Projeto

Mestrado em Engenharia Informática - Computação Móvel

Realidade Aumentada Aplicada ao Ensino Pré-escolar

João Pedro dos Santos Nobre Órfão

Leiria, 30 de Setembro de 2014

Projeto

Mestrado em Engenharia Informática - Computação Móvel

Realidade Aumentada Aplicada ao Ensino Pré-escolar

João Pedro dos Santos Nobre Órfão

Dissertação de Mestrado realizada sob a orientação do Doutor Luis Marcelino, Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria e co-orientação do Doutor Filipe Pinto, Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, 30 de Setembro de 2014

À minha família

Agradecimentos

A realização deste projeto não teria sido possível sem a ajuda do meu coordenador ao qual agradeço todo o apoio prestado, a disponibilidade e entrega. Queria agradecer também aos meus colegas, Luís Trigueiro e Nelson Ramos, que trabalharam juntamente comigo na bolsa de investigação VisualYzart. Sem o seu trabalho e apoio não teria sido possível concluir este projeto dentro do prazo estipulado. Agradeço também à YDreams pela disponibilização da framework YVision e pela relação de proximidade que permitiu ultrapassar muitas barreiras a nível da implementação. Um muito obrigado aos pais, educadores e claro, às crianças que participaram nos testes, sem eles não seria possível validar o trabalho desenvolvido. Por fim, não podia deixar de agradecer à minha família que me apoiou não só neste projeto, como também durante todo o percurso que levei para chegar até ele.

Resumo

Neste projeto foi abordado o tema da Realidade Aumentada e a sua aplicação no ensino a crianças em idade pré-escolar. Baseado nos desenvolvimentos já efetuados na área, os objetivos deste trabalho foram verificar a capacidade da framework YVision em produzir jogos que recorressem à realidade aumentada e validar a sua utilização na educação de crianças com idades compreendidas entre os 5 e 6 anos. Controlos pouco fiáveis, pouca tolerância na deteção dos movimentos, imaginários desinteressantes e destuantes da realidade das crianças, podem comprometer a experiência desta tecnologia. Por isso foram tidos em conta, na implementação dos jogos, os temas abordados e a jogabilidade de modo a cativar e despertar o interesse das crianças. Os resultados obtidos comprovaram a capacidade da framework e demonstraram que a realidade aumentada pode ser um bom complemento para as técnicas de ensino clássicas. Isto foi observado através da facilidade de adoção deste modo de interação por parte dos utilizadores e o conhecimento demonstrado após a utilização dos jogos desenvolvidos. Como este projeto não representa uma abordagem completa ao tema, identificámos também o que poderá ser implementado no futuro de forma a melhorar e massificar esta ideia.

Palavras-chave: (*Realidade Aumentada, Ensino, Jogos, YVision, Unity*)

Abstract

In this project we approach Augmented Reality and its application on education of preschoolers. Based in previous studies already taken in this subject, the main goals of this work were verifying the capacity of the framework YVsion in implementing games that use Augmented Reality and validate their usage in the education of children between 5 and 6 years old. Sluggish controls, low tolerance in movement detection, unattractive and unfamiliar themes to children could compromise the experience of this technology. Because of that, the subjects approached and the playability were considered durind the development fase. The results obtained proved the ability of YVision and demonstrated that Augmented Reallity could be a complementary method to today's educational system. This was concluded by observing the easily adoption of this interaction mode by the children and the knowledge shown after playing the developed games. As this project don't represent a complete apraoch to this subject, we also identified possible future work that can improve and expand this concept.

Keywords: (*Augmented Reality, Education, Games, YVision, Unity*)

Lista de Figuras

Lista de Tabelas

Lista de Siglas

CCI Child Computer Interaction

GPS Global Positioning System

HCI Human Computer Interaction

IDE Ambiente Integrado de Desenvolvimento

I&DT Sistema de Incentivos à Investigação e Desenvolvimento Tecnológico nas Empresas

IPL Instituto Politécnico de Leiria

IPS Instituto Politécnico de Santarém

NFT Natural Feature Tracking

NUI Interfaces Naturais para Utilizadores

PLU PLayer Learner User

PLU-E Playing Learning and Using for Evaluation

RA Realidade Aumentada

RFID Radio-Frequency Identification

RV Realidade virtual

SAR Spatial Augmented Reality

SDK Software Development Kit

UC Universidade Católica Portuguesa

UCP Universidade de Coimbra

VA Virtualidade Aumentada

Índice

Capítulo 1

Introdução

Hoje em dia e com a evolução que temos vindo a observar, existem cada vez mais exemplos de conceitos e ideias reinventadas nos tempos modernos. Ações que demoravam algum tempo no passado agora são mais rápidas e eficazes. Como tudo, o ensino também melhorou ao longo dos anos: quadros inteligentes, conteúdo digital e até mesmo o ensino à distancia já são realidades. Para além destas tecnologias, também se tem vindo a observar o aparecimento de um novo paradigma associado à educação: a realidade aumentada (RA) (Billinghurst, 2002). Existem já vários estudos referentes a esta matéria inclusive na área da educação. Mas este não é um conceito novo: a sua origem data aos anos 60 mas só mais recentemente é que o termo foi aplicado. Por volta de 1960, Morton Helig desenvolveu uma máquina a que apelidou de Sensorama. De fisionomia semelhante a uma máquina de jogos árcade, foi apresentada como uma experiencia cinematográfica inovadora: projetava uma espécie de estereoscopia arcaica, simulava vento e movimentava o assento do utilizador. Avançando para 1966, esta é a data da invenção de um dos mais importantes e impulsionadores conceitos da RA, o head-mounted display. Criado por Ivan Sutherland, era demasiado grande e pesado para ser usado por um humano e mostrava apenas formas geométricas simples. Embora limitado (a par do poder computacional da altura) era já um importante passo rumo às implementações atuais. Até esta altura, ainda não se tinha ouvido falar do termo em si. Pensa-se que a RA tenha sido cunhada por Tom Caudell, na altura a trabalhar no projeto Boeing's Computer Services Adaptive Neural Systems Research and Development, aquando o desenvolvimento de uma tecnologia que permitia sobrepor o desenho das plantas sobre o equipamento a ser montado. Isto permitia aos mecânicos da empresa saber o que montar e como montar sem recorrer a plantas ou ao auxílio de engenheiros. Praticamente em simultâneo, em 1992, outras duas empresas deram passos largos no domínio deste novo conceito. LB Rosenberg e um grupo de cientistas constituído por Steven Feiner, Blair MacIntyre e Doree Seligmann, apresentaram avanços semelhantes, o primeiro na área militar e o segundo num manual interativo. Até

1999 a RA tinha-se apresentado apenas no domínio científico: hardware complicado, dispendioso e de difícil acesso tornou o consumidor comum alheio a esta tecnologia. Tudo isto mudou com o aparecimento do ARToolKIt. Apresentado sob forma open source por Hirokazu Kato, este permitia o tracking de vídeo e consequente overlay de objetos virtuais em qualquer plataforma (sistema operativo). Foi isto que facultou a RA para o público em geral: bastava um simples dispositivo portátil com uma camara e conetividade com a internet. Praticamente todas as demonstrações até então foram baseadas nesta tecnologia. Com o aparecimento dos primeiros smartphones modernos em 2008, as aplicações de RA chegaram a uma nova era. Com o poder de processamento de um computador na palma da mão, a realidade aumentada chegou perto do patamar em que se encontra atualmente: globalmente aceite como uma tecnologia emergente e disponível ao utilizador comum. Com este projeto e recorrendo à RA temos como objetivo testar a introdução de novas tecnologias no meio da educação: abordar temas importantes de uma nova perspetiva e verificar resultados. Sendo o conceito em si relativamente simples, a sua implementação é um tanto menos trivial. Podemos afirmar que estamos simplesmente a lidar com um certo conjunto de sensores ligados a um dispositivo inteligente que trata a informação e a disponibiliza para o utilizador. Embora a anterior afirmação seja verdade, a parte complicada reside na forma como se obtém e trata essa informação: sendo o meio a estudar, analógico, quanto mais precisa é a informação recolhida, melhor o tratamento e por conseguinte, a sua aplicação. Saber o que utilizar ou não, distinguir os falsos dos verdadeiros positivos e acima de tudo aplicar isso a uma solução em tempo real, é o verdadeiro desafio e o fator diferenciador nesta indústria. Neste capítulo iremos abordar o enquadramento do projeto referindo os seus requisitos.

1.1 Problema

Como principais requisitos deste projeto, pretendemos estudar a introdução de tecnologias virtuais no contexto do ensino, em particular a RA. Para além disso, utilizaremos a framework YVision de modo a implementar uma solução que combine o real com o virtual de forma a abordar temas cruciais em idades pré-escolares. No final, tencionamos verificar a viabilidade pedagógica da RA e da framework YVision na criação de aplicações desta natureza e validar os temas abordados recorrendo a métricas definidas consoante o teste em questão.

1.2 Enquadramento

Este projeto encontra-se interligado com o consórcio VisualyzART, um projeto de Sistema de Incentivos à Investigação e Desenvolvimento Tecnológico nas Empresas (I&DT) e promovido pela YDreams, uma empresa tecnológica especializada em RA e Interfaces Naturais para o Utilizador (NUI). Englobados neste projeto encontram-se para além da YDreams, o Instituto Politécnico de Santarém (IPS), a Universidade de Coimbra (UC), Instituto Politécnico de Leiria (IPL) e a Universidade Católica Portuguesa (UCP). O objetivo deste consórcio é o de desenvolver uma plataforma que possa ser usada para tornar a RA e as NUI ubíquas no mundo. Encontra-se focado principalmente no desenvolvimento e investigação em plataformas móveis, web e cenários sociais de forma a trazer a RA e a NUI para o grande público. Para tal além dos estudo de tecnologias e algoritmos que tornem essa realidade alcançável, parte do trabalho passa por desenvolver ferramentas que tornem possível, ao utilizador comum, lidar com a complexidade e multidisciplinaridade destes sistemas. Recorrendo à framework em desenvolvimento YVision, cada elemento desta parceria tem objetivos específicos:

- YDreams: desenvolvimento da framework e integrar os resultados das várias entidades
- UCP: exploração da tecnologia no domínio das Artes Digitais
- IPS: exploração da tecnologia no domínio da Educação
- IPL: exploração da tecnologia no domínio da Informática e Programação Criativa
- UC: exploração da tecnologia no domínio do Design e Multimédia.

Para além das responsabilidades de cada parceiro, cabe a todos validar o resultado do trabalho efetuado e apresentar feedback da ferramenta em utilização.

1.3 Abordagem

Dentro da situação atual, o objetivo seria desenvolver uma solução simples e que utilizasse recursos já disponíveis no ambiente alvo. Seria preferencial também o uso de ferramentas de desenvolvimento rápido como é o caso de frameworks e Ambientes Integrados de Desenvolvimento (IDEs). O principal foco seria obter um produto completo

o mais depressa possível. Assim teríamos um prazo mais abrangente para efetuar testes de usabilidade junto do grupo de utilizadores alvo. Com isto, até ao final do mês de Maio está previsto a conclusão do primeiro protótipo a fim de ser testado e modificado durante o mês de Junho. Acabando a primeira faze de testes e com o feedback dos mesmos, começar a elaboração do produto final em meados de Junho de modo que a conclusão do projeto se encontre dentro dos prazos de entrega. Na tabela 1.1 encontra-se a calendarização do projeto.

1.4 Apresentação do documento

Após a primeira abordagem ao assunto no capítulo 1, onde introduzimos os conceitos base e os requisitos deste projeto, iremos desenvolver um pouco mais o conceito de RA no capítulo 2. Neste capítulo explicamos mais aprofundadamente o conceito de RA, apresentamos a arquitetura geral de uma aplicação deste género, descrevemos os seus constituintes, os métodos normalmente utilizados para interagir com o mundo real, IDEs e ferramentas de desenvolvimento bem como frameworks de RA. Após a análise das ferramentas disponíveis, no capítulo 3 serão apresentadas as ferramentas a utilizar. Cada ferramenta utilizada será apresentada e justificada a sua escolha. No capítulo 4 está descrita a solução proposta: aqui é apresentado o estudo do público-alvo, regras de desenvolvimento de jogos para crianças em idade pré-escolar e descritos os jogos elaborados. Posto isto, no capítulo 5, apresentamos a implementação da solução proposta. Neste capítulo esta descrito todo o desenvolvimento dos jogos, ferramentas e algoritmos, bem como as escolhas por detrás de algumas decisões. No seguinte capítulo, validamos e analisamos o trabalho elaborado através de testes documentados. Por fim, no capítulo 7, apresentamos as conclusões deste projeto e aferimos se foi resolvido o problema apresentado e se foram cumpridos todos os requisitos. Para além disso, é neste capítulo que também é referido o trabalho futuro.

Planeamento	
Estado de Arte	Março
Definição de Casos de Uso	Março
Implementação de prótotipos	de Março a Maio
Desenvolvimento da Solução Final	de Junho a Agosto
Validação	finais de Agosto
Conclusão da Documentação	Setembro
Entrega do Projeto	Setembro

Tabela 1.1: Planeamento do projeto

Capítulo 2

Estado de Arte

De forma a implementar uma solução de RA é preciso efetuar estudos em diversas áreas e matérias. A forma como são elaborados os sensores, como poderemos interpretar esses dados de forma a efetuar alguma espécie de tracking, como havemos de apresentar os resultados obtidos, como embrulhar tudo isto num dispositivo adequado ao caso de uso, representam uns dos principais pontos a ter em conta aquando o desenvolvimento de aplicações recorrendo à RA. Dependendo da aplicação em questão, terão de ser tidos em contas outros aspetos, mas no geral, todas as aplicações deste género terão de ser capazes de tratar fielmente a informação do meio, conseguir integrar sincronizadamente o meio virtual com o meio físico e efetuar estas operações sem latência percetível.

> A tracker must be accurate to a small fraction of a degree in orientation and a few millimeters in position; The combined latency of the tracker and the graphics engine must be very low; The tracker must work at long ranges; (Azuma, 1993)

Antes de abordar os constituintes de uma aplicação de RA, as técnicas utilizadas para interagir com o mundo real e o estado em que se encontram as ferramentas de desenvolvimento, iremos primeiramente elaborar o conceito de RA. A sua evolução ao longo dos tempos e o que a distingue de outras tecnologias semelhantes. Para além disso, e depois de abordar os temas anteriormente referidos, iremos analisar o estado atual da indústria com especial foco nas aplicações existentes e nas ferramentas de desenvolvimento disponíveis. Também analisaremos estudos prévios da aplicação desta tecnologia ao ensino.

2.1 Realidade Aumentada

Dado o termo, RA, partimos do princípio que se adiciona alguma informação ao meio que observamos. Pegamos em alguns elementos do mundo físico e aprofundamos a informação que nos é fornecida com dados que não estariam disponíveis de outra forma. Definir o que é RA é uma tarefa compilcada uma vez que o conceito em si se encontra em constante evolução. Em 1997, Azuma tentou definir RA utilizando três princípios (Ronald T. Azuma, 1997):

- Combinar o real e o virtual
- Interatividade em tempo real
- Rendering em 3D

Para a altura, todas as aplicações do género enquadravam-se nesta definição. Hoje em dia também se pode dizer o mesmo, mas o problema é que esta definição abrange muito mais do que aplicações de RA. Mais recentemente, no livro de Lester Maden de 2011 Augmented Reality Browsers for Smartphones Maiden (2011), é apresentada uma definição mais elaborada que tenta específicar melhor este conceito recorrendo a cinco características:

- Combinam o real com gráficos gerados por computador
- Interatividade com objetos em tempo real
- Tracking de ojectos em tempo real
- Providencia o reconhecimento de imagens ou objetos
- Dispõem de um contexto ou dados em tempo real

Podemos então observar que embora esta definição mais recente vá ou encontro de aprofundar a mais antiga, definir o que é RA não é trivial. Mais uma vez, as principais características de aplicações deste género encontram-se específicadas nos conceitos a cima, mas existem algumas falhas. Um dos exemplos é a RA baseada na localização. Embora sendo uma forma de RA, esta não verifica alguns dos aspetos apresentados: não se baseia em reconhecimento de imagens ou objetos e não recorre a qualquer tipo

de tracking. Outro problema com esta definição, como acontecia com a primeira, é a inclusão de tecnologias que não são do dominio da RA, como é o caso da leitura de códigos de barras numa superfície comercial.

Olhando o problema de outra forma, podemos pegar nos conceitos principais no que toca a aplicações desta área e tentar relacioná-los de forma a obter uma explicação mais adequada. De acordo com o Geroimenko (2012), temos então as seguintes ideias:

- Realidade
- Tempo real
- Objetos gerados por computador recorrendo a sensores
- Integração transparente dos vários componentes (realidade com o virtual)
- Dispositivo com capacidades de RA

A RA é então uma perceção em tempo real do mundo real mediada por um dispositivo que interliga a realidade com objetos sensoriais gerados por computador de forma transparente.

De forma a compreendermos melhor esta explicação, é importante perceber a relação entre a realidade Virtual (RV), aumentada e o mundo real. Ora a RA é uma variação da RV. Tomamos então o exemplo deste último conceito: na RV o utilizador é imerso num ambiente virtual sem acesso ao mundo real. Contrariamente, a realidade aumentada possibilita o acesso ao mundo real com objetos virtuais sobrepostos. Assim, a RA suplementa a realidade já existente sem a substituir. A fim de distinguir melhor estes conceitos, na figura 2.1 está representado o esquema Reality-Virtuality Continuum introduzido por Paul Miligram (Paul Miligram, 1994).

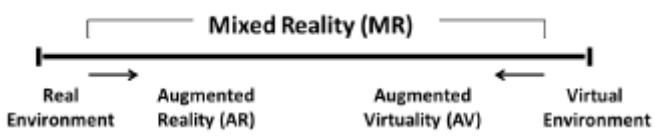


Figura 2.1: Reality-Virtuality Continuum (Paul Miligram, 1994)

Portanto, a RA situa-se no meio-termo entre o real e o virtual, próxima da Virtualidade Aumentada VA. Temos então o real, a realidade suplementada por objetos virtuais (RA), a virtualidade assistida por ações reais (VA) e a realidade virtual (RV).

2.1.1 Tipos de Realidade Aumentada

Pensando no conceito de RA, a primeira impressão é que é uma tecnologia que se manifesta maioritariamente no sentido visual. Sendo a afirmação acima verdade, a RA não se apresenta apenas desta forma. De acordo com os sentidos humanos (visão, audição, tato, olfato e paladar), existe para cada um, um tipo de RA. A figura 2.2 mostra os vários tipos de RA.

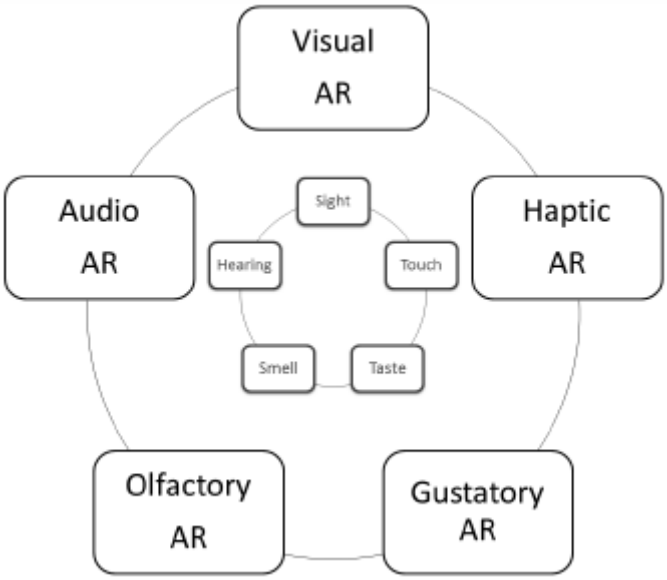


Figura 2.2: Vários tipos de Realidade Aumentada (Geroimenko, 2012)

Como foi referido anteriormente, o mais comum dos tipos apresentados é o Visual, mais vulgarmente descrito por RA. Este recorre a algum tipo de ecrã para projetar imagens que não se encontram visíveis no mundo real. No que toca ao Audio AR, em vez de impor imagens sobre o real, como o próprio nome indica, impõe sons. Existe para além destes dois, o Haptic AR, que permite ao utilizador sentir através do toque, objetos virtuais no mundo real. Os outros dois tipos de RA, o Olfativo e o Gustatório, baseiam-se em sentidos dificeis de simular através da tecnologia atual. Não obstante, já se encontram trabalhos realizados nestas áreas, como é o caso do Gustatory Display (Takuji Narumi, 2011). Como neste trabalho iremos abordar o tipo visual, iremos aprofundar melhor as suas características. A partir daqui iremo-nos referir a este tipo como RA apenas.

Atualmente podemos classificar as aplicações de RA como dois tipos diferentes: Marker e Markerless (Stephen Cawood, 2008). Ora Marker, ou marcador, é uma imagem, digital ou do mundo real, que tem um padrão único que pode ser reconhecido por uma aplicação de RA. Este padrão pode ser um QRCode, um quadro ou até mesmo

uma face. Aplicações desta natureza recorrem a câmaras e sensores para identificar este padrão. Uma vez identificado, o software procede ao cálculo da sua posição e orientação de modo a integra-lo no mundo real. Mas nem sempre foi assim. Inicialmente o termo marcador apenas era usado para imagens do tipo QRCode e código de barras. Com a evolução das tecnologias, passaram a ser utilizados como marcadores, o rosto humano, características de determinadas imagens entre outro aspetos. Com isto o termo Marker passou a ser mais abrangente. Podemos caracterizar os vários tipos de marcadores como técnicos (códigos de barras, QRCode...) ou como naturais (um quadro, um a face humana...). A figura 2.3 ilustra os vários tipos existentes.

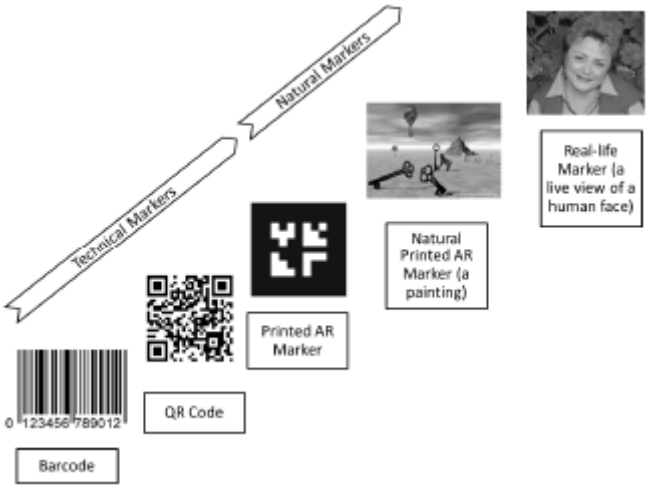


Figura 2.3: Géneros de marcadores (Geroimenko, 2012)

Posto isto, o outro tipo de RA, é a Markerless ou baseada em localização. Neste caso, nenhum marcador entra no processo e a integração com o mundo real é feita através da posição (latitude, longitude e altitude). Exemplo disso são muitas das aplicações de navegação que recorrem à tecnologia GPS para mostrar pontos de interesse e renderizar edifícios e monumentos em tempo real. Na figura 2.4 encontram-se demonstrados os dois tipos de Visual AR (ou só RA no âmbito do nosso projeto).

Dada a natureza do projeto, iremo-nos focar na RA baseada em marcadores. Normalmente o que é verificado atualmente, é uma junção dos dois tipos de forma a promover uma experiencia mais imersiva.

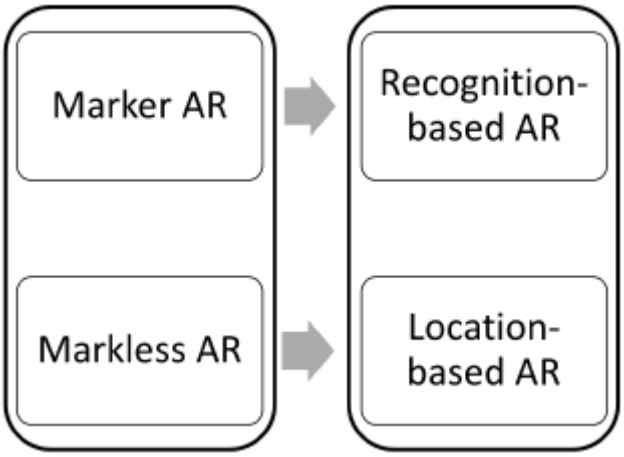


Figura 2.4: Tipos de Realidade Aumentada Visual (Geroimenko, 2012)

2.2 Aplicações de Realidade Aumentada

Tendo a sua origem no meio militar e indústrial, a RA já se espalhou por diversos ramos incluído os lúdicos. Atualmente, já é possível usufruir destes paradigmas em setores tão divergentes e críticos como a medicina e o comércio. Ações como uma ida ao centro comercial, uma viagem turística e até mesmo o ensino de uma determinada disciplina, já são possíveis de efetuar recorrendo à RA. Muita da publicidade atual recorre a tecnologias digitais e a RA não é exeção. Vulgarmente dissimulada através do uso de QRCodes, já se encontram exemplos de implementações onde se pode, por exemplo, experimentar uma peça de mobiliário no espaço que se pretende remodelar mesmo antes de efetuar a compra IKEA (2014). Como referido anteriormente, já existem exemplos de aplicações no ramo da educação. Um dos exemplos é o do ensino da música recorrendo a sensores óticos. Com isto é possível demonstrar como tocar um determinado instrumento de forma mais aprofundada Discovery (2014). No que diz respeito ao entretenimento, existe diverso conteúdo. Desde os mais variados jogos recorrendo a diferentes tipos de sensores (kinect (Microsoft, 2014), move (Sony, 2014), wii plus (Nitendo, 2014)), a experiencias mais imerssivas como é o caso dos simuladores. Um dos primeiros exemplos de uma aplicação deste género é o do ARQuake Lab (2014), uma versão do popular jogo Quake em RA. Toda esta diversificação de conteúdos prova a importância que esta tecnologia tem vindo a adquirir no panorama atual.

2.3 Arquitetura de uma aplicação de RA

Dada uma determinada aplicação de RA, é sempre necessário ter um método para analisar o meio em redor, processar essa informação, fornecer o resultado ao utilizador e permitir a interação do mesmo. Temos então a seguinte arquitetura geral de uma aplicação deste género 2.5.

