que criar uma *Framework* Modular, com recurso a *Game Development Librarys*, seria a melhor solução para este projeto.

Decidiu-se criar a própria *Framework* pelas seguintes razões:

Em primeiro lugar, ao contrário de todas as outras implementações (*Frameworks*, Motores de Jogo e *Game Makers*) já previamente analisadas, ao criar-se a própria *Framework* o programador/criador do videojogo tem completo controlo sobre tudo que se está a passar nas suas rotinas de jogo (Game Loop), sendo que estas podem ser melhoradas, tendo em

conta o tipo de videojogo que está a ser feito e para que plataformas está a ser desenvolvido. Isto é, como todo o código da *Framework* está acessível ao programador, coisa que muitas das vezes não acontece, este pode fazer as alterações que bem desejar.

Suporte a novos *chipsets* ou placas gráficas. Muitas das vezes as empresas que fabricam estes componentes alteram pequenas funcionalidades ao longo do tempo, de forma a rentabilizar/melhorar algum processo rotineiro. Isto pode levar a que certos componentes de um videojogo deixem de funcionar, ou que funcionem duma forma não desejada. Caso o criador do videojogo esteja "preso" a uma das implementações previamente discutidas, este tem de esperar que a empresa responsável pela ferramenta que está a usar faça uma atualização que passe a suportar estas novas funcionalidades, não podendo, em grande parte dos casos, ser o próprio a resolver o problema.

Com esta *Framework* Modular, também se consegue evitar que um componente fique obsoleto. Pois por vezes, algumas das *Frameworks* existentes no mercado utilizam *Game Development Librarys* que deixam de ser atualizadas, ou até que são completamente abandonadas pelos criadores das mesmas. Neste caso, se a empresa responsável pela *Framework* utilizada no desenvolvimento do videojogo não alterar a *Game Development Library* que utiliza para outra mais estável, pode-se dar o caso do criador do videojogo estar "preso" a uma biblioteca obsoleta. Isto não acontece nesta *Framework* Modular, pois todas as bibliotecas podem ser alteras a qualquer momento, com o mínimo de impacto no restante código do videojogo.

Ainda relacionado com o ponto anterior é o facto de que através duma *Framework* Modular é fácil adicionar novas funcionalidades à mesma. Por exemplo, se o criador de um videojogo desejar passar a suportar algum tipo de ficheiro ainda não suportado, ou até mesmo criar um *pipeline* direto com algum *software* de edição (ler ficheiros .xcf pertencentes à ferramenta de edição de imagem GIMP, conforme estes são alterados no próprio programa), este pode fazê-lo. Para isso terá simplesmente de acrescentar um novo módulo que lhe permita fazer esse tipo de processamento. Isto muito raramente acontece com ferramentas disponíveis no mercado. Nestes casos, os utilizadores costumam utilizar uma área comunitária (fóruns por exemplo) onde pedem que certa funcionalidade seja implementada numa próxima versão da ferramenta, ou se perceberem que a função que procuram é completamente impossível de vir a existir na ferramenta utilizada, acabam por ter de passar a utilizar uma ferramenta diferente, tendo de voltar a aprender como fazer todos os processos que já achava banais na ferramenta previamente utilizada.

Pode vir a suportar qualquer Linguagem/Ambiente. Quer-se com isto dizer que enquanto na maior parte das ferramentas no mercado disponibilizam um certo número de plataformas de exportação possíveis, um criador de videojogo que tenha a sua própria ferramenta, pode a qualquer momento implementar uma forma de exportar para uma nova plataforma que lhe pareça necessária para os seus projetos.

Finalmente, criar a própria *Framework* irá ajudar imenso a entender/aprender todos os conceitos básicos que funcionam por trás de um videojogo, sendo que estes muitas das

vezes ajudam de forma drástica na implementação de certas rotinas mais complexas, pois o criador da *Framework* sabe como esta está a funcionar.

Mas nem tudo são pontos positivos acerca da criação da própria *Framework*. Inicialmente requererá um tempo de construção e testes, que não deverão acontecer no caso de se utilizar uma ferramenta já existente no mercado, pois estas tendem a já ter sido testadas ao longo do tempo por outros utilizadores. Deixa de existir qualquer ajuda *online* para um processo, pois não existe documentação previamente feita, como no caso das ferramentas existentes, e muito provavelmente terá um número de *bugs* muito maior do que qualquer outra ferramenta (relacionado também com o facto de não ter sido ainda tão exaustivamente testada).

Apesar destes pontos negativos, acha-se que estes poderão a vir a ser superados com o tempo e que realizar a própria *Framework* traz uma liberdade completamente diferente de utilizar uma ferramenta já existente, podendo esta liberdade ser uma peça chave no desenvolvimento de videojogos futuros.

No que diz respeito à *Framework* Modular a ser criada durante o curso desta Dissertação, esta será unicamente pensada para videojogos 2D, apesar que, com algumas mudanças, esta poderia ser implementada para videojogos 3D, trocando alguns dos módulos existentes e adicionando novos.

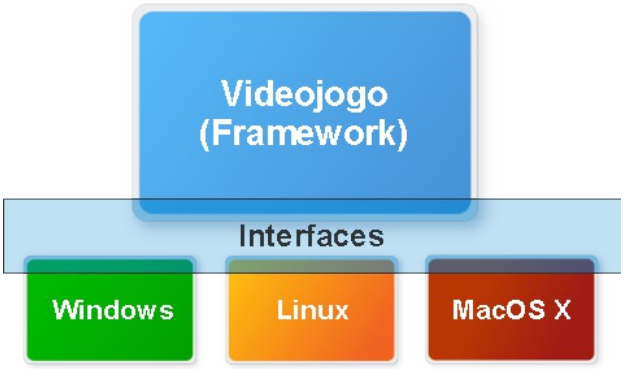


Figura 7 - Arquitetura de Alto-Nível

Tal como descrito no subcapítulo anterior, a *Framework* deverá suportar a plataforma PC e três dos seus Sistemas Operativos mais utilizados. Para isso, existirá uma camada de Interfaces que deverá ser responsável pelo tratamento dos diferentes componentes de cada SO. Desta forma, a *Framework* poderá abstrair-se da plataforma que está a ser exportada, sendo que toda a comunicação entre a *Framework* e os diferentes sistemas é feita através da camada de Interface (Figura 7).

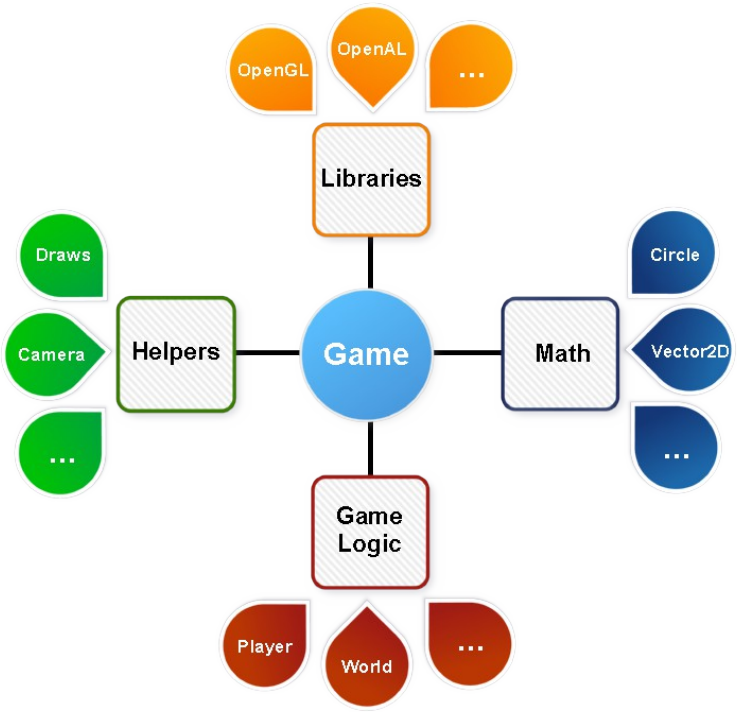


Figura 8 - Representação Gráfica da *Framework*

A *Framework* em si será dividia em cinco grandes partes (Figura 8), sendo estas: *Helpers, Libraries, Math, Game Logic* e *Game*. Cada uma destas partes pode ser entendida como se fossem grandes Módulos, que são compostos de Módulos mais pequenos, sendo que cada um destes tem a sua função na *Framework*.

No caso dos módulos presentes no grupo *Libraries*, estes devem fazer uma espécie de *Adapter* entre as *Game Development Librarys* utilizadas e a *Framework*, de forma a facilitar o acesso às mesmas por parte de todos os processos a serem realizados na construção do videojogo.

Os *Helpers* e *Math* serão módulos onde existirão classes que ajudarão o programador a realizar tarefas rotineiras duma forma mais simples ou com maior desempenho, no caso dos *Helpers*, ou ainda a simplificar cálculos, ou criação de formas geométricas simples, no caso do módulo *Math*.

O módulo *Game Logic* é onde todas as classes e/ou métodos, referentes à lógica do videojogo a ser criado, estarão guardados. Nesta secção são implementadas coisas como construção de níveis, ações dos personagens, e todos os restantes aspetos relacionados com a lógica do videojogo.

Finalmente existe ainda o Módulo Game, sendo este o módulo que faz a união entre todas as partes da *Framework*, sendo a base para o bom funcionamento desta. Apesar disso, poderá ocorrer o caso de algum dos módulos comunicar diretamente com outro de forma a facilitar

algum tipo de processo, por exemplo, uma classe que desenhe círculos no *Helper* poderá comunicar com o módulo *Math* de forma a receber por parte do programador um círculo já previamente criado e desenhar este duma forma mais direta, em vez do utilizador ter de passar todos os parâmetros referentes ao círculo.

Com isto consegue-se perceber que a *Framework* tem como base um processo de desenvolvimento muito focado na elevada flexibilidade, de forma a facilitar a implementação de novas funcionalidades no futuro, sem destruir todas as funcionalidades já previamente existentes.

3.3 Levantamento de requisitos

A especificação de requisitos é uma das tarefas mais importantes na fase de análise, pois quando os requisitos são mal especificados podem vir a produzir atrasos em qualquer projeto.

Os requisitos, de modo geral, podem ser classificados em três grandes grupos: os requisitos funcionais, os não funcionais e os requisitos de interface.

Neste relatório serão tratados os requisitos funcionais e não funcionais, sendo que os funcionais serão divididos em dois subgrupos: requisitos funcionais para a construção da *Framework* Modular e requisitos funcionais para a implementação do videojogo, tornando assim a leitura dos requisitos mais simples.

Os Requisitos funcionais do sistema descrevem as funções que este deve possuir, isto é, são todas as ações/serviços que o sistema deve conseguir desempenhar para que o projeto funcione.

Os Requisitos não funcionais do sistema demonstram as restrições, ou atributos de qualidade, presentes no sistema e/ou no processo de desenvolvimento do projeto. Podem ser vistos como padrões de qualidade do projeto.

3.3.1 Requisitos funcionais da *Framework*

A *Framework* deverá ser capaz de:

- Suportar a Plataforma PC e alguns dos seus sistemas operativos (Windows, Linux, MacOS);
- Suportar vários géneros de videojogos;
- Carregar Imagens (.PGN, .JPEG), Sons (.OGG), Vídeos (.OGG).
- Carregar e Criar ficheiros de dados (*savegames*, etc.)

- Permitir a reprodução de áudio;
- Permitir a visualização de vídeos;
- Visualização de Texturas;
- Suportar a criação de animações (Sprites);
- Desenhar formas primitivas (Quadrados, Rectangulos);
- Suportar ficheiros .TTF (True Type Fonts);
- Suportar várias resoluções;
- Receber *input* por parte do jogador de forma a este conseguir interagir com o videojogo (Teclado/Rato e Comando);
- Suportar criação de diversas cenas 2D;
- Suportar criação de objetos estáticos e dinâmicos;
- Testar colisões entre objetos presentes na cena;
- Representar a quantidade de *Frames* Por Segundo (FPS) do videojogo.

3.3.2 Requisitos funcionais do Videojogo

O videojogo deverá:

- Possuir menus para que o jogador consiga navegar entre as várias opções disponíveis (caso se aplique);
- Capacidade de Controlar volume e resolução do ecrã;
- Permitir que o jogador pare momentaneamente (pause) o videojogo a qualquer momento;
- Suportar um sistema de movimentação simples (esquerda, direita);
- Suportar a ação de saltar;
- Suportar forças físicas (gravidade).

3.3.3 Requisitos não funcionais

- **Desempenho**

  O videojogo deverá correr de forma estável e rápida em qualquer máquina, independentemente do seu sistema operativo, para que o jogador não sinta qualquer tipo de frustração não propositada. O número de FPS deve ainda ser alto de forma a não afetar o *gameplay* do videojogo.

- **Fiabilidade**

  O videojogo deverá guardar todos os dados necessários para que o jogador possa recomeçar a qualquer momento a sua aventura. Estes dados devem ser guardados de forma eficiente e eficaz, de maneira a não perturbar o bom funcionamento do videojogo, e para que o jogador não tenha qualquer problema com corrupção de dados, perdendo todo o seu progresso.

- **Usabilidade**

  A interface apresentada ao jogador deverá ser sempre o mais simples e intuitiva possível, de forma a evitar que este cometa erros ou que não consiga perceber algum dos elementos presentes. Todos estes fatores devem ainda ter em conta o público-alvo.

- **Manutenibilidade**

  Toda a *Framework* deve ser construída de forma a facilitar a alteração de qualquer um dos seus componentes duma forma simples e eficaz, isto é, um componente poderá ser substituído por outro mais adequado, ou até mesmo melhorado, sem que afete os restantes componentes da *Framework*.

- **Portabilidade**

  Todos os videojogos desenvolvidos nesta *Framework* devem suportar os seguintes sistemas operativos: Microsoft Windows, Linux e MacOS X. Desta forma, o videojogo deverá ser executável na maior parte dos computadores pessoais.

3.4 Metodologia de desenvolvimento

Para que o processo de desenvolvimento duma aplicação seja bem-sucedido, deve-se optar por uma metodologia de desenvolvimento adequada. O processo mais comum na área de desenvolvimento de *software* passa pelo Modelo em Cascata [63], modelo este onde se divide o projeto em cinco partes: Análise, Design, Implementação, Testes e Manutenção.

Apesar deste modelo ser bom, quando se enverga num projeto complexo, este deixa de ser o mais adequado. Muito devido à sua estrutura essencialmente sequencial, pois estes projetos mais complexos requerem que o criador alterne entre as várias fases.

