Tese

Mestrado em Engenharia Informática

Mobile Ambient Gaming

Daniel Filipe André Barbosa
(dbarbosa@student.dei.uc.pt)

Orientadores
Professor Fernando Penousal Machado
Professor Daniel Castro Silva

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Setembro 2014

Resumo

Atualmente a sociedade faz bastante uso de *smartphones*, tanto que estes parecem passar despercebidos à vista humana. Estes dispositivos permitem realizar e auxiliar inúmeras tarefas no quotidiano das pessoas. Dentro destas tarefas, existem algumas bastante comuns como tirar fotografias/vídeos. Esta atividade de fazer vídeos vem ao encontro com o objectivo do projeto, uma vez que este tenta relacionar algumas temáticas como o *ambient gaming*, os dispositivos móveis, a realidade aumentada e a criatividade computacional. O objectivo principal deste projeto é a construção de uma aplicação para dispositivos móveis, que seja capaz de criar música através da detecção de formas geométricas elementares que estejam presentes num vídeo. Foi feito um estudo sobre dispositivos móveis e as suas características e também um estudo sobre realidade aumentada e as suas possibilidades, tendo-se verificado que existe uma limitação por parte do *hardware* dos dispositivos móveis que não permite realizar o projeto com a qualidade necessária. Devido a esta limitação, foi necessário remover a componente de realidade aumentada e passar a ser feito pós-processamento dos vídeos. Em relação à geração de música foi feito um estudo que levou à opção de criar um sequenciador, com vários *loops* e efeitos. No final, com o acesso às variáveis visuais e com a geração musical, foi possível criar composições vídeo-musicais. Também foi criado um servidor que suporta o jogo e todas as suas componentes: missões, *badges*, *rankings* e votações. Este jogo tem uma ligação direta com as composições, uma vez que estas são necessárias para que haja dinamismo. No final do projeto foi também feita uma comparação com outras aplicações semelhantes, e foram apontadas as diferenças e as inovações implementadas no projeto.

Keywords: *Ambient Gaming*, Dispositivos Móveis, Criatividade Computacional.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste capítulo é apresentado o contexto onde se aplica o projeto e a importância que este pode ter na sociedade, assim como os objetivos principais deste projeto. Também é feita uma simples explicação da estruturação do documento.

1.1 Contexto

No mundo de hoje, as pessoas fazem bastante uso de *smartphones*. Tanto que eles parecem já estar um pouco banalizados entre a população, uma vez que passam despercebidos à vista humana. Atualmente estes dispositivos possibilitam realizar as mais variadas tarefas, através das várias aplicações para eles desenvolvidas. Existem certas atividades que as pessoas têm que passam completamente despercebidas e que são realizadas com frequência na sociedade em que nos inserimos [1]. Estas atividades são por exemplo enviar mensagens, fazer chamadas, tirar fotografias/vídeos, etc. Aprofundando mais a área de tirar fotografias e fazer vídeos, perceber-se que sempre foi um aspecto que teve relevância para o utilizador pois sempre se tentou criar novos dispositivos móveis com câmaras melhoradas e com bastante qualidade. Daí muitas das aplicações famosas para este tipo de dispositivos sejam relacionadas com a fotografia/vídeo, como é caso do *Instagram*[1] e do *Vine*[2]. Estas duas aplicações são usadas constantemente por milhões utilizadores em todo o mundo [2] [3].

1.2 Motivação

A música ambiente está presente na sociedade há já algum tempo e é algo que tem impacto e mudou a maneira de estar das pessoas em algumas situações. O conceito de *ambient ga-*

[1] mais informações em: http://instagram.com/
[2] mais informações em: https://vine.co/

ming baseia-se na música ambiente pretendendo criar jogos integrados na vida das pessoas. Este projeto está relacionado com este conceito de *ambient gaming* [4][5][6].

Neste seguimento, e conjugando este ato cada vez mais natural com os conceitos de música ambiente e *ambient gaming*, surge a ideia de gerar uma composição musical para acompanhar as imagens ou vídeos capturados pelo utilizador. Esta composição, criada automaticamente com base nas características visuais da imagem ou vídeo capturado, poderá também ser partilhada e ter associada uma pontuação, criando assim um jogo integrado na vida das pessoas.

1.3 Objetivos

O objetivo principal deste projeto é a construção de uma aplicação móvel capaz de criar música através do reconhecimento de formas geométricas elementares (quadrados, círculos, etc.) presentes num vídeo. Este objectivo decompõe-se em vários sub-objetivos como: a análise de imagem/vídeo, a detecção de formas geométricas elementares, o mapeamento entre as variáveis visuais e auditivas e a geração de música.

Uma componente importante neste projeto é o conceito de *gamification*, ou seja, a tornar a aplicação num jogo de modo a esta fique mais apelativa. Este jogo pretende promover a partilha de composições vídeo-musicais e a interação entre utilizadores, através de um sistema de pontos, de *badges*, de missões e de votações.

Pretende-se construir um protótipo de uma aplicação que possa mais tarde vir a ser colocado no mercado mais precisamente no *Google Play*.

1.4 Estrutura

Este documento está dividido nos seguintes capítulos: o Estado da Arte, a Descrição do Sistema, o Planeamento, o Processo da Análise de Vídeo, a Geração Musical, o Desenvolvimento da Aplicação, os Resultados e as Conclusões.

No capítulo do Estado da Arte foram estudadas algumas temáticas importantes para o projeto como é o caso do *ambient gaming*, dispositivos móveis e criatividade computacional.

No capítulo da Descrição do Sistema foram realizados alguns artefatos capazes de des-

crever o projeto, as suas funções e a sua arquitetura.

No capítulo do Planeamento foram feitas escolhas tecnológicas e dividiu-se o projeto em tarefas e sub-tarefas.

O Processo da Análise de Vídeo é um capítulo que descreve a detecção de formas geométricas e as variáveis visuais que foram recolhidas através dos passos que foram feitos até se chegar a uma solução pretendida.

No capítulo da Geração Musical é descrito o processo que levou a criar uma solução capaz de gerar música, através das variáveis visuais recolhidas.

Em relação ao capítulo do Desenvolvimento da Aplicação é explicado todo o processo atual de como é feita uma composição vídeo-musical e como é feito o *upload* da mesma.

No capítulo dos Resultados são apresentados alguns testes e o efeito que tiveram na implementação do projeto.

Em relação às Conclusões é feita uma descrição de todos os aspectos relevantes sobre a realização de algumas tarefas e também do trabalho futuro.

Além destes capítulos existe ainda uma secção de anexos que contém informação sobre outros áreas que foram analisadas, como é o caso do Estudo da Realidade Aumentada e dos *Mockups*.

Capítulo 2

Estado da Arte

Neste capítulo é apresentado o estado da arte, sendo exploradas temáticas relacionadas com o trabalho desenvolvido, assim como trabalhos prévios realizados nestas áreas. Estes temas são *ambient gaming*, dispositivos móveis e criatividade computacional que vêm alinhados com o projeto em causa. Sobre *ambient gaming* é referido um pouco da história que está por detrás do tema, assim como a sua definição. No caso dos dispositivos móveis é apresentada a sua evolução ao longo do tempo, são avaliados alguns sistemas operativos e também são expostas algumas estatísticas para perceber a importância que têm. Em relação à criatividade computacional é apresentada uma definição que é complementada com algumas áreas de trabalho e exemplos.

2.1 *Ambient gaming*

A humanidade procurou desde sempre maneiras de entretenimento e os jogos foram sempre um ponto forte neste tema [7]. Os jogos evoluíram muito ao longo dos tempos desde jogos físicos, jogos de mesa, jogos de sorte, jogos de tabuleiro até aos videojogos. Existe uma indústria forte no que toca a jogos, especialmente nos videojogos e é nos videojogos que o *ambient gaming* se insere. Este é um género relativamente recente, mas, apesar disso, tem vindo a evoluir com o tempo[8][9][10].

2.1.1 História

O termo *"ambient gaming"* está de certa forma ligado ao termo "música ambiente". Brian Eno foi quem apresentou a música ambiente ao mundo com o álbum *"Ambient 1: Music for Airports"* em 1978. Este género musical tende a adaptar-se ao nível de atenção de cada pessoa, e "deve ser tão ignorável como interessante"[5]. Este tipo de música foi criado com o intuito de desviar a atenção da pessoa de uma determinada atividade, normalmente algo

aborrecido. Assim sendo, o estado de espírito da pessoa é alterado, fazendo com que a atividade em questão deixe de ser tão aborrecida [4][5][6].

A música ambiente é um ponto de partida para a criação de *ambient gaming*, visto que a sua definição serve de referência e de contexto para este género de jogos. Para definir *ambient gaming* é necessário, primeiro, perceber o conceito de videojogo. Este é, normalmente, uma forma de entretenimento, que pode conter personagens/elementos inseridos num ambiente, criados por um computador. No entanto, um videojogo pode não ter a conotação de entretenimento, caso se trate de um ***serious game***. Os ***serious games*** são jogos desenvolvidos com o intuito de resolver problemas, como por exemplo na educação, na saúde, em publicidade, na política, na religião, entre outros [11]. Os videojogos podem ser de dois tipos: *single player* ou *multiplayer*. Um jogo *single player* é jogado apenas por um jogador que tem como missão enfrentar todas adversidades propostas pelo computador. No caso dos jogos *multiplayer*, existem vários jogadores que podem cooperar ou competir no jogo [4].

É sabido que existem jogos que exigem mais esforço mental (ou intelectual) do utilizador. Estes jogos requerem ao utilizador mais esforço e mais tempo para que sejam jogados corretamente (por exemplo jogos de estratégia como o *Civilization* [1] e *Age of Empires* [2], ver figuras 2.1a e 2.1b). Então, diz-se que a curva de aprendizagem neste tipo de jogos é muito acentuada. No entanto existem jogos em que não é necessário dispender muito tempo, nem muito esforço para que sejam entendidos (por exemplo o *Tetris*, ver figura 2.2) e que têm uma curva de aprendizagem menos acentuada [4].

No passado, o utilizador encontrava-se a jogar apenas num local, visto que não existia a possibilidade de mover o computador. Com o avançar do tempo, este aspecto de mobilidade mudou com aparecimento dos dispositivos móveis. Apesar de não ser obrigatório o utilizador mover-se, pode deslocar-se no espaço. Este fator do jogador não estar fixo a um local, foi bastante importante na criação do conceito de *ambient gaming*, na medida em que o jogador pode mover-se em ambientes/locais rotineiros. Uma vez que estes jogos podem ser enquadrados com a rotina, o esforço é reduzido por parte do utilizador. Assim o jogador tem a possibilidade de escolher qual o nível de esforço que pretende ter com o jogo, dependendo da sua motivação [4].

[1] mais informações em: http://www.civilization.com/
[2] mais informações em: http://www.ageofempires.com/

(a) *Civilization V*

(b) *Age of Empires 3*

Figura 2.1: Jogos de estratégia

Figura 2.2: *Tetris*

2.1.2 Definição

Ambient gaming é um conceito que ainda não foi completamente definido. No entanto os autores que estudam esta área partilham uma visão relativamente comum sobre o tema.[4]

Segundo estes autores, um *ambient game* deve ser o menos intrusivo possível, permitindo ao jogador fazer a escolha entre dar muita ou pouca atenção ao jogo [12], à semelhança da música ambiente. O jogo tem a capacidade de estar ativo sem que o jogador tenha consciência disso e de recolher dados das ações realizadas pelo jogador, analisá-los e traduzi-los em informação necessária para que jogo prossiga. Isto faz com que as ações realizadas pelo jogador na vida real tenham um efeito direto no jogo. Assim, o jogador tem a possibilidade de seguir com a sua rotina ou alterá-la em prol do jogo [4].

Existem duas maneiras de implementar um *ambient game*. A primeira, consiste no jogador transportar o sistema com ele. A segunda, é utilizando um ambiente inteligente, ou seja, o sistema está embebido no ambiente que envolve o jogador. Esta segunda opção normalmente necessita de contexto como a localização, informação do jogador, entre outros. Em relação a tecnologias, estas devem ser tidas em conta no desenvolvimento de um ambiente inteligente, como por exemplo: a interconectividade, a inteligência artificial e a

proliferação de computadores. Com este tipo de tecnologias é necessário pensar em ubiquidade, transparência e inteligência, uma vez que existem muitos computadores a comunicar entre si, que devem tentar passar despercebidos no ambiente e serem capazes de responder e interagir com os utilizadores de uma forma usável [13][14]. Para promover estes aspectos de transparência e de usabilidade podem-se usar a voz ou gestos para comunicar com o sistema, que são métodos mais naturais para o ser humano [4]. Também se podem usar sensores contidos em objetos utilizados diariamente, como por exemplo roupas, paredes, móveis, entre outros, podendo até ser o próprio corpo do jogador o sensor [15][16][17] (ver as figuras 2.3 e 2.4).

Figura 2.3: *SmartShirt HexoSkin* [3], *t-shirt* com sensores.

Figura 2.4: *Kinect* [4] a usar o corpo humano como sensor.

Os jogos digitais, nos dias de hoje, podem ser integrados numa perspectiva espacial, temporal e social devido à tecnologia existente e à importância que as redes sociais têm hoje em dia. Assim é possível integrar os jogos nas atividades do dia-a-dia de maneira que os limites entre o jogo e o mundo real desapareçam [18]. Pode-se então inserir um mundo virtual no mundo real através de vários dispositivos, tendo assim um ambiente inteligente. Este ambiente pode ser capaz de perceber quem está presente, onde estão os utilizadores, o que estão a fazer, quando e porque é que o estão a fazer, ou seja, extrair um contexto deste ambiente [12].

Os *ambient games* estão incluídos numa outra categoria de jogos chamada *pervasive gaming*(ver figura 2.5).

[3]mais informações em: http://www.hexoskin.com/en
[4]mais informações em: http://www.xbox.com/pt-PT/Kinect

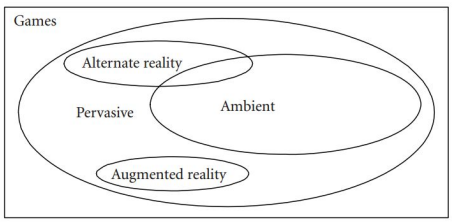


Figura 2.5: *Pervasive* e *Ambient Games* (adaptado de [4]).

Pervasive game é um jogo que tem lugar no mundo real, e tira partido de atividades normais que os jogadores têm no seu dia-a-dia. Neste tipo de jogos as ações efetuadas no mundo real têm efeito no mundo virtual, e vice-versa [19]. Dentro dos *pervasive games* existem mais tipos de jogos, como é o caso da realidade aumentada e da *alternate reality*.

A realidade aumentada (ver capítulo A) tira partido do local e do raio de visão do utilizador para acrescentar informação ao mundo real. Esta, normalmente, requer o transporte do equipamento necessário, o que exige esforço ao utilizador. A *alternate reality* costuma ser composta por puzzles, pistas, telefonemas, charadas, etc. que têm a capacidade de envolver o jogador com a história, com outros personagens (reais ou fictícios) e com o mundo real. Um exemplo disto é o *Perplex city* (figura 2.6) que é um jogo de *alternate reality* baseado em puzzles e pistas, disponibilizados em cartas, *websites*, *podcasts*, *emails*, textos ou até em eventos ao vivo.

Figura 2.6: *Perplex City* [5].

A *alternate reality* pode estar contida nas mais diversas áreas, como é o caso da música, mais especificamente no álbum *"Year Zero"* de *Nine Inch Nails* (figura 2.7) que fazia parte de um jogo composto por uma série de enigmas que eram encontrados em *flashdrives* que estavam em casas de banho nos concertos, *flyers*, *t-shirts*, *websites*, números de telefone

[5] mais informações em: http://perplexcity.com/

contidos em músicas por realizar e em paredes. Este jogo contava uma história, passada em 2022, contra o governo dos Estados Unidos da América. O próprio *CD* (figura 2.8) acaba por fazer parte do jogo uma vez que este depois de aquecido no leitor de *CD's* passa de preto para branco, revelando assim algumas pistas. Os *alternate reality games* estão muito próximos dos *ambient games*, mas necessitam de mais esforço por parte do utilizador visto que não fazem uso de ações rotineiras.[4]

Figura 2.7: *Year Zero* foi o álbum de *Nine Inch Nails* [6]que iniciou um jogo de *alternate reality*.

Figura 2.8: *CD* do álbum *Year Zero*.

Os *ambient games* deixam de ser orientados a um objectivo e passam a ser mais uma experiência divertida e por isso diz-se que existe uma ligação entre os conceitos de *ambient gaming* e de *gamification* pois ambos tentam tornar certas atividades divertidas. A *gamification* passa por introduzir um jogo numa aplicação normal de maneira a torná-la mais cativante para o utilizador. Os jogos tradicionais normalmente têm objetivos para se cumprir e utilizam mecanismos de recompensa e de punição. No entanto existem estudos psicológicos que provam que estes métodos podem deixar de ser eficazes se forem usados durante muito tempo e em demasia. Estes mecanismos devem ser utilizados eficientemente ou simplesmente utilizar outro método, como por exemplo *open-ended* ou *free-play* em que

[6]mais informações em: http://nin.com/

o objectivo é apenas a experiência em si e a interação social [12]. Um exemplo prático de um jogo em que o objectivo é a experiência pode ser o *Grand Theft Auto* [7] (figura 2.9) uma vez que o jogador pode ignorar as missões e fazer exatamente o que pretende: andar de carro, explorar o mapa, entre outros.

Figura 2.9: *Grand Theft Auto 5.*

O *ambient gaming* é um género de jogo que é capaz de oferecer uma experiência contínua, ao contrário dos jogos de consola em que o jogador faz um uso isolado e discreto. Isto é possível porque os *ambient games* coexistem com o mundo real e por isso têm a tendência de influenciar as ações realizadas pelo jogador, as suas emoções e até mesmo alterar a percepção que este tem do mundo real. Os jogadores normalmente obtêm *feedback* através de um avatar que pode ser controlado através de gestos, voz (ver figura 2.10), movimento (ver figura 2.11), batimento cardíaco, respiração, etc. Podem existir outros tipos de *feedback* através de outros sentidos, que não a visão, como por exemplo gerir temperaturas (ver figura 2.12), cheiros (ver figura 2.13), entre outros. Como estes jogos conseguem mudar a maneira como as pessoas reagem com o mundo real podem-se tornar úteis para melhorar a produtividade no trabalho, mudar estilos de vida, etc. Assim, é possível mudar o pensamento da sociedade uma vez que estes jogos podem ser sociais pois não limitam o número de jogadores [4]. É também importante ter em conta o controlo de privacidade do jogador, pois este tipo de jogo normalmente requer contexto. Sendo assim, o jogador deve ter a possibilidade de controlar a sua privacidade e de ter a consciência de que esta pode ser afetada [12].

[7] mais informações em: http://www.rockstargames.com/grandtheftauto/
[8] mais informações em: http://www.singstar.com/
[9] mais informações em: http://www.nintendo.com/wii
[10] mais informações em: http://www.sony.com/
[11] mais informações em: http://pt.playstation.com/psmove/
[12] mais informações em: http://sensoryacumen.com

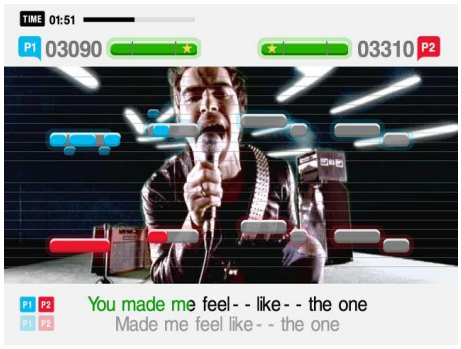


Figura 2.10: *SingStar* [8] utiliza a voz do jogador.

Figura 2.11: *Wii* [9] utiliza os movimentos do jogador para controlar o jogo.

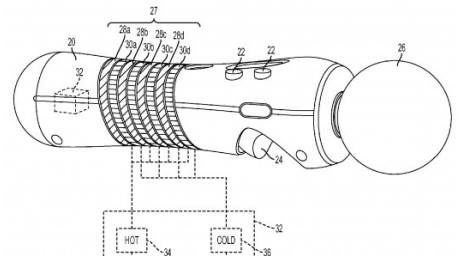


Figura 2.12: *Sony* [10] tem a patente que permite gerir a temperatura nos comandos da *PlayStation 3 Move* [11] [20].

Figura 2.13: *Game Skunk* [12] permite criar jogos com uso de recursos olfactivos [21].

2.2 Dispositivos Móveis

Antigamente os dispositivos móveis eram considerados um luxo, mas hoje em dia são bastante comuns, tão comuns que passam despercebidos entre a população [1]. Atualmente os dispositivos móveis estão cada vez mais próximos dos utilizadores, devido às variadas funções que conseguem desempenhar. Estes dispositivos são considerados, por algumas pessoas, parte de uma lista de objetos indispensáveis para o seu dia-a-dia, assim como as chaves, a carteira, entre outros. Quando são esquecidos, estas pessoas sentem-se desconfortáveis pois os dispositivos móveis transmitem-lhes uma sensação de estarem constantemente conectados ao mundo, ou seja, têm um cariz fortemente social [22].

Existe também uma ligação física e emocional entre os utilizadores e os dispositivos móveis. A ligação física é estudada pelas marcas, uma vez que seguem a moda para criar o design dos dispositivos móveis [1][22]. A possibilidade de os utilizadores poderem personalizá-los reforça esta ligação. A personalização passa por capas de diversos feitios, tons de toque, fundos de ecrã, entre outros [22]. A ligação emocional está relacionada com o fator social presente nos dispositivos móveis, ou seja, estes deixam de ser apenas dispositivos móveis e passam a ser algo mais [23][24][25]. Estes são, provavelmente, a tecnologia mais usada no dia-a-dia e torna-os quase como um companheiro, e daí o desconforto quando são perdidos ou esquecidos [26][27].

2.2.1 Evolução

Durante a Segunda Guerra Mundial houve uma grande pressão para se construir uma forma de comunicação portátil. Houveram grandes evoluções nesta área, mas os dispositivos continuavam a ser pouco portáteis, até que em **1973** a *Motorola* [13] mostrou ao mundo o primeiro telemóvel *handheld* [28]. Tinha-se então passado do "0G"para o "1G". Durante os **anos 80** continuou-se a apostar nesta tecnologia principalmente nos países nórdicos como a Suécia, a Dinamarca, Finlândia e Noruega.

- Em **1991** entrou-se na era "2G"através da primeira rede *GSM* (*Global System for Mobile Communications*) desenvolvida pela operadora finlandesa *Radiolinja* [29].
- Em **1992** apareceu o *Nokia* [14] *1011* que foi o primeiro telemóvel produzido em massa que utilizava as redes *GSM* [30].

[13] mais informações em: www.motorola.com
[14] mais informações em: http://www.nokia.com/global/

- Em **1993** a *IBM* [15] produziu o primeiro *smartphone* denominado *Simon*.
- Em **1997** a *Siemens* [16] lançou o modelo *S10* que foi o primeiro a ter um ecrã a cores.
- Em **1999** a *Nokia* construiu o modelo *3210* que veio a ser um dos modelos de telemóveis mais populares de sempre, com vendas superiores a 160 milhões de dispositivos. A *Samsung* [17] criou o primeiro telemóvel com a capacidade de tocar músicas em *mp3*. Também apareceu o primeiro telemóvel com GPS integrado, desenvolvido pela *Benefon*. Neste mesmo ano a *Nokia* ainda implementou no modelo *7110* um *browser* WAP(*Wireless Access Point*) em que era possível aceder a algumas aplicações *web*, como por exemplo o *e-mail*.
- Em **2000** a *Sharp* [18] criou o modelo *J-SH04* que tinha a primeira câmara fotográfica, apenas com 0.1 *megapixels*.
- Em **2001**, passado 10 anos desde aparecimento "2G"foi então desenvolvido o "3G". No mesmo ano a *Ericson* [19] lançou o modelo *T39* que foi o primeiro a ter *bluetooth*.
- Em **2007** a *Apple* [20] lançou o *iPhone* que veio revolucionar o mundo dos *smartphones*.
- Em **2008** a *Google* [21] entrou no mercado com o *Android* [22] que tinha a vantagem de ser *open-source*.
- Em **2010** foi criado o *TerreStar* [23] *Genus* e foi o primeiro *smartphone* que usa comunicações por satélite. O *LG* [24] *Optimus 2x* detém um *record* do *Guiness* por ser o primeiro *smartphone* a ter um processador *dual-core*. Também neste ano surgiu o sistema operativo *Windows Phone* [25] que era o sucessor do *Windows Mobile*.
- Em **2011** surgiu o "4G"com o *Android HTC* [26] *EVO 4G*. No mesmo ano a *LG* lançou o *Optimus 3D P920* que foi o primeiro *smartphone* em 3D, que permitia ver e partilhar conteúdos em 3D sem o auxílio de óculos.

Em relação aos *tablets* [31], os seus primeiros desenvolvimentos começaram em **1985** com computadores que usavam canetas como método de interação. Em **1990** a *Microsoft* [27]

[15] mais informações em: http://www.ibm.com/us/en/
[16] mais informações em: http://www.siemens.com/entry/cc/en/
[17] mais informações em: http://www.samsung.com/pt/
[18] mais informações em: http://sharp-world.com/
[19] mais informações em: http://www.ericsson.com/pt
[20] mais informações em: http://www.apple.com/
[21] mais informações em: https://www.google.com
[22] mais informações em: http://www.android.com/
[23] mais informações em: http://www.terrestarnetworks.com/
[24] mais informações em: http://www.lg.com/pt
[25] mais informações em: http://www.windowsphone.com/pt-pt
[26] mais informações em: http://www.htc.com/pt/
[27] mais informações em: http://www.microsoft.com/pt-pt/default.aspx

adaptou isto ao sistema operativo *Windows* [28]. No entanto só em **2002** é que a *Microsoft* inventou os *tablets* como são conhecidos hoje em dia [32]. Em **2010** apareceu o famoso *iPad* e um ano mais tarde apareceram os *tablets* com o sistema operativo *Android*. Em **2012** a Microsoft reinventou os seus *tablets* e surgiu então o *Microsoft Surface* [30].

2.2.2 Sistemas Operativos

Atualmente existem muitos sistemas operativos para dispositivos móveis como o *Android*, o *iOS*, o *Windows Phone*, o *BlackBerry* [29], entre outros. Por isso é importante ter em conta qual escolher quando se vai desenvolver para estes dispositivos. Nesta decisão é tido em conta, normalmente, o público-alvo, os conhecimentos dos programadores (linguagem, ferramentas, etc.), a documentação de apoio, o custo das licenças e de equipamentos, entre outros.

De todos os sistemas operativos, existem dois que são os preferidos para se desenvolver aplicações: o *Android* da *Google* e o *iOS* da *Apple*. Neste sub-capítulo é feita a comparação entre estes dois, uma vez que estes representam mais público alvo como é referido na secção 2.2.3, além disto existem mais *frameworks* de apoio para estes dois sistemas operativos [33]. Existem alguns autores que também fizeram uma comparação entre *Android* e *iOS* e chegaram a algumas conclusões [34].

Os autores Goadrich e Rogers [35] concluíram que desenvolver aplicações para *iOS* pode-se tornar mais difícil, pois é necessário equipamento e ferramentas específicos, e para além disso, a linguagem de programação é pouco comum.

Tracy [36] verificou que as maiores diferenças entre estes dois sistemas operativos passam pela interface visto que o *iOS* não possui um botão físico de retroceder e por isso é necessário incluir um botão de retroceder na aplicação.