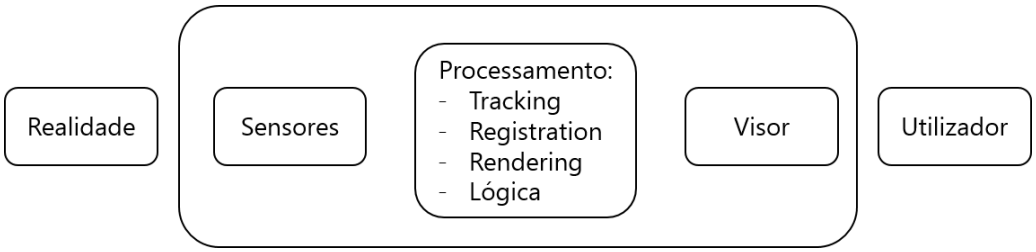


Figura 2.5: Arquitetura de uma aplicação de realidade aumentada

Assim sendo, temos alguns tipos de sensores que analisam o meio e alimentam informação ao processador. Estes sensores podem ir desde uma simples camara a sensores GPS. O processador é então responsável por tratar a informação vinda dos sensores, efetuar o tracking, rendering da imagem final e lógica da aplicação (iremos abordar melhor estes conceitos no seguimento do capítulo). Depois desse tratamento, a informação é então transmitida ao utilizador através de um tipo de display (simples ecrã, projetor, head-mounted display...). O utilizador apenas interage com a realidade em si embora seja transmitida mais informação a partir do display, daí o termo RA. De seguida serão melhor explicados cada um dos constituintes desta arquitetura e as tecnologias por detrás dos mesmos.

2.3.1 Hardware

Começando pelo hardware num sistema de RA, são sempre necessários algum tipo de processador, uma forma de apresentar informação e dispositivos de input, nomeadamente sensores. O poder computacional é um dos principais aspetos em RA. Processar a informação rapidamente melhora a usabilidade de uma aplicação e aumenta as possibilidades da mesma. Com o chegar dos smartphone modernos, este poder de processamento impulsionou a evolução da RA: com os variados tipos de sensores (camara, acelerómetro, GPS...), não só é possível agora uma nova interação como também novas autónomas e compactas aplicações.

Dispositivos de Entrada

De forma a poder gerar informação de acordo com o meio ou uma determinada interação, é necessário antes de mais recolher certos dados. Necessários para que as aplicações de RA efetuem várias técnicas de reconhecimento, quer de movimento, quer de som, os dados são recolhidos através de variados sensores presentes atualmente na maior parte dos smartphones. Entre os mais importantes destacam-se:

- Camara
- Microfone
- Acelerómetro, giroscópico, bussola, gps, entre outros sensores sem fios
- Tecnologias RFID
- Outros acessórios desenvolvidos específicamente para determinadas tarefas (como é o caso de braçadeiras desportivas)

Display e Interface

Outro dos aspetos importantes no que toca a RA é a forma como se interliga o virtual com o real. De forma a promover uma experiencia imersiva, é preciso ter em conta o modo de apresentação de uma dada aplicação. Várias tecnologias utilizadas são:

- Handheld: Dispositivos portáteis de tamanho diminuto. Devido à sua dimensão e ao fato de ser necessário utilizar o ecrã reduzido, distorce a perceção do real (ângulo de visão do ser humano é superior ao da objetiva do dispositivo.)
- Head-mounted: Ecrã montado no campo de visão do utilizador. Torna a experiencia mais imersiva devido à envolvência da aplicação. Um dos exemplos atuais é o Google Glass (Google, 2014).
- Spatial (SAR): usa projetores para suplementar o mundo real (em vez de visores). Não depende do utilizador o que possibilita uma experiencia para grupos. Como limitações apresenta uma grande suscetibilidade em relação à iluminação do meio e fraca mobilidade.

Para além das tecnologias mais usuais, encontram-se em desenvolvimento lentes de contacto (Lavars, 2014), projetores de retina (Michael Tidwell and III, 1995) e outros novos displays.

2.3.2 Software, Tracking e Registration

Para suportar a informação proveniente dos sensores, existem várias técnicas consoante o objetivo. Independentemente da aplicação, existem dois conceitos a ter em conta: tracking e registration. O tracking é a forma de continuamente saber a posição atual ou relativa do ponto de vista do utilizador. É um dos pontos-chave nas aplicações de RA. É através disto que é possível integrar os objetos ou informação que queremos aumentar no mundo real. A este segundo conceito dá-se o nome de registration. Falando de tracking, existem vários tipos de sensores: desde sensores mecânicos, magnéticos, ultrasónicos, Gps, wifi, bússola, acelerómetro, giroscópio até a computer vision (camaras). Dada a natureza do projeto, apenas nos iremos focar em tecnologias passivas de tracking, mais propriamente em tecnologias óticas (computer vision). Nesta matéria existem duas possibilidades consideradas: a utilização de marcadores e o Natural Feature Tracking (NFT). Em relação ao NFT podemos ter em conta vários aspetos, visto que se baseia no estudo das características do objeto: cor, vértices, histogramas, entre outros. Isto torna esta técnica bastante complexa e computacionalmente pesada. Neste projeto iremos recorrer a marcadores de cor e experimentar a utilização de marcadores técnicos. Dado o tema do projeto, de seguida abordaremos esta tecnologia no contexto da educação.

2.4 Realidade Aumentada Aplicada ao Ensino

Cada vez mais, as crianças têm acesso às novas tecnologias quer seja através da televisão, computadores ou até mesmo tablets. Têm acesso ao mais variados tipos de conteúdos disponíveis nos diversos tipos de plataformas mesmo os que não se mostram adequados a este grupo de utilizadores. Para produzir conteúdo digital para crianças é necessário seguir determinadas regras que não são tidas em conta aquando a produção para outras audiências. As crianças não são adultos em miniatura, estão em fase de desenvolvimento e qualquer interação pode alterar a sua mentalidade e o que se viram a tornar mais tarde. Não basta alterar as cores de um produto, a história por detrás de um jogo ou a banda sonora, é preciso ter em conta as suas capacidades cognitivas e

motoras. Da mesma forma, desenvolver aplicações em realidade aumentada direcionadas para crianças, implica um estudo prévio e um compreensão das características do público alvo. Analisando os estudos (Janet C. Read, 2011) desenvolvidos no âmbito de Child Computer Interaction (CCI), uma área de pesquisa dentro de Human Computer Interaction (HCI), começamos por justificar o porquê destas preocupações e identificar as principais normas a ter em conta no que toca ao desenvolvimento de software para crianças. Com o popularização dos primeiros computadores, na década de 80 e 90, surgiram os primeiros estudos sobre a interação das crianças com os computadores primeiro por (Page, 1980) e mais tarde por (Kafai, 1990). A partir daí, cada vez mais se tem vindo a estudar os efeitos da utilização destas tecnologias pelos mais novos, muito devido à adoção em larga escala de equipamentos desta natureza. A figura 2.6 apresenta a subida da adoção do computador pela população Portuguesa.

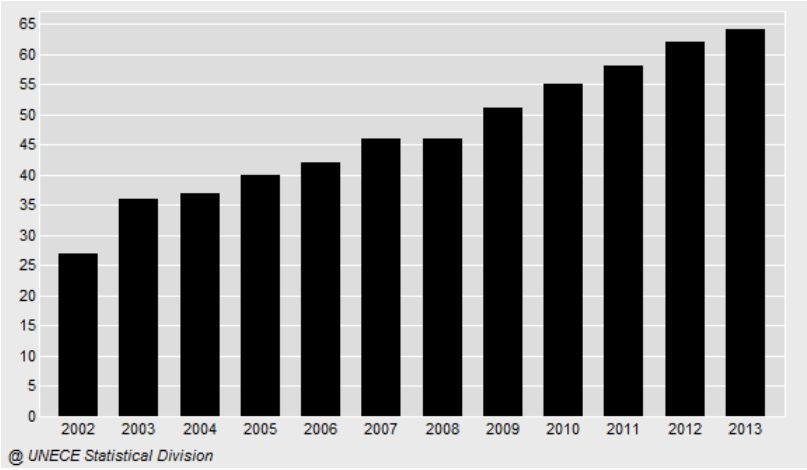


Figura 2.6: Percentagem da população Portuguesa, compreendida entre os 16 e os 74 anos que usa computador (UNECE, 2014)

Muito do que se tem vindo a observar, este acesso às novas tecnologias traz muitas vantagens aos mais novos mas também tem os seus pontos negativos. A utilização do computador (e outros dispositivos semelhantes) ajuda ao desenvolvimento cognitivo e a quebrar barreiras na interação social e facilita o desenvolvimento das capacidades motoras, coordenação e processamento visual. Por outro lado, pode expor as crianças a conteúdos nocivos, ameaçar a sua segurança através do cyber bullying e interação com estranhos, alterar rotinas de sono e levar a atrofias musculares e a problemas motores e visuais devido a longas horas de contínua utilização. Embora não tenha atualmente aplicação para todas as tarefas que uma criança possa levar a cabo num computador, uma das formas de contornar parte dos problemas do uso destas tecnologias pode passar pela adoção da realidade aumentada.

Trocar a interação normalmente utilizada por jogos didáticos para uma abordagem

mais imerssiva pode trazer benefícios não só a nível de aproveitamento como a também a nível ergonómico. Através do uso de RA, a criança tem outra interação com o computador. O facto de se terem que mexer físicamente evita possíveis problemas a nível motor como as atrofias musculares (ao contrario de estarem sentados durante longos períodos muitas vezes de forma errada). Isto aliado à maior concentração e imersão torna a realidade aumentada um fator a ter em conta também na sala de aula. Para além dos benefícios que foram referidos anteriormente, a realidade aumentada também pode ser explorada na educação. Através da exposição de texto sobreposto, visualização de objetos 3D, vídeos, imagens e outro conteúdo, a RA apresenta um bom complemento ao ensino já praticado nas escolas (Dede, 2009). Qualquer tema que não possa ser visualizado no mundo real, pode ser analisado recorrendo a esta tecnologia: um exemplo em particular é o fenómeno das reações químicas que não pode ser observada facilmente (Eric Klopfer, 2007). (Wei Liu, 2007) apresenta vários sistemas que exploram este campo: estes vão desde a observação do sistema solar até à análise do fenómeno da fotossíntese. Para além destes exemplos, existem já testados e implementados, protótipos que exploram a RA para abordar temas como geometria e matemática (Hannes Kaufmann, 2005).

Sendo a parte tecnológica da realidade aumentada bastante importante, o que se pode fazer com ela é que dita o seu futuro. A sua aplicação ao ensino, segundo o estudo (Hsin-Kai Wu, 2013), favorece o mesmo em cinco aspetos:

- Permite estudar conteúdo em perspetiva 3D
- Aprendizagem colaborativa e omnipresente
- Transmissão do sentido de presença e imersão
- Visualizar o que normalmente não é visível na natureza
- Permite fazer a ligação entre o ensino formal e informal

Estudos, como (Lucinda Kerawalla, 2006), comprovam o acima referido. Através de dois tipos de sessões diferentes (uma onde foi utilizada realidade aumentada e outra onde apenas foi usado o ensino normal) sobre os mesmos tópicos, concluí-se junto dos professores que os métodos de RA contemplam uma plataforma que disponibiliza conteúdos benéficos para a aprendizagem. Embora este estudo não comprove que a realidade aumentada enriqueça o desempenho escolar, estudos prévios identificaram alguns indícios que defendem esta afirmação: (Cynthia E Copolo, 1995) observaram que os alunos que usaram suporte digital para complementar os métodos clásssicos

tiveram um melhor aproveitamento. Mesmo que esta tecnologia traga melhorias ao ensino, não se deve rejeitar o que foi implementado até então: muitos dos alunos podem não se sentir confortáveis com novos métodos, cabe a cada um escolher o melhor para si. De salientar que esta tecnologia pode não ser só utilizada na sala de aula: existem já implementações de jogos educativos em dispositivos móveis que facilitam a aprendizagem em visitas de estudo (Eric Klopfer, 2008) e (Kurt D. Squire, 2007). Em suma, a realidade aumentada no seio da educação ainda é uma matéria muito recente. A sua abordagem implica um estudo prévio dos vários fatores inerentes: é preciso identificar o que falta e como é que a RA pode ser útil nessa situação. Problemas com a precisão dos equipamentos, lentidão nas ações, inflexibilidades do sistema, são tudo aspetos que podem tornar a experiencia frustrante. De seguida iremos abordar as principais ferramentas para o desenvolvimento de aplicações de realidade aumentada.

2.5 Ferramentas de Desenvolvimento Rápido

Passando para o modo de como desenvolve realidade aumentada, existem várias ferramentas de desenvolvimento rápido já disponíveis atualmente. Muitas utilizam IDEs comumente utilizados mas outras apresentam as suas próprias ferramentas de authoring. Abaixo destacam-se as principais frameworks até então 2.1.

Analisaremos agora cada uma das soluções apresentadas tendo em conta os aspetos inessenciais do projeto. Isto é, em que plataformas corre, que tipo de sensores são suportados, quais são os custos do desenvolvimento e em que linguagem é efetuado, se existe integração de bibliotecas externas ou se utilizam algoritmos proprietários e que funcionalidades apresentam no geral. Como a framework escolhida (dado ser um requisito do projeto) foi o YVison, iremos descrevê-la melhor no capítulo seguinte.

Framework	Linguagens	Plataformas
Metaio	Objective C, Java, C#	iOS, Android, Windows, Web
AR Media	Java, C#, C, C++	Windows, Mac OS, Linux
IN2AR	Objective C, Java, C#, Flash	iOS, Android, Mac OS, Windows, Web
Instant Reallity	Objective C, C++	Windows, Mac OS, Linux
D'Fusion	Flash	iOS, Android, Web
Wikitude		Android, iOS, Blackberry
Vuforia	Objective C, Java, C++	Android, iOS,
Layar		Android, iOS
YVision	C#, Java Script	Android, iOS, Windows, Mac OS, Linux

Tabela 2.1: Ferramentas de Desenvolvimento

2.5.1 Metaio

Empresa alemã de RA que nasceu de uma parceria com uma empresa de automóveis. Com raízes no sector indústrial, desenvolveu uma serie de ferramentas destinadas aos mais variados graus de conhecimento na área. Disponibiliza vários produtos consoante a necessidade:

- Mobile SDK - Permite o desenvolvimento de aplicações móveis de RA. Atualmente suporta Android e iOS.
- PC SDK - Permite implementar soluções de RA através da disponibilização de tracking 3D, deteção de faces, tracking infravermelho e laser e calibração extensiva dos sensores óticos.
- Web SDK - Facilita a implementação de soluções de e-commerce sem recorrer à instalação de editores e software de terceiros.
- Metaio Creator - Software básico que permite a criação de pequenas demonstração de RA. Focada para pessoas sem conhecimento de programação. 2.7
- Junaio - Browser de RA que permite aos utilizadores aceder a informação do meio utilizando a câmara dos smartphones.
- Metaio Engineer - Pacote focado para o meio indústrial com aplicações na área de montagem e design de componentes

Esta solução é utilizada em vários campos atualmente: destaque para vários jogos didáticos, catálogos de produtos e manuais de apoio automóvel. Metaio (2014)

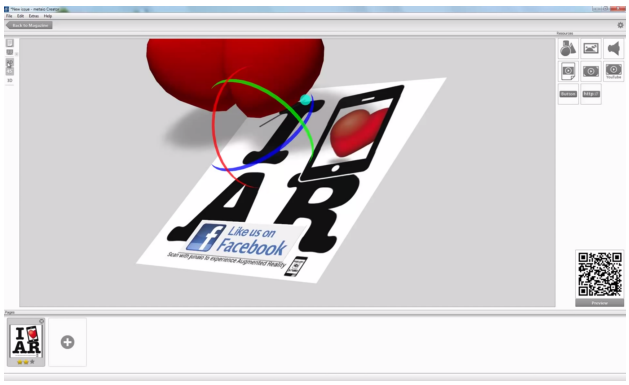


Figura 2.7: Metaio Creator

2.5.2 AR Media

Apresentada como uma plataforma modular e estruturada, estão incluídas nesta framework várias funcionalidades incluindo real-time tracking e real-time rendering (ambas independentes uma da outra). Implementada em C/C++, AR Media é multiplataforma o que a permite correr em sistemas Windows, Mac, Linux e sistemas móveis (IOS e Android) através de wrappers para cada uma das plataformas (objective-c e java). Dada a modularidade, é possível introduzir vários sistemas de tracking e rendering: até à data, esta framework incorpora as livrarias ARToolKit, alguns métodos de OpenCV e suporta o reconhecimento de gestos OpenNI. Para além das anteriores referidas, também está presente o Inglobe Tracker e várias extensões de OpenGL para rendering. Para desenvolvimento, são disponibilizados vários pacotes diferentes para diversas aplicações:

- ARToolKit: livraria de RA que corre em diversos sistemas operativos (Windows, Linux, Mac OS X e Irix);
- NyARToolKit: port da livraria ARToolKit que suporta a maioria das plataformas de maquinas virtuais;
- osgART: livraria em C/C++ que inclui para alem da ARToolKit a OpenSceneGraph que permite um desenvolvimento rápido de aplicações de RA ricas em conteúdo;
- FLARToolKit: versão Flash Actionscript da biblioteca ARToolKit;
- FLARManager: framework de desenvolvimento rápido em Flash;
- SLARToolkit: livraria para a plataforma Windows Phone (Silverlight) baseada nas livrarias NyARToolkit e ARToolkit;

Para além dos pacotes referidos anteriormente, ainda são disponibilizados plugins para o Unity e plataformas móveis, como é o caso de Android e iOS. É possível encontrar exemplos de aplicações que utilizam esta tecnologia nas mais variadas áreas da indústria como é o caso da arquitetura, medicina, publicidade e entretenimento. ARMedia (2014) Em baixo encontra-se um exemplo de uma aplicação que utiliza esta framework 2.8.

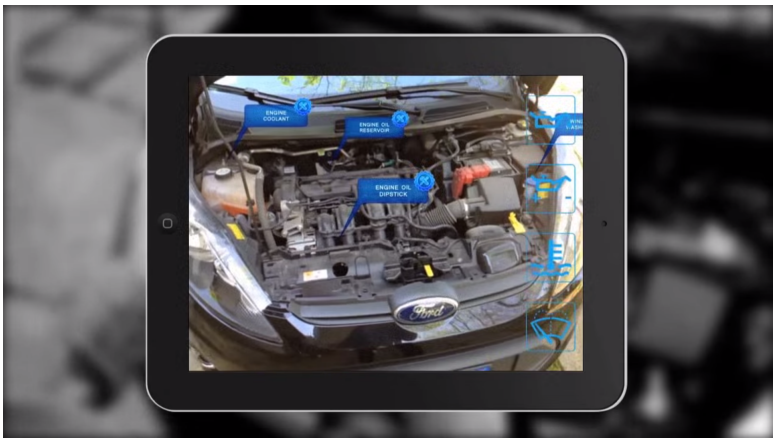


Figura 2.8: ARMedia 3D tracker

2.5.3 IN2AR

O IN2AR disponibiliza extensões para Adobe e Unity3D que permitem adicionar capacidades de RA a aplicações web e móveis. Permitem a utilização de multi marcadores recorrendo apenas às camaras normais que se podem encontrar nos portáteis e smartphones. Dadas as características dos programas que estendem podem correr em várias plataformas. Suporta teoricamente o reconhecimento de até 100 imagens diferentes na mesma aplicação. características do SDK para Flash e Adobe Air 2.9:

- IN2AR em action script e extensões nativas
- Exemplos básicos de aplicações de Realidade Aumentada em action script
- Imagens gratuitas e licenciadas
- Librarias de terceiros incluídas nos demos

características do SDK para Unity:

- IN2AR plugin para iOS, Android, Windows e Mac OSX
- Disponibilizado numa package
- Single Marker Demo
- Multi Marker Demo
- Masking Demo

Atualmente é possível encontrar exemplos do uso desta tecnologia em campanhas publicitárias e até mesmo num jogo para iOS (IN2AR, 2014).

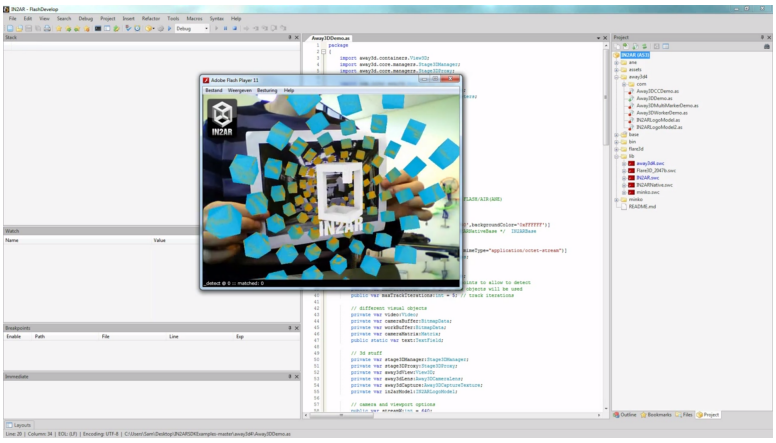


Figura 2.9: IN2AR FlashDevelop

2.5.4 Instant Reallity

A instantreallity-framework incorpora vários standarts da indústria de forma a disponibilizar uma arquitetura flexível e adaptável. A sua arquitetura é constituida por:

- OpenSG: Sistema de rendering Open-source scene-graph
- InstantIO: Sistema de gestão de comunicações em rede
- VisionLib: Pipeline de processamento visual
- Avalon: Gestão dinâmica de scenes e manipulação do sistema

O modo de desenvolvimento permite ao programador modelar uma aplicação em vez de a ter de programar totalmente, o que facilita e diminui o tempo deste processo. Através do uso de graphs, estas aplicações são extremamente flexiveis: cada graph define o comportamento e a relação entre componentes. Para além disso, esta arquitetura permite escalar aplicações desde um simples smartphone até um sistema complexo de vários monitores. De forma a funcionar da melhor maneira possível em qualquer hardware, são também disponibilizadas otimizações em run-time recorrendo a algoritmos próprios e delegação de processamento através da internet. A Instant Reallity dispõem de várias ferramentas de integração para programas de modelação permitindo uma compilados importação de modelos e conteúdos. Suporta várias plataformas (Windows, Linux, Mac OS X) 2.10. No que toca a exemplos da utilização desta

framework, as soluções mais interessantes encontram-se no meio científico onde foram elaborados planetários virtuais, simuladores de tato entre outros. (Reality, 2014)

Figura 2.10: Instant Reality Player MAC

2.5.5 D'Fusion

Solução concebida para várias plataformas de realidade aumentada. Assim encontra-se subdividida em vários produtos. O D'Fusion Mobile destinasse a plataformas móveis, mais propriamente os smartphones e tablets. Temos depois o D'Fusion @Home que se centra no desenvolvimento para computadores com integração nas redes sociais. Por fim temos o D'Fusion Pro com suporte para vídeo HD, múltiplas camaras, camaras de infravermelhos entre outros sensores, focado em aplicações profissionais (eventos empresariais, manutenção indústrial...). Como plataforma de desenvolvimento gratuito, esta disponível o D'Fusion Studio, representado na figura 2.11 que permite a criação de cenas 3D, cenários e o tracking de conteúdo multimédia. O D'Fusion Studio sendo uma ferramenta de authoring, permite a criação rápida e simples de aplicações ar recorrendo ao paradigma da programação orientada a objetos e a linguagens padrão. Permite a exportação para multiplataformas, tracking sem marcadores, suporte de até 10000 imagens na base de dados local (possibilidade de ser atualizada em runtime) e integração do sensor Kinect. Para além disso tem integrado um motor de rendering 3D, de vídeo e física.

O D'Fusion já se encontra representado por várias aplicações em várias plataformas. Jogos publicitários, eventos públicos, livros de realidade aumentada e até eventos virtuais de larga escala, são alguns contituintes deste portfolio. (Immersion, 2014).

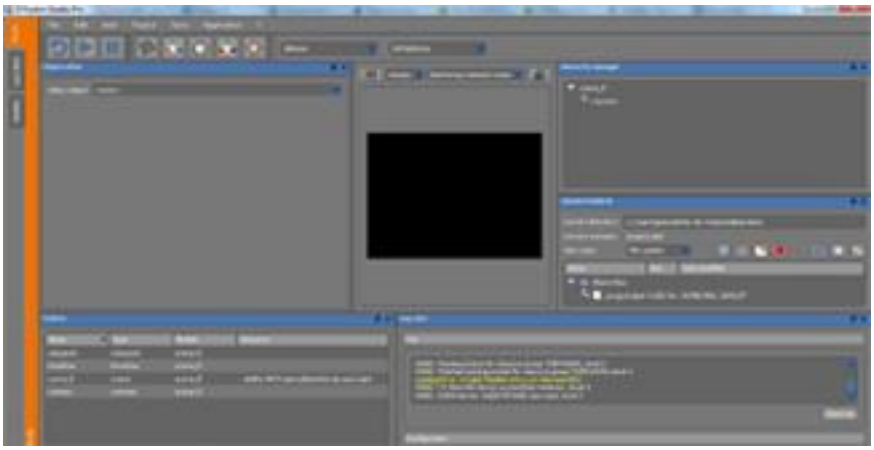


Figura 2.11: D'Fusion Studio

2.5.6 Wikitude

Wikitude é uma framework de realidade aumentada desenvolvida específicamente para dispositivos móveis. Utilizando os sensores disponibilizados pelos smartphone atuais, a informação de geolocalização e o conteúdo proveniente da camara permitem a aplicações que utilizem esta framework, implementar uma camada de informação que sobrepõe a realidade. É disponibilizado suporte a iOS, Android e BlackBerry. O conteúdo destas aplicações é elaborado recorrendo a HTML5, JavaScript e CSS. Em baixo encontra-se uma imagem referente ao Wikitude Studio 2.12: software central da framework.

