Universidade de Aveiro Departamento Comunicação e Arte
2012

António Celso Adrião Soares

Técnicas de análise em contextos de videojogos: o motor source

Universidade de Aveiro Departamento de Comunicação e Arte
2012

António Celso Adrião Soares

Técnicas de análise em contextos de videojogos: o motor source

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Comunicação Multimédia, realizada sob a orientação científica da Doutora Ana Veloso, Professor Auxiliar do Departamento de Comunicação e Arte da Universidade de Aveiro e sob co-orientação do Doutor Óscar Mealha, Professor Associado com Agregação do Departamento de Comunicação e Arte da Universidade de Aveiro.

Dedico este trabalho à minha família, especialmente aos meus pais que permitiram que eu conseguisse efetuar os meus estudos ao longo destes anos e com eles aprendi a ser a pessoa que sou hoje.

o júri

presidente

Prof. Doutor Luis Francisco Mendes Gabriel Pedro
Professor Auxiliar do Departamento de Comunicação e Arte da Universidade de Aveiro

Prof. Doutor Rui Filipe Fernandes Prada
Professor Auxiliar da Universidade Técnica de Lisboa

Prof.ª Doutora Ana Isabel Barreto Furtado Franco de Albuquerque Veloso
Professora Auxiliar do Departamento de Comunicação e Arte da Universidade de Aveiro

Prof. Doutor Óscar Emanuel Chaves Mealha
Professor Associado com Agregação do Departamento de Comunicação e Arte da Universidade de Aveiro

agradecimentos

Quero agradecer aos meus orientadores, à Professora Ana Isabel Barreto Furtado Franco de Albuquerque Veloso e ao Professor Óscar Emanuel Chaves Mealha, pelo apoio que me deram nesta investigação e pela paciência que tiveram comigo.
Quero agradecer também ao Samuel Almeida pela ajuda prestada ao longo desta investigação, que mesmo não sendo meu orientador, acabou por ter um papel semelhante a um.
Obrigado ao Luís Bastião pelo tempo que disponibilizou para discução de assuntos relacionados com algoritmia e programação.
O meu agradecimento ao Bruno Teles, Joaquim Vieira, João Mourão, Miguel Ribeiro e Simão Cardeal que se disponibilizaram para participar no estudo.
Quero agradecer aos professores que tive ao longo destes anos todos que me transmitiram parte do conhecimento que tenho hoje.
Obrigado também aos meus colegas, e sobretudo aos meus amigos mais próximos pelo apoio, amizade e companhia.
Finalmente, o maior obrigado aos meus pais e irmãs que me têm ajudado imenso ao longo destes anos, o que me permitiu chegar onde cheguei.

palavras-chave

videojogos, métricas, eye tracking, modding, visualização de dados, aplicação de análise, usabilidade.

resumo

Para muitos videojogos, durante o processo de desenvolvimento, um dos aspetos mais importantes no desenvolvimento de produtos é a supervisão (avaliação de usabilidade). Sem esta avaliação, pode acontecer que certos problemas relacionados com a conceção e/ou usabilidade do videojogo estejam presentes no primeiro contacto do jogador com o jogo. Daí ser importante analisar e avaliar um videojogo antes de ser lançado para o mercado, nomeadamente com métodos alternativos aos métodos clássicos de avaliação de aplicações digitais. Com este projeto pretende-se contribuir na melhoria da análise de videojogos. Nomeadamente na identificação dinâmica de elementos cénicos dum jogo, que estão a ser visualizados pelo jogador em tempo real. Para que isto seja possível, será necessário criar uma arquitetura de análise de videojogos para sustentar todo o processo. Este processo irá passar pelo uso do Source SDK e do eye tracker até à apresentação final dos resultados que serão apresentados a partir de uma aplicação, designada por “GAMEYE app” que será desenvolvida para o efeito em Flash.

keywords

video games, metrics, eye tracking, modding, data visualization, analysis application, usability.

abstract

For many video games, during the development process, one of the most important aspects in product development is the supervision process (usability evaluation). Without this evaluation it may happen that certain problems related to the design and/or usability of the videogame are present in the first contact of the player with the game. Hence, it is important to analyse and evaluate a video game before being released to the market, particularly with alternative methods to traditional methods used in the evaluation of digital applications. This project aims to contribute in improving the analysis of video games. Particularly, in identifying scenic elements of a dynamic game, which are being viewed by the player in real-time. To make this possible, an architecture for analysing video games to support the whole process is needed. This process will be based on the use of the Source SDK and an eye tracker, up to the final presentation of results that will be presented using an application, called "GAMEYE app" that will be developed for this purpose in Flash.

Índice

Índice de figuras

Índice de tabelas

Acrónimos

CS:S	CounterStrike: Source
EEG	Electroencephalogram
EMG	Electromyogram
EOG	Electro-OculoGraphy
FPS	First Person Shooter
GSR	Galvanic Skin Response
HCI	Human Computer Interaction
HR	Heart Rate
MMORPG	Massive Multi-player Online Role-Playing Game
NPC	Non-Player Character
POG	Photo-OculoGraphy
PoR	Point of Regard [de: (Duchowski, 2007)]
TRUE	Tracking Real-Time User Experience [de: (Kim et al., 2008)]
RIA	Rich Internet Application
RITE	Rapid Iterative Testing and Evaluation
RPG	Role-Playing Game
RTS	Real-Time System / Real-Time Strategy
SDK	Software Development Kit
UCD	User-centred Design
UDK	Unreal Development Kit
VOG	Video-OculoGraphy

1. Introdução

A indústria de videojogos gera biliões de dólares em cada ano que passa, e este negócio tenderá a crescer muito mais nos próximos anos (Ribeiro, 2010). As empresas de desenvolvimento de videojogos, novas ou já fortemente estabelecidas, têm a necessidade constante de desenvolver videojogos de qualidade para entreter o seu público-alvo. Esta necessidade não é estabelecida só a partir da grande exigência dos jogadores, como também a partir da concorrência que existe entre as empresas. No entanto, para muitos videojogos, durante o processo de desenvolvimento, um dos aspetos mais importantes no desenvolvimento de produtos é a supervisão (avaliação de usabilidade). Sem esta avaliação, pode acontecer que certos problemas relacionados com a conceção e/ou usabilidade do videojogo estejam presentes no primeiro contacto do jogador com o jogo, podendo ser cruciais na aceitação deste por parte do jogador. Daí ser importante analisar e avaliar um videojogo antes de ser lançado para o mercado, nomeadamente com métodos alternativos aos métodos clássicos de avaliação de aplicações digitais. Também é importante realçar que, com a análise de videojogos, os desenvolvedores podem ter uma perspetiva mais exata da eficiência da interface, *feedback* e pertinência dos elementos cénicos. E com isto perceber quais os elementos que poderiam ser evitados, poupando tempo e dinheiro no seu desenvolvimento.

A usabilidade tem vindo a ganhar importância, nomeadamente, em produtos web, na área de marketing e publicidade, aplicações *desktop* e *mobile*, entre outros. Mas poucos são aqueles que ainda falam em usabilidade de videojogos. Alguns autores como Melissa Federoff (Federoff, 2002) tentam interligar o conceito de usabilidade com o conceito de videojogos acabando-se por verificar que estes são difíceis de combinar. Federoff, no seu estudo, descobre que mesmo para as pessoas da área de videojogos, o conceito de *game usability* não é usado ou mesmo desconhecido. No entanto, esta situação não é problemática. Afinal, os videojogos não passam de um produto cujo principal objetivo é entreter o público, e não tanto uma ferramenta para facilitar as tarefas diárias de um utilizador.

Existem várias técnicas que complementam a avaliação de videojogos, tais como *eye tracking*, *logging*[1] e *game heurístics*. Tendo isto em conta, serão apresentados alguns estudos que se basearam nestas técnicas e algumas opiniões dos autores em relação ao seu uso. Os *designers* usam estudos de caso, muitas vezes, para analisar o que funciona, mas especialmente o que não funciona num videojogo (Salen & Zimmerman, 2005). Também serão abordadas algumas metodologias de recolha de dados de videojogos tais como *Focus Groups*, questionários, inquéritos e *logging* (técnica de base deste projeto).

[1] Processo de monitorização e gravação de dados

Finalmente, será abordado o conceito de *modding* que está diretamente relacionado com a modificação de videojogos. Este conceito tem vindo a ganhar mais força ao longo dos últimos anos graças às oportunidades que as produtoras de videojogos proporcionam aos jogadores em modificar os videojogos, como por exemplo, oportunidades de emprego àqueles que mostram ter potencial no desenvolvimento de *mods* (jogos modificados). Outras razões pelas quais o *modding* tem aumentado é a possibilidade do *modder* (pessoa que modifica) poder ser criativo na modificação do jogo, identificar-se com os jogo que desenvolve, aumentar a satisfação com o mesmo, entre outras (Postigo, 2007; Sotamaa, 2008; Scacchi, 2010). Para as produtoras de videojogos isto é uma boa notícia, pois aumenta do tempo de vida de um jogo através da sustentabilidade que é dada a partir da comunidade de fãs.

1.1. Problema de Investigação

Para muitos videojogos, durante o processo de desenvolvimento, um dos aspetos mais importantes no desenvolvimento de produtos é a supervisão (nomeadamente avaliação de usabilidade, *design* e testes de implementação). Sem esta avaliação, pode acontecer que certos problemas relacionados com a conceção e/ou usabilidade do videojogo estejam presentes no primeiro contato do jogador com o jogo, podendo ser cruciais na aceitação deste por parte do jogador. Daí ser importante analisar e avaliar um videojogo antes de ser lançado para o mercado, nomeadamente com métodos alternativos aos métodos clássicos de avaliação de aplicações digitais. Também é importante realçar que, com a análise de videojogos, os desenvolvedores podem ter uma perspetiva mais exata da eficiência da interface, *feedback* e pertinência dos elementos cénicos. E com isto perceber quais são os elementos que poderiam ser evitados, poupando tempo e dinheiro no seu desenvolvimento.

Ivory e Hearst (2001) argumentam que a avaliação de usabilidade pode ser cara em termos de tempo e recursos humanos, daí a automação ser um caminho promissor para aumentar as abordagens de avaliação (Dix et al., 1998; Shneiderman, 1998) já existentes.

Com este projeto pretende-se contribuir na melhoria da análise de videojogos, nomeadamente na identificação dinâmica (automação) de elementos cénicos do jogo, que estão a ser visualizados pelo jogador em tempo real (RTS - *Real Time System*). Em estudos anteriores realizados sobre este assunto (Almeida, 2009), esta identificação só foi possível através de um mapeamento manual, por intermédio do cruzamento de informação proveniente do *eye tracking* (PoR - *Point of Regard* - local x e y para onde o jogador está a olhar no ecrã) com a informação posteriormente recolhida nas coordenadas x e y do mapa do jogo. O problema de investigação que se coloca no âmbito desta dissertação é desenvolver uma arquitetura de análise de videojogos, como se pode ver na Figura 1, esta encontra-se dividida em várias fases sequenciais. Esta arquitetura de análise de videojogos será desenvolvida para um caso particular de um jogo FPS (*First Person Shooter*).

Numa primeira fase pretende-se saber qual é a posição do jogador no eixo z do ecrã (profundidade), informação que se consegue obter quando se recorre à implementação de algumas áreas do videojogo para registar em *logfiles* para posterior leitura (Figura 1). Com a informação desta coordenada do jogador será possível calcular automaticamente a posição do elemento cénico, para o qual o jogador está a olhar.

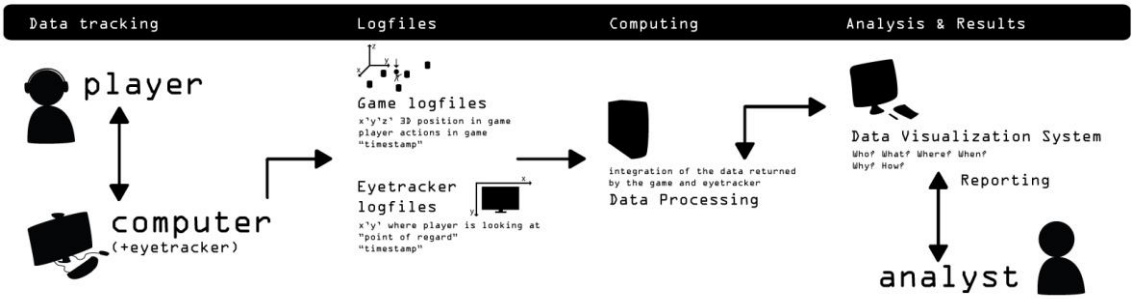


Figura 1 - Arquitectura de análise de videojogos pretendida

Com a abordagem apresentada na Figura 1, pretende-se resolver o problema da construção dinâmica de métodos de visualização de dados para a análise de videojogos e contribuir para uma análise mais completa com dados dos *logfiles* do jogo. Deste modo para além da posição do jogador nas coordenadas x, y, z que poderão ser também pertinentes para a análise (eventos/ações no jogo, por ex.: *kills*, direcção do jogador e pontuação). Para a execução desta experiência, será necessário um jogador que esteja a jogar um jogo num computador ligado a um *eye tracker.* Em tempo real serão, sincronamente, serão efetuados registos em *logfiles* com a informação de interação no jogo e a informação recolhida simultaneamente pelo *eye tracking*. Posteriormente, os dados desses *logfiles* serão processados de forma a serem integrados/cruzados entre si, com o intuito de desenvolver paradigmas de visualização de dados. Esta aplicação apresentará os resultados de forma clara para facilitar a análise e a avaliação do videojogo. Estes paradigmas de visualização de informação poderão apresentar os dados a partir da aplicação desenvolvida depois da sessão terminar (pós-análise de jogo).

Destaca-se ainda que a Figura 1 representa o universo dos dados/elementos envolvidos na construção da arquitetura deste sistema:

- o público alvo (jogador - cultura, idade, *level experience*);
- quais os comportamentos de interação (ações) relevantes para o sistema;
- onde decorre o jogo (mundo virtual/nível de jogo);
- qual a duração da interação jogador/pc (tempo de jogo);
- quais os resultados esperados (melhorar eficiência do *level design*, *enhancing game play*);
- como é feita a análise dinâmica do contexto de jogo (quantitativo - *logfiles*, qualitativo/qualitativo - questionários).

1.2. Perguntas de investigação

Após uma breve referência sobre a caracterização do problema de investigação e os objetivos a atingir, é importante definir uma linha orientadora do projeto. Assim, torna-se imperativo a formulação e definição de uma, ou várias, questões, no sentido de dar respostas objetivas ao projeto de investigação.

Que métodos/técnicas de visualização de informação permitem analisar o cruzamento de diferentes dados em contexto videojogo?

Sub-perguntas

1. *Quais os dados mais pertinentes para caracterizar um contexto de videojogo?*
2. *Quais as técnicas de cruzamento dos diferentes dados registados em contexto videojogo para efeitos de análise do mesmo?*
3. *Quais os esquemas de visualização adequados para análise dos dados que caracterizem um contexto videojogo?*

Com estas questões pretende-se perceber exatamente quais são os paradigmas de visualização que irão representar dados relativos a toda a experiência do jogador durante o jogo, assim como as suas ações, para onde estava a olhar naquele momento, quando aconteceu e com isto ser possível perceber o porquê das atitudes/escolhas tomadas pelo jogador (suportado numa análise quantitativa). A utilização futura destes paradigmas poderá melhorar a avaliação de videojogos, ou seja, tornar a avaliação de videojogos mais completa, e com isto, poder contribuir para a melhoria de usabilidade e conceção nos videojogos.

1.3. Objetivos e finalidades

A finalidade deste projeto é desenvolver uma proposta, com recurso aos paradigmas de visualização de informação que permita a apresentação da informação que caracterize um determinado contexto de videojogo. A aplicação cruza dados provenientes do mesmo contexto de videojogos, mas tem origens diferentes, o comportamento de interação do jogador com o jogo e os dados provenientes do comportamento ocular do jogador (recolhidos através do *eye tracker*) na mesma experiência de jogo.

O processo para atingir a finalidade é complexo e será faseado em várias etapas. Relativamente ao objeto de estudo será escolhido um jogo FPS (*First Person Shooter*) no qual será alterado e preparado para guardar dados em *logfiles* (interação do jogador com o jogo - eventos/ações desempenhadas) para posteriormente serem integrados com os dados resultantes do PoR (*Point of Regard*) através do *eye tracking.*

O videojogo a utilizar é o CounterStrike: Source (CS:S) e terá que ser desenvolvido um *mod* (instância da base de um videojogo - modificada pelo interveniente) do respetivo jogo com a finalidade de o preparar para guardar *logfiles* sobre os eventos/ações desempenhadas pelo jogador.

Por fim, também se pretende desenvolver uma aplicação para a visualização e análise integrada de dados provenientes dos *logfiles* do jogo e do *eye tracker*. Espera-se com esta

aplicação facilitar a análise dos resultados registados nos *logfiles*, atendendo a que a informação poderá ser apresentada através de técnicas de georreferenciação, linhas temporais, gráficos, esquemas ou tabelas (ex.: histórico de ações, *timestamp* do jogador, detalhes da ronda, relatório final gerado, entre outros).

Uma das perspetivas de desenvolvimento futuro que poderá ser tomada em consideração, dependendo da viabilidade temporal do desenvolvimento desta investigação, é a adaptação do *mod* de CS:S para um ambiente relacionado com a Universidade de Aveiro (camada de produto CounterStrike: Source - UA).

Objetivos

1. Perceber como guardar os dados dos comportamentos de interação para os *logfiles* (uso através do Source SDK - *Source Software Development Kit* -Valve *Software*);

2. Adaptar alguns dos conteúdos cénicos dos jogos de modo a perceber a influência que determinados objetos têm no desempenho do jogador;

3. Criar um método que cruze os dados provenientes dos *logfiles* do jogo (ações do jogador, coordenadas x, y e z e *timestamp*) com os *logfiles* dos dados resultantes do *eye tracker* (*Point of Regard* - PoR e *timestamp*);

4. Apresentar uma proposta de uma aplicação baseada num paradigma de visualização que consiga apresentar os dados disponíveis sobre o desempenho do jogador.

1.4. Abordagem Metodológica

A abordagem metodológica pode ser descrita como a estrutura para a investigação que ao cumprir com um grupo de padrões e situações, torna possível selecionar e articular técnicas designadas para auxiliar no desenvolvimento do processo de validação empírica (Pardal & Correia, 1995). A abordagem metodológica principal destinada a este projeto é o método exploratório. Contudo também será aplicado o método descritivo. A primeira é um método no qual uma área específica de estudo se encontra incompleta, e como o nome indica, ainda em exploração. O segundo método concentra-se em descrever minuciosamente um objeto de estudo específico (Carmo & Ferreira, 1998, p. 47).

A abordagem metodológica inicia-se com a elaboração da primeira parte do projeto - enquadramento teórico - isto é, o conjunto de factos e informações relacionadas com as áreas de usabilidade, videojogos e análise de dados. A parte do enquadramento teórico será elaborada através do uso de pesquisa bibliográfica. Além disso, o procedimento de recolha de dados será realizado através da elaboração de um estudo de caso simples, onde os participantes poderão jogar um videojogo baseado no motor Source, onde serão captados os movimentos oculares do jogador através do *eye tracker* (dados escritos num *logfile*) e, ao mesmo tempo, o *mod* desenvolvido estará a registar noutro *logfile* vários dados relativos às ações e eventos desempenhados durante o momento de jogo.

Nesta fase da abordagem metodológica pretende-se explorar a experiência dos jogadores num videojogo, como as suas ações, eventos, dados relativos à posição do jogador no ambiente

tridimensional, saber para onde está a olhar (PoR) e quando (*timestamp*), entre outros. Tudo isto é possível com a análise de *logfiles* de forma quantitativa. Outro recurso a considerar são os questionários (Shneiderman, 1998, pp. 132-135), e com isso, adquirir informação quantitativa e qualitativa sobre os pontos de vista dos jogadores e dados pertinentes quanto à experiência em jogo. Em suma, acaba-se por usar uma abordagem metodológica mista (quantitativa e qualitativa), através de uma análise dinâmica do contexto de jogo, com fim de detetar, por exemplo, o porquê de determinados comportamentos dos jogadores e detetar problemas de usabilidade e conceção do jogo. Segundo Bruckman (2006) os métodos quantitativos e qualitativos são geralmente mais poderosos quando utilizados de forma complementar constituindo-se como fontes de triangulação de dados.

A recolha de dados pela técnica de *eye tracking* (PoR e *timestamp*) será efetuada por meio do uso de um equipamento *eye tracker* T120 Tobii Eye Tracker, e como já foi dito será utilizada para recolher os dados quantitativos relacionados com os movimentos dos olhos do jogador. Os restantes dados (coordenadas x, y e z, ações e eventos do jogador no jogo e quando aconteceu - *timestamp*) serão recolhidos a partir do *mod* de Source (desenvolvido através do Source SDK - *Source Software Development Kit*, da Valve), também para análise quantitativa.

Neste projeto de investigação, pretende-se modificar um *software* e desenvolver outro. O primeiro é o *mod* baseado no motor Source para poder enviar dados para *logfiles*, o segundo, tem como objetivo ler os dados inerentes dos *logfiles* e apresenta-los da melhor forma (desenvolvimento em Adobe Flash na linguagem de programação ActionScript 3.0) com uma amostra de conveniência para o estudo de caso. Uma vez que o objetivo não é avaliar o videojogo, mas sim experimentar o produto desenvolvido. Contudo, o número da amostra não é importante para este estudo de caso, mas sim a distinção entre os utilizadores, ou seja, jogador inexperiente, jogador casual e jogador *hard-core* (com vista a visualizar diferentes análises da aplicação de análise desenvolvida - diferentes utilizadores têm diferentes comportamentos).

1.5. Motivações pessoais

Em primeiro lugar a escolha deste tema deve-se ao facto do investigador (autor) possuir desde muito cedo um fascínio em relação à área de videojogos. Com ligação a competições de videojogos desde 2001, passando por vários clãs de diferentes videojogos dentro da categoria de FPS. Com esta experiência é possível entender melhor muitos aspetos relacionados com a análise de videojogos e a necessidade de haver algo que nos monitorize a experiência de jogo após uma competição. Identificar falhas, problemas ocorridos com a equipa seria uma mais-valia para os clãs que competem em torneios.

Em segundo lugar, pode-se dizer que esta é uma área com crescimento em exponencial, com ainda muito por explorar e com uma grande aposta na inovação para o futuro.

1.6. Estrutura da dissertação

A dissertação está organizada em três partes diferentes. A primeira parte corresponde ao enquadramento teórico; a parte dois está relacionada com o estudo empírico e a terceira parte, com as conclusões do estudo. Cada uma destas partes é composta por vários capítulos. No entanto, estas três partes seguem-se a uma introdução ao estudo.

A introdução é composta por seis secções: o problema de investigação, perguntas de investigação, objetivos e finalidades, metodologia de investigação, motivações pessoais e o capítulo corrente, estrutura da dissertação. O problema de investigação mostra os vários aspetos atuais e o quão relevante é este estudo para a área de videojogos. A secção perguntas de investigação apresenta a pergunta que serve de espinha dorsal para o presente estudo enquanto que o capítulo dos objetivos e finalidades apresenta uma lista de objetivos e uma grande finalidade para o estudo em si. A secção de abordagem metodológica explica a metodologia que foi aplicada durante a investigação curso, enquanto que os motivos pessoais referem os vários motivos que motivaram esta investigação e consequentemente o desenvolvimento deste estudo. Finalmente, a estrutura da dissertação esclarece o leitor sobre a composição desta dissertação.

A primeira parte que compõe o enquadramento teórico do trabalho contém vários capítulos: (i) *Eye tracker* e o sistema visual humano, que inclui as componentes e funções do olho, bem como uma taxonomia dos movimentos dos olhos, técnicas de *eye tracking* e limitações e preeminências do *eye tracker*; (ii) os videojogos onde são apresentadas algumas listas de vários autores sobre géneros de videojogos e, são abordados conceitos como *game design*, *game mechanics* e *game experience*; (iii) Usabilidade na avaliação e conceção dos videojoso, abordando alguns temas como HCI, *game usability* e diferentes tipos de avaliação de videojogos (avaliar com *eye tracker*, através de *logfiles* e segundo *game heurístics*); (iv) métodos e técnicas de recolha e modificação de dados em videojogos, que inclui métricas de jogo para análise, instrumentos de recolha de dados em videojogos, sistema de *logfiles* nos videojogos e *videogame modding.* Esta parte é finalizada com alguns comentários finais sobre enquadramento teórico.

A segunda parte do documento, intitulado Investigação empírica, é composta por dois grandes capítulos: (i) investigação empírica, que aborda o paradigma do objeto de estudo, a integração e visualização de dados (explicação técnica e visual da aplicação desenvolvida) e a experiência empírica propriamente dita (caracterização da amostra, instrumentos de recolha de dados, preparação da experiência e caracterização do estudo, contexto presencial e *setup* da sala); (ii) o segundo grande capítulo, apresentação, análise e discussão dos resultados da experiência empírica é composta pela apresentação e análise do questionário e discussão dos resultados dos dados obtidos do protótipo a partir da experiência.

Finalmente, e para concluir, a última parte do documento é relativa às conclusões, onde vários aspetos do estudo são revistos e analisados. Também são referidos aspetos relacionados com a limitação do estudo e dificuldades encontradas, assim como perspetivas futuras desta investigação.

primeira parte
enquadramento teórico

2. Sistema visual humano e Eye tracker

2.1. Sistema visual humano

A visão é o processo fisiológico por onde se discernem as formas, as cores dos objetos e a distância aos mesmos. O objetivo do sistema visual passa por extrair a luz do mundo, filtrá-la e transformá-la em algo compreensível (Almeida, 2009). São o olho, os respetivos componentes e os processos cognitivos que lhe estão inerentes que tornam o processo de visão possível.

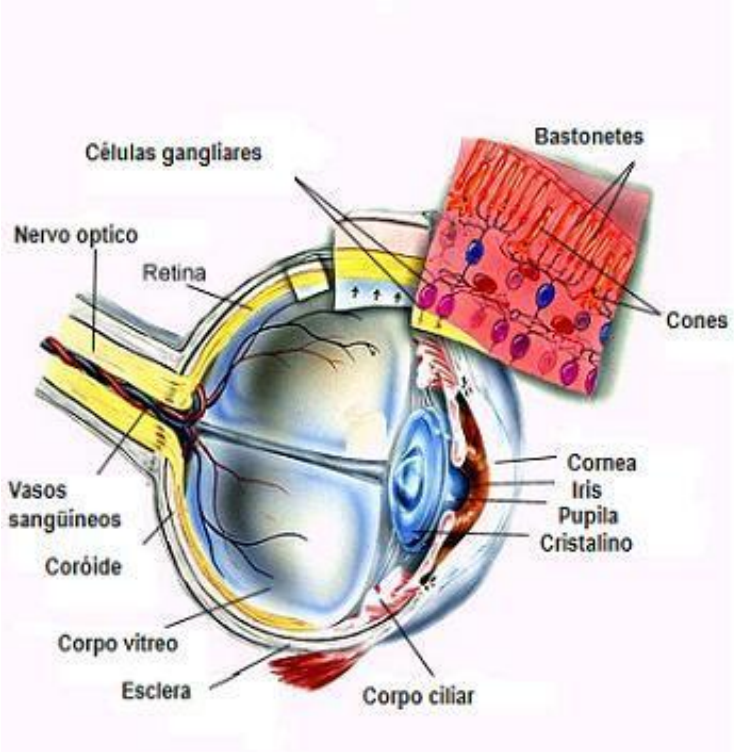


Figura 2 - Composição do olho [retirado de (MedicinaGeriátrica, 2007)]

O sistema visual humano é muito mais complexo do que é apresentado na Figura 2. No entanto, a legenda apresentada é uma indicação de algumas das partes mais importantes do sistema que permite ao humano ver o mundo tal como o conhecemos.

Os olhos são formados por um conjunto de estruturas transparentes (córnea, humor aquoso, humor vítreo e lente) e caracteriza-se por ser o único órgão no corpo humano capaz de permitir uma observação direta da micro-vascularidade. No fundo do olho, observa-se o leito

vascular, constituído de vasos de pequeno calibre (arteríolas e vênulas), envolvidos na resistência vascular periférica (Missagia, 2010).

A córnea é a parte transparente e ao mesmo tempo mais exterior do olho. Localizado sobre a íris, que é de cerca de meio milímetro de espessura. A córnea e o cristalino são similares na estrutura e na função das lentes de uma câmara fotográfica. Juntos, córnea e cristalino, são responsáveis pela capacidade de se concentrar pela refração da luz em pontos específicos sobre a retina. No entanto, a lente tem um papel mais específico no ajuste do foco de objetos em diferentes distâncias (Hubel, 1995, p. 34).

A íris é uma peça em forma de círculo no olho localizada entre as duas lentes do olho (córnea e cristalino) e é responsável por controlar a quantidade de luz disponível para ser processada pelo olho interior. A íris tem músculos, na sua estrutura, que lhe dão mobilidade e atuam como um diafragma de uma máquina fotográfica aumentando ou diminuindo a sua abertura (espaço no centro da íris - pupila) e selecionando a porção de luz que deve estimular a retina. Uma outra propriedade da íris é a cor dos olhos.

A retina, formada por células nervosas (cones e bastonetes), modifica as ondas de luz (ou energia de luz) que entram nos olhos em sinais nervosos e permite ao humano ver em diferentes tipos de condições. A retina discrimina o comprimento de onda para que o humano possa ver a cor (Hubel, 1995, p. 36), uma vez que o sinal é enviado através do nervo ótico ao córtex visual localizado na parte posterior do cérebro.

A retina, semelhante a muitas outras estruturas do Sistema Nervoso Central (CNS - Central Nervous System) tem a forma de um prato e é cerca de um quarto de milímetro de espessura. É composto de três camadas separadas de células nervosas que são mantidas juntas por duas camadas de sinapses feitas pelos axônios e dendritos das células nervosas (Hubel, 1995, p. 36).

Na parte mais interna da retina encontra-se uma camada de células que contêm os recetores de luz: os bastonetes, responsáveis pela visão em condições de pouca luz, e os cones responsáveis pela visão de cores e detalhes. Neste agrupamento de células, há aproximadamente 100 milhões de bastonetes e 7 milhões de cones (Bianco, 2000). Assim como os nomes indicam, a secção externa dos bastonetes são longos e finos enquanto a mesma secção de cones tem uma estrutura em forma de cone.

Os segmentos externos tanto dos cones como dos bastonetes contêm produtos químicos fotossensíveis. Nas hastes, o produto químico é chamado rodopsina (proteína dotada de um agrupamento cromatóforo), e nos cones, os produtos químicos são chamados de pigmentos de cor. Para reduzir a quantidade de reflexão que possa suceder, a retina encontra-se alinhada com a melanina de pigmento preto, que também desempenha um papel na redução da radiação de luz nociva (Bianco, 2000).

Quando a luz atinge a retina, sobrevém uma sequência de reações químicas complexas. O resultado é a formação de um químico - rodopsina ativada - que por sua vez cria vários impulsos elétricos no nervo ótico (Bianco, 2000). Estas reações complexas são responsáveis pela nossa visão. Cada reação produz uma série de impulsos elétricos que por sua vez no cérebro são convertidas em sensações de cor e luz. A capacidade humana para destrinçar a cor

não é um processo limitado aos componentes do olho, mas também ao resultado de processos no córtex cerebral. O que significa que a visão só se completa quando o cérebro humano recebe os impulsos da retina.

O córtex cerebral é responsável por traduzir os sinais eletroquímicos da retina, que em última análise, identifica as imagens e as suas características, como a cor, a forma, o formato, a distância, o tamanho e a orientação (Rosa, 2001 apud. (Almeida, 2009)).

2.2. Taxonomia dos movimentos dos olhos

De que forma o olho humano se movimenta? Quais as partes do olho que tornam os movimentos deste possíveis?

Existem seis músculos extraoculares (Figura 3) que controlam o movimento do olho, cujas ações dependem da posição do respetivo olho no momento da crispação muscular: o reto medial e lateral (encarregues pelo movimento lateral), os retos superior e inferior (encarregues pelo movimento vertical) e os oblíquos superior e inferior (encarregues pela torção).

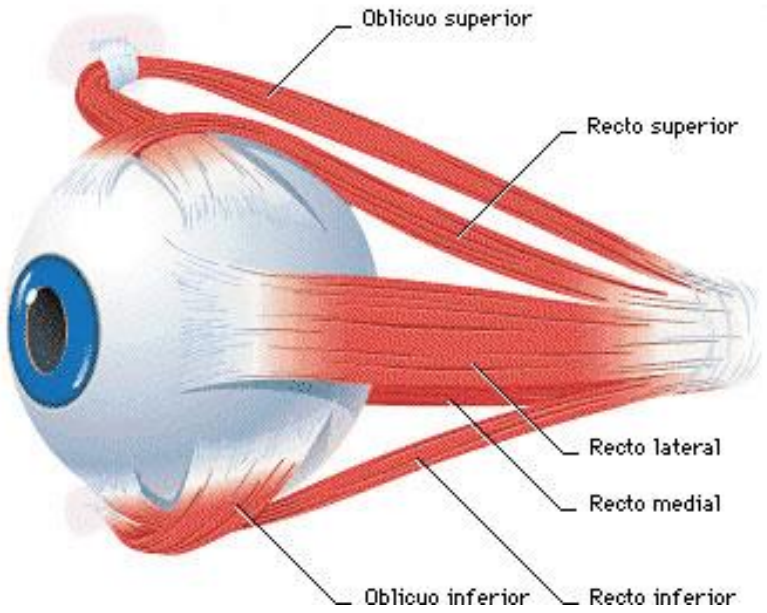


Figura 3 - Músculos extraoculares [retirado de (Álvarez, García, Garófano, & Jiménez, 2008)]

Estes seis músculos são, no entanto, responsáveis pela capacidade de realizar cinco movimentos distintos:

- **movimentos sacádicos -** segundo a perspetiva de Guyton e Hall (2006), movimentos sacádicos são movimentos rápidos dos olhos (Almeida, 2009). Têm como objetivo de posicionar a imagem de um alvo sobre a fóvea, região central da retina do olho humano onde se encontram os cones e se forma a imagem que é transmitida ao cérebro (Marchesin et al. 2005 apud. (Almeida, 2009)). Os movimentos sacádicos duram aproximadamente 10 ms a 100 ms (0,01 a 0,1 segundos). As funções de controlo neuronal sacádicas ainda são debatidas. Alguns acreditam que estes sejam

balísticos, ou seja, que os destinos dos movimentos sacádicos são pré-programados. Outros já consideram que sejam estereotipados, ou seja, os padrões de movimentos sacádicos podem ser lembrados repetidamente (Hubel, 1995, p. 42).

- **movimentos de perseguição lenta -** são movimentos com o intento de manter a imagem de um objeto em movimento na fóvea (Marchesin et al. 2005 apud. (Almeida, 2009)). Se um objeto estiver em movimento, os olhos têm a capacidade de se permanecerem fixos nele. Movimentos esses que se denominam por movimentos de perseguição ou perseguições suaves (Almeida, 2009). Um mecanismo complexo tem a idoneidade de compreender o movimento de um objeto e desenvolver um curso complementar de movimento para os olhos das pessoas (Guyton e Hall, 2006 apud. (Almeida, 2009)).
- **movimentos de vergência -** são os movimentos dos olhos em direcções opostas para que a imagem possa ser posicionada em ambas fóveas (Marchesin et al., 2005 apud. (Almeida, 2009)). Inverso da distância focal de um sistema ótico centrado. Os movimentos de vergência são bastante lentos quando comparados com os outros, com uma duração de 1 segundo ou mais. Há duas razões principais para a ocorrência de movimentos de vergência: o desfoque da imagem da retina e disparidade da retina. Se uma imagem é um pouco distorcida, significa que está muito próxima ou muito distante. Se um objeto está a lançar a sua imagem de forma desigual sobre a retina, movimentos oculares de vergência são usados para reajustar as linhas de visão para alcançar a visão binocular singular (Wong, 2007, p. 82).
- **movimentos vestibulares -** desencadeamento de movimentos rápidos e lentos dos olhos em resposta aos movimentos da cabeça (Marchesin et al. 2005 apud. (Almeida, 2009)), também conhecido como o reflexo vestíbulo-ocular (VOR - vestibular-ocular reflex). É um movimento capaz de manter a imagem na retina quando a cabeça está em movimento, o que é possível através da contra-rotação dos olhos na mesma velocidade que a cabeça se move na direção oposta (Wong, 2007, p. 22).
- **fixações -** as fixações são provavelmente os mais importantes dos movimentos oculares, que são responsáveis pela capacidade de fixar o olhar nos objetos. O movimento dos olhos do tipo fixação é controlado por dois mecanismos neuronais: o mecanismo de fixação voluntária, que permite aos seres humanos, voluntariamente, encontrar o objeto que pretendem fixar com os seus olhos, e em segundo, o mecanismo de fixação involuntária, que prende o olhar sobre um objeto uma vez que foi encontrado (Guyton & Hall, 2006 apud. (Almeida, 2009)). As fixações são identificadas por diversos movimentos pequenos dos olhos: micro-tremores, micro-sacadas e micro-trações, o último sendo responsável por evitar o desbotamento das imagens estáveis (Wong, 2007, p. 18). 90% de tempo de visualização de uma pessoa ocorre através de gravações que duram desde 150 ms a 600 ms (0,15 a 0,6 segundos) (Hubel, 1995, pp. 46-47).

Estes cinco movimentos são o resultado das exímias capacidades do olho. Além disso, com os avanços da tecnologia, esses movimentos dos olhos são capazes de ser registados e analisados através de sistemas de *eye tracking*, uma tecnologia que será especificada no seguinte segmento desta dissertação.

2.3. Técnicas de eye tracking

O olho humano é um órgão imprescindível para o funcionamento de um dos sentidos mais importantes, a visão. Já Marcus André dizia: "...a maior parte do que somos entrou primeiro pelos nossos olhos" ("Frases e Pensamentos", 2007). Estima-se que cerca de 80% da informação recebida do ambiente externo é captada pelo sentido da visão (Cotti, 2009). Possivelmente com esta ideia em mente, durante os últimos anos a comunidade científica tem presenciado avanços interessantes na área de *eye tracking*, uma técnica que possibilita aos investigadores estudar e analisar os movimentos dos olhos.

Embora o eye tracking pode parecer uma tecnologia nova, assim como Jacob e Karn (2003) o dizem, o estudo do movimento dos olhos já tem mais de um século, e a sua primeira aplicação no *design* ergonómico surge nos anos 50 (Fitts, Jones & Milton, 1950 apud. (Giannotto, 2009)).

No ponto de vista tecnológico, o *eye tracking* é uma técnica que muitas pessoas, principalmente investigadores, acham interessante. Além disso, o *eye tracking* é relativamente barato, assim como útil para monitorizar o espaço de uma imagem grande (seja estática ou dinâmica) observada por um utilizador. O equipamento de *eye tracking* com base no vídeo está a tornar-se relativamente barato, e as ferramentas que auxiliam na análise dos dados do *eye tracking* estão se a tornar disponíveis e melhores que as soluções anteriores.

O *eye tracker* é atualmente o mais comum dos dispositivos utilizados para determinar e medir o movimento dos olhos, onde normalmente são considerados dois tipos de técnicas:

- a técnica de medição da posição do olho em relação à cabeça;
- a técnica que mede a orientação do olho no espaço, também conhecida como o PoR (Young & Sheena, 1975 apud. (Duchowski, 2007, p. 51)).

A segunda técnica normalmente é usada para identificar os itens de uma cena visual, analisar os padrões de leitura e consulta visual dos utilizadores face ao monitor, assim como as interfaces web tradicionais, considerando que a tecnologia mais adequada para esta medição é o *eye tracker* com base no vídeo da reflexão da córnea (Duchowski, 2007, p. 51).

As técnicas de *eye tracking* têm um grande potencial de aplicação numa variedade de áreas de estudo, desde a publicidade e marketing até à investigação médica, passando por estudos de usabilidade. Neste último, o objetivo é determinar como o utilizador explora visualmente a interface na qual este interage.

Segundo Duchowski (2007), existem quatro categorias de medições dos movimentos oculares: Eletro-oculografia (EOG), lentes de contacto especiais (*Scleral Search Coil*), Foto-oculografia (POG) ou vídeo-oculografia (VOG) e VOG baseado no método centro-da-pupila/reflexão-na-córnea.

2.3.1. Electro-oculografia

A Electro-oculografia foi o método mais usado na monitorização do movimento do olho há cerca de 40 anos atrás. Esta técnica consiste na medição da diferença potencial elétrica (da ordem de alguns micro-volts) da pele com base em elétrodos em torno do olho, como se

verifica na Figura 4. No movimento horizontal ou vertical dos olhos, este potencial elétrico medido sofre pequenas variações que são registadas por um equipamento (Figura 5) e ao mesmo tempo são convertidos em graus que representam os movimentos e a posição do olho em relação à cabeça do utilizador (Metrovision, 2008; Duchowski, 2007; Morimoto; Mimica, 2004 apud. (Giannotto, 2009)).

Figura 4 - Exemplo de um EOG [retirado de (Duchowski, 2007, p. 52)]

Como o EOG mede os movimentos do olho relativamente à posição da cabeça, esta técnica deixa de ser a mais adequada para a medição de PoR (Duchowski, 2007, p. 52). Uma das caraterísticas vantajosas do EOG é de ser uma técnica não invasiva, e também tem a sua mais-valia de facilidade de montagem. Da mesma forma, a técnica não levanta qualquer problema para as pessoas que usam óculos. Além disso, pode ser facilmente utilizado enquanto uma pessoa movimenta livremente a sua cabeça e em simultâneo, ter os olhos fechados. Porém, a quantidade de luz presente no ambiente deve ser mantida de forma constante devido à influência sobre as eventuais taxas córneo-retinais (Deuschl e Eisen, 1999 apud. (Almeida, 2009)). Na Figura 5 é apresentado

Figura 5 - Unidade de gravação eletro-fisiológico [retirado de (Giannotto, 2009, p. 37)]

Esta técnica exige menos conhecimento especializado dos aplicadores (dispositivos/materiais) em relação à técnica baseada em lentes de contato (técnica explicada no tópico seguinte) e oferece menor risco aos utilizadores (Giannotto, 2009, p. 38). No entanto, Duchowski (2007)

afirma que não constitui uma solução razoável quando se espera obter o ponto observado pelo utilizador.

Uma das desvantagens inerentes desta técnica é o facto dela não oferecer recursos para a determinação da posição da cabeça, o que força o utilizador a permanecer completamente imóvel, o que pode invalidar a experiência. Em contrapartida, para pessoas com deficiências físicas que as impedem de movimentar a cabeça, a EOG ainda se apresenta como uma solução interessante e de baixo custo (Giannotto, 2009, p. 38).

2.3.2. Lentes de contato especiais

A técnica de lentes de contacto especiais, conhecida por *Scleral Search Coil*, é um dos melhores métodos em termos de precisão da medição dos olhos, embora esta seja extremamente intrusiva. Esta técnica consiste em anexar um objeto mecânico ou ótico numa lente de contacto que é usada no olho, como é mostrado na Figura 6.

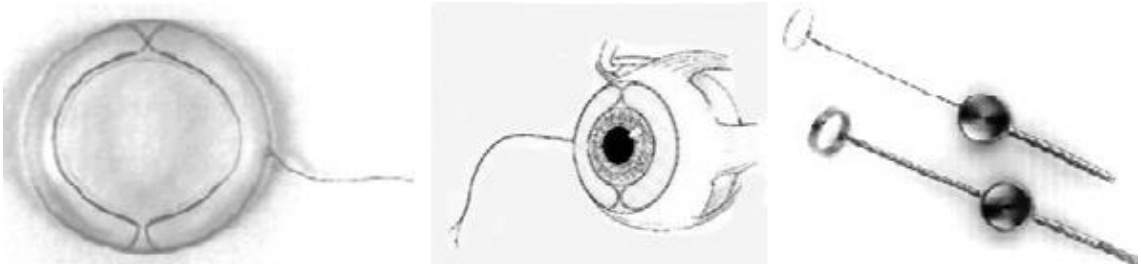


Figura 6 - Pequena bobina anexada numa lente de contacto (*Scleral Search Coil*) e ferramentas de instalação da lente [retirado de (www.skalar.nl/index2.html)]

A lente deve ser grande, de maneira a que esta cubra a córnea e a esclera. Caso fosse apenas para cobrir a córnea, a lente poderia acabar por se deslizar do olho. A técnica das lentes de contacto *Scleral Search Coil*, inicialmente proposta por Robison em 1963, tem evoluído ao longo dos anos e agora é composto por uma lente de contacto moderna em que a haste de fixação encontra-se unida à lente. Entre vários dispositivos anexados às lentes de contacto, Duchowski (2007) refere o uso de fósforos reflexivos e de diagramas formados por linhas, usados para facilitar a deteção do olho em imagens, espirais de fio e bobinas que captam variâncias no campo magnético enquanto o olho se movimenta, e também lentes conectadas mecanicamente a dispositivos de registo em papel e/ou em disco, utilizadas pelos primeiros *eye trackers*.

