ESCOLA
SUPERIOR
DE MÚSICA
E ARTES
DO ESPETÁCULO
POLITÉCNICO
DO PORTO

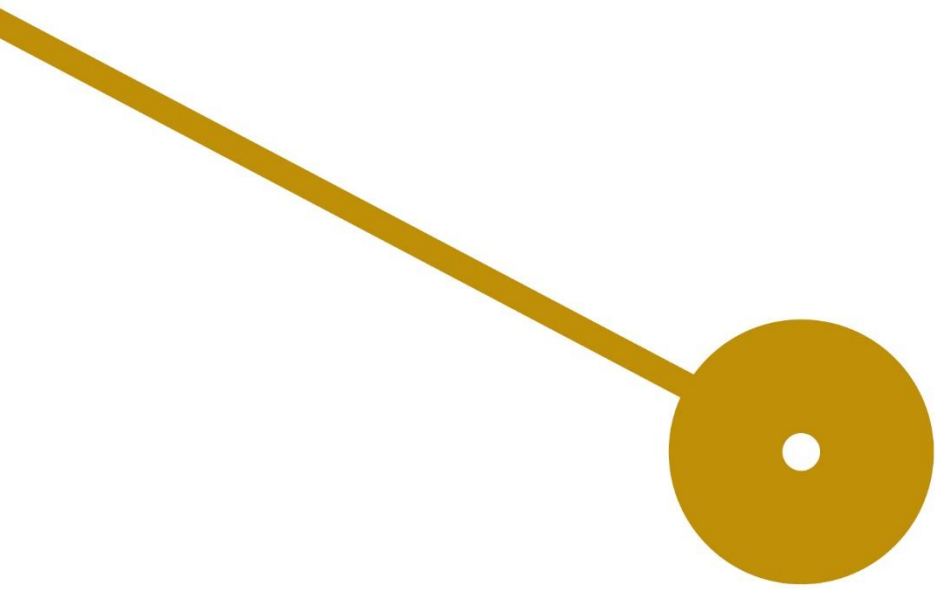
MESTRADO

ARTES E TECNOLOGIAS DO SOM

Jogo de Realidade Virtual com Foco na Experiência Sonora

José Ferreira Machado

2021



ESMAE

ESCOLA

SUPERIOR

DE MÚSICA

E ARTES

M

MESTRADO

ARTES E TECNOLOGIAS DO SOM

Jogo de Realidade Virtual com Foco na Experiência Sonora

José Ferreira Machado

Projeto apresentado à Escola Superior de Música e Artes do Espetáculo como requisito parcial para obtenção do grau de Mestre em Artes e Tecnologias do Som

Professor(es) Orientador(es)
Marco Paulo Barbosa Conceição
Pedro Emanuel Oliveira Santos

2021

À Emília Cândida.

Agradecimentos

Aos meus pais, pelo apoio e estabilidade
Aos meus orientadores, por me acompanharem neste caminho
Aos meus professores, por me ensinarem tanto
Aos meus colegas, por me ensinarem tanto mais
Á ESMAD, pelo empréstimo dos Oculus Rift

Obrigado

Resumo

Descrição do desenvolvimento de um jogo em realidade virtual, tendo como foco principal o som, a sua importância, qualidade e implementação, como tentativa de elevar o padrão atualmente em vigor para o som nos videojogos.

Fez-se o *design* da experiência, assim como listagens de *assets* a serem desenvolvidos, tendo sido tratado depois da criação e implementação dos mesmos. Finalmente, realizaram-se sessões de teste, cujos resultados foram depois considerados para o desenvolvimento de projetos futuros.

Para este desenvolvimento, utilizou-se o Unity como motor de jogo, FMod como *middleware* e Google Resonance como *plugin* de espacialização, tendo-se ainda experimentado, mas abandonado, o uso do Steam Audio.

Palavras-chave: Design de som; áudio espacial; imersividade; realidade virtual; videojogos; desenvolvimento de jogos.

Abstract

Description of the development of a virtual reality game, having as a focus the sound, its importance, quality, and implementation, as an attempt of elevating the current standard for sound in videogames.

We worked on designing the experience, as well as listing the assets to be developed, having later created and implemented them. Finally, we conducted playtest sessions, whose results were then considered for the development of future projects.

For this development, we used Unity as a game engine, FMod as the middleware and Google Resonance as the spatialization plugin, having also experienced, but abandoned, Steam Audio.

Keywords: Sound Design; spatial audio; immersivity; virtual reality; videogames; game development.

Índice

Índice de Figuras

Índice de Tabelas

Introdução

Ao longo dos últimos anos tem havido um grande investimento no desenvolvimento das tecnologias de Realidade Virtual (VR), especialmente lideradas pela indústria de videojogos. Disto resultaram grandes avanços, a nível da fidelidade de imagem, da *framerate* dos dispositivos utilizados, assim como da capacidade de posicionamento e rastreamento de movimento de utilizadores num mundo virtual. Contudo, apesar de haver investigação quanto à importância do som na localização da atenção dos indivíduos (Salselas, Penha, & Bermardes, 2020), ainda pouca é a literatura referente à implementação de técnicas de design de som como forma de melhor complementar a área de VR e a imersividade de indivíduos que a experienciem (Shindo & Kohata, 2018). Mesmo com recentes avanços a nível de implementações tecnológicas, ainda pouco parece ser o interesse da indústria no investimento da área do som como forma de elevar os projetos a ser desenvolvidos. Pela consulta efetuada, identificou-se que, nos títulos disponíveis ou em desenvolvimento para VR de momento, o som aparece em duas categorias abrangentes: em jogos cujas mecânicas base têm a ver com o som, normalmente classificados como sendo Jogos de Ritmo, como é o caso do Beat Saber (Beat Games, 2018), mas onde as mecânicas passam a ser atuadas num espaço real que se traduz para o espaço virtual, ao invés de com outro controlador, como o caso de outros jogos com mecânicas de ritmo sem serem em espaços imersivos (Brace Yourself Games, 2015); a outra categoria é relativa à imersividade dos/das jogadores/as no jogo, dando contexto e ambiência à ação passada no espaço virtual aumentando o fator de plausibilidade e daquilo que Lee descreve como o “Sensorimotor Engagement” (Lee, 2021), como é o caso do Half-Life Alyx (Valve, 2020), onde ambientes e efeitos sonoros tentam ajudar com a imersividade dos/das jogadores/as, mas de uma forma bastante análoga àquilo que são jogos eletrónicos não apresentados em contexto de VR (Valve, 1998).

Para além disso, a literatura na área, especialmente em termos de implementação ou de projetos que tenham chegado a um nível passível de ser apresentado ao grande público, menos *niche* do que o da Academia, é ainda escassa. Existe literatura sobre métodos de implementação e boas práticas a nível sonoro, e existe literatura sobre o desenvolvimento de aplicações e jogos em VR, mas a sobreposição de ambos os temas, onde haja o desenvolvimento de um jogo ou experiência interativa focado na vertente sonora, mostra-se um tópico mais ténue.

Esta é uma altura fulcral para o som na indústria dos videojogos graças ao VR, que vem da constatação que quando o som não acompanha de forma verosímil a visão (o que não implica que seja um som realista, mas sim um som adequado ao que se mostra), a experiência é insatisfatória (Lee, 2021). Com isto, surgiram várias tecnologias, como o Google Resonance, o Facebook Spatial Audio (originalmente desenvolvida pela Oculus e mais tarde adquirida pelo Facebook), a Steam Audio, entre outros, todas num espaço de tempo aproximado, para tentarem complementar a experiência com o que já era possível obter visual e mecanicamente, preenchendo assim uma lacuna no mercado.

Existem também tecnologias conhecidos como “middleware” que nos permitem fazer a integração de sons em motores de jogo como o Unreal ou o Unity, e que permitem que os objetos sonoros apareçam de maneira mais orgânica nos projetos em que são inseridos, dando um controlo mais direto a designers de som sem terem que recorrer a programação. Alguns exemplos destes *softwares* são o FMod e o Wwise, reconhecidos como fulcrais na indústria de desenvolvimento de jogos quando tomando em consideração o áudio.

Tendo tudo isto em conta, este projeto enquadra-se não só como um projeto para desenvolvimento a nível técnico, dando mais lugar e mais importância ao som dentro da experiência interativa, mas também como uma tentativa de contribuir para colmatar a algo escassa documentação desta área.

Capítulo I - Oportunidades Tecnológicas

Aquando do desenvolvimento de um videojogo, tal como com qualquer outro *software*, é necessário utilizar ferramentas específicas. Apesar de haver a possibilidade de o fazer através de "programação pura", usando apenas um *Integrated Development Environment* (IDE) e uma qualquer linguagem de programação, o mais comum é a utilização de um Motor de Jogo, ou *Game Engine* (Adams, 2013; Nystrom, 2014). Os motores de jogo permitem-nos unir todas as componentes que constituem um jogo – a geometria, a componente de programação, os vários elementos de arte e, de relevância para nós, o áudio.

Existe uma vasta panóplia de motores de jogo disponíveis (The Power of Video Game Engines: Every Game Developer's (Not-So-Secret) Weapon, n.d.). Contudo, nem todos eles estão preparados para o desenvolvimento de jogos compatíveis com VR, e desses há apenas dois que se destacam a nível de popularidade entre desenvolvedores de jogos: o "Unity" e o "Unreal Engine" (Gajsek, 2021). No desenvolvimento deste projeto, achou-se por bem utilizar o Unity, como se falará em mais detalhe no capítulo 2.2.1. .

Relativamente à implementação de sons nos videojogos existem duas possibilidades: ou se usa o nosso motor de jogo diretamente, que fornece ferramentas muito básicas e restritivas, ou se usa software externo que se interpõe entre a *Digital Audio Workstation* (DAW) e o motor de jogo, tendo assim acesso a um maior número de funcionalidades e controlo na implementação do áudio no projeto. Este *software* é normalmente referenciado como "middleware" (Gould, 2018; Shindo & Kohata, 2018; Nogueira, 2019).

Existem vários *middlewares* no mercado, que podem ser utilizados com os vários motores de jogo também disponíveis (incluindo o Unity). Contudo, dois destacam-se pelo seu uso mais proliferado e em títulos de maior importância na área dos videojogos: o FMod e o Wwise (Gould, 2018; Shindo & Kohata, 2018; Nogueira, 2019), sendo que, mais à frente nesta monografia, no capítulo 2.2.2. , iremos falar sobre a escolha feita para o desenvolvimento deste projeto.

Uma vez que os/as jogadores/as têm controlo sobre o movimento das personagens que controlam, e especialmente em jogos VR com a possibilidade de rodarem livremente a sua cabeça, têm controlo sobre o ponto de observação da experiência, ao contrário do que teríamos noutros *media* lineares como o cinema. Por esse motivo temos de dar especial atenção à forma de implementar a espacialização sonora (Salselas, Penha, & Bermardes, 2020; Kuzminski, 2016; Schütze & Irwin-Schütze, 2018). A implementação

do som deverá ter em consideração a forma como a pessoa irá experienciar o domínio aural. Seria necessário um *speaker array* com um elevado grau de complexidade de forma a cobrir o máximo espaço acústico possível, mas mesmo essa estratégia teria as suas desvantagens, visto que não seria possível rodar o espaço acústico de uma forma eficiente e verosímil, criando uma desconexão entre o que os utilizadores veriam e ouviriam (Schütze & Irwin-Schütze, 2018). Por estes motivos, jogos e outras experiências de VR são desenhadas de forma que sejam experienciadas com auscultadores, usando técnicas binaurais de reprodução de áudio (Kuzminski, 2016; Roginska & Geluso, 2018; Schütze & Irwin-Schütze, 2018).

Para atingir o objetivo de ter mais controlo sobre a espacialização de fontes sonoras num espaço tridimensional virtual, foram desenvolvidas ferramentas que se ligam ao *middleware*, *plugins* de espacialização de áudio. Os mais reconhecidos na indústria de momento, e que estão disponíveis para uso público de forma gratuita e não apenas para uso de certas companhias (os chamados *in-house softwares*), são o “Oculus Spatializer” desenvolvido inicialmente pela Oculus e mais tarde comprada pelo Facebook (Facebook, n.d.), o “Resonance Audio” desenvolvido pela Google (Google, n.d.) e o “Steam Audio” desenvolvido pela Valve (Gould, 2018). Existem ainda alguns outros, como o “DearVR” desenvolvido pela Dear Reality[1], mas uma vez que não existe um orçamento alocado a este projeto, essas opções foram colocadas de lado logo à partida. Todas estas opções têm as suas fraquezas e pontos fortes específicos, e por isso compilou-se a seguinte tabela com as informações mais relevantes a este projeto.

[1] Pode-se consultar o site no link https://www.dear-reality.com/products/dearvr-pro

Tabela 1 - Detalhes sobre opções de middleware

Plugin	Fmod	Atenuação	Oclusão e Reflexão
Oculus Spatializer (Facebook)	Sim	Raio Volumétrico ajustável à volta da fonte sonora	Não disponível
Steam Audio (Valve)	Sim	Baseada nas leis da física, sem possibilidade de ajustes criativos. Inclui o efeito de Absorção do ar, atenuando as frequências mais altas à medida que nos afastamos da fonte sonora.	Disponível, selecionando geometria e atribuindo-lhe o material acústico que deverá ser imitado. Dois métodos disponíveis de cálculo da oclusão: *raycasts* e parcial, permitindo diferentes graus de realismo. Requer preparação no motor de jogo e não pode ser alterado enquanto o jogo corre, deverá ser *baked*.
Resonance Audio (Google)	Sim	Atenuação com a distância configurável no *middleware*.	Disponível, através de uma versão simplificada de *raycasts*. Não inclui opção de alterar os materiais que a geometria individual deverá emular, mas permite a criação de “salas” (paralelepípedos) nas quais podemos mudar o material que deverá ser emulado em cada uma das seis superfícies. Pode ser alterado enquanto o jogo corre, podendo, portanto, reagir a mudanças do ambiente (como abrir/fechar portas).
	Não (Motor de Jogo)	Atenuação com a distância configurável nas variáveis disponíveis no motor de jogo	Disponível através de uma versão simplificada de *raycasts* ou atribuindo material acústico a texturas que podem ser aplicadas na geometria. Permite a criação de “salas”.

Capítulo II - Pré-Produção

2.1. Ideias e Processo de Game Design Iniciais

Começámos por várias sessões de *brainstorm*, numa tentativa de arranjar ideias concretas, que levassem a desenvolver um projeto coerente com a proposta inicial: um jogo de realidade virtual onde a experiência sonora estivesse em primeiro plano. Demorou, contudo, até chegar a ideias que se tornassem únicas, algo que só fosse possível com esta tecnologia, e não um simples reaproveitamento de elementos que já eram possíveis atingir sem recurso a tecnologias de VR ou que expandem a vertente aural e simplesmente aumentá-las (Adams, 2013; Lee, 2021).

As primeiras ideias que surgiram andavam todas à volta de memorização de padrões, como ritmos ou melodias, a serem repetidas num outro sistema, como alavancas ou botões, ou tentativas de igualar dois sons, por exemplo dando controlo sobre afinação, tempo ou reverberação. Contudo, chegava-se sempre à conclusão de que isto poderia ser igualmente atingido com algo que não som, como por exemplo substituindo o som por cores. Apesar disso, conseguiu-se nestas sessões iniciais chegar a algumas conclusões que permaneceram para o resto do projeto, nomeadamente a vontade de se explorar espaços acusticamente diferenciados. Inicialmente, teve-se uma abordagem simplificada, tendo em atenção a escala que o projeto poderia tomar e, dessa forma, chegou-se à intenção de explorar um espaço de menores dimensões e pouco reverberante, como uma sala mobilada, um espaço relativamente fechado mas bastante reverberante, como por exemplo um corredor relativamente vazio, um espaço amplo e bastante reverberante, como um armazém, e finalmente um espaço exterior. Considerou-se também inicialmente um espaço de grandes dimensões e acusticamente tratado, como um anfiteatro, mas a ideia foi descartada em sessões de *brainstorm* subsequentes, numa tentativa de reduzir a escala do projeto de forma a aumentar as probabilidades de sucesso do mesmo.

Referencias de jogos como como “Myst” (Cyan, Inc. 1993) e “The Witness” (Thekla 2016), serviriam de inspiração para o estilo de jogo pretendido, onde a exploração de ambientes virtuais com a combinação de puzzles é usada. Para tentarmos apontar nesta direção, seria então preciso melhor definir os espaços que iríamos explorar e quais seriam efetivamente os puzzles que o/a jogador/a teria que completar. Voltou-se ao *brainstorm*.

O nosso sentido da audição é usado por nós para ajudar a melhor entender o mundo que nos rodeia. Nesse sentido, o melhor tipo de puzzles seria precisamente aquele que fizesse com que a nossa audição nos guiasse no mundo virtual no qual seriamos inseridos, e que nos auxiliasse nas tarefas propostas no jogo. Pegando nessa ideia de "como é que podemos fazer com que o nosso sentido de audição nos guie", desenvolveu-se conceptualmente os puzzles "Labyrinth", "Dropper" e "Boxes".

No puzzle "Labyrinth" o/a jogador/a deveria encontrar o seu caminho num labirinto, seguindo uma fonte sonora e tentando chegar a esta. Tentando seguir a fonte numa série de corredores e perceber qual dos caminhos a tomar, apenas com base na audição; o local "corredor", referido anteriormente, seria o espaço privilegiado.

De seguida, "Dropper" tem como base a ideia que é possível distinguirmos o peso de um objeto pelo barulho do impacto deste quando cai ao chão. Nesse sentido, seria colocado em frente ao/à jogador/a dois botões, devidamente legendados, correspondentes a "pesado" e "leve", e deixar-se-ia cair um objeto onde eles não o conseguissem ver, mas conseguissem ouvir. De seguida, o/a jogador/a teria de carregar num dos botões determinando a que categoria o objeto pertencia.