Embora tenham começado por desenvolver produtos de navegação, como é o caso do Wikitude Drive, já desenvolveram também aplicações na área da publicidade, feiras tecnológicas, exposições de arte e jogos. (Wikitude, 2014)

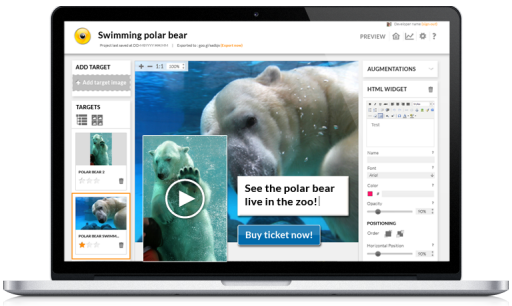


Figura 2.12: Wikitude Studio

2.5.7 Vuforia

O Vuforia é uma biblioteca de realidade aumentada desenvolvida pela Qualcomm. Suporta várias plataformas incluindo iOS, Android e ainda disponibiliza plugins para a plataforma Unity3d. Entre as funcionalidades implementadas destacam-se o reconhecimento de texto, imagem, marcadores e a possibilidades de incorporar modelos 3d e vídeos nesses mesmos alvos em tempo real. Para além das características normais de uma framework de realidade aumentada, o Vuforia ainda disponibiliza reconhecimento de marcadores a partir da Cloud (até 1 milhão simultaneamente), criação de marcadores em run-time, deteção de imagens e traking em superfícies cilindrícas e deteção de 5 marcadores em simultâneo. Todas estas funcionalidades encontram-se adaptadas para o mundo real: são assegurados bons resultados em ambientes de luz limitada e em condições em que os marcadores se encontram parcialmente escondidos. Para desenvolver em Vuforia é necessário criar marcadores personalizados: não existem marcadores específicos ditados pela plataforma. Tomando o caso de uma simples imagem como marcador, o developer efetua o upload da imagem que quer tornar como marcador para a plataforma de criação de marcadores do Vuforia 2.13. Após isso, este marcador pode ser obtido de duas maneiras por parte da aplicação: através da cloud ou através da base de dados local da aplicação. A partir daí e utilizando as funcionalidades do Vuforia, é possível criar aplicações simples e ricas em conteúdo. Atualmente, mais de 200 aplicações fazem parte do catalogo desta biblioteca: desde jogos a produtos relacionados com a saúde. (Vuforia, 2014)

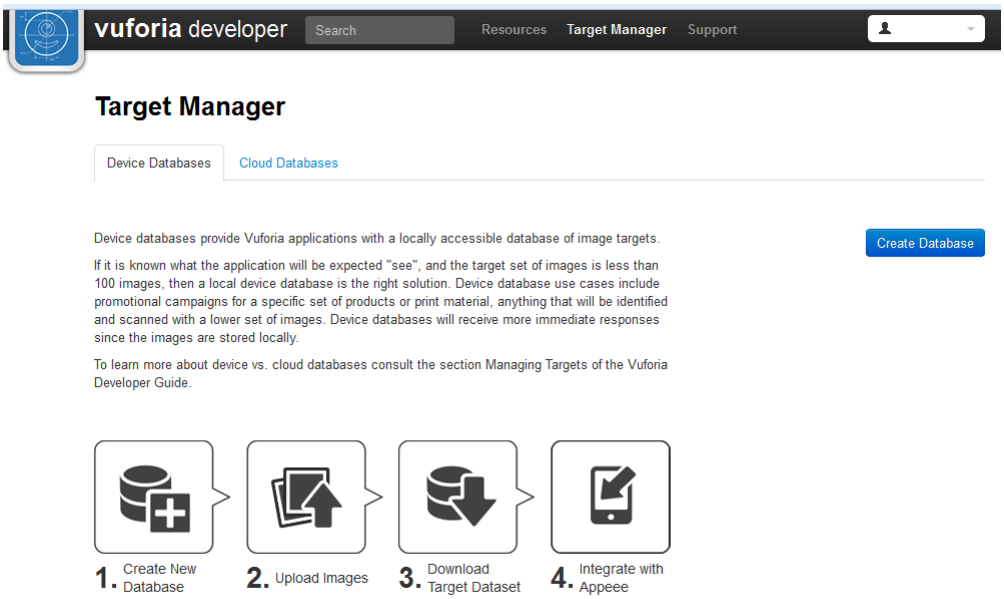


Figura 2.13: Vuforia Target Manager

Como forma de testar esta framework, foi desenvolvida uma aplicação, recorrendo ao exemplo Virtual Buttons, que permitia através da camara do smartphone, identificar os vários constituintes de um dispositivo e apresentar informação referente aos mesmos. Utilizando o exemplo da baia do motor de um automóvel, esta aplicação mostrava três secções diferentes: o motor, radiador e o filtro de ar. Ao apontar o telemóvel para esta imagem (pré carregada para a base de dados local elaborada recorrendo à plataforma web da Vuforia: Targets), o utilizador podia clicar nas secções virtuais, ou botões, e obter informação das diferentes peças. A partir desta implementação é possível lançar um site web, um ficheiro pdf ou até mesmo um vídeos instrucional.

2.5.8 Layar

Layar é uma plataforma direcionada específicamente para dispositivos móveis. É uma aplicação web que associa conteúdo digital a recursos impressos sem recorrer a QRCodes ou tecnologias semelhantes. Estas aplicações, devido a não requerer competências de programação, são largamente utilizadas. Com a sua interface gráfica torna-se simples criar uma experiencia de realidade aumentada em pouco tempo 2.14. Para utilizar estas aplicações web, chamadas de layers, o browser utiliza o acelerómetro, camara, bussola e GPS, sensores vulgarmente presente nos smartphones. Com estes é possível identificar a posição e a perspetiva de visão do utilizador. Estas layers são webservices REST que disponibilizam pontos de interesse próximos da localização do utilizador. Desenvolvidas por terceiros utilizando a API grátis disponibilizada, apenas são validadas e publicadas pela Layar. Muitos dos exemplos que se podem encontrar passam por revistas e catálogos interativos.

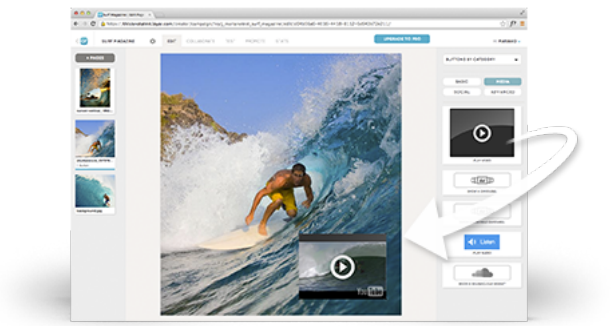


Figura 2.14: Layar creator (Layar, 2014)

2.5.9 YVision

YVision é uma framework que possibilita o desenvolvimento rápido de aplicações baseadas em Natural User Interfaces (NUI). Integra computer vision (recorrendo a webcams e outros sensores), rendering em tempo real, simulação de física, realidade aumentada, inteligência artificial, multitasking, entre outros. Desenvolvida de forma a efetuar a ponte entre os tradicionais motores de jogo e as NUI, é baseada em componentes o que permite o desenvolvimento rápido de sistemas complexos recorrendo a módulos pré-concebidos. Esta arquitetura encontra-se em muitos dos jogos apresentado atualmente na indústria o que valida a sua implementação. Recorrendo às fundações do Microsoft .Net, utiliza código managed o que proporciona um desenvolvimento rápido visto que muitas das operações básicas como a gestão da memória e o paralelismo são tratados automaticamente. Para além disso, também é compatível com Mono o que facilita o desenvolvimento multiplataforma, quer para desktop (PC e Mac), dispositivos móveis (iOS, Android e Windows Phone) e consolas (Xbox 360 e Wii). Juntamente com a framework, são disponibilizados alguns conteúdos para melhor integração com os ambientes de desenvolvimento, como é o caso de templates de projeto, code snippets, auto-completion, tutoriais e documentação. Já se encontram disponíveis alguns exemplos do uso desta framework na indústria quer em instalações interativas, demonstrações de realidade aumentada e até mesmo em robótica. 2.15 Esta framework encontra-se disponível em formato de biblioteca ou plugin para o editor Unity 3D. Iremos analisar mais pormenorizadamente esta ferramenta visto esta ter sido a escolha para este projeto.

Figura 2.15: Robôs autónomos no expositor da Santander(YVision, 2013)

2.6 Conclusão

Neste capítulo apresentámos o conceito de realidade aumentada. Efetuámos a sua definição e desenvolvemos a sua abrangência. Identificámos os vários tipos existentes com foco na realidade aumentada visual (dado este ser o abordado neste projeto). Aprofundámos as várias tecnologias por detrás de uma aplicação deste género e estruturámos uma arquitetura generalizada de um sistema de realidade aumentada. Por fim apresentámos vários exemplos da sua aplicação no quotidiano e identificámos as principais ferramentas de desenvolvimento neste meio.

Capítulo 3

Ambiente de Desenvolvimento

Dado um dos requisitos deste projeto ser o uso do YVision, neste capítulo iremos descrever o que é e quais as suas principais características e funcionalidades. Como plataforma de desenvolvimento iremos utilizar o Unity, um motor de jogo que dispõe do seu próprio IDE. Assim, também iremos explicar como o YVision se integra com o Unity e como este conjunto pode ser utilizado para aplicações de realidade aumentada. Para além disso iremos abordar o Unity sucintamente: iremos descrever o que é, como está organizado e como funciona. Para complementar a informação apresentada, também iremos abordar a preparação do ambiente de desenvolvimento.

3.1 Unity

Começando pelo Unity, este é um motor de jogo que permite desenvolver rapidamente para várias plataformas usando uma linguagem comum. Para isso, dispõe de várias ferramentas incorporadas no seu SDK. Integrados no Unity estão um editor 3D, um modelador simples, compilador, entre outras ferramentas. Como no YVision, o Unity baseia-se no conceito de Composition, o qual explicaremos em mais pormenor na seção referente ao YVision. O desenvolvimento em Unity é constituído principalmente pelos seguintes elementos:

- Scenes: Constituem uma cena no jogo. Isto é, um nível, menu, ecrã de carregamento...
- Game Objects: Elementos de uma cena. Qualquer elemento presente numa Scene é um objecto: camaras, cubos, luzes, áudio... Funciona como contentor para os vários componentes que implementam funcionalidades.

- Prefabs: Conjunto de elementos que formam um grupo lógico. Ideal para criar moldes de objetos que se usarão várias vezes.
- Components: add-ons que possibilitam funcionalidades novas aos objetos (física, colliders, YVision Behaviours...)
- Scripts: contém a lógica do jogo e normalmente encontram-se associados a objects. Semelhantes aos componentes, têm a particularidade de poder existir mais do que um associado ao mesmo GameObject.

No Unity o fluxo de trabalho é extremamente intuitivo: estão presentes várias ferramentas que permitem um desenvolvimento rápido e grande reaproveitamento de recursos. Começando pelo editor integrado, esta é a interface principal para o desenvolvimento de jogos. É aqui que é feita a importação dos recursos, a construção das cenas e onde é adicionada a interação através dos scripts. Aqui é também possível testar em tempo real o jogo, alterando as diversas variável e exportar para as plataformas desejadas. A sua interface simples 3.1 é constituída pelos painéis "Scene View" [1], "Game View" [2], "Hierarchy" [3], "Project" [4] e "Inspector" [5]. O "Scene View" permite ver e editar o conteúdo de uma determinada "Scene", movendo e alterando os objetos nela presentes. O "Game View" permite antever o produto final, dando uma perspetiva em Jogo. Para além disso, também é possível testar o jogo em tempo real dentro do editor. O painel "Hierarchy" lista todos os objetos de uma determinada "Scene", ordenados alfabeticamente e hierarquicamente. No "Project" são apresentados todos os recursos do nosso projeto. São incluídos os objetos 3d, sons, scripts, vídeos, entre outros. O "Inspector" é um painel que muda consoante o objeto ou recurso selecionado. Permite ver e editar as suas características dependendo do tipo selecionado. Para além dos principais painéis, uma ferramenta importante apresentada também na interface é a barra de ferramentas [6]. Esta contém as ferramentas de edição de "Scene" (escala, posição, rotação...), os controlos do jogo (play, pause, frame-by-frame) e dois drop-downs que permitem escolher que layers estão visíveis e que layout está ativo.

Esta interface, para além de simples e acessível, é também bastante flexível e personalizável. A sua organização pode ser facilmente alterada consoante as preferências do utilizador e novas funcionalidades podem ser adicionadas à mesma recorrendo scripts e extensões de terceiros. A modularidade do inspector permite inúmeras possibilidades de desenvolvimento, como por exemplo, exibir propriedades apenas de leitura, reforçar restrições de valor ou simplesmente alterar o modo de apresentação de uma opção. Tão importante como a interface, as funcionalidades disponíveis e a compatibilidade com os mais variados recursos são imprescindíveis num bom motor de jogo. O Unity pode importar modelos 3D, bones e animações a partir de quase todos os aplicativos 3D.

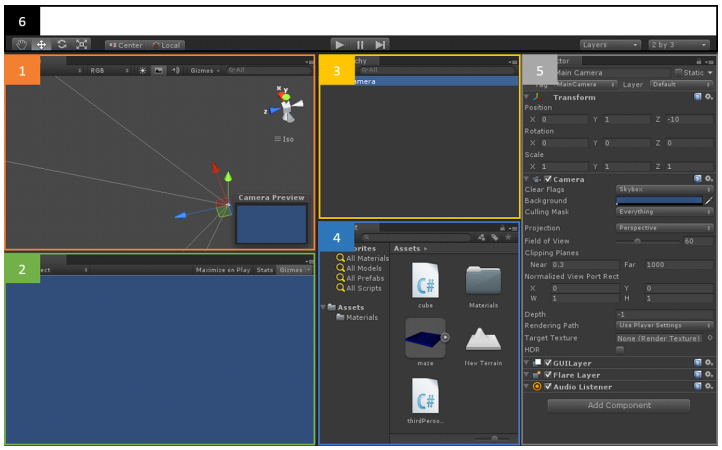


Figura 3.1: Interface do Unity

Suporta também vários formatos de áudio e imagem e permite organizar os recursos facilmente através da janela do Project. De seguida iremos explicar mais pormenorizadamente cada janela apresentada e descrever sucintamente o fluxo de trabalho neste IDE.

3.1.1 Scene View

É onde é construído e modelado o jogo. Nele podemos ver todos os objetos que formam uma determinada cena no jogo. A interação com este painel é feita maioritariamente pelas ferramentas localizadas imediatamente a cima na barra de ferramentas. São elas a "Hand Tool", que permite navegar na cena através de pan, freelook e orbit, "Translate Tool", que possibilita a deslocação de objetos no espaço 3D, "Rotate Tool", que permite rodar objetos de acordo com os vários eixos e a "Scale Tool" que facilita a alteração da escala dos objetos também nos diferentes eixos. Para além das ferramentas de interação, é possível alterar o modo como os conteúdos na "Scene View" são renderizados. Através dos controlos localizados na barra principal deste painel é possível escolher o que se está a visualizar: wireframe, textures, lights, áudio, entre outros. Também na barra principal do painel, encontram-se à direita uma drop down que permite identificar os diferentes tipos de objetos visualmente através de ícones configuráveis e ainda uma barra de pesquisa constituintes de uma Scene.

3.1.2 Game View

É a View que mostra o aspeto final que se irá obter do jogo. Esta é invocada quando se clica no botão "Play" na barra de ferramentas. A partir daí, é possível experimentar e testar o que será o resultado final do projeto. Também através da barra de ferramentas é possível pausar e avançar frame a frame de modo a poder ter um olhar mais detalhado do resultado. Na barra superior da "Game View" é possível alterar alguns aspetos de visualização: é possível indicar o rácio da imagem, resolução, mostrar indicadores como os fps's e ainda se queremos a "View" maximizada aquando a execução. Qualquer alteração feita durante execução do jogo (alteração de variáveis, valores de componentes...) é apresentada imediatamente nesta mesma janela, porém, essas alterações serão descartadas assim que terminar a execução.

3.1.3 Hierarchy

Este componente da interface lista todos os objetos presentes na Scene atual. Estes são listados alfabeticamente e de acordo com o seu grau de parentesco. Ou seja, é possível agrupar vários objetos hierarquicamente em que o objeto principal é o pai e os descendentes obedecem a certas características do pai. Um dos exemplos é a origem do referencial: a origem destes objetos passa a ser a posição do objeto pai. Com isto, podemos ordenar e agrupar os nossos objetos de forma simples e lógica.

3.1.4 Project

Neste painel são apresentados todos os recursos que constituem o projeto. É possível para além de importar recursos, criar recursos específicos do Unity (como scripts, prefabs...). Analogamente a outras View's, nesta também é possível pesquisar elementos através da barra superior. Esta View é um espelho da pasta Assets presente na pasta do nosso projeto: qualquer alteração efetuada fora do Unity nesta pasta, é também efetuada no editor. De notar que apesar de ser possível fazer alterações fora do Unity é aconselhável só mexer dentro do editor para evitar erros na linkagem dos recursos.

3.1.5 Inspector

Esta é uma View dinâmica que altera o conteúdo de acordo com o contexto: pode-se editar as características de um objeto, recurso e até mesmo as do editor. É nesta View que, por exemplo, se editam valores em scripts durante a preview do jogo. Se tivermos selecionado um objeto, o que é apresentado, para além das informações do mesmo, são os diversos compondes que lhe estão atribuídos e as suas respetivas variáveis. Caso tenha sido selecionada alguma configuração do editor, aparecerão as variáveis referentes a essa configuração e assim sucessivamente.

3.1.6 Fluxo de Trabalho

O desenvolvimento nesta plataforma passa por importar recursos para o projeto: objetos previamente modulados, plugins, texturas, sons, fontes, etc. Assim que os materiais se encontram incorporados no ambiente, o passo asseguir é alterar as propriedades dos mesmos e incorporá-los na cena de acordo com a nossa preferencia. Componentes adicionais podem ser dados aos objetos de forma a estender as suas capacidades. O painel da hierarquia pode ajudar em alguns aspetos: definir o grau de relação dos objetos e por conseguinte organizar de forma lógica a cena. Caso se identifique um determinado objecto que se pode reutilizar, é boa pratica elaborar um prefab de forma a guardar não só o objecto ou conjunto deles como também as suas propriedades. Assim é possível instanciar de forma mais simples objetos complexos. Posto isto, podemos proceder à elaboração da lógica por detrás do jogo. Recorrendo a um editor externo (Mono ou Visual Studio) é possível elaborar scripts que posteriormente iram ser adicionamos sob forma de componenetes aos objetos, definindo assim as interações dentro do jogo. De relembrar que para serem usados, os scripts necessitam de estar associados a um objecto dentro da cena. No fim da elaboração de uma ou mais cenas, é necessário interligá-las de modo a obter uma certa lógica no jogo. Definir qual cena é o menu, o que acontece quando determinada acção ocorre, que ecrã apresentar numa determinada situação, são alguns dos aspetos a ter em conta nesta etapa. Após isso e de forma a poder correr as várias cenas do projeto, é necessário adicioná-las às definições da build para ser compilada aquando o deploy ou debug. Resumidamente, este é o fluxo de trabalho neste IDE. Cada etapa não tem de ser necessariamente efetuada pela ordem apresentada: a criação de jogos é um processo iterativo e flexível. Dado o paradigma de programação ser semelhante ao implementado no YVision, a integração dos dois é bastante transparente. Para utilizar as funcionalidades do YVision no Unity, basta importar o package do mesmo e obedecer a certas regras de hierarquia consuante o caso

de estudo a implementar. Para uma abordagem mais específica encontra-se em anexo um pequeno tutorial elaborado para aprofundar os conhecimentos desta aplicação. De seguida iremos abordar de forma mais específica a framework YVision.

3.2 YVision

O YVision é uma framework que interliga Natural User Interfaces (NUI) com Game Engines a fim de proporcionar facilmente novas formas de interagir com conteúdo digital interativo. Baseado em .NET, é compatível com Mono, o que permite a compatibilidade, não só com os vários ambiente desktop, mas também com dispositivos móveis e consolas. Atualmente em Beta 4, apenas são suportados alguns sensores (maioritariamente camaras usb), mas está prevista a inclusão de dispositivos mais complexos, como é o caso do Kinect. Como parte integrante da framework, estão embutidos wrappers para algumas das bibliotecas mais usadas no meio: Alvar, Aforge, OpenCV e Firmata. Por fim, encontra-se também disponível uma package para o Unity3D o que permite a integração de todas as funcionalidades disponibilizadas neste motor de jogo.

3.2.1 Arquitetura

Esta framework favorece a composição à herança e baseia-se na ideia de uma arquitetura por componentes. Dado isto, temos então o conceito de Component e Container. Um Container, neste caso denominado de GameObject, é um contentor, um objecto que incorpora vários componentes. Por outro lado, os componentes definem as funcionalidades a adicionar a um determinado contentor. Da mesma forma, o Unity utiliza a mesma filosofia o que torna facil a integração das duas. Outra característica peculiar desta abordagem é a utilização de Behaviours. No YVision o fluxo da aplicação é controlado através de behavior trees. Em vez de recorrer a estruturas de decisão normais, utilizam-se vários behaviors agregados de forma a implementar arvores que gerem o comportamento e vida dos objetos. Estes behaviours não são nada mais do que componentes que podem sere adicionados aos GameObjects.

3.2.2 Behavior Tree

Responsáveis pelo controlo dos objetos, são elas que definem o tempo de vida dos componentes. São corrotinas modulares compostas por vários Behaviors. Podem implementar lógicas simples ou serem constituídas por outras Behavior Trees.

3.2.3 Behaviors

Definem a lógica por detrás dos objetos. São tarefas primitivas executadas em fibers (mais leves computacionalmente do que threads) e têm acesso ao Behavior pai e aos componentes do objecto (GameObject). São tarefas que executam ao longo de um determinado intervalo de tempo e que devolvem os seguintes estados:

- Sucess
- Failure
- Running

Os Behaviors são os constituintes das Behavior Trees e apresentam a seguinte composição:

- Um pai ou nenhum
- Vários filhos ou nenhum

Existem vários behaviors base cada um com uma função específica. Na figura 3.2 está exemplificado um desses behaviors, neste caso um selector. Começando por descrever o Sequence Selector, este permite executar em sequência os seus behaviors filhos. Como referido anteriormente, é devolvido um de três estados: sucesso quando todos os filhos devolverem Success, falha se algum dos filhos devolver Failure e Running quando algum dos filhos estiver a executar. No que diz respeito aos restantes selectors o comportamento é semelhante com as seguintes exceções: no caso do Priority Selector, este devolve Failure se todos os filhos devolverem Failure, o Shuffle Selector executa os filhos aleatoriamente e falha se algum dos filhos devolver Failure, o Stochastic Selector, funciona de forma semelhante ao anterior mas o processo de seleção é ponderado consoante

a prioridade de cada filho e o Concurrent Selector executa os seus filhos em sequência cada iteração e obdece às politicas pré estabelecidas (necessita de uma execução de um filho ou de todos bem sucedida para obter Success e necessita de uma falha de um filho ou de todos para obter Failure). Para além dos selectors existem outro tipo de behaviors standart que permitem a implementação de lógica na aplicação. Como muitos outros paradigmas, estão disponíveis Timers, Loops, Execution Limiters e Suspends.

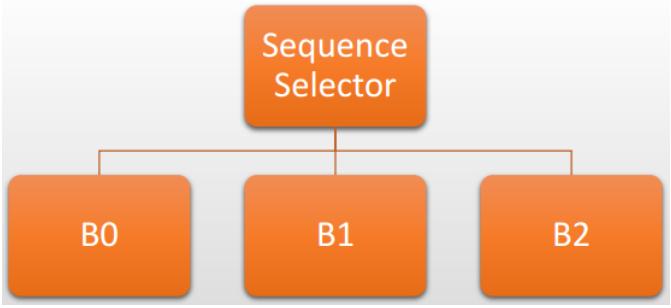


Figura 3.2: Sequence Selector (YVision, 2013)

3.2.4 Graph

No YVision os fluxos de informação, devido a requererem um tempo de processamento inferior, são tratados através de uma pipeline em contraste com o paradigma acima apresentado. Estes fluxos podem representar a informação proveniente de uma webcam: no âmbito do projeto é essencial a rapidez de tratamento destes mesmo dados. Denominadas de Graph's, estas pipelines incorporam vários blocos interligados por Pins de forma a tratar um fluxo de informação num determinado contexto. Para além disso e ao contrário das Behavior trees, os Graphs podem conter loops. O seu funcionamento encontra-se exemplificado na figura 3.3.

Figura 3.3: Graph (YVision, 2013)

3.2.5 Block

No contexto dos Graphs, um block é um encapsulamento de uma tarefa de tratamento de um fluxo de informação. Cada bloco tem três conjuntos diferentes de Pins:

- Input pins - usados para receber os dados a tratar pelo block
- Output pins - usados para devolver a informação tratada
- Property pins - usados para ditar como os parâmetros de tratamento dos dados

Em baixo, na figura 3.4, está representado um bloco de exemplo com os diversos pins.

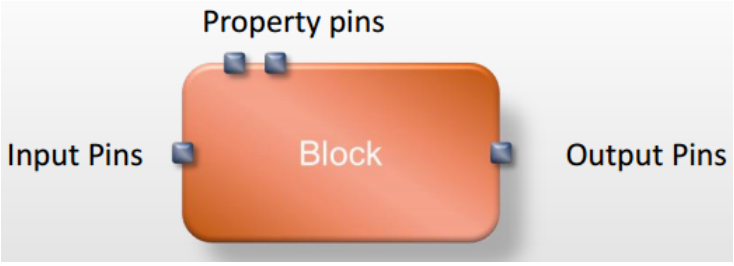


Figura 3.4: Block (YVision, 2013)

3.2.6 Data Flux

O fluxo de informação nestas pipelines realiza-se através da ligação dos diversos blocos que a constituem. Esta ligação é feita entre os pins dos blocos: os dados deslocam-se dos pins de output para os pins de input dos blocos ligados. Consoante a complexidade do Graph, é possível ter vários pins Input ligados a um só pin Output . Para além disso existem casos em que num Graph se encontra implementado um loop.

De forma a separar e organizar operações distintas num mesmo Graph, podem-se separar os fluxos por contextos diferentes (representado na figura 3.5). Assim cada contexto executa os seus blocos sincronamente. Para além disso, os pins dos blocos propagam automaticamente a ocorrência de eventos entre contextos diferentes.

De forma a controlar o fluxo de dados numa pipeline, existem blocos de controlo predefinidos. Estes blocos são o Join, Merge, Gate e Buffer. Usado para juntar os vários fluxos input, o Join aguarda os dados de todos os pins input e combina a informação dos dois. É utilizado, como por exemplo, para juntar a imagem proveniente de duas camaras com diferentes framerates. Semelhante ao Join, o Merge efetua a mesma operação mas apenas necessita de informação num dos pins. No que toca ao Gate, este permite a descarga do fluxo de dados. É normalmente utilizado no final das pipelines para disponibilizar a informação tratada para outros componentes. Por fim,

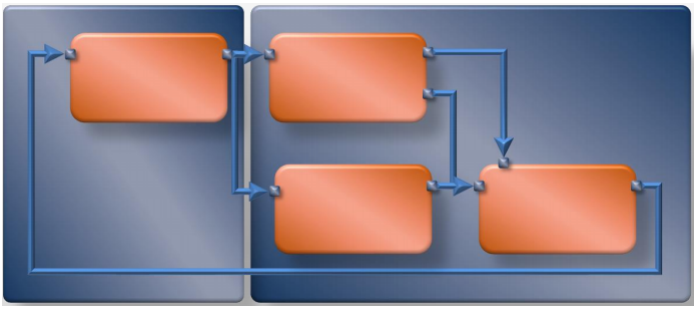


Figura 3.5: Graph Context (YVision, 2013)

o Buffer, como o próprio nome indica, permite a existencia de delays no Graph. Utiliza o algoritmo First in First Out e disponibiliza a informação assim que ultrapassar a sua capacidade.

3.2.7 Bibliotecas Externas Incorporadas

Para além de todas a funcionalidades implementadas nesta framework, ainda estão integradas as bibliotecas mais utilizadas no campo do tratamento de imagem e realidade aumentada. As bibliotecas incluídas são o Aforge, OpenCV e Alvar. Dependendo da implementação de cada uma delas (algumas não se encontram implementadas em C#) estão incluídos wrappers para poderem ser acedidas. Posto isto, o modo de funcionamento dos diversos filtros e do processamento de imagem é efetuado recorrendo aos Graphs e blocos implementados no YVision. Isto é, as funcionalidades de cada biblioteca podem ser incorporadas nesta framework através do encapsulamento em blocos e posterior introdução nas pipelines de processamento de imagem. Dando o exemplo da utilização do filtro HSL do Aforge, primeiro é necessário criar o bloco com essa funcionalidade, definir os pins de saída e entrada e definir as suas propriedades. No caso deste bloco, como pins de entrada temos o fluxo de imagens provenientes do bloco anterior e como saída temos esse mesmo fluxo alterado consoante as propriedades definidas. Posto isto, na pipeline de processamento, é necessário instanciar o bloco, definir o seu contexto e conectar os pins consoante o uso (neste caso o pin Input estará ligado ao Output do bloco anterior e o Output ao Input do posterior). O bloco terá de aceitar o formato proveniente do bloco anterior ou se se verificar o caso, proceder à sua conversão. Assim num processo simples é facil implementar novas funcionalidades a uma aplicação que utilize o YVision. De seguida serão apresentadas as bibliotecas incluídas na framework.