2.3.3. Foto-oculografia (POG) ou vídeo-oculografia (VOG)

Ambas as técnicas agrupam uma grande variedade de metodologias de gravação dos movimentos oculares que envolvem a medição de várias características do olho em rotação/translação. Estas técnicas estão agrupadas porque normalmente não permitem a medição do PoR (Duchowski, 2007, pp. 53-54).

Ao contrário das outras técnicas apresentadas até agora, tanto a POG como a VOG não são invasivas, ou seja, não exigem contacto com os olhos nem com a cabeça do utilizador, permitindo um maior conforto durante o processo. Estes métodos são baseados na utilização

de câmaras de vídeo que registam imagens dos utilizadores, que por sua vez são usadas nos processos de deteção da posição dos olhos e respetivos movimentos.

A medição dos movimentos dos olhos oferecidos por estas técnicas nem sempre é automático e, portanto, podem exigir uma inspeção visual dos movimentos oculares registados, que por sua vez é cansativo e sujeito a erros (Duchowski, 2007, pp. 53-54).

2.3.4. VOG baseado no método centro-da-pupila/reflexão-na-córnea

As técnicas mencionadas anteriormente, apesar de úteis para a medição do movimento dos olhos, não permitem a medição do PoR. Para fornecer a medição do PoR a cabeça do utilizador deve estar fixa de modo a que a posição dos olhos em relação à cabeça e o PoR coincidam, ou seja, várias características oculares devem ser medidas a fim de desambiguar o movimento da cabeça da rotação do olho. Duas dessas características são o reflexo da córnea (geralmente por meio de uma fonte de luz infravermelha) e do centro da pupila (Duchowski, 2007, p. 54).

Hoje em dia, dentro das técnicas referidas, a centro-da-pupila/reflexão-na-córnea é a mais usada na construção de um *eye tracker*. Esta técnica funciona a partir do processamento de imagens para a deteção da pupila e dos feixes de luz refletidos nos olhos dos utilizadores (Tobii, 2008 apud. (Giannotto, 2009, p. 41)).

Como ocorre com o VOG tradicional, os *eye trackers* baseados em centro-da-pupila/reflexão-na-córnea utilizam câmaras e equipamento de processamento de imagem para calcular o PoR em tempo real, mas em contrapartida esta técnica trabalha com o espetro de luz infravermelho que não é visível ao olho humano.

Os *eye trackers* baseados em centro-da-pupila/reflexão-na-córnea utilizam LED's como fontes de luz infravermelha para estabilizar a iluminação do rosto da pessoa e aumentar o contraste entre a pupila e a íris, o que vai facilitar a sua localização. A luz infravermelha não causa desconforto nem distrai o utilizador, devido ao facto desta luz não ser visível a olho nu (Morimoto et. al, 1999 apud. (Giannotto, 2009, p. 41)).

Ambos os dispositivos de medição, são semelhantes em termos óticos e são sistemas cada vez mais comuns, assim como também são úteis para medir os movimentos oculares em sistemas interativos. A reflexão na córnea do olho da fonte de luz é medida em relação à localização do centro da pupila. Estas reflexões são também conhecidas como reflexões *Purkinje* ou imagens *Purkinje*. Na realidade, quando a luz atinge o olho, a estrutura do olho humano reflecte quatro camadas diferentes, por outras palavras, quatro reflexões *Purkinje* são formadas. Os *eye trackers* baseados em vídeo são capazes de localizar a primeira imagem de *Purkinje* (Duchowski, 2007, pp. 54-56).

Para separar os movimentos dos olhos dos movimentos da cabeça requer-se dois pontos de referência. Duchowski afirma, que a diferença de posicionamento entre o centro da pupila e o reflexo da córnea muda com uma rotação pura do olho, mas permanece relativamente constante com os movimentos mais pequenos da cabeça (Duchowski, 2007, p. 57). Todavia, esta diferença mantém-se constante com os movimentos leves da cabeça. Desde que a fonte de luz esteja colocada normalmente numa posição fixa em relação aos olhos, a imagem de

Purkinje está relativamente estável enquanto o globo ocular e a pupila giram em sua órbita (Duchowski, 2007, p. 57).

Conforme Li, Badcock e Parkhurst (2006) indicam, o principal problema desta técnica é que esta não é muito adequada para ambientes externos à luz do dia, exatamente porque a luz natural, composta por luz em diversas faixas do espectro luminoso (incluindo luz no espectro infravermelho), cria interferências no processo, inviabilizando-o.

Isto resume uma visão geral sobre algumas das técnicas *eye tracking* disponíveis para medir o movimento dos olhos. De seguida destaca-se as técnicas que têm sido aplicadas na investigação em usabilidade, bem como quais as primazias e limitações existentes nesta tecnologia.

2.4. Limitações do eye tracking

Na investigação de Jacob e Karn (2003), são citados vários autores como Crowe e Narayanan (2000), e Redline e Lankford (2001) . Todos estes autores mencionados têm uma ideia em comum, que afirma que o *eye tracking* é uma técnica auspiciosa no campo da engenharia de usabilidade. Considerada auspiciosa devido ao seu grande valor na área, porém também tem as suas falhas. Jacob e Karn (2003), indicam três possíveis razões pelo início lento da técnica na investigação de usabilidade:

1. **problemas técnicos em estudos de usabilidade**

   Atualmente, os *eye trackers* são mais fáceis de operar quando comparados com as primeiras aplicações desenvolvidas. Basta imaginar as dificuldades que Paul Fitts e seus colegas encontraram há cerca de 60 anos atrás na criação da primeira aplicação de *eye tracking* na história (Fitts et al. 1950 apud. (Giannotto, 2009)).

   Os sistemas de *eye tracking* comercialmente disponíveis que são normalmente usados para estudos de laboratório, focam-se na medição do PoR. Os vendedores e fabricantes geralmente fornecem *software* para auxiliar o processo de instalação e calibração. A técnica baseada em vídeo juntamente com o processo de calibração simples torna os sistemas de *eye tracking* não só de fácil utilização, mas também inspiram confiança nos métodos de análise.

   Todavia, Jacob e Karn (2003) defendem a necessidade de dissipar, tanto quanto possível, a relação entre o dispositivo de *eye tracking* e o participante. Esta pode ser a maior barreira para o uso de *eye tracking* em mais estudos de usabilidade. Apesar da existência de duas opções válidas para a análise dos movimentos dos olhos, ambas impõem algumas limitações em termos de restrições de conforto e movimento. Os investigadores podem escolher entre usar uma tabela montada no sistema de *eye tracking*, o que limita a circulação do corpo de um utilizador, ou então usar um sistema montado na cabeça implicando que haja um aparelho fixo à cabeça do utilizador de forma desconfortável.

   Quanto à mobilidade, têm sido feitos avanços recentes no desenvolvimento de *eye trackers* portáteis (como telemóveis e PDAs), com o intuito de serem mais fáceis de executar. No entanto, apesar da evolução verificada nos últimos anos, muitos ainda

resistem, principalmente por questões financeiras, no uso desta tecnologia em investigação de usabilidade.

2. trabalho intensivo de extração de dados

Os *eye trackers* comuns produzem informação relacionada com a orientação visual do utilizador. Tipicamente, o sistema apresenta resultados com as coordenadas das ordenadas e abcissas. A quantidade de dados devolvidos pelo sistema pode aumentar rapidamente, dependendo da percentagem da amostra usada, bem como da duração da sessão (Jacob & Karn, 2003, p. 579). Trata-se de uma boa estratégia inicial para distinguir a análise de sacadas e de fixações, duas coleções de dados com base no movimento dos olhos. Geralmente, o *software* é capaz de filtrar estes dois tipos de dados, simplificando o trabalho do investigador.

Todavia, devido à natureza dinâmica das interfaces de utilizador mais comuns, estudar os movimentos dos olhos, fixações, pode ser difícil. Nas interfaces dos dias de hoje, animações, *pop-ups* e texto dinâmico são comuns e, por isso, torna a missão de filtrar fixações mais árdua do que com um estímulo estático.

Outra dificuldade com a extração de dados pode ser colocada sobre a cabeça do utilizador ou do movimento do seu corpo. Esta situação torna-se complicada, por exemplo, quando o sistema não está a captar o movimento dos olhos devido a algum problema de movimentação ou posição do utilizador. Neste caso, o uso de sistemas de *head-tracking* juntamente com os sistemas de *eye tracking* pode diminuir este problema (Jacob & Karn, 2003, p. 580).

3. dificuldades na interpretação dos dados

Mesmo que as duas últimas limitações não representem um obstáculo para o investigador, ainda permanece o problema de dar algum sentido aos dados devolvidos pelo *eye tracker*. Perante esta situação Jacob e Karn (2003) questionam-se da seguinte forma: "How does the usability researcher relate fixation patterns to task-related cognitive activity?" (Jacob & Karn, 2003, p. 580).

A interpretação de dados pode ser feita usando o método *top-down* (baseado quer na teoria cognitiva, quer na hipótese de *design*) ou o método de *bottom-up* (baseado em dados de observação, sem movimento do olho/relações de atividade cognitiva) (Goldberg, Stimson, Lewenstein, Scott & Wichansky, 2002 quote (Jacob & Karn, 2003, p. 580)). Para exemplificar, Jacob e Karn (2003) apresentam o seguinte:

O método *top-down* baseado na teoria cognitiva pode parecer o método mais atraente, mas em muitos casos, as investigações não têm uma teoria bem fundamentada ou uma hipótese que guie a análise (Jacob & Karn, 2003, p. 580). Se for esse o caso, os autores mencionados sugerem a aplicação de uma pesquisa de dados guiados, para padrões de fixação.

Para ser capaz de interpretar os dados resultantes do *eye tracking*, o investigador deve selecionar as variáveis ou métricas para analisar com base nos dados adquiridos. Jacob e

Karn (2003) apresentam algumas das métricas de *eye tracking* mais usadas pelos investigadores: fixações, duração do olhar, área de interesse e caminho percorrido.

As fixações, como mencionado na secção sistema visual humano, são os movimentos dos olhos relativamente estáveis. A duração do olhar é a duração cumulativa e a posição espacial média de uma série de fixações sucessivas dentro de uma área de interesse. Esta última normalmente inclui várias fixações, bem como sacadas pequenas entre as fixações. Logo que o olho se movimenta fora da área de interesse, o olhar termina. A área de interesse é uma área no ambiente visual que é de interesse para os investigadores e o foco do estudo e o caminho percorrido é uma combinação espacial de uma sequência de fixações.

Destaca-se a Tabela 1 com algumas das limitações (já referidas) do *eye tracking* e respetivas formas de evitar/diminuir essas mesmas fraquezas organizada por Karn (2006).

Limitações do *eye tracking*	**Técnicas para minimizar as limitações**
O *eye tracking* não é o instrumento mais adequado para responder à maioria das perguntas de usabilidade	• Não tentar usar o *eye tracking* para responder a perguntas, que podem ser respondidas de melhor forma a partir de outras ferramentas; • "*Cherry pick*" os problemas a serem abordados (aplicar o *eye tracking* apenas aos problemas que são suscetíveis de serem benéficos).
Não se pode aprender sobre a interação homem-máquina pelo *eye tracking* aplicado a pessoas que apenas olham para uma interface.	• Combinar o *eye tracking* com técnicas tradicionais de testes de usabilidade baseados em tarefas.
A definição e detecção de fixações e sacadas individuais é complexa e não necessária	• Analisar tempos em AOI's (não fixações individuais).
Pode ser um desafio para a consideração dos movimentos da cabeça e corpo, e movimento ou mudança na cena visual.	• Utilizar a tecnologia de *head-tracking*; • Considerar como hipóteses, as ferramentas disponíveis para o *tracking* de um *site web* e sincronizar esses dados com os movimentos monitorizados da cabeça e dos olhos; • Considerar a codificação de alguns dados manualmente, escolher amostras menores de dados para análise.
A alta precisão continua cara e dispendiosa em termos de tempo.	• Focar em grandes áreas de interesse; • Usar grandes ecrãs/simulações de produto.
A tecnologia de *eye tracking* ainda requer um investimento substancial na aprendizagem do uso do equipamento e *software* de análise.	• Contratar por fora um trabalho de *eye tracking* (empresa/equipa especializada no uso de *eye tracker*) • Indústria de suporte/colaboração universitária; • Incentivar a partilha de código fonte aberto

	de análises.
A interpretação dos dados pode ser difícil e demorada.	• Incentivar a publicação e partilha de técnicas de análise e de código aberto; • Ficar com análises básicas, sempre que for possível. Foco na visualização de dados.

Tabela 1- Fraquezas do *eye tracking* em avaliação de usabilidade e respetivos métodos para minimizar estas fraquezas (retirado de (Karn, 2006))

Provavelmente o maior obstáculo para o uso de *eye tracking* em estudos de usabilidade é a eventual dificuldade em relacionar os dados do *eye tracking* com a atividade cognitiva (Jacob & Karn, 2003). Com base nessa ideia, pode-se afirmar que os investigadores que optarem por utilizar este método de avaliação nos seus estudos, devem perguntar-se a si mesmos: "Quais os aspetos da posição do olho que vão ajudar a explicar os problemas de usabilidade?"

A afirmação de que o *eye tracking* é uma técnica de medição válida em usabilidade, pode levantar algumas dúvidas. Contudo, muitos acreditam nos seus benefícios e potencialidades. Na secção seguinte falar-se-á sobre algumas preeminências associadas ao uso do *eye tracking* em investigação de usabilidade.

2.5. Preeminências do eye tracking

Keith Karn (Karn, 2006) defende o uso de *eye tracking* em estudos de interação homem-computador, cujos benefícios têm sido demonstrados anteriormente. Isto inclui as seguintes áreas: pesquisa visual, aprendizagem, visibilidade das funcionalidades do produto e análise de certas tarefas que outros métodos tradicionais de testes de usabilidade indicaram um problema que o *eye tracking* pode esclarecer. Com isso em mente, Karn no workshop CHI 2006 apresenta uma lista extensa de benefícios associados à utilização do *eye tracking* para avaliação de usabilidade (Karn, 2006).

Como já foi referido, *eye tracking* pode ser aplicado a áreas onde os seus benefícios têm o maior impacto. Karn (2006) enumera-os da seguinte forma: o desenvolvimento da atenção visual durante a pesquisa de tarefas guiadas ou de uma visão mais geral; a aprendizagem (mudança de padrões de fixação ao longo do tempo); avaliação da eficiência de sistemas onde o tempo de reação visual-motora é crucial (por exemplo, os sistemas utilizados em situações de emergência) e, finalmente, na análise de tarefas em que os métodos tradicionais de testes de usabilidade têm indicado um problema que o *eye tracking* poderia resolver (por exemplo, atrasos durante o uso de produtos que numa forma tradicional de protocolos de pensamento em "voz alta" não esclarecem) (Karn, 2006).

Para além de estudar o tempo de reação de um utilizador, ao analisar o seu padrão de pesquisa visual, o *eye tracking* também pode ajudar a compreender as estratégias que um determinado utilizador exerce ao visualizar um estímulo. Isto pode ser aplicado em estudos nos quais é necessário compreender o padrão de pesquisa do utilizador pelos recursos de um produto, assim como por exemplo em páginas *web*. Além disso, pode ajudar a perceber quais as partes da interface que têm mais impacto no utilizador.

Quanto à componente de aprendizagem, o *eye tracking* pode ser usado para compreender a diferenciação entre utilizadores principiantes e utilizadores experientes em termos de padrões

de pesquisa, bem como, a forma como um determinado utilizador evolui em termos do seu próprio padrão de pesquisa.

O *eye tracking* também pode ser usado para a avaliação de sistemas e respetiva eficiência, onde o tempo de reação visual-motora é importante. Exemplos desses tipos de situações são os desembarques de aviões, emergências de centrais elétricas, combates a incêndios, situações de guerra, entre outros.

Finalmente, quando as técnicas tradicionais de avaliação de usabilidade encontram um problema com um produto, o *eye tracking* pode ser usado para clarificar qual é o problema. Claro que isto só é benéfico para alguns produtos, devido às limitações de contexto já referidas.

Há, portanto, muitas perguntas a serem respondidas quando ao desejo de aplicar o *eye tracking* em superfícies de testes de usabilidade. Em muitos casos, a técnica de *eye tracking* é um bom método para começar a avaliar um produto ou para complementar uma avaliação de usabilidade já previamente feita. Em outros casos, o *eye tracking* ainda apresenta algumas limitações e não é o método mais recomendado para a avaliação. Seja qual for o caso, cabe ao investigador compreender o que é necessário ser avaliado e se a escolha de determinados métodos, e a respetiva triangulação dos dados recolhidos, poderá produzir os resultados adequados e desejados.

3. Os Videojogos

No ponto de vista de Salen e Zimmerman (2004), um jogo é um sistema no qual os jogadores se envolvem num conflito artificial, definido por regras, que determina um resultado quantificável (Salen & Zimmerman, 2004). Já McLuhan encara o jogo de uma forma mais filosófica, comparando-o a um media, uma vez que considera que os jogos são extensões do homem social e do corpo político, tal como as tecnologias são extensões do organismo animal (McLuhan, 1994, p. 258). É neste novo media que grande parte deste trabalho se baseia, e neste capítulo serão abordados vários aspetos inerentes aos videojogos, tais como a sua taxonomia, *game design*, *game mechanics* e *game experience*.

Até finais do século XIX, o ato de jogar estava associado ao entretenimento e à diversão. O termo jogo provém do latim ludus que remete para divertimento, passatempo e distração. Com o desenvolvimento da tecnologia informática surge uma nova fase dos jogos, os videojogos ou jogos de computador. A evolução na área dos videojogos tem vindo a afirmar-se na cultura contemporânea, e tendem a evoluir para videojogos mais complexos em termos de grafismo e de jogabilidade, devido ao acompanhamento do *software* e do *hardware*.

Os videojogos são, actualmente, um negócio de vários biliões de dólares (na área dos media & entretenimento). No estudo de mercado da Global Entertainment and Media Outlook 2009-2013 – realizado pela PriceWaterHouseCoopers, a estimativa sobre o volume de negócios na indústria de videojogos é de 51,4 biliões de dólares no ano de 2008 com estimativa de crescimento para 73,5 biliões no ano de 2013. Recentes notícias apontam para um crescimento exponencial (e tudo indica ser bastante superior aos "conservadores" 6,9% de crescimento médio anual) (Ribeiro, 2010).

3.1. Géneros e Taxonomias de jogos

A essência dos videojogos passa pelo entretenimento e pela diversão, e estes são desenvolvidos a pensar no público-alvo, os jogadores. No entanto, os gostos são variados, e por isso, houve uma necessidade de criar diferentes categorias nos videojogos.

Um dos mais notáveis *designers* de videojogos do mundo, Chris Crawford (Crawford, 1982), menciona no seu livro *The Art of Computer Game Design*, uma taxonomia para classificar os jogos. O autor aponta para uma visão interessante sobre a categorização de géneros de videojogos, sugerindo que não existe uma taxonomia correta. Porém, deve-se ter em conta que as ideias de Crawford foram sugeridas há cerca de 30 anos atrás, numa altura em que os jogos estavam numa fase inicial de crescimento. Mesmo assim, Crawford dividiu os videojogos em duas grandes categorias: jogos *Skill-and-Action* e jogos de Estratégia, cada uma com as próprias subcategorias.

Dentro da categoria *Skill-and-Action* encontram-se os jogos de combate, de labirinto, de desporto, de *Paddle*, de corridas e diversos. Quanto à categoria de jogos de estratégia, Crawford inclui os jogos de aventura, D & D, jogos de guerra, de Azar, educativos para crianças e interpessoais (Crawford, 1982).

Na perspetiva de Wolf (2000) a divisão dos videojogos tem que ser feita através das suas características, nomeadamente, as experiências interativas que estes proporcionam, os objetivos do jogo, o tipo de personagem e a forma como esta é controlada. Com base nestas características, o autor categoriza os videojogos em 42 diferentes géneros. Wolf (2000) segue um caminho, cuja taxonomia é mais abrangente, sem grandes preocupações na categorização dos géneros como Crawford fez. Géneros ainda não mencionados até agora, como os jogos de demo, de diagnóstico, de ritmo e dança, e de utilidade. Wolf também especifica outros géneros, que apesar da sua originalidade, estão ligados em muitos aspetos. Exemplos destes são por um lado, os jogos de captura, de apanhada, de caça e de coleta, e por outro lado os jogos de esquivar, de escapar, de labirinto e de pistas de obstáculos (Wolf, 2000). Como observado anteriormente, o autor, em cada um destes géneros é único e existem muitos jogos que são classificados de acordo com estes géneros. No entanto, as suas descrições não indicam a proximidade entre eles. A primeira seleção, como os próprios nomes sugerem, envolve a busca e agrupamento de objetos enquanto a segunda seleção será revertida para evitar objetos, ou descobrir o caminho correto para alcançar um determinado objetivo.

Outra opinião sobre a taxonomia dos videojogos provém de Rollings e Adams (Rollings & Adams, 2003, p. 287). Estes autores apresentam a seguinte categorização de géneros de videojogos: jogos de ação, de estratégia, de *role-playing*, de desporto, de simulação de carros, de construção e gestão de simulações, de aventura, de vida artificial, de puzzles e jogos online. O género de videojogos que até agora ainda não foi mencionado, tanto pelo Crawford (1982) como pelo Wolf (2000), foi o género de jogos on-line. Talvez pelo facto de esses autores terem apresentado as suas ideias antes do negócio dos jogos on-line se expandir. Todavia, alguns autores mais recentes não o consideram um género, porque, na verdade, é mais uma tecnologia do que uma categoria de videojogos (Rollings & Adams, 2003, p. 489).

Uma visão mais recente, que será detalhada a seguir, é a de Lucas e Sherry (2004), que através das suas investigações a *sites* de jogos, revistas e lojas de jogos, acabaram por identificar 13 categorias de jogos (Lucas & Sherry, 2004, p. 511). Ventura (2009) achou pertinente acrescentar a esta lista a categoria de plataformas:

Ação e aventura - (por ex.: Prince of Persia, Splinter Cell ou Tomb Raider) é uma categoria mista, onde estão presentes algumas das características de outros géneros de videojogos: a resolução de quebra-cabeças, características dos jogos de aventura com elementos de ação, o jogador luta com e sem armas, corre e salta. Nesta categoria, a personagem é visualizada e controlada na terceira pessoa, sendo possível vê-la de corpo inteiro, ou apenas de cintura para cima.

Figura 7 - Prince of Persia: The Forgoten Sands

Arcade - Os jogos de *arcade* inicialmente eram vistos em máquinas de salões de jogos. Hoje em dia estes jogos podem ser descarregados para os telemóveis, emuladores de computador, e até on-line através de plataformas ou redes sociais como o Facebook. Bons exemplos deste tipo de jogo são: Pinball, Pac-man e Space Invaders.

Figura 8 - Pacman

Cartas ou dados - nesta categoria estão presentes os jogos tradicionais jogados com cartas e/ou dados, que de certa forma foram adaptados para jogos em formato digital (por ex.: Solitário e Vegas Fever 2000).

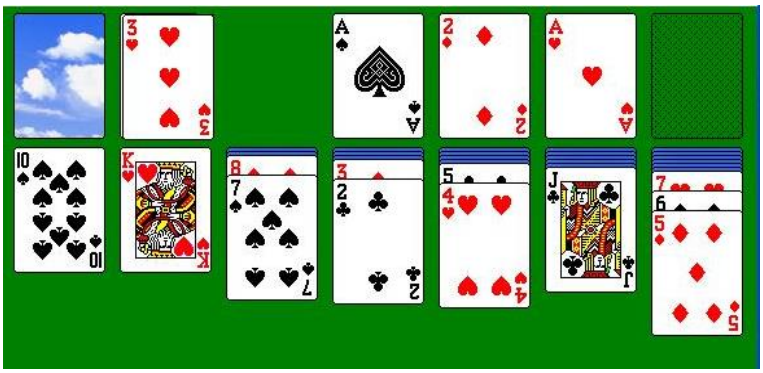


Figura 9 - Solitário da Microsoft

Corridas e velocidade - este tipo de jogo tem como principal objetivo de ganhar diversas corridas, através de uma condução de veículos a alta velocidade, onde é necessária perícia de condução para atingir os objetivos. Por vezes é permitida a escolha de veículos e personalização dos mesmos. (por ex. Grand Turismo, Need for Speed, Grid e Moto Racer).

Figura 10 - Grid 2

Clássicos de mesa - nesta categoria encontram-se os videojogos que se baseiam sobretudo em jogos de mesa tradicionais (por ex. Monopoly e Chess).

Figura 11 - Chess da Microsoft

Desporto - Pro Evolution Soccer, Wii Sports Resort e NBA Live são alguns exemplos de jogos de desporto de diferentes tipos. Consistem em equipas e eventos que existem na realidade, tais como futebol, basquetebol, golf, canoagem, entre outros.

Figura 12 - FIFA 13

Estratégia - normalmente, este tipo de jogo, tem a característica importante de possibilitar o jogador de controlar os elementos de jogo "visto de cima". Esta forma de visualização facilita o desempenho do jogador do ponto de vista de estratégia. Dentro desta categoria, existem dois tipos de estratégia: em tempo real/sem paragens (por ex. Desperados e Command & Conquer), e estratégia baseada em turnos, onde o jogador tem que esperar pelo seu turno para jogar (por ex. primeiro lançamento da série Commandos e Rome Total War). Os objetivos nesta categoria, normalmente, baseiam-se na conquista de territórios e/ou vencer batalhas/guerras/combates.

Figura 13 - Command & Conquer 4

Fantasia/role-playing - esta categoria tem a particularidade de dar ao jogador uma vasta lista de opções de personalização da sua personagem, tais como a sua raça, sexo, aparência, habilidades (inteligência, força, pontaria, entre outros), ocupação, entre outros. É comum haver combates entre jogadores em modo de competição ou em colaboração. Alguns permitem jogar em *single-player* (por ex. Neverwinter Nights ou Diablo), outros em *multi-player*. Hoje em dia, são cada vez mais comuns os *massive multi-player online* (MMORPG) como é o caso do World of Warcraft, Perfect World, SilkRoad, Sword of New World e Aion.

Figura 14 - Aion

Luta - o objetivo principal é a luta entre dois adversários. Um jogo desta categoria pode ser jogado por apenas um jogador contra o computador, ou dois jogadores combatendo entre si. Dependendo dos jogos, os personagens podem usar artes marciais, armas ou até poderes especiais (por ex. Street Fighter e Mortal Kombat).

Figura 15 - Mortal Combat

Plataformas - nesta categoria os personagens principais percorrem uma plataforma aos saltos, por vários obstáculos, coletar itens, e por vezes conseguem atirar objetos ou disparar com armas (por ex.: Super Mário, Sonic e Donkey Kong).

Figura 16 - Sonic

Puzzle - os jogos de puzzle concentram-se em desafios lógicos e conceptuais, embora ocasionalmente os jogos acrescentem a pressão de tempo ou outros elementos de ação (por ex. Tetris, Free Cell).

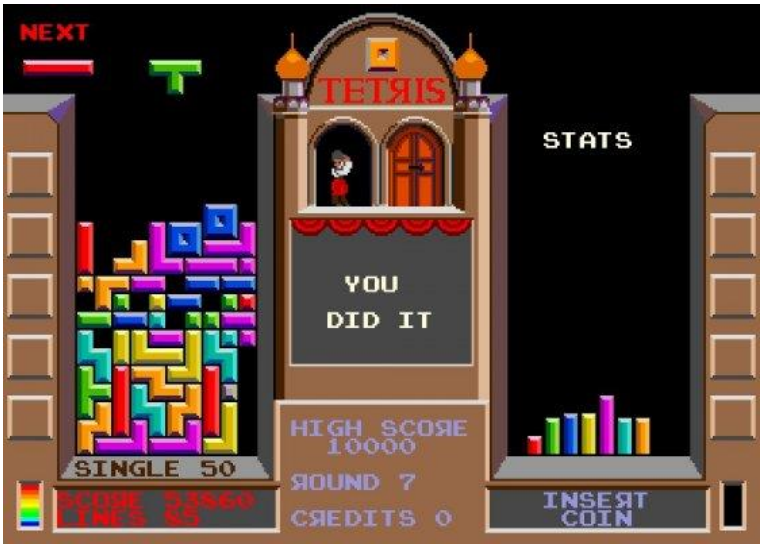


Figura 17 - Tetris

Quiz/trivia - objetivo principal, neste tipo de jogo, é responder corretamente às questões propostas. O número de repostas corretas ou a quantidade de dinheiro acumulada, são variáveis imprescindíveis para o resultado final que determinará a vitória ou derrota do jogador. Alguns destes jogos são baseados em programas da televisão ou em jogos de mesa (por ex.: Buzz, Jeopardy e Who Wants to be a Millionaire).

Figura 18 - Buzz

Simulação - nesta categoria Wolf (2000), afirma que os jogadores têm que utilizar os recursos (normalmente limitados) de forma controlada para construir ou expandir algum tipo de comunidade, instituição ou império. Ao mesmo tempo têm barreiras tais como o crime, a poluição, as catástrofes ambientais e monstros, assim como acontece com o jogo SimCity. Também pode acontecer que o objetivo principal seja a simulação de situações e atividades da vida real, como acontece com o The Sims, onde os personagens podem interagir com objetos dentro de casa e manter relações sociais com a vizinhança.

Figura 19 - Sims 3

Shooter - é um termo usado por Lucas e Sherry (2004) e por outros autores, no entanto, é mais conhecido por FPS (*First Person Shooter*), onde o jogador joga na perspetiva de primeira pessoa. Trata-se de um tipo de jogo onde há muita ação (uso de armas, situações de combate, entre outros), e é um sistema de jogo tridimensional onde a imersão ou sensação de proximidade da realidade, prevalece em maior percentagem em relação às outras categorias de videojogos. Grandes exemplos de sucesso deste tipo de jogo: Half-Life, Halo, Unreal Tournament, Wolfenstine, Doom, entre outros. Outro grande exemplo é a serie Battlefield, com o lançamento mais recente de Battlefield 3, que para além de ser um jogo FPS, também é considerado um jogo de estratégia.

Figura 20 - Battlefield Bad Company 2 Vietnam

Existem muitas perspetivas sobre a categorização dos videojogos, e como Crawford referiu, não existem taxonomias certas ou erradas. Qualquer taxonomia escrita é o resultado da interpretação dos videojogos por parte do autor.

3.2. Game Design

Game design é o processo de conceção do conteúdo e regras de um jogo na fase de pré-produção e do design do *gameplay*, ambiente, e narrativa. Dennis Sasse (2008) vê o *game design* de forma semelhante, onde estão presentes 4 aspetos importantes: narrativa, a estimulação, o nível de desafio, e mecânica de jogo. Na perspetiva de Crawford (1982) *game design*, antes de mais, é um processo artístico, mas também é um processo técnico. O termo também é usado para descrever tanto o *design* inerente de um jogo bem como a documentação que descreve um tal desígnio. *Game design* exige competência artística e técnica, bem como habilidades de escrita (Rollings & Adams, 2003).

Todd (2007), uma veterana em duas indústrias de entretenimento (Hollywood e videojogos), realça a importância que as personagens têm no *gameplay* de um jogo. Ao fazer um jogo que é rico em história, rico em mecânica, rico em gráficos, ou qualquer combinação dos três, uma coisa que está a tornar-se mais consistente e mais importante é forma como o jogador se relaciona com os personagens. Os criadores de jogos gastam uma quantidade significativa de tempo a abordar a composição das personalidades dos seus personagens, como se relacionam com a história e a experiência de jogo que estes estimulam (Todd, 2007, p. 71).

A conceção de videojogos, normalmente, é composta por vasta equipa de pessoas com diferentes competências e em diferentes áreas. A equipa poderá ser constituída da seguinte forma: artista, engenheiro de som, *designer*, escritor, programador, produtor, *tester* e marketing. Todd (2007) questiona-se sobre *"What makes a great game?"*. A Tabela 2 apresenta diferentes respostas, que variam conforme os diferentes papéis que os membros desempenham dentro da equipa.

Membro da equipa	Resposta
Artista	*You have to have killer graphics. Look at the number of artists working on a project. It's definitely the art, art direction, cinematics, theoverall look that is key to making a great game.*
Engenheiro de som	*Art is really important, it's true, but you can have cool graphics and if your music and sound effects are lousy you'll have a game nobody wants to play. Music is far more important than people might think. So, defi nitely, the music and quality of the sound recording are what I think make a great game.*
Designer	*It all starts with really solid game design. If you don't have a good design, good structure, integrating good mechanics and gameplay, all fi tting together, the whole thing falls apart. You can't even get it off the ground.*
Escritor	*You have to have characters that people will want to play, and of course you need a story wrapped around the game that makes sense. There's a lot that goes into it. Who are your characters? How do they relate to other characters? Where and how do they fi t in this environment and in the story? What is the story? The dialog has to be tight. It all has to be engaging. The story has to be compelling. The cut scenes have to be fi gured out and well delivered. Why is the player playing the game? Why are they here? Why do they care? It has to support all of these elements of the game. So, defi nitely it's story and characters that will engage the player.*

Programador	*The technology. The mechanics. What can we do that nobody's ever done before? How do we take this great mechanic and make a hot new game? Defi nitely the technology.*
Produtor	*Teamwork. Communication.*
Tester	*Getting the games in front of the users as early as possible, listening to what they say, fi nding out where you get held up, where the problems are, where things are working and where they aren't, and incorporating what you learn into the game so that when you deliver, it works.*
Marketing	*I don't care how good your game is, if you don't have marketing behind it you'll have a flop.*

Tabela 2 - Perspetivas de como fazer um bom jogo

Como é óbvio, cada um tende a focar a sua perspetiva na sua área de saber. Porém, todos eles estão corretos, mas por vezes uns mais que outros (Todd, 2007, p. 10).

Salen e Zimmerman (2005) referem que o *game design* deve ser interativo. A diversão e excitação de jogar não podem ser calculados de uma forma abstrata: estes devem ser experiênciados ou vividos (Salen & Zimmerman, 2005). Também é importante que uma equipa se conjugue bem, para que o produto final seja o mais agradável possível para o utilizador final e este tenha uma experiência de jogo estimulante, já que é esse um dos objetivos dos videojogos.

3.3. Mecânica de jogo (Gameplay)

Björk, Lundgren e Holopainen estudaram as várias comunidades de jogadores e *designers* de videojogos e descobriram que muitos deles utilizavam o conceito de mecânica ou mecanismos (Björk, Lundgren, & Holopainen, 2005). "*Part of a game's rule system that covers one general or specific aspect of the game*" (Boardgamegeek apud. (Björk et al., 2005)). Os mesmos afirmam ainda que esta definição de *game mechanics* é geral e não é útil para investigação académica.

Segundo Federoff (2002) mecânica de jogo remete ao conceito da física do mundo de um jogo. Ou seja, quais habilidades das personagens, a dinâmica de um determinado elemento ou objeto do jogo. Estes são desenvolvidos através de uma combinação de animação e programação (Federoff, 2002, p. 11).

Um típico de mecânica comum pode ser "rolar e mover-se", que simplesmente diz que os dados são lançados e que outra coisa (peça) é movida dependendo do resultado da jogada. A mecânica não diz como algo deve ser movido ou porquê, isso é determinado nas regras do videojogo.

Os *designers* de videojogos também costumam usar com frequência o termo de mecânica, mas o termo não é estritamente definido - este é usado tanto na forma como é utilizado para jogos de tabuleiro, como em contextos de programação técnica (Lundgren, 2002 apud. (Björk et al., 2005)).

Na visão de Björk e Holopainen, mecânica é um termo que engloba as regras que são aplicadas quando o jogador interage com o jogo, e não há necessidade de uma distinção entre as regras

e a mecânica. Mecânicas de jogo seriam descrições de baixo nível das regras do jogo ou dos grupos de regras do jogo.

Sicart (2008) já tem uma perspetiva diferente dos anteriores autores. "*Game mechanics are methods invoked by agents, designed for interaction with the game state*" (Sicart, 2008).

Segundo Sicart (2008), Järvinen (2008) afirma que a melhor forma de compreensão de mecânicas como métodos é formaliza-las como verbos, com outros elementos sintácticos/estruturais, tais como regras, tendo influência sobre a forma de atuação destes verbos no videojogo. Por exemplo, em *Shadow of the Colossus*, encontra-se as seguintes mecânicas: escalar, montar (o cavalo), esfaquear, pular, atirar (setas), apitar, agarrar, correr (e variações, como nadar ou mergulhar). No caso de *Gears of War* seria: proteger, atirar, recarregar, mandar (granada), olhar (num ponto de interesse), usar, dar ordens, mudar de arma. Todos estes verbos são métodos de atividade dentro do mundo do jogo, ações que o jogador pode assumir dentro do espaço possível criado pelas regras (Sicart, 2008).

Outra abordagem interessante de Sicart (2008) é a possibilidade de descrever *mechanics* que são acionados em função do contexto da presença dos jogadores no videojogo, que acaba por definir como "*context mechanics*". *Contextual mechanics* são conceitos analíticos que podem ser usados para entender como os jogadores descodificam a informação num nível de jogo, ou seja, como um jogador percebe certas estruturas e como essas estruturas são utilizadas para comunicar utilizações pretendidas ou comportamentos (Sicart, 2008). Sicart acrescenta ainda que nesta definição está implícita uma diferença ontológica entre as regras e a mecânica. Os *game mechanics* estão focados na interação atual com o estado do jogo, enquanto que as regras fornecem o possível espaço onde a interação é factível, regulando assim a transição entre estados. Posto isto, pode-se dizer que as regras são modeladas a partir da atividade, enquanto que a mecânica é modelada para a atividade.

Sicart refere que num contexto orientado a objetos, as regras poderiam ser consideradas propriedades gerais ou especiais do sistema de jogo (todos os objetos nos jogos têm propriedades). Essas propriedades são frequentemente regras ou determinadas pelas regras. Estas regras são avaliadas por um *loop* de jogo, um algoritmo que relaciona o estado atual do jogo e as propriedades dos objetos com uma série de condições que, consequentemente, pode modificar o estado do jogo. Por exemplo, a condição de vencedor, a condição de perder e os efeitos das ações na vida do personagem são calculadas durante a execução do *loop* de jogo. Este algoritmo relaciona as regras com a mecânica, exemplificando a aplicabilidade de uma distinção ontológica entre as regras e a mecânica (Sicart, 2008).

Mesmo que a definição de Sicart de *game mechanics* determine que os jogos são estruturados como sistemas com mecânica, regras e desafios, entendida como a gramática essencial dos jogos de computador (e provavelmente de todos os jogos), há mais o ato de jogar um jogo que apenas interage com mecânica limitada por regras. Além disso, pode acontecer que o que foi concebido como uma mecânica de jogo seja usado num comportamento *non-gameplay*: os jogadores de *Shadow of the Colossus* usam a mecânica de escalada para alcançar algumas das áreas mais distantes do mundo do jogo, que não teve influência, ou interesse, para a sequência central do *gameplay* e narrativa do jogo. O *game mechanics* é desenhado para o *gameplay*, mas pode ser usado para *toyplay* (Bateman e Boon, 2006 apud. (Sicart, 2008)). A

única variação seria o nível de abstração: por um jogador que está a jogar o jogo, uma mecânica serve um conjunto específico de efeitos, enquanto que um jogador que está a jogar com ou dentro do jogo, o *game mechanics* perde a sua origem formal do projeto do jogo e torna-se um instrumento para a atividade (Sicart, 2008).

Existe alguma confusão quanto à diferença entre a mecânica de jogo e o *gameplay*. Para alguns autores, o *gameplay* não é nada mais do que um conjunto de mecânicas de jogo. Para outros, *gameplay* - especialmente quando relacionado no termo de "*gameplay* básico" - refere-se à essência das mecânicas de jogo que determinam as características gerais do jogo em si. Segundo Rollings e Adams (2003) é difícil de definir o conceito de *gameplay*. Cada autor tem a sua definição pessoal de *gameplay*, que é formada ao longo das suas carreiras. Para além disso, é difícil definir *gameplay*, porque não existe uma entidade única que se pode apontar e dizer *"There! That's the gameplay"* (Rollings & Adams, 2003). Para Federoff (2002) o *gameplay* não é nada mais que um processo através do qual o jogador realiza o seu objetivo. Refere-se aos desafios e problemas que um jogador precisa de superar para vencer o videojogo (Federoff, 2002, pp. 11-12).

O *game mechanics* acaba por ser uma construção de regras destinadas a produzir um jogo agradável ou com *gameplay*. Todos os jogos usam mecânica, no entanto, as teorias e os estilos diferem quanto à sua importância definitiva para o jogo (diferentes géneros - diferentes mecânicas). Em geral, o processo e estudo de *design* do jogo é o esforço para chegar ao game *mechanics* que permite que as pessoas joguem um jogo tendo uma experiência divertida e envolvente.

3.4. Game Experience

Nacke et al. (2009) argumenta que a jogabilidade (*playability*) é o processo de avaliação direcionado para jogos, enquanto que a experiência do jogador (*player experience*) é direcionado para os jogadores. Mais precisamente, os métodos de jogabilidade avaliam os jogos com fim em melhorar o *design*, enquanto que os métodos de experiência de jogador avaliam os jogadores para melhorar os jogos. Nacke et al. (2009) cria um esquema que mostra que essa separação de conceitos torna-se importante no processo de *design* de jogos, principalmente dentro de uma equipa de investigação de utilizadores para decidir quais os métodos para implementar e em que fase do processo (L. E. Nacke et al., 2009).

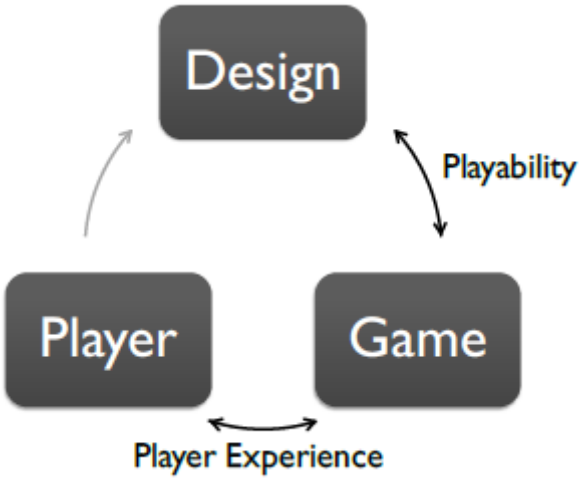


Figura 21 - Interface entre jogador, jogo e *design* [retirado de (L. E. Nacke et al., 2009, p. 1)]

As interfaces entre o jogador, jogo e *designer* de jogos mostram que a jogabilidade é direcionada para avaliação de *design* de jogos, onde a experiência do jogador tem que ser analisada no processo de interação jogador-jogo (L. E. Nacke et al., 2009).