Finalmente, "Boxes" tem como base inicialmente o género de puzzles *lock and key* (Adams, 2013), onde teríamos que de alguma forma ultrapassar uma porta, e finalmente estaríamos no exterior, local final deste jogo. Pensando em como poderíamos alcançar este objetivo através do som, revisitamos o jogo "1-2-Switch", mais especificamente o seu minijogo "Ball Counting" (Nintendo, 2017). Neste, os/as jogadores/as devem tentar adivinhar quantas bolas se encontram dentro de uma caixa fechada, apenas pelas vibrações do controlador que seguram. Mas, e se para além de não conseguirmos ver o que está dentro da caixa, também não a conseguíssemos sentir? Nesse caso, restar-nos-ia ainda uma opção: tentar discernir quantas bolas estão dentro da caixa pela audição. Juntando a ideia de um puzzle *lock and key* com esta ideia de objetos com uma quantidade específica de elementos no seu interior, chegou-se então à ideia de um local onde pudessem ser depositados estes objecto-chave e que abrissem a porta para o exterior.

A primeira caixa estaria no local onde começaríamos o jogo, e onde se encontraria também o mecanismo para abrir a porta de saída, de forma a deixar marcado no/na jogador/a a ideia de que algo mais seria preciso neste local. As outras duas caixas seriam a recompensa de completar os outros dois puzzles, fechando desta forma o *loop*

do jogo (Adams, 2013; Schell, 2015), ou seja, completar puzzle de forma a receber parte da chave que nos leva ao final.

Por fim, faltava ainda organizar estes vários puzzles e espaços de forma coerente. Passou-se então a um exercício de *level design* (Adams, 2013; Schell, 2015), tentando traçar um percurso a ser percorrido durante a sessão de jogo, e definindo como essas decisões afetariam o conteúdo sonoro desta experiência (Schütze & Irwin-Schütze, 2018). Após a exploração de algumas possibilidades, chegou-se ao seguinte resultado:

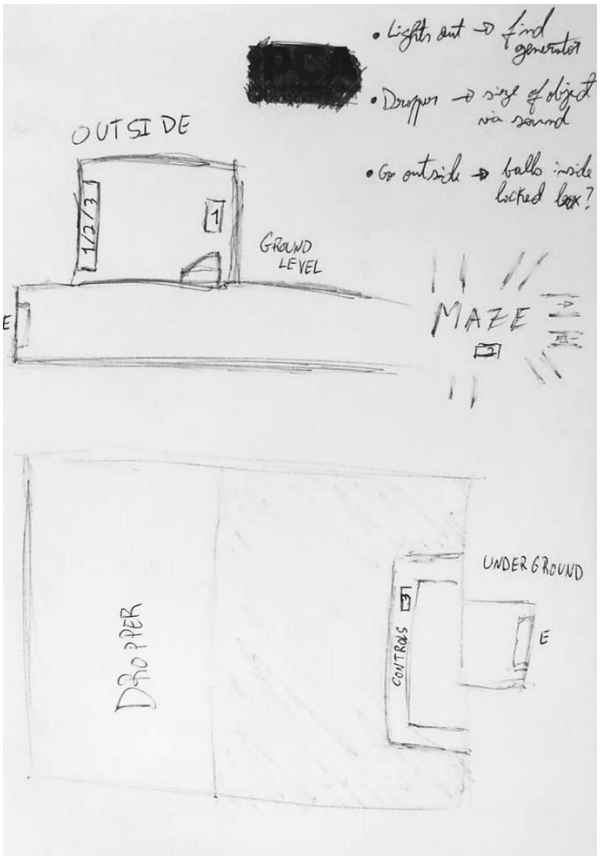


Figura 1 - Diagrama do Level Design

O jogo teria início numa sala, no nível térreo de um edifício. Lá, encontrar-se-ia o mecanismo que abre a porta para o exterior e uma outra porta que leva para o corredor adjacente. Para além disso, encontraríamos aqui também a primeira caixa com bolas que deverá depois ser inserida no mecanismo para abrir a porta para o exterior. Esta sala estaria mobilada e seria o espaço relativamente pequeno e menos reverberante, discutido anteriormente.

Passando para o corredor adjacente, perto da porta da sala, estaria um elevador, e o corredor continuaria para o outro lado. Quer o/a jogador/a tente interagir com o elevador ou se afaste no outro sentido, as luzes do edifício iriam abaixo, e ouvir-se-ia o gerador

a fazer barulho na continuação do corredor. Aqui, o/a jogador/a teria que passar por um labirinto de corredores, sempre atento ao som do gerador, tentando discernir em que direção ir. Este corredor e sequente labirinto são o local fechado, mas bastante reverberante previamente discutido. Chegando ao gerador, haverá uma maneira de o voltar a pôr a trabalhar, e uma segunda caixa com bolas no interior. Voltando para o elevador, seguindo a música proveniente deste, encontramos 2 botões no seu interior: um que indica o piso térreo, onde nos encontramos, e um que indica um piso inferior.

Indo para o piso inferior, saímos num complexo semelhante a uma junção entre um laboratório e uma casa das máquinas, num varandim que olha uma área iluminada por candeeiros elétricos. Nessa área, conseguiremos ver as caixas que cairão quando carregarmos num dos botões do painel que se encontra à nossa frente no varandim. Em primeiro lugar, cairá uma caixa leve, e, portanto, teremos que carregar no botão de igual significância. De seguida, cairá uma caixa pesada, que causará um curto-circuito nas luzes, fazendo com que não consigamos ver as próximas caixas e, portanto, fazendo-nos guiar apenas pelo som. Após algumas caixas, cumprindo uma quota arbitrária, desbloquearemos um contentor nesse varandim, onde se encontra a caixa final que precisamos para desbloquear a saída para o exterior.

Trazendo todas as caixas de volta para a sala, e verificando qual das caixas deverá ser inserido em qual parte do mecanismo, consoante o número de objetos no seu interior, abrir-se-á uma porta que nos levará ao exterior. Este será o último local a ser visitado.

Definida a metodologia para o jogo seria levado a cabo a implementação prática dos mesmos, que será documentada nos capítulos seguintes.

2.2. Processo de Pesquisa

2.2.1. Desenvolvimento de Jogos para VR

Como referenciado anteriormente no capítulo 1 desta monografia, existem dois motores de jogo que se destacam quando se fala de desenvolvimento de jogos VR: o Unity e o Unreal Engine. Ambos são perfeitamente capazes de produzir conteúdo de alta qualidade. Contudo, uma vez que o calendário para a realização deste projeto é bastante apertado, valorizou-se a rapidez com que se consegue trabalhar com a ferramenta, atendendo que ambas consiguem produzir resultados idênticos e, neste caso, o facto de o autor ter já experiência a nível profissional com o Unity, e o facto de

este ser um dos *softwares* falados durante este mestrado, fez com que o Unity fosse a escolha natural.

A 23 de Março de 2021, no período de transição entre o processo de pré-produção e de produção, o Unity lançou a sua versão *Late Term Suport* (LTS) do Unity 2020, fazendo assim com que esta fosse a versão recomendada de desenvolvimento mais atualizada (Hauwert, 2021). Uma vez que esta versão é uma melhoria direta da versão Unity 2019 LTS, que se tinha ponderado usar, e que aparenta ser totalmente compatível na integração de VR planeada, escolheu-se seguir com esta nova versão para o desenvolvimento do projeto.

Com o nosso motor de jogo escolhido, restou ainda pensar como fazer a integração da componente VR neste projeto. Verificando a documentação oficial do Unity (Unity, 2021), verifica-se que existem várias possibilidades de desenvolver aplicações de VR com o Unity, sendo preferível recorrer a bibliotecas externas, sendo elas suportadas diretamente pelo Unity ou *third-party*. Para desenvolvimento de aplicações e jogos em VR, é necessário acesso a tecnologias de reprodução dos mesmos. Existem várias opções de equipamentos capazes de correr este tipo de *software*, tais como os HTC Vive, os Sony Playstation VR, os Valve Index, entre outros (Greenwald, 2021). Uma vez que este projeto não tem financiamento externo, contactou-se outras instituições pedindo ajuda na forma de acesso aos seus equipamentos – nomeadamente à ESMAD, que graciosamente concedeu acesso a um dos seus Oculus Rift. Tendo essa informação em conta, e olhando de novo para a documentação oficial do Unity, podemos verificar que quando estamos a desenvolver para os Oculus Rift, deveremos usar preferencialmente o Oculus XR Plug-in, de forma a conseguir fazer corretamente a integração com o ambiente de desenvolvimento da Oculus.

2.2.2. Implementação e Espacialização de Som em Jogos

Como referenciado anteriormente no Capítulo I -desta monografia, relativamente ao conteúdo áudio, são dois os *middlewares* de referência: o FMod e o Wwise. Apesar de haver prós e contras em usar cada um destes, é importante ter uma coisa em consideração no decorrer deste projeto: o calendário é bastante apertado para haver liberdade de aprender novos softwares. Nesse sentido, e uma vez que as diferenças encontradas na fase de pesquisa não foram substanciais (revolvendo primariamente sobre preço para projetos comerciais e acesso a *plugins* extra), decidiu-se avançar com o Fmod, uma vez que o autor tem experiência com este. O recomendado na

documentação (FMOD, n.d.) é utilizar-se a última versão verificada pelo Unity, que à data da escrita desta monografia é a 2.01.05, pelo que foi esta a versão adotada.

Como referenciado anteriormente no capítulo 1 desta monografia, existem vários plug-ins de espacialização, cada um com diferentes focos e, portanto, diferentes pontos fortes e fracos relativos a este projeto.

Com acesso à informação compilada na Tabela 1 (pág. 5), podemos descartar o “Oculus Spatializer”, devido à sua falta de controlo de oclusão, o que faria com que o puzzle referente ao labirinto de corredores se tornaria imensamente complexo de ser implementado.

Uma vez que há um interesse no realismo que, apesar de mais pesado a nível de processamento é um ponto de interesse a explorar neste projeto, o facto de termos controlo sobre o material físico que a geometria irá replicar, em vez de termos acesso apenas a salas paralelipipédicas, foi um ponto positivo suficiente para priveligiar inicialmente o “Steam Audio” em detrimento do “Resonance Audio”. Apesar de não ser perfeito, especialmente faltando a possibilidade de mudanças de oclusão em tempo real, as falhas que apresenta poderão ser ultrapassadas tendo cuidado aquando da finalização do *level design*, tomando em atenção especialmente as fontes sonoras perto de qualquer geometria amovível e reduzindo ao máximo alterações na geometria enquanto o jogo está a decorrer, que possam levar a uma quebra de imersão por parte do/da jogador/a.

Assim sendo, fechamos esta secção da pesquisa com o plano de se usar os *softwares* FMod e Steam Audio neste projeto. No momento da escrita desta monografia, a versão disponível do Steam Audio é a versão 2.0-beta.20, o que revela ser uma versão ainda bastante instável e volátil.

Devido a problemas técnicos que serão expostos no capítulo 3.1 e após algumas tentativas de utilização da versão mencionada do Steam Audio, foi necessário abandonar este *middleware* e em alternativa usar o Resonance Audio. A versão do SDK para o Unity mais atualizada, à data da escrita desta monografia, é a versão v1.2.1, e será essa a utilizada neste projeto.

2.3. Listagem dos Elementos a Serem Produzidos

Tendo em conta o processo de *game design* e *level design* discutidos no ponto 2.1. desta monografia, dadas as mecânicas e ações que se irão poder tomar nesta

experiência e o tipo de materiais que se pretende recriar virtualmente, foi desenvolvida a seguinte listagem de elementos sonoros a serem produzidos, tendo havido já uma preocupação sobre nomenclaturas que se viriam a usar aquando do processo de implementação no *middleware* e motor de jogo.

Tabela 2 - Listagem de Sons a Ser Desenvolvidos

Som	Dur.	*Loop*	Integração	Notas
Passo	< 1s	Não	One-Shot/Trigger "carpete", "madeira", "metal", "erva"	Deverão ser desenvolvidas várias versões para cada um dos materiais onde o/a jogador/a poderá andar, nomeadamente carpete, madeira, metal e erva.
Bola a Rolar	2s	Sim	"velocidade" (0-1)	Bola de metal de pequenas dimensões (~2cm de diâmetro) a rolar numa superfície de madeira
Bola a Bater	< 1s	Não	One-Shot/Trigger	Bola de metal de pequenas dimensões (~2cm de diâmetro) a bater numa superfície de madeira
Caixa de Madeira a Colidir	< 1s	Não	One-Shot/Trigger "carpete", "madeira", "metal", "cimento"	Som de caixa de madeira a bater em superfícies
Caixa Pesada a Cair	< 1s	Não	One-Shot/Trigger	Várias variações Caixa cairá em cimento
Caixa Leve a Cair	< 1s	Não	One-Shot/Trigger	Várias variações Caixa cairá em cimento
Correto	< 1s	Não	One-Shot/Trigger	Este som tocará quando o/a jogador/a acertar corretamente no peso da caixa que caiu na arena
Errado	< 1s	Não	One-Shot/Trigger	Este som tocará quando o/a jogador/a errar no peso da caixa que caiu na arena
Clicar em botão	< 1s	Não	One-Shot/Trigger	Este som tocará sempre que o/a jogador/a interaja com um botão, seja no elevador, no puzzle de deixar cair as caixas, ou outro
Música Elevador	1 min	Sim	Bool "ativo"	
Alarme Falta de Energia	5s	Sim	Loop	Indicará onde está o gerador que precisa de ser reiniciado.
Gerador a Trabalhar	30s	Sim	One-Shot/Trigger	
Luzes a tremer	1s	Não	One-Shot/Trigger	Luzes intermitentes, falhando de vez em quando
Tom de Sala	1 min	Sim	Loop	
Falha de Potência	3s	Não	One-Shot/Trigger	Indicará que a eletricidade foi abaixo
Elevador a Subir/Descer	10s	Não	One-Shot/Trigger	
Portas Elevador Abrir/Fechar	2s	Não	One-Shot/Trigger	

2.4. MVPs e como os melhorar

Um dos pontos fortes do design deste projeto da maneira acima descrita é que ele é essencialmente modular. Apesar de ser preferível que toda a experiência tenha o mesmo nível de polimento, seja estético ou mecânico, de forma a tornar a experiência o mais coerente possível (Adams, 2013; Schell, 2015; Irish, 2005), o facto dos três puzzles que vão ser desenvolvidos para este projeto não terem uma ligação direta faz com que possam ser trabalhados em separado.

Desta forma, se se conseguir atingir o *minimum viable product* (MVP) (Wirtz, 2021) em qualquer um dos puzzles, o estado em que eles estão funcionais e prontos a serem implementados no produto geral, e se conseguirmos isso ainda com tempo de continuar a iterar no seu desenvolvimento, teremos então oportunidade de expandir algumas facetas de cada um desses puzzles, de forma a melhorar o projeto e possivelmente aumentando o seu fator de *replayability* – o que determina se há interesse em ser jogado por um/a qualquer jogador/a. Resumidamente, atingir o MVP faz com que o puzzle esteja completo do ponto de vista de desenvolvimento, mas não ótimo, podendo, portanto, ainda ser melhorado.

Peguemos em cada um dos puzzles individualmente, e vejamos quais os seus MVPs segundo o que acima descrevemos, e quais as várias camadas de melhoramentos que poderemos atingir, caso tenhamos tempo.

2.4.1. Sobre o *Labyrinth*

No puzzle do *Labyrinth*, como descrito previamente, o/a jogador/a terá de navegar um labirinto de corredores, seguindo um som de forma a atingir o gerador de energia e reiniciá-lo, de forma a restaurar a energia ao complexo onde se encontra.

O MVP deste puzzle seria, então, a construção de um labirinto e das características sonoras desse espaço e de uma fonte sonora que esteticamente se assemelhe a um gerador, e que o/a jogador/a consiga navegar do ponto inicial até à fonte sonora de forma viável apenas através da sua perceção aural e que, interagindo com o gerador, se voltem a ligar as luzes do espaço onde se encontra. Chegando a este ponto, ter-se-ia atingido o necessário para dar este puzzle como completo.

Contudo, se tivermos mais tempo de desenvolvimento no nosso calendário, poderíamos fazer algumas alterações que elevem este puzzle.

Em primeiro lugar, poderíamos pensar neste labirinto como sendo composto de várias características sonoras, e não como sendo sempre aproximadamente a mesma, ou seja, em vez de por exemplo todos os corredores que compõem este espaço terem chãos de cimento, paredes de azulejos e tetos de tijolo, poderá haver corredores carpetados, ou onde as paredes sejam feitas de madeira, ou onde o teto tenha uma proteção de cortiça, entre várias outras hipóteses. De forma a conseguirmos atingir este patamar, teríamos de fazer mudanças visuais no espaço virtual de forma a representar coerentemente a característica sonora com o seu material, e também mudanças de implementação, atribuindo corretamente a característica do material no nosso *middleware* e *plugin* de espacialização.

Para além disso, poderíamos também ir para um outro patamar que envolveria mudanças mais técnicas a nível da implementação, que seria não ter um labirinto estático, mas sim composto de vários módulos que pudessem ser interligados, e que fosse depois gerado no início do jogo, fazendo com que de cada vez que se iniciasse uma nova sessão de jogo tivéssemos acesso a um labirinto diferente.

2.4.2. Sobre o *Dropper*

No puzzle *Dropper*, o/a jogador/a terá de identificar se o objeto que caiu num determinado local pertence à categoria de objetos leves ou objetos pesados.

Neste sentido, o MVP será ter uma plataforma onde o/a jogador/a poderá acionar a queda de um objeto, ter esse mesmo objeto a embater numa superfície e produzir um som correspondente, e permitir que o/a jogador/a escolha o *input* de “leve” ou “pesado”. Inicialmente o/a jogador/a poderá ver as caixas, de forma a perceber como funciona o puzzle, mas a dado momento a sua visão para o local onde as caixas caem seria obstruído pelo corte da luz que ilumina o local de queda. Tendo isso em conta, na nossa listagem inicial de sons tínhamos descrito os sons a serem produzidos como sendo:

Tabela 3 - Listagem de Sons do MVP do puzzle Dropper

Som	Dur.	*Loop*	Integração	Notas
Caixa Pesada a Cair	< 1s	Não	One-Shot/Trigger	Várias variações Caixa cairá em cimento
Caixa Leve a Cair	< 1s	Não	One-Shot/Trigger	Várias variações Caixa cairá em cimento
Luzes a tremer	1s	Não	One-Shot/Trigger	Luzes intermitentes, falhando de vez em quando

Contudo, temos ainda a possibilidade de, após completo este estado, elevar o puzzle, fazendo com que os objetos que caem sejam não só “caixas indiscriminadas”, mas sim objetos específicos. Ou seja, em vez de apenas termos caixas leves a cair, teríamos também bolas de futebol, quadros, livros, carrinhos de brincar, etc. e em vez de apenas termos caixas pesadas a cair teríamos também máquinas complexas, móveis, carros, etc. Um carrinho telecomandado soa bastante diferente de um carro, quando deixados cair da mesma altura.