Ribeiro et al. [37] concluíram que os pontos fortes do *Android* são: a linguagem utilizada ser *Java* pois é mais utilizada que *Objective-C*; existirem mais funcionalidades no emulador de *Android*; e a licença ser livre o que permite mais e melhores aplicações. Em relação ao *iOS* as vantagens são: o *xCode* [30] (ferramenta de trabalho para desenvolver as

[28] mais informações em: http://windows.microsoft.com/pt-pt/windows/home
[29] mais informações em: http://pt.blackberry.com/
[30] mais informações em: https://developer.apple.com/Xcode/

aplicações) que apresenta mais funcionalidades; e existirem menos problemas com compatibilidade entre hardware e aplicações pois a variedade de dispositivos é menor.

De uma maneira mais detalhada, para se desenvolver para *iOS* é necessário um computador *Apple Macintosh* que consiga correr o sistema operativo *Mac OS X 10.6 (Snow Leopard)*. Este fator é um impedimento muito forte para a criação de novas aplicações para *iOS*. Apesar disto, grande parte do trabalho pode ser realizado no computador, uma vez que o simulador consegue realizar a maior parte dos testes mesmo que se tratem de testes de GPS, câmara, acelerómetro, *multi-touch*, entre outros. É claro que este tipo de interação não é natural pois não existe toques mas sim cliques de rato. As aplicações para *iOS* são escritas no *xCode*, que permite especificar para que dispositivo é que o projeto é direcionado (*iPhone* ou *iPad*). Este também permite fazer *debug*, mas para esse efeito existe um programa mais sofisticado para fazer isto chamado *Instruments*. Para ajudar a criar aplicações existem *frameworks* como é o caso da *Cocoa Touch* [31], que permite criar elementos da interface, organiza o código, entre outros. O código é escrito em *Objective-C* que, por questões de performance, não tem *garbage colector*, obrigando assim o programador a fazer a gestão da memória [34][35].

Ao contrário do *iOS*, o *Android* não está limitado a um sistema operativo. Este necessita de um computado que tenha, no mínimo, *Windows XP*, *Mac OS X (10.5.8)* ou *Linux* (*kernel* 2.6). Mais uma vez, grande parte dos testes podem ser realizados com o simulador; além disso, é possível escolher vários parâmetros como a resolução, memória, processador, entre outros. No que toca a desenvolver aplicações para *Android*, o IDE mais comum é o *Eclipse* [32]. A interface de uma aplicação consiste basicamente em *XML (eXtensible Markup Language)* , mas também é possível usar o *Eclipse* ou até outro programa para desenhar a interface de uma forma mais visual [34][35].

O custo de desenvolvimento para o *iOS* pode ser muito elevado em relação ao *Android*, pois o computador necessário e as licenças para desenvolver aplicações pode chegar a ser o triplo do preço. No que toca a dispositivos passa-se o mesmo, visto que os *iPhone's* e os *iPad's* têm custos mais elevados do que os dispositivos *Android*. Isto porque existem mais fabricantes para *Android* e assim é possível escolher mais facilmente a gama dos *smartphones/tablets* pretendida [34].

[31] mais informações em: https://developer.apple.com/technologies/ios/cocoa-touch.html
[32] mais informações em: http://www.eclipse.org/

Depois de concluída a aplicação, é necessário partilhá-la e por isso existe a *AppStore* [33] e o *Google Play* [34]. A *AppStore* é a única plataforma de distribuição para *iOS* e é um pouco rígida, visto que é necessário ter uma licença que tem de ser paga anualmente (99$). Para além disto a aplicação só pode ser testada num determinado número de dispositivos. O *Google Play* já tem alguma concorrência por parte de outras plataformas e o utilizador é livre de testar a sua aplicação em quantos dispositivos desejar. Também necessita de uma licença mas neste caso esta é paga apenas uma vez (25$) [34].

Apesar de um dispositivo *Android* ter um custo mais baixo, por existirem várias marcas a fabricar dispositivos, este prende-se com um problema pertinente. Este problema está relacionado com a compatibilidade entre a aplicação e o dispositivo, visto que existem muitos aparelhos e cada aparelho tem a sua resolução de ecrã. Isto muitas vezes é resolvido com o simulador do *Android*, em que se pode testar a aplicação em várias resoluções [34].

Existe um aspecto muito importante que é necessário ter em conta para se desenvolver uma aplicação que é a documentação de apoio. Neste aspecto os dois sistemas operativos estão repletos de informação estruturada e organizada, o *Android* tem o *Android Developers* [35] e o *iOS* tem o *iOS Dev Center* [36], onde se podem encontrar exemplos, guias de boas práticas, entre outros. Uma parte fundamental desta documentação passa pelas comunidades específicas de desenvolvimento que apresentam muitas soluções para resolver os problemas dos programadores nesta área [34].

2.2.3 Estatísticas de utilização

Como já foi referido anteriormente, existem dois sistemas operativos para dispositivos móveis que lideram o mercado: o *Android* e o *iOS*. Num estudo feito pela *International Data Corporation* [38], estes dois sistemas operativos juntos equivalem a 91.1% do mercado (tabela 2.1 e figura 2.14). Já em 2011 estes representavam 68,1% dos 494,5 milhões de dispositivos vendidos e em 2012, na mesma altura, estes tinham 87,6% das 722,4 milhões de vendas (tabela 2.2). Isto mostra tanto o crescimento destes dois sistemas operativos como também o crescimento dos *smartphones/tablets*. Durante o ano de 2012, foram então vendidos 497,1 milhões de aparelhos *Android* e 135,9 milhões de unidades de *iOS*. O *Android* teve um crescimento de 104,1% em relação a 2011 enquanto que o *iOS* teve um crescimento de 46,0%.

[33] mais informações em: http://store.apple.com/pt
[34] mais informações em: https://play.google.com/
[35] mais informações em: http://developer.android.com/index.html
[36] mais informações em: https://developer.apple.com/devcenter/ios/index.action

Sistema Operativo	*Market Share* último trimestre de 2011	*Market Share* último trimestre de 2012
Android	52.9%	70.1%
iOS	23.0%	21.0%
BlackBerry	8.1%	3.2%
Windows Phone	1.5%	2.6%
Linux	2.4%	1.7%
Outros	12.1%	1.3%
Total	100.0%	100.0%

Tabela 2.1: *Smartphone Market Share*

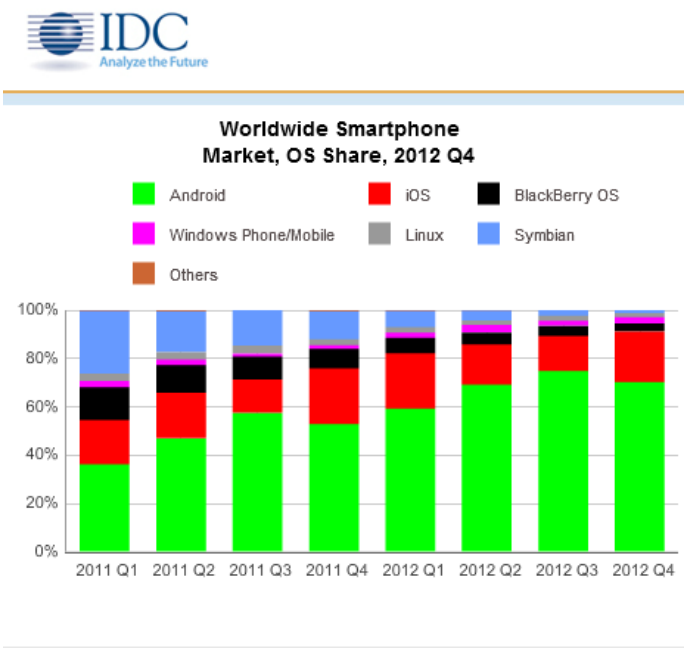


Figura 2.14: *Smartphone Market Share*

Sistema Operativo	Unidades vendidas em 2011	Unidades vendidas em 2012	Crescimento de 2011 para 2012
Android	243.5	497.1	104.1%
iOS	93.1	135.9	46.0%
BlackBerry	51.1	32.5	-36.4%
Symbian	81.5	23.9	-70.7%
Windows Phone	9.0	17.9	98.9%
Outros	16.3	15.1	-7.4%
Total	494.5	722.4	46.1%

Tabela 2.2: Crescimento dos sistemas operativos de 2011 para 2012

Em relação aos *downloads* das aplicações feitos pelos utilizadores pode-se verificar que os sistemas operativos *Android* e *iOS* continuam superiores ao resto, como se pode ver na

figura 2.15. Foram feitos 51% de *downloads* de aplicações no *Google Play*, o que significa que existem mais *downloads* feitos para *Android*. O seu concorrente direto, o *iOS*, tem cerca de 40%, ou seja, não está muito distante do *Android*. Os restantes 9% equivalem aos *downloads* feitos para *BlackBerry* e *Windows Phone* [39][40]. No total foram feitos 13.4 mil milhões de *downloads* [39].

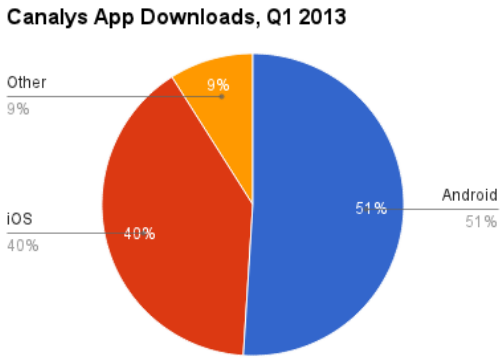


Figura 2.15: *Downloads* de aplicações no primeiro trimestre de 2013 [40].

2.3 Criatividade Computacional

A criatividade é uma característica que é atribuída geralmente aos seres humanos. Mas esta também pode ser realizada por sistemas artificiais que foram programados para simular a criatividade exercida pelos humanos.

2.3.1 Definição

A criatividade computacional também é conhecida por criatividade artificial ou por criatividade mecânica. Este é um conceito que tenta misturar as áreas da inteligência artificial, da psicologia cognitiva, da filosofia e das artes, com o intuito de simular a criatividade humana através de um computador. Por isso é necessário compreender o que é a criatividade e como é que ela se relaciona com os computadores.

Newell, Shaw e Simon [41], chegaram à conclusão que nenhuma definição era suficientemente completa para descrever a criatividade, então elaboraram quatro critérios para perceber se uma solução é de facto criativa. Os quatro critérios são: a solução ser original

e útil; ser uma solução que nos faça rejeitar ideias que já tinham sido aceites anteriormente; ser uma solução resultante da motivação e persistência; e ser uma solução que surge para clarificar um problema que inicialmente era vago. Para além disso, os programas desenvolvidos não têm que ser criativos podem ser simplesmente para melhorar a criatividade do humano que os está a utilizar.

A criatividade pode ser dividida em duas categorias segundo Margaret Boden [42][43]. Pode ser *P-creativity* quando a solução é original apenas para o agente que a produz, ou pode ser *H-creativity* se a solução for original também para a sociedade. Além disto, a criatividade pode ainda ser distinguida entre exploratória e transformacional. A criatividade exploratória acontece quando a solução surge através da exploração de um determinado espaço conceptual, enquanto que a transformacional tenta transformar ou transcender o espaço conceptual.

2.3.2 Áreas de trabalho

A criatividade computacional pode ser aplicada em muitas áreas como por exemplo na criatividade linguística, na criatividade visual e artística, na criatividade musical, etc.

No que toca à criatividade linguística, esta pode incluir a geração de histórias como é o caso do projeto *Tale-spin* [44] que entendia que as histórias tinham o objectivo de resolver problemas, e neste sentido este projeto criava histórias com um determinado problema que as personagens eram obrigadas a resolver. Outro exemplo de criatividade linguística é a criação de metáforas, que está presente num sistema chamado *Sardonicus*. Este sistema acede a uma base dados com várias comparações que são analisadas posteriormente para que lhes sejam dadas uma conotação. Estas comparações, depois deste processo, são utilizadas por outro sistema chamado *Aristotle* [45] que consegue sugerir metáforas dado um *input*. Nesta área também da linguística também existe a criação de piadas, como é o caso do projeto *HAHAcronym* [46] que tira partido da linguagem natural quando esta gera desentendimento. A poesia também está englobada nesta área da linguística e requer certos cuidados nas rimas, no sentido do texto, na estrutura, entre outros. Neste campo existe um projeto chamado *Aspera* [47] que recebe como *input* de alguns fragmentos de texto que têm associados significados. Depois estes fragmentos são organizados e é construído um poema que esteja de acordo com os cuidados referidos anteriormente.

Na área da criatividade artística existe por exemplo o sistema *Aaron*, desenvolvido por Harold Cohen, que permite a criação de imagens a preto e branco ou a cores, abstratas

ou não. Estas imagens podem ser figuras humanas, rochas, plantas, entre outros objetos. Algumas criações deste sistema já foram expostas e reconhecidas em galerias de renome [48].

No que toca à criatividade musical existem vários propósitos para a sua criação pois pode ter o objectivo de ser usada para ajudar um determinado músico ou pode simplesmente ser usada como trabalho final em que o artista acaba por ser o computador. David Cope [49] criou um sistema chamado *EMI*[50] que analisava música criada por humanos e a partir dessa tentava criar música dentro do mesmo estilo musical. Na área da música clássica foi criado o *Iamus* que foi o primeiro computador capaz de criar música sem receber qualquer *input*. Uma das criações deste projeto foi interpretada pela Orquestra Sinfónica de Londres [40].

Outro tipo de criatividade musical é música visual que usa características visuais para gerar música ou vice-versa. Os primeiros passos nesta área foram dados pelo Louis Bertrand Castel por volta de 1730 quando este construiu um cravo com a capacidade apresentar cores quando uma nota era tocada. Posteriormente foram realizadas algumas experiências com outros instrumentos, neste caso um órgão (ver figura 2.16). Este órgão tinha a capacidade de apresentar uma cor que correspondia à nota que tinha sido tocada [51][52].

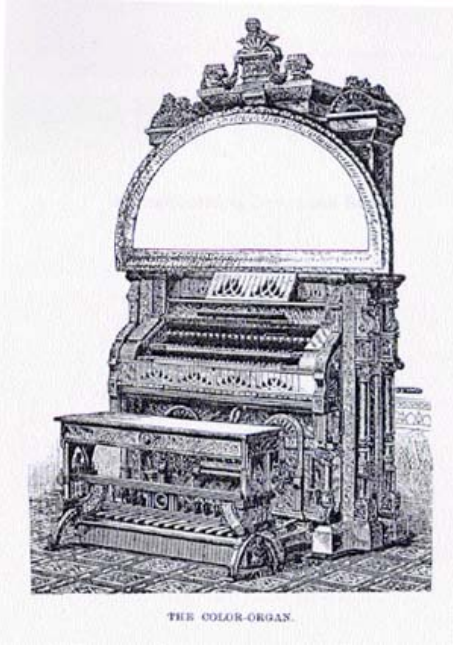


Figura 2.16: *Colour Organ* usado para a música visual

Mais tarde, com acesso a computadores foi possível expandir os horizontes e começar a criar música a partir de características visuais. Com isto começaram a aparecer aplicações

como o *MetaSynth* [37] (ver figura 2.17) que era capaz de gerar música a partir de imagens ou até mesmo de desenhos feitos com as ferramentas próprias da aplicação. Esta aplicação tinha em conta as cores que estavam contidas na imagem. Outra aplicação é o *AudioPaint*[38] (ver figura 2.18), que através de uma imagem de *input* tem a capacidade de analisar cada pixel e através da sua cor e da sua posição é capaz de criar uma música.

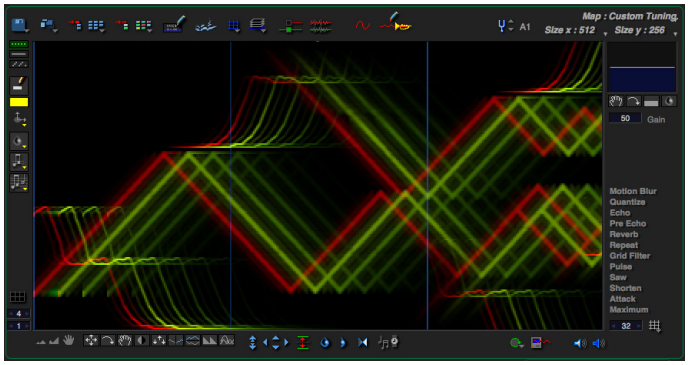


Figura 2.17: *MetaSynth*

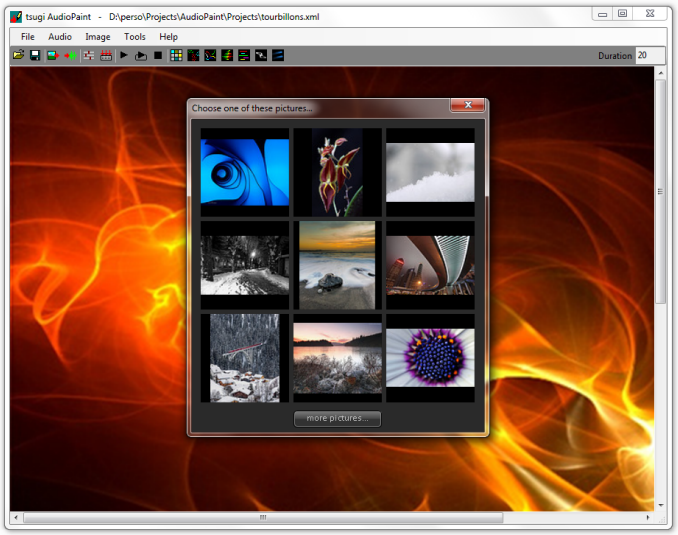


Figura 2.18: *AudioPaint*

Atualmente existem algumas aplicações móveis que são capazes de receber um *input* visual e transformá-lo num *output* musical. Um exemplo disto é o *Virtual ANS* [39] (ver figura 2.19) que permite ao utilizador fazer desenhos e inserir objetos para que a estes possam ser analisados pela aplicação com a finalidade de gerar som. Neste caso o formato dos

[37] mais informações em: http://www.uisoftware.com/MetaSynth/index.php
[38] mais informações em: http://www.nicolasfournel.com/audiopaint.htm
[39] mais informações em: http://www.warmplace.ru/soft/ans/

objetos têm importância assim como a posição onde estes estão colocados. Além disto a aplicação permite alterar algumas variáveis musicais, como por exemplo as batidas por segundo. A aplicação está disponível para *Android* e para *iOS*, e também para *Windows*, *Linux* e *OSX*. Outra aplicação móvel é a *Audible Ink* [40] (ver figura 2.20) que recebe como *input* uma imagem, analisa as cores que estão contidas na mesma e retorna como *output* uma música. Esta está disponível apenas para *iOS*.

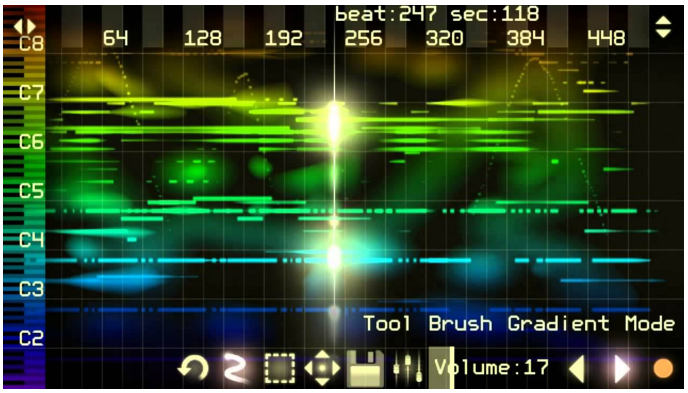


Figura 2.19: *Virtual ANS*

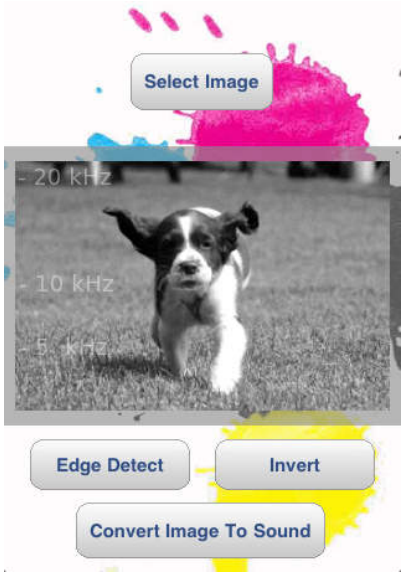


Figura 2.20: *Audible Ink*

[40] mais informações em: https://itunes.apple.com/us/app/audible-ink/id331533183?mt=8

Capítulo 3

Descrição do Sistema

Neste capítulo é descrito o jogo, e assim como os requisitos principais do projeto através de *user stories* e de casos de uso. Também é feita uma análise sobre a arquitetura do sistema de maneira a descrever os diversos componentes do projeto.

3.1 Jogo

O jogo está contido, em grande parte, no servidor, uma vez que é o servidor que está em condições de poder realizar as operações necessárias para que todas as componentes do jogo funcionem.

A maior parte da componente de jogo, neste caso *ambient gaming*, é realizada no servidor. No entanto, esta componente também existe na aplicação móvel apesar de acabar por passar despercebida, o que acaba por ir ao encontro dos princípios do *ambient gaming* uma vez que o utilizador pode escolher entre dar muita ou pouca atenção ao jogo.

O jogo é composto por várias componentes: acumulação de pontos por parte do jogador; *rating* que o jogador obtém nas composições; missões diárias que aparecem e às quais o jogador poderá dar resposta para as completar; e *badges* que o utilizador poderá tentar obter.

Em relação aos pontos existe um sistema de *ranking* entre os utilizadores em que aqueles que tiverem mais pontos têm uma classificação melhor que os que tiverem menos. Este sistema de pontos tem a funcionalidade motivar a competitividade entre utilizadores.

O *rating* corresponde à votação que os utilizadores podem atribuir às várias composições. Esta votação pode ter valores entre uma e cinco estrelas, em que uma estrela significa

que a composição não foi bem conseguida e cinco estrelas corresponde a uma composição muito bem conseguida. Este *rating* é uma componente que é muito subjectiva uma vez que depende dos gostos e interesses dos utilizadores que estão a avaliar as composições.

No entanto o facto de ser subjectivo não implica que seja uma desvantagem uma vez que isto pode trazer mais dinamismo ao jogo. Além disto, esta componente está diretamente associada aos pontos, ou seja, por cada votação feita o criador da composição recebe pontos consoante as estrelas recebidas. A pontuação é dada de acordo com a seguinte tabela 3.1.

Estrelas	Pontuação
1	1
2	2
3	4
4	8
5	16

Tabela 3.1: Votações e os respectivos pontos.

As missões são um aspecto importante no jogo, pois são um dos factores que cativa os utilizadores a jogarem todos os dias. Estas missões são diárias e requerem que o utilizador efetue um determinado procedimento para ganhar pontos, mais concretamente vinte e cinco pontos. Estas missões têm o objectivo de dinamizar a aplicação, no que toca a enviar e a visualizar composições e fazer o *rating* das composições dos outros utilizadores. Neste momento existe uma lista limitada de missões que todos os dias é escolhida uma aleatoriamente para cada utilizador. Dentro desta lista de missões o utilizador poderá ter que fazer *upload* de composições com um certo número de formas, ter que ganhar um determinado número de pontos através do *rating* de uma composição de que tenha feito *upload* naquele dia, fazer *upload* ou *rating* de composições, entre outras. Mais uma vez, estas podem ser ignoradas pelos utilizadores caso eles pretendam.

Por fim os *badges* são uma recompensa pelo bom desempenho do utilizador e têm o objectivo de promover a interação do utilizador com a aplicação. O utilizador tem à sua disponibilidade alguns objetivos e quando estes são cumpridos é lhe atribuído um prémio, neste caso uma medalha digital. As medalhas têm um título, uma descrição do objectivo e uma imagem (as imagens foram desenvolvidas pela equipa de design do CDV [1]), e devem ser o mais chamativas possíveis para que o utilizador tenha vontade de cumprir o objec-

[1] mais informações em: http://cdv.dei.uc.pt/people/

tivo. Existe também uma lista limitada de *badges* com os mais variados objetivos, entre eles criar uma conta na aplicação para motivar e para mostrar ao utilizador que existe esta possibilidade de ganhar *badges*, fazer *upload* da primeira composição, ter uma composição com cinco estrelas, ter um certo número de pontos, ter todas as formas geométricas numa única composição e também ter uma composição com uma estrela. Este último objectivo não promove o bom desempenho como foi mencionado em cima, mas pode motivar um utilizador que tenha tido um mau resultado a continuar a enviar as suas composições até ter um *rating* melhor.

3.2 *User Stories*

Os *user stories* definem de uma forma simples os requisitos do projeto. Para isso é feita uma descrição simples dos requisitos do ponto de vista da pessoa pretende utilizar a aplicação/jogo. Normalmente é usada uma estrutura capaz de clarificar a descrição:

Enquanto <tipo de utilizador>
Quero <ação a executar>
De forma a <proposta de valor>

Esta estrutura permite perceber quem é a pessoa que está a usar a aplicação/jogo, qual a função que pretende realizar e porque é que a pretende realizar.

User story 1:

Enquanto utilizador **quero** capturar um novo vídeo ou escolher um vídeo já existente **de forma a** criar uma nova composição vídeo-musical.

User story 2:

Enquanto utilizador **quero** reproduzir as minhas composições vídeo-musicais **de forma a** revê-las.

User story 3:

Enquanto utilizador **quero** partilhar a composição vídeo-musical **de forma a** que outras pessoas possam visualizá-la.

User story 4:

Enquanto utilizador **quero** pausar a reprodução da composição vídeo-musical **de forma a** fazer uma pausa ou aceder às opções do vídeo.

User story 5:

Enquanto utilizador **quero** voltar à reprodução da composição vídeo-musical que tinha sido pausada **de forma a** continuar a visualizá-la.

User story 6:

Enquanto utilizador **quero** apagar a composição vídeo-musical **de forma a** removê-la do meu dispositivo móvel pois não a pretendo manter.

User story 7:

Enquanto utilizador **quero-me** autenticar **de forma a** ter acesso à aplicação.

User story 8:

Enquanto utilizador **quero** atribuir uma pontuação a uma composição vídeo-musical **de forma a** avaliá-la.

User story 9:

Enquanto utilizador **quero** ver a pontuação de uma composição vídeo-musical **de forma a** perceber se está bem classificada.

User story 10:

Enquanto utilizador **quero** ver o *ranking* das pontuações dos utilizadores **de forma a** comparar com a minha pontuação.

User story 11:

Enquanto utilizador **quero** ver as missões que tenho disponíveis **de forma a** ganhar pontos a cumpri-las.

User story 12:

Enquanto utilizador **quero** ver os *bagdes* que tenho **de forma a** visualizar a minha coleção.

3.3 *Casos de uso*

Os casos de uso foram usados para definir os requisitos do projeto através dos objetivos, das pré-condições, das pós-condições, do fluxo de eventos primário, do fluxo de eventos secundário e prioridade.