A jogabilidade pode ser definida como um conjunto de propriedades que descrevem a experiência do jogador usando um sistema de jogo específico, cujo principal objetivo é proporcionar diversão e entretenimento, quando o jogador joga sozinho ou acompanhado (Sánchez, Zea, & Gutiérrez, 2009). A jogabilidade é caracterizada por atributos e propriedades diferentes para medir a experiência de um jogador de videojogos:

- **Satisfação -** o grau de satisfação ou prazer do jogador para com um videojogo dependente de alguns dos seus aspetos como: mecânica, gráficos, interface do utilizador, história, entre outros. Satisfação é também um atributo com uma elevada subjetividade que provoca uma medição difícil devido às preferências e prazeres que o jogador tem. Isto tem influência na satisfação para com os elementos específicos do jogo: personagens, mundo virtual, desafios, e assim por diante.
- **Aprendizagem -** a facilidade de compreender e dominar o sistema de jogo e a respetiva mecânica (objetivos, regras, como interagir com o videojogo, entre outros).
- **Eficiência -** definido com o tempo e os recursos necessários para oferecer diversão e entretenimento aos jogadores quando estes atingem os diferentes objetivos de jogo e chegam ao objetivo final. Um videojogo eficiente é capaz de capturar a atenção do jogador desde o primeiro instante, e provocá-lo a continuar a jogar até o fim do jogo.
- **Imersão** - a capacidade de acreditar no conteúdo de videojogo e integrar o jogador no mundo virtual do mesmo. A imersão provoca uma sensação ao jogador de estar envolvido no mundo virtual, tornando-se parte do presente e interativo com ele.
- **Motivação** - características que provocam o jogador a realizar ações concretas e persistir nelas até ao seu ponto culminante. Para obter um alto grau de motivação, o jogo deve ter um conjunto de recursos para garantir a perseverança dos jogadores nas ações realizadas para superar os desafios do jogo (tornar um comportamento positivo, atingir recompensas pelos atos, promover a confiança e prazer ao alcançar os objetivos).

- **Emoção -** impulso involuntário, originado em resposta ao estímulo do videojogo e induz a sentimentos ou desencadeia reações automáticas e condutas. A utilização das emoções nos videojogos ajudam a obter uma melhor experiência ao jogador e leva os jogadores a diferentes estados emocionais: felicidade, medo, intriga, curiosidade ou tristeza, usando os desafios do jogo, história, aparência estética ou as composições musicais que são capazes de comover, fazer sorrir ou a chorar a um jogador.
- **Socialização -** grau do conjunto de atributos de jogo, elementos e recursos que promovem o fator social da experiência de jogos em grupo. Esse tipo de experiência é uma forma diferente de jogar, graças às relações que se estabelecem com outros jogadores ou com outros personagens de jogo que ajudam o jogador a resolver em conjunto os desafios do jogo de forma colaborativa, competitiva ou cooperativa (interação - ex.: pedir ajuda na resolução de um determinado problema do jogo, ou negociar um determinado item de jogo com outro jogador).

Uma boa jogabilidade de um jogo deve ser um pré-requisito para a avaliação do *game experience* (L. E. Nacke et al., 2009). Um *game design* não deve conter quaisquer problemas que possam ficar no caminho de uma experiência de jogo individual. Um método de avaliar a jogabilidade é a revisão por especialistas (*expert review*) ou heurísticas, que é uma técnica de custo/eficiência e efetiva para identificar problemas de jogabilidade e usabilidade, tema que será continuado no capítulo seguinte.

4. Usabilidade na Avaliação e Conceção dos videojogos

Conta-se que a indústria dos videojogos está num crescimento exponencial e que gera biliões de dólares em vendas. A necessidade de oferecer mais jogos agradáveis já foi estudado por várias pessoas. O problema surge no "como" fazer isso. Neste capítulo, serão abordados alguns dos trabalhos já realizados nesta área e como estes contribuíram para melhorar os videojogos e experiência geral dos videojogos. Além disso, neste capítulo são apresentadas algumas das investigações existentes e projetos feitos na área de desenvolvimento de videojogos que não usam a tecnologia de *eye tracking*, bem como alguns que adotaram a tecnologia explorada. Também são abordados alguns projetos que optaram por usar *logfiles* para avaliar videojogos.

Para complementar este capítulo, são discutidos alguns temas importantes na área da usabilidade, nomeadamente, sobre o HCI, sobre usabilidade em geral, e posteriormente sobre usabilidade em videojogos (*game usability*).

4.1. HCI e usabilidade

No contexto da Interação Humano - Computador (HCI - *Human-computer interaction*) que tem promovido continuadamente o conceito de usabilidade, conceito onde está inerente a preocupação de avaliar, retificar ou sugerir melhores formas de interação, de forma a que um sistema cumpra os objetivos para o qual foi desenhado/construído, tornando assim a sua utilização mais acessível a qualquer tipo de utilizador. Desta forma, a principal finalidade é facilitar a vida dos utilizadores na utilização e perceção dos recursos disponíveis pelos sistemas. Existem fatores, como a faixa etária, os níveis de conhecimento do utilizador, os tipos de aplicações, entre outros, que podem influenciar a interação entre o homem e o computador através da qualidade da interface. Assim sendo, devem-se ter em conta fatores humanos e ergonómicos. Contudo algo usável é algo que é fácil de aprender a usar, com um uso efetivo e que proporciona uma experiência agradável (Nielsen, 2003).

Conforme a *Internacional Standard Organization* (ISO/IEC) pode-se descrever a usabilidade como (ISO, 2009)):

- **ISO 9126** - "a usabilidade refere-se à capacidade de um *software* de ser compreendido, aprendido, utilizado e ser atrativo para o utilizador, em condições específicas de utilização";

- **ISO 9241-11** - "a medida em que um produto pode ser usado por utilizadores específicos para atingir objetivos com eficácia, eficiência e satisfação num contexto de uso específico" (UPA - *Usability Professionals' Association*).

De acordo com Jakob Nielsen[2], a usabilidade mede a qualidade da experiência do utilizador quando interage com o sistema. Algumas características de usabilidade que devem estar presentes num sistema são (Nielsen, 2003):

- **intuitividade** - consiste na facilidade de uso apresentada pelo sistema, permitindo o desenvolvimento de um trabalho de forma satisfatória, mesmo sendo realizado por um utilizador sem experiência;
- **utilização eficiente** - consiste na capacidade do sistema em ser eficiente no seu desempenho, apresentando um alto nível de produtividade. Está relacionado com a rapidez de execução de tarefas;
- **facilidade de memorização** - consiste na capacidade do sistema em apresentar facilidade de memorização, permitindo a sua utilização por utilizadores ocasionais depois de um intervalo de tempo relativamente longo;
- **minimização de erros** - a quantidade de erros apresentados pelo sistema deve ser o mínimo possível, além disso, eles devem apresentar soluções simples e rápidas mesmo para utilizadores inexperientes. Não podem ocorrer erros graves ou sem solução;
- **satisfação subjetiva** - consiste na capacidade do sistema em agradar aos utilizadores, permitindo uma interação agradável. É o grau de satisfação do utilizador no uso do sistema.

Rubin e Chisnell (2008) apresentam uma abordagem similar no conceito de usabilidade. O modelo destes autores é composto por seis atributos (Rubin e Chisnell, 2008 apud. (Almeida, 2009)): utilidade, eficiência, eficácia, capacidade de aprendizagem, satisfação e acessibilidade.

- **utilidade** - descreve a medida em que um produto permite ao utilizador completar os seus objetivos com precisão;
- **eficiência** - à semelhança do que foi mencionado a cima, esta é medida pela rapidez com que um utilizador pode completar a sua meta com precisão;
- **eficácia** - refere-se ao grau de realização do produto tal como o utilizador espera que aconteça;
- **capacidade de aprendizagem** - está relacionada com a eficácia e refere-se à capacidade do utilizador para usar, com um certo grau de competência, um produto após um determinado período de tempo;
- **satisfação** - assim como nas definições anteriores, refere-se às opiniões pessoais do utilizador pelo produto;
- **acessibilidade** - de acordo com Rubin e Chisnell, esta está ligada com a usabilidade. Contudo, o acesso concentra-se na forma que um produto é feito e utilizável por utilizadores com deficiência. A pergunta que se segue, no entanto é: o que faz um produto mais usável?

[2] Jakob Nielsen é um famoso consultor de usabilidade com doutoramento em Interação Humano-Computador da Technical University of Denmark

Segundo Rubin e Chisnell entender o que faz com que um produto possa ser menos útil é um passo importante para saber como fazer um produto mais funcional. Os autores sugerem três ideias para fazer com que os produtos sejam mais utilizáveis (Rubin & Chisnell, 2008 apud. (Almeida, 2009)): um enfoque nos estudos sobre utilizadores e as suas tarefas, avaliar e medir a utilização do produto, e finalmente, *design* de interação. No primeiro contato precoce entre os utilizadores finais e a equipa de *design* do produto, ao longo do desenvolvimento do produto, é importante identificar os principais utilizadores do produto, mas também é importante estabelecer uma conexão entre as duas partes que vai resultar no desenvolvimento de um produto que responde, de forma mais precisa, às necessidades de um utilizador. O segundo contato serve para avaliar e medir o produto, ambas as análises devem ser feitas constantemente ao longo do ciclo de desenvolvimento com ênfase na facilidade de aprendizagem, bem como a facilidade de usar o produto. Finalmente, no último, o *design* de interação e de teste deve ser um processo que acompanha o ciclo de desenvolvimento do produto e não apenas nos momentos finais. Como os autores indicam, o verdadeiro *design* de interação permite "moldar o produto" através de um processo de conceção, teste, redesenho e reteste de atividades. Estes três pontos são uma pequena parte de uma visão global - *design* centrado no utilizador (UCD - *User Centered Design*) - um conceito introduzido inicialmente por Donald Norman em Design of Everyday Things (Norman, 1998), que incidiu no projeto de produtos baseados nas necessidades dos utilizadores e com menos preocupação com a estética do produto. Rubin e Chisnell definem UCD, representando as técnicas, processos, métodos e procedimentos para a conceção de produtos e sistemas usáveis, assim como também é importante a filosofia que coloca o utilizador no centro do processo (Rubin & Chisnell, 2008 apud. (Almeida, 2009)).

O que foi visto até agora é apenas um vislumbre de como é complexo o *design* de usabilidade. Fazer um produto ou sistema utilizável para um vasto grupo de utilizadores não é de forma alguma um objetivo simples e requer muito debate dentro da equipa de desenvolvimento.

Assim como é importante saber quais são os atributos a seguir ao desenvolvimento de um produto usável também os métodos utilizados são importantes para avaliar a usabilidade de um produto, uma vez que este já esteja concluído. Entretanto, a avaliação de usabilidade não deve ser considerada a última fase do ciclo de desenvolvimento de um produto. O processo de avaliação deve ocorrer em paralelo com o desenvolvimento do produto. Porém, parece razoável que a avaliação intensiva não é possível durante o ciclo completo. Existem outras técnicas que podem ser aplicadas e utilizadas (Dix, Finlay, Abowd, & Beale, 1998, p. 406).

Dix et al. (1998, pp. 406-407) incide sobre HCI, afirmando que a avaliação tem três objetivos principais: (i) avaliar a extensão da funcionalidade do sistema, (ii) avaliar os efeitos da interface no utilizador; e (iii) identificar eventuais problemas específicos com o sistema. No que diz respeito à funcionalidade do sistema, esta deve ser capaz de atender as necessidades do utilizador, assim como as suas expectativas. A avaliação de um sistema em termos de interface significa que o impacto do *design* do sistema no utilizador deve ser avaliado, ou seja, a facilidade do sistema de uso e a capacidade de aprendizagem, bem como a atitude do utilizador perante o sistema. Finalmente, identificar problemas específicos com o sistema remete a identificação de problemas com um sistema que, no seu contexto de uso, causou erros inesperados, tanto em termos de funcionalidade como de *design*.

Muitos autores dividem e classificam os métodos de testes de usabilidade de diferentes formas. Dix et al. (1998, pp. 405-435) apresenta uma divisão hierárquica enquanto que Ben Shneiderman (1998, pp. 124-151) apresenta uma visão mais linear sobre o assunto. Apesar desta diferença, tanto Dix et al. Shneiderman e discutem técnicas e métodos idênticos.

Dix et al. apresenta dois estilos de avaliação (Dix et al., 1998, pp. 407-408):

- **estudos laboratoriais** - os estudos de laboratório, como o nome sugere, são os estudos realizados em laboratório. Os laboratórios podem ser equipados com tecnologia e em condições que os tornam em ambientes mais favoráveis. No entanto, a falta de contexto, como os autores definem, pode inibir, para alguns utilizadores, de uma avaliação mais natural. Contudo, o uso de um estudo de laboratório, em alguns casos é a única opção.
- **estudos de campo** - os estudos de campo levam o avaliador para o ambiente natural do utilizador, um lugar onde este se sente mais confortável. Ainda assim, a observação de campo apresenta muitas distrações possíveis que, por um lado, pode beneficiar a avaliação, mas por outro lado, também pode pô-la em causa.

Dix et al. (1998, pp. 408-415) separa a avaliação de *design* da avaliação de implementação. O *cognitive walkthrough* e a avaliação heurística são dois métodos importantes de avaliação de *design* de um produto. A base principal do método *cognitive walkthrough*, tal como sugerido por Dix et al. (1998), é verificar como um sistema é fácil de aprender. Neste método um avaliador apresenta uma lista de passos que o avaliando deve seguir para completar uma determinada tarefa. O segundo método (avaliação heurística) centra-se no seguimento de uma lista de regras que orientam uma decisão de projeto ou que complementam uma crítica de uma decisão já tomada (Dix et al., 1998, p. 412). O conceito de avaliação heurística foi introduzido por Jakob Nielsen e Rolf Molich em 1990. Normalmente, a avaliação heurística é realizada por avaliadores diferentes que independentemente avaliam e analisam um determinado sistema à procura de potenciais problemas de usabilidade. Quantos mais avaliadores estiverem envolvidos, maior é a probabilidade de encontrar um maior número de problemas de usabilidade. Dez são o número de regras que compõem a lista de heurísticas utilizadas na avaliação heurística, tal como apresentado na seguinte tabela (Tabela 3).

Heurística	Descrição
Visibilidade do estado do sistema	O sistema deve sempre manter os utilizadores informados sobre o que está a acontecer, através de *feedback* apropriado e em tempo razoável.
Compatibilidade do sistema com o mundo real	O sistema deve falar a língua do utilizador, com palavras, frases e conceitos familiares ao este, em vez de termos orientados ao sistema. Deve seguir as convenções do mundo real, para que a informação apareça numa ordem lógica e natural.
Controlo e liberdade do utilizador	Os utilizadores frequentemente escolhem funções do sistema por engano e vão precisar de uma "saída de emergência" para sair do estado indesejado sem terem que passar por um extenso diálogo. Dar a possibilidade ao utilizador de voltar atrás (desfazer e refazer).
Consistência e padrões	Os utilizadores não precisam de adivinhar que diferentes palavras, situações ou ações significam a mesma coisa. O sistema

	deve seguir as convenções da plataforma.
Prevenção de erros	Ainda melhor do que boas mensagens de erro é um projeto cuidadoso que impede que um problema ocorra. Eliminar as condições de erro propício ou verificá-las e apresentar ao utilizador uma opção de confirmação antes de se comprometer com a ação.
Reconhecimento em vez de lembrança	Minimizar a peso de memória do utilizador, tornando os objetos, ações e opções visíveis. O utilizador não deve ter que estar a lembrar-se de informação sempre que muda de sítio. As instruções para a utilização do sistema devem estar visíveis e facilmente recuperáveis quando necessário.
Flexibilidade e eficiência de uso	Uso de aceleradores para facilitar e acelerar a interação do utilizador avançado. Deve estar preparado tanto para utilizadores experientes, como para inexperientes. Permitir aos utilizadores adaptar ações frequentes.
Estética e *design* minimalista	Os diálogos não devem conter informação irrelevante ou raramente necessária. Cada unidade extra de informação num diálogo pode pôr em causa as unidades relevantes de informação e diminuir a sua visibilidade.
Ajudar os utilizadores a reconhecer, diagnosticar, e corrigir erros	As mensagens de erro devem ser expressas em linguagem clara e concisa (sem códigos). Devem também indicar com precisão o problema e construtivamente sugerir uma solução.
Ajuda e documentação	Mesmo que seja melhor que o sistema seja usado sem documentação, pode ser necessário fornecer ajuda e documentação. Toda a informação deve ser fácil de pesquisar, focada nas tarefas do utilizador, lista de passos concretos a serem realizados, e não deve ser muito extensa.

Tabela 3 - As dez heurísticas de Nielsen

Estes princípios são chamados de "heurísticas" porque estão mais na natureza das regras de ouro do que as *guidelines* de usabilidade específicas (Nielsen, 2005).

Quanto à implementação de avaliação, Dix et al. (1998, p. 415) apresenta três métodos possíveis:

- **método empírico: avaliação experimental -** um dos métodos mais fortes para avaliar, fornece evidências empíricas para sustentar uma hipótese proposta, bem como a sua potencial utilização nos contextos mais diversos. Qualquer estudo empírico (ou experiência) segue os mesmos procedimentos: o avaliador escolhe a hipótese que deseja testar e que pode ser determinada por medição de algum atributo do comportamento de um indivíduo. Então o avaliador considera as condições experimentais, que por usa vez são influenciadas pelos valores de determinadas variáveis. Apesar desta visão simplista do método empírico, fatores como as experiências com indivíduos, variáveis escolhidas para testar e manipular, bem como as hipóteses testadas são cruciais.
- **técnicas de observação -** Dix et al. (1998, pp. 427-431) divide este método em 4 partes: processo de “pensar em voz alta” ou “pensar alto” (*Think aloud*), a análise do protocolo, a análise automática de protocolo e *post-task walkthrough*. Pensar em voz

alta é um processo utilizado para observar a interação dos utilizadores com o sistema enquanto completam uma série de tarefas, ou numa segunda instância, observar o dia-a-dia do utilizador. Enquanto os utilizadores completam as suas tarefas, estes são convidados a "pensar alto", ou seja, descrever o que sentem, porque o fazem e o que estão a tentar fazer. Os resultados neste método podem ser subjetivos e seletivos, dependendo da tarefa a ser realizada. O *post-task walkthrough* consiste em reproduzir e analisar com o sujeito que fez durante a tarefa, para uma maior compreensão. O que quer dizer que em muitos casos os dados obtidos através da observação direta não são completos ou totalmente compreensíveis. Quanto aos outros dois métodos, esses não são relevantes para o estudo presente.

- **técnicas de consulta -** também são utilizadas para a avaliação (Dix et al., 1998, pp. 431-435). Apesar de serem menos controladas pelo analista, estas técnicas podem fornecer informações úteis sobre a opinião do utilizador em relação ao produto ou sistema. Positivamente, são métodos simples e de baixo custo de avaliação. Porém, estes retornam resultados muito com alguma ambiguidade, o que pode ser difícil de compreender. Consultas técnicas são divididas em duas categorias principais: entrevistas e questionários (Dix et al., 1998, p. 432). As entrevistas são um meio eficaz de obter informações dos utilizadores relacionadas com as suas preferências, atitudes e impressões, bem como detetar os problemas que o analista não tinha previsto. Para efeito pleno, as entrevistas devem ser planeadas com antecedência para que a consistência seja mantida. Existem vários tipos de questionários, cada um com uma finalidade diferente. Estes levam menos tempo para ser concluídos, dando a possibilidade de concentração em questões mais específicas. No entanto, a elaboração de um questionário exige uma reflexão anterior sobre o tipo de informação que o analista espera obter.

Como já foi referido, Shneiderman (1998) também apresenta a sua perspetiva no que diz respeito aos métodos de avaliação de usabilidade. A sua lista é composta por testes de usabilidade em laboratórios, opiniões de especialistas (*expert reviews*), inquéritos, testes de aprovação, avaliação durante o uso ativo, experimentos orientados e controlados psicologicamente, resumo do praticante e agenda do investigador. Como os 5 primeiros métodos de Ben Shneiderman são os únicos relevantes para este trabalho, também serão os únicos a serem discutidos a seguir:

- **teste de usabilidade em laboratório** - como o nome indica, refere-se ao teste num ambiente de laboratório (Shneiderman, 1998, pp. 127-132). Como foi visto com Dix et al. (1998, p. 407), os testes em laboratório oferecem condições controladas e a possibilidade de obtenção de respostas mais concretas para os problemas. Os laboratórios de testes de usabilidade discutido por Shneiderman são aqueles com especialistas com conhecimentos na área de testes de usabilidade e que executam testes como o que foi discutido anteriormente (e.g. processo de *post-task walkthrough*).
- o método de **opiniões de especialistas** apresentado por Shneiderman (1998, pp. 125-127) inclui vários métodos: a avaliação heurística à semelhança da avaliação sugerida por Dix et al. (1998), revisão de *guidelines*, a inspeção de consistência, *cognitive walkthroughs* e inspeções de usabilidade formal. De certa forma, alguns destes

métodos já foram referenciados acima, contudo, no que diz respeito à avaliação heurística de Shneiderman, existem alguns aspetos diferentes em relação à avaliação de Dix et al. (1998, pp. 412-414), como por exemplo, a composição pelas oito regras de ouro (Shneiderman, 1998, pp. 74-75).

- o uso de **inquéritos** (Shneiderman, 1998, pp. 132-135) compara-se aos questionários de Dix et al. 's (Dix et al., 1998, pp. 432-435). Ambos procuram obter opiniões dos utilizadores sobre o uso e a funcionalidade de um sistema ou produto.
- **testes de aprovação** - lista de pré-requisitos que um produto deve seguir, ou que um utilizador deve ser capaz de fazer. Caso isso não aconteça, o produto ou sistema deve ser revisto até que este cumpra os resultados esperados (Shneiderman, 1998, p. 135).
- **avaliação durante o uso ativo** - refere-se à avaliação de um sistema depois de estar em uso. Shneiderman sugere algumas abordagens como a obtenção de feedback através de entrevistas e "*focus groups*", *logging* do desempenho dos utilizadores são duas de várias sugestões que Shneiderman menciona (Shneiderman, 1998, p. 145).

Nesta presente secção mencionaram-se alguns dos métodos de avaliação de usabilidade usados para a avaliação do produto ou sistema. Todavia, muitas destas técnicas foram pensadas especialmente para os sistemas e produtos, como *sites web* ou aplicações desktop. O problema agora é associar videojogos e usabilidade. Tradicionalmente, estes são dois conceitos com pouca correlação apesar do facto de que os videojogos, sendo eles próprios um produto, devem ser usáveis. Porém, na secção seguinte poderemos ver que a compreensão do conceito de usabilidade é uma tarefa difícil, especialmente quando associada a videojogos.

4.2. Game Usability

Os videojogos e usabilidade são dois conceitos que poderiam ser facilmente interconectados. Porém, como veremos mais adiante nesta secção, estes são difíceis de combinar. No estudo de Melissa Federoff (Federoff, 2002), ela descobre que mesmo para as pessoas da área de videojogos, o conceito de *game usability* não é usado ou mesmo desconhecido. No entanto, esta situação não é assim tão grave quanto parece, afinal, os videojogos não passam de um produto cujo principal objetivo é entreter o público e não tanto uma ferramenta para facilitar as tarefas diárias de um utilizador.

Uma das primeiras tentativas de abordagem de *game usability* surgiu em 1980, por Thomas Malone (Malone, 1980 apud. (Almeida, 2009)). Mais recentemente, Chuck Clanton (Clanton, 1998) dividiu alguns dos principais problemas de *game usability* em três categorias: *game interface*, *game mechanics* e *gameplay*. Estes dois últimos termos já foram mencionados no secção 3.3. Quanto ao *game interface* refere-se à parte visual e motora de um jogo, por exemplo, como o controlador funciona ou quais os instrumentos e elementos que são exibidos no ecrã. Federoff (2002) descreve *game interface* como os elementos que são realmente utilizados para controlar um videojogo: teclado, controlador de videojogos, joystick ou rato. Além disso, o *game interface* é a representação visual das diversas ações que o jogador executa num jogo: movimentar-se pelo jogo, começar o jogo, sair do jogo, entre outros (Federoff, 2002).

No trabalho de Melissa Federoff (2002, pp. 12-13), ela reconhece que a questão da usabilidade, tanto nos videojogos como noutros produtos ou sistemas, está sempre dependente do contexto em que se encontra. Como será verificado na próxima secção, Federoff foi capaz de associar as questões de usabilidade tradicionalmente ligadas a outros produtos, com os videojogos, desenvolvendo uma lista de heurísticas de *game design*, para avaliação de usabilidade.

As heurísticas de Nielsen estão normalmente focadas para avaliar interfaces de produtos e *design*, no entanto, estas também podem ser úteis para a avaliação de videojogos. Além disso, se for considerado que um jogo usável é aquele que oferece ao utilizador entretenimento e satisfação, então as heurísticas de videojogos devem abranger aspetos de *design* que garantam essa satisfação (Federoff, 2002, pp. 15-16).

O trabalho de Melissa Federoff (2002) visa compensar a falta de métodos para avaliar a usabilidade dos videojogos. Outros autores como Pinelle, Wong e Stach (Pinelle, Wong, & Stach, 2008) abordam esta questão da usabilidade em termos semelhantes. A abordagem deste conjunto de autores para com o *game usability* consiste na definição quanto ao grau de aprendizagem de um jogador, controlo e entendimento de um jogo (Pinelle et al., 2008).

Outros autores como Desurvire, Caplan & Toth (Desurvire, Caplan, & Toth, 2004) indicam que quando se evocam os videojogos e a jogabilidade, não se pode limitar à avaliação da usabilidade da interface do jogo. Estes autores defendem a necessidade de avaliar outras propriedades dos videojogos, como o *gameplay*, *gamestory*, *game mechanics* e usabilidade.

Blake Snow (Snow, 2007) apresenta uma lista de dez características, que segundo ele, devem ser seguidas pelos *designers* de videojogos. Essa lista, fornecida pela Next Generation (http://www.next-gen.biz), são outra perspetiva heurística para a avaliação de videojogos. Uma das características apresentadas, "apresentar sempre tutoriais no jogo, FAQs e menus de ajuda para os jogadores novatos", assemelha-se à décima heurística de Nielsen, "ajuda e documentação". Ambas perspetivas defendem o fornecimento de ajuda aos utilizadores, com o intuito de solucionar ou evitar problemas. A ajuda necessária para participar num jogo deve ser apresentada principalmente por meio de um tutorial (Federoff, 2002, p. 19).

Segundo Cockton e Woolrych (Cockton & Woolrych, 2002, apud. (Almeida, 2009)) os *so-called discount methods* (avaliação heurística incluída) não são seguros, e sugerem que podem e devem ser melhorados. Os autores caracterizam estes métodos como os que "cortam cantos" e que são usados na esperança de que o uso de qualquer método de usabilidade é melhor do que não ter usado método algum. Além disso, estes *discount methods* são definidos como aqueles que cortam nos custos de avaliação através da redução dos recursos que são essenciais para a avaliação: tempo, instalações, dinheiro e habilidade. Cockton e Woolrych afirmam ainda que estes métodos são suscetíveis a erros levando ao desacreditar dos profissionais de usabilidade, e que nem deveriam ser tomados em consideração em HCI (Cockton e Woolrych, 2002 apud. (Almeida, 2009)). Contudo, eles reconhecem que estes métodos são úteis como dispositivos de formação. Ou seja, eles são úteis para a compreensão de que existem problemas que exigem testes reais de utilizadores. Além disso, eles reconhecem as diferenças existentes entre os vários tipos de produtos e que, em alguns

contextos, os erros decorrentes da utilização de *discount methods* têm maior impacto sobre alguns produtos do que em outros.

Nesta secção apresentam-se alguns dos conceitos e métodos utilizados tradicionalmente na avaliação de usabilidade de produtos e sistemas. Assim como as técnicas tradicionais estão a ser propostas como métodos de avaliação de usabilidade de jogos. Na secção seguinte serão apresentados vários estudos relacionados com a avaliação de videojogos.

4.3. Avaliação de videojogos

Adquirir e analisar dados da experiência do jogador não só permite medir os efeitos que os videojogos têm nos jogadores, como também dá uma visão aos *designers* de como os jogadores vêm os seus jogos (Sasse, 2008). Segundo Kennerly (2003), ao analisar a experiência de um jogador pode-se ampliar uma perspetiva do *designer* do jogo, provar ou refutar hipóteses, e substituir opiniões por factos, tornando-se assim uma poderosa ferramenta para validar o *game design* (Kennerly, 2003).

Nesta secção são apresentadas algumas das investigações e projetos existentes feitos na área de avaliação de videojogos que usaram a técnica de *eye tracking*, outros através de *logfiles e* também aqueles que adotaram os *game heurístics*. Alguns destes estudos usaram mais que uma das técnicas referidas.

4.3.1. Avaliar com Eye tracking

Magy Seif El-Nasr e Su Yan em 2006 procuram estudar o processo de atenção visual dentro de um ambiente de um videojogo (El-Nasr & Yan, 2006). Para estes autores compreender os padrões da atenção visual dos jogadores dentro de um ambiente de jogos interativos 3D é uma importante área de investigação que pode melhorar do *level design* e gráficos de um jogo. Para além disso, Magy Seif e Su Yan pretenderam também explorar a diferença entre os padrões de atenção visual exibido em dois géneros de jogo distintos (*first person shooter* e jogos de ação e aventura).

Como eles referem, a atenção visual pode ajudar a melhorar os videojogos e seu *design*, aumentando a sua sedução para com o jogador e diminuindo a frustração deste durante o momento de jogo. Muitos jogadores novatos perdem-se nos ambientes de jogos 3D, ou não chegam a apanhar um item importante do jogo porque não percebem a sua importância (El-Nasr & Yan, 2006). Se os *designers* do jogo soubessem como os jogadores visualizam um jogo, seria mais fácil para eles entenderem onde devem colocar os itens num ambiente virtual ou até mesmo como misturar cores para chamar a atenção do jogador, eliminando portanto, os problemas mencionados acima.

El-Nasr e Yan também têm como objetivo compreender se a atenção visual dentro de um jogo 3D segue as teorias visuais *bottom-up* ou *top-down*, mencionadas na secção 2.4.

A possibilidade de estudar estas questões, apesar de não ser impossível, têm as suas complicações. Devido ao facto das interfaces 2D serem mais fáceis de estudar, um ambiente 3D apresenta novos desafios. Os autores apresentam, portanto, um novo método de *eye tracking* para analisar os dados dentro de ambientes 3D.

Após o estudo efetuado, com o envolvimento de seis participantes da Universidade do estado de Pensilvânia (submetidos a jogar 2 jogos durante 10 minutos: Legacy of Kain Blood Omen 2 e Halo 2), os autores concluem que nos videojogos 3D, estão presentes ambos padrões visuais, *bottom-up* e *top-down.* Destacam que o *eye tracking* pode fornecer informações que verifica esta conclusão (El-Nasr & Yan, 2006). No entanto, as limitações relacionadas ao número de participantes e o número de jogos usados para executar o estudo devem ser considerados em futuras investigações.

Outro estudo onde foi predominante o uso de *eye tracking* para a avaliação de videojogos foi o estudo de Sune Alstrup Johansen, Mie Nørgaard e Janus Rau. O estudo denominado por *Can Eye Tracking boost usability evaluation of computer games* (2008) explora a forma como o *eye tracking* pode responder a três desafios fundamentais enfrentados pelos produtores de videojogos IO Interactive (IOI) da Dinamarca, com vista em garantir que os jogos são divertidos, usáveis e desafiadores (Johansen et al., 2008). Tais desafios são:

- persuadir os designers de jogos acerca da relevância dos resultados de usabilidade;
- envolver os designers de jogos no trabalho de usabilidade;
- identificar métodos que fornecem novos dados sobre o comportamento e experiência do utilizador.

A partir do momento que os objetivos específicos de um videojogo são diferentes dos sistemas regulares, os métodos tradicionais de avaliação de usabilidade falham, com frequência, quando estes são usados para avaliar os jogos. Segundo os autores, é impensável usar o protocolo "pensar em voz alta" (Dix et al., 1998) enquanto um jogador está a jogar um jogo, visto que esta técnica pode distrair o jogador. Por essa razão, os autores discutem o uso do *eye tracking* como uma ferramenta para avaliar alguns dos problemas de usabilidade com que a empresa em causa (IOI) se depara. Os autores afirmam, no entanto, que não têm a intenção de provar que todos os problemas de usabilidade podem ser corrigidos por meio dos dados provenientes do *eye tracking*. O que eles esperam é que os resultados do *eye tracking* podem fornecer algumas informações sobre a importância dos resultados da usabilidade.

Johansen, Nørgaard, e Rau acreditam que o uso do *eye tracking* pode fornecer informações interessantes sobre o comportamento do utilizador e que por sua vez podem ser usadas com outros métodos de avaliação de usabilidade. No entanto, os autores acabam por afirmar que graças à natureza quantitativa dos dados recolhidos através do *eye tracking*, este pode produzir resultados que os *designers* de jogos na IOI consideram mais persuasivos, ao contrário dos resultados do exemplo dos tradicionais testes retrospetivos de "pensar em voz alta".

Os autores acreditam ainda que *o eye tracking* pode ser útil na avaliação de jogos, especialmente quando a atenção do jogador é crucial para o *gameplay*, bem como as distrações do jogador que são devastadoras para a sobrevivência de um jogador no jogo. Além disso, a originalidade dos dados do *eye tracking* acabam por ser um incentivo para os jogadores a participar na avaliação de usabilidade de um jogo.

4.3.2. Avaliar através de Logfiles

Em 2008 surge um estudo denominado por *Tracking Real-Time User Experience* (TRUE) com o intuito de estudar uma solução abrangente de instrumentação para sistemas complexos. Os autores, Kim, Gunn, Schuh, Phillips, Pagulayan, e Wixon, combinaram a recolha e análise da instrumentação comportamental com outros métodos de HCI para desenvolver um sistema de *tracking* da experiência do utilizador em tempo real (Kim et al., 2008). Os autores fizeram dois casos de estudo como exemplo, demonstrando assim como evoluíram de instrumentação e metodologia de análise para melhorar o *design* dos videojogos.

Segundo os autores, o TRUE é altamente adaptável e funciona muito bem em desenvolvimento de jogos, que exigem uma recuperação rápida dos resultados para cumprir prazos apertados. Os autores acreditam que esta solução não afetará apenas a área dos videojogos, mas também será igualmente útil para a compreensão de interação de utilizador com outros sistemas de computação (Kim et al., 2008).

O TRUE partilha traços em comum com outros tipos de instrumentação e técnicas de análise de *logfiles*. Com essas abordagens, os utilizadores interagem com sistemas que automaticamente gravam os eventos de interesse da aplicação em *logfiles* (*logging*), que o sistema envia para um servidor para posterior análise. O TRUE oferece a capacidade de navegar em grandes conjuntos de dados que representam várias horas de jogo de um jogador, para identificar potenciais problemas no ambiente do mesmo. O TRUE permite também detalhar situações específicas da missão, compreender o problema, e usar essas informações para fornecer recomendações para corrigir os problemas encontrados.

A partir dos estudos efetuados, os autores acabam por provar que o sistema TRUE é de facto uma tecnologia válida e útil. Assim como os autores explicam, estes precisavam de um método que lhes poderia permitir executar quatro ações (Kim et al., 2008):

- detetar problemas e entender as causas de raiz, tal como acontece em testes de usabilidade;
- suportar design de interação na medida em que os testes RITE (*Rapid Iterative Testing and Evaluation*) proporcionam;
- incorporar os comportamentos e atitudes na forma de inquéritos;
- compreender o uso naturalista dos produtos, como se encontra em métodos etnográficos.

Isto é necessário no contexto da indústria de videojogos, onde os orçamentos de desenvolvimento são grandes, o risco do negócio é alto, os cronogramas são apertados, e a apresentação de um valor de uma forma atempada é uma necessidade (Kim et al., 2008). Para além disso, o sistema TRUE tem sido um trunfo valioso para a indústria de videojogos. Nos últimos cinco anos, a tecnologia tem ajudado a melhorar mais de 20 jogos de vários géneros e plataformas de jogo.

Ainda no mesmo ano (2008), um outro estudo foi realizado com base numa técnica já mencionada no estudo anterior, o *logging*. Neste estudo, que fez parte da tese de Dennis Sasse, foi desenhado e implementado um ambiente para desenvolvimento rápido de jogos que usa um registo de dados psicofisiológicos. O autor compara métodos de recolha de dados

relacionados com o jogador e outros campos de aplicações conexas, a fim de estabelecer uma base teórica para a conceção de uma *framework* de registo (*logging*). Com base nisso, foram revistos alguns motores de jogos, plataformas de desenvolvimento e *frameworks*, que acabaram por levar Dennis Sasse à seleção da *framework* XNA e do motor Torque X como tecnologias adequadas para a implementação desta mesma *framework* de registo (Sasse, 2008).

Neste estudo o autor ainda integra a técnica de *eye tracking* para recolher dados do olhar do jogador, e acaba por desenvolver também um jogo de exemplo a fim de demonstrar a aplicação geral do ambiente e as funcionalidades do *eye tracking* e do *logging*.

Segundo Dennis Sasse, uma vez que os dados dependentes do contexto de *logging* estão profundamente ligados à lógica do jogo, a etapa de personalizar a funcionalidade de *logging* para um jogo específico nunca pode ser omitida. No entanto, através do desenvolvimento de um conjunto de componentes de *logging*, esta etapa pode ser ainda mais simplificada.

O autor acrescenta ainda que a realização de experiências psicofisiológicas que mensuram a forma como os jogadores experienciam os videojogos, não só permite investigar os efeitos que os videojogos têm sobre os jogadores, como também oferece aos desenvolvedores de videojogos uma ferramenta para validar o *game design* (Sasse, 2008).

4.3.3. Avaliar segundo Game Heuristics

Existem vários estudos de vários autores que optaram por seguir o caminho das heurísticas para a avaliação de videojogos. Autores como Malone (1982), Melissa Federoff (Federoff, 2002), Heather Desurvire, Martin Caplan e Jozsef Toth (Desurvire et al., 2004), Blake Snow (Snow, 2007), David Pinelle, Nelson Wong, Tadeusz Stach (Pinelle et al., 2008) e Samuel Almeida (Almeida, 2009).

Melissa Federoff propõe uma lista de heurísticas e sugestões que poderiam ser usadas por *designers* de jogos para estes desenvolverem e criarem melhor os videojogos. Para fazer isto, Federoff examinou heurísticas implícitas e explícitas e processos de avaliação de usabilidade utilizados por um líder em desenvolvimento de videojogos (Federoff, 2002). Além disso, Federoff passou uma semana de observação e posteriormente entrevistou cinco pessoas de uma empresa de criação de jogos, cada um com um importante papel e contribuição para o processo de desenvolvimento de jogos.

Com os dados recebidos dos participantes, Federoff foi capaz de identificar algumas heurísticas assim como cada uma foi identificada. Federoff codifica ainda as heurísticas para uma das áreas específicas de usabilidade dos videojogos (interface, mecânica e *gameplay*). Após a codificação e contagem do número de vezes que cada heurística foi mencionada, a autora comparou a lista com a heurística que tinha identificado durante a revisão da literatura.

A autora acaba por apresentar ainda uma lista de quatro sugestões para a execução de um procedimento mais formal de usabilidade que estão relacionados com (Federoff, 2002):

- **prototipagem** - Federoff sugere que a prototipagem deve ser uma parte de cada jogo. Implementando uma fase de prototipagem poderia evitar fracassos financeiros ou, no

mínimo, permitir a uma empresa entender se um jogo vai ou não funcionar antes que a produção já esteja numa fase avançada e que grandes investimentos estejam feitos. Além disso, também permite a criação de protótipos para que os problemas sejam identificados antes do processo de desenvolvimento;

- **teste** - a segunda sugestão é com os testes. Embora seja possível apenas para testar completamente com um produto acabado, um protótipo permite testar numa versão simplificada do videojogo;
- **fase *post-mortem*** - esta sugestão está relacionada com a fase *post-mortem*, uma fase onde um dos participantes do estudo mencionou não achar que fosse o suficiente. A possibilidade de discutir o que correu mal ou o que correu bem, durante o processo de desenvolvimento do jogo depois de estar finalizado, pode ajudar a evitar a repetição de erros. Durante esta fase o facto de que o jogo já estar no mercado, permitindo a indivíduos externos jogar o videojogo e avaliá-lo, pode fornecer informações importantes. Se o teste de usabilidade não for possível, a exploração dos métodos de avaliação de peritos de usabilidade pode ser útil para determinar os problemas existentes num jogo;
- **avaliador de usabilidade** - finalmente, uma quarta sugestão de Federoff passa pela existência de um avaliador de usabilidade com conhecimentos de jogos na equipa de produção de videojogos da empresa. Mesmo que seja apenas uma pessoa, este pode ajudar no desenvolvimento e avaliação de protótipos, bem como ajudar com testes e avaliações de usabilidade.

O estudo *Using Heuristics to Evaluate the Playability of Games* de Desurvire, Caplan e Toth (Desurvire et al., 2004), centra-se, tal como o estudo de Melissa Federoff, sobre o uso de heurísticas para avaliar jogos. Os autores apresentam a avaliação heurística para a jogabilidade (HEP - *Heuristic Evaluation for Playability*), um conjunto de heurísticas para avaliar jogos de computador, de vídeo e de tabuleiro e testaram-nos para compreender a sua validade e eficácia real em relação às metodologias de testes de utilizador padrão.

Os autores propuseram então uma lista de heurísticas, considerando a revisão da literatura que fizeram (mencionando nomes como Nielsen, Malone e Federoff), onde as dividiram em quatro áreas específicas (*gameplay*, *gamestory*, mecânicas e usabilidade). Estas heurísticas HEP também foram examinadas por vários especialistas na área de jogabilidade e *game design*. Posteriormente, a lista foi colocada à prova através de um jogo que estava a ser desenvolvido.

Os autores acreditam que os testes de utilizador são uma parte essencial de qualquer avaliação de jogabilidade, principalmente porque um *designer* de jogos nunca pode prever o comportamento de um jogador. Também acabam por admitir que a avaliação heurística para a jogabilidade (HEP) é muito útil para as fases iniciais de desenvolvimento de um jogo, em vez do desenvolvimento de protótipos possivelmente caros (Desurvire et al., 2004).

Blake Snow (Snow, 2007), destacado na secção 4.2, argumenta ainda que um bom programa ou produto deve permitir aos utilizadores de realizar as suas tarefas destinadas da forma mais fácil e direta possível. Quanto menos tempo que uma pessoa leva para concluir uma tarefa desejada (mesmo que apenas por alguns segundos), mais satisfatório se torna. Quando isso

acontece, as pessoas estão mais propensas a usar um produto com maior frequência e voltar para mais (Snow, 2007).

David Pinelle, Nelson Wong e Tadeusz Stach no estudo *Heuristic Evaluation for Games: Usability Principles for Video Game Design* (Pinelle et al., 2008), apresentam um novo conjunto de heurísticas que podem ser usados para os mesmos fins que todas as outras já referenciadas acima. Os autores desenvolveram a heurística por meio de uma análise de opiniões de jogos para PC a partir de um *site* popular de jogos, e a revisão abrangeu 108 jogos diferentes e incluiu 18 de cada um dos 6 principais géneros de videojogos. Os autores analisaram as opiniões e identificaram 12 classes comuns de problemas de usabilidade observados em videojogos. Desenvolveram então dez heurísticas de usabilidade com base nos problemas das categorias, e descreveram como se podem evitar os problemas mais comuns de usabilidade de videojogos.

Segundo os autores, os *designers* que têm grande experiência com jogos podem identificar muitos problemas de usabilidade sem a necessidade de um conjunto de heurísticas. No entanto, ainda há muitos jogos que são lançados com sérios problemas de usabilidade, por isso Pinelle et al. (Pinelle et al., 2008) acreditam que as abordagens de avaliação mais organizadas, tais como avaliações de usabilidade que usam heurísticas, ainda têm valor. Acrescentam ainda que esta é uma das razões para o sucesso das heurísticas de Nielsen, os princípios de *design* descrito nas heurísticas são amplamente aceites na comunidade HCI, e são bem conhecidas pela maioria dos designers.

Neste estudo, os autores utilizaram uma revisão crítica de *software* para identificar problemas de usabilidade e desenvolver um conjunto de princípios de design de videojogos. As heurísticas apresentadas neste estudo, segundo os autores, representam uma nova forma de adaptação de avaliação de usabilidade para jogos, e permitem aos *designers* avaliar tanto maquetes como protótipos funcionais. Ao contrário de outras heurísticas de videojogos, a lista apresentada pelos autores focam-se, segundo eles, especificamente sobre a usabilidade do jogo, e é baseada numa análise estruturada dos problemas de usabilidade de um grande número de jogos.

Mais recentemente, Samuel Almeida (Almeida, 2009) propõe uma metodologia que é uma parte de um estudo empírico que contou com participantes com distintos níveis de experiência com os videojogos e fez uso de dois instrumentos: um questionário e em segundo lugar, uma tecnologia que, segundo o autor, não gera consenso – o *eye tracking*. Os dados obtidos através dos questionários bem como os resultados do *eye tracking* foram analisados e serviram de base para o objetivo maior do estudo: o desenvolvimento de *guidelines* (linhas orientadoras) que pudessem assistir na melhoria da conceção e do desenvolvimento de videojogos.