O desafio de aumentar a complexidade deste puzzle seria apenas de um ponto de vista de design sonoro: quanto maior a nossa base de dados de sons, maior a mistura que poderíamos fazer quando apresentamos este puzzle ao publico.

Assim sendo, poderíamos estar perante uma listagem de sons semelhante à seguinte:

Tabela 4 - Possível Listagem de Sons do puzzle Dropper

Som	Dur.	*Loop*	Integração	Notas
Caixa Pesada a Cair	< 1s	Não	One-Shot/Trigger	Cairá em cimento
Caixa Leve a Cair	< 1s	Não	One-Shot/Trigger	Cairá em cimento
Luzes a tremer	1s	Não	One-Shot/Trigger	Luzes intermitentes, falhando de vez em quando
Bola de futebol	< 1s	Não	One-Shot/Trigger	Cairá em cimento
Bola de destruição	< 1s	Não	One-Shot/Trigger	Semelhante às que se encontram em escavadoras para destruir edifícios. Cairá em cimento
Carrinho telecomandado	< 1s	Não	One-Shot/Trigger	Cairá em cimento
Carro	< 1s	Não	One-Shot/Trigger	Cairá em cimento

Deve-se, contudo, ressalvar que apenas fará sentido desenvolver mais a fundo esta ideia, e subsequentemente esta tabela, caso consigamos de facto atingir o MVP com tempo de sobra na calendarização de desenvolvimento.

2.4.3. Sobre as *Boxes*

No puzzle *Boxes*, o/a jogador/a terá de identificar o número de objetos dentro de uma caixa, e utilizar essa informação para abrir uma porta.

Assim, o MVP deste puzzle será dividido em dois: 1) a criação de três caixas iguais, cada uma contendo um número diferente de objetos; 2) uma forma de o/a jogador/a poder obter e demonstrar esse conhecimento. Achou-se por bem, neste projeto, que

isso se refletisse na criação de três caixas de madeira onde se encontram várias bolas metálicas, e que o/a jogador/a terá de colocar num determinado sítio pela ordem que for indicada – por exemplo se o código for 3-5-7, terá de colocar primeiro a caixa com 3 bolas, depois a com 5 e finalmente a que contém 7 bolas.

Tendo isto em consideração, chegou-se então à seguinte listagem de sons:

Tabela 5 - Listagem de Sons do MVP do puzzle Boxes

Som	Duração	*Loop*	Integração	Notas
Bola a Rolar	2s	Sim	"velocidade" (0-1)	Bola de metal de pequenas dimensões (~2cm de diâmetro) a rolar numa superfície de madeira
Bola a Bater	< 1s	Não	One-Shot/Trigger	Bola de metal de pequenas dimensões (~2cm de diâmetro) a bater numa superfície de madeira
Caixa de Madeira a Colidir	< 1s	Não	One-Shot/Trigger "carpete", "madeira", "metal", "cimento"	Som de caixa de madeira a bater em superfícies

Contudo, à semelhança do puzzle *Dropper*, uma simples maneira de introduzir complexidade e variedade neste puzzle, é introduzir novos elementos de *sound design*. E se as bolas não fossem de metal, mas sim também de madeira? Ou de plástico? Ou de vidro? Ou se dentro da mesma caixa houvesse bolas de diferentes materiais?

Ou, por outro lado, e se as caixas não fossem de madeira, mas sim de metal? Ou de plástico? Ou de cerâmica?

Ou, virando-nos mais para a parte de simulação de físicas no motor de jogo e o como elas afetam o áudio a ser reproduzido, se em vez de caixas cúbicas ou paralelepipedais tivessem geometrias mais complexas?

Todas estas hipóteses seriam viáveis e interessantes neste contexto. Contudo, estamos sempre limitados pelo tempo disponível para o desenvolvimento deste projeto. Mesmo assim, após desenvolvermos o nosso MVP, poderá ser interessante explorar alguns destes territórios, e formar uma nova listagem de sons semelhante à seguinte:

Tabela 6 - Possível Listagem de Sons do puzzle Boxes

Som	Duração	*Loop*	Integração	Notas
Bola de Metal a Rolar	2s	Sim	"velocidade" (0-1) Superfície: “madeira”, “metal”, “plástico”	Bola de metal de pequenas dimensões (~2cm de diâmetro) a rolar em [superfície]
Bola de Metal a Bater	< 1s	Não	One-Shot/Trigger Superfície: “madeira”, “metal”, “plástico”	Bola de metal de pequenas dimensões (~2cm de diâmetro) a bater numa [superfície]
Bola de Plástico a Rolar	2s	Sim	"velocidade" (0-1) Superfície: “madeira”, “metal”, “plástico”	Bola de plástico de pequenas dimensões (~2cm de diâmetro) a rolar numa [superfície]
Bola de Plástico a Bater	< 1s	Não	One-Shot/Trigger Superfície: “madeira”, “metal”, “plástico”	Bola de plástico de pequenas dimensões (~2cm de diâmetro) a bater numa [superfície]
Bola de Vidro a Rolar	2s	Sim	"velocidade" (0-1) Superfície: “madeira”, “metal”, “plástico”	Bola de vidro de pequenas dimensões (~2cm de diâmetro) a rolar numa [superfície]
Bola de Vidro a Bater	< 1s	Não	One-Shot/Trigger Superfície: “madeira”, “metal”, “plástico”	Bola de plástico de pequenas dimensões (~2cm de diâmetro) a bater numa [superfície]
Caixa a Colidir	< 1s	Não	One-Shot/Trigger Superficie: "carpete", "madeira", "metal", "cimento" Material: “madeira”, “metal”, “plástico”	Som de caixa de [material] a bater em [superfície]

Estas sugestões introduziriam complexidade tanto na parte de design de som como na parte de implementação, mas seriam benéficas para fazer o projeto ainda mais robusto. Contudo, deverão apenas ser consideradas e/ou expandidas no caso de se verificar que temos essa capacidade a nível de calendário de desenvolvimento.

Capítulo III - Desenvolvimento

3.1. Prototipagem

3.1.1. Tentativa falhada com Steam Audio

Tendo em conta as decisões que tinham sido tomadas com base na informação recolhida no capítulo 2.2.2. desta monografia, a primeira tentativa de prototipagem serviu-se de recurso ao Unity como motor de desenvolvimento, e ao FMod e Steam Audio como elementos do motor de áudio do projeto.

3.1.1.1. Protótipo do *Dropper*

Começou-se por se desenvolver o puzzle "Dropper" e, de acordo com o MVP apresentado, numa nova cena do Unity, criou-se um plano com um Mesh Collider e com um Physical Material com um baixo fator de Bounciness (0.1), e dois cubos. A cada um dos cubos adicionou-se um Box Collider como trigger (Is Trigger = true) de Size 1.01 no X, Y e Z, um Rigidbody com Use Gravity = true e finalmente um outro Box Collider onde se adicionou um Physical Material correspondente a uma "Heavy Crate", na qual se definiu um Bounciness = 0.1, e a uma "Light Crate", com Bounciness = 0.8, podendo assim criar o prefab de um cubo que reagirá na simulação de física como se fosse bastante pesado e outro que reagirá como se fosse relativamente leve.

Criou-se depois um novo *script* em C#, e foi adicionado a ambos os `prefabs` criados acima. Neste *script*, adicionou-se um `enum` referenciando se se tratava do objeto "heavy" ou "light", e referência a dois eventos do FMod: o de impacto, e um de feedback aural para questões de *debugging*. Depois, tendo em conta o `Mesh Collider` que tínhamos preparado anteriormente com `Is Trigger = true` e usando o método `OnTriggerEnter()` fez-se com que, de cada vez que um outro objeto entra no `trigger` do nosso cubo, cria-se uma instância sonora com referência ao som de impacto, são passados os parâmetros de `weight` e `fallVelocity` que descrevem se se trata de uma caixa "heavy" ou "light" e a sua velocidade, e reproduz-se de seguida o som que lhe for associado no FMod. Uma vez que conseguimos controlar onde o cubo deverá aparecer, conseguimos ter a certeza de que o único objeto que terá oportunidade de passar neste `trigger` será o chão, o nosso plano criado inicialmente, não precisando de confirmarmos com que é que o cubo está a colidir.

De seguida, criou-se a interação do/da jogador/a com este puzzle. De forma a facilitar este desenvolvimento inicial, definiu-se que, depois de instanciado um dos `prefabs` na

cena, caso o/a jogador/a carregasse na tecla “p” estaria a dizer que o cubo correspondia a um objeto pesado (ou “heavy”) e se carregasse no “l” correspondia a um objeto leve (ou “light”). Uma vez que o cubo, através do nosso *script*, tem referência se ele próprio é “heavy” ou “light”, fez-se com que caso o/a jogador/a carregasse na tecla correspondente ao peso atribuído ao cubo, seria despoletado uma indicação sonora de “correto” (um “ding”) e o cubo seria destruído, ou caso carregasse na tecla não correspondente ao peso atribuído ao cubo seria despoletado uma indicação sonora de “incorreto” (um “bzz”).

De seguida implementou-se um *spawner* onde, de cada vez que o/a jogador/a carregasse na tecla “espaço”, caso não houvesse já um dos `prefabs` na cena, criaria aleatoriamente um dos cubos ligeiramente acima do nosso plano de chão, de forma que o cubo pudesse impactar e ressaltar, fazendo o som correspondente de cada uma das vezes, como foi anteriormente tratado.

É de notar que as ações que estão até aqui ligadas às teclas “l”, “p” e “espaço” seriam numa fase seguinte ligadas a botões virtuais presentes na cena, e com os quais se poderia interagir com os controlos de realidade virtual.

Finalmente, de forma a completar este primeiro protótipo, passámos para o FMod, de forma a criar o suporte sonoro para o qual se esteve a desenvolver. Criaram-se dois eventos 3D: o de impacto e o de feedback.

No evento de impacto criaram-se dois novos parâmetros, de forma a receber a informação que estava a ser enviada pelo Unity: o `weight` que tinha referência “heavy” ou “light”, tocando um dos `Multi Instruments` que foram criados para estas diferentes opções, e o `fallVelocity` referente à velocidade instantânea do cubo no momento do impacto, onde se controla, através de automação, o volume e a presença de médios e graves consoante a velocidade, de forma a tornar o som mais verosímil.

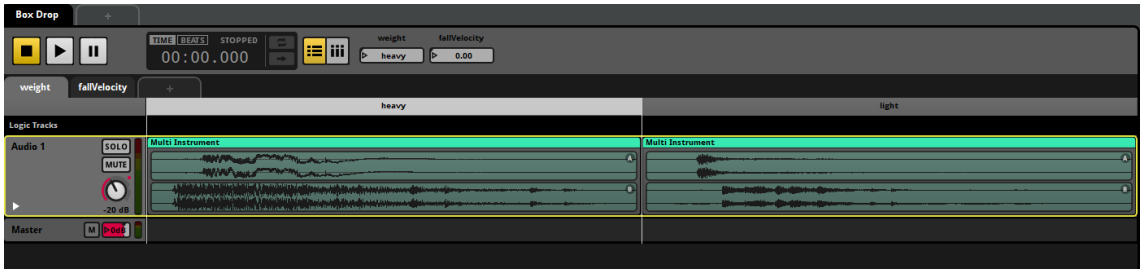


Figura 2 - Parâmetro weight do som de impacto no protótipo do Dropper

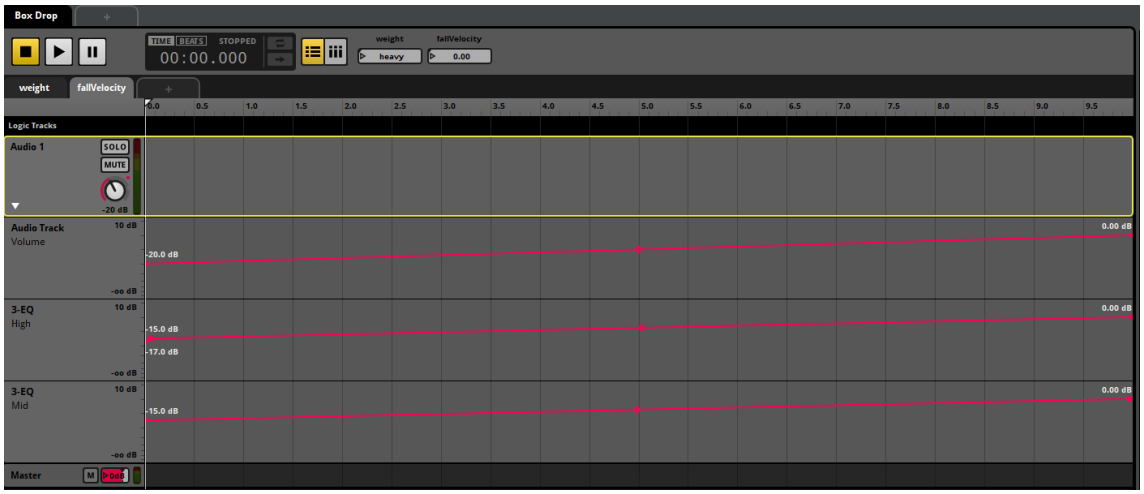


Figura 3 - Parâmetro fallVelocity do som de impacto no protótipo do Dropper

No evento de *feedback*, criou-se um novo parâmetro *feedback* com as opções “correct” e “wrong”, correspondentes a se o/a jogador/a acertou ou errou na sua escolha, respetivamente. Aqui, usou-se apenas um `Simple Instrument` para cada uma das opções, de forma a manter coerentes o *feedback* com as respostas.

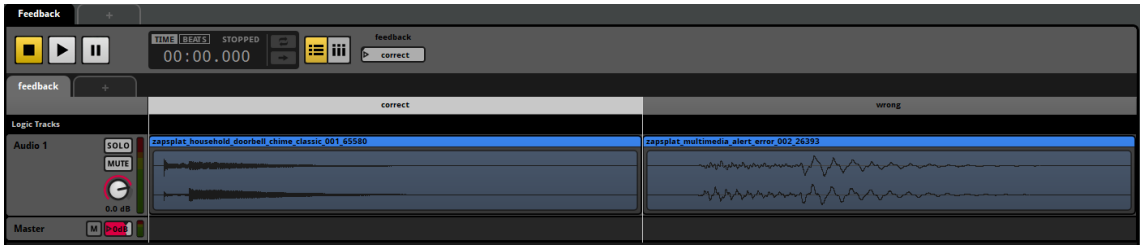


Figura 4 - Parâmetro feedback do som de Feedback no protótipo do Dropper

Estes sons foram arranjados de bibliotecas de som online (MTG, 2021; Zapsplat, 2021), e como estavam a ser usados apenas para efeitos de *debug* não foram alteradas. Com o continuar do trabalho, estes sons foram mais trabalhados, aproximando-os do pretendido, consoante a nossa listagem inicial.

3.1.1.2. Protótipo Falhado do *Labyrinth*

Para ao desenvolvimento do protótipo inicial deste puzzle, começou-se por implementar o *plugin* do Steam Audio no projeto de Unity e no FMod (Valve, n.d.), uma vez que a reflexão e reverberação do som tendo em conta a geometria da cena e a posição da fonte sonora relativa ao avatar do/da jogador/a seriam essenciais para que este funcionasse.

A implementação do *plugin* no FMOD é um processo simples: de acordo com a documentação (Valve, n.d.), basta fazer o *download* do .zip com os ficheiros corretos, e

copiá-los para a pasta onde o FMod lê os *third-party plugins*. E, de facto, como podemos ver na imagem seguinte, seguindo esses passos conseguimos ter acesso às funcionalidades de espacialização providenciadas pelo Steam Audio.