Aforge

AForge.NET é uma biblioteca open source desenvolvida em C# com o intuito de disponibilizar funcionalidades de tratamento de imagem, inteligência artificial e realidade aumentada. (Aforge.Net, 2014) Esta encontra-se subdividida em várias bibliotecas com destaque das seguintes:

- AForge.Imaging - contém rotinas de processamento de imagens e filtros
- AForge.Vision - biblioteca de computer vision
- AForge.Vídeos - conjunto de funcionalidades de processamento de vídeos

É disponibilizada através do seu repositório onde podemos encontrar não só o código fonte como também exemplos de aplicações que demonstra as funcionalidades e a sua implementação. A sua documentação é disponibilizada em formato HTML o qual pode também ser acedido online. Como é uma biblioteca em constante desenvolvimento, podem ser incluídas outras funcionalidades ao longo do seu ciclo de vida.

OpenCV

OpenCV (Open Source Computer Vision Library) é uma biblioteca open source de computer vision e machine learning. (OpenCV, 2014) Com mais de 2500 algoritmos otimizados para o processamento de imagem e com a utilização nas mais variadas áreas (desde juntar imagens de streetview a detetar intrusões no sistema de vigilância de Israel) é uma biblioteca conceituada nesta área. Os algoritmos desenvolvidos podem ser utilizados para a deteção e reconhecimento facial, identificação de objetos, classificação de ações humanas em vídeoss, efetuar o tracking dos movimentos da camara, tracking de objetos, extração de modelos 3D de objetos entre outros. Mantida por mais de 47 mil pessoas, o OpenCV disponibiliza interfaces C++, C, Python, Java e Matlab e suporta Windows, Linux, Andoid e Mac OS. Para além disso, está a ser desenvolvido suporte a CUDA e OpenCL.

Alvar

ALVAR é uma biblioteca desenvolvida específicamente para a criação de aplicações de realidade aumentada. (Alvar, 2014) Oferece ferramentas de alto nível e métodos para criar facilmente aplicações de realidade aumentada. Para além disso, também disponibiliza interfaces para liguagens de baixo nível de forma a providenciar ferramentas para o desenvolvimento de soluções customizadas e outros algoritmos. Apresenta como principais pontos fortes uma boa deteção de marcadores com deteção fiável de distancias, rotações e dimensionamentos, possibilita a criação de marcadores próprios, dispões de ferramentas para a calibração de sensores opticos e dispõe ainda de deteção recorrendo a imagens pré-definidas. Atualmente, esta biblioteca está disponível em Windows e Linux e apenas depende da biblioteca OpenCV para funcionar. Consoante a sua utilização, será necessário recorrer a outras bibliotecas de renderização mas o Alvar não depende destas para implementar as suas funcionalidades.

3.3 Preparação do Ambiente de Desenvolvimento

Primeiramente, instalou-se e testou-se o editor Unity. Um dos recursos gerados foi o pequeno tutorial incluído em anexo. Posto isto procedeu-se à integração do YVision com o Unity. O YVision (na versão Beta 4 à data), encontra-se distruibuido num ficheiro comprimido de acesso restrito: não está disponível publicamente. Encontra-se subdividido nas seguintes pastas:

- Licenses
- Packages
- Resources
- Samples
- Tools

Na pasta Licenses encontram-se as licenças de distribuição e uso de todo o software contido na distribuição (bibliotecas externas, extensões e até a própria framework). Packages contém os plugins a serem importados para Unity de forma a podermos estender as funcionalidades do editor com esta framework. Avançado para Resources,

nesta pasta estão presentes os recursos utilizados nas demonstrações presentes na pasta Samples: imagens, prefeabs e até materials. Como referido anteriormente, na pasta Samples estão reunidos os demos das várias funcionalidades da framework. Exemplos de deteção de caras, movimento e até deteção de marcadores, encontram-se nesta pasta em formato de projeto de Unity. Em Tools estão disponíveis os dlls e alguns utilitários distribuídos em conjunto com a biblioteca Alvar. De forma a utilizar esta framework em conjunto com o Unity é necessário importar os plugins desejados e obedecer a uma certa hierarquia dentro da cena. De seguida iremos explicar como se encontra organizado um projeto desta natureza e parte do desenvolvimento inicial antes da criação do produto final.

3.3.1 BoxFall

De forma a explorar as várias funcionalidades disponibilizadas pela framework, começou-se por estudar o exemplo BoxFall presente no pacote disponibilizado pela YDreams. Este exemplo consiste na interação do utilizador, recorrendo a uma webcam, com vários objetos que caiem aleatoriamente no ecrã. 3.6

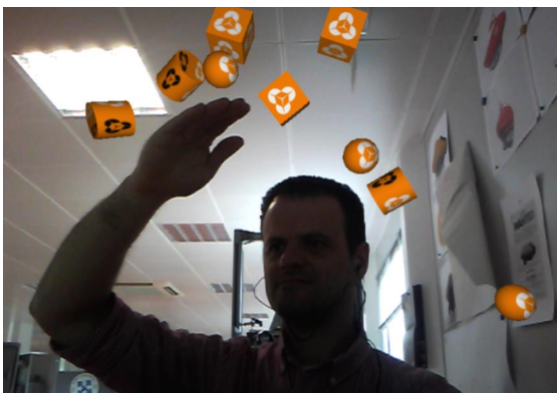


Figura 3.6: Boxfall em funcionamento (YVision, 2013)

Iremos começar por apresentar a estrutura deste exmplo, apresentar os seus constituintes e descrever sucintamente a sua implementação. Começando pela organização dos objetos, os constituintes desta demonstração encontram-se na imagem da hierarquia apresentada em baixo. 3.7

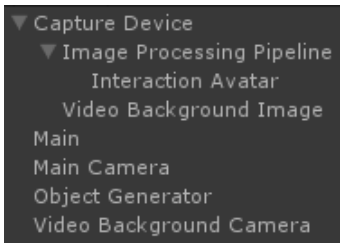


Figura 3.7: Hierarquia do Boxfall (YVision, 2013)

Main

Behavior tree principal responsável por controlar o fluxo do projeto. É neste objeto que se encontra o script que dita quando termina a sua execução.

Main Camera

Objecto Camera do Unity. Responsável por definir a área visivel para o utilizador final. Representa a imagem mostrada no ecrã no memento da execução do jogo.

Capture Device

Objeto que implementa a captura de imagem através da webcam. Contém os vários objetos responsáveis pela captura e processamento da informação proveniente da camara e interação com os restantes objetos da cena. Este é um objeto vazio apenas com o script "WebCamCaptureDevice" presente no package da Yvision. Para além disso, tem como filhos os objetos "ImageProcessingPipeline" e "VídeosBackgroundImage". Alternativamente poderá ter mais pipelines de processamento de imagem caso se justifique (como por exemplo adicionar efeitos visuais que não entrem na deteção de objetos).

Image Processing Pipeline

Implementa a pipeline de tratamento de imagem a fim de produzir a interação com os restantes elementos da cena. Tem como componentes o script "BackgroundContext" da biblioteca do Yvison e o "ImageProcessingPipeline" e "ContourBroker". Tem como "filho" o objeto "InteractionAvatar". O processamento de imagem segue a seguinte

lógica:

- Capturar uma frame da webcam
- Encolher o tamanho de forma a melhor a performance
- Remover o fundo
- Encontrar os contornos
- Gerar a mesh a partir dos contornos descobertos anteriormente
- Associar a mesh ao objeto "InterationAvatar"

Para implementar esta lógica, o script "ImageProcessingPipeline" é responsável pelo processo de captura, aplicação de filtros (resize, mirror...), deteção dos contornos (através do service provider "ContourBroker") e geração de mesh. Efetua estas operações através da implementação de uma pipeline: constituída por vários blocos, cada um com uma função diferente, o output de um é o input doutro.

Vídeos Background Image

Objeto do tipo GUITexture (tipo nativo do Unity) que apresenta a imagem capturada pela webcam. Implementa o script "VídeosBackgroundBehaviourTree".

Object Generator

Behavior tree que trata da criação e destruição dos objetos que caiem do topo do ecrã. Este behavior recorre a um "Stochastic Selector" de forma a instanciar os prefabs dos vários objetos de acordo com uma dada probabilidade.

3.4 Conclusão

Foi abordado, neste capítulo, o ambiente de desenvolvimento escolhido: Unity + YVision. Foi apresentado o motor de jogo utilizado e a sua interface, descritas as suas

funcionalidades e a sua organização interna e explicado, embora que de forma sucinta, o seu fluxo de trabalho. Posteriormente, foi exposta a framework YVision. Foi apresentada a sua arquitetura, descrevendo cada constituinte e o modo como se interligam entre si e descritas as bibliotecas externas a que esta recorre para o tratamento de dados. Para finalizar, referimos como foi preparado o ambiente de desenvolvimento e apresentado um pequeno exemplo de como funciona o mesmo.

Capítulo 4

Solução Proposta

Neste capítulo abordaremos todos os aspetos que foram tidos em conta aquando a elaboração da solução proposta. Iremos descrever o processo de decisão tomado e os vários fatores que influenciaram algumas das decisões importantes do projeto. Apresentaremos então o público-alvo, quais os requisitos para a elaboração de jogos para o mesmo e os pontos chave a ter em conta no momento da implementação. Também iremos identificar os vários géneros existentes de jogos que podemos desenvolver e descrever os jogos elaborados ao nível das respetivas regras e mecânicas.

4.1 Público-alvo

Antes de proceder à elaboração dos respetivos jogos, tivemos que estudar o público-alvo. Compreender a partir de que idades fazia sentido utilizar este tipo de aplicações e moldar o resultado consoante as características dos utilizadores. Existem, antes de mais e independentemente da idade, uma série de princípios básicos a ter em conta quando se desenvolve software infantil. (Ellen Wolock, 2006) São eles:

- Diversão: as crianças escolhem as atividades que gostam de fazer e evitam atividades que são frustrantes, estáticas ou chatas. As primeiras atividades devem ser intuitivas, fáceis e divertidas.
- Controlo: as crianças evitam atividades em que não têm nenhum controlo. Um bom software aumenta a sensação de controlo das crianças, fornecendo um ambiente em que as suas ações têm impacto. A aplicação deve ser responsiva e enviar mensagens de controlo frequentes. Uma interface nítida e ágil aumenta esta sensação.

- Interesse: As crianças têm mais probabilidade de se envolver numa atividade que suscite o seu interesse. Cada jogo deve oferecer algo novo ou incorporar elementos abertos. As crianças adoram surpresas e novidades. O jogo deve ir ao encontro dos interesses das crianças. Elas gostam de personagens com quem se identificam, boas histórias, música de qualidade, humor, e temas familiares.

- Sentimento de Competência: As crianças desenvolvem sentimentos de competência, se pensarem que têm uma hipótese razoável de sucesso. O desafio deve estar na própria atividade, e não na operação física do programa. O programa deve incluir elementos que dependem do sucesso anterior, permitindo um maior desafio adaptado às habilidades da criança. A estimulação adequada e nívelamento das atividades são fundamentais, pois se for demasiado rápido, a criança constrói um banco de fracassos, em vez de um banco de competências. Se for muito lento, a motivação intrínseca diminui.

Reduzindo a faixa de idades para os 3 a 6, torna todo o processo de desenvolvimento um bocado mais específico. No entanto é preciso ter em conta a limitação das crianças, as suas preferências e capacidades. Nestas idades existem tantas competências dispares que é necessário identificar e dividir as principais por grupos etários. Assim agrupámos as crianças dos 3 aos 4 anos, dos 4 aos 5 e dos 5 aos 6. Uma criança compreendida no primeiro grupo de idades é capaz de reconhecer a maioria das cores e identificar formas geométricas simples (é o caso do quadrado, círculo, triangulo...). Consegue também compreender os conceitos de ïgualë d̈iferente; seguir duas instruções simples na ordem correta e manusear alguns utensílios com sucesso (lápis, régua...). Estas capacidades aliadas à demonstração de conhecimento de algumas relações, reconhecimento de algumas letras, capacidade de completar puzzles de 4 peças e construir torres com 10 ou mais blocos, fazem desta faixa etária um bom alvo para jogos de lógica simples que explorem princípios básicos de relação e de numeração. (WebMD, 2014a) Passando para o grupo a seguir, a grande diferença prende-se com o facto de que nestas idades a perceção do mundo à sua volta aumenta significativamente: conseguem distinguir e ordenar objetos de acordo com algumas características (tamanho, cumprimento), conseguem classificar consoante o tipo (animais, comida, roupa...), começam a perceber alguns puzzles lógicos, as suas capacidades motoras melhoram e conseguem acompanhar regras bem definidas podendo até exprimir alguma opinião em relação à atividade que estão a praticar. Nestas idades já é possível desenvolver jogos mais complexos e que requeiram mais destreza. (WebMD, 2014b) No que toca ao último grupo, este já é um grupo que já dispõe de alguma experiencia com computadores: normalmente utilizam tablets com frequência e já navegam na internet com auxilio dos pais. A. Driscoll (2014) Já é uma idade que disfruta em pleno de atividades com computadores: ex-

plora as diferentes aplicações, consegue interagir de forma relativamente facil com os dispositivos e sabe normalmente as regras de utilização. Dado o seu intelecto estar mais desenvolvido em relação às faixas etárias anteriores, é também um grupo que possibilita um maior feedback no que toca aos testes das aplicações: crianças destas idades já conseguem exprimir-se quando têm um problema e normalmente dizer o que acham que está errado ou menos correto. Thomas Keenan (2006) Após a análise destes grupos de idades, decidimos escolher o grupo compreendido entre os 5 e 6 anos devido à sua capacidade motora e cognitiva.

4.2 princípios para desenvolvimento dos Jogos

Muitas das decisões no que toca a desenvolvimento deste género de software prendem-se com o facto de tornar a experiencia mais adequada para um público que não tenha tanta experiencia com este tipo de tecnologias.(Chiasson, 2005) Assim sendo as principais ideias a ter em conta passam por:

- Elaboração de menus percetiveis que não necessitem de leitura para serem compreendidos
- Criação de menus de uma camada que disponibilizem acesso direto às atividades
- Tempos de espera curtos de forma a igualar o tempo de atenção das crianças
- Interação rápida e fidedigna
- Fluxo do jogo sem interrupções (vídeos de entrada, animações, transições animadas entre cenas...)
- Tolerância a grande fluxo de informação sem apresentar falhas (exemplo de vários clique em simultâneo no teclado)
- Mecanismo de ajuda transmitido por uma voz percetível e dentro do mesmo contexto
- Ícones grandes, percetíveis e que transmitam significado às crianças. Evitar imagens que transmitam a ideia de botões mas que não deem para clicar
- Feedback constante para transmitir a sensação de progresso. Também é necessário apresentar mensagens de incentivo em caso de perda, ganho e sugestões para facilitar o jogo

Para além disso e como forma de cativar o público-alvo, neste caso as crianças, o jogo deve garantir uma iniciação compilados e que permita preferencialmente sucesso nos primeiro 10 segundos de experimentação. Caso contrário, deve fornecer ajuda e alguns tópicos que facilitem a chegada ao sucesso quando a criança demonstra dificuldades. O modo de jogar deve ser consistente e não confundir o jogador. É imperativo retirar o máximo de obstáculos possíveis: deve ser a criança que está em controlo e não o jogo. Definir bem os objetivos, quer numa explicação inicial ou durante o jogo, e inserir o ideal do jogo num contexto com significado para a criança. Possibilitar vários caminhos para o sucesso e preferencialmente disponibilizar sempre uma solução independentemente do problema. Basicamente é preciso pensar num jogo para jogadores ocasionais, que tenham pouca ou nenhuma experiencia com este tipo de software Fisher (2014).

4.3 Interação

De forma a definir que tipo de implementações iriamos desenvolver, tivemos em conta vários fatores no que toca às mecânicas da realidade aumentada. Para isso foi necessário identificar que tipo de interação iriamos adotar. Entre marcadores e deteção de movimento, decidimos optar pelo primeiro, visto o feedback obtido pela deteção de movimento não ser preciso o suficiente (sensores como o Kinect Microsoft (2014) poderiam contornar esta falha, mas atualmente apenas são suportadas webcams no YVision). Nesta categoria, ainda temos mais duas opções a ter em conta: marcadores de cor e marcadores do género de QRCodes (técnicos). Ambos mostraram ser mais fidedignos que a deteção de movimento sendo o segundo o mais preciso dos dois.

Na primeira categoria de marcadores, os marcadores de cor, a chave para uma boa deteção prende-se com a escolha da cor do marcador, cores mais dispares, cores que não se confundam facilmente com o fundo são a melhor opção (no capítulo de testes iremos referir que cores em concreto foram utilizadas para os respetivos jogos). Isto facilita o contraste com outros elementos da imagem o que facilita a deteção da cor que por conseguinte favorece a interpretação dos movimentos. Para o segundo caso de marcadores, os marcadores técnicos, não existem tanto essa preocupação. Dadas as suas características, a deteção dos mesmos é apenas afetada pela iluminação do meio. Visto serem compostos por cores contrastantes, a sua deteção é muito mais eficaz. Para além disso ainda temos a opção de utilizar dois tipos dentro desta categoria: o single marker ou multi marker. Explicados melhor no capítulo da implementação, ambas as opções são mais precisas em caso de fraca luminosidade.

Independentemente das características dos marcadores, a interação, no âmbito deste projeto é efetuada da seguinte forma: procede-se à deteção do marcador, obtêm-se os dados do mesmo (posição, rotação, distancia quando aplicável) e integra-se essa informação no jogo, quer seja a mover um objecto ou qualquer outra ação. Esta é a mecânica base por detrás da interação com os jogos criados. Dependendo do jogo, as regras serão diferentes e por conseguinte, a jogabilidade também.

4.4 Tipos de Jogos

Para além da interação em si, também tivemos de definir que género de jogos iriamos elaborar. Tendo em conta o público alvo o objetivo seria desenvolver jogos simples em que o foco principal seria a destreza e a coordenação motora. Jogos com regras complexas e requisitos de atenção elevados ficaram fora das escolhas. Por isso, foram tidos em conta jogos que normalmente são jogados por crianças desta idade e que poderiam ser implementados recorrendo à realidade aumentada: puzzles, jogos simples de lógica, ordenação e categorização de conjuntos, entre outros. Tomando como referência os primórdios dos jogos de vídeos, inspirámo-nos em alguns clássicos aquando o desenvolvimento. Jogos como o Pong e o Tetris, são exemplos disso mesmo: são jogos simples, com regras básicas e com jogabilidade que poderá ser implementada em RA. Categorizando-os, estes jogos são do tipo casual, arcade, simples de curta duração. Este foi o tipo de jogos que identificámos como mais adequado para realização no âmbito do projeto.

4.5 SinglepLayer vs Multiplayer

Tendo sido definido anteriormente os tipos de jogos a desenvolver, a próxima questão que se levantou foi que modos implementar: singleplayer ou multiplayer. Ambos os modos têm as suas vantagens e desvantagens no que toca ao projeto. Falando em primeiro lugar do modo singleplayer, este tem como vantagem a possibilidade de se desenvolver jogos com regras e jogabilidade mais complexas. É também o modo de jogo em que se pode validar mais facilmente a implementação e observar todos os aspetos envolventes da utilização do jogo visto poder ser testado num ambiente mais controlado. Como se encontra focado apenas num jogador, é um bom candidato para a apresentação da tecnologia a novos utilizadores. Em contrapartida, não oferece uma experiência tão competitiva e social como se fosse jogado a mais que um jogador. Em multiplayer, o

facto de existir mais que uma pessoa a interagir com o mesmo jogo, embora possibilite jogabilidades que não poderiam ser implementadas em singleplayer, também trás outros aspetos a ter em conta. Neste modo de jogo e dadas as características da plataforma escolhida, o espaço para cada jogar passa para metade (no caso de dois jogadores) visto a área disponível para o jogo ser a mesma (apenas é utilizado um monitor ou projetor). Isto torna mais dificil a interação e, visto que os jogadores partilham o mesmo espaço físico, facilita a interferência entre os dois (contacto físico).

Cada um com as suas características, ambos os modos permitem testar diferentes aspetos da realidade aumentada. Podem validar não só a capacidade da framework de suportar os diferentes requisitos como também demonstrar vários cenário em que a RA traga mais-valias.

4.6 Jogos Elaborados

De seguida apresentaremos as informações referentes às mecânicas e regras de cada jogo implementado. Muitas destas decisões foram baseadas nos princípios e capacidades do público-alvo apresentados anteriormente neste capítulo. Aqui apenas descreveremos as regras e as mecânicas de cada jogo de forma sucinta. Como forma de experimentar a framework a ser utilizada, foram desenvolvidos inicialmente dois jogos, o FruitCatch e o AlvarPong: o primeiro sendo um simples jogo de apanha de fruta e o segundo uma implementação de realidade aumentada do famoso Pong (as regras e implementação serão abordadas mais pormenorizadamente no capítulo seguinte). Estes jogos foram essenciais para definir e aprimorar as mecânicas de interação: foi nos possível identificar as falhas e melhorar o controlo e deteção dos marcadores a fim de produzir jogos mais complexos a nível de jogabilidade. Posto isto, decidimos então, após testar e analisar os jogos anteriores, desenvolver mais dois jogos com focos destintos: o primeiro teria como função introduzir a tecnologia e o segundo testar a aprendizagem dos utilizadores. Temos então o Falling Fruits, um jogo com pouco raciocino e de facil controlo, e um spinoff do Tetris, onde é testada a destreza e o racicionio rápido. Com estes dois tipos de jogos conseguimos, aquando a elaboração dos testes, obter dados para as duas principais questões deste projeto: a capacidade da framework de desenvolver aplicações de realidade aumentada e a sua utilidade no sei da educação. Informação mais aprofundada da implementação está presente no próximo capítulo.

4.6.1 Fruit Catch

Um jogo multi-jogador em que o objetivo consiste em apanhar o maior número de peças de fruta consoante a respetiva caixa. O jogo desenrola-se no dado período de tempo que pode ser configurado. Neste período, o ecrã encontra-se dividido em duas áreas verticais: uma para cada um dos dois jogadores. Nestas áreas encontram-se caixas (uma em cada área) que são controladas pelos jogadores de forma a apanhar a fruta respetiva a cada caixa. A fruta a apanhar cai do topo do ecrã de forma aleatória. Cada fruta apanhada corretamente corresponde a uma bonificação na pontuação e o cada fruta não pertencente à caixa, corresponde a uma penalização. O jogo termina quando o tempo esgotar e ganha o jogador que tiver melhor pontuação. De seguida é mostrado um ecrã com as melhores pontuações obtidas. Caso a pontuação obtida for superior a alguma das existentes no high score, esta é introduzida no mesmo. Para começar o jogo, basta posicionar o marcador de cada jogador conforme a posição apresentada no ecrã.

4.6.2 Alvar Pong

O jogo Alvar Pong é uma interpretação do famoso Pong recorrendo à realidade aumentada. Desenvolvido para testar os marcadores Alvar, as regras deste jogo são simples: jogado a dois, cada jogador controla um paralelipípedo no eixo vertical tentando evitar que a bola de jogo ultrapasse a borda do ecrã imediatamente por trás do respetivo paralelipípedo. Para começar o jogo, cada jogador tem de colocar o seu marcador no sitio devidamente marcado no ecrã. Posto isto, a bola presente no centro do campo movimenta-se para o lado esquerdo ou direito aleatoriamente. Cada vez que toca no limite superior, inferior, ou nos paralelipípedos, é ricocheteada. Cada bola vale um ponto e ao fim de 5 pontos jogados o jogo termina indicando o jogador vencedor.

4.6.3 Falling Fruits

Falling Fruits é um jogo singleplayer onde é testado o raciocínio do utilizador e a destreza de forma a escolher os frutos certos para uma determinada receita. O objetivo do jogo é introduzir apenas os alimentos certos, definidos aleatoriamente no início de cada jogo, numa cesta que se movimenta horizontalmente no ecrã antes do tempo terminar. Para isso, o jogador controla uma colher, através de um marcador de cor,

com o intuito de desviar os objetos que caem do topo do ecrã (para fora ou para dentro da cesta conforme o caso). De forma a distinguir os certos dos errados, existe um conjunto de alimentos, mostrados ao utilizador, que vão desaparecendo à medida que são introduzidos na cesta. Para além de alimentos, também cairão outros objetos que, se introduzidos na cesta, contam como alimentos errados. Se o jogador conseguir apanhar os alimentos que foram definidos no início do jogo, este será premiado de acordo com a rapidez com que o fizer. Por cada alimento ou objecto que seja introduzido no cesto que não faça parte do conjunto a apanhar, o jogador será penalizado: é reduzir o tempo restante de jogo. O jogo termina quando o jogador apanhar todos os alimentos certos ou quando o tempo terminar. Para que se dê início ao jogo, o jogador terá de apresentar no ecrã, o marcador que controlará a colher. Depois, durante 5 segundos, serão mostrados ao jogador quais os alimentos que este terá de apanhar. Durante o jogo, o jogador terá na faixa lateral esquerda os alimentos que faltam apanhar. Do lado direito, o tempo restante é mostrado em formato numérico e representado por uma barra cilíndrica cuja cor se altera consoante o tempo restante. O sistema de pontuação funciona da seguinte maneira:

- Conclusão com sucesso até 1/3 do tempo de jogo equivale a 3 taças
- Conclusão com sucesso até 2/3 do tempo de jogo equivale a 2 taças
- Conclusão com sucesso após 2/3 tempo de jogo equivale a 1 taça
- Falha nos objetivos não apresenta recompensa

O jogador é ainda premiado com uma imagem de jogo captada logo após o jogo ganho, no qual figurará o número de taças atribuídas. Caso o tempo definido se esgote sem o objetivo atingido, o jogador perde sem nenhuma recompensa. Para jogar novamente, basta repetir o processo referido anteriormente.

4.6.4 Tetris

Utilizando as peças do famoso jogo Tetris, o objetivo deste jogo é empilhar o maior número de peças possíveis sem deixar cair nenhuma. Sendo um jogo em que se testa a perícia e o raciocínio do jogador, este é apenas single player. No início do jogo, é apresentada no ecrã, uma base onde serão empilhadas as peças e uma peça no canto superior direito. Para colocar a peça apresentada, o utilizador deverá movimentar o marcador até a onde a peça se encontra e arrastar a mesma para a base. A peça a ser

controlada movimenta-se de acordo com a orientação e posição do marcador no espaço 2D. Uma vez colocada, aparece outra peça no canto superior e o conceito repete-se. De salientar caso a peça toque na base ou noutra peça já colocada na mesa, o utilizador perde o seu controlo. Quanto mais peças empilhar e quanto mais alta for a torre produzida, maior será a pontuação. O efeito da gravidade é aplicado às peças o que torna o jogo de destreza superior aos restantes jogos. Caso uma peça caia para baixo do nível da base, o jogo termina. Analogamente ao jogo T̈alling Fruits;, este também dispões de um sistema de recompenssas e de scores que poderá ser alterado conforme isso seja necessário.