Todos estes estudos mencionados são abordagens que podem ser usadas por investigadores para compreender as questões de *design* vistos noutros tipos de software da especialidade. Para além disso, o uso da avaliação heurística numa análise de um videojogo em conjunto com técnicas como *logging* e *eye tracking*, acaba por tornar uma avaliação mais completa e eficaz na deteção de problemas no jogo.

5. Métodos e técnicas de recolha e modificação de dados em videojogos

Na secção 4.1 são mencionadas algumas técnicas de recolha de dados propostas por Dix et al. (1998), tais como técnicas de consulta e técnicas de observação. Nesta secção serão ainda apresentadas alguns métodos e técnicas semelhantes, mas numa perspetiva focada em videojogos.

5.1. Métricas de jogo para análise

As métricas de jogo estão relacionadas com a interação do utilizador com o videojogo. O primeiro tipo de métricas de jogo são na essência eventos iniciados pelo utilizador como, por exemplo, o movimento dentro do ambiente virtual, a utilização da interface do jogo, o uso de habilidades específicas no jogo, interação com entidades ou objetos no mapa do jogo ou outros jogadores, entre outros. As métricas são objetivas, podem ser recolhidas em grande quantidade e mapear pontos específicos no jogo (Tychsen, 2008).

As métricas de interação com o utilizador podem assumir diferentes formas. Alguns podem ser registados numa base contínua (e.g. movimento no mundo virtual) ou ser gravados utilizando frequências específicas (e.g. determinar a localização do jogador virtual a cada 3 segundos). Também é possível registar métricas sempre que um evento é despoletado (e.g. registar cada vez que o jogador salta ou dispara uma bala). Com vista a complementar estas informações, também é necessário capturar outros dados, como por exemplo, o momento da ocorrência (*timestamp*), coordenadas do jogador virtual, autor do evento, entre outros.

Tychsen (2008) afirma que a escolha das métricas a ser registada, o seu registo como conjunto de eventos ou conjunto de dados agregados depende do jogo em questão e das exigências da análise atual. O uso de métricas na análise de videojogos permite responder a perguntas relacionadas com o quê, onde e quando; contudo, já não têm resposta para o porquê e o como (Nacke et al., 2009).

5.2. Instrumentos de recolha de dados em videojogos

Sasse (Sasse, 2008) menciona duas formas metodológicas para obter dados da experiência do jogador no jogo: primeiro, pedir aos jogadores para preencher um questionário após o jogo, e em segundo, acompanhar jogadores enquanto jogam o videojogo, e gravar os dados pertinentes relacionados com estes. Este processo automatizado de monitorização e recolha de dados é normalmente referido como *logging* (explicado com mais pormenor na secção 5.3).

5.2.1. Questionários

As experiências do utilizador nos videojogos são baseadas nas necessidades do jogador, expectativas, referências e satisfação. Portanto, os questionários podem ser considerados um instrumento válido para os jogadores relatarem as suas experiências de jogo. Acredita-se que os questionários, em geral, são mais objetivos, nomeadamente quando comparados com as entrevistas. Além disso, Harvey e Council (1998) afirmam que é relativamente rápido recolher informação através de um questionário. No entanto, em algumas situações, os questionários podem levar um longo tempo não só para a sua conceptualização, mas também para a aplicar e analisar. Potencialmente, os questionários podem ser usados para colectar dados de um grande grupo, mas geralmente sofrem de baixa percentagem de retorno (Gamboa et al., 2004; Harvey e Council, 1998 apud. (Sasse, 2008)).

Um dos problemas dos questionários é que estes, assim como outros métodos de avaliação, ocorrem após o evento, e com isso os participantes podem se esquecer de questões importantes (Harvey e Council, 1998 apud. (Sasse, 2008)). Nos casos em que o questionário contém perguntas abertas as respostas podem demorar muito tempo para processar e analisar qualitativamente.

Os questionários também são facilmente dispersos, porque os jogadores não precisam de reportar os seus próprios comportamentos em inquéritos ou em feedback de clientes (Kennerly, 2003).

5.2.2. Aquisição não intrusiva de Dados

Existe uma infinidade de mecanismos de controlo para validar os dados adquiridos através de questionários (Pressione, 2004 apud. (Sasse, 2008)), a fim de combater o "ruído social" como Kennerly refere (Kennerly, 2003). No entanto, a natureza intrusiva dos questionários pode dificultar a recolha de dados não distorcidos que representam as experiências do jogador (Sasse, 2008).

Desde que a imersão no jogo se tornou um pré-requisito para os jogadores desfrutarem plenamente o *game experience*, uma possível distração de coleta de dados de forma intrusiva deve ser ponderada para não afetar o momento de jogo. Os questionários não podem ser preenchidos pelos jogadores enquanto estão a jogar. Isto implica que os jogadores tenham de se abstrair do próprio jogo, pondo em causa a fiabilidade das suas respostas aos questionários.

Alguns conceitos como o protocolo de "jogar em voz alta" (Ericsson e Simon, 1993 apud. (Sasse, 2008)) exigem aos jogadores que se pronunciem sobre as suas ações no jogo, enquanto estão a jogar, e é gravado ou transcrito por um observador. Além disso, os jogadores são

gravados em vídeo e pedem-lhes para comentar os vídeos após a sessão de jogo. A tarefa adicional de comentar o jogo pode colocar uma carga cognitiva adicional sobre os jogadores, distraindo-os do jogo e impedindo estes de estarem imersos no jogo. Ao comentar sobre um vídeo após o jogo, os jogadores podem já ter esquecido as suas motivações para fazer uma determinada ação (Sasse, 2008). Também existe o problema do subconsciente, distorções pessoais dos comentários mencionados por Kennerly (Kennerly, 2003) podem ocorrer nessa situação. Por exemplo, Sennersten (2004, p. 20) descobriu que os jogadores sempre alegaram que olhavam para o rosto de um personagem não-jogador (NPC - *non-player character*), quando os dados recolhidos através de *eye tracking* mostraram que eles afinal estavam a olhar para as mãos do NPC.

Outro conceito muito semelhante foi executado pela bolt|peters, que optou por deixar de parte a investigação baseada em "Focus Groups" (metodologia mencionada por Shneiderman (1998, p. 145)), para criar a sua própria abordagem metodológica que assentou em observar os jogadores a distância, de forma a que estes pudessem agir de forma natural no local que bem entendessem e sem receber quaisquer instruções do que deviam fazer no jogo (bolt|peters, 2008). A equipa de investigação gravou em vídeo e áudio durante 6 horas em 2 noites a cada jogador, acabando por se tornar um trabalho dispendioso em termos de tempo para a sua análise. Para além do tempo necessário para tirar notas enquanto o jogador jogava. Mas em compensação os resultados sempre foram mais viáveis, tendo em conta que os jogadores agiram de forma natural enquanto jogavam.

A fim de evitar os problemas de discrepância e distorção percetiva do subconsciente, nada melhor que o uso de ambos os métodos (através de triangulação) para um conjunto completo de dados. Os questionários podem ser usados para coletar informações gerais dos jogadores (idade, sexo, preferências de jogador e de dados adicionais mais subjetivos) e informarem os jogadores após uma sessão de jogo. O processo automático de monitorização ou *logging* pode ser usado para uma recolha de dados não intrusiva, em tempo real sobre o comportamento e desempenho dos jogadores durante o jogo. Segundo Ravaja (Ravaja, 2004 apud. (Sasse, 2008)), o ideal é recolher os dados dos jogadores sem que estes se apercebam a fim de evitar um fenómeno, conhecido como o "efeito observador", obtendo assim um vislumbre sobre a personalidade real e desempenho do jogador (Kennerly, 2003).

5.3. Sistema de *logfiles* nos videojogos

Logfile é um ficheiro que grava ações ocorridas num determinado sistema. O computador consegue registar todos os comandos digitados pelos utilizadores, e em alguns casos, todas as teclas pressionadas. Nos casos em que os utilizadores interagem apenas *online*, pode-se recolher um registo detalhado de todas as suas interações. Um bom exemplo disto são os servidores *web* que listam todos os pedidos feitos ao servidor. Com alguns instrumentos de análise de *logfiles*, os administradores podem ter uma ideia do local por onde os visitantes ligaram ao *site*, como navegaram no mesmo, páginas mais visitadas, entre outros. Para além do exemplo do *site* existem outros tipos de sistemas onde os *logfiles* acabam por ter um grande impacto a nível de armazenamento de informação e utilidade na sua análise. Nesta secção mostra-se a utilidade dos *logfiles* em geral e aplicados em videojogos, onde, por exemplo, podem ser registados dados como as coordenadas x, y e z do jogador em tempo real,

ações desempenhadas pelo mesmo ao longo do jogo, e também coordenadas do olhar do jogador em relação ao ecrã através da técnica de *eye tracking* (esta técnica automatizada, como já foi mencionado acima, pode ser referida como *logging*).

5.3.1. Tipologias de análise de logfiles

É possível encontrar uma variedade de abordagens, dependendo da natureza das perguntas de investigação. Como Jenny Preece refere, o primeiro passo importante antes de usar o registo de dados e métricas é rever os objetivos e as perguntas de investigação (Preece, 2000 apud. (Bruckman, 2006)). Dentro dos vários tipos de análises de *logfiles* pode-se encontrar a análise quantitativa e análise qualitativa, que por sua vez também podem ser manuais ou automatizadas (Bruckman, 2006).

Segundo Amy Bruckman, a análise qualitativa de um *logfile* é geralmente realizada manualmente (por uma pessoa), no entanto, existem ferramentas informatizadas que podem ajudar as pessoas na sua análise. Por exemplo, no caso de haverem grandes quantidades de dados, o *software* pode ajudar na sua organização, como também na procura e identificação de trechos que atendam a um conjunto de critérios desejados (Bruckman, 2006). O leitor humano numa análise qualitativa pode usar uma série de modelos teóricos para organizar essa análise, tais como teoria fundamentada (Glaser e Strauss, 1967 apud. (Bruckman, 2006)), cognição distribuída (Hutchins, 1995 apud. (Bruckman, 2006)), teoria da atividade (Engeström et al., 1999 apud. (Bruckman, 2006)), entre outros.

Os *logfiles* não devem ser a única fonte de dados numa investigação ou estudo (Bruckman, 2006). Convém recorrer a outras metodologias, tais como, entrevistas e questionários com participantes. Estas abordagens podem ajudar significativamente na compreensão da informação existente num *logfile* pelo investigador, e vice-versa, um trecho de um *logfile* pode ser um ponto de partida para obter, das entrevistas, dados mais sólidos.

Hulshof (2004) também menciona uma série de métodos que podem ser usados para obter *logfiles*. Segundo o autor os três métodos que são mais usados são:

- **registo dos movimentos dos olhos** - este método é discutido em detalhes no livro de Duchowski na metodologia de *eye tracking*, também já mencionado na secção 2.3. O método implica que as fixações oculares sejam registadas quando uma pessoa está a realizar uma tarefa. Depois, os pontos de fixação podem ser combinados com as coisas que aconteceram num ecrã. O registo dos movimentos oculares resultam em *logfiles* muito detalhados e quando uma tarefa tem uma grande componente visual, os movimentos oculares podem fornecer aos investigadores mais informação do que com outros métodos de obtenção de dados através de *logfiles*.
- **análise de protocolos** - também conhecido por Van Someren, Barnard, Sandberg e Dix et al. (1998) como o método de "pensar em voz alta". Este método, já mencionado na secção 4.1, é um método popular para obter um relatório detalhado em tempo real do processamento cognitivo dos utilizadores.
- **operações de registo de computador** - um método que consiste em manter o controlo de toda a interação, que é definida como o comportamento do desempenhado pelo utilizador e que leva a uma mudança, visível ou invisível, para o programa de

computador no qual o padrão de interação é estudado. Na prática, isso significa registar todos os cliques nos botões do rato, impressoras e operações de teclado que o utilizador compromete-se durante uma tarefa. Os *logfiles* resultantes geralmente são extensos, mas as pessoas não têm o raciocínio por trás das ações que já foram realizadas, o que torna a interpretação uma questão complexa. Uma vantagem importante deste método é que este pode ser aplicado a um conjunto de pessoas em simultâneo, o que pode ser muito prático.

Apesar de métodos diferentes, estes podem ser combinados e geralmente apenas um tipo de medição é escolhido (Hulshof, 2004).

Bruckman (2006) ainda divide um *logfile* em duas tipologias distintas: análise quantitativa manual (um ser humano a traduzir as entradas no *logfile* em métricas específicas) e análise quantitativa automática (um software computacional que realiza a tradução). Alguns tipos de dados de *logfiles* se concedem a análises automatizadas mais facilmente que outros. Contudo existe uma linguagem de script interessante, o Perl (http://www.perl.org/) que contém poderosas capacidades de correspondência de padrões, e é frequentemente útil para automatizar processos com os dados de *logfiles*.

5.3.2. Supremacias dos logfiles

Segundo Bruckman (2006), não se deve desperdiçar um bom *logfile* (Bruckman, 2006). O uso de dados provenientes de *logfiles* como ferramenta de estudo pode beneficiar de:

- complementação na utilização de métodos qualitativos e quantitativos;
- métodos de análise manuais e automatizados;
- encontrar padrões ou regularidades nos dados de um determinado ficheiro (*data detective*);
- reconhecimento da facilidade com que um conjunto de dados pode ser lido por pessoas, e de que forma isso molda a sua possível análise.
- uso de ferramentas informatizadas para auxiliar o leitor humano;
- complementar a utilização de dados de outras fontes (tais como entrevistas, notas de campo e pesquisas);
- atenção para as questões éticas relativas ao registo e análise de dados de *logfiles*, mantendo o respeito pela privacidade das pessoas e pelos seus direitos como seres humanos (em situações em que a obra constitui “pesquisa com seres humanos”).

A análise do *logfile* é um processo poderoso para ganhar introspeção nos processos cognitivos que fundamentam o comportamento, especialmente a interação com um programa de computador. Obter um registo detalhado da interação pode ser alcançado de forma relativamente discreta através de uma série de diferentes métodos, e os *logfiles* resultantes podem ser estruturados de tal forma que podem ser aplicados diferentes métodos de análise. No entanto, cada método tem as suas próprias vantagens e desvantagens. Os *logfiles* podem ainda fornecer uma riqueza de informações independentemente da área de aplicação. Contudo a análise dessa informação deve ser precisa e adequada para a situação em questão.

5.3.3. Sistema de logging em videojogos

Logging é o processo de monitorização e gravação de dados (registo - normalmente em *logfiles*), selecionado pelos investigadores, antes da sessão de jogo. Estes dados, segundo Sasse (2008), podem ser classificados pela sua dependência no contexto do jogo pela qual foram recolhidos:

- **contexto de dados independentes** - um contexto cujo dados podem ser adquiridos sem ter em conta o contexto ou o conteúdo do jogo. Esses dados consistem em:
    - input do utilizador (e.g. teclas que são pressionadas)
    - dados provenientes do *eye tracking* (coordenadas do local (ecrã) que o jogador está a olhar - *PoR*)
    - notável emoção do jogador através da combinação de eletromiografia (EMG) e atividade eletrodérmica (EDA)
    - capturas de ecrã (captura de vídeo), da sessão de jogo

  Os dados com nenhuma dependência do contexto do jogo podem ser registados a partir do exterior por um *software* especializado para o efeito, sem necessidade de modificação da integridade do jogo.

- **contexto de dados dependentes** - um contexto onde os dados requerem conhecimentos sobre elementos chave do jogo. Tais dados podem depender de apenas um único valor chave (e.g. pontuação do jogador, tempo para terminar o nível), ou numa combinação de variáveis (ex.: objetos que o jogador está a olhar, o número de inimigos que o jogador atacou, entre outros). O registo de dados neste tipo de contexto, obriga a que o jogo seja modificado para integrar a funcionalidade de registo de dados.

Embora haja uma série de soluções de *software* disponíveis que permitem a aquisição e visualização de dados psicofisiológicos provenientes de fontes diversas de hardware, esses sistemas só podem registar dados comportamentais independentemente do contexto em que se encontram, uma vez que geralmente não são concebidos para integração noutras aplicações de software ou jogos em particular. No entanto, alguns fabricantes de hardware oferecem kits de desenvolvimento de software (SDK's - *Software Development Kits*) para os seus produtos assim como o SDK desenvolvido para o Half-Life 2 (ValveCorporation, 2004) que surgiu após uma abordagem desenvolvida pela Agência Sueca de Defesa de Investigação, cuja finalidade se assentou na integração de um sistema de Eye Tracking Tobii 1750 num mecanismo de alta fidelidade (Sasse, 2008)).

Nem sempre haverá a possibilidade de seguir o caminho do contexto de dados dependentes, caso o jogo em questão seja um produto comercial, pois o código fonte deste tipo de produtos raramente encontra-se disponível ao público.

Sasse (2008) defronta duas possibilidades de integração da funcionalidade de *logging* num jogo:

- **Modificação de um jogo comercial** - um jogo comercial que permite que os jogadores possam alterar o seu conteúdo ou para criar conteúdo personalizado e modifica-lo

(muitas vezes referido como *mod*). No entanto, isso só é possível, se o acesso ao código fonte for permitido.

- **Desenvolvimento de um jogo personalizado** - um jogo feito sob encomenda pode ser desenvolvido de raiz para o propósito único de recolha de dados.

Na modificação de jogos comerciais, tem-se a oportunidade de se beneficiar de uma base de jogo já implementada. Normalmente existem editores de níveis associados a estes jogos que oferecem ferramentas de edição da inteligência artificial, *scripting*, geometria do nível, iluminação e por aí adiante. Por exemplo, Source SDK da Valve (ValveCommunity, 2010). Desenvolver jogos personalizados em torno da funcionalidade de *logging* permite uma maior flexibilidade e fácil integração de *hardware* de aquisição de dados através de terceiros. Uma vez que todo o conteúdo tem que ser criado a partir do zero, esta abordagem provavelmente irá exigir mais tempo de desenvolvimento e os resultados serão de qualidade inferior.

A combinação de um motor de jogo flexível, altamente reutilizável na funcionalidade de *logging* e editores de nível de jogo intuitivos permitem um rápido desenvolvimento de jogos, facilitando o processo de *logging* das experiências dos utilizadores no jogo.

Na secção seguinte serão mencionados em pormenor esta possibilidade de modificar um videojogo, assim como as suas prevalências e impacto na indústria dos videojogos.

5.4. Videogame modding

As modificações, também conhecidas como *mods*, são uma forma intrigante e de longa duração de produção por parte do jogador e têm sido uma parte essencial dos jogos de computador há mais de uma década (Olli Sotamaa, 2008). Modificação ou alteração é um termo geral aplicado a jogos de computador, nomeadamente, nas categorias de *first person shooter*, fantasia*/role-playing* e jogos de estratégia. Os *mods* são desenvolvidos pelo público em geral ou por alguém com conhecimentos de programação, normalmente denominados por *modders* (pessoas que modificam). Segundo Sotamaa (2008), *mods* são artefactos digitais realizados por jogadores desejosos em brincar com os seus jogos favoritos. No entanto, ao criar *mods*, os utilizadores têm a oportunidade de inovar no *game design* e no *gameplay*, mas para os desenvolver estes têm que adquirir a versão original do jogo base ou uma licença do mesmo, pois este tipo de jogos não pertencem a um sistema independente (Scacchi, 2010). Os utilizadores podem incluir novos itens como armas, personagens, inimigos, modelos, texturas, níveis, linhas de história, música e modos de jogo. Nesta secção serão explicados alguns aspetos relacionados com o ato de modificar videojogos, assim como as motivações, prevalências, impacto na indústria dos videojogos e nas comunidades.

5.4.1. Motivações

Hector Postigo em 2007 identifica três motivações centrais do *modding* de videojogos. Segundo o autor muitos dos *modders* consideraram que o ato de *modding* ou *mapping* é uma saída criativa que lhes permite contribuir com algo de belo. Para outros o *modding* permite às pessoas se identificarem com os jogos e aumentar a satisfação com os mesmos. Finalmente, há casos em que vêm o *modding* como uma oportunidade de adquirir emprego na indústria dos videojogos (Postigo, 2007, pp. 309-310). Scacchi (2010) confirma esta última motivação de

Postigo, afirmando que o *modding* é uma forma viável de conseguir um emprego na indústria de desenvolvimento de jogos. O autor reforça esta ideia com o exemplo da Epic Games que por muitos anos tem incentivado os desenvolvedores de jogos a fazer *download* das ferramentas de edição do Unreal (UDK - *Unreal Development Kit*) para modificar o próprio jogo. Com isto os desenvolvedores têm a oportunidade de serem contratados para um estúdio de desenvolvimento de jogos como desenvolvedores de jogos experientes.

Olli Sotamaa (2008), no seu artigo, afirma que a motivação aumenta pela necessidade de extrair o máximo possível dos jogos. A criação de novas áreas e objetos de jogo tornam o ato de jogar significativo depois dos jogadores conhecerem detalhadamente os ambientes originais. Com ambientes alterados, os jogos não só são alargados, mas também personalizados. O autor indica também que o *modding* pode ser visto como uma manifestação cultural contemporânea do jogo, o "legado *hacker*". Os programadores de *mods*, muitas vezes, realçam o desafio que os jogos colocam como sistemas complexos baseados em código. Além disso, os conhecimentos avançados em *modding* não são adquiridos através da educação formal, mas sim no ato de jogar com o código. Sotamaa (2008) compara os *hackers* e os *mooders* orientados para a investigação, onde o primeiro entusiasticamente examina os detalhes do código e o segundo pretende esclarecer os detalhes do assunto em questão. Isso pode significar qualquer coisa como um pedaço de informação histórica ou uma visão clara do objeto a ser modelado. Além disso, o autor acrescenta que os *modders* usam os jogos como um meio de expressão e que a motivação pode variar desde a pura estética até à motivação política (Olli Sotamaa, 2008, p. 8).

Sotamma (2008) menciona que, para algumas pessoas, visitar fóruns e participar em projetos de *mods* é uma forma essencial de encontrar outras pessoas com os mesmos interesses. Além disso, quanto maior for o projeto de *modding* mais importantes se tornam as competências sociais.

Um motivo importante, para além dos referidos anteriormente, é a perspetiva com que se pode ficar, após um conjunto de modificações feitas em videojogos, sobre a estrutura requerida para trabalhar na indústria de videojogos: *"The practical advice I would give, if you have absolutely no experience in the games industry, is to work on mods. That's a great way to get an idea if this is really something you want to do for a living, because many mod groups can be just as organized, with similar timetables, schedules, and politics as an actual production team. You get a real feeling for the structure that's required to work in this business."* Justin Hayward (Todd, 2007). *Modding* é uma forma de aprendizagem - aprender como modificar, aprender a ser um desenvolvedor de jogos, aprender a tornar-se um desenvolvedor de conteúdo/*software*, aprender a ciência do videojogo dentro ou fora de um ambiente académico, entre outros (El-Nasr & Smith 2006; Hayes & Games, 2008; Scacchi 2004 apud. (Scacchi, 2010)).

5.4.2. O impacto do modding

Muitas empresas têm reconhecido abertamente o valor do conteúdo que os fãs criam na produção de *mods* (Postigo, 2007). Os desenvolvedores de sistemas de jogos estão cada vez mais a oferecer ferramentas de *software* para modificar os jogos que criam ou distribuem, como forma de aumentar as vendas de jogos e a participação do mercado. SDKs e outras

ferramentas de *modding* fornecidos aos utilizadores pelos estúdios de desenvolvimento de jogos representam uma estratégia contemporânea de negócios para incentivar os utilizadores a ajudar a conduzir a inovação de produtos fora do estúdio (Hippel e Katz, 2002; Jepperson, 2005 apud. (Scacchi, 2010)).

Epic Games, Valve Software, e Id Software são exemplos de produtoras de videojogos que reconheceram o valor das suas comunidades de jogos. Segundo Hector Postigo (2007), Wagner James Au (2002), entrevistou especialistas do setor sobre o papel das comunidades de fãs do *game design*. Nesse artigo, Scott Miller, da 3D Realms afirma que "os desenvolvedores assistiram estupefactos como os *mods* realmente ajudaram a prolongar a vida de um jogo, oferecendo conteúdo adicional grátis para os jogadores explorarem." No mesmo artigo, Cliff Bleszinski da Epic Games (criador da série popular Unreal) estimou que 5 a 10 por cento de jogadores de Unreal tivessem usado as ferramentas de edição e que pelo menos metade dos 2 milhões de jogadores fizeram *download* e jogaram mods para o jogo.

Hector Postigo (2007) afirma que nenhuma empresa tem feito tanto quanto a Id Software e a Valve Software para incorporar add-ons[3] para fãs no processo de desenvolvimento. A emrpesa Id Software disponibiliza o código fonte para todos os seus jogos e incorporou o mais bem sucedido nível de jogo desenvolvido por fãs numa das suas distribuições de Doom e contratou o programador e fã Robert Duffy para criar o kit de desenvolvimento QERadiant, para o Quake3. A referida empresa Id Software usou esse mesmo editor para criar outros títulos de grande sucesso, como Return to Castle Wolfenstein.

No caso da Valve Software, esta usou o Worldcraft, um kit de desenvolvimento produzido por fãs para o Quake original, para desenvolver o título popular Half-Life. Em 2002 a Valve lançou o Steam Online - um programa de computador de gestão de direitos digitais para tentar combater a pirataria e fornecer serviços como a *steam store* e actualização automática de jogos. Hoje em dia o Steam conta com mais de 30 milhões de utilizadores ativos e serve como um ponto de encontro de amigos e comunidades de videojogos (ValveCorporation, 2010). O Steam possibilita ainda aos utilizadores de partilhar os seus *mods* criados facilitando a sua divulgação (ValveCorporation, 2008).

5.4.3. A comunidade

Segundo Sotamma (2008), as formas de colaboração entre os membros da comunidade de *mods* são relativamente diversificadas. Os concelhos sobre detalhes técnicos, normalmente, são solicitados em fóruns e também podem ser usados como fonte de feedback, onde por exemplo, são anunciadas as versões beta a fim de receber comentários e relatórios de bugs. Também existem *sites* e blogs específicos para apresentarem novidades das ferramentas de desenvolvimento e afins. Além disso, os *modders* mais experientes costumam desenvolver tutoriais e publicá-los nos seus *sites*.

No caso particular do kit de desenvolvimento Source (Source SDK) da Valve Software, existe a comunidade *The Valve Developer Community* (http://developer.valvesoftware.com). A partir

[3] Significa adicionar valor ao jogo que pode variar desde simples objetos, plantas e construções, veículos e armas, esquadrões modificados e a ilhas completas (Sotamaa, 2008).

de uma wiki, esta comunidade, apresenta detalhadamente como usar o kit de desenvolvimento e explica também com pormenor as várias camadas de um projeto de *modding* como *level design*, programação, modelação, sons, partículas, materiais e coreografia. Nesta wiki também são disponibilizados tutoriais e ferramentas desenvolvidas pelos desenvolvedores em Source SDK. A comunidade está diretamente relacionada com a Steam que dá a possibilidade ao *modder* de distribuir o seu *mod* nesta plataforma. Também existe um fórum de discussão onde a comunidade expõe as suas dúvidas e opiniões. A partir do Source SDK os *modders* podem alterar jogos pertencentes à produtora Valve como Half-Life2, CounterStrike:Source, Team Forteress 2 e o recente Left 4 Dead 2.

Outro kits de desenvolvimento muito conhecido, que já foi mencionado antes, é o *Unreal Development Kit* da Epic Games que também tem a sua comunidade de desenvolvimento em UDK (http://udk.com/). Este kit usa o unreal *engine* que serviu de base a títulos como Unreal Tournament Series, Medal of Honor Airbone, Army of Two e Gears of War (EpicGames, 2010).

As comunidades que existem por trás destes programas de desenvolvimento são importantes para sua manutenção. O nível de detalhe da documentação disponível e a entre ajuda existente nos fóruns de discussão são fatores importantes que facilitam o processo de *modding* sobretudo aos *modders* iniciantes.

5.4.4. Tipos de modding

Scacchi (2010) apresenta 5 tipos de *modding*, onde cada um deles permite diferentes tipos de possibilidades que determinam os *mods*, *modders* e práticas de *modding*:

- **personalização da interface do utilizador** - as interfaces do utilizador para jogos personalizam a prática e a experiência de interface dos utilizadores (jogadores) para o sistema de jogo e experiência de jogo concebida por desenvolvedores de jogos. Os desenvolvedores de jogos agem de forma a restringir e controlar o que os utilizadores podem fazer, e quais os tipos de experiência que eles podem realizar. Alguns utilizadores, por sua vez procuram atingir algum tipo de vantagem competitiva durante o jogo através do *modding* do *software* de interface do utilizador para o jogo, se os desenvolvedores o permitirem, para adquirir ou revelar informações adicionais que os utilizadores acreditam que irá ajudar no seu desempenho e na sua experiência de jogo. Scacchi (2010) enumera três tipos de personalizações de interface do utilizador. O primeiro e mais comum, é a habilidade do jogador para selecionar, vestir ou colocar acessórios nos personagens que representam a identidade do jogador no jogo. Em segundo lugar, a personalização da palete de cores e da representação das fronteiras elaboradas do ecrã do jogo dentro da interface humano-computador. Em terceiro lugar, são as componentes de interface do utilizador (*add-ons*) que modificam a informação do painel de gestão do jogador em jogo, não modificando quaisquer regras ou funções do mesmo. Esses *add-ons* fornecem informações adicionais sobre o jogo que podem melhorar a experiência de jogo, bem como aumentar a sensação de imersão de um jogador ou consciência dentro do mundo do jogo através da expansão sensorial ou percetual.
- **conversões do jogo** - são talvez a forma mais comum de *modding*. Muitas das conversões são parciais, na medida em que adicionam-se ou modificam-se

personagens no jogo, incluindo a aparência ou capacidade do personagem controlado pelo utilizador, *bots*[4] adversários, *cheat bots*, personagens não-jogador (NPCs - *Non-Player Characters*), objetos como armas, poções, magias e outros recursos, níveis de jogo como zonas, terrenos, paisagens e também a alteração de regras do jogo e *game mechanics*. Segundo Scacchi (2010), alguns *modders* mais ambiciosos vão tão longe que chegam a realizar uma conversão total a partir dos jogos já existentes, tornando-se jogos completamente novos e diferentes em relação à versão original.

- ***mods* de *machinima* e de arte** - podem ser vistos como produtos de esforços de *modding* com a intenção de modificar a experiência de jogo. Estes produtos utilizam os videojogos como meios criativos, de modo que estes sejam mobilizados para outros fins, como por exemplo a criação de cinema *online* ou exibição de arte interativa. *Machinima* é uma forma de modificar a experiência de jogar um jogo específico através de uma gravação da sua história visual durante a sessão de jogo, a fim de conseguir outros objetivos para além da diversão (ou frustração) durante a interação com o jogo. Estas histórias de sessão de jogo podem ser modificadas através da edição de vídeo ou da mistura com outros meios (e.g. gravações de áudio) para permitir uma melhor narrativa cinematográfica ou documentação do desempenho criativo. *Machinimas* podem ser criados a partir de quaisquer jogos que permitam gravar a ação. Há também jogos cuja finalidade é a criação de machinimas, como The Movies, da Activision. Os *mods* de arte também modificam a experiência de jogo através de manipulação, intervenção, apropriação, ou outras formas de transformação criativa do conteúdo visual original de um jogo que é consumido pelos utilizadores durante uma sessão de jogo.
- **personalização de computadores para jogos** - é outra expressão de práticas de *modding* do sistema de jogo. Neste caso os jogadores direcionam os seus esforços na reconfiguração do computador pessoal em algo especializado para suporte de jogos competitivos ou exibição de um ambiente multi-jogador. Scacchi (2010) refere o ponto de vista de Simon (2007) que menciona os PCs "*hot rod*" e "*custom car*", plataformas de *hardware* de computador modificadas tanto para a máxima velocidade como performance, ou aparência especial para o caso dos *mods*. Um PC *hot rod*, normalmente, apresenta um alto desempenho através de CPUs *over-clocked* com dispositivos de refrigeração líquida, múltiplos cartões de memória, várias placas gráficas, rede, áudio de jogo, entre outros. Para alguns jogadores este tipo de computadores especiais encaram o desejo de alcançar uma vantagem competitiva no momento de interação com o jogo em relação aos seus adversários, noutros casos apenas para correr jogos que requerem muitos recursos. Um caso semelhante é o que acontece com os automóveis, onde os seus proprietários alteram ou praticam o *modding* nestes (*custom car*) com o objetivo de aumentar as suas capacidades de condução na estrada, ou para uma situação de corrida de rua.
- ***hacking* em consola de jogos** - é uma prática cujo objetivo, muitas vezes, parece estar diretamente relacionado com o desafio à autoridade dos fabricantes de consolas de jogos. O *hacking* de consolas, em contraste com a personalização de computadores,

[4] Um *bot* (abreviatura de *robot*) é um jogador controlado pelo computador num jogo *multiplayer*, geralmente usado para comportar-se como um jogador real. (retirado de https://developer.valvesoftware.com/wiki/Bot)

> muitas vezes não é muito focado sobre como melhorar a vantagem competitiva em jogos multi-jogador, mas está focada em ampliar o leque de experiências que os utilizadores podem encontrar através de uma consola de jogos. Contudo, o *modding* de consolas é uma expressão de jogadores que estão dispostos a passar à frente da "proteção" e garantias de qualidade que os desenvolvedores de consolas fornecem através garantias de produto, a fim de experimentar a liberdade, competências e aquisição de conhecimentos, assim como o potencial de inovação, que o domínio de engenharia reversa oferece.

A partir desta abordagem pode-se verificar que o conceito de *modding* não se aplica apenas aos jogos de computador, mas também a outros elementos que aliciam os utilizadores a modificá-los, como os automóveis, computadores e consolas de jogos.

5.4.5. Implementação do mod

Antes de partir para a implementação do *mod* existem aspetos que devem ser considerados, como a constituição da equipa. Segundo a comunidade da Valve Software (*Valve Developer Community* (2010)) uma equipa de produção de *mods* deve ser o mais pequena possível. Isto porque liderar uma equipa *online*, ou mesmo local, requer tempo para a gestão dos membros desta, e quantas mais pessoas estiverem envolvidas, maior é o tempo despendido. Para reforçar esta ideia, a comunidade da Valve Software dá o exemplo de dois casos de grande sucesso: Team Forteress com uma equipa de três pessoas e o CounterStrike com apenas uma. Acrescentam ainda que deve-se contratar/nomear para a equipa, pessoas que são absolutamente essenciais para o seu desenvolvimento. Também aconselham que uma pessoa por cada área do *mod* (código, som, modelos, mapas) é o suficiente para a primeira versão do *mod* (ValveCommunity, 2010).

Outro aspeto importante para o autor do *mod* é questionar-se da seguinte forma, "Porque é que alguém deveria jogar o meu *mod*?" (ValveCommunity, 2010). A comunidade da Valve Software argumenta que esta é uma pergunta difícil de responder, mas se o autor do *mod* conseguir responder então está num bom caminho. Pensar nos outros *mods* que já existem e o que eles oferecem aos jogadores. Pensar no que o *mod* poderá oferecer de novo e se será o suficiente para cativar os jogadores que jogam outros *mods*. Estas são algumas sugestões que comunidade da Valve Software sugere aos *modders* iniciantes.

A comunidade de desenvolvimento da Valve (2010) lista alguns componentes de produção que fazem parte do desenvolvimento de um *mod*:

- ***level design*** - construção de mapas, ambientes virtuais onde o jogo decorre. Normalmente quem constrói mapas denomina-se por *level designer* ou *mapper*;
- **programação -** programação de código que define a forma como o mundo virtual e as mecânicas se comportam;
- **modelação** - modelação de objetos 3D que aparecem no ambiente do jogo, como personagens, armas ou casas;
- **materiais** - constituídos pelas texturas e sombras que fazem parte dos objetos modelados;
- **sons** - áudio que acrescenta uma quarta dimensão ao ambiente virtual;

- **partículas** - efeitos como fumo, faíscas, sangue e fogo são criados a partir de partículas;
- **coreografia** - criar cenas coreografadas, através de animação, nas personagens do jogo.

Como já foi referido nos tópicos anteriores, alguns estúdios de desenvolvimento criaram *kits* de desenvolvimento (SDK's) e outras ferramentas de *modding* para fornecer aos utilizadores com a finalidade de sustentar os videojogos desenvolvidos pelas empresas e de certa forma receber contributos inovadores por parte dos utilizadores. As empresas id Software e a Epic Games, criadores da série de jogo Unreal, começaram a fornecer aos entusiastas em jogos (que também tinham conhecimentos de programação) ferramentas de *software* que permitem aos utilizadores editar o conteúdo do jogo, a mecânica de jogo, entre outros (Scacchi, 2010).

A Valve Software, como já foi mencionado, também desenvolveu ferramentas de *software* tal como a Id Software e Epic Games - kit de desenvolvimento Souce (Source SDK). Este *kit* estende-se em vários programas de desenvolvimento, cada um com a sua particularidade e objetivo que possibilitam aos *modders* realizar algumas camadas acima descritas (*level design*, programação, modelação, materiais, sons, partículas e coreografias). Um desses programas é o Hammer, que tem a principal particularidade de servir para desenhar níveis de jogo para o motor de jogo Source da Valve Software (posicionar modelos 3d, texturas, iluminação, geometria, entradas de *gameplay*, scripts I/O, estrutura dos NPCs[5] e compilação e execução dos níveis de jogo). Outro programa é o Model Viewer que permite aos utilizadores visualizar os modelos já criados. Quanto ao Face Poser, este serve para aceder a animações faciais e aos sistemas de coreografia dos personagens (editar expressões faciais, gestos, movimentos, fonema). Os *mods* desenvolvidos a partir deste *kit* de desenvolvimento são compilados a partir do VisualStudio da Microsoft e desenvolvidos neste na linguagem de programação C++ (ValveCommunity, 2010). Para este ou outros *kits* de desenvolvimento, a modelação dos objetos pode ser feita a partir de programas independentes da produtora, como por exemplo, Maya, Blender, 3D Studio Max, entre outros.

Conforme a comunidade da Valve Software, após o lançamento do *mod* desenvolvido, os jogadores podem entrar numa fase de adoração do novo *mod*. Mas segundo a experiência deles no campo de multi-jogador *online*, um *mod* apenas permanece popular, desde que este seja suportado constantemente. Mesmo que o *mod* seja de grande dimensão, não será um fator que leva a angariar grande número de jogadores numa primeira versão. O número de pessoas a aderir ao *mod* aumenta ao longo do tempo através de repetidos lançamentos de novos conteúdos, correções de erros, e apoio da comunidade. Tanto o CounterStrike como o Team Fortress começaram pequenos e cresceram ao longo do tempo e cada vez que são lançadas novas versões, mais jogadores os experimentam e começam a jogá-los. "*Knowing what to fix, what to change, and how to listen to your community is a continual learning process*" (ValveCommunity, 2010).

[5] Non-Player Characters

Comentários finais do enquadramento teórico

Neste enquadramento teórico pode-se verificar que o olho humano é um órgão imprescindível para o funcionamento de um dos sentidos mais importantes, a visão. Estima-se que cerca de 80% da informação recebida do ambiente externo é captada pelo sentido da visão (Cotti, 2009). Possivelmente, com esta ideia em mente, durante os últimos anos a comunidade científica tem presenciado avanços interessantes na área de *eye tracking*, uma técnica que possibilita aos investigadores estudar e analisar os movimentos dos olhos.

Uma área importante na qual este trabalho se baseia é a área dos videojogos, que segundo Salen e Zimmerman (2004) são sistemas no qual os jogadores se envolvem num conflito artificial, definido por regras, que determinam um resultado quantificável. Esta área de entretenimento, nos últimos anos, tem vindo a aumentar em termos de negócio e tende agravar-se até aos 73,5 biliões de dólares no ano de 2013 (Ribeiro, 2010).

Quanto ao conceito de usabilidade, destacou-se a preocupação de a interligar com o conceito de videojogos acabando-se por verificar que estes são difíceis de combinar. No estudo de Melissa Federoff (Federoff, 2002), ela descobre que mesmo para as pessoas da área de videojogos, o conceito de *game usability* não é usado ou mesmo desconhecido. No entanto, esta situação não é assim tão grave quanto parece. Afinal, os videojogos não passam de um produto cujo principal objetivo é entreter o público, e não tanto uma ferramenta para facilitar as tarefas diárias de um utilizador.

Adquirir e analisar dados da experiência do jogador não só permite medir os efeitos que os videojogos têm nos jogadores, como também dá uma visão aos *designers* de como os jogadores vêm os seus jogos (Sasse, 2008). Segundo Kennerly (2003), ao analisar a experiência de um jogador pode-se ampliar uma perspetiva do *designer* do jogo, provar ou refutar hipóteses, e substituir opiniões por factos, tornando-se assim uma poderosa ferramenta para validar o *game design* (Kennerly, 2003). Tendo isto em mente, foram apresentadas algumas das investigações e projetos existentes feitos na área de avaliação de videojogos que usaram as técnicas de *eye tracking, logging* e avaliação com *game heurístics*.

Como este trabalho também passa por uma modificação de um videojogo, houve a necessidade de abordar tal conceito (*Videogame Modding*). Pode-se verificar que o ato de *modding* (modificar), para além de ser uma mais-valia para as produtoras de videojogos, aumentando o tempo de vida dos seus videojogos, também pode ser uma forma de ensinar aos jogadores (*modders*) o processo de desenvolvimento de um jogo e eventualmente poderá surgir uma oportunidade de emprego numa produtora de videojogos. Para além disso, foram abordados alguns fatores e dicas importantes para o desenvolvimento de um *mod* de jogo.

segunda parte
investigação empírica

6. Investigação empírica

Na área dos videojogos, é importante ter em consideração que a atividade que proporcionam deve ser compreendida através de um processo de análise. A análise proposta nesta dissertação passa pela verificação minuciosa das ocorrências durante sessões de jogo através de métodos alternativos às abordagens clássicas de avaliação de aplicações digitais. Desta forma, poderão ser encontrados problemas relacionados com usabilidade e conceção do jogo ou ainda com aspetos relacionados com as escolhas de cada jogador.

No presente capítulo são apresentadas várias fases de desenvolvimento de um *mod* baseado no motor Source e de um protótipo (aplicação de análise e visualização) baseado em Flash. Esta aplicação de análise e visualização é capaz de representar dados obtidos por um jogo (registo em *logfiles*) e dados obtidos através de um *eye tracker* representativos do olhar do jogador, para uma dada situação de jogo. Com esta aplicação é possível obter um maior conhecimento da forma como os jogadores interagem individualmente e em equipa dentro de um espaço de jogo.

Neste capítulo também são apresentados os preparativos de uma experiência de jogo (*mod* desenvolvido) composta por 6 jogadores e 6 rondas onde cada jogador teve a oportunidade de usar o *eye tracker* numa das rondas. Também são apresentados e analisados os resultados originados dessa mesma experiência.

Finalmente, os comentários finais e conclusões sobre todo este estudo onde são confrontadas as hipóteses com os objetivos e perguntas de investigação, é apresentada uma reflexão crítica final sobre o estudo, são numeradas algumas limitações e problemas encontrados e por fim as perspetivas futuras de investigação.

6.1. Paradigma do objeto de estudo

Esta secção visa explicar com algum detalhe a fase de desenvolvimento de um *mod* e de uma aplicação de análise e visualização. O *mod* tem como objetivo ser preparado para registar *logfiles* sobre ações e comportamentos do jogador para que posteriormente estes sejam lidos, processados e apresentados pela aplicação desenvolvida.

Este processo é dividido por várias fases: em primeiro lugar, (i) escolher um motor de jogo para desenvolver o *mod*; depois, (ii) investigar a ferramenta de desenvolvimento nesse mesmo motor de jogo (*framework* ou kit de desenvolvimento); (iii) instalar o *mod* de base ou *template*; (vi) criar um mapa de jogo; (v) fazer as modificações necessárias no *mod* para que seja possível registar em *logfiles* os dados relativos aos jogadores; (vi) fazer um teste com apenas um jogador para experimentar o *mod* juntamente com *eye tracker*; (vii) escolher uma

ferramenta de autoria (e.g. Adobe Flash) para desenvolvimento de *software*; (viii) preparar a aplicação de análise e visualização para interpretar e processar os dados dos *logfiles* registados pelo *mod*; finalmente, (ix) preparar a aplicação, usando paradigmas de visualização, para representar os dados sobre os jogadores.