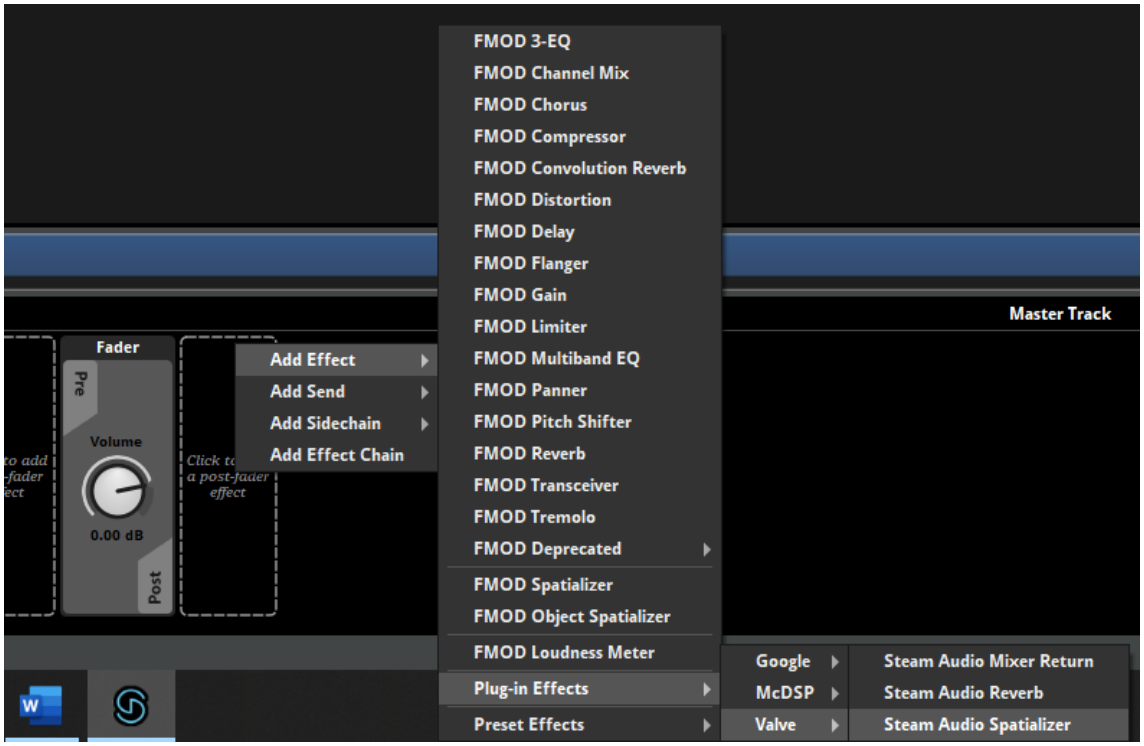


Figura 5 - Ferramentas do Steam Audio no FMod depois de instalado o plugin

O problema surge, contudo, quando se tenta associar o Steam Audio ao Unity. Seguindo de novo a documentação (Valve, n.d.), é necessário primeiramente fazer o *download* e importar o `Custom Package` disponibilizado. De seguida, da seção de Audio dos `Project Settings` deveremos escolher a opção “Steam Audio Spatializer” tanto nas opções de `Spatializer Plugin` como de `Ambisonic Decoder Plugin`. Utilizando a opção de menus “Window > Steam Audio” abrirá um editor que nos permitirá escolher o FMod Studio como sendo o nosso *Audio Engine*. Finalmente, nas opções do Fmod no Unity, disponíveis através dos menus em “FMOD > Edit Settings”, deveremos adicionar “`phonon_fmod`” nos plugins associados. Contudo, quando completamos este passo final, de todas as vezes que carregamos em `Play` de forma a correr o nosso projeto, o Unity apresenta-nos um erro e “crash”, terminando forçosamente o processo. No Anexo 1 podemos ver uma gravação disso a acontecer, apesar de não mostrar o erro de “paragem forçada”.

Não existe nenhuma explicação na documentação que se refira a este erro, nem nenhuma maneira de evitar este último ponto do processo. Chegámos à conclusão que,

devido à sua instabilidade, seria preferível abandonar-se o Steam Audio, em detrimento de outro *software*.

3.1.2. Tentativa com Google Resonance

Tendo-se apercebido que, nas condições atuais, seria impossível completar este projeto com o Steam Audio, passou-se então para a segunda opção na lista de prioridades definidas no capítulo 2.2.2. e seguiu-se com o Google Resonance como *plugin* de espacialização.

Uma vez que estamos a utilizar o FMod como *middleware*, o uso do Google Resonance com o Unity torna-se bastante trivial de implementar: há algumas questões técnicas que iremos discutir nos capítulos 3.1.2.1. e 3.1.2.3. , como a criação de zonas de reverberação, mas se quisermos espacializar uma fonte sonora basta retirarmos ou desativarmos o especializador por defeito do Evento do FMod que quisermos espacializar, adicionar o item “Resonance Audio Source” (Figura 6) e finalmente, no *master bus* do projeto do FMod (ao qual podemos aceder através do menu superior em Window > Mixer) deveremos adicionar o item “Resonance Audio Listener” (Figura 7).

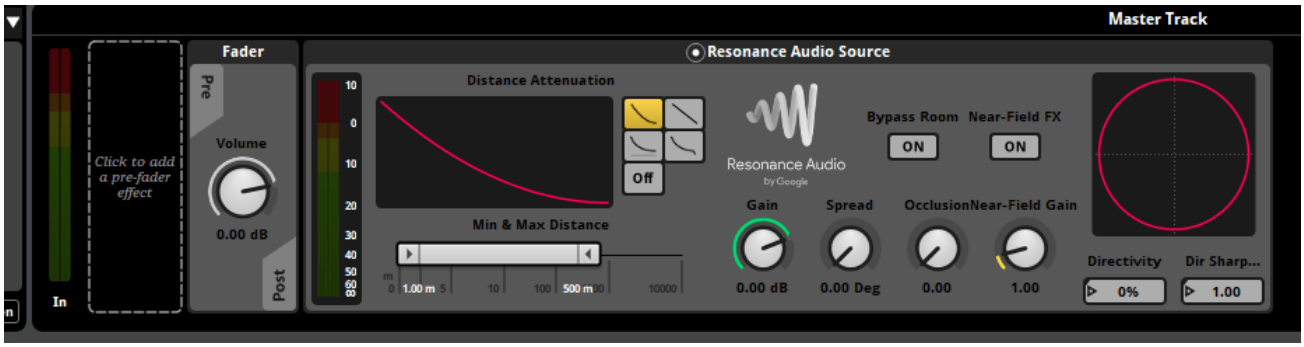


Figura 6 - Exemplo de Evento FMod espacializado

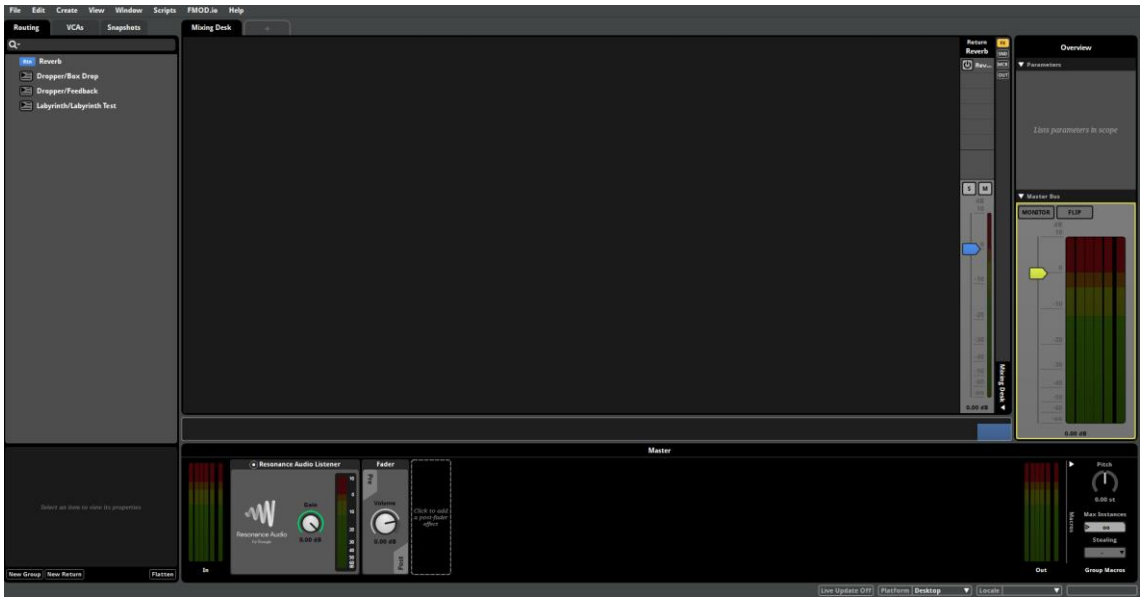


Figura 7 - Master Bus com item Resonance Audio Listener

Uma vez que o Google Resonance vem acoplado automaticamente com o FMod, não é necessária a instalação de mais nenhum software nem de mais nenhuma preparação para termos acesso por defeito aos seus efeitos no FMod.

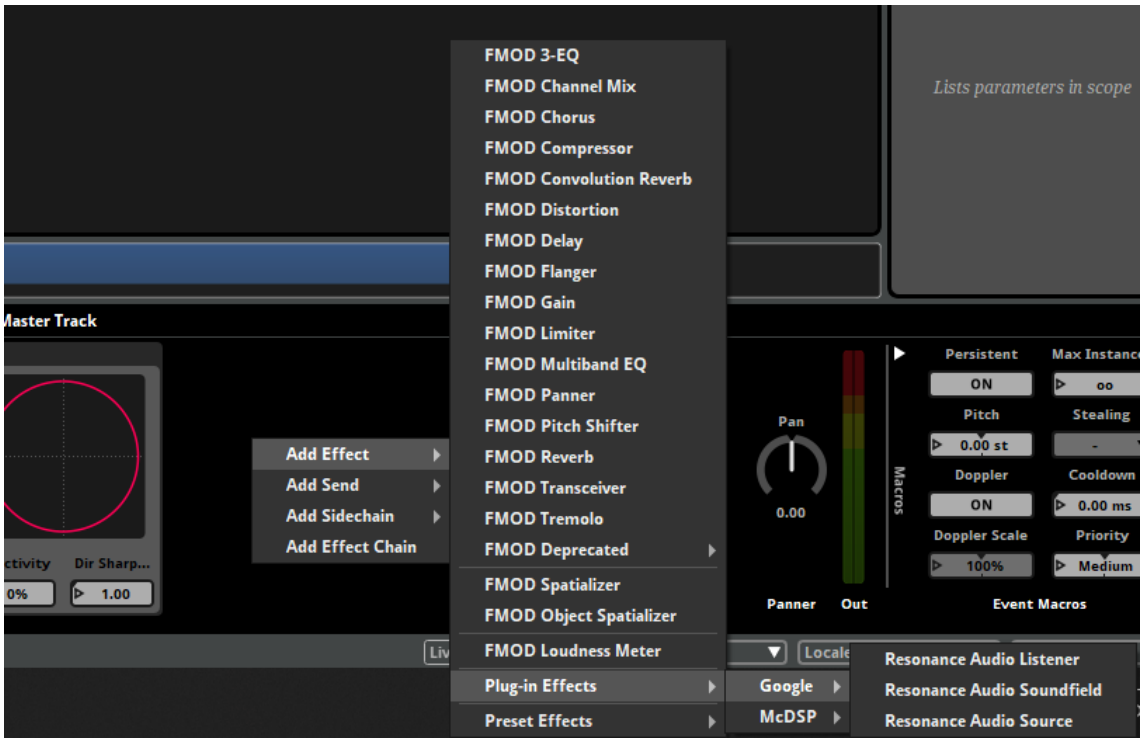


Figura 8 - Ferramentas do Google Resonance nativamente presentes no FMod

Para podermos utilizar as componentes do Google Resonance no Unity, precisamos de fazer o download e importar o `Unity package` mais recente, que à data da escrita

desta monografia é a versão 1.2.1. Para além disso, precisaremos apenas que a nossa câmara tenha uma componente `FMod Listener`.

3.1.2.1. Protótipo Funcional do *Labyrinth*

O puzzle do *Labyrinth* vive de som direcional e zonas de reverberação. O posicionamento da fonte sonora, assim como a sua direccionalidade, será tratado automaticamente pelo FMod com a sua componente de espacialização ligada ao Google Resonance. Existem, contudo, duas formas de criar zonas de reverberação com o Google Resonance num motor de jogo: 1) através da criação de `Room Effects`, também referidas apenas como *Rooms*; 2) fazendo *bake*[2] utilizando `Reverb Probes`. (Google, n.d.)

A primeira abordagem foi utilizar-se *Rooms*, uma vez que seria a abordagem mais simples de ser implementada. *Rooms* são salas virtuais, cujas dimensões e materiais acústicos replicados podem ser definidos, mas mantendo sempre uma forma paralelepipédica. Isto significa que, para o nosso caso do labirinto, teríamos de ter uma *room* por cada segmento de corredor, criando uma nova *room* sempre que tivéssemos de rodar à esquerda ou direita, uma vez que não podemos criar, por exemplo, uma forma L ou Estrela. O problema com esta técnica, é que não existe *bleed* de reverberação entre salas adjacentes – isto é, há uma mudança muito notória quando estamos na *room* onde se encontra a fonte sonora ou não, porque elas são estanques entre si, não criando um "túnel de reverberação", mas sim segmentos individuais onde a reverberação acontece, tornando o efeito que se pretende atingir com este puzzle impossível. Podemos ver um exemplo disto a acontecer no Anexo 2.

Por este motivo, fez-se uso de *Reverb Probes*. A desvantagem desta técnica é o ser necessário fazer-se *bake* da informação de reverberação, impossibilitando que se possa simplesmente mudar os materiais acústicos do labirinto numa dada sessão de jogo. Contudo, essa vontade de potencialmente mudar os materiais acústicos do labirinto é apenas algo a ter em consideração caso se consiga atingir o objetivo de ter o puzzle a funcionar e com tempo de desenvolvimento de sobra, e não um pré-requisito para que ele funcione. Para além disso, pode-se também, em vez de fazer a substituição dos materiais, construir o labirinto de forma modular e fazer com que cada um dos módulos

[2] *Bake* ou *Baking* é definir um conjunto de elementos numa projeção estática que é calculada no momento de desenvolvimento em vez de ser continuamente calculada durante uma sessão de jogo

possa ser substituído por um outro pré-calculado com diferentes materiais acústicos – um processo mais moroso, mas que atingiria resultados semelhantes no final.

Iniciou-se criando um bloco de corredor, que serviria como a base para cada um dos módulos do labirinto. Cada um destes blocos é constituído por um chão, um teto, quatro paredes e um `reverb probe`, todos filhos de um objeto vazio para mais fácil edição. As superfícies são feitas com cubos redimensionados e reposicionados de forma coerente, e todos eles têm um `Box Collider`, de forma a evitar que o/a jogador/a possa sair do cenário. O `reverb probe` é um objeto vazio, ao qual adicionamos o componente `ResonanceAudioReverbProbe`, o que faz com que apareça um visualizador da área abrangida por este *`probe`*. O *`probe`* foi depois redimensionado de forma a abranger todas as superfícies deste módulo.

Para fazer o *bake* da reverberação, precisamos de atribuir materiais a cada uma das superfícies. Nesse sentido, criou-se alguns materiais visuais que viriam a servir como referência a materiais acústicos, e atribuiu-se a cada uma das superfícies – para estes testes iniciais, o chão seria feito de mármore, as paredes de tijolo e o teto de madeira. De seguida, é necessário atribuir as características acústicas a cada um dos materiais criados num `Resonance Audio Material Map`. Utilizou-se o mapa que vinha no exemplo de testes no *package* que tem de ser importado do Google Resonance para o Unity, mas para criar um novo basta seguir-se os passos que se seguiriam para criar qualquer outro objeto no Unity: clique com o botão direito do rato no inspetor do projeto ou ir no menu superior direito a Assets > Create > ResoanceAudio > Material Map. Selecionando esse `material map`, aparecer-nos-á no inspetor todos os materiais visuais existentes nos *assests* do nosso projeto, e deveremos de seguida atribuir a cada um deles qual o material acústico a que queremos que ele corresponda, selecionando "`Transparent`" caso não queiramos que ele seja considerado nos cálculos. Com essa parte tratada, devemos de seguida no menu do topo selecionar ResonanceAudio > Reverb Baking e, na janela que nos aparece, escolher o `material map` no qual trabalhamos, selecionar todos os `reverb probes` que queremos e clicar em `Bake` para dar início aos cálculos. É preferível realizar este último passo apenas quando fizermos alterações que queiramos testar, de forma a não abrandar o processo de desenvolvimento, ou no final antes de exportarmos o projeto.

Com este módulo pronto, criou-se uma primeira sequência de módulos, editando as suas superfícies de forma a garantir a continuidade do espaço virtual, de forma a criar um labirinto teste, e fez-se *bake* de todos os *probes*. Com esse *layout* definido para já,

criou-se um cubo que servisse como fonte sonora e, no FMod, criou-se um evento 3D chamado *Labyrinth Test* onde foi colocada uma música para efeitos de teste. Na *master track* deste evento, apagou-se o `Spatializer` que vem por defeito e adicionou-se o `Resonance Audio Source` e, finalmente de novo no Unity, adicionou-se o componente `FMOD Studio Event Emitter` à fonte sonora, com a referência a este nosso evento teste.

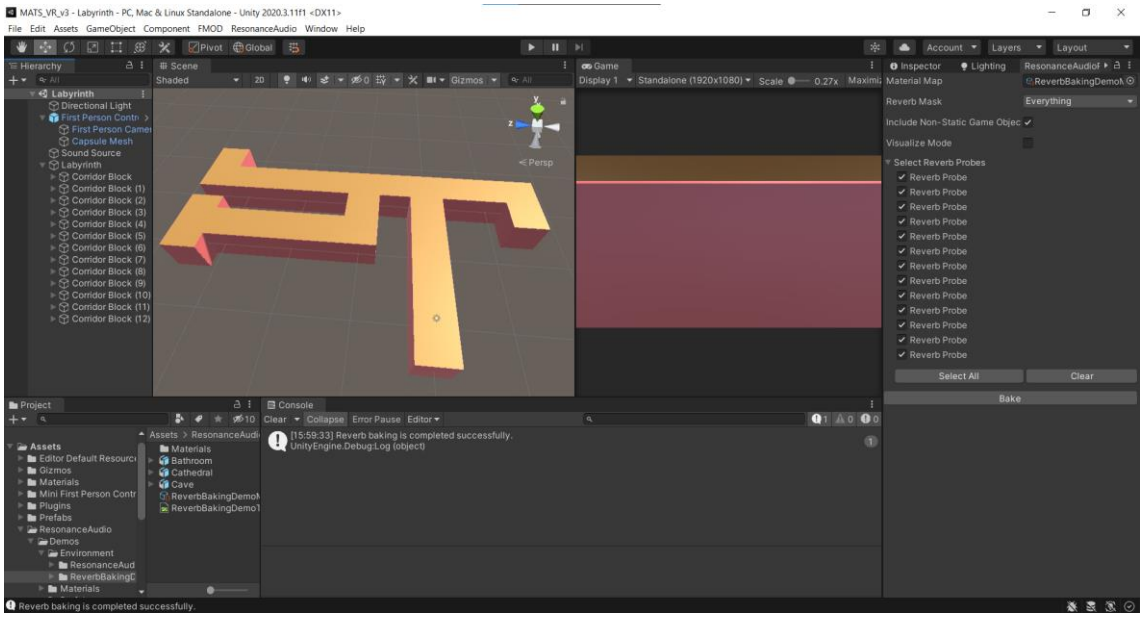


Figura 9 - Labirinto Teste no Unity

De forma a podermos explorar o ambiente 3D criado, utilizou-se um *asset* disponível na loja online do Unity chamado “Mini first person controller” desenvolvido pelo grupo Vitefait. Fazendo uns pequenos ajustes ao seu `prefab First Person Controller`, nomeadamente fazendo com que a câmara utilize o componente `FMod Studio Listener` em vez do que vem por defeito no Unity, é possível agora navegar o labirinto criado e confirmar que tudo está a funcionar dentro do espectável, como podemos ver no Anexo 3.