4.7 Conclusão

Como forma de idealizar os jogos a desenvolver, foi necessário estudar o público-alvo e os vários princípios utilizados para o desenvolvimento para estas faixas etárias. Neste capítulo analisámos todas essas nuances e estudámos que tipos e modos de jogos implementaríamos. Posto isto, descrevemos cada um dos jogos a implementar e o modo de interação dos mesmos.

Capítulo 5

Implementação

No que diz respeito ao desenvolvimento, começámos por testar várias formas de implementar funcionalidades no jogo. Como este projeto se encontra inserido num consórcio e como o software em utilização é relativamente recente e ainda se encontra em desenvolvimento, parte do trabalho inicial prendeu-se com a aprendizagem da ferramenta em si. Além disso, dada a ligação próxima com a empresa responsável pela framework, têm sido ao longo do projeto reportados bugs e indicadas sugestões de forma a suportar o desenvolvimento da mesma. Posto isto, foram elaboradas várias aplicações e jogos de forma a estudar o funcionamento da framework e analisar possíveis casos de estudo. Foram testadas várias formas de interação e implementados diversos casos de estudo, isto numa primeira fase. Após esta fase preliminar de testes e desenvolvimento, aprimorámos os vários jogos e definimos dois casos de estudo em concreto. Aplicando algumas das guias no que toca a desenvolvimento de jogos para crianças, modificámos alguns aspetos e introduzimos outros mecanismo de forma a tornar os jogos mais apelativos. Assim de seguida iremos descrever os vários jogos desenvolvidos e mostrar alguma da evolução conseguida através de testes e introdução de algumas normas. Iremos também neste capítulo, abordar o trabalho efetuado antes de começar a desenvolver o produto final: desenvolvimento de filtros de cor e ferramentas auxiliares.

5.1 Desenvolvimento Preliminar

Passando para a descrição do que foi elaborado, após o teste do exemplo referido no capítulo do Ambiente de Desenvolvimento, foram definidos alguns objetivos para uma primeira abordagem. Como a interação com os objetos eram apenas colisões simples, decidiu-se animar a destruição dos mesmos. Observou-se também que a imagem

proveniente da webcam não se encontrava espelhada, o que dificultava a interação do utilizador. Para isso, era preciso implementar algum mecanismo que efetua-se o espelho da imagem.

Começando pela animação dos objetos, para a implementação desta funcionalidade, foi preciso modelar um novo objecto, desta vez destruído, de modo a ser trocado pelo objecto original aquando a deteção da colisão. Depois disso, foi alterado o behaviour associado ao objecto de forma a proceder a essa mesma substituição. Assim foi possível implementar um mecanismo simples de destruição de objetos. No caso do espelho da imagem, o processo foi um pouco mais complexo. Como o código que existia para efetuar esse operação não suportava os dados provenientes da webcam, tivemos de alterar o código fonte da classe Mirror da biblioteca Aforge (biblioteca utilizada pelo YVision) e encapsula-lo num bloco de forma a ser utilizado na pipeline de captura e tratamento de imagem. A lógica por detrás deste filtro é a seguinte: trocar sucessivamente os pixéis com coordenadas inversas em X ou Y dependendo do tipo de mirroring efetuado. Como não queremos alterar a funcionalidade do filtro, apenas queremos suportar um novo formato, tivemos que modificar o código de forma a aceitar o novo tamanho dos pixéis.

Começámos então por adicionar um novo formato ao dicionário (32bppArgb é o formato disponibilizado pela webcam) de forma a aceitar o novo tipo de dados. De seguida, como este novo formato implica que cada pixel tenha 32 bits, tivemos de acrescentar uma nova condição: se o formato for 32bppArgb, o offset será 4 bytes (32 bits / 8 = 4 bytes) e tivemos também de trocar, não só os parâmetros RGB como também o Alpha (característica deste novo formato).

Com isto já temos o nosso bloco mirror a aceitar o formato produzido pela webcam. Como por defeito, no Unity, não é suportada a keyword "unsafe", necessária para efetuar operações com ponteiros em C#. Para podermos correr este código necessitámos de criar o ficheiro "smcs.rsp" com o conteúdo "-unsafe" e meter na pasta "Assets" do projeto. Em alternativa á alteração do bloco, pode-se alterar o valor da escala da textura associada ao stream da camara e assim inverter a imagem apresentada na mesma.

5.1.1 Filtros de cor

De modo a conseguir alguma forma de interação com o utilizador (para além da deteção de movimento já disponibilizada nos exemplos da framework), decidiu-se elaborar um bloco que efetuasse um filtro por cor à imagem proveniente da camara. Na figura 5.1 está representada a filtragem da cor vermelha. Com isto seria possível reduzir as limitações impostas pela abordagem anterior e introduzir novas maneiras de interação: possibilidade de interação com mais utilizadores ao mesmo tempo, utilização de objetos mais adequados à tematica do jogo e aumentar a fiabilidade da deteção.

Figura 5.1: Filtro do vermelho recorrendo ao bloco elaborado

Recorrendo ao filtro HSL, disponível pela biblioteca Aforge incluída na framework YVision, foi desenvolvido então um bloco que filtra a imagem da webcam de acordo com os valores passados ao mesmo. A escolha do formato HSL em oposição ao mais tradicional RGB prendeu-se com o facto de ser mais simples a sua utilização e definição. No modelo RGB são utilizadas as intensidades de três cores aditivas (vermelho, verde e azul). Embora mais simples e intuitivo para uso por equipamentos eletrónicos, o mesmo não se verifica com o ser humano, por isso optámos pelo modelo HSL. Baseado no modelo RGB, é definido pelos valores de hue (cor), saturation (saturação) e lightness (luminosidade). 5.2 Assim é mais facil definir uma determinada cor e os seus aspetos.

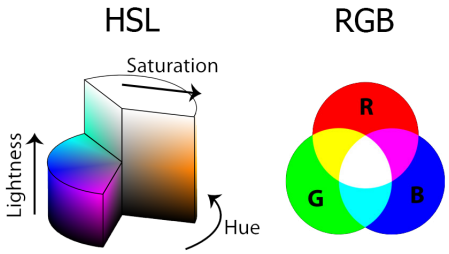


Figura 5.2: HSL vs RGB

Sendo um formato mais simples de usar, a sua utilização implica uma conversão de formatos: o formato proveniente da webcam é RGB. Outro aspecto tido em conta foi

o facto de no Yvision a escala de cor ser tratada de maneira diferente: o valor em 0 é o azul em vez de ser o vermelho. Posto isto e de forma a estudar a integração com o OpenCV, desenvolveu-se o mesmo género de filtro mas desta vez recorrendo a esta biblioteca. Assim podemos ambranger um maior numero de plataformas e conhecer o funcionamento das várias tecnologias envolvidas. A transposição do filtro para esta biblioteca ocorreu sem grandes problemas. Foi encapsulado da mesma forma e pode ser utilizado mas mesma maneira que o anterior. O único problema é a diferença das escalas: neste caso o hue pode tomar valores entre 0 e 180 em oposição ao padrão de 0 a 360. Posto isto, iremos de seguida explicar o desenvolvimentos dos jogos implementados.

5.1.2 Marcadores Alvar

Para além da deteção de cor explicada anteriormente, com a disponibilização da ver são Beta 4 do YVision, foi introduzida a compatibilidade com a biblioteca Alvar (referida no capítulo do Ambiente de Desenvolvimento). Esta biblioteca disponibiliza a deteção de marcadores técnicos o que possibilita uma interação mais refinada através da webcam. Estes marcadores, para além das funcionalidades permitidas pelos marcadores de cor, também possibilitam a deteção de rotação e profundidade (funcionalidades não disponíveis nos anteriores). Isto permite a implementação de mecânicas de interação diferentes e por conseguinte, outros tipos de jogos.

A introdução desta biblioteca para além de permitir o uso deste género de marcadores, também disponibiliza a deteção de multi marcadores 5.3. Constituídos por um grupo de marcadores técnicos, estes para além do padrão de cada um, têm como característica a relação de posição entre eles. Isto permite a deteção do marcador mesmo que não estejam a ser detetados todos os marcadores pertencentes ao conjunto (multi marcador). Esta possibilidade torna ainda mais eficaz a deteção em condições extremas. Outra das possibilidades é a criação de marcadores customizadoss. Estes marcadores são multimarcadores criados pelo utilizador. A biblioteca já disponibiliza um pequeno aplicativo que torna ajuda à criação dos mesmos.

5.2 FruitCatch

Jogo realizado no âmbito do consórcio por Luís Trigueiro com algumas adaptações posteriores para se enquadrar com o tema do projeto. Primeiramente elaborado para

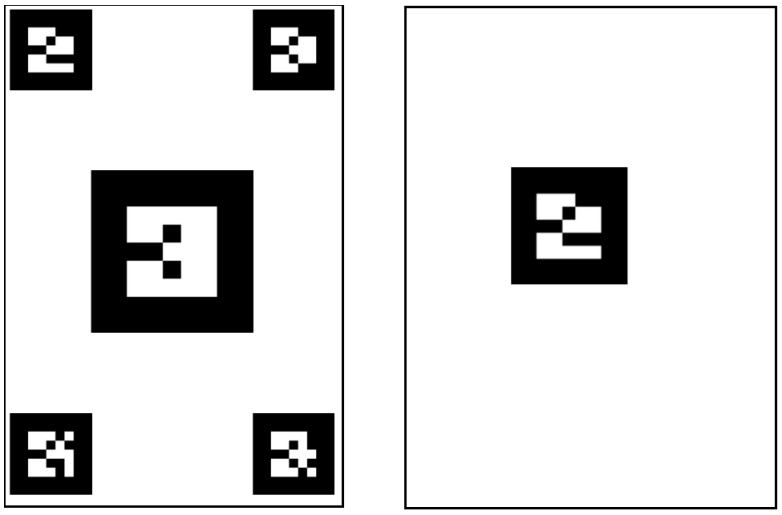


Figura 5.3: Multimarker à esquerda e marcador único à direita

utilizar ”marcadores de cor”, funcionalidade desenvolvida no decorrer do projeto, este jogo tinha como objetivo apanhar o maior número de fruta de acordo com o tipo de caixa (a caixa laranja deveria apanha laranjas e a verde, maçãs). Jogado com dois jogadores, existia um limite de tempo e penalização para quem apanhasse fruta errada. 5.4 Foi o primeiro exemplo para testar a fiabilidade da deteção por cor. Utilizando o exemplo do controlo por movimento, neste jogo em vez de se recorrer a diferentes imagens de forma a criar uma mesh para interação, esta mesh era criada pela cor detetada nas imagens provenientes da camara. Outras da funcionalidades implementas foi um sistema de high score recorrendo a estruturas de dados já definidas pelo Unity. Neste caso, a interface ainda não se encontrava adaptada para os utilizadores alvo.

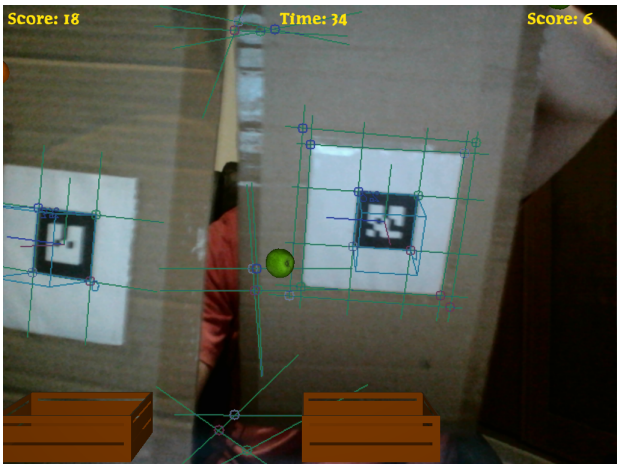


Figura 5.4: FruitCatch utilizando marcadores Alvar

Posteriormente e com a disponibilização do Beta 4 do YVision, foi testada a implementação da interação recorrendo a marcadores do género de QRCodes. Através da

biblioteca Alvar foi possível a implementação deste sistema. Mais fiável que o sistema de cores, não necessita da elaboração de uma mesh 3d pois já fornece informação suficiente para interagir com os objetos virtuais de forma simples e relativamente mais eficaz. Para além disso tem suporte a multimarcadores e combinações mais complexas. Como pontos negativos, não tem compatibilidade com alguns sistemas o que poderá ser um problema dependendo da plataforma alvo. Este jogo foi reaproveitado para a criação do produto final "Falling Fruits". Como a deteção e interação com dois jogadores não era a mais adequada, esta ideia foi retirada no produto final, resultando num jogo singleplayer mais maduro. Para além disso, o jogo sofria de fraco desempenho em alguns computadores caso se estivesse a utilizar o modelo de deteção por cor. Outras das razões para retirar este jogo da fase final de testes prendeu-se com o facto de não respeitar muitos dos "princípios" que ditam se o jogo é aceitável para o público-alvo.

5.3 AlvarPong

Como forma de testar a introdução de marcadores recorrendo à biblioteca Alvar, foi elaborado um jogo baseado no clássico Pong. Anterior à adaptação referida no Fruit-Catch, foi o primeiro exemplo de teste recorrendo a esta biblioteca. Utilizando os marcadores como base, cada utilizador recorria a um marcador impresso dos mais de 65000 marcadores predisponíveis para movimentar uma placa em altura de forma a evitar que o adversário fizesse com que a bola transpusesse a margem oposta e se possível fazer o contrario. 5.5 Sendo um exemplo básico, facilitou a aprendizagem desta tecnologia: a possibilidade de produzir marcadores próprio inclusive multimarcadores que recorrem a vários códigos de forma a tornar mais fiável o reconhecimento.

Figura 5.5: Demonstração AlvarPong

Após alguns testes realizados, entre eles uma pequena demonstração no Dia Aberto da Escola Superior de Tecnologia e Gestão de Leiria, foi elaborado um pequeno ecrã de entrada. 5.6 Criado para colmatar a falha de uma janela de iniciação, este ecrã consistia

na apresentação de dois quadrados os quais teriam de conter o marcador, de cor ou do Alvar, de forma a dar início ao jogo. Após os dois jogadores posicionarem os respetivos marcadores nas áreas adequadas, iniciava-se uma contagem decrescente ilustrada por um círculo de loading e iniciava-se o jogo. De forma a implementar este mecanismo foi elaborada uma animação de loading e desenhada uma interface recorrendo ao GUI do Unity. Nesta interface eram detetadas as colisões entre os marcadores físicos e os retângulos ou áreas virtuais. Um aspecto tido em conta para a implementação desta funcionalidade foi as diferentes coordenadas em que se encontram os vários objetos. Para além de terem de ser efetuadas conversões entre as unidades do espaço do Unity e do espaço do ecrã como as coordenadas os elementos GUI estão invertidas em relação ao ecrã, tiveram de ser efetuadas algumas alteração de forma a aprimorar a interação. Não resultou num produto final devido a não se adequar às capacidades das crianças da ”faixa etária alvo”.

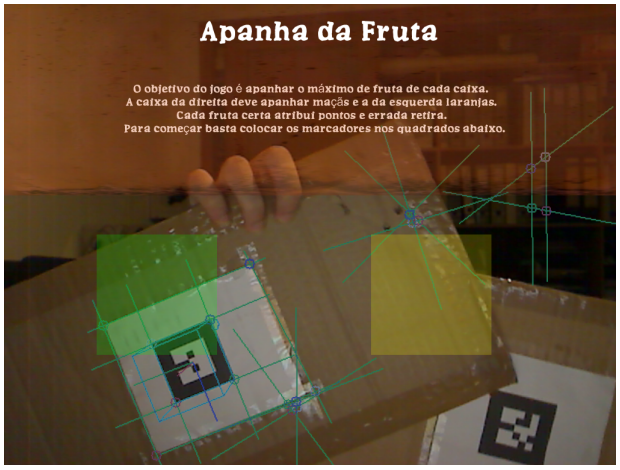


Figura 5.6: Ecrã de entrada com deteção dos marcadores

5.4 Tetris

Após as várias experiencias junto de variados grupos de utilizadores, como referido anteriormente, decidiu-se utilizar dois jogos para o teste mais intensivo: um spin-off do famoso jogo Tetris desenvolvido por Luís Trigueiro e uma versão melhorada do jogo de apanha de fruta. Em relação ao primeiro, este consiste em empilhar peças umas nas outras de forma a obter a maior pontuação, usando elementos do famoso jogo Tetris como peças e sons. 5.7 Após iniciado o jogo é visível no ecrã uma base onde serão colocadas as peças e uma peça no lado direito. O utilizador deverá movimentar o marcador até onde a peça se encontra para a “agarrar” e passar a ter controlo sobre a mesma. Uma vez adquirido o controlo da peça o jogador pode agora mover e orientar

a peça de acordo com o marcador. A peça devera ser colocada na base ou numa outra peça que já esteja colocada. Sempre que a peça em nosso controlo entra em contacto com a base ou outra peça já colocada, perde-se o controlo da peça e é criada uma nova peça que fica disponível para ser "agarrada". O objetivo é fazer a torre mais alta e com o maior número de peças possíveis, se uma peça cair fora da base, o jogo termina.

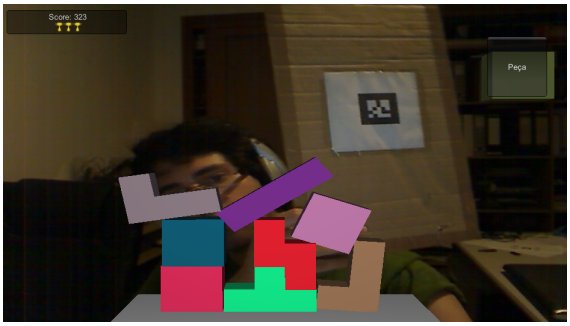


Figura 5.7: Spinoff do jogo Tetris

De acordo com os "princípios" estudados, foram integradas várias funcionalidades e definidas algumas características de modo a tornar o jogo mais interessante. Foi implementado um mecanismo de highscores de forma a recompensar o utilizador :são guardados as melhores pontuações e atribuídas recompensas consoante o desempenho. Foram integrados efeitos visuais e sonoros de forma a aproximar a interação à realidade e limitada a introduzido um sistema de dificuldade dinâmico: quanto maior a torre criada pelo jogador, maior a dificuldade de a aumentar. Todo o visual encontra-se adaptado ao público-alvo: reduzido uso de texto e quando utilizado este é adequado e legível, uso de objetos facilmente identificáveis e utilização de cores garridas. Passando agora à implementação, foram desenvolvidas duas versões deste jogo: uma recorrendo a marcadores de cor e outra a marcadores Alvar (semelhantes a QRCodes). A implementação do jogo contém três scenes: Menu, Main, e Highscores que constituem o menu da aplicação, o jogo, e a tabela de highsores respetivamente. A primeira scene, Menu, apenas apresenta um menu, conseguido através do OnGUI do Unity 3D, com três opções: Jogar, Highscores e Sair. O visual do menu foi criado recorrendo a um tema customizados com texturas e fontes próprias. De forma a facilitar a interação por parte dos utilizadores alvo, os intems do menu contém imagens explicativas e letra bem legível. A navegação é conseguida através do rato, no caso da versão com marcadores de cor, ou através dos próprios marcadores Alvar. Das opções apresentadas no menu, a primeira, Jogar, serve para iniciar o Jogo. Highscores serve para apresentar a tabela com as pontuações mais elevadas e o Sair, como o próprio nome indica, serve para terminar o jogo. No que toca aos Highscores, decidimos implementar esta funcionalidade como forma de recompensar o utilizador. Assim o jogo torna-se mais cativante e promove a competição entre os utilizadores. No que diz respeito à implementação,

estes são lidos a partir de um ficheiro XML e apresentados, também usando OnGUI, no ecrã. É também apresentada uma imagem para cada score, gravada aquando a obtenção do mesmo. Mais complexa é a implementação da scene Main. Nesta scene existem 3 GameObject mais importantes: Capture Device (ou Marker Tracker no caso da implementação recorrendo ao Alvar), GameMaster e PhysicsConfig. Capture Device é responsável pela criação de um avatar que é uma representação 3D do marcador no espaço virtual. Previamente implementado nas demos da YVision (em concreto, BoxFall) foi apenas alterado para implementar um filtro HSL para a distinção das cores e para as configurações deste filtro (hue, saturation e luminance) serem modificadas através do inspector do Unity 3D. Analogamente, no caso do Marker Tracker, utilizado também num exemplo da framework, apenas serve para detetar o marcador que irá interagir com os restantes elementos do jogo. O GameObject GameMaster, como o nome indica trata de dos elementos relativos à execução do jogo. É responsável pela geração de novas peças e pela atualização da tabela de highscores. Através do inspector podemos definir todos os parâmetros que lhe estão associados: score ganho por colocar uma peça, score ganho pela altura, sons que são tocados, tipo de peças que são usadas. Para ter um certo controlo sobre a dificuldade do jogo foi implementado o GameObject PhysicsConfig que permite que se configure no inspector a força da gravidade, e as características físicas das peças: atrito, dinâmico e estático, e elasticidade. Para as peças foi criado um prefab para cada uma. Na realidade todos os prefabs partilham os mesmos components, a única alteração consiste apenas na forma da peça. O componente com mais interesse será o que contém o script MovePiece. Este script atualiza a posição e orientação da peça de acordo com o avatar do marcador. No caso dos marcadores de cor, foi implementado um algoritmo próprio para identificar a rotação, informação que já é disponibilizada no caso dos marcadores Alvar. Para além disso, este script também responsável por detetar as colisões com outros objetos.

Ao longo do desenvolvimento surgiram algumas dificuldades. Numa primeira fase, com a necessidade de implementar o filtro HSL, houve problemas ao tentar obter um contorno diretamente através desse filtro, ficando no fim a funcionar usando o Diference (bloco já criado) após o HSL. A escolha dos parâmetros do HSL é bastante dependente das condições de captura de imagem pelo que foi necessário ter um controlo sobre as suas configurações em tempo real. Posteriormente foi desenvolvida uma aplicação para calibrar estes aspetos de forma mais intuitiva. Também se explorou a deteção de cores distintas em simultâneo (no caso da utilização de vários marcadores de cor). Os dois métodos utilizados para tal efeito foram: 2 Pipelines (um para cada cor) e um pipeline cujas propriedades eram alteradas periodicamente. Na primeira, apesar de se obter o resultado esperado com maior facilidade, incluindo ter avatares distintos para cada cor, causava uma maior carga na performance da aplicação, tornando-a lenta. Com a

segunda alternativa a dificuldade prendia-se com o sincronismo entre a cor que estava a ser detetada pelo HSL e a cor que estava a ser testada no avatar. Posto isto, e como um dos requisitos deste jogo era a deteção de rotação do avatar, foi necessário desenvolver um algoritmo que efetuasse essa mesma deteção. Uma vez que apenas temos acesso ao tamanho e posição do avatar apenas se consegue recolher o ângulo através da função Atan que nos dá um ângulo entre -90º e 90º. Na realidade, como apenas temos altura e largura do avatar, Atan apenas dará um ângulo entre 0º e 90º. Para descobrir qual o lado da inclinação em está o marcador, foi utilizado um método que calcula um centro superior e um centro inferior (ver figura). Comparando as posições relativas a ambos os centros passa a ser possível detetar a orientação correta. Finalmente, para se obter um espaço de 0º a 360º precisamos de "memória", isto é, passamos a guardar a última orientação do marcador e assim testamos nas fronteiras entre quadrantes se houve uma mudança (descontinuidade da função Atan). Na parte final do trabalho houve dificuldades em apresentar na tabela de highscore as fotografias (tiradas no fim de cada jogo) da respetiva posição na tabela. Isto porque as fotografias são gravadas e carregadas (posteriormente) fora do projeto Unity 3D. Para tal efeito, utilizou-se o módulo WWW. O entrave dava-se na leitura das mesmas, e em vez de apresentar as fotografias apenas se obtinha imagens descaracterizadas, o motivo era o facto de não se estar a dar tempo suficiente à aplicação para fazer o load das mesmas. Solucionou-se a questão criando um ciclo while que espera pelo load de caga imagem. No que diz respeito à implementação recorrendo a marcadores Alvar, existia um problema que se prendia com o desfasamento da posição do marcador em relação ao objecto virtual. Como a câmara utilizada se encontra em modo de perspetiva, existia uma certa destorção de acordo com a distancia a que o marcador se encontrava da câmara. Isso iria introduzir erros nos cálculos da posição dentro do próprio jogo tornando o seu manuseamento quase impossível. Para resolver este problema, recorremos ao cálculo da interseção da normal do marcador com um plano virtual à distancia que se encontra a câmara. Assim conseguimos obter a posição exata do objecto em relação à realidade e não à imagem da câmara. Outro fator condicionante foi o facto da movimentação da peça ser um pouco errática: num instante o marcador deixava de ser detectado e a sua posição alterava para valores fora do ecrã, o que fazia com que a peça saltitasse ou na pior das hipóteses, desaparecesse completamente. Para isso foi confinada a posição à área do ecrã e limitada a frequência de atualização da mesma. Dado que muitas das vezes, dependendo das condições do ambiente em questão, existiam ainda alguma descrepâncias entre a posição do marcador e da peça, adotou-se uma configuração multimarcador. Assim em vez de um, utilizamos cinco marcadores relacionados entre si (multimarcador), o que aumenta o grau de fiabilidade e melhora a interação com o jogo (não é necessário estarem visíveis todos os marcadores, a relação entre eles é tida em conta na deteção o que a torna menos propicia a falhas).