No fim desta secção é caracterizada a amostra do estudo, apresentados os instrumentos de recolha de dados usados na experiência, como se foi preparada a experiência e o processo durante a experiência propriamente dita.

6.1.1. Ferramentas para a construção do contexto empírico

a) Motor de jogo

Um motor de jogo é um sistema concebido para a criação e desenvolvimento de videojogos. Os principais motores de jogo fornecem uma *framework*, também conhecida por API (*Application Programming Interface*) ou SDK (*Software Development Kit*), que os programadores usam para criar videojogos para consolas e computadores pessoais. A principal funcionalidade tipicamente fornecida por um motor de jogo inclui um motor de renderização para gráficos 2D ou 3D, um motor de física ou de deteção de colisão, som, script, animação, inteligência artificial, redes, *streaming*, gestão de memória, *threading*[6], suporte de localização, e um gráfico de cena. O processo de desenvolvimento de jogos é muitas vezes economizado, em grande parte, pela reutilização ou adaptação do mesmo motor de jogo para criar jogos diferentes, ou para tornar mais fácil a importação de jogos para várias plataformas (Ward, 2008).

São inúmeros os motores de jogo existentes nos dias de hoje e são vários os tipos de motores. Contudo, apenas serão referidos aqueles que já ofereceram jogos notáveis na categoria de *First Person Shooter;* por exemplo:

- **CryEngine** - Este motor de jogo foi desenvolvido pela Crytek (2012) que já vai na terceira versão. Também contém um SDK para desenvolvimento de videojogos, disponível para qualquer pessoa interessada na área. No entanto, precisará de licença caso queira comercializar o videojogo desenvolvido nesta *framework*. O motor foi base de vários videojogos de sucesso como o conhecido FarCry e os famosos Crysis e Crysis 2.
- **Frostbite Engine** - Um motor de jogo desenvolvido pela EA Digital Illusions CE, criadores da série Battlefield. A DICE (equipa de desenvolvimento) usaram duas versões (v1.0 e v1.5) do motor para as suas séries de Battlefield: Bad Company, Battlefield 1943 e Battlefield: Bad Company 2. A nova geração do motor Frostbite2 (2011) é usada numa das maiores revelações de videojogos do ano de 2011, o Battlefield 3. Este motor também serviu de base de desenvolvimento de outros videojogos como Medal of Honor (*multi-player)* e Need for Speed: The Run. Quanto ao *modding* do Battlefield 3, ainda não existe uma resposta definitiva sobre a possibilidade de criar *mods* do videojogo. O novo motor Frostbite 2.0, segundo Toshiro Tanaka (2011), é simplesmente demasiado complexo para os *modders* poderem usar,

[6] Um *thread* de execução ou *threading* é a menor sequência de instruções programadas, que podem ser geridas de forma independente por um agendador do sistema operativo.

acrescentando ainda que existem complicadas questões de licenças com o *software* de terceiros no Frostbite.

- **GoldSrc** - Uma versão modificada do código base do motor QuakeWorld, que por sua vez é um desenvolvimento de base de código do motor Quake. Este motor já se encontra oficialmente inactivo, mas ainda continua a ser usado por alguns programadores. Os clássicos Half-life em 1988 e Counter-Strike em 2000 foram desenvolvidos neste motor (half-lifeWiki, 2012).
- **id Tech** - id Tech é a família de motores de jogo desenhado e desenvolvido pela id Software. Antes da apresentação do Tech ID 5 baseado no videojogo RAGE, os motores não tinham designação oficial e, como tal, eram conhecidos como Doom Engine e Quake Engine. As versões do id Tech, 1, 2, 3 e 4 foram lançados como *software* livre sob a GNU General Public License.
- **IW Engine** - IW Engine é um motor de jogo desenvolvido pela Infinity Ward e é baseado no motor id Tech 3, que foi usado pela primeira vez sobre o jogo Call of Duty altamente bem-sucedido. Desde então, tem sido continuamente melhorado e utilizado nos jogos Call of Duty 4, Call of Duty: World at War, Call of Duty: Modern Warfare 2 e Call of Duty: Black Ops (InfinityWard, 2012).
- **Quake engine** - Também conhecido por **id Tech 2,** este motor de jogo foi escrito em 1996 para o desenvolvimento do clássico Quake, escrito pela id Software.
- **Source engine** - Source é um motor de jogo 3D desenvolvido pela Valve Corporation. Foi lançado em junho de 2004 com o Counter-Strike: Source e logo em seguida Half-life 2, e tem estado em desenvolvimento ativo desde então. Excecionalmente para um motor de jogo, Source foi projetado para receber constantes atualizações incrementais e não tem um esquema numerado de versões como outros motores de jogo. O Source SDK está disponível gratuitamente via Steam com a compra de um jogo baseado nesta fonte, como Half-Life 2 ou Counter-Strike: Source. A tecnologia Source é a base de alguns dos mais aclamados jogos de PC e consola, incluindo: Half Life 2: Episódios Um e Dois, Portal e Portal 2, Team Fortress 2 e Counter-Strike: Source (Valve, 2007).
- **Unity** - É um motor de desenvolvimento totalmente integrado para a criação de conteúdo 3D interativo. Fornece funcionalidades completas *out-of-the-box*[7] para criações de alta qualidade, conteúdo de alto desempenho e com a possibilidade de publicação em múltiplas plataformas. O Unity é um motor que ajuda programadores e *designers* independentes, grandes e pequenos estúdios, corporações multinacionais, estudantes e entusiastas a reduzir drasticamente o tempo, esforço e custo de desenvolvimento de videojogos (UnityTechnologies, 2012). Este motor contém um programa de desenvolvimento muito acessível para iniciados na área de videojogos.
- **Unreal Engine** - O Unreal Engine, que já vai na terceira versão, é um motor de jogo desenvolvido pela Epic Games, com o primeiro lançamento do FPS Unreal em 1998. Embora originalmente desenvolvido para *first-person shooters*, tem sido utilizado com sucesso noutros géneros, incluindo *stealth*, MMORPGs e RPGs. Com o seu núcleo escrito em C++, o Unreal Engine possui um alto grau de portabilidade e é uma

[7] Out-of-the-box é uma expressão que descreve o pensamento criativo. O termo é usado como um advérbio para descrever o pensamento ou como um adjetivo para descrever as ideias. [retirado de http://searchcio.techtarget.com/definition/out-of-the-box]

ferramenta usada por muitos programadores de videojogos nos dias de hoje. Este motor contém uma *framework* (UDK – *Unreal Development Kit*) que serve de base para quem quer desenvolver videojogos com o motor do Unreal (I. EpicGames, 2012).

b) Motor "Source": porquê?

A escolha do motor em si não é importante (nível gráfico, rapidez de processamento, detalhe dos modelos, entre outros) mas sim o que pode oferecer para este estudo. Um fator importante a considerar num motor de jogo é a possibilidade de se poder editar a base de um jogo já desenvolvida e ter ao dispor ferramentas para criação de mapas, e edição da lógica de jogo. Documentação muito completa e uma comunidade ativa são outros fatores muito importantes que servirão de guia para o desenvolvimento do estudo.

Dois dos motores de jogo, que perante a comunidade de *modding* são considerados os mais poderosos, são o Source Engine e Unreal Engine. Ambos oferecem uma framework muito completa para a prática de *modding*, e são sustentados pela linguagem de programação C++.

O Unreal Engine oferece um jogo de base dentro da categoria FPS, mas num estilo de jogo rápido e não tão tático. No caso do Source Engine já existe um jogo de base, também dentro da categoria FPS, mas num estilo de jogo mais tático e de *teamplay*, sendo esta a maior razão de escolha deste motor de jogo para este estudo, visto que se pretende estudar as várias reações/decisões dos jogadores perante um trabalho de equipa e individual. Outra grande vantagem do Source Engine é a imensa documentação disponível na *Valve Developer Community* wiki[8] e em vários fóruns de discussão oficiais e não oficiais.

A Valve Software tem uma visão muito otimista em relação ao Source Engine afirmando o seguinte: "*Valve's Source engine is widely recognized as the most flexible, comprehensive, and powerful game development environment available.*" (Valve, 2007)

6.1.2. Ambiente de desenvolvimento do *software*

Constata-se que o motor de jogo mais adequado para este estudo é o Source Engine devido ao estilo de jogo base que é oferecido e à vasta informação disponível pela comunidade.

Nesta secção pretende-se mostrar detalhadamente os passos efetuados para a investigação e configuração/utilização da *framework* do Source Engine, nomeadamente a sua instalação, prática de *modding* em C++ e criação de um mapa de jogo para testes.

a) State of the art técnico sobre frameworks de desenvolvimento (SDKs)

Cada motor de jogo tem a sua framework de desenvolvimento, mas apenas algumas estão disponíveis ao público. Na seguinte lista encontram-se alguns ambientes de desenvolvimento que são acessíveis a qualquer pessoa que pretenda desenvolver um videojogo:

- **Source Software Development Kit –** Kit escolhido para desenvolvimento pelas razões já explicadas.
- **Unreal Development Kit –** Muito semelhante ao Source SDK, apesar de não se enquadrar no tipo de jogo pretendido para análise.

[8] https://developer.valvesoftware.com/wiki/Main_Page

- **Unity –** Não oferece um jogo base para poder desenvolver, o que implicaria maior trabalho de *design* e implementação.

b) Investigação sobre SDK

Como já foi referenciado na secção anterior, o motor de jogo escolhido para este estudo é o motor Source que por sua vez traz uma framework denominada por Source SDK. Esta *framework* disponibilizada pela Valve Software oferece aos programadores e *designers* de videojogos um conjunto de ferramentas necessárias para a criação de mapas ou *mods* para o motor Source, com a exceção de Left 4 Dead, Left 4 Dead 2 e Alien Swarm. Estes jogos são embalados com os seus próprios SDKs, chamados de Left 4 Dead Authoring Tools, Left 4 Dead 2 Authoring Tools, e do Alien Swarm SDK respetivamente. Estas *frameworks* são todas extremamente parecidas com o Source SDK, mas adaptadas ligeiramente para atender às necessidades de cada jogo.

O Source SDK está disponível para todos os compradores de jogos Source da Valve, ou seja, para ter acesso a estas ferramentas é necessária a compra de um produto da Valve na Steam, como por exemplo, Half-Life 2 ou Team Forteress 2. Após a compra do produto pode-se verificar que na zona de Ferramentas da Steam já se pode fazer a transferência do Source SDK e Source SDK base.

Source SDK Base é um jogo/ferramenta, compartilhado entre todos os proprietários de videojogos Source da Valve Software, onde os *modders* podem basear os seus projetos. Ambos os *mods singleplayer* e *multi-player* podem usá-lo. O Source SDK Base também é um conjunto de ficheiros necessários para reproduzir qualquer tipo de *mods* criados usando o Source SDK. Por exemplo, quando se pretende apenas jogar *mods* de origem, só necessita do Source SDK Base. Mas se o objetivo também passa por desenvolver *mods* e mapas, será necessário tanto o Source SDK como o Source SDK Base (ValveCommunity, 2010).

Segundo Sasse (2008) as vantagens da modificação em Source SDK são:

- Fornecimento de diversas ferramentas de desenvolvimento para apoiar no desenho de mapas padrão e modificações do jogo base;
- Uma extensa biblioteca de modelos e recursos incluídos na framework;
- Permite a modificação do código fonte do motor de jogo, desta forma permitindo mudanças internas da mecânica de jogo principal;
- Comunidade de programadores de suporte on-line fornece documentação e tutoriais para os recursos (de outra forma menos documentados) do Source SDK.

O Source SDK também tem as suas desvantagens, apesar de a maioria não ser relevante para este estudo (Sasse, 2008):

- As modificações que não sejam do tipo FPS são possíveis, mas exigem um enorme esforço e experiência, a fim de realizar as modificações extensas que são necessárias;
- A integração de novos modelos personalizados, animações e entidades é possível, mas exigem mudanças de enorme esforço e grandes alterações no código fonte.
- A extensão e a complexidade do motor Source requerem programadores experientes e qualificados, bem como profundo conhecimento da arquitetura do motor.

Destaca-se que esta última desvantagem referida é a maior preocupação para este estudo. Isto porque independentemente da experiência que um programador possa ter numa linguagem de programação, o facto de não estar dentro da arquitetura/estrutura do motor, poderá vir a dificultar imenso no desenvolvimento do *mod*. No entanto, existe uma boa documentação e comunidade que ajudam a resolver o problema como poderemos constatar nas seguintes secções.

c) Configurações e instalação do SDK

Antes de criar um *mod* em Source SDK de raiz deve garantir-se que a transferência da *framework* já foi efetuada. A aplicação Source SDK deve correr-se a partir da Steam na secção de ferramentas. Após a execução da aplicação aparecerá uma janela onde estão listadas as várias aplicações, *links* para documentação e utilidades que estão incluídas no pacote da *framework*. Para além dos itens referidos também existe um campo de escolha extremamente importante para selecionar a versão do motor em que se vai trabalhar:

- Source engine 2006
- Source engine 2007
- Source engine 2009

A Figura 22 representa a janela inicial que aparece quando se corre a *framework* Source SDK a partir do Steam.

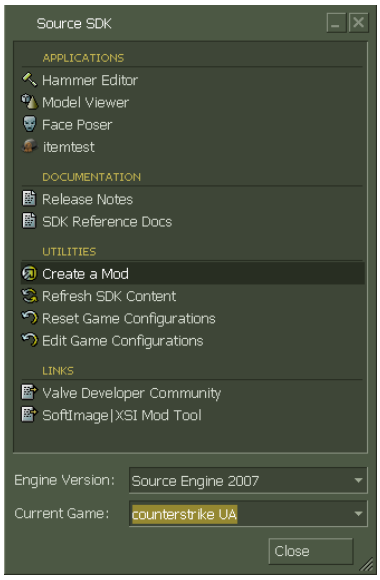


Figura 22 - Source SDK: janela principal

Todas as configurações necessárias dependem da versão escolhida. Por exemplo, ao escolher a versão 2007, todos os mapas desenvolvidos a partir da framework serão compatíveis com o motor da mesma versão.

Após a escolha da versão para trabalhar já se pode criar o *mod* clicando-se numa das utilidades listadas "Create a *Mod*" (Figura 23) onde de seguida aparecerá um wizard para fazer as configurações de base.

Em primeiro lugar escolhe-se a opção “Start a Multiplayer *mod* from a template” que inclui um pacote da valve com o código base em C++ para se poder editar e o próprio *mod* já instalado e pronto a executar através do Steam. No segundo passo (Figura 24) escolhe-se o caminho onde serão tranferidos todos os ficheiros em C++ necessários para editar o *mod* e ao mesmo tempo introduz-se o nome que o *mod* vai ter.

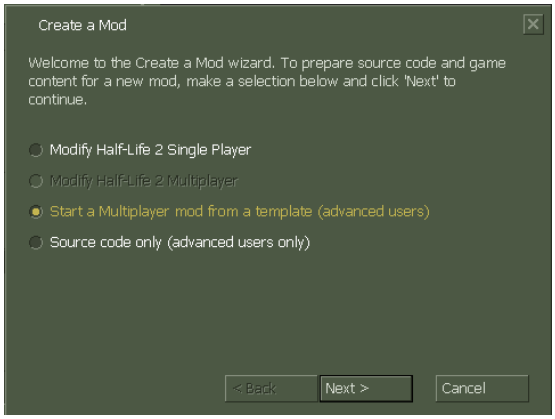


Figura 23 - Criação de um Mod: tipo de mod

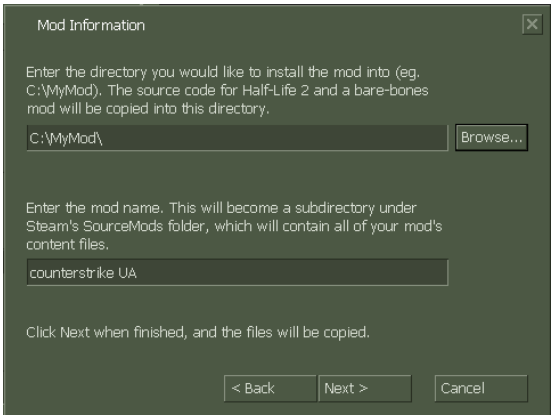


Figura 24 - Criação de um Mod: destino e nome do mod

No terceiro passo (Figura 25) é possível escolher quais as funcionalidades base que o *mod* já vai ter como ativas. No caso do *mod* desenvolvido, optou-se por escolher todas as opções para proporcionar aos jogadores uma experiência de jogo mais completa. No passo seguinte (Figura 26), todos os ficheiros para edição são transferidos para a diretoria escolhida no passo 2 e é instalado o *mod* base para que já seja possível experimentá-lo através do Steam.

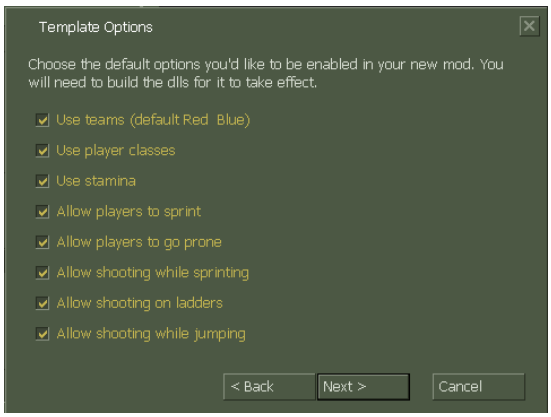


Figura 25 - Criação de um Mod: selecção de extras

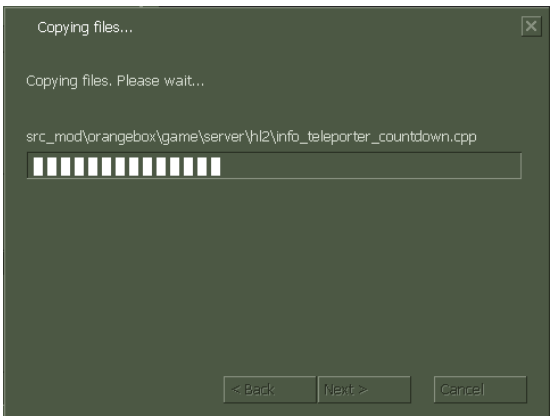


Figura 26 - Criação de um Mod: transferência e instalação

Finalmente, após a conclusão da transferência e instalação, o *mod* está pronto a ser editado ou executado. Se a opção “Open readme file” (Figura 27) estiver ativa o Steam reencaminhará o utilizador para uma página específica da documentação da Valve Development Community, onde são apresentadas algumas instruções básicas para quem está a fazer o seu primeiro *mod*.

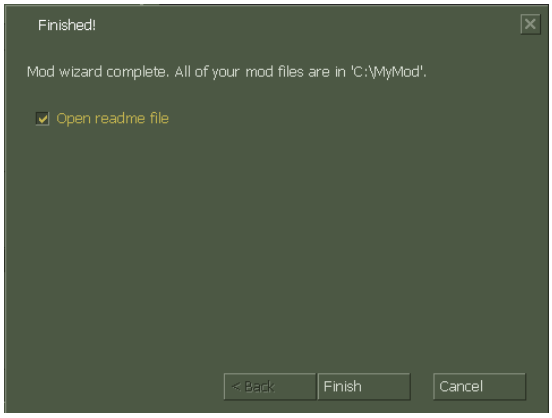


Figura 27 - Criação de um Mod: finalização e ajuda

d) Problemas relacionados

Na fase inicial do desenvolvimento do *mod* foram encontrados vários problemas relacionados com a instalação do Source SDK. Os criadores do Source SDK lançaram a última versão com problemas que têm dificultado o processo de instalação a muitos *modders* iniciantes.

Segundo a comunidade de desenvolvimento em SDK (2010), a Valve Software não teve os cuidados necessários para lançar a última versão sem *bugs*, algo que se tem arrastado há muitos meses sem qualquer intervenção para melhoria da situação.

Existem 3 bases de SDK desenvolvidas pela Valve: sdk2006, sdk2007 e sdk2009. A versão mais recente encontra-se com vários problemas graves, nomeadamente a sua instalação não conter quaisquer ficheiros que possam contribuir para o desenvolvimento de um *mod*.

No entanto, a partir de algumas pesquisas efetuadas, muitos dos *modders* já experientes dizem que o ideal é esquecer a versão sdk2009 e desenvolver em sdk2007, visto ser a última versão do sdk com menos problemas.

Seguindo a sugestão dada pelos *modders* mais experientes, instalou-se a versão sdk2007. Esta versão já contém os ficheiros e arquitectura de pastas que a última versão não tem. No entanto, esta versão ainda é portadora de vários problemas.

Ao correr a solução em Visual Studio, para poder programar no jogo base oferecido pela Valve, verifica-se que ainda assim existem falta de ficheiros como por exemplo, “weapon_sdk.h” e “weapon_sdk.cc” (ficheiros que contêm a lógica relacionada com as armas de jogo). Estes ficheiros são necessários para poder compilar o jogo. Na sua falta, e após uma pesquisa e interação em fóruns da comunidade, surgiram várias propostas desses ficheiros em falta. De acordo com a comunidade, este é mais um dos problemas que a Valve Software não resolveu, enviando para o mercado uma versão (sdk2007) com erros e falta de ficheiros.

Alguns destes ficheiros propostos resolvem o problema, mas depois surgem outros erros, até que alguns *modders* explicam que existem várias versões desses mesmos ficheiros e acabam por dizer quais são os que funcionam. Esta informação foi pertinente para poder passar para o passo seguinte (correr o jogo a partir do Visual Studio, sem erros de compilação).

6.1.3. Modding

A Valve Software lançou um Source Development Kit (SDK) para que a comunidade pudesse modificar a base do Half-Life 2. Este kit de desenvolvimento, baseado na linguagem de programação C++, serviu de base para o projeto prático que inclui o desenvolvimento de um *mod multi-player* para registar dados dos jogadores em *logfiles*. Este *mod* desenvolvido é similar ao Counter Strike: Source, mas sem grande parte das funcionalidades disponíveis no jogo original.

a) Registo em logfiles

Uma fase deste trabalho (teste com um jogador onde se experimentou o *mod* juntamente com *eye tracker*) passou pela obtenção de informação dos jogadores (métricas) que por sua vez são registados em ficheiros para que depois fossem analisados pela aplicação de análise desenvolvida. Esta informação pode ser adquirida diretamente a partir da modificação do jogo e também pelo programa de leitura do *eye tracker* (Tobi T120). No caso do *eye tracker*, este já inclui um *software* (Tobii Studio) que permite exportar para *logfiles* as coordenadas de visualização no ecrã. Assim, não existe qualquer dificuldade no registo do *logfile*, pois todo ele é feito automaticamente pelo sistema já existente. Quanto ao resto da informação necessária como, por exemplo, a posição do jogador no espaço do jogo em xyz, propriedades do jogador e respetivos eventos, já terá que ser adquirida através do desenvolvimento em C++. Assim, foi necessário modificar o código do videojogo para que este em tempo real possa estar a registar dados dos jogadores em *logfiles*.

O cenário ideal seria desenvolver uma estrutura *standard* para o registo dos *logfiles*. Desta forma, a aplicação de análise desenvolvida poderia ser usada noutros estudos para outros videojogos específicos, que por sua vez estariam a registar *logfiles* com uma estrutura compatível. No entanto, esta regra estaria a obrigar outros programadores a desenvolver um registo que seguisse a mesma estrutura, caso contrário a aplicação de análise não conseguiria interpretar tais registos.

Antes da fase de testes com um jogador ocorrer, foi necessário desenvolver em C++ (no *mod*) o sistema de registo em *logfiles*. Este sistema foi preparado para buscar em determinadas zonas do código do jogo a informação relativa aos jogadores para que depois pudessem ser indexados e registados em ficheiros (*logfiles*).

O registo em *logfiles* é subdividido em 4 ficheiros, cada um com diferentes dados que se complementam. Três dos *logfiles* são oriundos do *mod* enquanto que um é registado automaticamente pelo *eye tracker*:

- ***Logfile* Mapa** - O *logfile* do mapa contém informação relativa às dimensões gerais do mapa que são definidas como coordenadas máximas e mínimas, sendo possível a partir destes dados calcular o comprimento, largura e altura total do mapa. Neste *logfile* também são gravados os dados referentes às entidades/objetos envolventes do mapa (e.g. coordenadas x, y e z, tipo de modelo e respetivo nome). Com a informação das dimensões, é possível criar uma matriz base para que depois se possa simular todos os acontecimentos e posições apresentados no programa em desenvolvimento.

- ***Logfile* Jogadores** - Neste *logfile* são registados, em tempo real, alguns dados referentes a todos os jogadores que participam na experiência. São registados dados como: nome do jogador, equipa, coordenadas x, y e z da posição do jogador, ângulo x, y e z, vida, armadura, arma em uso, se está a correr, a saltar, a disparar, agachado, entre outros. A partir destes dados é possível verificar os percursos dos jogadores/equipas durante o momento de jogo, assim como as áreas de maior incidência no mapa. Também é possível ver em pormenor quais são os vários estados e propriedades do jogador num momento específico como, por exemplo, saber se estava a saltar, a correr ou até agachado.
- ***Logfile* Eyetracker** - Este *logfile* é atualizado em tempo real pelo sistema de *eye tracking* (*eye tracker* do tobii T120[9] guarda 120 amostras/segundo referentes ao comportamento do olhar)que contém os dados referentes aos movimentos oculares do jogador principal e dados gerais relativos às configurações que o mesmo está a usar no computador. Os dados gerais registados são: a resolução do ecrã; unidade das coordenadas; raio de fixação; data e tempo do início do registo; entre outros. Quanto aos dados do jogador, são registados o *timestamp*, duração da fixação num determinado ponto e coordenadas x e y do ecrã para o qual o jogador estava a olhar. É importante realçar que os dados gerais deste *logfile* como a resolução são importantes para servirem de base para a implementação da funcionalidade de deteção automática dos objetos que foram olhados pelo jogador durante o jogo, assim como os dados relativos à data e tempo de início do registo que têm utilidade na sincronização entre os dados existentes no *logfile* dos jogadores e o *logfile* do *eye tracker*.
- ***Logfile* Eventos** – O *logfile* de eventos tem como principal objetivo de mostrar na aplicação um histórico do que aconteceu durante a experiência de jogo. A partir deste *logfile* é possível saber quando um jogador se conectou ao servidor, quando fez *spawn* no mapa, quando foi atingido e qual a quantidade de energia perdida, quando foi morto e por quem, quando saiu do jogo, quando entrou/mudou de equipa, entre outros. Este *logfile* também serve para dar apoio à interpretação do *logfile* dos jogadores, isto para que ao registar os percursos dos jogadores, seja possível identificar os pontos em que o jogador não está ativo (possivelmente morto ou em modo espetador), e desta forma mostrar no percurso os locais onde o jogador "nasceu" ("*spawn*" num contexto de videojogos) e onde foi morto, dando assim maior riqueza à informação apresentada pela aplicação desenvolvida.

b) Sistema de debug

O Visual Studio oferece um sistema de *Debug* capaz de detetar erros de compilação do *mod*. No entanto, também é necessário detetar erros de lógica, de forma que os procedimentos armazenados sejam executados corretamente. Isto é possível com as funções do ambiente de desenvolvimento integrado de depuração. Estes sistemas permitem parar em locais de procedimento, inspecionar memória e registar valores, variáveis de mudança, observar tráfego de mensagens e obter um olhar mais atento ao que o código faz (Microsoft, 2010).

Durante a implementação do *mod*, por vezes é necessário saber dentro do jogo quais os valores que estão a passar dentro das variáveis. Para isto, o Source SDK oferece uma função

[9] Tobii - http://www.tobii.com/en/eye-tracking-research/

em C++ denominada por DevMsg(), que escreve no topo do ecrã durante o momento de jogo. Esta função é útil, por exemplo, para saber o que está a ser registado nos *logfiles*:

```
DevMsg("Health VALUE: %i\n", m_pSDKPlayer->GetHealth());
```

Este exemplo faz com que apareça no ecrã os valores da energia do jogador em tempo real.

c) Problemas encontrados

A maior dificuldade nesta fase passa por saber quais os ficheiros e respetivos métodos onde é possível encontrar a informação de jogo necessária: posições, eventos, estados e propriedades dos jogadores. Para quem está dentro da arquitetura do motor Source esta questão é fácil. o entanto, para quem não está, a melhor opção é procurar ajuda na comunidade de *modders* do Source.

6.1.4. Mapa de jogo: porquê?

O mapa é usado para descrever arenas em videojogos de *multi-player* competitivo onde a jogabilidade é fortemente dependente da conceção do terreno (como jogos de estratégia em tempo real (*real-time strategy*) e em primeira pessoa (*first-person shooters*)). Este termo é também muitas vezes transportado para videojogos de *singleplayer*, para descrever os níveis com um elevado grau ou extensão no *design* de terreno, ou simplesmente como a soma de todas as áreas do jogo.

De início, pretendia-se criar dois mapas de jogo: (i), um mapa muito simples com poucos obstáculos e distrações (um número de objetos reduzido); e (ii) um mapa mais complexo, mais próximo de um mapa final com mais prédios, andares e obstáculos (um número de objetos elevado). À partida, com dois mapas diferentes, seria possível fazer estudos distintos: com o mapa simples, os jogadores estariam mais concentrados em eliminar o inimigo o mais rapidamente possível visto que mal se iniciava o jogo ambas as equipas estariam expostas uma à outra. No mapa mais complexo, os jogadores teriam outras abordagens mais táticas para atingir os objetivos de jogo. No entanto optou-se por criar apenas um mapa simples mas com alguns objetos para poderem ser identificados pelo sistema em desenvolvimento (*eye tracking* + *logging*) e para testar o mesmo.

Também existem alguns mapas já desenvolvidos e disponibilizados na Internet pela comunidade. No entanto, não houve acesso aos componentes de edição do mapa para, por exemplo, poder definir os nomes dos objetos como “Cadeira”, “Mesa”, “Candeeiro”, “Árvore”, entre outros. Estes nomes são importantes para que depois na aplicação de visualização e análise seja possível identifica-los mais facilmente nos *logfiles*. Por isso optou-se por fazer um mapa de raiz que tem a vantagem de poder controlar totalmente a disposição e propriedades de todos os elementos que compõem o mapa de jogo e, com isto, saber exatamente o que é esperado no *logfile* na fase de desenvolvimento.

6.1.5. Hammer: breve descrição

Como já foi descrito na secção 5.4.5, "Hammer" é um programa que permite desenhar níveis de jogo para o motor de jogo Source da Valve Software.

a) Configurações necessárias

Para que seja possível a integração do mapa a criar no *mod* em desenvolvimento é necessário ter em atenção a versão do Hammer. A versão do mapa tem que corresponder à versão do *mod*, ou seja, a versão Source Engine 2007.

Como se trata de um *mod multi-player*, significa que vão existir mais que um jogador em jogo. Para além disso também vão existir duas equipas para tornar a experiência competitiva: a equipa Azul e a equipa Vermelha. Estas equipas têm que ser configuradas num documento[10] de texto relacionado com o Hammer, de forma que seja possível criar os *spawn points* (locais onde as equipas entram no jogo) de ambas as equipas. Se esta configuração não for feita, apenas será possível criar mapas de *single-player*, sendo possível apenas criar *spawn points* para um jogador.

b) Desenvolvimento do mapa

Após as configurações necessárias para criar um mapa *multi-player* já se pode iniciar o desenvolvimento do mapa (modelação, iluminação, aplicação de texturas, entre outros) usando o programa de desenvolvimento Hammer. A Figura 28 representa uma imagem do programa de desenvolvimento Hammer enquanto se desenvolvia o mapa de jogo para ser usado neste estudo.

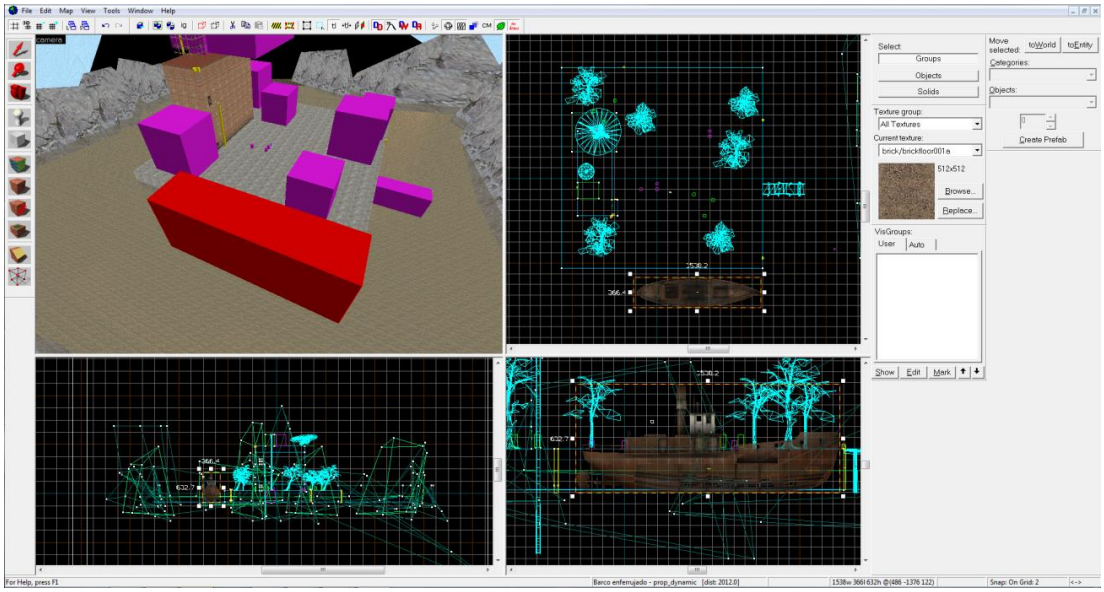


Figura 28 - Hammer: ferramenta de criação de mapas pertencente ao pacote Source SDK

Em primeiro lugar criou-se o terreno do jogo, ou seja, tudo o que é estático (e.g. montanhas, chão, mar, plataformas, blocos) e aplicaram-se as respetivas texturas. Tendo em conta que a

[10] Ver Anexo 1 – Ficheiro de configuração das equipas

água do mar é transparente, teve que haver a preocupação de configurar a textura de forma que se parecesse com a água.

De seguida criaram-se as entidades simples (e.g. árvores, bidons, candeeiros, barcos) que foram distribuídas pelo mapa. Estas entidades podem ser do tipo *static* ou *dynamic* e serão renderizadas de forma independente do terreno do jogo. Assim, os constituintes do terreno de jogo serão transformados em apenas um objeto enquanto as entidades serão transformadas em objetos independentes e com propriedades próprias (nome, posição, material, entre outras propriedades dependendo do tipo de entidade).

Outro passo importante é a iluminação do ambiente virtual porque a partir dela é possível definir o espaço temporal do mapa. Para definir iluminação também se usam entidades, mas em vez de serem do tipo *static* ou *dynamic*, são do tipo *light*. Neste mapa são usadas 4 entidades de *light:* a primeira é uma *light_environment* que vai definir a luminosidade geral do terreno de jogo; as outras 3 fazem parte da iluminação simples de dois holofotes e um candeeiro de cor vermelha.

Outro fator importante que também define o espaço temporal do mapa é o céu – a textura envolvente de todo o mapa. Esta textura pode ser um céu limpo, com algumas nuvens, com estrelas, entre outros. Para fazer o céu do mapa é preciso criar uma *skybox* que não passa de um cubo gigante que envolve o terreno do jogo com uma textura em forma de céu no interior deste.

Para dar um aspeto mais característico de um porto criou-se e configurou-se uma entidade do tipo *fog* para adicionar nevoeiro ao ambiente em desenvolvimento.

Por fim, adicionaram-se 6 entidades do tipo *player* que definem a posição de *spawn* dos jogadores. Três das entidades fazem parte da *Blue team* (equipa Azul) e as outras 3 da *Red team* (equipa Vermelha).

c) Proposta de mapa

Na secção 6.1.4 foi explicado o porquê da escolha de um mapa simples a desenvolver, visto ser a opção mais adequada para o estudo em questão.

Um mapa relativamente pequeno e com poucos objetos é o ideal para pôr em prática os testes que se pretendem efetuar. Esta abordagem de distribuição justifica-se pela pouca quantidade de objetos distribuídos pelo mapa, facilitando a sua identificação, e evitando sobreposições de vários objetos que se poderia tornar conflituoso no momento de identificar o objeto para o qual o jogador estava a olhar.

O tema do mapa passa por uma pequena plataforma situada no meio do mar com um prédio central, sendo possível subir ao topo deste através de uma escada vertical. No topo do prédio encontra-se uma plataforma de aterragem para helicópteros e um depósito de água. Em baixo, na plataforma principal, podemos visualizar algumas árvores e 2 grupos de bidons. Para finalizar, num dos lados da plataforma principal existe uma doca de madeira e um barco perto deste. A Figura 29 representa uma parte do mapa desenvolvido para este trabalho.

Figura 29 - Mapa de jogo do ponto de vista do jogador

Em suma, e enumerando os vários objetos possíveis a serem identificados pelo mapa desenvolvido, verificam-se:

- 1 barco;
- 1 doca em madeira;
- 6 árvores;
- 6 bidons;
- 1 depósito de água;
- 1 plataforma com helicóptero.
- 2 holofotes
- 1 candeeiro de cor vermelha

Em relação às outras especificações do mapa, existem duas áreas de *spawn*: uma para a equipa Azul e outra para a equipa Vermelha, ambas com uma distância considerável entre si para que os jogadores não nasçam (*spawn*) uns ao lado dos outros. Quanto ao estado temporal do ambiente, este passa-se durante uma tarde de nevoeiro e com céu nublado.

6.2. Integração e visualização de dados

Nesta secção, é explicado com pormenor o desenvolvimento da aplicação de análise, no fundo uma aplicação que passamos a designar de gameye, começando pela escolha do *software* de desenvolvimento e respetiva linguagem de programação usada. São apresentadas soluções usadas para a leitura e interpretação dos dados dos *logfiles* assim como algoritmos importantes para tratamento de dados e apresentação de informação. Também é explicada a estrutura e o funcionamento da aplicação de análise, assim como as suas áreas de interação e visualização. Resolveu-se adotar, doravante, a designação "GAMEYE app" para a aplicação porque este instrumento integra estratégias de análise para o contexto jogo, "*game*", utilizando técnicas de visualização intrinsecamente relacionadas com o olho humano, "*eye*", e por fim, porque o objetivo é apoiar a análise da experiência do jogador, "*Experience Analyser".* Sistematizando o acrónimo GAMEYE, traduz *Game Experience Analyser*, subentendido que será de forma visual pelo sufixo "EYE" que ganha visibilidade no acrónimo.

6.2.1. Visualização correlacionada de logfiles

a) Software utilizado: Adobe Flash

Flash é uma ferramenta multimédia originalmente lançada pela Macromedia em 1996 e posteriormente adquirida pela Adobe em 2005. O Flash é usado para adicionar animação, vídeo e interatividade a páginas web. Flash é frequentemente utilizado para anúncios, jogos e animações. Mais recentemente, posicionou-se como uma ferramenta para *Rich Internet Applications* (RIAs). Esta ferramenta permite a exportação de aplicações imersivas e interativas para várias plataformas como desktops e dispositivos múltiplos, incluindo *tablets*, *smartphones* e televisões (Adobe, 2012).

A escolha desta ferramenta de autoria, deve-se ao facto de ser fácil no desenvolvimento de um *layout* complexo e ao mesmo tempo ser eficaz e eficiente na leitura e tratamento de dados externos.

b) Linguagem de programação

A linguagem de programação usada em Flash denomina-se por ActionScript que atualmente se encontra na versão 3.0. É uma linguagem orientada a objetos o que possibilita ao programador usufruir de uma melhor organização do código e reaproveitamento de código (Grossman & Huang, 2012).

Segundo Grossman e Huang (2012) o ActionScript 3.0 (versão usada neste projeto) foi desenvolvido para lidar com os seguintes objetivos:

- **Segurança** - A linguagem suporta um tipo de segurança para os programadores poderem escrever inequívoca e facilmente código sustentável.
- **Simplicidade** - A linguagem é bastante intuitiva para os programadores serem capazes de ler e escrever programas sem consultar um manual de referência constantemente.
- **Performance** - A linguagem permite aos programadores criarem programas complexos que se executam eficientemente e responsavelmente.
- **Compatibilidade** - A linguagem fornece uma compatibilidade entre diferentes versões da mesma. ActionScript 3.0 é um dialeto do ECMAScript que formaliza as características do ActionScript 2.0, adiciona as capacidades do ECMAScript para XML (E4X), e unifica a linguagem inteira numa linguagem coerente.

6.2.2. Algoritmo de leitura e representação do/s jogador/es

Para representar os movimentos físicos de um personagem do jogo no programa em Flash deve-se ler os *logfiles* que contêm os dados relativos aos jogadores. Com esses dados, e a partir de alguns cálculos que serão explicados na alínea c), será possível apresentar visualmente os movimentos e rotas originadas pelos diferentes jogadores que participaram na experiência de jogo.

a) Propriedades do jogador virtual

A apresentação visual 2D dos percursos dos jogadores durante a experiência de jogo depende essencialmente de alguns dados presentes no *logfile* "Jogadores" e no *logfile* "Mapa".

Dados importantes inerentes do *logfile* Jogadores:

- *Timestamp*;
- Posição x;
- Posição y;
- Ângulo x;
- Ângulo y;

O *Timestamp* indica o momento pontual do registo dos dados e será essencial para a sincronização dos vários momentos de registo dos jogadores na *timeline*. A posição x e y na escala do jogo refere-se ao local exato onde o jogador estava no momento de registo. O ângulo x e y indica para que zona o jogador virtual estava virado.

Quanto aos dados pertinentes do *logfile* Mapa:

- world_maxs;
- world_mins;

Estes dados são importantes para o cálculo da dimensão total do mapa. O world_maxs e world_mins são constituídos por 3 coordenadas: x, y e z. Tal como os nomes sugerem, o world_maxs contém as coordenadas máximas do mapa assim como o world_mins contém as coordenadas mínimas.

b) Interpretação de logfiles (parsing)

Antes de se usarem os dados provenientes dos *logfiles* no Flash, é necessário que estes sejam lidos e indexados para a memória ou mais propriamente para objetos. Desta forma será possível aceder às propriedades de cada jogador sempre que necessário.

Como já foi referido na secção 6.1.3, os *logfiles* seguem uma estrutura otimizada para as necessidades desta experiência. No caso do *logfile* "Jogadores", como se pode verificar no Anexo 2 – Logfile "Jogadores", é possível entender a sua estrutura da seguinte forma:

> "data][timestamp][id do jogador|nome do jogador|id da equipa][posição x|posição y|posição z][ângulo x|ângulo y|ângulo z][sentido|movimento lateral][energia|armadura|cançasso|mortes|pontuação|arma actual|se está a correr|se está agachado|se está a saltar|disparos instantâneos|se está morto".

De uma maneira mais simples, pode-se interpretar a referida estrutura da seguinte forma:

> "data][timestamp][info jogador][posição][ângulo][movimentação][propriedades e estados"

O *logfile* "Mapa"[11] é um caso especial porque contém uma estrutura diferente dos outros *logfiles*. Este *logfile* não é orientado a momentos de jogo, mas sim orientado aos vários objetos ou entidades pertencentes ao mapa de jogo. Esta estrutura é gerada automaticamente pelo Source SDK, sendo apenas necessário encontrar no código onde esta estrutura é tratada e gravada num *logfile* recorrendo ao C++. Esta estrutura é semelhante ao formato JSON que é

[11] Ver Anexo 3 – Logfile "Mapa

frequentemente usado para a serialização de dados estruturados e transmissão através de uma conexão em rede (json, 2011).

Após a leitura dos *logfiles* em Flash, é necessário fazer *parsing*[12] de cada linha do documento lido, ou seja, transformar texto em dados. Este processo passa por uma leitura do documento linha a linha, e separação dos vários dados através de um caracter ou um conjunto de caracteres. Por exemplo, no caso do *logfile* "Jogadores", a estrutura contém dois tipos de separações de dados: "][" e "|".

Para fazer esta separação em através de ActionScript, pode-se recorrer à função "split()". Esta função divide um objeto String em substrings segundo o parâmetro delimitador especificado e retorna os substrings num Array para depois ser usado conforme necessário (Adobe, 2010).

