Universidade Federal de Pernambuco

Centro de Informática

Graduação em Ciência da Computação

Ferramenta para Criação de Trilhas Sonoras Adaptativas e Personalizadas em Jogos Digitais

Daniel Filgueira Bezerra (dfb2@cin.ufpe.br)

Trabalho de Graduação

Recife, Agosto de 2021

Universidade Federal de Pernambuco

Centro de Informática

Graduação em Ciência da Computação

Ferramenta para Criação de Trilhas Sonoras Adaptativas e Personalizadas em Jogos Digitais

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Geber Ramalho

Recife, Agosto de 2021

Aos meus pais, Selma e Carlos.
Ao meu irmão, Matheus.
À minha tia, Norma.
Aos meus avós, Iara, Helena e Dionísio (in memoriam).
À minha namorada, Sol.

Agradecimentos

À toda comunidade independente de desenvolvimento de jogos digitais de Pernambuco e do Brasil, onde sempre encontrei acolhimento, inspiração, e suporte. Muitas vezes buscamos nossos sonhos nas tangentes de nossos objetivos. Sonhamos e trabalhamos para que o Nordeste e o Brasil possua títulos importantes para a indústria, e experiências marcantes para nossos jogadores.

À minha família pela inspiração e apoio desde criança, construindo uma formação que valoriza o estudo, esforço, simplicidade e honestidade, antes mesmo de eu saber a importância de tudo isso. Aos meus pais, Selma e Carlos; ao meu irmão, Matheus; à minha tia, Norma; e aos meus avós, Iara, Dionísio e Helena, meu muito obrigado.

À minha namorada, Sorely, que comigo divide lutas e conquistas, toda estrada é mais agradável com o nosso companheirismo. O futuro é sempre promissor quando se está em volta de boas pessoas.

Aos meus amigos e colegas do CIn-UFPE, que compartilharam essa jornada juntos comigo, ao longo de dias e madrugadas. Sem a comunidade e a irmandade criada em épocas de luta, tudo seria muito mais difícil.

Aos meus companheiros de mesa de RPG. Foram as primeiras cobaias de trilhas sonoras adaptativas pelo Spotify, mesmo que em um formato mais analógico, e associaram a esse projeto memórias leves e preciosas.

Aos meus orientadores e referências. Em especial Geber Ramalho, Giordano Ribeiro e Filipe Calegario, que guiaram uma mente academicamente inexperiente, e lapidaram juntos comigo o projeto como ele está hoje.

A todos os funcionários do CIn-UFPE, que juntos compõem um centro de aprendizado e inovação que vira casa de tantas pessoas, em suas passadas por lá. Em especial a Augusto Sampaio, que me apresentou o CIn e permitiu a ele me cativar.

Por fim, a todas as pessoas que passaram por perdas e dificuldades durante a pandemia, que foram resilientes para continuar com suas conquistas e adaptações, em um período emocionalmente crítico. Estamos todos nisso juntos.

Resumo

Trilha sonora adaptativa para jogos digitais é um campo de estudo em constante evolução: jogos que modificam sua música dependendo do que está acontecendo com e ao redor do jogador são cada vez mais frequentes. Enquanto isso, desenvolvedores de jogos independentes lutam para compor trilhas sonoras, e ferramentas de trilhas sonoras adaptativas não são acessíveis para eles, principalmente por causa de sua complexidade. Isso força que a maioria deles procure por faixas de áudio online, com licenças permissivas para fazer uma colagem e formar suas próprias trilhas. Além disso, existe um hábito crescente, por parte de jogadores, de silenciar a música do jogo e ouvir suas próprias músicas. Essa monografia introduz *MySoundtrack,* um *plugin* para o motor de desenvolvimento de jogos Unity, que permite o jogador jogar escutando suas próprias músicas no Spotify, mas escolhidas de acordo tanto com o gosto musical do jogador quanto com as emoções projetadas de cada momento do jogo. Nós revisamos abordagens existentes de criação de trilhas sonoras adaptativas, explicamos como o protótipo de MySoundtrack funciona e suas decisões de projeto, e discutimos planos futuros para a ferramenta. A validação até agora evidencia interesse e curiosidade dos desenvolvedores e jogadores, indicando a relevância da ferramenta proposta.

Palavras-chave: adaptive music, video game music, mood playlists, soundtrack, personalized music.

Abstract

Adaptive soundtrack design for video games is an ever-evolving field of study: games that modify its music depending on what is happening around and with the player. Meanwhile, independent game developers struggle with composition of soundtracks, and adaptive soundtrack tools are not accessible to them, mostly because of their complexity. This forces most of them to collage soundtracks with permissive licenses they find online. Besides, there is a growing habit of players to mute the game's original soundtrack to listen to their own songs. This paper introduces MySoundtrack, an asset for Unity that allows the player to keep playing while listening to Spotify songs, chosen according both to the player musical preferences and to the intended emotions on each moment in the game. We review existing approaches on adaptive soundtracks, explain how MySoundtrack's prototype works and its design choices, and discuss future plans for the tool. Validation so far indicates interest and curiosity by game developers and players, indicating the relevance of the proposed plug-in.

Keywords: adaptive music, video game music, mood playlists, soundtrack, personalized music.

Lista de tabelas

Tabela 1: Detalhamento dos valores escolhidos para cada Vibe. 23

Lista de figuras

Figura 1 - Exemplo de transição de trilha sonora, de uma música de combate para uma de exploração, utilizando-se uma abordagem de re-sequenciamento horizontal. 16

Figura 2 - Exemplo de uma trilha sonora, utilizando-se uma abordagem de verticalização em camadas. 16

Figura 3 - Modelo circumplexo de afeto 22

Figura 4 - Custom Inspector desenvolvido na Unity 24

Figura 5 - possibilidade de customização do sentimento associado à área, caso o desenvolvedor não se satisfaça com as vibes pré-definidas 24

Figura 6 - Exemplo de uma composição de nível, com áreas associadas a sentimentos específicos. 25

Figura 7 - Arquitetura básica de MySoundtrack 26

Figura 8 - Uso de BackupPlaylists 27

Figura 9 - Lógica de decisão para seleção de músicas da conta do jogador. 27

Figura 10 - Resultados de questionário sobre hábitos de uso do Spotify 28

Figura 11 - Média das pontuações obtidas no questionário SUS e distribuição das pontuações de cada participante. 36

Figura 12 - Distribuição das pontuações do questionário SUS aplicado com desenvolvedores de jogos, para cada item. 37

Lista de Algoritmos

Algoritmo 1 - Algoritmo de Ranqueamento 29

Sumário

1 Introdução **11**
1.1 Contexto e motivação 11
1.2 Objetivos gerais 11
1.3 Objetivos específicos 12
1.4 Organização do trabalho 12

2. Caracterização do Problema **13**
2.1. Complexidade de composição de trilhas em times indies 13
2.2. Hábito do jogador de silenciar o áudio do jogo 14

3. Estado da Arte **15**
3.1 Abordagens 15
3.1.1 Métodos baseados em reprodução de faixas 15
3.1.2 Geração Procedural de Trilha Sonora 17
3.2 Avaliação sobre as abordagens 18

4. Metodologia e Implementação **20**
4.1 A nossa ferramenta e sua concepção 20
4.2 Classificando a emoção geral de uma música 21
4.3 Design e lógica básica de MySoundtrack 23
4.3.1 Design 23
4.3.2 Lógica 25
4.4 Algoritmo de Ranqueamento 29

5. Validação e Experimentos **31**
5.1 Prova de Conceito 31
5.1.1. Objetivo 31
5.1.2. Método 31
5.1.3. Resultados 32
5.1.4. Discussão 33
5.2 Validação da usabilidade da ferramenta 33
5.2.1. Objetivo 33
5.2.2. Método 34
5.2.3. Resultados 35
5.2.4. Discussão 36
5.3 Riscos e Limitações 37

6. Conclusão e Trabalhos Futuros **39**
6.1 Conclusão 39
6.2 Trabalhos Futuros 41
6.2.1 Escolha de músicas sem royalties para criadores de conteúdo 41
6.2.2 Criar uma playlist com músicas tocadas enquanto jogou 41
6.2.3 MySoundtrack para outras plataformas 42
6.2.4 Filtros de áudio 42

7. Referências **43**

1 Introdução

1.1 Contexto e motivação

Nos últimos anos, o desenvolvimento independente de jogos (daqui em diante referenciado como *indie*), cresceu e recebeu muita atenção. Porém, ainda é uma atividade lenta e custosa, parcialmente por causa da diversidade de especialidades necessárias a um desenvolvedor indie, para que seja capaz de criar um jogo inteiro sozinho ou em um pequeno grupo de pessoas, e uma das especialidades mais raras nestes grupos é a de design de som. Muitas vezes, não há uma pessoa na equipe com a habilidade de compor e projetar trilhas sonoras, e o que acaba por acontecer é os desenvolvedores buscarem trilhas gratuitas e de licenças permissivas para usarem em seus projetos.

Ainda por cima, a música em jogos digitais pode ser não linear, reagindo a qualquer conjunto arbitrário de ações que o jogador possa fazer ao decorrer da jogatina, o que é muito importante para reforçar a emoção de cada momento e aumentar a imersão. Essa estratégia, comumente chamada de música adaptativa ou música interativa, é uma prática comum entre grandes empresas de jogos. Porém, é mais um nível de complexidade no desenvolvimento, separando ainda mais times pequenos de projetos de alto orçamento.

Enquanto isso, existe um hábito crescente de se jogar silenciando a música do jogo, e ouvindo suas próprias músicas, como demonstrado em McGowan (2011). Segundo o estudo, a principal razão para isso é os jogadores acharem a música muito repetitiva ou desinteressante, seguida por os jogadores acharem que a trilha sonora não reflete os sentimentos da jogatina. Tal hábito pode prevenir que o design musical da obra tenha o impacto desejado pelos criadores dela, já que o jogador pode estar escutando músicas que não tenham qualquer relação emocional com o momento vivido na narrativa.

1.2 Objetivos gerais

Este trabalho tem como objetivo propor uma alternativa à implementação de músicas adaptativas em jogos digitais, que não envolva composição de trilhas

sonoras ou qualquer tecnicidade envolvida no processo. O desenvolvedor será capaz de associar *vibes* musicais a espaços geográficos ou momentos no jogo. Quando o jogador estiver naquela área ou momento, a nossa ferramenta, chamada MySoundtrack, irá buscar por músicas no seu Spotify que melhor combinam com a vibe definida, respeitando as preferências musicais do jogador.