Nome	**Objectivo**	**Pré-condições**	**Pós-condições**	**Fluxo de eventos principal**	**Fluxo de eventos secundários**	**Prioridade**
Capturar um vídeo.	Permitir que o utilizador possa capturar um vídeo.	Ter espaço suficiente em disco para capturar o vídeo.	A composição vídeo-musical fica guardada e o espaço em disco diminui.	O utilizador indica que quer capturar um vídeo. A aplicação apresenta imagens vindas da câmara. O utilizador indica que quer iniciar a captura. A aplicação inicia a captura. O utilizador indica que quer acabar captura. A aplicação pára a captura. A aplicação cria a composição vídeo-musical, e reproduz a mesma quando acabada.		*Must*
Compor um vídeo já existente.	Permitir que o utilizador possa compor um vídeo já existente no dispositivo.	Ter espaço suficiente em disco para criar a composição vídeo-musical.	A composição vídeo-musical fica guardada e o espaço em disco diminui.	O utilizador indica que quer compor um vídeo existente. A aplicação apresenta uma lista de vídeos contidos no disco do dispositivo móvel. O utilizador indica qual o vídeo que pretende. A aplicação cria a composição vídeo-musical, e reproduz a mesma quando acabada.		*Must*
Reproduzir uma composição vídeo-musical.	Permitir que o utilizador possa reproduzir uma composição vídeo-musical.	Ter uma composição vídeo-musical.	Não tem.	O utilizador indica que pretende escolher uma composição vídeo-musical. A aplicação apresenta-lhe uma lista com composições. O utilizador indica qual pretende reproduzir. A aplicação reproduz a composição vídeo-musical.		*Must*
Pausar a reprodução de uma composição vídeo-musical.	Permitir que o utilizador possa pausar a reprodução de uma composição vídeo-musical.	Estar a reproduzir uma composição vídeo-musical.	A composição vídeo-musical fica em pausa.	O utilizador indica que quer pausar a reprodução da composição vídeo-musical. A aplicação faz uma pausa na reprodução.		*Should*

Nome	**Objectivo**	**Pré-condições**	**Pós-condições**	**Fluxo de eventos principal**	**Fluxo de eventos secundários**	**Prioridade**
Retomar a reprodução de uma composição vídeo-musical.	Permitir que o utilizador possa retomar a reprodução de uma composição vídeo-musical.	Estar uma reprodução de uma composição vídeo-musical pausada.	A composição vídeo-musical está a reproduzir novamente.	O utilizador indica que quer retomar a reprodução da composição vídeo-musical. A aplicação retoma a reprodução.		*Should*
Enviar a composição vídeo-musical para o servidor.	Permitir que o utilizador possa enviar uma composição vídeo-musical para o servidor.	A aplicação acabou de reproduzir uma composição vídeo-musical ou esta está em pausa.	A composição vídeo-musical é enviada para o servidor.	O utilizador indica que pretende enviar a composição vídeo-musical para o servidor. A aplicação requer alguns dados relacionados com a composição (nome). O utilizador indica os dados necessários. A aplicação envia a composição para o servidor. O servidor guarda a composição.	Caso não haja conexão à internet: A aplicação envia a composição para o servidor quando existir conexão à internet. Caso a composição já esteja no servidor a aplicação não a envia novamente.	*Must*
Partilhar a composição vídeo-musical nas redes sociais.	Permitir que o utilizador possa partilhar uma composição vídeo-musical nas redes sociais.	A aplicação acabou de reproduzir uma composição vídeo-musical ou esta está em pausa.	A composição vídeo-musical é partilhada nas redes sociais.	O utilizador indica que pretende partilhar a composição vídeo-musical nas redes sociais. A aplicação requer a verificação de alguns dados relacionados com a composição (nome, local e escolha das redes sociais em que a composição será partilhada). O utilizador verifica os dados. A aplicação envia as atualizações dos dados da composição para o servidor e faz a partilha para as redes sociais escolhidas. O servidor guarda a composição.	Caso não haja conexão à internet: A aplicação envia as atualizações dos dados da composição para o servidor caso seja necessário e faz a sua partilha quando existir conexão à internet. Caso a composição já esteja no servidor a aplicação não a envia novamente.	*Could*
Apagar a composição vídeo-musical.	Permitir que o utilizador possa apagar uma composição vídeo-musical.	A aplicação acabou de reproduzir uma composição vídeo-musical ou esta está em pausa.	A composição vídeo-musical foi removida.	O utilizador indica que pretende apagar a composição vídeo-musical. A aplicação apaga essa composição.		*Must*

Nome	Objectivo	Pré-condições	Pós-condições	Fluxo de eventos principal	Fluxo de eventos secundários	Prioridade
Atribuir uma pontuação a uma composição vídeo-musical.	Permitir que o utilizador possa avaliar uma composição vídeo-musical.	Estar a visualizar uma composição vídeo-musical na *página web*.	A pontuação é atribuída.	O utilizador indica que pretende atribuir uma certa pontuação à composição vídeo-musical. O servidor guarda essa informação.		*Must*
Ver uma pontuação de uma composição vídeo-musical.	Permitir que o utilizador consiga perceber se uma composição vídeo-musical está bem classificada.	Estar a visualizar uma composição vídeo-musical na *página web*.	Não tem.	O utilizador indica que pretende a pontuação da composição vídeo-musical. O servidor apresenta a pontuação.		*Must*
Ver o *ranking* das pontuações dos utilizadores.	Permitir que o utilizador consiga comparar a sua pontuação com o *ranking* dos utilizadores.	Estar na *página web*.	Não tem.	O utilizador indica que pretende o *ranking* dos utilizadores. O servidor apresenta o *ranking*.		*Should*
Ver um *badge*.	Permitir que o utilizador consiga ver um *badge* e a sua descrição.	Estar na *página web*.	Não tem.	O utilizador indica que pretende ver a coleção de *badges*. O servidor apresenta a coleção. O utilizador escolhe um *badge*. O servidor apresenta o *badge* e a sua descrição.		*Must*
Obter um *badge*.	Permitir que o utilizador obtenha um *badge*.	Estar na *página web* ou na aplicação.	O utilizador obtém o *badge*.	O utilizador realiza o que é descrito pelo *badge* para que este seja desbloqueado. O servidor desbloqueia o *badge*.		*Must*
Ver uma missão.	Permitir que o utilizador consiga ver uma missão e a sua descrição.	Estar na *página web*.	Não tem.	O utilizador indica que pretende ver as missões. O servidor apresenta as missões e as suas descrições.		*Must*
Concluir uma missão.	Permitir que o utilizador conclua uma missão.	Estar na *página web* ou na aplicação.	O utilizador ganha pontos.	O utilizador realiza o que é descrito pela missão. O servidor atribui os pontos ao utilizador.		*Must*

Nome	**Objectivo**	**Pré-condições**	**Pós-condições**	**Fluxo de eventos principal**	**Fluxo de eventos secundários**	**Prioridade**
Registar um utilizador.	Permitir que o utilizador se registe.	Estar na *página web* ou na aplicação.	O utilizador fica registado.	O utilizador indica que pretende fazer o registo. O servidor apresenta o formulário. O utilizador preenche. O servidor valida o formulário e adiciona o utilizador à base de dados.	Caso o formulário esteja preenchido incorretamente: O utilizador corrige o problema e volta a submeter o formulário. O servidor volta a validar. Caso esteja correto o servidor adiciona o utilizador à base de dados, no entanto se estiver ainda incorreto o processo é repetido.	*Must*
Iniciar a sessão.	Permitir que o utilizador inicie a sessão.	Estar na *página web* ou na aplicação, e estar registado.	O utilizador fica com a sessão iniciada, ou seja, está autorizado a fazer todos os outros casos de uso (excepto "registar um utilizador").	O utilizador indica que pretende iniciar a sessão. O servidor pede os dados do utilizador (*e-mail* e *password*). O utilizador insere os dados. O servidor valida os dados e autoriza a que a sessão seja iniciada.	Caso os dados do utilizador estejam incorretos: O utilizador corrige o problema e volta a submeter os dados. O servidor volta a validar. Caso esteja correto o servidor autoriza a que a sessão seja iniciada, no entanto se estiver ainda incorreto este processo é repetido.	*Must*

3.4 Arquitetura do sistema

Nesta secção é analisado o sistema e os seus componentes principais.

Na figura 3.1 é apresentada uma arquitetura de alto nível que representa a comunicação entre o dispositivo móvel e o servidor, e entre o dispositivo móvel e as redes sociais. Pode-se verificar que este dispositivo interage com o utilizador e possui o sistema operativo *Android* que utiliza a *framework OpenCV* para realizar a detecção de formas geométricas. Nesta arquitetura também está presente o servidor que inclui o jogo e o *front-end* necessário para a interação com o utilizador.

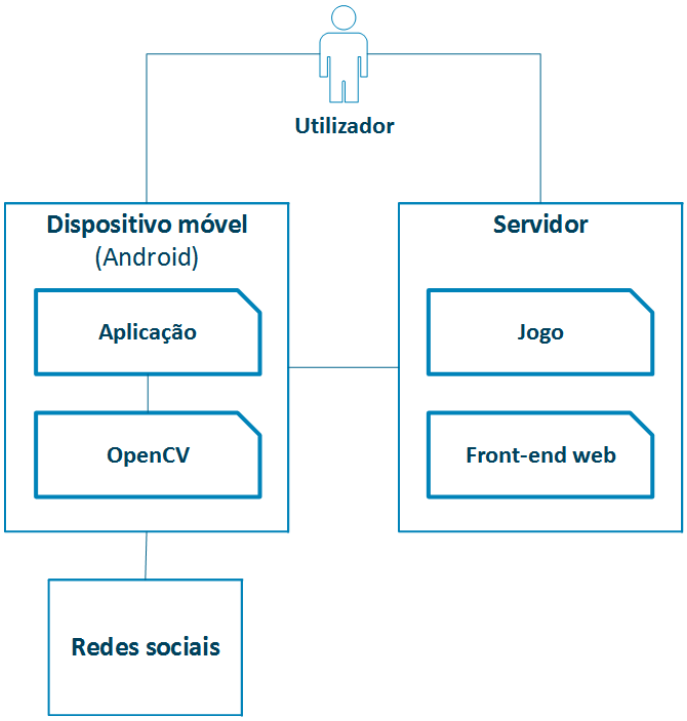


Figura 3.1: Diagrama de alto nível da arquitetura

No caso da figura 3.2 é apresentado um diagrama de componentes que exemplifica como são feitas as interações entre os componentes do sistema. Dentro do dispositivo móvel *Android* existe um componente responsável pela captura de vídeo que por sua vez disponibiliza o vídeo para outro componente que é responsável pelo pós-processamento da detecção de formas geométricas elementares. A componente que tem a função gerar composições vídeo-musicais cria a música através do vídeo processado anteriormente e gera a composição vídeo-musical. A componente responsável por reproduzir composições vídeo-musicais obtém a composição através da componente de geração de composições ou da galeria. A galeria contém todas as composições realizadas no dispositivo uma vez que as obtém assim que estas são geradas. As composições que se encontram na galeria são enviadas para as

redes sociais através da componente de partilha, e para o servidor através da componente de comunicação. A componente de interface está responsável por mapear alguns destes componentes no ecrã do dispositivo, como é o caso da captura, da reprodução e da galeria.

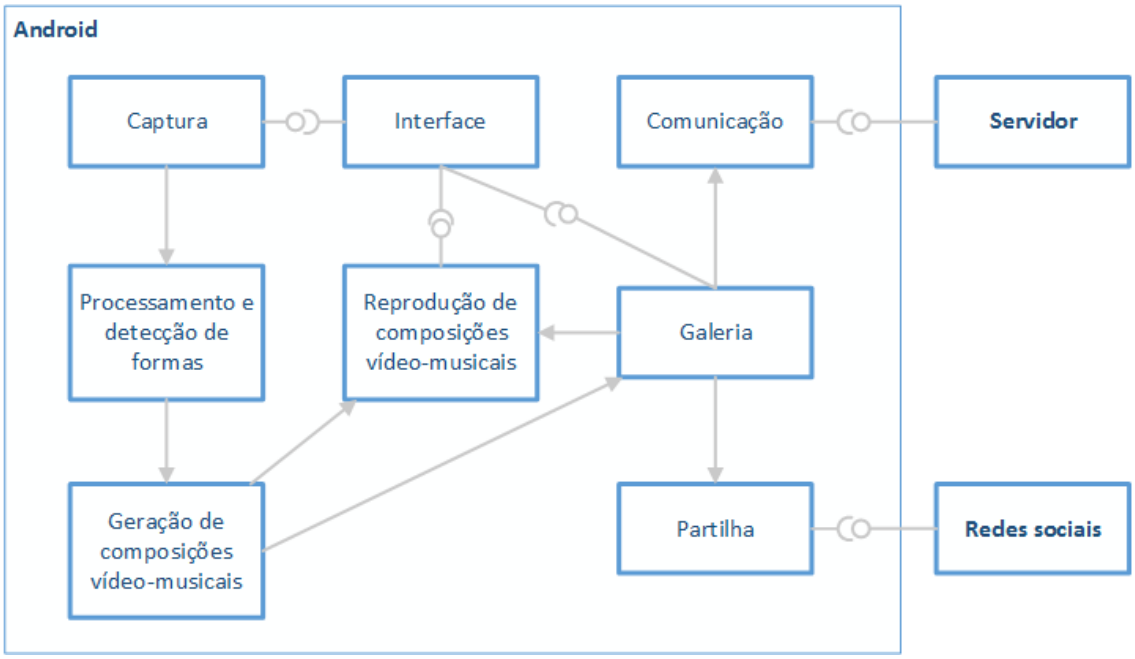


Figura 3.2: Diagrama de componentes da arquitetura do sistema

3.5 Diagrama ER

O servidor tem como função guardar e apresentar a informação necessária, e neste sentido foi necessário analisar qual a melhor maneira de a organizar. Para isto foi criado um diagrama ER (figura 3.3) para definir a estrutura da base de dados.

Existem quatro componentes bastante importantes na base de dados que são: os utilizadores, as composições, os *badges* e as missões. O elemento mais importante são os utilizadores uma vez que estão ligados a todos os outros.

Os utilizadores tem alguns campos como: o nome, o *username*, a *password*, o *email*, o número de pontos, e um campo para verificar se o utilizador é *admin*. Neste caso o *email* e a *password* são bastante importantes para se conseguir realizar o *login*, o *username* serve para distinguir utilizadores que tenham o mesmo nome e os pontos servem para manter a contagem de todos os pontos ganhos pelo utilizador. Os utilizadores têm uma ligação de um para muita composições o que significa que aquelas composições foram criadas por um utilizador, além disto existe outra ligação com as composições que é feita por uma tabela

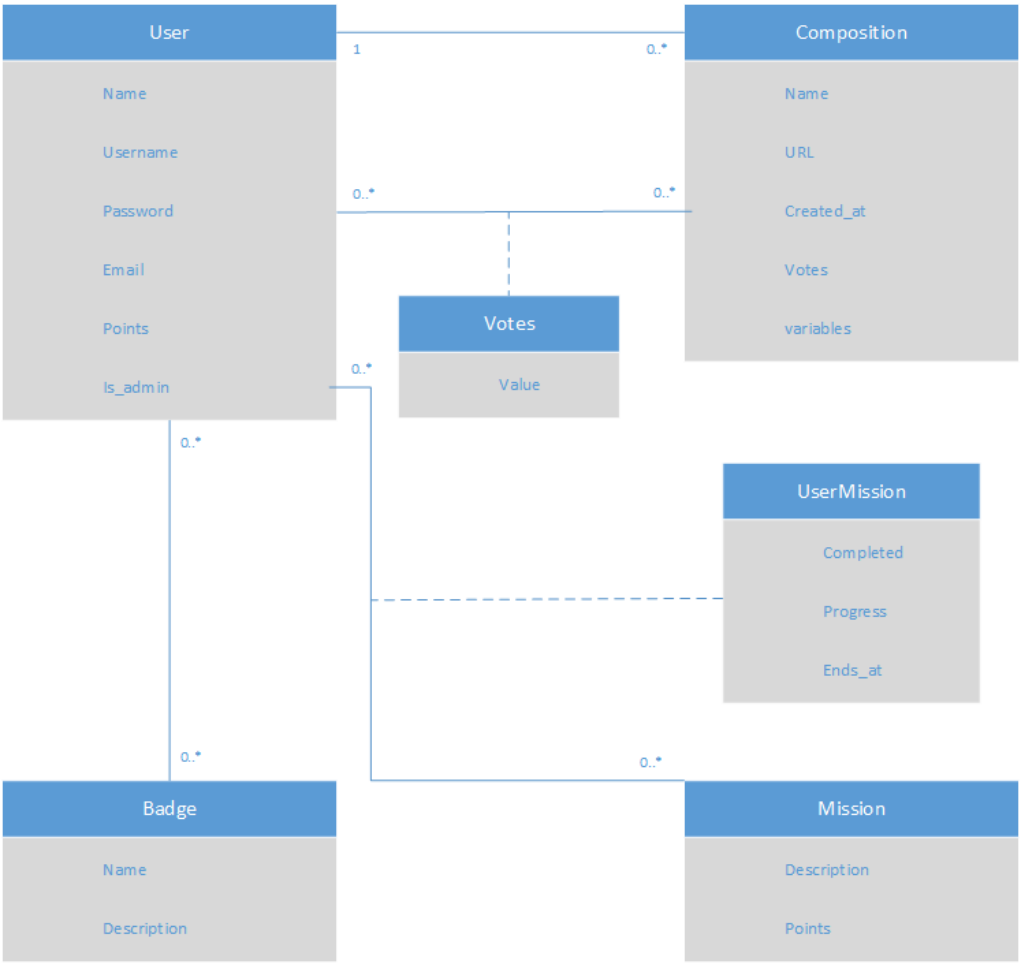


Figura 3.3: Diagrama ER

intermediária chamada votos. Esta tabela serve para guardar os votos que utilizadores deram a uma composição e qual o valor (de um a cinco) do voto que lhe foi atribuído. Neste caso cada utilizador pode fazer muitos votos, mas cada voto pertence a uma única composição, da mesma maneira que uma composição tem muitos votos, mas cada voto foi feito apenas por um utilizador.

Os utilizadores têm uma ligação muito semelhante com as missões, uma vez que também existe uma tabela intermediária. Essa tabela serve para perceber se a missão foi concluída pelo utilizador, o seu progresso, e qual é o tempo limite da mesma. Estes campos não estão contidos na missão porque uma missão pode ser atribuída a muitos utilizadores, por isso com o uso desta tabela é possível os utilizadores terem muitas missões e as missões terem muitos utilizadores.

Para finalizar os utilizadores ainda têm outra ligação com os *bagdes*. Esta ligação mostra que existem *badges* que podem ser atribuídos a mais que um utilizador e que os utilizadores

podem ter muitos *badges*.

As composições guardam a informação como: o nome, o *url*, a data de criação, os votos e as variáveis. O nome foi escolhido pelo utilizador, o *url* corresponde ao *link* do *Youtube* onde se encontra a composição, a data de criação serve para saber quando a composição foi adicionada ao servidor, os votos representam a média de todos os votos realizados pelo utilizador e as variáveis servem para guardar informação como o número de formas encontradas, o tipo formas, entre outros. No caso dos votos esta variável serve como optimização, uma vez que desta forma não é necessário recontar todos os votos já feitos para saber qual é a média dos valores atribuídos.

As missões têm apenas dois parâmetros: a descrição e os pontos. A descrição é uma explicação do que é necessário fazer para completar a missão e os pontos correspondem à pontuação que é atribuída ao utilizador caso este consiga concluir a missão com sucesso.

Os *bagdes* são bastante simples uma vez que também só têm dois parâmetros: o nome e uma descrição. O nome serve para identificar o *badge* em questão e a descrição serve para informar o utilizador o que deve fazer para conseguir ganhar aquele *badge* e juntá-lo à sua coleção.

Capítulo 4

Planeamento

Este capítulo apresenta a forma como o trabalho foi desenvolvido no primeiro e segundo semestre. Descreve também as alterações feitas ao planeamento e as escolhas e decisões tomadas sobre algumas tecnologias importantes para o projeto

4.1 Planeamento inicial

Esta secção descreve o planeamento que foi feito inicialmente, antes de qualquer alteração do projeto. Este planeamento acabou por sofrer algumas alterações no segundo semestre, devido a algumas dificuldades encontradas.

4.1.1 Primeiro Semestre

Este projeto começou com a definição da ideia a desenvolver, ou seja, no que consiste, qual a motivação e o seu contexto. Depois de definida a ideia foi necessário fazer um estudo sobre as áreas abrangidas, como a realidade aumentada, o *ambient gaming* e os dispositivos móveis. Neste estudo foram contempladas definições, alguns exemplos práticos já existentes, e também algumas comparações de tecnologias existentes. Em algumas destas comparações, como é o caso das *frameworks* de realidade aumentada, foi necessário utilizar um dispositivo móvel para perceber como é que elas reagiam e se estavam alinhadas com a ideia do projeto.

Após ter sido escolhida aquela que estava mais próxima do pretendido, começou-se por fazer um pequeno teste com algumas das funções que iriam ser necessárias mais tarde para o projeto, como por exemplo a detecção de círculos. Nesta detecção, foi encontrado o primeiro problema uma vez que a aplicação de teste ficava com o número de *FPS* (*Frames Per Second*) demasiado baixo (cerca de 1.8, quando o valor mínimo aceitável seria 7.5)

o que era um impedimento para a realidade aumentada, uma vez que esta tem que ser feita em tempo real. Numa tentativa de melhorar este aspecto tentou-se modificar então a aplicação de teste para que deixasse de existir este problema. A aplicação foi melhorada mas ainda assim não era suficiente, pois naquela aplicação teste estava apenas a função de detectar círculos, ou seja, o problema ia-se agravar assim que fossem adicionadas novas funções.

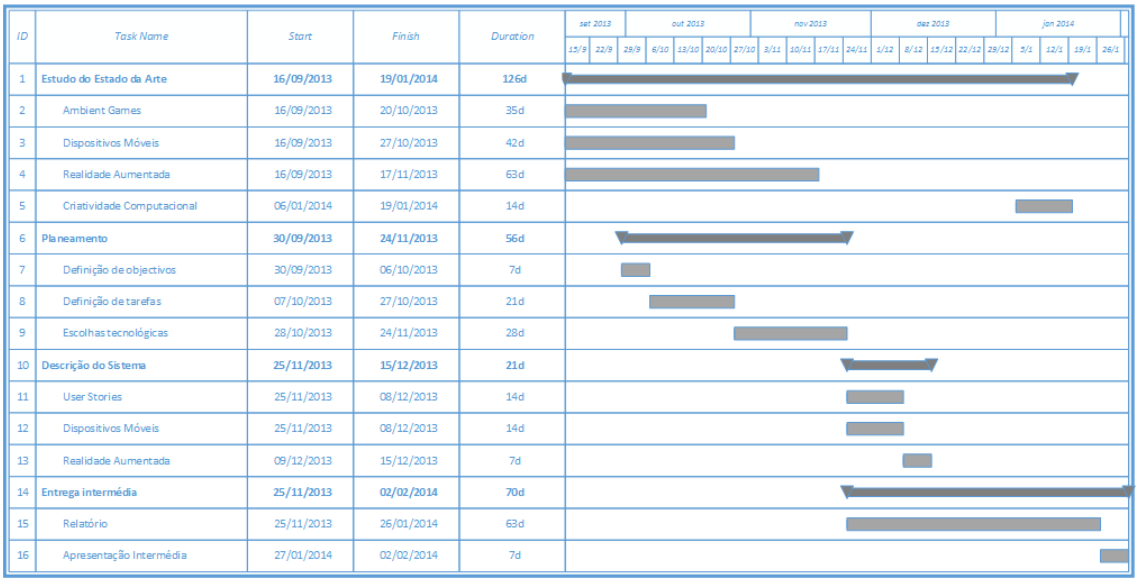
Uma vez que o problema se mantinha aparentemente sem nenhuma forma viável de o solucionar, foi necessário reformular a ideia do projeto. Nesta reformulação chegou-se ao consenso que a aplicação deixaria de ter a componente de realidade aumentada devido à impossibilidade da aplicação permitir tempo real. A nova ideia foi deixar de parte o tempo real e fazer o pós-processamento do vídeo, sendo que o *output* seria o mesmo que na ideia anterior. Esta solução tem também a vantagem de ser possível voltar facilmente à ideia inicial, assim que existam melhorias de *hardware* por parte dos dispositivos móveis.

Para planear o trabalho foi necessário criar algumas tarefas e sub-tarefas de maneira a estabelecer ordem e prazos. Foram então criadas sete tarefas principais, quatro das quais são respectivas ao primeiro semestre e as restantes ao segundo semestre. As quatro tarefas delineadas para o primeiro semestre são:

1. Estudo do estado da arte
2. Planeamento
3. Descrição do sistema
4. Entrega intermédia

A figura 4.1 apresenta o diagrama de *Gantt* das tarefas relativas ao primeiro semestre.
A primeira tarefa, estudo do estado da arte, foi dividida em quatro sub-tarefas, que representam as temáticas exploradas neste trabalho. Estas sub-tarefas podem ser observadas na tabela 4.1.

A tarefa de planeamento corresponde à organização e à tomada de algumas decisões relacionadas com o projeto. Esta tarefa foi dividida em três sub-tarefas como se pode analisar na tabela 4.2.



ID	Task Name	Start	Finish	Duration
1	**Estudo do Estado da Arte**	**16/09/2013**	**19/01/2014**	**126d**
2	Ambient Games	16/09/2013	20/10/2013	35d
3	Dispositivos Móveis	16/09/2013	27/10/2013	42d
4	Realidade Aumentada	16/09/2013	17/11/2013	63d
5	Criatividade Computacional	06/01/2014	19/01/2014	14d
6	**Planeamento**	**30/09/2013**	**24/11/2013**	**56d**
7	Definição de objectivos	30/09/2013	06/10/2013	7d
8	Definição de tarefas	07/10/2013	27/10/2013	21d
9	Escolhas tecnológicas	28/10/2013	24/11/2013	28d
10	**Descrição do Sistema**	**25/11/2013**	**15/12/2013**	**21d**
11	User Stories	25/11/2013	08/12/2013	14d
12	Dispositivos Móveis	25/11/2013	08/12/2013	14d
13	Realidade Aumentada	09/12/2013	15/12/2013	7d
14	**Entrega intermédia**	**25/11/2013**	**02/02/2014**	**70d**
15	Relatório	25/11/2013	26/01/2014	63d
16	Apresentação Intermédia	27/01/2014	02/02/2014	7d

Figura 4.1: Diagrama de *Gantt* relativo ao primeiro semestre

1. Estudo do estado da arte	Descrição	Data de início	Data de fim
1.1. *Ambient gaming*	Estudo sobre o conceito de *ambient gaming*, a sua história e alguns projetos desenvolvidos nesta área.	16/09/13	20/10/13
1.2. Dispositivos móveis	Estudo sobre a evolução dos dispositivos móveis, as suas características, os sistemas operativos e as estatísticas de utilização.	16/09/13	27/10/13
1.3. Realidade aumentada	Estudo sobre o conceito de realidade aumentada, a utilidade desta área, os seus desafios, a comparação de *frameworks* e a sua ligação aos dispositivos móveis.	16/09/13	17/11/13
1.4 Criatividade computacional	Estudo sobre o conceito de criatividade computacional e suas áreas de trabalho.	06/01/14	19/01/14

Tabela 4.1: Estudo do estado da arte

2. Planeamento	Descrição	Data de início	Data de fim
2.1. Definição de objetivos	Definição dos principais objetivos do projeto	30/09/13	06/10/13
2.2. Definição de tarefas	Definição das principais tarefas e sub-tarefas do projeto	07/10/13	27/10/13
2.3. Escolhas tecnológicas	Estudo e comparação de diferentes tecnologias, e escolha das mais adequadas para o projeto.	28/10/13	24/11/13

Tabela 4.2: Planeamento

A tarefa de descrição do sistema consiste em descrever o sistema e os seus requisitos, e foi dividida em três sub-tarefas presentes na tabela 4.3.

