Jogo digital de ritmo com análise musical em tempo real

João Pedro B. A. Zanchett[1], Bruno Santos[2], Leonardo B. Estacio[3]

[1]Instituto Federal de Santa Catarina (IFSC) - Câmpus Lages
Curso Superior de Ciencia da Computação
R. Heitor Villa Lobos, 222 - Sao Francisco, Lages - SC, 88506-400

{joao.pba, bruno.S1}@aluno.ifsc.edu.br, leonardo.bravo@ifsc.edu.br

Abstract. *Digital rhythm games are musical themed, where the player must perform actions in the rhythm of the music. In many scenarios, mainly by smaller teams, the elements that accompany the music rhythm are manually organized by the developers, however, the use of Music Information Retrieval (MIR), a science that aims to extract information from music, can be used to automate this process. This article aims to contextualize the reader about this type of game and also to develop a rhythm game with an algorithm that automatically generates musical notes with real-time analysis of songs.*

Resumo. *Jogos digitais de ritmo possuem temática musical, onde o jogador deve realizar ações no ritmo da música. Em muitos cenários, principalmente por equipes menores, os elementos que acompanham o ritmo da música são organizados manualmente pelos desenvolvedores, no entanto, o uso de Music Information Retrieval (MIR), uma ciência que visa extrair informações de músicas, pode ser utilizado para automatizar esse processo. Este artigo visa contextualizar o leitor sobre esse tipo de jogo e, também, desenvolver um jogo rítmico com um algoritmo que gera notas musicais automaticamente através da análise em tempo real de músicas.*

1. Introdução

Jogo é um termo do latim "*jocus*", que significa brincadeira, divertimento. São atividades estruturadas praticadas com fins recreativos e, em alguns casos, fazem parte de instrumentos educacionais usados para passar uma mensagem aos jogadores. Jogos geralmente envolvem estimulação mental ou física e muitos deles ajudam a desenvolver habilidades práticas, servindo como uma forma de exercícios, expressão de algum tipo de arte, participação no processo educativo, entre outros. "Um jogo é uma forma de arte na qual os participantes, denominados jogadores, tomam decisões, a fim de gerir os recursos através de elementos de jogo na busca de um objetivo" (Costikyan et al., 1994).

Música, um termo do latim "*canticum*", é uma forma de arte que se constitui na combinação de sons e ritmos, de forma a se tornar agradável ao ouvido. É considerada por diversos autores como uma prática cultural e humana seguida por todas as civilizações, tendo funções didáticas, de lazer, entre outros (de Moraes, 2017).

Músicas são ferramentas que auxiliam no suporte cognitivo desde a infância, em brincadeiras e aprendizados. Estando presente em nosso cotidiano desde a antiguidade, a

música vai muito além da indústria musical, sendo fator de grande importância no desenvolvimento humano no geral, tanto como atividade de lazer, quanto em atividades práticas, como exercícios, estudos e jogos (Betti et al., 2013).

Um jogo digital (ou videojogo ou jogo eletrônico), termo que se refere a jogos desenhados para serem jogados num computador, num console ou outro dispositivo tecnológico, como *smartphones*, pode ser definido como um jogo onde existe interação entre humano e computador, recorrendo ao uso de tecnologia (Pivec e Kearney, 2007).

Após inúmeras discussões entre os historiadores, entrou-se em um consenso de que o primeiro jogo digital da história surgiu em 1958. Segundo Amorin (2006), esse jogo foi criado pelo físico Willy Higinbotham e recebeu o nome de *Tennis Programing*, também conhecido como *Tennis for Two*. Ele era um jogo muito simples, jogado por meio de um osciloscópio.

Jogos de digitais são, no mundo moderno, uma das principais formas de lazer e até mesmo renda para muitas pessoas, independentemente de faixa etária, cor, etnia, etc, como é dito na PGB (2021) (Oitava Edição da Pesquisa Game Brasil). Desenvolver jogos é algo que se utiliza de vários elementos computacionais, como Computação Gráfica, Inteligência Artificial, Redes, etc. E seu principal objetivo é, no final, trazer diversão, entretenimento e cultura ao jogador (Clua e Bittencourt, 2005).