3.1.2.2. Protótipo *Boxes*

Com os inícios do *Dropper* e com um protótipo do *Labyrinth*, virei-me finalmente para o puzzle *Boxes*. Relembrando, o MVP deste protótipo seria ter a criação de três caixas iguais, cada uma contendo um número diferente do mesmo objeto, e uma maneira de o/a jogador/a poder obter e demonstrar esse conhecimento. Para termos três caixas, temos de começar por apenas uma.

Criou-se um objeto vazio, e seis cubos filhos – cada um representando uma face de um cubo por si, e cada um com um `Box Collider`. Redimensionou-se e reposicionou-se os objetos de forma a terem, todos, as mesmas dimensões e criassem um espaço no seu interior, criando assim uma caixa fechada. Ao objeto pai, adicionei um componente `Rigidbody`. Criei um outro cubo de apoio, sobre o qual esta caixa repousa, apenas com um `Box Collider`, de forma que a caixa não caísse para o infinito.

Com o esqueleto da caixa criado, criei um objeto esférico de relativas pequenas dimensões no seu interior, que servirá como a base do que virão a ser as "bolas dentro da caixa". A esta esfera adicionei também um componente `Rigidbody` que usa a gravidade, e dois componentes `Sphere Collider` – um que servirá de forma que a bola consiga interagir com as paredes da caixa e não saia quando esta é movida, e outro com o elemento `Is Trigger` ativo e com um `Radius` ligeiramente maior que o raio da esfera, que servirá como o detetor de colisões e irá fazer com que consigamos detetar com o que é que a esfera colidiu e, portanto, que som deverá ser reproduzido.

Com estes dois elementos criados, era possível testar as colisões. Correndo a cena e, utilizando as ferramentas do editor, conseguíamos ver que, de facto, as simulações de física parecem estar a funcionar conforme o esperado, e que a bola se desloca quando movemos a caixa. Contudo, se fizermos movimentos demasiado bruscos, a bola "atravessa" as paredes, e perde-se, caindo para o infinito. De forma a testar se isto se devia ao uso das ferramentas do editor em vez de interagirmos com os objetos programaticamente, possivelmente fazendo com que a deteção de física quebrasse, fez-se um pequeno *script* em que, se pressionarmos a tecla numérica "1" no teclado, é possível adicionar uma força de `1000` unidades no eixo do y (vertical) à caixa, e se pressionarmos a tecla "2" do teclado é possível adicionar uma força de `10000` unidades nesse mesmo eixo. Como é possível verificar no Anexo 4, o que acontece é que com uma força relativamente pequena, a física funciona normalmente, mas se adicionarmos uma força muito grande o sistema de física tem dificuldades em calcular essa mudança e a esfera sai da caixa. No vácuo, é complicado saber se, ao interagir com a caixa, os/as jogadores/as conseguirão atingir forças suficientes tais que o motor de física tenha dificuldades em acompanhar, e, portanto, será necessário verificar se este problema se revela mais à frente no processo de desenvolvimento. Contudo, há formas de forçar a que as bolas sejam transportadas para dentro das caixas novamente, caso a força seja de grande magnitude, e por isso foi adicionado uma condição que testa isso no *script* acima descrito: se a magnitude da velocidade do `rigidbody` for acima do valor dado e essa condição estiver ativa, então deve-se fazer com que a posição da esfera seja uma

pré-definida, dentro da caixa. Isto será uma forma de combater este erro, caso se continue a verificar mais à frente no desenvolvimento, mas, para já, vamos apenas manter isso em mente.

No FMod criou-se três eventos 3D: *Ball Roll*, *Ball Impact* e *Box Impact*, correspondentes ao som da esfera a rolar, a bater na caixa e da caixa a bater noutras superfícies, respetivamente. Uma vez que todos estes eventos serão espacializados, adicionou-se na *master track* de todos o *Resonance Audio Source* e removeu-se o *Spatializer* que vem por defeito.

Uma vez que tanto o *Ball Roll* como o *Ball Impact* são elementos sonoros que vão estar a acontecer dentro de caixas fechadas (até este ponto idealizadas como sendo de madeira), adicionou-se, também, um *3-EQ*, removendo -6dB aos *Mid* e -12dB aos *High*. Para além disso, é de notar que ambos os eventos poderão vir a ter um parâmetro *label/string*, consoante avancemos no desenvolvimento, se chegarmos ao ponto de se mudar o material da caixa e/ou das bolas, como denotado no capítulo 2.4.3. desta monografia

No evento *Ball Roll*, adicionou-se um parâmetro contínuo (`float`) chamado *rollVelocity*, que fica responsável pela atenuação de volume, através de uma automatização do ganho da *master track*.

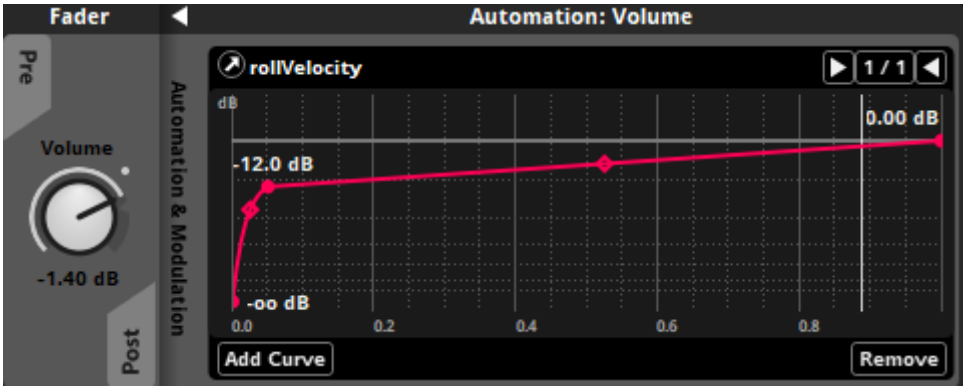


Figura 10 - Automatização do ganho do master, ligado à rollVelocity

No evento *Ball Impact*, foi adicionado um parâmetro contínuo chamado *ballVelocity*, que fica responsável pela atenuação de volume, através de uma automatização do ganho da *master track*, com uma curva idêntica à da Figura 10.

Da mesma forma como no capítulo 3.1.1.1. desta monografia, para estes testes iniciais foram usados sons provenientes de uma biblioteca de sons online, com a intenção de se apurar o design sonoro mais à frente no desenvolvimento. Para já, as únicas

alterações foi tornar o som que vai estar associado à esfera rolar um *loop* perfeito, cortando o som em duas partes e trocando-as de ordem com um *crossfade* entre elas. Para garantir esse *loop* quando o evento fosse tocado, no FMod foi adicionado uma *Loop Region* ao evento “Ball Roll”.

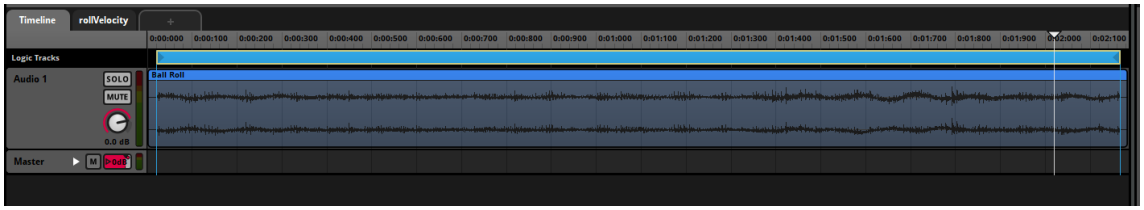


Figura 11 - Ball Roll com loop region

Criou-se um *script BallController* e adicionou-se à esfera presente na cena. Este *script* faz referência aos eventos de FMod *rollEvent* e *impactEvent*, assim como ao `Rigidbody` da esfera. De seguida, instanciou-se dois esses dois eventos do FMod na cena e iniciei, no método `Start()` que é chamado quando o objeto é inicializado na cena, o evento `rollEvent`. Uma vez que temos a curva de atenuação e que podemos passar em tempo real a velocidade da esfera, este evento poderá estar a tocar constantemente e o seu volume estará dependendo da velocidade instantânea. Assim, no método `FixedUpdate()` está-se a ser a dado *set* do parâmetro “rollVelocity”, passando a magnitude da velocidade angular da esfera. Para além disso, no método `OnTriggerEnter()`, despoletado de cada vez que um objeto entre no `trigger` da esfera previamente mencionado, está-se a dar *set* do parâmetro “ballVelocity” do evento de impacto, e a iniciá-lo, de forma a que quando a esfera embata na caixa (ou noutra esfera), se oiça esse impacto, de acordo com a velocidade da esfera naquele momento.

Finalmente, faltou uma forma de comparar se a caixa selecionada é, de facto, correspondente com o pretendido. A ideia é usar depois três locais onde são colocadas três caixas com diferentes bolas dentro delas, correspondendo a caixa com um certo número de bolas ao local pretendido (por exemplo, se tivermos um 3 escrito nesse local, colocar lá a caixa com 3 bolas dentro dela). Para isso, no Unity, foram criadas novas `tags`, que irão corresponder ao número de bolas dentro de cada caixa, e criou-se para efeitos de teste um novo objeto vazio com a propriedade de `BoxCollider` mas sem qualquer tipo de `MeshRenderer`, ao que se chamou “Boxes Comparer”. Adicionou-se também um `BoxCollider` ao objeto pai da caixa, que tem dentro dele como filhos todos os lados que a componham, e ativou-se a opção de `Trigger`, com um `size` de 1.5 em todas as dimensões. Foi criado um novo *script* e adicionou-se a este objeto “pai caixa”, onde no método `OnTriggerEnter()` se consegue comparar o objeto com que

se colidiu com aquilo que seria esperado. Para estes primeiros testes, foi confirmado se o objeto com que se colidiu foi o “Boxes Comparer” e, se sim, qual a sua `tag`. Se a `tag` for a mesma que a da caixa, estamos a fazer um `Debug.Log("correct")` e se não for um `Debug.Log("incorrect")`. Estes `logs` são puramente para efeitos de testes, mas permitem duas coisas: confirmar que a logica está a funcionar corretamente, e ter um local onde, assim que quisermos, podemos inserir nova lógica consoante o que quisermos fazer com a completação do puzzle.

De momento, pode ser deixado apenas assim, com a referência que, quando deixarmos de ter este puzzle no vácuo, consegue-se controlar a lógica do mesmo.

3.1.2.3. Melhoria do Protótipo do *Dropper*

Agora que o nosso *plugin* de espacialização está a funcionar corretamente, foi possível melhorar o protótipo *Dropper*, que já estava funcional. Começou-se por criar um ambiente virtual que se aproximasse do finalizado: um espaço amplo reminiscente de um armazém, com uma plataforma de visualização, que o/a jogador/a poderá explorar, com 3 botões (um para fazer deixar cair as caixas, e 2 para selecionar se a caixa era pesada ou leve) e um espaço onde se irá dar a recompensa por ter acabado o puzzle. Moveu-se o *spawner* das caixas para o fundo deste armazém, oposto à zona de visualização e adicionou-se uma série de luzes do tipo `Spotlight` a iluminar essa área. Criou-se também um `Point Light` na zona de visualização para iluminar esse espaço. Adicionou-se também o nosso “Mini first person controller” adaptado, que havia sido usado para o *Labyrinth*, como descrito no ponto 3.1.2.1. desta monografia.

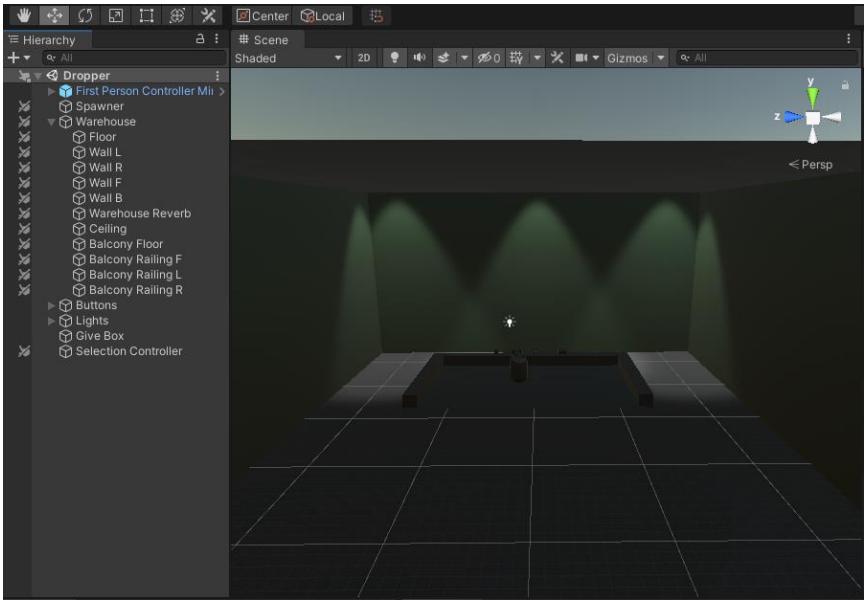


Figura 12 - Screenshot do espaço virtual criado para o puzzle Dropper

Uma vez que a intenção é, depois de garantir que o/a jogador/a entendeu como o puzzle funciona, remover-lhe a visão do espaço onde caem as caixas, criou-se um objeto vazio chamado “Lights Controller” onde se adicionou um novo *script*, que fica à espera que se pressione a tecla “1” do teclado para iniciar uma pequena animação onde as luzes piscam e depois desligam e ligamos a componente de renderização `fog`, criando um nevoeiro escuro em volta do/da jogador/a, impedindo que se consiga ver o local onde as caixas caem, mas conseguindo ainda ver toda a zona de visualização, e podendo portanto ainda interagir com os botões que lá se encontram. A animação de cada luz a piscar está a ser tratada num *script* separado, que em tempos aleatórios entre `0.1` e `0.5s` dá `enable` ou `disable` do objeto a que está associado, sendo que este *script* foi associado a todas as luzes que deveriam apresentar este comportamento.

Com a componente visual tratada, para já, avançou-se para a componente acústica. Com o Google Resonance, criou-se uma nova *Room* e fez-se tal que a sua dimensão fosse igual à do armazém criado. Depois, fez-se com que as suas superfícies apresentassem as propriedades acústicas de Metal exceto o chão que apresenta a propriedade de um bloco de cimento grosso.

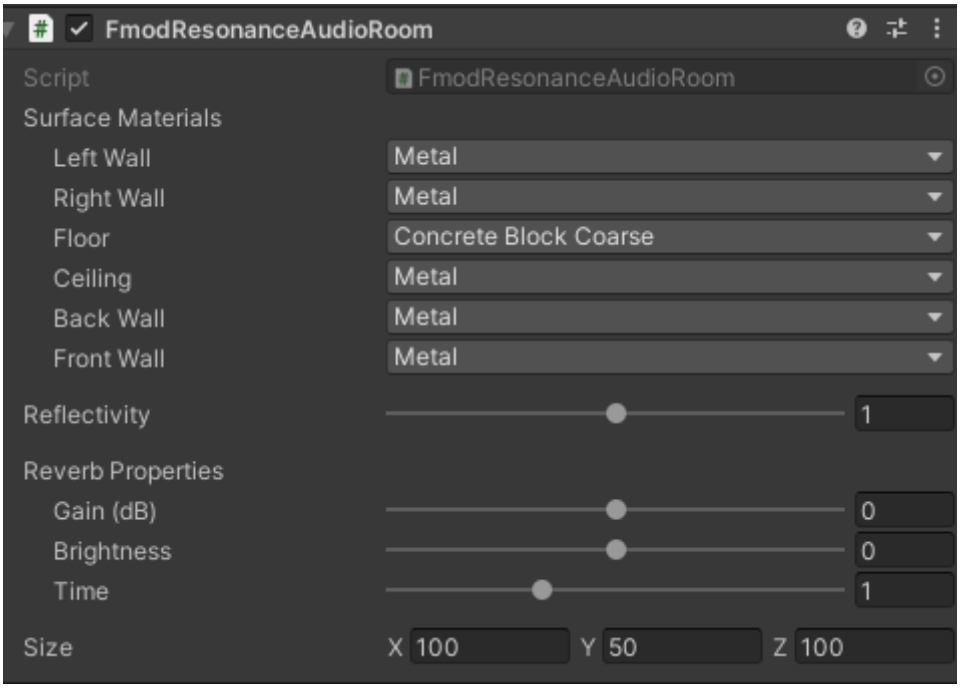


Figura 13 - Propriedades da Room do Dropper

Assim, quando as caixas embatem no chão e fazem barulho, conseguimos ter uma melhorada experiência aural, mais coerente com o que vemos no espaço virtual, comparando com a nossa experiência de espaços e sons do nosso dia-a-dia.

Agora que se tinha um espaço aproximado ao produto final, foi também altura de tratar de ter interações com o ambiente que se aproximem ao produto final. Em vez de se clicar em teclas no nosso teclado para fazer com que as ações aconteçam, usar-se-á os botões que colocamos na nossa zona de observação para fazer com que as caixas sejam instanciadas e para escolher se se trata de uma caixa pesada ou leve. Uma vez que a lógica já está tratada, basta passar-se o *input* do teclado para o mundo virtual.

De forma a se interagir com os botões do mundo virtual, criou-se um `raycast` centrado na câmara do jogo. Isto criará uma linha invisível do centro da câmara estendendo-se para a frente, e da qual se poderá retirar informações sobre os objetos que se atravessarem nessa área (Unity, 2021). Deu-se `tags` aos objetos que serviriam de botão, de forma que se pudesse comparar com a informação recebida no `raycast` e, quando o/a jogador/a estiver a olhar diretamente para o objeto, se clicar com o botão esquerdo do rato, despoletará a função associada a esse botão. De forma a indicar que o objeto é de facto algo com que se possa interagir, pôs-se de momento uma mudança de material, tornando-o do seu material *default* para um amarelo. Com estas alterações, o/a jogador/a poderá agora clicar no botão mais à esquerda para instanciar uma caixa, e usar os outros dois botões para mostrar se se trada de uma pesada ou leve – se bem que ainda é necessário fazer uma diferenciação aos botões de forma a demonstrar qual é qual.