Para colmatar este problema decidiu-se escolher o Modelo em Estrela (Figura 9). Esta metodologia, proposta por Hix e Hartson em 1993 [64], ao contrário da metodologia em Cascata, deixa de ser sequencial e passa a ser iterativa, sendo que os criadores de *software* podem pegar em qualquer uma das fases, isto é, permite que, por exemplo, se inicie a fase de implementação sem que a fase correspondente à análise de requisitos esteja completamente fechada.

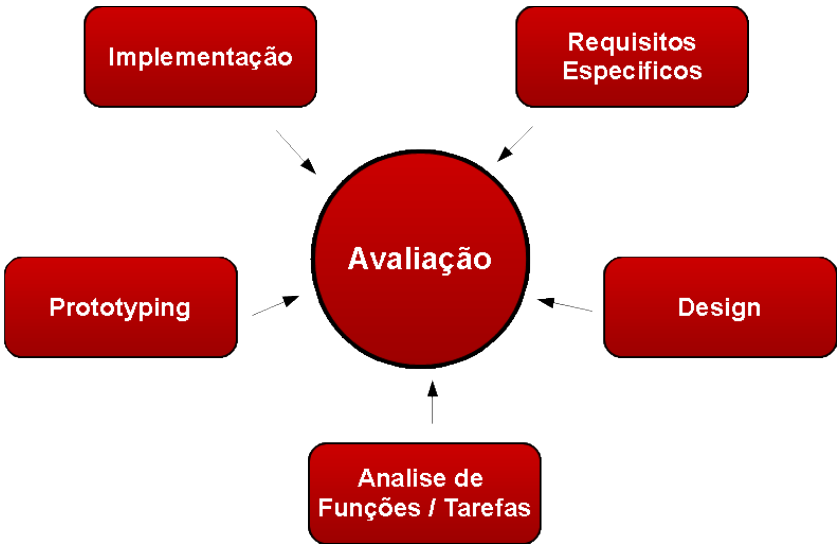


Figura 9 - Metodologia em Estrela

Hix e Hartson verificaram ainda que este tipo de abordagem é preferível, contrariamente ao Modelo em Cascata, quando existem muitas componentes de interação com o utilizador durante o desenvolvimento. Isto é, existe muita imprevisibilidade no comportamento do utilizador, o que por sua vez, obriga a que exista um processo iterativo de desenvolvimento que facilite as correções a serem realizadas.

Com o uso do Modelo em Estrela, o projeto criado para esta Dissertação, pode vir a sofrer diversas alterações ao longo da sua implementação, conforme o sistema é testado/avaliado, sem que seja necessária uma completa revisão de todas as fases, como acontece no modelo em Cascata.

Pode-se então concluir que esta metodologia se foca na avaliação, promovendo sucessivas correções e aperfeiçoamentos do sistema em desenvolvimento.

3.5 Conclusão

Neste capítulo foi realizado o desdobramento do videojogo a ser realizado, onde se pode perceber qual o objetivo deste, os controlos a serem utilizados, as plataformas suportadas e outros pontos importantes com a ajuda do desenvolvimento de um GDD. GDD este que pode ser encontrado no Anexo A.

Para além disso, foi ainda discutida a forma como este videojogo seria implementado, tendo sido escolhida a criação de uma *Framework* Modular, possuindo um forte foco na sua evolução e manutenção, muito devido à fácil substituição dos diferentes módulos.

Ainda neste capítulo foram analisados os diferentes requisitos funcionais e não funcionais para a criação da *Framework* e do videojogo, de forma a evitar atrasos durante a fase de implementação.

Finalmente foi também decidida a metodologia de desenvolvimento, tendo sido o Modelo em Estrela o escolhido, devido à liberdade que este oferece durante o desenvolvimento de um projeto.

4 Interface com o Jogador

Nesta secção, serão discutidos todos os elementos necessários para a criação de uma boa interface gráfica.

Estas interfaces têm como principal objetivo a comunicação entre o jogador e o videojogo. Apesar disso, muitas das vezes, um bom videojogo perde por ter uma interface inadequada, tornando todas as ações do jogador mais complicadas do que deveriam ser, fazendo com que este último perca o interesse no videojogo. De forma a evitar este problema, deve-se sempre olhar para a interface como uma componente extremamente importante do videojogo, e não como algo extra. Para além disso, muitas das vezes a interface é a primeira coisa que o jogador vê, e se esta não for apelativa e responsiva, muitos jogadores deixam de ter vontade de jogar o videojogo.

De forma a não cometer os erros acima descritos, serão inicialmente abordados os tipos de interface mais utilizados nos videojogos de hoje. De seguida, serão expostas as principais regras a que estas devem obedecer, de forma a criar algo que esteja dentro dos padrões existentes. Finalmente, será criada a janela base, quer para os menus, quer para *In-game*, de forma a manter uma consistência sólida entre todas as componentes do videojogo.

4.1 Tipos de Interfaces

Existem imensas formas de criar uma interface para um videojogo, sendo que algumas tem como grande foco a imersão do jogador, outras a facilidade de uso. Nenhuma destas formas está errada, pois diferentes pessoas têm diferentes gostos, e nunca se consegue agradar a todos os jogadores. Por essa razão, deve-se sempre fazer uma escolha baseada nos estudos realizados por outras empresas presentes na área da criação de videojogos, pois estas já contam com muitos mais anos de experiência.

Apesar de existirem imensos tipos de interface [65], como a Interface através de linha de comandos (CLI), os videojogos optam quase sempre por uma Interface Gráfica com o Utilizador (GUI), pois estas acabam por ser as que trazem mais vantagens entre a relação

jogador/criador, como o facto do jogador não ser obrigador a decorar comandos, ou não serem apelativas. Desta forma, os tipos de interfaces em foco nesta dissertação serão unicamente as do tipo GUI.

Para além de todos estes pontos, outro fator a ter em conta na criação de uma Interface com o jogador é o tipo de controlos suportados pelo videojogo, pois o jogador deve sentir-se à vontade a navegar entre os menus, independentemente do controlador escolhido.

De forma a fazer uma análise mais precisa, serão inicialmente analisados os tipos de interface mais comuns na navegação dos menus dos videojogos, e numa secção posterior, os tipos de interface utilizados no decorrer dos videojogos. Esperando desta forma encontrar uma solução prática para os menus do videojogo a ser criado no âmbito desta Dissertação.

4.1.1 Menus

Para os menus, foram analisadas várias formas de interação com o jogador, visto que não existe um tipo/forma perfeita. De todas as formas, foram selecionadas as quatro mais comuns nos videojogos dos dias de hoje. Nos menus de exemplo criados, todos contém botões, para que o jogador perceba mais facilmente a ação que está prestes a iniciar. Em vez de botões, estes menus poderiam ter imagens relacionas com as ações, ou qualquer outra forma de apresentação, mas esses detalhes não são tidos em conta, pois nesta Dissertação, o ponto que se pretende avaliar é a forma de navegação preferida pelos jogadores.

A primeira forma, **Botões + Rato** (Figura 10), consiste na utilização do Rato, ou do Analógico (no caso do *Gamepad*), para mover um ponto no ecrã (ponteiro do Rato, mira, ou algo do género) de forma a selecionar a ação pretendida. Desta forma os jogadores conseguem chegar a todos os menus duma forma rápida e eficaz, na maioria das vezes. Esta forma costuma ser muito utilizada em videojogos para a plataforma PC, pois corresponde à utilização normal de um Rato num dos vários SO.

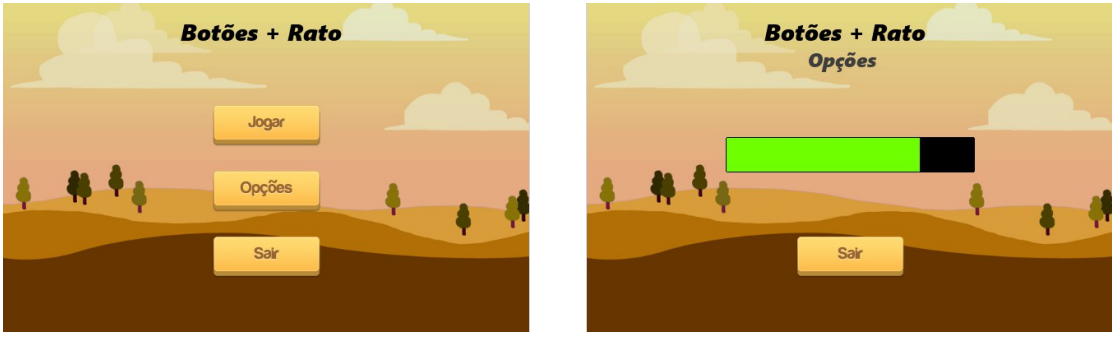


Figura 10 - Menu Botões + Rato

Na segunda forma, **Botões + Teclas (setas)** (Figura 11), o jogador deverá utilizar as setas do teclado, ou o Analógico e/ou *D-pad*, para navegar entre os menus. Neste caso, a ação que o

jogador tem selecionada tem sempre algum tipo de destaque. Neste caso, o destaque é a linha, a inversão das cores do gradiente nos botões e o contorno na barra de volume.

Esta forma de interação por parte do utilizador é muito utilizada em todas as plataformas, apesar que muitas das vezes acaba por requerer mais passos para fazer uma tarefa do que a forma número um. Apesar disso, é um padrão muito utilizado pela indústria dos videojogos.

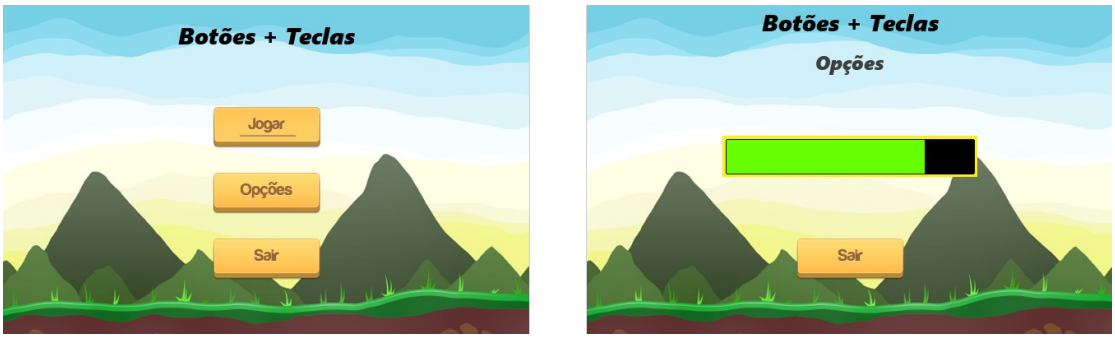


Figura 11 - Menu Botões + Teclas (setas)

A terceira forma, **Botões + Teclas (Mapeadas)** (Figura 12), tem como principal conceito o mapeamento de diferentes teclas, ou botões do *Gamepad*, para que cada um dispute uma ação. Desta forma o jogador só precisa de premir a tecla certa e a ação é disputada imediatamente. Todas as ações que o utilizador pretende tomar acabam por requerer um menor número de passos por parte do mesmo, pois estão à distância de um botão/tecla.

Apesar disso, pode gerar dois tipos de problemas. O primeiro problema seria no caso de ter muitos menus na mesma janela, pois pode tonar todo o processo mais confuso, ou no caso do *Gamepad*, até não existirem botões suficientes. O segundo problema ocorre quando o jogador não está muito habituado ao controlador que está a usar, e neste caso pode demorar mais tempo a realizar as ações, visto que tem de procurar pelos botões/teclas que lhe são apresentados. Devido a estes problemas não existem muitos videojogos que utilizem esta forma de interação.

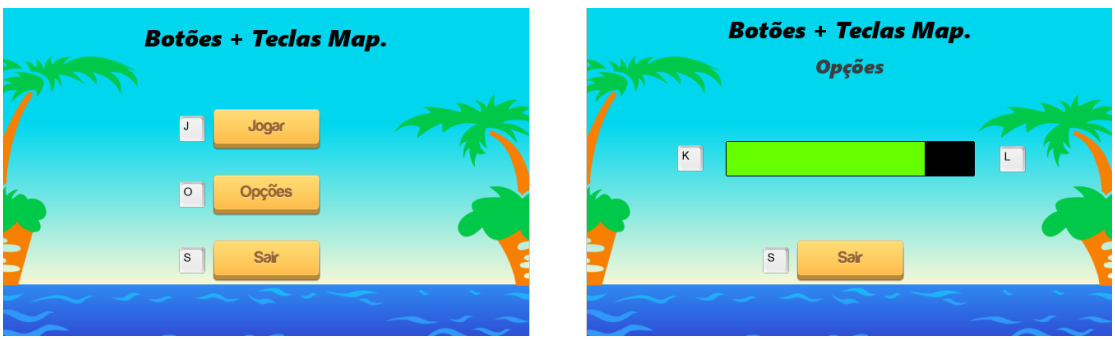


Figura 12 - Menu Botões + Teclas (Mapeadas)

Finalmente, a última forma, ***In-game*** (Figura 13), tem uma maior imersão para os jogadores, pois os menus são integrados no videojogo. Neste exemplo o jogador controla um personagem, que naturalmente seria o personagem a utilizar durante o videojogo em si, que se movimenta na horizontal. Para disputar uma ação, o jogador deve movimentar o personagem para que este fique debaixo de um botão, e clicar na tecla/botão de ação.

Este tipo de menus acaba por agradar a imensos utilizadores, pois mal entram no videojogo, estes estão a jogar, e não a passar por um número de passos para entrar no videojogo. Apesar de tudo, esta forma de interação pode não ser a mais indicada no caso do videojogo conter muitos menus, pois apesar de ser uma forma imersiva, também é a forma mais lenta de aceder ao menus, muito devido às regras do videojogo estarem presentes na navegação dos menus.

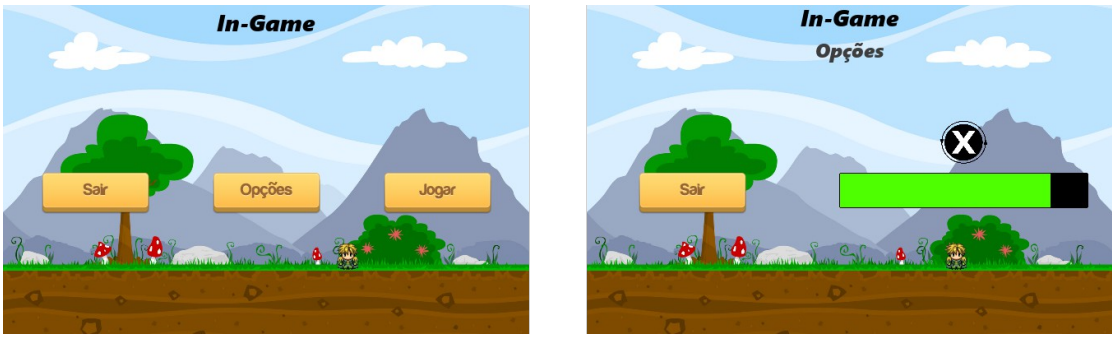


Figura 13 - Menu *In-game*

Apesar de existirem todas estas formas de interagir com o jogador, não se pode escolher uma como a mais "certa" para um videojogo, pois são todas funcionais e utilizadas por criadores de videojogos.

