INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Efeitos especiais mediados por Kinect em contexto de espetáculo de dança ao vivo

TIAGO NUNO MOURA GONÇALVES
Outubro de 2015

POLITÉCNICO
DO PORTO

Efeitos especiais mediados por Kinect em contexto de espetáculo de dança ao vivo

Tiago Nuno Moura Gonçalves

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Arquiteturas, Sistemas e Redes

Orientador: Prof. Doutor António Vieira de Castro

Júri:
Presidente:
[Nome do Presidente, Categoria, Escola]
Vogais:
[Nome do Vogal1, Categoria, Escola]
[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, Outubro de 2015

Aos meus pais, á minha namorada e a toda a minha família e amigos pela sua presença constante na minha vida e por todo o apoio incondicional ao longo da realização deste trabalho.

Resumo

A dança e os espetáculos foram atividades desenvolvidas e praticadas pelo homem desde praticamente a sua existência. Ao longo de todo esse período de tempo e até aos dias atuais estas atividades foram sofrendo evoluções que as fizeram manterem-se relevantes e de grande importância na sociedade humana e na sua cultura.

A evolução não se fez sentir apenas no estilo das danças e espetáculos mas também nos acessórios e efeitos que estas implementam de forma a torna-las mais atrativas para quem as vê. Apesar desta evolução, a maioria dos efeitos não permite um nível de interação com a dança ou espetáculo, fazendo com que exista uma clara separação entre a componente pura da dança e o cenário do espetáculo no que diz respeito á componente acessória de efeitos.

Com o intuito de colmatar esta clara divisão de componentes, iniciamos um estudo no sentido de criar um sistema que permitisse derrubar essa barreira e juntar as duas componentes com o intuito de criar efeitos que interajam com a própria dança tornando o espetáculo mais interativo, e que não seja apenas mais um componente acessório, isto ao mesmo tempo torna todo o espetáculo mais apelativo para o público em geral. Para conseguir criar tal sistema, recorremos às tecnologias de sensores de movimento atuais para que a ponte de ligação entre o artista e os efeitos fosse conseguida.

No mercado existem diversas ofertas de sensores de movimentos que serviriam para criar o sistema, mas apenas um poderia ser escolhido, então para tal numa primeira parte foi feito um estudo para determinar qual destes sensores seria o mais adequado para ser utilizado no sistema, tendo em conta uma diversidade de fatores.

Após a escolha do sensor foi então desenvolvido o sistema *MoveU* e tendo no final sido feitos uma série de testes que permitiram validar o protótipo e verificar se os objetivos propostos foram atingidos.

Por fim, o *MoveU* foi demonstrado a uma série de pessoas (dançarinos e espectadores), para que pudessem opinar sobre ele e indicar possíveis melhoramentos. Foram também criados uma série de questionários para que o público a quem foi demonstrado o protótipo, com a finalidade de realizar uma análise estatística para determinar se este sistema seria do agrado das pessoas e também permitir retirar conclusões sobre este trabalho.

Palavras-chave: Dança, Espetáculos, Sensores de Movimento, Efeitos, Interação, MoveU.

Abstract

Dance and their shows are activities that were practiced by Man since almost their existence. Throughout this period of time and up to modern times these activities have been undergoing a series of changes that made them remain relevant and with great importance in human society and its culture.

The evolution did not occur only in the styles of dances and performances but also in the accessories and effects that they implement, in order to make them more attractive to those who see them.

Despite these changes, most effects do not allow a level of interaction with the dance or show, making a clear separation between the pure dance and the scenery of the spectacle and the accessory component of effects.

In order to bridge this clear division of components, we started a study to conceive a system which would break down this barrier and join the two components in order to create interactive effects as result of a dance and movements of the dancer making the performance more interactive, and that are not just one more accessory component, this also makes the whole spectacle more appealing to the general public. To be able to create such a system, it was necessary to resort to the current motion sensing technologies in order to the bridge between the artist and the effects be achieved.

In the market there are several motion sensors offerings that serve to create the system, but only one could be chosen, in order to do that a study was done to determine which of these sensors would be most suitable for use in the system, having into account a variety of factors.

After choosing the sensor the *MoveU* system was developed, and at the end a series of tests was done to validate the prototype and verify that the proposed goals have been achieved for this.

Finally, the *MoveU* prototype was shown to a number of people (dancers and the spectators), so they could express their opinion on it and also indicate possible improvements. It was also created a series of questionnaires for the public to which the prototype was demonstrated so they could reply, in order to perform a statistical analysis to determine whether this system would be to the liking of the people, and also allow to draw conclusions about this work.

Keywords: Dance, Shows, Motion Sensors, Effects, Interaction, MoveU.

Agradecimentos

Ao meu orientador, o Professor Doutor António Vieira de Castro docente do Departamento de Engenharia Informática do ISEP (Instituto Superior de Engenharia do Porto) e investigador do GILT (*Games, Interation and Learning Technologies*) pela sua paciência, dedicação, incentivo e por todo o valiosíssimo apoio que incansavelmente me deu ao longo da realização deste trabalho.

Ao ISEP pela qualidade de ensino proporcionada e por ter proporcionado as condições necessárias à realização do presente estudo.

Ao LAMU (Laboratório Multimédia) pela cedência do Kinect de apoio ao desenvolvimento do protótipo "MoveU".

Á academia de dança Danciart e muito particularmente á professora Alexandra Rodrigues pela cedência do espaço para testes e pela autorização de participação de alguns dos seus alunos para experimentação do protótipo.

Á academia Global Dança e aos seus professores Liliana Guedelha e Carlos Guedelha pela cedência do espaço.

Aos dançarinos e professores de dança Teresa e Jorge pela experimentação do protótipo e pelas sugestões dadas.

Aos meus país e á minha namorada por todo o apoio incansável que me deram ao longo da realização deste trabalho.

A todos os outros que de alguma forma, tenham facilitado, cooperado ou permitido que esta possa ter sido uma tese melhor.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos e Símbolos

2D	Duas Dimensões, Bi-Dimensional
3D	Três Dimensões, Tri-Dimensional
API	*Application Programming Interface*
AVC	Acidente Vascular Cerebral
CCD	*Charge-Coupled Device*
FPS	*Frames per Second*
GUI	*Graphical User Interface*
HDR	*High Dynamic Range*
LED	*Light Emitting Diode*
MoveU	*Nome dado ao protótipo*
PC	*Personal Computer*
RGB	*Red Green Blue*
SDK	*Software Development Toolkit*
ToF	*Time-of-Flight*
USB	*Universal Serial Bus*
WSDL	*Web Service Definition Language*
ZDK	*Zigfu Development Toolkit*

Capítulo 1 - Introdução

"Dança telemática demonstra interação entre tecnologia e cultura"

Juliana Berriel

Neste capítulo é feito o enquadramento dos meios de comunicação e interação humana, mais concretamente a dança, isto tanto num contexto histórico como também temporal. São também mencionados os diversos aspetos da dança e a sua importância até á atualidade.

Também são identificadas as principais limitações que os artistas encontram atualmente nos seus espetáculos, sendo que são sugeridas soluções para resolver e eliminar essas mesmas limitações a fim de melhorar o espetáculo.

Tendo em conta as soluções sugeridas, são definidos objetivos que permitirão desenvolver e sustentar conclusões a fim de validar os resultados desde trabalho.

Por fim são também demonstradas as motivações por parte do autor na realização desde mesmo trabalho, como também um pequeno guia que demostra com brevidade a organização desta tese assim como também um pequeno resumo sobre cada um dos capítulos constituintes deste mesmo trabalho.

1.1 Introdução

Segundo Diniz [Diniz, 2015] ao longo dos séculos a dança foi evoluindo em diversos aspetos, tendo uma evolução a cada era que passava. Para os Egípcios a dança manifestava-se na forma de homenagear os deuses, para os Romanos a dança envolvia os temas de Reis, República e Império, no renascimento a dança já se tornou em algo mais artístico. Ao longo da história da humanidade a dança esteve presente nos mais diversos contextos, sendo estes religiosos, políticos, artísticos, cerimoniais, entre outros. No entanto, voltando às origens da dança, em que esta não passava mais do que uma forma de comunicar, recorrendo a gestos, padrões e expressões transmitir ideias, valores e estados de espirito, pode-se atestar que com a evolução ao longo da história esta passou a ser algo que transcende á própria comunicação, passando a ser sobretudo uma forma de arte.

Nos dias de hoje a dança tem um contexto mais artístico, dando origem aos mais diversos tipos de espetáculos de dança, com imensas variantes. Um facto relevante é que ao longo da história, a dança foi sendo acompanhada por diversos tipos de instrumentos. Nos seus primórdios seriam instrumentos básicos que ao longo dos tempos foram evoluindo, permitindo um acompanhamento percussivo que dava á própria dança uma outra dimensão. Atualmente esse acompanhamento transcendeu o mero apoio musical, pelo que nos dias de hoje quando se fala em dança, não se fala apenas no dançarino e nos instrumentos musicais mas também é necessário considerar efeitos, sejam eles de luz ou de alteração ao cenário.

Tendo em conta que hoje em dia a tecnologia está presente nas mais diversas áreas, e sendo que existem sensores que permitem capturar e analisar uma nova gama de métricas, foi idealizado utilizar uma dessas tecnologias para adicionar outra dimensão á dança, no sentido de tornar o espetáculo mais interativo. Tal como mencionado acima, as danças recorrem atualmente a uma série de efeitos de luz e de outros tipos como forma de acompanhamento da dança, para criar cenários apropriados ao espetáculo. Existem também efeitos tridimensionais que podem ser projetados numa tela de forma a criar cenários onde o dançarino possa executar a sua atuação. O problema desses efeitos é que não têm uma interação direta e em tempo real com o dançarino. Devido a esta “lacuna” foi idealizado este trabalho recorrendo á tecnologia que existe atualmente na área dos sensores de captura de movimento, para tornar esses efeitos tridimensionais sem interação em efeitos interativos controlados pelo dançarino, ou seja, á medida que o dançarino executa os seus movimentos de dança durante o seu espetáculo, os efeitos projetados no cenário interagem com ele tornando assim o espetáculo ainda mais imersivo e aliciante.

1.2 O Problema

Atualmente os dançarinos e outros tipos de artistas, vêm as suas atuações em espetáculos ao vivo (realizadas normalmente num palco) acompanhadas de inúmeros efeitos especiais como a luz, fumos, projetores de vídeo, efeitos pré gravados, áudio ou elementos como por exemplo a água e o fogo, que sincronizados com a coreografia proporcionam ao espectador sensações que se prolongam para além da dança proporcionando um espetáculo melhor.

Cada vez mais estes espetáculos são mediados por tecnologia e estes efeitos apesar de já serem um excelente acompanhamento para a atuação do bailarino, têm as suas limitações, como por exemplo o facto de não interagirem em tempo real com o artista.

Partimos do pressuposto que podemos contribuir para melhorar ainda mais o espetáculo de dança, dotando-o de um mecanismo interativo capaz de reagir em tempo real aos movimentos do bailarino. Deste modo, considerámos que podem ser colmatadas as limitações relacionadas com a interação em tempo real e que a sua inclusão num espetáculo poderá eventualmente contribuir para um cenário mais agradável, aumentando a própria interação entre o artista e o seu público.

Tendo em conta a panóplia de tecnologias que permitem a interação dos utilizadores em tempo real e de forma imersiva, como por exemplo o sensor *Kinect*, foi pensada uma forma de proporcionar aos bailarinos um mecanismo de interação com o meio que lhes permitisse acrescentar elementos gráficos á coreografia provenientes da sua performance durante a atuação em tempo real, tornando o espetáculo mais apelativo ao público.

1.3 Contributos esperados

Com o presente estudo pretende-se analisar o potencial de um mecanismo capaz de mediar a interação entre os movimentos corporais do bailarino transformando-os numa série de efeitos multimédia para integrar a coreografia nos espetáculos ao vivo.

Com este sistema a interação e a dança estarão de mãos dadas permitindo ao artista jogar com os efeitos para produzir um espetáculo único e irrepetível.

Tendo em conta a reflexão apresentada acima e considerando que existem diversas tecnologias no mercada que permitem interação em tempo real com o utilizador, foi decidido como objetivo preliminar analisar cada uma dessas tecnologias com o intuito de perceber as vantagens e constrangimentos de cada uma delas.

Seria importante perceber até que ponto essas tecnologias poderiam limitar os objetivos definidos para o presente estudo.

Teria também de ser estudada a viabilidade da aplicação de uma tecnologia desta natureza no contexto a que este projeto se destina, ou seja, a tecnologia escolhida teria que se adaptar e desempenhar as suas funções em meios como palcos de espetáculos, teatros, pavilhões, entre outras possibilidades. Cada um destes meios pode apresentar diferentes limitações á aplicação da tecnologia em análise.

Depois de estudada a tecnologia e a viabilidade da sua implementação para o fim a que se destina, será necessário estudar os *softwares e recursos* existentes que permitem não só

controlar a tecnologia como também criar aplicações que tirem maior partido desta, e como é claro, que também permita a realização dos objetivos definidos.

Assim sendo pretende-se criar um protótipo que interprete os movimentos do dançarino e que os transforme em diversos efeitos com a finalidade de serem projetados numa tela que ficará posicionada atrás do artista enquanto este desempenha a sua atuação. Ao mesmo tempo que a atuação é realizada para o público, existirá um sensor que irá capturar em tempo real os movimentos realizados pelo artista e tendo em conta esses mesmos movimentos transmita-os á aplicação e que por consequência permitam uma interação com os efeitos projetados na tela atrás do artista, ou seja, por exemplo quando o artista ergue o seu braço e este provoca um determinado efeito baseado numa reação em tempo real do sensor. Esse efeito será projetado no cenário em tempo real.

Deste modo, pretendemos contribuir para um tipo de espetáculos de dança mais interativos e estudar o seu potencial para cativar ainda mais o público.

1.4 Motivação

O autor do presente estudo é licenciado em Engenharia Informática e tem exercido a sua atividade profissional no âmbito da programação.

Desde a frequência do mestrado em Engenharia Informática no ISEP, que a sua visão sobre a aplicabilidade de técnicas de programação a outras áreas, como a dos sensores, lhe abriu horizontes e perspetivas.

Dado que nos dias de hoje os sistemas de interação estão em grande expansão e evolução o tema tornou-se de grande interesse para o autor e a área dos efeitos especiais resultante da aplicação destes sensores, assumiu um especial interesse.

A conjugação de áreas distintas é altamente motivante pelo facto de se poder verificar a aplicabilidade da programação a outas áreas de interesse social e humano.

Perante a hipótese a ligar o mundo da programação aos efeitos especiais aplicados a uma área como a dança, através de uma tecnologia interativa recente e inovadora, o autor começo por estudar os aspetos relacionados como a dança, os efeitos especiais aplicados durante os seus espetáculos e os sensores que poderão eventualmente acrescentar á dança a possibilidade de interação por parte do próprio interveniente, ou seja, o bailarino.

Toda esta envolvência e o contacto com pessoas e uma área tão distinta como é a dança são aspetos motivantes sobretudo quando associados ao que o autor mais gosta de fazer que é programar. Antevendo a possibilidade de criar um sistema desta natureza, e a sua possível implicação no mundo dos espetáculos de dança deram ao presente estudo um interesse natural e especial por parte do autor.

1.5 A organização desta tese

Este documento encontra-se dividido em 5 capítulos.

No capítulo 1 apresenta-se uma introdução geral ao tema, sendo identificado o problema que será abordado ao longo deste trabalho. Apresentam-se os contributos esperados no sentido de analisar novos meios para tornar um espetáculo de dança mais interativo e a motivação do autor para a realização do presente estudo. O capítulo termina com a apresentação da organização do presente documento.

No capítulo 2 apresenta-se um estudo sobre a dança, onde é feita uma análise a diferentes tipos de dança e efeitos tradicionalmente utilizados nelas. É também feita uma reflexão sobre os tipos mais adequados para a utilização com o protótipo que será desenvolvido.

São também analisados os sistemas de interação e os sensores de movimento, apresentando-se uma reflexão sobre os tipos de utilização desses sensores, terminando-se o capítulo com uma análise relacionada com o uso dos sensores aplicados á dança em espetáculos ao vivo.

No capítulo 3 é feito o estudo de todos os *softwares* passiveis de serem utilizados na construção do protótipo proposto, bem como também uma análise a todas as vantagens e desvantagens de cada um deles. Além disto é também referida a proposta de modelo feita para o trabalho, detalhando todas as fases da construção do protótipo. Sendo que numa última parte são referidos os testes efetuados ao protótipo tanto numa fase inicial como também numa fase final.

No capítulo 4 são feitas as validações e avaliação ao modelo proposto usando para isso uma análise estatística a fim de interpretar os resultados obtidos nos questionários que foram realizados por quem teve contacto com o protótipo.

No capítulo 5 são tiradas as conclusões acerca do presente trabalho bem como também os possíveis trabalhos futuros que possam advir do estudo realizado.

Capítulo 2 – A dança, os movimentos corporais e os sensores de movimentos

"A coisa mais indispensável a um homem é reconhecer o uso que deve fazer do seu próprio conhecimento."

Platão

Neste capítulo apresenta-se a correlação entre a dança e os seus movimentos corporais e os sensores de movimentos.

Inicia-se com um estudo da dança analisando os seus tipos e distinguindo as danças a solo de danças em grupo. Faz-se uma reflexão sobre os tipos de dança mais adequados ao protótipo. Analisam-se os espetáculos de dança e efeitos tradicionais inseridos neste tipo de espetáculos.

Analisam-se os sistemas de interação e os sensores de movimento. Apresenta-se uma reflexão sobre os tipos de utilização dos sensores de movimento, terminando-se o capítulo com uma análise relacionada com o uso dos sensores aplicados á dança em espetáculos ao vivo.

2.1 A dança

Dado que a área de aplicação do presente estudo é uma área distinta da engenharia da programação, ou seja, a área de trabalho do autor, o primeiro aspeto importante foi o de analisar o fenómeno que é a dança. Para podermos propor um sistema interativo nesta área, mediado por sensores e alicerçado em programação, seria necessário estudar e perceber previamente alguns dos principais aspetos relacionados com o tema.

Foi possível perceber, durante a fase de estudo preliminar, que a dança é "um tipo de arte" que geralmente envolve movimentos do corpo, sejam eles rítmicos ou simplesmente seguindo a sonoridade de uma música.

A dança é realizada em diversas culturas como uma forma de expressão emocional, de interação social ou simplesmente como um exercício físico em contexto espiritual, desportivo, lúdico recreativo ou de espetáculo, sendo muitas vezes utilizada para expressar

ideias ou contar até contar uma história. Para muitas pessoas dançar é uma paixão, um estado de espírito mas pode também ser uma profissão [Khan, 2015].

No próximo tópico são apresentados alguns dos principais tipos de dança praticados no mundo bem como uma breve descrição sobre cada tipo. São ainda explorados e explicados alguns efeitos tradicionalmente usados em diferentes tipos de dança.

A dança aparece sob inúmeros tipos sendo vários os benefícios para o corpo humano. Abaixo, referem-se alguns deles:

- Flexibilidade: Para praticar a dança, é importante muita flexibilidade. Por esse motivo, a pessoa deve realizar uma boa série de alongamentos antes de começar a praticar. Durante a própria dança, é exigido do dançarino que procure trabalhar o extremo de cada músculo do seu corpo mesmo que isso possa causar dores musculares [Tipos de Dança, 2015];
- Força: Quando a pessoa dança, ela está a forçar o seu corpo para que ele resista ao peso corporal. Os saltos de alguns tipos de dança exigem o uso de muita força [Tipos de Dança, 2015];
- Resistência: É importante preparar os músculos com uma série de exercícios para aplicar durante as coreografias. Com o corpo cada vez mais adaptado à prática de dança, o bailarino sentirá menos dores e desconforto muscular [Tipos de Dança, 2015];
- Bem-Estar: Com a dança, as pessoas podem criar um convívio e privilegiar o seu estado de espírito e valorização pessoal e social. A sensação de bem-estar é adquirida com as conversas e com a convivência com outras pessoas que compartilham a prática. Dançar, é acima de tudo um meio para manter uma vida mais saudável e feliz [Tipos de Dança, 2015].

2.1.1 Os tipos de dança

Nos próximos subcapítulos serão abordados diferentes tipos de dança, sendo estes divididos em 3 categorias distintas: danças a solo, danças a par e danças em grupo. Dentro de cada categoria são descritos 3 tipos de dança sendo feita uma breve descrição que apresenta o tipo de dança, abordando aspetos de índole cultural, como por exemplo as suas origens mas focalizando que tipo de movimentos são feitos.

2.1.1.1 As danças a solo

Este estilo consiste em danças que podem ser praticadas por uma única pessoa. Algumas destas danças podem também ser praticadas em grupo ou até mesmo em pares, mas não é um requisito obrigatório, sendo que o mínimo necessário é apenas uma pessoa.

- O ballet – O ballet é um tipo de dança que originou em Itália no século 15. É um tipo de dança bastante complicado que é ensinado em diferentes escolas de ballet em todo o mundo. A dança é geralmente coreografada com música de orquestra e envolve bastante fluidez e movimentos acrobáticos bastante precisos (fig. 1) [Khan, 2015]. No ballet existem 9 movimentos principais que os bailarinos executam, eles são: plié, tendu, jeté, rond de jambé, fondu, frappé, grand battement, adagio e en dehors [Lopes, 2015];

Figura 1 - Bailarina a praticar Ballet[1]

- A dança do ventre - A dança do ventre consiste principalmente no movimento das partes do ventre e coxas, sendo um tipo de dança praticado apenas por mulheres (fig. 2). Este tipo de dança teve as suas origens no médio oriente [Khan, 2015];

Figura 2 - Praticante de dança do ventre[2]

- A dança contemporânea – A dança contemporânea é um tipo de dança que tenta romper com as molduras clássicas da dança. Não apresenta nenhuma técnica específica nem um "corpo ideal". Esta tenta inovar nas temáticas e na relação com os espaços e outras artes. Este tipo de dança surgiu nos anos 60 nos Estados Unidos no seguimento da dança moderna

[1] Imagem retirada de http://2.bp.blogspot.com/-O-L9MhSgL1o/Ui87q6hhYdI/AAAAAAAABAk/yWeuDpZ6Bhw/s1600/state%2Bstreet%2Bdancer.png
[2] Imagem retirada de http://www.belasdicas.com/wp-content/uploads/2012/12/20150226-aulas-da-danca-do-ventre.jpg

[Duarte, 2008]. Uma aula de dança contemporânea começa com alguns exercícios simples de alongamento do corpo com ênfase numa postura correta, dando por sua vez origem a exercícios de técnica de chão que visa a consciencialização do movimento do corpo através do centro, permitindo assim que o mesmo mantenha o equilíbrio e previna lesões. Gradualmente a aula vai progredindo para exercícios mais complexos que permitem o corpo experimentar a expansão dos movimentos trabalhados anteriormente em pormenor (fig. 3) [Leite, 2015].

Figura 3 - Bailarina de dança contemporânea[3]

2.1.1.2 As danças a par

As danças a par são um estilo de dança que necessitam obrigatoriamente de um par de dançarinos para que possam serem dançadas. Este estilo de dança demarca-se dos restantes por apresentar danças bastante sensuais e românticas, sendo elementos que surgem naturalmente de danças feitas por pares.

- A salsa - A salsa é um tipo de dança originado em Cuba e geralmente requer um parceiro de dança, ou seja, é uma dança de pares (fig. 4), embora existem alguns estilos reconhecidos em solo. A salsa é geralmente dançada ao som de música salsa embora muitas vezes seja também utilizada música oriunda da América Latina. Este é um tipo de dança geralmente coreografada embora possa haver lugar a alguma improvisação. Esta dança é bastante popular na América Latina, mas ao longo do tempo expandiu-se para a América do Norte, Europa, Austrália, Asia e Médio Oriente [Khan, 2015]; Quando se dança Salsa, os bailarinos estão ligeiramente separados e mantêm todo o seu peso na ponta dos pés. A mão direita da mulher e a mão esquerda do homem podem estar dadas ou entrelaçadas, enquanto a mão direita do homem é pousada no ombro esquerdo da mulher ou, em alternativa, na sua anca.

[3] Imagem retirada de http://www.espacomovimente.com/site/wp-content/uploads/2015/09/dan%C3%A7a-contempor%C3%A2nea2-500x424.jpg

O braço esquerdo da mulher deve estar levemente apoiado no do homem. Apesar do passo de Salsa ser compacto, é extremamente relaxado, permitindo uma liberdade fluida e sensual. Os movimentos das ancas, principalmente dos homens, são igualmente tranquilos, até subtis [Passo Base, 2015a];

Figura 4 - Par de bailarinos a dançar salsa[4]

- O tango – O tango é um tipo de dança originário da Argentina no início do século 20. Este tipo de dança é realizado em pares, geralmente por um homem com uma mulher (fig. 5). O Tango é um tipo de dança em que o romantismo é exprimido pelos movimentos executados pelos bailarinos, sendo considerada uma dança sensual [Bedinghaus, 2015a]. Dançado de perto, no tango é o ombro esquerdo que conduz, enquanto o resto do corpo se mantém ligeiramente inclinado. Na posição inicial, os corpos estão em contacto, os joelhos estão ligeiramente comprimidos e os pés juntam-se verticalmente, ou seja a ponta do pé direito está junto à parte interna do pé esquerdo. A mulher posiciona-se sempre um pouco à direita do homem [Passo Base, 2015b];

Figura 5 - Praticantes de tango[5]

[4] Imagem retirada de http://www.danstudyo.com/eng/images/salsa-dance.png
[5] Imagem retirada de http://1.bp.blogspot.com/-Dfo8pQNA_lM/UWTXSiU4brI/AAAAAAAAtTM/kZjCCZ8A1PQ/s1600/tango-0102a.png

- O cha-cha-cha – O cha cha cha é um tipo de dança rítmica, que possuí uma energia e batidas constantes. Este tipo de dança é originário de Cuba e é contruído sobre a música do mesmo nome. Esta dança é uma variação do mambo [Danças de Salão, 2015]. A posição inicial dos dançarinos não requer o contacto dos corpos (cerca de 15 cm de distância entre o par) que devem manter-se o mais relaxado possível. Os bailarinos dão as mãos ao nível do queixo; enquanto a mão direita do homem pousa na região abaixo do ombro esquerdo da mulher, esta apoia o seu braço esquerdo sobre o braço direito do homem. Ambos os bailarinos mantêm os pés juntos (fig. 6), colocando a maioria do peso sobre o pé esquerdo (no caso do homem) e o pé direito (no caso da mulher). Os movimentos mais importantes do cha-cha-cha a dominar numa fase inicial são: Passo Básico, Passo Básico com Leque, Passo Básico terminado em Posição de Contra-Promenade Aberta e Posição de Contra-Promenade Aberta [Passo Base, 2015c].

Figura 6 - Par de dançarinos de Cha-Cha-Cha[6]

2.1.1.3 As danças em grupo

Este estilo de dança demarca-se dos restantes pelo facto destas danças serem feitas em grupos de várias pessoas ou um grupo de vários pares. As danças podem ter todos os seus participantes sincronizados a realizarem os mesmos movimentos ou podem também não serem sincronizados.

- A line dance - Este tipo de dança é coreografado e requer que uma determinada sequência de passos seja executada por um grupo de pessoas numa ou várias linhas (fig. 7), isto sem qualquer discernimento pelo sexo da pessoa, sendo que todas elas estão normalmente

[6] Imagem retirada de http://www.teachballroomdancing.com/wp-content/uploads/2013/07/dancing-couple-adult-dip.png

viradas na mesma direção e executam todos os movimentos ao mesmo tempo. Neste tipo de dança não existe qualquer tipo de contacto entre os participantes [Khan, 2015];

Figura 7 - Grupo de pessoas a praticar Line Dance[7]

- A roda de casino – Este tipo de dança tal como a salsa e o cha-cha-cha é originário de Cuba. O "Rueda de Cacino" é dançado ao som de música salsa, sendo que todos os participantes dançam em pares e formam um círculo (fig. 8), durante a dança os pares vão trocando constantemente, daí este tipo de dança ser considerado uma dança de grupo e torna esta um espetáculo vivido e alegre [The Latin World, 2015];

Figura 8 - Grupo de pares em roda a dançarem Rueda de Cacino[8]

- Dança folclórica – A dança folclórica é um tipo de dança desenvolvido por um grupo de pessoas e que reflete o modo de vida tradicional das pessoas de certa região ou país (fig. 9). Este tipo de dança originou-se no século 18 como forma de distinguir entre as danças das pessoas comuns das pessoas de classe alta [Bedinghaus, 2015b].

Figura 9 - Grupo de folk dance[9]

[7] Imagem retirada de http://i.ytimg.com/vi/CXkyw8wTRmI/maxresdefault.jpg
[8] Imagem retirada de http://i.ytimg.com/vi/Ry9qd5c1cDA/maxresdefault.jpg
[9] Imagem retirada de http://www.themoscowtimes.com/upload/iblock/ca3/5224-16b-moiseyev.jpg

2.1.2 Os tipos de dança mais adequados ao protótipo

O protótipo originalmente focalizará apenas um único tipo de dança para o qual está mais preparado e adaptado. Dadas as especificidades técnicas dos sensores, iremos fazer incidir o estudo no tipo de danças a solo. Isto não significa que não consiga desempenhar adequadamente as suas funções noutros estilos, mas o seu principal foco será o tipo de dança a solo para o qual foi adaptado. Esse tipo é o mais adequado para utilização com o protótipo por diversas razões, dentro das quais, as características do próprio sensor de movimentos. O sensor de movimentos tem uma limitação na quantidade de pessoas que consegue detetar e processar, portanto danças que contenham um elevado número de participantes não são de forma alguma adequadas para a utilização deste tipo de sensor. Em segundo lugar, e tendo em conta a limitação anteriormente descrita, os efeitos que serão desenvolvidos terão como foco as danças a solo, salvo algumas exceções. Tendo em conta que o sensor não consegue processar adequadamente uma grande quantidade de pessoas não serão desenvolvidos efeitos para essas situações. Quanto ao estilo de danças a par mais uma vez existe alguma limitação por parte do próprio sensor, isto porque o sensor tem dificuldades em detetar os movimentos do par quando estes se encontram numa posição em que estão bastante próximos um do outro, o que torna difícil ao sensor distinguir as formas básicas do corpo de cada um dos dançarinos levando a erros de interpretação. Quanto aos efeitos que serão desenvolvidos existe um que se pode utilizar com danças de pares (mas que obviamente terá a limitação por parte do hardware do sensor descrita anteriormente) e outros que são apenas e exclusivamente adequados a danças a solo.

2.1.3 Os espetáculos de dança e os efeitos especiais tradicionais

A dança tal como foi referido no início deste capítulo pode ter contextos distintos, tais como expressões emocionais, interações sociais, exercícios físicos, atividades lúdicas, recreativas, espirituais e de espetáculos. Nos próximos tópicos serão abordados diversos tipos de efeitos que são tradicionalmente utilizados no contexto de espetáculos de dança.

Ao longo dos anos, as danças fizeram-se acompanhar de efeitos baseados em métodos tradicionais, que proporcionam uma experiência sensorial e visual para a audiência. Estes espetáculos de dança podem recorrer a inúmeros efeitos ou serem mesmo inteiramente baseados neles. Apresentam-se em seguida alguns efeitos tradicionais que são utilizados em espetáculos de dança para criar um ambiente mais agradável e apelativo para os espectadores.

2.1.3.1 Utilização de sombras

Este tipo de dança recorre ao efeito de luz e sombras para proporcionar um espetáculo diferente do habitual. Basicamente recorre-se a uma ou várias fontes de luz que depois são projetadas numa tela semitransparente. Os dançarinos colocam-se entre a tela e a fonte de luz e fazem a sua atuação, sendo que as suas sombras são projetadas na tela e a audiência consegue assistir á dança. Como se pode ver na figura seguinte, a audiência apenas verá as sombras dos dançarinos a movimentarem-se e a dançarem.

Figura 10 - Dança com Sombras[10]

Este efeito é muito semelhante ao que acontece no típico teatro de sombras.

2.1.3.2 Efeitos de luz ou iluminação

A dança com luzes recorre a variados efeitos de luz com o intuito de tornar a dança mais apelativa aos espectadores. Os efeitos de luz podem ser concretizados de maneiras distintas. Podem ser luzes que se encontram no próprio corpo dos dançarinos, sendo que neste caso não existe nenhuma iluminação ambiente para que as luzes nos corpos dos dançarinos se destaquem, como se pode ver na figura abaixo.

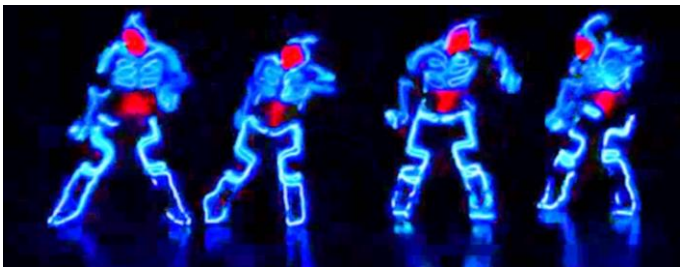

Figura 11 - Tron Dance[11]

[10] Imagem baseada no conteúdo disponibilizado em https://www.youtube.com/watch?v=GpG3Oq58xKY
[11] Imagem baseada no conteúdo disponibilizado em https://www.youtube.com/watch?v=OUeX95_mLwk

Podem também ser usadas projeções de luzes que incidem nos dançarinos á medida que estes desempenham a sua atuação (fig. 12 e 13).

Figura 12 - Dança com projeção de luzes sobre um grupo[12]

Figura 13 - Dança com projeção de luz para cada dançarino[13]

Em todos os casos estes efeitos proporcionam uma experiência de dança bastante mais imersiva e apelativa ao público face às danças sem qualquer efeito.

Como se ilustra na figura seguinte, os efeitos de luz são também bastante utilizados em ambientes lúdico-recreativos como discotecas, eventos festivos e afins, sendo que representam uma parte vital do ambiente proporcionado por esse tipo de eventos.

Figura 14 – Típicos efeitos de luz utilizados em ambientes lúdicos[14] [15] [16]

[12] *Imagem disponível em* http://wp.clicrbs.com.br/mundoitapema/files/2012/01/FESTIV1.jpg
[13] Imagem disponível em https://viagensculturais.files.wordpress.com/2013/01/nirmanika-vastu.jpg
[14] Imagem disponível em http://images.quebarato.com.br/T440x/super+discoteca+em+sua+festa+dj+som+e+luz+garcons+festas+teens__234388_2.jpg
[15] Imagem disponível em https://pixabay.com/static/uploads/photo/2014/03/11/07/15/silhouettes-285028_640.jpg
[16] Imagem disponível em http://4.bp.blogspot.com/_hKKn6qgVi94/TKPoBybDKtI/AAAAAAAABTQ/r4ulXEZmyyA/s1600/revival+capital.jpg

Os efeitos de luz, representam um valor acrescentado para a dança pelo que a procura de novos meios para tornar estes ambientes cada vez mais imersivos e interativos tem sido constante nos últimos anos e um exemplo do fenómeno é o uso de light floors como se ilustra na figura seguinte, onde o próprio chão se ajusta á posição do dançarino ao mesmo tempo que apresenta efeitos luminosos.

Figura 15 - Light Floor numa pista de dança[17]

2.1.3.3 A dança com fogo

A dança com fogo socorre-se de efeitos baseados em fogo para dar uma experiência única aos espectadores. Nestas danças os dançarinos podem vestir trajes com fogo (mas que os protegem de se queimar), objetos que emitem fogo, ou até mesmo líquidos inflamáveis que podem ser expelidos pelo dançarino com a finalidade de criar projeções de chamas (fig. 16). Em todos estes casos o fogo é o elemento predominante e este acompanha a atuação dos dançarinos tornando o seu espetáculo muito apelativo ao público.

Figura 16 – Dança com objeto que emite fogo[18] *e Dança com fogo emitido pelos dançarinos*[19]

[17] Imagem disponível em http://g02.a.alicdn.com/kf/HTB1WmG0IFXXXXbraXXXq6xXFXXXS/2-p%C3%A7s-lote-nova-etapa-interativo-levou-pista-de-dan%C3%A7a-luz-China-para-discoteca-boate-bar.jpg

[18] Imagem disponível em http://focacultural.com.br/wp-content/uploads/2012/08/IMG_28131.jpg

[19] Imagem disponível em http://1.bp.blogspot.com/-OH0LpREGrgs/Tr_6-WAWzDI/AAAAAAAAEwU/TrSUNE6g4xA/s400/I_Festival_de_Capoeira3-por_Silvia_Schneiders_%2528118%2529.jpg

De salientar que este tipo de efeitos podem ser perigosos e dar origem a acidentes que podem causar danos aos dançarinos, ao público ou às infraestruturas se forem utilizados sem as precauções necessárias.