1.3 Objetivos específicos

- Desenvolver uma ferramenta para servir de alternativa à implementação de trilhas sonoras adaptativas.
- Desenvolver uma ferramenta que permita que o jogador não perca a experiência emocional do jogo ao ouvir suas próprias músicas enquanto joga.
- A ferramenta deve ser fácil de ser utilizada, e deve conseguir escolher músicas do jogador que bem representam a vibe desejada.

1.4 Organização do trabalho

- Capítulo 2: Explana e detalha as duas premissas em que se apoia o projeto, explicando os problemas e como eles afetam a indústria indie e a experiência do jogador.
- Capítulo 3: Introduz os conceitos mais usados no desenvolvimento de trilhas sonoras adaptativas e exemplifica seus usos na indústria dos jogos.
- Capítulo 4: Apresenta a solução proposta, explicando como ela pode resolver os problemas citados no capítulo 2, e detalhando seu design e implementação.
- Capítulo 5: Apresenta e discute os resultados obtidos ao longo das fases de validação (Prova de Conceito, Validação dos Desenvolvedores, Validação dos Jogadores). Também pondera sobre os riscos e limitações de MySoundtrack.
- Capítulo 6: Conclui o trabalho e pondera perspectivas de trabalhos futuros.

2. Caracterização do Problema

Este capítulo busca apresentar os principais pontos a serem atacados pelo projeto: a dificuldade de projetar trilhas sonoras, quando trabalhado em times sem uma pessoa com especialidade em música (maioria dos times indies), e o hábito de jogadores silenciarem a música do jogo e ouvirem suas próprias.

2.1. Complexidade de composição de trilhas em times indies

Quando se trabalha em um jogo sozinho, ou em um pequeno grupo de pessoas, música e som é uma da mais rara especialidade encontrada na equipe. Isso muitas vezes leva ao uso de trilhas sonoras gratuitas encontradas online, o que pode resultar em um jogo com pouca identidade musical, mesmo após um grande trabalho de pesquisa e curadoria de boas músicas gratuitas e com licenças permissivas que combinem bem umas com a outras.

Segundo De Salas *et al.* (2016), Steinke *et al.* (2016) e Borg *et al.* (2020) a área de áudio é uma das com menos representantes em *Game Jams*, eventos onde pessoas se reúnem para fazer um jogo em um determinado período de tempo, como um *Hackathon.* Nos trabalhos, compara-se a frequência dos participantes da área de áudio principalmente com as áreas de programação, arte (2d e 3d) e design, e segundo eles: na *TasJam: Voices* do ano de 2015, apenas cerca de 10% dos participantes alegaram ter alguma habilidade na área de áudio, e nas *Global Game Jam*s, apenas 12% . Essa escassez eventualmente leva a organização do evento a solicitar às pessoas de áudio que atuem em mais de uma equipe simultaneamente. O cenário de *Game Jams* é uma representação do cenário de desenvolvimento independente, já que a maioria de seus participantes são pessoas com menos experiência, como também demonstrado nesses trabalhos. Portanto, é latente a demanda desta comunidade por instrumentos que facilitem o processo de composição de áudio, ou que forneçam alternativas para ele.

Outra opção para compor músicas seria usar as ferramentas de trilhas sonoras adaptativas, que frequentemente são usadas em grandes empresas. Porém, elas muitas vezes não são disponíveis para todos, e quando são, como a

barelyMusician (Gungormusler *et al.* 2015), ainda requerem a presença de alguém com especialização em música, já que foram projetadas para serem usadas por designers de som ou funções afins. A maior parte dessas ferramentas foi feita para agilizar a manipulação da variabilidade da trilha, por parte dos *sound designers*, e não para permitir que pessoas sem conhecimento musical consigam criar as suas próprias trilhas sonoras. Portanto, perpetua-se a necessidade da rara presença de pessoas especializadas em composição e design de som.

2.2. Hábito do jogador de silenciar o áudio do jogo

Atualmente, existe o crescente hábito de se escutar suas próprias músicas enquanto joga, e normalmente isso envolve silenciar a trilha sonora do jogo. *Consoles*, como o *Playstation 4*, possuem integrações com serviços como o Spotify, e até seu próprio serviço de *streaming* de música, como o *PlaystationMusic,* para facilitar aos jogadores escutarem suas próprias músicas ao jogar.

Esse hábito pode fazer com que o grande esforço de composição e projeto de trilha sonora dos jogos possa ser contraproducente, especialmente em times menores, com recursos mais limitados. Adicionalmente, ele pode prejudicar a experiência do jogo, já que muito provavelmente ocorrerão dissonâncias emocionais entre o que o jogador está ouvindo e o momento que ele está vivendo no jogo.

Essa cultura vai além de apenas jogar, mas também está presente na criação de conteúdo online: *streamers* de jogos(pessoas que se gravam enquanto jogam), normalmente buscam incrementar a interação com o público permitindo-o controlar que música está tocando na *stream*.

Outro caso cultural desse hábito pode ser visto nos *bots* do Discord, um aplicativo de comunicação caracteristicamente voltado para o público mais *nerd* e *gamer.* Alguns de seus bots, como o *Rythm*, podem ser adicionados às chamadas de grupo para ficarem tocando música no plano de fundo, enquanto aquele grupo conversa ou joga. As músicas ou playlists a serem tocadas através de *bots* como o *Rythm* são especificadas pelos participantes da chamada, analogamente a uma *jukebox*.

Casos como esses evidenciam motivações também sociais para o hábito, e normalizam e fortalecem ainda mais o crescimento dessa cultura..

3. Estado da Arte

Este capítulo tem como objetivo apresentar as principais tendências da indústria no que tange trilhas sonoras adaptativas.

Nos anos recentes, se testemunha um crescimento na pesquisa de trilhas sonoras adaptativas, em busca de melhorar e refinar a experiência do jogador, como aponta Young (2012). O objetivo é, na maioria das vezes, fazer com que o tom emocional da trilha sonora se encaixe no sentimento do de cada momento vivido pelo jogador, é isso que a faz adaptativa. Há duas categorias de métodos para compor trilhas sonoras adaptativas: os métodos baseados em faixas já existentes de música, ou *playback-based*, e a geração procedural de trilhas sonoras.

3.1 Abordagens

3.1.1 Métodos baseados em reprodução de faixas

O método se baseia na reprodução e modificação de peças de músicas pré-existentes. Usa-se um arsenal de músicas já compostas, que são tocadas em modelos diferentes, dependendo da abordagem utilizada. As duas mais famosas, como mencionadas em (Kähärä, 2018), são: re-sequenciamento horizontal, quando as faixas são tocadas em sequência arbitrária, e verticalização em camadas, quando utilizam-se camadas de áudio separadas, sobrepostas umas sobre as outras.

O re-sequenciamento horizontal se concentra na transição entre faixas completas de músicas, como demonstra a Figura 1, que demonstra um exemplo de transição de uma música de combate para uma de exploração, utilizando-se essa abordagem. No re-sequenciamento horizontal, as faixas possuem informações sobre batida e tempo, possibilitando escolher os melhores momentos para transitar entre a atual e a próxima, mantendo a suavidade da trilha. As faixas pré-compostas dessa abordagem normalmente são músicas independentes, podendo contar percussão, harmonia e melodia em uma mesma faixa.

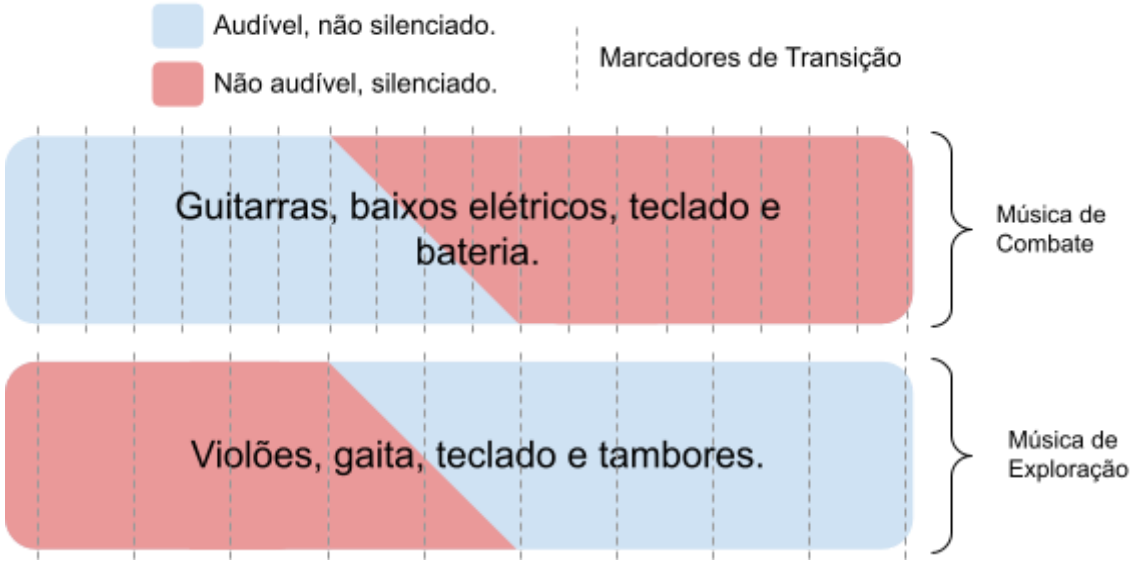


Figura 1: Exemplo de transição de trilha sonora, de uma música de combate para uma de exploração, utilizando-se uma abordagem de re-sequenciamento horizontal.

Por sua vez, a abordagem de verticalização em camadas utiliza múltiplas faixas pré-compostas, tocadas em conjunto. Cada faixa, portanto, pode representar apenas uma fração do que se está tocando ao todo, como por exemplo, haver uma faixa apenas para percussão ou apenas para harmonia, como explanado na Figura 2. A adaptação de uma trilha que utiliza verticalização em camadas envolve alterar características de cada camada em tempo real, ou até mesmo permutar as camadas, adaptando portanto a trilha sonora como um todo ao sentimento desejado.