A entrega intermédia é a última tarefa do primeiro semestre e corresponde à elaboração do relatório e à preparação da apresentação intermédia, como está descrito na tabela 4.4.

3. Descrição do sistema	Descrição	Data de início	Data de fim
3.1. *User stories*	Criação de *user stories* para responderem aos vários requisitos do jogo.	25/11/13	08/12/13
3.2. Casos de uso	Definição de casos de uso para os requisitos do jogo.	25/11/13	08/12/13
3.3. Arquitetura do sistema	Definição da arquitetura do sistema	09/12/13	15/12/13

Tabela 4.3: Descrição do sistema

4. Entrega intermédia	Descrição	Data de início	Data de fim
4.1. Relatório intermédio	Elaboração do relatório intermédio.	25/11/13	26/01/14
4.2. Apresentação intermédia	Preparação da apresentação intermédia	27/01/14	02/02/14

Tabela 4.4: Entrega intermédia

4.1.2 Segundo semestre

Como foi dito na subsecção 4.1.1, o projeto foi dividido em sete tarefas principais em que quatro destas já foram analisadas. No entanto faltam analisar três destas tarefas que correspondem ao segundo semestre:

5. Desenvolvimento
6. Testes e melhorias
7. Entrega final

A figura 4.2 apresenta o diagrama de *Gantt* das tarefas relativas ao segundo semestre.

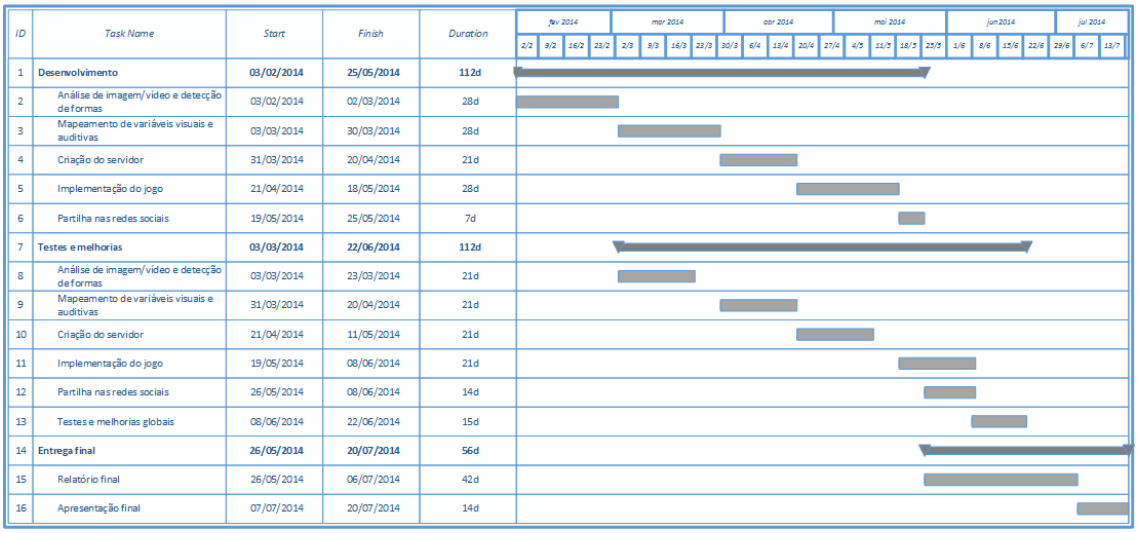


Figura 4.2: Diagrama de *Gantt* relativo ao segundo semestre

A tarefa de desenvolvimento foi dividida em cinco sub-tarefas, com o intuito de mapear os diferentes componentes necessários para o projeto. As sub-tarefas podem ser analisadas com mais rigor na tabela 4.5.

5. Desenvolvimento	Descrição	Data de início	Data de fim
5.1. Análise de imagem/vídeo e detecção de formas	Desenvolvimento da componente responsável por analisar a imagem/vídeo e por detectar de formas geométricas.	03/02/14	02/03/14
5.2. Mapeamento de variáveis visuais e auditivas	Desenvolvimento da componente responsável por mapear as variáveis visuais das formas geométricas detectadas em variáveis auditivas.	03/03/14	30/03/14
5.3. Criação do servidor	Criação do servidor responsável pela gestão dos dados dos utilizadores.	31/03/14	20/04/14
5.4. Implementação do jogo	Implementação do jogo na aplicação e servidor	21/04/14	18/05/14
5.5. Partilha nas redes sociais	Desenvolvimento da componente responsável por fazer partilha das composições vídeo-musicais nas redes sociais	19/05/14	25/05/14

Tabela 4.5: Desenvolvimento

Os testes e melhorias, fazem parte de uma tarefa principal que foi dividida em seis sub-tarefas. Os testes e melhorias, que são feitos aos componentes referidos em cima, podem ser analisados na tabela 4.6.

6. Testes e melhorias	Descrição	Data de início	Data de fim
6.1. Análise de imagem/vídeo e detecção de formas	Realização de testes e melhorias sobre a análise de imagem/vídeo e detecção de formas.	03/03/14	23/03/14
6.2. Mapeamento de variáveis visuais e auditivas	Realização de testes e melhorias sobre o mapeamento de variáveis visuais e auditivas.	31/03/14	20/04/14
6.3. Criação do servidor	Realização de testes e melhorias sobre o servidor.	21/04/14	11/05/14
6.4. Implementação do jogo	Realização de testes e melhorias sobre o jogo.	19/05/14	08/06/14
6.5. Partilha nas redes sociais	Realização de testes e melhorias sobre a partilha nas redes sociais.	26/05/14	08/06/14
6.6. Testes e melhorias globais	Realização de testes e melhorias sobre todo o projeto.	08/06/14	22/06/14

Tabela 4.6: Testes e melhorias

Por fim, a entrega final que corresponde à conclusão do relatório final e à preparação da apresentação final. A entrega final pode ser observada com mais detalhe na tabela 4.7.

7. Entrega final	Descrição	Data de início	Data de fim
7.1. Relatório final	Elaboração do relatório final.	26/05/14	06/07/14
7.2. Apresentação final	Preparação da apresentação final.	37/07/14	20/07/14

Tabela 4.7: Entrega final

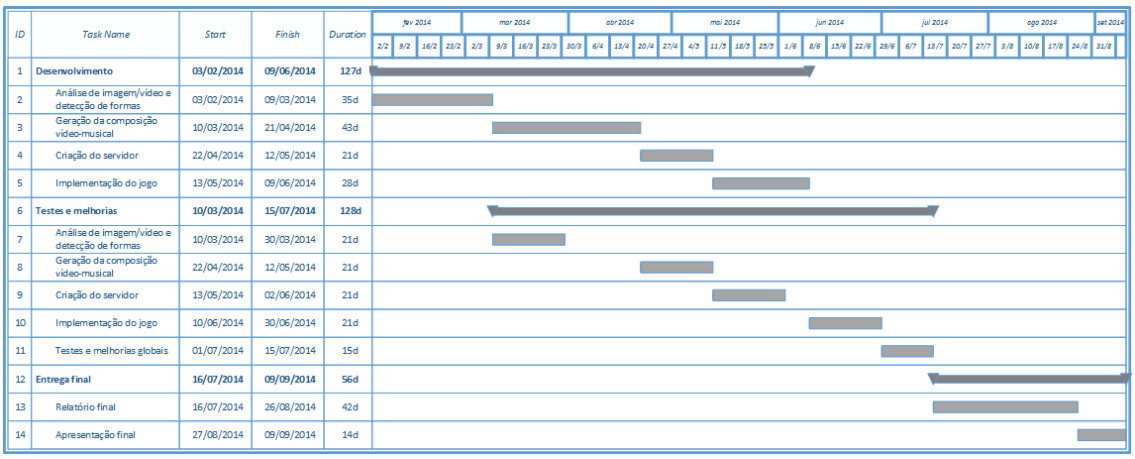
4.2 Planeamento reformulado

Esta secção tem o intuito de explicar o desvio que foi feito, em termos temporais, no segundo semestre. Inicialmente tinha sido planeado que a tarefa de desenvolvimento iria começar no início de Fevereiro e terminar no final de Maio, no entanto existiram algumas dificuldades ao realizar esta tarefa. Estas dificuldades começaram, quando estava a ser realizada a sub-tarefa "Análise imagem/vídeo e detecção de formas", uma vez que a *framework OpenCV* que estava a ser usada para a detecção de formas não tinha funções capazes de detectar formas geométricas que não fossem círculos. Como não existiam outras *frameworks* capazes de detectar formas foi necessário realizar um estudo mais intensivo sobre o *OpenCV* de maneira a encontrar uma maneira a encontrar uma solução. A solução foi encontrada mas o tempo gasto nesta sub-tarefa maior do que o esperado.

Ultrapassada esta dificuldade, foi necessário passar ao próximo passo, ou seja, o "Mapeamento de variáveis visuais e auditivas". Este passo foi o mais prolongado de todos, uma vez que foram encontradas muitas dificuldades devido a incompatibilidades de formatos de áudio. Inicialmente era suposto usar o formato *midi* para gerar música, uma vez que este permite a fácil manipulação de variáveis musicais. No entanto, este formato impedia que a música fosse adicionada ao vídeo de forma a criar uma composição vídeo-musical pois este não era compatível com as ferramentas usadas pelos dispositivos móveis. Depois de algum tempo a tentar encontrar uma solução, percebeu-se que esta área de gerar música em dispositivos móveis está pouco explorada e por isso está pouco desenvolvida. Após muitas bibliotecas estudadas, foi encontrada uma solução que envolvia outros formatos de áudio. No entanto, esta acabava por perder a manipulação de variáveis musicais que o *midi* disponibilizava, e por isso foi necessário reformular a ideia de como era gerada a música. Foi então pensado criar um sequenciador de *loops* musicais, mas as dificuldades continuavam pois era necessário sobrepor músicas e não existia nenhuma biblioteca que fosse capaz de fazer isso. Mais uma vez, foi feita uma pesquisa exaustiva para encontrar uma solução. Foi então encontrada uma solução que era capaz de sobrepor as músicas no formato *wav*, mas este continuava a ser incompatível na junção ao vídeo. Com isto, foi necessário encontrar uma maneira de converter este formato para um que fosse o desejado, e após algum tempo foi encontrada uma biblioteca que conseguia fazer a conversão. Com esta tabela biblioteca finalmente era possível realizar a geração de uma composição vídeo-musical.

Existiram também outros aspectos que fizeram com que o plano acabasse por ser alterado, como é o caso do jogo. O jogo e o servidor começaram a ter outra prioridade após algumas ideias que surgiram como por exemplo: as missões e os *badges*. A prioridade destes passou a ser mais elevada, no entanto existiam outras componentes que começaram a perder a prioridade que tinham como era o caso da partilha nas redes sociais. Esta componente tinha como função proliferar a aplicação pelas redes sociais, no entanto com o uso do *Youtube* para fazer o *upload* das composições a urgência de partilhar as composições noutras redes sociais tornou-se menos relevante, uma vez que o *Youtube* era capaz de realizar a mesma função.

A figura 4.3 apresenta o diagrama de *Gantt* do segundo semestre já reformulado.



ID	Task Name	Start	Finish	Duration
1	**Desenvolvimento**	**03/02/2014**	**09/06/2014**	**127d**
2	Análise de imagem/vídeo e detecção de formas	03/02/2014	09/03/2014	35d
3	Geração da composição vídeo-musical	10/03/2014	21/04/2014	43d
4	Criação do servidor	22/04/2014	12/05/2014	21d
5	Implementação do jogo	13/05/2014	09/06/2014	28d
6	**Testes e melhorias**	**10/03/2014**	**15/07/2014**	**128d**
7	Análise de imagem/vídeo e detecção de formas	10/03/2014	30/03/2014	21d
8	Geração da composição vídeo-musical	22/04/2014	12/05/2014	21d
9	Criação do servidor	13/05/2014	02/06/2014	21d
10	Implementação do jogo	10/06/2014	30/06/2014	21d
11	Testes e melhorias globais	01/07/2014	15/07/2014	15d
12	**Entrega final**	**16/07/2014**	**09/09/2014**	**56d**
13	Relatório final	16/07/2014	26/08/2014	42d
14	Apresentação final	27/08/2014	09/09/2014	14d

Figura 4.3: Diagrama de *Gantt* relativo ao segundo semestre com as reformulações

A tarefa de desenvolvimento foi dividida em quatro sub-tarefas, com o intuito de mapear os diferentes componentes necessários para o projeto. As sub-tarefas podem ser analisadas com mais rigor na tabela 4.8.

5. Desenvolvimento	Descrição	Data de início	Data de fim
5.1. Análise de imagem/vídeo e detecção de formas	Desenvolvimento da componente responsável por analisar a imagem/vídeo e por detectar de formas geométricas.	03/02/14	09/03/14
5.2. Geração da composição vídeo-musical	Desenvolvimento da componente responsável por gerar música a partir das variáveis visuais das formas geométricas detectadas e por criar a composição vídeo-musical.	10/03/14	21/04/14
5.3. Criação do servidor	Criação do servidor responsável pela gestão dos dados dos utilizadores.	22/04/14	12/05/14
5.4. Implementação do jogo	Implementação do jogo e dos seus componentes (missões, *badges*, ranking e votações).	13/05/14	09/06/14

Tabela 4.8: Desenvolvimento

Os testes e melhorias, fazem parte de uma tarefa principal que foi dividida em seis subtarefas. Os testes e melhorias, que são feitos aos componentes referidos em cima, podem ser analisados na tabela 4.9.

6. Testes e melhorias	Descrição	Data de início	Data de fim
6.1. Análise de imagem/vídeo e detecção de formas	Realização de testes e melhorias sobre a análise de imagem/vídeo e detecção de formas.	10/03/14	30/03/14
6.2. Geração da composição vídeo-musical	Realização de testes e melhorias sobre a geração da música e da composição vídeo-musical	22/04/14	12/05/14
6.3. Criação do servidor	Realização de testes e melhorias sobre o servidor.	13/05/14	02/06/14
6.4. Implementação do jogo	Realização de testes e melhorias sobre o jogo.	10/06/14	30/06/14
6.6. Testes e melhorias globais	Realização de testes e melhorias sobre todo o projeto.	01/07/14	15/07/14

Tabela 4.9: Testes e melhorias

Por fim, a entrega final que corresponde à conclusão do relatório final e à preparação da apresentação final. A entrega final pode ser observada com mais detalhe na tabela 4.10.

7. Entrega final	Descrição	Data de início	Data de fim
7.1. Relatório final	Elaboração do relatório final.	16/07/14	26/08/14
7.2. Apresentação final	Preparação da apresentação final.	27/08/14	09/09/14

Tabela 4.10: Entrega final

4.3 Escolhas Tecnológicas

Para realizar este projeto foram analisadas e escolhidas várias tecnologias que se adequassem aos objetivos já delineados.

4.3.1 Aplicação Móvel

Primeiro foram analisados os dispositivos móveis, que foram escolhidos inicialmente por permitir incorporar o tempo real necessário para implementar a realidade aumentada e também por ser um objecto que à partida é comum para as pessoas o que se encaixa perfeitamente no tema do *ambient gaming*. Apesar do tema da realidade aumentada ter sido posto de parte, os dispositivos móveis continuam enquadrados, uma vez que assim que existam melhorias no seu *hardware*, como tem acontecido durante estes últimos anos, o projeto será facilmente adaptado para tempo real.

Dentro dos dispositivos móveis analisaram-se os vários sistemas operativos, mais especificamente o *Android* e o *iOS*, pois estes representavam mais utilizadores. Entre estes dois verificou-se que o *iOS* traria mais gastos de desenvolvimento e que o *Android* apresenta mais utilizadores e *downloads* de aplicações. Pelos motivos apresentados na secção 2.2 foi escolhido o sistema operativo *Android*.

Após ter sido escolhido o sistema operativo, foram analisadas algumas *frameworks* compatíveis para ajudar na detecção de formas e neste sentido foi escolhido o *OpenCV*, uma vez que este é capaz de fazer vários tipos de reconhecimento de objetos que se aproximavam dos objetivos do projeto. Todas as outras *frameworks* estudadas na secção A.4 apresentavam um reconhecimento que não era bem o pretendido uma vez que era reconhecimento de marcadores.

4.3.2 Servidor

Nesta subsecção são explicadas as ferramentas e as base de dados que foram analisadas para construir o servidor.

Ferramentas

Para começar, foi feito um estudo sobre as funcionalidades que o servidor iria ter que incorporar. Estas funcionalidades são: o jogo com as componentes dos pontos, *badges*, missões e *ratings*, múltiplos utilizadores, informação sobre as composições, a interação entre utilizadores e composições, e a comunicação com a aplicação. Tendo este tipo de funcionalidades como base era necessário uma *framework* compatível com rápido desenvolvimento.

Após analisadas todas as funcionalidades passou-se à escolha da ferramenta mais adequada. Na procura por esta ferramenta surgiram algumas das mais utilizadas, como por exemplo: *Play Framework* [1], *Ruby on Rails* [2] e *Node.js* [3]. Foram então estudadas as suas capacidades para poder ser tomada uma decisão.

A *Play Framework* utiliza *java* como linguagem de programação e já era conhecida pelo autor uma vez que já tinha sido usada noutros projetos de dimensão menor. A comunidade que desenvolve aplicações *web* através desta *framework* acaba por ser menor que nas outras duas assim como a documentação. Apesar de o autor já ter alguns conhecimentos sobre a *Play Framework* estes não eram suficientes para desenvolver todo o servidor, o que levaria

[1] mais informações em: https://www.playframework.com/
[2] mais informações em: http://rubyonrails.org/
[3] mais informações em: http://nodejs.org/

à necessidade de uma aprendizagem para melhorar os conhecimentos. Isto associado ao facto de existir menos documentação do que as outras ferramentas e o facto de ser mais rápido desenvolver o mesmo projeto em *Ruby on Rails* ou *Node.js*, fez com que esta *framework* fosse posta de parte [53].

Ruby On Rails é uma *framework* atual que utiliza a linguagem *ruby*. Hoje em dia esta *framework* é bastante utilizada e por isso tem uma das maiores comunidades que existe nesta área. Através desta comunidade existe muita documentação e também muitas *gems* [4] que facilitam o trabalho. As *gems* são bibliotecas escritas em *ruby*, que foram desenvolvidas pela comunidade com o objectivo de optimizar o desenvolvimento do código, uma vez que estas tratam de problemas que são bastante comuns tais como: *logins*, registo de utilizadores, paginação e até mesmo *debug* [53]. Além das *gems*, o *Ruby on Rails* dispõe um conjunto de comandos que permite a construção de uma aplicação *web* em pouco tempo [54]. Através de todos estes aspectos conclui-se que com esta *framework* é possível desenvolver rapidamente o servidor e todos os aspectos a ele referentes [55][56][57][58].

Node.js também é uma ferramenta bastante atual e a linguagem de programação é *javascript*. Tem também uma grande comunidade, o que faz com haja uma grande documentação. Esta ferramenta é muito utilizada em aplicações *real-time* [59][60]. No entanto esta ferramenta, no que toca ao tempo de desenvolvimento fica atrás do *Ruby on Rails* uma vez que não tem o apoio das *gems* ou de algo semelhante.

Após terem sido analisadas todas as ferramentas, a escolhida acabou por ser o *Ruby on Rails*. Em comparação com a *Play Framework*, iria acabar por demorar mais tempo o desenvolvimento do projeto com esta do que com *Ruby On Rails*, uma vez que os conhecimentos do autor sobre a *Play Framework* não eram suficientes. Em relação ao *Node.js* o autor teria também que estudar esta ferramenta uma vez que era totalmente nova e além disso o conhecimento que este tinha sobre *ruby* comparado com *javascript* é muito maior. Isto associado ao facto do *Node.js* não ter tantas bibliotecas como o *Ruby on Rails*, iria fazer com que o processo de desenvolvimento fosse mais demorado. Além disto o *Ruby on Rails* é uma ferramenta mais matura que *Node.js*, uma vez que o *Node.js* é muito recente o que faz com que exista mais estabilidade no *Ruby on Rails*[56][57][58].

[4] mais informações em: https://rubygems.org/

Base de dados

Após a analise das várias componentes e das várias funcionalidades que o servidor tem que incorporar foi necessário definir qual é a base de dados capaz de suportar isto. Consequentemente foram analisadas algumas base de dados: *SQLite* [5], *MySQL* [6], *PostegreSQL* [7] e *Oracle* [8].

A *SQLite* é a base de dados que vem por omissão no *Ruby On Rails*, no entanto esta é demasiado simples para o projeto. Esta base de dados é normalmente utilizada para testes ou para aplicações que apenas tenham um utilizador. Tendo isto, o *SQLite* foi descartado por se mostrar incapaz de cumprir com as funcionalidades deste projeto [61].

Outra base de dados analisada foi o *MySQL* que é uma base de dados bastante comum. Esta tem uma segurança reforçada em comparação com o *SQLite* e é bastante fácil de configurar o que não acontece com outras. É bastante rápida no que toca a leituras, comparando com outras como é o caso do *PostegreSQL* [61][62][63].

O *PostegreSQL* que é muito utilizada em situações mais complexas (muitos dados, muitas entidades, muitas relações entre entidades, bases de dados de *backup*, entre outros), do que o *MySQL*, o que torna a sua configuração mais complicada. É utilizada quando existem muitos dados por gerir melhor a concorrência. No entanto, acaba por se tornar mais lenta tanto nas leitura como nas escritas [61][62].

A *Oracle* ainda foi analisada mas rapidamente se concluiu que não compensava adquirir uma base dados proprietária quando tanto o *MySQL* e o *PostegreSQL* eram capazes de cumprir com as necessidades do servidor sem qualquer custo [61][62].

Após a analise, concluiu-se que o mais adequado seria o *MySQL*. Como a base de dados não era muito complexa não fazia sentido estar a usar o *PostegreSQL* e assim acaba por se tirar partido da velocidade proporcionada pelo *MySQL*. Além disto, o autor também já tinha algum conhecimento adquirido com o *MySQL* o que não acontecia com o *PostegreSQL*[61][62].

[5] mais informações em: http://www.sqlite.org/
[6] mais informações em: http://www.mysql.com/
[7] mais informações em: http://www.postgresql.org/
[8] mais informações em: http://www.oracle.com/index.html

Capítulo 5

Processo da Análise de Vídeo

Neste capítulo é descrita a análise de vídeo, desde a detecção de formas geométricas até à recolha de variáveis associadas às formas detectadas.

5.1 Detecção de formas geométricas

Antes de ser iniciado o desenvolvimento da detecção de formas geométricas foi necessário definir que tipo de formas geométricas iriam ser consideradas no que toca a esta detecção. Foram selecionados três tipos simples de formas geométricas: círculos, quadriláteros e triângulos.

Para detectar formas geométricas num vídeo com alguma precisão é necessário preparar as *frames* que vão ser analisadas. Para isto aplica-se uma escala de cinzentos que é requerida por algumas funções do *OpenCV* e aplica-se também o *blur* que permite reduzir erros na detecção de formas. Esta preparação é importante para se conseguir detectar os limites das formas geométricas com rigor, de maneira a que não existam formas geométricas que na realidade são falsas. Sendo assim, foi necessário aplicar também o filtro *canny* sobre a *frame* que permite detectar os limites de todos os objetos presentes na *frame*. Tanto este filtro como o *blur* sofreram bastantes testes de maneira a ser possível detectar o maior número de formas geométricas sem erros (ver a secção 8.1).

5.1.1 Quadriláteros

Em relação à detecção de quadriláteros, o método utilizado para o reconhecimento de quadriláteros foi encontrado depois de uma pesquisa exaustiva para o reconhecimento de

quadriláteros e triângulos uma vez que não existe nenhuma função do *OpenCV* capaz de fazer o reconhecimento. Depois desta pesquisa foram encontradas algumas soluções, uma das quais era capaz de reconhecer quadriláteros, no entanto só conseguia detectar aqueles que tinham uma determinada cor, neste caso amarelo. Apesar desta ser uma solução possível, esta acabou por não ser escolhida uma vez que limitava o reconhecimento de muitas formas pois estas podiam ter cores que não estavam a ser consideradas. Outra solução encontrada foi aquela que acabou por ser usada. Inicialmente foi encontrado um exemplo escrito em *c++* com algumas chamadas a funções do *OpenCV*. No entanto existiram algumas dificuldades a converter este código para *Java*, uma vez que as funções do *OpenCV* usadas em *c++* não existiam em *Java* ou tinham parâmetros diferentes. No entanto, existiam outras funções que conseguiam ter resultados semelhantes ou então era necessário adequar os parâmetros para as funções em java.

Através da função *FindContours* do *OpenCV* é possível recolher os vários contornos através dos limites detectados anteriormente, através do filtro *canny*. Esta função tem a capacidade de perceber quais são os contornos que formam um polígono fechado e simples, retornando assim uma lista de contornos. Posteriormente é utilizada a função *ApproxPolyDP* do *OpenCV* que permite aproximar com mais precisão o contorno a um polígono, uma vez que alguns contornos contêm linhas ligeiramente curvas e assim é possível aproximá-las a linhas rectas.

Finalmente os quadriláteros encontrados são analisados para perceber se são realmente adequados e para eliminar resultados que não são de facto quadriláteros. Para isso é verificada a área, as arestas diagonais, entre outros (ver a secção 8.1).

5.1.2 Triângulos

No que toca à detecção de triângulos, acontece o mesmo que com os quadriláteros. São encontrados os contornos para se conseguir obter as arestas, que neste caso devem ser obrigatoriamente três arestas. No entanto, a detecção de triângulos sofreu mais algumas alterações de parâmetros uma vez que a probabilidade de erro aumenta quando existem menos arestas. Estes erros tem mais frequência quando a luminosidade da *frame* não é favorável e por isso foi necessário alterar certos parâmetros como por exemplo o filtro de *blur*. Com o aumento do impacto deste filtro foi então possível obter resultados mais favoráveis pois a questão da luminosidade foi minimizada (este processo é explicado com mais detalhe na 8.1).

5.1.3 Círculos

Em relação aos círculos são utilizados dois métodos de reconhecimento. O primeiro é feito através de uma função já existente no *OpenCV* chamada *HoughCircles*. Esta função recebe alguns parâmetros como por exemplo o tamanho mínimo e máximo do raio do círculo, permitindo melhorar a detecção e diminuir o tempo de processamento (ver também a secção 8.1). Este método é muito rígido no que toca a detectar um círculo, ou seja, para que este seja detectado os limites devem ser claros, o que por vezes não acontece por questões de luminosidade.

Para tentar resolver este problema foi utilizado outro método para que fosse possível detectar também estes círculos com os contornos menos delineados. Este segundo método acaba por ser uma adaptação do método usado para a detecção de quadriláteros e triângulos, uma vez que quando uma forma tem muitas arestas esta pode ser considerada um círculo. Este método passa então por reconhecer os contornos das formas geométricas e perceber quantas arestas é que estas têm. Para ser considerado um círculo a forma detectada deve ter sete ou mais arestas.

Depois dos dois métodos de detecção de círculos aplicados é necessário verificar se existe sobreposição de formas uma vez que estes dois métodos podem detectar o mesmo círculo. Para acabar com esta redundância é percorrida a lista de círculos detectados pelo segundo método e é verificado se o centro de um círculo está contido num outro detectado pelo primeiro método, caso isto se verifique um dos círculos é removido.