2.1.3.4 A dança com água

A dança com água recorre a efeitos baseados em água para proporcionar um espetáculo bastante diferente do habitual á audiência (fig. 17).

Figura 17 - Dança com efeitos de água[20]

Neste tipo de danças são adicionados efeitos de água que podem interagir com os dançarinos de formas distintas. Por exemplo, jatos de água podem ser projetados nos dançarinos ao mesmo tempo que estes atuam a fim de os respingos serem desviados em diversas direções conforme os movimentos da dança. Outro caso é por exemplo os próprios dançarinos realizarem a sua atuação debaixo de água (fig. 18), criando assim um verdadeiro espetáculo subaquático. Em todos estes casos, a utilização de água como efeito acompanhante numa dança é algo que cativa o público.

Figura 18 - Tango de baixo de água[21]

[20] Imagem disponível em http://g1.globo.com/Noticias/PopArte/foto/0,,14659462,00.jpg
[21] Imagem disponível em http://revistapegn.globo.com/Revista/Pegn/foto/0,,69774874,00.jpg

2.2 Os Sistemas de Interação

Com a evolução dos sistemas de informação surge a necessidade de criar meios mais eficazes que permitam uma interação dos sistemas com o mundo real manipulada pelos utilizadores. Foi assim que começou o desenvolvimento de soluções que viessem possibilitar tais necessidades. Um dos primeiros a ser criado, e ainda hoje utilizado, é o teclado. O teclado foi utilizado em diversos sistemas de escrita e comunicação, mesmo antes de terem aparecido sistemas de informação digitais. Eram já usados por exemplo em máquinas de escrever, que se podem designar como dispositivos analógicos. Após essa fase, começou a era dos sistemas digitais, e o teclado mais uma vez esteve presente como o principal dispositivo de interação entre o Homem e a máquina, mesmo até aos dias de hoje.

Apesar da grande importância do teclado, algum tempo depois, já durante a era digital, mas ainda nos seus primórdios, surgiu outro meio de interação que veio revolucionar o mundo, o rato. O rato foi de grande importância pois permitiu que as interfaces com o utilizador sofressem uma revolução nunca antes vista. A interação que até esse ponto era apenas feita via linha de comandos, passou a poder ser feita através de grafismos e movimentos da mão do utilizador, o que tornou a experiencia de interação mais simplista e agradável.

No mundo dos videojogos os principais meios de interação nos seus primórdios foram o *joystick* e o controlador (*gamepad*), meios estes também bastante revolucionários, que tal como o teclado e o rato são ainda utilizados nos dias de hoje.

Apesar destes dispositivos de grande importância e verdadeiramente revolucionários, foram também desenvolvidos ao longo dos tempos outros meios de interação que tinham como objetivo complementar ou substituir por completo os existentes acima mencionados. Muitos desses meios porém não tiveram qualquer sucesso.

Durante a década de 90, começaram a aparecer no mundo dos videojogos dispositivos que permitiam detetar por exemplo para onde o utilizador estava a apontar o dispositivo e outros que permitiam ao utilizador ficar imersivo num mundo 3D através de óculos especiais. A maioria das tecnologias desta era não teve grande sucesso, muito devido a serem pouco refinadas e a terem problemas que impediam uma interação fluida com o utilizador. Foi a partir dos meados dos anos 2000, que começaram a aparecer tecnologias com o potencial para singrar no mundo da interação. Algumas delas serão abordadas com mais detalhe seguidamente. Durante o período entre a década de 90 e os anos 2000, não surgiram tecnologias de relevância nos meios de interação com o utilizador, sendo que o teclado e o

rato continuaram a ser o meio dominante. Após esse período surgiu uma tecnologia de interação que está atualmente no seu exponencial máximo, que já provou estar, em termos de popularidade e utilização, ao nível do rato e do teclado. Essa tecnologia é o ecrã de toque (*touch screen*), que está presente atualmente em força no segmento dos dispositivos móveis e que já se provou ser um meio de interação bastante eficaz. Não falarei com maior detalhe sobre esta tecnologia, pois está fora do contexto deste trabalho.

A tecnologia de interação que está em foco neste estudo é a dos sensores de movimentos. Estes sensores são uma tecnologia mais ou menos recente, e que começou por aparecer em maior força no mundo dos videojogos. Mais especificamente este trabalho vai focar-se nos sensores de movimento desenvolvidos pela *Microsoft* (*Kinect*) e serão analisados os seus respetivos concorrentes, sendo estas tecnologias analisadas a seguir.

Para concluir esta breve análise dos sistemas de interação, pode-se referir que foi uma área que foi evoluindo ao longo dos anos, mas que dia após dia está a tomar uma posição cada vez mais importante na sociedade atual. Algo que começou apenas como sistemas usados num segmento muito restrito, tornou-se em algo que é utilizado á escala mundial por praticamente todas as pessoas no mundo, e precisamente devido a este fator, é algo que cada dia assume uma maior importância mas também reações críticas.

2.3 Os Sensores de Movimento

Os sensores de movimento são dispositivos que permitem aos utilizadores interagirem com um determinado sistema através de movimentos do corpo, gestos, comandos de voz e outros tipos de ações naturais. Esses movimentos são reconhecidos, processados e traduzem-se em determinadas ações no sistema com que o utilizador está a interagir [Rouse, 2011].

Ao longo do tempo os sensores de movimento foram sofrendo uma grande evolução como se apresenta de seguida.

2.3.1 A evolução

Ao longo dos tempos foram várias as empresas a desenvolverem tecnologias que permitissem uma maior interação com o utilizador. Foi um tipo de tecnologias que atraíram bastante as empresas, mas que nos seus primeiros estágios, sofriam de graves falhas, que fizeram com que este tipo de tecnologia não tivesse uma grande adoção [Greenberg, 2012], [Norris, 2014].

Uma das primeiras empresas a desenvolver este tipo de tecnologia foi a *Sega*, mais concretamente, desenvolveu um sistema de deteção de movimentos para as pistolas (acessório dos jogos do tipo *shooter*) que eram usadas nas consolas *Master System* e *Megadrive (fig. 19)*. Basicamente o sistema de deteção de movimento era utilizado nestas pistolas para detetar a posição para a qual o utilizador está a apontar e verificar se coincide com o alvo do ecrã. De forma a este sistema funcionar a pistola continha um foto-díodo no cano, sendo que este foto díodo é capaz de capturar a luz emitida pelo ecrã. Ao efetuar um disparo se a luz capturada pelo díodo coincidir com o tipo de luz esperada pelo alvo, significa que o utilizador acertou no alvo, caso contrário errou, portanto o funcionamento do sistema era á base da captura de luz e em cálculos de trajetórias [Gamers Teresina, 2012].

Figura 19 – O Sega Light Phaser[22]

Após a *Sega* ter criado esta tecnologia, a *Nintendo* decidiu criar um rival, o *Super Scope*. Ao contrário da *Sega* o *Super Scope* operava já sem fios e fazia uso já de um sensor acessório, que teria de ficar em cima da televisão onde seria utilizado o *Super Scope (fig. 20)*. Sem esse acessório não seria possível utilizar a tecnologia, pois o sinal emitido pelo *Super Scope* era recebido e processado pelo sensor acessório. Este pode se dizer que foi o sistema percursor ao da *Nintendo Wii* [Gamers Teresina, 2012].

Figura 20 – O Nintendo Super Scope e sensor acessório[23][24]

[22] Imagem disponível em http://upload.wikimedia.org/wikipedia/commons/6/61/Sega_Master_System_lightphaser.jpg
[23] Imagem disponível em http://38.media.tumblr.com/6f3adaec4efc9b0359f94e8d20ca3c06/tumblr_inline_n3s0r7k8we1rjw2mu.png
[24] Imagem disponível em http://www.thestrong.org/online-collections/images/Z005/Z00534/Z0053469.jpg

Depois destes primeiros sensores, que se podem atualmente denominar como arcaicos, foi a vez da *Sony* mostrar algo de verdadeiramente revolucionário, essa revolução veio pelo nome de *Eye Toy (fig. 21)*. Este sensor é baseado numa câmara, e é o percursor mais parecido com o *Kinect*. A câmara deteta o utilizador e os seus movimentos, permitindo que este tenha interação com jogos usando apenas o seu corpo, tal como acontece com o *Kinect* atualmente [Gamers Teresina, 2012].

Figura 21 – O PlayStation Eye Toy[25]

Pouco tempo depois do *Eye Toy* aparecer a *Nintendo* lançou o *Wii Mote* (fig. 22), que nada mais é que uma espécie de comando remoto que deteta os movimentos feitos sobre este mesmo com um elevado nível de precisão [Gamers Teresina, 2012].

Figura 22 – O Nintendo WiiMote[26]

Em seguida foi a vez de a *Microsoft* lançar o *Kinect* (fig. 23) [Gamers Teresina, 2012], tecnologia que irá ser aprofundada em mais detalhe abaixo, pois será sobre ela que irá incidir o tema deste trabalho.

Figura 23 – O Microsoft Kinect V1[27]

[25] Imagem disponível em http://upload.wikimedia.org/wikipedia/commons/8/8e/PS2-Eyetoy.jpg
[26] Imagem disponível em http://theultimategamer.com/wp-content/uploads/2014/03/nintendo-wiimote.png
[27] Imagem disponível em http://upload.wikimedia.org/wikipedia/commons/f/fe/KinectSensor.png

Para concluir esta breve análise da história dos sensores de movimentos, pouco depois de a *Microsoft* ter lançado o *Kinect*, a *Sony* decidiu lançar um "clone" do *Wii Mote*, chamado *PlayStation Move* (fig. 24). Este em quase todos os aspetos similar ao *Wii Mote* da *Nintendo*, tirando o facto de que além dos controlos remotos conterem sensores de deteção de movimentos, estes também são auxiliados por uma camara que deteta a luz emitida por estes mesmos e assim permitir ainda uma maior precisão [Gamers Teresina, 2012].

Figura 24 – O PlayStation Move[28]

2.3.2 O Leapmotion

O *Leapmotion* (fig. 25) é uma tecnologia criada pela empresa do mesmo nome, e que tem como finalidade ser um sensor capaz de detetar os movimentos das mãos do utilizador e por sua vez usar esses movimentos para controlar uma variedade de aspetos no dispositivo ao qual o *Leapmotion* foi conectado [Alves, 2014].

Figura 25 - O dispositivo Leapmotion[29]

Esta tecnologia foi apresentada ao mercado em 2013, sendo que uma das características que o distingue mais dos seus concorrentes é o tamanho compacto [Alves, 2014]. Esta tecnologia faz uso de 3 LEDs infravermelhos e 2 câmaras CCD, o dispositivo captura uma área hemisférica com dimensão de cerca de 0.23 metros cúbicos, dentro da qual os LEDs geram um padrão de pontos 3D que são depois refletidos nas mãos do utilizador e por sua vez essas mesmas reflecções são capturadas 200 vezes por segundo pelas câmaras de infravermelhos, isto tudo com o objetivo de capturar os movimentos da mão do utilizador, sendo que depois esses movimentos são processados através de uma série de algoritmos matemáticos pelo *software* e consequentemente traduzem-se em interação no ecrã [Weichert et al., 2013], [Leapmotion, 2015c]. É de notar também que esta tecnologia é capaz de distinguir os

[28] Imagem disponível em http://qwentertain.com/wp-content/uploads/2013/12/PS-Move-1.jpg
[29] Imagem disponível em http://blog.leapmotion.com/wp-content/uploads/2013/04/device-hero.png

movimentos dos 10 dedos das mãos do utilizador, sendo que consegue até detetar diferenças de movimentos na ordem dos centésimos de milímetro individualmente para cada dedo, e tudo isto com uma latência tão baixa que é completamente impercetível ao olho humano [Alves, 2014]. Isto claro traduz-se numa experiencia de utilização bastante fluida, natural e agradável ao utilizador.

Para que o processamento dos movimentos do *Leapmotion* seja convertido em instruções para o sistema operativo interpretar é necessário um *software*, *software* esse que é necessário instalar para que o *Leapmotion* funcione corretamente [Alves, 2014]. Este *software* também se faz acompanhar de uma loja de aplicações chamada *Airspace* (fig. 26), muito ao estilo do *Google Play Store* e da *Apple App Store*, que permite ao utilizador fazer *download* de uma gama de aplicações que permitem interagir com o *Leapmotion* [Leapmotion, 2013d]. Esta loja de aplicações é constituída tanto por aplicações pagas como também aplicações gratuitas [Leapmotion, 2014a].

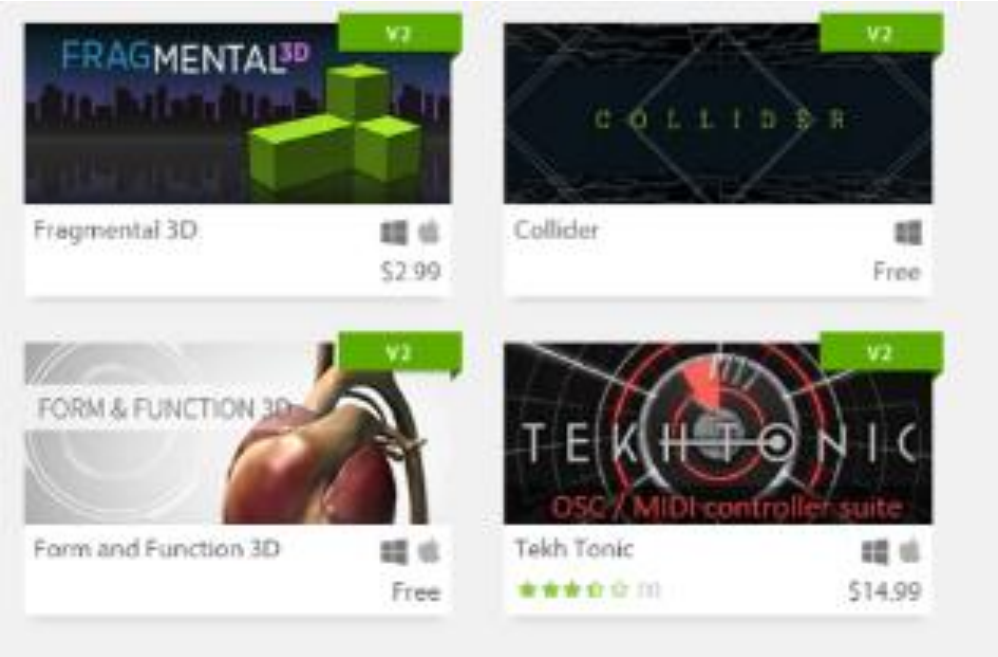

Figura 26 - Loja de Aplicações do Leapmotion, a Airspace[30]

Para o utilizador o uso desta tecnologia é bastante simples e intuitiva, basta apenas posicionar o *Leapmotion* entre o teclado do PC e o próprio utilizador, depois conectar o *Leapmotion* ao PC, instalar o respetivo *software* e por fim treinar os movimentos básicos recorrendo às suas mãos (fig. 27). Após isso o utilizador está pronto para tirar todo o partido do *Leapmotion* [Alves, 2014].

[30] Imagem baseada no conteúdo disponível em https://airspace.leapmotion.com/

Figura 27 - Exemplos do uso do Leapmotion[31] [32] [33]

Quanto aos movimentos em si, estes são capturados sensivelmente em duas zonas distintas, uma mais próxima do próprio utilizador, que permite capturar movimentos com pouca profundidade, como por exemplo o simples mover do cursor do rato, e outra zona mais próxima do monitor do PC, sendo que esta zona tem como objetivo capturar movimentos com uma maior profundidade, como por exemplo cliques num botão no ecrã [Alves, 2014].

A empresa também disponibiliza para os programadores um SDK que tem como objetivo permitir que estes criem aplicações que tirem partido do *Leapmotion*, aplicações essas que podem depois ser publicadas na loja de aplicações do *Leapmotion*, a *Airspace* [Leapmotion, 2014b].

Foram também estabelecidas parcerias com várias empresas, como por exemplo a *Asus* e a *HP*, a fim de integrar a tecnologia *Leapmotion* nos seus respetivos produtos, mais concretamente nos seus *Laptops* e *all-in-one* PC [Gorman, 2013], [Lee, 2013].

Este produto tem um preço de venda direto ao cliente de 89.99€, sendo o dispositivo comercializado diretamente do seu *site*, pois noutras plataformas de venda o seu preço poderá variar [Leapmotion, 2015e].

Quanto ao sucesso deste produto existem várias opiniões distintas, umas mais otimistas, outras menos otimistas, sendo que a opinião geral é de que apesar de ser um produto bem concebido e de desempenhar bem as suas funções, a sua utilidade na vida real fica um pouco aquém das expectativas [Hutchinson, 2013], [Etherington, 2013].

No contexto deste trabalho, esta é uma tecnologia que apesar de inovadora e bastante interativa, tem um problema que logo á partida que impede a sua escolha em detrimento de tecnologias rivais, esse problema é o facto de esta tecnologia só captar os movimentos das

[31] Imagem disponível em http://s2.glbimg.com/ekMhgAABVGnpR0Gq5ORIQxHvryU=/695x0/s.glbimg.com/po/tt2/f/original/2014/05/19/depois-de-instalado-basta-movimentar-os-dedos-sobre-os-sensores-para-operar-o-pc.png

[32] Imagem disponível em http://s3.amazonaws.com/digitaltrends-uploads-prod/2013/09/leap-motion-hp-laptop.jpg

[33] Imagem disponível em http://ecx.images-amazon.com/images/I/71BvThxnsXL._SX522_.jpg

mãos, sendo que o objetivo neste trabalho é de captar os movimentos do corpo inteiro do utilizador. Assim sendo esta tecnologia fica de fora das escolhas e outras alternativas terão de ser encontradas.

2.3.3 O Panasonic D-IMager

Depois do lançamento do *Kinect* por parte da *Microsoft*, várias foram as empresas que viram uma oportunidade de competir num novo mercado, uma dessas empresas foi a *Panasonic*, e assim sendo decidiram lançar um concorrente ao sensor da *Microsoft*, o *D-IMager (fig. 28)*.

Figura 28 - Panasonic D-IMager[34]

Este sensor, tal como o *Kinect*, permite reconhecer os movimentos do corpo e gestos das mãos a fim de utilizar essa mesma informação para controlar de forma interativa a mais variada espécie de conteúdos, tais como jogos, aplicações medicinais, aplicações imersivas multimédia, entre outras. [Panasonic, 2014a], [Vochin, 2010].

O *D-IMager* faz uso do princípio *Time-of-Flight* (ToF) (fig. 29) em conjunção com um sensor CCD e um *array* de LEDs Infravermelhos para detetar os movimentos do corpo e mãos do utilizador. Outras características interessantes do *D-IMager* prendem-se com o seu consumo energético de apenas 0.4A, o que permite que o seu *design* seja bastante pequeno, e que não tenha de recorrer a ventoinhas para arrefecimento. Os ângulos de visão máximos de captura do sensor são de 60 graus na horizontal e de 44 graus verticais, sendo que a distância mínima de deteção é de 1.2m e a máxima de 9m. O sensor é também capaz de capturar as imagens a uma velocidade de 30 FPS [Panasonic, 2014a], [Panasonic, 2014b], [Vochin, 2010].

[34] Imagem disponível em http://i1-news.softpedia-static.com/images/fitted/620x348/Panasonic-Develops-Kinect-Killer-the-D-IMager.jpg

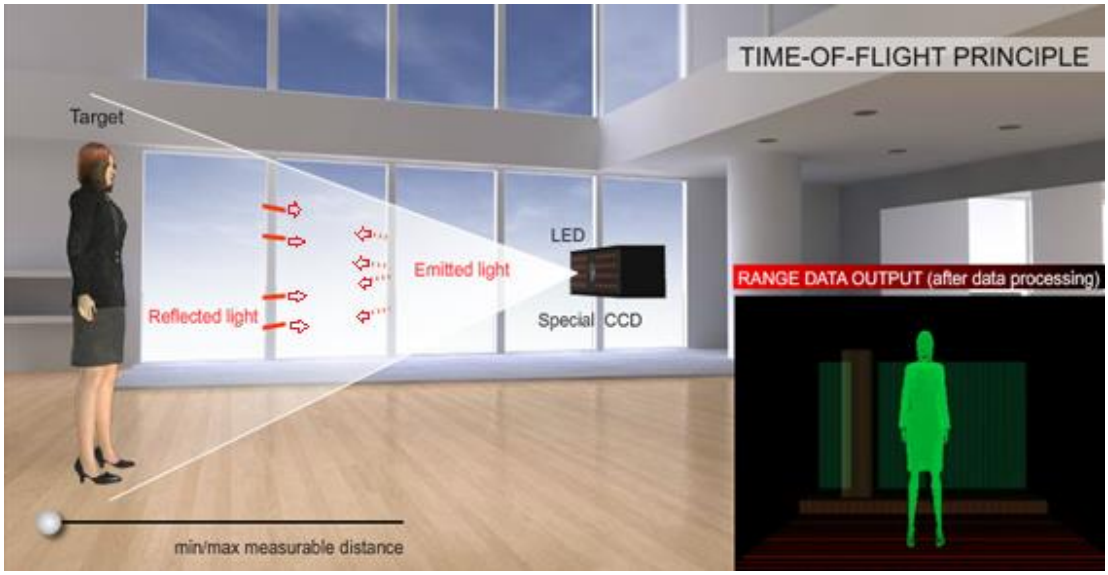

Figura 29 - Principio Time-of-Flight[35]

A *Panasonic* também permite o acesso gratuito (mediante a compra do *D-IMager*) a um SDK que permite desenvolver aplicações para o seu respetivo sensor, este SDK tem o nome de *Omek Beckon™ Development Suite*, e não foi criado pela *Panasonic* mas sim pela empresa *Omek* [Panasonic, 2014c].

É importante de referir que no dia 5 de Março de 2014 a *Panasonic* anunciou que este produto irá ser descontinuado nos finais de Dezembro de 2014. As razões por de trás deste anúncio não são inteiramente conhecidas, mas suspeita-se que tal terá acontecido devido ao baixo volume de vendas e pouco sucesso do produto [Panasonic, 2014a].

Esta é uma tecnologia com um enorme potencial, e que cumpre exatamente aquilo a que se compromete, mas devido a uma serie de fatores, esta tecnologia não foi escolhida para a realização deste trabalho. Um desses fatores é o facto de não existir uma grande base comunitária de programadores que desenvolvam para este dispositivo, ou seja, não existe uma grande base de documentação nem de suporte, o que dificulta naturalmente a utilização desta plataforma para o trabalho. Outro motivo prende-se com o facto de esta plataforma num futuro próximo ser completamente descontinuada, portanto invalidando definitivamente qualquer suporte futuro por parte da própria *Panasonic*.

[35] Imagem baseada no conteúdo disponibilizado em http://www2.panasonic.biz/es/densetsu/device/3DImageSensor/en/tech.html

2.3.4 O Creative Senz3D

A *Creative* decidiu lançar um dispositivo que pudesse concorrer com o *Kinect* da *Microsoft*, e como tal criou o *Creative Senz3D* (fig. 30). Este dispositivo captura os movimentos do utilizador para que estes sejam interpretados e permitam interação com diversas aplicações.

Figura 30 - Creative Senz3D[36]

O *Senz3D* permite detetar movimentos em distâncias entre 15 a 99 cm [Creative, 2014] (sendo que dispositivos rivais como o *Kinect* e o *Asus Xtion Pro Live* conseguem uma melhor gama de distâncias), estas distâncias podem não parecer longas, e de facto não o são, mas apesar disso o *Senz3D* consegue distinguir movimentos individuais das mãos e até dos dedos, o que permite por sua vez uma ótima experiência de utilização [Williams, 2013].

Em termos de características técnicas, o sensor da *Creative* consegue capturar imagens RGB a uma resolução de 1280x720 e imagens de profundidade a uma resolução de 320x240, e ambas a 30 FPS. Tal como mencionado anteriormente o sensor consegue detetar movimentos nas distâncias entre 15 a 99 cm e num angulo máximo de 74 graus [Creative, 2014].

Tal como os outros sensores o *Senz3D* faz-se acompanhar de um SDK de desenvolvimento de aplicações, sendo que o SDK utilizado é o *Intel Perceptual Computing* SDK [Creative, 2014]. Além do SDK também é acompanhado de uma variada gama de *software* que permite ao utilizador tirar partido do sensor, sendo que contem *software* desde jogos até *software* multimédia interativo [Creative, 2014]. O preço de venda deste sensor é de 199$ (157€) [Williams, 2013].

O sensor da *Creative* não foi escolhido para este trabalho devido a uma série de motivos, primeiro, e tal como outros sensores, tem uma base comunitária muito pequena, e além disto apresenta um alcance de deteção pequeno, o que podia se mostrar insuficiente para o contexto deste trabalho, onde serão necessárias maiores distâncias de deteção. Por último

[36] Imagem disponível em http://assets.hardwarezone.com/img/2013/10/creative_senz3d_610x443.jpg

o preço é também outro fator que deixa a desejar neste produto, pois outros sensores de outros fabricantes apresentam características mais atrativas a um preço igual ou inferior ao do *Senz3D*.

2.3.5 O Kinect

O *Kinect* é uma tecnologia que emergiu para dar resposta á falta de meios de interação diversificada no mundo dos videojogos, mundo este que até á pouco estava concentrado apenas em controlar a interação com os jogos via comando/teclado e rato. O *Kinect* veio mudar a forma como a interação é feita entre o utilizador e o jogo, permitindo-o controlar através do seu corpo, gestos, etc. Isto permite um maior nível de imersão entre o utilizador e o jogo [Microsoft, 2014a].

Não seria de esperar que este dispositivo de interação abrisse caminho para utilizações fora dos videojogos, mas depressa se começou a ver a sua utilização nas mais diversas áreas e campos, como por exemplo, na medicina (utilização em salas de cirurgia de forma a apoiar o procedimento como pode ser visto na figura 31), lojas (paredes interativas que reagem á passagem do cliente e mostram informações dos respetivos produtos da loja), demonstrações interativas, montras virtuais (que interagem com o cliente de forma a mostrar "virtualmente" por exemplo como determinada peça de roupa ficará ao cliente como se pode ver na figura 32), entre outros. [Globo, 2010], [Lact tin, 2013], [KinectHacks.net, 2010], [Vsauce, 2010], [Royal Institute of Technology, 2011], [MSDN, 2014c].

Figura 31 - Utilização do Kinect em medicina[37]

[37] Imagem disponível em http://images.gameskinny.com/gameskinny/17efa887bf0c1679798c6c921441562a.jpeg

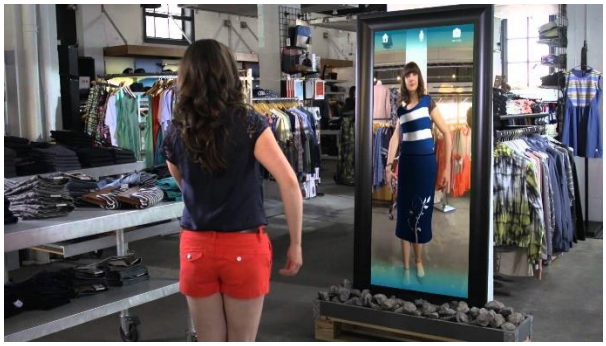

Figura 32 - Kinect utilizado numa montra interativa[38]

Devido á grande adesão a este sistema e diversas aplicações fora do mundo dos vídeo jogos, a *Microsoft* decidiu criar uma versão do *Kinect* que fosse suportada no PC, e mais tarde até desenvolveu um SDK que permite de uma forma mais fácil, desenvolver novas formas de aplicar este dispositivo em novos cenários [MSDN, 2014a]. Inclusive recentemente outras tecnologias do âmbito de imersão interativa decidiram usar o *Kinect* para aprimorar e aperfeiçoar a experiencia oferecida pelas suas tecnologias, um caso exemplo é a da tecnologia do *Oculus Rift* (tecnologia esta que permite ao seu utilizador ficar imersivo num ambiente 3D e que reage de acordo com o movimento da sua cabeça recorrendo a giroscópios e lentes que permitem imersão tridimensional), apesar de ser já neste momento uma tecnologia com imenso potencial, e que, devido a esse mesmo potencial foi adquirida pelo *Facebook*, ainda não se encontra totalmente aperfeiçoada [McEntegart, 2014].

Com o objetivo de colmatar as imperfeições que restam, a *Oculus* pensou em recorrer á tecnologia do *Kinect* para desta forma melhor detetar os movimentos e posições do utilizador, ao contrário daquilo que simples giroscópios e acelerómetros podem oferecer. Isto apenas indica o quão importante e quanto potencial tem para oferecer a tecnologia do *Kinect* [McEntegart, 2014].

Quanto a características técnicas o *Kinect* tem uma câmara RGB que permite capturar imagens a uma resolução de 640x480 a 30 FPS e uma câmara de profundidade com uma resolução também ela de 640x480 a 30 FPS. O *Kinect* consegue detetar movimentos em distancias entre os 0.7m a 6m e os ângulos máximos de captura são de 57 graus na horizontal, 43 graus na vertical, mas devido ao facto do *Kinect* ter um motor na base que permite rodar verticalmente o sensor os 43 graus verticais são estendidos, pois o motor consegue rodar o sensor por 27 graus para cima ou 27 graus para baixo [MSDN, 2015e]. Ao contrário de alguns

[38] Imagem disponível em http://i.ytimg.com/vi/Mr71jrkzWq8/maxresdefault.jpg

sensores de outras marcas, o *Kinect* necessita de uma fonte de energia externa, o que torna a portabilidade do sensor um pouco mais difícil [Gilbert, 2010].

Tal como as tecnologias rivais, o *Kinect* faz uso da projeção de infravermelhos para criar um mapa de pontos, que depois por sua vez são refletidos pelos obstáculos (utilizador e outros objetos presentes no ambiente) e por sua vez capturados pelo sensor de infravermelhos (fig. 33), e que depois são traduzidos numa imagem de profundidade. Sendo que essa informação de profundidade é depois processada através de algoritmos de forma a detetar se houve movimentos por parte do utilizador, movimentos esses que depois são traduzidos em ações nos mais diversos *softwares* ou jogos que o utilizador esteja a controlar [MacCormick, 2015].

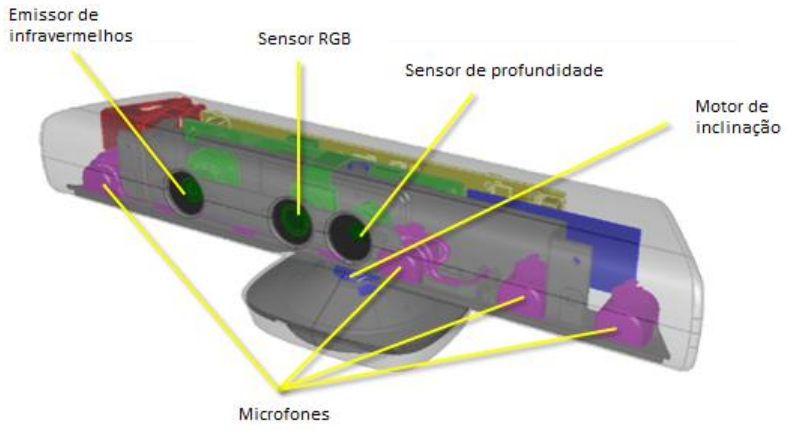

Figura 33 - Os vários elementos constituintes do Kinect[39]

Tal como mencionado acima, o *Kinect* faz-se acompanhar de um SDK de desenvolvimento de *software*, SDK esse que foi produzido pela própria *Microsoft*, e tem como nome simplesmente *Kinect* SDK [Eisler, 2012]. Este SDK, tal como outros que acompanham os sensores de outras marcas, permite que programadores consigam desenvolver aplicações que façam uso do *Kinect*, e assim até poder vender essas mesmas aplicações [Eisler, 2012].

Um facto importante a reter, é que a tecnologia utilizada pelo *Kinect* foi desenvolvida pela empresa *PrimeSense*, tecnologia essa que também foi utilizada no sensor da *Asus*, o *Xtion Pro Live* [MIT Technology Review, 2015], [Hollister, 2011]. Devido a isto, é possível utilizar em alternativa ao *Kinect* SDK, um SDK diferente, que se chama *OpenNI Framework*, isto porque a tecnologia da *PrimeSense* é *open source*, e isso permitiu que fosse criada uma *framework*

[39] Imagem adaptada com base na disponibilizada em https://i-msdn.sec.s-msft.com/dynimg/IC584396.png

que funcionasse com a tecnologia da *PrimeSense* [Mitchell, 2010]. Quanto ao preço de venda ao público do *Kinect*, este situa-se nos 210€[40].

Apesar de haver diversas ofertas no mercado dos sensores de movimento 3D, este encontra-se dominado pelo sensor da *Microsoft*, o *Kinect*, isto muito por culpa do facto de este sensor ser acompanhado das *Xbox 360* e *Xbox One* que a empresa vende, e por essa mesma razão, á semelhança do que acontece nos PC que vêm de origem com o *Microsoft Windows*, isto resulta na dominação do mercado, pois os clientes não têm tendência a mudar ou a comprar outro sensor, se a sua consola já vem acompanhada de origem com um sensor que funciona bem e que satisfaz as suas expectativas [Barrett, 2011]. Devido a esta mesma razão, o mercado dos sensores de outros fabricantes pode-se chamar de um nicho, que apenas apela a um grupo restrito de clientes, sendo que a maioria destes são programadores, que fazem uso destas tecnologias com vista a desenvolver aplicações que façam uso destas mesmas.

2.3.6 O Asus Xtion Pro Live

O *Asus Xtion Pro Live* é um dispositivo que tal como o *Kinect*, permite detetar os movimentos do utilizador e traduzi-los em interação com as mais diversas aplicações ou jogos. Este dispositivo foi lançado pela *Asus* para rivalizar com o *Kinect*.

Ambos os dispositivos apresentam imensas semelhanças, quer no seu formato, como também nas suas tecnologias. Isto deve-se principalmente ao facto da tecnologia que se encontra dentro de cada um destes dispositivos ter sido desenvolvida pela mesma empresa, a *PrimeSense* [Ipisoft, 2013]. Posto isto ambos apresentam um desempenho bastante similar, mas apesar disso, existem algumas pequenas diferenças que fazem distinguir estes dois produtos.

Uma das principais diferenças reside na qualidade da imagem produzida pela câmara RGB, sendo que o *Asus Xtion Pro Live* produz uma qualidade de imagem RGB superior á do *Kinect* [Ipisoft, 2013]. O sensor da *Asus* também não necessita de uma fonte de energia externa, sendo que pode ser inteiramente alimentado pela própria ligação USB ao computador [Ipisoft, 2013]. No caso do *Kinect*, este necessita de alimentação externa, o que pode tornar-se num inconveniente em termos de portabilidade [Ipisoft, 2013]. Uma característica que favorece o *Kinect* em relação ao *Xtion*, é o facto de o *Kinect* possuir um motor na base que permite controlar a inclinação do sensor para variados ângulos, esta é uma característica bastante útil, e que no caso do *Xtion* (fig. 34) não se encontra presente [Ipisoft, 2013]. Por

[40] Preço consultado através do site KuantoKusta.pt

ultimo é de notar que os *drivers* do *Kinect* têm uma qualidade superior do que os utilizados pelo *Xtion* [Ipisoft, 2013].

Figura 34 - Diferentes tipos de câmaras e sensores do Xtion Pro Live[41]

Quanto às características do *Asus Xtion Pro Live* destaca-se a distância de deteção, que tem um valor mínimo de 0.8m e um valor máximo de 3.5m. Consegue também detetar movimentos num angulo horizontal máximo de 58 graus, vertical de 45 graus e diagonal de 70 graus [Asus, 2014b]. O dispositivo consegue capturar imagens de profundidade com a resolução máxima de 640x480 a 30 FPS, ou de 320x240 a 60 FPS, sendo que a resolução máxima da câmara RGB é de 1280x1024 [Asus, 2014b].

É também de notar que o *Asus Xtion Pro*, tal como os sensores analisados anteriormente, se faz acompanhar de um SDK de desenvolvimento de aplicações, sendo que este SDK é o *OpenNI* SDK [Asus, 2014b]. Além do SDK o sensor também terá no futuro uma loja de aplicações que irá permitir os seus utilizadores descarregarem aplicações que façam uso do sensor, e também permitir aos programadores, colocarem á venda as suas próprias aplicações [Asus, 2014a]. Para além disto o sensor faz-se acompanhar já de alguns jogos e aplicações para que utilizador possa logo tirar partido do sensor [Landim, 2013]. Quanto ao custo, o *Asus Xtion Pro Live* está á venda com um preço de 171,50€[42].