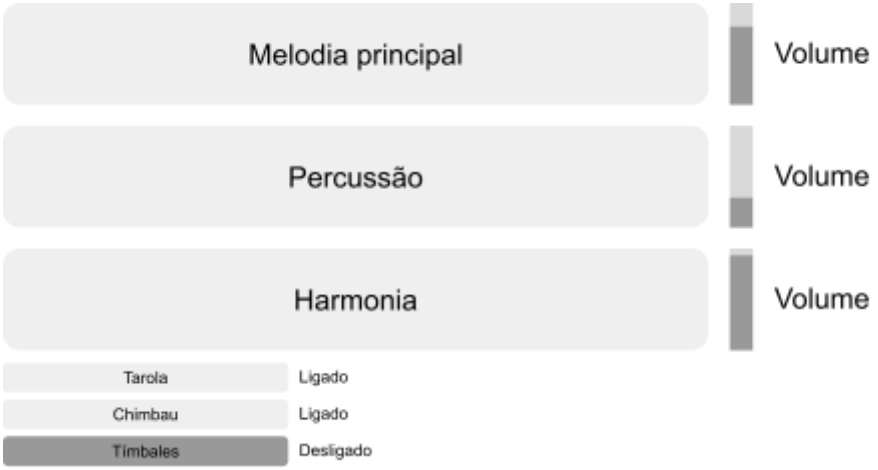


Figura 2: Exemplo de uma trilha sonora, utilizando-se uma abordagem de verticalização em camadas.

A *Sequence Music Engine*, apresentada por Sporka (2017), é um bom exemplo deste método. Implementada para desenvolver a trilha sonora de *Kingdom*

Come: Deliverance (KC:D), pode ser ligada ao motor de jogos *Cry*, e à própria Unity. A ferramenta foi projetada para ser manipulada pelos autores das trilhas musicais, para que eles possam configurar várias sequências, camadas, e definir variações para cada uma. Na experiência descrita pelos autores, eles levaram em consideração variáveis do jogo sempre sujeitas à mudança, como hora do dia, clima ou localização do jogador no mundo. KC:D usa músicas orquestrais, incluindo canto gregoriano, e a *Sequence Music Engine* também administra transições suaves entre cada trilha. O objetivo dessa funcionalidade é preservar a imersão do jogador e não chamar sua atenção para a troca de músicas, dando o sentimento que a música foi composta daquela maneira. Finalmente, essa ferramenta consegue juntar e interpolar músicas complexas, previamente compostas, conquistando resultados impressionantes em KC:D, mas com a necessidade da presença de profissionais de música altamente capacitados para operá-la.

3.1.2 Geração Procedural de Trilha Sonora

Gerar proceduralmente uma trilha sonora em tempo real é uma outra opção para se atingir música adaptativa em jogos digitais. A principal abordagem de trilhas sonoras procedurais, ou algorítmicas, é modificar em tempo de execução os parâmetros do algoritmo de geração da trilha de acordo com o que está acontecendo no jogo, levando à adaptação do tom da música às circunstâncias dele.

A principal diferença entre esta abordagem e a baseada em reprodução de faixas é que aqui a granularidade pode ser imensamente maior. Ferramentas de geração procedural de músicas comumente a fazem nota por nota, utilizando sintetizadores para “tocar” instrumentos digitais, como ocorre no *middleware* VORPAL, apresentado por Mizutani (2017). Esta abordagem também tem maior risco de gerar resultados insatisfatórios ou desagradáveis, justamente pela edição em nível mais atômico, e quase sempre em tempo de execução, e não de projeto.

Um dos primeiros grandes jogos a utilizar essa abordagem foi *Spore*, um jogo onde se joga como um organismo que está em constante evolução: você cresce, se alimenta e evolui. Como detalhado por Kosak (2008), a música de Spore acompanha o estilo de jogo e as diversas situações do mundo em constante evolução à sua volta.

Outro grande exemplo é o jogo *Doom* (2016), um jogo de tiro em primeira pessoa sobre derrotar hordas de demônios. Segundo Smith (2016), o compositor Mick Gordon utilizou vários canais de áudio, cada um com uma sequência de filtros, e dependendo do momento do jogo, os parâmetros de cada filtro são alterados em tempo real, modificando o canal de áudio e portanto a emoção geral da trilha sonora.

3.2 Avaliação sobre as abordagens

Os métodos de composição de trilhas sonoras baseadas em faixas pré-compostas podem facilitar a implementação da adaptabilidade da trilha sonora, mas continuam requerendo a posse das músicas. Portanto, desenvolvedores *indies* continuam necessitando da posse de boas faixas sonoras, o que é ainda mais difícil caso utilizem a abordagem de verticalização em camadas, já que teriam que compor ou conseguir acesso a boas faixas, separadas em camadas. Um segundo fator dificultante para equipes independentes, principalmente na abordagem da verticalização em camadas, é o da necessidade de um entendimento musical intermediário, para melhor administrar as camadas de som, e configurar uma trilha sonora que seja harmônica, como resultado.

Por sua vez, os métodos procedurais requerem um conhecimento musical muito mais aprofundado, já que o controle de composição da música é mais complexo e granular. Normalmente, os profissionais que operam ferramentas de geração procedural de trilhas sonoras têm um sólido entendimento de quais variáveis estão envolvidas na boa sonoridade de uma música, e do que as faz cativantes.

Além desses fatores, existe o fato de que muitas dessas ferramentas são implementadas para uso de uma empresa ou um projeto específico, não comumente disponíveis no mercado. As que são, por sua vez, disponíveis, como as citadas *Sequence Game Engine,* e VORPAL*,* têm sua usabilidade pensada para *sound designers*, já que esperam que eles sejam os que detenham as faixas musicais compostas e manipulem as ferramentas, inserindo-as no jogo.

Outro ponto importante ao se avaliar as ferramentas existentes é que, como muitos conteúdos gerados em tempo de execução, elas estão sujeitas a gerar resultados insatisfatórios. Esses resultados podem tanto ser desagradáveis por

falhas ou falta de refinamento de seus algoritmos, podendo gerar trilhas repetitivas e previsíveis (motivo principal do hábito de silenciar músicas sonoras, segundo McGowan (2011)), quanto por desagradarem o gosto musical do jogador.

Por fim, nota-se que ambos os métodos explicados ainda se apoiam ou na posse de boas faixas de música, ou no profundo entendimento de criação musical. Tais recursos raramente estão disponíveis para o público independente, e portanto, as abordagens afastam-se da capacidade de muitas equipes e desenvolvedores com recursos humanos e financeiros limitados.

4. Metodologia e Implementação

Este capítulo demonstra como a solução proposta, MySoundtrack, ataca os problemas evidenciados nos capítulos anteriores. Ele também detalha a metodologia utilizada neste trabalho e os passos seguidos no projeto e sua implementação. Ele está dividido em quatro seções: a primeira apresenta a ferramenta desenvolvida e fala do seu processo de concepção; a segunda discute como associar músicas do Spotify a sentimentos, e quais sentimentos foram escolhidos como representativos; a terceira explica as principais decisões de design da ferramenta e mostra seu funcionamento básico, e a quarta detalha o algoritmo de ranqueamento utilizado para selecionar as músicas que melhor se encaixam com os sentimentos desejados.

4.1 A nossa ferramenta e sua concepção

O principal objetivo de MySoundtrack é prover para desenvolvedores de jogos digitais, principalmente para aqueles sem conhecimento musical extenso ou alguém que possa fazer a composição de músicas para o projeto, uma alternativa a esse processo.

MySoundtrack pode ser facilmente utilizada para selecionar e tocar as músicas do Spotify do jogador, baseando-se na conta dele, e em parâmetros definidos pelos criadores do jogo, para que o hábito existente de se escutar música enquanto joga não ignore o fluxo emocional daquela experiência.

Nossa ferramenta utiliza-se da SpotifyWebAPI, e as músicas selecionadas são tocadas no aplicativo do jogador, seja no desktop, no browser, ou em algum dispositivo móvel. Esse fato, além de flexibilizar a experiência do usuário, também poupa discussões sobre direitos autorais, já que o áudio da música não está dentro do jogo, o produto não detém nem armazena aquela propriedade intelectual.

A concepção de MySoundtrack se iniciou durante a disciplina de Tópicos Avançados em Interface, ofertada no curso de Ciência da Computação da UFPE, no período de 2020.3. Durante a sua concepção, o Spotify foi escolhido após uma investigação que envolveu a SpotifyWebAPI, a API do Deezer e a API da Alexa (Amazon). Essas três opções foram levantadas pois são os serviços de streaming de

música mais populares atualmente (Spotify, Deezer e Amazon Music). A API do Deezer, porém, ainda está em sua fase inicial e atualmente não é possível controlar o que se está tocando, apenas interagir com as informações do usuário. Nos restou a API da Alexa e a SpotifyWebAPI, mas logo concluímos que o serviço do Spotify nos daria uma abrangência melhor do público, que facilitaria também a fase de teste inicial, já que haveriam mais voluntários assinantes, aptos a testarem. Porém, dar suporte a outros serviços é um ponto importante de trabalhos futuros.

Tendo o serviço através do qual faríamos a ferramenta, havia também a decisão de para qual plataforma iríamos fazer o nosso plug-in. Escolhemos a Unity3D por causa de sua popularidade, como demonstrada por Cowan & Kapralos (2014) e Borg *et al.* (2020). Essa popularidade é ainda maior entre os desenvolvedores independentes, que são o principal público de MySoundtrack.

Mesmo tendo escolhido a Unity, houve a preocupação de componentizar o projeto o máximo possível. A parte do código relativa à comunicação com a SpotifyWebAPI foi separada em uma *dll* .NET, facilitando o seu futuro reuso caso decidamos fazer o MySoundtrack para outras *engines*.

4.2 Classificando a emoção geral de uma música

O primeiro passo a ser feito foi analisar como a API do Spotify poderia identificar o sentimento de uma música específica. Foi visto que é possível recuperar informações que podem ser úteis para isso usando as *audio features* de cada música. As *audio features* são parâmetros que caracterizam aquela faixa de acordo com vários fatores. Esses parâmetros são:

- *Danceability*: descreve o quão apropriada a música é para ser dançada. Leva em consideração o ritmo geral e a batida.
- *Energy*: representa a intensidade da música. Quanto menos, mais a música parece lenta e calma, e quanto mais, mais ela parece veloz e barulhenta.
- *Loudness*: o volume geral, em decibéis.
- *Speechness*: a quantidade de presença de textos falados na música. Poesias, por exemplo, se aproximam do valor máximo.
- *Acousticness*: detecta se a música é acústica (sem instrumentos elétricos).