A criação de um jogo *Indie* (jogos feitos por pequenas empresas ou por apenas uma pequena equipe) ou *Triple A* (jogos com maiores orçamentos e níveis de qualidade, geralmente feitos por grandes empresas), pode levar meses ou até mesmo anos de desenvolvimento, visto que são necessários vários conhecimentos de áreas diferentes trabalhando em conjunto para entregar o produto final, como conhecimentos em efeitos sonoros, trilha sonora, *level design*, programação, *marketing*, entre outros (Schell, 2008). Uma das áreas que toma um grande tempo de desenvolvimento diz respeito ao *Level Design*, que corresponde a criação de mapas e fases, mundos, algo que acarreta em uma série de adiamentos e falta de polimento, visto que tudo é feito manualmente pelos desenvolvedores.

Em um jogo muito conhecido, *Guitar Hero* (Activision, 2005), o jogador deve, através de controles padronizados, atingir notas musicais coloridas que vêm em direção a ele no ritmo da música que é tocada ao fundo, no tempo e ordem certa para acumular pontos, simulando um instrumento real. Esse tipo de jogo, popularmente conhecido como "Jogo Rítmico", sofre do problema de tempo do *Level Design*, pois cada nota da música costuma ser posicionada manualmente pelo desenvolvedor e, levando em conta que músicas possuem vários minutos de duração e várias notas diferentes, o tempo de desenvolvimento de um jogo nesse estilo pode levar anos, já que estes possuem várias músicas diferentes, com vários níveis de dificuldade (cada nível possuindo notas a mais para serem posicionadas).

Mesmo com a grande evolução dos jogos rítimicos, como é possível ver em *Osu!* (Dean Herbert, 2007), ainda existe um trabalho em colocar as notas manualmente, ou em *Beat Saber* (Beat Games, 2018) onde há a necessidade de adicionar modificações (popularmente chamados "Mods") ao jogo (Bégel et al., 2017).

A ideia de um jogo ritmico, onde os níveis são gerados proceduralmente de acordo com os Batimento Por Minuto (BPM) de uma música escolhida pelo jogador, a fim de ter sua canção favorita disponível ao invés de jogar algo desconhecido, ainda é escassa e não tão

explorada como deveria ser. Com as leituras feitas para embasamento deste artigo, como visto na seção 2.1, pode-se notar uma falta de pesquisas na área de jogos rítmicos, tanto nacionais quanto internacionais.

Pelo exposto, o artigo pretende apresentar, como objetivo geral, o processo de desenvolvimento de um jogo digital de ritmo, sendo o objetivo principal a adição de um sistema inteligente que gere os pontos e obstáculos de forma automática, fazendo uma análise em tempo real de músicas escolhidas pelo jogador utilizando *links* de servidores *web*, assim facilitando o processo de *level design*.

Todo o processo será listado na seção de Desenvolvimento, como:

- O processo de *game design* e programação;
- Modificação de objetos 3D e criação de cenários;
- Codificar o processo de interpretação com base em músicas escolhidas por serviço web;
- Utilização dessas músicas para a criação dos obstáculos que surgirão no cenário.

A estrutura restante deste documento se compõe em seis seções organizadas da seguinte forma: A Seção 2 apresenta as definições e os elementos teóricos referenciais que são necessários para um melhor entendimento da proposta. A Seção 3, apresenta a abordagem proposta e detalha a arquitetura de sistema projetada, sendo assim uma extenção aprofundada da Seção 2. Além disso, apresenta os trabalhos correlatos, onde foram buscadas referências para o desenvolvimento do trabalho. A Seção 4 mostra os aspectos de implementação da aplicação proposta destacando como os componentes da arquitetura de sistema projetada foram implementados. A Seção 5 informa os resultados gerados com o desenvolvimento, além de discussões gerais sobre o tema assim como as dificuldades encontradas. Por fim, na Seção 6 são apresentadas as considerações sobre o trabalho e possibilidade de trabalhos futuros.

2. Materiais e Métodos

O processo metodológico foi divido em três etapas macro, a primeira abrangendo pesquisas contextuais sobre o tema, com a utilização de ferramentas como Google Acadêmico, a segunda, escolha das ferramentas que serão utilizadas, assim como seus estudos através de cursos e documentação, e a terceira, com a concepção do GDD (*Game Design Document*, um documento altamente descritivo sobre o desenvolvimento do jogo), definindo a arquitetura de *software*, levantamento e implementação de requisitos, como visto na Figura 1. O método escolhido para o desenvolvimento do trabalho foi uma forma derivada do SCRUM (derivada pois os prazos de entrega de cada seção não eram completamente definidos). Para o desenvolvimento deste trabalho foram utilizados os *softwares 3D Studio Max* como ferramenta de modelagem, além da *engine Unity* na tarefa de estruturação do jogo.