O Asus Xtion Pro Live é sem duvida um excelente produto, com um bom desempenho, mas sofre de alguns problemas que o levam a não ter sido escolhido para a realização deste trabalho. Um desses problemas é, tal como o D-IMager da Panasonic analisado anteriormente, não ter uma grande base comunitária, nem um grande suporte por parte da própria fabricante, que apesar de ser claramente superior á da Panasonic, mesmo assim não é suficiente para convencer a ser o dispositivo escolhido para este trabalho [Ipisoft, 2013].

[41] Imagem disponível em http://robosavvy.com/RoboSavvyPages/ASUS/01.jpg
[42] Preço consultado através do site KuantoKusta.pt

2.4 Tipos de utilização dos sensores do movimento

Os sensores de movimento são utilizados nas mais diversas aplicações, como os jogos, a medicina, os painéis de publicidade, lojas interativas, espetáculos, entre outros. Serão referidos de seguida alguns exemplos de utilizações típicas dos sensores nos mais diversos ambientes.

2.4.1 Em Jogos

Ao longo dos anos foram vários os sensores que foram criados com o intuito de dar ao utilizador uma experiência mais imersiva de interação. Tal como abordado anteriormente, ao longo da história dos sensores a sua utilização primária foi sobretudo no ramo dos vídeo jogos.

Uma vasta lista de jogos fizeram uso de sensores de movimento, abaixo são referidos alguns exemplos desse tipo de jogos:

2.4.1.1 O Dance Central

O Dance Central é um jogo que saiu para a consola Xbox 360 e que recorre ao sensor de movimento Kinect. Neste jogo os movimentos do utilizador refletem-se em movimentos do avatar. O objetivo é que o utilizador tente executar os movimentos de dança pedidos e apresentados no ecrã da forma o mais correta possível, tal como pode ser visto na figura 35, onde ambos os jogadores executaram os movimentos corretamente. [Harmonix, 2015a].

Figura 35 - Avatares em jogo no Dance Central[43]

[43] Imagem disponível em http://2.bp.blogspot.com/-jzhxx3sdXpw/U_jrgj6AObI/AAAAAAAAMzo/wJsw8mFFpfc/s1600/2515703-7702568408-dance.png

2.4.1.2 O EyeToy: AntiGrav

Este jogo saiu para a consola PlayStation 2 fazendo utilização do sensor de movimento Eye Toy da Sony. Neste jogo o utilizador controla um avatar que percorre um circuito com obstáculos numa hoverboard (fig. 36). O utilizador tem de realizar uma serie de movimentos, como saltar ou esquivar-se, para se desviar e evitar os obstáculos do jogo [Harmonix, 2015b].

Figura 36 - Nível do jogo EyeToy: AntiGrav[44]

2.4.1.3 O Grand Slam Tennis

Este jogo é um título pertencente á consola Wii que faz uso do controlo WiiMote de forma a proporcionar uma experiência de interação superior ao utilizador. Neste jogo o utilizador joga ténis (fig. 37) recorrendo ao WiiMote e recriando os movimentos tal como se estivesse a jogar ténis na realidade [EA, 2015].

Figura 37 - Partida de Ténis no Grand Slam Tennis[45]

[44] Imagem disponível em http://www.todojuegos.com/modules/coppermine/albums/PS2/Antigrav/antigrav_00.jpg
[45] Imagem disponível em http://static.giantbomb.com/uploads/original/0/27/1022557-ea_sports_grand_slam_tennis___wimbledon.jpg

2.4.2 Na área da saúde

Com o aparecimento dos sensores de movimento para as consolas de jogos não se demorou muito tempo para perceber que esses mesmos sensores teriam imensas vantagens noutros campos para além dos videojogos. Assim depressa se entendeu o uso dos sensores como por exemplo o *Kinect* da *Microsoft* a outras áreas e aplicados a outros cenários. No mundo da medicina, é usado com o intuito de facilitar e agilizar os mais diversos processos de intervenção médica. Eis alguns exemplos de aplicações dos sensores de movimento na área da saúde.

2.4.2.1 Manipulação e observação de exames durante cirurgia

Alguns hospitais começaram a realizar cirurgias aos seus pacientes recorrendo ao sensor de movimentos *Kinect*. O objetivo seria de substituir o procedimento tradicional de ver os exames de raio x e afins nas máquinas de painel branco, que se encontram nas salas de cirurgia, por um sistema digital em que esses mesmos exames seriam vistos num ecrã e poderiam ser manipulados (rodar, fazer zoom e movimentar) recorrendo apenas a gestos no ar que seriam depois interpretados pelo *Kinect* (fig. 38), isto tudo para evitar que o cirurgião tenha necessidade de dizer ao assistente para mudar ou manipular o exame atualmente a ser analisado, ou caso não tenha assistente, ser ele mesmo a ter que ir manipular o exame, sendo que neste caso após a manipulação o cirurgião teria de desinfetar as mãos para poder continuar a operar o paciente. Com o *Kinect* todos estes problemas de logística ficam resolvidos e como tal permite também reduzir ao tempo da operação, o que é positivo para todos [Taborda, 2012], [Silva, 2011], [Redação Olhar Digital, 2012], [Crounse, 2014], [Gantenbein, 2012].

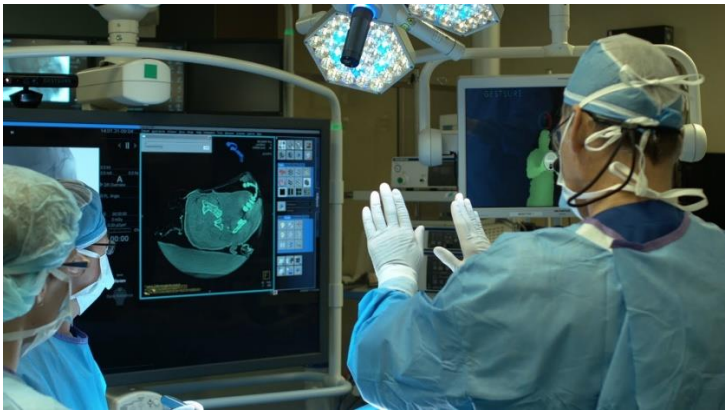

Figura 38 - Utilização do Kinect na sala de cirurgias para manipulação de exames[46]

[46] Imagem disponível em http://www.voletic.com/wp-content/uploads/2014/07/maxresdefault1.jpg

2.4.2.2 Exames em realidade aumentada

A *Microsoft Research Cambridge* criou um protótipo que recorre ao sensor de movimentos *Kinect* para mostrar num ecrã os dados de uma ressonância magnética, com realidade aumentada, ou seja, o *Kinect* filma por exemplo a cabeça do paciente e em seguida os resultados dessa ressonância são sobrepostos sobre a cabeça do paciente no monitor (fig. 39). Assim o cirurgião pode em tempo real e de forma tridimensional examinar os resultados do exame como se estivesse dentro da cabeça do paciente.

Esta técnica ajuda principalmente numa questão de localização e perspetiva sobre o incidente reportado no exame, já que normalmente os resultados desses exames são mostrados em 2D, o que torna mais difícil localizar o incidente reportado no corpo do paciente que é 3D. O exame torna-se assim em 3D e é sobreposto ao corpo do paciente que é também um objeto 3D, ou seja, a localização de incidentes e afins é muito mais facilitada e simplificada facilitando a função do cirurgião [Kleina, 2013].

Figura 39 - O protótipo da realidade aumentada desenvolvido pela Microsoft Research Cambridge[47]

2.4.2.3 Sistema de ajuda em recuperação de AVC dos pacientes

A *Microsoft Research* usou também o *Kinect* para desenvolver um sistema que pode ajudar pacientes em recuperação de um AVC (acidente vascular cerebral) em casos de ataque cardíaco (fig. 40). Este sistema consiste de três exercícios que os pacientes têm de realizar em frente ao *Kinect* para que este possa avaliar se existem certos sinais suspeitos que possam levar a crer que o paciente ainda não se encontra totalmente recuperado. Segundo Jacob,

[47] Imagem baseada no conteúdo disponibilizado em http://www.tecmundo.com.br/medicina/37466-kinect-pode-ver-dentro-de-sua-cabeca-e-ajudar-na-hora-da-cirurgia-video-.htm

com este sistema o acompanhamento destes pacientes após o incidente fica mais facilitado e flexível [Siegal, 2014].

Figura 40 - Sistema desenvolvido pela Microsoft Research para ajudar a recuperação de pacientes que sofreram de AVC[48]

2.4.3 Em painéis de publicidade

Além das aplicações em videojogos e medicina, os sensores de movimento tiveram também a sua expansão para a área publicitária, mais concretamente em anúncios interativos ao público.

Ao serem usados sensores de movimento nos painéis publicitários, automaticamente os anúncios tornam-se mais interativos e apelativos o que por sua vez atrai mais o publico e chama a sua atenção para o conteúdo que está a ser publicitado. De seguida serão mostrados alguns exemplos de utilização dos sensores de movimento na área publicitaria.

2.4.3.1 O painel publicitário da PrimeCredit

A *JCDecaux* e a *PrimeCredit* criaram um painel publicitário interativo que recorre ao sensor de movimentos *Kinect* para que numa estação de metro de Hong Kong seja dada a oportunidade dos passageiros que passam em frente ao painel de jogarem um jogo interativo que lhes permite ganhar prémios. A interação com o jogo é feita através de movimentos das mãos que são depois captados pelo sensor *Kinect* e que por sua vez se traduzem em movimentos no jogo (fig. 41) [JCDecaux, 2013].

[48] Imagem disponível em http://l.yimg.com/bt/api/res/1.2/gTXyVHy05YJT06ZoB.tx5Q--/YXBwaWQ9eW5ld3M7Zmk9ZmlsbDtoPTM0MztpbD1wbGFuZTtweW9mZj0wO3E9NzU7dz02MDk-/http://media.zenfs.com/en_US/News/BGR_News/stroke-recovery-with-kinect.jpg

Figura 41 - Painel publicitário interativo em Hong Kong[49]

2.4.3.2 O painel publicitário da Ballantines

A *Ballantines* em associação com outras empresas decidiu criar uma serie de painéis publicitários interativos no aeroporto de *Schiphol* em Amesterdão. Nestes painéis os passageiros que passam são captados pelo sensor *Kinect* que depois passa a informação para um algoritmo que cria uma versão "artística" dos passageiros (fig. 42). O passageiro pode depois mover-se para provocar mais alterações á sua versão "artística" criada pelo algoritmo, isto porque com cada movimento do utilizador a sua representação no ecrã sofre alterações. O utilizador pode então depois partilhar a sua versão artística nas redes sociais Facebook, Twitter e Pinterest [DisplayInsight, 2013].

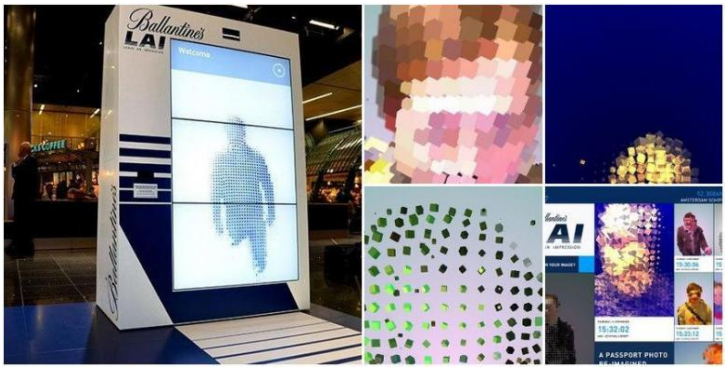

Figura 42 - Painel publicitário interativo em Amesterdão[50]

2.4.3.3 O painel publicitário criado pela Nsquared

A *Nsquared* criou um painel publicitário para uma loja num centro comercial Australiano. Esse painel usou o sensor de movimentos *Kinect* para dar aos clientes que iam passando uma experiencia interativa. Basicamente os clientes ao passarem pelo painel veriam as suas

[49] Imagem baseada no conteúdo disponível em http://www.jcdecaux-transport.com.hk/video/primecredit-showed-its-commitment-via-multimedia-panel-with-interactive-gam/123/P40

[50] Imagem disponível em http://www.displayinsight.com/wp-content/uploads/2012/11/Ballantine_install.jpg

sombras representadas numa serie de ecrãs (fig. 43), com isto a atenção dos clientes seria despertada o que os levaria a aproximarem-se do painel, ao fazerem isso o painel publicitário permitir-lhes-ia interagir com uma espécie de latas de tinta que depois faria revelar uma serie de desenhos artísticos que os conduziriam ao interior da loja publicitada [Al-Riyami, 2015].

Figura 43 - Painel publicitário da Nsquared num centro comercial da Austrália[51]

2.4.4 Conclusão

Já existem atualmente imensas implementações e soluções a utilizar a tecnologia do *Kinect*. Para este trabalho, e depois de uma grande e exaustiva pesquisa, foi determinado utilizar esta mesma tecnologia para ajudar artistas e outras entidades da área dos espetáculos a oferecerem um acompanhamento de efeitos especiais mais dinâmico e interativo às suas atuações. Isto tornará todo o espetáculo mais envolvente para a audiência em geral, e consequentemente melhorará a sua experiência interativa. A escolha do *Kinect* como tecnologia neste trabalho, ao invés de outras similares, recai no facto de existir uma maior base comunitária e de suporte tanto por parte da própria fabricante (*Microsoft*) como dessa mesma comunidade (programadores), que permite uma maior facilidade no desenvolvimento nesta plataforma, algo que não seria tangível nas plataformas rivais, pela falta dos recursos acima mencionados. Portanto a escolha da plataforma não foi muito difícil, apesar de se ter estudado todas as alternativas disponíveis no mercado.

2.5 Os sensores, a dança e os espetáculos ao vivo

Com o aparecimento dos sensores de movimento não se demorou muito até se descobrir o potencial que estes teriam na área dos espetáculos.

[51] Imagem disponível em http://www.winbeta.org/sites/default/files/news/Wall.JPG

O objetivo da utilização destes sensores no mundo dos espetáculos seria o de dar uma maior interação a esses mesmos e assim descobrir outras formas criativas de dar á audiência um espetáculo ainda mais cativante. De seguida vão ser referidos alguns exemplos de utilizações destes sensores em espetáculos.

2.5.1 Exemplos de utilização de sensores de movimento em espetáculos

2.5.1.1 O as·phyx·i·a

O *as·phyx·i·a* é um espetáculo de dança 3D, em que os movimentos e a dança do artista são capturados previamente com um sensor *Kinect*. Depois essa informação gerada pelo *Kinect* é transferida para um *software* de modelação 3D, no qual é criado um ambiente onde uma representação virtual do artista é gerada e depois renderizada (fig. 44). Posteriormente é acrescentado e sincronizado o áudio utilizado para acompanhar o espetáculo e tem-se como resultado uma espécie de videoclipe 3D [Rodrigues, 2015].

Figura 44 - Espetáculo virtual as·phyx·i·a[52]

2.5.1.2 O Connect

O *Connect* é um projeto desenvolvido pela *Microsoft Research* em que torna possível criar uma dança que é afetada pela interação dos espectadores. Mais concretamente existe um sensor *Kinect* que captura os movimentos de um dos membros da audiência e conforme os seus movimentos é escolhida a música que os artistas em palco têm de dançar (fig. 45). A escolha da música com base nos movimentos de um membro da audiência é feita recorrendo a um algoritmo que faz essa análise e aconselhamento [Mentis, 2012].

[52] Imagem disponível em http://edrodrigues.com.br/wp-content/uploads/2015/03/asphyxia-3.jpg

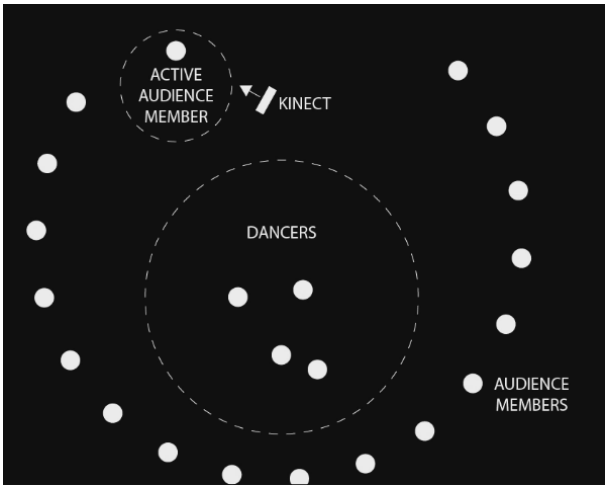

Figura 45 - Esquema do funcionamento da dança com o Connect[53]

2.5.1.3 O Kinect Illusion

O *Kinect Illusion* é um espetáculo de dança interativo, em que os movimentos dos dançarinos interagem com uma série de diversos efeitos em tempo real, recorrendo para tal ao sensor de movimentos *Kinect*. Os efeitos são maioritariamente baseados em partículas e fluidos para conferir um grande dinamismo e interação ao espetáculo (fig. 46) [Yoon, 2011].

Figura 46 - Espetáculo de dança Kinect Illusion[54]

[53] Imagem baseada no conteúdo disponível em http://research.microsoft.com/apps/video/default.aspx?id=161884

[54] Imagem baseada no conteúdo disponível em https://www.youtube.com/watch?v=5Qjl4BQw8qw&feature=youtu.be

2.5.1.4 O flow no. 1

O *flow no. 1* é um espetáculo de dança interativo que recorre ao sensor *Kinect* para proporcionar interação entre os dançarinos e os efeitos gráficos. Os efeitos deste espetáculo são baseados em movimentos de fluidos (fig. 47) e segundo o seu criador, Christian Mio Loclair, este espetáculo foi criado com a intenção de investigar os benefícios da sua aplicação em danças urbanas como por exemplo o *hip-hop* [Loclair, 2013].

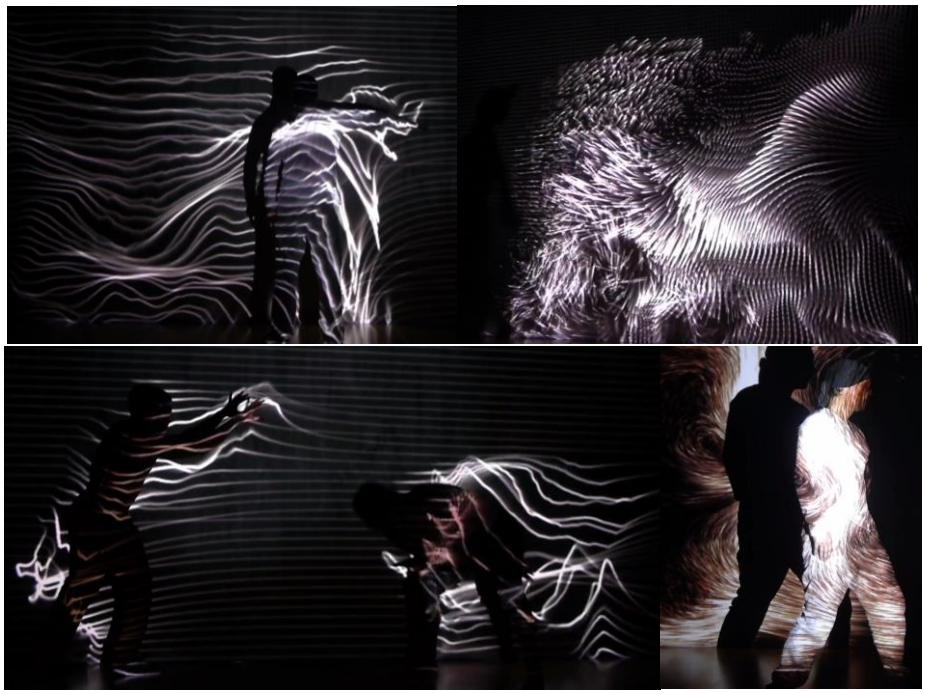

Figura 47 - Espetáculo de dança flow no. 1[55]

2.5.2 A interação proporcionada pelos Sensores de Movimento

Os sensores de movimento permitem ampliar o nível de interação de um determinado espetáculo. Isto é conseguido pelo facto dos sensores permitirem criar efeitos que interagem em tempo real com os artistas ou até mesmo com o próprio público. Com isto consegue-se criar um espetáculo que apela o público de uma forma mais intensa e que consegue igualar ou até mesmo superar o fator apelativo gerado pelos efeitos mais tradicionais.

Posto isto, sem dúvida alguma que os sensores de movimento começarão a ser utilizados com uma maior frequência nos espetáculos da atualidade prevendo-se uma maior utilização no futuro, pois permitem acrescentar uma nova dimensão áquilo que já é atualmente proporcionado por estes.

[55] Imagem baseada no conteúdo disponibilizado em http://princemio.net/portfolio/flow-1-kinect-projector-dance/

Capítulo 3 – O protótipo “MoveU”

“A universalidade dos conhecimentos

é necessária para se ser superior em qualquer parte.”

Madame de Stael

Neste capítulo são analisados os diferentes tipos de softwares de desenvolvimento do protótipo ou suporte que são utilizados no âmbito deste trabalho. Além disto também será detalhado e aprofundado o processo de construção do protótipo e da Aplicação Controladora.

Numa primeira parte será feita uma breve análise a cada software, mencionando os seus pontos mais importantes, vantagens e desvantagens. São apresentadas as escolhas dos softwares e são analisadas as razões pelas quais determinado software foi escolhido.

Numa segunda parte é apresentada e explicada a interface com o utilizador do protótipo e da própria Aplicação Controladora. Também é apresentado o processo de desenvolvimento do protótipo bem como os passos que foram necessários para a sua construção. Por último é também explicado o desenvolvimento e construção da Aplicação Controladora que permite o controlo remoto ou local do protótipo.

Por fim são explicados e detalhados alguns testes que foram realizados tanto ao protótipo MoveU como também á Aplicação Controladora.

3.1 Análise de software para desenvolvimento

Para a realização deste trabalho não será apenas necessário a utilização de componentes de *hardware* (sensores), também será necessária a utilização de *software* que permita tirar partido destes sensores. Tal como foi mencionado no capítulo anterior, cada um dos sensores estudados fazia-se acompanhar de um SDK para que pudessem ser criadas aplicações para esses mesmos sensores. Neste capítulo será feita uma análise a alguns desses SDK e também outros tipos de *software* necessários para o desenvolvimento do protótipo que serve de suporte ao presente estudo.

3.1.1 O Microsoft Kinect SDK

A *Microsoft* decidiu criar um SDK que permitisse programadores criarem as suas próprias aplicações recorrendo ao *Kinect*. Desde a sua primeira versão, já foram lançadas diversas evoluções deste SDK, sendo que a versão estável mais recente é a 1.8 [MSDN, 2014b], mas já existe uma versão experimental, a 2.0 [Microsoft, 2014b].

Com este SDK é possível criar um vasto leque de aplicações, sendo que alguns dos exemplos possíveis são, reconhecer pessoas em movimento utilizando para isso o rastreamento do esqueleto, determinar a distância entre um objeto e o sensor recorrendo á informação de profundidade por ele gerada, capturar áudio recorrendo a tecnologia de remoção de ruído e cancelamento de eco ou até determinar o ponto de origem do som e criar aplicações que reconheçam voz e façam processamento gramatical [MSDN, 2014b].

Este SDK trás incluídos uma série de componentes importantes ou opcionais mas que melhoram a experiência de utilização do SDK para o *Kinect*, desses componentes destacam-se os *drivers* e documentação técnica do *Kinect*, uma série de APIs para programação de aplicações e respetiva documentação, exemplos que demostram boas práticas na utilização do *Kinect*, vários exemplos de código que dividem cada um dos exemplos em tarefas para o utilizador [MSDN, 2014b].

A principal vantagem na utilização deste SDK é o facto de este oferecer uma total integração de *drivers*, APIs e ter uma boa documentação, o que torna o desenvolvimento mais fácil e com um menor esforço em termos de integração das diferentes componentes necessárias para o desenvolvimento de aplicações recorrendo ao *Kinect*. Outro aspeto é que sendo um SDK desenvolvido pela própria marca do sensor leva a que exista, pelo menos teoricamente, uma maior estabilidade e fiabilidade no mesmo ao contrário de outros que sejam externos á marca. É também importante salientar que este SDK pode ser utilizado em conjunto com outras ferramentas da *Microsoft*, nomeadamente o *Microsoft Visual Studio* ou o *Microsoft XNA Game Studio.* O *XNA Game Studio* é analisado e apresentado em seguida.

3.1.2 O Microsoft XNA Game Studio

A *Microsoft* criou um conjunto de ferramentas que permitisse aos programadores facilmente criar videojogos e outros tipos de conteúdos 3D. Para este fim a *Microsoft* criou a ferramenta denominada de "*XNA Game Studio"*. Este *software* é baseado na *Framework .NET* da própria *Microsoft* e pode ser executado nas plataformas *Windows*, *Windows Phone* e *Xbox* [MSDN, 2015f]. Esta ferramenta foi criada principalmente para que os programadores pudessem

desenvolver uma variedade de jogos que poderiam facilmente ser executados nas mais diversas plataformas da *Microsoft* [MSDN, 2015f].

Pode-se assim dizer que o *XNA Game Studio* é análogo da API *DirectX* da *Microsoft*, sendo que nesta última o programador tem de construir todo o motor de jogo, e todos os pilares de base necessários para a execução do jogo, para além de também ter que desenvolver a própria lógica do jogo em si [MSDN, 2009g].

Portanto, estas duas plataformas têm dois públicos-alvo distintos, sendo que tal como referido acima o *XNA Game Studio* é direcionado para programadores que queiram criar jogos que facilmente funcionem em diversas plataformas *Microsoft*, sem terem que se preocupar com as partes que dizem respeito ao motor de jogo ou integração com o *hardware* gráfico [MSDN, 2009g]. O *DirectX* é direcionado a programadores que pretendam ter o máximo de controlo sobre o seu desenvolvimento e sobre o seu código, e que pretendam ter a potencialidade de extrair o máximo desempenho gráfico possível, ou seja, para que possam desenvolver jogos mais exigentes [MSDN, 2009g].

O *XNA Game Studio* oferece o seu próprio ambiente de desenvolvimento, sendo que este é baseado no ambiente de desenvolvimento *Microsoft Visual Studio*, desta forma não é necessário a um programador que queira utilizar o *XNA Game Studio* obter o *Visual Studio* para poder desenvolver os seus jogos, tornando assim o *XNA Game Studio* uma ferramenta bastante versátil, e simples de utilizar [MSDN, 2009g].

É de notar também que esta é uma ferramenta gratuita da *Microsoft*, sendo que qualquer pessoa a pode descarregar para tirar proveito dela [MSDN, 2009g]. Foram lançadas várias versões ao público desta ferramenta, sendo a versão 4.0 a mais recente, mas a *Microsoft* anunciou a 1 de Abril de 2014 que esta ferramenta iria ser descontinuada, e consequentemente o seu suporte terminado [MSDN, 2015f], [Rose, 2013].

3.1.3 O Unity

O *Unity* tal como o *XNA Game Studio* é um ambiente totalmente integrado de desenvolvimento de jogos criado pela *Unity Technologies* [Unity, 2015q].

O *Unity* funciona sobre *Windows* ou *Mac OSX*, mas permite a criação de jogos para uma grande variedade de plataformas como por exemplo *Windows*, *Linux*, *Mac OSX*, *Windows Phone*, *Blackberry 10*, *Android*, *IOS*, *Adobe Flash*, e a vasta maioria das consolas de jogos da atualidade [Unity, 2015p]. O *Unity* é em si um conjunto de diversos componentes, é um ambiente de desenvolvimento, um motor gráfico e motor de jogo [Unity, 2015r], [Brodkin,

2013]. O *Unity* agrega numa só ferramenta diversos componentes necessários para a criação de jogos, acrescendo ainda o bónus de ser possível desenvolver para um vasto leque de plataformas, portanto isto faz do *Unity* uma ferramenta extremamente flexível e também simples de utilizar [Unity, 2015r], [Unity, 2015p].

É de notar que a componente de programação (*scripting*) do *Unity* é baseada no *Mono*, sendo que o *Mono* é a implementação *open source* do *.NET Framework* da *Microsoft* [Mono, 2015a], [Mono, 2015b]. Quanto às linguagens suportadas pelo *Unity* destacam-se o *C#*, *UnityScript* (que é semelhante ao *JavaScript*) e *Boo*, sendo que o programador é livre de escolher com qual linguagem pretende trabalhar no *Unity*, atestando mais uma vez a grande flexibilidade do *Unity* [Unity, 2015r].

3.1.4 O Zigfu ZDK

O *Zigfu* ZDK é um *plugin* que pode ser instalado em conjunto com o *Unity* para que seja possível criar jogos ou aplicações 3D que façam uso de sensores de movimento. Este *plugin* é o que permite fazer a ponte entre o *Kinect* e o ambiente de desenvolvimento *Unity*, sendo que o *Zigfu* permite utilizar tanto o *Microsoft Kinect SDK* como o *OpenNI NITE* como forma de comunicação com o *Kinect* [Zigfu, 2014].

Tal como o *Unity* o *Zigfu* ZDK permite o desenvolvimento de jogos ou aplicações tanto em *Windows* como em *Mac OSX* tornando-o assim num *plugin* multiplataforma [Zigfu, 2014].

É de notar que o *Zigfu* ZDK é um *software* pago, mas que disponibiliza uma versão *trial* que não é restrita em funcionalidades ou em termos de tempo de utilização, a única diferença entre a versão *trial* e a versão paga é que na *trial* aparece uma marca de água com o logótipo *Zigfu* nas aplicações criadas recorrendo a este *plugin* [Zigfu, 2014].

3.1.5 O OpenNI NITE

O *OpenNI NITE* é um *middleware* desenvolvido pela *PrimeSense* para uso com os seus sensores [Mitchell, 2010]. Este *middleware* vem facilitar o acesso de aplicações às funcionalidades dos sensores, e sendo que este *software* é de natureza aberta, ou seja *open source*, ele é disponibilizado a todos e de forma livre e gratuita [Armstrong, 2014].

É de notar que dois dos sensores analisados no capitulo anterior fazem uso de tecnologias da *PrimeSense*, sendo estes sensores o *Kinect* e o *Xtion Pro Live*. Portanto no caso destes sensores é possível utilizar o *middleware NITE* como forma de utilizar estes dispositivos.

É também importante referir que a *PrimeSense* foi adquirida pela *Apple*, e devido a esta razão todo o *middleware* que era *open source* foi descontinuado e a página de onde era possível transferir os *softwares* (openni.org) foi encerrada (no dia 23 de Abril de 2014) [Armstrong, 2014]. No entanto cópias do *software* encontram-se nos mais diversos sites da *web*, sendo que desta forma ainda é possível descarrega-lo.

Apesar desta aquisição por parte da *Apple* e da descontinuação do *middleware*, existem uma série de entidades (como a *Structure*) que vão continuar o desenvolvimento do *OpenNI*, assegurando o suporte continuado dos sensores que usam tecnologia da *PrimeSense* [Structure, 2014].

3.1.6 A justificação da escolha

Foram vários os *softwares* que foram analisados por forma a escolher os melhores e mais adequados para a realização do presente estudo.

O motor gráfico escolhido para o âmbito deste trabalho foi o *Unity* com os respetivos *softwares* de desenvolvimento. O *Unity* foi escolhido pois apresenta um elevado nível de flexibilidade e conjuga numa única ferramenta uma série de componentes necessários para o desenvolvimento deste trabalho.

O *Microsoft XNA Studio* foi um excelente candidato, mas não foi escolhido pela simples razão da *Microsoft* ter decidido descontinuar o *software* e acabar com o seu suporte, portanto a escolha do *Unity* foi pacífica, pois além das suas grandes qualidades o *software* concorrente teve em sua desvantagem o término do seu suporte.

Depois de ter sido escolhido o *Unity* como plataforma de desenvolvimento, teria também de ser escolhido um *software* que permitisse fazer a ponte entre o *Unity* e o *Kinect*. Após alguma pesquisa foi encontrado o *Zigfu* ZDK, este *plugin* permite interligar as duas plataformas, e fá-lo de uma forma bastante simples e eficaz, sem serem necessárias grandes configurações.

Após a escolha do *Zigfu* era necessário escolher o *middleware* que disponibilizaria a comunicação do *Kinect* com o próprio *Zigfu*. Aquando da leitura da documentação do *Zigfu* foi concluído que este suporta o *middleware OpenNI NITE* e o *Kinect* SDK, portanto um dos dois teria de ser escolhido. No final a escolha recaiu no *Kinect* SDK, pois ao contrário do *OpenNI*, este ainda tem suporte total por parte da *Microsoft*, além de que este é o *software* que foi feito de raiz pela própria marca (*Microsoft*) para suportar o *Kinect*.

3.2 O modelo conceptual do MoveU

Antes de avançarmos para a conceção da solução que pretendíamos desenvolver decidimos esboçar e desenhar o modelo conceptual que o suportaria. Deste modo seria possível não só identificar os tipos de utilizadores mas também as necessidades de equipamento relacionado com o funcionamento do protótipo e prever aspetos como:

- Interação Dançarino/Protótipo;
- Projeção sobre o cenário;
- Localização do projetor;
- Localização do dançarino.

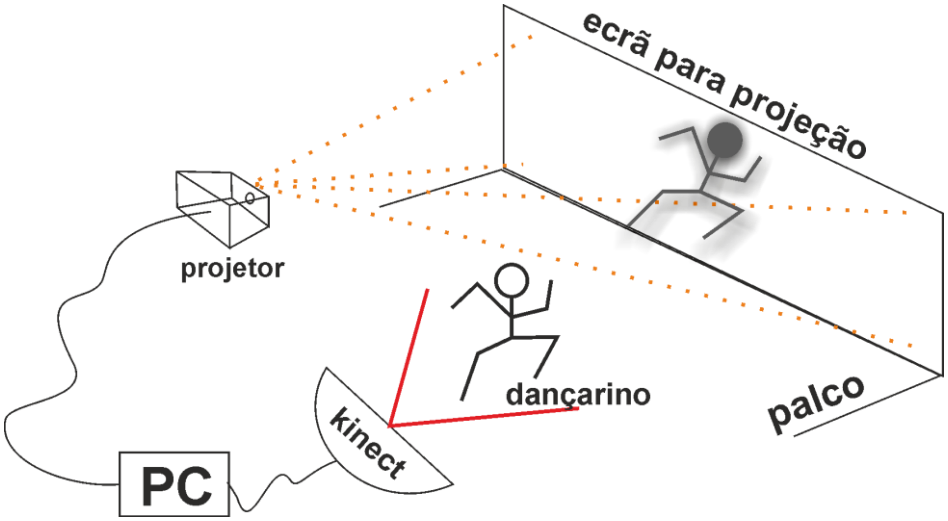

Figura 48 - Modelo conceptual do protótipo[56]

Como se pode ver na figura anterior, o *MoveU* teria as seguintes necessidades mínimas de *hardware*:

- Um computador;
- Um sensor (por exemplo Kinect);
- Um projetor.

O dançarino deverá ficar numa zona frontal ao sensor e a aplicação a desenvolver deverá capturar os seus movimentos que por sua vez serão projetados no palco.

Como se pode ver na figura seguinte, o objetivo é que se possam visualizar efeitos especiais no cenário que são provocados mediante a interação do dançarino com o sensor.

[56] Imagem da autoria do autor do presente trabalho

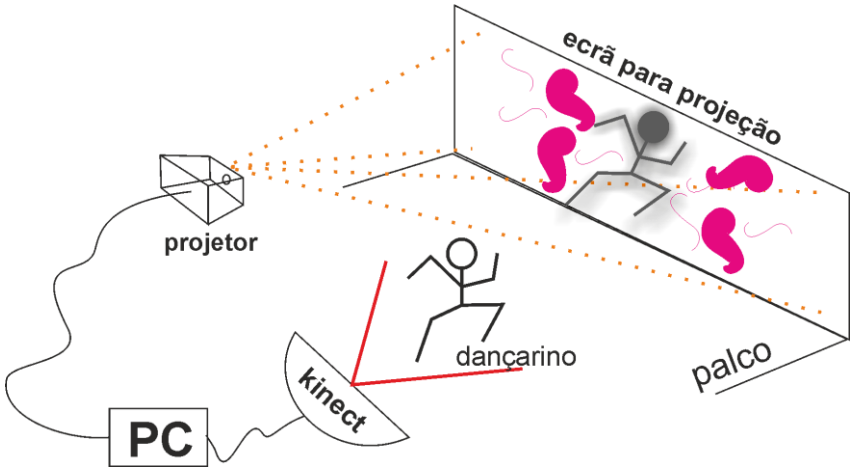

Figura 49 - Elementos interativos acrescentados ao cenário[57]

Na figura anterior pode verificar-se que este modelo conceptual deverá proporcionar a adição de efeitos especiais ao cenário mediante a atuação e performance do dançarino.