5.2 Variáveis visuais

Após a detecção das formas referidas na secção anterior, é necessário ter em conta quais as variáveis que se podem recolher. Para esta recolha foram estudadas as várias formas de uma maneira geral, ou seja, as formas foram analisadas de maneira a encontrar variáveis em comum. Posteriormente também foram analisadas em singular de maneira a encontrar variáveis mais específicas à forma geométrica correspondente.

5.2.1 Variáveis comuns a todas as formas

No que toca à generalidade das formas, as variáveis mais óbvias que surgiram foram o perímetro e a área. Estas duas variáveis foram calculadas facilmente, uma vez que os quadriláteros e os triângulos tinham as coordenadas dos vértices e os círculos tinham o tamanho do raio.

Outra variável analisada foi o centro das formas geométricas, ou seja, as coordenadas centrais x e y para cada forma. No caso dos círculos detectados através da função *HoughCircles* do *OpenCV* estas coordenadas eram devolvidas. No entanto, em relação à segunda forma de detecção foi necessário encontrar outro método. Nesse método, faz-se inicialmente a distinção entre formas com o número de arestas impares das pares. Caso o número de arestas seja par seleciona-se um vértice e encontra-se o seu vértice oposto de maneira a obter um segmento de recta. Depois calcula-se o ponto central desse segmento. Após obtidas essas coordenadas, o centro do círculo é também encontrado pois as coordenadas são as mesmas (ver figura 5.1).

No caso da forma ser impar também se começa por escolher um vértice e encontrar a aresta oposta. Posteriormente, calcula-se o centro dessa aresta de forma a encontrar as suas coordenadas x e y. Após isto, o processo é semelhante ao anterior. Encontra-se o segmento de recta entre o vértice escolhido no início e o ponto encontrado no passo anterior. Seguidamente, calcula-se o ponto central do segmento e obtém-se assim as coordenadas do centro do círculo (ver figura 5.2).

No caso dos quadriláteros escolhem-se dois vértices que sejam adjacentes e encontram-se os seus vértices opostos. Através de cada par de pontos é possível usar a expressão $y = mx + b$, uma vez que se consegue calcular o m com a expressão $m = \frac{y1 - y2}{x1 - x2}$, sendo o *x1* e *y1* as coordenadas de um ponto e *x2* e *y2* as coordenadas do ponto oposto. Em relação ao b basta substituir o x e o y pelas coordenadas de um dos vértices e o m pelo valor calculado anteriormente. No final destes passos, existem dois m's e dois b's correspondentes às duas rectas que vão ser utilizados para calcular o x através da expressão $x = \frac{b1 - b2}{m1 - m2}$ em que o *m1* e o *b2* correspondem a uma recta e o *m2* e o *b2* a outra. Com isto só falta calcular o y para obter as coordenadas do centro do quadrilátero e, para isso, usa-se a expressão já mencionada anteriormente: $y = mx + b$. No fim disto obtém-se o y que era a coordenada que faltava para saber o centro do quadrilátero.

Já no caso dos triângulos, para calcular o centro é necessário encontrar a aresta com a dimensão maior, ou seja a base do triângulo. Após isto é necessário encontrar o ponto central desta aresta para que seja possível encontrar o segmento de recta que une este ponto e o vértice oposto à base do triângulo. Com este segmento de recta é possível encontrar as coordenadas centrais da forma geométrica uma vez que o ponto central da recta é igual.

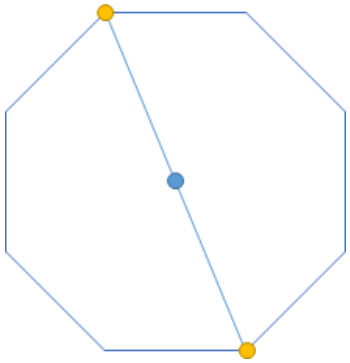


Figura 5.1: Centro de uma forma com arestas pares calculado através de dois vértices opostos.

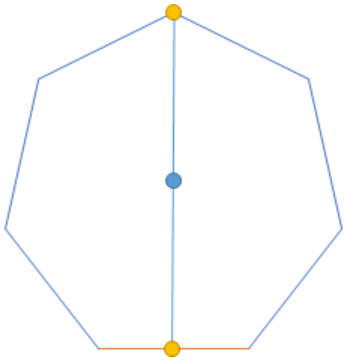


Figura 5.2: Centro de uma forma com arestas impares calculado através de um vértices e da sua aresta oposta.

Por fim a última variável analisada em comum para todas as formas geométricas foi a cor. Para recolher estes valores foram selecionados vinte e nove pontos de cada forma, de maneira a obter o valor RGB de cada ponto para mais tarde ser feita a média e obter um único valor para cada forma. Foram escolhidos apenas alguns pontos de maneira a que a análise das formas fosse mais rápida, uma vez que se esta fosse feita sobre todos os pontos contidos numa forma verificava-se que a aplicação ficava muito mais lenta. Além disso, com base nas experiências que foram realizadas, verificou-se que não existia grande alteração nos valores de RGB com uma amostra de pontos ou todos os pontos. Os pontos escolhidos são aqueles que estão nas imagens 5.3, 5.4 e 5.5. Estes pontos foram escolhidos pela facilidade de obtenção das suas coordenadas, uma vez que são pontos cujos cálculos já foram feitos anteriormente e também para que a aplicação seja mais rápida nesta fase de análise.

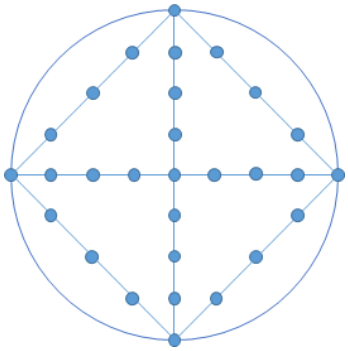


Figura 5.3: Pontos escolhidos para os círculos

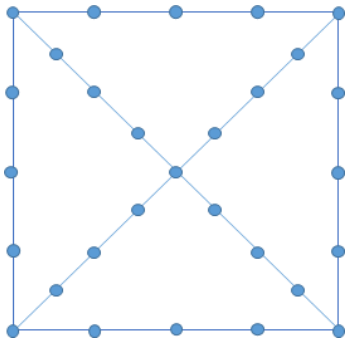


Figura 5.4: Pontos escolhidos para os quadriláteros

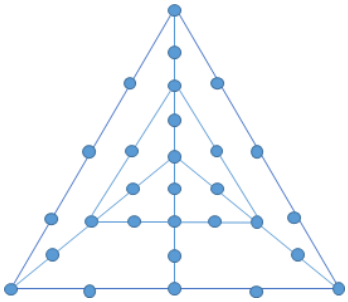


Figura 5.5: Pontos escolhidos para os triângulos

5.2.2 Variáveis específicas a cada forma

No que toca aos círculos, o raio foi a única variável considerada. Obter o raio no primeiro método de detecção foi bastante simples visto que o valor era devolvido pela função *Hough-Circles*. Em relação ao segundo método foi necessário escolher um vértice e usar o ponto central do círculo calculado anteriormente para que fosse possível calcular a distância entre estes dois pontos.

Para os quadriláteros, a variável específica analisada foi a rotação. Esta variável pode ter valores entre 0 e 179 e para a calcular é necessário encontrar a base do quadrilátero, ou seja, a sua maior aresta. Depois desta aresta ter sido encontrada é necessário encontrar qual dos vértices contidos na aresta tem a maior coordenada y para se conseguir encontrar

mais um ponto para se obter um triângulo. Esse ponto deve ter a coordenada y igual à do vértice com o y maior, e deve ter a coordenada x igual à do outro vértice. Tendo isto, consegue-se calcular o ângulo da rotação, uma vez que se consegue calcular o cateto oposto com o ponto obtido anteriormente e com o vértice com o y mais pequeno. Neste momento tem-se a maior aresta do quadrilátero, ou seja, a hipotenusa e um cateto oposto ao ângulo que se pretende calcular e através disto, usando o seno consegue-se obter o ângulo da rotação (ver figura 5.6). Basta ter atenção a alguns pormenores como por exemplo: quando o cateto oposto e a hipotenusa têm o mesmo tamanho o ângulo é de noventa, caso a coordenada x do vértice com o y maior seja maior que a coordenada x do outro vértice é necessário retirar a 180 o ângulo calculado por causa da inclinação da aresta. Já no caso das coordenadas y dos vértices serem iguais o ângulo da rotação é igual a zero.

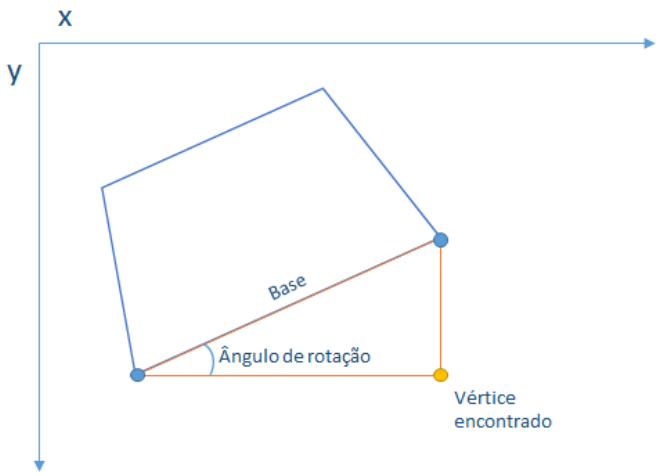


Figura 5.6: Exemplo da rotação de um quadrilátero

No caso dos triângulos a rotação também foi considerada, e o processo utilizado foi igual ao dos quadriláteros, ou seja, encontrar a base, calcular o cateto oposto ao ângulo que se pretende calcular e por fim calcular o ângulo da rotação.

Continuando nos triângulos foi também analisada outra variável, o tipo, ou seja, se o triângulo é equilátero, escaleno ou isósceles. Para encontrar o tipo do triângulo é necessário calcular primeiro os seus ângulos internos através dos vértices. Tendo estes pontos consegue-se obter três rectas e assim obter também três declives. Depois para calcular cada ângulo interno usa-se a expressão $\alpha = tan^{-1}(\frac{m1 - m2}{1 + m1 * m2})$.

Depois de calculados todos os ângulos internos é necessário compará-los, por exemplo se todos forem iguais significa que o triângulo é equilátero, se dois forem iguais é isósce-

les, e finalmente se forem todos diferentes é escaleno. Para obter melhores resultados por motivos de perspectiva foi atribuído um intervalo desvio de cinco graus nas comparações entre ângulos.

Capítulo 6

Geração Musical

Neste capítulo é descrita a criação de música, as dificuldades de implementação que surgiram, as bibliotecas analisadas e o processo escolhido. Foi neste tema que existiram mais dificuldades uma vez que este aspecto ainda não está bem desenvolvido em *smartphones*, mais especificamente em *Android*. Outro problema que surgiu, foi o facto dos formatos dos ficheiros de música e vídeo que se tornarem incompatíveis com o objectivo do projeto. No entanto, estas dificuldades foram ultrapassadas com a ajuda de bibliotecas ou com uma análise mais exaustiva do problema.

6.1 Processo

A música sempre foi o aspecto mais criativo do projeto, e por isso as soluções escolhidas teriam de estar sempre dirigidas para um *output* o mais variado e original possível. Neste sentido a solução mais óbvia para obter liberdade na criação de música seria usar o formato *midi*. Por exemplo o *midi* permite criar partituras com a ajuda de uma sintaxe própria. Tem também a capacidade de reproduzir, artificialmente, vários instrumentos, é fácil de se manipular, tem uma amplitude enorme de sons, ocupa pouco espaço, dá acesso a variáveis musicais (timbre, volume, etc.) e também é muito conhecido, o que se traduz em muita documentação. Apesar de todas estas vantagens, tem uma grande desvantagem de a música parecer muito artificial [64][32].

Por todas as vantagens já descritas anteriormente, foi então escolhido o *midi* para fazer a criação musical. Neste sentido foi necessário estudar qual a melhor abordagem para criar este tipo de música através de um *smartphone Android*. Depois de alguma procura, foram encontradas algumas bibliotecas que manipulavam este tipo de formato, mas todas tinham o mesmo problema: a falta de documentação, uma vez que a criação de música em

dispositivos móveis ainda está pouco desenvolvida. Durante esta procura a biblioteca que se destacou mais, por ter mais referências e alguns exemplos práticos, foi a *android-midi-lib* [1]. Esta biblioteca foi estudada com mais detalhe e foram feitos testes para verificar se era possível criar a música da maneira pretendida.

Foi então construída uma música de exemplo através de comandos que representavam a inserção de várias notas (em escala) e foi também experimentado a sobreposição de notas e vários tipos de instrumentos. Depois deste passo da geração de música foi necessário passar ao próximo, a agregação da música ao vídeo.

Verificou-se desde de início a dificuldade de encontrar uma biblioteca que fosse capaz de juntar a música ao vídeo, e dentro dos resultados encontrados as soluções nunca ofereciam compatibilidade com o formato *midi*.

Dentro dos resultados, existiam soluções que não referenciavam diretamente os formatos com que trabalhavam, e foram alvo de experimentação pois podiam ser compatíveis com *midi*. Este foi o caso de uma biblioteca incluída pelo suporte do próprio *Android*, uma vez que na documentação de áudio (em geral), referia que o sistema operativo suportava vários ficheiros, entre eles o *midi*. No entanto, esta biblioteca do *Android* chamada *MediaMuxer* [2], que permite juntar música a vídeos, na sua fraca documentação não referia incompatibilidade com os tipos de formatos referenciados na secção de áudio do sistema operativo. Tendo esta incógnita, foi então tentada esta abordagem de incorporar o *MediaMuxer* no projeto. Para o realizar foi necessário utilizar um exemplo disponibilizado na documentação, mas este tinha um problema pois estava incompleto e tinha chamadas a funções que não existiam dentro do suporte do *Android*. Este problema era com funções que pretendiam extrair informação da música de maneira a gerar o *output* certo para que fosse processado pelo resto do código de exemplo. Com isto, foi necessário fazer mais pesquisas para solucionar o problema, e dos poucos resultados encontrados todos referenciavam duas outras bibliotecas do *Android* chamadas *MediaExtractor* [3] e *MediaCodec* [4]. Integradas as bibliotecas os resultados eram negativos, não existindo compatibilidade com o *midi*.

Então decidiu-se abandonar esta abordagem uma vez que os resultados esperados não correspondiam com os obtidos e também pelo facto de a versão mínima do *Android* para utilizar estas novas bibliotecas ter aumentado para versão 4.3. Esta versão 4.3 é muito

[1] mais informações em: https://code.google.com/p/android-midi-lib/
[2] mais informações em: https://developer.android.com/reference/android/media/MediaMuxer.html
[3] mais informações em: http://developer.android.com/reference/android/media/MediaExtractor.html
[4] mais informações em: http://developer.android.com/reference/android/media/MediaCodec.html

recente e por isso não era benéfico desenvolver a aplicação nestas condições pois o público alvo seria muito pequeno como se pode verificar na tabela 6.1.

Versão	Utilizadores
2.2	0.7%
2.3.3 - 2.3.7	13.6%
4.0.3 - 4.0.4	10.6%
4.1.x	26.5%
4.2.x	19.8%
4.3	7.9%
4.4	20.9%

Tabela 6.1: Distribuição das versões do *Android* pelos utilizadores [65].

A abordagem seguinte foi alterar o formato de *midi* para outro que fosse capaz de cumprir com os objetivos. O passo seguinte foi tentar encontrar uma maneira de converter noutro formato. Depois de muito tempo sem se ter obtido resultados, percebeu-se que não existia nada que fosse capaz de converter aquele formato noutro, em *Android*. Uma razão encontrada para existir pouca compatibilidade com este formato, foi o facto de não existir um som mas sim uma nota, ou seja, o ficheiro é interpretado e é tocado um som correspondente mas esse som pode ser completamente diferente noutro dispositivo. E sendo a música um tema pouco explorado em dispositivos móveis e este um problema complexo, não existe ainda uma solução. Uma das soluções seria atribuir esta tarefa de converter o *midi* noutro formato a um servidor, mas esta solução criava dependência do acesso à internet que seria um fator que iria desmotivar o público alvo. O facto de existir mais dados na comunicação também era desmotivador pois o utilizador possivelmente pretende usar a aplicação em qualquer lado e caso este não tenha acesso a internet sem fios teria de usar o seu plano de dados o que não seria o indicado para a utilização por parte do público alvo.

Decidiu-se então abandonar de vez este formato *midi* e procurar outras maneiras de se criar música. Outro formato é era capaz de recriar as funcionalidades do *midi* era o *ABCnotation* [5]. Através de um ficheiro *xml*, este é capaz de criar música, mas com o aprofundar deste tema percebeu-se que o problema que o *midi* apresentava acontecia também com este formato. Para além disso o problema era ainda mais difícil de resolver neste caso em particular, porque este formato não é tão conhecido como o *midi* logo existe menos documentação para resolver qualquer tipo de problema. O *ABCnotation* foi também descartado como solução para este projeto.

[5] mais informações em: http://abcnotation.com/

Como os formatos mais fáceis de manipular têm problemas de compatibilidade no que toca à inserção nos vídeos, foi necessário recorrer a outro tipo de solução. Uma alternativa estudada foi utilizar uma biblioteca capaz de gerar som, como por exemplo um sintetizador. Foram encontradas algumas, mas nenhuma permitia gravar o *output* num ficheiro. Todas estas bibliotecas tinham pouca ou nenhuma documentação o que tornou difícil perceber as suas funcionalidades. Ainda foram feitos alguns testes em algumas bibliotecas mas estas não referiam a sua incapacidade de gravar *output* como foi o caso do *Music Synthesizer for Android* [6]. Após estes testes percebeu-se que não existia uma maneira de poder guardar o *output* num ficheiro com estas bibliotecas.

Após se verificar que estas soluções mais fáceis de manipular não tiveram sucesso foi necessário, mais uma vez, mudar de abordagem. Para isso a ideia inicial teve de ser um pouco modificada, ou seja, em vez de a música ser construída dinamicamente através de notas e outras variáveis musicais, a música passa a ser formada através de pequenos ficheiros de som. Estes ficheiros iriam ser sobrepostos e concatenados consoante o *input* das variáveis visuais e no final era obtida a música. Para se conseguir escolher o tipo de formato correto foi necessário estudar qual seria a biblioteca que iria ser usada para realizar a inserção da música ao vídeo. Já tinha sido encontrada uma biblioteca capaz disto que fazia parte do suporte do *Android* mas esta tinha um problema. O problema era o facto da versão de *Android* mínima para correr a aplicação ser demasiado alta (versão 4.3), e por isso a quantidade de utilizadores que a poderiam vir utilizar era muito menor (como se pode ver na tabela 6.1). Neste sentido foi encontrada uma solução que era preferida entre os programadores de aplicações para o sistema operativo *Android* chamada *MP4Parser* [7].

O *MP4Parser* é uma biblioteca *open source* que funciona a partir da versão 2.3.3 do *Android*. Esta biblioteca é capaz de juntar vídeo com música desde que estes tenham formatos compatíveis. Em relação à música os formatos devem ser *mp4* ou *m4a*, já o vídeo deve ter o formato *mp4* ou *m4v*. Estes formatos de vídeo são normalmente usados pelo *Android*, mais concretamente pela aplicação da câmara que está contida no sistema operativo. Esta biblioteca pode fazer com que hajam mais utilizadores uma vez que a versão mínima necessária é mais baixa que as outras soluções existentes, os formatos compatíveis permitem que as pessoas que já têm um vídeo antigo o possam usar com esta aplicação. Além disto a documentação acaba por ser melhor que a do *MediaMuxer*. Tendo todos estes factores, o *MP4Parser* foi escolhido para integrar o projeto.

[6] mais informações em: https://code.google.com/p/music-synthesizer-for-android/
[7] mais informações em: https://code.google.com/p/mp4parser/

No entanto, a escolha da biblioteca *MP4Parser* acabou por limitar os formatos de áudio que podiam ser usados. Neste sentido era necessário utilizar os ficheiros com estes formatos e sobrepô-los. Foram feitas algumas tentativas sobre isto com a ajuda desta biblioteca, mais especificamente foi experimentado o uso de *layers*. Assim, cada ficheiro ficava numa *layer* diferente e estava concluída a sobreposição. No entanto, esta solução não correu como esperado, uma vez que quando a música era gerada apenas se ouvia a última música que tinha sido adicionada, ou seja, as outras tinham sido substituídas.

Neste sentido foi necessário encontrar uma solução capaz de sobrepor várias músicas, uma vez que o *MP4Parser* consegue apenas concatenar e juntar músicas com vídeo. Após muita pesquisa não foi encontrada nenhuma solução capaz de fazer o pretendido. No entanto, existiam algumas sugestões para se usar o *PCM* para realizar a sobreposição [66]. O *PCM* é aconselhado porque os dados da música não têm qualquer tipo de compressão e normalmente o formato mais usado para isto é o *wav* (que não é compatível com o *MP4Parser*). Com isto, todos os ficheiros de som foram convertidos para *wav* e posteriormente eram carregados para aplicação em *PCM*, neste caso em *arrays* de *shorts*. Depois disto os valores contidos nos *arrays* são percorridos e são divididos por 32768 uma vez que os *shorts* têm valores entre -32768 e 32767. Desta maneira obtêm-se valores entre menos um e um que são posteriormente somados para se conseguir gerar uma única música a partir de várias. Caso esta soma seja superior a um ou inferior a menos um, os valores passam a ser um e menos um respectivamente. Também é nesta altura que o volume da música pode ser ajustado. Finalmente os valores calculados são multiplicados por 32768 para voltarem a ser guardados em shorts. Desta maneira obtém-se novamente o *pcm*, que depois vai ser guardado no formato *wav*. Para isso é necessário configurar o cabeçalho do *wav* com a frequência correta, a duração, o número de canais, entre outros [67].

No final de todos estes passos era necessário converter este *output* para outro formato que fosse compatível com o *MP4Parser*. Após alguma pesquisa verificou-se que não existia nenhuma solução para converter de *PCM* para *mp4* ou *m4a*, e a única opção seria gravar com o formato *wav* novamente. Como a solução era então gravar em *wav* foi necessário estudar este formato com mais detalhe para perceber os campos do cabeçalho. Após ter um cabeçalho correto foi guardada a informação com os vários sons sobrepostos num ficheiro *wav*. No entanto, este formato era incompatível com o *MP4Parser*, logo era impossível adicionar uma música ao vídeo. Foi então necessário converter este formato para um que fosse compatível. Após algum estudo, verificou-se que a maioria indicava o uso da biblio-

teca *javasound* [8] que já vem incluída com o *java*. No entanto, implementar a aplicação não reconhecia os métodos chamados pela biblioteca e era impossível importá-la. Verificou-se então que a biblioteca tinha sido removida do *Android*, ou seja, esta solução não era válida para a aplicação móvel. No entanto existiam outras bibliotecas que tinham a funcionalidade de converter os formatos, mas depois de se verificar a documentação percebeu-se que um dos requisitos era a biblioteca *javasound*.

Após muita pesquisa, foi encontrada uma solução que apesar de não ser a melhor, cumpria os requisitos e como já se tinha dispendido muito tempo nesta tarefa foi necessário seguir por este caminho. Esta solução era a biblioteca *android-aac-enc* [9], e esta faz uso da biblioteca *MP4Parser* modificada. Por causa destas modificações foi necessário ter em conta todos os conflitos que poderiam existir entre a versão original e modificada. A biblioteca *android-aac-enc* é capaz de receber o ficheiro *wav* como *input* e convertê-lo para *aac*. No entanto, o formato *aac* também não era compatível com o *MP4Parser* de maneira que ainda é necessário converter este formato para outro que fosse compatível. Então foi necessário passar de *aac* para *m4a* também com esta biblioteca.

Depois das várias sobreposições de sons e de todos os ficheiros *m4a* gerados foi necessário concatená-los e incorporá-los no vídeo. Este passo já foi mais fácil uma vez que o *MP4Parser* conseguia fazer estes dois passos. No final foi gerado um vídeo em *mp4* com a música criada nos outros passos.

6.2 Geração de música

Depois de se ter obtido uma solução funcional foi necessário definir como é que as variáveis visuais iriam influenciar a geração de música. Inicialmente foram recolhidas várias variáveis que já foram referenciadas anteriormente, mas depois de algum estudo foram escolhidas aquelas que representavam maiores variações e que permitiam mais *outputs* diferentes numa tentativa de exponenciar a criatividade musical.

De todas as variáveis anteriormente foi utilizado o número de formas, ou seja, a quantidade de um tipo de forma geométrica dentro de uma *frame*. Outra variável escolhida foi a área média de um determinado tipo de forma numa *frame*. Por fim foi utilizado o centro médio de cada tipo de forma de uma *frame* para criar outra variável chamada posição. Esta posição utiliza a largura da *frame*, o que corresponde ao eixo dos *xx*, e é encontrado

[8] mais informações em: http://www.oracle.com/technetwork/java/index-139508.html
[9] mais informações em: https://github.com/timsu/android-aac-enc

ponto central daquele segmento de recta, ou seja, a largura é dividida ao meio para se obter este valor. Depois, verifica-se se o centro de um tipo de formas é menor ou maior que o valor encontrado no passo anterior. Caso seja menor, a posição do conjunto de formas é negativa, caso contrário é positivo, obtendo-se assim uma variável binária (ver figura 6.1).

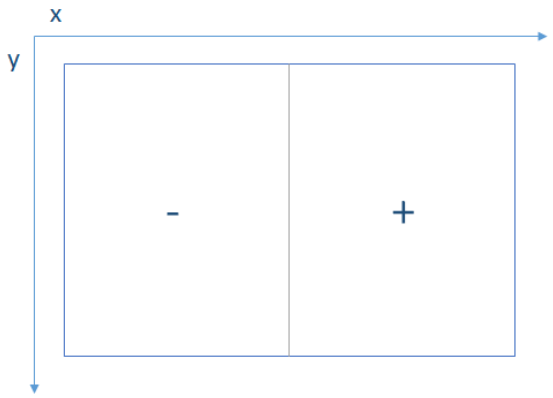


Figura 6.1: *Frame* dividida ao meio, com uma área negativa e outra positiva. Equivalente à posição recolhida através dos conjuntos de formas.

Após estarem estas variáveis visuais definidas foi necessário definir o processo de geração de música. Neste sentido, foi pensado utilizar pequenos ficheiros de som que seriam tocados quando determinada variável musical aparecesse. Logo nos primeiros testes existiu um problema que era o facto de por vezes não existirem formas geométricas no vídeo. Caso estas não existissem a composição musical não teria música. Então a solução encontrada foi ter sempre música de fundo mesmo quando as formas geométricas são inexistentes.

Como a análise de *frames* é um processo que demora muito tempo foi necessário arranjar uma solução que melhorasse a rapidez de análise e criação de uma composição musical. Para conseguir isto são analisadas apenas dez *frames* por segundo, o que reduz o tempo de processamento. São então analisadas dez *frames* em cada segundo, e de seguida é feita a média de cada variável para cada tipo de forma. Desta maneira é possível escolher que tipo de som é adequado para aquele segundo.

Inicialmente cada som tinha três segundos de duração, e estes ficheiros eram divididos em outros três ficheiros de apenas um segundo. Assim por cada segundo, dez *frames* eram analisadas, eram recolhidas as variáveis e era adicionado um ficheiro de um segundo. Caso as condições das variáveis musicais se mantivessem era adicionado o próximo ficheiro relativo ao mesmo som. No entanto, estes resultados não estavam a ser suficientemente satisfatórios, uma vez que a música gerada fazia pouco nexo e por vezes soava mal com todos aqueles sons.