A pesquisa realizada para o desenvolvimento deste trabalho possui caráter aplicado, por apresentar um problema identificado, tendo como objetivo a resolução deste através da sua alteração, melhoria ou criação. Marconi (2002) debate que a pesquisa aplicada possui o interesse prático no qual busca solucionar problemas da realidade, sendo assim, os resultados utilizados e aplicados no problema em questão.

Este artigo apresenta como procedimento técnico, a pesquisa bibliográfica e documental, na qual Prodanov e De Freitas (2013) citam que esta procura reunir dados e informações

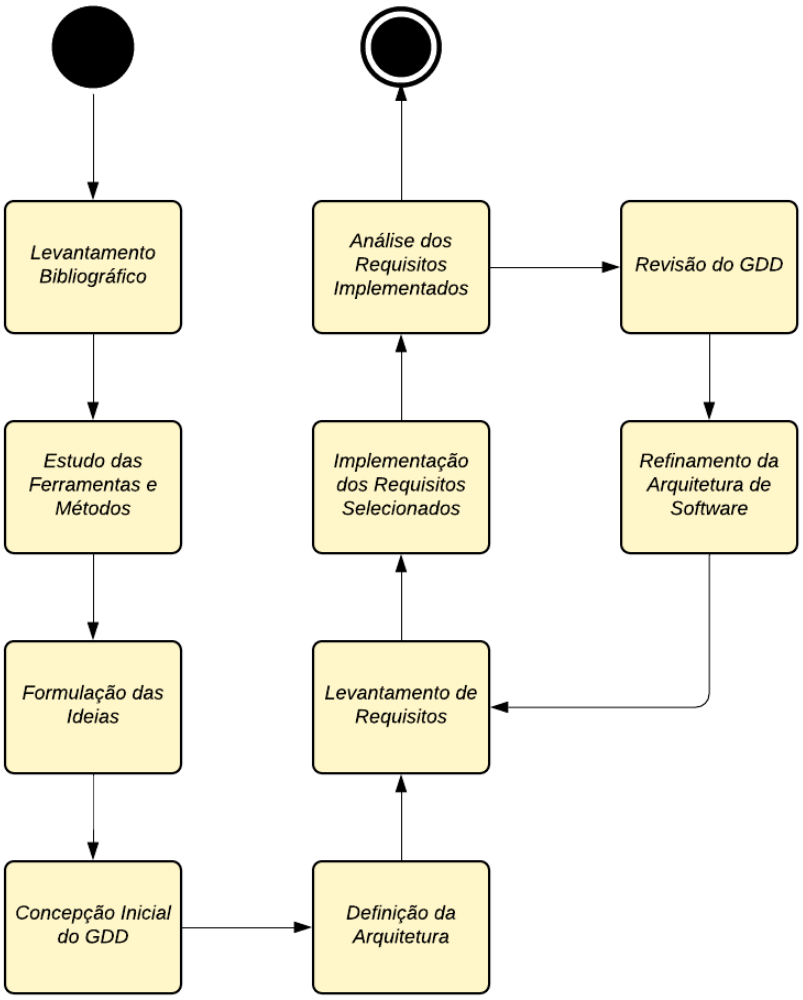

Figura 1. Fluxograma de Desenvolvimento

sobre o estudo em questão, através da análise de materiais relacionados ao tema e a pesquisa documental utiliza de fontes secundárias para a realização do presente estudo.

A forma de abordagem do problema se dá através de uma pesquisa qualitativa, na qual o estudo é realizado sobre informações adquiridas do ambiente. "Os dados coletados nessas pesquisas são descritivos, retratando o maior número possível de elementos existentes na realidade estudada" (Prodanov e De Freitas, 2013).

Por fim, o trabalho apresenta caráter exploratório, por se tratar de uma questão pouco estudada atualmente, com o objetivo de realizar novas descobertas capazes de facilitarem futuros usos de ferramentas com geração procedural.