Nesta fase foi ainda possível perceber que seria necessário um outro tipo de utilizador para controlar a solução idealizada.

Neste sentido o "utilizador controlador" poderá assumir um papel preponderante na mudança dos efeitos a acoplar ao sistema com o intuito de tornar o espetáculo ainda mais interessante e agradável para os espectadores.

3.3 A interface do protótipo MoveU

O protótipo deste trabalho foi denominado *MoveU* (basicamente a tradução para inglês do termo "move-te") e está dividido em duas partes distintas, sendo que a primeira corresponde ao ambiente 3D criado pelo protótipo e que será projetado numa tela atrás do artista que está a atuar, e uma segunda parte que corresponde a uma aplicação que é executada noutra máquina (ou na mesma) e que controla todos os efeitos do ambiente, sendo que a comunicação entre as duas partes é feita através de *Web Services*. Esta comunicação é remota, dando uma maior flexibilidade na escolha das máquinas que irão executar o ambiente 3D selecionado pelo utilizador controlador na aplicação controladora.

Os utilizadores desta aplicação terão de conectar o sensor *Kinect* (selecionado para integrar o protótipo) e um projetor ao PC que irá executar o ambiente 3D. Como referido, a aplicação controladora pode ser executada num segundo PC (embora tal não seja obrigatório pois tanto o ambiente 3D como a aplicação controladora podem funcionar no mesmo PC), após

[57] Imagem da autoria do autor do presente trabalho

isso o artista terá de se posicionar em frente ao sensor *Kinect* e começar a desempenhar a sua atuação, sendo, como planeado no modelo conceptual que deverá ser outra pessoa a operar a mudança dos diferentes efeitos no PC onde se encontra a aplicação controladora.

A aplicação controladora é nesta fase muito simples e tem como objetivo tornar a alteração de efeitos de cenário possível e, como se pode ver na figura seguinte, permite ao utilizador controlador escolher alguns efeitos, podendo, inclusivamente dentro do próprio efeito alterar funcionalidades.

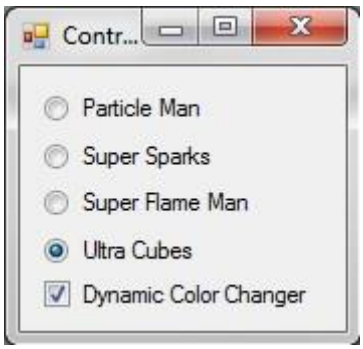

Figura 50 - Aplicação Controladora

Em seguida serão referidos cada um desses efeitos.

Denominou-se o primeiro efeito de *Particle Man*, basicamente trata-se de um efeito que captura as formas do corpo do utilizador e preenche essas formas com partículas animadas dando uma espécie de efeito "*glow"* que deixa um certo rasto quando o utilizador se mexe, como se pode ver na seguinte figura.

Figura 51 - Particle Man

Este efeito pode também ser combinado com a opção *Dynamic Color Changer*, esta opção faz com que o fundo do ambiente 3D passe de uma cor sólida para uma variação dinâmica entre diversas cores, auferindo assim um efeito complementar ao efeito principal, tal como se apresenta na figura 52.

Figura 52 - Particle Man com Dynamic Color Changer

É também de notar que todos os efeitos presentes na aplicação controladora podem ser combinados com o efeito *Dynamic Color Changer*.

Outro efeito desenvolvido e que pode ser escolhido é o *Super Sparks*, este efeito deteta os movimentos do utilizador e substitui o seu corpo por uma espécie de estrelas brilhantes e á medida que o utilizador mexe os seus braços estes deixam um rasto luminoso que proporciona um efeito artístico elegante, tal como se pode comprovar pela figura seguinte.

Figura 53 - Super Sparks

Outro efeito desenvolvido designamo-lo por *Super Flame Man*. Este efeito deteta o corpo do utilizador e representa no ecrã um avatar, desenhado com formas geométricas, mais concretamente cilindros, cubos, esferas e paralelepípedos que representam a estrutura do corpo do dançarino baseado no esqueleto desse utilizador.

Estas formas geométricas têm também uma mudança de cor dinâmica ao estilo do efeito *Dynamic Color Changer*. Além disto acrescenta-se também um efeito do tipo fogo tanto nas mãos como nos pés do esqueleto que representa o utilizador.

Como efeito adicional foram colocadas no fundo uma série de estrelas que deslocam de uma extremidade á outra do ecrã a alta velocidade e com diferentes sentidos de orientação, sendo que alguns conjuntos de estrelas se movimentam da esquerda para a direita e outras da direita para a esquerda. O efeito é ilustrado pela seguinte figura.

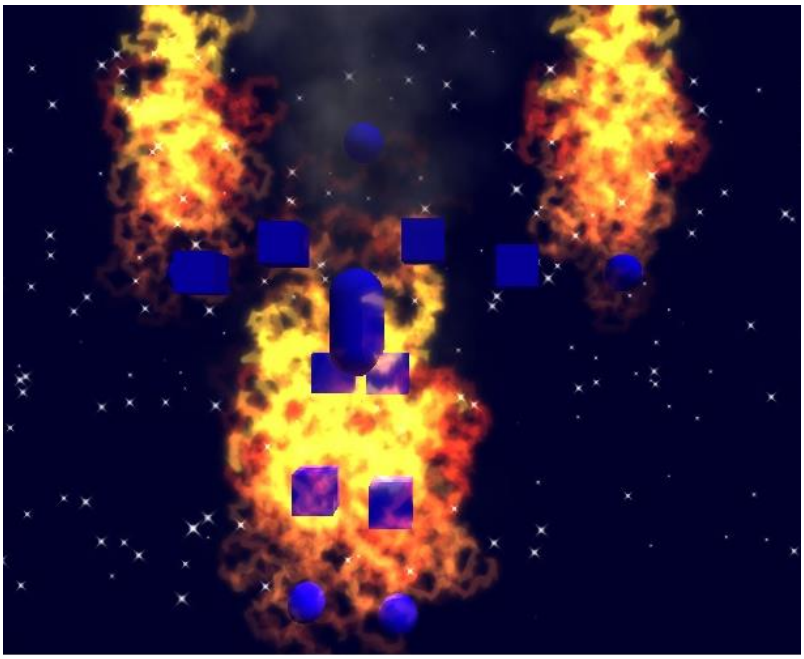

Figura 54 - Super Flame Man

Demos o nome de *Ultra Cubes* ao último efeito desenvolvido. Este efeito é representado por uma serie de cubos no ecrã que o preenchem totalmente. Quando o utilizador muda de posição, quer seja para a frente, para trás, esquerda, direita, cima ou baixo, os cubos acompanham o seu movimento através de uma rotação sobre eles mesmos, ou seja, por exemplo se o utilizador se mover para o lado direito, os cubos também rodam para o lado direito. Quando mais o utilizador for para o lado direito, mais estes rodam.

Além disto os cubos também podem ter (por opção) uma mudança de cor dinâmica e quando o efeito *Ultra Cubes* é selecionado deve ser ativado efeito *Dynamic Color Changer* para que a cor de fundo acompanhe também as cores dinâmicas dos cubos, dando assim um efeito bastante visual e artístico que será projetado no cenário.

Se o utilizador controlador assim o entender também pode desativar a opção *Dynamic Color Changer* e deste modo o fundo irá ficar com uma cor estática. Na figura seguinte apresenta-se uma representação deste efeito em ação.

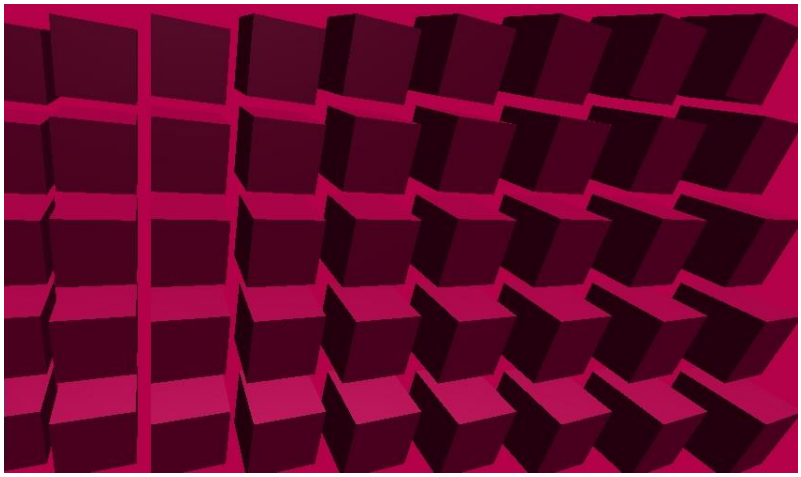

Figura 55 - Ultra Cubes com Dinamic Color Changer

3.4 Desenvolvimento do protótipo

O desenvolvimento do protótipo iniciou-se numa primeira fase realizando alguns testes que permitissem estudar e validar a forma como todos os *softwares* e a plataforma *Kinect* comunicam e funcionam. Foram utilizados os exemplos que são fornecidos com o ZDK *Zigfu* para o *Unity*. Alguns desses exemplos foram executados e após a sua execução foi possível concluir que todos os componentes de *software* e *hardware* estavam operacionais. Assim deu-se início ao desenvolvimento do protótipo no *Unity*.

No *Unity* foi criado um novo projeto e dentro desse projeto foram criados diversos *GameObjects*, sendo que foi idealizado criar um *GameObject* para cada um dos efeitos interativos além de também criarem *GameObjects* auxiliares conforme as necessidades do projeto.

Dentro de cada *GameObject* foram criados uma série de componentes, que incluem as diferentes partes constituintes do efeito interativo uma vez que são necessárias para que este faça o pretendido.

3.4.1 A MainCamera

Em primeiro lugar, antes de criar objetos 3D no projeto, é necessário criar uma câmara. Essa câmara permite ajustar a área do ambiente 3D que está a ser mostrada no ecrã.

No *Unity* para criar uma câmara é necessário criar um *GameObject* do tipo *Camera*, após criar este objeto este já vem automaticamente constituído por diversos outros componentes que são necessários para a câmara.

O primeiro desses componentes é o *Transform.* E que permite mudar a posição na qual o *GameObject* se encontra no ambiente 3D e aplicar diversas operações de transformação, como por exemplo rotações ou escalamentos. Neste caso foi definido que o *GameObject MainCamera* tivesse as seguintes coordenadas de posição: (0, 5, -10) sendo que o primeiro valor corresponde ao eixo do X, o segundo o do Y e o terceiro o do Z. Foi associado o valor 0 para o eixo do X para que a câmara tivesse uma posição centrada no ambiente 3D, o valor de 5 para o eixo do Y teve como objetivo dar um pouco de elevação á câmara para que esta não ficasse "colada ao chão" e assim permitindo uma melhor visualização da cena 3D, por fim o valor de -10 para o eixo do Z tem como objetivo fazer com que a câmara se mantenha recuada para que seja possível ver os objetos 3D com uma certa distância, ou seja, para que a câmara não fique muito em cima dos objetos ficando quase que "colada" a estes.

O segundo componente chama-se *Camera*, e é neste componente que são definidas uma serie de opções específicas da câmara em si. A primeira opção deste componente chama-se *Clear Flags*. Este componente é útil quando existem diversas câmaras e foi definida uma *Skybox* (objeto que rodeia o cenário 3D para simular o céu) no sentido de que quando se muda a visualização de uma câmara para outra, os *buffers* que armazenam diversas informações como a profundidade, e as partes da *Skybox* que estavam a ser mostradas pela câmara que agora não é utilizada sejam limpos, isto, com a finalidade de libertar espaço nos *buffers* e consequentemente aumentar a performance global [Unity, 2014a], [Unity, 2014b].

No caso deste protótipo como só foi criada uma única câmara esta opção torna-se relativamente irrelevante, mas apesar disso foi definida com a opção *Skybox*.

A seguinte opção chama-se *Background* e apenas define, como o nome indica, a cor de fundo do ambiente 3D, caso nenhuma *Skybox* seja criada, neste caso foi escolhido um tom de azul-escuro para que os objetos 3D sejam mais facilmente vistos [Unity, 2014b].

Em seguida temos a função *Culling Mask* que permite selecionar determinados tipos de objetos para poderem serem renderizados, ou seja se nenhum objeto for selecionado todos os objetos 3D serão invisíveis á câmara, neste caso foi selecionado *Everything* para que todos os objetos sejam mostrados [Unity, 2014b].

A função *Projection*, permite definir o tipo de projeção a ser utilizada pela câmara. Tem duas possibilidades, a *Perspective* e a *Orthographic.* Com a opção *Perspective* selecionada os objetos 3D serão mostrados em perspetiva, caso a opção *Ortographic* seja selecionada os objetos serão mostrados uniformemente sem qualquer perspetiva, neste caso foi

selecionado a opção *Perspective* pois foi determinado que seria o ideal para os objetos 3D deste protótipo [Unity, 2014b].

O *Field of View* representa o campo de visão da câmara e é um valor definido em graus que por defeito é de 60 graus tendo sido decidido deixar esta opção com o seu valor por defeito pois oferece um campo de visão satisfatório [Unity, 2014b].

A opção *Clipping Planes*, representa as distâncias mínima e máxima a partir das quais os objetos começam a ser renderizados, ou seja, dentro do intervalo destes valores são renderizados os objetos 3D, fora destes valores não são [Unity, 2014b]. Neste caso foi decidido deixar os valores por defeito, sendo que o valor *Near* é de 0.3 e o valor *Far* é de 1000.

A opção *Viewport Rect* representa o local onde a *viewport* vai ser desenhada no ecrã, existem 4 valores a definir, sendo que X e Y representam as coordenadas a partir das quais a *viewport* irá ser desenhada, e W e H representam respetivamente a largura e altura da *viewport* [Unity, 2014b]. Como se pretendia que a *viewport* ocupasse a posição de todo o ecrã foram definidos os valores de 0 para as coordenadas X e Y e 1 para W e H, isto para que a *viewport* da câmara ocupasse o ecrã na sua totalidade.

A opção *Depth* representa a posição da câmara na fila da ordem pela qual estas são desenhadas. Valores mais baixos representam uma maior prioridade na fila, enquanto que valores mais altos representam uma menor prioridade na fila [Unity, 2014b]. Neste caso e uma vez que só existe uma câmara no projeto foi determinado dar-lhe a máxima prioridade e assim ficou com o valor de -1.

A opção *Rendering Path* define quais são os métodos de renderização a ser utilizados pela câmara, sendo que por defeito está selecionado *Use Player Options* [Unity, 2014b]. Deixamos esta opção com o seu valor original.

A opção *Target Texture* permite criar uma referência para uma *Render Texture*, sendo que se for criada esta referência a câmara deixará de renderizar os objetos 3D [Unity, 2014b]. Colocamos esta opção em branco, ou seja sem referência, pois é irrelevante para o MoveU e além disso é uma opção que só se encontra disponível na versão *Pro* do *Unity* [Unity, 2014b].

A opção *Occlusion Culling* permite desativar a renderização dos objetos que não estão neste momento a ser vistos pela câmara, isto com o intuito claro de poupar recursos e consequentemente aumentar a performance [Unity, 2014f]. Foi deixada ativa, pois é do interesse do *MoveU* aumentar a performance sempre que possível.

Por ultimo existe a opção HDR que permite ativar uma componente de luz avançada em que as luzes terão um aspeto mais realista e mais de acordo com a luz real [Unity, 2014b]. Para o MoveU esta opção é também um pouco irrelevante, pois é apenas utilizado um tipo de luz básico, portanto não existe necessidade de a ativar até porque afeta a performance gráfica.

Os terceiro, quarto e quinto componentes são o GUI *Layer*, *Flare Layer* e *Audio Listener.* Estes três componentes vêm por defeito associados ao *GameObject Camera*, mas para este projeto especifico são irrelevantes.

O GUI *Layer* é um componente que permite renderizar uma interface de utilizador por cima da renderização 3D, como neste projeto não foi utilizada qualquer interface de controlo na própria renderização este componente torna-se irrelevante [Unity, 2014c].

No caso do *Flare Layer* este permite renderizar efeitos do tipo *Lens Flare*, estes efeitos simulam o que acontece quando uma câmara real filma fontes de luz de grande intensidade como o sol [Unity, 2014d]. Mais uma vez este componente foi irrelevante pois não existem fontes de luz intensas neste projeto para que este componente tivesse relevância.

Por fim o componente *Audio Listener* é semelhante a um microfone de uma câmara real, ou seja captura os sons transmitidos por objetos da renderização 3D [Unity, 2014e]. No caso deste projeto como não foram associados quaisquer efeitos sonoros aos objetos 3D este componente torna-se ele também irrelevante, podendo no futuro vir a ser utilizado caso se pretenda capturar o som de maneira a interagir com o sistema.

O sexto componente é o *script ColorChanger*, que foi criado com a intenção em caso do utilizador controlador ative o efeito *Dynamic Color Changer* as cores do fundo mudarem dinamicamente, sendo esse efeito controlado por este *script*.

Este *script* está dividido em dois métodos, o primeiro método é o *Start*, que é executado mal a aplicação seja inicializada, portanto todo o código que se encontre dentro deste é imediatamente executado, sendo um método ideal para inicialização de valores e criação de construtores. O método *Start* contém a inicialização de 3 variáveis que irão ser utilizadas no segundo método deste *script*. Essas variáveis são a cr, a estado e a t. Cr é uma variável do tipo *Camera*, a sua função neste *script* é obter uma instância do componente *camera* de modo a se poder aceder e editar as suas propriedades, sendo que neste caso mais concretamente é pretendido aceder á opção *BackgroundColor* a fim de se alterar a cor de fundo do cenário 3D.

A segunda variável, estado, é uma variável de tipo inteiro em que a sua única função é indicar qual é o segmento de cores que está atualmente a ser percorrido pelo método *Lerp* (este método será explicado mais adiante). Por fim a variável t, que é do tipo *float*, indica qual é a posição temporal da mudança de cores para o método *Lerp*.

No método *Start* a variável Cr é inicializada com uma chamada ao método *GetComponent*, para que a variável tenha uma atribuição com uma instância do componente *Camera*, tal pode ser feito recorrendo á seguinte linha de código:

```
cr = GetComponent <Camera> ()
```

A variável estado tem atribuído o valor de 1 e a t o valor de 0.0f.

O segundo método chama-se *Update*, e é invocado a cada *frame* renderizada pelo motor do *Unity*, sendo o local ideal para a colocação de código que contenha a lógica que faça alterações sobre objetos ou sobre o ambiente 3D.

Neste caso o método *Update* contém uma porção de código que controla a lógica da mudança de cores de fundo dinâmica. Um excerto desse mesmo código é apresentado abaixo.

```
if(t <= 1.0f && estado == 1){
col = Color.Lerp (Color.red, Color.yellow, t);
    t = t + 0.01f;
    if(t > 1.0f){
        estado = 2;
        t = 0.0f;
    }
}
if(t <= 1.0f && estado == 2){
    col = Color.Lerp (Color.yellow, Color.blue, t);
    t = t + 0.01f;
    if(t > 1.0f){
        estado = 3;
        t = 0.0f;
    }
}
if(t <= 1.0f && estado == 3){
    col = Color.Lerp (Color.blue, Color.red, t);
    t = t + 0.01f;
    if(t > 1.0f){
        estado = 1;
        t = 0.0f;
    }
}
```

Código 1 – Algoritmo que permite a mudança de cor dinâmica

Col, é a variável que armazena cores, daí esta ser do tipo *Color*. Como se pode ver no código acima existe uma serie de atribuições á variável Col, estas atribuições são feitas recorrendo ao método *Lerp* da classe *Color*. Este método permite fazer uma interpolação entre duas cores distintas, essas cores são definidas através de parâmetros de entrada, sendo que

correspondem aos dois primeiros, o terceiro parâmetro de entrada corresponde ao momento temporal da mudança de cor, ou seja, supondo que o *Lerp* tem como parâmetro de entrada as cores vermelho e amarelo e o momento temporal de zero, o retorno do método *Lerp* seria a cor vermelha, mas se o momento temporal fosse 1 o resultado já seria amarelo, enquanto que se o momento temporal fosse 0.5 o resultado já seria um tom de alaranjado, ou seja o meio termo entre vermelho e amarelo [Unity, 2014g].

O problema principal na utilização do *Lerp* é que este está apenas limitado a duas cores, e o objetivo do efeito *Dynamic Color Changer* era que este percorreria uma vasta palete de cores e não apenas duas. Para ultrapassar essa limitação teve que se formular um algoritmo que contornasse essa questão, assim foi criada uma variável de estado que permitisse mudar entre vários *Lerps*, ou seja, quando um *Lerp* terminasse o seu ciclo completo a variável estado mudava de valor e o *Lerp* seguinte continuaria a mudança de cor.

Foram criados 3 *Lerp's* distintos, o primeiro começa com as cores vermelho e amarelo, o segundo com amarelo e azul e o terceiro com azul e vermelho, portanto um determinado *Lerp* tem a sua primeira cor igual á segunda cor do *Lerp* anterior, isto para que a transição entre cores seja contínua e correta.

Cada vez que um *Lerp* é executado a variável temporal t é incrementada em 0.1 isto para que na próxima iteração do *Lerp* a transição entre cores seja um pouco mais incrementada. Quando a variável temporal t atinge o valor de 1.0 o ciclo de iteração de um determinado *Lerp* termina, o estado incrementa o seu valor para que a execução do programa passe para o próximo *Lerp* e a variável temporal é reposta para o seu valor 0.0 para que o próximo *Lerp* comece de forma correta, ou seja a partir de 0.0.

Este ciclo repete-se vezes e vezes sem conta sendo que a variável estado do último *Lerp* volta a colocar a execução do programa de volta ao primeiro *Lerp* e a continuar a partir desse ponto.

É de notar que a cada iteração do programa (chamadas ao método *Update*) no final de cada execução de um determinado *Lerp* é executada também uma linha de código que atribuí a cor resultante do *Lerp* á cor de fundo do ambiente 3D, pois caso isto não fosse feito não se iria ver nenhuma mudança de cor no ambiente 3D. A linha de código em questão é a seguinte:

```
cr.backgroundColor = col
```

Esta linha atribuí a variável col que contem o resultado da cor de determinado *Lerp* e faz a sua atribuição á propriedade *backGroundColor* do componente *Camera*. A iteração é definida

por cada *frame* mostrada, ou seja, a velocidade da iteração das cores está diretamente relacionada com a taxa de *frames* por segundo, portanto quantas mais *frames* por segundo, mais rápida e suave será a transição entre cores.

O sétimo e último componente é também um *script* e tem o nome Cliente, este *script* tem como objetivo ser o cliente de *Web Service* e assim comunica com a aplicação controladora onde o utilizador seleciona os efeitos a ativar ou desativar. Portanto este *script* tem a função de comunicar com um *Web Service* e ativar ou desativar efeitos conforme os pedidos recebidos da aplicação controladora.

Este *script* está dividido em dois métodos, sendo que o primeiro se denomina de *Start* e é chamado sempre que a aplicação inicia, portanto neste método foram definidos alguns construtores e inicializações de variáveis.

Start contém uma variável chamada cc que é do tipo *ColorChanger* (ou seja corresponde á classe *ColorChanger* referida anteriormente) e é executado o seguinte método:

```
cc = GetComponent<ColorChanger>()
```

Isto permite obter o componente que contem o *script ColorChanger* para que possa ser possível realizar operações sobre este mesmo.

De igual forma é também atribuída á variável c uma chamada ao *GetComponent*, mas neste caso é com a intenção de obter o componente *Camera*, isto é feito da mesma forma como foi feito acima para a variável cc.

A criação de um construtor para a variável col tem a intenção de armazenar os dados relativos a uma cor. Por último são inicializados os parâmetros individuais da cor da variável col, a cor definida é igual á cor de fundo por defeito do ambiente 3D (azul escuro) e a razão por de trás da criação desta variável será revelada mais á frente.

Antes de se falar no segundo método, convém mencionar algumas declarações globais ao nível da classe Cliente. Para além das habituais declarações globais de objetos e variáveis que são utilizadas pelos métodos desta classe destaca-se o construtor criado para a variável svc.

Este é um construtor da classe *WebServiceClient*, esta classe não é nada mais que um ficheiro de *script* que se encontra na pasta *WebServices* e que permite a comunicação entre o *Web Service* da aplicação controladora e a classe Cliente que é a classe que está a ser neste momento referida. O *script WebServiceClient* foi automaticamente gerado por uma ferramenta chamada de *Web Services Description Language Tool* [MSDN, 2014d], sendo que

esta é uma ferramenta que vem incluída com o ambiente de desenvolvimento *Visual Studio* e que permite de forma bastante fácil e rápida gerar o *script* WSDL relativo a um *Web Service*.

O construtor para a variável svc tem também que fornecer um *Endpoint Andress*, ou seja tem de fornecer o endereço pelo qual o *Web Service* recebe os pedidos dos clientes, sendo assim o construtor foi criado da seguinte forma:

```
WebServiceCliente svc = new WebServiceClient(new BasicHttpBinding(), new
EndPointAdress(http://localhost:7777/server)
```

Neste caso o endereço foi definido para *local host*, ou seja o serviço encontra-se na própria máquina onde a aplicação em *Unity* está a ser executada, mas este endereço pode facilmente ser modificado para contemplar máquinas externas àquela onde a aplicação *Unity* está a ser executada, isto como é claro tira partido do fundamento básico dos *Web Services* que é a comunicação entre máquinas distintas.

Voltando agora aos métodos do *script* Cliente, o segundo método, que se chama de *Update*, é executado cada vez que uma *frame* do ambiente 3D é renderizada, portanto foi decidido colocar neste método toda a logica da interação do *WebService* com os objetos do ambiente 3D.

Foram feitas duas chamadas a serviços do *Web Service*, a primeira chamada:

```
svc.getEstado()
```

Esta pretende obter qual é o efeito atualmente selecionado na aplicação controladora, esta chamada ao serviço retorna um valor do tipo inteiro que é atribuído á variável estado, sendo que o número corresponde a um determinado efeito.

A segunda chamada:

```
svc.getColorChanger()
```

Esta pretende obter o estado do efeito *Dynamic Color Changer*, sendo que neste caso é retornado um valor do tipo booleano que é atribuído á variável *colorChanger*.

Um valor de *true* indica que o efeito deve de estar ligado e um valor de *false* indica que deve de estar desligado. Estas duas chamadas ao serviço existem em separado pois o efeito *Dynamic Color Changer* pode ser ligado ao mesmo tempo que qualquer outro efeito.

Após estas duas chamadas, foram criadas uma série de estruturas condicionais que avaliam as duas variáveis que indicam o estado dos efeitos e que após essa avaliação ativam e

desativam os respetivos efeitos de acordo com o estado dessas variáveis. O excerto de código correspondente a essas estruturas condicionais pode ser visto de seguida.

```
if (estado == 1) {
    UltraCubes.SetActive (false);
    SuperFlameMan.SetActive (false);
    SuperSparks.SetActive (false);
    ParticleMan.SetActive (true);
}
if (estado == 2) {
    UltraCubes.SetActive (false);
    SuperFlameMan.SetActive (false);
    SuperSparks.SetActive (true);
    ParticleMan.SetActive (false);
}
if (estado == 3) {
    UltraCubes.SetActive (false);
    SuperFlameMan.SetActive (true);
    SuperSparks.SetActive (false);
    ParticleMan.SetActive (false);
}
if (estado == 4) {
    UltraCubes.SetActive (true);
    SuperFlameMan.SetActive (false);
    SuperSparks.SetActive (false);
    ParticleMan.SetActive (false);
}
if (colorChanger == true) {
    cc.enabled = true;
} else {
    cc.enabled = false;
    c.backgroundColor = col;
}
```

Código 2 – Algoritmo que permite a mudança dos efeitos consoante as seleções feitas na aplicação controladora

Para realizar a mudança de efeitos é necessário ativar e desativar os objetos do ambiente 3D correspondentes, para isso basta apenas executar o método *SetActive* correspondente ao objeto que se pretende ativar ou desativar, e enviar como parâmetro de entrada um booleano *true* ou *false*, sendo que caso seja *true* o objeto ficará ativo e caso seja *false* ele ficará inativo.

No caso do efeito *Dynamic Color Changer* uma chamada ao método *SetActive* seria impossível, pois o controlo do efeito *Dynamic Color Changer* é feito pelo *script ColorChanger*, como tal é apenas necessário especificar a propriedade *enabled* da classe *ColorChanger* com um booleano, sendo que com *true* o *script* ficará ativo, ou seja, executará o efeito *Dynamic Color Changer*, e com *false* ficará inativo.

É também de notar que para além de desativar ou ativar o *script ColorChanger* é também necessário, caso o *script* seja desativado, restituir a cor de fundo original, isso é possível atribuindo uma cor á propriedade *backgroundColor* do componente *Camera*, daí inicialmente ter sido criada uma variável col que armazena-se a cor por defeito do ambiente

3D. Para restituir a cor por defeito do ambiente 3D após o efeito *Dynamic Color Changer* ter sido desligado é necessário atribuir a variável col á propriedade *backgroundColor*.

3.4.2 O MainLight

Após ter sido criada a câmara é agora necessário criar uma fonte de luz que permita iluminar de forma uniforme todos os objetos 3D criados, ou seja, uma luz do tipo ambiente, como por exemplo o sol, isto para que estes objetos possam ser vistos facilmente pelo utilizador.

Assim é necessário um *GameObject* do tipo *Directional Light*, este objeto já vem automaticamente preenchido com alguns componentes necessários á construção da iluminação.

O primeiro desses componentes é o *Transform*. Tal como no caso da câmara, este componente permite realizar diversas operações de transformação ao *GameObject* como por exemplo mudar a sua posição, realizar rotações e escalar o objeto.

Neste caso foi então definido colocar o *GameObject MainLight* nas coordenadas x: 164.3078, y: -45 e z: 132.9017. Este conjunto de coordenadas coloca a luz de forma semelhante á posição da luz solar natural, ou seja, bastante acima dos objetos do cenário 3D. Foi também realizada uma rotação do *GameObject MainLight*, esta rotação incidiu apenas nos eixos x e y, sendo que no x foi feita uma rotação de 50 graus e no y de 330 graus. Estas rotações tornaram possível que a luz se projetasse sobre os objetos 3D num ângulo inclinado, ou seja da mesma maneira que o sol faz, garantindo assim que este sistema de iluminação fosse o mais realista possível.

O segundo componente é o *Light*, este componente tem uma vasta gama de opções para personalizar e modificar o comportamento e o tipo de luz. A opção *Type* permite definir o tipo de luz a ser usado, sendo que foi selecionado o tipo *Directional* pois permite construir uma luz que incide numa determinada direção, ou seja, permite de forma bastante realista simular a luz solar.

A opção *Color*, esta tal como o nome indica, permite escolher a cor da luz emitida, neste caso como é pretendido simular uma luz ambiental e que tem apenas como função tornar visíveis os objetos no ambiente 3D, a cor ideal para este tipo de situação é a cor branca.

A próxima opção chama-se *Intensity*, e tal como o nome indica permite definir qual é a intensidade da luz, nesta opção, e sendo pretendida uma luz ambiental é mais adequado escolher uma intensidade baixa que apenas permita ter um nível de iluminação suficiente

para que todos os objetos 3D possam ser visíveis, portanto foi escolhida uma intensidade de 0.37.

A opção *Cookie* permite utilizar uma determinada textura para fazer com que a luz emitida siga esse padrão [Unity, 2014h]. Esta opção não é relevante, pois é apenas pretendido que a luz seja emitida de forma uniforme e sem qualquer tipo de efeitos extra.

A *Cookie Size*, tal como a anterior opção não tem relevância, e apenas permite controlar o tamanho da *Cookie* criada anteriormente [Unity, 2014h].

O *Shadow Type* permite determinar se são criadas sombras no ambiente 3D geradas pelo incidir da luz nos objetos 3D. Sendo que o cenário 3D criado não é suposto ser realista e nem ter uma superfície térrea onde essas sombras seriam projetadas, não faz sentido ter a opção ligada para criar sombras. Portanto foi escolhido deixar esta opção com *No Shadows*, ou seja, sem sombras.

Draw Halo permite simular o efeito que acontece quando se está diante uma fonte de luz intensa e esta projeta uma espécie de brilho á sua volta [Unity, 2014h]. Tal como as sombras, não tem interesse que um efeito realista como este seja apresentado no ecrã, portanto esta opção foi desligada.

A *Flare* permite associar um tipo de *Lens Flare* a uma fonte de luz [Unity, 2014h]. Uma *Lens Flare* é um efeito que acontece quando uma câmara filma uma fonte de luz intensa [Unity, 2014i]. Este efeito tal como o referido anteriormente é realista, e portanto não faz sentido tê-lo ligado. Assim foi determinado não associar qualquer *Flare* a esta fonte de luz.

O *Render Mode* determina a importância de uma determinada luz na renderização [Unity, 2014h]. Foi determinado deixar esta opção em *Auto*, e assim deixar o *Unity* determinar da melhor forma a importância desta luz.

Culling Mask permite selecionar determinados objetos para não serem iluminados pela luz [Unity, 2014h]. Neste caso como é pretendido que todos os objetos sejam iluminados de forma uniforme esta foi deixada em *Everything*, ou seja, para que todos os objetos sejam afetados pela luz.

O *LightMapping* permite determinar se as luzes são renderizadas em tempo real, ou seja se algum tipo de pré cálculo da iluminação é efetuado, sendo que esta tem uma implicação direta na performance [Unity, 2014h]. Foi decidido deixar em *Auto*, para que o *Unity* escolha automaticamente a melhor forma de renderizar a iluminação.

3.4.3 O ZigFu

Para que se possa utilizar o *Kinect* como meio de interação com o ambiente 3D criado no *Unity*, é necessário adicionar uma serie de componentes que permitam estabelecer essa interação, esses componentes são fornecidos pelo *plugin Zigfu*.

Foi criado um *GameObject* chamado *Zigfu* e dentro deste foram adicionados uma série de componentes que são necessários para garantir a comunicação e interação com o *Kinect*.

O primeiro componente, tal como em todos os *GameObjects* é o *Transform*, este permite realizar uma série de transformações ao objeto 3D, como por exemplo, rotações, escalamentos, e mudanças de posição. Neste caso como os objetos que vão ser adicionados não têm uma manifestação física no ambiente 3D, ou seja, não são sequer renderizados pois não são apresentados no ambiente 3D, torna-se irrelevante a alteração dos valores do componente *Transform*, portanto os valores foram deixados como os originalmente definidos.

O *script Zigfu* que é fornecido pelo próprio *plugin Zigfu,* permite estabelecer as funções básicas que dão suporte ao *Kinect* no *Unity*. Este fornece uma série de opções configuráveis, sendo que a primeira é o *Input Type*, esta permite escolher qual é o *middleware* a ser utilizado como meio de comunicação com o *Kinect*. Tal como foi indicado no final do subcapítulo anterior, o *middleware* escolhido foi o *Kinect* SDK, portanto foi esse o selecionado.

De seguida existe o grupo de opções *Settings*, onde são apresentadas uma série de definições gerais do *Kinect* e outras especificas para cada tipo de *middleware* de comunicação utilizado. Sendo que o *middleware* escolhido foi o *Kinect* SDK, as opções referentes ao *OpenNI* não têm interesse.

Em primeiro existe um grupo de definições gerais, estas determinam que partes do sensor Kinect vão ser utilizadas. No caso deste protótipo é apenas necessário saber a informação relacionada com a captura da profundidade e da deteção dos utilizadores na câmara, portanto foram selecionadas as opções *Update Depth* e *Update Label Map*, as restantes, *Update Image* e *Align Depth to* RGB não foram selecionadas.

O próximo grupo é específico para o *Kinect* SDK, neste é possível modificar diversos parâmetros do *Kinect*

O primeiro chama-se *Use SDKSmoothing*, este permite que a deteção do esqueleto do utilizador seja mais suave e não tão tremida. Associado a este parâmetro está um grupo de

definições que se chamam de *Smoothing Parameters*, este grupo apresenta uma série de parâmetros como fator de *Smoothing*, fator de Predição, entre outros, que permitem refinar melhor a funcionalidade de *Smoothing*. Foi determinado ligar o parâmetro *SDKSmoothing* e os valores dos *Smoothing Parameteres* foram deixados como por defeito, pois estes produzem resultados satisfatórios em termos da deteção do esqueleto do utilizador.

O *Seated Mode* permite que o *Kinect* detete a pessoa estando sentada, no caso deste protótipo este parâmetro torna-se desnecessário pois não é pretendido detetar utilizadores sentados.

O *Near Mode* permite que o *Kinect* detete os utilizadores a uma distância bastante próxima do sensor. No contexto do *MoveU*, como as distâncias que vão desde o sensor ao utilizador são maiores, este torna-se desnecessário.