De forma a facilitar a escolha do tipo de interface a ser criada para o protótipo a desenvolver foi realizado um pequeno questionário de forma a que diferentes jogadores, uns mais casuais do que outros, pudessem avaliar os diferentes menus, e assim perceber qual a forma mais apreciada pelos mesmos.

4.1.1.1 Estudo Realizado

Como referido na secção acima, foi realizado um pequeno questionário de forma a perceber qual seria a melhor forma de interação com os jogadores. Este questionário encontra-se disponível no Anexo B.

Neste questionário foram disponibilizados os quatro tipos de menus acima descritos. As pessoas questionadas teriam de seguir um número de passos, passos esses disponíveis numa das partes introdutórias do questionário, e realizar estes passos em todos os menus. Desta forma, conseguiam ter uma noção das diferentes formas de navegação presentes em cada

um dos menus, e desta maneira perceber as dificuldades que poderiam existir nos diferentes menus.

Depois de testarem todos os menus, teriam ainda de responder a cinco perguntas e dar uma nota pessoal (avaliando os diferentes campos com valores entre 1 a 5, sendo 1 o pior valor e 5 o melhor), a cada um dos menus. Para além disso, poderiam ainda escrever alguns comentários que achassem importantes para a avaliação deste menus.

Antes de serem expostos e analisados os diferentes resultados, é de notar que 20 jogadores portugueses de ambos os sexos, tanto casuais como habituais, entre os 16 e 50 anos, foram questionados durante o mês de Junho de 2014. Os questionários foram distribuídos online e mais tarde recolhidos da mesma forma.

Todos os resultados serão expostos em forma de percentagem, para tornar a análise mais direta.

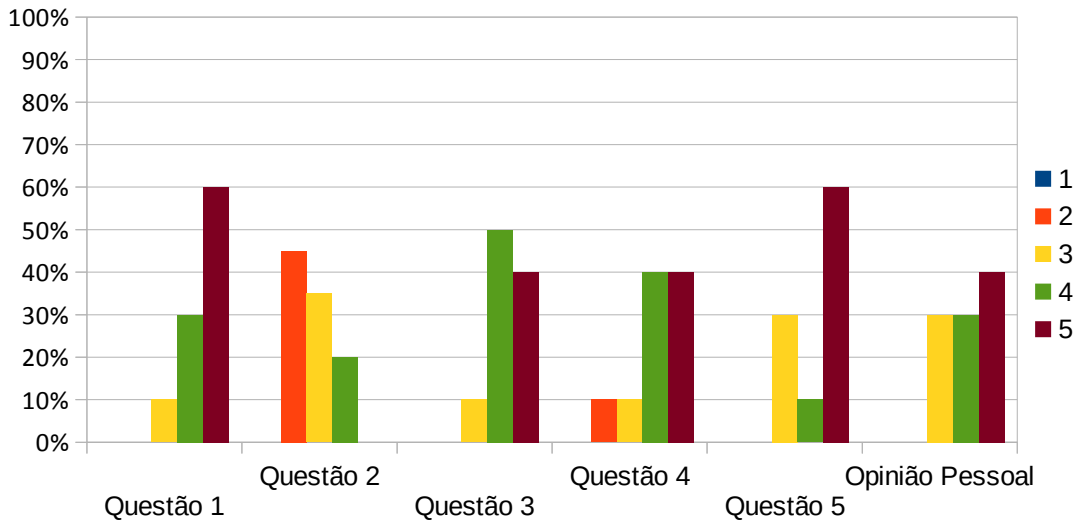


Gráfico 3 - Respostas do questionário referentes ao Menu 1

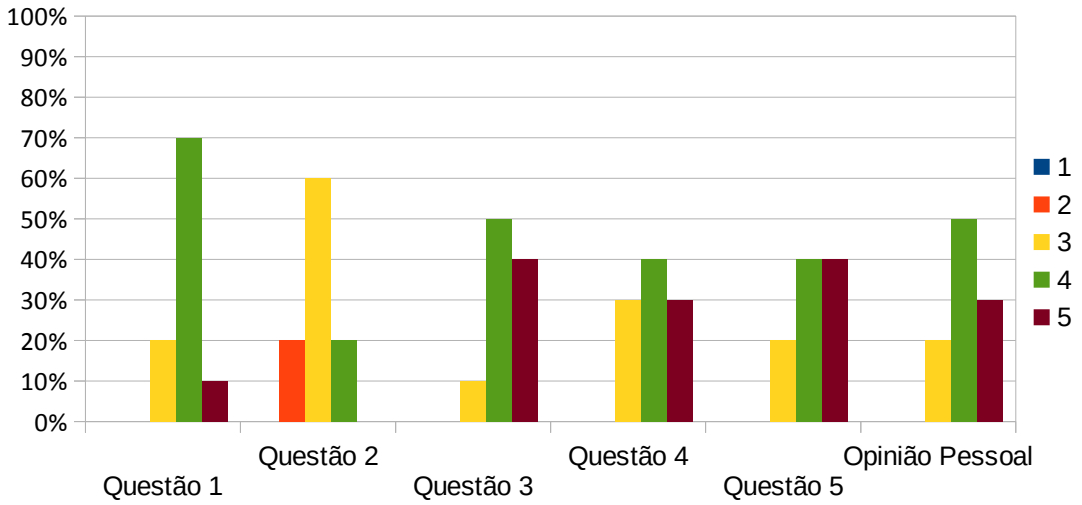


Gráfico 4 - Respostas do questionário referentes ao Menu 2

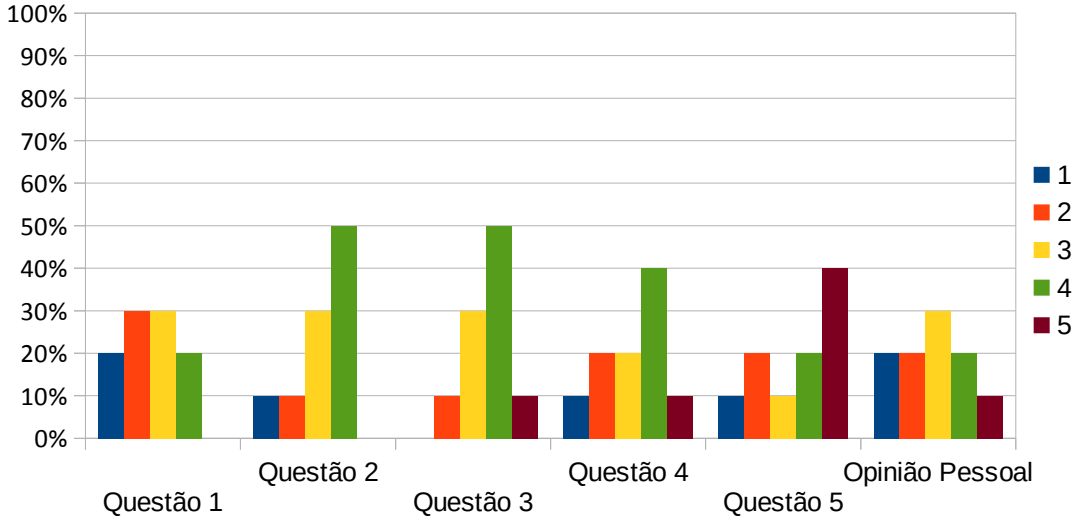


Gráfico 5 - Respostas do questionário referentes ao Menu 3

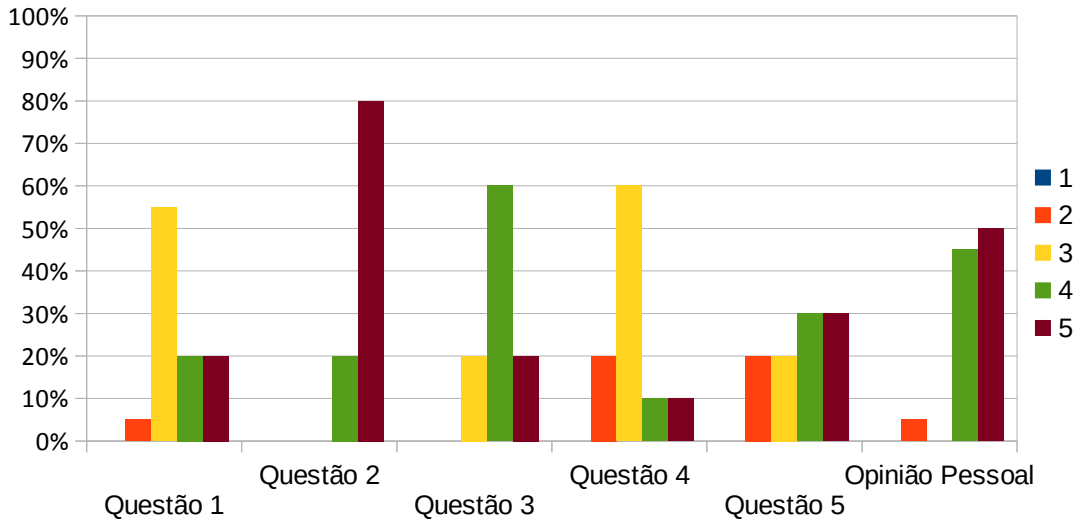


Gráfico 6 - Respostas do questionário referentes ao Menu 4

Como se pode ver nos gráficos acima (Gráficos 3, 4, 5 e 6), o Menu menos apreciado é sem dúvida o "Menu 3". Algumas pessoas disseram que este menu, para além de ser difícil de navegar, acabava por ser pouco intuitivo, e que por isso não era uma boa escolha.

Quanto aos restantes menus, todos eles acabam por ser bem aceites pelas pessoas questionadas. Estas acham que os Menus 1 e 2, acabam por ser os mais normais, mas também os mais estáveis, simples de usar e com um acesso bastante fácil a todas as ações, sendo que o Menu 1 acaba por ter uma preferência mais elevada, quando tido em conta as preferências pessoais de cada um.

Quanto ao Menu 4, apesar de ser visto como um menu que poderá gerar alguma confusão por parte do jogador, e também o menu mais lento de navegar, acaba por ser o mais apreciado pela grande quantidade dos questionados, pois pensam que este é o mais divertido e original, tornando a tarefa de navegar nos menus menos monótona.

Com este pequeno estudo, consegue-se perceber que uma forma de interação baseada nos Menus 1, 2 ou 4 não será uma má forma de interação, visto serem formas que agradam à maior parte das pessoas.

4.1.2 In-Game

As Interfaces existentes dentro do videojogo (*In-Game*) não podem ser vistas como todos os outros tipos de Interface com o Utilizador (UI), pois neste caso temos um novo elemento misturado com todos os outros, sendo este: a ficção (onde se encontra inserido o personagem do jogador, e outros elementos pertencestes ao mundo criado para o videojogo).

Graças a este novo elemento (ficção), as UI dos videojogos acabam por ser divididas em quatro grandes grupos [66], sendo *estes: Diagetic, Non-diegetic, Spatial, Meta.*

Para se perceber em que grupo se encontra uma interface, deve-se ter duas perguntas em mente. "A representação do elemento está dentro do mundo do videojogo?" e "A representação do elemento existe no mundo/história do videojogo?". Conforme as respostas dadas, consegue-se perceber rapidamente a que grupo pertence a UI (Figura 14).

Figura 14 - Diagrama dos diferentes grupos de UI nos videojogos

De forma a explicar melhor em que consiste cada um destes grupos, será criada uma imagem (Figura 15) que poderia ser uma *screenshot,* sem qualquer elemento da UI*,* de um possível videojogo, e em cada uma das secções abaixo será explicado duma forma muito simples e concisa como são definidos os diferentes grupos, e será adicionado um, ou vários, elementos à imagem, de forma a criar uma UI do tipo do grupo que está a ser discutido, demonstrando assim, duma forma mais clara, as diferenças entre cada grupo.

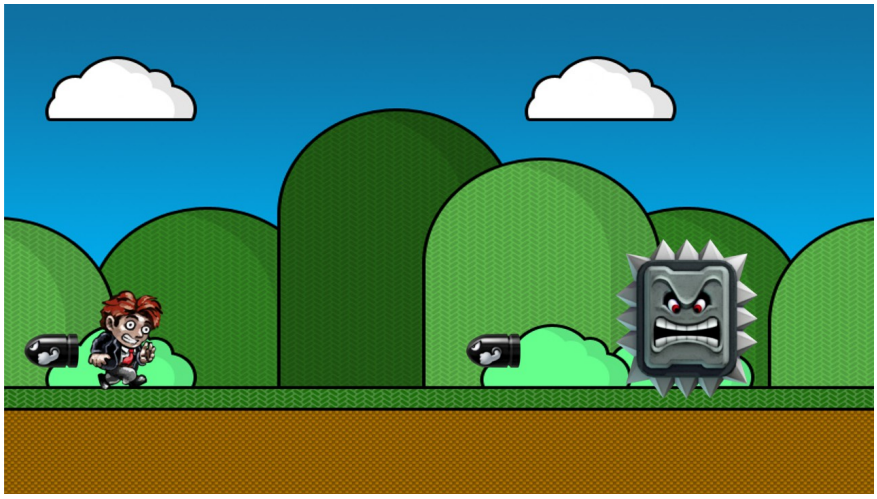


Figura 15 - *Screenshot* base (jogador sofreu dano)

4.1.2.1 *Diagetic*

Elementos do grupo *Diagetic* têm como principal objetivo passar informação ao jogador, através de pequenas pistas, assim evitando que o jogador se distraia da narrativa do videojogo. Muitas das vezes estas pistas são algo que acontece ao personagem principal, ou a um dos personagens controlados pelo computador (NPC). Estes elementos tornam assim toda a experiência do jogador mais imersiva.