5.5 Falling Fruits

Falando agora do outro caso de estudo implementado, este por outro membro do consórcio, Nelson Ramos, neste caso o objetivo consiste em fazer com que, por meio de uma colher virtual que o jogador controla, um alimento caia dentro de um cesto em movimento. 5.8 Para o efeito, a posição e a inclinação da colher é calculada com base num objeto real que o jogador segure com as mãos e que será reconhecido por intermédio de uma webcam. No início do jogo, são definidos aleatoriamente quais os alimentos a apanhar e o jogador tem um determinado período de tempo para o fazer. Os alimentos caem segundo uma cadência regular. Adicionalmente, cairão objetos que não são alimentos. Se o jogador conseguir captar os alimentos definidos no início do jogo, este será premiado de acordo com a rapidez com que o fez. Por cada alimento que seja introduzido no cesto que não faça parte do conjunto a apanhar, o jogador será penalizado reduzindo o tempo restante de jogo. Os objetos (não-alimentos) que caiam no cesto provocarão sempre uma penalização, pois não se incluem no grupo de alimentos a apanhar. Para que se dê o início do jogo o objecto que controlará a colher terá de ser apresentado no ecrã e durante 5 segundos serão mostrados ao jogador quais os alimentos que este terá de apanhar, em simultâneo com a contagem decrescente de início de jogo. Durante o jogo, o jogador terá numa faixa lateral esquerda os alimentos que faltam apanhar. Do lado direito, o tempo restante é mostrado em formato numérico e, em simultâneo, com uma barra cilíndrica cuja cor se altera com as cores verde, amarelo e vermelho, correspondendo ao tempo restante do jogo corrente. O jogo será terminado caso o jogador atinja o objetivo. Este será premiado com uma, duas ou três taças de acordo com o tempo demorado para captação dos alimentos. Caso o tempo definido se esgote sem o objetivo atingido, o jogador poderá jogar novamente. Como forma de alterar alguns aspetos da jogabilidade, foi incluído no jogo um menu escondido. Premindo a tecla "M" pode-se aceder a este mesmo menu onde é possível alterar alguns aspetos como a duração do jogo, as penalizações em caso de erro e as velocidades quer do cesto quer da queda dos objetos. Da mesma forma que foram adoptados elementos que facilitam e promovam a utilização por parte de utilizadores mais jovens no caso anterior, neste esses elementos também estão presentes (grafismo adequado, sistema de recompensa e feedback visual e sonoro). No que diz respeito ao desenvolvimento deste jogo, a base assenta nos mesmos fundamentos utilizados no jogo do Tetris: filtro de cor de forma a geral uma mesh 3D para a interação com os objetos virtuais. No caso deste jogo em vez das peças de tetris, o utilizador controla uma colher. O seu movimento é feito recorrendo aos mesmos princípios apresentados anteriormente. Complementando esta implementação, na camara principal da cena, encontra-se a lógica principal do jogo. É aqui que é ditado quando começa o jogo,

a sua duração, o acesso ao menu de configuração, entre outras funcionalidades. Para gerar os objetos que caem, foi implementado um Object Generator que recorre a behavior trees do próprio YVision. Aqui os objetos são gerados aleatoriamente pela árvore, dentro de um conjunto pré-definido, e são eliminados assim que deixam de ser visíveis. Por fim, o cesto é um objecto a parte controlado pelo próprio script associado.

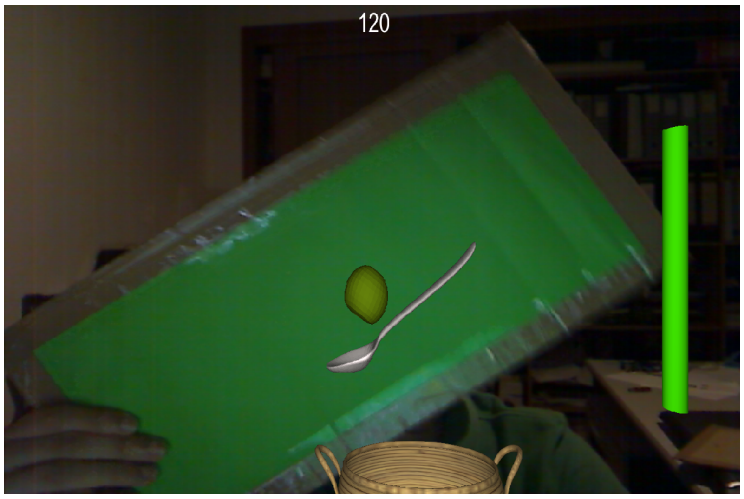


Figura 5.8: Falling Fruits

Além das dificuldades apresentadas no desenvolvimento anterior (controlo utilizando marcadores de cor, deteção da rotação e carregamento de imagens), houve alguns problemas com o controlo da colher e com as colisões com os objetos: esporadicamente a colher mexia-se de forma errática e a interação com os objetos não era a mais fidedigna. Para contornar estes problemas foi dada uma certa tolerância na deteção do movimento (para não ocorrerem pequenos saltos da colher) e atribuídos materiais diferentes aos vários objetos. Outro fator que teve um grande impacto no desenvolvimento do jogo foi a escolha da cor a utilizar: é importante que a cor definida seja uma cor primária ou secundária, excetuando os vermelhos, laranjas e castanhos para que o tom de pele não impeça o isolamento do objeto físico a usar.

Para além do desenvolvimento específico que foi levado a cabo para os jogos, foram desenvolvidas algumas funcionalidades e ferramentas globais. Foi necessário implementar a leitura e criação de ficheiros (xml e csv) de forma a guardar configurações e outras informações como relatórios e logs, de forma a compilar informação útil em situações de teste, efetuar screenshots e manipulação de imagens e incorporar um sistema do género das strings implementadas em Android (compilados desenvolvimento para multi lingua e alteração de texto dentro do jogo). Assim foi desenvolvido um conjunto de utilitários comuns aos dois jogos e compilados numa classe propria. Para além disso, surgiu de necessidade implementar uma especie de calibrador de cor de forma a facilitar a de-

finição dos marcadores nos jogos. Diferentes tipos de iluminação alteram a perceção de cor por parte da camara, o que no limite, tornava impossível a utilização do jogo recorrendo a marcadores pré-definidos. Assim foi elaborado um utilitário chamado de Color Calibration que permite, visualmente, isolar a cor pretendida, recorrendo à camara disponível, e gravar essa mesma informação num ficheiro csv que será lido mais tarde pelos jogos. 5.9 Assim tornou-se mais facil adaptar o jogo ao ambiente em que este está a ser executado.

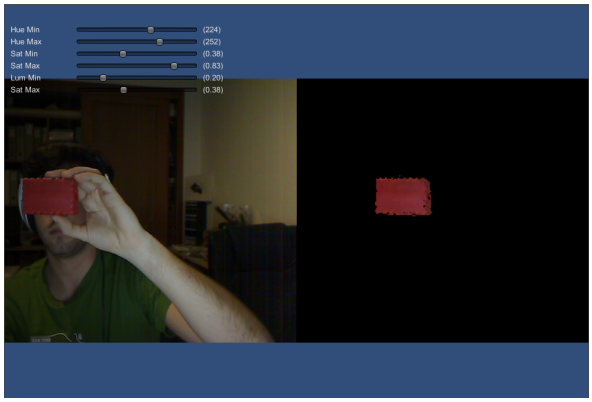


Figura 5.9: Aplicação para a calibração da cor

5.6 Conclusão

Neste capítulo apresentámos o desenvolvimento de todo o trabalho deste projeto. Explicámos o que foi elaborado em cada jogo e o trabalho anterior de modo a produzir as mecânicas de interação apresentadas. Como foi referido, encontrámos algumas dificuldades e retiramos algumas conclusões da implementação conseguida. No que toca aos modos de interação, tivemos alguma dificuldade inicialmente com a interação através dos marcadores Alvar. Eles mostravam ser mais precisos mas em algumas situações a sua posição era destorcida devido à camara utilizada. Esse problema foi contornado bem como afinado o controlo da rotação recorrendo a marcadores de cor. Outros dos aspetos a retirar deste capítulo prende-se com os modos de jogo: single player vs multiplayer. Referido no capítulo anterior, foi aprofundado neste capítulo e decidido que para a validação final é preferível utilizar jogos de um jogador só devido às interferências apresentadas pelo outro modo.

Capítulo 6

Avaliação da Solução

De forma a validar as implementações efetuadas, procedemos a uma série de teste tendo em conta vários aspetos desde a implementação, forma de interação e conteúdo. Após comprovar as possibilidades apresentadas por esta framework, é preciso verificar a sua utilidade na vertente pedagógica. Assim, neste capítulo iremos abordar os testes efetuados, o modo como foram elaborados e apresentar os respetivos resultados obtidos.

Para além das pequenas oportunidades que surgiram para validar alguns aspetos do projeto, como vários testes junto de pessoas externas elaborados no decorrer do mesmo, incluíndo testes levados a cabo durante o evento Dia Aberto, foi necessário validar os jogos junto do público alvo de forma a retirar resultados concretos. Antes de proceder a estes testes, tivemos de ter em conta vários aspetos: a preservação da privacidade das crianças, a forma como se expressam, o modo de interação no decorrer do teste e a duração adequada do mesmo. Assim sendo, não nos foi possível gravar qualquer tipo de material (imagem ou som) no decorrer dos testes, para posterior análise. Como tal, recorremos a outras fontes de dados: em cada teste preenchemos uma tabela com informações referentes ao comportamento e desempenho do utilizador e registámos dados autonomamente através de um sistema de logs implementado em cada jogo.

6.1 O Modelo PLU

Dada a natureza do projeto, houve necessidade de encontrar uma solução que auxiliasse de alguma forma o processo de teste. Visto que se tratam de crianças, isto impos alguns requisitos que não puderam ser satisfeitos com métodos clásssicos. Assim e após alguma pesquisa, decidiu-se apoiar os testes na framework Playing Learning and Using for Evaluation (PLU-E), baseada no modelo Player Learner User (PLU). Este modelo

apresentado por (Ester Baauw, Ester Baauw) e discutido mais pormenorizadamente por (R.J.W. Sluis-Thiescheffera, 2011), desenvolvido para ajudar a compreender como as crianças interagem com a tecnologia, assenta na relação das mesmas com os produtos interativos. Assim, são definidos três tipos de interações:

- A criança como jogador: neste tipo de relação, a criança vê o software como um brinquedo e como tal espera que este seja divertido
- A criança como aluno: o produto é visto como um mecanismo de ensino. É esperado que este transmita informação sobre algum assunto e que apresente algum de sistema de desafios e recompensas
- A criança como utilizador: aqui o produto é visto como uma ferramenta. A sua utilidade é avaliada consoante a sua facilidade de utilização de acordo com o objetivo definido

Escolher um jogo baseando-se neste modelo implica mapear as suas características consoante as relações pretendidas. Se quisermos que um produto seja divertido e educativo, temos de visualizar a criança do ponto de vista de jogador e aluno: o produto tem de cumprir os requisitos de um brinquedo e transmitir conhecimentos de um determinado assunto (ensinar enquanto diverte). Desta ideologia, pode ser extrapolada uma framework de testes baseada no objetivo do produto final. Sendo um software desenvolvido com base nestas relações, faz todo o sentido testá-lo de forma a compreender se cumpre os requisitos para implementar estas mesmas relações. Assim foi desenvolvida a frameworkd de testes PLU-E (Lorna McKnight, 2011). Esta framework de testes assenta nos seguintes princípios:

- Definir o objetivo e o foco do produto em termos de requisitos do projeto e PLU
- Identificar o grupo de teste e os utilizadores mais experientes dentro desse grupo
- Indicar o peso de cada componente PLU de acordo com os pontos anteriores
- Decidir em que fase do projeto devem ser efetuados os testes
- Planear as fases de avaliação

6.2 Procedimento

De forma a testar a viabilidade da framework YVision em produzir aplicações de realidade aumentada e a estudar os seus benefícios para a educação, nomeadamente em idades pré-escolares, decidimos testar dois jogos pela respetiva ordem: Falling Fruits e Tetris. Desta maneira podemos analisar as dificuldades das crianças na adoção desta tecnologia recorrendo a um jogo menos complexo e com uma jogabilidade mais simples para o início do teste. Após o primeiro contacto com este modo de interação, o segundo jogo (Tetris) vem testar a aprendizagem desta tecnologia, a capacidade motora e de raciocínio do utilizador. Assim conseguimos introduzir os utilizadores à realidade aumentada de forma gradual sem comprometer o resultado final dos testes. Os utilizadores escolhidos para a validação do produto final foram crianças com idade compreendida entre os 5 e os 6 anos de idade. Escolhemos este grupo de utilizadores devido aos argumentos apresentados no capítulo da Solução Proposta. Todas as crianças testadas já tinham utilizado o computador antes do teste mas nenhuma tinha interagido com o mesmo através de realidade aumentada. Visto as idades dos utilizadores testados terem sido praticamente as mesmas, não efetuamos distinção entre elas. Posto isto, o ambiente de teste foi o seguinte:

- Computador com Windows 8.1 64 bits
- Processador i7-3610QM
- Placa gráfica Nvidia Geforce GT 640m
- 8GB de ram
- Webcam com resolução 1920x1080
- Monitor de 37 polegadas com resolução 1366x768
- Visual Studio 2013 (ou redistributable)
- Unity 4.3.4f1

  O teste foi realizado num ambiente com boa iluminação natural. A webcam foi colocada no topo do monitor e encontrava-se ligeiramente a cima da altura das crianças. Como marcadores foram utilizados objetos com cores distintas elaborados de acordo com as necessidades. Na figura 6.1 encontram-se os marcadores utilizados. O ambiente de teste foi montado antes das crianças chegarem e foi assegurado o funcionamento de todos os equipamentos.

Figura 6.1: Marcadores utilizados durante os testes

Para além da preparação do ambiente de teste, foi necessário efetuar a identificação precisa da cor do marcador a utilizar de forma a disponibilizar uma interação fidedigna com os jogos. Foi também imperativo assegurar que o local onde se realizaram os testes se encontrasse bem iluminado e que a mesma iluminação fosse constante ao longo do mesmo, caso contrário poderia ser preciso proceder a uma nova calibração de cor no decorrer do teste, o que influenciaria o resultado final. Uma vez obtidos os valores da cor, foi necessário introduzir os mesmos no jogo. Como ambos os jogos aceitam os valores a partir de um ficheiro gerado pela aplicação de calibração, a preparação dos testes foi facil e sem grandes demoras. Abaixo está exposta a cronologia levada a cabo durante os testes. Cada iteração obedeceu a esta agenda.

- Apresentação do cenário de teste
- Introdução das várias tecnologias
- Teste do jogo da FallingFruits:

  Apresentadas as regras do jogo

  Breve demonstração das mecânicas

  Ronda de teste
- Teste do jogo Tetris:

  Apresentadas as regras do jogo

  Breve demonstração das mecânicas

  Ronda de teste (com marcador de cor)
- Finalização do Teste

Começámos então por apresentar o cenário indicando aos utilizadores o funcionamento geral do sistema e a forma como se iria proceder o teste. Explicámos de uma forma sucinta o conceito de realidade aumentada e como é efetuada a interação com a mesma. Depois iniciámos os testes propriamente ditos. As crianças foram individualmente introduzidas aos jogos e testadas da mesma forma: foram apresentadas as regras, efetuada uma ronda de habituação e de seguida a ronda de teste (para cada um dos respetivos jogos). Para o primeiro jogo foram dados 180 segundos para a concretização do teste enquanto que para o segundo não houve limite devido à curta duração imposta pela jogabilidade. As crianças situaram-se aproximadamente a 1,70m do monitor, de pé. Durante a execução dos testes, os restantes utilizadores não estiveram presentes. Como nenhuma criança tinha tido contato anterior com a RA, limitaram-se assim possíveis interferências nos resultados: cada teste teve o mesmo ponto de partida, ou seja, não existiu nenhuma criança que partisse com vantagem (conhecimento prévio do jogo a testar).

Ao longo da sessão de testes, procederam-se a recolhas de dados através do registo de eventos dos próprios jogos e da observação da experiência dos utilizadores. Esta recolha foi auxiliada por um formulário de respostas rápidas (em anexo) de forma a tornar a mesma o menos intrusiva possível e não influenciar os resultados dos testes. De seguida iremos proceder à análise e cruzamento dos vários dados.

6.3 Análise de Resultados

Começando pelo teste do primeiro jogo, o Falling Fruits, observámos um certo desconforto por parte das crianças: muitas não utilizavam o marcador da forma correta o que dificultava a deteção. Ou seja, rodavam o marcador para um ângulo que não permitia a deteção pela camara. Para além disso, efetuavam movimentos bruscos o que não era transmitido da melhor forma para o jogo. Estes aspetos aliados ao facto de este ter sido o primeiro contacto com esta tecnologia, influenciaram o resultado final deste primeiro teste. Abaixo podemos observar o gráfico de pontuações 6.2: dada a natureza do jogo, quanto maior for o tempo restante, melhor é a pontuação (regras referidas no capítulo da Solução Proposta). O resultado apresentado a 0 significa que não conseguiu concluir o jogo com sucesso. Isto não se deveu ao fato da criança em questão não ter compreendido as regras ou a jogabilidade, mas sim ao infortunio de não terem sido gerados os objetos necessários para completar a sequência.

Podemos observar que dos 180 segundos disponibilizados para a concretização do

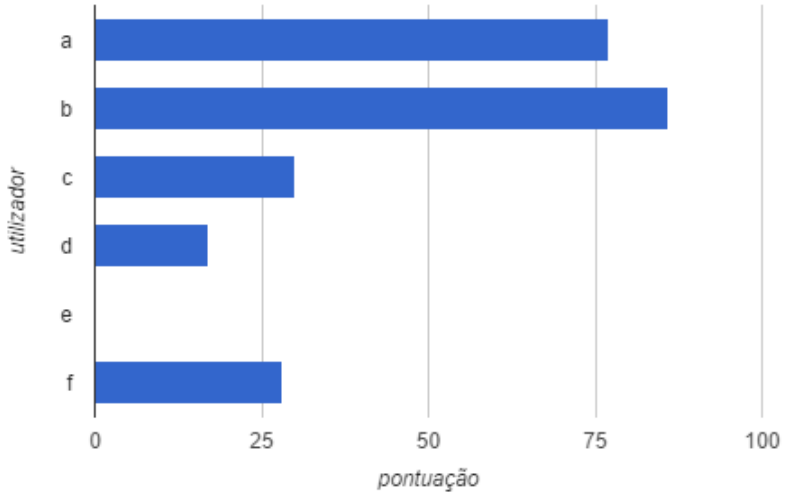


Figura 6.2: Pontuações do jogo Falling Fruits

jogo, a maior parte dos utilizadores gastou mais de metade do tempo para concretizar o teste. Devido ao fator aleatório com que são definidos os objetivos e os objetos que caem, parte desse tempo foi gasto a aguardar que os objetos corretos fossem disponibilizados. O restante, foi geralmente gasto a aprender a jogabilidade dado que este foi o primeiro contacto com o jogo. Todas as crianças perceberam a mecânica e regras do jogo. Ocorreram poucas situações em que o objecto correto tenha sido introduzido no cesto sem interação da criança: muitas das vezes o que acontecia era uma ausência de contato intencional de forma a pontuar. Observou-se também durante o teste, que as crianças percebiam quando erravam ou acertavam no objecto a introduzir na cesta, muito devido ao feedback sonoro e visual do jogo. Estas ocorrências mostraram o sentido de realização e recompensa sentidos por elas quando cumpriam os objetivos.

Passando para o segundo jogo, o Tetris, a atitude das crianças mudou ligeiramente: agora já conheciam a tecnologia e o seu modo de funcionamento. Como já se sentiam mais à vontade, os resultados obtidos forma ligeiramente diferentes. Algumas das crianças chegaram até a realizar observações e comparações entre os dois jogos testados. Disseram que preferiam o primeiro pelas cores e sons utilizados e que o segundo era de certa forma mais complicado ao nível da interação. Abaixo está apresentado o gráfico das pontuações para este jogo 6.3. Estas pontuações foram atribuídas consoante o número de peças empilhadas com sucesso e a altura atingida com a torre elaborada (regras referidas no capítulo da Solução Proposta).

Observámos que a média de pontuações rondou os 320. Mesmo as crianças que sentiram dificuldades inicialmente conseguiram obter boas pontuações neste segundo teste. Analogamente ao jogo anterior, todas as crianças compreenderam as regras e a

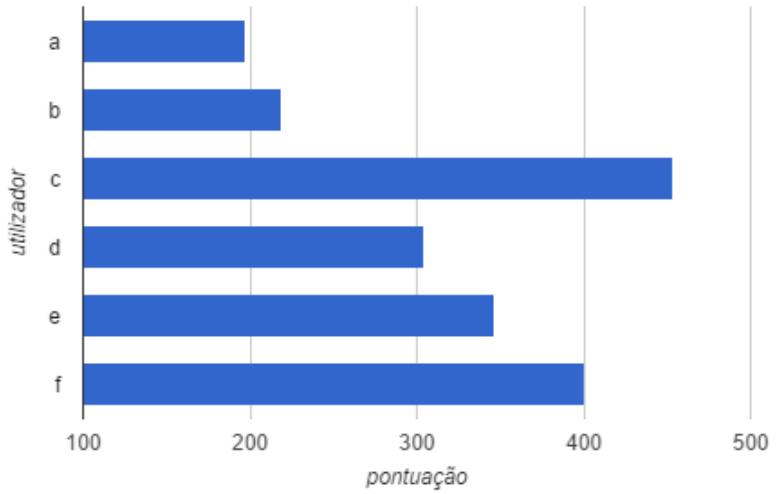


Figura 6.3: Pontuações do jogo Tetris

forma de interação. Por ser um jogo onde a destreza física é mais valorizada, foi visivel na maior parte das crianças alguma frustação. Como utilizavam o marcador de forma rapida e imprecisa, qualquer peça mal colocada influenciava drasticamente a pontuação final. Como neste jogo não existiam tantos eventos que necessitassem de algum tipo de feedback, as crianças sentiram menos empatia por este em relação ao primeiro. Outro aspecto a ter em conta foi o facto dos utilizadores não se identificarem com as peças do jogo: a utilização de modelos inspirados em jogos infantis poderia alterar esta situação.

No geral, as crianças acharam ambos os jogos divertidos e mostraram-se autónomas após a primeira interação. Os elementos utilizados mostraram ser adequados e até apelativos. Todas as crianças mostraram interesse em voltar a jogar ambos os jogos, preferencialmente o primeiro (pelas razões já referidas). Através do formulário preenchido durante o teste e baseando-nos na framework de testes PLU-E, observámos as várias relações das crianças com os jogos. No primeiro jogo, observámos a componente educativa, lúdica e utilitária. As crianças divertiram-se com o jogo, aprenderam a identificar objetos e a enquadra-los num conjunto e avaliaram a jogabilidade tendo mostrado algumas dificuldades. No caso do segundo jogo, apenas não se verificou tanto a componente educativa, muito devido às características do mesmo. No entanto, as crianças divertiram-se na mesma e melhoraram o controlo do marcador.

6.4 Conclusão

Conseguimos, neste capítulo, validar a capacidade da framework YVision, em conjunto com o Unity, de produzir jogos de realidade aumentada adequados a crianças com idades compreendidas entre os 5 e 6 anos. Verificámos também a viabilidade destas aplicações no contexto do ensino: foi demonstrado um bom método complementar aos métodos já praticados atualmente. No entanto foram identificadas algumas dificuldades por parte dos utilizadores e observados alguns aspetos a melhorar. O facto das crianças não se mostrarem à vontade com a tecnologia, deu origem a alguma frustação. Embora as regras e jogabilidade dos jogos tenham sido compreendidas, foram observados alguns comportamentos erráticos dos jogos devido à implementação e ao modo como as crianças movimentavam o marcador. No final, todos os utilizadores se divertiram com o jogos, aprenderam uma nova forma de interação e mostraram interesse a voltar a experimentar jogos semelhantes.

Capítulo 7

Conclusões

Foi proposto, neste projeto, uma nova maneira de abordar o ensino de crianças em idades pré-escolares. Apresentámos um conceito ainda novo neste contexto e analisámos os vários aspetos inerentes ao mesmo. Percorremos os vários passos desde a definição do conceito à implementação. Identificámos as tecnologias por detrás da realidade aumentada e analisámos o que já foi feito e como foi feito. Descrevemos as várias ferramentas que utilizámos e o processo de desenvolvimento, com destaque para alguns detalhes importantes da implementação. No final, obtemos quatro jogos destintos, dos quais selecionámos dois para validar a capacidade do ambiente de desenvolvimento escolhido de produzir jogos com recurso à RA e analisar o impacto no público alvo. Foram respondidos os dois problemas impostos por este projeto e identificados alguns pontos importantes no que toca à utilização deste tipo de tecnologia por crianças entre os 5 e os 6 anos de idade. Isto é, os temas abordados, a jogabilidade, o feedback produzido pelo jogo, tudo isso influencia a experiencia dos mais novos e dita se os mesmos terão vontade de voltar a jogar estes jogos.

A realidade aumentada constitui um universo vasto de possibilidades. Mesmo limitando a sua aplicação ao ensino, existem muitas vertentes por explorar. Neste projeto apenas abordámos uma pequena parte do problema.

7.1 Trabalho Futuro

Como trabalho futuro, temos como ponto de começo o melhoramento dos algoritmos de deteção de forma a reduzir as limitações da interação através de marcadores. Isto melhoraria a experiencia de utilização e possibilitaria o desenvolvimento de jogos mais complexos e abrangentes. Seria também interessante, implementar um sistema de pon-

tuações local e global. Isto gerava uma competição intra e inter escolas que motivaria as crianças a voltar a jogar e com isso melhorar os seus conhecimentos referentes aos temas dos jogos. Outros fatores a ter em conta para trabalho futuro, passam por basear a apresentação dos jogos em temas do conhecimento das crianças. Isto aliado à introdução de novos modos de jogo, como o multiplayer, elevaria o grau de divertimento obtido e a longevidade dos jogos.

Posto isto, novos jogos terão de ser desenvolvidos de forma a conjugar os temas apresentados com o plano curricular pretendido. Assim seria possível, em testes futuros, observar os benefícios desta tecnologia num cenário real. Ainda no contexto pedagógico, foi mostrado interesse por parte dos educadores em tornar a experiência mais interativa, isto é, delegar mais controlo aos professores de forma a ser possível aos mesmos definir o conteúdo e a regras dos jogos em tempo real. Assim seria possível, por exemplo, alterar a pontuação concedida pela conquista de um determinado objetivo ou a velocidade do jogo consoante a observação do desempenho das crianças.

Bibliografia

A. Driscoll, N. N. (2014). Developmental highlights in 5- and 6- year-old children. http://www.education.com/reference/article/developmental-highlights-5-6-year-old/.

Aforge.Net (2014). Aforge.net framework. http://www.aforgenet.com/.

Alvar (2014). Alvar. http://virtual.vtt.fi/virtual/proj2/multimedia/alvar/.

ARMedia (2014). Applications. http://www.armedia.it/applications.php.

Azuma, R. T. (1993). Tracking requirements for augmented reality. *CACM paper* (1), 1.

Billinghurst, M. (2002). Augmented reality in education. *New Horizons for Learning 42*, 1–5.

Chiasson, S. (2005). Design principles for children's technology. *Technical Report HCI-TR-05-02, Computer Science Department, University of Saskatchewan*.

Cynthia E Copolo, P. B. H. (1995). Using three-dimensional models to teach molecular structures in high school chemistry. *Journal of Science Education and Technology 4*(4).

Dede, C. (2009). Immersive interfaces for engagement and learning. *Science 1*(2).

Discovery (2014). Augmented reality 1m15s. http://science.discovery.com/tv-shows/pop-scis-future-of/videos/popscis-future-of-augmented-reality.htm.

Ellen Wolock, W. B. (2006). Child development 101 for the developers of interactive media: An overview of influential theories of child development. *Active Learning Associates 1*(1).

Eric Klopfer, K. S. (2007). Environmental detectives—the development of an augmented reality platform for environmental simulations. *Educational Technology Research and Development 56*(2).

Eric Klopfer, K. S. (2008). Environmental detectives—the development of an augmented reality platform for environmental simulations. *Educational Technology Research and Development 56*.

Ester Baauw, P. M.

Fisher, C. (2014). Guidelines for successful mobile interactive apps for children. http://www.gdcvault.com/play/1015634/Guidelines-for-Successful-Mobile-Interactive.