2.1. Pesquisa e Estudo Bibliográfico

Esta etapa foi realizada inicialmente em junho de 2021, sendo feita até agosto de 2021, durante este período houve uma leitura dos artigos, através de um método denominado revisão sistemática, mostrada na Figura 2. A ferramenta de busca foi o Google Acadêmico, com foco

em estudos do período de 2005 até 2021, procurando sempre buscar informações atualizadas.

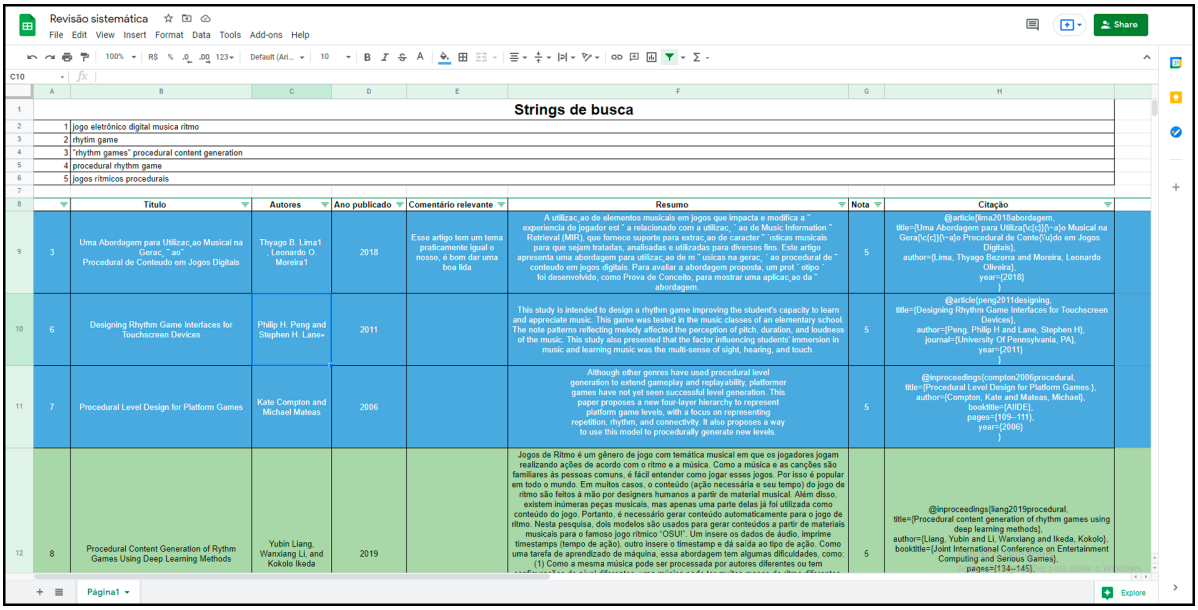

Figura 2. Revisão Sistemática

2.1.1. Revisão Sistemática

Na revisão sistemática foram analisados cerca de 30 artigos, utilizando o Google Acadêmico, com *strings* de busca relacionadas ao tema de geração procedural, desenvolvimento de jogos, jogos rítmicos e música, mostradas das linhas 2 à 6 na Figura 2.

A partir da linha 8, foram escritas as principais informações sobre os trabalhos pesquisados, como título, autores, ano de publicação, resumo e BibTex.

Após a leitura de todos os artigos, foram selecionados os mais relevantes para o tema. Esses artigos foram lidos por completo e, após isso, foram resumidos, sendo 7 deles utilizados na Seção 3 de Referencial Teórico.

2.2. Estudo técnico e prático das ferramentas de desenvolvimento

As ferramentas utilizadas no projeto foram selecionadas com base em conhecimentos prévios dos autores desse projeto, sendo elas: *3DS Max* utilizado na edição de modelos 3D e *Unity* como o motor responsável pela criação da estrutura do jogo. Todos os *softwares* utilizados neste trabalho serão apresentados e descritos na Seção 3.

Estudos dessas ferramentas se deram principalmente por vídeo aulas no *YouTube* e *Udemy*, além de páginas de sites especializados em conteúdos para desenvolvedores, como *Stackoverflow* e fóruns do *Unity*, juntamente com suas próprias documentações.

2.3. Desenvolvimento do Jogo Digital

O Jogo Digital foi desenvolvido seguindo os passos da Figura 1. Em primeiro momento, foram estudadas as ferramentas e métodos de desenvolvimento. O segundo ponto, foi estabelecer os requisitos de software. O terceiro passo foi a prototipagem do jogo.