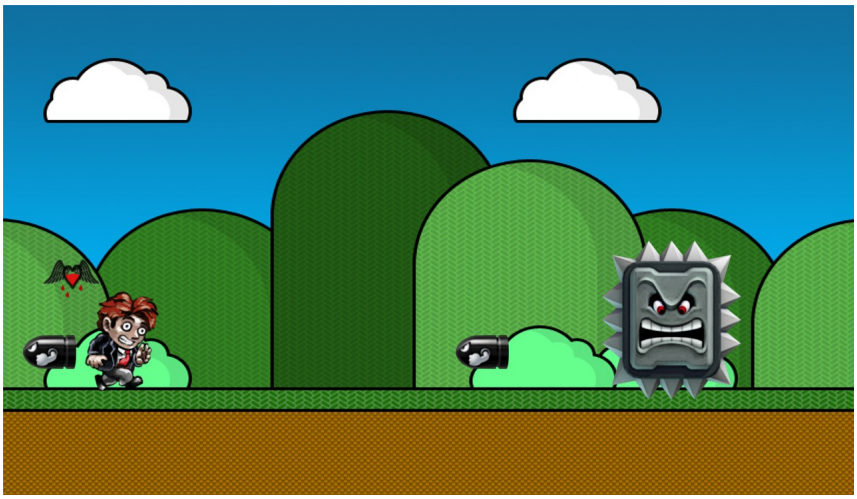


Figura 16 - *Screenshot* Diagetic (Adicionado um companheiro ao videojogo)

Para demonstrar o uso deste grupo de elementos, foi adicionado à *screenshot* base um novo companheiro, tendo este a forma de um coração com asas (Figura 16). Este NPC teria como principal objetivo a representação da vida do jogador. Para além disso poderia ainda ter mais funções. Este poderia ter uma voz e interagir com o jogador, envolvendo-o assim muito mais com o jogador.

4.1.2.2 *Non-Diagetic*

Estes elementos podem ser qualquer coisa, pois não existem dentro do mundo nem da história do videojogo. Apesar disso, muitas das vezes, estes elementos acabam por estar relacionados com o estilo de arte presente no videojogo, de forma a não parecerem algo completamente à parte do mesmo.

Quase todos os videojogos do género *Massive Multiplayer Online Role Playing Game* (MMORPG) usam este tipo de elementos, pois possuem um elevado número de elementos presentes no ecrã ao mesmo tempo. Um dos pontos mais fortes destes elementos, é que devido a sua abstração do mundo/história do videojogo, podem ser reposicionados muito facilmente, dando assim a possibilidade ao jogador de os dispor da forma que mais lhe agrada.

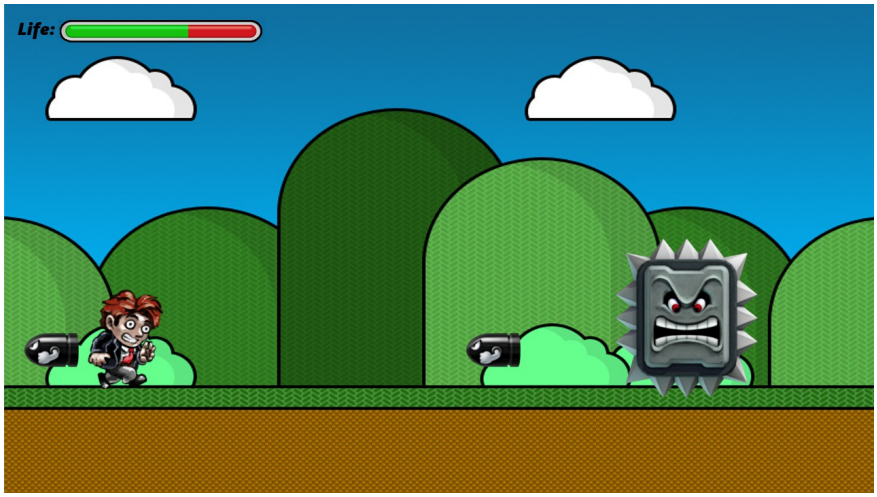


Figura 17 - *Screenshot* Non-Diagetic (Adicionada barra de vida)

Neste caso (Figura 17), foi adicionado ao *screenshot* base uma barra que representa a vida do jogador, para que este consiga facilmente perceber a vida que resta ao seu personagem.

4.1.2.3 *Spatial*

Os elementos do grupo *Spatial*, são elementos que fazem parte do mundo do videojogo, mas que são completamente ignorados pela história e pelos personagens do videojogo.

A informação que estes elementos pretendem passar ao jogador é sempre apresentada em locais muito precisos, para que este perceba facilmente essa mesma informação.

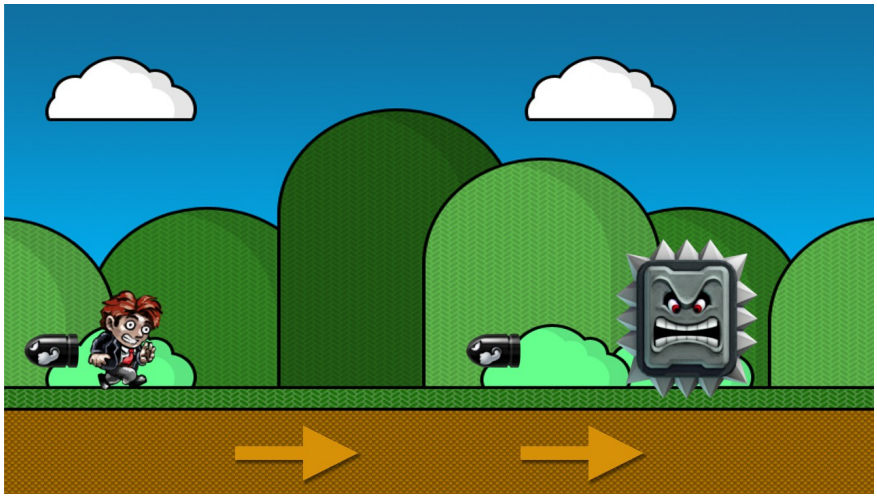


Figura 18 - Screenshor Spatial (Adicionado setas que indicam a direção a seguir)

Na Figura 18, pode-se ver que foram adicionadas umas setas à *screenshot* base. Estas setas têm como função indicar a direção que o jogar deve seguir para completar o nível. Este elemento, apesar de estar presente no mundo do videojogo (debaixo da terra), é completamente ignorado pelos personagens.

4.1.2.4 *Meta*

Finalmente, os elementos Meta, são elementos que, apesar de serem expressados como parte da narrativa do videojogo, não pertencem ao mundo do mesmo. Isto é, estes elementos são muitas vezes apresentados no ecrã.

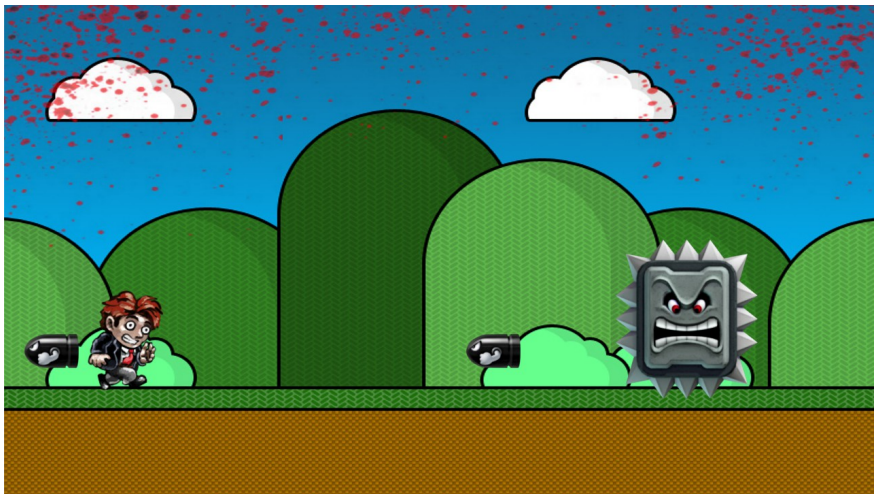


Figura 19 - *Screenshot* Meta (Adicionado efeito sangue para demonstrar dano)

Grande parte das vezes estes efeitos são utilizados de forma a criar uma interação mais próxima com o jogador. Pode-se, por exemplo, criar um efeito de vidro partido, dando a ideia que foi o ecrã do jogador se partiu.

Neste caso (Figura 19), foi adicionado um efeito de salpicos de sangue. Este efeito pretende avisar o jogador que o personagem sofreu algum tipo de dano e que não está completamente bem. Para isso são apresentados alguns salpicos no ecrã, ignorando o mundo do videojogo, mas transmitindo uma ideia de perigo e dor ao jogador.

4.2 Interface Final

Depois da análise realizada a todas as formas de interfaces com o utilizador estudadas neste capítulo, foram escolhidos as seguintes para serem implementadas no videojogo a ser desenvolvido:

Menus: *In-Game*.

Esta forma foi escolhida pois, apesar de dar algum trabalho extra aos jogadores, é uma das interfaces mais apreciadas pelos mesmos. E tendo em conta que o produto a ser criado é um produto de entretenimento, este aspeto da apreciação por parte do jogador torna-se extremamente importante. O grande inconveniente deste tipo de menus é uma maior probabilidade de erros por parte do jogador. De forma a combater este problema, as várias opções disponibilizadas ao utilizador deverão ser bem explicadas e muito claras, evitando assim muita da confusão que poderá existir à volta deste problema. Esta forma de interação adapta-se bastante bem quer ao *Gamepad,* quer ao Rato+Teclado.

***In-game*:** *Diagetic*.

Foi selecionada a interface do tipo *Diagetic,* pois é a que melhor se enquadra quando se pretende que o jogador se sinta imerso na aventura a ser criada. Como o videojogo que se deseja criar, pertence ao género Plataforma, acaba também por ter uma interface com poucos elementos, o que facilita a aplicação deste tipo de interface. Para além de todos estes pontos, os menus do videojogos também fazem parte do "mundo" do videojogo,pelo que esta forma ainda faz mais sentido de ser aplicada.

4.3 Conclusão

Como se pôde perceber durante este capítulo, a construção de uma Interface com o Jogador acaba por ser uma das tarefas mais difíceis durante o processo de criação de um videojogo. Como não existe uma maneira certa, mas sim várias, e independentemente da que seja escolhida, esta não irá ao encontro do que todos os jogadores mais gostam, acabam por ser geradas imensas interfaces ao longo da vida de um videojogo.

Para facilitar a compreensão de cada um destes tipos, foram apresentados neste capítulo os tipos de interface mais utilizados quer em menus, quer durante o decorrer do videojogo. Desta forma, foi mais fácil de compreender os pontos fortes e fracos que cada uma traria para o protótipo a desenvolver. Para além disso, foi ainda feito um questionário a diferentes pessoas sobre o tipo de interação que mais lhes agrada, obtendo assim uma nova perspetiva sobre o assunto.

Finalmente, depois de todas as opções terem sido tidas em conta, de todas foram escolhidas as que melhor encaixariam no videojogo a ser desenvolvido.

5 Implementação da Framework Modular

Depois de todos os requisitos terem sido levantados, e as várias ferramentas estudadas, neste capítulo serão descritos alguns dos passos mais importantes na implementação da *Framework* Modular que irá suportar todo o videojogo.

Numa primeira fase será apresentado e discutido o esqueleto da *Framework*, reforçando mais uma vez, duma forma simples, todas as ligações que existem entre as diferentes componentes, possibilitando assim o bom funcionamento da *Framework*.

Depois, serão expostos alguns dos pontos mais importantes da construção dos componentes da *Framework*, implementados pelo autor tendo sempre em vista a elevada manutenibilidade da *Framework*.

É ainda de realçar que a *Framework* foi desenvolvida por duas pessoas, sendo que as partes não discutidas nesta Dissertação podem ser encontradas na Dissertação de Ricardo Moreira [62].

5.1 Esqueleto da Framework

Como já discutido no capítulo 3.2 (Figura 8), o esqueleto da F*ramework* está dividido em cinco grandes componentes. Sendo que, destes, só quatro são desenvolvidos na parte da *Framework*: *Game*, *Helper*, *Math* e *Libraries*. A componente de *Game Logic* só será implementada no videojogo em si, pois é esta componente que muda conforme o videojogo. Todas as restantes componentes são comuns para qualquer videojogo desenvolvido nesta *Framework* Modular.

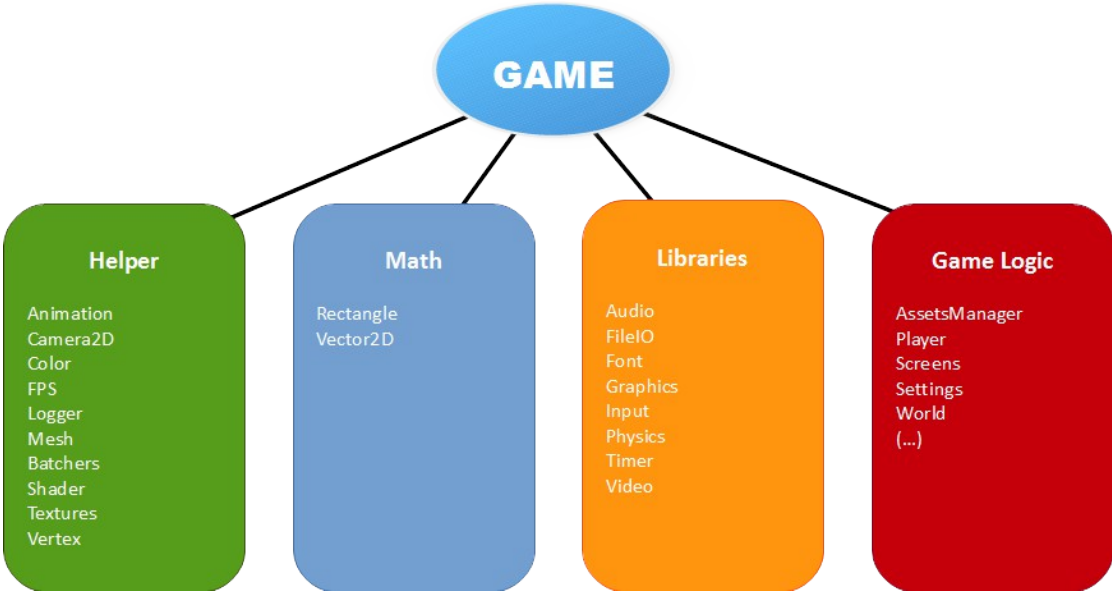


Figura 20 - Esqueleto da *Framework* Modular

Na Figura 20, pode-se ver o esqueleto completo da *Framework* desenvolvida para esta Dissertação.

Tal como já fora referido no capítulo 3.2, toda esta *Framework* foi criada com vista a ter uma elevada manutenibilidade, separando assim todas as partes umas das outras. Com isto quer-se dizer que, se no futuro fosse necessário trocar a biblioteca de Áudio, neste caso FMOD, por outra qualquer, as únicas classes que teriam de ser alteradas seriam as classes de Áudio, que se encontram inseridas na componente *Libraries.* Pois todos estes componentes foram desenvolvidos tendo as normas de encapsulamento sempre em mente, de forma a facilitar futuras modificações, tornando assim toda a *Framework* Modular mais flexível.