Visto que é boa prática dar *feedback* num jogo quando uma ação é efetuada (Adams, 2013; Schell, 2015), achou-se por bem criar uma pequena animação nos botões que, quando fossem pressionados, descessem e subissem novamente – semelhante a botões reais. Para isso, adicionou-se o *plugin* “DoTween”, que permite a criação de animações de objetos por código (Demigiant, n.d.). Desta forma, poderá ser criada uma animação genérica que todos os botões obedecerão, sem termos de nos preocupar com a componente de animação e animadores do Unity, o que poderia custar mais tempo de desenvolvimento, uma vez que teria de ser efetuado botão a botão. Criou-se então uma animação em que o objeto a que o *script* seja associado move-se `0.3` unidades no eixo vertical para baixo em `0.5s` e de seguida volta à sua posição original em `0.5s`.

Chegou-se, assim, a uma versão melhorada do *Dropper*, faltando ainda definir após quantas tentativas se desligam as luzes e quantas depois se considerará o puzzle completo e se irá oferecer a recompensa ao/à jogador/a – coisas a testar ainda.

3.2. Composição coerente das cenas jogáveis

Tendo-se chegado a um ponto em que se acreditava que todos os puzzles estavam prontos a ser testados num cenário único, fez-se `prefab` de cada um dos puzzles nas suas cenas de teste e importaram-se para uma cena única, à qual se chamou “Main”. Organizaram-se as geometrias de forma semelhante à descrita no *level design* presente no capítulo 2.1. . Criou-se também uma sala inicial, onde o/a jogador/a iria começar o jogo, e onde estavam os pilares de comparação do puzzle *Boxes*, e um elevador que transportaria o/a jogador/a da zona do *Labyrinth* para a zona do *Dropper*.

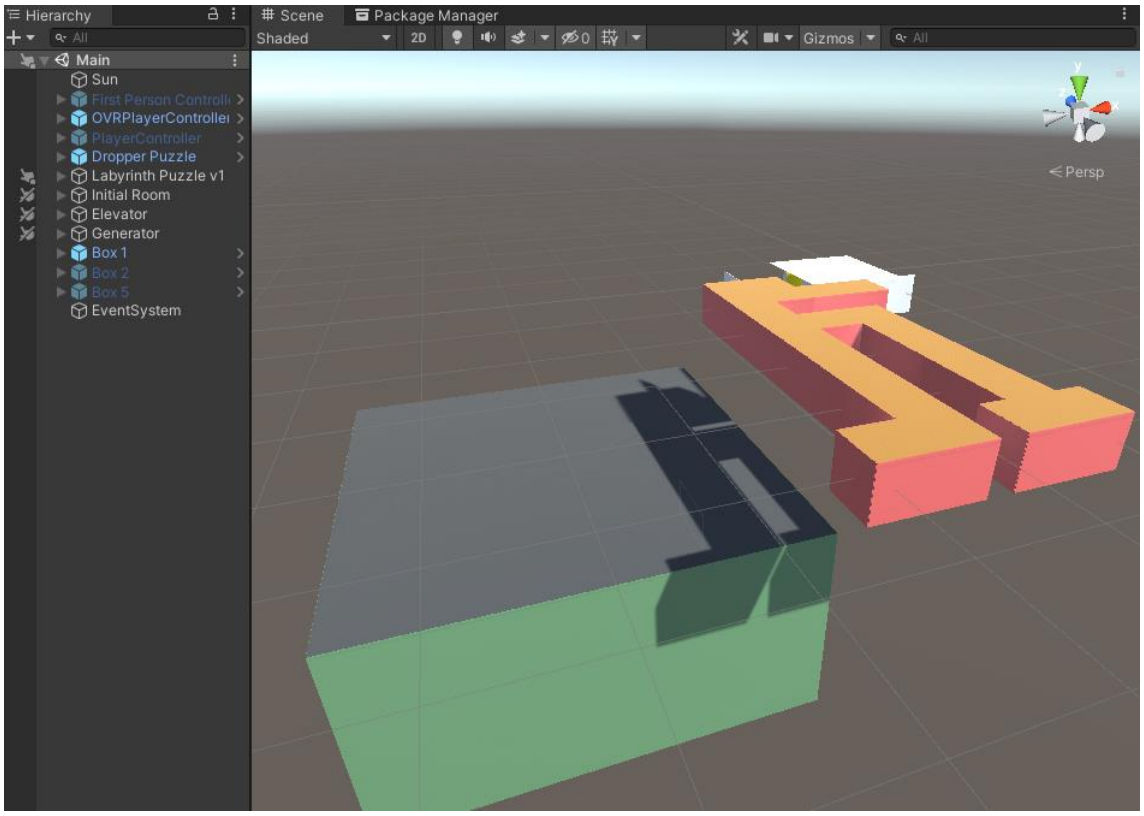


Figura 14 - Puzzles Dropper, Labyrinth e Boxes, todos na mesma cena de Unity

Como descrito no capítulo 2.1. , a ideia inicial era o/a jogador/a começar na zona inicial com os pilares e uma única caixa, explorar o labirinto e encontrar o elevador. Assim que tentasse interagir com o elevador, simular-se-ia uma falha de eletricidade, e o/a jogador/a teria que explorar o labirinto, utilizando dicas sonoras para chegar ao gerador e reiniciar a corrente. Fazendo isto, receberia a segunda caixa, e poderia agora mover-se para o puzzle do *Dropper* utilizando o elevador. Aí, poderia interagir com os botões de forma a deixar cair caixas e completar o puzzle, recebendo a terceira caixa. Trazendo todas as caixas de volta para a sala inicial, e colocando-as nos pedestais de comparação certos, terminaria o jogo. Contudo, como se vai entrar mais em detalhe no capítulo 3.3. , não foi possível chegar a um ponto onde esta sequência de eventos

estivesse estável e apresentável, tanto a tempo das sessões de *playtest* como para a entrega final. Não o sabendo ainda nesta altura do projeto, continuou-se o trabalho.

De forma que o puzzle *Boxes* funcionasse, criaram-se então 3 caixas com um diferente número de bolas dentro delas: a caixa da sala inicial teria 1 bola, a do gerador 5 e o do *Dropper* 3. Desativaram-se as caixas do gerador e do *Dropper*, e fez-se tal que apenas se ativassem novamente quando se interagisse com o botão para reativar o gerador e como recompensa de terminar o puzzle do *Dropper*, respetivamente. Para além disso, associou-se o *script* de controlo de luzes descrito no capítulo 3.1.2.3. para que as luzes piscassem e desligassem quando o/a jogador/a interagisse com o elevador da primeira vez fosse mais claro que se perdeu energia, voltando a ligá-las quando se interage com o gerador para o reiniciar.

Para se criar o elevador, criou-se inicialmente um objeto vazio, e com uma série de cubos filhos desse objeto criou-se a geometria do espaço do elevador. Depois, associou-se a esse objeto um *script* com um método que ficava responsável por fazer a translação de movimento da posição da porta do elevador e do elevador em si, tendo uma animação de fecho da porta, seguida de uma translação vertical até ao ponto inferior do *Dropper* e de abertura da porta no final dessa deslocação, e obviamente o equivalente inverso de forma a voltar a subir.

De forma que os/as jogadores/as melhor associassem o espaço do elevador ao mesmo, criou-se um evento “Elevator Music” no FMod, onde se pôs em *loop* a música “Elevator Music” de Kevin Macleod, por se tratar de uma música com um estilo semelhante ao encontrado em elevadores, criando assim uma associação direta na mente dos/das jogadores/as, e por se tratar de uma música *royalty free*, sobre a licença Attribution 3.0 Unported (CC BY 3.0)[3]. Para além disso, fizeram-se gravações de *foley* de um elevador, tendo sido captados os sons da porta a fechar e a abrir, e da viagem do elevador, do lado de dentro do elevador, uma vez que não seria possível sair do elevador do jogo uma vez começada a viagem. Fez-se uma passagem pelo iZotope RX8 dessas gravações, de forma a limpar ruídos indesejados presentes na gravação, e aplicou-se um pouco de compressão de forma a tornar as gravações mais homogéneas. Utilizou-se o Reaper como DAW para fazer a edição das gravações. Exportados os sons, pode-se finalmente incluir no projeto de FMod, criando os eventos “Elevator Travel”, “Elevator

[3] Pode-se encontrar mais informações sobre esta referência no link https://creativecommons.org/licenses/by/3.0/

Door Open" e "Elevator Door Close" para os sons de ambiente da viagem, da porta a abrir e a fechar, respetivamente.

Uma vez que os eventos sonoros relacionados com a porta do elevador estavam anexados à geometria da porta, esta precisava de ter um componente de `Rigidbody`, e o mesmo para o objeto vazio pai da geometria do elevador relativamente ao evento sonoro da viagem de elevador. Contudo, apesar das animações destes objetos estarem a funcionar corretamente antes, a introdução deste novo componente teve efeitos adversos. Como podemos observar no Anexo 5, apesar das fontes sonoras estarem corretamente espacializadas, assim que começava a animação de descida do elevador, a porta apresentava erros e começava-se a afastar, e eventualmente ficava presa no teto do armazém onde se encontra o *Dropper*. Para além disso, a música presente no elevador, que deveria descer com este, mantinha-se estática no piso superior, voltando a poder ser ouvida quando o elevador subia novamente. Estes foram os primeiros sinais que o *layout* desta cena teria de ser repensado – alterações que serão discutidas em maior detalhe no capítulo 3.3. .

3.3. Integração da componente de Realidade Virtual no motor de jogo

Com a cena aparentemente preparada para ser explorada em VR, fez-se a integração do módulo de VR no Unity. Uma vez que a ESMAD cedeu gentilmente um dos seus Oculus Rift para que se pudesse completar este projeto, como mencionado no capítulo 1 desta monografia, o módulo que foi necessário integrar foi o da Oculus (Facebook, n.d.).

Para se poder trabalhar com o Oculus Rift no Unity, são necessários três elementos além do equipamento em si: o *software* da Oculus, `Integration Package` disponibilizado pela Oculus e `Android Build Support` no Unity, permitindo fazer *builds* para Android. O módulo de suporte para Android pode ser instalado para o Unity utilizando o Unity Hub, indo à secção "Installs" e clicando em "Add Modules" à nossa instalação de Unity. O *software* da Oculus pode ser adquirido na página de *setup* da Oculus[4], e basta seguir os passos do instalador, sendo notório que neste passo será necessário configurar também o dispositivo em si – neste caso, os Oculus Rift. Finalmente, o pacote de integração da Oculus poderá ser adquirido na *Asset Store* do

[4] Pode-se consultar o site de *setup* da Oculus no link https://www.oculus.com/setup/

Unity, a sua loja virtual, procurando por “Oculus integration”, adicionando aos nossos *Assets*, e acedendo a este depois na janela “Package Manager” dentro do nosso projeto do Unity. A versão mais recente deste pacote à escrita desta monografia era a versão 33.0, sendo que foi essa a utilizada na continuação deste projeto.

Com tudo instalado e corretamente configurado, é possível integrar os serviços e capacidades de VR presentes no módulo de VR da Oculus (OVR). A primeira mudança que foi implementada foi a alteração do “Mini first person controller” como avatar controlável para o “OVRPlayerController” disponibilizado no OVR. Este avatar é disponibilizado como um `prefab`, sendo que é preciso apenas arrastar os `prefabs` das mãos, disponibilizadas também no OVR, para ter um avatar que conseguimos controlar com os controladores e *headset* do Oculus Rift. Desativando o “Mini first person controller” e ativando o “OVRPlayerController” com as mãos adicionadas, podemos agora explorar o cenário em VR.

Há a ressalvar que, neste processo de experimentação em VR, se verificou que existe a possibilidade de os/as jogadores/as sentirem vertigem ao moverem-se no cenário virtual, uma vez que o movimento é linear, podendo causar uma sensação de desconforto. Uma maneira de combater esta sensação é mover o corpo como se estivesse a andar de facto, ou dar passos sem sair do lugar. Por causa disto, achou-se por bem não incluir os efeitos sonoros de passos, como discutido no capítulo, uma vez que se achou que seria mais propenso a atrapalhar os/as jogadores/as do que a ajudar na sua imersão, podendo ir contra a sua Presença Física e *Sensorimotor Engagement* (Lee, 2021).

Uma vez que os botões que se haviam sido desenvolvidos para interagir com o cenário dependiam da posição do rato, através de `raycasts`, e que com estas alterações se havia introduzido uma nova maneira de *input*, criou-se um novo objeto “Button”, que serviria como interface no jogo. Houveram várias tentativas como fazer a interação das mãos do avatar do OVR com os botões: o botão ter uma zona de `trigger` que detetasse as mãos, as mãos terem uma zona de `trigger` nas mãos que detetassem os botões, adicionar componentes de `rigidbody` nas mãos de forma a que fosse registado um impacto entre os dois elementos pela componente de física; infelizmente nenhuma das opções estava a parecer funcionar, parecendo que os objetos mãos e botões estivessem em *layers* diferentes, mas com o tempo a escassear para a conclusão deste projeto, decidiu-se avançar com uma solução que funcionasse em vez de gastar mais tempo a tentar perceber o porquê destas outras não funcionarem. A solução encontrada foi, em vez de se “clicar” nos botões, é possível “agarrá-los”,

adicionando uma componente `Grababble`, disponível no OVR, que faz com que com o gesto de fecho de punho se consiga agarrar em objetos. Utilizando esta técnica, era possível interagir com o objeto botão e, por isso, identificar quando o/a jogador/a teve essa interação, despoletando qualquer método que lhe fosse atribuído. Uma vez que já se havia programado as interações que seriam esperadas dos vários botões, bastou posicionar os vários novos botões na cena, e atribuir-lhes os métodos que a si correspondiam: o botão do elevador deverá primeiro desligar a energia e tudo o que isso implica, e só depois servir como método de ascender e descender o elevador; os botões do *Dropper* deveriam fazer cair uma caixa e possibilitar a resposta ao puzzle; o botão do gerador deveria restituir a energia e reativar a segunda caixa com bolas.

Com a possibilidade de interagir com os botões do cenário, recorreu-se a bibliotecas de som (MTG, 2021; Zapsplat, 2021) para se arranjar os sons base para a interação com os mesmos. Uma vez que temos a possibilidade de identificar quando o/a jogador/a agarra o botão e o larga (que é, nesta versão, o equivalente a clicar e tirar o dedo do botão), criaram-se dois eventos no FMod referentes a cada uma dessas ações: “Click On” e “Click Off”. Numa biblioteca de som (Zapsplat, 2021) arranjou-se o som de um clique de um rato de computador e dividiu-se esse som em dois, a ativação do mecanismo e o momento em que ele deixa de ser ativado. Aplicou-se um equalizador a estes sons, e utilizou-se o iZotope Trash 2 para se adicionar um pouco de *drive* ao som, de forma a tentar-se aumentar o som de um clique de rato para um clique de botão mais complexo. Associou-se cada um destes dois sons aos dois eventos do FMod, e fez-se com que cada um fosse acionado no início e final da interação com cada botão.

Com recurso a bibliotecas de som, encontraram-se o som de motores a trabalhar normalmente e com problemas. Aplicou-se um equalizador a ambos, e foram editados de forma que pudessem ser *looped* num evento do FMod. Criaram-se dois novos eventos, e associaram-se ao gerador para cada um dos seus estados: a funcionar e a não funcionar, quando a energia fosse abaixo.

Da mesma forma que se adicionou uma componente `Grababble` aos botões de forma que o/a jogador/a pudesse agarrá-los, o mesmo se fez às caixas com as bolas que haviam sido previamente preparadas. Agora, quando o/a jogador/a se aproximasse das caixas e as agarrasse, conseguiria pegar nelas e, esperava-se, que conseguisse perceber apenas pelo contexto sonoro quantas bolas existiam dentro da caixa. Contudo, surgiram dois novos problemas com a implementação dos princípios da física que impossibilitaram a continuação do protótipo desta forma. Em primeiro lugar, os receios explícitos no capítulo 3.1.2.2. confirmaram-se, e quando o/a jogador/a pegava numa

caixa, havia uma alta probabilidade de as bolas caírem fora da mesma, como podemos observar no Anexo 5. Adicionalmente, a solução de programaticamente mudar a posição das bolas caso a velocidade delas fosse maior que um certo patamar, como descrito também no capítulo 3.1.2.2., não resolvia o problema. Em segundo lugar, uma vez que tanto as caixas como o `OVRPlayerController` têm componentes de `Collider`, caso o/a jogador/a se aproximasse da caixa, existiria uma alta probabilidade de o sistema de física impor uma força ao avatar, atirando sem controlo do/da jogador/a, como podemos também observar no Anexo 5.

3.4. Separação e finalização dos puzzles

Estando-se a aproximar o prazo de entrega para este projeto, e uma vez que este era ainda um projeto embrionário que apresentava aspetos que impediam ainda a sua demonstração pública, decidiu-se reavaliar a forma como se apresentaria o projeto. Achou-se por bem instaurar uma separação dos vários puzzles, para tentar perceber onde estavam os problemas maiores, e como se poderiam resolver caso fosse possível.

Fazendo esta reavaliação, chegou-se à conclusão de que todos os problemas estavam concentrados no puzzle *Boxes*. O *Labyrinth* parecia funcional, apesar de se terem revelado problemas com as reflexões e oclusões que serão explorados mais à frente neste capítulo, e o *Dropper* estava completo também; apenas as caixas tinham problemas, devido aos erros de física, que o tornavam o *Boxes* não jogável. Tendo isso em conta, decidiu-se por bem abandoná-lo, deixando a referência ao trabalho desenvolvido nesta monografia, mas não investindo mais tempo no seu desenvolvimento. Assim sendo, tinha-se agora dois puzzles funcionais em cenas separadas.