Tendo os primeiros passos realizados, foi organizado o processo de construção. Por fim, foi desenvolvido o jogo, implementando e testando a cada iteração do ciclo de desenvolvimento baseado em SCRUM.

3. Referencial Teórico

As próximas subseções visam mostrar ao leitor informações sobre as ferramentas utilizadas no desenvolvimento do projeto, bem como outras tecnologias. Essas subsções estão divididas em Processamento de Sinais Musicais, Tecnologias para Desenvolvimento de Jogos Digitais, Unity, 3Ds Max, Adobe Photoshop, C Sharp, GitHub e Trello.

Os parágrafos a seguir mostram os trabalhos correlatos utilizados como referência no desenvolvimento deste trabalho.

3.1. Trabalhos Correlatos

Lima e Moreira (2018) iniciam o artigo dando uma ideia geral de jogos rítmicos, com exemplos de jogos conhecidos e comentam um pouco sobre a definição de Geração Procedural e *Music Information Retrieval* (MIR) (ciência por trás de extrair elementos musicais e transformá-los em objetos). O artigo possui um foco mais teórico, dando ênfase em contextualizações de temas e exemplos, abordando a exploração de uma arquitetura padronizada para obtenção de dados MIR e a utilização desses dados na geração procedural de fases para um jogo Rítmico com tema espacial. Os resultados obtidos mostraram que, quanto maior a velocidade da música (BPM), mais inimigos e obstáculos surgiam na tela para dificultar o jogador.

Peng e Lane (2011) comentam em seu artigo sobre a evolução da interação humana com computadores através de dispositivos *touch screen*. Juntamente com isso, fazem a ligação com jogos rítmicos, falando que seu sucesso funciona através da responsividade dos controles como um todo. Em seus estudos, foram utilizados diferentes dispositivos *touch screen*, com diferentes eficiências, no jogo *Beats2 Prototypes*, como forma de teste de jogabilidade. A abordagem utilizada foi de testar vários jogos rítmicos do mercado, como forma de categorizá-los referente a suas interfaces. Após isso, desenvolveram um jogo, implementando 8 modos de jogabilidade diferentes, baseados nos jogos testados, e com isso, receberam *feedbacks* de *testers* sobre qual jogabilidade era mais intuitiva, divertida, única, entre outros. Após o período de testes, foram anotados os resultados, indicando qual jogabilidade entregou a melhor experiência de usuário.

Compton e Mateas (2006) iniciam comentando sobre a inovação no mercado de jogos através da geração procedural de fases. Com o jogo *Rogue*, os autores comentam sobre a evolução no desenvolvimento, mostrando que o jogo possui fases ilimitadas que são geradas automaticamente, e que, apesar das fases geradas não serem tão complexas quanto criadas por humanos, a inovação da criação sem supervisão humana é grande. Nesse trabalho, é citada

a dificuldade em criar algoritmos de geração procedural, visto que qualquer *bug* pode tornar uma fase difícil em uma fase impossível de ser completada. Sobre jogos rítmicos, os autores ainda comentam que não dependem apenas de cálculo de distância de pulos, por exemplo, mas também do senso de ritmo do jogador, o *flow*. O resto do trabalho mostra o desenvolvimento de um algoritmo inteligente de geração procedural.

Webster e Mendes (2014) iniciam seu artigo falando um pouco sobre a origem de jogos ritmicos, citando o clássico jogo *Simon*, um brinquedo com quatro botões coloridos que piscam em ordem aleatória, onde o jogador deve pressionar os botões na mesma ordem para vencer. Após citarem o sucesso e evolução de jogos deste tipo, chegam à seu tópico principal, onde falam sobre o jogo *DJ Hero*, um jogo ritmico em que o controle possui formato de uma mesa de *DJ*, com a proposta de serem tocadas músicas eletrônicas famosas no ritmo para acumular pontos. O artigo comenta, principalmente, sobre como o cenário todo é montado de forma a aumentar ao máximo a imersão do jogador, utilizando músicos famosos como personagens. A conclusão do artigo fala mais sobre como jogos ritmicos influenciam no comportamento do jogador, de forma divertida e envolvente.