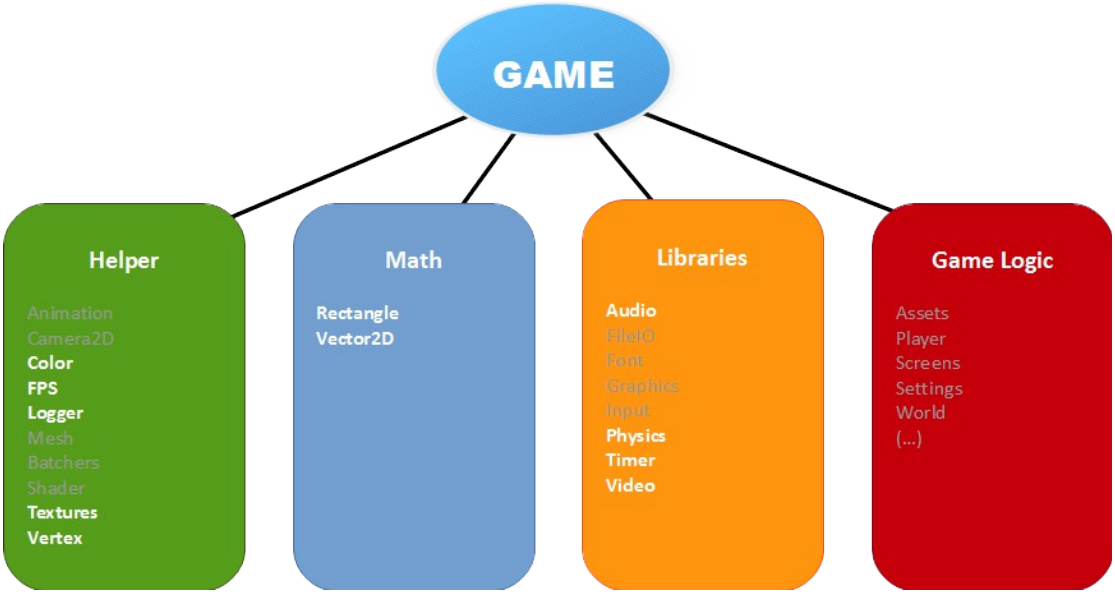


Figura 21 - Esqueleto *Framework* Modular (divisão do trabalho)

A Figura 21, foi feita de forma a demonstrar, com maior simplicidade, as diferentes partes desenvolvidas para este projeto. Todas as partes realçadas foram desenvolvidas no âmbito desta Dissertação pelo autor, enquanto que todas as outras partes foram desenvolvida pelo aluno Ricardo Moreira [62], ou no caso da *Game Logic,* só será desenvolvida e documentada no próximo capítulo, que se destina à Implementação do Videojogo.

5.2 Implementação

Nesta secção serão referidos alguns dos aspetos mais interessantes acerca da implementação da *Framework*, e alguns dos problemas encontrados no decorrer do desenvolvimento da mesma.

5.2.1 *Helpers*

Na componente *Helpers* foram implementadas cinco classes diferentes: *Color*, FPS, *Logger*, *Textures* e *Vertex*.

A classe **Color** tem como principal função guardar cores que serão usadas mais tarde para colorir retângulos, ou até texturas. Esta classe pode receber uma cor quer pelo seu código RGB, quer por valores *float* (entre 1.0f e 0.0f). Sendo que, internamente, já contém mais de 140 cores previamente definidas, e à disposição do programador.

A classe **FPS**, é uma classe de *Debug,* e tem como principal objetivo o cálculo do número de *Frames* apresentados por segundo.

A classe ***Logger,*** tal como a classe FPS, também é uma classe de *Debug*. Esta tem como fim a apresentação de mensagens de texto na consola. Desta forma, consegue-se perceber duma maneira simples, o que pode estar a acontecer com certa variável, ou ser alertado no caso de acontecer um erro durante o tempo de execução. A mensagem a ser apresentada na consola pode ser totalmente personalizada, e conforme a sua gravidade esta muda de cor. No caso de ser reportado um erro, este aparecerá a vermelho na consola, por exemplo.

A classe ***Textures*** serve para fazer a gestão de todos os componentes das diferentes texturas do videojogo (filtros, tamanho, carregamento para a memória, etc.). Para além disso, são ainda suportados Atlas de texturas. Estes Atlas normalmente possuem um número elevado de texturas numa só imagem, e de forma a facilitar a utilização dessas mesmas texturas na construção do videojogo, a classe guarda em objetos as coordenadas e os tamanhos das diferentes texturas. Desta forma, o criador do videojogo só tem de dizer qual a textura e a área da mesma que pretende apresentar no ecrã.

Finalmente, a classe ***Vertex***, tem como objetivo a gestão de todos os vértices necessários para a criação de formas geométricas e apresentação de texturas no ecrã.

5.2.2 *Math*

Na componente **Math** existem duas classes: *Rectangle* e Vector2D.

A classe **Vector2D** tem a finalidade de criar, e gerir, vetores com coordenadas nos eixos X e Y. Internamente a classe está construída de forma a suportar todo o tipo de operações, como por exemplo, soma de vetores, normalização, projeções, distância entre vetores, entre outras.

A classe ***Rectangle*** tem como principal função a criação e manipulação de retângulos. Para além das ações mais simples, como obter a altura do retângulo, também é possível com esta classe perceber se um retângulo interseta outro, ou um ponto no espaço, que pode ser o clique do rato. Este tipo de ações torna-se bastante útil para o funcionamento básico de *hitboxes,* que é muito utilizado na criação de menus.

5.2.3 *Timer*

O ***Timer*** é responsável pela gestão do tempo de jogo, ou seja, tem como função perceber quanto tempo passou entre a última *frame* apresentada e a próxima *frame*. Este processo é fundamental para que todo o videojogo corra direito, caso contrário o som poderia não estar de acordo com a imagem, ou as áreas "físicas" dos objetos poderiam estar em sítios diferentes dos apresentados no ecrã.

Para além disto, o *Timer* é ainda responsável pela gestão de FPS do videojogo, sendo que por padrão, está programado para apresentar, no máximo, 60 FPS. Este valor pode ser alterado pelo criador do videojogo, de forma a ter uma *frame rate* estável no seu videojogo. Desta forma, para computadores mais potentes, onde o videojogo poderia correr a 120 FPS, o *Timer* trata de gerir o tempo extra que tem e dividi-lo de forma a manter os FPS máximos pretendidos, em vez dos 120 FPS. Este foi um dos problemas iniciais que ocorreram durante o desenvolvimento da *Framework*. Como não existiam muitas imagens no ecrã, era bastante fácil obter mais de 1000 FPS, o que tornava os valores dos métodos de atualização pequenos demais para serem devidamente processados pelas diferentes bibliotecas.

5.2.4 *Audio*

Para o processamento de áudio foi utilizada a biblioteca **FMOD** [46]. O FMOD apresentou-se como uma das melhores opções pois, para além de ser muito utilizada pelos grandes criadores de videojogos espalhados pelo mundo, suporta praticamente todos os sistemas operativos, lê todos os formatos de ficheiros que eram pretendidos (.ogg), tem uma boa documentação e é agora gratuito no caso do desenvolvimento de videojogos independentes.

Através das classes de *Audio* passou a ser bastante fácil gerir todas as músicas e sons do videojogo. Para carregar um novo ficheiro de som, o programador só precisa de dizer se se

trata de um som, algo curto, ou uma música, definir o volume, e apontar para onde o ficheiro se encontra. Todos os outros passos são internos e o programador não se tem de preocupar com nada.

O programador pode depois manipular o som como bem entender. Pode ligá-lo, ou desligá-lo, pode pôr este num ciclo infinito, ou ajustar o volume do mesmo em tempo real. Para além disso, pode ainda manipular cada som individualmente, ou pode escolher manipular todos os sons, ou músicas, ao mesmo tempo. Quer-se com isto dizer que o programador, a dado momento pode, por exemplo, desligar todas as músicas do videojogo com uma simples linha de código. Este foi um dos métodos mais importantes da classe de áudio, a nível de controlo do áudio, pois no caso de estarem a tocar muitos sons ao mesmo tempo, e se pretendesse desligar todos ao mesmo tempo, antes de ter este método, tinha de se desligar um som de cada vez.

Para além disso, se todos os sons forem desligados através do método que controla todos os sons, novos sons que deveriam tocar são ignorados. Logo, o programador não tem de fazer testes para perceber se deve ou não tocar aquele som.

5.2.5 *Physics*

Para gerir todas as componentes físicas do videojogo, foi escolhida a biblioteca **Box2D** [48]. Esta biblioteca foi escolhida pois tem uma integração bastante fácil e simples, e já foi bastante utilizada em videojogos. Ademais, esta biblioteca tem uma documentação bastante boa, e uma comunidade ativa, o que constitui um grande ponto a seu favor. Bibliotecas como Bullet Physics Library foram postas de parte logo desde início, pois apesar de serem bibliotecas muito boas, são bibliotecas 3D, e utilizar uma destas bibliotecas só iria acrescentar complexidade desnecessária ao código a ser criado.

As classes *Physics* são bastantes simples de operar. O programador só terá de inicializar o seu "mundo" com a gravidade pretendida (por padrão é -9,81ms^{-2} no eixo dos Y, força gravitacional do planeta Terra) e depois disso só terá de adicionar os objetos que pretende que a cena do videojogo contenha.

Os objetos são compostos por uma posição central inicial e por um tipo, sendo que este tipo pode ser dinâmico (reage a forças), ou estático (não reage a forças). Depois de ter o objeto criado, o programador pode adicionar formas de contacto, círculos ou retângulos, e quando este colidem com alguma coisa, uma nova colisão é registada. Quando o programador achar conveniente, pode pedir que lhe seja enviada uma lista de todas as colisões que ainda não foram tratadas.

Para além de poder adicionar formas de contacto normais, pode ainda adicionar formas de contacto do tipo sensor. Estas formas registam colisões como uma forma normal, mas não têm um corpo sólido. Este tipo de formas é muito utilizado para perceber, por exemplo, se um *avatar* se encontra no chão e pode saltar ou se colidiu com uma parede. No caso de estar

no chão o sensor (devidamente aplicado na área dos pés) deverá já ter previamente colidido com a mesma área.

Finalmente, de forma a criar movimento nos diferentes objetos, o programador pode aplicar forças, quer no eixo do X como do Y, a qualquer objeto do tipo dinâmico.

Um dos maiores problemas que ocorreram no desenvolvimento destas classes foi a má utilização das unidades. O que acontecia era que apesar de o videojogo ser todo em píxeis, o Box2D trabalha em metros, o que fazia com que uma cena com altura de 720 píxeis equivalesse a 720 metros. Isto tornava todas as forças existentes no mundo mínimas, ou até impercetíveis. Foram então feitas alterações de modo a que, internamente, as classes da parte *Physics* trabalhassem com metros, mas para o programador está tudo em píxeis, logo este último não tem de se preocupar com nada. Caso este pretenda fazer algo mais complexo pode alterar os valores de conversão internos, sendo que o valor padrão é: cada 100 píxeis equivalem a 1 metro.

5.2.6 *Video*

A **Theora Playback Library** [50], foi a biblioteca selecionada para apresentação de vídeos durante o videojogo. Na construção desta *Framework,* um dos pontos importantes era a utilização de bibliotecas gratuitas para o desenvolvimento de videojogos independentes. E percebeu-se desde cedo que grande parte dos vídeos usam *codecs* pagos, como o caso do H264, o que seria um problema para o ponto de manter a *Framework* gratuita.

Depois de bastante pesquisa, foram encontrados os *codecs* da Xiph.Org Foundation (ogg, vorbis, theora, etc...). Estes conseguiam fazer tudo que era pretendido para a *Framework* e eram gratuitos. Logo tornaram-se na escolha óbvia para este projeto.

De forma a não perder imenso tempo a criar um descodificador para vídeos .ogg foi encontrada a biblioteca Theora Playback Library, que para além de corresponder ao que se pretendia, já tinha sido utilizada por diversos videojogos com sucesso no mercado, tal como Octodad.

Apesar da biblioteca ser perfeita para o que se pretendia, esta utilizava uma combinação de OpenGL e GLUT para a representação gráfica do vídeo, e a biblioteca OpenAL era utilizada para reprodução de áudio. Foram então feitas todas as transformações necessárias, para que todos os processos envolvessem as mesmas bibliotecas que eram utilizadas na *Framework*. Todo o conteúdo pertencente ao GLUT foi alterado de forma a só utilizar OpenGL para a apresentação gráfica do vídeo, e foi ainda criado um adaptador em FMOD para o áudio do vídeo, de forma a substituir todo o código OpenAL.

5.2.7 Outros componentes

De forma a rematar este subcapítulo, serão aqui referidas as outras bibliotecas utilizadas na *Framework* Modular, desenvolvidas por Ricardo Moreira [62].

A **FileIO** utiliza exclusivamente código C++, e suporta a leitura e escrita de ficheiros.

A **Font** utiliza a biblioteca Freetype2 [49]. Esta classe tem como função apresentar texto no ecrã e suporta fonts do tipo .ttf e .otf.

Os **Graphics** são compostos por duas bibliotecas, sendo que a janela é gerida através da biblioteca GLFW [47], mas todos os aspetos de renderização são efetuados em OpenGL.

O **Input** é obtido através da biblioteca GLFW, para depois ser tratado e mais tarde processado pela *Game Logic*. Os tipos de controladores suportados são: *GamePad*, Teclado e Rato.

5.3 Conclusão

Neste capítulo foi discutida e apresentada toda a informação relevante sobre a criação e desenvolvimento da *Framework* Modular.

Numa fase inicial, foi criado um esqueleto da *Framework* de forma a perceber como todos os elementos da mesma se ligam e interagem. Depois foi distribuído de forma igual o trabalho pelos dois estudantes que iriam implementar a *Framework*.

Finalmente, foi feita a implementação da *Framework* Modular, que correu como esperado, visto que todos os componentes ficaram funcionais, e a cumprir toda a lista de requisitos previamente realizada no capítulo 3.3.

6 Implementação do Videojogo (Protótipo)

Neste capítulo serão discutidos alguns dos aspetos da implementação do Videojogo (Protótipo). Este Protótipo tem como principal objetivo a demonstração do bom funcionamento da *Framework* Modular previamente implementada.

Numa primeira instância irá ser explicado o *GameFlow* do Protótipo. Desta forma será possível demonstrar e perceber, todas as janelas e menus necessários para o bom funcionamento do Protótipo.