Geroimenko, V. (2012). Augmented reality technology and art: The analysis and visualization of evolving conceptual models. *16th International Conference on Information Visualisation*.

Google (2014). Glass. https://www.google.com/glass/start/.

Hannes Kaufmann, Karin Steinbugl, A. D. J. G. (2005). General training of spatial abilities by geometry education in augmented reality. *Annual Review of CyberTherapy and Telemedicine: A Decade of VR 3*.

Hsin-Kai Wu, Silvia Wen-Yu Lee, H.-Y. C.-J.-C. L. (2013). Current status, opportunities and challenges of augmented reality in education. *Computers Education 62*.

IKEA (2014). Place ikea furniture in your home with augmented reality. http://www.youtube.com/watch?feature=player_embedded&v=vDNzTasuYEw.

Immersion, T. (2014). Augmented reality projects gallery. http://www.t-immersion.com/projects.

IN2AR (2014). Augmented reality showcases. http://www.augmented-reality-games.com/showcase.asp.

Janet C. Read, M. M. B. (2011). The nature of child computer interaction. (1), 1.

Kafai, Y. B. (1990). From barbie to mortal kombat, gender and computer games. *MIT Press 1*(1).

Kurt D. Squire, M. J. (2007). Mad city mystery: developing scientific argumentation skills with a place-based augmented reality game on handheld computers. *Journal of Science Education and Technology 16*.

Lab, W. C. (2014). Arquake: Interactive outdoor augmented reality collaboration system. http://wearables.unisa.edu.au/projects/arquake/.

Lavars, N. (2014). ioptik augmented reality contact lens prototype to be unveiled at ces. http://www.gizmag.com/ioptik-ar-contact-lens-ces/30310/.

Layar (2014). Innovative ar and interactive print solutions. https://www.layar.com/.

Lorna McKnight, J. C. R. (2011). Plu-e: A proposed framework for planning and conducting evaluation studies with children. *Proceeding BCS-HCI 11 Proceedings of the 25th BCS Conference on Human-Computer Interaction*.

Lucinda Kerawalla, Rosemary Luckin, S. S. A. W. (2006). Making it real: exploring the potential of augmented reality for teaching primary school science. *Virtual Reality 10*.

Maiden, L. (2011). *Augmented Reality Browsers for Smartphones* (Second ed.). Amsterdam: Wiley Pubishing.

Metaio (2014). Featured customers and ar apps. http://www.metaio.com/featured/.

Michael Tidwell, Richard S. Johnston, D. M. and T. A. F. III (1995). The virtual retinal display – a retinal scanning imaging system. *Ph.D. Human Interface Technology Laboratory, University of Washington 1* (1).

Microsoft (2014). Kinect for windows. http://www.microsoft.com/en-us/kinectforwindows/.

Nitendo (2014). Comando wii plus. https://www.nintendo.pt/Wii/Acessorios/Acess-oacute-rios-626430.html.

OpenCV (2014). Opencv (open source computer vision). http://opencv.org/.

Page, S. P. (1980). *Mindstorms: Children, Computers, and Powerful Ideas* (Second ed.). Basic Books.

Paul Miligram, Haruo Takemura, A. U. F. K. (1994). *Augmented Reality: A class of displays on the reality-virtuality continuum* (Second ed.).

Reality, I. (2014). Exhibit. http://www.instantreality.org/exhibition/.

R.J.W. Sluis-Thiescheffera, M.M. Bekkera, J. E. A. V. H. d. R. (2011). Development and application of a framework for comparing early design methods for young children. *Interacting with Computers 23*.

Ronald T. Azuma, H. R. L. (1997). A survey of augmented reality. *Teleoperators and Virtual Environments 6* (1), 1–35.

Sony (2014). Playstation move. http://pt.playstation.com/psmove/.

Stephen Cawood, M. F. (2008). *Augmented Reality: A Pratical Guide* (Second ed.). The Pragmatic Programmers.

Takuji Narumi, Shinya Nishizaka, T. K. T. T. M. H. (2011). Augmented reality flavors: Gustatory display based on edible marker and cross-modal interaction. *Session: Olfaction, Breath & Biofeedback*.

Thomas Keenan, S. E. (2006). *An Introduction to Child Development* (Second ed.). SAGE.

UNECE (2014). Computer use by age and sex. http://w3.unece.org/pxweb/dialog/varval.asp?ma=$01_GEICT_{C}omputerUse_rpath = ../database/STAT/30 - GE/09 - Science_ICT/lang = 1ti = Computer + use + by + age + and + sex$.

Vuforia (2014). App gallery. https://www.vuforia.com/app-gallery.

WebMD (2014a). 3 to 4 year-olds: Developmental milestones. http://www.webmd.com/parenting/guide/3-to-4-year-old-milestones.

WebMD (2014b). 4 to 5 year-olds: Developmental milestones. http://www.webmd.com/parenting/guide/4-to-5-year-old-milestones.

Wei Liu, Adrian David Cheok, C. L. M.-L. Y.-L. T. (2007). Mixed reality classroom: learning from entertainment. *DIMEA 07 Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts*.

Wikitude (2014). Wikitude showcases. http://www.wikitude.com/showcases/.

YVision (2013). Yvision4unity. *Workshop Santarém*.

Anexos

Anexo 1

Tutorial de Unity

Introdução

O Unity é um motor de jogo que permite desenvolver rapidamente para várias plataformas usando uma linguagem comum. Para isso, dispõe de várias ferramentas incorporadas no seu SDK. Integrados no Unity estão um editor 3D, compilador, entre outras ferramentas.

De seguida iremos mostrar alguns aspectos do Unity incluindo a sua interface e o seu modo de funcionamento. Também iremos elaborar um pequeno jogo de demonstração a fim de aplicar alguns dos conceitos apreendidos. A informação contida neste documento teve como base a informação disponivel no site oficial do Unity (www.unity.com).

Interface do Unity

A interface do Unity é constituída maioritariamente pelos paíneis “Scene View” [1], “Game View” [2], “Hierarchy” [3], “Project” [4] e “Inspector” [5]. O “Scene View” permite ver e editar o conteúdo de uma determinada “Scene”, movendo e alterando os objetos nela presentes. O “Game View” permite antever o produto final, dando uma perspetiva em Jogo. Para além disso, também é possível testar o jogo em tempo real dentro do editor. O painel “Hierarchy” lista todos os objetos de uma determinada “Scene”, ordenados alfabeticamente e hierarquicamente. No “Project” são apresentados todos os recursos do nosso projeto. São incluídos os objetos 3d, sons, scripts, vídeos, entre outros. O “Inspector” é um painel que muda consoante o objeto ou recurso selecionado. Permite ver e editar as suas características dependendo do tipo selecionado.

Para além dos principais painéis, uma ferramenta importante apresentada também na interface é a barra de ferramentas [6]. Esta contém as ferramentas de edição de “Scene” (escala, posição, rotação…), os controlos do jogo (play, pause, frame-by-frame) e dois drop-downs que permitem escolher que layers estão visíveis e que layout está ativo.

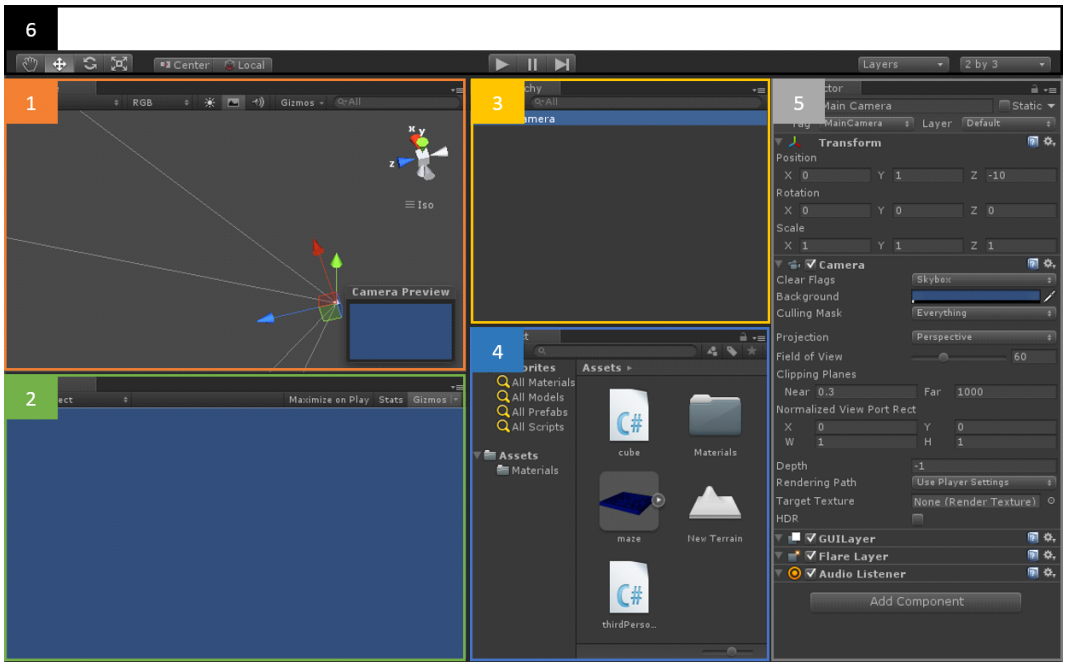


Figura : Interface do Unity4

Scene View

É onde é construído e modelado o jogo. Nele podemos ver todos os objetos que formam uma determinada cena no jogo. Através da drop down “Layers” é possível escolher os modos de visualização do “Scene View”. A interação com este painel é feita maioritariamente pelas ferramentas localizadas imediatamente a cima na barra de ferramentas. São elas a “Hand Tool”, que permite navegar na cena através de pan, freelook e orbit, “Translate Tool”, que possibilita a deslocação de objetos no espaço 3D, “Rotate Tool”, que permite rodar objetos de acordo com os vários eixos e a “Scale Tool” que facilita a alteração da escala dos objetos também nos diferentes eixos. Uma forma também bastante útil de visualizar as várias perspetivas de uma Scene é utilizando o “View Gizmo”. Localizado no canto superior direito, este permite visualizar a “Scene” de vários eixos incluindo um modo de perspetiva.

Para além das ferramentas de interação, é possível alterar o modo como os conteúdos na “Scene View” são renderizados. Através dos controlos localizados na barra principal deste painel é possível escolher o que se está a visualizar: wireframe, textures, lights, áudio, entre outros. Também na barra principal do painel, encontram-se à direita uma drop down que permite identificar os diferentes tipos de objectos visualmente através de ícones configuráveis e ainda uma barra de pesquisa constituintes de uma Scene.

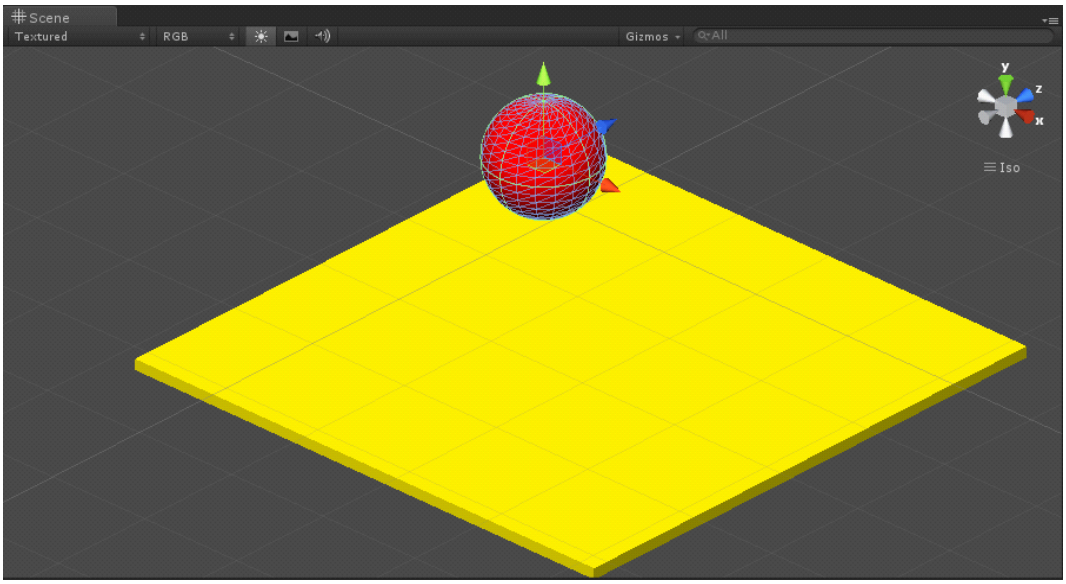


Figura 2: Scene View

Game View

É a View que mostra o aspeto final que se irá obter do jogo. Esta é invocada quando se clica no botão "Play" na barra de ferramentas. A partir daí, é possível experimentar e testar o que será o resultado final do projeto. Também através da barra de ferramentas é possível pausar e avançar frame a frame de modo a poder ter um olhar mais detalhado do resultado.

Na barra superior da "Game View" é possível alterar alguns aspectos de visualização: é possível indicar o rácio da imagem, resolução, mostrar indicadores como os fps's e ainda se queremos a "View" maximizada aquando a execução. É também possível mostrar os ícones dos tipos de objetos como na "Scene View".

Qualquer alteração feita durante execução do jogo (alteração de variáveis, valores de componentes...) é apresentada imediatamente nesta mesma janela, porém, essas alterações serão descartadas assim que terminar a execução.

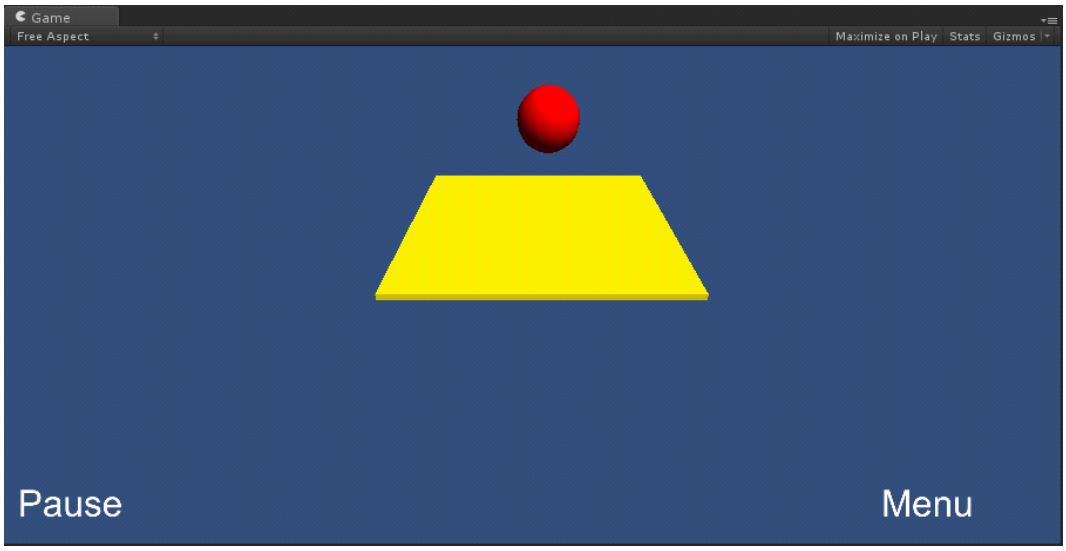


Figura : Game View

Hierarchy

Este componente da interface lista todos os objetos presentes na Scene atual. Estes são listados alfabeticamente e de acordo com o seu grau de parentesco. Ou seja, é possível agrupar vários objetos hierarquicamente em que o objeto principal é o pai e os descendentes obedecem a certas características do pai. Um dos exemplos é a origem do referencial: a origem destes objetos passa a ser a posição do objeto pai. Com isto, podemos ordenar e agrupar os nossos objetos de forma simples e lógica.

Também nesta janela está presente uma barra superior. Através desta é possível a criação de novos objetos predefinidos no Unity bem como a pesquisa de objetos na lista. Esta pesquisa pode ser efetuada através do nome ou então por características dos objetos como o tipo.

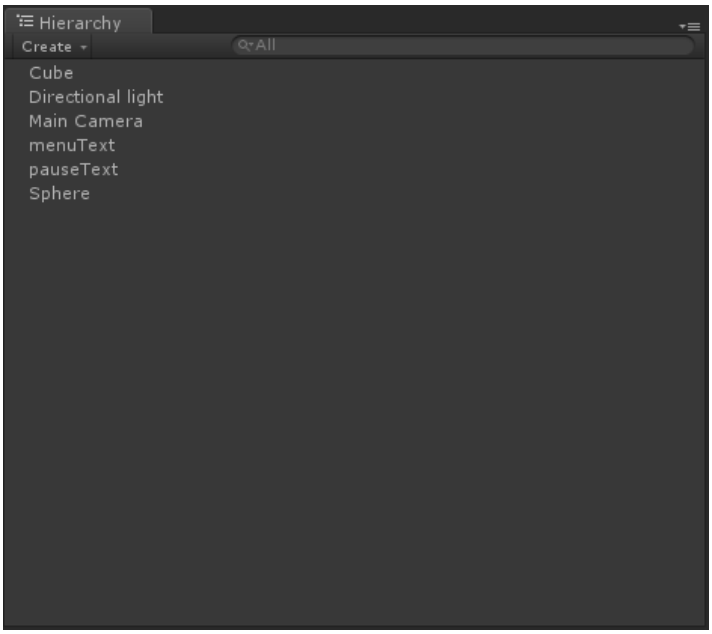


Figura : Hierarchy

Project

Neste painel são apresentados todos os recursos que constituem o projeto. É possível para além de importar recursos, criar recursos específicos do Unity (como scripts, prefabs…). Analogamente a outras View's, nesta também é possível pesquisar elementos através da barra superior. Para além disso, também é possível alterar a visualização da mesma: pode-se alternar entre uma ou duas colunas. Esta View é um espelho da pasta Assets presente na pasta do nosso projeto: qualquer alteração efetuada fora do Unity nesta pasta, é também efetuada no editor. De notar que apesar de ser possível fazer alterações fora do Unity é aconselhável só mexer dentro do editor para evitar erros na linkagem dos recursos.

Figura : Project

Inspector

Esta é uma View dinâmica que altera o conteúdo de acordo com o contexto: pode-se editar as características de um objeto, recurso e até mesmo as do editor. É nesta View que, por exemplo, se editam valores em scripts durante a preview do jogo.

Se tivermos selecionado um objeto, o que é apresentado, para além das informações do mesmo, são os diversos compondes que lhe estão atribuídos e as suas respetivas variáveis. Caso tenha sido selecionada alguma configuração do editor, aparecerão as variáveis referentes a essa configuração e assim sucessivamente.

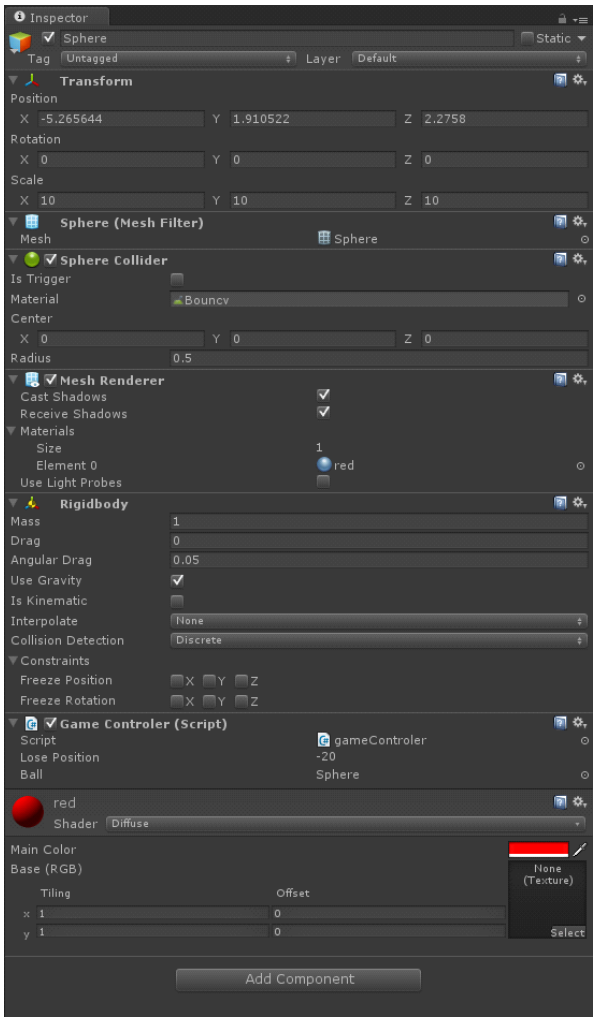


Figura : Inspector

Constituintes de um projeto

O desenvolvimento em Unity é constituído principalmente pelos seguintes elementos:

- Scenes: Constituem uma cena no jogo. Isto é, um nível, menu, ecrã de carregamento…

- Objects: Elementos de uma cena. Qualquer elemento presente numa Scene é um objecto: cameras, cubos, luzes, áudio…

- Prefabs: Conjunto de elementos que formam um grupo lógico. Ideal para criar moldes de objetos que se usaram várias vezes;

- Components: add-ons que possibilitam funcionalidades novas aos objectos;

- Scripts: contém a lógica do jogo;

Iremos abordar mais pormenorizadamente cada constituinte no exemplo prático de seguida.

Exemplo prático

Como demonstração do funcionamento desta ferramenta, iremos elaborar um pequeno jogo. O jogo será muito simples e consistirá em equilibrar uma bola num plano móvel controlado pelo utilizador. Observando estes requisitos na perspetiva do Unity, iremos então ter 2 objetos principais, são eles a bola e o plano onde a bola irá cair. Como queremos que tenha algumas características de um jogo normal, iremos também elaborar um menu simples e um ecrã de fim de jogo.

Abrimos então o Unity e criamos um projeto novo com o nome “Example1”:

File > New Project > Create

De forma a organizar o nosso projeto, no “Project View”, iremos criar 3 pastas: scenes, scripts e materials. Clicando com o botão direito do rato dentro do “Project”:

Create > Folder

Com as nossas pastas criadas, avancemos para as Scene’s. O jogo irá ter 3: são elas o menu inicial, o jogo em si e o ecrã de fim do jogo. Adicionamos então uma nova Scene:

File > New Scene

Podemos observar na Hierachy que já temos um objecto na Scene: “Main Camera”. É este objeto que irá indicar o que o utilizador final irá ver. Posto isto, nesta Scene iremos elaborar um menu simples que irá ser constituído por duas opções: “Start” e “Quit”.

Figura : Menu

Para isso iremos utilizar dois objetos do tipo “3D Text”. Criamos então um objeto desse tipo:

Game Object > Create Other > 3D Text

O objeto é então criado e aparecerá na Hierarchy. Vamos alterar o nome deste objeto para “startButton”, clicando nele e premindo “F2”. De seguida alteramos alguns aspetos de forma a permitir a interação normal de um menu (neste caso iremos usar o rato). Quando se quer alterar algumas características de um objeto, clicamos nele e desseguida, no “Inspector”, irão aparecer todas as variáveis que podemos modificar. Vamos então clicar no objeto, modificar o texto presente em “Text”, no separador “Text Mesh”, adicionar um componente através do botão “Add component” localizado no final do “Inpector” e através da barra de pesquisa da janela pop-up que aparece, localizar e selecionar o componente “Box collider”.

Como já temos as características comuns aos dois botões do menu, podemos facilmente duplicar o objeto que acabámos de alterar bastando selecioná-lo e clicar em “Ctrl+D”. Assim iremos ter dois objetos do tipo “3D Text” na nossa cena. Alteramos o nome do segundo, como foi explicado anteriormente, para “quitButton”. Para organizar o menu como foi apresentado na figura 7, alteramos as posições dos botões e da câmara como indicam as imagens abaixo.

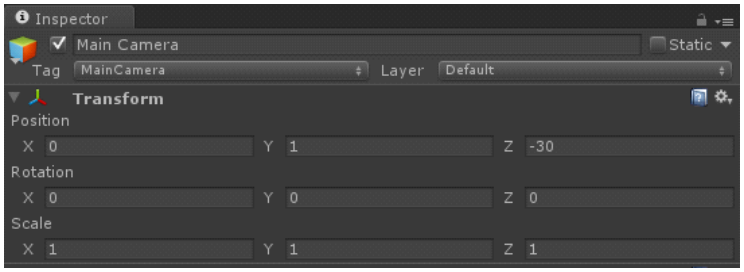


Figura : posição da câmara

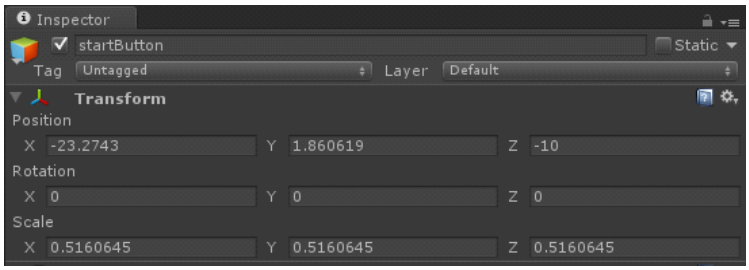


Figura : posição do botão "Start"

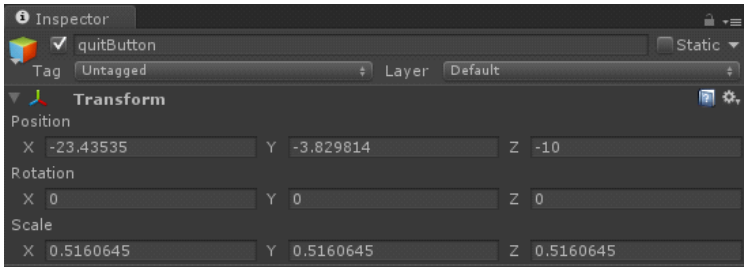


Figura : posição do botão "Quit"

Com o nosso menu desenhado, iremos então tratar da lógica por detrás do grafismo. Para isso, vamos recorrer a um Script. Em Unity podemos utilizar 3 linguagens diferentes para a criação de scripts:

-UnityScripting: javascript adaptado ao Unity

-Boo: python adaptado ao Unity

-C#: dispõe dos recursos disponibilizados pelas livrarias do mono

Neste exemplo iremos recorrer à linguagem C#.

No “Project”, dentro da pasta “scripts” que criámos anteriormente, clicamos com o botão direito do rato:

Create > C# Script

Após isto alteramos o nome para “menu” e abrimos o editor “mono”, incluído no Unity, clicando duas vezes no script com o botão esquerdo do rato. Dentro do editor, escrevemos o seguinte código:

```
using UnityEngine;
using System.Collections;

public class menu : MonoBehaviour {

 public bool isExit = false;

 void  OnMouseEnter() {
```

```
        renderer.material.color = Color.yellow;
    }


    void OnMouseExit () {

        renderer.material.color = Color.white;
    }

    void OnMouseDown () {

        if(isExit){
            Application.Quit();
        }else{
            Application.LoadLevel(1);
        }
    }
    }
```

O código acima altera a cor do texto quando o rato está por cima do botão e executa uma ação quando clica: se o botão tiver o boolean "isExit" a true, sai do jogo, se não, inicia o jogo. Guardamos o script e voltamos para o Unity. Neste momento temos o menu modelado e a lógica elaborada mas se primirmos o botão play observamos que o menu não está a funcionar. Isto acontece porque o script tem de estar associado a algum objeto da Scene para ser executado. No caso do menu iremos associar o nosso script aos dois botões. Para isso arrastamos o script que fizemos para cima dos dois botões, na "Hierarchy". Se clicarmos agora nos botões, podemos ver, no "Inspector", que ambos têm um componente novo: o nosso script "menu". Como os botões são diferentes, temos de colocar a flag "isExit" a true no botão quit, como mostra a baixo.