Tomczak (2005) inicia seu trabalho falando sobre o que é uma batida e sobre os algoritmos para determinar a batida de uma música em tempo real e sobre os resultados variados em diferentes tipos de músicas, algo que, em geral, é procurado em picos quase periódicos de um determinado recurso. Em seguida, os autores falam sobre como a batida pode ser melhorada, considerando as informações múltiplas da música, construindo uma biblioteca de dois ou mais detectores de batidas com uma interface em comum, a fim de construir um conceito geral de música e melhorar a precisão de detecção. Por fim, os autores falam do quão valiosa a biblioteca pode ser para trabalhos futuros, da implementação de mais detectores, aumentar as funções de fábrica já existentes com a capacidade de consumir descrições serializadas e também sobre o grande obstáculo no projeto de detectores de batimento precisos que é a eficiência computacional.

Oliveira (2015) inicia seu artigo explicando sobre a dificuldade de que as preferências e emoções de cada jogador podem diferir muito umas das outras quando se trata do mesmo jogo, sobre as vantagens da geração procedural de conteúdo e sobre o quão importante é a música para a imersão do jogador. Após, o autor segue comentando de como um conteúdo gerado em base da música pode ser diversificado, melhorar cada estilo de jogo e as vontades do jogador. A abordagem utiliza duas *APIs* principais, *Meapsoft API1* e *The Echo Nest API2* utilizadas em sua versão *jEN3* do cliente java, que permitem que o usuário escolha sua própria música e vá passo a passo até que o jogo esteja pronto para ser tocado com aquela música. Além disso, também permite ao usuário utilizar músicas já analisadas anteriormente. A conclusão do artigo responde a quatro perguntas: 1. É possível gerar um jogo furtivo proceduralmente apenas com base em parâmetros de entrada? 2. Quais são os recursos musicais mais importantes a serem usados na geração de conteúdo do jogo? 3. Quão viável é o uso de recursos musicais na geração de conteúdo do jogo? 4. É possível alterar o estilo de jogo com base no conteúdo criado? E como trabalhos futuros, sugere a melhoria na renderização, melhoria no processo de segmentação, percepção da batida e harmonização com os ouvidos do jogador.

Lyra e de Jesus (2014) comentam em seu artigo sobre a criação de um jogo musical para plataformas móveis com o objetivo de auxiliar o aprendizado em exercícios de

execução rítmica com partituras e sequências sonoras que devem ser reproduzidas utilizando-se da execução de sons ritmados, capturados através de um microfone ou como alternativa, a utilização de toques na tela. Criado para dispositivos móveis, por ser uma plataforma que, normalmente, já tem um microfone e por conta de sua mobilidade. Falam sobre alguns conceitos, música, gêneros e o gênero *runner*. Segue apresentando trabalhos de criações de jogos similares. Após, apresenta as ferramentas utilizadas e a arquitetura do *game*. Por fim, discutem sobre os testes realizados e concluem sobre a importância da *framework Unity*, sobre como correu o desenvolvimento e sugerindo a melhoria na captura de ritmo e melodia como trabalho futuro.

3.2. Processamento de Sinais Musicais

Para comentar sobre processamentos de sinais musicais, primeiro deve-se entender o que é *MIR*. *MIR* é a sigla para *Music Information Retrieval*, que é a ciência que retira informações de músicas, tendo aplicações para musicistas, psicólogos, escolas, processamento de sinais, *machine learning*, informática, entre outros (Abreu, 2021).

A *MIR* possui várias classificações, uma delas chamada de *Automatic Music Transcription*, cuja função é converter áudio em notações simbólicas, como *MIDI Files* (acrônimo para *Musical Instrument Digital Interface*, resumidamente, um padrão musical que usa pulsos elétricos para tocar, gravar e editar músicas), ou *MP3*, como mencionado por Herrera-Boyer et al. (2006). Essas transcrições analisam vários pontos do áudio, como *pitch*, duração, identificação do instrumento, ritmo, melodia, velocidade, etc.

Algumas das principais informações extraídas através do *MIR* são *Mel-Frequency Cepstral Coefficient* (MFCC), que são a medida do timbre, além de outras como compassos, acordes, melodia, bpm, entre outros (Moffat et al., 2015), tendo grande importância em aplicações como criação procedural de músicas, com algum sucesso no que se diz respeito a apreciação dos resultados por outros humanos.