Depois serão explicados os menus existentes no Protótipo e os diferentes controlos de opções disponíveis. De seguida será exposto todo o tipo de iterações que o jogador pode realizar com a sua personagem, e como são compostos os níveis do Videojogo. Mais uma vez, é de referir que a arte e os sons criados e escolhidos para este projeto servem apenas para fins de demonstração; não existe qualquer estudo acerca destes.

Finalmente, é ainda de realçar que em anexo (Anexo A) existe um GDD sobre o Protótipo criado, que contém alguma informação adicional, tal como as personagens.

6.1 *GameFlow*

Com a ajuda do diagrama de *GameFlow,* o criador do videojogo consegue perceber muito mais facilmente, o número de ecrãs que irá ter de criar. Para além disso, também ajuda a compreender todas as ligações que existem entre eles. Percebendo assim todas as ligações necessárias para que um jogador consiga ir de um ponto A até um ponto B.

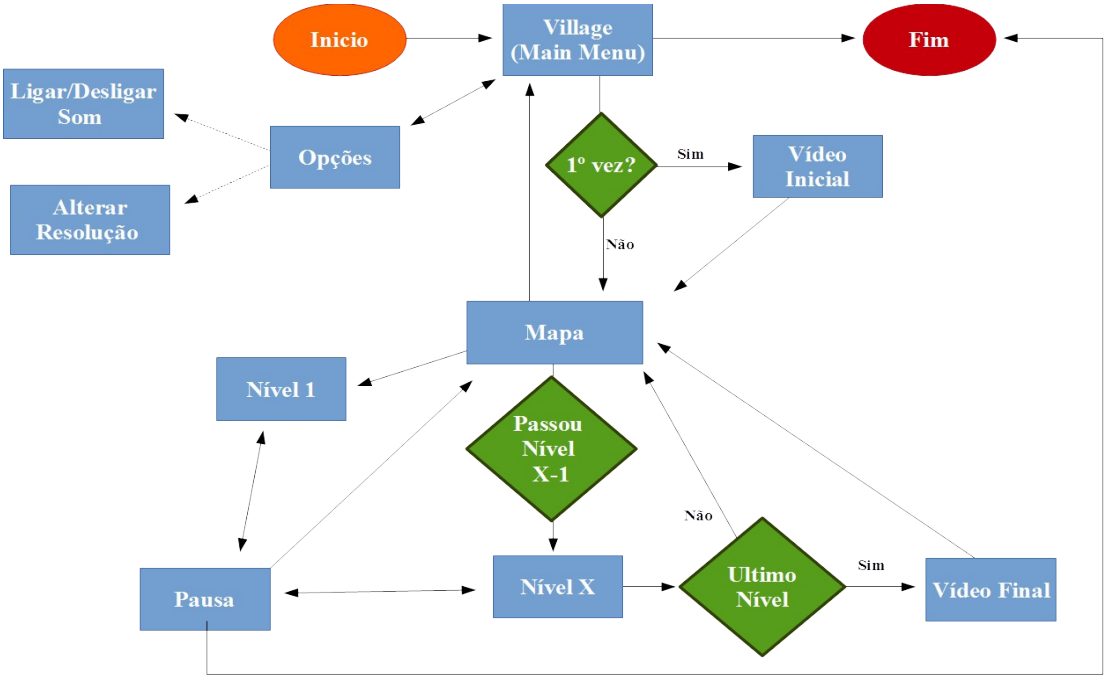


Figura 22 - Diagrama do GameFlow

Quando o jogador inicia o Protótipo, este será enviado para o Main Menu, que neste Protótipo é tratado como um "nível", pois as opções estão embebidas dentro do mundo criado para o Protótipo. Do Main Menu o jogador pode decidir se quer alterar alguma das opções ou começar a jogar.

Se decidir começar a jogar, ser-lhe-á mostrado um vídeo para introduzir a história. A qualquer momento, o vídeo pode ser pausado ou avançado.

Apesar da ação não se encontrar no diagrama, quando o jogador completa um nível, tendo ou não já completado este previamente, o seu progresso é guardado para que os vídeos não sejam novamente apresentados, ou para que este tenha de completar o Protótipo todo sem parar.

Quando o vídeo acaba, o jogador é levado para o mapa. Este é, na realidade, o menu de Seleção de níveis, sendo que todos os níveis presentes no videojogo, incluindo o Main Menu, se encontram disponíveis, ou bloqueados, neste menu. O utilizador começa com os níveis bloqueados, e conforme avança na história, estes vão sendo desbloqueados.

Quando chega ao último nível, o jogador é confrontado com mais um vídeo, de forma a concluir a história do videojogo.

O jogador pode então repetir qualquer nível previamente completado, ou sair do videojogo.

6.2 Menus

O Protótipo só contem um menu “tradicional”, o menu de Pausa. Como definido no Capítulo 4, o Protótipo, todas as opções deveriam fazer parte do mundo do videojogo em vez de serem apresentadas em menus.

O menu de Pausa é a única exceção, pois desta forma, o jogador pode aceder a este menu a qualquer momento (sendo este um dos requisitos funcionais do videojogo), tornando a ação mais confortável e viável para o jogador.

Dentro desse menu, o jogador pode sair do Pause Menu, pode sair do nível e ir diretamente para o Mapa, ou ainda, se preferir, pode sair para o ambiente de trabalho, fechando o Protótipo.

6.3 Controlo das Opções do Videojogo

Tal como já foi referido no subcapítulo anterior, as opções são controladas através de personagens. Personagens estas, que existem no mundo criado para o videojogo, com as quais o jogador pode interagir.

Neste Protótipo o jogador pode controlar duas opções de jogo:

Figura 23 - Personagens das opções

Ligar/Desligar o Som: existe um personagem (baterista, no lado esquerdo na Figura 23) no Protótipo que está por omissão a tocar bateria (representação do som ligado). Quando o jogador interage com o baterista, este adormece e deixa de tocar (som desligado). Interagindo novamente, este começa novamente a tocar a sua bateria e o som volta a tocar.

Alterar resolução: um personagem com uma tabuleta (no lado direito na Figura 23) apresenta a resolução a que o videojogo está a correr. Interagindo com o mesmo altera a resolução do videojogo, em tempo real, para a próxima resolução disponível.

Desta forma, o jogador sente-se muito mais imersivo, pois não tem acesso aos menus tradicionais, mas começa logo a jogar com o seu personagem, mesmo que só esteja a alterar algumas definições de videojogo.

Sempre que as opções são alteradas, as alterações são guardadas. Para isso é utilizada a componente responsável pela gestão de ficheiros da *Framework* (FileIO), gravando e carregando os dados conforme seja necessário.

6.4 Personagem do Jogador

O jogador, durante todo o Protótipo, controla um StickMan. Este StickMan consegue saltar, movimentar-se para os lados e interagir com certos elementos.

Figura 24 - StickMan (da Esq. para Drt.: Ação, Normal, Saltar)

A nível de implementação, o personagem contém vários métodos para gerir os seus movimentos, e ações, conforme os *inputs* recebidos, quer estes provenham de um Teclado, ou de um *Gamepad* (*inputs* geridos pela componente *Input*).

Também existem vários testes geridos pela componente *Physics* para perceber se o jogador se encontra no chão caso pretenda saltar (senão poderia saltar infinitas vezes consecutivas).

O seu salto e movimentos são aplicados através de forças físicas, também geridas pela componente de *Physics.* Isto torna tudo mais simples, sendo que o programador não precisa de se preocupar com o número de píxeis que o personagem tem de subir em determinado espaço de tempo. Todo este processo é gerido pela componente *Physics*.

Para além de tudo isto, existe ainda um método que é chamado sempre que é detetada uma colisão, colisões essas geridas pelo componente *Physics*, para que esta seja tratada conforme o seu tipo (se colidir com um inimigo morre, etc.).

6.5 Níveis

Neste protótipo existem dois tipos de níveis. O primeiro é um nível padrão onde o jogador tem de ultrapassar vários obstáculos, e decifrar alguns puzzles, para poder chegar ao fim do mesmo. Quando chega ao fim do nível este é dado por completo.

Os obstáculos consistem em pequenos inimigos que se movimentam da esquerda para a direita, que podem ser mortos se o jogador saltar para cima destes, e em catos. Caso o jogador entre em contacto com algum destes, é imediatamente morto.

Para além disso existem plataformas para as quais o jogador pode saltar, e caixas que pode empurrar, e que deve usar para ultrapassar obstáculos.

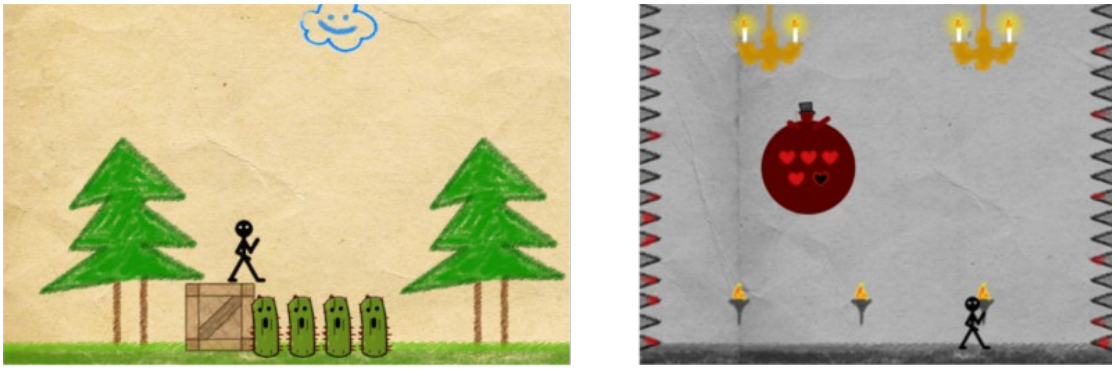


Figura 25 - Nível 1 (Esquerda) e Nível 2 (Direita)

O segundo, é um nível que não contém qualquer tipo de obstáculos, mas contém um *Boss* (neste caso concreto BadGuy), sendo este um inimigo mais robusto que os restantes, subindo o nível de dificuldade do videojogo. O *Boss,* para além de ser mais difícil, tem também uma mecânica de movimento diferente da dos restantes inimigos. Neste tipo de nível o jogador não tem de chegar a um certo sítio, mas sim derrotar o *Boss* para o ultrapassar.

No caso do *Boss* BadGuy, este salta dum lado para o outro do ecrã, perdendo vida sempre que colidi com uma das paredes. O jogador tem de evitar que este lhe toque. Para além disso, BadGuy, conforme perde vida, vai ganhando massa (alterando o efeito da força física aplicado), o que faz com que os seus saltos sejam cada vez mais baixos e difíceis de desviar.

6.6 Conclusão

Neste capítulo foram apresentados os aspetos mais essenciais do Protótipo criado.

Inicialmente foi criado um diagrama do *GameFlow* do Protótipo, para que assim se conseguisse perceber, duma forma simples, todas as janelas e ações que seria necessário implementar.

Em seguida discutiu-se os diferentes pontos presentes no Protótipo, como: os Menus existentes, as Opções do videojogo, a Personagem principal e a composição dos diferentes níveis.

Com a construção deste Protótipo foi possível perceber, e demonstrar, que a *Framework* se encontra capaz de suportar o tipo de videojogos plataforma, duma forma bastante estável.

7 Conclusão

Depois de uma extensa explicação de todos os componentes retratados nesta Dissertação, este capítulo tem como principal objetivo, a reflexão sobre o produto final que foi estudado e implementado.

Para isso será então feita uma análise para que se verifique se todos os objetivos foram alcançados, as limitações do projeto e as implementações futuras.

Finalmente, ainda neste capítulo, será apresentada uma apreciação pessoal sobre toda esta dissertação.

7.1 Objetivos Alcançados

O objetivo desta Dissertação passou, inicialmente, pela construção de um Videojogo. Mas conforme esta era desenvolvida, percebeu-se que o rumo da mesma iria ser um pouco alterado.

Mantendo a construção de um Videojogo um dos grandes focos da Dissertação, não deixando de analisar as diferentes interfaces, o mercado existente, e algumas das ferramentas disponíveis para a criação de videojogos. A dissertação passou a ter como foco principal, a construção e implementação de uma *Framework* Modular com fim a criação de Videojogos.

A *Framework* Modular foi criada com sucesso, sendo que foram procuradas e estudadas algumas das bibliotecas mais populares para a criação de aplicações gráficas, e mais tarde incorporadas na *Framework*, criando assim os diferentes módulos necessários para o bom funcionamento da mesma.

Foi então implementado um pequeno Protótipo que teria como principais funções: a demonstração da estabilidade e as várias funcionalidades da *Framework* criada, e a

representação de um futuro videojogo a ser criado com todos os aspetos previamente estudados ao longo da Dissertação.

Apesar disso, como nunca foi intenção criar um videojogo completo (mecânicas complexas, longa duração, etc.), todos os objetivos inicialmente propostos foram alcançados com sucesso.

7.2 Limitações e Trabalho Futuro

Devido à grandeza do projeto existem diversas limitações, mas de certa forma estas acabam por estar ligadas a todo o trabalho futuro.

Tudo que foi implementado está funcional, só que poderia conter muito mais funcionalidades do que as que existem neste momento. Por exemplo, na componente das forças físicas, neste momento é possível alterar a velocidade a que um objeto se move, ou aplicar uma força simples. Mas, a biblioteca utilizada (Box2D) permite fazer muito mais do que isso ao nível da gestão de forças aplicadas sobre um corpo. Futuramente poderiam ser criados novos métodos, tornando assim, todas as aplicações disponíveis a todos os objetos do videojogo.

Outro caso em que a *Framework* poderia ser melhorada, seria ao nível do controlo dado ao jogador, através de um elevado suporte de técnicas gráficas e de som. Por exemplo, poderia existir a possibilidade do jogador poder escolher um filtro gráfico (alterando o aspeto do videojogo) ou até mesmo o suporte das técnicas: V-Sync e diferentes modos de *Anti-aliasing* (*FSAA*, MSAA, FXAA, etc.) à escolha do jogador.

Para além de tudo isto, existem ainda muitos módulos que não estão sequer implementados. Módulos estes como o de Rede, para poder existir algum conteúdo *online*, ou até mesmo cooperação entre jogadores *online*.

Para terminar a secção de trabalho futuro sobre a *Framework* Modular, seria ainda muito apreciado, um teste à utilidade da mesma. Pois apesar desta ter sido testada por parte integral de um estúdio independente de desenvolvimento de videojogos, seria extremamente importante a realização de um estudo em maior escala. Este estudo deveria ser realizado por um grupo de criadores de videojogos profissionais, de forma a perceber se estes achariam a *Framework* Modular aqui criada uma possível candidata à utilização em grandes projetos.