Track Skeleton in Near Mode também é desnecessário pois só é utilizado caso o *Near Mode* esteja ativo.

Por fim tem-se os 3 últimos parâmetros, *Show Web Player Logger*, que é irrelevante por só ser necessário no caso do *Unity Web Player*, que neste trabalho não é utilizado, o *Verbose*, que produz informação de diagnóstico mais detalhada na consola do *Unity*, isto para fins de resolução de erros, que está ligado por ajudar na resolução de problemas, e por fim o *Listeners* que permite adicionar objetos para que estejam dependentes das ações do *GameObject Zigfu.* Não foi adicionado nenhum objeto *Listener*, pois é irrelevante para o protótipo MoveU.

Outro componente é o *script Zig Engage Single User*, este é fornecido pelo próprio *plugin Zigfu* e permite detetar um único utilizador fazendo o rastreamento do seu esqueleto, para que depois este possa dar interação a diversos elementos do ambiente 3D.

Existem diversas opções configuráveis neste *script*, a primeira delas chama-se *Skeleton Tracked* e permite tal como o nome indica rastrear o esqueleto do utilizador. Visto que esta opção tem utilidade para este projeto, esta foi deixada ligada.

Raise Hand determina se para que o corpo do utilizador seja detetado, este tenha que levantar a sua mão. No protótipo é pretendido que o utilizador seja detetado de forma imediata sem ter que levantar a sua mão, portanto esta foi desligada.

Por fim têm-se o grupo *Engaged Users*, neste podem ser associados objetos do ambiente 3D para que sejam animados com base na informação do esqueleto do utilizador retornada por

este *script*. No contexto do MoveU a este foi associado o objeto *Blockman*, que irá ser referido mais á frente no presente documento quando se explicar o *GameObject SuperFlameMan*.

3.4.4 O ParticleMan

Depois de terem sido criados os *GameObjects* básicos que permitem dar as funcionalidades como a câmara, a iluminação e a interação com o *Kinect*, chegou a vez de criar os *GameObjects* que asseguram os efeitos especiais que dão a essência ao MoveU.

O primeiro efeito é o *ParticleMan.* Tal como já foi explicado anteriormente, o efeito *ParticleMan* captura os contornos do corpo do utilizador e preenche-os com uma série de partículas que dão um efeito gráfico bastante artístico. Para isto começou-se por criar um *GameObject* vazio com o nome *ParticleMan*, e dentro desde foram adicionados vários componentes que permitem construir o efeito.

O primeiro desses componentes, tal como em todos os outros *GameObjects*, é o *Transform*. Este componente permite realizar uma série de transformações aos objetos 3D, e essas transformações podem ser simples rotações, como também escalamentos ou simplesmente mover o objeto. Era pretendido que o efeito aparecesse á frente da câmara e que permitisse visualizar de forma satisfatória o corpo do utilizador, então foi definido que seriam necessárias diversas transformações, a primeira dessas transformações seria a posição. Neste caso foi determinado que a melhor posição seria atingida usando as coordenadas x: -8, y: 11 e z: 0, isto tendo em conta que este *GameObject* começa a ser desenhado a partir do canto inferior esquerdo. Em termos de rotação não é necessário realizar qualquer alteração, pois o objeto já tem a sua orientação correta de origem. A última transformação necessária prende-se com o escalamento, sendo que é necessário realizar um escalamento para que o objeto tenha as dimensões apropriadas com o objetivo de ocupar a grande parte da área de ecrã disponível. Para o efeito foram definidas as coordenadas de x: 0.05, y: -0.05 e z: 0.001 para efetuar os escalamentos nos respetivos eixos. Estes valores permitiram que o objeto ocupasse a maior parte da área do ecrã para que o efeito fosse perfeitamente visível.

O *Zig Dethmap to Particles* vem incluído com o *plugin Zigfu* e tem como função ler do *Kinect* o mapa de profundidade, de forma a conseguir-se distinguir as formas do corpo do utilizador dos restantes objetos e cenário em sua volta, e de seguida preencher no ecrã as formas do corpo do utilizador com um conjunto de partículas.

Este componente tem também vários parâmetros configuráveis para se poder refinar e ajustar o efeito, o primeiro parâmetro tem o nome de *Grid Scale* e tal como o escalamento do componente *Transform* este permite escalar as dimensões do objeto 3D gerado. Como o escalamento já foi feito recorrendo ao *Transform*, o *Grid Scale* torna-se irrelevante, e por essa razão os seus valores foram deixados os originais.

Desired Resolution permite definir qual é a resolução á qual o objeto 3D gerado pelo *script* irá ser renderizado. Regra geral, quanto maior a resolução, maior o detalhe e a nitidez do objeto gerado. Visto que o preenchimento do corpo do utilizador por centenas de partículas é uma tarefa graficamente exigente para o *hardware*, foi decidido utilizar uma resolução relativamente baixa, neste caso de 200 por 150 pixéis.

Outro fator a ter em atenção é que esta é uma resolução que tem um aspeto de 4:3, isto deve-se ao facto do *script* só suportar resoluções neste formato, portanto a resolução escolhida teria que ser também deste formato.

O Only Users permite escolher se a área preenchida por partículas é apenas a do corpo do utilizador, ou se é também para preencher todo o cenário e objetos que envolvem o utilizador dançarino. Como só era pretendido preencher a área do corpo do utilizador dançarino com partículas esta opção foi ligada.

O *World Space* permite ignorar o escalamento feito pelo componente *Transform* e utilizar em sua vez o escalamento proveniente da opção *Grid Scale.* Como o escalamento foi feito através do *Tranform*, este parâmetro foi desligado.

Velocity permite definir para que eixo é que as partículas devem emitir e a respetiva velocidade, ou seja, se o eixo do x tiver o valor 1 as partículas irão ser emitidas no sentido desse eixo com velocidade 1. Quanto maior o valor de velocidade, mais rápido as partículas emitem.

Podem também ser usadas combinações de eixos e com diferentes velocidades, como por exemplo, eixo do x com valor de 3 e eixo y com valor de 1. Isto iria emitir as partículas numa espécie de diagonal, em que no sentido do eixo do x iria ter uma velocidade de 3 e no sentido de y iria ter uma velocidade de 1. Neste caso simplesmente se definiu que as partículas iriam emitir no eixo do y com uma velocidade de 1.

No *Particle Prefab* pode-se adicionar um *Prefab* de uma partícula, esse *Prefab* será então utilizado para se replicar quantas vezes forem necessárias, a fim de ser possível preencher toda a área do corpo do utilizador.

Um *Prefab* é um tipo de objeto no *Unity* que pode ser criado previamente, adicionando todos os componentes e parâmetros pretendidos, isto para que possa ser utilizado todas as vezes que sejam necessárias, evitando assim a cópia desnecessária de objetos [Unity, 2014j]. Neste contexto um *Prefab* é estritamente necessário devido ao elevado volume de partículas a serem geradas e também devido ao facto de previamente (antes do programa ser executado), não se saberem quantas partículas terão de ser geradas para preencher o corpo do utilizador.

Desta forma o *script* do *Zigfu* apenas analisa a informação de profundidade do *Kinect*, detetando os limites do corpo do utilizador, calcula quantas partículas serão necessárias para o preenchimento do corpo e por último, gera as partículas necessárias usando o *Prefab* e faz o preenchimento da área do corpo do utilizador.

Foi criado um *Prefab* com a partícula pretendida e depois foi adicionado ao *Particle Prefab* para poder ser utilizado pelo *script*. Adiante será descrito como foi criado o *Prefab* das partículas para este efeito.

O parâmetro *Color* define a cor que será utilizada para preencher o corpo do utilizador caso nenhum *Prefab* seja associado ao parâmetro *Particle Prefab*. Foi definida a mesma cor que foi utilizada como cor de fundo do cenário 3D.

Size define o tamanho de cada partícula individual emitida, tendo sido definido com o valor de 1.

O *Energy* define a intensidade da emissão de cada partícula, sendo que foi também definido com o valor de 1.

Por fim *Cycles* indica quantas vezes a partícula é emitida para cada *frame* renderizada, tendo sido o valor definido como 5.

Para terminar é necessário descrever o processo de criação do *Prefab* que foi necessário para que o *script* descrito anteriormente pudesse funcionar.

O primeiro passo passa por escolher um local onde o *Prefab* possa ser guardado. Após o local ter sido escolhido, basta apenas aceder ao menu de contexto e na opção *Create*, escolher *Prefab*. Depois é apenas necessário adicionar os componentes e os parâmetros necessários tal como se trata-se de um *GameObject* normal. Neste contexto foi criado um *Prefab* chamado *Fireworks*, dentro deste foram adicionados vários componentes que são necessários para a criação de um sistema de partículas.

O primeiro é o *Ellipsoid Particle Emitter*, este define o espaço onde as partículas começam a ser emitidas e também a sua velocidade, tamanho, intensidade, entre outros [Unity, 2014k].

O *Particle Animator* anima a partícula após esta ter sido emitida pelo *Ellipsoid Particle Emitter*, neste são definidas as cores da animação, força da animação, entre outras opções [Unity, 2014l].

O *Particle Renderer* é responsável pela renderização da partícula desde a sua emissão até á fase da animação [Unity, 2014m]. Este permite definir opções como por exemplo, definir sombras para as partículas e o tamanho máximo das partículas [Unity, 2014m].

Por último, é apenas necessário adicionar uma textura para ser utilizada como base da partícula [Unity, 2014m].

Após todos estes componentes terem sido adicionados e configurados o *Prefab* fica pronto para ser utilizado pelo *script*.

3.4.5 O SuperSparks

O segundo efeito especial desta aplicação chama-se *SuperSparks*. Tal como foi referido anteriormente, este efeito deteta os movimentos do utilizador e substituí o seu corpo por estrelas brilhantes, e á medida que o utilizador mexe os seus braços estes deixam um rasto luminoso.

Para a criação deste efeito começou-se por criar um novo *GameObject* com o nome *SuperSparks*. Dentro deste foram adicionados outros *GameObjects* (neste caso *child GameObjects*) e também diversos componentes.

No *GameObject SuperSparks* foi apenas adicionado o componente *Transform*, que permite mover, rodar ou escalar objetos do cenário 3D. Decidiu-se movimentar o *GameObject* para o seguinte par de coordenadas, x:-8, y:5 e z:5. Além da mudança de posição do *GameObject* não foram realizadas mais operações de transformação.

Em seguida foi criado um novo *GameObject* vazio que ficaria como *child* do *GameObject SuperSparks*, com o nome de *Sparkle Rising*. Este *GameObject* representa a agregação de todos os componentes necessários para a criação do efeito e sobre essa agregação é também adicionado um *script* que permite detetar a posição do utilizador dançarino em frente do *Kinect* em termos de movimentos laterais e movimentos de altura.

Este *script* ao detetar os movimentos do utilizador afeta o componente *Tranform* do *GameObject Sparkle Rising*, e que por sua vez afeta a posição de todos os outros *child GameObjects* que estejam dentro do *Sparkle Rising*. Assim o que se pretendia era que todo o conjunto de *GameObjects* que constituí o efeito *SuperSparks* acompanha-se a posição do utilizador, e não apenas um dos *GameObjects*, daí se ter criado esta estrutura hierárquica de *GameObjects*, que permite que a posição global do *parent GameObject* afete os seus *child GameObjects*.

Assim dentro do *GameObject Sparkle Rising* foi adicionado o componente *Transform*, e neste não foram feitas quaisquer alterações (ficou com os valores de origem), e foi também adicionado um *script* chamado *ZigUsersRadarHeight*. Este tem por base um *script* do *plugin Zigfu* que permite detetar a posição atual do utilizador. Sendo que no *script* original as posições que eram detetadas eram as laterais e longitudinais e para este caso era pretendido as posições laterais e de altura. Assim foi necessário criar um novo *script* que tivesse em conta esta situação.

O *script* que foi criado está dividido em dois métodos. O primeiro método tem o nome de *Start* e é sempre executado cada vez que a aplicação se inicia. Logo este é um bom método para a inicialização de variáveis e a criação de construtores.

Antes do método *Start* foram declaradas algumas variáveis globais, a primeira dessas tem o nome de *RadarRealWorldDimensions* e é um vetor de duas posições que indica o fator pelo qual as posições lidas pelo *Kinect* serão multiplicadas, a fim de os valores poderem ser utilizados pelos objetos do cenário 3D, ou seja, as posições no mundo real onde o utilizador se encontra são superiores aquelas do mundo virtual, portanto é necessário converter essas posições para valores que possam ser usados no mundo virtual. Esse vetor armazena o fator pelo qual os valores são convertidos.

De seguida tem-se a declaração do objeto *Transform*, que será depois utilizado para armazenar uma instância do componente com o mesmo nome, a fim de se poder movimentar o *GameObject* conforme a posição do utilizador no mundo real.

Por fim foi criado outro vetor de três posições chamado *position* que servirá para armazenar os valores finais da posição do utilizador.

Voltando ao método *Start* este foi utilizado para atribuir ao objeto *Transform* uma instância do componente *Transform*. Para isto recorreu-se ao método *GetComponent* do *Unity*. Tal foi realizado com recurso á seguinte linha de código:

```
tr = GetComponent <Transform> ()
```

O método *Update* deste *script*, é um método que é chamado cada vez que uma *frame* seja renderizada, assim este é o método ideal para conter toda a lógica de interação sobre objetos do cenário 3D. Foi então construído um algoritmo que permitisse ler a posição do utilizador no mundo real e em seguida convertesse essa informação em coordenadas que pudessem ser utilizadas no cenário 3D e aplica-las ao *Transform* do *GameObject Sparkle Rising*. Um excerto do código que foi criado para realizar essas tarefas pode ser visto de seguida.

```
foreach (ZigTrackedUser currentUser in ZigInput.Instance.TrackedUsers.Values)
{
Vector3 com = currentUser.Position;
Vector2 radarPosition = new Vector2(com.x / RadarRealWorldDimensions.x, -com.y
/ RadarRealWorldDimensions.y);
radarPosition.x += 0.5f;
radarPosition.x = Mathf.Clamp(radarPosition.x, 0.0f, 1.0f) * 50;
radarPosition.y = Mathf.Clamp(radarPosition.y, 0.0f, 1.0f) * 100;
position = new Vector3(radarPosition.x-25, -radarPosition.y+8, 5);
tr.position = position;
}
```

Código 3 – Algoritmo que permite a leitura, conversão e atribuição das coordenadas do utilizador ao GameObject Sparkle Rising

Como pode ser visto no excerto de código acima, é criado um ciclo que lê todas as informações relativas a um utilizador que esteja a ser detetado e em seguida, obtém-se os valores da posição do utilizador e faz-se a sua atribuição a um vetor chamado *com*.

Após isso é necessário converter as coordenadas da posição do utilizador no mundo real para coordenadas que possam ser utilizadas no cenário 3D, para tal atribuiu-se ao vetor de duas posições *radarPosition* a divisão de cada uma das coordenadas pelos fatores que foram armazenados no vetor *RadarRealWorldDimensions* (é de notar que o eixo do z foi suprimido. Isto acontece pois esse eixo não é necessário, sendo que o efeito *SuperSparks* não se move em termos de profundidade. Portanto o eixo do z tem um valor fixo que é definido na ultima atribuição do vetor *position*). Isto torna as coordenadas adequadas para a utilização no cenário 3D.

É também de notar que na divisão do eixo do y é efetuada uma troca de sinal, neste caso é aplicado o sinal negativo para que o valor fique negativo. Tal acontece porque nas coordenadas do mundo real no eixo do y os valores são inversos aos valores do eixo do y do cenário 3D, ou seja, se esta troca de sinal não fosse feita, quando o utilizador se movesse para baixo, o efeito no cenário 3D iria para cima e vice-versa. Portanto a inversão do sinal teve de ser realizada para que esta situação fosse resolvida.

Em seguida é necessário somar 0.5 ao valor do eixo do x. Isto acontece porque no mundo real quando o utilizador está centrado em frente ao *Kinect* a sua posição tem um valor de 0, mas em termos de cenário 3D, o facto de estar ao centro no cenário não equivale a este valor, mas sim a 0.5. Portanto tem de se compensar esta situação somando mais 0.5 ao valor do eixo do x.

Depois de efetuado este passo é necessário converter estas coordenadas para que o seu número esteja entre 0.0 e 1.0, ou seja, se um determinado eixo tiver o valor de 50, ao realizar esta conversão ele teria de passar a ser 0.5. Para tal é necessário recorrer á função `Mathf.Clamp`, e nesta foi então especificado que os valores dos eixos do x e do y seriam convertidos para valores que estivessem entre 0.0 e 1.0. Além disto o resultado da conversão é multiplicado por um fator, neste caso de 50 para o x e de 100 para o y. Estes fatores influenciam o valor de distância que é percorrida pelo *GameObject Sparkle Rising* por cada unidade de distância percorrida pelo utilizador no mundo real. Estes valores foram decididos tendo por base uma experimentação de tentativa e erro até que o movimento do objeto *Sparkle Rising* fosse adequado.

Após as operações anteriores foi atribuído ao vetor *position* as coordenadas já convertidas, mas além disso o eixo do x acresce uma subtração de 25 e o eixo do y uma soma de 8, para refinar melhor o movimento do objeto *Sparkle Rising*.

Para terminar é apenas atribuído ao parâmetro *position* do objeto *Transform* as coordenadas finais que se encontram no vetor *position*.

Depois de todos os componentes necessários para o *GameObject Sparkle Rising* estarem criados é necessário criar *GameObjects* que são *child* do *GameObject Sparkle Rising*. Para tal foi criado um *child GameObject* chamado de *Sparkle Particles* que consiste no *GameObject* que trata da primeira parte da animação do efeito *SuperSparks*. Este é constituído por uma série de componentes que em conjunto formam um sistema de partículas. Um sistema de partículas tal como já referido anteriormente é constituído por um *Ellipsoid Particle Emitter*, *Particle Animator* e um *Particle Renderer*, sendo que a explicação da função de cada um destes componentes foi já descrita anteriormente.

Para este sistema de partículas foi definido usar uma textura em forma de estrela, sendo que as estrelas emitidas por este sistema farão contraste com as do próximo *GameObject* que será referido de seguida.

O próximo *child GameObject* tem o nome de *Sparkle Particles Secondary*, e tal como o *Sparkle Rising* é também constituído por uma série de componentes que formam um sistema de partículas. Neste contexto este sistema de partículas tem como papel emitir uma espécie de pontos brilhantes que têm de contrastar com as estrelas emitidas pelo *GameObject* anterior.

Por último tem-se os *child GameObject Left Hand* e *Right Hand*. Estes dois *GameObject* geram a última parte do efeito *SuperSparks*, o efeito do rasto do movimento dos braços. Este efeito pretende deixar uma espécie de rasto á medida que os braços do utilizador se mexem, sendo que o *Left Hand* fica posicionado á esquerda dos *GameObject Sparkle Particles* e *Sparkle Particles Secondary*, e o *Right Hand* fica á direita.

Posto isto os componentes e os parâmetros que constituem os dois *GameObjects* são exatamente iguais, sendo que a única diferença reside no valor do eixo do x do parâmetro *position* do componente *Transform*, em que no caso do *Left Hand* tem o valor de -3 e no *Right Hand* de 3, isto com o objetivo de o *Left Hand* ficar á esquerda e o *Right Hand* á direita.

Assim só irão ser descritos os vários componentes do *GameObject Left Hand* pois são exatamente iguais aos do *Right Hand*.

Para a concretização do efeito do rasto foi necessário usar uma série de componentes, sendo o primeiro, como é habitual, o *Transform*. Tal como foi mencionado acima o *Transform* foi usado para colocar os efeitos no lado esquerdo e direito respetivamente, como tal foi necessário utilizar as seguintes coordenadas para o parâmetro *position*: x:-3, y:2, z:0 (sendo que no caso do efeito do lado direito x tem um valor de 3). Além do parâmetro *position* não foram utilizados mais parâmetros, como o *rotation* e *scale*, pois neste caso não era necessário nem rodar nem escalar os *GameObject* dos efeitos.

O *Zig Follow Hand Point* é um *script* que já vem incluído no *plugin Zigfu* e que permite detetar a posição das mãos do utilizador e atribui-la ao parâmetro *position* do componente *Transform*. Neste *script* existe uma variedade de parâmetros configuráveis, o primeiro tem o nome de *scale*, e tal como um *scale* de um *Transform,* permite aumentar a área deste componente. Visto que este parâmetro não pode ter o valor de 0, foi decidido neste contexto que teria um valor de 0.02 para todos os eixos, isto porque existe um parâmetro que será explicado em seguida que permite controlar de melhor maneira o tamanho deste componente.

O *Bias* permite definir um valor para cada eixo e quanto maior for esse valor, mais severidade terá o movimento da mão do utilizador no respetivo eixo. Pretendia-se um fator de severidade igual em todos os eixos, portanto todos eles foram definidos com o valor de 0.

Damping permite suavizar o movimento da mão do utilizador, ou seja, quando maior o valor definido, mais suave será o movimento. Foi determinado que este parâmetro teria o valor de 5, pois representa em termos visuais uma suavidade de movimento boa.

O parâmetro *Bounds*, este permite definir os limites de movimento que são alimentados ao parâmetro *position* do componente *Transform*, ou seja, se este parâmetro for definido com o valor de 4 para todos os eixos e o utilizador mexer a sua mão para perto dos limites de deteção do *Kinect* isso iria gerar um valor no parâmetro *position* grande, mas devido á limitação do parâmetro *Bounds* de 4 o valor máximo alimentado ao parâmetro *position* é de 4.

Neste contexto como se pretende que os efeitos tenham uma área de movimento limitada para que não saiam fora das suas posições originais, foi definido que o parâmetro *Bounds* teria o valor de 10 para todos os eixos, assim limitando o raio de movimento dos efeitos, mas mantendo-os nas suas posições corretas.

Por fim o último componente necessário chama-se *Trail Renderer*. Este é o componente que permite que seja desenhado um rasto recorrendo a uma textura, á medida que o *GameObject* é movido no ambiente 3D. Portanto este componente associado ao *script* que foi descrito anteriormente permite que conforme o movimento da mão do utilizador seja desenhado um rasto que o acompanha.

O *Trail Renderer* é constituído por diversos parâmetros configuráveis, o primeiro deles chama-se *Cast Shadows*, este indica se o rasto gera sombras noutros objetos. Para o protótipo as sombras são irrelevantes e por isso este parâmetro foi desligado.

Receive Shadows, ao contrário do anterior indica se o rasto deve receber as sombras geradas por outros objetos. Tal como no caso anterior este parâmetro é irrelevante, logo foi desligado.

Materials permite definir uma partícula para ser utilizada na renderização do rasto [Unity, 2014n]. Para o protótipo foi utilizada uma partícula com uma forma de bolha de sabão, que permite dar um efeito estilizado ao rasto.

O *Use Light Probes* e o *Light Probe Anchor* têm funções específicas de cálculo da luz em determinadas situações [Unity, 2014o]. Para o contexto do MoveU ambos os parâmetros não têm interesse, pois só existe uma única luz no cenário 3D e é uma luz que tem como função iluminar todo o cenário de forma uniforme. Portanto estes dois parâmetros foram deixados com os seus valores por defeito.

Time indica quando tempo demora o rasto a desaparecer [Unity, 2014n]. Para tal foi definido um valor de 1 segundo, pois valores maiores causam uma certa confusão na dinâmica do rasto além de provocar um grande impacto na performance gráfica.

Start Width e *End Width* definem a largura do rasto no início e no fim respetivamente [Unity, 2014n]. Foi determinado que os valores ideais para estes parâmetros seria de 1 para ambos.

Colors permite definir uma série de cores a serem utilizadas ao longo do rasto [Unity, 2014n]. Neste contexto não houve interesse em gerar cores ao longo do caminho do rasto, logo não foi definida qualquer cor para este parâmetro.

Min Vertex Distance dita qual é a distância mínima entre segmentos do rasto onde um *vertex* é criado [Unity, 2014n]. Quanto menor for este valor, mais suave será o rasto gerado ou vice-versa. Quanto mais baixo for o valor mais impacto na performance gráfica terá o rasto gerado. Como não existem elementos gráficos que estejam a exigir grande performance do *harware* este parâmetro foi definido com o valor mínimo de 0.1, ou seja, os rastos gerados terão a maior suavidade possível.

Por último, resta o parâmetro *AutoDestruct*, este permite que o rasto seja destruído caso esteja parado por mais que um certo tempo definido [Unity, 2014n]. Como é pretendido que o rasto esteja sempre ativo, este parâmetro foi desligado.

3.4.6 O SuperFlameMan

O terceiro efeito especial desta aplicação chama-se *SuperFlameMan*. Tal como referido anteriormente este efeito cria um *avatar* do utilizador e coloca um efeito de fogo nas suas mãos e pés, sendo que o *avatar* acompanha os movimentos do utilizador á medida que este se mexe, assim como também as chamas. Além disto, no plano de fundo estão também presentes uma espécie de efeito de estrelas que se movem em direções opostas.

Para a construção deste efeito foi necessário criar um *GameObject* vazio chamado *SuperFlameMan*, dentro desse objeto foi necessário criar também vários *child GameObjects* que permitem construir cada uma das partes necessárias a este efeito. Assim os primeiros 5

child GameObjects criados chamam-se *Lines* e todos eles são idênticos exceto num aspecto, que é a sua posição. Estes 5 *GameObjects* não são mais do que sistemas de partículas (já foi referido anteriormente a criação deste tipo de sistemas) que emitem uma série de estrelas como parte do plano de fundo deste efeito. A razão para terem sido criados 5 destes *GameObjects* ao invés de apenas 1 é pelo facto de ter sido determinado que de uma perspetiva artística seria mais interessante se houvesse 5 fluxos de estrelas distintos que se movessem em direções opostas mas paralelas entre elas e que fizessem o preenchimento do plano de fundo.

Para que tal fosse possível seria necessário criar 5 sistemas de partículas distintos, ou seja, criar 5 *GameObjects* que contivessem esses mesmos sistemas. Para o correto posicionamento de cada um deles foi necessário recorrer ao componente *Transform* de cada um dos *GameObjects* e mudar a posição e rotação de cada um deles.

Depois dos 5 *child GameObjects* foi criado um *child GameObject* chamado *BlockManContainer*, este contém tudo o que é necessário para a construção da segunda parte do efeito, o *avatar*.

O *BlockManContainer* tem apenas um único componente, o *Transform*, neste foi apenas feita uma mudança de posição ao longo do eixo do y com o valor de 5.8 e um escalamento em todos os eixos com o valor de 3, a fim de tornar o *avatar* um pouco maior para que aproveite melhor a área do ecrã.

Dentro do *GameObject BlockManContainer* foi criado também um *child GameObject* chamado *BlockMan*, a este foi adicionado um *script* com o nome de *Zig Skeleton*, este já vem incluído com o *plugin Zigfu* e permite detetar o esqueleto do utilizador e representa-lo no cenário 3D através de objetos que podem ser criados em separado.

Este *script* tal como muitos outros tem uma série de parâmetros que podem ser configurados, logo no início uma grande parte deles representa cada membro do esqueleto, ou seja, estes parâmetros têm nomes como por exemplo *Head*, *Torso*, *Neck, entre outros*. Nesses parâmetros é possível indicar que objeto 3D representa determinada parte do esqueleto. Para o *MoveU* foram criados vários cubos, esferas e cilindros que serão utilizados por este *script* para representar o esqueleto. Esses objetos serão descritos mais á frente.

Tirando estes parâmetros que apenas referem as partes do esqueleto têm-se também os parâmetros *Update Joint Positions*, *Update Root Position* e *Update Orientation*.

O parâmetro *Update Joint Positions* indica ao *script* se as posições dos membros do esqueleto são para ser atualizadas conforme os movimentos do utilizador. Sendo este um parâmetro necessário no MoveU ele foi ligado.

Update Root Position indica se a posição global do esqueleto é para ser atualizada conforme a posição do utilizador. Este foi também ligado.

Por fim, o parâmetro *Update Orientation* determina se a direção para a qual o esqueleto se vira é atualizada conforme a direção para a qual o utilizador está virado. Este também foi ligado.

O Mirror indica se os movimentos e posições do esqueleto do ambiente 3D devem ser como uma espécie de espelho do utilizador. Isto é, se o parâmetro for ligado e se por exemplo o utilizador levantar o seu braço direito, o esqueleto do cenário 3D levantará também o braço direito, caso contrário levanta o esquerdo, isto em analogia com o que acontece nos espelhos do mundo real.

Rotate to Psi Pose faz com que o esqueleto do utilizador rode automaticamente para a posição na qual o utilizador se encontra virado mediante este executar a *Psi Pose*. A *Psi Pose* é uma posição em que o utilizador levanta os seus braços no ar num angulo de 90 graus, como se fosse a imitar um cato. Este parâmetro não tem interesse para o protótipo logo foi desligado.

Os parâmetros *Rotation Damping* e *Damping* relacionam-se com o fator de suavização dos movimentos do esqueleto, sendo que o *Rotation Damping* corresponde á suavidade do movimento ao rodar e o *Damping* á suavidade dos movimentos em do esqueleto. Foi definido que os valores ideais para estes dois parâmetros seria de 0 para *Rotation Damping* e 30 para o *Damping*, estes garantem níveis de suavidade satisfatórios nos movimentos.

Scale permite efetuar o escalamento das dimensões do esqueleto representado no ambiente 3D. Estes valores foram definidos em 0.01 para todos os eixos de forma garantir que o esqueleto aparece, mas que as suas medidas originais definidas por cada um dos modelos dos membros não sejam afetadas. Se o valor fosse definido para 0 o esqueleto não seria representado, daí a necessidade de ter um valor de 0.01.

Position Bias permite atribuir valores a cada um dos eixos para fazer mover o esqueleto em certas direções consoante os valores definidos, isto tendo em conta os valores de posição originais. No contexto do MoveU não havia necessidade de realizar qualquer movimentação recorrendo a este parâmetro, portanto todos os valores foram definidos a 0.

Dentro do *child GameObject Blockman* foram criados uma série de outros *GameObjects* que representam cada um dos membros individuais do esqueleto. Estes foram construídos recorrendo a várias formas geométricas, e muitos dos membros são iguais devido às formas simétricas de alguns membros do corpo (ossos do lado direito e esquerdo), devido a este facto será apenas detalhada a construção dos membros essenciais, sendo que os membros simétricos, que são apenas duplicações não serão referidos por motivo de evitar repetição.

Os *GameObjects LeftShoulder*, *LeftElbow*, *LeftHip*, *RightElbow*, *RightHip*, *RightShoulder*, *RightKnee* e *LeftKnee* são iguais e têm os mesmos componentes e parâmetros.

O primeiro desses componentes é o *Transform*. Neste *Tranform* foi apenas utilizado o parâmetro *Scale* para realizar um pequeno escalamento para que os membros aparecessem no ambiente 3D. Foi então definido um escalamento com um valor de 0.1055571 em todos os eixos.

Cube (*Mesh Filter*) permite criar a forma geométrica de um cubo que será utilizada para representar o membro do esqueleto no ambiente 3D.

Box Colider permite gerar as colisões entre vários objetos no ambiente 3D com o cubo que foi criado. Todos os parâmetros deste componente foram deixados com os seus valores por defeito, pois apenas são necessárias colisões simples.

Mesh Renderer permite realizar a renderização do objeto cubo. Os dois primeiros parâmetros deste componente são o *Cast Shadows* e *Receive Shadows*, estes indicam se o objeto cubo recebe e emite sombras de ou nos outros objetos. Ambos foram ligados.

Materials é um componente em que se atribuí um tipo de material para ser renderizado em forma de cubo. Neste contexo foi determinado usar um material com reflecção de luz difusa.

Por fim é criado um componente tendo por base o *script* anteriormente criado chamado *Color Changer Cubes* (que é igual ao *script Color Changer*, mas apenas difere no aspeto de mudar a cor do *GameObject* onde este se encontra). Este *script* permite efetuar a interpolação de cores dinâmica a objetos, tendo o objeto criado, o cubo, ficado com o efeito de cores dinâmicas caso essa opção seja selecionada pelo utilizador na aplicação controladora.

O *GameObject Head* é em tudo semelhante aos descritos anteriormente, sendo que a única diferença reside no componente *Mesh Renderer*, que neste caso, se trata da parte da cabeça

do esqueleto, que terá de ser uma esfera. Portanto em vez de ter sido selecionado um cubo foi selecionado uma esfera para a renderização.

O Torso tal como o *Head* é também igual aos anteriores, mas o *Mesh Filter* é diferente. Visto se tratar da dorsal do esqueleto foi definido que seria mais adequado escolher a forma Capsule (que é semelhante a uma capsula de um comprimido) para representar a dorsal do esqueleto.

Os *GameObjects LeftHand*, *RightHand*, *LeftFoot* e *RightFoot* são também eles iguais aos anteriores, sendo que os seus *Mesh Filter* renderizam a forma geométrica de uma esfera, que foi determinada ser a mais adequada para representar as mãos e os pés do esqueleto.

O que difere substancialmente estes *GameObjects* dos demais é que estes incluem um *child GameObject* chamado Flame. Este representa a construção do efeito que faz com que as mãos e os pés do esqueleto tenham chamas a emanar deles. Dentro do *Flame* foram criados 4 *child GameObjects* que representam os diferentes constituintes do efeito das chamas, esses são o *InnerCore*, *OuterCore*, *Lightsource* e *Smoke*. Cada um deles é constituído por um sistema de partículas que emite as partículas que permitem dar a vida ao efeito das chamas, á exceção do *GameObject Lightsource* que é apenas um sistema de iluminação.

Anteriormente neste capítulo foi referido como é construído um sistema de partículas e um sistema de iluminação, portanto será apenas referido o que é que cada sistema de partículas emite e qual é o propósito do sistema de iluminação.

No caso do *InnerCore*, o sistema de partículas deste *GameObject* emite a zona central e mais amarelada das chamas.

O *OuterCore* tal como o nome indica emite a zona mais externa das chamas, que é representada por uma cor mais avermelhada.

Para tornar o efeito da chama mais realista foi necessário acrescentar elementos extra. Para tal foram criados os *GameObject Lightsource* e *Smoke*.

Lightsource é o sistema de iluminação que foi criado para simular a luz brilhante emitida por uma chama, portanto é uma luz simples de cor alaranjada e que é emitida a partir do centro das chamas.

Por fim o *GameObject Smoke* trata-se de um sistema de partículas que emite o fumo que é normalmente emitido por uma chama ativa.

Todos estes *GameObjects* em conjunto representam um efeito de chamas realista e que é emitido a partir das mãos e dos pés do esqueleto. É também importante referir que este efeito de chamas é igual nos *GameObjects LeftHand*, *RightHand*, *LeftFoot* e *RightFoot*, sendo que é desnecessário referi-lo individualmente para cada um deles.

3.4.7 UltraCubes

O quarto e último efeito especial desta aplicação tem o nome de *UltraCubes*. Este tal como já foi referido anteriormente consiste numa serie de cubos alinhados que formam uma espécie de parede. Esses cubos rodam conforme a posição e movimentos do utilizador. Para construir este efeito foram necessários criar vários *child GameObjects* chamados *Cube*, cada um destes representa um dos cubos apresentados no ambiente 3D, e dentro deles encontram-se os componentes necessários á criação do cubo, bem como também os *scripts* que permitem a rotação destes conforme os movimentos do utilizador.

Visto que todos os cubos são exatamente iguais apenas será detalhada a construção de um deles. É também importante referir que visto que os cubos estão dispostos numa espécie de parede em que todos eles estão alinhados entre si, o componente *Transform* terá valores de posição diferentes em cada um dos *GameObjects,* isto para que estes estejam nas suas respetivas posições corretas e não sobrepostos uns nos outros.

O primeiro componente que é necessário para a criação do cubo é o *Transform*, para o primeiro cubo de todos, este *Transform* fica com as coordenadas de posição a 0. Para os cubos seguintes estas coordenadas vão mudando para que os estes fiquem alinhados em grelha.

Quanto aos parâmetros *Rotation* e *Scale* estes ficam com os seus valores por defeito em todos os cubos, pois não foi necessário realizar qualquer alteração em termos de escalamento ou rotação.

O componente *Mesh Filter* neste contexto, e visto que a forma geométrica desejada era um cubo, foi-lhe definido que a forma em uso seria o *Cube*.

O Mesh Renderer permite renderizar a forma geométrica escolhida no *Mesh Filter*. Além disso o *Mesh Renderer* apresenta vários parâmetros configuráveis, os dois primeiros são o *Cast Shadows* e *Receive Shadows*. Estes ditam se o cubo receberá ou projetará sombras de ou nos outros objetos do ambiente 3D. Ambos os parâmetros foram ativados.