A Figura 30 representa parte da função do processo de *parsing* usado para o *logfile* Jogadores:

```
47      var playerTS:playerTimeStamp = new playerTimeStamp();
48
49      for(var i=0;i<myArrayOfLines.length;i++) {
50          if(myArrayOfLines[i]) {
51              globalLine = myArrayOfLines[i].split("][");
52              simulationTime = globalLine[1];
53              playerLogInfo = globalLine[2].split("|");
54              playerLogCoordinates = globalLine[3].split("|");
55              playerLogAngles = globalLine[4].split("|");
56              playerLogMoves = globalLine[5].split("|");
57              playerLogPropreties = globalLine[6].split("|");
58              playerTS = new playerTimeStamp();
59
60              playerTS.setSimulationTime(simulationTime);
61              playerTS.setX(convertGameScaleToFlashX(playerLogCoordinates[0]));
62              playerTS.setY(convertGameScaleToFlashY(playerLogCoordinates[1]));
63              playerTS.setPlayerId(playerLogInfo[0]);
64              playerTS.setPlayerName(playerLogInfo[1]);
65              playerTS.setTeamId(playerLogInfo[2]);
66              playerTS.setAngX(playerLogAngles[0]);
67              playerTS.setAngY(playerLogAngles[1]);
68              playerTS.setForwardMove(playerLogMoves[0]);
69              playerTS.setSideMove(playerLogMoves[1]);
70              playerTS.setHealth(playerLogPropreties[0]);
71              playerTS.setArmor(playerLogPropreties[1]);
72              playerTS.setStamina(playerLogPropreties[2]);
73              playerTS.setDeaths(playerLogPropreties[3]);
74              playerTS.setFrags(playerLogPropreties[4]);
75              playerTS.setWeapon(playerLogPropreties[5]);
76              playerTS.setIsSprinting((playerLogPropreties[6] == 1) ? true : false);
77              playerTS.setIsDucking((playerLogPropreties[7] == 1) ? true : false);
78              playerTS.setIsJumping((playerLogPropreties[8] == 1) ? true : false);
79              playerTS.setInstantShotsFired(playerLogPropreties[9]);
80              playerTS.setIsDead((playerLogPropreties[10] == 1) ? true : false);
81              playerTS.setTime(globalLine[0]);
82
83
84              globalPlayerInfoSimulationTime[i] = simulationTime;
85
```

Figura 30 - Exceto de código em ActionScript 3.0 usado para efeitos de *parsing* do *lofile* Jogadores

Como se pode ver na Figura 30, é criada uma instância da *class* playerTimeStamp. Esta *class* contém uma estrutura para gravar em memória a informação de cada momento do jogador no jogo. Depois do *parsing* ser processado, a informação é registada na instância criada, e posteriormente adicionada a um Array do respetivo jogador. Desta forma, quando a função terminar o seu processo, haverá um Array de instâncias do tipo playerTimeStamp, que

[12] Processo de análise de um texto, feito de uma sequência de símbolos (e.g. palavras), para determinar a sua estrutura gramatical relativamente a uma gramática formal.

representa uma lista de todos os movimentos (posições, ângulos, propriedades, entre outros) do jogador efetuado durante a experiência de jogo.

c) Criação de mini-map

O mini-map é uma representação global do mapa de jogo. Nesta experiência o mini-map encontra-se em apenas duas dimensões, omitindo a representação visual das coordenadas z.

O Source SDK não oferece nenhuma forma de criar o mini-map automaticamente do mapa que foi construído pelo Hammer. A comunidade Steam propõe uma via para criar um mini-map do mapa de jogo desenvolvido, sendo necessário correr o *mod* desenvolvido e o respetivo mapa criado, seguir as instruções de configuração apresentadas por eles e no fim fazer um *Print Screen*. De seguida faz-se os ajustes finais necessários num editor de imagem para que o mini-map fique com as dimensões corretas (ValveCommunity, 2011). A Figura 31 representa o mini-map produzido seguindo as instruções dadas pela comunidade Valve e ajustado às necessidades deste projeto:

Figura 31 - Mini-map produzido e preparado a ser integrado na GAMEYE app

Um dos passos das instruções de configuração efetuados para preparar o mini-map no jogo é o uso do comando “cl_leveloverview x” onde o x é o fator da escala que o mini-map apresenta no momento e esse valor será essencial para os cálculos de transformação da escala do jogo para a escala em Flash. Se na consola do jogo se executar o comando “cl_leveloverview 6”, por exemplo, o mapa de jogo será apresentado no modo de vista ortográfica, exatamente da maneira que se pretende. Neste caso a escala do mini-map em relação ao mundo do jogo é de 6, ou seja, o mini-map tem uma largura e altura 6 vezes menor em relação ao tamanho real no jogo.

d) Cálculo da escala (do jogo) para escala (em Flash)

Os dados provenientes dos *logfiles* "Jogadores" e "Mapa" estão à escala do jogo. No entanto, para que seja possível representar em Flash as posições dos jogadores corretamente, é necessário que haja uma conversão de coordenadas para uma escala compatível com o Flash.

Como a representação visual é feita em 2D (mapa visto de cima), é apenas necessária a conversão das coordenadas x e y.

Depois da leitura do *logfile* "Mapa", deve-se regular os valores máximos e mínimos. Como os valores mínimos podem ser negativos ou diferentes de zero, para facilitar o processo de *rescale*, coloca-se os valores mínimos x e y a zero e compensa-se nos valores máximos. Por exemplo, como se pode verificar no Anexo 3 – Logfile "Mapa", os valores máximos e mínimos referentes ao mapa jogado estão com os seguintes valores:

- **Máximos (x|y|z):** 2592|2880|1236
- **Mínimos (x|y|z):** -2008|-2624|-176

No entanto a ideia é que esses valores fiquem da seguinte forma:

- **Máximos (x|y|z):** 4600|5504|1412
- **Mínimos (x|y|z):** 0|0|0

O passo seguinte passa pela conversão dos valores máximos e mínimos referentes ao mapa de jogo numa escala definida para a app desenvolvida em Flash. Esta escala será a mesma que foi usada para a criação do mini-map a partir do comando "cl_leveloverview x" como foi referido no ponto c). A nova escala em Flash será então calculada a partir desse valor x (fator escala) onde a largura será igual ao "valor máximo x" sobre o "fator escala" e a altura será igual ao "valor máximo y" sobre o "fator escala". Por exemplo, ao considerar os valores máximos já apresentados anteriormente e o fator escala de 6, teríamos uma largura de 4600/6 = 767 pixéis e uma altura de 5504/6 = 917 pixéis.

Depois da regulação dos máximos e mínimos, faz-se a conversão das coordenadas x e y de cada jogador em cada momento. Para isso foram criadas três funções para que as coordenadas transformadas pudessem coincidir com a escala do Flash. A Figura 32 representa as três funções que executam a conversão da escala.

```
262 function convertGameScaleToFlashX(coordinate:int):int {
263     return ((miniMapMaxX*(Number(coordinate)+Number(adjustXCoordsValue)))/gameMapMaxX)
264 }
265
266 function convertGameScaleToFlashY(coordinate:int):int {
267     return (miniMapMaxY*(Number(coordinate)+Number(adjustYCoordsValue)))/gameMapMaxY;
268 }
269
270 function convertGameScaleToFlashZ(coordinate:int):int {
271     return (miniMapMaxZ*(Number(coordinate)+Number(adjustZCoordsValue)))/gameMapMaxZ;
272 }
273
```

Figura 32 - Excerto de código em ActionScript 3.0 referente à conversão de escala

Matematicamente o valor final (x, y ou z) que se pretende encontrar é calculado a partir de uma regra de três simples, uma forma de descobrir um valor a partir de outros três.

$$\frac{\boldsymbol{Coordenada\ x\ final\ (Flash)}}{Coordenada\ x\ original\ (jogo)} = \frac{Dimensão\ máxima\ x\ (Flash)}{Dimensão\ máxima\ x\ (jogo)}$$

$$\boldsymbol{Coordenada\ x\ final\ (Flash)} = \frac{Coordenada\ x\ original\ (jogo)\ \times\ Dimensão\ máxima\ x\ (Flash)}{Dimensão\ máxima\ x\ (jogo)}$$

6.2.3. Estrutura da GAMEYE app

a) Organização por *Layers*

A utilização de diferentes *layers* para diferentes propósitos de código facilita o processo de organização de código. Desta forma evita-se o uso de apenas um documento para listar toda a lógica programada, o que pode dificultar o processo de leitura do mesmo pelo programador.

b) Classes e programação *object-oriented*

Essencialmente, a programação OO (*object oriented*) pode-se resumir a uma forma de olhar para os problemas particulares e dividi-los em partes menores chamadas objetos. Estes objetos formam os blocos de construção de aplicações *object-oriented*, e quando são devidamente desenhados ajudam a partir de uma *framework* sólida (neste caso o Flash) a construir um projeto (Elst, Jacobs, & Yard, 2007).

Ao estudar programação OO, não se pode ignorar as classes e os objetos, como aqueles que são os blocos básicos de construção de qualquer projeto. Há uma ligeira diferença entre uma classe e um objeto. Uma classe é uma descrição autossuficiente para um conjunto de serviços relacionados e dados. As classes listam serviços sem revelar como funcionam internamente. Normalmente, as classes não são capazes de trabalhar por conta própria; estas precisam de instanciar pelo menos um objeto em seguida que é capaz de agir sobre os serviços e dados descritos na classe (Elst et al., 2007).

Para este estudo foram desenvolvidas 5 classes:

- **playerTimeStamp**: a partir desta classe é possível gravar em memória um momento ocorrido durante o jogo por um jogador, isto é, as sua posição, ângulo, propriedades e estados daquele preciso momento registado no *logfile*. Recorrendo ao tipo *Array* será possível registar todos os momentos de um determinado jogador, ou seja, vários objetos da class playerTimeStamp num *Array*. Nesta *class* são registados dados como id do jogador, nome do jogador, equipa, pontuação corrente, mortes, posição corrente (x, y e z), ângulo, movimentação instantânea (se está a andar para a frente/trás e esquerda/direita), tempo, energia, entre outros.
- **playerStatistics –** playerStatistics é uma *class* usada para guardar em memória os resultados finais de jogo relativos a cada jogador. Esses resultados são calculados a partir dos dados tratados na *class* palyerTimeStamp. Pode-se gravar dados como pontuação total, número de mortes, relação entre os pontos e as mortes (*kill/death* ratio), número de saltos, agachamentos, corridas e disparos efetuados.
- **playerEyeTracker -** esta *class* guarda os dados relativos à posição ocular do jogador que tem o *eye tracker* instalado no seu computador. À semelhança da *class* playerTimeStamp, esta também recorre ao *Array* para registar as várias leituras oculares do jogador durante o jogo. Esta *class* tem como principais propriedades:

resolução do monitor (x e y), raio de fixação, *timestamp*, duração da fixação e ponto de fixação x e y.

- **playerEvent -** a class playerEvent tem o papel de permitir a instanciação dos vários eventos realizados durante a experiência de jogo de todos os jogadores, tais como, quando alguém matou um adversário ou se alguém foi atingido por uma bala. Alguns dos dados tratados nesta *class* são: o momento da ocorrencia (*timestamp*), nome do evento e respetiva descrição.
- **Entity -** esta *class* gere os dados relacionados com uma entidade ou objeto pertencente ao mapa de jogo como por exemplo uma árvore ou bidon. São guardados dados como a sua posição em coordenadas x, y e z, nome do objeto, nome do modelo e tamanho do objeto em termos de *boundingbox.*

c) Organização por pastas e execução da GAMEYE app

Para executar a GAMEYE app apenas é necessário correr o ficheiro executável produzido pelo Flash (.swf). De forma a facilitar o processo de desenvolvimento e análise dos *logfiles* produzidos, foi criada uma pasta na mesma diretoria do ficheiro executável, onde foram colocados todos os ficheiros necessários para analisar a experiência de jogo realizada:

- *logfile* jogadores
- *logfile* eventos
- *logfile* mapa
- *logfile eyetracker*
- video da experiência de jogo do jogador com *eye tracker*
- mini-map em formato JPG

Assim sendo, ao haver uma nova experiência a ser analisada, apenas será necessário copiar os novos ficheiros para esta diretoria e executar a aplicação Flash (.swf).

6.2.4. Visualização global correlacionada

A partir desta GAMEYE app, é possível analisar e visualizar situações ocorridas pelos vários jogadores que participam numa sessão de jogo. A Figura 33 representa uma visão global da app de análise e visualização desenvolvida.

Existem seis áreas de interação ou visualização na GAMEYE app: (a) sistema de *timeline*, (b) filtragem, legenda e escalas, (c) dados correntes (propriedades e organização dos jogadores), (d) resultados finais e estatística, (e) vídeo com *eye tracker,* (f) histórico de eventos e (g) zona central de representação e estatística.

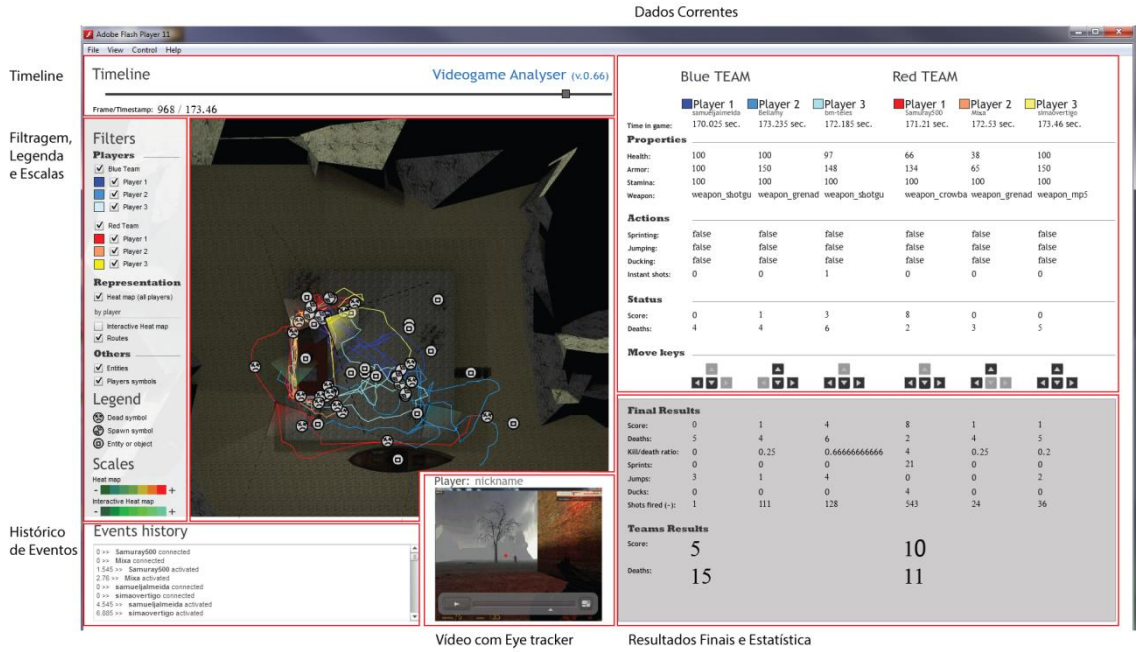


Figura 33 - Representação de um momento de jogo na GAMEYE app

São considerados vários paradigmas de visualização de informação que permitem analisar o cruzamento de diferentes dados em contexto videojogo, nomeadamente correlações (*Bubble Chart*), quantidades contínuas e discretas (*Stacked Area Chart*, *Starplot* e *Isometric Bar Chart*), proporções (*Pie Chart* e *Ring Chart*), fluxos (*Sankey Diagram* e *Thread Arc*), hierarquias (*Tree Diagram* e *Treemap*), redes (*Diagram Map* e *Relation Circle*), configurações espaciais (*Topographic Map* e *Thematic Map*) e considerações (*Heat Map*) (Correia, 2009, pp. 21-31; Costa, 2009, pp. 23-31). Para que fosse possível interagir com a app foi considerado um paradigma de controlo (*timeline*).

Algumas áreas de representação, como a *timeline*, zona central de representação e dados correntes com dados resultados finais e estatística, usam paradigmas de visualização que transformam os dados recolhidos em informação visual para o analista.

Na área de visualização de informação existem diversos paradigmas (Correia, 2009, pp. 21-31; Costa, 2009, pp. 23-31), no entanto, para o desenvolvimento da GAMEYE app apenas alguns dos paradigmas foram considerados:

- ***Timeline*** **-** A interação com este paradigma de navegação pode ser feita através de botões que representam intervalos de tempo ou então através de *sliders* que se movem ao longo da linha de tempo, permitindo selecionar a informação que se pretende ver, com base na evolução cronológica.
- ***Topographic Map*** **-** o mapa topográfico é basicamente uma representação de uma área geográfica com as características naturais inerentes a essa área (espaço, localização, entre outros).
- ***Heat Map*** **-** mapa de calor térmico é um paradigma de representação visual das zonas com maior incidência de valores num determinado contexto. Por vezes é usado para identificar em sites as zonas que têm mais destaque perante os utilizadores.

No entanto, a intenção futura de desenvolvimento passa por complementar a app existente com outros paradigmas de visualização, com vista em enriquecer a informação relativamente ao momento de jogo a ser analisado.

a) Sistema de timeline

Quando existem dados dispostos seguindo uma ordem cronológica, um paradigma bastante usual é a linha temporal (*timeline*). Permite navegar pelos dados consoante a janela temporal pretendida - logo estes dados devem ter uma característica temporal, que vai ser o objeto de interação do utilizador.

Neste caso a "*timeline*" é um controlador que permite ao utilizador da GAMEYE app visualizar o momento de jogo desejado (com base no tempo total de jogo registado).

Esta *timeline* tem o comportamento de um controlo *slider*, onde sempre que o utilizador muda o marcador para outra posição da *timeline*, são apresentados visualmente os dados referentes àquele momento de jogo. Simultaneamente é feito um *search* no vídeo originado pelo *eye tracker* de forma a mostrar o que o jogador estava a ver no ecrã naquele preciso momento. A Figura 34 representa o sistema de *timeline* da GAMEYE app.

Figura 34 - GAMEYE app: sistema de timeline

A *timeline* consiste numa linha que define o início e o fim da sessão de jogo, e por um marcador que indica nessa mesma linha o exato momento a ser representado. Como complemento à informação, também é indicado a *frame* corrente e o respetivo tempo de jogo em segundos após o início da sessão.

b) Filtragem, legenda e escalas

A área de "filtragem, legenda e escalas" permite ao utilizador ativar e desativar os jogadores/equipas que pretende ver na zona de representação da experiência de jogo. Permite também selecionar diferentes tipos de visualização:

- Rotas dos jogadores
- *Heat map* global (escala de cores frias e quentes)
- *Heat map* interativo (escala de cores frias)

Também é possível ativar e identificar as posições dos objetos envolventes no mapa de jogo e dos eventos ocorridos durante o jogo, e.g. mortes e *spawns* dos jogadores. A Figura 35 representa a secção de Filtragem, legenda e escalas da GAMEYE app.

Figura 35 - GAMEYE app: filtragem, legenda e escalas

No lado esquerdo aparecem os ícones ou cores seguidos de uma breve descrição para que durante a análise se possa compreender o que é mostrado sobre o mapa de jogo (mini-map).

No fim da barra pode-se ver duas escalas distintas para representação de *heat map.* A primeira escala contém cores frias e cores quentes, onde as cores frias representam as zonas com menor incidências no mapa de jogo enquanto que as cores quentes representam as zonas com maior incidências por parte dos jogadores. O mesmo acontece para a segunda escala, mas com a diferença de apenas existirem cores frias em tons de verde para azul esverdeado claro. A existência de duas escalas têm uma simples explicação: possibilitar ao analista comparar o *heat map* global (de todos os jogadores) com o *heat map* dos jogadores selecionados na secção de filtragem.

c) Dados correntes (propriedades e organização dos jogadores)

A área "dados correntes" (Figura 36) encontra-se na zona lateral direita da GAMEYE app e está dividida em 6 colunas de informação. Cada coluna representa um jogador e contém as propriedades (e.g. quantidade de vida, quantidade de estamina, arma que está a ser usada); ações (e.g. se o jogador está a correr, agachado ou a saltar); e estado dos jogadores no preciso momento indicado pelo marcador da *timeline*. Estes conceitos também são considerados métricas de jogo.

Para efeitos de prototipagem e demonstração, a app foi desenvolvida para acolher e representar dados de, no máximo, 6 jogadores.

Para facilitar a interpretação, a representação visual e identificação dos jogadores, foi associada uma cor a cada um. Ao mesmo tempo, para que as equipas também se distinguissem facilmente, foram associadas cores frias à equipa Azul (azul escuro, azul ciano e azul claro) e cores quentes à equipa Vermelha (vermelho, laranja e amarelo). A Figura 36 representa a secção da GAMEYE app que contém dados sobre os jogadores dentro do jogo.

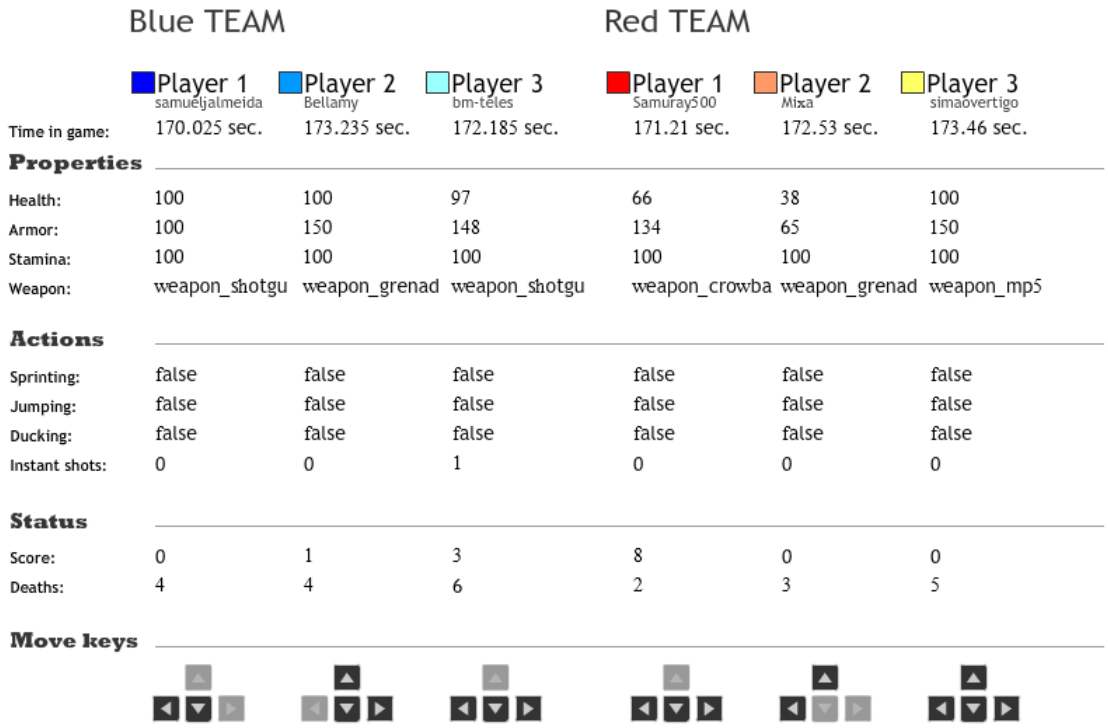


Figura 36 - GAMEYE app: dados correntes

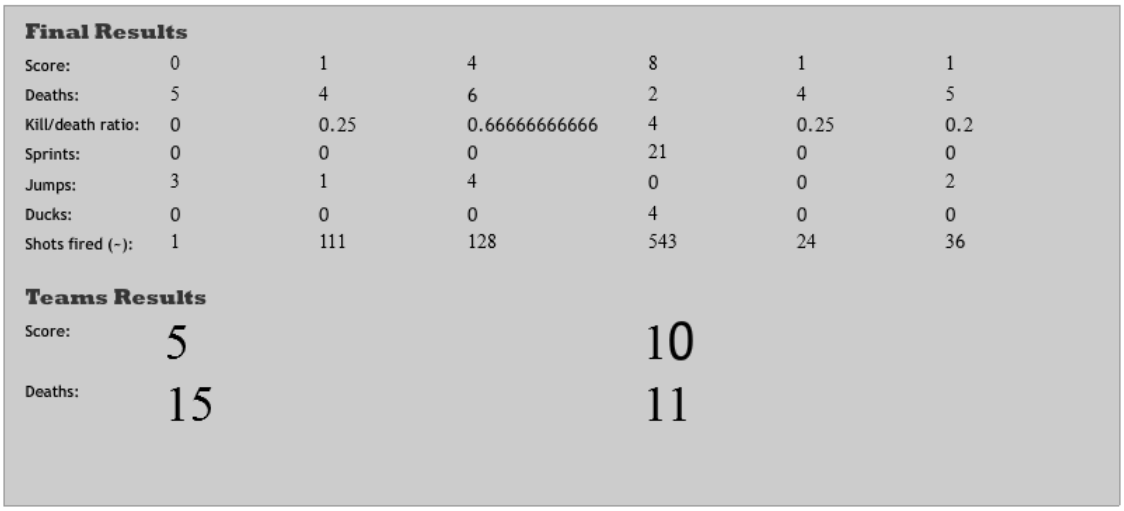
d) Resultados finais e estatística

A área de "Resultados finais e estatística" é visualmente a continuação da área de "Propriedades e organização dos jogadores" visto que ambas as áreas partilha do mesmo cabeçalho onde é possível verificar a que coluna pertence cada jogador e respetiva equipa. Esta área encontra-se imediatamente em baixo da área de "Propriedades e organização dos jogadores". O conjunto das duas áreas acaba por ser a representação de uma tabela de dados dividida em primeiro lugar numa zona relativa à *timeline* e em segundo lugar numa zona onde são representados os dados totais ou finais.

Cada coluna desta área contém a informação que cada jogador tinha no fim da sessão. São apresentados dados como pontuação total, número total de mortes, relação entre a pontuação e as mortes (*points/death ratio*, por exemplo neste caso se o resultado for maior que 1, significa que o jogador teve uma relação positiva entre o número de pontos e as suas mortes) , número total de disparos durante o jogo, entre outros.

Por último, talvez a informação mais pertinente que define qual a equipa que saiu vitoriosa, é o resultado final de cada equipa, ou seja, a soma da pontuação final de cada jogador. Olhando para a Figura 36 e Figura 37 é possível verificar que a equipa Vermelha teve uma maior

pontuação que a equipa Azul. A Figura 37 representa a secção da GAMEYE app com os dados finais de cada jogador e de cada equipa.



Final Results

Score:	0	1	4	8	1	1
Deaths:	5	4	6	2	4	5
Kill/death ratio:	0	0.25	0.66666666666	4	0.25	0.2
Sprints:	0	0	0	21	0	0
Jumps:	3	1	4	0	0	2
Ducks:	0	0	0	4	0	0
Shots fired (-):	1	111	128	543	24	36

Teams Results

Score:	5	10
Deaths:	15	11

Figura 37 - GAMEYE app: resultados finais e estatística

e) Vídeo com eye tracker

Na área de "vídeo com eye tracker" (Figura 38), situada no lado direito dos "eventos ocorridos", pode-se visualizar o vídeo extraído do *eye tracker*. Esta camada de informação adicional está sincronizada com o *timeline* e mostra um vídeo do percurso do jogador, que estava no *eye tracker*, durante a sessão do jogo, bem como – através de um ponto vermelho (o PoR) – o local exato para onde o jogador estava olhar num determinado momento. Para efeitos de demonstração da GAMEYE app, preparou-se a mesma para receber dados de um único *eye tracker*. Assim, apenas a informação "visual" de um jogador é exibida na app. A Figura 38 representa a zona de vídeo com *eye tracker* da GAMEYE app.

Figura 38 - GAMEYE app: vídeo com *eye tracker*

Por cima do vídeo encontra-se a identificação do jogador, que nesta sessão de jogo, utilizou o computador equipado com *eye tracker*. Como apenas existe um vídeo, existe a necessidade de mostrar ao analista qual o jogador que naquele momento estava a utilizar o *eye tracker*.

Também é possível expandir o vídeo para o modo *fullscreen* para que o analista possa ver com mais pormenor os vários pontos importantes inerentes do vídeo.

f) Histórico de eventos

O "histórico de eventos" é outra área da GAMEYE app. Consiste numa área onde encontram-se listados todos os eventos decorridos durante a experiência de jogo. Nesta área podem ser descritos vários eventos: identificação de quando um jogador é morto ou é feito um *respawn* (jogador é introduzido novamente dentro do jogo); quando um jogador dispara ou é atingido, entre outros. De uma forma geral, trata-se de um histórico de jogo, onde o utilizador poderá tirar partido deste para compreender melhor o que aconteceu durante o jogo de uma forma generalizada. A Figura 39 representa um excerto do histórico de eventos decorridos para uma sessão de jogo.

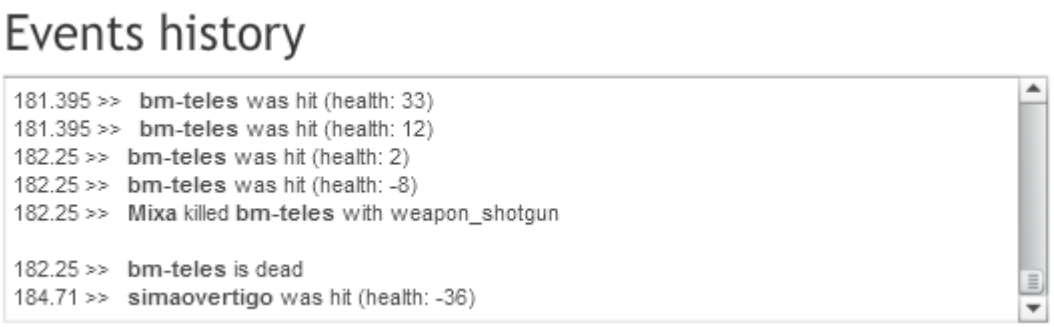


Figura 39 - GAMEYE app: eventos ocorridos

g) Zona central de representação

A área "zona central de representação" contém um mapa de pequena dimensão (mini-map) e em formato de imagem (.JPG) que pode ser alterado conforme o mapa que foi jogado numa determinada sessão. Este mini-map acaba por ser o paradigma de visualização (*Topographic Map*) usado para representar geograficamente os acontecimentos ocorridos na área de jogo. A Figura 40 representa em pormenor a "zona central de representação".

Figura 40 - GAMEYE app: secção principal de representação de jogo

Na Figura 40 verifica-se que os vários jogadores são representados por um "ponto negro" (posição atual no jogo) que é acompanhado por uma representação visual em forma de "V" com um ângulo de 90º (representa o ângulo de visão do jogador dentro do jogo). Este "V" indica o sentido para o qual o jogador virtual (avatar) está virado no jogo e o que é possível ser visualizado por ele. Para facilitar a identificação dos jogadores foi adicionado um gradiente com a mesma cor existente na legenda lateral.

Na Figura 40 também estão representados alguns símbolos que estão dispersos pelo mini-map. Estes símbolos representam situações ocorridas num determinado ponto do mapa e também identificam as posições de origem dos vários objetos existentes no mesmo.

A rota de cada jogador é representada através de uma linha, também ela com a mesma cor do respetivo jogador. Existem dois *heat maps* diferentes: o primeiro (Figura 41) trata-se de um *heat map* global composto por cores frias e quentes que mostra as zonas de menor e maior incidência respetivamente por parte dos jogadores no mapa de jogo. O *heat map* interativo (Figura 42) está representado pelas múltiplas zonas em cor de tons de verde e tem a particularidade dinâmica de mostrar apenas as zonas de incidência dos jogadores selecionados na secção de filtragem onde o verde escuro representa zonas com pouca incidência enquanto que o azul esverdeado claro representa zonas com maior incidência.

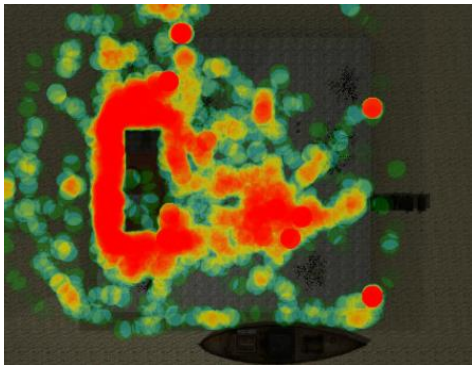


Figura 41 - GAMEYE app: representação do heat map global

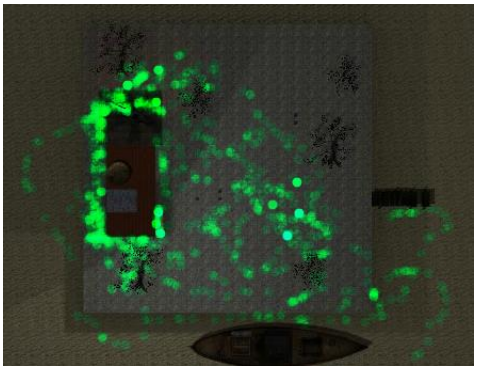


Figura 42 - GAMEYE app: representação do heat map interativo

A representação visual desta área depende necessariamente dos sistemas falados nas duas outras secções anteriores: o sistema de *timeline* e o sistema de filtragem. Através da *timeline* é possível visualizar os vários momentos ocorridos dos jogadores durante a sessão e a partir da filtragem lateral é possível estudar casos específicos, como por exemplo, zonas de maior incidência de um determinado jogador ou de uma equipa.

6.3. Experiência empírica

Para a experiência empírica que se pretendia realizar – de forma a testar a utilidade da aplicação desenvolvida – foram definidos 2 tipos de jogadores distintos. Esta abordagem permitiria visualizar diferentes análises da app desenvolvida, pois diferentes utilizadores têm diferentes formas de agir. Nesta secção também se identificam os vários instrumentos de recolha de dados que utilizados na experiência, assim como a caracterização de cada um. Para finalizar, são referidos alguns aspetos relacionados com a preparação da experiência.

6.3.1. Caracterização da Amostra

A divisão mais comum de jogadores é feita entre jogadores novatos (*newbies* ou jogadores inexperientes) e especialistas (jogadores experientes). Esta classificação é útil principalmente quando se define a dificuldade do jogo e se ajusta a interface para servir jogadores com diferentes níveis de experiência. Outro modelo básico é para os jogadores do grupo *hardcore* e jogadores casuais. Jogadores *hardcore* podem ser descritos como pessoas que jogam como uma preferência de estilo de vida e gastam quantidades substanciais de tempo e dinheiro em jogos. Jogadores casuais podem ser entendidos como um grupo mais diverso. Estes jogam para se divertir ou para "matar o tempo"; estes têm algum conhecimento sobre as convenções de jogo e jogam alguns jogos (O. Sotamaa, 2007). Posto isto, para esta experiência, definiram-se 2 tipos de jogadores distintos de forma a visualizar diferentes análises do *software* desenvolvido:

- **jogador inexperiente –** com pouca ou quase nenhuma experiência de jogo. Estes jogadores têm uma certa dificuldade a adaptarem-se ao ambiente de jogo quando jogam pela primeira vez um determinado mapa. Por vezes, no caso de jogos da categoria *Shooter*, chegam a andar às voltas pelo mapa de jogo sem qualquer objetivo definido.

- **jogador experiente –** um jogador experiente pode ser subdividido em 2 diferentes tipos de jogador:
    - **jogador casual –** jogador que joga videojogos com alguma frequência. Normalmente já tem alguma experiência de jogo e uma interpretação de jogo moderada. Apesar de dedicarem menos tempo ao jogo que os jogadores *hardcore*, não significa que joguem pior que eles.
    - **jogador *hardcore* –** com muita experiência de jogo. Estes jogadores definem-se por terem várias horas diárias dedicadas a videojogos. Por norma, o nível de interpretação de situações de jogo é elevada. No entanto, não significa que estes jogadores sejam melhores que os jogadores casuais. Simplesmente dedicam mais tempo aos videojogos. Normalmente pertencem a clãs de videojogos e participam em torneios.

Para esta experiência foi selecionada uma amostra de conveniência composta por 6 jogadores dado que o objetivo não era avaliar o *mod* desenvolvido, mas experimentar o processo de recolha e análise de dados de forma a provar que se trata de um método útil para a área de análise de videojogos. O processo de seleção passou pela pré-seleção de cerca de 10 pessoas onde se perguntou pessoalmente a cada uma qual a sua qualidade/experiência de jogo. Dentro desta lógica procurou-se selecionar, segundo as respostas destes jogadores, o mais variado tipo de jogadores para ter uma amostra o mais heterogénea possível a nível de experiência de jogo.

6.3.2. Instrumentos de recolha de dados

O procedimento de recolha de dados foi realizado através da elaboração de um estudo de caso simples, onde os participantes podem jogar um videojogo (*mod* desenvolvido), onde são captados os movimentos oculares do jogador através do *eye tracker* (dados escritos num *logfile* e gravação do ecrã durante o momento de jogo) e, simultaneamente, o *mod* desenvolvido em Source SDK regista noutro *logfile* vários dados relativos às ações e eventos desempenhados durante o momento de jogo. Outro instrumento utilizado para recolha de dados foi um questionário (Shneiderman, 1998, pp. 132-135) antes da primeira sessão. Com o preenchimento destes questionários, foi possível obter informação quantitativa e qualitativa sobre os pontos de vista dos jogadores e quanto à experiência de jogo de cada um.

a) Questionário inicial

Foi desenvolvido um questionário[13] com vista a adquirir informação relativa à experiência de jogo de cada participante. Este questionário foi submetido aos participantes antes de ser iniciada a primeira sessão de jogo. O questionário é composto por 7 perguntas onde 3 delas são perguntas de categoria pessoal e 4 estão relacionadas com hábitos e experiência de jogo.

A primeira questão está relacionada com o ID do participante (*nickname* de jogador) que é igual ao ID usado dentro do jogo. Desta forma, durante a análise de dados, será possível poder associar os dados recolhidos pelo questionário aos dados recolhidos pelo *Mod* desenvolvido. A segunda e terceira questão, idade e sexo respetivamente, servem apenas para complementar

[13] Ver Anexo 4 – Questionário usado no estudo empírico

a informação recolhida de cada jogador. Estas questões poderão vir a ser úteis, por exemplo, para questões de estatísticas relacionadas com fachas etárias e a sua experiência de jogo.

A partir da quarta questão é possível recolher informação pertinente para se poder definir o tipo de jogador (jogador inexperiente, jogador casual, jogador *hardcore*). Nesta questão é perguntado ao participante qual o número de horas, em média, por semana que dedica aos videojogos. Um dos fatores que distingue um jogador casual de um jogador *hardcore*, segundo O. Sotamaa (2007), é o número de horas dispendidas a jogar videojogos. Quanto à quinta pergunta ("Já pertenceu a um clã de videojogos do género Shooter? ") existem 3 respostas possíveis: "sim, mas apenas com amigos"; "sim, em competições"; "não". Este fator é importante para saber se o participante já tem alguma experiência em competições (o que exige um determinado tipo de *skills* relacionadas com trabalho em equipa, prever comportamentos da equipa adversária e dos seus companheiros), se apenas jogou com um grupo de amigos (com uma componente menos aperfeiçoada de trabalho de equipa) ou se simplesmente não tem qualquer experiência em jogar em equipa.

A sexta questão inquira sobre o tipo de plataformas de jogo a que o participante está mais habituado a usar. Como neste estudo pretende-se fazer uma experiência na plataforma PC, é importante saber se os participantes têm alguma experiência relacionada com videojogos para computador. A maneira de interagir com o jogo difere de plataforma para plataforma, ou seja, é diferente jogar um *First-Person Shooter* com um comando (consola), do que jogar com um teclado e um rato (computador).

Finalmente, e com um objetivo semelhante à questão anterior, a questão 7 procura recolher informação acerca das categorias de videojogos que os participantes mais jogam. A partir desta questão é possível entender se os participantes têm alguma experiência na categoria de jogo presente na experiência a efetuar (*First-Person Shooter*).

b) Eye Tracker

O *eye tracker* (Tobii Eye Tracker) e o correspondente programa Tobii Studio, fazem parte do segundo instrumento de recolha de dados usado neste estudo. Enquanto que os questionários forneceram informações relacionadas com os participantes e sua experiência de jogo, o *eye tracker* fornece informação visual concreta relacionada com os movimentos oculares de cada participante durante as 6 sessões de jogo que foram realizadas (cada jogador foi submetido ao uso do computador equipado com *eye tracker* apenas uma vez, um jogador diferente por sessão). A Figura 43 representa o Tobii T120, o *eye Tracker* utilizado neste estudo.

Figura 43 - Tobii T120 Eye Tracker [retirado de http://www.tobii.com]

A configuração de um novo projeto usando o *software* da Tobii é de natureza simples. O primeiro passo consiste em criar um novo projeto, ao qual é atribuído um nome e autor (se desejado). Em seguida, dependendo do número e tipo de tese que se pretende fazer, deve-se atribuir e criar subprojetos independentes. Assim, para exemplificar, um nome para um novo projeto com dois testes poderiam ser "Estudo de Usabilidade da Web" e o nome dos testes poderiam ser "página inicial do Google" e "página inicial do Sapo".

Quando essas etapas forem concluídas, o utilizador está no ambiente de *software* onde pode executar os testes e manipular dados através do uso das opções presentes nos três painéis do software.

O programa do *eye tracker* suporta a gravação de vários elementos multimédia e meios de comunicação, bem como a combinação de vários destes meios. A configuração dos elementos multimédia é feito no painel de estrutura e registro. A Figura 44 representa um ensaio com um único elemento multimédia - uma gravação de ecrã.

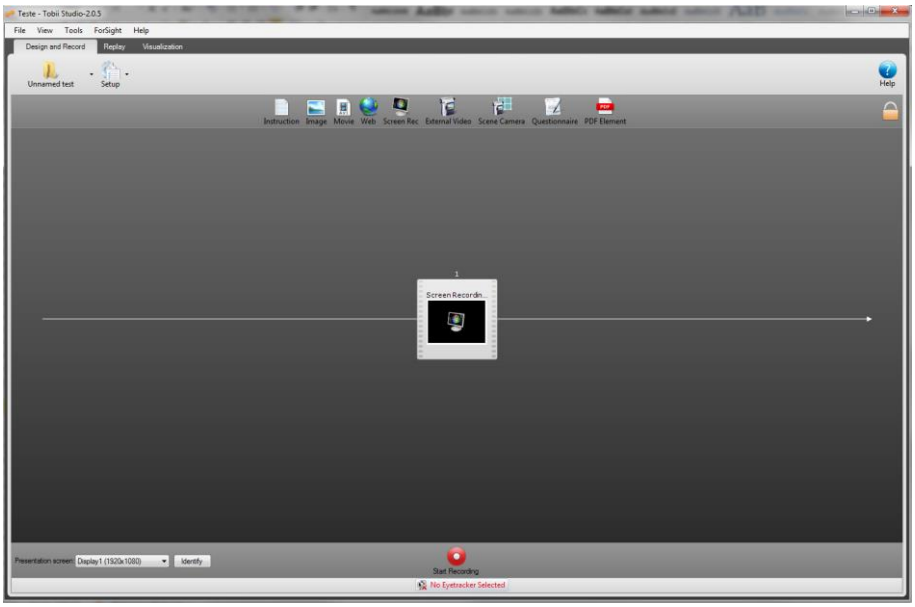


Figura 44 - Tobii Studio Software: Painel de *Design and Record*

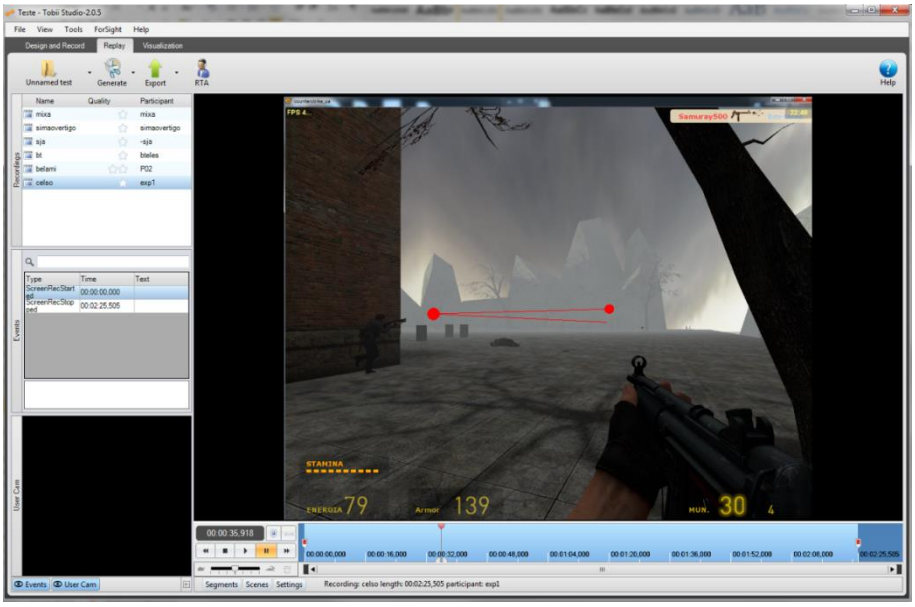


Figura 45 - Tobii Studio Software: Painel de *Replay*

Após o teste ter sido completado, os dados visuais podem ser vistos e analisados por meio do uso do painel de *replay* e de *visualization*. O painel de *replay* (Figura 45) mostra uma lista das diferentes gravações associadas ao teste em questão (à esquerda). Para cada gravação, há um vídeo, que pode ser visto no centro e controlado utilizando os botões perto da *timeline*. Aqui, também é possível criar marcadores para registar e definir os pontos de interesse. Para alguns media, é necessário segmentar o vídeo o que também é feito no painel de *replay*. Além disso, imagens e vídeos também podem ser exportados para análise exterior através do uso da opção Exportar também encontrado neste painel.