No que toca ao Videojogo (Protótipo), este deveria sofrer imensas alterações. Sendo que a primeira seria o devido estudo da arte e sonoplastia a ser utilizada durante todo o processo de criação do videojogo.

Para além disso, o videojogo deveria ser muito mais enriquecido. Isto poderia acontecer através da criação de: novos personagens, novos níveis, uma história mais complexa, mais habilidades especiais ao personagem principal, etc.

Finalmente, o videojogo deveria ser exportado como um instalador e com um ícone customizado, de forma a tornar muito mais fácil todo o processo de instalação do videojogo para o utilizador final.

7.3 Apreciação final

Do ponto de vista pessoal, este foi um projeto muito enriquecedor, e uma experiência extremamente gratificante.

Com este projeto, não só foi possível aprofundar o conhecimento em C++, e realmente começar a perceber o verdadeiro poder que esta linguagem tem para oferecer, mas para além disso houve ainda a oportunidade de poder trabalhar com algumas das bibliotecas mais utilizadas neste campo pelos vários criadores profissionais de videojogos. Graças a isso, o autor sente-se hoje muito mais a vontade, e capaz, para discutir algum assunto dentro desta área do que anteriormente.

Para além disso, conseguiu aprofundar os seus conhecimentos em muitos aspetos da indústria dos videojogos, que é uma indústria que sempre o interessou, quer a nível das ferramentas utilizadas pelas várias companhias, quer por toda a dinâmica que existe no processo de criação de um videojogo, desde a conceção do mesmo, até às diferentes plataformas de venda.

Resta-lhe dizer que espera um dia poder continuar a desenvolver esta *Framework* Modular, tornando-a numa ferramenta cem por cento viável para a criação de um videojogo. E quem sabe, ainda vir a criar um videojogo através da mesma.

Referências

[1] Fast Company (7 de Novembro, 2013). Why video games succeed where the movie and music industries fail, [Online]. http://www.fastcompany.com/3021008/why-video-games-succeed-where-the-movie-and-music-industries-fail [Acedido pela última vez em: 29 de Janeiro, 2014]

[2] Steam (n.a.). [Online] http://store.steampowered.com/.

[3] Desura (n.a.). [Online] http://www.desura.com/.

[4] Facebook (n.a.). [Online] https://www.facebook.com/.

[5] Twitter (n.a.). [Online] https://twitter.com/.

[6] GamingEnthusiast (12 de Setembro, 2012). The importance of sound design in video games, [Online]. http://www.gamingenthusiast.net/the-importance-of-sound-design-in-video-games/ [Acedido pela última vez em: 30 de Janeiro, 2014]

[7] Gamasutra (30 de Janeiro, 2013). The Aesthetics of Game Art and Game Design, [Online]. http://www.gamasutra.com/view/feature/185676/the_aesthetics_of_game_art_and_.php [Acedido pela última vez em: 30 de Janeiro, 2014]

[8] Hitfix (5 de Maio, 2013). 10 biggest opening weekends of all time, [Online]. http://www.hitfix.com/galleries/10-biggest-opening-weekends-of-all-time-harry-potter-dark-knight-hunger-games [Acedido pela última vez em: 29 de Janeiro, 2014]

[9] Metro (9 de Outubro, 2013). GTA 5 breaks seven Guinness World Records as global addiction continues, [Online]. http://metro.co.uk/2013/10/09/gta-5-breaks-seven-guinness-world-records-as-global-addiction-continues-4139807/ [Acedido pela última vez em: 29 de Janeiro, 2014]

[10] AllGame (n.a.). Game Genres, [Online]. http://www.allgame.com/genres.php [Acedido pela última vez em: 18 de Fevereiro, 2014]

[11] About.com (n.a.). Game Industry – AAA Game, [Online]. http://gameindustry.about.com/od/glossary/g/Aaa-Game.htm [Acedido pela última vez em: 18 de Fevereiro, 2014]

[12] IndieGameMag (15 de Julho, 2010). What Exactly is an Indie Game?, [Online]. http://www.indiegamemag.com/what-is-an-indie-game/ [Acedido pela última vez em: 18 de Fevereiro, 2014]

[13] Pong-Story (n.a.). Magnavox Odyssey, [Online]. http://www.pong-story.com/odyssey.htm [Acedido pela última vez em: 21 de Fevereiro, 2014]

[14] OUYA (Setembro, 2013). OUYA Fact Sheet September 2013.pdf, [Online]. https://www.dropbox.com/sh/67fld7t4vgsikay/_iMoAHvlLj/Docs/OUYA%20Fact%20Sheet%20September%202013.pdf [Acedido pela última vez em: 21 de Fevereiro, 2014]

[15] PlayStation (n.a.). Press Releases, [Online]. http://us.playstation.com/corporate/about/press-release/ [Acedido pela última vez em: 21 de Fevereiro, 2014]

[16] IGN (23 de Novembro, 2013). Xbox One Release Date, [Online] http://uk.ign.com/wikis/xbox-one/Xbox_One_Release_Date [Acedido pela última vez em: 21 de Fevereiro, 2014]

[17] TechTerms (n.a.). Crossplatform, [Online] http://www.techterms.com/definition/crossplatform [Acedido pela última vez em: 21 de Fevereiro, 2014]

[18] Giant Bomb (23 de Agosto, 2013). Platform Exclusive, [Online] http://www.giantbomb.com/platform-exclusive/3015-5670/ [Acedido pela última vez em: 21 de Fevereiro, 2014]

[19] 1up (26 de Dezembro, 2011). How Retail Games are Made, [Online] http://www.1up.com/features/how-retail-games-are-made [Acedido pela última vez em: 24 de Fevereiro, 2014]

[20] About.com (n.a.). Pros and Cons of Digital Distribution for PC Games, [Online] http://internetgames.about.com/od/gamerentals/i/digitaldistro.htm [Acedido pela última vez em: 24 de Fevereiro, 2014]

[21] Daily Infographic (15 de Junho, 2012). Video-Games: Digital vs Retail Sales, [Online] http://dailyinfographic.com/video-games-digital-vs-retail-sales-infographic [Acedido pela última vez em: 25 de Fevereiro, 2014]

[22] Gamasutra (29 de Maio, 2012). Digital vs. Retail: Five big publishers, [Online] http://gamasutra.com/view/news/171148/Digital_vs_Retail_Five_big_publishers.php [Acedido pela última vez em: 26 de Fevereiro, 2014]

[23] Kinect (n.a.). [Online] http://www.xbox.com/pt-pt/kinect/ [Acedido pela última vez em: 25 de Outubro, 2014]

[24] PSMove (n.a.). [Online] http://pt.playstation.com/psmove/ [Acedido pela última vez em: 25 de Outubro, 2014]

[25] Oculus Rift (n.a.). [Online] http://www.oculus.com/rift/ [Acedido pela última vez em: 25 de Outubro, 2014]

[26] Kotaku (15 de Outubro, 2013). Motion Controls, The Most Popular And Most Broken Idea Gaming Ever Had [Online] http://kotaku.com/motion-controls-the-most-popular-and-most-broken-idea-1445766816 [Acedido pela última vez em: 27 de Fevereiro, 2014]

[27] XNA Framework (n.a.). [Online] http://www.microsoft.com/en-us/download/details.aspx?id=20914 [Acedido pela última vez em: 25 de Fevereiro, 2014]

[28] Unreal Engine (n.a.). [Online] http://www.unrealengine.com/udk/ [Acedido pela última vez em: 25 de Fevereiro, 2014]

[29] GameMaker: Studio (n.a.). [Online] https://www.yoyogames.com/studio [Acedido pela última vez em: 25 de Fevereiro, 2014]

[30] Bullet Physics Library (n.a.). [Online] http://bulletphysics.org/ [Acedido pela última vez em: 24 de Fevereiro, 2014]

[31] Gifford (n.a.). Evolution of the Game Pad [Online] http://www.gifford.co.uk/~coredump/gpad.htm [Acedido pela última vez em: 28 de Fevereiro, 2014]

[32] Old-Computers (n.a.). Binatone TV Master MK IV [Online] http://www.old-computers.com/museum/computer.asp?st=3&c=1035 [Acedido pela última vez em: 27 de Fevereiro, 2014]

[33] NES Wiki (n.a.). R.O.B. (Robotic Operating Buddy) [Online http://nes.wikia.com/wiki/R.O.B._(Robotic_Operating_Buddy) [Acedido pela última vez em: 27 de Fevereiro, 2014]

[34] Digital Trends (30 de Novembro, 2013). The 10 Weirdest Computer Peripherals Ever Made[Online] http://www.digitaltrends.com/computing/10-weirdest-computer-peripherals-ever-made/ [Acedido pela última vez em: 28 de Fevereiro, 2014]

[35] Game Dev (n.a.). Framework vs Engine [Online] http://www.gamedev.net/topic/598999-framework-vs-engine/ [15 de Novembro, 2013]

[36] Indie Games Guild (23 de Fevereiro, 2010). Framework vs Engine [Online] http://indiegamesguild.com/sgtflame/2010/02/23/frameworks-vs-engines/ [Acedido pela última vez em: 10 de Novembro, 2013]

[37] Game Dev (19 de Março, 2013). Your First Step to Game Development Starts Here [Online] http://www.gamedev.net/page/resources/_/technical/game-programming/your-first-step-to-game-development-starts-here-r2976 [Acedido pela última vez em: 27 de Novembro, 2013]

[38] RPG MAKER (n.a.). [Online] http://www.rpgmakerweb.com/ [Acedido pela última vez em: 10 de Março, 2014]

[39] AZ Central (n.a.). What Is the Microsoft XNA Framework? [Online] http://yourbusiness.azcentral.com/microsoft-xna-framework-20700.html [Acedido pela última vez em: 24 de Janeiro, 2014]

[40] MSDN Blogs (25 de Agosto, 2006). What is the xna framework [Online] http://blogs.msdn.com/b/xna/archive/2006/08/25/what-is-the-xna-framework.aspx [Acedido pela última vez em: 12 de Março, 2014]

[41] MCV (19 de Março, 2013). Microsoft's XNA framework no longer in active development [Online] http://www.mcvuk.com/news/read/microsoft-s-xna-framework-no-longer-in-active-development/0110259 [Acedido pela última vez em: 25 de Janeiro, 2014]

[42] CPlusPlus (n.a.). A Brief Description [Online] http://www.cplusplus.com/info/description/ [Acedido pela última vez em: 25 de Março, 2014]

[43] Reddit Gaming (17 de Outubro, 2012). I am a programmer for Guild Wars 2 [Online] http://www.reddit.com/r/gaming/comments/11mz4k/i_am_a_programmer_for_guild_wars_2_amaa/ [Acedido pela última vez em: 25 de Março, 2014]

[44] Forums Epic Games (17 de Fevereiro, 2014). UNREAL ENGINE 4 FAQ [Online] http://forums.epicgames.com/threads/970073-UNREAL-ENGINE-4-FAQ-Read-this-before-asking [Acedido pela última vez em: 25 de Março, 2014]

[45] OpenGL (n.a.). [Online] https://www.opengl.org/ [Acedido pela última vez em: 25 de Outubro, 2014]

[46] FMOD (n.a.). [Online] http://www.fmod.org/ [Acedido pela última vez em: 25 de Outubro, 2014]

[47] GLFW (n.a.). [Online] http://www.glfw.org/ [Acedido pela última vez em: 25 de Outubro, 2014]

[48] Box2D (n.a.). [Online] http://box2d.org/ [Acedido pela última vez em: 25 de Outubro, 2014]

[49] FreeType (n.a.). [Online] http://www.freetype.org/index.html [Acedido pela última vez em: 25 de Outubro, 2014]

[50] Theora Playback Library (n.a.). [Online] http://libtheoraplayer.cateia.com/wiki/index.php/Main_Page [Acedido pela última vez em: 25 de Outubro, 2014]

[51] Code::Blocks (n.a.). [Online] http://www.codeblocks.org/ [Acedido pela última vez em: 25 de Outubro, 2014]

[52] LibreOffice Writer (n.a.). [Online] http://www.libreoffice.org/discover/writer/ [Acedido pela última vez em: 25 de Outubro, 2014]

[53] LibreOffice Draw (n.a.). [Online] http://www.libreoffice.org/discover/draw/ [Acedido pela última vez em: 25 de Outubro, 2014]

[54] GIMP (n.a.). [Online] http://www.gimp.org/ [Acedido pela última vez em: 25 de Outubro, 2014]

[55] Mark Fennell (13 de Janeiro, 2012). Game Development : The Game Definition Document [Online] http://markfennell.com/?p=25 [Acedido pela última vez em: 10 de Junho, 2014]

[56] GiantBomb (n.a.). DuckTales [Online] http://www.giantbomb.com/ducktales/3030-15799/ [Acedido pela última vez em: 27 de Abril, 2014]

[57] GiantBomb (n.a.). Super Meat Boy [Online] http://www.giantbomb.com/super-meat-boy/3030-25114/ [Acedido pela última vez em: 27 de Abril, 2014]

[58] GiantBomb (n.a.). Rayman Legends [Online] http://www.giantbomb.com/rayman-legends/3030-38141/ [Acedido pela última vez em: 27 de Abril, 2014]

[59] GiantBomb (n.a.). Super Mario Bros. [Online] http://www.giantbomb.com/super-mario-bros/3030-15544/ [Acedido pela última vez em: 27 de Abril, 2014]

[60] GiantBomb (7 de Novembro, 2013). Ristar [Online] http://www.giantbomb.com/ristar/3030-16386/ [Acedido pela última vez em: 27 de Abril, 2014]

[61] The Verge (n.a.). Play this: 'Rayman Fiesta Run' is mobile platforming perfection [Online] http://www.theverge.com/2013/11/7/5074460/play-this-rayman-fiesta-run-ios-android [Acedido pela última vez em: 21 de Abril, 2014]

[62] Ricardo Moreira (n.a.). Desenvolvimento de uma framework para a criação de videojogos [Livro]

[63] Prof. Hußmann (n.a.). User-Centered Development Process [Online] https://www.medien.ifi.lmu.de/lehre/ws0607/mmi1/mmi4.pdf [Acedido pela última vez em: 15 de Abril, 2014]

[64] Deborah Hix, H. Rex Hartson (Junho, 1993). Developing User Interfaces: Ensuring Usability Through Product & Process [Livro]

[65] CartouCHe (26 de Janeiro, 2012). Graphical User Interface - Layout and Design [Online] http://www.e-cartouche.ch/content_reg/cartouche/ui_access/en/html/index.html [Acedido pela última vez em: 20 de Abril, 2014]

[66] Gamasutra (23 de Fevereiro, 2010). Game UI Discoveries: What Players Want [Online] http://www.gamasutra.com/view/feature/4286/game_ui_discoveries_what_players_.php [Acedido pela última vez em: 29 de Junho, 2014]

Anexo A

Game Design Document (GDD)

[StickMan Universe]

Game Design Document

Ultima Versão:

1.2.0

(25/09/2014)

Criado e Desenvolvido por Gil Manuel Canizes

Game Overview

Pitch (Conceito)

O jogador terá de controlar um personagem, recorrendo à sua locomoção e ao seu poder de salto, ultrapassando diversos obstáculos e inimigos, de forma a concluir o nível sem sofrer qualquer tipo de dano.