Além dos anteriormente referidos o parâmetro *Materials* permite definir o tipo de material com o qual o cubro será preenchido. No contexto do protótipo foi definido utilizar uma textura simples de cor cinza com um *shader* do tipo especular.

Zig Users Radar é um *script* que permite os cubos do ambiente 3D rodarem conforme o movimento e a posição do utilizador. Este está dividido em dois métodos distintos, um método *Start* e outro *Update*.

O método *Start* é executado logo após a aplicação ter sido iniciada e o método *Update* é executado sempre que cada *frame* é renderizada. Posto isto, o *Start* é ideal para a inicialização de variáveis e a criação de construtores e o *Update* é ideal para o código que contenha a lógica da aplicação.

Este *script* é muito idêntico ao *Zig Users Radar Height*, sendo que o método *Start* é totalmente igual, e a primeira parte do método *Update* é também semelhante. Desta forma será apenas detalhada a parte que é diferente do *script Zig Users Radar Height*.

A parte que se diferencia é a que calcula o ângulo de rotação dos cubos tendo em conta a posição do utilizador no mundo real. Para tal foi criada uma equação que permitisse estabelecer uma relação entre um determinado ângulo de rotação e uma determinada posição do utilizador dançarino. Recorreu-se então a uma calculadora gráfica e construiu-se uma tabela em que foram estabelecidas algumas relações entre a posição do utilizador dançarino e o ângulo de rotação, como pode ser visto de seguida.

Tabela 1 - Tabela que representa a relação entre as posições do utilizador e os ângulos de rotação

Posição Utilizador	Ângulo de Rotação do Cubo
0.6	18°
0.5	0°
0.1	-72°

Como pode ser concluído pela análise da tabela, se a posição do utilizador dançarino for 0.5, ou seja, se este se encontrar no ponto central tanto do eixo do x como do y, os cubos terão uma rotação de 0 graus. Mas se o utilizador dançarino se mover para a posição de 0.1 os cubos terão uma rotação de -72 graus. É também importante referir que as posições do utilizador se encontram entre 0.0 e 1.0, ou seja, 0.0 e 1.0 representam as extremidades máximas em termos de posição que o sensor *Kinect* consegue capturar. Portanto tendo em conta tal a posição central é equivalente ao valor 0.5. É também importante referir que os valores da tabela, e os valores dos extremos são válidos para o eixo do x e do y. Para além

disso existem também valores máximos em termos de ângulos de rotação dos cubos, sendo que os extremos vão desde 90 graus a -90 graus.

Recorrendo á calculadora gráfica, os dados representados na tabela acima foram traduzidos numa equação de reta, como pode ser visto abaixo.

$$y = 180x - 90$$

Esta equação, a um determinado valor de entrada x, que representa a posição atual do utilizador, faz sair um determinado valor y, que representa o ângulo que o cubo terá que rodar. Abaixo pode ser visto o gráfico que representa a reta desta mesma equação.

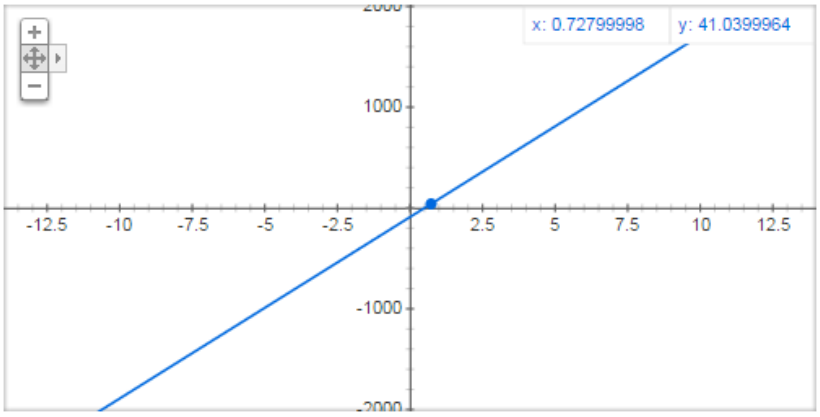

Figura 56 - Gráfico que representa a reta da equação y = 180x - 90

Depois de ter sido estabelecida a equação que permite fazer a tradução das posições do utilizador em ângulos para a rotação dos cubos, foi então criado no código do *script Zig Users Radar* a linha de código que representa essa mesma equação, como pode ser visto abaixo.

```
angulo = new Vector3(180 * radarPosition.y - 90, 180 * radarPosition.x - 90, 0)
```

Nesta linha é atribuída a uma variável ângulo um vetor de 3 valores, esses valores representam o ângulo de rotação de cada eixo. No caso do eixo do x e y, pode ser visto que contém a equação que foi determinada anteriormente, ou seja:

```
180 * radarPosition.y – 90
```

Sendo que `radarPosition.y` e `radarPosition.x` representam as posições atuais do utilizador nos respetivos eixos e já com os seus valores limitados aos extremos de 0.0 e 1.0.

É de notar que o terceiro valor do vetor que representa o eixo z, tem um valor fixo de 0, isto porque foi definido que os cubos não iriam realizar rotação conforme a posição da altura do utilizador. Portanto este valor foi então fixado a 0 para que neste eixo não houvesse qualquer alteração.

Após essa linha de código realizar o cálculo dos ângulos de rotação para os cubos é então necessário limpar os antigos ângulos em que os cubos estariam rodados, esta operação pode ser realizada com recurso á seguinte linha de código.

```
tr.rotation = Quaternion.identity
```

Esta linha atribuí um vetor de ângulos em identidade á propriedade *rotation* do componente *Transform*, por forma a limpar as rotações anteriormente realizadas.

Por último, é necessário atribuir os novos ângulos que foram calculados á propriedade *rotation* do componente *Transform*. Para tal é necessário recorrer ao método *Rotate* do componente *Transform*, como pode ser visto na linha de código abaixo.

```
tr.Rotate(angulo)
```

Nesta linha pode-se ver que é chamado o método *Rotate* do componente *Transform*, e é enviado como parâmetro de entrada o vetor que contém os novos ângulos de rotação para que as rotações dos cubos sejam atualizadas.

Abaixo, apresenta-se um excerto de todo o algoritmo do método *Update* do *script Zig Users Radar*.

```
foreach (ZigTrackedUser currentUser in ZigInput.Instance.TrackedUsers.Values)
{
Vector3 com = currentUser.Position;
    Vector2 radarPosition = new Vector2(com.x / RadarRealWorldDimensions.x, -
com.z / RadarRealWorldDimensions.y);
    radarPosition.x += 0.5f;
    radarPosition.x = Mathf.Clamp(radarPosition.x, 0.0f, 1.0f);
    radarPosition.y = Mathf.Clamp(radarPosition.y, 0.0f, 1.0f);
    angulo = new Vector3(180 * radarPosition.y - 90, 180 * radarPosition.x -
90, 0);
    tr.rotation = Quaternion.identity;
    tr.Rotate(angulo);
}
```

Código 4 – Algoritmo que permite a leitura, conversão, cálculo de ângulos e atribuição destes ao componente Tranform do GameObject Cube

O último componente é um *script* que tem o nome de *Color Changer Cubes*. O objetivo deste é permitir que todos os cubos tenham uma mudança de cor dinâmica. Este *script* é exatamente igual ao *Color Changer* e apenas difere no facto de mudar dinamicamente as cores dos cubos. Por essa razão não será explicado o código referente a este *script* visto que este já foi referido anteriormente por via do *script Color Changer*.

3.5 Desenvolvimento da aplicação controladora

A aplicação Controladora tal como foi referido no inicio deste capítulo é uma aplicação cujo propósito é controlar remotamente os efeitos do protótipo. O objetivo desta aplicação seria de ter uma simples janela com uma série de botões tipo rádio, que permitisse comutar entre os diferentes efeitos e recorrer á tecnologia dos *Web Services* para realizar a comunicação da aplicação com o protótipo. Para tal foi criado um novo projeto no *Microsoft Visual Studio* e criou-se uma *Form* e respetiva classe, e também criaram-se duas classes adicionais chamadas *WebService* e *IWebService* para suportar as funcionalidades dos *Web Services*.

3.5.1 Form

Esta *Form* foi criada para suportar a interface base da aplicação. Nesta interface foram criados quatro *Radio Buttons* e uma *Check Box*. Cada um dos *Radio Buttons* representa um efeito diferente do protótipo, enquanto que a *Check Box* representa o efeito *Dynamic Color Changer* que pode ser ligado em conjunto com qualquer um dos efeitos representados pelos *Radio Buttons*, daí este efeito estar numa *Check Box* e não num Radio Button.

Dentro da classe da *Form* foram inicializadas as variáveis que representam o estado atual dos efeitos, ou seja, as variáveis guardam qual é o efeito atualmente selecionado pelo utilizador e também se este ligou ou não o efeito *Dynamic Color Changer*. Para tal foram então criadas as variáveis estado e *estadoColorChanger*. A variável estado é do tipo inteiro, e é inicializada com o valor de 1 para que ao abrir a aplicação o efeito selecionado por defeito seja o o *Particle Man*.

A variável *estadoColorChanger* é do tipo booleano, e é inicializada com o valor de *false* para que quando a aplicação seja aberta o efeito *Dynamic Color Changer* esteja desligado por defeito.

Além disso é também criado o construtor da classe do *Web Service* que será referida mais á frente e o respetivo inicializador do serviço.

Por fim é também selecionado por defeito o *Radio Button* do efeito *Particle Man* para que esteja em concordância com o valor por defeito da variável estado.

Depois de todas as inicializações terem sido efectuadas, foram criados métodos que são executados cada vez que o utilizador carrega num dos *Radio Buttons* ou na *Check Box*. Dentro de cada um destes métodos foi criado um pequeno algoritmo que altera o valor das variáveis estado e *estadoColorChanger* conforme a *Radio Button* ou *Check Box* selecionados. Esse

algoritmo é igual para todos os métodos referentes aos *Radio Buttons*. De seguida fica um excerto do algoritmo.

```
if (ParticleMan.Checked == true)
{
estado = 1;
}
if (SuperSparks.Checked == true)
{
estado = 2;
}
if (SuperFlameMan.Checked == true)
{
estado = 3;
}
if (UltraCubes.Checked == true)
{
estado = 4;
estadoColorChanger = true;
ColorChanger.Checked = true;
}
```

Código 5 – Algoritmo que permite a alteração das variáveis estado e estadoColorChanger conforme o Radio Button selecionado

Como pode ser visto no algoritmo cada *if* verifica qual é o *Radio Button* que foi selecionado e muda a variável de estado de acordo com essa mesma seleção. É de notar que no caso do efeito *UltraCubes* é também automaticamente ativado o efeito *Dynamic Color Changer*, isto porque foi definido que quando o efeito *UltraCubes* é ativado a mudança dinâmica de cores é também ativada, Por isso pode ser visto no algoritmo, que no *if* que faz a verificação do *Radio Button* do *UltraCubes*, é alterada a variável estado e *estadoColorChanger*, além de que também é automaticamente feito o *check* á *Check Box* do efeito *Dynamic Color Changer*.

No caso relativo ao método que é executado cada vez que uma alteração é feita na *Check Box* do efeito *Dynamic Color Changer*, foi feito também um pequeno algoritmo que permite alterar o estado da variável *estadoColorChanger* conforme a *Check Box* tenha sido clicada pelo utilizador ou não. De seguida pode ser visto um excerto desse algoritmo.

```
if (ColorChanger.Checked == true){
estadoColorChanger = true;
}
else { estadoColorChanger = false;
}
```

Código 6 – Algoritmo que permite a alteração da variável estadoColorChanger conforme o estado da Check Box

Como pode ser visto no algoritmo acima o *if* verifica se a *Check Box* foi clicada pelo utilizador para ativar o *check* e caso isso tenha acontecido a variável *estadoColorChanger* passa para o valor *true*, caso o utilizador tenha clicado para retirar o *check* a variável *estadoColorChanger* passa para o valor *false*.

3.5.2 WebService e IWebService

O *Web Service* disponibilizado por esta aplicação é construído tendo por base duas classes, a classe *WebService* e *IWebService*, sendo que a primeira é constituída por todos os métodos inerentes ao próprio *Web Service* e a segunda é apenas uma interface dos métodos disponibilizados pelo *Web Service*.

Dentro da classe *WebService* existem três métodos, os dois primeiros chamam-se *getEstado* e *getColorChanger*. Estes dois métodos apenas fazem o retorno do valor contido nas variáveis estado e *estadoColorChanger* que já foram descritas anteriormente. Esses valores são retornados e enviados via *Web Service* para o protótipo que depois os irá receber e através deles determinar qual o efeito a apresentar atualmente.

O último método chama-se *run* e contem a definição do *endpoint* do *Web Service* e a abertura da conexão dele mesmo, como pode ser visto no seguinte excerto de código.

```
ServiceHost host = new ServiceHost(typeof(WebService), new
Uri("http://localhost:7777/server"));
host.Open();
```

Código 7 – Código que cria um novo endpoint num endereço específico e faz a respetiva abertura de conexão

Como se pode ser no código acima, é criado um novo *endpoint* no endereço da própria máquina local, na porta 7777 e na localização *server*. Após este ter sido criado é executada a linha `host.Open()` para que sejam aceites conexões no respetivo endereço definido.

A classe *IWebService* tal como foi referido anteriormente, é apenas uma interface para os métodos *getEstado* e *getColorChanger* da classe *WebService*. Estes são os dois métodos que são disponibilizados para acesso via *Web Service* e assim têm de estar na interface *IWebService* para que tal seja possível.

3.6 Testes ao protótipo e á aplicação controladora

Durante e após a construção do protótipo e da aplicação controladora, foram efetuados vários testes no sentido de validar o funcionamento de ambos. Os testes realizados cobriram uma diversidade de funcionalidades para assegurar que tudo funcionaria efetivamente como esperado e planeado.

Apresentam-se alguns dos testes realizados tanto ao protótipo como á aplicação controladora, mostrando a metodologia adotada para os testes e uma descrição sobre o tipo de teste e sobre os seus resultados.

3.6.1 Testes ao protótipo na sua fase preliminar

Numa fase inicial o protótipo era apenas constituído por um único efeito, esse efeito era o *Particle Man*. Além desse efeito também continha alguns elementos extra, como por exemplo algumas partículas e texturas, elementos esses que não foram mantidos na versão final do protótipo.

O objetivo dos testes nesta fase inicial foi perceber se o conceito planeado do protótipo funcionaria no mundo real. Para validar esse conceito foi determinado que o ideal seria convidar algumas pessoas a testar o protótipo inicial para saber posteriormente a sua opinião após o terem experimentado. As figuras seguintes pretendem ilustrar o momento dos testes preliminares sendo possível visualizar na figura abaixo dois dos participantes á esquerda sendo o efeito projetado no quadro á direita.

Durante estes primeiros testes e momentos de experimentação do protótipo MoveU, foi possível perceber que os participantes revelaram um grande interesse pela interação que tinham experimentado constatando-se que os gestos que faziam saiam naturalmente quase que como a desafiar e a experimentar as limitações do sistema de interação corpo/imagem.

Figura 57 - Testes ao Protótipo inicial no Instituto Superior de Engenharia do Porto

A figura abaixo mostra o detalhe de uma projeção resultante dos movimentos reais executados pelos participantes nesta fase de testes beta. Como pode ver-se, é também possível acrescentar elementos gráficos ao cenário para além da imagem representativa do utilizador sendo neste caso uma bola de fogo que liberta bolas de sabão.

Figura 58 - Protótipo numa fase Inicial

Questionamos os participantes destes testes sobre qual a sua opinião sobre o conceito deste protótipo inicial. Ambos responderam que gostaram bastante do conceito e acharam que o protótipo já estava bastante funcional e interativo.

Para além do teste anteriormente descrito solicitou-se também a um grupo de 21 alunos que assistissem a um teste realizado a fim de simular a utilização do protótipo num ambiente de espetáculo, sendo que os 21 alunos representariam o público que iria assistir ao espetáculo (fig. 59). Este teste foi importante para poder demonstrar que os efeitos proporcionados pelo protótipo cativariam o público de forma a tomar uma maior atenção aos efeitos do que á dança desempenhada pelo dançarino, sendo que caso tal acontecesse o objetivo do protótipo seria positivo.

Figura 59 - Apresentação a um público simulado

Depois de realizado esse teste foi possível aferir que a grande maioria dos 21 alunos que participaram no demonstraram um grande agrado no protótipo e revelaram também que prestaram uma maior atenção aos efeitos por ele proporcionados do que á dança realizada pelo dançarino.

O *feedback* transmitido pelos participantes foi bastante valioso e motivador uma vez que permitiu validar, pelo menos inicialmente, todo o conceito que foi planeado para este projeto e testar o seu nível de funcionalidade e interação existentes no protótipo inicial.

Estes participantes sugeriram a integração no protótipo de novos efeitos visuais que foram tidos em conta. Para além disso todos se mostraram bastante agradados com o conceito o que foi motivador tendo-se nesta fase decidido que o desenvolvimento do protótipo e do seu conceito poderiam e deveriam seguir em frente, considerando-se nesta etapa integrar novos efeitos e planear atividades que pudessem ser testadas por bailarinos profissionais futuramente.

3.6.2 Testes ao protótipo final

Quando o protótipo foi concluído idealizaram-se uma série de testes para validar se este estaria de acordo com aquilo que tinha sido planeado no início, e se estaria adequado á utilização em ambientes de espetáculos.

O primeiro teste a ser criado tinha como intuito determinar se num ambiente de total escuridão o protótipo conseguiria detetar os movimentos do utilizador dançarino sem qualquer problema.

Para este teste foi montado o protótipo numa sala completamente escura (apenas com a luz do projetor) mas com a zona de captura não iluminada, o autor, com a ajuda de um participante voluntário a simular a ação do utilizador controlador, fez alguns movimentos para que o protótipo os capturasse (fig. 60). Caso o protótipo conseguisse capturar todos os movimentos feitos pelo participante sem qualquer problema ou dificuldade, poder-se-ia afirmar que o teste seria concluído com sucesso.

Figura 60 - Testes com o ambiente escurecido

Após a realização do teste é possível afirmar que foi concluído com sucesso, pois durante todo o teste o protótipo conseguiu capturar cada um dos movimentos do participante sem

qualquer problema e sem que a escuridão afetasse de alguma forma a performance e a qualidade de captura.

Posto isto pode-se afirmar que o protótipo está pronto para ser utilizado em ambientes escuros (que são típicos dos espetáculos) sem grandes problemas.

O segundo teste tinha como objetivo determinar quais seriam as distâncias laterais e longitudinais máximas que o protótipo conseguiria capturar movimentos feitos pelos utilizadores dançarinos. Para tal o protótipo foi montado numa sala e com a ajuda de um participante, foi-lhe pedido numa primeira fase que se movimentasse o mais possível para trás (isto no eixo cartesiano y) até que o protótipo deixasse de capturar os seus movimentos. No momento em que os movimentos deixassem de ser capturados foi pedido ao participante para parar de andar para trás e socorreu-se de uma fita métrica para medir a distância entre o sensor *Kinect* e o participante, esta distância representa o máximo que o protótipo consegue capturar em termos longitudinais. Depois de medida a distância, foi determinado que esta era de aproximadamente 3,60m (fig. 61).

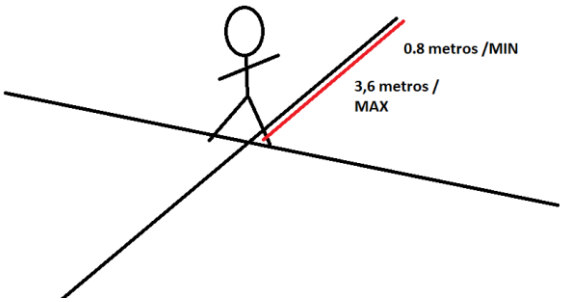

Figura 61 - Distâncias máxima e mínima longitudinal de captura do utilizador[58]

O mesmo procedimento foi feito para determinar a distância mínima longitudinal a partir da qual o sensor Kinect deixa de capturar os movimentos. Foi determinado que essa distância mínima é de aproximadamente 0,80m.

Foi depois pedido participante para se mover lateralmente (isto no eixo cartesiano x) até ao protótipo deixar de capturar os seus movimentos. No momento em que tal acontecesse era pedido ao participante para parar e em seguida a distância entre o *Kinect* e este era medida, isto para que se conseguisse determinar qual é a distância máxima lateral que o protótipo consegue capturar os movimentos dos utilizadores.

[58] Imagem da autoria do autor

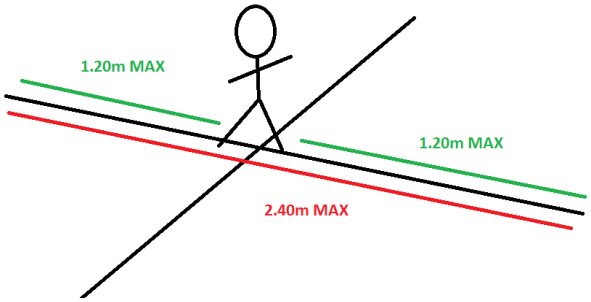

Figura 62 - Distâncias laterais máximas de captura do utilizador[59]

Após ter sido concluído o teste foi determinado que a distância lateral máxima em que o protótipo consegue capturar movimentos é de aproximadamente 2,40m, ou seja, 1,20m a partir do ponto central onde se encontra o *Kinect* para ambos os lados (esquerdo e direito) (fig. 62).

Além destes dois testes de distância foram também medidas as mesmas distâncias mas em ambiente de total escuridão (tal como no primeiro teste) a fim de determinar se a escuridão teria alguma influência nas distâncias máximas de captura dos movimentos dos utilizadores dançarinos.

Depois dos testes terem sido feitos em ambiente escuro pôde-se afirmar que as distâncias não sofreram qualquer alteração em relação às obtidas em ambiente luminoso.

O último teste criado tinha o objetivo de determinar se o sensor *Kinect* poderia ser utilizado numa posição invertida, ou seja, isto seria como se o *Kinect* ficasse virado ao contrário para que pudesse ser acoplado ao teto, em vez de ficar em cima de uma mesa ou no chão.

Este teste foi idealizado porque poderá haver algum interesse num cenário real de espetáculo em colocar o *Kinect* no teto em vez de no chão ou mesa, e para validar se tal é possível é necessário testar se o protótipo consegue lidar com este fator sem que os seus efeitos gerem problemas ou simplesmente não funcionem corretamente.

Para tal, foi montado o protótipo numa sala e o sensor *Kinect* foi colocado numa posição invertida. De seguida foi pedido a um participante que se movimentasse em frente ao *Kinect* a fim de testar o protótipo e cada um dos seus efeitos. Após o teste ter sido concluído pode-se afirmar que todos os efeitos com a exceção do *UltraCubes* não funcionam corretamente com o *Kinect* invertido.

[59] Imagem da autoria do autor

O efeito *UltraCubes*, apesar de mostrar os cubos com interação, não funciona totalmente bem pois o movimento lateral dos cubos em relação ao movimento do utilizador dançarino no mundo real fica invertido. Portanto o protótipo não se encontra preparado para lidar com a situação da inversão do posicionamento do sensor *Kinect*.

Muitas outras possibilidades haveria de interligar os cenários interativos proporcionados pelo protótipo *MoveU*. A título meramente ilustrativo, apresenta-se na imagem seguinte uma solução que poderia ser aplicada ao chão onde o dançarino efetuaria a sua performance tornando-o interativo através dos movimentos provenientes da dança (fig. 63).

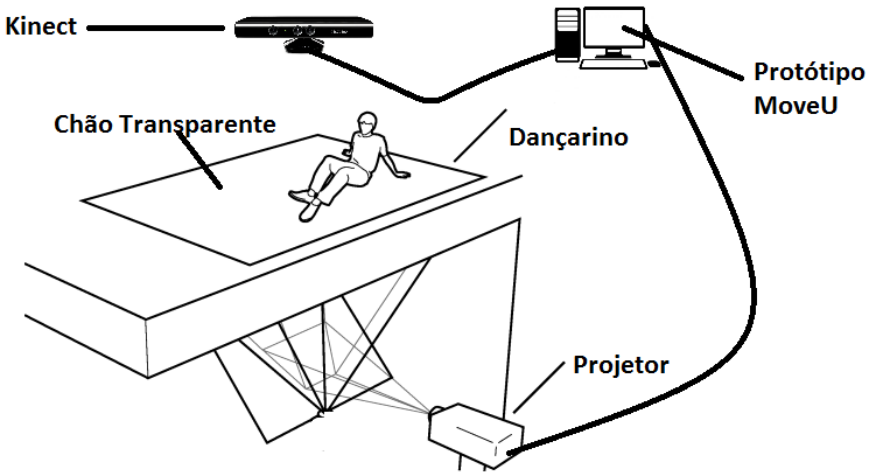

Figura 63 - Exemplo de outros formatos possíveis de aplicação do MoveU[60]

Para além dos testes referidos anteriormente foram também realizados alguns testes em ambiente real, mais concretamente em algumas escolas de dança. Para tal montou-se nesses locais o protótipo e pediu-se para alguns dançarinos profissionais que realizarem algumas performances e enquanto testava-mos o *MoveU*.

Nas figuras seguintes pode observar-se alguns dos vários testes efetuados aos diferentes efeitos do protótipo *MoveU* na primeira escola de dança. Nesta escola, testou-se o protótipo tanto com danças a solo como também com danças a par (nos efeitos mais adequados para tal situação). Em ambos os casos todos os efeitos funcionaram sem qualquer problema a reportar.

Na figura seguinte pode ver-se uma bailarina a executar uma performance sendo utilizado o efeito *ParticleMan* que, é projetado numa parede da sala podendo perceber-se que os

[60] Imagem adaptada de http://1.bp.blogspot.com/-Yqh4poc0HNk/Ugokn7AZWnI/AAAAAAABsBM/5KqQRyTOOyA/s1600/Figure_4.png

movimentos capturados eram precisos. Como pode ver-se na figura, praticamente todos os presentes centraram a sua atenção na projeção, incluindo a bailarina.

Figura 64 - Efeito ParticleMan em ação aplicado a ballet

Na imagem seguinte apresenta-se uma performance com dois bailarinos de ballet podendo ver-se a projeção do efeito na parede resultante dos seus movimentos. Este efeito funcionou muito bem com danças a pares.

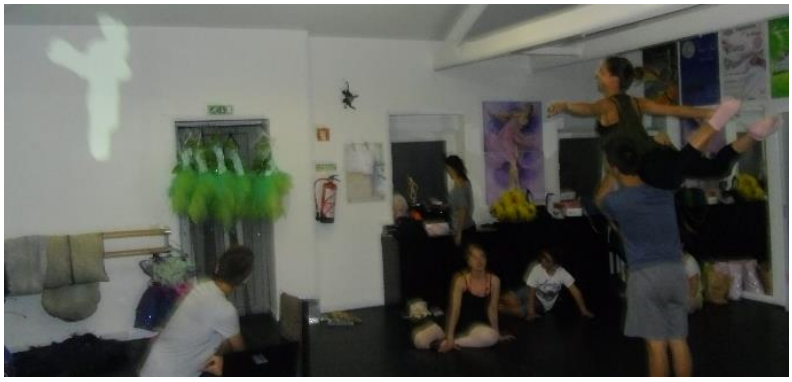

Figura 65 - Efeito ParticleMan com dois dançarinos

Ainda com a primeira bailarina foram testados outros efeitos. Neste caso, como se apresenta na figura seguinte foi o *UltraCubes*.

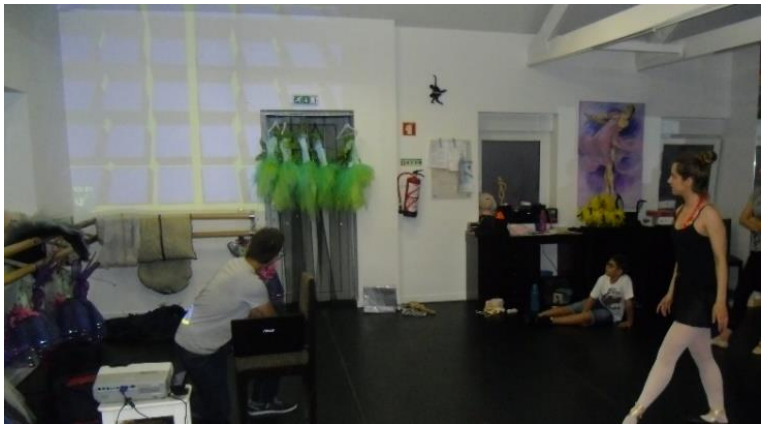

Figura 66 - Demonstração do efeito UltraCubes

Foi curioso notar que, ao contrário do que seria de esperar a bailarina se adaptou rapidamente ao efeito começando a fazer passos mais suaves porque percebeu que o *UltraCubes*, apesar da sua boa sensibilidade ao movimento poderia adaptar-se melhor a esse tipo de movimentos.

Podem ser observados, nas figuras seguintes alguns dos diferentes testes realizados na segunda escola de dança, a GlobalDança. Foram realizados testes a todos os efeitos proporcionados pelo protótipo *MoveU*, sendo que tal como na primeira escola, foram testados com danças a solo e também danças a par (para os efeitos preparados para tal situação). Na maioria dos testes realizados não foram encontrados quaisquer problemas, tal como pode ser visto nas figuras 67 e 68, contudo, foram encontradas duas situações que causaram alguns constrangimentos sendo que estes pequenos problemas serão detalhados nos parágrafos seguintes.

Figura 67 - Efeito UltraCubes em ação

Na figura anterior pode ver-se um teste a solo com o efeito *UltraCubes*. Á semelhança dos testes realizados na primeira escola também aqui foi possível perceber que o dançarino começou a fazer movimentos mais suaves.

Figura 68 - Demonstração do efeito ParticleMan

Como se observa na figura 69, quando o dançarino ultrapassa a distância mínima de deteção do *Kinect* o efeito *ParticleMan* deixa de reconhecer os movimentos do dançarino. Isto vem comprovar as distâncias mínimas e máximas determinadas anteriormente. Neste caso para evitar o problema demonstrado na figura 69, o dançarino não deve estar a menos de 0,80m do sensor *Kinect*, sendo que na figura ele encontra-se a 0,40m do sensor estando assim a causar o não reconhecimento dos seus movimentos.

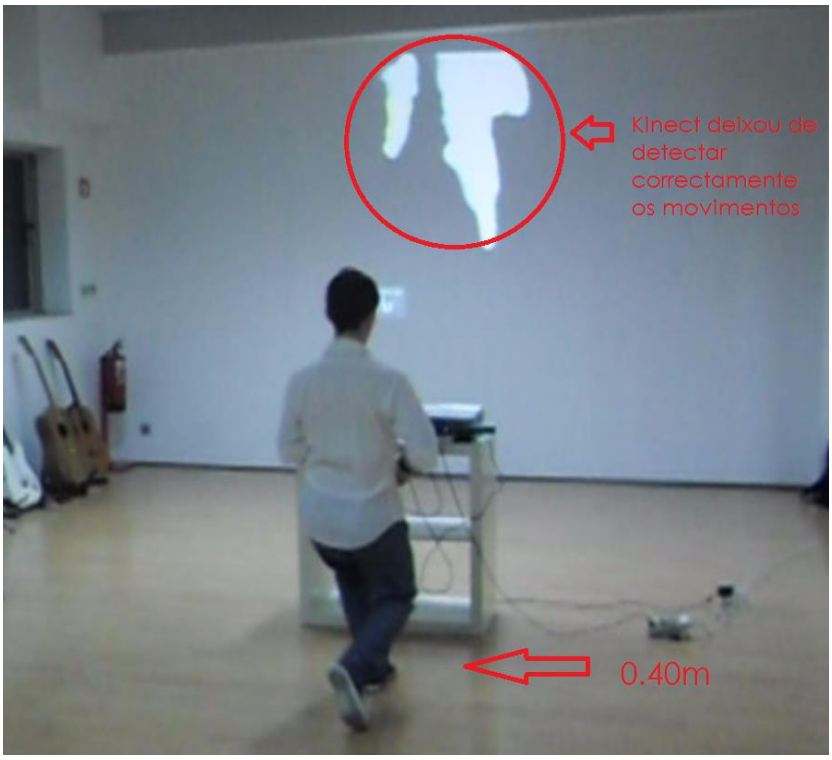

Figura 69 - Figura que demonstra o que acontece quando a distância mínima de deteção do Kinect é ultrapassada

Na figura 70 pode-se ver o problema que acontece quando o efeito *ParticleMan* está a ser utilizado e os dois dançarinos dançam muito próximos um do outro.

Nesta situação, e mesmo cumprindo as distâncias mínimas e máximas de reconhecimento, os dois dançarinos não são reconhecidos pelo *Kinect*. Portanto o protótipo não se encontra preparado para lidar com este tipo de situações em que ambos os dançarinos estão muito próximos um do outro, sendo que esta até poderá ser uma limitação do próprio sensor *Kinect*.

Figura 70 - Figura que demonstra o que acontece quando dois dançarinos dançam de uma forma muito próxima

Capítulo 4 – Avaliação do Protótipo

"Conhecer não é demonstrar

nem explicar, é aceder à visão."

Antoine de Saint-Exupéry

Neste capítulo é feita uma introdução á problemática da avaliação do protótipo MoveU e também serão abordados os métodos estatísticos utilizados para validar o protótipo. Serão também demonstrados os respetivos resultados, nas várias perspetivas de utilização (dançarino, espectador e controlador).

4.1 Introdução

Após a criação do protótipo *MoveU* foi necessário perceber até que ponto é que os objetivos propostos foram alcançados com sucesso. Foi também importante determinar se o protótipo seria do agrado dos vários públicos-alvo e também se estaria preparado para ser utilizado num ambiente real.

Para responder a todas as questões anteriores, foram criados alguns inquéritos (anexos 1,2 e 3) para que fossem respondidos pelos vários públicos-alvo após estes terem tido contacto como espectador ou experimentado o protótipo *MoveU*.

Depois de terem sido obtidas todas as respostas por parte dos diferentes tipos de utilizador ou público, os resultados foram exportados da plataforma *Google Forms* (os questionários foram realizados nesta plataforma) a fim de se realizarem uma série de análises estatísticas que serão detalhadas no seguinte subcapítulo.

4.2 Análise dos resultados

Foram criados três tipos de inquérito distintos (anexos 1, 2 e 3), isto a fim de melhor perceber a opinião de cada um dos públicos-alvo em diferentes perspetivas de utilização. O primeiro tipo de inquérito destinou-se a pessoas que tinham experimentado o protótipo *MoveU* na perspetiva do dançarino ou artista, ou seja, estas pessoas realizaram uma performance ou espetáculo em frente ao sistema.

O segundo tipo de inquérito destinou-se a pessoas que tivessem assistido á experimentação do sistema por dançarinos, ou seja, estas representaram o público que assistiu a um espetáculo onde o protótipo *MoveU* foi utilizado.

Por fim o terceiro tipo de inquérito destinou-se a pessoas que tivessem experimentado controlar os efeitos do protótipo recorrendo á aplicação controladora.

Estes três tipos distintos de inquérito cobriram as três perspetivas distintas de utilização do protótipo e cobriu também os três públicos-alvo deste sistema, os dançarinos/artistas, o público que assiste a espetáculos e os responsáveis por controlar os efeitos do espetáculo.

Ao inquérito destinado a quem experimentou o protótipo na perspetiva do dançarino, responderam no total 45 pessoas, sendo que uma parte destas pessoas eram dançarinos que faziam parte de algumas academias de dança (tal como foi referido no capitulo anterior) e outra parte eram pessoas sem qualquer ligação a academias de dança.

Quanto ao inquérito destinado ao público que assiste a um espetáculo em que é usado o protótipo *MoveU*, responderam no total 74 pessoas.

Por fim ao inquérito destinado a quem experimentou controlar os efeitos do protótipo recorrendo á aplicação controladora, responderam no total 38 pessoas.

A próxima secção apresentará uma serie de gráficos que foram criados tendo em conta os resultados estatísticos exportados da plataforma *Google Forms* e que por sua vez foram tratados no *Microsoft Excel*.

4.2.1 Análise aos inquéritos da perspetiva do dançarino

Naturalmente, um dos pontos de vista mais importantes na relação utilizador/*MoveU* é o dos dançarinos que representam neste processo o dinamismo que irá provocar e interagir com o sistema proposto.

No gráfico apresentado na figura 71 pode-se observar que 56% dos inquiridos são do sexo feminino sendo os restantes 44% do masculino.