- *Instrumentalness*: valores mais altos indicam músicas com menor presença de vocalidade.
- *Liveness*: valores mais altos indicam maior presença de público.
- *Valence*: descreve a positividade de uma música. Valores mais altos tendem a ser músicas mais alegres e eufóricas, enquanto valores mais baixos, mais tristes ou raivosos.
- *Tempo*: consiste basicamente na cadência da música. Considera diretamente a duração média das batidas.

Os parâmetros escolhidos para implementar o algoritmo de ranqueamento foram Energy (Energia) e Valence (Valência). Essa escolha foi feita baseada no estudo de Russell (1980), onde é demonstrado um modelo circumplexo de afeto, uma relação bi-dimensional entre *arousal* (excitação) e *pleasure* (positividade), como demonstrada na Figura 1. Ela é mapeada do contexto de psicologia para o das *audio features* encarando excitação como energia e positividade como valência, como demonstrado no trabalho de Helmholz *et al.* (2017), através de exemplos de músicas famosas, posicionadas no modelo.

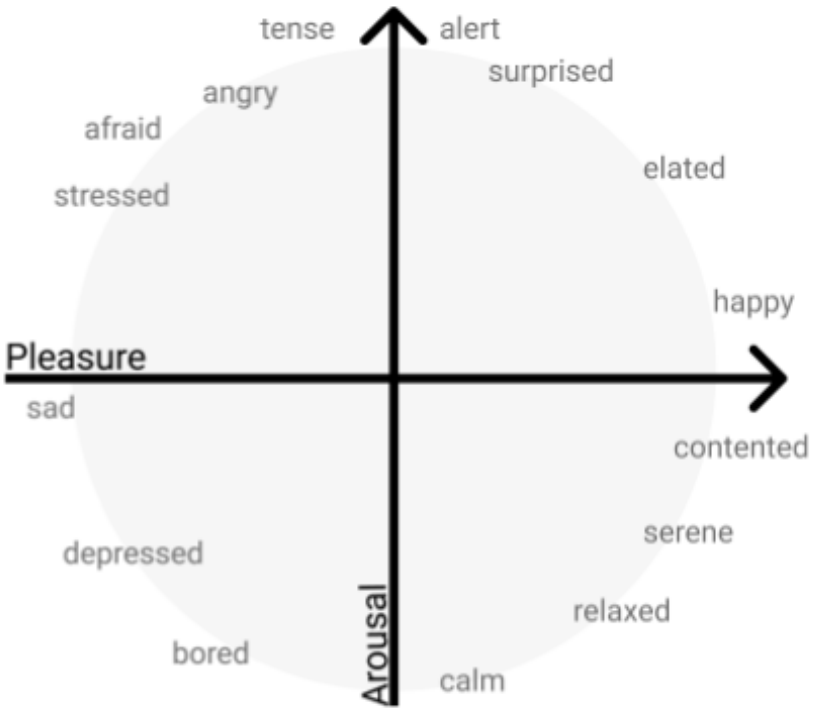


Figura 3 - Modelo circumplexo de afeto, por (Russell 1980)

Com essa simples relação estabelecida, quatro sentimentos pré-definidos, ou *vibes*, foram escolhidos, cada um localizado em um quadrante do gráfico, tentando cobrir as principais vibes presentes em jogos digitais. As *audio features* restantes serão usadas no futuro para melhor filtrar o ranqueamento baseado em preferências pessoais do jogador, como músicas mais instrumentais ou barulhentas.

4.3 Design e lógica básica de MySoundtrack

4.3.1 Design

O protótipo inicial foi implementado usando 4 emoções pré-definidas, ou *vibes*. A Tabela 1 mostra os nomes e valores de energia e valência escolhidos para cada uma. Os valores pré-definidos de cada vibe foram escolhidos e balanceados através de testes informais, buscando bem representar cada vibe comumente presente nos jogos, independentemente de seu gênero.

Nome da Vibe	**Audio Feature**	
	Energia	**Valência**
Combat	1	0.4
Adventure	0.5	0.3
Sad	0	0
Romance	0.2	1

Tabela 1: Detalhamento dos valores escolhidos para cada Vibe.

A vibe *romance*, por exemplo, teve seu valor de energia escolhido como 0.2 para dar maior prioridade aos contrastes de valência. Isso foi decidido através de observações de que os valores de energia e valência raramente se aproximam das extremidades, e tendem a flutuar de 0.2 a 0.8. Portanto, caso na vibe *romance* fosse pré-definido o valor de (0, 1) (onde 0 corresponde à energia, e 1 à valência), uma música com valores de (0.1, 0.7) seria priorizada no ranqueamento, à uma música com valores de (0.3, 0.8), por exemplo. Essas observações e a decisão dos valores de cada vibe se baseiam, por enquanto, apenas na nossa experimentação e nas dos

usuários que participaram dos testes com a ferramenta. Porém, caso o usuário não se agrade com os valores pré-definidos, ele pode customizá-los como desejar, como exemplificado na Figura 3.

Então, foi implementada um uma janela de edição customizada na Unity, através de uma classe *Custom Inspector*, para que o desenvolvedor pudesse escolher qual é a vibe associada a uma determinada área, como demonstrado na Figura 2. Nessa janela, foram desenvolvidos ícones que representassem aquela combinação de energia e valência, para não usar palavras e correr o risco de induzir a vibe a apenas um tipo de situação.

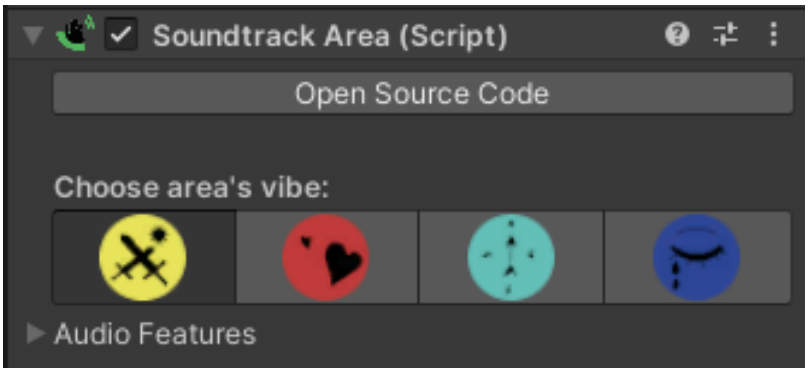


Figura 4: Custom Inspector desenvolvido na Unity

O desenvolvedor pode escolher por uma das vibes pré-definidas, ou customizar sua própria, através do menu colapsável “Audio Features”, como demonstrado na Figura 3:

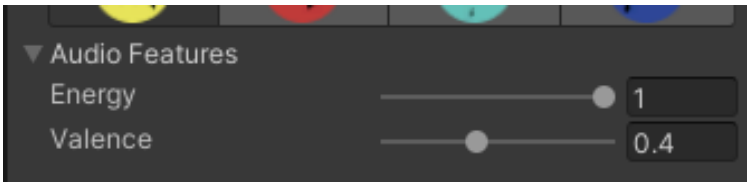


Figura 5: possibilidade de customização do sentimento associado à área, caso o desenvolvedor não se satisfaça com as vibes pré-definidas.

A versão atual conta apenas com energia e valência, pois atualmente são os únicos valores levados em consideração no ranqueamento das músicas que melhor se encaixam com os sentimentos definidos.

O resultado do design de um nível em um jogo 3D, como exemplificado na Figura 4, mostra a área envolta por cada colisor, permitindo edições básicas no seu

formato (posição, rotação e escala). Um ícone customizado e a cor do colisor ajudam a organizar as áreas entre si, principalmente quando houver muitas.

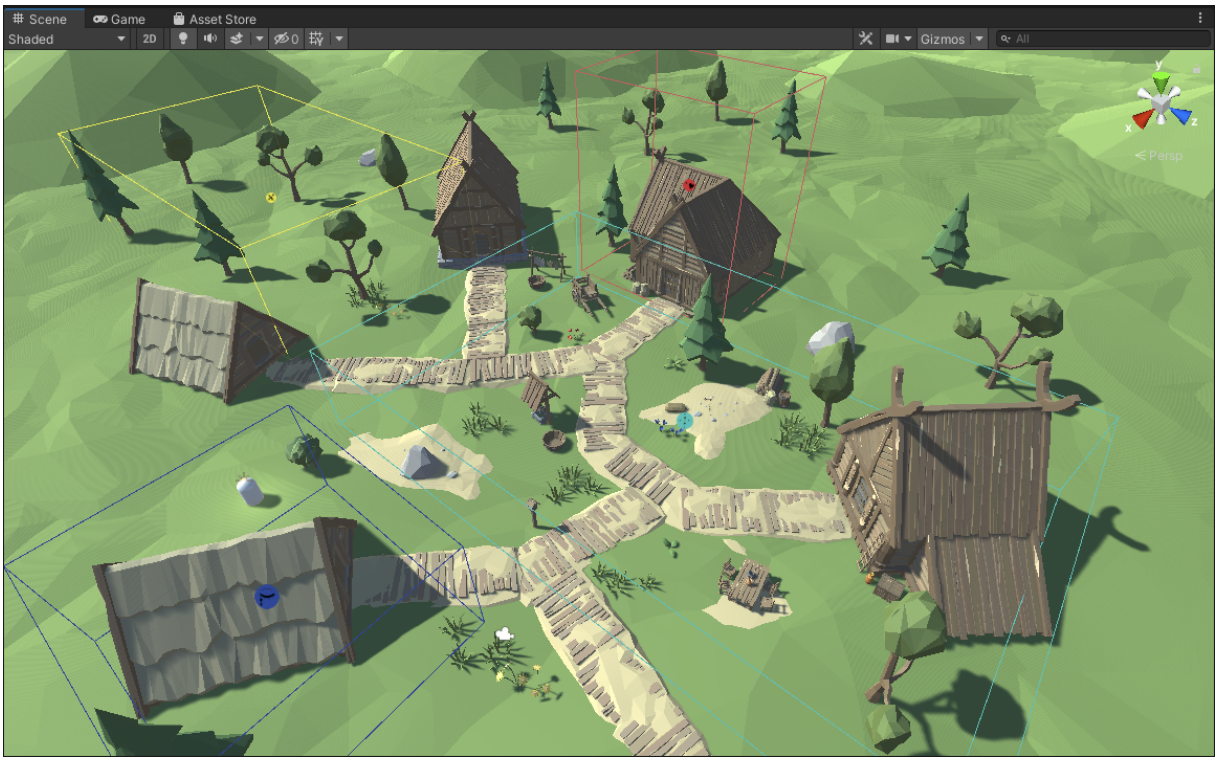


Figura 6: Exemplo de uma composição de nível, com áreas associadas a sentimentos específicos.