Depois de algum estudo e de aconselhamento por parte do Professor Pedro Martins [10], percebeu-se que o caminho a seguir, para que a música criada fosse mais agradável, seria uma espécie de sequenciador. Para realizar isto foi necessário encontrar músicas que fossem *loops* e que tivessem o mesmo número de *bpm's*(*beats per minute*) para que as músicas soassem bem ao ouvido humano. Tendo isto foram pesquisados alguns ficheiros de *loops* gratuitos e de seguida foram testados no programa *Audacity* [11]. Foram analisados outros programas para testar sobreposições, criar efeitos e editar áudio na generalidade. Dentro destes programas foram analisados o *Audacity*, o *GoldWave* [12] e o *Adobe Audition* [13], mas todos estes eram pagos excepto o *Audacity* que tinha as mesmas funcionalidades que os outros. Além disso o autor já tinha alguma experiência com o *Audacity*, e por isso este acabou por ser escolhido. Experimentaram-se várias sobreposições no *Audacity* entre os ficheiros para se perceber se o conjunto era agradável. Depois de escolhidos os conjuntos foi necessário obter ainda mais ficheiros que pudessem lhe pertencer, uma vez que atualmente um conjunto tinha apenas sete ficheiros de áudio. Foi então feita a divisão da seguinte forma: dois para cada forma e um para a música de fundo, mas isto era muito pouco para as variáveis visuais. No sentido de se obter mais ficheiros de áudio foram testados vários efeitos para se poder replicar os *loops*. Então foram estudados alguns efeitos que não pusessem em causa a sobreposição dos *loops* quanto a soarem bem. Dentro destes efeitos do *Audacity* foi escolhido o *wah wah*. Com isto, cada ficheiro acaba por se dividir em mais dois uma vez que estes dois ficheiros tem o efeito com intensidades diferentes. Assim caso a variável seja maior que a anterior o efeito aplicado vai ter uma intensidade também maior.

Estes *loops*, que têm todos a mesma duração (superior a sete segundos), foram divididos em ficheiros com a duração de um segundo para que possam ser adicionados assim que exista uma variável que o indique. Após alguns testes, chegou-se à conclusão que a ordem e o momento que estes sons eram adicionados era bastante importante. Inicialmente estes sons eram adicionados e se a variável que os despoletou permanecesse semelhante o som escolhido seria o próximo ficheiro daquele *loop*. Mas, neste caso o primeiro ficheiro do *loop* seria sempre escolhido caso aparecesse a variável correspondente independentemente do momento. Neste caso existia a tendência da sobreposição soar mal, uma vez que os *loops* tinham sido feitos para serem tocados ao mesmo tempo, ou seja, se o primeiro *loop* estava no primeiro segundo, o *loop* que ia ser sobreposto tinha de estar também no primeiro segundo. Tendo isto, foi necessário descobrir em que segundo vai o vídeo para poder escolher

[10] mais informações em: www.dei.uc.pt/ pjmm
[11] mais informações em: http://audacity.sourceforge.net/
[12] mais informações em: http://www.goldwave.com/
[13] mais informações em: https://creative.adobe.com/products/audition

o ficheiro correspondente. Caso o segundo em que vai o vídeo fosse maior que o número de segundos dos *loops* voltava-se ao início dos *loops*.

Capítulo 7

Desenvolvimento da Aplicação

Neste capítulo é feita a análise de como a aplicação realiza a geração da composição e do *upload* da composição. São ainda apresentados os passos principais que dizem respeito ao processo escolhido para estas duas áreas.

7.1 Geração da composição

Nesta secção é explicado o processo atual que aplicação segue para gerar uma composição vídeo-musical. Tendo em conta os passos da análise de vídeo e da geração geração de música.

Para iniciar o processo é necessário que o utilizador tenha um vídeo que possa ser analisado. O utilizador pode usar um vídeo que já esteja na memória do *smartphone* ou pode filmar um vídeo no momento e utilizá-lo.

Depois do vídeo escolhido, é necessário que o utilizador espere um pouco para que a aplicação consiga analisar o vídeo e criar a composição a partir do mesmo. É nesta altura que a aplicação começa a analisar as dez *frames* por segundo com as suas devidas preparações, desde de filtros de *blur*, de *canny* e de escala de cinzentos. A utilização destes filtros é descrita com mais detalhe na secção 8.1. A aplicação analisa todas as possíveis formas que possam existir sejam elas círculos, triângulos ou quadriláteros e guarda as diversas formas e o id da *frame* assim como todas as variáveis selecionadas já referidas no capítulo 5.

Depois de todas as variáveis recolhidas passa-se então à construção da música. O primeiro passo é escolher a música de fundo para que a composição tenha sempre música mesmo que não existam formas. Esta é escolhida através de duas variáveis que dizem respeito a

todas as formas: o tamanho e o número de formas geométricas. A posição foi retirada das variáveis escolhidas uma vez que só existe uma música de fundo, que depois sofre algumas alterações através de efeitos. Esta música de fundo deve tentar passar despercebida, e existe alguma dificuldade em obter este tipo de músicas que fiquem bem com as outras que vão ser sobrepostas. Por isso optou-se por ter apenas uma música e retirar a variável posição porque era aquela que variava menos, e numa tentativa de se obter resultados mais diferenciados permaneceram as outras variáveis.

Após a análise das *frames* e da recolha de formas, utiliza-se o número de formas encontradas para escolher uma componente da música. Esta componente indica se a música terá um efeito, chamado *wahwah* feito pelo *Audacity*. Caso o número de formas seja no máximo nove a música não terá este efeito, mas se existirem entre dez e vinte e nove formas a música terá o efeito porém com pouca intensidade. No caso do número de formas ser superior a trinta o efeito também será adicionado mas com uma intensidade maior (ver figura 7.1).

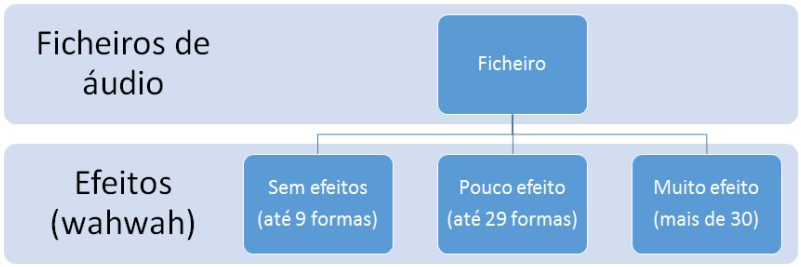


Figura 7.1: Efeitos sobre a música de fundo.

De seguida é utilizada a variável correspondente ao tamanho médio de todas as formas encontradas. Associado a esta variável está o volume da música, ou seja, quanto maiores forem as formas mais alto será o volume final. O volume mínimo da música é de cinquenta por cento uma vez que com valores menores a música já ficava com um volume demasiado baixo e também pelo facto da sobreposição de músicas o exigir, ou seja, caso exista uma música com um volume muito baixo e outra muito alto depois de sobrepostas a música final não fica agradável.

Posteriormente à seleção da música de fundo é necessário fazer as sobreposições das outras músicas. Para isso são analisadas as formas de cada tipo recolhidas nas dez *frames* correspondentes a cada segundo. Cada forma tem duas músicas originais, sem quaisquer efeitos, e por isso é necessário escolher aquela que vai ser usada. Utilizando as formas

de um determinado tipo que foram recolhidas no primeiro segundo, mais propriamente a variável da posição média das formas é possível selecionar uma música; caso a posição seja negativa é escolhida uma música, caso contrário é escolhida a outra. De seguida é utilizada outra variável, o número de formas geométricas do tipo que está a ser analisado. Através desta variável vai ser decidido o uso ou não do efeito *wahwah* referido anteriormente.

O processo é semelhante ao que foi utilizado na música de fundo, no entanto os limites usados são diferentes. No caso de existirem no máximo quatro formas a música não terá o efeito mas a partir de cinco até nove formas o efeito já estará presente. Caso existam mais de dez formas, inclusive, o efeito usado terá mais intensidade (ver figura 7.2). Depois disto, apenas é usada a variável associada ao tamanho das formas para definir o volume da música. Este processo é igual aquele utilizado na definição do volume da música de fundo. Após os ficheiros de áudio escolhidos correspondentes às formas (ou não, caso não existam formas) e com o ficheiro escolhido para a música de fundo é necessário os sobrepô-los. Para isto é preciso converter a sua extensão de *wav* para *pcm*. Analisam-se os sinais de áudio, ajusta-se o volume, o *fade-in* e o *fade-out* caso necessário para cada ficheiro, e depois sobrepõem-se os sinais para gerar um único ficheiro de áudio. Converte-se novamente de *pcm* para *wav* e posteriormente de *wav* para *aac* através da biblioteca *android-aac-enc*. Por fim converte-se este formato *aac* para *m4a*, que vai ser guardado com o nome correspondente ao segundo do vídeo que está a ser analisado. Acabada a análise do primeiro segundo do vídeo é necessário realizar o mesmo processo para os restantes segundos.

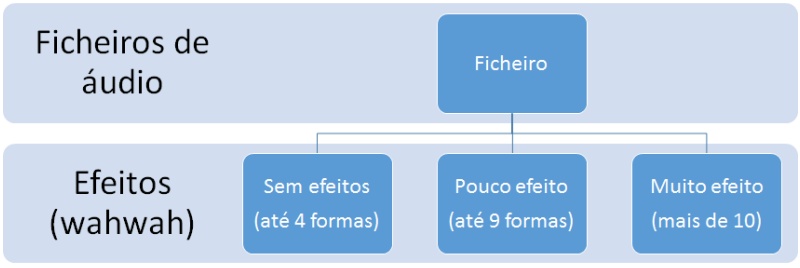


Figura 7.2: Efeitos sobre a música de um determinado tipo de forma.

Após todos os segundos analisados, é feita a criação dos vários ficheiros para todos os segundos do vídeo. De seguida, é necessário concatená-los e adicioná-los ao vídeo para que este se torne numa composição vídeo-musical. Neste sentido é necessário usar a biblioteca *MP4Parser*, o vídeo escolhido no primeiro passo e os ficheiros áudio gerados. Neste momento são adicionados os ficheiros à fila (por ordem crescente uma vez que estes foram guardados com o nome correspondente ao segundo a que pertence). A biblioteca

MP4Parser consegue concatenar os ficheiros de áudio e juntá-los ao vídeo e é neste momento que é gerada a composição. Para finalizar o processo o utilizador atribui-lhe um nome e faz *upload* para o servidor.

7.2 *Upload* da composição

Após todo o processo de geração de uma composição o utilizador tem a possibilidade de fazer o *upload* da mesma. Para isto é apenas necessário atribuir um nome à composição e de seguida a composição é enviada para a conta do *Youtube* associada ao projeto. A conta é mesmo do projeto e não do utilizador visto que assim é possível obter mais controlo sobre as composições. Caso contrário existiria por exemplo, o risco de o utilizador retirar o vídeo sem qualquer aviso, ou seja, perderia a integridade dos dados. Para além disso, outro motivo por se usar uma conta própria é o facto desta poder ser considerada uma conta de *Youtuber*, ou seja, é possível esta gerar dinheiro com o número de visitas e construir a partir disto uma fonte de receitas.

Após concluído o *upload*, a aplicação recebe o *id* do vídeo. Através deste *id* é possível aceder ao vídeo e obter as imagens de *preview* do mesmo. A aplicação neste momento encarrega-se de enviar o *id*, o nome da composição para o servidor e a informação sobre as formas geométricas detectadas, ou seja, apenas é enviada uma *string* em *json* para o servidor. Esta informação é utilizada para criar uma instância da composição que é automaticamente associada ao utilizador que a criou. Desta maneira a base dados usada terá apenas que lidar com informação relativamente pequena quando comparada ao tamanho ocupado por uma composição.

Capítulo 8

Resultados

Neste capítulo são apresentados os resultados dos testes e experimentações realizados sobre a análise de vídeo e da geração de música. É ainda feita uma comparação com algumas aplicações semelhantes a este projeto.

8.1 Análise de vídeo

A análise de vídeo foi uma das componentes que sofreu algumas melhorias derivadas dos testes. Estas melhorias prendem-se com o facto de ser possível detectar mais e melhor formas geométricas, uma vez que existiam, inicialmente, algumas formas que não eram reconhecidas e outras que eram detectadas mas não correspondiam a nenhuma forma.

Para reduzir estes erros foram usados alguns filtros que eram capazes de melhorar a *frame* que estava a ser analisada. O primeiro filtro que foi aplicado foi o *blur*, através da função *GaussianBlur* do *OpenCV*. Esta função é importante para reduzir erros na detecção de formas que na verdade não as são. No entanto, foram feitos muitos testes sobre isto, uma vez que era necessário encontrar a intensidade certa de *blur*. Para realizar estes testes foram usadas algumas imagens e vídeos de exemplo, como é o caso da figura 8.1.

Depois de aplicado o filtro obteve-se o resultado apresentado na figura 8.2. Neste caso a *frame* tem uma intensidade de *blur* demasiado alta, uma vez que a aplicação não é capaz de detectar nenhuma forma. Neste sentido foi necessário realizar várias experiências para baixar a intensidade gradualmente até ser encontrada a desejada. Quando esta foi encontrada conclui-se que a intensidade deve ser muito baixa, apenas o suficiente para reduzir algum "ruído" presente na *frame* (ver figura 8.3).

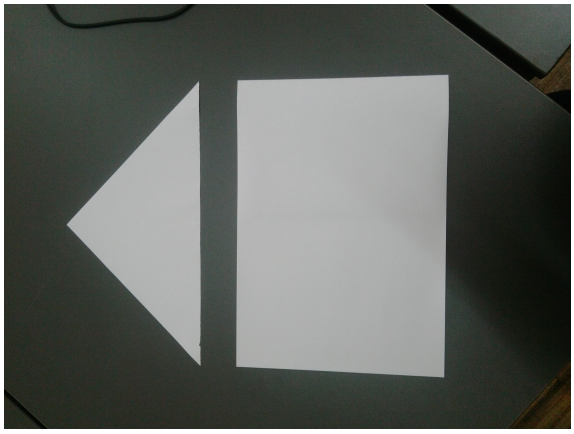


Figura 8.1: Imagem exemplo para testes.

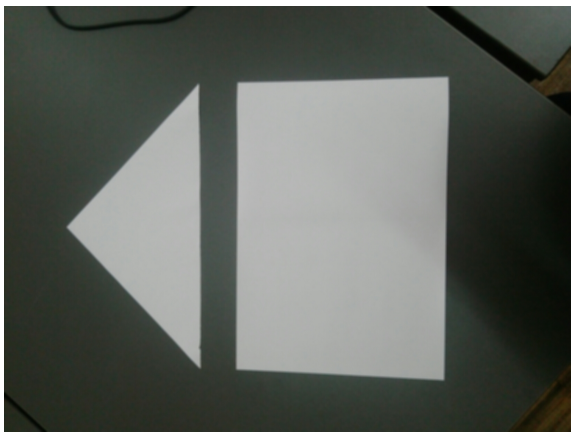


Figura 8.2: Imagem exemplo com *blur* com uma intensidade alta.

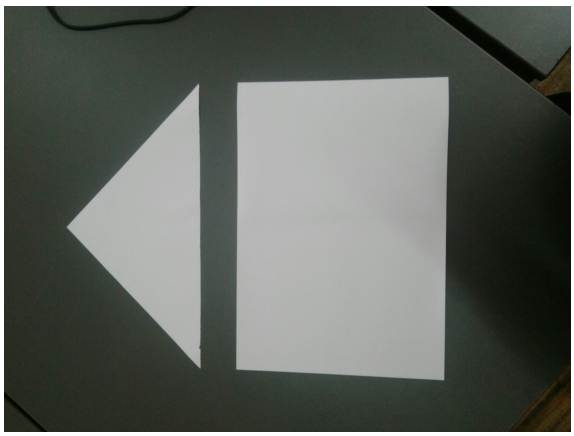


Figura 8.3: Imagem exemplo com *blur* com uma intensidade ideal.

No entanto a *frame* teve de ser analisada de maneira diferente para serem detectados os triângulos, uma vez que estes apenas têm três arestas podem ser facilmente confundidos com algo que na verdade não seja um triângulo. Este problema foi detectado com mais frequência em situações em que a *frame* tinha mais luminosidade. A solução encontrada para resolver este problema foi simplesmente aumentar mais um pouco a intensidade do *blur* utilizado anteriormente, e assim ter uma intensidade para quadriláteros e círculos e

outra para triângulos.

Posteriormente foi adicionado outro filtro do *OpenCV* chamado *Canny* que tem a capacidade de delimitar os contornos de objetos através de alterações bruscas no que toca à cor. Este filtro é bastante importante para mais tarde ser possível detectar corretamente os contornos das formas geométricas. Depois do filtro aplicado obteve-se a seguinte figura 8.4 que apresenta um detalhe demasiado elevado, uma vez que se consegue ver ainda algum "ruído".

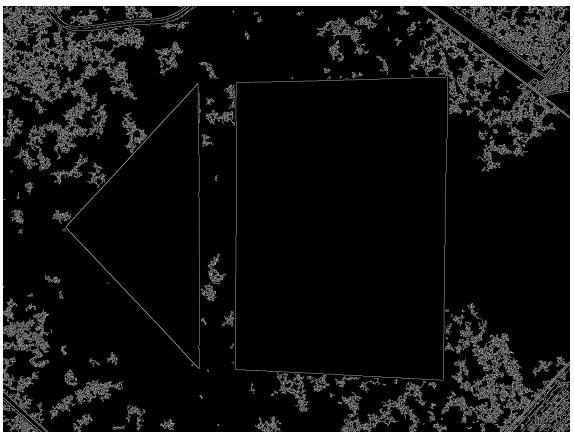


Figura 8.4: Imagem exemplo com *canny* com um detalhe demasiado elevado.

Assim, como no filtro anterior, foi necessário ir reduzindo o detalhe do filtro gradualmente até se encontrar o desejado. O nível de detalhe que foi encontrado é aquele que está presente na figura 8.5. Ainda se pode ver algum ruído presente na *frame*, e por isso houve uma tentativa de o retirar como se pode ver na figura 8.6. No entanto esta tentativa não era a melhor uma vez que existiam algumas formas que não era detectadas, pois nem todos os contornos eram encontrados como se pode reparar no canto superior esquerdo da figura 8.6. Tendo assim estes resultados conclui-se que era preferível existir algum "ruído", uma vez que existiam mais formas geométricas reconhecidas corretamente.

O *blur* e o *canny* são dois filtros muito importantes para o reconhecimento de formas geométricas, porque os dois em conjunto conseguem eliminar muitos erros que poderiam acontecer caso estes não fossem usados. Um exemplo disto é a figura 8.7 em que não é usado o filtro de *blur* e se pode verificar a quantidade de ruído que está presente na *frame*. Todo este ruído iria provocar a detecção de falsas formas geométricas.

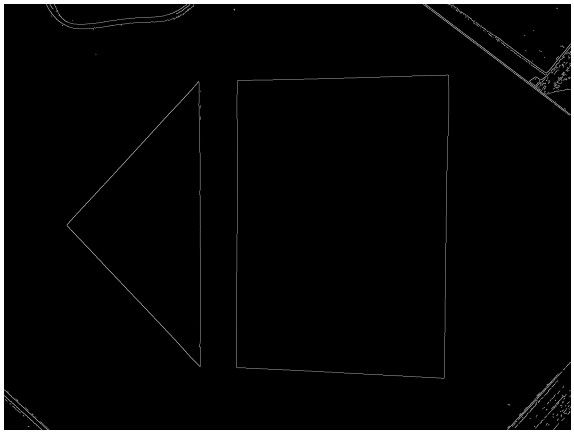


Figura 8.5: Imagem exemplo com *canny* com um detalhe ideal.

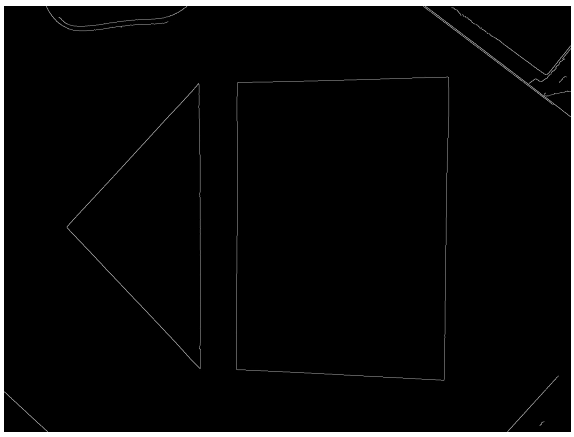


Figura 8.6: Imagem exemplo com *canny* com um detalhe demasiado baixo.

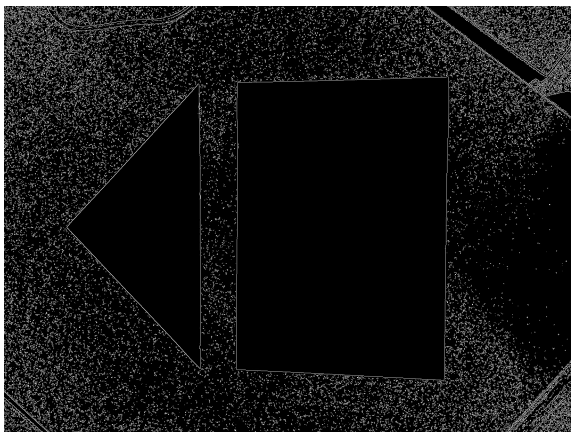


Figura 8.7: Imagem exemplo com *canny* e sem *blur*.

Outro aspecto que foi alvo de alguns testes foi a função *HoughCircles* do *OpenCV* que tinha alguns parâmetros como o raio mínimo e máximo de um círculo para que fosse detectado. Inicialmente o raio mínimo tinha um valor demasiado baixo e isso gerava muitos círculos que na realidade não existiam e além disso aumentavam o tempo de processamento. Esse parâmetro foi gradualmente aumentado de maneira a encontrar um tamanho onde não se verificasse os erros encontrados anteriormente. No caso do raio máximo este

inicialmente também era um valor demasiado grande o que aumentava também o tempo de processamento, e por isso foi um pouco diminuído para melhorar este tempo mas sempre tendo em conta a quantidade de círculos que eram detectados. Neste caso, através de todas as experiências feitas, todos os círculos que foram detectados com o raio máximo inicial foram detectados com o raio máximo final.

Por fim, houve ainda outro aspecto que foi importante para eliminar falsas formas geométricas, mais particularmente no caso dos quadriláteros e dos triângulos. Este aspecto está relacionado com o problema identificado na figura 8.8, que apresenta formas que são detectadas devido a ruído presente na *frame*.

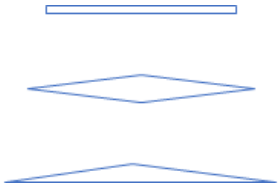


Figura 8.8: Falsas formas geométricas.

Para resolver este problema no caso dos quadriláteros, foram comparadas as arestas e as diagonais, ou seja, a aresta mais pequena tem que ter pelo menos 30% do tamanho da aresta maior, a segunda aresta menor tem que ter também pelo menos 30% da aresta que resta. Para além disso, a diagonal menor tem que ser maior do que 30% da outra diagonal. No caso dos triângulos é analisada a altura do triângulo e a aresta maior, ou seja, a altura deve ter no mínimo 30% do tamanho da aresta maior. Tendo isto, era necessário verificar se os resultados correspondiam com o que era esperado e para isso foi feita uma imagem de teste (ver figura 8.9), em que as formas válidas são as três que estão no topo.

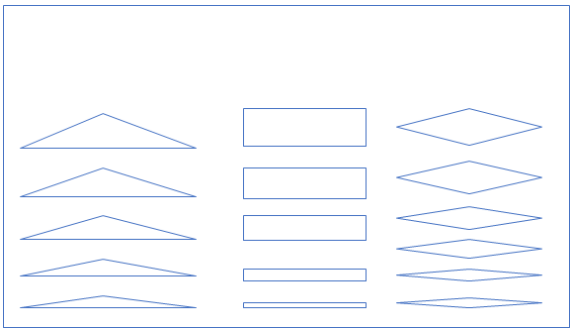


Figura 8.9: Imagem de teste para verificar as formas.

No entanto ainda existiam muitas formas geométricas que eram originadas através de erros de detecção. Estas formas eram originadas pelo ruído que era demasiado pequeno e que

parecia uma determinada forma geométrica. Para evitar isto foi utilizada a área de cada forma e a área de uma *frame*, ou seja, para se considerar uma forma válida é necessário que esta tenha pelo menos 2,5% da área total da *frame*. Para validar os quadriláteros foi usada a figura 8.10 em que o único quadrilátero válido é o maior. Assim como existe a imagem teste para os quadriláteros existe também para os triângulos (ver figura 8.11), os círculos detectados pelo primeiro método (ver figura 8.12) e pelo segundo método (ver figura 8.13).

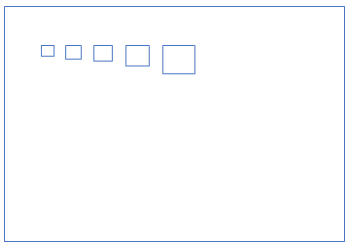


Figura 8.10: Imagem de teste para verificar a área dos quadriláteros.

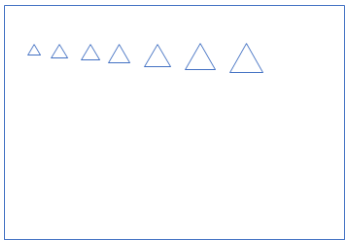


Figura 8.11: Imagem de teste para verificar a área dos triângulos.

Figura 8.12: Imagem de teste para verificar a área dos círculos (primeiro método).

8.2 Geração de música

A geração de música também foi alvo de muita experimentação, sendo a maior parte realizada através do *Audacity*. O *Audacity* serviu inicialmente para escolher os *loops* corretos,

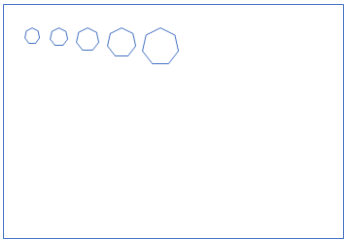


Figura 8.13: Imagem de teste para verificar a área dos círculos (segundo método).

uma vez que era necessário ver se os ficheiros encontrados eram compatíveis (ver figura 8.14). Para estes serem compatíveis têm que ter a mesma batida por minuto, caso isso não acontecesse o *output* originado era de fraca qualidade pois obtém-se uma música onde não parece haver sintonia e harmonia. Depois de muitos ficheiros analisados foi feita uma divisão por conjuntos, uma vez que uns *loops* ficavam bem apenas com alguns *loops*. O maior conjunto conseguido tem apenas sete loops, e por isso foi necessário criar mais *loops* a partir destes. Neste sentido foram usados efeitos, neste caso o *wahwah* do *Audacity* com diferentes intensidades. Mais uma vez estas intensidades foram testadas no conjunto todo para perceber se a sintonia e harmonia que antes existia ainda continuava a existir, e também se o efeito tinha uma intensidade suficiente para que pudesse ser ouvido. Foram testados muitos efeitos disponíveis no *Audacity*, como por exemplo o tremolo, o eco, o reverberar, entre outros, mas muitos passavam despercebidos e outros alteravam demasiado os *loops*, o que acabava por quebrar a harmonia entre eles. Por isso, acabou por ser escolhido o efeito *wahwah* que consegue melhores resultados.