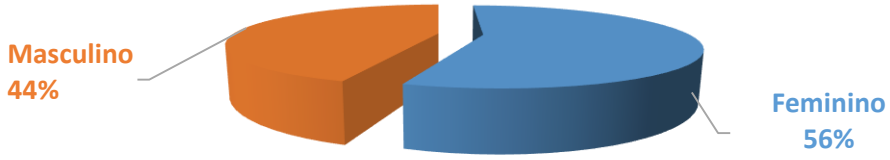

Figura 71 - Distribuição por sexo dos inquiridos dançarinos

Foi possível verificar que 71% dos inquiridos se situam numa faixa etária entre os 25 e os 45 anos, 22% têm uma idade abaixo de 25 anos e 7% acima de 45 anos (fig. 72). Portanto isto demonstra que a maioria dos inquiridos são adultos mas relativamente jovens.

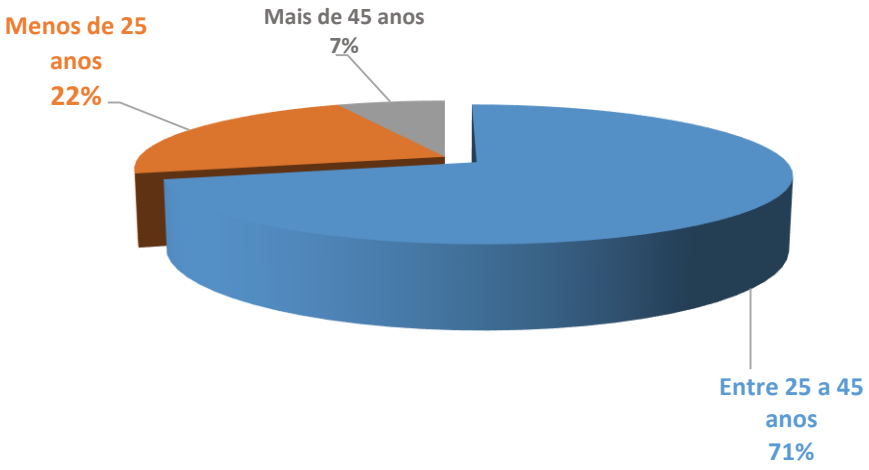

Figura 72 - Distribuição dos inquiridos dançarinos quanto á idade

Quanto á formação académica constatamos que 51% dos inquiridos têm o 12º ano de escolaridade e 24% são detentores de uma licenciatura. Os restantes inquiridos estão distribuídos da seguinte forma: 2% o 6º ano de escolaridade, 9% têm o 9º ano, 7% um bacharelato, 5% o grau de mestre e 2% são detentores de um doutoramento (fig. 73). Tratando-se de pessoas relativamente jovens percebemos que a maioria dos inquiridos têm o 12º ano de escolaridade mas que a dança não apresenta impedimentos para evolução para outros graus académicos.

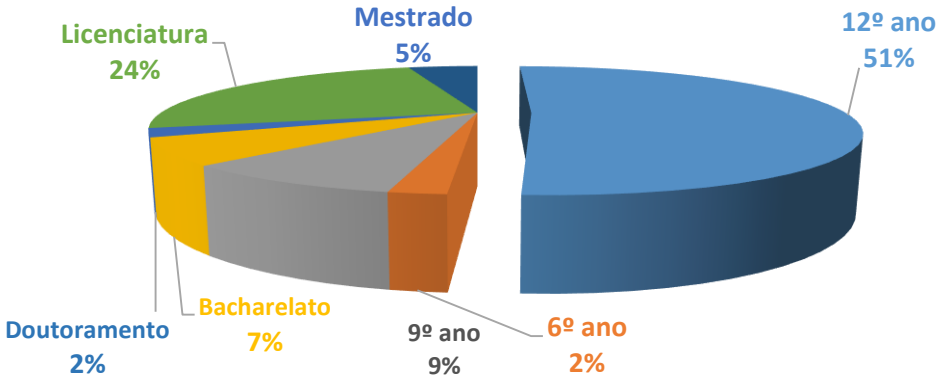

Figura 73 - Distribuição dos inquiridos dançarinos quanto às habilitações literárias

No que concerne á participação em espetáculos de dança ao vivo, foi possível constatar que 56% dos inquiridos afirma ainda não o ter feito, sendo que os restantes 44% responde afirmativamente (fig. 74).

Tratando-se de escolas e de alunos de dança é possível perceber o contexto dos dados recolhidos. De salientar que todos mostraram interesse em participar nesse tipo de espetáculos.

Figura 74 - Participação dos dançarinos inquiridos em espetáculos ao vivo

Quanto á participação dos inquiridos em escolas ou academias de dança é possível constatar que 58% respondeu que não faz parte de qualquer escola ou academia e 42% respondeu que sim. Ou seja a maioria dos inquiridos não faz parte de qualquer instituição de dança ou espetáculos (fig. 75). Estes resultados estão de acordo com os resultados da figura 74, pois nela também a maioria dos inquiridos respondeu que nunca participou num espetáculo de dança.

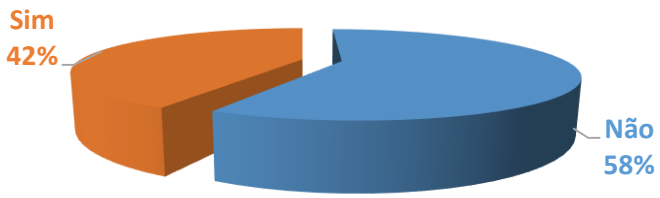

Figura 75 – Participação dos dançarinos em escolas ou academias de dança

Relativamente á facilidade de utilização do protótipo *MoveU* pode-se verificar que 97% dos inquiridos dançarinos o achou fácil (45%) ou muito fácil (53%) de utilizar. Apenas 2% referiu que o achou difícil e ninguém o considerou muito difícil (fig. 76).

Portanto é possível concluir que a maioria das pessoas achou que o protótipo *MoveU* é de uma utilização muito fácil.

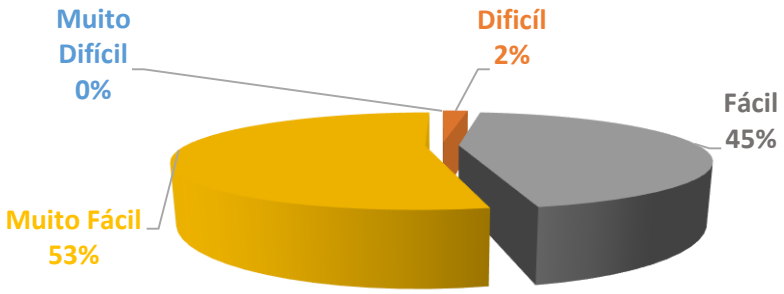

Figura 76 - Distribuição de resultados quanto á facilidade de utilização do protótipo MoveU pelos dançarinos

Quanto á opinião dos dançarinos sobre a utilização do protótipo *MoveU* no seu estado atual em espetáculos reais, pode-se atestar que 91% responderam que sim e 9% responderam que não (fig. 77). Portanto isto indica que a esmagadora maioria dos inquiridos acha que o protótipo no seu estado atual já poderia ser utilizado em situações de espetáculos reais.

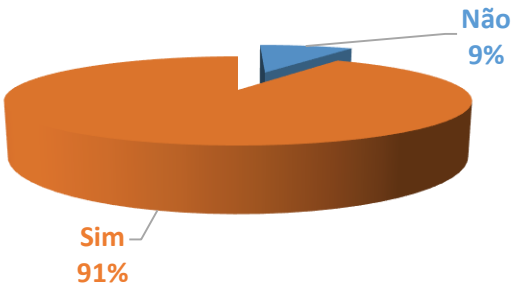

Figura 77 – Opinião dos dançarinos sobre a utilização do protótipo MoveU no seu estado atual em espetáculos reais

Quanto aos efeitos que os dançarinos mais gostaram pode-se constatar que 29% dos inquiridos respondeu o efeito *ParticleMan*, 27% o efeito *SuperFlameMan*, 24% os *UltraCubes* e 20% o *SuperSparks* (fig. 78).

Isto indica que a maioria das pessoas gostou mais do efeito *ParticleMan*, mas que apesar de tudo todos os efeitos têm percentagens bastante igualadas, o que mostra que no geral os inquiridos gostaram de todos os efeitos, mas com uma pequena preferência para o *PartiicleMan*.

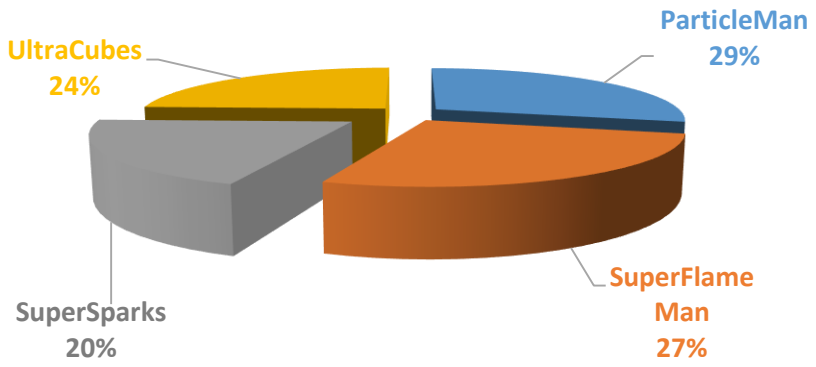

Figura 78 – Os efeitos do protótipo MoveU que mais agradaram aos dançarinos

Quanto á opinião dos dançarinos sobre se achavam que os efeitos proporcionados pelo protótipo MoveU tornariam um espetáculo protagonizado por eles mais interessante para o público que o fosse assistir, pode-se verificar que 96% dos inquiridos responderam que sim e apenas 4% responderam que não (fig. 79).

Isto mostra claramente que a maioria dos inquiridos acha que os efeitos que o protótipo *MoveU* acrescenta aos espetáculos tornam-no mais interessante para o público que o está a assistir.

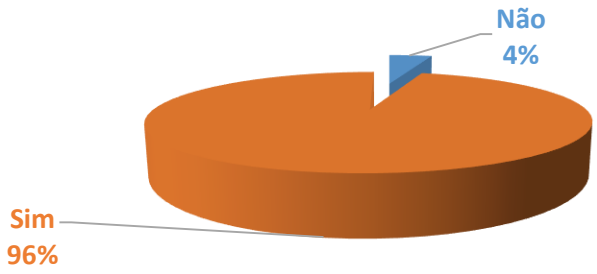

Figura 79 - Distribuição da opinião dos dançarinos sobre se achavam que os efeitos proporcionados pelo protótipo MoveU tornariam um espetáculo protagonizado por eles mesmos mais interessante para o público que o fosse assistir

Quanto á avaliação por parte dos dançarinos do protótipo *MoveU* em termos gerais pode-se verificar que 31% deu nota 9, 29% deu nota máxima de 10, 24% deu nota 8, 9% deu nota 7, 5% deu nota 6, 2% deu nota 4 e 0% deram as notas 1, 2, 3 e 5 (fig. 80). Com isto pode-se concluir que a grande maioria dos inquiridos deram notas altas ao protótipo *MoveU*, o que mostra que é do seu pleno agrado.

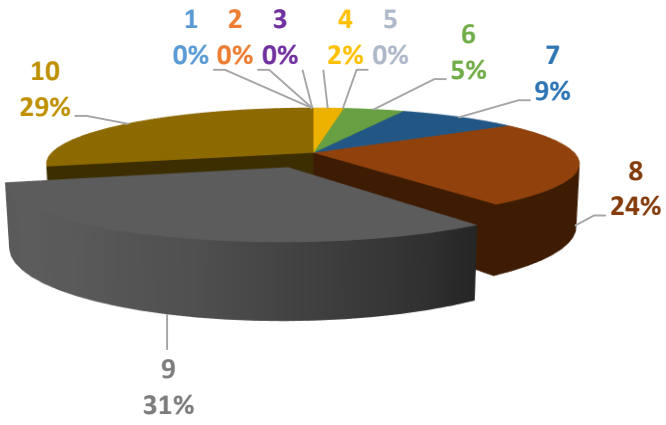

Figura 80 - Distribuição dos resultados da avaliação do protótipo MoveU pelos dançarinos

Por fim quanto á opinião dos dançarinos sobre se achavam que o protótipo poderia ser melhorado e refinado em relação ao que viram, 64% responderam que sim e 36% responderam que não (fig. 81). Isto mostra que a maioria das pessoas acha que o protótipo pode ser ainda melhorado em relação ao seu estado atual.

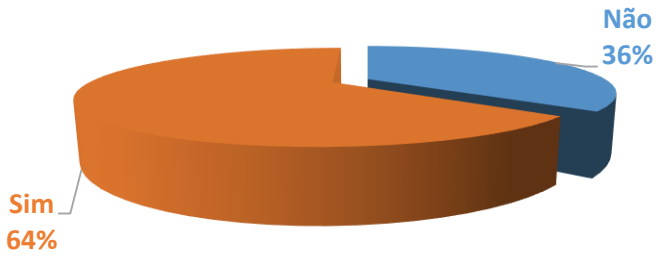

Figura 81 – Opinião dos dançarinos sobre se achavam que o protótipo poderia ser melhorado e refinado em relação ao que viram

4.2.2 Análise aos inquéritos da perspetiva do espectador

O espectador representa uma parte fulcral do sistema proposto, pois foi para este que o protótipo foi desenvolvido, isto porque o acréscimo que os efeitos proporcionam a um espetáculo é direcionado para o este. É então importante perceber a opinião dos espectadores sobre os mais variados aspetos abordados nos próximos parágrafos.

No gráfico da figura 82 pode-se observar que 53% dos inquiridos são do sexo feminino e 47% do masculino, ou seja, a maioria da população é do sexo feminino.

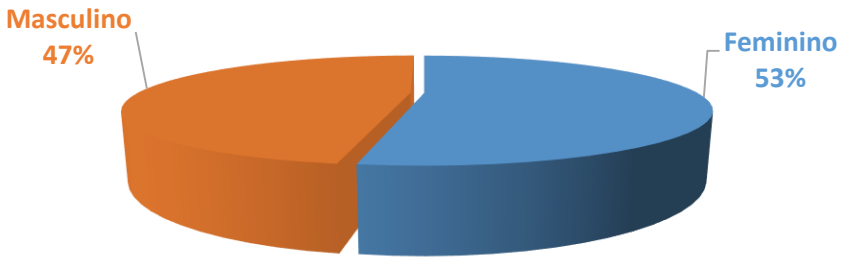

Figura 82 - Distribuição dos inquiridos espectadores quanto ao sexo

Quanto á distribuição de idades pode-se concluir que 58% dos inquiridos têm menos de 25 anos, 30% têm entre 25 a 45 anos e 12% têm mais de 45 anos (fig. 83). Portanto isto demonstra que a maioria dos inquiridos são jovens.

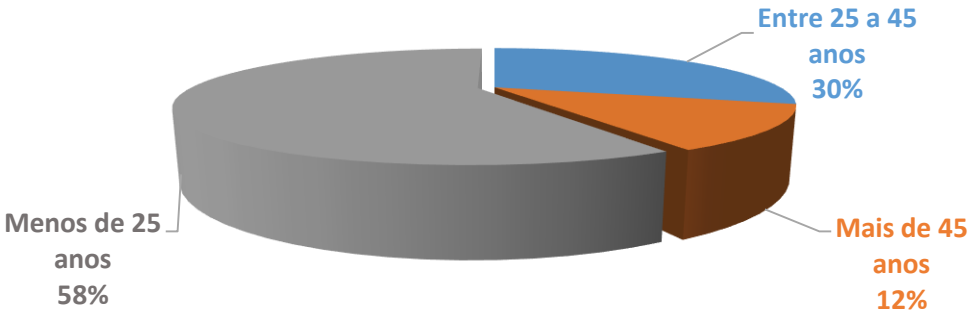

Figura 83 - Distribuição dos inquiridos espectadores quanto á idade

Nas habilitações literárias 28% dos inquiridos têm o 12º ano de escolaridade, 26% têm licenciatura, 15% têm o 6º ano, 14% têm o 9º ano, 7% o doutoramento, 5% o mestrado e 5% o bacharelato (fig. 84). Pode-se afirmar que a maioria dos inquiridos têm o 12º ano de escolaridade.

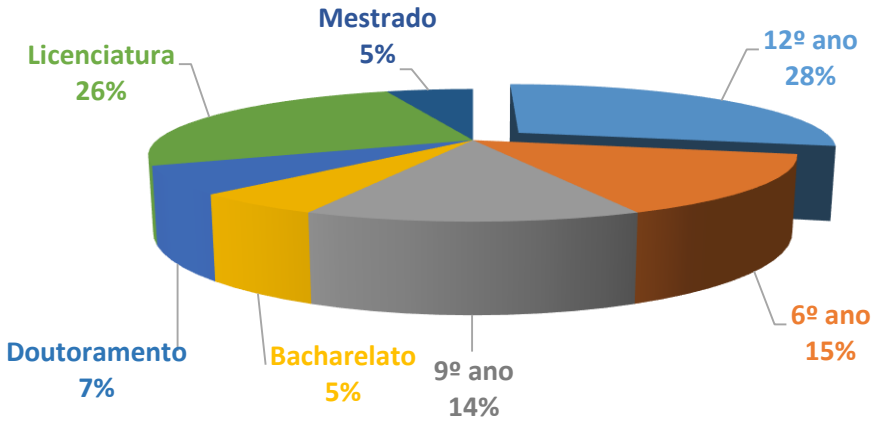

Figura 84 - Distribuição dos inquiridos espectadores quanto às habilitações literárias

Quanto aos espectadores inquiridos aos quais foi perguntado se costumavam assistir a espetáculos de dança ou a outro tipo de espetáculos ao vivo pode-se auferir que 41% dos

inquiridos responderam que assistem algumas vezes, 35% responderam que nunca assistem e 24% responderam que assistem regularmente (fig. 85). Pode-se afirmar que a maioria dos inquiridos assiste algumas vezes a espetáculos mas também existe uma grande parte de inquiridos que nunca vêm.

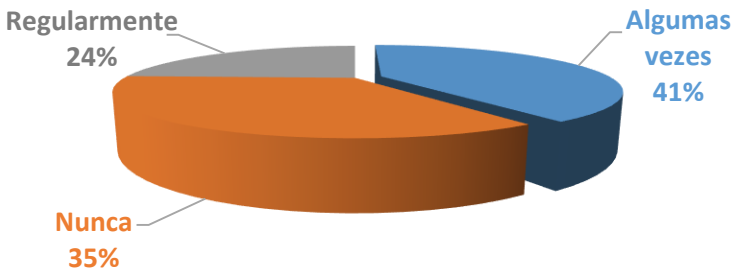

Figura 85 - Distribuição de resultados da taxa de espectadores que assiste a espetáculos de dança ou outro tipo de espetáculos ao vivo

Relativamente á facilidade de utilização do protótipo *MoveU* pode-se verificar que 43% dos inquiridos achou muito fácil de utilizar, 41% achou fácil, 11% achou difícil e 5% achou muito difícil (fig. 86). Portanto pode-se concluir que a maioria das pessoas achou que o protótipo *MoveU* é de uma utilização muito fácil.

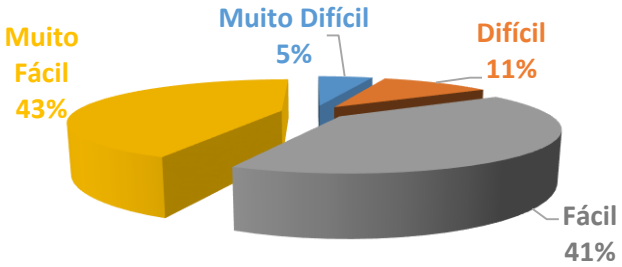

Figura 86 - Distribuição de resultados quanto á facilidade de utilização do protótipo MoveU visto pelos espectadores

Quanto á opinião dos espectadores sobre a utilização do protótipo *MoveU* no seu estado atual em espetáculos reais, pode-se atestar que 91% das pessoas responderam que sim e 9% responderam que não (fig. 87). Portanto isto indica que a esmagadora maioria dos inquiridos acha que o protótipo no seu estado atual já está pronto para ser utilizado em situações de espetáculos reais.

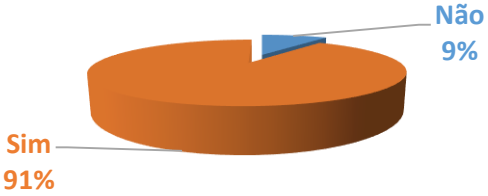

Figura 87 - Opinião dos espectadores sobre a utilização do protótipo MoveU no seu estado atual em espetáculos reais

Quanto aos efeitos que os espectadores mais gostaram pode-se constatar que 31% dos inquiridos respondeu o efeito *SuperFlameMan*, 29% o efeito *UltraCubes*, 24% o *SuperSparks* e 16% o *ParticleMan* (fig. 88). Isto indica que a maioria das pessoas gostou mais do efeito *SuperFlameMan*, mas que apesar de tudo todos os efeitos têm percentagens bastante igualadas, o que mostra que no geral os inquiridos gostaram de todos os efeitos, mas com uma pequena preferência para o *SuperFlameMan*.

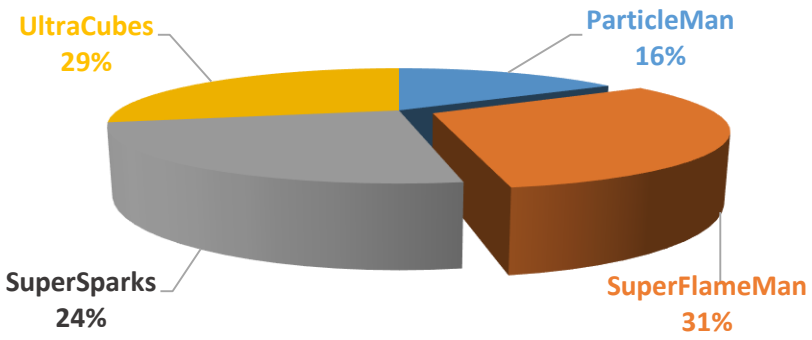

Figura 88 - Os efeitos do protótipo MoveU que mais agradaram aos espectadores

Quanto á opinião dos espectadores sobre se achavam que um espetáculo em que fosse utilizado o protótipo *MoveU* o tornaria mais interessante para eles, pode-se verificar que 86% dos inquiridos responderam que sim e 14% responderam que não (fig. 89). Isto mostra que a maioria das pessoas acha mais apelativo um espetáculo onde o protótipo *MoveU* é utilizado.

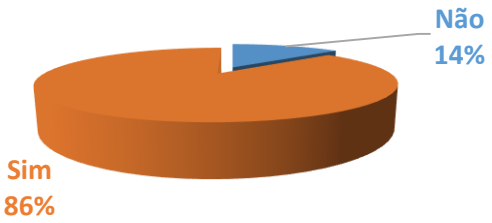

Figura 89 - Distribuição da opinião dos espectadores sobre se achavam que um espetáculo em que fosse utilizado o protótipo MoveU o tornaria mais interessante para eles

Quanto á avaliação por parte dos espectadores do protótipo *MoveU* em termos gerais pode-se verificar que 31% deu nota 10, 19% deu nota 9, 18% deu nota 8, 12% deu nota 7, 11% deu nota 6, 5% deu nota 5, 3% deu nota 4, 1% deu nota 3 e 0% deu notas 1 e 2 (fig. 90). Com isto pode-se concluir que a grande maioria dos inquiridos deram notas altas ao protótipo *MoveU*, o que mostra que é do seu pleno agrado.

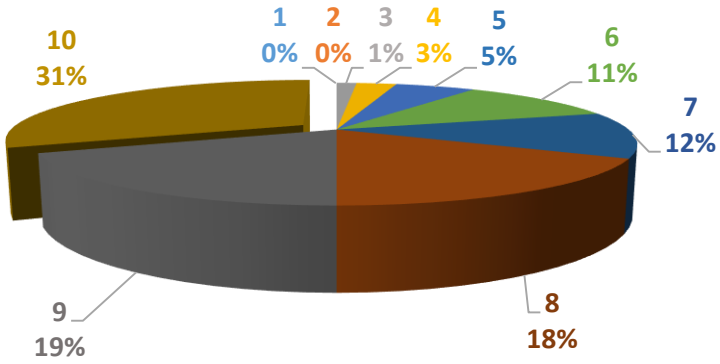

Figura 90 - Distribuição dos resultados da avaliação do protótipo MoveU pelos espectadores

Por fim quanto á opinião dos espectadores sobre se achavam que o protótipo poderia ser melhorado e refinado em relação ao que viram, 55% responderam que sim e 45% responderam que não (fig. 91). Isto mostra que a maioria das pessoas acha que o protótipo pode ser ainda melhorado em relação ao seu estado atual.

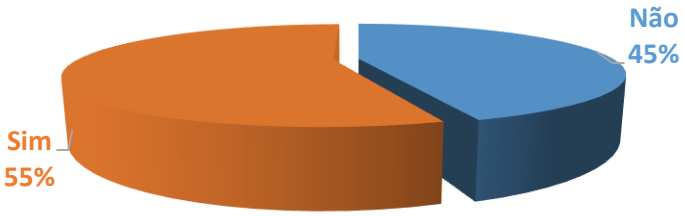

Figura 91 - Opinião dos espectadores sobre se achavam que o protótipo poderia ser melhorado e refinado em relação ao que viram

4.2.3 Análise aos inquéritos da perspetiva do controlador

O controlador representa uma parte tão importante no sistema como o dançarino e o espectador, isto porque o papel do controlador é o de escolher e dinamizar os efeitos que serão demonstrados ao longo do espetáculo. É por isso importante entender a opinião de quem controla o protótipo *MoveU* a fim de perceber se o mecanismo de controlo corresponde às necessidades dos espetáculos reais.

No gráfico da figura 92 pode-se observar que 55% dos inquiridos são do sexo masculino e 45% do feminino, ou seja, a maioria da população é do sexo masculino.

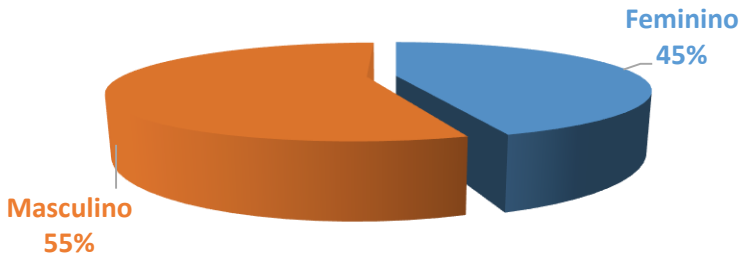

Figura 92 - Distribuição dos inquiridos controladores quanto ao sexo

Quanto á distribuição de idades pode-se concluir que 60% dos inquiridos têm entre 25 a 45 anos, 32% têm menos de 25 anos e 8% têm mais de 45 anos (fig. 93). Portanto isto demonstra que a maioria dos inquiridos são jovens.

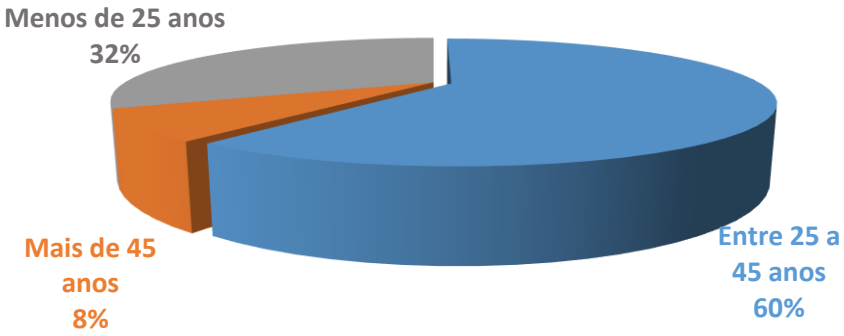

Figura 93 - Distribuição dos inquiridos controladores quanto á idade

Nas habilitações literárias 40% dos inquiridos têm o 12º ano de escolaridade, 26% têm licenciatura, 13% têm o mestrado, 8% têm o bacharelato, 5% o doutoramento, 5% o 9º ano e 3% o 6º ano (fig. 94). Pode-se afirmar que a maioria dos inquiridos têm o 12º ano de escolaridade.

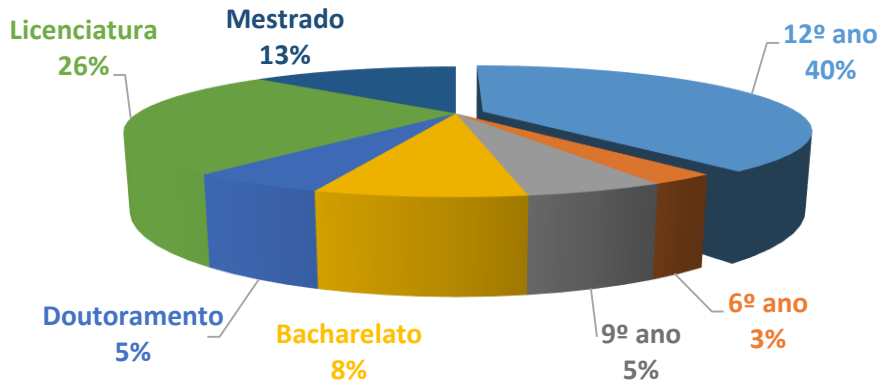

Figura 94 - Distribuição dos inquiridos controladores quanto às habilitações literárias

Relativamente á facilidade de utilização da interface de controlo do protótipo *MoveU* pode-se verificar que 43% dos inquiridos achou muito fácil de utilizar, 52% achou muito fácil de

utilizar, 45% achou fácil, 3% achou difícil e 0% achou muito difícil (fig. 95). Portanto pode-se concluir que a maioria das pessoas achou que é muito fácil controlar o protótipo *MoveU*.

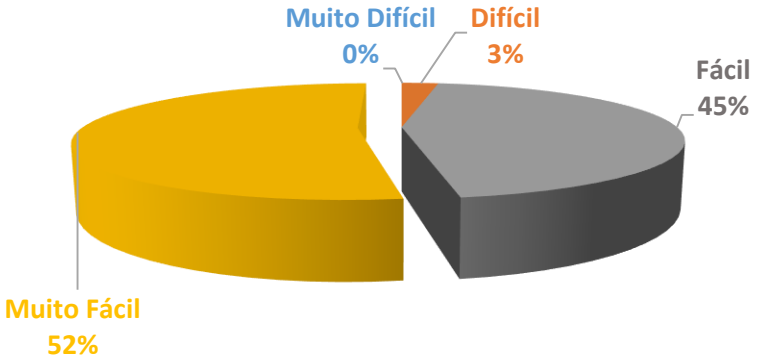

Figura 95 - Distribuição de resultados quanto á facilidade de utilização da interface de controlo do protótipo MoveU pelos inquiridos controladores

Quanto á opinião dos controladores sobre a utilização do protótipo *MoveU* no seu estado atual em espetáculos reais, pode-se atestar que 97% das pessoas responderam que sim e 3% responderam que não (fig. 96). Portanto isto indica que a esmagadora maioria dos inquiridos acha que o protótipo no seu estado atual já está pronto para ser utilizado em situações de espetáculos reais.

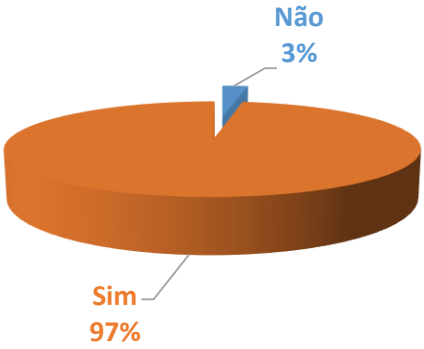

Figura 96 - Opinião dos inquiridos controladores sobre a utilização do protótipo MoveU no seu estado atual em espetáculos reais

Quanto aos efeitos que os controladores mais gostaram pode-se constatar que 31% dos inquiridos respondeu o efeito *SuperFlameMan*, 29% o efeito *UltraCubes*, 24% o *SuperSparks* e 16% o *ParticleMan* (fig. 97). Isto indica que a maioria das pessoas gostou mais do efeito *SuperFlameMan*, mas que apesar de tudo todos os efeitos têm percentagens bastante igualadas, o que mostra que no geral os inquiridos gostaram de todos os efeitos, mas com uma pequena preferência para o *SuperFlameMan*.

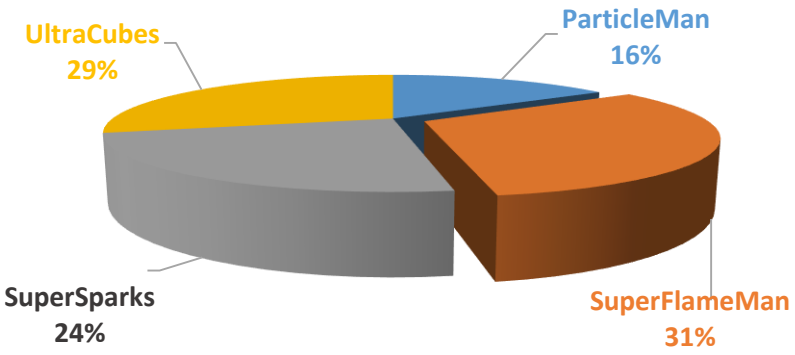

Figura 97 - Os efeitos do protótipo MoveU que mais agradaram aos controladores

Quanto á opinião dos controladores acerca da utilidade da funcionalidade que permite controlar os efeitos do protótipo *MoveU* usando um computador distinto ou remoto pode-se atestar que 52% dos inquiridos responderam que é bastante útil, 37% que é útil, 8% que é pouco útil, 3% que é muito pouco útil e 0% que não é útil (fig. 98). Isto mostra que a maioria dos inquiridos acha que a funcionalidade é de grande utilidade.

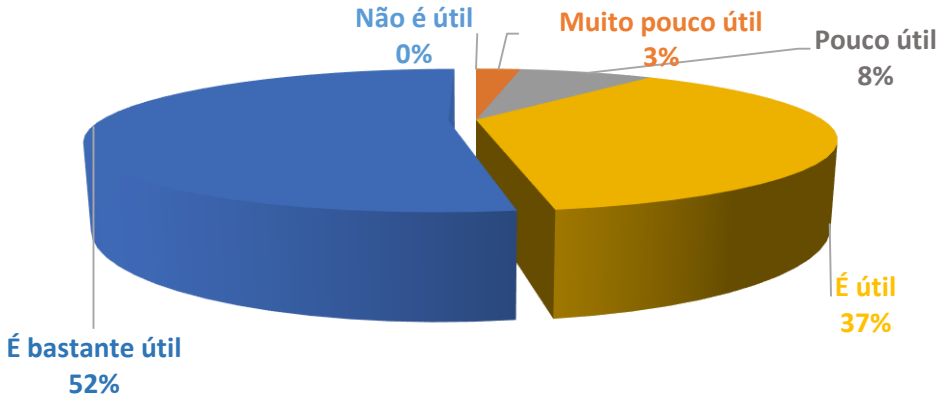

Figura 98 - Distribuição de resultados da opinião dos controladores acerca da utilidade da funcionalidade que permite controlar os efeitos do protótipo MoveU usando um computador distinto ou remoto

Quanto á avaliação por parte dos controladores do protótipo *MoveU* em termos gerais pode-se verificar que 42% deu nota 10, 21% deu nota 9, 16% deu nota 8, 16% deu nota 7, 3% deu nota 6, 2% deu nota 5, 0% deu notas 1, 2, 3 e 4 (fig. 99). Com isto pode-se concluir que a grande maioria dos inquiridos deram notas altas ao protótipo *MoveU*, o que mostra que é do seu pleno agrado.

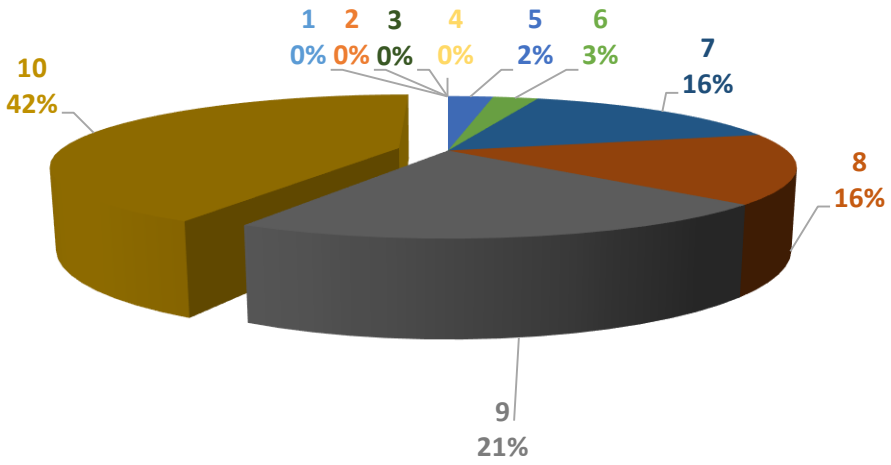

Figura 99 - Distribuição dos resultados da avaliação do protótipo MoveU pelos controladores

Por fim quanto á opinião dos controladores sobre se achavam que o protótipo poderia ser melhorado e refinado em relação ao que viram, 55% responderam que sim e 45% responderam que não (fig. 100). Isto mostra que a maioria das pessoas acha que o protótipo pode ser ainda melhorado em relação ao seu estado atual.