4.3.2 Lógica

MySoundtrack possui duas classes principais, como demonstrado na Figura 5, e ambas são *Monobehaviours*: código C# que executa acoplado a um *GameObject* dentro da cena da Unity.

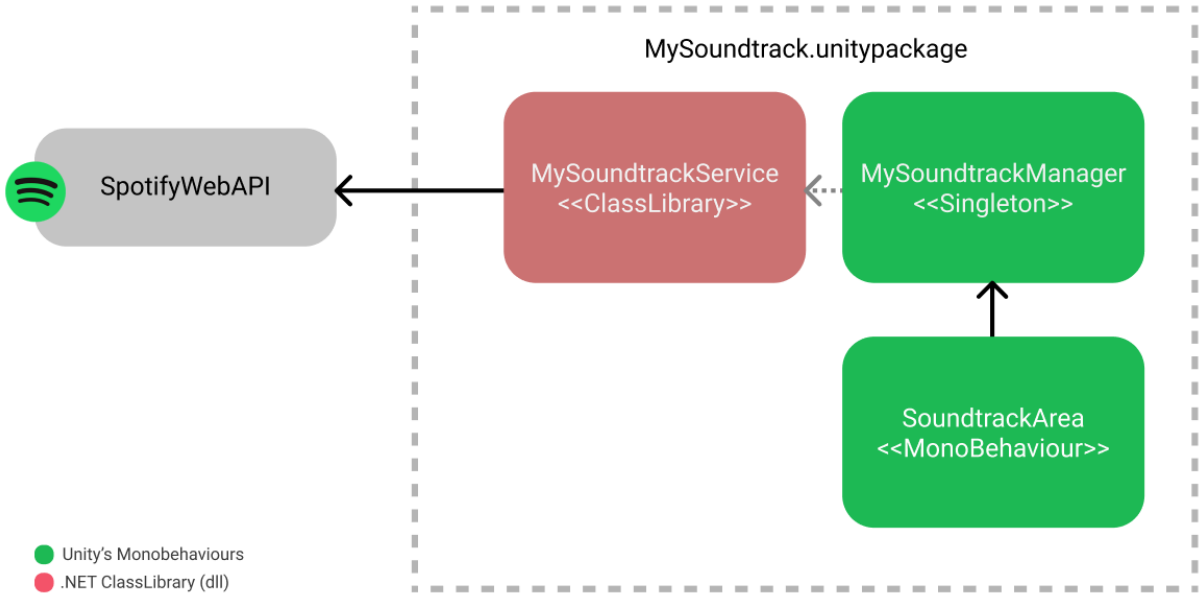


Figura 7: Arquitetura básica de MySoundtrack

Fora os dois *Monobehaviours*, há o *MySoundtrackService*, uma classe inserida no formato de uma *ClassLibrary*, responsável por fazer a comunicação entre o *manager* e a SpotifyWebAPI, desde a autenticação no aplicativo até a requisição de tocar uma música. O serviço é instanciado pelo *manager*.

No exemplo implementado e submetido à loja de pacotes da Unity, assim que o jogo começa, é pedido para o jogador entrar em sua conta do Spotify, de modo que seja possível acessar suas músicas. O momento em que isso acontece não necessariamente precisa ser o início do jogo, e fica a critério do desenvolvedor.

MySoundtrack sempre fará o ranqueamento com as músicas curtidas do jogador, a não ser que ele não tenha músicas o suficiente (o valor mínimo padrão foi definido como 150). Nesse caso, a ferramenta irá buscar se o desenvolvedor definiu *BackupPlaylists*: playlists que podem ser inseridas no MySoundtrackManager, como demonstra a Figura 6, para quando o jogador não tiver músicas curtidas o suficiente. Caso o jogador não tenha músicas curtidas o suficiente e o desenvolvedor também não tenha definido BackupPlaylists, MySoundtrack irá usar as *Featured Playlists* do jogador. *Featured Playlists* são playlists que o Spotify sugere para seus usuários, de acordo com seus algoritmos de recomendação, e normalmente aparecem na página inicial do aplicativo. As *Featured Playslists* são o último passo dessa lógica, já que mesmo um usuário recém-chegado no Spotify já as possui. Essa lógica, portanto,

sempre selecionará músicas o suficiente da conta do jogador, e encontra-se esquematizada na Figura 7.

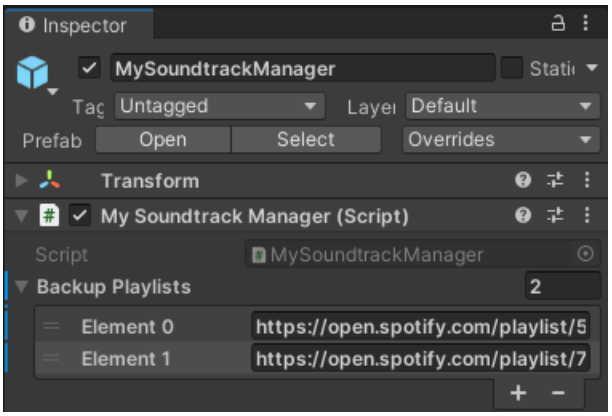


Figura 8: Uso de BackupPlaylists

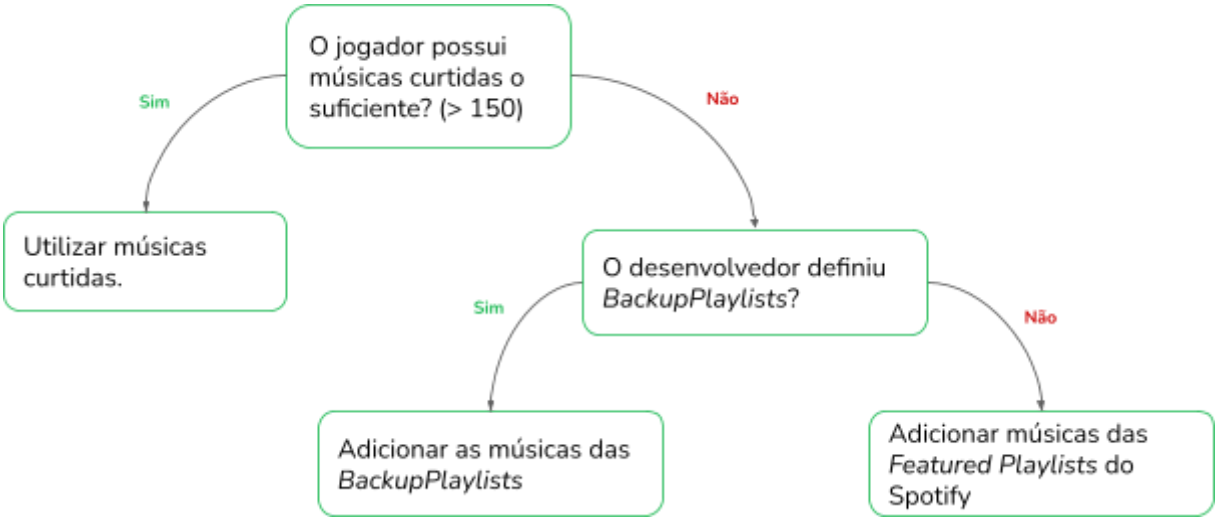
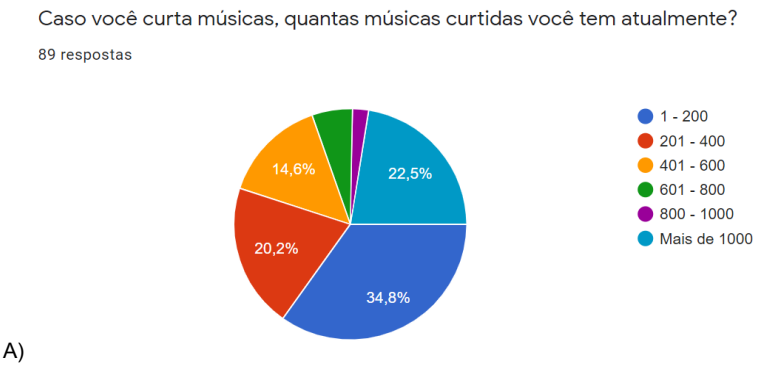


Figura 9: Lógica de decisão para seleção de músicas da conta do jogador.

As decisões de primeiro considerar as músicas curtidas do usuário, e do valor mínimo que ele deve ter para utilizar também a BackupPlaylist, se basearam em uma pesquisa realizada por nós com 102 pessoas, sobre o padrão de uso delas do Spotify. A pesquisa foi composta no formato de questionário, enviado na fase de concepção desse projeto, para alunos, professores e funcionários do CIn-UFPE. Como mostra a Figura 7, 88 pessoas (86,7%) afirmaram curtir músicas no aplicativo, gerando uma lista de músicas na biblioteca do usuário. Além disso, 54 (52,9%) pessoas costumam ouvir músicas no Spotify em uma sequência aleatória, dentro de todas as suas músicas curtidas, e 30 (29,4%) costumam ouvir de playlists curtidas.

Totalizando, portanto, uma maioria de 82,3% que na maior parte do tempo, escutam suas músicas curtidas.



A)

B)

C)

Figura 10: Resultados de questionário sobre hábitos de uso do Spotify.

O desenvolvedor pode também preferir fazer a troca de *vibes* independente da localização geográfica do jogador e das *SoundtrackAreas* (nomenclatura escolhida para o objeto que possui o colisor e representa uma área com um sentimento). Para tal, a classe MySoundtrackManager provê a ele acesso programático a um método (*PlaySongsOfVibe(energy, valence)*) que toca músicas

de acordo com as *audio features* desejadas. Dessa forma, permitimos também controlar a troca de emoções de acordo com momentos do jogo, e não apenas espaço.