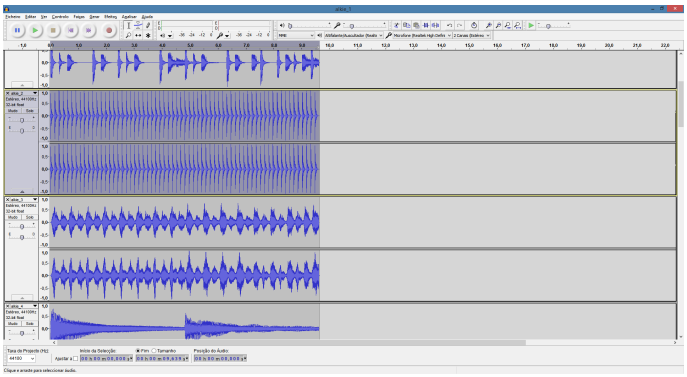


Figura 8.14: Escolha de *loops* no *Audacity*.

Também houve muita experimentação feita na aplicação, uma vez que à medida que esta ia sendo testada apareciam alguns problemas. Um dos problemas que apareceu, foi quando os *loops* foram divididos em ficheiros de um segundo e estes eram tocados por outra ordem que não a original, ou quando existia sobreposição de ficheiros de segundos diferentes. Isto originava *outputs* em que não existia harmonia, apesar de estes terem a mesma batida por

segundo este não era o único fator que permitia a existência de harmonia. Para existir harmonia era então necessário que os ficheiros de um segundo fossem adicionados por ordem e na sobreposição fossem ficheiros do mesmo segundo (ver figura 8.15).

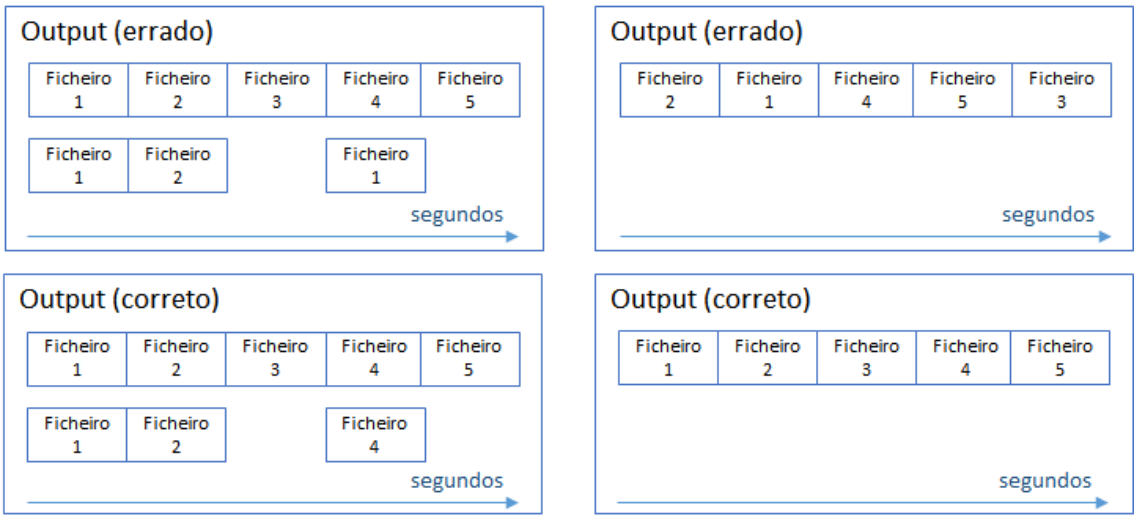


Figura 8.15: Comparação entre *outputs* corretos e errados.

Outra melhoria foi feita para optimizar a harmonia, na geração musical, foi utilizar *fade-in* e *fade-out* quando era iniciado um novo *loop* e quando acabava respectivamente. Esta melhoria fez com que o *output* fosse mais agradável uma vez que os ficheiros não começavam nem acabavam de tocar repentinamente.

8.3 Comparação de aplicações

Esta aplicação é muito semelhante a um sequenciador, no entanto esta tem muitas mais componentes que afetam a geração musical do que num simples sequenciador. Além disto existem algumas aplicações móveis com algumas parecenças com esta aplicação, como é o caso da *Virtual ANS* e da *Audible Ink*.

Em relação à *Virtual ANS*, esta permite que o utilizador insira objetos e possa desenhar mas não permite utilizar fotos e muito menos vídeos. No entanto a *Audible Ink* permite usar uma imagem como *input*, o que acaba por ser limitador uma vez que não permite o uso de vídeo. Além disto estas aplicações não têm um jogo associado, como a aplicação que foi desenvolvida neste projeto. Este jogo tem a capacidade de poder ser jogado pelo utilizador tenha ele muito ou pouco interesse em jogar, uma vez que a aplicação pode fazer parte da rotina do utilizador. Em relação a isto a *Virtual ANS* não tem esta capacidade uma vez que o seu funcionamento não é rotineiro. Outro aspecto que este projeto tem que

as outras aplicações não tem é um espaço onde vários utilizadores podem partilhar as suas criações.

Capítulo 9

Conclusões

Neste projeto foi inicialmente idealizado que a aplicação seria um jogo que estivesse relacionado com temas como *ambient gaming*, dispositivos móveis e realidade aumentada. Por isso, estas temáticas foram estudadas no sentido de perceber o que existe atualmente e de construir um projeto mais completo e diferente do que existe hoje em dia. Após este estudo, foram feitas algumas provas de conceito que incorporavam os dispositivos móveis e a realidade aumentada. Durante estas provas percebeu-se que o *hardware* dos *smartphones* ainda não está preparado para realizar este projeto, no que toca à componente de detecção de formas geométricas em tempo real. Esta componente exige um processamento e memória que não estão disponíveis nos *smartphones*. Ainda existiram alguns esforços no sentido de se tentar adaptar esta componente às características dos dispositivos mas conclui-se que era necessário chegar a outra solução. A solução encontrada passa por retirar o tempo real da detecção de formas geométricas passando-se a fazer um pós-processamento. Desta forma a limitação encontrada foi minimizada de maneira a prosseguir com o projeto. Mais tarde foi também estudado o tema criatividade computacional de maneira a acrescentar mais valor à aplicação.

Posteriormente, passou-se para a implementação onde existem três grandes componentes: a análise de vídeo, a geração de música e o jogo. Em relação à análise vídeo, inicialmente, existiram algumas dificuldades com a detecção de formas, uma vez que o *OpenCV* não tem funções capazes de detectar triângulos e quadriláteros. No entanto estas dificuldades acabaram por ser ultrapassadas, e passou-se à recolha de variáveis e à sua análise, uma vez que estas tinham de ser selecionadas para gerar música.

A geração de música foi a componente onde se obteve mais dificuldades, pois esta é uma área pouco desenvolvida no que toca a dispositivos móveis e por isso existem poucas so-

luções capazes de ajudar projetos inovadores como este. Inicialmente existia uma ideia para gerar música a partir do formato *midi* através de variáveis musicais. No entanto, a ideia teve de ser modificada por causa de incompatibilidades do formato de áudio com o vídeo. Para encontrar uma solução foi realizada uma análise e uma procura exaustiva. No entanto para incluir a solução encontrada foi necessário reformular a ideia. A ideia foi criar um sequenciador, uma vez que era necessário mudar o formato de áudio e com isto perder o dinamismo que era possível com variáveis musicais. No entanto o sequenciador como foi desenhado também é capaz de incluir bastante dinamismo através da variedade de *loops* e efeitos.

A geração de uma composição surge da conjunção destas duas áreas: a análise de vídeo e a geração de música. As variáveis visuais são adaptadas à geração de música e é então criada a música que é incorporada no vídeo, obtendo-se uma composição vídeo-musical. As composições são então construídas com base na premissa da criatividade computacional, mais especificamente da criatividade musical uma vez que a aplicação é capaz de transformar algo visual em algo musical.

O jogo está maioritariamente contido no servidor pois não é necessário que este esteja integrado na aplicação uma vez que é *ambient gaming*. Este jogo, sendo *ambient gaming*, não necessita de atenção por parte do utilizador para ser jogado pois o utilizador pode obter *badges*, concluir missões e ganhar pontos apenas com a sua rotina. No entanto, caso o utilizador decida dar mais atenção ao jogo consegue obter uma melhor experiência e avançar mais depressa no mesmo.

A aplicação que foi desenvolvida tem potencial pela sua originalidade e inovação, e por esse mesmo motivo o trabalho ainda não está acabado uma vez que devem existir melhoramentos e novas funções a incorporar. No entanto, todo o trabalho que foi já desenvolvido apresenta uma base sólida para o desenvolvimento deste trabalho futuro.

9.1 Trabalho futuro

Este é um projeto complexo que está muito de longe de ter fim, uma vez que existem sempre maneiras de o melhorar e funcionalidades a acrescentar. Algumas destas melhorias podem ser ao nível da aplicação sejam elas referentes à geração de música como à detecção de formas, ou até mesmo novas funcionalidades que ainda não foram pensadas. O servidor também pode ser melhorado uma vez que pode promover mais interação entre utilizadores, mais funcionalidades no jogo, entre outras. Outro aspecto que deve ser também analisado

é a comercialização, ou seja, como é que aplicação se pode tornar rentável.

9.1.1 Aplicação

No que toca a aplicação é possível melhorar algumas funcionalidades e acrescentar muitas mais. Existem duas componentes que podem ser melhoradas, mas talvez no futuro distante uma vez que existem poucos desenvolvimentos atualmente nas áreas são a análise de vídeo e a geração de música.

Em relação à analise de vídeo, poderia ser melhorado o tempo que demora a processar cada *frame*. No entanto, não existem alternativas neste momento. Existem também algumas funcionalidades que podem ser acrescentadas como por exemplo: novas formas geométricas e mais variáveis visuais capazes de criar mais dinamismo no que toca a geração de música.

A geração de música é um dos pontos que deve ser analisado uma vez que podem ser adicionadas muitas funcionalidades pois esta área está bastante relacionada com a criatividade. Neste sentido deve-se procurar e desenvolver uma maneira de trazer mais dinamismo à geração da música através das notas musicais, timbre, e outras variáveis musicais. Outro aspecto que pode ser melhorado é o processo de criação e tentar evitar tantas conversões de formatos, e por isso deve ser analisada uma forma de se fazer as alterações necessárias aos ficheiros de som, e manter sempre o seu formato. A diversidade de músicas também é um fator a ter em conta. Este pode ser melhorado ao adicionar novos *loops* de música, incluir mais variáveis visuais e utiliza mais efeitos nos ficheiros de som.

Além disto, existem outras melhorias que podem ser feitas à aplicação de uma maneira geral, como é o caso do *design* uma vez que este deve ser sempre revisto ao longo dos tempos numa tentativa de que a aplicação esteja atual. Podem ser feitos também testes de usabilidade e dar mais atenção nesta área.

Podem também ser adicionadas mais funções à aplicação, como por exemplo a componente do jogo que está presente no servidor ser incorporada na aplicação.

9.1.2 Servidor

O servidor também pode ser melhorado principalmente no que toca à componente de jogo, uma vez que este é um fator que cativa os utilizadores a registarem-se e a continuar a utilizar a aplicação. Neste sentido podem ser adicionadas mais missões à base de dados, e até mesmo criar missões dinâmicas, ou seja, em missões como "Ter 3 triângulos numa

composição", o número de formas passava a ser decidido dinamicamente. Além disto este número poderia ser definido através do histórico do utilizador, por exemplo caso o utilizador recolhesse poucos círculos a missão poderia ter um número maior para recolher. Ainda nas missões poderiam ser adicionadas novos tipos de missões, e passariam a existir não só missões diárias mas também semanais.

Os *badges* também devem ser analisados de maneira a criar mais variedade de objectivo, para cativar o interesse dos utilizadores. Além disto poderiam existir *badges* que só poderiam ser obtidos com um tempo limitado, por exemplo poderia existir um *badge* referente ao campeonato mundial de futebol de 2014 e este só poderia ser obtido enquanto decorresse o evento e assim que este acabasse o *badge* já não podia ser conquistado. Isto tornava os *badges* mais raros, e incentivava o interesse de obter aquele *badge* em específico.

Outro aspecto que pode ser melhorado é a interação entre utilizadores. Neste momento existe pouca interação como é o caso das votações nas composições e os comentários do *Youtube*. Este pode ser resolvido ao incorporar comentários nas composições e integrar os comentários do *Youtube*, e até mesmo realizar trocas de *badges* entre utilizadores. Estas trocas poderiam ser de *badge* por *badge* ou pontos.

9.1.3 Comercialização

A comercialização da aplicação é um aspecto que deve ser considerado num futuro próximo, sendo esta uma aplicação que não tem concorrência uma vez que não existem aplicações semelhantes no mercado móvel.

A aplicação poderia ser colocada no *Google Play* para ser rentabilizada através de várias maneiras, como por exemplo apresentar publicidade. A publicidade poderia estar tanto na aplicação móvel como no servidor. Neste sentido a aplicação seria gratuita, mas teria também uma versão paga que não apresentasse publicidade para aqueles utilizadores que não gostam de ser incomodados.

Outro conceito que é muito usado atualmente é o *downloadable content*. Este consiste em disponibilizar algum conteúdo gratuitamente mas ter outro que é pago caso os utilizadores o pretendam usar. Neste caso isto pode ser aplicado aos *loops* de música, ou seja, existiria apenas um conjunto de *loops* gratuitos e os outros seriam pagos para se poderem usar. Para incentivar as pessoas a comprar os *loops* as missões poderiam ter o objectivo da composição fazer uso de um conjunto que fosse pago.

Apêndice A

Estudo da Realidade Aumentada

A interação das pessoas com os computadores começou por ser muito básica, avançando o grau de complexidade com o passar dos tempos. Inicialmente as interfaces que permitiam aos utilizadores comunicar com os computadores eram à base de comandos (*Command Line Interfaces*). Seguiram-se as *Graphical User Interfaces*, que já permitiam ao utilizador comunicar com o computador de forma gráfica e não só através de texto. Hoje em dia está-se a tentar que as interfaces sejam mais naturais e retratem a interação dos utilizadores com os objetos do quotidiano. É neste tipo de interfaces que se enquadra a realidade aumentada [68]. A realidade aumentada tem o objectivo de fazer a ponte entre o mundo virtual com o mundo real [69].

A.1 Definição

A realidade aumentada é uma variação de realidade virtual. No que toca à realidade virtual o utilizador não consegue ver o mundo real mas sim um mundo virtual. Já a realidade aumentada pretende que o utilizador veja o mundo real mas com a imposição de elementos gerados computacionalmente [69][70][71]. Esta tecnologia é utilizada em tempo real e tira partido dos cinco sentidos do utilizador, mas o mais comum é utilizar a visão o que leva ao uso de um ambiente 3D para representar os vários elementos [71]. Por vezes, a realidade aumentada é confundida com editar um vídeo ou uma imagem, ou até com o reconhecimento de objetos em imagens obtidas por uma câmara de filmar. Outras vezes, a realidade aumentada é também confundida com realidade virtual uma vez que a realidade virtual tem em comum o facto da informação ser gerada por um computador. A diferença entre as duas é que no caso da realidade virtual o utilizador é transposto para outro mundo, um mundo completamente digital, enquanto que a realidade aumentada apenas acrescenta mais informação àquela que já existe no mundo real. Para as aplicações serem conside-

radas realidade aumentada, devem ser usadas em tempo real e adicionar algum tipo de informação ao mundo real [69].

A realidade aumentada tem sido uma tecnologia muito estudada nos últimos anos [69].

- Em **1992**, Tom Caudell e David Mizell cunharam o termo realidade aumentada, quando encontraram uma maneira de ajudar a Boeing nos processos de engenharia e de fabrico. Assim criaram um software que permitisse sobrepor as posições de alguns cabos sobre os diversos componentes.
- Em **1996**, Jun Rekimoto teve a ideia de utilizar um marcador 2D. Um marcador é um objecto que está contido no mundo real e que faz a ponte entre o mundo virtual e mundo real. Assim, o dispositivo reconhece o marcador e coloca a informação sobre este.
- Em **1997**, Ronald Azuma identificou três características importantes para a realidade aumenta: é em tempo real, está em 3D e combina o mundo real com o mundo virtual.
- Em **1999**, Hirokazu Kato lançou o *ARToolKit* [1]. Este é *open source* e permite combinar o mundo real com objetos digitais através de vídeo.
- Em **2004**, Mathias Möhring criou o primeiro sistema a usar marcadores 3D através de dispositivos móveis.
- Em **2006**, a *Nokia* iniciou o projeto MARA (*Mobile Augmented Reality Applications*) que consiste em criar um ambiente em que o utilizador pode ver outros utilizadores, texto e gráficos em tempo real através de um dispositivo móvel.

A.2 Utilizações

Hoje em dia, a realidade aumentada está a ser muito utilizada em áreas como a publicidade. Por exemplo em que marcas como a *Lego* [2] (figura A.1) ou a *Nissan* [3], *Toyota* [4], *BMW* [5] e *Mini* [6] a utilizam-na para promover os seus produtos de uma forma mais interativa.

[1] mais informações em: http://artoolkit.sourceforge.net/
[2] mais informações em: http://www.lego.com/en-us/
[3] mais informações em: http://www.nissan.pt/
[4] mais informações em: http://www.toyota.pt/
[5] mais informações em: http://www.bmw.pt/pt/pt/
[6] mais informações em: http://www.mini.pt/mini/cooper/

Figura A.1: O *Lego Kiosk* permite ver o conteúdo que está dentro da caixa.

Também é usada para orientação em que aplicações como o *Yelp* [7] (figura A.2) e o *NRU* (figura A.3) que indicam locais (restaurantes, bares, lojas, entre outros).

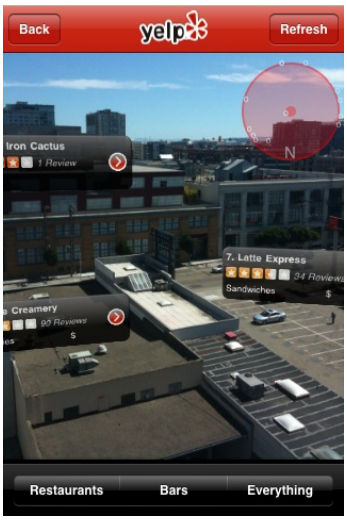


Figura A.2: A aplicação *Yelp* encontra vários locais próximos e apresenta a sua informação no ecrã.

Figura A.3: A *NRU* é uma aplicação muito semelhante à *Yelp*, com a diferença de mostrar a informação em forma de radar.

A arte é outra área onde esta tecnologia está bastante presente. A aplicação *Konstruct* [8] é um exemplo disso, pois permite ver elementos escondidos, a olho nu, dentro do museu (figura A.4).

[7] mais informações em: http://www.yelp.com/yelpmobile
[8] mais informações em: http://apps.augmatic.co.uk/konstruct

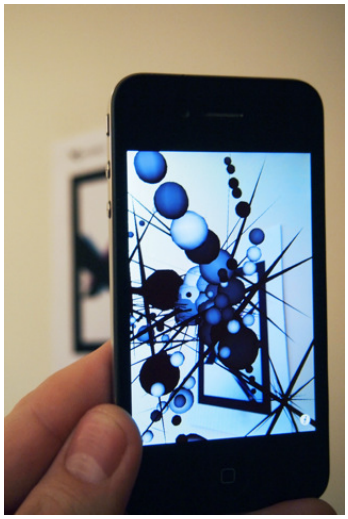


Figura A.4: *Konstruct.*

Os jogos representam uma grande fatia no que toca à realidade aumentada, visto que é uma área que promove um novo tipo de interação com o jogador. Exemplo disto é o *Parrot AR.Drone* [9] em que é utilizado um *iPhone/iPad* para controlar um brinquedo voador (figura A.5).

Figura A.5: *Parrot AR.Drone.*

Atualmente as redes sociais são muito importantes e por isso não podiam ficar de parte. Existe por exemplo uma aplicação chamada *Reconizr* que permite reconhecer uma pessoa e obter algumas informações sobre ela (figura A.6).

A educação é outro campo onde a realidade aumentada pode trazer alguns frutos e criar aplicações como quadros interativos ou tentar ensinar a fazer uma tarefa sem que a pessoa necessite de estar num certo local ou utilizar um material específico. Também pode ser

[9] mais informações em: http://ardrone2.parrot.com/

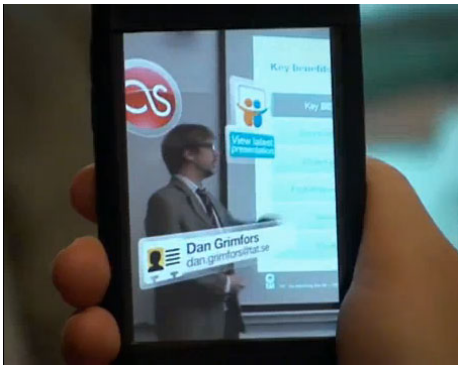


Figura A.6: *Recognizr.*

utilizada na tradução, com aplicações como a *Word Lens* [10] (figura A.7) e *Intelligent Eye* que permitem que ao apontar a câmara para um determinado texto e este seja traduzido para outra língua.

Figura A.7: *Word Lens.*

Esta tecnologia é usada também noutras tarefas. Por exemplo, na compra de uma televisão em que não se sabem quais as medidas, é possível, utilizando uma escala, perceber o espaço que esta vai ocupar apontando apenas a câmara de um dispositivo móvel. Outro exemplo comum de realidade aumentada está presente em desportos quando estes são transmitidos na televisão, como é o caso do início de uma prova de natação onde aparecem os nomes e os países de cada desportista (figura A.8) [69].

Figura A.8: Realidade aumentada numa prova de natação.

[10] mais informações em: http://questvisual.com/pt/

A.3 Desafios

Existem dois potenciais grandes desafios no que toca à realidade aumentada, os técnicos e os sociais. Os técnicos estão relacionados com aspectos de *hardware* e *software*, como por exemplo técnicas de reconhecimento, sensores e dispositivos. Os sociais passam por definir a motivação, que problemas é que se vão resolver e porque é que a sociedade deve usar esta tecnologia [69].

Em relação ao *hardware* é necessário um computador (ou um dispositivo móvel capaz de fazer o processamento necessário) e um monitor ou ecrã para apresentar os resultados. Pode também ser possível utilizar alguns sensores como GPS, acelerómetro, bússola, entre outros. O acesso à *internet* pode ser importante para produzir dados a apresentar, ou apenas para comunicar com a aplicação. A realidade aumentada pode ser utilizada em computadores pessoais através do uso de uma câmara, em quiosques, *smartphones* ou *tablets* visto que estes têm câmaras, GPS e outros sensores que podem ser necessários para a aplicação, e também pode ser utilizada em óculos especiais para realidade aumentada. Um exemplo de realidade aumentada é o *HUD* (*heads-up display*), utilizado pelos pilotos de aviões de guerra, que tem a capacidade de acrescentar ao mundo real informação importante para a condução destes veículos (figura A.9).

Figura A.9: *HUD* de um piloto de aviões.

Sabendo isto, pode-se dividir a realidade aumentada em duas categorias: fixa e móvel. Na fixa o sistema não pode ser movido e é utilizado no local onde se encontra, enquanto que a móvel acompanha o utilizador, ou seja, o sistema pode ser transportado e utilizado em qualquer lugar [69].

Geralmente existem três tipos de monitores possíveis para a realidade aumentada: dispositivos móveis, *Spatial Augmented Reality* e roupa. Dentro dos dispositivos móveis existem os *smartphones* e os *tablets*. No caso dos *Spatial Augmented Reality* as dimensões são maiores

(normalmente são televisões, projetores, hologramas, etc.) e por isso não são móveis. Em relação à roupa (e acessórios) deve ser algo que o utilizador use e que passe despercebido (como por exemplo óculos) [69].

A maneira como se interage com as aplicações de realidade aumentada é um aspecto importante e que deve ser analisado. Existem alguns métodos que foram pensados para melhorar a interação do utilizador: *tangible, collaborative, hybrid* e *multimodal* [69].

- O *tangible* tenta misturar o mundo real com o mundo digital de maneira a que o utilizador consiga ver os objetos digitais e quando interage com eles exista um *feedback* táctil.
- O *collaborative* permite uma interação de vários utilizadores ao mesmo tempo.
- O *hybrid* combina muitas interações, e acaba por ser mais flexível no que toca à adaptação deste método ao dispositivo.
- O *multimodal* combina vários métodos de interação, sendo possível interagir com objetos reais com o uso do toque, fala, gestos, olhar, etc.

Em relação ao *software*, o reconhecimento de objetos é um grande desafio, que tem sido muito trabalhado durante estes anos, mas que ainda pode ser melhorado. Os problemas mais comuns são: a oclusão, a câmara desfocada, o movimento e a luminosidade irregular. A oclusão acontece quando existe uma obstrução ou quando algo está a bloquear a vista. A câmara desfocada reduz a percepção dos elementos, o que afeta a precisão do reconhecimento. O movimento também afeta a precisão, especialmente quando é muito rápido. A luminosidade irregular acontece quando sombras incidem sobre os objetos, o que faz com que estes não sejam reconhecidos [69].

Existem vários métodos de realizar o reconhecimento de objetos. Alguns exemplos disso são: por padrão, por contorno, de localização e de superfície. O método por padrão consiste no reconhecimento de padrões em marcadores ou formas geométricas básicas. O método de contorno permite reconhecer partes do corpo, sendo assim possível interagir com objetos digitais (exemplo disto é o *Magic Mirror* [11] (figura A.10) em que se pode experimentar óculos digitais, peças de roupa, etc).

O método de localização é baseado em GPS e permite colocar objetos digitais no ecrã dependendo das coordenadas do utilizador. Para obter melhores resultados podem ser

[11] mais informações em: http://www.magicmirror.me/

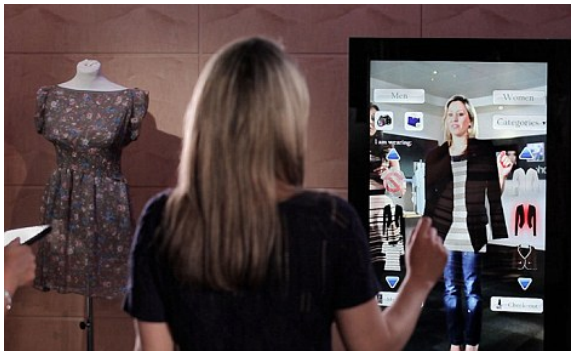


Figura A.10: *Magic Mirror*

usados acelerómetros e bússolas. Este método é usado por aplicações como o *Layar* [12] e o *Wikitude* [13]. O método de superfície consiste em ecrãs, paredes, chão, etc. que sejam interativos após o toque do utilizador ou de outro objecto. A *Microsoft* desenvolveu um projeto, em 2007, chamado *LightSpace* que fazia uso deste método, em que era possível interagir com objecto digitais. Outro exemplo disto é o *AR floor*, que faz uso do chão para simular neve, areia, relva, entre outros, e através das vibrações exercidas pelo utilizador reage com sons e com efeitos visuais [69].