Para este estudo a funcionalidade mais importante é a de exportação de vídeo e do *logfile* de cada sessão para posteriormente serem tratados e integrados na GAMEYE app.

A exportação de cada vídeo (gravação do ecrã do momento de jogo de um jogador) é feita para o formato ".avi". Porém, em alguns vídeos gravados existem 2 partes que não se referem à experiência no jogo e que devem ser cortadas do vídeo final. Esta edição pode ser feita no Tobii Studio antes da exportação ou mesmo depois da exportação usando um programa de edição de vídeo como o Adobe Premiere como é apresentado na Figura 46.

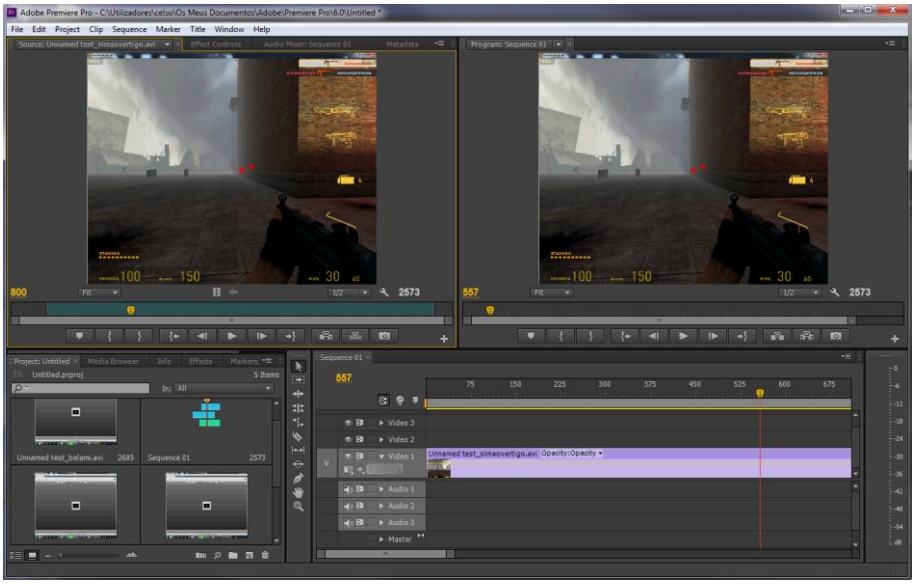


Figura 46 - Adobe Premiere Pro: Edição de vídeo

Como a app desenvolvida para análise é realizada em Adobe Flash, esta requer que o vídeo seja de formato compatível (e.g. f4v). Desta forma é necessário que após a edição/exportação, o vídeo seja convertido do formato ".avi" para o formato ".f4v" usando um programa de conversão como por exemplo o Adobe Media Encoder. A Figura 47 mostra o decorrer da conversão de um vídeo em formato ".avi" para formato ".f4v".

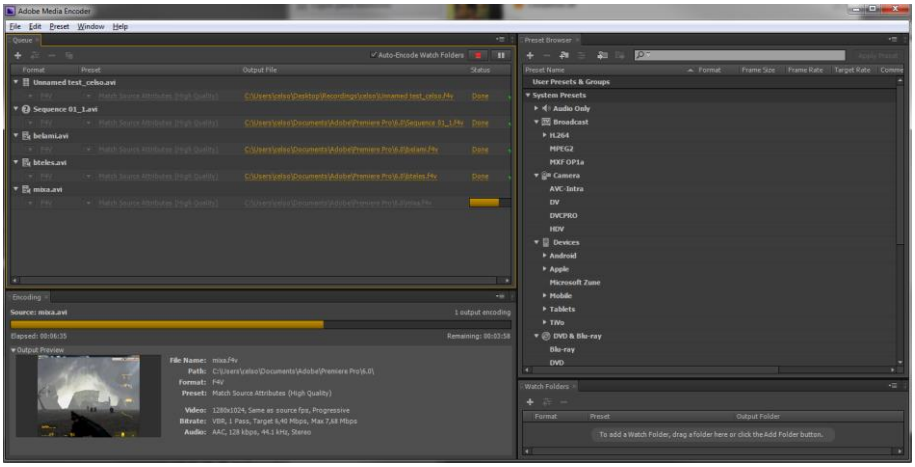


Figura 47 - Adobe Media Encoder: Conversão de vídeo

Após a conversão de todos os vídeos, estes estão aptos para serem integrados e interpretados pela GAMEYE app.

Estes vídeos contêm informação pertinente para análise. O círculo vermelho (*point of regard*) que aparece na Figura 48 representa a localização para onde o jogador estava a olhar num determinado momento. Para além disso, o tamanho destes círculos acrescentam outra variável de informação (tempo), duração em que o jogador fixou aquele ponto. A partir destes dados visuais é possível, por exemplo, identificar os elementos de maior destaque durante o jogo. Na Figura 48 é apresentado um momento de jogo onde o jogador estava a olhar para o tronco de uma árvore.

Figura 48 – Representação do jogador com foco no tronco da árvore

c) *Mod* em Source

Como já foi referido na secção 6.1.3, foi criado um *mod* com base no motor Source. A necessidade de modificar um *template* base oferecido pela Valve Software passa pelo facto de ser possível (usando a linguagem C++) registar em 3 *logfiles* os dados referentes ao mapa de jogo, às propriedades/estados (métricas) e eventos dos jogadores durante a experiência de jogo: *logfile* mapa, *logifle* jogadores e *logfile* eventos.

Este registo (*logging*) da experiência de jogo de cada jogador é um dos principais passos até chegar à GAMEYE app, que terá que interpretar cada *logfile* produzido pelo *mod* para depois apresentar resultados visualmente e prontos a serem analisados.

6.3.3. Preparação da experiência

Antes de efetuar a experiência final, foi realizada uma experiência preliminar para efetuar alguns testes de modo a garantir que a experiência final iria correr como pretendido.

a) Teste com jogadores reais

Em primeiro lugar, era importante ter a certeza que o *mod* desenvolvido estava a registar os dados de forma correta. Como a fase de desenvolvimento foi toda à base de testes com bots (jogadores controlados pelo computador com inteligência artificial), não havia certezas que a experiência iria correr da mesma forma com jogadores reais, ligados a uma rede local. Desta forma, criou-se uma rede para ligar dois computadores com a versão *Release* (versão de instalação) do *mod* instalada. Um dos computadores criou o servidor de jogo para que o outro se pudesse ligar e ambos jogarem juntos num ambiente de jogo partilhado (modo *multi-jogador*). Após algum tempo de experiência de jogo, já é possível verificar se os *logfiles* (a serem registados no computado/servidor) estavam a ser registados conforme acontecia com *bots*.

b) Os participantes

Após ter sido ajustado com os participantes a agenda para a realização da experiência foi necessário preparar os computadores de cada participante envolvidos para o dia da experiência. Nos dias anteriores à data marcada distribuiu-se a versão *Release* do *mod* pelos participantes e instalou-se o Steam e o respetivo *mod*. Também foi pedido aos participantes que ainda não tinham conta de Steam para que criassem uma conta, pois esta era uma das exigências do *mod*.

c) Teste na rede UA com o Steam e com o jogo

Para que o participante possa se conectar no Steam com a sua conta precisa de estar ligado a uma rede com ligação à internet. No entanto, como a experiência ia ter lugar na Universidade de Aveiro, foi necessário fazer um teste para verificar se o processo de login passava na rede da universidade. A suspeita de não ser possível fazer login na rede da universidade partiu de uma referência[14] da equipa de suporte da Steam sobre este assunto.

"Many university networks and proxies block required ports for Steam operation - please consult your network administrator to ensure the required ports are open if you are using a university network or a proxy. Ports required for Steam can not be re-mapped to HTTP or reconfigured to a custom port range."

A equipa de suporte da Steam refere que em algumas universidades os portos que o Steam usa estão bloqueados, o que não permite ao utilizador se conectar estando ligado a uma rede dessas universidades incluindo a Universidade de Aveiro (após uma tentativa falhada).

Para resolver esta situação surgiram 3 possíveis soluções:

- **Ter os computadores em modo de suspensão com o Steam já ligado –** Pedir aos participantes para trazerem já de casa o computador em modo de suspensão com a autenticação no Steam já feita. Visto que para esta experiência apenas é necessário o Steam estar ligado para depois correr um jogo em LAN, já não seria necessário usar a internet da universidade.
- **Dar permissão, na rede da Universidade de Aveiro, aos computadores que iriam estar envolvidos na experiência –** Este processo exige a recolha do *MAC address* dos computadores de cada participante para poderem ser fornecidos a um técnico responsável pela rede da universidade. O técnico, com os *MAC Addresses,* podia dar permissões apenas aos respetivos computadores, não comprometendo a segurança global da rede. Desta maneira, os computadores dos participantes, estando ligados à rede da universidade já se podiam autenticar na plataforma da Steam.
- **Criar uma rede WiFi "ad hoc" com internet de uma pen –** Se possuir dois computadores ou mais equipados com adaptadores sem fios (placas WiFi), é possível ligá-los em rede, criando uma rede "ad hoc", quer significa uma rede de igual para igual, sem estar a utilizar ponto de acesso. Se um dos computadores da rede "ad hoc" possuir uma conexão à Internet, é então possível partilhá-la com os outros computadores da rede, como no caso de uma rede local tradicional. Esta rede com

[14] https://support.steampowered.com/kb_article.php?ref=8571-GLVN-8711

partilha de Internet serve apenas para que os participantes se possam autenticar pela primeira vez na Steam.

Decidiu-se apostar na última solução, sendo a mais viável, primeiro porque nem todos os participantes podiam lembrar-se de trazer os computadores em modo de suspensão com o Steam ligado de casa. Em segundo lugar, não haviam garantias por parte do técnico da universidade que a abertura das portas necessárias para correr o Steam ia funcionar. Testou-se a terceira opção com êxito e concluiu-se que esta seria a solução a seguir.

Para ligar os computadores de cada participante em rede para jogarem o mod desenvolvido, foi necessário criar uma outra rede (LAN) onde foi possível ligar os computadores a através de um ponto de acesso (*wireless switch*). O motivo da criação de uma segunda rede deve-se pelo facto de não ser possível ligar ao servidor de jogo estando ligados à rede "ad hoc". O mesmo aconteceu com um teste efetuado com a rede da universidade, que também não permite a ligação a servidores de jogos dentro da rede.

d) Computador com Eye tracker

O computador com *eye tracker* tem dois papéis importantes: ser o servidor de jogo onde vão ser registados os *logfiles* relativos aos acontecimentos de jogo e correr o Tobii Studio que vai estar a registar em *logfiles* os pontos de fixação do jogador no ecrã e a gravar um vídeo. Este processo complexo exige um processamento elevado por parte do computador. Este problema pode ser resolvido usando dois computadores tal como Almeida (2009) sugere: um para correr o servidor de jogo e transmitir o sinal, e o outro para receber o sinal e correr o *eye tracker.* Esta solução pode resolver grande parte do problema relacionado com o processamento, no entanto surgem outros pequenos problemas como atraso no envio do sinal entre os dois computadores, a não garantia de que os *logfiles* do jogo e os *logfiles* do *eye tracker* estejam sincronizados e o facto de ser necessário o uso de outros equipamentos para converterem/transmitirem o sinal entre os computadores.

A solução ideal para o problema de processamento foi o uso de um computador com elevada capacidade de processamento capaz de suportar as exigências desta experiência, deixando de parte a primeira solução de partilha do processamento entre dois computadores. Para que o computador ficasse totalmente apto para ser usado na experiência, foi instalado o Steam e a versão *Release* do *mod* desenvolvido e o Tobii Studio para poder correr o *eye tracker*.

6.3.4. Caraterização do estudo, contexto presencial e setup da sala

O estudo empírico consistiu num conjunto de participantes a jogar um *mod* baseado no motor Source (categoria *First-Person Shooter*). O estudo foi dividido por 6 rondas de jogo com uma duração aproximada de 3 minutos cada. Trata-se de um *mod* do tipo TDM (*Team Death Match*) onde o objetivo principal de jogo é iliminar o maior número possível de jogadores da equipa adversária. É possível escolher entre ser da equipa Azul ou da equipa Vermelha, assim como escolher uma das três classes apresentadas aos jogadores (a primeira o jogador tem apenas uma pistola e um ferro, mas em compensação movimenta-se mais rápido; a segunda é composta por uma arma automática, uma caçadeira e uma granada, o jogador movimenta-se a uma velocidade moderada; finalmente a terceira classe o jogador tem á sua disposição um ferro, uma arma automática, uma caçadeira e duas granadas, mas em contrapartida

movimentar-se-á mais lentamente em relação às outras classes referidas). Os jogadores podem fazer *spawn* as vezes que quiserem na mesma ronda e sempre que eliminam um inimigo ganham um ponto. Ganha a equipa que juntar maior número de pontos, ou seja, a equipa que eliminou um maior número de inimigos. O ambiente de jogo passa-se num porto de embarcações rodeado por água (mapa desenvolvido para a experiência) como se pode ver na Figura 49.

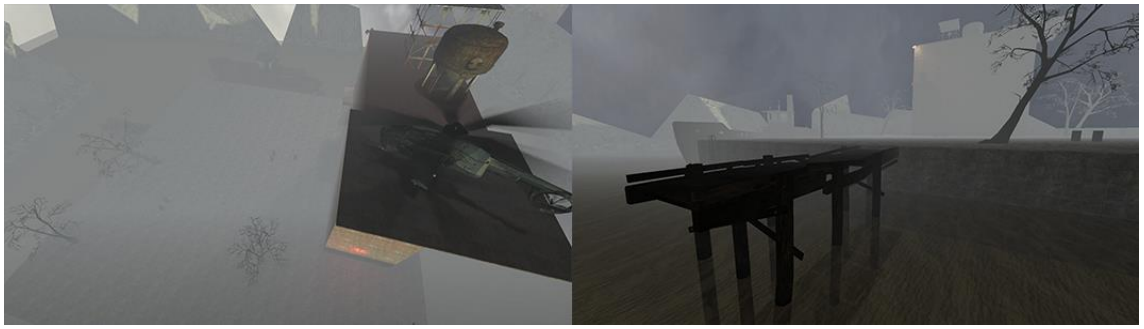


Figura 49 - Mod Source: representação do mapa de jogo desenvolvido para a experiência

Ao entrar na sala destinada para a experiência, foi solicitado cada participante que escolhesse um lugar para se sentar em redor da mesa situada no centro da sala (Figura 50). Nessa mesa os participantes puderam colocar os seus portáteis e acomodarem-se de forma a ficarem confortáveis para a experiência de jogo. De seguida foi explicado de uma forma resumida o conceito geral do estudo em questão e uma explicação básica sobre os comandos do jogo para quem nunca tinha experienciado um jogo do género (*First-Person Shooter*). Dos 6 participantes apenas um não tinha conhecimento de quais as teclas a usar para se movimentar no jogo. Todos os outros já tinha experiência suficiente o que acabou por poupar tempo nas explicações.

Após os participantes já estarem enquadrados com o tema de estudo, pediu-se que se ligassem à "ede 1", uma rede "ad hoc" por WiFi com partilha de Internet através de uma *pen* de banda larga móvel. Esta rede estava a ser produzida por um portátil destinado para esse efeito como se pode ver na Figura 50. Os computadores dos participantes, incluindo o servidor com *eye tracker* (torre), têm placa *wireless*, o que permite ligar os computadores evitando o uso de cabos de rede.

Antes que os participantes se começassem a preparar para iniciar o Steam e correr o *mod*, foi lhes submetido um questionário online com o objetivo principal de averiguar qual a experiência de cada um em relação com os videojogos da categoria *Shooter.* Depois de estes já terem terminado de preencher o questionário foi lhes pedido que se autenticassem na plataforma Steam conforme previsto na alínea (c) da secção 6.3.3.

A partir do momento em que já têm o Steam a correr, a "rede 1" já não é necessária e os participantes podem começar a ligar-se à "rede 2" que serve exclusivamente para jogar em rede. Esta "rede 2" é produzida por um ponto de acesso WiFi (*wireless switch*) (Figura 50).

Na Figura 50 também pode-se identificar o servidor de jogo com *eye tracker* que se encontram ligeiramente separados da mesa central onde estiveram 5 participantes. Na Figura 50 é apresentada a disposição pela sala dos vários elementos físicos inerentes da experiência.

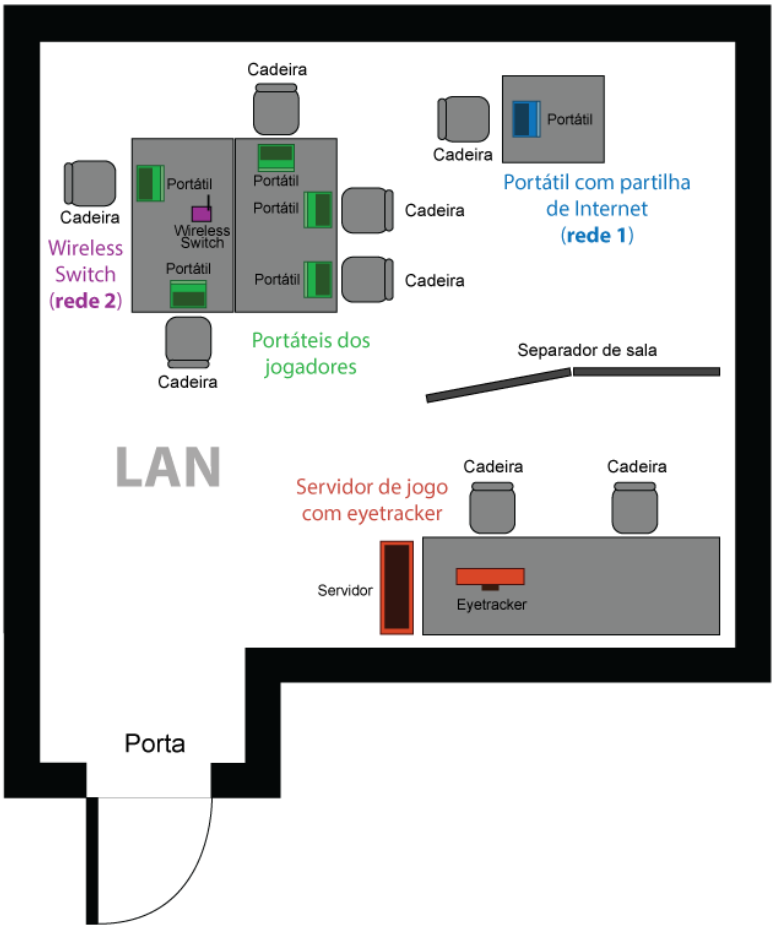


Figura 50 - Setup da sala para o estudo empírico

Outro aspeto importante foi a definição de equipas entre os participantes. Escolher quem ia pertencer á equipa Azul e que ia pertencer à equipa Vermelha. Com o fim de se tornar uma experiência mais competitiva, decidiu-se equilibrar o melhor possível as equipas seguindo as características de cada um e experiência de jogo.

Uma vez que os participantes já estão autenticados no Steam, ligados à "rede 2" e com equipas definidas já é possível iniciar o *mod* para jogar. No entanto, o participante que estiver a jogar no servidor de jogo com *eye tracker* precisa de se preocupar com duas coisas: (i) iniciar o *mod* de jogo através do Steam e criar um servidor de jogo para que os outros jogadores possam entrar no jogo; e (ii) criar um novo teste no Tobii Studio com o seu nome para poder prosseguir (este processo foi acompanhado por um investigador com conhecimentos da ferramenta). Assim que iniciou a gravação com *eye tracker*, o participante teve que se colocar na "posição de jogo", numa posição confortável de modo a que não se mova constantemente na sua cadeira e o *eye tracker* não consiga captar corretamente os movimentos dos olhos.

Seguidamente, a sua posição dos olhos foi calibrada pelo *eye tracker* e o participante pôde escolher a sua equipa e classe para iniciar a primeira sessão de jogo.

Antes do participante iniciar a gravação com o *eye tracker* é preciso tomar em atenção se os outros participantes já entraram no servidor de jogo, porque é importante que todos comecem a jogar ao mesmo tempo.

Passados cerca de 3 minutos de jogo, o investigador junto do participante com o servidor definia a ronda como terminada e através de uma opção do *mod* desenvolvido desligava o jogo. De seguida foi a vez de outro participante ir para o servidor e submeter-se a uma experiência com o *eye tracker*. Repetiu-se o processo de calibragem com o novo participante e iniciou-se a segunda ronda e assim sucessivamente até chegar ao sexto participante e sexta ronda. Após todos os participantes terem rodado pelo servidor deu-se a experiência de jogo como terminada.

Após a experiência de jogo ter acabado procurou-se saber informalmente o que os participantes acharam da experiência e se sentiram algum tipo de transtorno por estarem a usar o *eye tracker*.

7. Apresentação, análise e discussão dos resultados da experiência empírica

Tendo explorado os instrumentos utilizados para coletar dados para o presente estudo de caso, as secções seguintes vão agora abordar os resultados obtidos através desses instrumentos do ponto de vista analítico.

7.1. Apresentação e análise do questionário

Na secção 6.3.1 são apresentados 2 tipos de jogador: jogador inexperiente e o jogador experiente onde um jogador experiente pode ainda ser definido como jogador casual ou jogador *hardcore*. Segundo O. Sotamaa (2007) o número de horas dedicadas a videojogos é extremamente relevante para a definição da experiência de jogo de um determinado jogador. Tendo isto em conta, na Tabela 4 são apresentados alguns dados obtidos a partir dos questionários[15] onde através deles foi possível definir o tipo de jogador de cada participante na experiência.

ID	Plataformas de jogos que mais utiliza	Géneros de jogo que costuma jogar	Tempo em média que joga videojogos durante a semana	Se já pertenceu a um clã de videojogos do género Shooter	Tipo de jogador
J1	PlayStation,**PC,** Online	**Shooter,** Desporto	1-5 horas	**Sim.** Mas só como grupo de amigos	Exp. **casual**
J2	Xbox/Kinect, **PC,** Telemóvel, Android	**Shooter,** Desporto, Corridas/Carros, Plataformas	1-5 horas	**Sim.** Com participação em competições	Exp. **casual**
J3	Xbox/Kinect, **PC,** Online	**Shooter,** Role-playing game, Aventura, Desporto Corridas/Carros	**> 10 horas**	**Sim.** Com participação em competições	Exp. ***hardcore***
J4	**PC**	**Shooter,** Ação, Desporto Corridas/Carros	< 1 hora	Não	Inexp.
J5	mobile	**Shooter,** Real-time strategy, Corridas/Carros	< 1 hora	Não	Inexp.
J6	Xbox/Kinect, **PC,** Online	**Shooter,** Role-playing game, Ação, Aventura, Corridas/Carros, Plataformas	< 1 hora	Não	Inexp.

Tabela 4 - Resultados obtidos pelo questionário e definição de tipo de jogador

[15] Ver Anexo 4 – Questionário usado no estudo empírico

O número de horas foi o fator mais relevante para a definição de cada tipo de jogador, contudo, não é o único fator importante para definir a experiência de jogo dos participantes. As plataformas de jogo que o participante mais utiliza pode ser relevante a nível de familiarização com os controlos usados na plataforma em questão. Algo semelhante acontece com os géneros de jogo que o participante costuma usar. Na Tabela 4 é possível verificar que todos os participantes já tiveram experiências em jogos da categoria *Shooter* o que significa que não tiveram qualquer dificuldade a aprender a jogar o estilo de jogo do *mod* desenvolvido. Finalmente, saber se um jogador já alguma vez pertenceu a um clã de videojogos, é de extrema importância porque indica que o participante já tem alguma experiência em relação ao trabalho de equipa.

Concluindo, neste estudo de caso participaram 3 jogadores inexperientes mas com alguma familiarização em relação a videojogos da categoria *Shooter*. Os outros 3 jogadores são considerados jogadores experientes, onde 1 é um jogador *hardcore* e 2 são jogadores casuais.

7.2. Discussão dos resultados dos dados obtidos do protótipo a partir da experiência

7.2.1. Componentes gerais

Para efeitos de constituir equipas equilibradas, a equipa Azul foi constituída por dois jogadores casuais e um inexperiente. A equipa Vermelha foi constituída por 1 jogador *hardcore* e 2 jogador inexperientes. A equipa Azul fazia *spawn* numa zona aberta e ampla, enquanto que a equipa Vermelha fazia *spawn* ao lado de um prédio um pouco mais protegido.

7.2.2. Comportamentos dos jogadores

A partir da Figura 51 é possível ver que os jogadores considerados experientes (azul escuro, azul ciano e cor-de-laranja) tinham um comportamento mais tático e estratégico. Estes contornaram a zona de *spawn* do inimigo enquanto que os ditos jogadores inexperientes apenas se limitaram a usar uma pequena zona do mapa de jogo. Também é possível verificar que os jogadores experientes raramente repetiam os mesmos trajetos apanhando o adversário desprevenido.

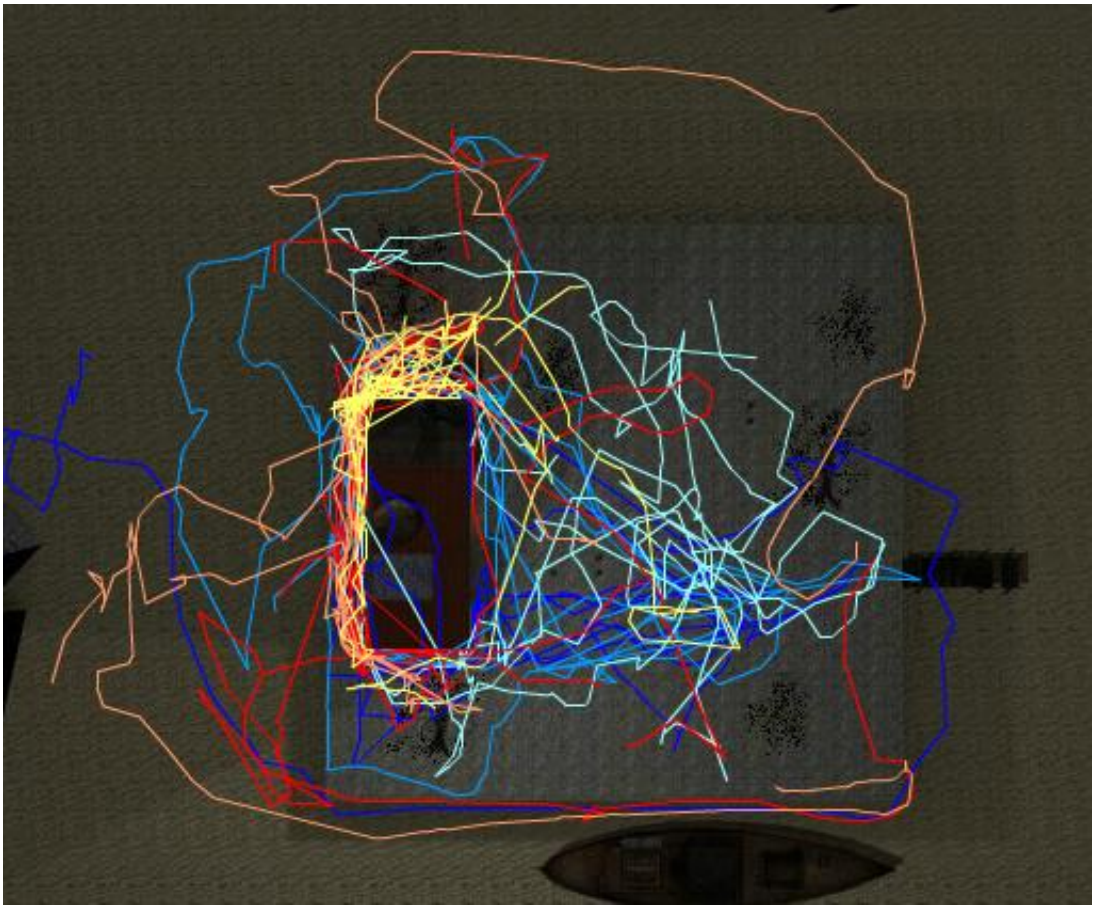


Figura 51 - Representação dos trajetos dos jogadores numa experiência de jogo

Como se pode ver na Figura 51, nota-se que os trajetos dos jogadores experientes são mais lineares e com poucas mudanças de direção enquanto os trajetos dos jogadores inexperientes são mais confusos e com mudanças mais frequentes. Estes trajetos sugerem que os jogadores experientes têm um comportamento em que sabem o que estão a fazer no jogo enquanto os inexperientes fazem um percurso menos preciso e mais aleatório.

No caso particular do jogador inexperiente de cor amarela (Figura 51), repara-se que este apenas usou uma pequena área do mapa, como é possível confirmar a partir da Figura 52. A pontuação deste jogador nesta ronda foi uma das pontuações mais baixas com 4 pontos[16] e também foi o jogador que morreu mais vezes durante a mesma ronda (19 vezes). Na Figura 53 destacam-se as zonas onde o jogador morreu, e de certa forma dá para compreender que o jogador raramente conseguia sair da zona de *spawn* da sua equipa.

[16] Os pontos de um jogador são compostos pelo número total de vezes que o jogador eliminou um adversário. Caso o jogador se tenha suicidado durante o jogo (e.g. morrer com a própria granada) é retirado um ponto ao total existente.

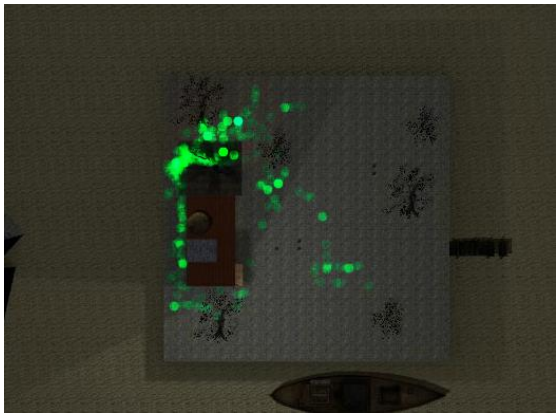


Figura 52 - Heatmap de um jogador inexperiente

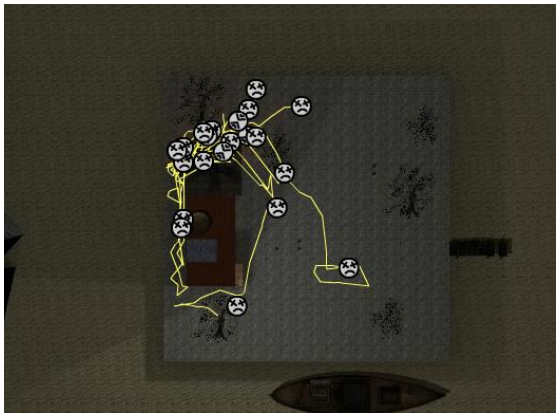


Figura 53 - Localizações onde um jogador inexperiente morreu

Outra observação pertinente, acontece quando o jogador amarelo consegue sair da zona de *spawn* pelo facto dos dois jogadores experientes da equipa adversária estarem ocupados noutra parte do mapa, como é possível verificar na Figura 54. Se for feita uma comparação entre a Figura 53 e a Figura 54, que por sua vez representam o mesmo momento de jogo, verifica-se que esta foi a única vez em que este jogador conseguiu sair da zona de *spawn*.

Figura 54 - Representação de um momento de jogo onde o jogador inexperiente consegue sair da zona de *spawn*

Este caso particular do jogador de cor amarela (pertencente à equipa Vermelha) demonstra os comportamentos típicos de um jogador inexperiente, tal como já foi referido acima. Os jogadores inexperientes, principalmente quando têm adversários com nível de experiência superior a eles (neste caso 2 jogadores com mais experiência), tendem a não conseguir mostrar resultados positivos.

Agora (ainda na mesma ronda) analisando um jogador experiente, a azul ciano (J1), é possível verificar que, este em comparação com o jogador amarelo, atinge uma maior percentagem do espaço disponível no mapa. Este jogador experiente chega a contornar várias vezes a zona de *spawn* do adversário fraguando[17] em média 2 a 3 inimigos por cada vez que é morto. A partir da secção de “Histórico de Eventos" da GAMEYE app, é possível verificar que este jogador,

[17] Fraguando é um termo usado pela comunidade *gamming* e significa o ato de matar o adversário. Frags, por sua vez, é o número total de vezes que o jogador matou o adversário.

morreu apenas 8 vezes, das quais 5 foram mortes causadas pelo único jogador experiente da equipa adversária (jogador laranja – J3).

Finalmente, analisando um caso muito específico, passa-se quando um dos jogadores inexperientes (jogador azul claro – J6) morre após uma sequência de erros de jogo. Na Figura 55 que é apresentada a seguir encontra-se uma situação específica do jogador com *eye tracker* que o correu uns instantes antes de este morrer.

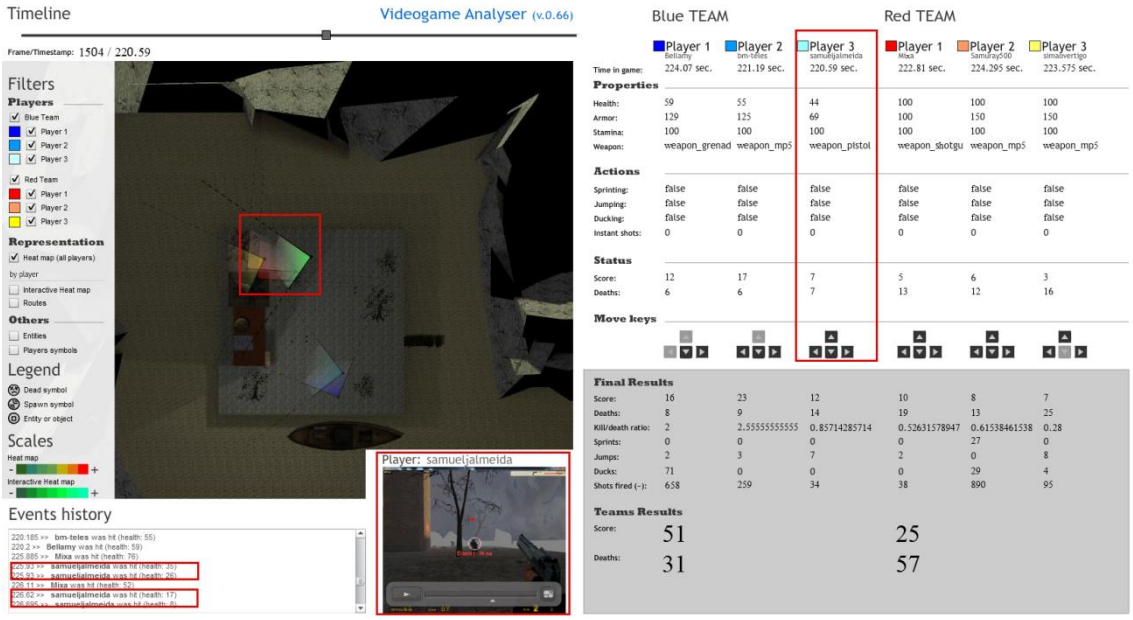


Figura 55 - Situação específica de jogo do jogador 6 (J6)

Na figura acima pode-se verificar os seguintes factos: em primeiro lugar, o jogador (J6) estava de frente para 3 adversários e os seus companheiros de equipa estavam demasiado distantes para o ajudar; em segundo, o jogador encontrava-se com 44 de vida e 69 de armadura enquanto que os seus 3 adversários tinham 100 de vida e 150 de armadura; em terceiro, o jogador estava parado à frente dos adversários (nenhuma tecla de movimentação estava a ser pressionada no momento – secção "Move keys") sendo um alvo mais fácil de atingir; finalmente, destaca-se o vídeo, onde se pode verificar que o jogador estava a olhar para cima do adversário que tinha de frente e não diretamente para ele (eventualmente estava distraído com algum *feedback* de jogo – neste caso um símbolo vermelho que aparece em cima da mira que indica que está a ser atingido pela frente).

Ainda na Figura 55, no histórico de eventos, também é possível verificar que o jogador (J6) foi atingido gradualmente até que o valor de vida chegou a zero (jogador morreu). A partir dos factos que foram indicados anteriormente desta situação específica de jogo, é possível prever o que aconteceu a seguir. Isto também deve-se ao facto de este jogador ser inexperiente e não ter qualquer ideia de trabalho de equipa. Um jogador experiente, em princípio não cometeria alguns erros que foram cometidos por este jogador como ficar parado à frente dos adversários tornando-se um alvo fácil. Acrescentando ainda, um jogador que tenha noções de trabalho de equipa, não vai atacar sozinho 3 adversários, mesmo que tenha a vida a 100%, o que não era o caso.

7.2.3. Observações e discussão

Como seria de esperar, segundo os resultados finais[18] de cada sessão, os jogadores mais experientes tiveram uma pontuação superior em relação aos outros jogadores. As métricas principais que o comprovam são a pontuação total (P), número de mortes (M) e a relação entre pontos e mortes (*points/death ratio*) (P/M). Na Tabela 5 são apresentados os resultados finais de cada participante das 6 rondas que constituíram a experiência.

Equipa	ID	Ronda 1			Ronda 2			Ronda 3			Ronda 4			Ronda 5			Ronda 6		
		P	M	P/M	P	M	P/M	P	M	P/M	P	M	P/M	P	M	P/M	P	M	P/M
Azul	J1 (exp.)	4	6	0,66	10	6	1,67	6	5	1,2	23	9	2,56	9	4	2,25	21	8	2,66
	J2 (exp.)	1	4	0,25	5	4	1,25	0	3	0	16	8	2	7	5	1,4	12	9	1,33
	J6 (inexp.)	0	5	0	2	**8**	**0,25**	**1**	6	0,16	12	14	0,86	2	8	0,25	3	16	0,19
Vermelha	J3 (exp. hardcore)	**8**	2	4	7	6	1,17	7	4	1,75	8	13	0,62	7	7	1	10	10	1
	J5 (inexp.)	1	4	0,25	3	8	0,27	1	2	0,5	10	19	0,52	2	6	0,33	11	13	0,85
	J4 (inexp.)	1	5	0,2	2	8	0,25	1	7	0,15	7	**25**	0,28	3	8	0,38	4	19	0,21
Jogador com *eye tracker*		J3			J2			J1			J6			J4			J5		

Tabela 5 - Resultados finais de cada jogador em 6 rondas e jogador com *eye tracker*

Em média os jogadores experientes tiveram uma relação entre pontos e mortes (*points/death ratio*) maior ou igual a 1 o que é um resultado positivo. O contrário aconteceu com os jogadores inexperientes que não conseguiram fazer uma ronda com uma relação de pontos e mortes superior a 1 o que significa que morreram mais vezes do que mataram.

A partir da Tabela 5 verifica-se que o jogador *hardcore* conseguiu, na primeira ronda, ter a relação de pontos e mortes mais alta de todos os jogadores (*points/death ratio* de 4, ou seja, matou quatro vezes mais em relação às suas mortes). Também é possível encontrar o jogador que morreu mais vezes de todos os jogadores com um total máximo de 25 mortes, que por sua vez pertencia ao grupo de jogadores inexperientes.

Na Figura 56 é apresentada a evolução da relação entre pontos e mortes dos jogadores ao longo das 6 rondas.

[18] Ver Anexo 5 – Resultados finais das sessões apresentados pela GAMEYE app

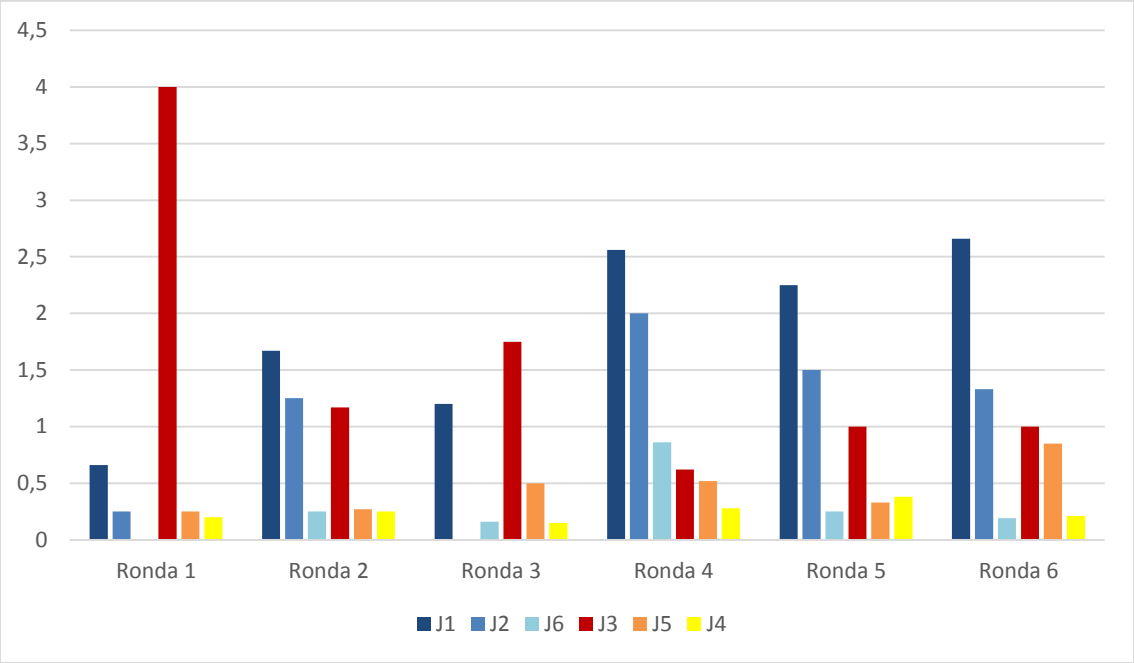


Figura 56 - Relação *points/death* dos jogadores ao longo das 6 rondas

Na Tabela 6 encontram-se os resultados finais de cada equipa, onde se pode verificar, a partir da pontuação (P) de cada uma, que a equipa Azul venceu 4 vezes e a equipa Vermelha venceu 2 vezes, o que dá 66.7% de vitórias à equipa Azul e 33,3% de vitórias à equipa Vermelha. Por coincidência 66,7% da equipa Azul é composta por jogadores experientes (2 jogadores de 3) enquanto que a equipa Vermelha é composta por 33,3% de jogadores experientes (1 jogador de 3).

Equipa	Ronda 1			Ronda 2			Ronda 3			Ronda 4			Ronda 5			Ronda 6		
	P	M	P/M	P	M	P/M	P	M	P/M	P	M	P/M	P	M	P/M	P	M	P/M
Azul	5	15	0,33	17	18	0,94	7	14	0,5	51	31	1,65	18	17	1,06	36	33	1,09
Vermelha	**10**	11	0,91	12	22	0,55	9	13	0,69	25	47	0,53	12	21	0,57	25	42	0,6

Tabela 6 - Resultados finais de cada equipa em 6 rondas

Com estes factos encontrados sobre os jogadores analisados na secção 7.2.2 e os resultados finais na presente secção entende-se que a experiência de cada jogador e o equilíbrio de equipas é extremamente influenciante no resultado final.