Contudo, explorando-se com maior cuidado agora o puzzle *Labyrinth*, foi possível perceber que as reflexões não estavam a funcionar corretamente, e a oclusão não estava a funcionar de todo. Revendo a documentação da Google e do FMod relativamente ao Google Resonance, foi possível perceber que os `Reverb Probes` tinham sido mal implementados, uma vez que eles funcionam com o Google Resonance para o Unity, mas não com o Google Resonance para o FMod para o Unity, ou seja, era necessário utilizar os componentes da Google Resonance diretamente no Unity, sem a camada de abstração do FMod (FMOD, n.d.; Google, n.d.). A resolução deste problema foi, felizmente, simples: retirou-se à fonte sonora a componente `FMod Event Emitter` e adicionou-se `Resonance Audio Source`, passando de um evento do FMod para uma fonte embutida no Unity. Uma vez que já não existia a metáfora de tentar encontrar

o gerador, decidiu-se substituir também o som que a fonte sonora produzia para um dos sons que vem por defeito com o OVR, o `vocal1`. Para além disso, adicionou-se uma nova *layer* ao Unity chamada "Occlusion", e fez-se com que esta fosse a *layer* de oclusão do `Resonance Audio Listener` adicionado ao avatar, substituindo o `FMod Studio Listener` que lá existia, de forma a implementar corretamente a oclusão na cena (Google, n.d.). Uma vez que o Google Resonance não é capaz de detetar a profundidade da geometria nos cálculos da oclusão, foi necessária a criação de "paredes artificiais", também elas com a *layer* "Occlusion", de forma a simular a densidade das paredes. Neste processo, achou-se por bem também aumentar ao labirinto original, acabando então com o labirinto da figura 15.

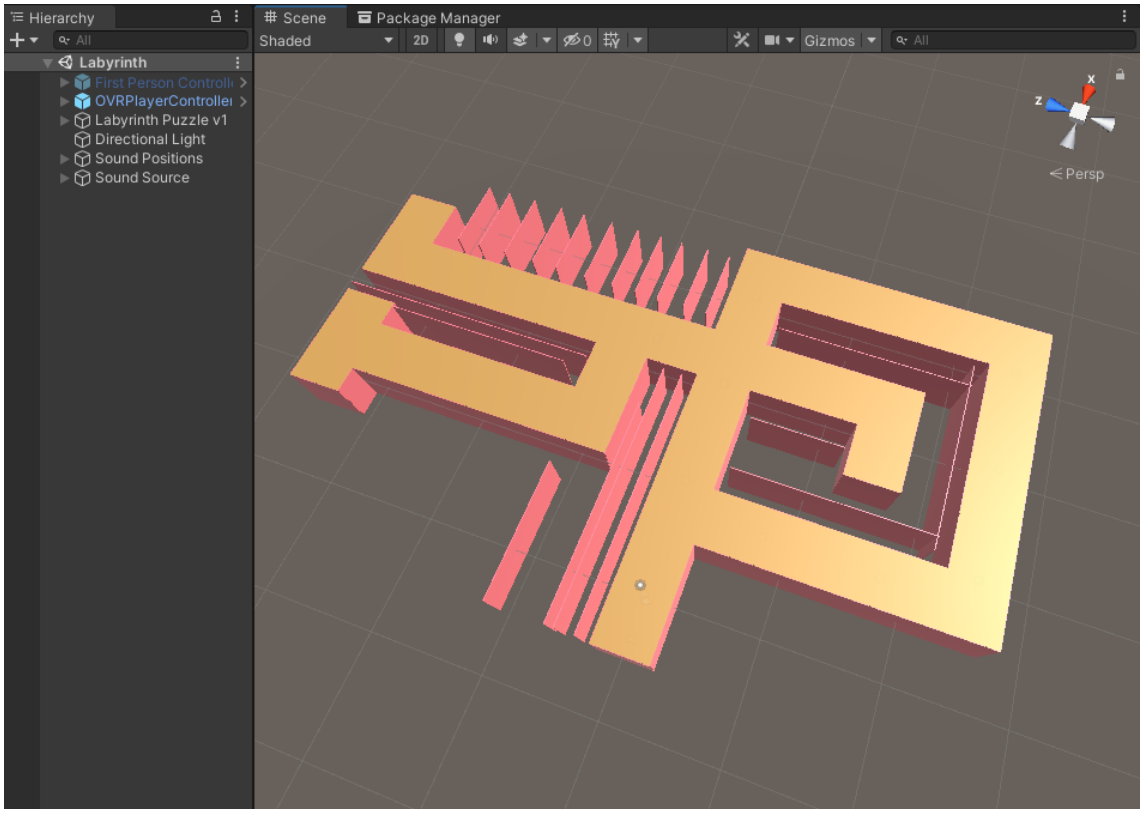


Figura 15 - Labirinto mais complexo e com paredes extra para oclusão

Uma vez que se estava a isolar este puzzle, achou-se por bem manter o conceito original de "procurar a fonte sonora num labirinto", destacando-o do *level design* criado. Por isso, decidiu-se que, quando o/a jogador/a encontrasse a fonte sonora e carregasse no botão sobre ela colocada, em vez de "reiniciar um gerador", a fonte sonora iria mover-se para um outro local do labirinto, e o/a jogador/a deveria reencontrar essa fonte um determinado número de vezes. Criou-se então um `array` de posições, através do posicionamento de vários objetos invisíveis na cena, e um método que faz com que, de

cada vez que o/a jogador/a carrega no botão da fonte sonora, ela desloca-se para um outro local desse mesmo `array`, diferente de onde se encontrava. Após um número de vezes desta repetição, o/a jogador/a seria transportado/a para o segundo nível, o *Dropper*.

No *Dropper*, não havendo agora a necessidade de movimentação no cenário, uma vez que estava estanque, decidiu-se retirar o movimento do `OVRPlayerController`. O/A jogador/a ainda se poderia mover, no espaço delimitado como sendo o seu espaço de jogo, mas não conseguiria controlar o movimento do avatar com os comandos, e redimensionou-se a plataforma do armazém, encurtando-a. Adicionou-se também um objeto semelhante a um altifalante na parede atrás do/da jogador/a, e associou-se a ele o os eventos de *feedback* sonoros deste puzzle. Aos eventos sonoros de *feedback*, adicionou-se no FMod um efeito de *pitch shift* e *distortion*, de forma a mais verosimilmente se assemelharem a saírem de um altifalante do tipo selecionado.

Com recurso a bibliotecas de som (MTG, 2021; Zapsplat, 2021), arranjaram-se vários efeitos sonoros de “coisas” a cair, tendo o cuidado de tentar que se diferenciassem sonoramente uns dos outros, e que se assemelhassem a objetos de diferentes pesos. Todos eles foram depois cortados, removendo as partes não interessantes de cada som, e foi-lhes aplicado um equalizador cortando os graves aos associados a objetos leves e os agudos associados a objetos pesados e um compressor. Depois, fez-se *layering* de vários sons de forma a criar novos, mais complexos, sendo que nos sons associados a objetos pesados adicionou-se também um som de um tambor, de forma a complementar a faixa dos graves, tentando vender mais a ideia de impacto (Viers, 2008). Finalmente, alterou-se ligeiramente o código da reprodução dos eventos de FMod no impacto das caixas, de forma a deixar aleatório, mas consistente, o som de cada tipo de caixa.

Capítulo IV - Playtesting

4.1. Sessões de Teste

Com o projeto numa fase passível de ser demonstrada ao público para testes, organizaram-se sessões de *playtest*, para se recolher *feedback* externo sobre o mesmo (Adams, 2013), tentando validar o trabalho desenvolvido até então. As sessões decorreram maioritariamente no Estúdio C da ESMAE, tendo uma outra sessão acontecido em casa do autor com pessoal externo à ESMAE, tendo em todas elas tido em atenção as recomendações da DGS[5] relativamente a aglomerados de pessoas.

Nestas sessões, começou-se por introduzir o projeto aos indivíduos de teste, fazendo uma pequena descrição da intenção do mesmo. Seguiu-se uma breve explicação sobre o funcionamento dos Oculus Rift, entrando-se em maior detalhe caso fosse a primeira experiência em VR do indivíduo, e uma explicação de quais os controlos disponíveis no jogo que iriam testar. Houve uma pequena explicação do que era esperado deles em cada um dos puzzles, e finalmente iniciou-se a experiência.

Durante o tempo de jogo de cada um dos indivíduos, o autor apontou notas sobre comentários e dificuldades que cada teve, que podem ser consultadas no Apêndice 1 – Notas das Sessões de Playtest

e, no final de cada sessão de jogo, pediu-se que se preenchesse um questionário, cujas perguntas e respostas de cada indivíduo poderão ser consultadas no Apêndice 2 – Questionário Completo das Sessões de PlaytestNo Anexo 6, podemos observar a sessão de teste do indivíduo 19. Devido a constrições de espaço e configuração dos Oculus Rift, teve de ser reiniciada quando se chegou ao puzzle do *Dropper*, mas, de resto, foi uma experiência semelhante à dos restantes indivíduos, e, portanto, considera-se representativa, para efeitos de análise de comportamento.

No final das sessões de teste, haviam sido recolhidos dados de 19 indivíduos, com idades entre os 19 e os 49 anos, de várias áreas profissionais, mas todos eles com experiência musical ou técnica sonora em alguma capacidade, verificando uma falha de dados relativos a indivíduos sem esse tipo de experiência. Para além disso, para 12 dos

[5] Estas recomendações devem-se ao estado de pandemia provocado pela COVID-19.

19 indivíduos (~63.3%) esta foi a primeira experiência de VR, o que poderá ter influenciado a sua perceção deste projeto como um todo.

4.2. Resultados dos testes

Observando os dados recolhidos com as notas das sessões, presentes no Apêndice 1 – Notas das Sessões de *Playtest*, e as respostas ao questionário, presentes no Apêndice 2 – Questionário Completo das Sessões de Playtest, pode-se então tirar algumas conclusões sobre o projeto.

É importante ressalvar o facto que, apesar de ter sido a primeira experiência em VR para cerca de 63.3% indivíduos, todos eles conseguiram chegar ao final da experiência, passando pelos dois puzzles disponíveis. Todos os indivíduos mostraram grande interesse pela experiência durante e após o seu término.

4.2.1. "Houve alguma interação que devia ter áudio, mas não teve? Se sim, qual/quais?"

Relativamente à presença de áudio nas interações, 5 dos 19 indivíduos disseram que todas tiveram os comportamentos esperados. Apenas 3 indivíduos sugeriram implementar-se som de passos, questão discutida no capítulo 3.3. desta monografia. Só 1 indivíduo comentou que os botões no *Labyrinth* deveriam ter *feedback* sonoro, mas uma vez que eles de facto já tinham esse *feedback*, especula-se que deveria ser mais notório.

4.2.2. "Quão fácil foi chegar à fonte sonora no primeiro nível?"

Relativamente a quão fácil foi chegar à fonte sonora no *Labyrinth*, sendo 1 "Muito fácil" e 5 "Muito complicado", temos uma média cerca 2.37, estando assim próximo do ponto intermédio perfeito, com alguma tendência para ser fácil.

4.2.3. "O que acha que @ ajudaria a chegar à fonte sonora?"

De acordo com a análise das respostas à pergunta, seria preciso melhorar ainda o espaço acústico, tentando melhorar a implementação das reflexões e oclusões de forma que se o seu comportamento se aproximasse ao do mundo real.

4.2.4. “Quão complicado foi perceber o peso das caixas no segundo nível?”

Relativamente ao quão complicado foi perceber o peso das caixas no *Dropper*, sendo 1 “Muito complicado” e 5 “Muito fácil”, apresenta-se uma média de aproximadamente 3.47. O grau de dificuldade do jogo e o grau de dificuldade da perceção sonora são métricas a ter em diferentes considerações, que nesta pergunta poderão ter ficado acidentalmente interligadas. Contudo, encontramo-nos próximos do ponto intermédio, com alguma tendência para ser fácil. Retira-se então a conclusão que nos num ponto ainda a ser mais trabalhado, tendo em conta que poderá ter havido entraves tecnológicos na facilidade da distinção sonora.

4.2.5. O que @ ajudaria a perceber melhor o peso das caixas?

De acordo com a análise das respostas à pergunta, percebe-se que seria necessário aumentar a intensidade sonora das caixas enquanto fontes sonoras, de forma a melhor se poder distinguir entre os vários tipos de som.

4.2.6. “Como descreveria a qualidade dos sons utilizados na experiência no geral?”

Relativamente à qualidade geral dos sons utilizados na experiência, sendo 1 “Pouco cuidados / Baixa Qualidade” e 5 “Bem Trabalhados / Alta Qualidade”, apresenta-se uma média de 3.47, não havendo nenhuma resposta abaixo de 3 nem acima de 4, estando assim próximos do ponto intermédio, mas tendo-os em boa consideração.

4.2.7. “O que melhoraria neste jogo?”

Quando inquiridos relativamente a alterações que efetuariam no projeto, ignorando questões visuais, foram apresentados maioritariamente os mesmos problemas acima descritos, relativamente à melhoria do espaço acústico do labirinto e aumento da intensidade sonora das caixas no *Dropper*.

Capítulo V - Conclusão

Com este projeto, espera-se ter demonstrado que é possível a criação de um jogo, especialmente um jogo VR, onde o foco principal é a experiência aural dos/das jogadores/as, apresentando o caso para um investimento e cuidado com a componente sonora no desenvolvimento de videojogos e experiências análogas. Pode-se encontrar as *builds* deste projeto no Anexo 7, e encoraja-se que, caso se tenha os recursos que possibilitem isso, se experimente para poderem observar os resultados atingidos. Podem-se também consultar o código fonte, projetos do Unity e do FMod deste projeto, no Anexo 8.

Após o conceito e *game design* da experiência, partiu-se inicialmente para o desenvolvimento do projeto, confiando que as ferramentas necessárias ao desenvolvimento estariam já num ponto estável para serem utilizadas. Contudo, como se pode observar com os problemas encontrados com o Steam Audio, e com a impossibilidade de integração do Google Resonance com o FMod para o tratamento de reflexões baseadas na geometria da cena, denota-se que estas ferramentas são ainda embrionárias, apresentando ainda grandes obstáculos ao seu uso em projetos com maior foco no áudio.

Tendo esses obstáculos em consideração, contudo, o *workflow* proporcionado por estas ferramentas, e que se aplicou neste projeto, permite-nos chegar ao resultado final esperado. São tarefas trabalhosas, mas não impossíveis de alcançar, especialmente se já se tiver passado pelos buracos de “tempo gasto” que nos mostram as limitações do sistema, e como as contornar. Espera-se assim, também, que este projeto sirva de aviso sobre o estado das ferramentas a projetos que o sigam, e como indicação aos desenvolvedores dessas mesmas ferramentas de pontos a melhorar.

Após todo este processo, fica apenas por refletir o que ficou ainda por fazer. Numa primeira fase, seria necessário reavaliar a experiência em si como um todo, de acordo com o *feedback* recolhido, de forma a se alcançar um ponto coerente, em vez de terminarmos com dois níveis estanques. Para além disso, seria também necessário corrigir os problemas identificados, nomeadamente um apuramento das reflexões e oclusões no *Labyrinth*, e uma melhoria dos elementos sonoros no *Dropper*.

Sobre o *Dropper*, há também trabalho a ser reconsiderado: talvez em vez da dicotomia “leve/pesado” se pudesse reformular o puzzle sem perder a sua essência, tentando ao invés disso adivinhar o material do qual os objetos que caem são feitos (por exemplo se

são de madeira, metal, pedra, vidro, etc.), apenas com recurso à audição. Dessa forma, talvez fosse possível alcançar um maior potencial no sentido de dar importância adicional às informações que o som pode comunicar.

Finalmente, e apesar de não ser esse o intuito deste projeto em específico, mas como ideia para uma eventual apresentação ao público geral, dever-se-ia ainda fazer uma melhoria gráfica a todo o projeto. Ter-se em foco uma das vertentes sensoriais do projeto não implica, para um lançamento público, desprezar-se as outras vertentes.

Tendo todas estas informações em mente, estamos confiantes em dizer que, havendo ainda espaço para melhorias, o projeto teve sucesso nos objetivos aos quais foi proposto.

Apresenta-se ainda a hipótese que não só em projetos VR o som tem potencialidade de ser elevado na sua importância e qualidade, mas também noutros estilos de jogos haverá potencialmente esse espaço. Uma hipótese a ser explorada em eventuais projetos que se sigam a este.

Pela criação de produtos de melhor qualidade, e de obras de arte mais cuidadas, no mundo dos videojogos.

Bibliografia

Adams, E. (2013). *Fundamentals of Game Design.* USA: Pearson Education.

Demigiant. (n.d.). *DoTween Documentation*. Retrieved from DoTween: http://dotween.demigiant.com/documentation.php

Facebook. (n.d.). *Oculus App Development in Unity*. Retrieved from Oculus For Developers: https://developer.oculus.com/documentation/unity/

Facebook. (n.d.). *Oculus Native Spatializer for Unity*. Retrieved from Oculus for Developers: https://developer.oculus.com/documentation/unity/audio-osp-unity/

FMOD. (n.d.). *FMod Documentation*. Retrieved from FMod: https://fmod.com/learn#documentation

Gajsek, D. (2021, March 16). *Unity vs Unreal Engine for XR Development: Which One Is Better?* Retrieved from Circuit Stream: https://circuitstream.com/blog/unity-vs-unreal/

Google. (n.d.). *Resonance Audio*. Retrieved from Resonance Audio: https://resonance-audio.github.io/resonance-audio/

Gould, R. (2018, March 29). *Let's Test: 3D Audio Spatialization Plugins*. Retrieved January 2021, from Designing Sound: https://designingsound.org/2018/03/29/lets-test-3d-audio-spatialization-plugins/

Greenwald, W. (2021, January 28). *The Best VR Headsets for 2021*. Retrieved from PC Mag: https://www.pcmag.com/picks/the-best-vr-headsets

Hauwert, R. (2021, March 23). *Unity 2020 LTS and Unity 2021.1 Tech Stream are now available*. Retrieved from Unity: https://blogs.unity3d.com/2021/03/23/unity-2020-lts-and-unity-2021-1-tech-stream-are-now-available/?utm_source=twitter&utm_medium=social&utm_campaign=engine_global_release_2021-03-23_2020-20211-social-combo-video

Huizinga, J. (2003). *Homo Ludens: um estudo sobre o elemento lúdico da cultura.* Lisboa: Edições 70.