Género

Plataforma

Público Alvo

Idades : 15-55 anos.
Sexo : Masculino e Feminino
Consumidores de videojogos no formato digital, na plataforma PC.

Créditos

Gil Canizes → Criador do videojogo (Arte, Conceito e Implementação)
Vários → Música, Sons, animação do Stickman a tocar bateria

História

Personagens

StickMan

Idade : Desconhecida.
Sexo : Masculino.
Interesses : Dinheiro.

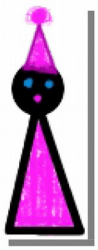
Avatar controlado pelo Jogador durante todo o videojogo.

BadGuy

Idade : Desconhecida.
Sexo : Masculino.
Interesses : Dinheiro.

NPC principal. Boss final do Videojogo.

Princess



Idade : Desconhecida.
Sexo : Feminino.
Interesses : Arco-íris e Unicórnios.

NPC secundário.

Resumo

StickMan está na sua vida normal, quando se depara com uma cena anormal. O vilão BadGuy está a raptar a princesa Princess. O herói fica bastante aborrecido, e decide ir atrás da princesa.

StickMan tem de passar por vários desafios (níveis), até que finalmente consegue chegar ao castelo do BadGuy.

BadGuy decide fazer frente ao StickMan, mostrando o seu derradeiro poder. Mas de nada lhe vale, pois é derrotado por StickMan.

A Princess fica completamente agradecida a este desconhecido herói, pela sua valentia. Mas o nosso herói não tem a mesma ideia em mente que a princesa.

Na realidade ele não a veio salvar, ele veio ficar com o castelo e todo o tesouro, pois a razão para este ter ficado aborrecido, foi por BadGuy ter tido a mesma ideia que ele. A ideia de raptar a Princess.

Gameplay

Objetivos

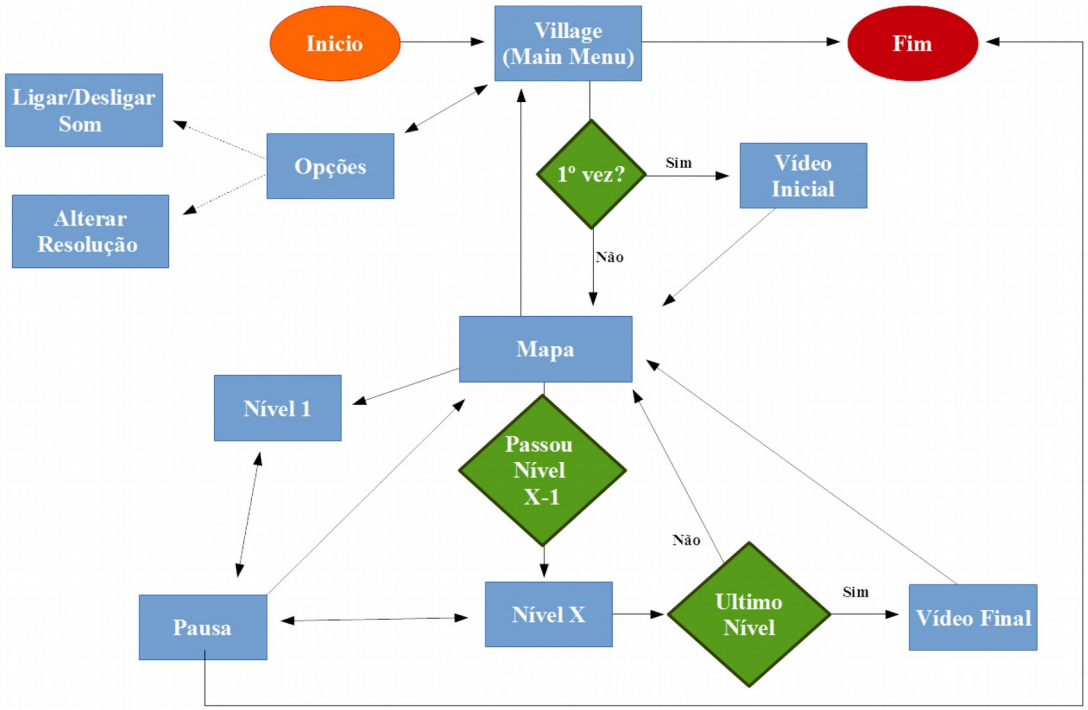
O jogador deve conseguir acabar os diferentes níveis sem sofrer qualquer tipo de dano.

Regras de Videojogo

O jogador pode utilizar as plataformas de forma a avançar no nível;
O jogador pode mover-se e saltar;
Para ganhar o jogador deve conseguir passar o nível todo sem morrer;
As armadilhas e inimigos dispersos ao longo do nível matam o jogador;
O jogador quando morre deve reiniciar o nível;
O jogador não deve iniciar um nível superior ao que está a fazer.

Gil Canizes [StickMan Universe]

Esquema Game Flow



Progressão

O jogador começa por ter um nível desbloqueado.

Conforme o jogador progride e acaba os níveis desbloqueados, novos níveis vão sendo desbloqueados até o jogador chegar ao último nível. Neste caso não é desbloqueado mais nenhum nível.

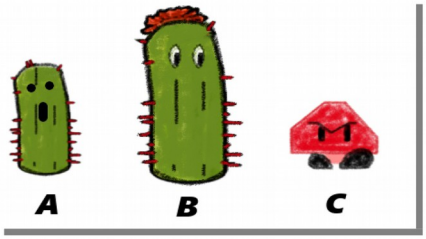
Antagonistic Elements

Obstáculos:

- Cactaceae (**A**)
- Grande Cactaceae (**B**)

Inimigos:

- Crawler (**C**)



Replay e Saves

O jogador a qualquer momento pode refazer os níveis previamente completados, não tendo estes nenhum impacto adicional a nível de progressão no videojogo.

O seu progresso deverá ser guardado sempre que:

- Acaba um nível;
- Altera alguma das definições do videojogo.

Controlos

Ações Ingame:

- Mover Personagem (**A**)
- Saltar (**B**)
- Ação (**C**)
- Pausar (**D**)

Ações Menus:

- Escolher Item Selecionado (**E**)
- Aceitar Item Selecionado (**F**)
- Retomar (**G**)

Ilustração 1: Mapeamento Teclado

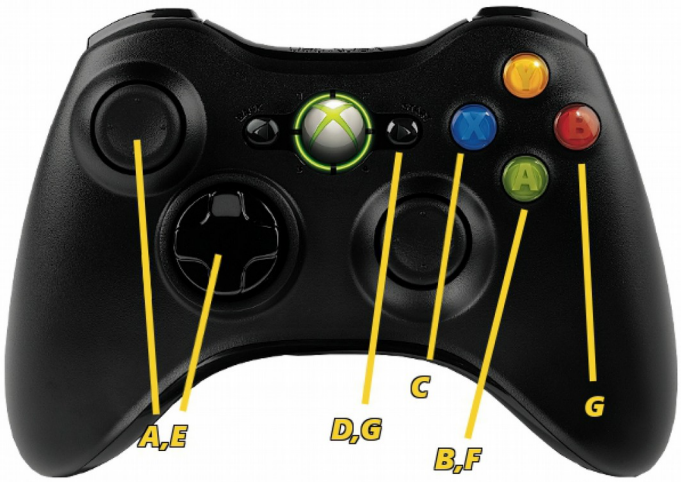


Ilustração 2: Mapeamento Gamepad

Câmara

O jogador não tem controlo sobre a câmara, esta movimenta-se conforme a posição do personagem controlado pelo jogador, de forma a ter sempre o jogador bem visível no ecrã.

Mecânicas de Videojogo

Físicas

Existe uma força gravitacional física que puxa o personagem para baixo, muito parecida com a força gravitacional do planeta Terra.

Movimento do jogador

O jogador pode realizar os seguintes movimentos:

- Andar – Pode-se mover para a esquerda ou para a direita conforme os obstáculos encontrados
- Saltar – Pode fazer com que o personagem salte de forma a conseguir algum tempo aéreo.
- Ação – Pode interagir com alguns dos diferentes elementos presentes no videojogo.

Inteligência Artificial

I.A. Inimigos

Toda a “Inteligência Artificial” dos inimigos está programada para que o seu comportamento seja sempre o mesmo, independentemente do posicionamento do avatar do jogador.

Os Crawlers andam para a frente e para trás, dentro dum espaço previamente definido.

Ao Boss BadGuy são aplicadas forças no eixo do X e Y, para que este se mexa (aos saltos). Conforme leva dano, a sua massa é alterada, de forma a modificar o seu padrão de saltos.

Níveis

Level Design

Os níveis devem sempre conter os seguintes elementos:

- Checkpoint Inicial
- Checkpoint Final
- Chão

Os níveis podem conter os seguintes elementos:

- Plataformas Aéreas
- Inimigos
- Obstáculos
- Caixas
- Boss

Os níveis devem ser sempre da esquerda para a direita.
Os níveis devem ter uma duração curta/média.

Menus

Menus

Pause Menu:

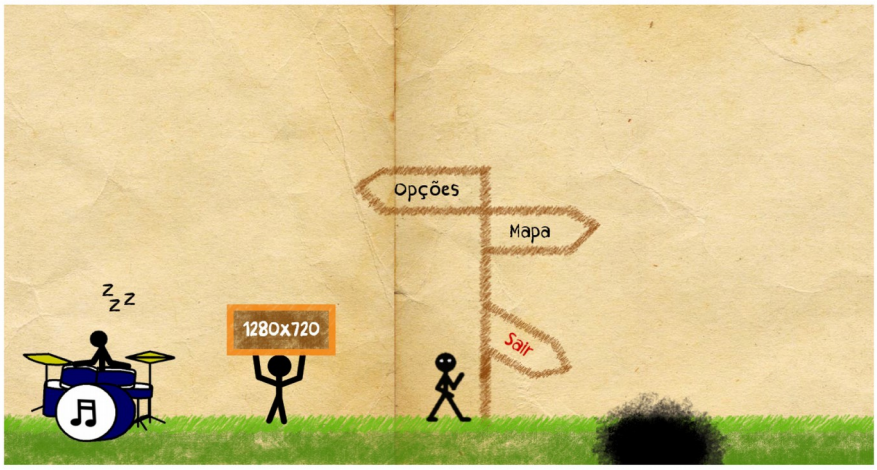
Ações:

- Retomar videojogo
- Sair para o Mapa
- Sair do videojogo

Gil Canizes [StickMan Universe]

In-game

Main Menu:



Ações:

- Alterar Resolução (através do Stickman com tabuleta)
- Ligar/Desligar Som (através do Stickman na bateria)
- Sair do videojogo (saltar para o buraco)
- Entrar no Mapa (seguir direção da tabuleta)

Mapa Menu:

As nuvens servem para mostrar que certas áreas ainda não estão disponíveis.

Ações:

- Selecionar nível

Ajuda

O videojogo final deverá conter um tutorial (parte do nível 1) para ajudar o jogador a perceber todas as mecânicas presentes no videojogo.

Para além disso, o jogador deverá receber pequenas notificações que expliquem o que cada NPC da Just Village (Main Menu) faz.

Especificações Técnicas

Requisitos de Rede

Não existem requisitos de rede para este videojogo.

Conteúdos

Extensões Utilizadas Para Imagens

Extensão 1 - .png

Extensões Utilizadas Para Sons

Extensão 1 - .ogg
Extensão 2 - .wav
Extensão 3 - .mp3

Extensões Utilizadas Para Vídeos

Extensão 1 - .ogv

Anexo B

Questionário Diferentes tipos de Interface

Diferentes tipos de Interfaces

Objetivo:

Com este Questionário pretende-se perceber quais são as formas de navegação que os utilizadores mais gostam, visto não existir uma formula perfeita para tal.

Para isso ser-lhe-ão disponibilizadas quatro formas viáveis de mostrar o conteúdo dos menus ao utilizador, e de receber a ação que este pretende disputar. Isto servirá para reforçar a(s) forma(s) escolhida(s) durante a escrita/estudo da Dissertação.

Teste a Realizar:

Dentro de todos os menus, pretende-se que siga os seguintes passos:

1. Ir ao menu das Opções
    1. Pôr a Barra a 30% (+/-)
    2. Sair das Opções
2. Ir a Jogar
3. Ir ao menu das Opções
    1. Pôr a Barra a 85% (+/-)
    2. Sair das Opções
4. Sair da Aplicação

Para iniciar os testes inicie o Menu1.exe, por favor.

Notas:

Dentro de cada Aplicação, irá obter as informações básicas para realizar os testes. Mas caso tenha alguma duvida deve consultar as notas expressas abaixo para cada Menu.

Não deve avaliar o menu em termos de Design, pois essa componente não foi, nem será tida em conta.

O botão Jogar não desencadeia nenhum tipo de ação. Este serve unicamente para que o utilizador tenha uma ideia da sua posição.

Diferentes tipos de Interfaces

Menu1.exe:

Deve utilizar o Rato e o Clique Esquerdo para navegar nos menus.

Menu2.exe:

Deve utilizar as Setas do Teclado para navegar nos menus.
Deve utilizar o 'Enter' para selecionar a opção pretendida.

Menu3.exe:

Deve utilizar as Teclas do Teclado apresentadas no exemplo para navegar nos menus.

Menu4.exe:

Deve utilizar as Setas do Teclado para mover o personagem, de forma a navegar nos menus.
Deve utilizar a tecla 'X' para selecionar a opção pretendida.

Diferentes tipos de Interfaces

Questionário:

Só deve realizar este questionário após ter realizado os testes descritos acima.

Pode ir respondendo conforme acaba de testar cada Menu se assim o preferir.

Utilize numeração de 1 a 5 para avalizar os seguintes campos, sendo:

1 => Muito Mau.

5 => Muito Bom.

Pergunta	Menu1	Menu2	Menu3	Menu4
Facilidade de Acesso (botões/barra)				
Mais Original (forma de navegar)				
Consistência da Interface				
Prevenção contra Erros pela parte do Utilizador				
Clareza das Ações a serem disputadas				
Opinião Pessoal sobre Menu				

Comentários Finais:

Muito obrigado pela sua colaboração.