No que toca aos desafios sociais, a realidade aumentada pode ajudar o utilizador em tomadas de decisão através de informação que lhe é disponibilizada. Também pode ter como função criar um ambiente artificial, ou seja, permitir a criação de elementos digitais que não existem no mundo real. Com isto, existem muitas áreas onde se pode aplicar a realidade aumentada, como por exemplo: entretenimento, educação, medicina, negócios, serviços públicos, justiça e militar [72][73][71]. Apesar de todas estas áreas terem potencial, apenas será explorado neste estudo, com detalhe, o entretenimento. A realidade aumentada pode estar inserida nas diversas formas de entretenimento sejam elas jogos, arte, cultura, filmes, música, etc. Os jogos em particular estão inseridos num mercado que move muito dinheiro, e por isso existem apostas na realidade aumentada por parte de grandes empresas, como é o caso da *Sony* que ao desenvolver a *PS Vita* [14] incorporou uma componente de realidade aumentada. Outra empresa que faz uso da realidade aumentada é a *Microsoft* no seu produto *Kinect*. Este tem o lema "Tu és o comando" pois este sistema tem a capacidade de reconhecer o corpo humano e reagir consoante os seus movimentos [69].

[12] mais informações em: https://www.layar.com/
[13] mais informações em: http://www.wikitude.com/
[14] mais informações em: http://pt.playstation.com/psvita/

A.4 Comparação entre plataformas de realidade aumentada

Para se criar uma aplicação de realidade aumentada é necessário considerar algumas plataformas para ajudar no seu desenvolvimento. Por isso foram analisadas algumas que pareceram ser as mais adequadas ao projeto. Foram examinadas sete ferramentas de forma mais intensiva de forma a perceber qual acrescenta mais valor ao projeto.

A *Vuforia* [15] pertence à empresa *Qualcomm* [16]. Esta ferramenta é capaz de reproduzir um grafismo personalizado na aplicação. Também consegue detectar texto, imagens, objetos 3D, entre outros. Pode fazer uso da *cloud*, caso seja necessário guardar imagens para utilizar na aplicação. Um dos aspectos mais importantes é a forte documentação, uma vez que contém bastantes guias, tutoriais e exemplos.

A *ARPA* [17] é muito semelhante à *Vuforia* pois também é capaz de fazer o reconhecimento de caras e de texturas. No entanto não utiliza a *cloud* para guardar imagens, nem tem uma documentação tão completa como a anterior.

A *D'Fusion* [18] é um produto da empresa *Total Immersion*. No entanto esta ferramenta tem a vantagem de ter o seu próprio IDE, o *D'Fusion Studio*.

A *Metaio* [19] também é muito semelhante às duas ferramentas iniciais. Esta faz mais uso da *cloud*, e de outra aplicação chamada Junaio que serve para obter mais informação sobre um determinado local, objeto, etc. Esta ferramenta tem uma versão grátis mas que tem a limitação de ter uma marca de água sempre presente no ecrã.

A *Wikitude* [20] é uma ferramenta que permite utilizar informação da *Wikipedia*, *Twitter*, *Flickr*, entre outros. Mas esta ferramenta apenas é grátis para fins académicos e ainda assim contém uma marca de água no ecrã.

A ferramenta *OpenCV* [21] é bastante poderosa no que toca a reconhecimento de objetos, sejam eles marcadores ou não. É *open-source* o que permite fazer melhorias ao código

[15] mais informações em: https://www.vuforia.com/platform
[16] mais informações em: http://www.qualcomm.com/
[17] mais informações em: http://www.arpa-solutions.net/
[18] mais informações em: http://www.t-immersion.com/products/dfusion-suite/dfusion-mobile
[19] mais informações em: http://www.metaio.com/products/sdk/
[20] mais informações em: http://www.wikitude.com/
[21] mais informações em: http://opencv.org/

Aplicação	**Modo Offline**	**Plataforma**	**Licença**	**Documentação**
Vuforia	Sim	*Android, iOS*	Grátis	Muito Boa
ARPA	Sim	*Android, iOS*	Grátis	Boa
D'Fusion	Sim	*Android, iOS*	Grátis	Boa
Metaio	Sim	*Android, iOS*	Paga	Boa
Wikitude	Sim	*Android, iOS*	Paga	Boa
OpenCV	Sim	*Android, iOS*	*Open-source*	Boa
Layar	Não	*Android, iOS*	Paga	Razoável

Tabela A.1: Comparação entre aplicações para desenvolver realidade aumentada

e também permite que a comunidade contribua para o seu desenvolvimento. No entanto não tem algumas funções como é o caso de *cloud* e também é um pouco mais difícil de se trabalhar o grafismo.

A *Layar* [22] tem como ponto forte a geo-localização, mas fica a perder no que toca a estar sempre ligada à *internet* e na documentação, que poderia ser melhor. Além disto esta ferramenta é paga.

A tabela A.1 compara de uma forma mais simplificada estas sete ferramentas através da análise de quanto aspectos: se a aplicação contem um modo offline, quais as plataformas que suportam, se a sua licença é grátis ou paga e qual a qualidade da sua documentação.

[22] mais informações em: https://www.layar.com/

Apêndice B

Mockups

Neste secção são apresentados os *mockups* (desenvolvidos pela equipa de design do CDV [1]) da aplicação. Estes *mockups* tem também a capacidade de mostrar como se processa a navegação da aplicação.

A figura B.1 corresponde a uma legenda para os botões presentes na aplicação.

A figura B.2 representa como é feita a interação com a aplicação e o *feedback* que esta retorna.

A figura B.3 mostra como é possível atualizar os dados do jogador.

Na figura B.4 está representado o processo de captura de um vídeo e de composição de um vídeo já existente. Pode-se verificar que depois de se criar uma composição vídeo-musical, o utilizador tem a possibilidade de reproduzir novamente a composição, de criar uma nova, de alterar o instrumento escolhido anteriormente, e de guardar composição e depois partilhá-la.

Na figura B.5 está representada a galeria de composições vídeo-musicais. Esta galeria permite reproduzir as composições, apagá-las e partilhá-las.

No entanto estes *mockups* estão desatualizados uma vez que existiram algumas reformulações à ideia inicial.

[1] mais informações em: http://cdv.dei.uc.pt/people/

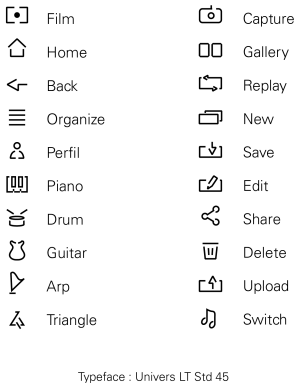


Figura B.1: Legenda de botões

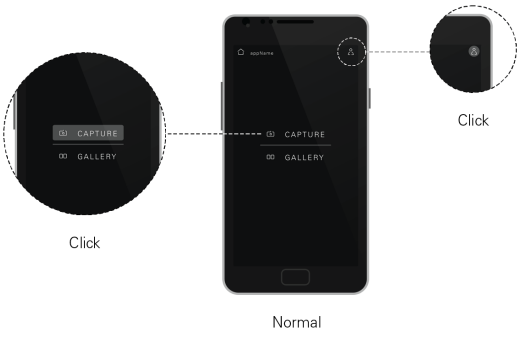


Figura B.2: Interação com a aplicação

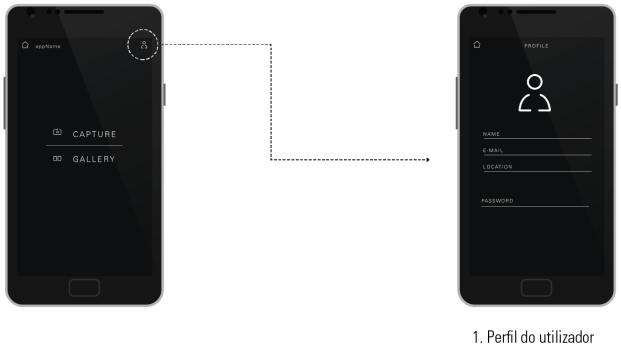


Figura B.3: Atualização dos dados de utilizador

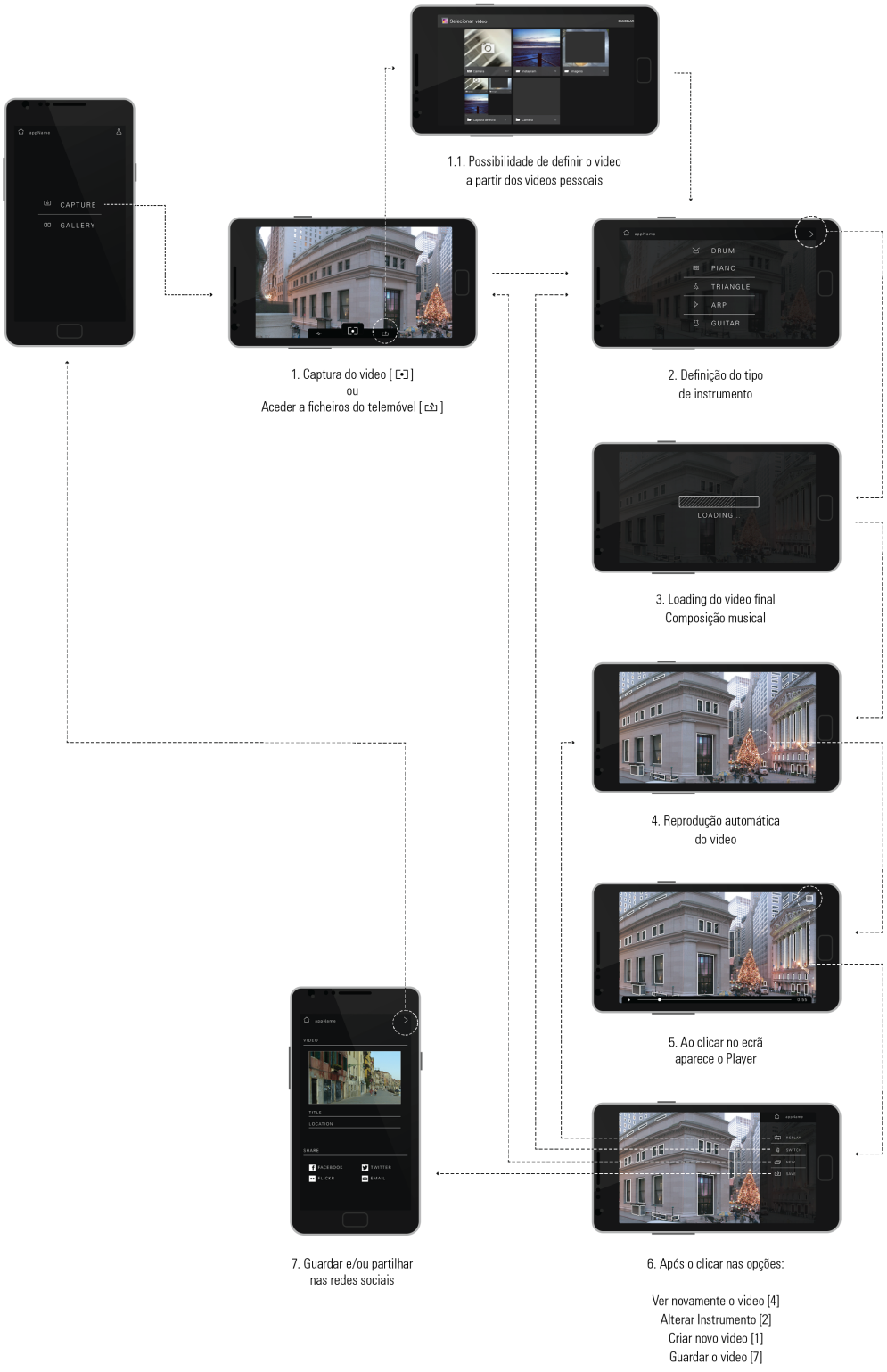


Figura B.4: Captura

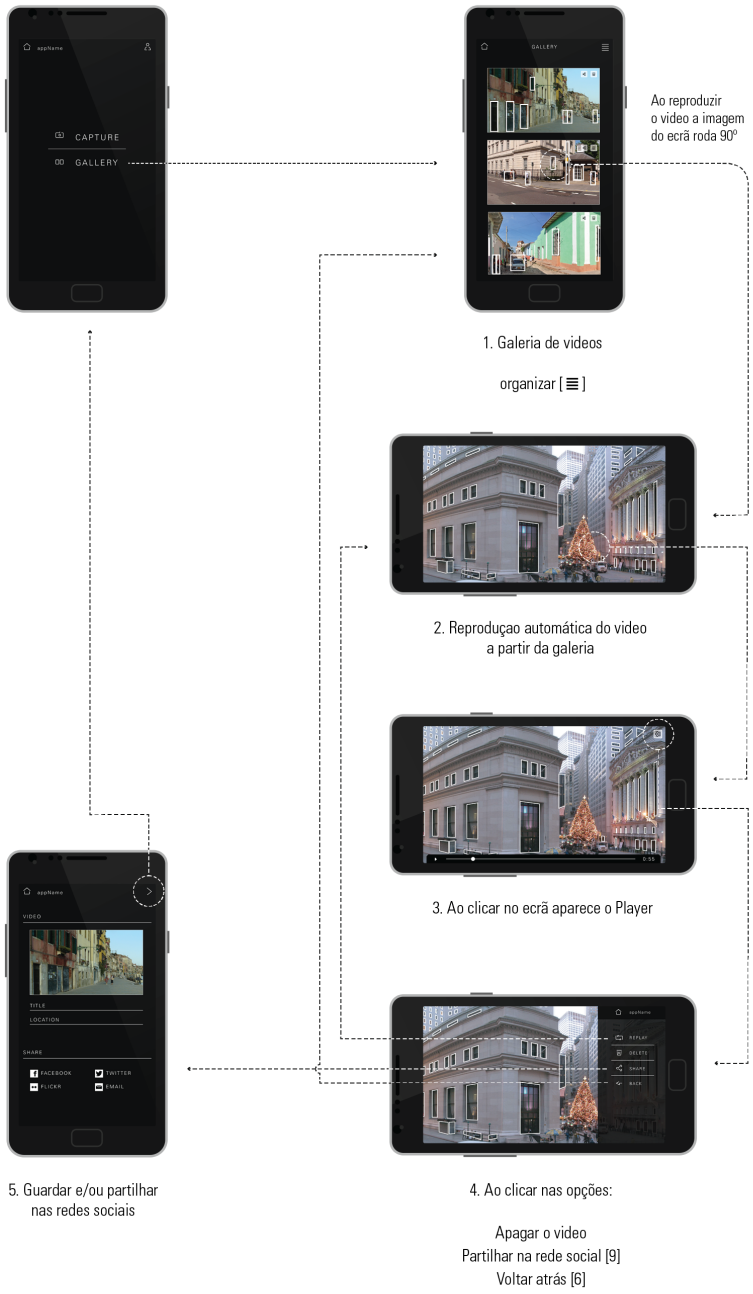


Figura B.5: Galeria

Bibliografia

[1] S. Kumar and C. Zahn. Mobile communications: Evolution and impact on business operations, 2003.

[2] Instagram. disponível online em `http://instagram.com/press/`, consultado pela última vez em 25 Janeiro 2014.

[3] Vine hits 40 million registered users, but how many are active? disponível online em `http://www.theverge.com/2013/8/20/4641980/vine-hits-40-million-registered-users-but-how-many-are-active`, consultado pela última vez em 25 Janeiro 2014.

[4] Mark Eyles and Roger Eglin. Ambient games, revealing a route to a world where work is play? *Int. J. Computer Games Technology*, 2008, 2008.

[5] Brian Eno. Ambient 1: Music for airports, 1978.

[6] Brian Eno. The long now-transcript, November 2003. disponível online em `http://www.enoshop.co.uk/words.asp`.

[7] History of social games. disponível online em `http://radoff.com/blog/2010/05/24/history-social-games/`, consultado pela última vez em 12 Outubro 2013.

[8] History of sports and games. disponível online em `http://www.historyworld.net/wrldhis/PlainTextHistories.asp?historyid=ac02`, consultado pela última vez em 25 Janeiro 2014.

[9] History of sports. disponível online em `http://inventors.about.com/od/sstartinventions/tp/History-Of-Sports.htm`, consultado pela última vez em 25 Janeiro 2014.

[10] History of games timeline. disponível online em `http://historicgames.com/gamestimeline.html`, consultado pela última vez em 25 Janeiro 2014.

[11] Ernest Adams. The designers notebook: Sorting out the genre muddle, 2009. disponível online em `http://www.gamasutra.com/view/feature/132463/the_designers_notebook_sorting_.php?page=4`, consultado pela última vez em 25 Janeiro 2014.

[12] Janienke Sturm and Ben A. M. Schouten. Ambient gaming and play: Opportunities and challenges. In Reiner Wichert, Kristof Van Laerhoven, and Jean Gelissen, editors, *AmI Workshops*, volume 277 of *Communications in Computer and Information Science*, pages 213–217. Springer, 2011.

[13] R. Harwig E. Aarts and M. Schuurmans. *Ambient intelligence*, pages 235–250. McGraw-Hill, Inc., New York, NY, USA, 2002.

[14] Mark Weiser. Ubiqutous computing, 1996. disponível online em `http://www.ubiq.com/hypertext/weiser`, consultado pela última vez em 25 Janeiro 2014.

[15] R.-D. Vatavu and I.-A. Zaiti. Exploration for gaming applications. proceedings of the first workshop on ambient gaming (amgam'11). 2011.

[16] Pepijn Rijnbout, Linda De Valk, Mark de Graaf, Tilde Bekker, Ben A. M. Schouten, and Berry Eggen. i-pe: A decentralized approach for designing adaptive and persuasive intelligent play environments. In *AmI Workshops*, pages 238–244, 2011.

[17] Rui Neves Madeira, Octavian Postolache, and Nuno Correia. Gaming for therapy in a healthcare smart ambient. In Reiner Wichert, Kristof Van Laerhoven, and Jean Gelissen, editors, *AmI Workshops*, volume 277 of *Communications in Computer and Information Science*, pages 224–228. Springer, 2011.

[18] M. Montola, J. Stenros, and A. Waern. *Pervasive Games: Theory and Design*. Morgan Kaufmann Game Design Books. Taylor & Francis, 2005.

[19] D. A. Waern. Iperg, 2006. disponível online em `http://www.pervasive-gaming.org/index.php`, consultado pela última vez em 25 Janeiro 2014.

[20] Sony patents temperature feedback games controller. disponível online em `http://www.gizmag.com/sony-controller-temperature-feedback/24599/`, consultado pela última vez em 25 Janeiro 2014.

[21] Game skunk. disponível online em `http://sensoryacumen.com/index.php?option=com_content&view=article&id=89&Itemid=62`, consultado pela última vez em 25 Janeiro 2014.

[22] L. Srivastava. Mobile phones and the evolution of social behaviour. *Behaviour & IT*, 24(2):111–129, 2005.

[23] Emanuel A. Schegloff. *Perpetual Contact: mobile communication, private talk, public performance*, chapter Opening sequencing, pages 326–385. Cambridge University Press, New York, NY, 2002.

[24] Christian Licoppe and Jean Philippe Heurtin. Managing one's availability to telephone communication through mobile phones: A french case study of the development dynamics of mobile phone use. *Personal and Ubiquitous Computing*, 5(2):99–108, 2001.

[25] Christian Licoppe and Jean-Philippe Heurtin. Perpetual contact. chapter France: Preserving the Image, pages 94–109. Cambridge University Press, New York, NY, USA, 2002.

[26] Richard H. R. Harper. People versus information: The evolution of mobile technology. In Luca Chittaro, editor, *Mobile HCI*, volume 2795 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2003.

[27] Leopoldina Fortunati. The mobile phone: An identity on the move. *Personal and Ubiquitous Computing*, 5(2):85–98, 2001.

[28] April 3, 1973: Motorola calls at&t... by cell. disponível online em `http://www.wired.com/science/discoveries/news/2008/04/dayintech_0403`, consultado pela última vez em 25 Janeiro 2014.

[29] Umts / 3g history and future milestones. disponível online em `http://www.umtsworld.com/umts/history.htm`, consultado pela última vez em 25 Janeiro 2014.

[30] Cell phone from brick to slick. disponível online em `http://www.wilsonelectronics.com/img/Infographs/MobliePhoneEvolution/MobliePhoneEvolution-800.jpg`, consultado pela última vez em 25 Janeiro 2014.

[31] Fabio Silva. Livros electrónicos - detecção e correcção de erros tipográficos: Hi-reader, September 2013.

[32] Who invented the first tablet pc? disponível online em `http://www.ehow.com/facts_5813990_invented-first-tablet-pc_.html?ref=Track2&utm_source=ask`, consultado pela última vez em 25 Janeiro 2014.

[33] The smartphone operating system complete comparison. disponível online em `http://myphonedeals.co.uk/blog/33-the-smartphone-os-complete-comparison-chart,consultadopelaúltimavezem25Janeiro2014.`

[34] Lucas Bulcão Jailson de Brito Vaninha Vieira Adolfo Duran Marivaldo Mascarenhas, Mario Martins. Um estudo de caso com análise comparativa entre plataformas para aplicações móveis aberta e proprietária: Android e ios. 2013.

[35] Mark H. Goadrich and Michael P. Rogers. Smart smartphone development: Ios versus android. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 607–612, New York, NY, USA, 2011. ACM.

[36] K. Tracy. Mobile application development experiences on apple's ios and android os. 2012.

[37] Ferraz C. A. G. Ribeiro, F. I. N. and F. S. Ferraz. Uma abordagem comparativa do desenvolvimento de aplicações para dispositivos móveis. 2011.

[38] IDC. Android and ios combine for 91.1% of the worldwide smartphone os market in 4q12 and 87.6% for the year, according to idc, 2013.

[39] 11% quarterly growth in downloads for leading app stores. disponível online em `http://www.canalys.com/newsroom/11-quarterly-growth-downloads-leading-app-stores`, consultado pela última vez em 25 Janeiro 2014.

[40] Who's winning, ios or android? all the numbers, all in one place. disponível online em `http://techland.time.com/2013/04/16/ios-vs-android/`, consultado pela última vez em 25 Janeiro 2014.

[41] Shaw J. G. Newell, Allen and Herbert A. Simon. The process of creative thinking. *Contemporary Approaches to Creative Thinking*, pages 63–119, 1963.

[42] Margaret A. Boden. *The Creative Mind - Myths and Mechanisms (2. ed.)*. Routledge, 2003.

[43] Margaret Boden. Computational models of creativity. *Handbook of Creativity*, pages 351–373, 1999.

[44] James Meehan. Inside computer understanding: Five programs plus miniatures. 1981.

[45] Tony Veale and Yanfen Hao. Comprehending and generating apt metaphors: A web-driven, case-based approach to figurative language. In *AAAI*, pages 1471–1476. AAAI Press, 2007.

[46] Oliviero. HAHAcronym: A Computational Humor System.

[47] Pablo Gervás. An expert system for the composition of formal spanish poetry. *Knowl.-Based Syst.*, 14(3-4):181–188, 2001.

[48] Learn about aaron's history. disponível online em `http://www.kurzweilcyberart.com/aaron/history.html`, consultado pela última vez em 25 Julho 2014.

[49] Roger B. Dannenberg. David cope, computer models of musical creativity, mit press (2005). *Artif. Intell.*, 170(18):1218–1221, 2006.

[50] David Cope. *Experiments in musical intelligence*. A-R Editions, Inc., 1996.

[51] The dream of color music, and machines that made it possible. disponível online em `http://www.awn.com/mag/issue2.1/articles/moritz2.1.html`, consultado pela última vez em 25 Julho 2014.

[52] Colour and sound. disponível online em `http://homepage.tinet.ie/~musima/visualmusic/visualmusic.htm`, consultado pela última vez em 25 Julho 2014.

[53] Play!framework vs ruby on rails. disponível online em `http://dillonbuchanan.com/programming/playframework-vs-ruby-on-rails/`, consultado pela última vez em 25 Julho 2014.

[54] The rails command line. disponível online em `http://guides.rubyonrails.org/command_line.html`, consultado pela última vez em 25 Julho 2014.

[55] Getting started with rails. disponível online em `http://guides.rubyonrails.org/getting_started.html`, consultado pela última vez em 25 Julho 2014.

[56] Here's why ruby on rails is hot. disponível online em `http://www.businessinsider.com/heres-why-ruby-on-rails-is-hot-2011-5`, consultado pela última vez em 25 Julho 2014.

[57] Why ruby on rails is my go-to app development platform. disponível online em `http://tech.co/yes-ruby-rails-go-app-development-platform-2014-05`, consultado pela última vez em 25 Julho 2014.

[58] Five reasons why we use ruby on rails. disponível online em `http://www.infront.com/blogs/the-infront-blog/2013/1/4/five-reasons-why-we-use-ruby-on-rails`, consultado pela última vez em 25 Julho 2014.

[59] Why node.js is becoming the go-to technology in the enterprise. disponível online em `http://www.nearform.com/nodecrunch/node-js-becoming-go-technology-enterprise#.U_Y3evldV8E`, consultado pela última vez em 25 Julho 2014.

[60] Why the hell would i use node.js? a case-by-case tutorial. disponível online em `http://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js`, consultado pela última vez em 25 Julho 2014.

[61] Sqlite vs mysql vs postgresql: A comparison of relational database management systems. disponível online em `https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems`, consultado pela última vez em 25 Julho 2014.

[62] Mysql vs postgresql: Why mysql is superior to postgresql. disponível online em `https://www.udemy.com/blog/mysql-vs-postgresql/`, consultado pela última vez em 25 Julho 2014.

[63] Top reasons to use mysql. disponível online em `http://www.mysql.com/why-mysql/topreasons.html`, consultado pela última vez em 25 Julho 2014.

[64] Advantages and disadvantages of midi data. disponível online em `https://forrestercomputing.wikispaces.com/Advantages+and+disadvantages+of+MIDI+data`, consultado pela última vez em 25 Julho 2014.

[65] Android developers dashboards. disponível online em `https://developer.android.com/about/dashboards/index.html?utm_source=ausdroid.net`, consultado pela última vez em 25 Julho 2014.

[66] Audio mix and record in android. disponível online em `http://mobilengineering.blogspot.pt/2012/06/audio-mix-and-record-in-android.html`, consultado pela última vez em 25 Julho 2014.

[67] Wave pcm soundfile format. disponível online em `https://ccrma.stanford.edu/courses/422/projects/WaveFormat/`, consultado pela última vez em 25 Julho 2014.

[68] Rachel Hinman. *The Mobile Frontier: A Guide for Designing Mobile Experiences.* 2012.

[69] Greg Kipper and Joseph Rampolla. *Augmented Reality: An Emerging Technologies Guide to AR.* Syngress Publishing, 1st edition, 2012.

[70] Realidade aumentada. disponível online em `http://www.marcasepatentes.pt/files/collections/pt_PT/1/300/301/Realidade%20Aumentada.pdf`, consultado pela última vez em 25 Janeiro 2014.

[71] Flávio da Silva Pereira. Desenvolvimento de uma aplicação móvel para o turismo. 2013.

[72] Daniel Castro Silva and Vasco Vinhas. An interactive augmented reality battleship game implementation. In Proceedings of Learning with Games 2007, September 24–26 2007, Sophia Antipolis, France (Marco Taisch and Jacopo Cassina eds.), pp. 213-219, 2007.

[73] Pedro Abreu and Pedro Mendes. Mastermind: an augment reality approach: [porting a legacy game to new interaction paradigms]. In Sofia Tsekeridou, Adrian David Cheok, Konstantinos Giannakis, and John Karigiannis, editors, *DIMEA*, volume 349 of *ACM International Conference Proceeding Series*, pages 205–210. ACM, 2008.