4.4 Algoritmo de Ranqueamento

Quando a autenticação com o Spotify é realizada com sucesso, as *SoundtrackAreas* se inicializam. Nesse processo, elas pedem que o MySoundtrackManager faça o ranqueamento das músicas reunidas pelo processo anterior, passando seus valores de energia e valência. As músicas vencedoras do ranqueamento têm suas referências armazenadas, para que sejam tocadas quando o jogador entrar na área correspondente. As músicas selecionadas de cada área são tocadas aleatoriamente, para diminuir a chance de repetição de músicas. O algoritmo de ranqueamento é detalhado no Algoritmo 1.

Algorithm 1 Getting best tracks for audio features

Input: E, the float value of the desired audio feature energy
V, the float value of the desired audio feature valence
T, the list Tracks to rank
S, the size of the returned ranked list
Output: R, the list of ranked songs, with the size of S

1: $\tau \leftarrow$ new empty list of tuples (track, grade)
2: **for** $t \epsilon\ T$ **do**
3: $\quad a \leftarrow \text{GetAudioFeatures}(t)$
4: $\quad grade \leftarrow |GetEnergy(a) - E| + |GetValence(a) - V|$
5: $\quad Insert(\tau, t, grade)$
6: **end for**
7: $\text{SortByGrade}(\tau)$
8: $R \leftarrow \text{Sublist}(\tau, 0, \text{S})$
9: **return** $\text{SelectFirstItemFromTuple}(R)$

Algoritmo 1: Algoritmo de ranqueamento das músicas levando em consideração energia e valência da área.

O cálculo do ranqueamento é relativamente simples: quão menor a variável *grade* for, mais próxima aquela música está da combinação energia-valência em questão. Já que ambos os valores variam entre 0 e 1 e têm a mesma importância para a definição do sentimento, não foi necessário fazer nenhum tipo de ponderação

no cálculo. Essa necessidade existirá quando outras variáveis forem incluídas no ranqueamento, especialmente aquelas cujos valores possam ser menores que 0 ou maior que 1, como *loudness* e *tempo*.

5. Validação e Experimentos

Este capítulo se dedica a detalhar as validações e os experimentos conduzidos com MySoundtrack. Ele se divide em duas seções de validação, representando os dois tipos de validações executadas: prova de conceito e validação de usabilidade com desenvolvedores de jogos. No final, dedicou-se uma seção para mencionar riscos e limitações, encontradas também através dos experimentos conduzidos.

5.1 Prova de Conceito

5.1.1. Objetivo

Nós observamos a escassez de material acadêmico acerca das duas premissas levantadas no início da concepção da ferramenta: a dificuldade de composição de trilhas sonoras adaptativas, por parte das equipes independentes, e o hábito dos jogadores silenciarem a música do jogo para ouvirem suas próprias. Portanto, foram conduzidas entrevistas com pessoas no contexto de desenvolvimento de jogos para verificar a veracidade e relevância dessas duas premissas, e também para validar o conceito do protótipo de MySoundtrack e gerar *feedbacks*.

5.1.2. Método

Entrevistados foram encontrados majoritariamente através da recomendação de um entrevistado anterior, que no fim da entrevista, era pedido por mais contatos que aceitassem participar dela.

O conteúdo da entrevista foi dividido em algumas partes. Parte 1 focava no perfil técnico do participante, para melhor guiar o que seria discutido na entrevista. Parte 2 apresentava perguntas sobre sua experiência, e de suas equipes, em composição de trilhas sonoras. Parte 3 se dedicava a entender os hábitos musicais daquela pessoa, quando o assunto era jogos digitais: as perguntas procuravam entender como aquela pessoa interagia com a música do jogo. E por fim, Parte 4 buscava apresentar MySoundtrack, validar sua proposta e gerar feedbacks.

5.1.3. Resultados

Foram realizadas 23 entrevistas, com pessoas de áreas diferentes de desenvolvimento de jogos, entre 19 e 52 anos. Dos participantes, 3 se identificam pelo gênero feminino, e 20 pelo masculino. Entre eles havia 12 programadores (52%), 4 game designers (17%), 2 produtores (8%), 1 sound designer (4%), 2 professores (8%) e 2 presidentes de empresas independentes (8%). Desses 23, 19 estavam ou já estiveram envolvidos em desenvolvimento independente de jogos, e 11 (47%), são usuários *premium* do Spotify.

Apesar de a abordagem de cada entrevistado recomendar mais possíveis entrevistados ter levado o escopo da pesquisa a 4 estados diferentes do Brasil, ainda há o risco de ela estar enviesada pelo tamanho diminuto do mercado de jogos independentes do país.

Acerca do hábito de silenciar a música dos jogos enquanto jogam, 16 participantes (69%) alegaram praticá-lo, em frequências variadas. Segundo eles, a prática é mais frequente em jogos mais casuais, e principalmente jogos *mobile*. Foram também destacados jogos de caráter mais repetitivos, como *MMORPGs*, onde se gasta muito tempo aperfeiçoando seu personagem, e onde muitos jogadores afirmam até escutar *podcasts* enquanto jogavam. Outro grande ponto mencionado nesse quesito foi o aspecto social, onde 10 (43%) reportaram utilizar frequentemente ferramentas com o *bots* musicais do Discord, onde ao se reunir com os amigos em uma chamada de voz, o *bot* tocava música ao fundo. Um participante (P15), comentou sobre isso e a ferramenta: “A gente [ele e amigos] sempre joga com um bot tocando música, mas temos que selecionar uma por uma. Seria massa usar essa ferramenta para tornar esse processo automático e de acordo com o jogo”.

Nas discussões sobre a escassez de profissionais de som e música em equipes independentes, dos 19 com experiência em desenvolvimento indie, 14 (73%) afirmaram que raramente ou nunca tiveram apoio de um especialista em música para compor trilhas sonoras. Segundo eles, a abordagem comum é buscar por músicas com licença permissiva na internet, e relataram que muitas vezes essa abordagem produz resultados medíocres. As principais queixas são a perda de

identidade musical da obra, e o tempo gasto procurando músicas que se encaixem com o desejado e entre si. Essa frustração é exemplificada pelas afirmações do participante (P04): “Já joguei jogos que tinham músicas que eu já havia usado em projetos pessoais. A sensação é bem chata”, e do participante (P11): “Isso de pegar músicas online só é bom quando você tem dinheiro pra gastar em músicas únicas. Fora isso, gera resultados muito medíocres.”

Apenas 2 (8%) dos entrevistados afirmaram não ter interesse nas funcionalidades oferecidas pela ferramenta, como declarou o participante (P17): “Não gosto de ouvir outras coisas enquanto jogo. Valorizo todas as partes da obra, inclusive sua música, e não gostaria de interferir na experiência do jogador com fatores externos, como o Spotify”. O restante (92%) declarou ao menos curiosidade de testá-las em projetos pessoais ou profissionais, e alguns destes (P02, P04, P05, P13, P16, P18, P21, P23), disseram se interessar na ferramenta fora do contexto de substituir toda a trilha sonora, como afirmou o P05: “É uma ferramenta muito interessante para se fazer uma Game Jam, e ver que tipo de mecânica surge ao redor dela, sem ser tirar a trilha sonora de dentro do jogo”. Esse participante tem experiência em organização de Game Jams, e se interessou em executar uma que utilizasse MySoundtrack, em sua turma de alunos.

Alguns casos de uso interessantes foram descritos por entrevistados, que se afastam do conceito de usar o MySoundtrack como um substituto de trilha sonora. Um exemplo recorrente foi o sistema de rádio encontrado em alguns jogos da série *Grand Theft Auto*, onde ao jogador dirigir pela cidade, no carro tocam músicas de rádio, em vez da trilha sonora normal do jogo. Alguns entrevistados (P02, P04, P16, P21) se imaginaram implementando esse sistema, mas personalizando-o com os hábitos musicais do jogador, localizando aquela experiência ao contexto dele. Um outro caso sugerido foi o uso de MySoundtrack para quebrar a quarta parede e falar diretamente com o jogador, como no jogo *Doki Doki Literature Club!*, onde a antagonista lê arquivos do computador do jogador e dialoga com ele baseado nisso. O entrevistado (P13) disse querer fazer um jogo onde uma lógica parecida seria implementada com a ajuda de MySoundtrack, e onde o antagonista manipularia o que estaria tocando no Spotify do jogador de maneira cínica, de acordo com a narrativa.

5.1.4. Discussão

Os resultados das entrevistas apontam para a validação positiva das premissas em que MySoundtrack se apoia, e gerou uma poderosa reflexão sobre o aspecto social do hábito de ouvir outras músicas enquanto se joga, que pode ser explorada no futuro. Também é notória a carência da comunidade por ferramentas que facilitem o custoso processo de desenvolvimento de jogos em um time pequeno, e a nossa ferramenta se viu também validada no contexto de trilhas sonoras, um campo cuja especialidade não está tão presente nas equipes indie.

Com esse experimento, obteve-se também uma nova noção de prioridades dos trabalhos futuros, e que tipo de funcionalidade agrega mais valor para os usuários. Dois pontos que foram priorizados, a partir das entrevistas, foram: ter uma versão de MySoundtrack que considerasse apenas músicas sem direitos autorais, e fazer com que a ferramenta suporte outras plataformas de *streaming*, para não se limitar aos usuários do Spotify. Sobre o primeiro ponto, participantes (P02, P18) afirmam que é fundamental, para poderem gravar conteúdo dos jogos feitos com MySountrack sem preocupações. Sobre o segundo, entrevistados (P07, P10), afirmaram que adorariam testar, mas ficaram frustrados ao saber que as funcionalidades estavam disponíveis apenas aos assinantes do Spotify: “dá até vontade de assinar o Spotify pra isso”.

5.2 Validação da usabilidade da ferramenta

5.2.1. Objetivo

Tendo validado o conceito de MySoundtrack, esta etapa de experimentação envolve testar e analisar a experiência do desenvolvedor utilizando a ferramenta. Este estudo busca contemplar tanto a relevância da ferramenta, sentida pelo desenvolvedor, quanto sua usabilidade dentro dos controles da Unity.