Figura : flag "isExit"

Terminado o menu, guardamos a nossa Scene na pasta "scenes" com o nome menu:

File > Save Scene > Save

De seguida iremos construir o jogo em si. Criamos então uma nova Scene. Nesta Scene, iremos criar um plano, uma esfera, um elemento de texto e uma luz para dar outra cor ao jogo.

Inserimos então um objeto do tipo esfera com as seguintes características:

GameObject > Create Other > Sphere

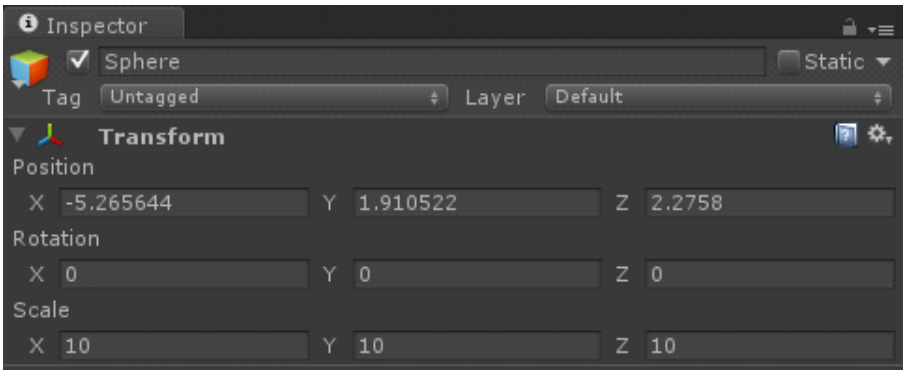


Figura : Caracteristicas da bola

Inserimos um objecto do tipo cube, que será o nosso plano, com as seguintes características:

GameObject > Create Other > Cube

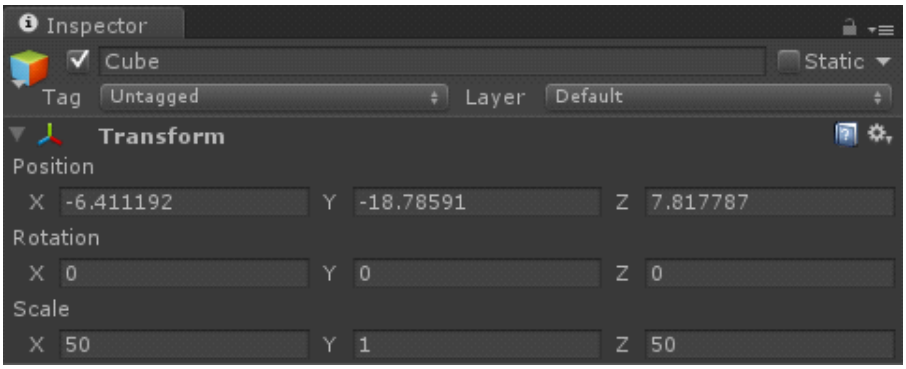


Figura : Caracteristicas do plano

Adicionamos também um elemento de texto que será usado para ir para o menu principal:

Game Object > Create Other > 3D Text

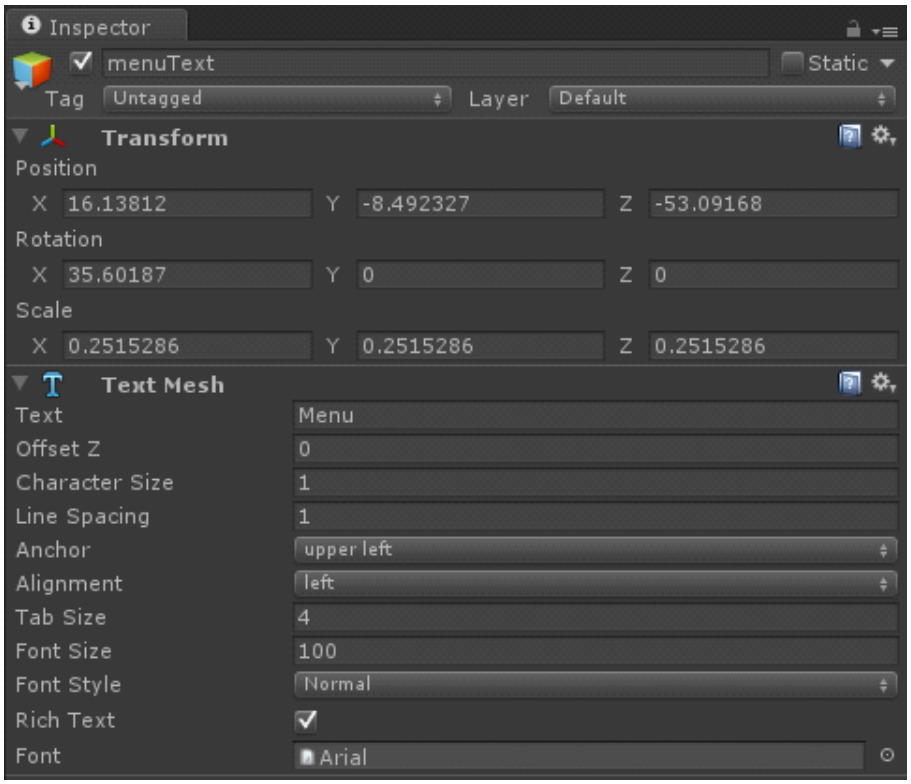


Figura : Caracteristicas do texto

Criamos também uma luz direcional e alteramos as características da camara da Scene:

Game Object > Create Other > Directional Light

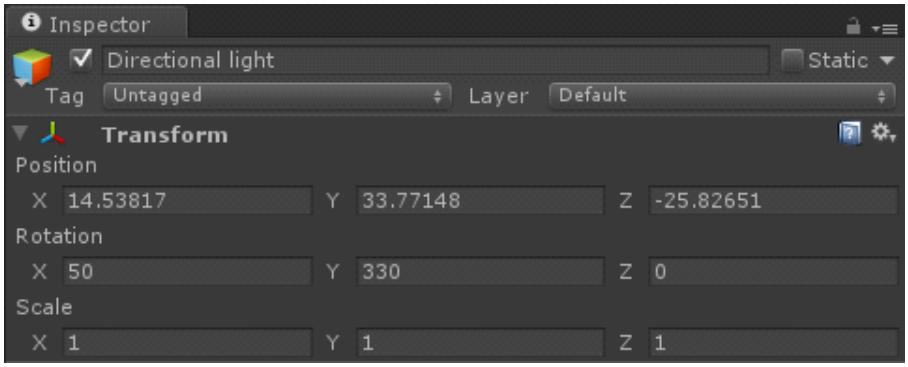


Figura : Caracteristicas da luz

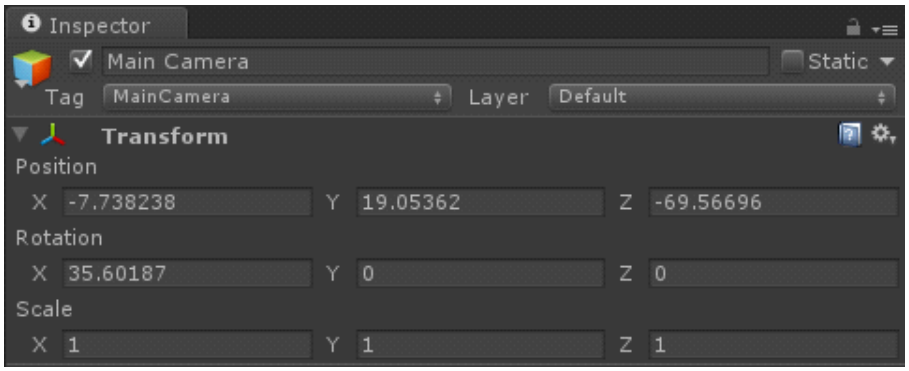


Figura : Caracteristicas da câmara

Com os objetos que precisamos já em cena, iremos agora alterar alguns aspetos de forma a dar outro ar ao jogo e a poder implementar toda a mecânica que irá ser imposta mais tarde por scripts.

Começando pela cor dos objetos do nosso jogo. De forma a termos uma bola vermelha e um plano amarelo iremos criar dos materiais para mais tarde aplicarmos aos nossos objetos.

Abrimos então a pasta “materials”, previamente criada, e com o botão direito do rato abrimos um menu e selecionamos a opção “Create” e depois “Material”. Chamemos “red” a este material e selecionamos a cor vermelha no inspector, na opção “Main Color”.

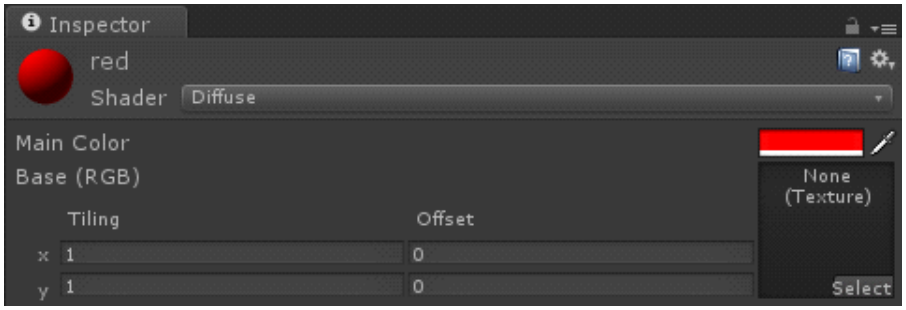


Figura : Seleção de cor

Da mesma forma, criamos um novo material neste caso com o nome “yellow” e selecionamos desta vez a cor amarela.

Agora o próximo passo é aplicar estes materiais que fizemos aos nossos objetos. Para isso basta arrastar os materiais para cima dos objetos: arrastamos então o “red” para cima da esfera e o “yellow” para cima do plano.

Para além das cores, temos de criar outro tipo de material de forma a conceber um comportamento de “bola saltitona” á nossa esfera. Para isso fazemos como criamos os materiais para as cores só que em vez de selecionar “Material” escolhemos “Physic Material”. As suas características serão:

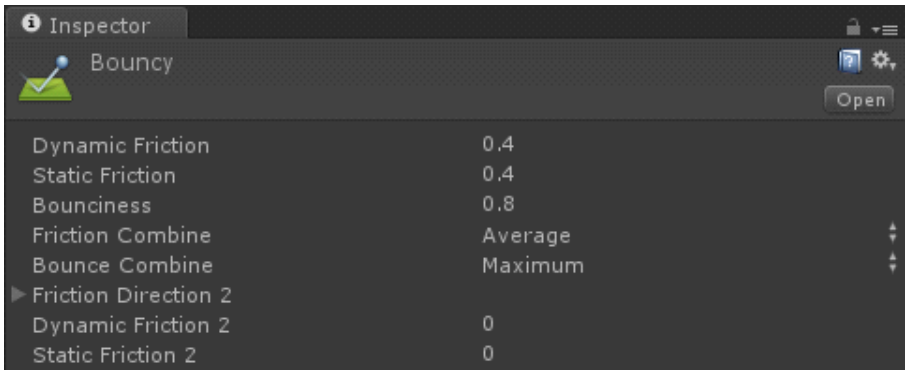


Figura : Physic Material

Arrastamos então este material para cima da nossa esfera. Com isto já temos algumas das propriedades que necessitamos para o nosso jogo. Contudo ainda faltam alguns aspetos antes de podermos programar os nossos scripts. Como os objetos que inserimos já têm coliders (componente que permite detetar colisões) basta-nos apenas inserir o componente Rigidbody à esfera. Com este componente adicionamos características ao objeto "bola" que o tornam afeto à gravidade. Então para adicionar o componente basta selecionar o objeto "Sphere" na vista de Hierarchy e clicar em "Add Component" no "Inspector". De seguida e utilizando a barra de pesquisa, adiciona-se o componente "RigidBody". Para terminar, utilizamos então o "Physic Material" que criámos, arranstando o mesmo da view "Project" para a aba "Sphere Collider", campo "Material", no "Inspector".

Figura : Adicionar o "Physic Material"

Neste momento pode-mos então codificar a lógica por detrás do jogo. Iremos então criar três scripts diferentes: "planeControler", "gameController" e "menuLevel". Começando pelo "menuLevel": é um script semelhante ao do menu principal.

```
using UnityEngine;

using System.Collections;

public class menuLevel : MonoBehaviour {

    void  OnMouseEnter() {
```

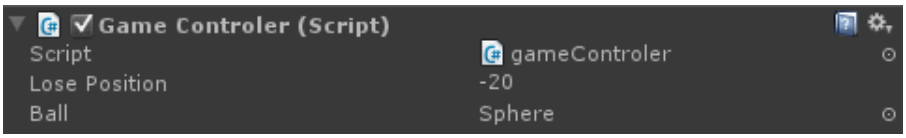
```
            renderer.material.color = Color.yellow;

        }


        void OnMouseExit () {

            renderer.material.color = Color.white;

        }

        void OnMouseDown () {

                Application.LoadLevel("menu");
        }

}
```

Da mesma forma que fizemos para o menu principal temos então de arrastar o script para o nosso 3DText object, no painel de “Hierarchy”.

Passando agora para o script “gameControler”, este é o que vai controlar o jogo em si: vai passar para o ecrã de game over quando uma certa condição se verificar.

```
using UnityEngine;
using System.Collections;

public class gameControler : MonoBehaviour {

        public int losePosition = -20;
        public GameObject ball;


        void Update () {

            if(ball.transform.position.y < losePosition){
                Application.LoadLevel("gameOver");
            }


        }
}
```

Temos então duas variáveis publicas que irão conter o objeto “bola” e a posição que este tem de atingir para o jogo terminar. Quando a posição indicada é atingida, o script mudar para a Scene “gameOver”. Adicionamos então este script ao objeto esfera e desseguida temos de arrastar o objeto esfera para a variável criada no script, de forma a este ter acesso às sua propriedades. Isto é feito selecionando o objeto em questão e arrastando o mesmo para a variável “Ball” na aba do script.



Falta agora poder controlar o plano onde a bola cairá. Para isso criamos o script “planeController” que irá tratar o input do teclado (usando as teclas w, a, s, d para o controlo).

```
using UnityEngine;
using System.Collections;

public class planeController : MonoBehaviour {

	public GameObject plane;
	public int keysDown = 0;
	public int keysPressed = 0;
	public Quaternion originalPos;

	void Start () {
		originalPos = plane.transform.localRotation;
	}


	void Update () {

		Debug.Log(keysDown);
		Debug.Log(keysPressed);
		if(Input.GetKeyDown("w") == true){
			keysDown += 1;
			plane.transform.Rotate(10,0,0);
		}else if(Input.GetKeyUp("w") == true){
			keysDown -= 1;
			keysPressed += 1;
			if(keysDown == 0){
				plane.transform.localRotation = originalPos;
			}else{
```

```
                plane.transform.Rotate(-10,0,0);
            }
        }
        if(Input.GetKeyDown("s") == true){
            keysDown += 1;
            plane.transform.Rotate(-10,0,0);
        }else if(Input.GetKeyUp("s") == true){
            keysDown -= 1;
            keysPressed += 1;
            if(keysDown == 0){
                plane.transform.localRotation = originalPos;
            }else{
                plane.transform.Rotate(10,0,0);
            }
        }

        if(Input.GetKeyDown("a") == true){
            keysDown += 1;
            plane.transform.Rotate(0,0,10);
        }else if(Input.GetKeyUp("a") == true){
            keysDown -= 1;
            keysPressed += 1;
            if(keysDown == 0){
                plane.transform.localRotation = originalPos;
            }else{
                plane.transform.Rotate(0,0,-10);
            }
        }
        if(Input.GetKeyDown("d") == true){
            keysDown += 1;
            plane.transform.Rotate(0,0,-10);
        }else if(Input.GetKeyUp("d") == true){
            keysDown -= 1;
            keysPressed += 1;
            if(keysDown == 0){
                plane.transform.localRotation = originalPos;
            }else{
                plane.transform.Rotate(0,0,10);
            }
        }


}
```

}

Da mesma forma que fizemos para a esfera, também teremos de o fazer com este script: arrastar para o objeto “plano” e desseguida arrastar o objeto para a variável do script.

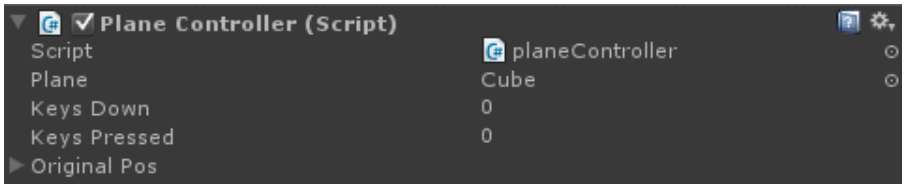


Figura : Script "planeController"

Desta Scene está tudo feita, resta guardar com o nome de “level1” e criar a última que será o ecrã de fim de jogo.

Criamos então a última Scene. Esta irá conter dois elementos de texto: um que irá indicar o fim de jogo e outro que irá avisar o jogar para clicar em qualquer tecla para continuar. Inserimos então dois objetos do tipo “3DText” com as seguintes propriedades:

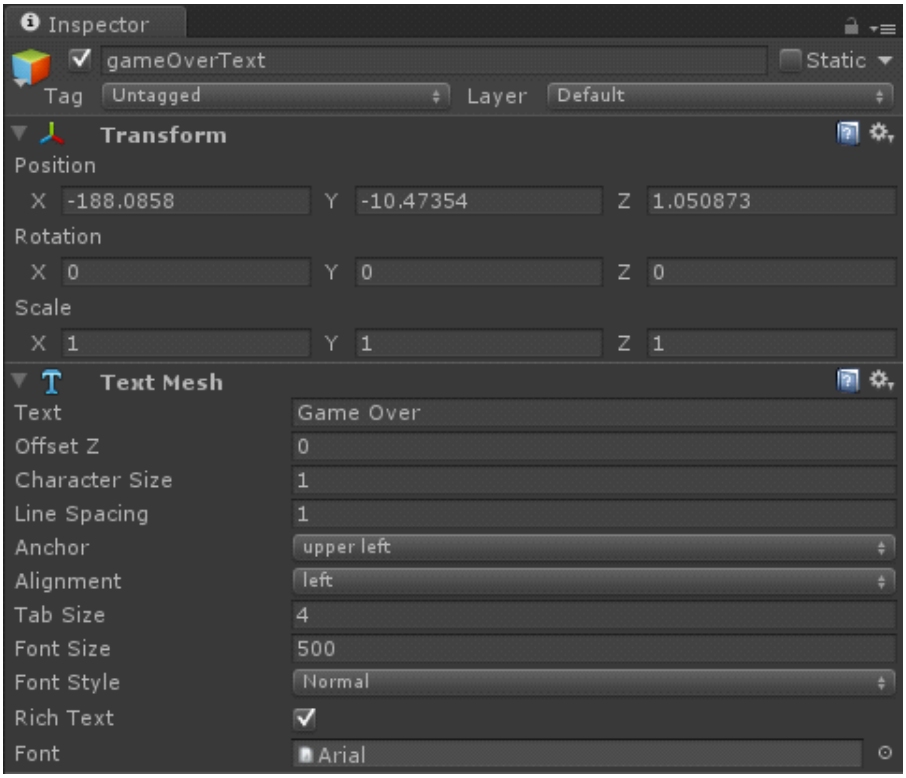


Figura : Propriedades do "gameOverText"

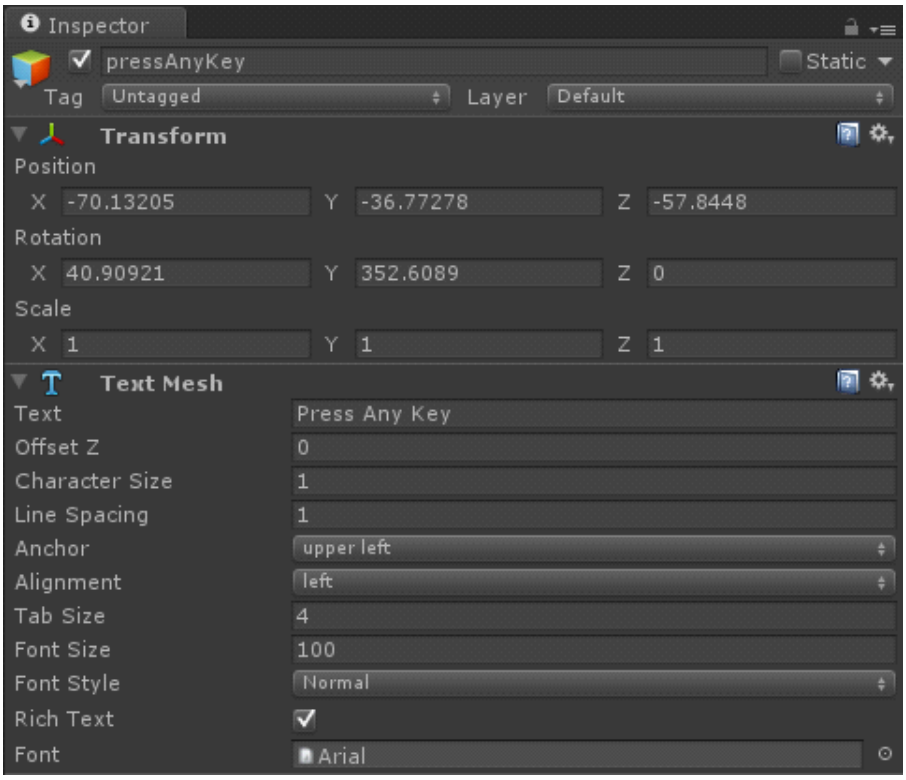


Figura : Propriedades de "pressAnyKey"

Não esquecer também do posicionamento da câmara:

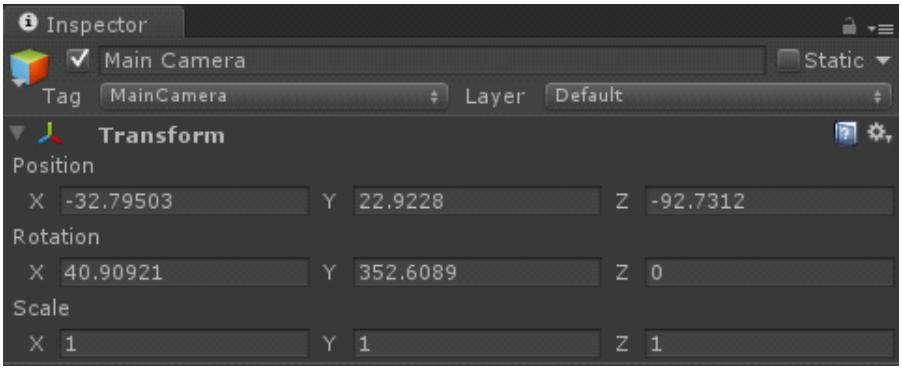


Figura : Posição da câmara da cena

Por fim, temos de tratar o input de forma a voltar para o ecrã inicial. Para isso criamos então outro script: o “gameOver”.

```
using UnityEngine;

using System.Collections;

public class gameOver : MonoBehaviour {
```

```
	public float waitTime = 1.0f;
	private float timePassed = 0.0f;

	void Update () {

		Debug.Log(timePassed);
		timePassed += 1 * Time.deltaTime;

		if(timePassed > waitTime){
			if(Input.anyKey){
				Application.LoadLevel("menu");
			}
		}
	}
}
```

Como para ser executado necessita de estar associado a um objeto na cena, arrastamos este ultimo script para o objeto camara.

Guardamos a cena com o nome de “gameOver” e passamos então ao teste do jogo. Antes de testar, teremos de adicionar as várias Scenes às “Build Settings” caso contrário o jogo não irá funcionar. Para isso vamos a:

File > Build Settings

Colocamos um visto em todas as cenas e ordenamo-las pela seguinte ordem:

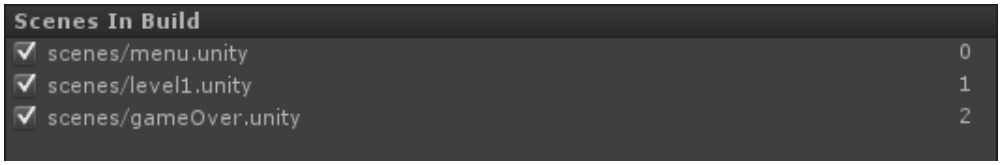


Figura : "Scenes In Build"

Para obter o produto final, basta selecionar a plataforma desejada na mesma janela e clicar em “Build”.

Com isto, aprendemos os conceitos básicos do Unity e elaborámos um jogo funcional.

Anexo 2

Formulário de Teste

Facilidade de utilização

1. A criança consegue interagir de forma correta usando o marcador

2. A criança consegue jogar de forma idependente depois do primeiro uso

3. Os controlos são adequados

4. A criança apercebe-se quando comete erros

5. A interação é simples

6 criança percebeu se o jogo acabou

. Aperciação Geral da Facilidade de Utilização

Factor Educacional

1. Apresenta uma apresentação familiar/adequada

2. Apresenta conteudo suficiente

3. Conteúdo é indentificável pela criança

4. O feedback apresenta conteúdo grafico e sonoro adequado

5. É transmitido algum tipo de conhecimento

6. Denota evolução

7. Não apresenta conteudo nocivo para a criança

8. criança percebeu o objetivo do jogo

. Apreciação Geral do Factor Educacional

Entertenimento

1. É divertido

2. O grafismo é adequado e contribui para a diversão da criança

3. É apelativo

4. A criança voltaria a jogar

5. A criança apercebe-se do feedback (dos vários sons e efeitos visuais)

6. Motiva a criança a melhorar a prestação

Anotações

O que aprendeu?

O jogo é divertido?

Gostava de o voltar a jogar?

Teve dificuldades? Com o jogo em si ou a interação?

Quais as alterações que gostava de ver?

Glossário

Ambientes Integrados de Desenvolvimento software que reúne características e ferramentas de apoio ao desenvolvimento com o objetivo de agilizar a implementação

feedback dar resposta a uma determinado pedido ou acontecimento

framework software reutilizável que providencia uma determinada funcionalidade de forma a facilitar o desenvolvimento

Handheld dispositivo portatil

head-mounted display dispositivo usado na cabeça do utilizador que disponibiliza uma interface virtual

Interfaces Naturais para o Utilizador interfaces homem-máquina intuitivas e emerssivas quase imperceptiveis

open source código fonte disponibilizado para uso e modificação

QRCode tipo de marcador técnico

rendering processo de geração de imagem apartir de um modelo 3D

smartphones telefones com capacidade computacional e conectividade superiores aos telefone comuns

tracking determinar a posição de um determinado objecto