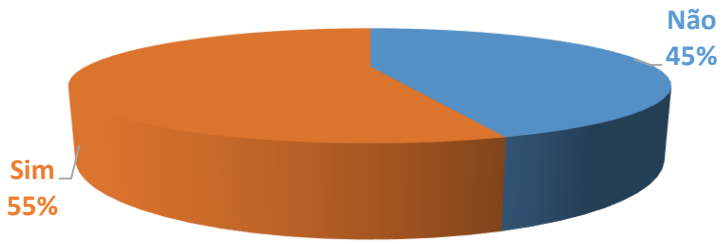

Figura 100 - Opinião dos controladores sobre se achavam que o protótipo poderia ser melhorado e refinado em relação ao que viram

Capítulo 5 – Conclusões e Trabalho Futuro

"A essência do conhecimento

consiste em aplicá-lo, uma vez possuído."

Confúcio

Neste capítulo são apresentadas as conclusões ao presente estudo refletindo sobre o trabalho realizado.

São relatados os possíveis trabalhos futuros que possam advir da realização deste estudo culminando-se o capítulo com algumas considerações finais.

5.1 Conclusões

Durante muitos anos o mundo da dança e dos espetáculos baseou-se em meros efeitos básicos e sem qualquer tipo de interação que proporciona-se um espetáculo mais interessante e interativo tanto para os espectadores como também para os artistas e bailarinos. Tendo em conta esta problemática foi decidido realizar um estudo que permitisse criar um sistema que pudesse colmatar esta falha no mundo dos espetáculos.

Para a criação desse sistema seria necessário a utilização de algum *hardware* que capturasse os movimentos dos artistas e que depois pudesse passar essa informação para o sistema que seria criado. Para decidir que *hardware* seria utilizado foi realizada uma pesquisa que abordou os diferentes sensores de movimentos presentes no mercado a fim de determinar qual seria o mais indicado para a utilização neste trabalho, tendo em conta uma diversidade de fatores distintos. Depois de ter sido analisado vários sensores, foi determinado que o mais adequado para ser utilizado neste trabalho foi o *Microsoft Kinect*.

Dado o exposto foi também feito um estudo com o intuito de perceber que outros tipos de sistemas interativos existem com o intuito de perceber se de facto existia uma lacuna no mercado em que houvesse uma necessidade de ser criado um sistema que permitisse um alto nível de interação com dançarinos e artistas. Depois de concluído o estudo percebeu-se que de facto não existia ainda nenhum sistema que realizasse o nível de interação que foi proposto para este trabalho. Para além disto foram também estudados os efeitos que são também tradicionalmente utilizados em danças e espetáculos, afim de também perceber que

de facto existe uma necessidade de criar algo que disponibilize efeitos com alto nível de interação para espetáculos.

Após ter sido realizado o estado da arte, avançou-se com a criação do sistema protótipo, mas antes disso foi necessário realizar uma análise aos mais variados *softwares* necessários para a criação do sistema. Mais uma vez, e tal como no caso da escolha do sensor de movimento, foram analisadas as diferentes vantagens e desvantagens de cada um dos *softwares* e com base nessa informação foram feitas as escolhas.

A fase da criação do sistema decorreu sem problemas a relatar, e dela surgiu o protótipo *MoveU*. Numa primeira fase, e quando ainda se estava a desenvolver o protótipo, foi feito um pequeno teste inicial com alguns alunos do Instituto Superior de Engenharia do Porto, a fim de perceber se o protótipo estava a ser desenvolvido na direção certa e se cumpriria os objetivos propostos inicialmente. Este pequeno teste foi um sucesso e então continuou-se o desenvolvimento do protótipo até este estar totalmente concluído.

Nesta fase contactou-se algumas escolas de dança para que se pudesse realizar uma serie de testes finais, sendo que neles se detetou alguns pequenos problemas, que não afetam em grande medida o funcionamento do protótipo, mas que no futuro e numa versão já final, terão de ser colmatados.

Depois de o protótipo ter sido finalizado e testado foi realizado uma serie de inquéritos para tentar perceber até que ponto todos os objetivos propostos foram conseguidos com sucesso. Assim sendo foi pedido a uma diversidade de pessoas que tiveram contacto com o protótipo que preenchessem um pequeno questionário sobre este. Com o objetivo de perceber a opinião das pessoas em diferentes perspetivas de utilização distintas, foram criados três tipos diferentes de questionários que permitiram concluir o nível de satisfação de cada individuo nas diferentes perspetivas de utilização.

Para os questionários realizados nas três perspetivas distintas pode-se concluir que no geral todos os inquiridos mostram-se satisfeitos com o protótipo e acham que é uma mais-valia para os espetáculos de dança. Além disso a maioria dos inquiridos referem que o protótipo já se encontra preparado para uma utilização em espetáculos reais. Estes também destacaram a facilidade de utilização do protótipo. Os inquiridos que responderam ao questionário na perspetiva de controlador afirmam que a funcionalidade do controlo dos efeitos de forma remota tem grande utilidade. No que diz respeito ao questionário na perspetiva do dançarino, pode-se afirmar que a grande maioria dos inquiridos acha que o protótipo tornaria um espetáculo de dança protagonizado por eles, mais interessante para o

público a que o fosse assistir. No geral todos os inquiridos referem que o protótipo pode ser melhorado em relação ao seu estado atual.

Após os resultados auferidos pela estatística realizada aos questionários respondidos pelos inquiridos pode-se concluir que no geral todos se mostraram bastante satisfeitos e interessados pelo protótipo *MoveU*.

5.2 Trabalho Futuro

Face a estes resultados, é do interesse do autor continuar com este projeto, colmatando as falhas encontradas durante os processos de testes e no futuro lançar uma versão final que esteja já preparada para ser comercializada a interessados neste tipo de tecnologias para utilização nos seus espetáculos, sendo este um produto mais vocacionado para instituições e academias de dança ou espetáculos.

Este trabalho permitiu ao autor auferir conhecimentos na área da dança e dos espetáculos e também na área da modelação tridimensional e na utilização de sensores de movimento, o que enriquece bastante o seu portefólio de conhecimentos.

Numa primeira fase, o autor perspetiva iniciar uma fase de escrita científica no sentido de dar a conhecer o seu estudo e partilhar os resultados obtidos com outras pessoas que trabalham na mesma área.

A procura de novos sensores e a produção de novos efeitos estão na mira do autor bem como o desenvolvimento de uma aplicação controladora com um interface mais apelativo e mais complexo uma vez que a fase de conceito e estudo dos aspetos funcionais do protótipo já passou.

5.3 Considerações finais

Partimos do pressuposto que seria possível tornar um espetáculo de dança ao vivo mais interativo e apelativo desde que fosse possível integrar no seu cenário elementos gráficos interativos resultantes da dança capturada por sensores em tempo real. Para o efeito desenvolvemos um protótipo que testamos juntos de escolas de dança e com bailarinos profissionais e com algumas audiências simuladas ou seja conjunto de pessoas que assistiram á demonstração do protótipo que nos transmitiram que um sistema desta natureza pode ser extremamente útil e interessante no contexto de espetáculos ao vivo de dança.

Os espectadores referiram que este tipo de abordagem transformava a atuação num espetáculo mais interativo e motivador pelo que a sua integração com os mesmos poderá tornar este tipo de espetáculos mais agradáveis. Nos bailarinos constatamos que embora a maioria salientasse o interesse do uso desta tecnologia para as suas atuações houve quem referisse que esta tecnologia o intimidava de alguma forma pois sentiu durante a sua atuação de teste com o protótipo que de algum modo perdeu protagonismo tendo percebido que os espectadores centraram mais a sua atenção nos efeitos do que propriamente na sua performance enquanto profissional de dança. De uma forma geral consideramos ter conseguido alcançar os objetivos propostos e estamos conscientes de que o produto aqui apresentado e desenvolvidos é apenas um protótipo que serve como prova de conceito ao estudo realizado e que este tipo de trabalho pode ainda ser mais elaborado e testado em ambientes de espetáculo real. Quer a interface quer o conjunto de efeitos serão alvo de intervenção senso intensão do autor reunir com profissionais da dança no sentido de poder vir a colocar esta solução no mercado.

Referências

A

[Al-Riyami, 2015] Al-Riyami F., Shopping mall store uses Kinect to captivate shoppers, http://www.winbeta.org/news/shopping-mall-store-uses-kinect-captivate-shoppers [último acesso: Mar 2015]

[Alves, 2014] Alves P., O que é e como funciona o leap motion?, http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2014/05/o-que-e-leap-motion.html [último acesso: Ago 2014]

[Armstrong, 2014] Armstrong A., OpenNI To Close, http://www.i-programmer.info/news/194-kinect/7004-openni-to-close-.html [último acesso: Set 2015]

[Asus, 2014a] Asus, Xtion PRO LIVE, https://www.asus.com/us/3D-Sensor/Xtion_PRO_LIVE/ [último acesso: Set 2014]

[Asus, 2014b] Asus, Xtion PRO LIVE, https://www.asus.com/us/3D-Sensor/Xtion_PRO_LIVE/specifications/ [último acesso: Set 2014]

B

[Barrett, 2011] Barrett E., 'Kinect Effect' Magic Pushes Beyond the Living Room, http://news.microsoft.com/2011/10/31/kinect-effect-magic-pushes-beyond-the-living-room/ [último acesso: Out 2014]

[Bedinghaus, 2015a] Bedinghaus T., Tango, http://dance.about.com/od/typesofdance/p/Tango.htm [último acesso: Out 2015]

[Bedinghaus, 2015b] Bedinghaus T., What Is Folk Dance?, http://dance.about.com/od/groupdancing/f/Folk_Dance.htm [último acesso: Out 2015]

[Brodkin, 2013] Brodkin J., How Unity3D Became a Game-Development Beast, http://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/ [último acesso: Set 2015]

C

[Creative, 2014] Creative, Creative Senz3D, http://pt.creative.com/p/web-cameras/creative-senz3d [último acesso: Out 2014]

[Crounse, 2014] Crounse B., Microsoft Kinect Sensor applications in health and medicine, http://blogs.msdn.com/b/healthblog/archive/2014/01/10/microsoft-kinect-sensor-applications-in-health-and-medicine.aspx [último acesso: Mar 2015]

D

[Danças de Salão, 2015] Danças de Salão, Cha Cha Cha, http://dancasdesalao.webnode.pt/dan%C3%A7as%20de%20salao/cha-cha-cha/ [último acesso: Out 2015]

[Diniz, 2015] Diniz T., História da Dança – Sempre, http://www.uel.br/eventos/sepech/sepech08/arqtxt/resumos-anais/ThaysDiniz.pdf [último acesso: Set 2015]

[DisplayInsight, 2013] DisplayInsight, NexNix supplies video panels for Kinect-powered interactive art installation in Schipol airport, http://www.displayinsight.com/nexnix-case-study-nexnix-supply-video-panels-for-kinect-powered-interactive-art-installation-in-schipol-airport/ [último acesso: Mar 2015]

[Duarte, 2008] Duarte M., O que é a dança contemporânea?, http://webcache.googleusercontent.com/search?q=cache:http://jpn.up.pt/2008/12/22/o-que-e-a-danca-contemporanea/&gws_rd=cr&ei=470rVqfiHMKgafDbo6AF [último acesso: Out 2015]

E

[EA, 2015] Electronic Arts, Grand Slam Tennis, http://www.ea.com/grand-slam-tennis-1 [último acesso: Set 2015]

[Eisler, 2012] Eisler C., Kinect for Windows releases SDK update and launches in China, http://blogs.msdn.com/b/kinectforwindows/archive/2012/10/08/kinect-for-windows-releases-sdk-update-and-launches-in-china.aspx [último acesso: Set 2015]

[Etherington, 2013] Etherington D., Leap Motion Launches With Limited Appeal, But It Could Be A Ticking Time Bomb Of Innovation, http://techcrunch.com/2013/07/22/leap-motion-launches-with-limited-appeal-but-it-could-be-a-ticking-time-bomb-of-innovation/ [último acesso: Set 2015]

G

[Gamers Teresina, 2012] Gamers Teresina, A historia dos sensores de movimento nos consoles, http://gamersthe.blogspot.pt/2012/07/a-historia-dos-sensores-de-movimento.html [último acesso: Ago 2014]

[Gantenbein, 2012] Gantenbein D., Kinect Launches a Surgical Revolution, http://research.microsoft.com/en-us/news/features/touchlesssurgery-060712.aspx [último acesso: Mar 2015]

[Gilbert, 2010] Gilbert B., Original Xbox 360 requires separate power connection for Kinect, http://www.engadget.com/2010/06/14/original-xbox-360-requires-separate-power-for-kinect/ [último acesso: Set 2015]

[Globo, 2010] Globo, https://www.youtube.com/watch?v=1JdWomEyyr0 [último acesso: Ago 2014]

[Gorman, 2013] Gorman M., ASUS partners up with Leap Motion, PCs with 3D motion control to debut in 2013, http://www.engadget.com/2013/01/03/asus-leap-motion-partnership/ [último acesso: Set 2015]

[Greenberg, 2012] Greenberg J., The Best NES Accessories That Failed Big Time, http://ezinearticles.com/?The-Best-NES-Accessories-That-Failed-Big-Time&id=7386669 [último acesso: Out 2015]

H

[Harmonix, 2015a] Harmonix, Dance Central, http://www.harmonixmusic.com/games/dance-central/ [último acesso: Set 2015]

[Harmonix, 2015b] Harmonix, EyeToy: AntiGrav, https://www.playstation.com/pt-pt/games/eyetoy-antigrav-ps2/ [último acesso: Set 2015]

[Hollister, 2011] Hollister S., PrimeSense and ASUS team, bring Kinect-like Wavi Xtion to your living room TV, http://www.engadget.com/2011/01/03/primesense-and-asus-team-bring-kinect-like-wavi-xtion-to-your-h/ [último acesso: Set 2015]

[Hutchinson, 2013] Hutchinson L., Hands-on with the Leap Motion Controller: Cool, but frustrating as hell, http://arstechnica.com/gadgets/2013/07/hands-on-with-the-leap-motion-controller-cool-but-frustrating-as-hell/ [último acesso: Set 2015]

I

[Ipisoft, 2013], Ipisoft, Depth Sensors Comparison, http://wiki.ipisoft.com/Depth_Sensors_Comparison [último acesso: Out 2014]

J

[JCDecaux, 2013] JCDecaux, MTR Advertising : PrimeCredit showed its Commitment via Multimedia Panel with Interactive Game at MTR Mong Kok Station, http://www.jcdecaux-transport.com.hk/video/primecredit-showed-its-commitment-via-multimedia-panel-with-interactive-gam/123/P40 [último acesso: Mar 2015]

K

[Khan, 2015] Khan E., 10 Most Famous Dance Styles in the World, http://www.wonderslist.com/10-most-famous-dance-styles-in-the-world/ [último acesso: Set 2015]

[KinectHacks.net, 2010] KinectHacks.net, Tracking on arbitrary planes with Kinect, https://www.youtube.com/watch?v=zKQ108YSYY0&list=UUUHuUNpf3nafsc45eZjZUIg [último acesso: Ago 2014]

[Kleina, 2013] Kleina N., Kinect pode ver dentro de sua cabeça e ajudar na hora da cirurgia, http://www.tecmundo.com.br/medicina/37466-kinect-pode-ver-dentro-de-sua-cabeca-e-ajudar-na-hora-da-cirurgia-video-.htm [último acesso: Mar 2015]

L

[Lact tin, 2013] Lact tin, https://www.youtube.com/watch?v=7_lsCB2TdJc [último acesso: Ago 2014]

[Landim, 2013] Landim W., Análise: ASUS Xtion Motion Sensor, http://www.tecmundo.com.br/asus/42660-analise-asus-xtion-motion-sensor.htm [último acesso: Out 2014]

[Leapmotion, 2014a] Leapmotion, Leapmotion Airspace, https://airspace.leapmotion.com/ [último acesso: Set 2014]

[Leapmotion, 2014b] Leapmotion, Leapmotion Developers Portal, https://developer.leapmotion.com/ [último acesso: Set 2014]

[Leapmotion, 2014c] Leapmotion, Leapmotion, https://www.leapmotion.com/product [último acesso: Set 2014]

[Leapmotion, 2013d] Leapmotion, Preview: Leap Motion Apps + Airspace™, http://blog.leapmotion.com/preview-leap-motion-apps-airspace-tm/ [último acesso: Set 2015]

[Leapmotion, 2015e] Leapmotion, Leapmotion Store, http://store-eur.leapmotion.com/ [último acesso: Set 2015]

[Lee, 2013] Lee D., Leap Motion seals HP deal to embed gesture control technology, http://www.bbc.com/news/technology-22166424 [último acesso: Set 2015]

[Leite, 2015] Leite A., Iniciação á Dança Contemporânea, http://fulloutda.weebly.com/iniciaccedilatildeo-agrave-danccedila-contemporacircnea.html [último acesso: Out 2015]

[Loclair, 2013] Loclair C., flow no. 1 | kinect projector dance, http://princemio.net/portfolio/flow-1-kinect-projector-dance/ [último acesso: Out 2015]

[Lopes, 2015] Lopes V., Nomes de passos de Ballet, http://artes.umcomo.com.br/articulo/nomes-de-passos-de-ballet-7034.html [último acesso: Out 2015]

M

[MacCormick, 2015] MacCormick J., How does the Kinect work?, http://users.dickinson.edu/~jmac/selected-talks/kinect.pdf [último acesso: Set 2015]

[McEntegart, 2014] McEntegart J., Kinect Sensors Used to Enhance Oculus Rift Experience, http://www.tomshardware.com/news/oculus-rift-3d-body-tracking,26778.html [último acesso: Ago 2014]

[Mentis, 2012] Mentis H., Connect: Kinect Interactive Dance Performance, http://research.microsoft.com/apps/video/default.aspx?id=161884 [último acesso: Mar 2015]

[Microsoft, 2014a] Microsoft, Microsoft Kinect Dev Center, http://www.microsoft.com/en-us/kinectforwindows/ [último acesso: Ago 2014]

[Microsoft, 2014b] Microsoft, http://web.archive.org/web/20140720183041/http://www.microsoft.com/en-us/download/details.aspx?id=43661 [último acesso: Out 2014]

[Mitchell, 2010] Mitchell R., PrimeSense releases open source drivers, middleware that work with Kinect, http://www.engadget.com/2010/12/10/primesense-releases-open-source-drivers-middleware-for-kinect/ [último acesso: Set 2015]

[MIT Technology Review, 2015] MIT Technology Review, 50 Disruptive Companies, http://www2.technologyreview.com/tr50/primesense/ [último acesso: Set 2015]

[Mono, 2015a] Mono, Mono, http://www.mono-project.com/ [último acesso: Set 2015]

[Mono, 2015b] Mono, Companies using Mono, http://www.mono-project.com/docs/about-mono/showcase/companies-using-mono/ [último acesso: Set 2015]

[MSDN, 2014a] Microsoft MSDN, Kinect for Windows SDK, http://msdn.microsoft.com/en-us/library/hh855347.aspx [último acesso: Ago 2014]

[MSDN, 2014b] Microsoft MSDN, Kinect for Windows Programming Guide, http://msdn.microsoft.com/en-us/library/hh855348.aspx [último acesso: Out 2014]

[MSDN, 2014c] Microsoft MSDN, Coding4fun, http://channel9.msdn.com/coding4fun/kinect [último acesso: Ago 2014]

[MSDN, 2014d] Microsoft MSDN, Web Services Description Language Tool, http://msdn.microsoft.com/en-us/library/7h3ystb6(VS.80).aspx [último acesso: Nov 2014]

[MSDN, 2015e] Microsoft MSDN, Kinect for Windows Sensor Components and Specifications, https://msdn.microsoft.com/en-us/library/jj131033.aspx [último acesso: Set 2015]

[MSDN, 2015f] Microsoft MSDN, Getting Started with XNA Game Studio Development, https://msdn.microsoft.com/en-us/library/bb203894.aspx [último acesso: Set 2015]

[MSDN, 2009g] Microsoft MSDN, XNA Frequently Asked Questions, http://wayback.archive.org/web/20090908145646/http://msdn.microsoft.com/en-us/xna/aa937793.aspx [último acesso: Set 2015]

N

[Norris, 2014] Norris S., 13 near-useless video game accessories, http://www.timeslive.co.za/scitech/2014/05/22/13-near-useless-video-game-accessories [último acesso: Out 2015]

P

[Panasonic, 2014a] Panasonic, Panasonic D-IMager, http://www2.panasonic.biz/es/densetsu/device/3DImageSensor/en/index.html [último acesso: Set 2014]

[Panasonic, 2014b] Panasonic, Technology, http://www2.panasonic.biz/es/densetsu/device/3DImageSensor/en/tech.html [último acesso: Set 2014]

[Panasonic, 2014c] Panasonic, Beckon SDK Bundle, http://www2.panasonic.biz/es/densetsu/device/3DImageSensor/en/bundle.html [último acesso: Set 2014]

[Passo Base, 2015a] Passo Base, Salsa, a dança sensual, http://passobase.com/artigos/salsa-danca-sensual [último acesso: Out 2015]

[Passo Base, 2015b] Passo Base, Vamos dançar o tango?, http://passobase.com/artigos/vamos-dancar-o-tango [último acesso: Out 2015]

[Passo Base, 2015c] Passo Base, Chá-Chá-Chá sem parar!, http://passobase.com/artigos/cha-cha-cha-sem-parar [último acesso: Out 2015]

R

[Redação Olhar Digital, 2012] Redação Olhar Digital, Kinect é utilizado em salas cirúrgicas de Londrina, PR, http://olhardigital.uol.com.br/noticia/kinect-e-utilizado-em-salas-cirurgicas-em-londrina,-pr/25504 [último acesso: Mar 2015]

[Rodrigues, 2015] Rodrigues D., http://edrodrigues.com.br/blog/as-phyx-i-a-um-espetaculo-de-danca-capturada-com-um-sensor-kinect-e-visualizado-com-software-3d/ [último acesso: Mar 2015]

[Rose, 2013] Rose M., It's official: XNA is dead, http://www.gamasutra.com/view/news/185894/Its_official_XNA_is_dead.php [último acesso: Out 2014]

[Rouse, 2011] Rouse M., motion gaming (motion-controlled gaming), http://whatis.techtarget.com/definition/motion-gaming-motion-controlled-gaming [último acesso: Mar 2015]

[Royal Institute of Technology, 2011] Royal Institute of Technology, Kinect + Google Earth!, https://www.youtube.com/watch?v=5vYnwiCkk-g [último acesso: Ago 2014]

S

[Siegal, 2014] Siegal J., Medical researchers are finding incredible uses for Microsoft's Kinect, http://news.yahoo.com/medical-researchers-finding-incredible-uses-microsoft-kinect-213152507.html [último acesso: Mar 2015]

[Silva, 2011] Silva R., Kinect ajuda cirurgiões a operar pacientes com câncer, https://tecnoblog.net/60254/kinect-ajuda-cirurgioes-a-operar-pacientes-com-cancer/ [último acesso: Mar 2015]

[Structure, 2014] Structure, Structure, http://structure.io/openni [último acesso: Out 2014]

T

[Taborda, 2012] Taborda C., Hospital brasileiro usa Kinect para fazer cirurgias, http://exame.abril.com.br/tecnologia/noticias/hospital-brasileiro-usa-kinect-para-fazer-cirurgias [último acesso: Mar 2015]

[The Latin World, 2015] The Latin World, Rueda de Cacino, http://www.thelatinworld.nl/rueda.html [último acesso: Out 2015]

[Tipos de Dança, 2015] Tipos de Dança, Tipos de Dança, http://tipos-de-danca.info/ [último acesso: Out 2015]

U

[Unity, 2014a] Unity, Skybox, http://docs.unity3d.com/Manual/class-Skybox.html [último acesso: Nov 2014]

[Unity, 2014b] Unity, Camera, http://docs.unity3d.com/Manual/class-Camera.html [último acesso: Nov 2014]

[Unity, 2014c] Unity, GUI Layer (Legacy), http://docs.unity3d.com/Manual/class-GUILayer.html [último acesso: Nov 2014]

[Unity, 2014d] Unity, Flare Layer, http://docs.unity3d.com/Manual/class-FlareLayer.html [último acesso: Nov 2014]

[Unity, 2014e] Unity, Audio Listener, http://docs.unity3d.com/Manual/class-AudioListener.html [último acesso: Nov 2014]

[Unity, 2014f] Unity, Occlusion Culling, http://docs.unity3d.com/Manual/OcclusionCulling.html [último acesso: Nov 2014]

[Unity, 2014g] Unity, Color.Lerp, http://docs.unity3d.com/ScriptReference/Color.Lerp.html [último acesso: Nov 2014]

[Unity, 2014h] Unity, Light, http://docs.unity3d.com/Manual/class-Light.html [último acesso: Nov 2014]

[Unity, 2014i] Unity, Lens Flare, http://docs.unity3d.com/Manual/class-LensFlare.html [último acesso: Nov 2014]

[Unity, 2014j] Unity, Prefabs, http://docs.unity3d.com/Manual/Prefabs.html [último acesso: Dez 2014]

[Unity, 2014k] Unity, Ellipsoid Particle Emitter (Legacy), http://docs.unity3d.com/Manual/class-EllipsoidParticleEmitter.html [último acesso: Dez 2014]

[Unity, 2014l] Unity, Particle Animator (Legacy), http://docs.unity3d.com/Manual/class-ParticleAnimator.html [último acesso: Dez 2014]

[Unity, 2014m] Unity, Particle Renderer (Legacy), http://docs.unity3d.com/Manual/class-ParticleRenderer.html [último acesso: Dez 2014]

[Unity, 2014n] Unity, Trail Renderer, http://docs.unity3d.com/Manual/class-TrailRenderer.html [último acesso: Dez 2014]

[Unity, 2014o] Unity, Renderer.useLightProbes, http://docs.unity3d.com/ScriptReference/Renderer-useLightProbes.html [último acesso: Dez 2014]

[Unity, 2015p] Unity, Multiplataforma, http://unity3d.com/pt/unity/multiplatform/ [último acesso: Set 2015]

[Unity, 2015q] Unity, Unity, http://unity3d.com/pt/unity [último acesso: Set 2015]

[Unity, 2015r] Unity, Unity Editor, http://unity3d.com/pt/unity/editor [último acesso: Set 2015]

V

[Vochin, 2010] Vochin A., Panasonic Develops Kinect 'Killer', the D-IMager, http://news.softpedia.com/news/Panasonic-Develops-Kinect-Killer-the-D-IMager-169425.shtml [último acesso: Set 2014]

[Vsauce, 2010] Vsauce, 12 BEST Kinect HACKS, https://www.youtube.com/watch?v=ho8KVOe_y08&noredirect=1_[último acesso: Ago 2014]

W

[Weichert et al., 2013] Weichert F., Bachmann D., Rudak B., Fisseler D., Analysis of the Accuracy and Robustness of the Leap Motion Controller, https://ls7-www.cs.uni-dortmund.de/publications/sensors-13-06380.pdf [último acesso: Set 2015]

[Williams, 2013] Williams A., Hands-on with the Creative Senz3D, http://www.gizmag.com/hands-on-with-the-creative-senz3d/28975/ [último acesso: Out 2014]

Y

[Yoon, 2011] Yoon K., Kinect Illusion, https://www.youtube.com/watch?v=5Qjl4BQw8qw&feature=youtu.be [ultimo acesso: Out 2015]

Z

[Zigfu, 2014] Zigfu, ZDK For Unity3D, http://zigfu.com/en/zdk/unity3d/ [último acesso: Out 2014]

Anexos

Nesta secção foram colocados os anexos relacionados com o presente estudo.

Anexo 1 - Questionário MoveU para o espetador:

Questionário MoveU

Um breve questionário sobre o protótipo MoveU que acabou de ver

*Obrigatório

1. **1 - Indique qual é o seu sexo** *
   Marcar apenas uma oval.
   - ◯ Masculino
   - ◯ Feminino

2. **2 - Indique qual é a sua idade** *
   Marcar apenas uma oval.
   - ◯ Menos de 25 anos
   - ◯ Entre 25 a 45 anos
   - ◯ Mais de 45

3. **3 - Indique as suas habilitações literárias** *
   Marcar apenas uma oval.
   - ◯ 6º ano
   - ◯ 9º ano
   - ◯ 12º ano
   - ◯ Bacharelato
   - ◯ Licenciatura
   - ◯ Mestrado
   - ◯ Doutoramento

4. **4 - Costuma assistir a espectáculos de dança ou a outro tipo de espectáculos com regularidade?** *
   Marcar apenas uma oval.
   - ◯ Nunca
   - ◯ Algumas vezes
   - ◯ Regularmente

5. **5 - Relativamente ao protótipo MoveU que teve oportunidade de ver, o que achou da facilidade de utilização?** *
   Marcar apenas uma oval.

	1	2	3	4	
Muito Difícil	◯	◯	◯	◯	Muito Fácil

6. **6 - No presente estado actual acha que o protótipo MoveU se encontra já pronto para uma utilização em espectáculos reais?** *
   Marcar apenas uma oval.

   - ◯ Sim
   - ◯ Não

7. **7 - Dos efeitos que teve oportunidade de ver, qual foi o que mais gostou?** *
   Marcar apenas uma oval.

   - ◯ ParticleMan (efeito do corpo "fantasma")
   - ◯ SuperSparks (efeito das estrelas brilhantes)
   - ◯ SuperFlameMan (efeito do homem de fogo)
   - ◯ UltraCubes (efeito dos cubos)

8. **8 - Se assistisse a um espectáculo em que fosse utilizado o protótipo MoveU, acha que isso tornaria o espectáculo mais apelativo para si?** *
   Marcar apenas uma oval.

   - ◯ Sim
   - ◯ Não

9. **9 - Em termos gerais, se tivesse que atribuir uma nota de 1 a 10 ao MoveU, qual seria a nota que daria?** *
   Marcar apenas uma oval.

1	2	3	4	5	6	7	8	9	10
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

10. **10 - Acha que o protótipo MoveU poderia ser melhorado e refinado em relação ao que viu?** *
    Marcar apenas uma oval.

    - ◯ Sim
    - ◯ Não

Anexo 2 - Questionário MoveU para o dançarino:

Questionário MoveU

Um breve questionário sobre o protótipo MoveU que acabou de experimentar

*Obrigatório

1. **1 - Indique qual é o seu sexo** *
   Marcar apenas uma oval.
   - ◯ Masculino
   - ◯ Feminino

2. **2 - Indique qual é a sua idade** *
   Marcar apenas uma oval.
   - ◯ Menos de 25 anos
   - ◯ Entre 25 a 45 anos
   - ◯ Mais de 45

3. **3 - Indique as suas habilitações literárias** *
   Marcar apenas uma oval.
   - ◯ 6º ano
   - ◯ 9º ano
   - ◯ 12º ano
   - ◯ Bacharelato
   - ◯ Licenciatura
   - ◯ Mestrado
   - ◯ Doutoramento

4. **4 - Já participou em algum tipo de espectáculo de dança?** *
   Marcar apenas uma oval.
   - ◯ Sim
   - ◯ Não

5. **5 - Faz parte de alguma escola ou associação de dança ou espectáculos?** *
   Marcar apenas uma oval.
   - ◯ Sim
   - ◯ Não

6. **6 - Relativamente ao protótipo MoveU que teve oportunidade de experimentar, o que achou da facilidade de utilização?** *

Marcar apenas uma oval.

	1	2	3	4	
Muito Difícil	◯	◯	◯	◯	Muito Fácil

7. **7 - No presente estado actual acha que o protótipo MoveU se encontra já pronto para uma utilização em espectáculos reais?** *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

8. **8 - Dos efeitos que teve oportunidade de experimentar, qual foi o que mais gostou?** *

Marcar apenas uma oval.

- ◯ ParticleMan (efeito do corpo "fantasma")
- ◯ SuperSparks (efeito das estrelas brilhantes)
- ◯ SuperFlameMan (efeito do homem de fogo)
- ◯ UltraCubes (efeito dos cubos)

9. **9 - Acha que os efeitos do protótipo MoveU tornariam um espectáculo de dança protagonizado por si mais interessante para o publico que o fosse assistir?** *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

10. **10 - Em termos gerais, se tivesse que atribuir uma nota de 1 a 10 ao MoveU, qual seria a nota que daria?** *

Marcar apenas uma oval.

1	2	3	4	5	6	7	8	9	10
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

11. **11 - Acha que o protótipo MoveU poderia ser melhorado e refinado em relação ao que viu?** *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

Anexo 3 - Questionário MoveU para o controlador:

Questionário MoveU

Um breve questionário sobre o protótipo MoveU que acabou de experimentar

*Obrigatório

1. **1 - Indique qual é o seu sexo** *
   Marcar apenas uma oval.
   - ◯ Masculino
   - ◯ Feminino

2. **2 - Indique qual é a sua idade** *
   Marcar apenas uma oval.
   - ◯ Menos de 25 anos
   - ◯ Entre 25 a 45 anos
   - ◯ Mais de 45

3. **3 - Indique as suas habilitações literárias** *
   Marcar apenas uma oval.
   - ◯ 6º ano
   - ◯ 9º ano
   - ◯ 12º ano
   - ◯ Bacharelato
   - ◯ Licenciatura
   - ◯ Mestrado
   - ◯ Doutoramento

4. **4 - Relativamente à interface de controlo do protótipo MoveU que teve oportunidade de experimentar, o que achou da facilidade de utilização?** *
   Marcar apenas uma oval.

	1	2	3	4	
Muito Difícil	◯	◯	◯	◯	Muito Fácil

5. **5 - No presente estado actual acha que o protótipo MoveU se encontra já pronto para uma utilização em espectáculos reais?** *
   Marcar apenas uma oval.
   - ◯ Sim
   - ◯ Não

6. **6 - Dos efeitos que teve oportunidade de controlar, qual foi o que mais gostou?** *
Marcar apenas uma oval.

- ParticleMan (efeito do corpo "fantasma")
- SuperSparks (efeito das estrelas brilhantes)
- SuperFlameMan (efeito do homem de fogo)
- UltraCubes (efeito dos cubos)

7. **7 - O que acha da funcionalidade que permite controlar os efeitos do protótipo MoveU usando um computador distinto ou remoto?** *
Marcar apenas uma oval.

	1	2	3	4	5	
Não é útil						É bastante útil

8. **8 - Em termos gerais, se tivesse que atribuir uma nota de 1 a 10 ao MoveU, qual seria a nota que daria?** *
Marcar apenas uma oval.

1	2	3	4	5	6	7	8	9	10

9. **9 - Acha que o protótipo MoveU poderia ser melhorado e refinado em relação ao que viu?** *
Marcar apenas uma oval.

- Sim
- Não

Anexo 4 - Cópia do email enviado para os inquiridos acerca do questionário sobre o protótipo MoveU:

Boa tarde,

Como se devem recordar, ontem (dia 21/07/2015) foi demonstrado na Academia Danciart um sistema de efeitos que interagem com a dança dos utilizadores, sistema esse que teve a oportunidade de ver e experimentar. Este sistema foi criado como meu projeto de dissertação/tese e para a sua conclusão preciso da sua assistência na resposta a um pequeno questionário constituído por algumas perguntas que visam conhecer a sua opinião sobre o sistema. Este questionário não demorará mais de 2/3 minutos a responder e é totalmente anónimo.

Este questionário é dividido em duas partes, a primeira é referente ao utilizador que experimentou o sistema e o outro é referente a quem assiste a um espetáculo em que o sistema é utilizado. Ambas as partes têm algumas perguntas em comum e tendo em conta que todos os envolvidos participaram tanto como utilizador do sistema como também assistente do espetáculo, agradecia que respondesse a ambas as partes.

Deixo de seguida os links para cada uma das partes e desde já agradeço a sua colaboração.

Questionário Parte 1 (para quem experimentou e utilizou o sistema)

- https://docs.google.com/forms/d/1mlQmBOVS9S-ciq36YIHi-D5jN3SusmL3v8raZBWAxQM/viewform?usp=send_form

Questionário Parte 2 (para quem assistiu e viu outros a usarem e experimentarem o sistema)

- https://docs.google.com/forms/d/1bMtQAKnIdkaimLQl6m0Lm16XSq_9UCufoxOdQC7d_fU/viewform?usp=send_form

Agradeço desde já a vossa colaboração.

Com os meus melhores cumprimentos,

Tiago Gonçalves