5.2.2. Método

Foi utilizado o *System Usability Scale* (SUS) (Brooke, 1996) para avaliar a usabilidade da ferramenta. Primeiramente, houve uma breve apresentação do

propósito de MySoundtrack, contemplando apenas a motivação de sua existência e o que a ferramenta possibilita fazer, sem qualquer instrução ou demonstração de uso, a fim de não prejudicar os dados do experimento. Ao único usuário que não possuía qualquer experiência na Unity, foi ensinado como instalar a *engine* e como importar um pacote nela. Foi pedido, então, que usuário seguisse uma sequência de passos para testá-la, e depois respondesse o formulário SUS sobre sua experiência. Os passos sugeridos no experimento foram:

1) Crie um projeto novo na Unity ou abra um existente;
2) Importe o pacote MySoundtrack;
3) Crie uma cena nova ou abra uma já existente;
4) Adicione à cena o prefab MySoundtrackManager;
5) Adicione à cena quantas vezes desejar, o prefab SoundtrackArea;
6) Pelo inspetor, escolha a vibe associada à cada área, ou customize a vibe da maneira que desejar.
7) Com o Spotify aberto em algum dispositivo, teste a cena montada.

Foram concedidos também, juntamente com o pacote, alguns recursos de demonstração, como código de um personagem que responde à entrada do mouse e teclado, de modo a facilitar a composição de uma cena para testagem. Aos desenvolvedores que já possuíam experiência com a Unity, não foram dadas mais instruções além desses passos. Nos poucos casos onde o desenvolvedor possuía pouco conhecimento da *engine*, foram dadas instruções mínimas de manipulação de cena a dos objetos nela.

Essa experimentação foi restrita, por limitação da SpotifyWebAPI, aos desenvolvedores de jogos que têm assinatura no aplicativo.

5.2.3. Resultados

A validação teve a participação de 10 pessoas, com níveis de experiência variados na Unity, como demonstrado no eixo horizontal da Figura 8. Todos os participantes se identificam com o gênero masculino, e estão distribuídos entre 22 e 30 anos. Quanto à formação, 6 (60%) deles são alunos ou recém-formados no curso

de Ciência da Computação da UFPE; 1 (10%) deles é aluno de Sistemas de Informação da UFPE; 2 (20%) deles são formados no curso de Jogos Digitais da Universidade Católica de Pernambuco; e 1 (10%) é formado em Design e atua como artista 3D.

MySoundtrack alcançou uma pontuação de 82.22, dando à ferramenta uma nota A, segundo Lewis *et al*. (2018), e o adjetivo Excelente, segundo Bangor *et al.* (2009). A Figura 8 mostra a distribuição dos resultados de cada participante, agrupados por seu nível de experiência com a Unity, e a média das pontuações, com o seu respectivo desvio padrão (8.99).

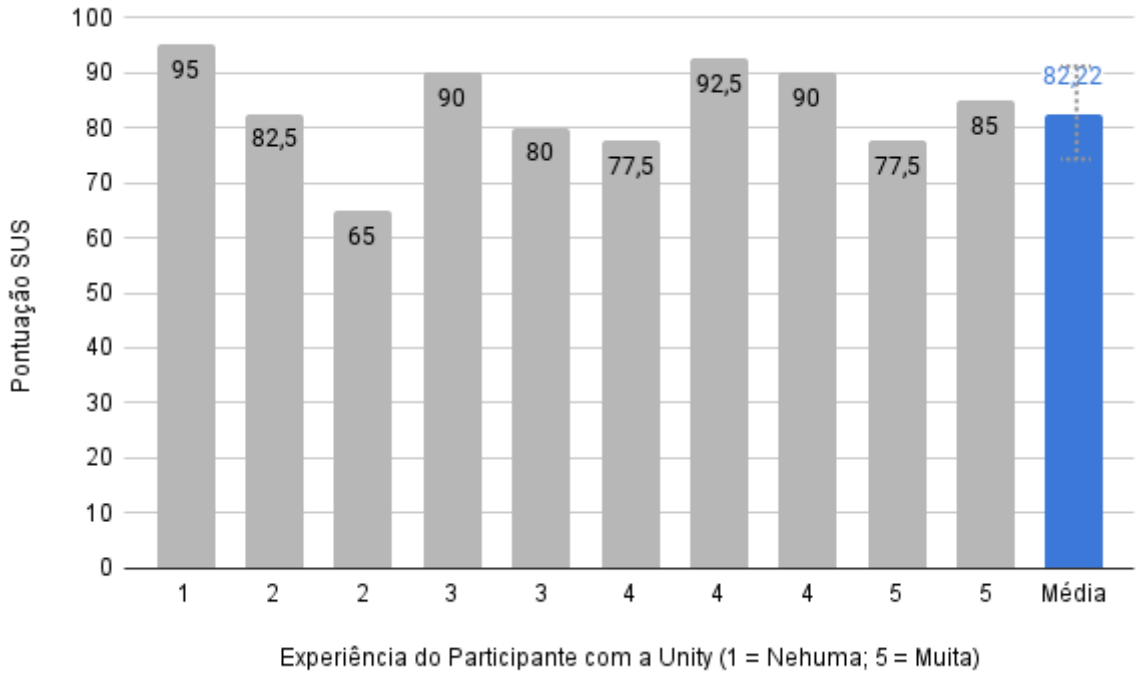


Figura 11: Média das pontuações obtidas no questionário SUS e distribuição das pontuações de cada participante, agrupadas pela experiência afirmada possuir com a Unity.

A Figura 9 mostra a distribuição das pontuações do questionário, para cada item. No SUS, cada item recebe uma pontuação de 0 a 4, sendo 4 o melhor resultado possível. Por exemplo, nos itens de tom negativo, como “Achei MySoundtrack desnecessariamente complexo”, o melhor resultado, que geraria a pontuação 4, seria a resposta “Discordo Totalmente”. Já nos itens de tom positivo, como “Eu acho que gostaria de usar MySoundtrack frequentemente”, o resultado que geraria a pontuação 4 seria “Concordo Totalmente”.

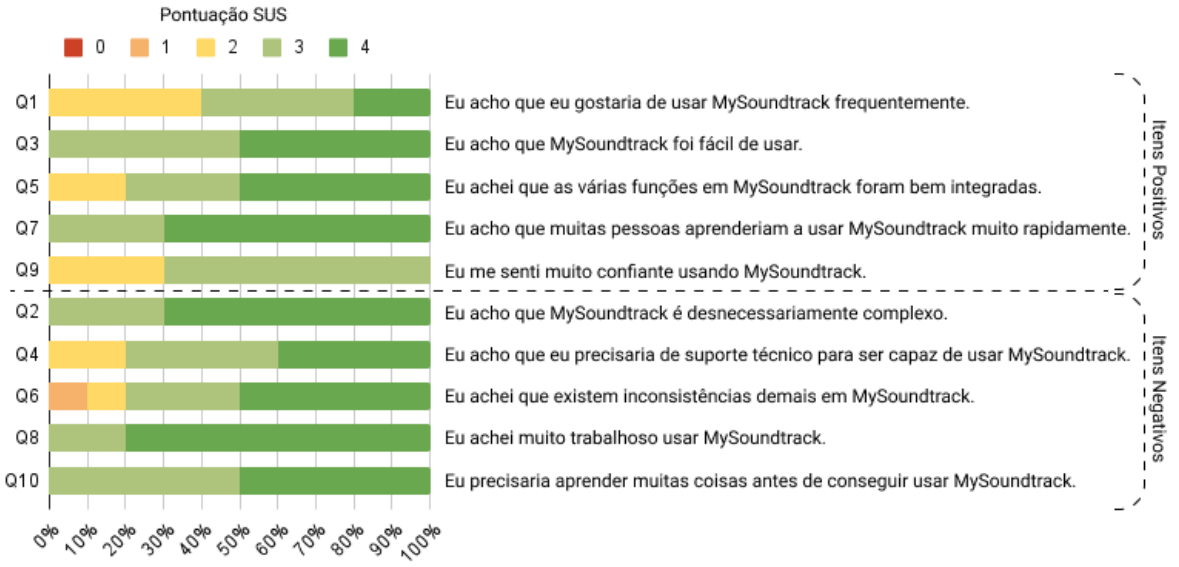


Figura 12: Distribuição das pontuações do questionário SUS aplicado com desenvolvedores de jogos, para cada item.

5.2.4. Discussão

Os resultados do questionário SUS sugerem que MySoundtrack teve uma boa aceitação pelos desenvolvedores de jogos: houve mínima dificuldade na sua utilização e causou boas impressões com sua relevância.

Um ponto interessante a se notar é a relativa uniformidade das pontuações dos participantes, independente do nível de domínio da Unity. A nossa hipótese para a causa desse fato é que a ferramenta não se utiliza de funcionalidades da Unity, em sua interface. Ela consegue assemelhar-se com interfaces usuais, e não traz, para o seu entendimento, o contexto específico da Unity, permitindo portanto, uma intuição maior de seu funcionamento.

Outro ponto relevante é que apesar da ausência de respostas negativas no primeiro item, houve uma grande porcentagem (40%) de respostas neutras. Isso pode indicar uma dificuldade de visualização de casos de usos, que pode ser mitigada com o desenvolvimento de mais exemplos de caso de uso no pacote.

5.3 Riscos e Limitações

Todo o feedback a respeito da relevância e da qualidade de MySoundtrack foi colhido da perspectiva dos desenvolvedores, gerando um risco de viés. Se faz necessário ainda, no futuro breve, validar o ponto de vista do usuário final, o jogador,

ao utilizar um jogo que se utilize da nossa ferramenta. Essa validação precisará distinguir a qualidade do ranqueamento de MySoundtrack da qualidade de escolha de qual sentimento se encaixa em cada momento, do desenvolvedor, principalmente se ele definir vibes de valores customizados.

Por limitações da SpotifyWebAPI, só é permitido manipular o que está tocando quando o usuário possui uma assinatura paga do aplicativo. Esse fato prejudicou os experimentos pois limitou o seu público, já que algumas pessoas utilizam o Spotify sem ser um assinante pago. Esta é, ao nosso ver, a principal limitação de MySoundtrack em questão de abrangência de público, e é também uma queixa recorrente dos participantes da Prova de Conceito que não eram assinantes do aplicativo. Portanto, prioriza-se como trabalho futuro a atividade de abranger o público alvo da nossa ferramenta para também outros serviços de *streaming* de música.

Outra limitação é em relação à escalabilidade da ferramenta, em números de usuários simultâneos: a SpotifyWebAPI possui um limite de requisições por minuto. O serviço não especifica precisamente o número limite, porém, experimentações foram feitas com até 30 requisições por segundo, e não houve quebra do limite. Portanto, MySoundtrack estará bem guarnecida, mas é uma limitação necessária de se ter em mente, caso o uso da ferramenta cresça demasiadamente, e otimizar as requisições feitas à API é um melhoramento que pode se tornar necessário em breve.