7.2.4. Opiniões dos participantes

Alguns jogadores referiram que o nevoeiro existente dificultava a visibilidade dos jogadores ao longe, levantando dúvidas entre o inimigo e um membro de equipa. Porém outros jogadores disseram que o facto de haver nevoeiro acrescentou maior competitividade de jogo aumentando assim a dificuldade de eliminar o adversário.

Um dos jogadores inexperientes (J4), no final da sessão, disse que sentiu estar a ser observado quando foi a sua vez de jogar com o computador equipado com *eye tracker*. Acrescentou ainda que sentiu que isso influenciou na sua performance de jogo. Esta pequena referência foi importante, e com isso pode-se verificar que das 6 sessões efetuadas, a sessão em que o jogador foi submetido a usar o computador equipado com *eye tracker* foi aquela em que ele teve uma relação entre pontos e mortes mais alta Tabela 5. Este facto é no mínimo contraditório, mas também é importante considerar que o computador equipado com *eye tracker* era o computador mais rápido da sala (com um taxa de atualização superior a 60 *frames* por segundo) o que pode ter influenciado para uma melhoria substancial da performance do jogador.

terceira parte

conclusões do estudo

8. Comentários finais/Conclusões

8.1. Confrontar as hipóteses com os objetivos e com o trabalho desenvolvido e apresentar

A presente investigação tinha como objetivos:

1. Perceber como guardar os dados dos comportamentos de interação para os *logfiles* (uso através do Source SDK - *Source Software Development Kit* -Valve *Software*);

2. Adaptar alguns dos conteúdos cénicos dos jogos de modo a perceber a influência que determinados objetos têm no desempenho do jogador;

3. Criar um método que cruze os dados provenientes dos *logfiles* do jogo (ações do jogador, coordenadas x, y e z e *timestamp*) com os *logfiles* dos dados resultantes sobre do *eye tracker* (*Point of Regard* e *timestamp*);

4. Apresentar uma proposta de uma aplicação baseada num paradigma de visualização que consiga apresentar os dados disponíveis sobre o desempenho do jogador.

As questões de investigação traçadas foram:

Que métodos/técnicas de visualização de informação permitem analisar o cruzamento de diferentes dados em contexto videojogo?

E as sub-perguntas foram:

1. *Quais os dados mais pertinentes para caracterizar um contexto de videojogo?*
2. *Quais as técnicas de cruzamento dos diferentes dados registados em contexto videojogo para efeitos de análise do mesmo?*
3. *Quais os esquemas de visualização adequados para análise dos dados que caracterizem um contexto videojogo?*

Após o trabalho desenvolvido e apresentado no capítulo 6 e tendo em conta os resultados apresentados no capítulo 7 do presente documento, verificou-se que:

A partir do trabalho realizado neste projeto, pode-se ter uma noção da utilidade que a GAMEYE app poderá vir a despertar na área de análise de videojogos. Existem imensos casos específicos que podem ser estudados a partir da app desenvolvida como foi possível verificar na secção 7.2. Estes casos podem ser particulares ou gerais.

Identificaram-se alguns paradigmas de visualização de informação que permitiram analisar o cruzamento de diferentes dados em contexto videojogo, nomeadamente configurações espaciais (*Topographic Map*) e considerações (*Heat Map*) (Correia, 2009, pp. 21-31; Costa, 2009, pp. 23-31). Como interação principal da GAMEYE app foi usado um controlo *timeline* para que fosse possível representar o comportamento dos jogadores ao longo do tempo.

Verifica-se que os dados provenientes dos comportamentos/interação do jogador com o jogo são os dados mais pertinentes para caracterizar um contexto de videojogo no âmbito desta investigação. Estes dados (métricas de jogo) são:

- Propriedades
    - id de jogo;
    - nome;
    - equipa;
    - *timestamp;*
    - posição (x, y e z);
    - ângulo de visão (x, y e z);
    - movimentação;
    - vida;
    - armadura;
    - arma;
- Ações
    - a disparar;
    - a saltar;
    - a correr;
    - agachado;
- Estados (pontuação)
    - pontos;
    - mortes;
- Eventos
    - conectou-se;
    - ativou-se;
    - escolheu/mudou de equipa;
    - foi atingido (e vida do jogador no momento);
    - morreu (e quem matou);
    - desconetou-se (e razão porque se desconetou).

Quanto às técnicas de cruzamento dos diferentes dados registados em contexto videojogo para efeitos de análise do mesmo, mostra-se serem aquelas que a partir de um vídeo referente a um jogador com *eye tracker* e a geo-representação do mesmo jogador através de um mini-map, numa sequência de tempo, possam determinar a localização dos elementos cénicos para o qual o jogador olhou durante a sua experiência de jogo. Ao mesmo tempo, nesta técnica, são apresentados dados relativos aos comportamentos do jogador para poderem ser comparados/cruzados com os dados anteriormente calculados.

Relativamente aos esquemas de visualização verifica-se que o *Topographic Map* e *Heat Map* são paradigmas de apresentação adequados para análise dos dados que caracterizam um contexto videojogo, como se pôde comprovar na secção 7.2 sendo adequados para representar os dados recolhidos na análise de situações específicas sem descontextualizar da situação efetiva e de ocorrência durante o jogo

8.2. Reflexão Crítica

Com a metodologia de análise de videojogos proposta neste trabalho pretende-se melhorar significativamente a experiência do jogador durante o momento de jogo. A análise resultante desta metodologia poderá resultar em algumas vantagens tanto a nível empresarial como de entretenimento. Destacam-se como exemplos: a deteção de falhas na mecânica de jogo; a melhoria de aspetos visuais e de usabilidade no mundo virtual; evitar lançamento do produto com bugs; evitar lançamento de *patches* com correções pontuais; aprendizagem na deteção de problemas no jogo; poupar tempo de desenvolvimento por parte dos criadores/*designers* em futuras versões do jogo; uso como ferramenta de estratégia/treino e pós-jogo para clãs e comunidades de videojogos; criação de estatísticas e rankings comunitários, entre outros.

A GAMEYE app, por enquanto, implica apenas as técnicas de *logging* (métricas de jogo) e *eye tracking*, porém sugere-se também o uso de outras técnicas de recolha de dados que não são visíveis com as técnicas referidas. Como já foi referido na secção 4.1 e 4.3 existem outras técnicas vantajosas e que podem vir a complementar este estudo: técnicas de observação; técnicas de consulta; opiniões de especialistas; inquéritos e questionários; testes de aprovação; avaliação durante o uso ativo (entrevistas, *focus groups*) e *game heuristics*. L. Nacke et al. (2009) também refere outras técnicas ainda não referidas como relatórios verbais, *play-personas*, resposta galvânica da pele (GSR - *Galvanic Skin Response*), medição cardiovascular (HR – *Heart Rate*), eletromiografia (EMG - *Electromyogram*), respiração, elétrodos de eletroencefalograma (EEG - *Electroencephalogram*) e eletromiografia.

Neste estudo, optou-se por fazer um estudo em laboratório onde os jogadores foram convidados a ir a um local para poderem realizar a experiência. Como é referido na secção 4.1, um estudo em laboratório tem a desvantagem de não ser uma avaliação mais natural devido á falta de contexto. Contudo, era a única opção visto que era necessário usar o equipamento *eye tracker*, e pretendia-se juntar os vários jogadores numa sala para fazerem um jogo em LAN. Por outro lado, também se pode considerar um estudo de campo, já que quando os jogadores querem jogar em LAN, têm que se deslocar para um local em comum onde todos possam jogar presencialmente, tal como aconteceu neste estudo. Mas isto também depende da categoria de jogo. No caso da categoria FPS é usual fazerem-se LANs presenciais, mas no caso de um MMORPG o jogo é todo ele feito a distância (Internet) e aí o estudo de campo seria em locais como quarto, sala ou escritório de casa. Para estes casos o ideal seria o estudo a distância, à semelhança do estudo da bolt|peters referido na secção 5.2.2.

8.3. Limitação do estudo

Durante o desenvolvimento deste projeto surgiram algumas barreiras, nomeadamente relacionado com o SDK, dada a complexidade na configuração do kit. Ainda, a base ou *template* que a Valve Software oferece à comunidade é muito útil, mas incompleta em alguns aspetos, tais como a falta de inteligência artificial nos *bots* e pedaços de código pouco desenvolvidos em algumas livrarias.

Tornou-se difícil a procura de situações específicas no código do *mod*, como por exemplo, encontrar as funções que gerem os eventos ocorridos durante o jogo. Este processo de procura e de teste foi moroso, pois necessitou de um constante recurso ao *debugging* de forma a perceber que informação estava a passar em determinadas variáveis.

O facto de apenas existir um *eye tracker* disponível para o estudo fez com que a GAMEYE app se limitasse a representar os movimentos dos olhos de apenas um jogador durante o jogo. A melhor situação possível para este estudo, considerando o facto de ser o desenvolvimento de um protótipo, seria a existência de 6 *eye trackers*, um para cada jogador.

Rede na Universidade de Aveiro dificultou a realização de alguns testes durante a pré-fase em relação á experiência final, no entanto encontraram-se soluções alternativas usando, nomeadamente redes independentes da universidade, uma rede "ad hoc" com partilha de Internet através de uma *pen* de banda larga e outra rede local originada por um ponto de acesso *switch* com capacidade *wireless*.

Pretendia-se captar a direção ocular do utilizador para identificar os objetos para qual o jogador olhou durante a sessão de jogo. Esta funcionalidade, talvez a mais complexa de todas as existentes, requeria o domínio de vários conceitos relacionados com a área de matemática. Conceitos esses que estão ligados ao cálculo de projeções 2D/3D e cálculo matricial. A complexidade é tal que o tempo necessário para a sua implementação transcende o tempo disponível para a execução deste estudo. Contudo, a GAMEYE app já foi preparada para interpretar os *logfiles* inerentes ao *eye tracker* que por sua vez também já estão a ser interpretados e sincronizados temporalmente com os *logfiles* devolvidos pelo *mod* de jogo desenvolvido.

8.4. Perspetivas futuras de investigação e desenvolvimento

Uma das perspetivas de desenvolvimento futuro passará pela adaptação do *mod* para um ambiente relacionado com a Universidade de Aveiro (camada de produto CounterStrike: Source - UA), na expectativa de sensibilizar e envolver a comunidade de jogadores da Universidade de Aveiro neste projeto.

Em relação à GAMEYE app, espera-se melhora-la e otimizá-la das seguintes formas:

- Migrar parte do código desenvolvido em ActionScript 3.0 para ficheiros externos;
- Dinamizar a app para poder suportar no mínimo 1 jogador até 64 jogadores por experiência, mesmo sendo um número pouco provável, mas possível de acontecer como em alguns videojogos das séries de Battlefield da Electronic Arts;

- Preparar a app para suportar, em vez de um vídeo, vários vídeos com *eye tracker* para cada participante;
- Desenvolver a funcionalidade que identifica os objetos para o qual o jogador olhou durante a sessão de jogo. Esta funcionalidade está relacionada com o uso do *logfile* produzido através do *eye tracker* e a sua integração com os *logfiles* produzidos pelo *mod* desenvolvido. Alguma parte desta funcionalidade já está implementada como a sincronização de *logfiles*. Resta desenvolver o algoritmo que irá identificar os objetos no ambiente cénico (mapa de jogo) para qual o jogador olhou durante a sessão e ao mesmo tempo, representar visualmente na app um vetor com a direção do olhar do jogador geo-representado (posição e sentido).
- Acrescentar a funcionalidade para criar um relatório com base nos resultados apresentados através de paradigmas de visualização de dados.
- Adicionar vista de *timeline* complexa em forma de gráfico com marcadores dos vários eventos ocorridos ao longo da sessão de jogo.
- Possibilitar ao analista de adicionar comentários/notas em cada momento de jogo.

O grande contributo desta dissertação fica assim refletido no desenvolvimento da GAMEYE app que consegue demonstrar o método que cruza os dados provenientes dos *logfiles* do jogo com o vídeo resultante do *eye tracker,* e na proposta da app baseada em paradigmas de visualização onde se consegue representar os dados disponíveis sobre o desempenho dos jogadores.

Bibliografia

Adobe. (2010). split (String.split method), 2012, from http://help.adobe.com/en_US/as2/reference/flashlite/WS5b3ccc516d4fbf351e63e3d118ccf9c47f-7ea2.html

Adobe. (2012). Adobe Flash Professional CS6, from http://www.adobe.com/products/flash.html

Almeida, S. (2009). *Augmenting Video Game Development with Eye Movement Analysis.* Aveiro University, Aveiro.

Álvarez, S. B. D., García, A. G., Garófano, C. J., & Jiménez, M. d. P. M. (2008). Bases Optométricas para una Lectura eficaz, from http://www.visiondat.com/index.php?mod=articulos&art=70

Bianco, C. (2000). How Vision Works Retrieved 22 Outubro, 2010, from http://health.howstuffworks.com/human-body/systems/eye/eye2.htm

Björk, S., Lundgren, S., & Holopainen, J. (2005). Game Design Patterns.

Bruckman, A. (2006). Chapter 58: Analysis of Log File Data to Understand Behavior and Learning in an Online Community Retrieved from http://www.springerlink.com/content/u30334l51020x236/

Carmo, H., & Ferreira, M. M. (1998). Metodologia da Investigação: Guia para Auto-aprendizagem. Retrieved from

Clanton, C. (1998). *An Interpreted Demonstration of Computer Game Design.* Paper presented at the CHI 98 conference summary on Human factors in computing systems, Los Angeles: ACM.

Correia, P. (2009). *Monitorização e Visualização de Notícias: estudo de caso do Portal Sapo.* Universidade de Aveiro, Aveiro.

Costa, J. (2009). *Monitorização e Geovisualização de Pesquisas Web no Portal Sapo.* Universidade de Aveiro, Aveiro.

Cotti, L. (2009). O Sentido da Visão, from http://www.administradores.com.br/informe-se/artigos/o-sentido-da-visao/30440/

Crawford, C. (1982). The Art of Computer Game Design W. S. University (Ed.) Retrieved from http://arcarc.xmission.com/Magazines%20and%20Books/Art%20of%20Game%20Design.pdf

Crowe, E. C., & Narayanan, N. H. (2000). Comparing interfaces based on what users watch and do. *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications*, 7.

Crytek. (2012). Crytek - MyCryEngine, 2012, from http://mycryengine.com/

Desurvire, H., Caplan, M., & Toth, J. A. (2004). *Using Heuristics to Evaluate the Playability of Games*. Paper presented at the CHI 2004, Vienna, Austria. http://portal.acm.org/citation.cfm?id=986102

Dix, A., Finlay, J., Abowd, G., & Beale, R. (1998). *Human-Computer Interaction* (2 ed.): Prentice Hall.

Duchowski, A. (2007). Eye Tracking Methodoloy: Theory and Practice

El-Nasr, M. S., & Yan, S. (2006). *Visual Attention in 3D Video Games.* Paper presented at the Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, Hollywood: ACM.

Elst, P., Jacobs, S., & Yard, T. (2007). Object-Oriented ActionScript 3.0.

EpicGames. (2010). Unreal Technology Retrieved 14 de janeiro, 2011, from http://www.unreal.com/

EpicGames, I. (2012). Game Engine Technology by Unreal, from http://www.unrealengine.com/

Federoff, M. (2002). *Heuristics and Usability Guidelines for the Creation and Evaluation of Fun in Video Games.* Master, Indiana University. Retrieved from http://melissafederoff.com/heuristics_usability_games.html

Frostbite2. (2011). Unofficial News on Frostbite 2, from http://www.frostbite2.com

Giannotto, E. (2009). *Uso de Rastreamento do Olhar na Avaliação da Experiência do Tele-Usuário de Aplicações de TV Interativa.* Escola Politécnica da Universidade de São Paulo, São Paulo. Retrieved from http://www.teses.usp.br/teses/disponiveis/3/3141/tde-15042009-151212/publico/dissertacao.pdf

Grossman, G., & Huang, E. (2012). ActionScript 3.0 overview from http://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html

half-lifeWiki. (2012). GoldSrc. Retrieved from http://half-life.wikia.com/wiki/GoldSrc

Hubel, D. (1995). Eye, Brain and Vision Retrieved from http://hubel.med.harvard.edu/book/bcontex.htm

Hulshof, C. D. (2004). Log file analysis *Encyclopedia of Social Measurement.* Twente.

InfinityWard. (2012). IW Engine - Mod DB, from http://www.moddb.com/engines/iw-engine

ISO. (2009). ISO 9241-11:1998 - Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on Usability Retrieved 24 de Dezembro, 2010, from http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16883

Ivory, M. Y., & Hearst, M. A. (2001). The State of the Art in Automating Usability Evaluation of User Interfaces Vol. 33. (pp. 470–516).

Jacob, R. J. K., & Karn, K. S. (2003). Eye Tracking in Human–Computer Interaction and Usability Research: Ready to Deliver the Promises *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*

Johansen, S. A., Nørgaard, M., & Rau, J. (2008). *Can Eye Tracking Boost Usability Evaluation of Computer Games?* http://www.itu.dk/docadm/detail.php?DocID=1706

json. (2011). Introducing JSON, from http://json.org/

Karn, K. S. (2006). *Eye Tracking for Usability Testing, You've Got to Know Your Strengths and Weaknesses*. Paper presented at the workshop CHI 2006, Rochester, NY 14623 USA. http://www.amber-light.co.uk/index.shtml

Kennerly, D. (2003). Better Game Design Through Data Mining Retrieved 22 de Novembro, 2010, from http://www.gamasutra.com/view/feature/2816/better_game_design_through_data_.php

Kim, J. H., Gunn, D. V., Schuh, E., Phillips, B. C., Pagulayan, R. J., & Wixon, D. (2008). *Tracking Real-Time User Experience (TRUE): A comprehensive instrumentation solution for complex systems.* Paper presented at the CHI 2008 Proceedings, Florence, Italy.

Li, D., Babcock, J., & Parkhurst, D. J. (2006). *openEyes: a low-cost head-mounted eye-tracking solution*. Paper presented at the The Human Computer Interaction Program Iowa State University, Ames, Iowa, 50011. http://thirtysixthspan.com/openEyes/li_etal06.pdf

Lucas, K., & Sherry, J. L. (2004). Sex Differences in Video Game Play: A Communication-Based Explanation. *31*, 499-523. Retrieved from http://icagames.comm.msu.edu/cr.pdf

McLuhan, M. (1994). Understanding Media: The Extensions of Man

MedicinaGeriátrica. (2007). Envelhecimento do olhos, 2010, from http://www.medicinageriatrica.com.br/2007/05/14/envelhecimento-dos-olhos/

Microsoft. (2010). Debugging in Visual Studio, from http://msdn.microsoft.com/en-us/library/sc65sadd.aspx

Missagia, L. (2010). E o olho com isso? *Hipertensão*.

Nacke, L., Ambinder, M., Canossa, A., Mandryk, R., & Stach, T. (2009). *Game Metrics and Biometrics: The Future of Player Experience Research.* Paper presented at the Future Play 2009, Vancouver, BC - Canada.

Nacke, L. E., Niesenhaus, J., Poels, K., Drachen, A., Korhonen, H. J., IJsselsteijn, W. A., . . . Kort, Y. A. W. d. (2009). *Playability and Player Experience Research*. http://www.bth.se/fou/forskinfo.nsf/all/e0a8cdd8cfc0c7e6c125762c005557c0/$file/Nacke-etal-Panel%20Playability%20and%20Player%20Experience.pdf

Nielsen, J. (2003). Usability 101: Introduction to Usability., 2010, from http://www.useit.com/alertbox/20030825.html

Nielsen, J. (2005). Ten Usability Heuristics, 2010, from http://www.useit.com/papers/heuristic/heuristic_list.html

Norman, D. A. (1998). The Design of Everyday Things. Retrieved from

Pardal, L., & Correia, E. (1995). Métodos e Técnicas de Investigação Social. Retrieved from

Pinelle, D., Wong, N., & Stach, T. (2008). *Heuristic Evaluation for Games: Usability Principles for Video Game Design.* Paper presented at the CHI 2008, Florence, Italy.

Postigo, H. (2007). Games and Culture. *2*. Retrieved from http://gac.sagepub.com/cgi/content/abstract/2/4/300

Redline, C. D., & Lankford, C. P. (2001). *Eye-Movement Analysis: A New Tool for Evaluating the Design of Visually Administered Instruments (Paper and Web)*. Paper presented at the American Association of Public Opinion Research.

Ribeiro, N. (2010). A indústria dos videojogos Retrieved 10 de Dezembro, 2010, from http://cibertransistor.com/2010/02/19/a-industria-dos-videojogos/

Rollings, A., & Adams, E. (2003). Andrew Rollings and Ernest Adams on Game Design

Salen, K., & Zimmerman, E. (2004). Rules of Play: Game Design Fundamentals

Salen, K., & Zimmerman, E. (2005). The Game Design Reader - A Rules of Play Anthology T. M. Press (Ed.) Retrieved from http://www.gamersmob.com/files/gameDesignReader/GameDesignReader_TOC.pdf

Sánchez, J. L. G., Zea, N. P., & Gutiérrez, F. L. (2009). Playability: How to Identify the Player Experience in a Video Game Retrieved from http://www.springerlink.com/content/2n30xkuk34164606/ doi:10.1007/978-3-642-03655-2_39

Sasse, D. B. (2008). *A Framework for Psychophysiological Data Acquisition in Digital Games.* Blekinge Institute of Technology, Karlshamn, Sweden. Retrieved from http://www.gamecareerguide.com/thesis/080520_sasse.pdf

Scacchi, W. (2010). *Computer Game Mods, Modders, Modding, and the Mod Scene*. http://www.ics.uci.edu/~wscacchi/GameLab/ModSquad-Scacchi.pdf

Sennersten, C. (2004). *Eye movements in an Action Game Tutorial*, Lund University, Sweden.

Shneiderman, B. (1998). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (3 ed.): Addison Wesley Longman.

Sicart, M. (2008). Defining Game Mechanics Retrieved 23 de Dezembro, 2010, from http://gamestudies.org/0802/articles/sicart

Snow, B. (2007). Gaming Usability 101. Retrieved from http://www.businessweek.com/innovate/content/oct2007/id20071012_041625.htm

Sotamaa, O. (2007). *Perceptions of player in game design literature.* Paper presented at the Situated Play: Proceedings of the 2007 Digital Games Research Association Conference, B. Akira, Ed., The University of Tokyo (Tokyo.

Sotamaa, O. (2008). When The Game is Not Enough: Motivations and Practices among Computer Game Modding Culture. Retrieved from http://www.uta.fi/~tlolso/documents/Games&Culture_Sotamaa.pdf

Tanaka, T. (2011). DICE: "We're considering mod tools". Retrieved from http://bf3blog.com/2011/08/dice-were-considering-mod-tools/

Todd, D. (2007). Game Design: From Blue Sky to Green Light S. Editorial, and Customer Service Office (Ed.)

Tychsen, A. (2008). *Crafting user experience via game metrics analysis*.

UnityTechnologies. (2012). Create Games With Unity, from http://unity3d.com/#creategames

Valve. (2007). Source Engine, from http://source.valvesoftware.com/

ValveCommunity. (2010). Source SDK Documentation, from http://developer.valvesoftware.com/wiki/SDK_Docs

ValveCommunity. (2011). Level Overviews, from https://developer.valvesoftware.com/wiki/Level_Overviews

ValveCorporation. (2004). Half-Life 2 Retrieved 17 de Janeiro, 2011, from http://orange.half-life2.com/hl2.html

ValveCorporation. (2008). Friday, September 26, 2008 Retrieved 14 de Janeiro, 2011, from http://storefront.steampowered.com/Steam/Marketing/message/1843/

ValveCorporation. (2010). Steam Surpasses 30 Million Account Mark Retrieved 14 de Janeiro, 2011, from http://store.steampowered.com/news/4502/

Ventura, M. A. A. (2009). *Etnografia de uma comunidade de jogadores de FPS.* Faculdade de Engenharia da Universidade do Porto, Minho.

Ward, J. (2008). What is a Game Engine? , from http://www.gamecareerguide.com/features/529/what_is_a_game_.php

Wolf, M. (2000). Genre and the Video Game.

Wong, A. (2007). Eye Movement Disorders

Anexos

Anexo 1 – Ficheiro de configuração das equipas

O texto seguinte representa um excerto de código utilizado na configuração das equipas a serem utilizadas no mapa de jogo implementado.

```
//------------------------------------------------------------------------
//
// Game data for Half-Life 2 Multiplayer.
//
//------------------------------------------------------------------------

@include "halflife2.fgd"


@PointClass base(PlayerClass, Angles) studio("models/editor/playerstart.mdl") = info_player_deathmatch :
    "This entity indicates the position and facing direction at which the player will spawn during a deathmatch map. Any number of "+
    "info_player_deathmatch entities may be placed in a map."
[
]


@PointClass base(PlayerClass, Angles) studio("models/editor/playerstart.mdl") = info_player_blue :
    "This entity indicates the position and facing direction at which the player will spawn during a deathmatch map. Any number of "+
    "info_player_deathmatch entities may be placed in a map."
[
]


@PointClass base(PlayerClass, Angles) studio("models/editor/playerstart.mdl") = info_player_red :
    "This entity indicates the position and facing direction at which the player will spawn during a deathmatch map. Any number of "+
    "info_player_deathmatch entities may be placed in a map."
[
]
```

Anexo 2 – Logfile “Jogadores”

As seguintes linhas de texto representam alguns registo efetuados no *logifle* Jogadores

```
10/17/12-23:26:10][11.34][4|Mixa|3][-768|816|64.0313][0|0|0][0|0][100|100|100|0|0|weapon_shotgun|0|0|0|0|0
10/17/12-23:26:10][11.835][6|Samuray500|3][-512|944|64.0313][0|0|0][0|0][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:10][12.03][6|Samuray500|3][-512|944|64.0313][0|-41.5897|0][0|0][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:11][12.27][2|Bellamy|2][679.221|-258.878|64.0313][0|44.9684|0][0|0][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:11][12.435][4|Mixa|3][-768|816|64.0313][0|0|0][0|0][100|100|100|0|0|weapon_shotgun|0|0|0|0|0
10/17/12-23:26:11][12.525][6|Samuray500|3][-552.709|957.688|64.0313][0|-58.552|0][-141.421|141.421][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:11][12.75][2|Bellamy|2][731.43|-208.109|64.0313][0|67.9448|0][200|0][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:11][12.9][4|Mixa|3][-768|816|64.0313][0|0|0][0|0][100|100|100|0|0|weapon_shotgun|0|0|0|0|0
10/17/12-23:26:11][13.005][5|bm-teles|2][647.597|-343.444|64.0313][0|128.093|0][200|0][100|150|100|1|-1|weapon_shotgun|0|0|0|0|0
10/17/12-23:26:11][13.14][6|Samuray500|3][-665.938|929.735|64.0313][0|-57.2444|0][141.421|141.421][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:12][13.32][6|Samuray500|3][-692.194|906.89|64.0313][0|-130.603|-0][141.421|141.421][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:12][13.47][5|bm-teles|2][586.999|-273.254|64.0313][0|140.245|0][200|0][100|150|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:12][13.605][5|bm-teles|2][566.956|-255.165|64.0313][0|140.369|0][200|0][100|150|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:12][13.77][6|Samuray500|3][-795.069|890.946|64.0313][0|-148.365|0][253.741|53.4017][100|150|78|1|-1|weapon_mp5|1|0|0|0|0
10/17/12-23:26:12][13.92][5|bm-teles|2][518.917|-214.41|64.0313][0|141.37|0][200|0][100|150|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:12][14.13][2|Bellamy|2][528.592|-214.008|64.0313][0|-139.626|0][188.355|-67.2482][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:13][14.25][3|samueljalmeida|2][448|-192|64.0313][0|0|0][0|0][100|100|100|1|-1|weapon_shotgun|0|0|0|0|0
10/17/12-23:26:13][14.325][6|Samuray500|3][-908.172|807.702|64.0313][0|-108.539|0][175.504|175.504][100|150|67|1|-1|weapon_mp5|1|0|0|0|0
10/17/12-23:26:13][14.415][6|Samuray500|3][-925.471|793.684|64.0313][0|-93.4566|0][174.231|174.231][100|150|66|1|-1|weapon_mp5|1|0|0|0|0
10/17/12-23:26:13][14.64][2|Bellamy|2][496.877|-308.941|64.0313][0|-66.9389|0][200|0][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:13][14.745][3|samueljalmeida|2][446.537|-195.215|64.0313][0|-44.6629|0][0|0][100|100|100|1|-1|weapon_shotgun|0|0|0|0|0
10/17/12-23:26:13][14.88][2|Bellamy|2][509.845|-355.02|64.0313][0|-62.6962|0][200|0][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:13][14.94][6|Samuray500|3][-1013.07|702.63|64.0313][0|-86.856|0][235.9|0][100|150|55|1|-1|weapon_mp5|1|0|0|0|0
10/17/12-23:26:13][15.075][6|Samuray500|3][-1020.72|671.994|64.0313][0|-86.856|0][233.2|0][100|150|52|1|-1|weapon_mp5|1|0|0|0|0
10/17/12-23:26:14][15.195][5|bm-teles|2][306.559|-98.0142|64.0313][0|138.596|-0][141.421|-141.421][100|150|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:14][15.375][2|Bellamy|2][527.156|-445.55|64.0313][0|-96.2254|0][141.421|141.421][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:14][15.495][4|Mixa|3][-1104.35|737.1|64.0313][0|-62.5565|0][220|0][100|100|100|0|0|weapon_shotgun|0|0|0|0|0
10/17/12-23:26:14][15.6][2|Bellamy|2][493.22|-474.949|64.0313][0|-104.743|0][141.421|141.421][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:14][15.69][6|Samuray500|3][-1025.61|532.424|64.0313][0|-89.364|0][220.9|0][100|150|40|1|-1|weapon_mp5|1|0|0|0|0
10/17/12-23:26:14][15.87][4|Mixa|3][-1087.79|656.975|64.0313][0|-70.9227|-0][220|0][100|100|100|0|0|weapon_grenade|0|0|0|0|0
10/17/12-23:26:14][15.975][2|Bellamy|2][433.121|-519.431|64.0313][0|-155.522|-0][200|0][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:14][16.065][5|bm-teles|2][161.116|-60.4723|64.0313][0|135.593|0][0|0][100|150|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:15][16.215][3|samueljalmeida|2][446.537|-195.215|64.0313][0|178.602|0][0|0][100|100|100|1|-1|weapon_shotgun|0|0|0|0|0
10/17/12-23:26:15][16.335][2|Bellamy|2][365.031|-534.148|64.0313][0|112.136|-0][141.421|-141.421][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:15][16.485][2|Bellamy|2][338.439|-520.489|64.0313][0|92.4991|-0][141.421|-141.421][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:15][16.605][4|Mixa|3][-1068.67|501.1|64.0313][0|-82.6321|0][220|0][100|100|100|0|0|weapon_shotgun|0|0|0|0|0
10/17/12-23:26:15][16.725][2|Bellamy|2][302.837|-488.376|64.0313][0|85.0772|-0][141.421|-141.421][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:15][16.815][3|samueljalmeida|2][427.487|-177.111|64.0313][0|137.111|0][0|0][100|100|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:15][16.905][6|Samuray500|3][-1037.59|280.079|64.0313][0|-89.562|0][139.017|139.017][100|150|16|1|-1|weapon_mp5|1|0|0|0|0
10/17/12-23:26:15][17.04][5|bm-teles|2][163.321|48.7417|64.0313][0|139.311|0][0|0][100|150|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:15][17.145][6|Samuray500|3][-1052|236.212|64.0313][0|-89.562|0][191.8|0][100|150|11|1|-1|weapon_mp5|1|0|0|0|0
10/17/12-23:26:16][17.295][6|Samuray500|3][-1054.82|207.79|64.0313][0|-89.562|0][188.8|0][100|150|8|1|-1|weapon_mp5|1|0|0|0|0
10/17/12-23:26:16][17.445][2|Bellamy|2][194.824|-393.478|64.0313][0|89.937|-0][141.421|-141.421][100|150|100|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:16][17.52][5|bm-teles|2][121.672|83.9276|64.0313][0|140.14|0][0|0][100|150|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:16][17.685][3|samueljalmeida|2][331.945|-91.1175|64.0313][0|138.371|-0][220|0][100|100|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:16][17.775][5|bm-teles|2][110.132|93.6677|64.0313][0|140.111|0][0|0][100|150|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:16][17.925][3|samueljalmeida|2][292.619|-55.8851|64.0313][0|138.042|0][220|0][100|100|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:16][18.015][6|Samuray500|3][-1055.65|95.0553|64.0313][0|-86.724|0][100|0][100|150|0|1|-1|weapon_mp5|1|0|0|0|0
10/17/12-23:26:16][18.18][3|samueljalmeida|2][250.915|-18.3622|64.0313][0|138.042|0][220|0][100|100|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:17][18.255][6|Samuray500|3][-1054.35|70.8203|64.0313][0|-86.724|0][102.4|0][100|150|2|1|-1|weapon_mp5|0|0|0|0|0
10/17/12-23:26:17][18.375][5|bm-teles|2][110.072|93.7178|64.0313][0|140.054|0][0|0][100|150|100|1|-1|weapon_grenade|0|0|0|0|0
10/17/12-23:26:17][18.555][3|samueljalmeida|2][189.566|36.7966|64.0313][0|138.282|0][220|0][100|100|100|1|-1|weapon_grenade|0|0|0|0|0
```

Anexo 3 – Logfile “Mapa”

As seguintes linhas de texto representam uma parte do *logifle* Mapa produzido pelo Mod quando o servidor de jogo está a ser iniciado.

```
[{
"world_maxs" "2592 2880 1236"
"world_mins" "-3008 -2624 -176"
"spawnflags" "0"
"skyname" "sky_day01_01"
"maxpropscreenwidth" "-1"
"detailvbsp" "detail.vbsp"
"detailmaterial" "detail/detailsprites"
"classname" "worldspawn"
"mapversion" "19"
"hammerid" "0"
}
{
"origin" "756.875 -709.309 64"
"targetname" "Arvore E"
"StartDisabled" "0"
"spawnflags" "0"
"solid" "6"
"skin" "0"
"SetBodyGroup" "0"
"rendermode" "0"
"renderfx" "0"
"rendercolor" "255 255 255"
"renderamt" "255"
"RandomAnimation" "0"
"pressuredelay" "0"
"PerformanceMode" "0"
"model" "models/props_foliage/tree_deciduous_01a-lod.mdl"
"mindxlevel" "0"
"MinAnimTime" "5"
"maxdxlevel" "0"
"MaxAnimTime" "10"
"fadescale" "1"
"fademindist" "-1"
"fademaxdist" "0"
"ExplodeRadius" "0"
"ExplodeDamage" "0"
"disableshadows" "0"
"disablereceiveshadows" "0"
"angles" "0 0 0"
"classname" "prop_dynamic"
"hammerid" "6"
}
{
"origin" "-192 736 73"
"targetname" "Arvore C"
"StartDisabled" "0"
```

Anexo 4 – Questionário usado no estudo empírico

Questionário | Videojogos

* Required

ID *

Idade *

Sexo *

- Masculino
- Feminino

Quanto tempo, em média, joga videojogos durante a semana? *

- <1 hora
- 1-5 horas
- 6-10 horas
- > 10 horas

Já pertenceu a um clã de videojogos do género Shooter? *

- Sim. Mas só como grupo de amigos
- Sim. Com participação em competições
- Não

Qual ou quais plataformas de jogos mais utiliza? *

Indique todasque se aplicam.

- PlayStation
- Xbox/Kinect
- Wii
- PC
- Online
- Other:

Qual ou quais géneros de jogo que costuma jogar? *

Indique todosque se aplicam.

- Shooter (First-person; Third-person; outros)
- Role-playing game (RPG)
- Real-time strategy (RTS)
- Ação
- Aventura
- Desporto (Simulação)
- Corridas/Carros (Simulação)
- Outros simuladores
- Plataformas
- Other:

Submit

Powered by Google Docs

Report Abuse - Terms of Service - Additional Terms

Anexo 5 – Resultados finais das sessões apresentados pela GAMEYE app

Resultados da Ronda 1

	Blue TEAM			Red TEAM		
	Player 1 samueljalmeida	Player 2 Bellamy	Player 3 bm-teles	Player 1 Samuray500	Player 2 Mixa	Player 3 simaovertigo
Time in game:	184.425 sec.	184.245 sec.	179.295 sec.	184.635 sec.	183.15 sec.	183.795 sec.
Properties						
Health:	100	90	85	53	100	23
Armor:	100	150	141	127	100	109
Stamina:	100	100	100	5	100	100
Weapon:	weapon_shotgu	weapon_grenad	weapon_pistol	weapon_grenad	weapon_shotgu	weapon_mp5
Actions						
Sprinting:	false	false	false	false	false	false
Jumping:	false	false	false	false	false	false
Ducking:	false	false	false	false	false	false
Instant shots:	0	0	0	0	0	0
Status						
Score:	0	1	4	8	1	1
Deaths:	5	4	6	2	4	5
Move keys						

Final Results						
Score:	0	1	4	8	1	1
Deaths:	5	4	6	2	4	5
Kill/death ratio:	0	0.25	0.66666666666	4	0.25	0.2
Sprints:	0	0	0	21	0	0
Jumps:	3	1	4	0	0	2
Ducks:	0	0	0	4	0	0
Shots fired (~):	1	111	128	543	24	36

Teams Results		
Score:	5	10
Deaths:	15	11

Resultados da Ronda 2

	Blue TEAM			Red TEAM		
	Player 1 samueljalmeida	Player 2 bm-teles	Player 3 Bellamy	Player 1 simaovertigo	Player 2 Mixa	Player 3 Samuray500
Time in game:	225.915 sec.	227.19 sec.	227.115 sec.	226.545 sec.	224.085 sec.	225.315 sec.
Properties						
Health:	90	6	83	37	7	80
Armor:	100	100	146	118	56	150
Stamina:	100	100	100	100	100	100
Weapon:	weapon_shotgu	weapon_shotgu	weapon_mp5	weapon_mp5	weapon_grenad	weapon_mp5
Actions						
Sprinting:	false	false	false	false	false	false
Jumping:	false	false	true	false	false	false
Ducking:	false	false	false	false	false	false
Instant shots:	0	0	0	0	0	0
Status						
Score:	2	10	5	2	3	7
Deaths:	8	6	4	8	8	6
Move keys						

Final Results						
Score:	2	10	5	2	3	7
Deaths:	8	6	4	8	8	6
Kill/death ratio:	0.25	1.66666666666	1.25	0.25	0.375	1.16666666666
Sprints:	0	0	0	1	0	28
Jumps:	14	1	2	8	0	3
Ducks:	0	0	2	3	0	0
Shots fired (-):	12	189	107	41	25	440

Teams Results		
Score:	17	12
Deaths:	18	22

Resultados da Ronda 3

	Blue TEAM			Red TEAM		
	Player 1 bm-teles	Player 2 Bellamy	Player 3 samueljalmeida	Player 1 Mixa	Player 2 Samuray500	Player 3 simaovertigo
Time in game:	148.02 sec.	146.07 sec.	146.7 sec.	147.6 sec.	148.425 sec.	148.095 sec.
Properties						
Health:	100	93	100	100	21	37
Armor:	150	146	100	100	105	115
Stamina:	100	100	100	100	100	100
Weapon:	weapon_shotgu	weapon_grenad	weapon_shotgu	weapon_grenad	weapon_mp5	weapon_shotgu
Actions						
Sprinting:	false	false	false	false	false	false
Jumping:	false	false	false	false	false	false
Ducking:	false	false	false	false	false	false
Instant shots:	0	0	0	0	13	0
Status						
Score:	6	0	1	1	7	1
Deaths:	5	3	6	2	4	7
Move keys						

Final Results						
Score:	6	0	1	1	7	1
Deaths:	5	3	6	2	4	7
Kill/death ratio:	1.2	0	0.16666666666	0.5	1.75	0.14285714285
Sprints:	0	0	0	0	1	0
Jumps:	2	3	5	0	1	1
Ducks:	0	9	0	0	0	0
Shots fired (-):	4	174	5	9	717	59

Teams Results

Score:	7	9
Deaths:	14	13

Resultados da Ronda 4

	Blue TEAM			Red TEAM		
	Player 1 Bellamy	Player 2 bm-teles	Player 3 samueljalmeida	Player 1 Mixa	Player 2 Samuray500	Player 3 simaovertigo
Time in game:	366.675 sec.	366.48 sec.	311.265 sec.	367.785 sec.	367.875 sec.	367.08 sec.
Properties						
Health:	100	6	100	100	45	100
Armor:	150	103	100	100	116	150
Stamina:	100	100	100	100	100	100
Weapon:	weapon_grenad	weapon_shotgu	weapon_grenad	weapon_shotgu	weapon_mp5	weapon_mp5
Actions						
Sprinting:	false	false	false	false	false	false
Jumping:	false	false	false	false	false	false
Ducking:	false	false	false	false	false	false
Instant shots:	0	0	0	0	4	0
Status						
Score:	16	23	9	10	8	7
Deaths:	8	9	11	19	13	25
Move keys						

Final Results						
Score:	16	23	12	10	8	7
Deaths:	8	9	14	19	13	25
Kill/death ratio:	2	2.55555555555	0.85714285714	0.52631578947	0.61538461538	0.28
Sprints:	0	0	0	0	27	0
Jumps:	2	3	7	2	0	8
Ducks:	71	0	0	0	29	4
Shots fired (-):	658	259	34	38	890	95

Teams Results		
Score:	51	25
Deaths:	31	57

Resultados da Ronda 5

	Blue TEAM			Red TEAM		
	Player 1 Bellamy	Player 2 bm-teles	Player 3 samueljalmeida	Player 1 Mixa	Player 2 simaovertigo	Player 3 Samuray500
Time in game:	366.675 sec.	366.48 sec.	311.265 sec.	367.785 sec.	367.875 sec.	367.08 sec.
Properties						
Health:	100	6	100	100	45	100
Armor:	150	103	100	100	116	150
Stamina:	100	100	100	100	100	100
Weapon:	weapon_grenad	weapon_shotgu	weapon_grenad	weapon_shotgu	weapon_mp5	weapon_mp5
Actions						
Sprinting:	false	false	false	false	false	false
Jumping:	false	false	false	false	false	false
Ducking:	false	false	false	false	false	false
Instant shots:	0	0	0	0	4	0
Status						
Score:	7	9	2	3	2	7
Deaths:	5	4	8	8	6	7
Move keys						

Final Results						
Score:	7	9	2	3	2	7
Deaths:	5	4	8	8	6	7
Kill/death ratio:	1.4	2.25	0.25	0.375	0.33333333333	1
Sprints:	0	0	0	0	0	20
Jumps:	0	0	3	0	0	1
Ducks:	11	1	0	0	0	7
Shots fired (-):	325	42	14	31	9	559

Teams Results		
Score:	18	12
Deaths:	17	21

Resultados da Ronda 6

	Blue TEAM			Red TEAM		
	Player 1 Bellamy	Player 2 bm-teles	Player 3 samueljalmeida	Player 1 Mixa	Player 2 Samuray500	Player 3 simaovertigo
Time in game:	360.03 sec.	359.85 sec.	347.55 sec.	359.16 sec.	355.11 sec.	355.53 sec.
Properties						
Health:	54	19	100	30	5	36
Armor:	130	103	100	62	100	115
Stamina:	100	100	100	100	100	100
Weapon:	weapon_crowba	weapon_shotgu	weapon_pistol	weapon_shotgu	weapon_mp5	weapon_mp5
Actions						
Sprinting:	false	false	false	false	false	false
Jumping:	false	false	false	false	false	false
Ducking:	false	false	false	false	false	false
Instant shots:	0	0	0	0	0	0
Status						
Score:	12	21	3	11	10	4
Deaths:	9	8	16	13	10	19
Move keys						

Final Results

Score:	12	21	3	11	10	4
Deaths:	9	8	16	13	10	19
Kill/death ratio:	1.33333333333	2.625	0.1875	0.84615384615	1	0.21052631578
Sprints:	0	0	0	0	76	0
Jumps:	6	3	15	1	8	1
Ducks:	47	0	0	0	0	0
Shots fired (-):	583	272	27	38	721	79

Teams Results

Score:	36	25
Deaths:	33	42