Irish, D. (2005). *The Game Producer's Handbook.* Boston: Course Technology PTR.

Kuzminski, A. (2016, August 9). *VR Audio: Trends and Challenges of Pioneering a New Field*. Retrieved January 2021, from Desiging Sound: https://designingsound.org/2016/08/09/vr-audio-trends-and-challenges-of-pioneering-a-new-field/

Lee, H. (2021, January 8). *A Conceptual Model of Immersive Experience in Extended Reality.* doi:10.31234/osf.io/sefkh

MTG. (2021). *freesound*. Retrieved from Free Sound: https://freesound.org/

Nogueira, T. (2019, July 7). *Audio Middleware: Why would I want it in my game?* Retrieved from Gamasutra: https://www.gamasutra.com/blogs/TheoNogueira/20190719/346915/Audio_Middleware_Why_would_I_want_it_in_my_game.php

Nystrom, R. (2014, November 14). Game Programming Patterns. Genever Benning.

Roginska, A., & Geluso, P. (2018). *Immersive Sound: The Art and Science of Binaural and Multi-Channel Audio.* New York: Routledge.

Salselas, I., Penha, R., & Bermardes, G. (2020). Sound design inducing attention in the context of audiovisual. *Personal and Ubiquitous Computing*. Springer-Verlag London Ltd. doi:https://doi.org/10.1007/s00779-020-01386-3

Schell, J. (2015). *The Art of Game Design : A Book of Lenses.* Natick: Taylor & Francis Inc.

Schütze, S., & Irwin-Schütze, A. (2018). *New Realities in Audio: A Pratical Guide for VR, AR, MR and 360 Video.* Boca Raton, Florida: CRC Press.

Shindo, M., & Kohata, S. (2018, December 14). *A Wwise approach to acoustics in NieR:Automata- Part 1*. Retrieved January 2021, from Platinum Games: https://www.platinumgames.com/official-blog/article/10021

Shindo, M., & Kohata, S. (2019, January 18). *A Wwise approach to acoustics in NieR:Automata- Part 2*. Retrieved January 2021, from Platinum Games: https://www.platinumgames.com/official-blog/article/10027

Smuts, A. (2005). Are Video Games Art? *Contemporary Aesthetics, 3*. Retrieved from https://contempaesthetics.org/newvolume/pages/article.php?articleID=299

The Power of Video Game Engines: Every Game Developer's (Not-So-Secret) Weapon. (n.d.). Retrieved from Game Designing: https://www.gamedesigning.org/career/video-game-engines/

Unity. (2021, June 21). *Getting started with VR development in Unity*. Retrieved from Unity Documentation: https://docs.unity3d.com/Manual/VROverview.html

Unity. (2021). *Unity User Manual 2020.3 (LTS)*. Retrieved from Unity3D: https://docs.unity3d.com/Manual/index.html

Valve. (n.d.). *Steam Audio*. Retrieved from Steam Audio: https://valvesoftware.github.io/steam-audio/

Viers, R. (2008). *The Sound Effects Bible: How To Create and Record Hollywood Style Sound Effects.* Studio City: Michael Wiese Productions.

Wirtz, B. (2021, June 24). *Minimum Viable Product: Why You Need To Allow Gamers To Get A Taste of What You Offer*. Retrieved from Game Desining: https://www.gamedesigning.org/career/mvp/

Zapsplat. (2021). *Zapsplat*. Retrieved from Zapsplat: https://www.zapsplat.com/

Zotter, F., & Frank, M. (2019). *Ambisonics: A Pratical 3D Audio Theory for Recording, Studio Production, Sound Reinforcement, and Virtual Reality* (Vol. 19). Switzerland: Springer Open.

Beat Saber (PC Version) [Video Game]. (2018). Beat Games.

Crypt of the NecroDancer (PC Version) [Video Game]. (2015). Brace Yourself Games.

Regamey, K. (2015). Phonopath (Web Version) [Video Game]. Retrieved from https://www.phonopath.com/

Half-Life (PC Version) [Video Game]. (1998). Valve.

Half-Life: Alyx (PC Version) [Video Game]. (2020). Valve.

1-2-Switch (Nintendo Switch Version) [Video Game]. (2017). Nintendo.

The Witness (PC Version) [Video Game]. (2016). Thekla.

Myst (PC Version) [Video Game]. (1993). Cyan, Inc.

NieR:Automata (Playstation 4 Version) [Video Game]. 2017. Platinum Games.

Apêndices

Apêndice 1 – Notas das Sessões de *Playtest*

Indivíduo #1

Comentou que “paredes não parecem paredes. Não é fidedigno como espaço.”

No puzzle do Dropper, dificuldade em chegar aos botões. Não conseguiu distinguir entre opção certa ou errada no Dropper.

No fim, depois de ter preenchido o questionário, comentou que, apesar de ter respondido que “não saberia como melhorar”, lembrou-se do comentário que “A fonte sonora só muda no eixo horizontal, mas não no vertical”.

Indivíduo #2

Roda-se sem querer, com o joystick do controlador direito.

Dificuldade com botões no Dropper.

Indivíduo #3

Dificuldade em pegar nos botões.

Indivíduo #4

Sem comentários.

Indivíduo #5

Comentou “porquê estas cores?”.

Indivíduo #6

Problema a fazer o movimento de agarrar. Usa mais o gatilho que os botões laterais.

Queixou-se que oclusão tinha falhas nalgumas paredes.

Nota: Indivíduos 1 a 6

Não conseguiam concordar uns com os outros sobre se havia feedback sonoro a indicar se a escolha era errada no Dropper.

Indivíduo #7

Dificuldade em chegar aos botões no Dropper.

Indivíduo #8

Sem comentários.

Indivíduo #9

Sugeriu aumentar volume das caixas no Dropper.

Indivíduo #10

Sem comentários.

Indivíduo #11

Dificuldade em chegar aos botões no Dropper.

Indivíduos #12 a #15

Sem comentários.

Indivíduo #16

Problema de localização da fonte sonora no labirinto.

“O que é para fazer?” no Dropper.

Queixou-se que as pessoas a falar na sala, fora do jogo, incomodaram e “estragaram” a experiência.

Indivíduo #17

Aumentar reverberação do labirinto.

No Dropper, aumentar a intensidade sonora das caixas, especialmente nas pesadas.

Indivíduo #18

Comentou numa das vezes no labirinto, “Parece que o som vem de cima”.

Indivíduo #19

Problemas com o fio do VR → várias piruetas durante a experiência

Não conseguiu chegar ao botão do “pesado” no Dropper. Foi necessário reiniciar a experiência.

Comentou que se deveria aumentar a intensidade sonora das caixas no Dropper

Apêndice 2 – Questionário Completo das Sessões de Playtest

Indivíduo	Idade	Profissão	Tem experiência musical ou técnica sonora?	Esta foi a sua 1ª experiência com VR?	Se não, que outras experiências teve?	Conseguiu chegar ao final do jogo?
1	26	Designer de Luz e de Som	Sim, sou estudante de técnic@ de som	Sim	-	Sim
2	21	Estudante	Sim, sou estudante de técnic@ de som	Sim	-	Sim
3	24	ESTUDANTE LUZ E SOM	Sim, sou estudante de técnic@ de som	Sim	-	Sim
4	20	Estudante	Sim, sou estudante de técnic@ de som	Sim	-	Sim
5	26	Estudante	Sim, sou estudante de técnic@ de som	Sim	-	Sim
6	47	Estudante Luz e Som (Teatro)	Sim, sou estudante de técnic@ de som	Sim	-	Sim
7	45	Designer de Luz	Sim, sou músic@ amador, Sim, sou técnic@ de som amador	Não	Experiências em exposições	Sim
8	21	Técnico de Luz	Sim, sou estudante de técnic@ de som	Não	Jogos	Sim
9	36	Engenheiro de som	Sim, sou técnic@ de som profissional	Sim	-	Sim
10	41	Desempregado	Sim, sou músic@ amador, Sim, sou técnic@ de som amador	Sim	-	Sim
11	21	Estudante	Sim, sou estudante de técnic@ de som	Sim	-	Sim
12	19	Estudante	Sim, sou estudante de técnic@ de som, Sim, sou técnic@ de som profissional	Sim	Vídeo 360º	Sim
13	21	Estudante	Sim, sou estudante de técnic@ de som	Sim	-	Sim
14	19	Estudante	Sim, sou músic@ amador, Sim, sou estudante de técnic@ de som	Sim	-	Sim
15	49	Estudante	Sim, sou técnic@ de som profissional	Não	Instalações interativas com realidade virtual	Sim
16	42	Professora	Sim, sou músic@ amador, Sim, sou estudante de música	Não	Instalações	Sim
17	44	Professor	Sim, sou técnic@ de som profissional	Não	Jogos VR	Sim
18	42	Professor/Performer	Sim, sou músic@ profissional	Não	Aplicações VR	Sim
19	24	Professora	Sim, sou músic@ amador	Não	Jogos	Sim

Indivíduo	Houve alguma interação que devia ter áudio, mas não teve? Se sim, qual/quais?	Quão fácil foi chegar à fonte sonora no primeiro nível? (1- Muito fácil / 5- Muito difícil)	O que acha que @ ajudaria a chegar à fonte sonora?	Alteraria o número de vezes que tem de encontrar a fonte sonora? Se sim, para quantas vezes?
1	Sim, os passos	3	O espaço acústico não parece estar representado de forma fidedigna	Não
2	Talvez	2	A consistencia das paredes serem consideradas no som, a sua textura/grossura/distância, de forma a ser mais percetivel perceber de onde o som vem	Não no número de vezes, mas mudaria a música algumas vezes, para haver variedade da forma e do som que procuramos, também para não se tornar muito secante, principalmente na última parte onde o som vem sempre do mesmo sitio
3	Não	2	A parte sonora sendo em 360 ajuda muito a achar a fonte sonora	Não
4	Passos. 'Plim' quando aperto o botão no labirinto.	3	Resposta fácil, setas	Sim. 5 vezes
5	Todas tiveram interação áudio	2	melhor espacialização, reverberaçao e absorsçao sonora por parte das paredes	no estado actuaç da experiencia, sim, reduzia para umas 5 vezes talvez no entanto se o audio for sempre diferente e o ambiente 3d for um pouco mais imersivo (iluminaçao/ texturas)
6	Não	1	Melhor simulação do espaço acústico, barreiras sonoras (paredes) demasiado permeáveis	Talvez, se mudasse seria para menos 2 ou 3
7	Não	2	qualidade dos auscultadores	um pouco menos (-2)
8	Não	2	O facto da mesma estar super realista	Não
9	Não, está óptimo	2	Está óptimo.	6
10	Não	1	maior realismo no efeito de isolamento acustico das paredes	Não
11	Não	3	o aumento do nivel de volume	Não
12	Não	2	Surgimento de setas em caso de dificuldade	10
13	Acredito que as interações sonoras foram suficientes para o desenvolvimento do projeto.	5	Utilizar uma qualidade sonora mais envolvente.	Talvez menos vezes em ambos os jogos em troca de um jogo extra.
14	Não	2	Se ao chegar perto do caminho a seguir o nivel sonoro aumentasse	Não
15	Os passos podiam ter algum som associado, apesar de ténue.	1	Não tenho nada a declarar, foi bastante simples chegar lá	Pareceu-me equilibrado o número de vezes selecionado
16	Sim, o in´cio de cada jogo deveria ter um objetivo claro	4	Saber que seria esse o meu objetivo	sim, 5 vezes no máximo
17	Som dos passos	3	Maior realismo (mais atenuação com a distância, mais filtragem), relação da reverberação vs som direto	-
18	Não	2	menor "transparência" das paredes	Menos vezes
19	Não	3	O uso de stereo	Não

Indivíduo	Quão complicado foi perceber o peso das caixas no segundo nível? (1- Muito complicado / 5- Muito fácil)	O que @ ajudaria a perceber melhor o peso das caixas?	Alteraria o número de caixas com que tem que interagir? Se sim, para quantas vezes?	Como descreveria a qualidade dos sons utilizados na experiência no geral? (1- Pouco cuidados / Baixa Qualidade ; 5- Bem Trabalhados / Alta Qualidade)
1	4	a relação audiovisual das caixas, tendo em conta o material aparente das mesmas e o som que estas produzem, poderia ajudar, assim como o tamanho das mesmas, ou seja, a relação visual deveria corresponder à relação auditiva, isto porque a refêrencia visual é anulada apenas a meio do jogo	Não	3
2	4	O som ser realmente diferente de caixa para caixa, nas caixas pesadas senti que o som era igual assim como nas caixas mais leves. Ou seja, não no sentido de ajudar, mas sim para se tornar um desafio maior e talvez mais interessante ao ouvinte tentar perceber qual é qual	Não	3
3	4	A questão de não haver graves	Não	4
4	5	Existirem sons diferentes para as caixas. Após ter identificado os dois sons, e perceber a repetição, foi fácil.	Não	4
5	3	um pouco mais de presença (volume) ou caixas maiores, maior reverberação	Não	3
6	2	Mais grave nas caixas pesadas	Talvez, mais uns 4 a 6	3
7	2	menos reverb, som mais proximo. O som não pareceu realista	não. qb.	4
8	3	O som não é muito audive	Não	4
9	4	Perfeito, mas o som talvez mais alto para sentir melhor a reverberação.	6	4
10	5	nada	Não	3
11	3	perceber que estavam a uma distancia perto ou nao	Não	4
12	3	Talvez um som mais alto	10	4
13	4	Talvez focar mais na densidade do som do que propriamente na sua "qualidade", isto é, não definir o peso da caixa pela quantidade de vezes é que ressalta	Não	4
14	3	Aumentar os graves nas cixas pesadas	Não	3
15	4	Aumentar um pouco o nível do sinal ou reduzir um pouco a rverberação do mesmo.	Não, está equilibrado	4
16	5	Em primeiro lugar que não fosse possível ouvir as restantes pessoas na caixa, em segundo lugar saber que era esse o meu objetivo	sim, porque facilemnte se perde o objetivo do jogo. 6 vezes	3
17	2	Som mais realista da fonte sonora (material), e mais alto!	-	3
18	3	maior pressão acústica nos sons do impacto	Não	3
19	3	Os sons serem mais altos	Não	3

Indivíduo	Há algum som que teria alterado? Se sim, qual/quais?	O que melhoraria neste jogo?
1	O som das caixas	Não sei
2	Já respondi mais ou menos em cima, mas sim tentaria arranjar uma forma de ter uma maior variedade de som, tanto como no primeiro jogo, com músicas diferentes, e como no segundo jogo, com o som das caixas ser mais diferente	O que já respondi em cima, a variedade dos sons e no primeiro jogo a construção sonora das paredes
3	O som das caixas	Nada
4	Som abafado da moça a cantar quando ela está do outro lado da parede; experienciei que ela estaria à minha beira, mas tinha de dar a volta ao quarteirão para chegar à fonte.	Primeiro, Parabéns! Foi a minha primeira experiência VR. Foi divertido. Melhoraria a acústica do labirinto: em termos de profundidade e direção está bem, mas a cena de passar as paredes.
5	Sim, na verdade. Todos... No primeiro nivel, como ja referi alguma variedade na fonte sonora. No segundo nivel talvez nao alterar mas aumentar o volume reverberaçao	Texturas, iluminaçao mas imersiva. Melhorar o sistema de absorçao do som (paredes/obstaculos)
6	O som das caixas, pouco realista	As alterações acima
7	o som das caixas	nteração com os botões. Facilidadede acesso aos botoes no jogo das caixas
8	O das caixas	Nada
9	Tudo adequado.	No nível das caixas como já disse teria aumento o volume das caixas a cair.
10	o som da caixa leve nao parece o som de uma caixa	a precisão da interação por vezes interfere com a qualidade da experiência
11	nao	os sons poderiao estar mais atos
12	As vocais do primeiro nivel	Apenas as vocais do primeiro nivel
13	Talvez nas caixas, conseguir "enconcorpar" mais o som, para a distinção ser mais óbvia, não que não tenha sido	No primeiro jogo, seria interessante ter sons diferentes e não só a monotonia da rapariga a cantar, pois com a habituação fica mais fácil de encontrar. Se as paredes tivessem "mais densidade" seria mais uma maneira de adicionar dificuldade ao nível.
14	O som das caixas	A qualidade do som das caixas
15	Manteria a cantora, mas a cantar melodias diferentes.	A melodia da cantora.
16	Sim, o som das caixas	Acrescentaria instruções ao início de cada etapa; pontuação; e elementos onde fosse possível ter um objetivo por cada vez que se acerta. Por exemplo, o objetivo geral do jogo é fazer um puzzle , portanto por cada vez que acertássemos na fonte sonora completaríamos o puzzle.
17	O som das caixas, música escolhida com informação espectral mais diferenciada e dinâmica	Nível 1: - Instruções claras no início do jogo - reverberação do espaço deveria ter mais preponderância - movimento a simular passos (demasiado linear e suave) - interação com o botão estranha Nível 2: - Instruções claras no início do jogo - Som das caixas deveria ser bastante mais alto e mais realista, em função dos materiais - interação com os botões estranha, acrescido de os botões estarem demasiado afastados - botão "drop" dispensável, poderia ser automático após a resposta dada - não deveria ser necessário responder certo para continuar, até porque se não é leve, é pesada. A resposta dada deveria ser contabilizada e um sinal sonoro e/ou visual denotaria estar certo ou errado, continuando com nova caixa de seguida.
18	podia ser interessante ter músicas diferentes em cada vez que se encontra a fonte sonora. Os sons das caixas talvez pudessem ser mais trabalhados	interligação entre as partes
19	Não	O som das caixas ser mais alto

ESCOLA
SUPERIOR
DE MÚSICA
E ARTES
DO ESPETÁCULO
POLITÉCNICO
DO PORTO

P.PORTO

M

MESTRADO

ARTES E TECNOLOGIAS DO SOM

[Nome]: Jogo de Realidade Virtual com Foco na Experiênica Sono

José Ferreira Machado