Por fim, como toda lógica procedural, que tira a exatidão da experiência das mãos do desenvolvedor, existe o risco de o ranqueamento gerar resultados insatisfatórios. Isso pode acontecer quando o usuário tiver poucas músicas ou todas suas músicas serem de um estilo específico, dificultando fornecer à ferramenta o alcance de sentimentos diferentes. A lógica implementada, de se utilizar também *BackupPlaylists* e as *Featured Playlists* do usuário, mitiga esse risco, porém ele ainda é um ponto importante para a boa experiência do jogador. Esse pode ser ultimamente contornado através de uma avaliação prévia das músicas curtidas do jogador, a fim de verificar a amplitude de sentimentos nelas. Dessa maneira, utiliza-se as *Backup Playlists* e *Featured Playlists*, caso o jogador não possua uma

variedade de sentimentos em suas músicas curtidas, mesmo que ele tenha uma quantidade suficiente delas.

Em suma, concluímos que MySoundtrack ainda possui pontos pertinentes de melhoramentos e riscos a serem mitigados. Pontos negativos levantados pelos usuários na fase de experimentação foram priorizados em trabalhos futuros, como a funcionalidade de filtrar músicas com direitos autorais e o suporte a outras plataformas de *streaming* de música.

6. Conclusão e Trabalhos Futuros

Este capítulo conclui o trabalho, resgatando seus objetivos estabelecidos no início, e apresenta as perspectivas de trabalhos futuros, se baseando principalmente na validação executada e nos experimentos realizados.

6.1 Conclusão

Nesse trabalho, apresentamos o MySountrack, um plug-in da Unity que permite guiar as músicas que se escutam enquanto se joga, de maneira adaptativa e personalizada. A ferramenta utiliza-se da SpotifyWebAPI, e toma proveito do perfil do usuário do Spotify para personalizar seu sistema de ranqueamento.

Usos e objetivos de trilhas sonoras adaptativas em desenvolvimento de jogos digitais foram detalhados, ponderando-se sobre a dificuldade de implementação de alto nível de trilha sonora, especialmente em cenários indies. Adicionalmente, foi explanado o hábito de jogadores silenciarem a música do jogo enquanto jogam, e como esse fato pode tornar contraproducente o grande esforço na composição musical do jogo.

A lógica básica foi exposta, passando pelas decisões na concepção da ferramenta, pela base teórica, escolhas de design, detalhes de implementação, limitações e possíveis riscos. O código da ferramenta foi disponibilizado no Github[1] e o plug-in foi também publicado na *Unity Asset Store*[2].

O pacote foi publicado na loja de pacotes da Unity no dia 4 de agosto de 2021, e até o dia 28 do mesmo mês, obteve 606 visualizações, 199 vendas, 47 *downloads* e nenhuma avaliação. A compra de um pacote gratuito, como o MySoundtrack, representa a adição do pacote à conta de um usuário da Unity, sendo o mesmo processo da compra de um pacote pago, mas sem a etapa do pagamento.

Validações foram feitas em duas etapas, e cada uma indicou a relevância da ferramenta proposta. A Prova de Conceito reiterou a demanda por uma alternativa ao processo de composição de músicas em equipes independentes, e o Teste de Usabilidade executado com desenvolvedores de jogos obteve uma pontuação de

[1] Disponível em: https://github.com/Danielfib/MySoundtrack

[2] Disponível em: https://assetstore.unity.com/packages/audio/music/mysoundtrack-197430

82,22 no SUS, sugerindo sua facilidade de uso e relevância. Por fim, foram ponderadas as limitações e riscos de MySoundtrack, e também pontos negativos levantados na experimentação da ferramenta.

[1] Disponível em: https://github.com/Danielfib/MySoundtrack

[2] Disponível em: https://assetstore.unity.com/packages/audio/music/mysoundtrack-197430

6.2 Trabalhos Futuros

6.2.1 Escolha de músicas sem *royalties* para criadores de conteúdo

Mencionada em muitas das entrevistas de Prova de Conceito, uma opção *royalty-free* de MySoundtrack já havia sido idealizada, e foi recebida como “muito relevante”, às vezes até desejada pelos entrevistados. Tal funcionalidade consiste em filtrar, no ranqueamento, as músicas com direitos autorais, que poderiam ser um problema para produtores de conteúdo.

Esse tópico engloba uma constante discussão existente por parte de criadores de conteúdo e principalmente, *streamers*, que se preocupam em ter seu conteúdo desmonetizado por alguma reivindicação de direitos autorais, como detalha Oliver *et al.* (2021). Portanto, esses criadores escolhem criteriosamente tudo que é mostrado e tocado em seus conteúdos, dando prioridade a músicas de domínio público ou de licenças permissivas. Um exemplo dessa demanda são as *playlists* públicas no Spotify, com músicas sem direitos autorais, feitas exclusivamente nesse intuito. A importância dessa funcionalidade cresce ainda mais quando se considera que grande parte do marketing de jogos, principalmente dos indies, dependem do patrocínio de criadores de conteúdo, para jogarem o jogo e darem uma opinião positiva dele para o seu público.

Essa funcionalidade adiciona complexidade ao sistema de ranqueamento, já que muito provavelmente o jogador não possuirá músicas suficientes sem direitos autorais, e será necessário inferir quais dessas músicas melhor combinam com o gosto musical do jogador.

6.2.2 Criar uma playlist com músicas tocadas enquanto jogou

Alguns desenvolvedores, enquanto entrevistados, deram a muito bem recebida sugestão de quando o jogo terminar, MySoundtrack gerar uma playlist no Spotify. Essa playlist representaria o resumo da experiência do jogo, usando músicas que foram ranqueadas e tocadas em momentos chaves, tal qual um álbum de trilha sonora original.

Essa é uma funcionalidade *nice-to-have* muito interessante, e é validada pelos esforços do Spotify de personalizar e recompensar a experiência do usuário no aplicativo: o *Spotify Wrapped* consiste na principal evidência desse esforço e do sucesso dele. No *Spotify Wrapped*, o usuário recebe um panorama musical do seu ano, com estatísticas interessantes e até uma playlist personalizada que representa aquele ano do usuário.

6.2.3 MySoundtrack para outras plataformas

Talvez a principal limitação de MySoundtrack atualmente seja estar restrita a assinantes do Spotify. De acordo com feedbacks recebidos na Prova de Conceito, se faz necessário o suporte a outras plataformas de streaming de música, como o Amazon Music, Deezer, e até mesmo algum bot de música no Discord. A arquitetura atual já foi pensada abstraindo código específico a uma plataforma, de modo que as mudanças seriam mínimas na classe principal, MySoundtrackManager.

6.2.4 Filtros de áudio

Outra funcionalidade frequentemente usada em design e som para jogos é a aplicação de filtros de som, com o objetivo de incrementar a imersão do jogador. Isso normalmente acontece quando o jogador está debaixo d'água, ou sofre um grande impacto, como por exemplo, quando uma granada explode próxima a ele, todo o áudio se torna abafado.

Já que não é uma opção atual fazer com as músicas toquem dentro do áudio do jogo, foi brevemente explorada a possibilidade de fazer com que o jogo analisasse que música está sendo tocada, e tocar um áudio que, junto à música, a faria parecer estar com um filtro sonoro, como o exemplo mencionado. Essa funcionalidade pareceu complexa demais para receber atenção prioritária agora, e foi listada como investigação futura.

7. Referências

McGowan, J. (2011). Harmonious: An Emotion-Matching System for Intelligent use of player's own music libraries with game soundtracks. MSc project, Leeds Metropolitan University.

De Salas, K., Lewis, I., & Bindoff, I. (2016). Game jams as an opportunity for industry development. In 1st International Joint Conference of DiGRA and FDG (DiGRA/FDG'16) (pp. 1-14).

Steinke, Thomas & Linsenbard, Max & Fiske, Elliot & Khosmood, Foaad. (2016). Understanding a Community: Observations from the Global Game Jam Survey Data. 15-21.

Borg, Markus & Garousi, Vahid & Mahmoud, Anas & Olsson, Thomas & Stalberg, Oskar. (2020). Video Game Development in a Rush: A Survey of the Global Game Jam Participants. IEEE Transactions on Games. PP. 1-1.

Gungormusler, Alper & Paterson, Natasa & Haahr, Mads. (2015). barelyMusician: An Adaptive Music Engine For Video Games.

Young, D. M. (2012). Adaptive Game Music: The Evolution and Future of Dynamic Music Systems in Video Games [Undergraduate thesis, Ohio University]. OhioLINK Electronic Theses and Dissertations Center.

Kähärä, L. (2018). Producing adaptive music for non-linear media.

Sporka, A. and Jan Valta. "Design and implementation of a non-linear symphonic soundtrack of a video game." New Review of Hypermedia and Multimedia 23 (2017): 229 - 246.

MIZUTANI, Wilson Kazuo. VORPAL: a middleware for real-time soundtracks in digital games. 2017. Dissertação (Mestrado em Ciência da Computação) - Instituto de Matemática e Estatística, University of São Paulo, São Paulo, 2017.

D. Kosak, "The bet goes on: Dynamic music in Spore," GameSpy, Feb. 2008. Available: http://uk.pc.gamespy.com/pc/spore/853810p1.html

B. Smith, "Rip and Tear: Deconstructing the Technological and Musical Composition of Mick Gordon's Score for DOOM (2016)". The University of Adelaide, 23-Oct-2017.

B. Cowan and B. Kapralos, "A Survey of Frameworks and Game Engines for Serious Game Development," 2014 IEEE 14th International Conference on Advanced Learning Technologies, 2014, pp. 662-664.

Russell, J. A. (1980). A circumplex model of affect. Journal of Personality and Social Psychology, 39(6), 1161–1178.

Helmholz, Patrick and Siemon, Dominik and Robra-Bissantz, Susanne. (2017). Summer hot, Winter not! – Seasonal influences on context-based music recommendations.

J. Brooke, “Sus: a “quick and dirty usability,” Usability evaluation in industry, p. 189, 1996.

Lewis, J. R., & Sauro, J. (2018). Item benchmarks for the system usability scale. Journal of Usability Studies, 13(3).

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. Journal of usability studies, 4(3), 114-123.

Oliver, P. G., & Lalchev, S. (2021). Music copyright, creators and fans. The Present and Future of Music Law, 82.