3.3. Tecnologias para Desenvolvimento de Jogos Digitais

Nesta subseção, serão apresentadas as ferramentas a serem utilizadas no desenvolvimento do jogo, todas possuindo licença gratuita ou de estudante para utilização.

3.3.1. Unity

A *Unity* é uma *engine* para o desenvolvimento de aplicações 2D ou 3D, desenvolvida pela *Unity Technologies*. Ela possui diversos módulos que garantem a efetividade e agilidade do processo de desenvolvimento.

A *Unity* conta com a presença de *game objects*, que são os objetos presente na cena. Imagine a cena como uma fase de um jogo, nessa fase devem ser adicionados diversos itens, como os obstáculos, personagens, câmera do jogo entre outros. Esses itens adicionados são os *game objects*.

Cada objeto possui ações diferentes, podendo emitir sons, andar, rotacionar, dependendo do componente adicionado. Por exemplo, o componente *transform* é responsável pelas transformações desse objeto, contando com atributos de posição, rotação, escala, e cada

modificação nesse item implicará diretamente no objeto. Além dos componentes padrões, é possível adicionar *scripts*, que são usados para a criação de efeitos gráficos, controle de comportamento físico de objetos ou outras funções personalizadas.

Com o Unity é possível, com pouca experiência, desenvolver jogos, graças aos módulos apresentados pela desenvolvedora, e também ao conceito de arquitetura baseada em *game assets* (componentes).

3.3.2. 3Ds Max

O *3Ds Max*, originalmente chamado por *3D Studio Max*, foi desenvolvido pela *Autodesk* e assim como outros *softwares* de modelagem tridimensional, foi desenvolvido com o intuito de auxiliar modeladores que iniciavam todo projeto do zero, sendo uma das principais ferramentas para a criação de renderizações e animações (Almeida, 2007).

Por meio desses *softwares*, a modelagem passou a ser feita com a modificação de polígonos, arestas e vértices, tendo como ponto inicial objetos primitivos, como cubos, esferas ou cilindros, onde neles são feitas as alterações e aplicações de modificadores disponíveis até a obtenção do modelo desejado. *Polycount* é o número de polígonos de um objeto, sendo *highpoly* e *lowpoly* quando possui muitos e poucos polígonos, respectivamente.

3.3.3. Adobe Photoshop

O *Adobe Photoshop* é um *software* de criação e edição de imagens bidimensionais rasterizadas. É considerado o líder no mercado de editores de imagem profissionais, sendo utilizado por fotógrafos, designers, profissionais da *web* e de vídeo. Para este trabalho, o *Photoshop* foi utilizado na parte gráfica da interação com o usuário, com a criação das telas do jogo, como menu inicial, tela de configurações, *HUD* (informações que aparecem na tela dentro do jogo, como pontuação, por exemplo), botões, entre outros.

3.3.4. C Sharp

C Sharp é uma linguagem de programação, multiparadigma, de tipagem forte, desenvolvida pela *Microsoft* como parte da plataforma *.NET*, com sintaxe orientada a objetos baseada em *C++*.

A linguagem de programação utilizada no *software Unity* é *C Sharp*, pois possui uma melhor curva de aprendizado, além do histórico de parcerias da *Microsoft* com a *Unity*. *Scripts* em *C Sharp* são os arquivos de código que guardam comportamentos de objetos no *Unity*, sendo eles os responsáveis pelo total funcionamento dessa *engine*. Mesmo com a existência de ferramentas que permitem a criação no *Unity* sem esses *scripts*, eles ainda são a melhor forma de criar ações e interações customizadas dentro do jogo.

3.3.5. GitHub

GitHub é uma plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou *Open Source* de qualquer lugar do mundo. Foi usado como repositório para esse projeto, sendo utilizado principalmente pelos recursos de edição simultânea, versionamento e histórico de alteração.

3.3.6. Trello

Trello é um aplicativo de gerenciamento de projeto *online*. Funciona principalmente como um Kanban, organizando as atividades em cartões e filas. Neste projeto, foi utilizado para ter controle das tarefas através das filas ”A Fazer”, ”Em Andamento”e ”Concluído”.

4. Desenvolvimento

O desenvolvimento se iniciou com a formulação de ideias, sendo um de seus passos a criação dos *Concept Arts* (artes conceituais) de cenários, pontuação, *HUD* (informações da tela), obstáculos e veículos, além do funcionamento geral do jogo, como visto nas Figuras 3 e 4.

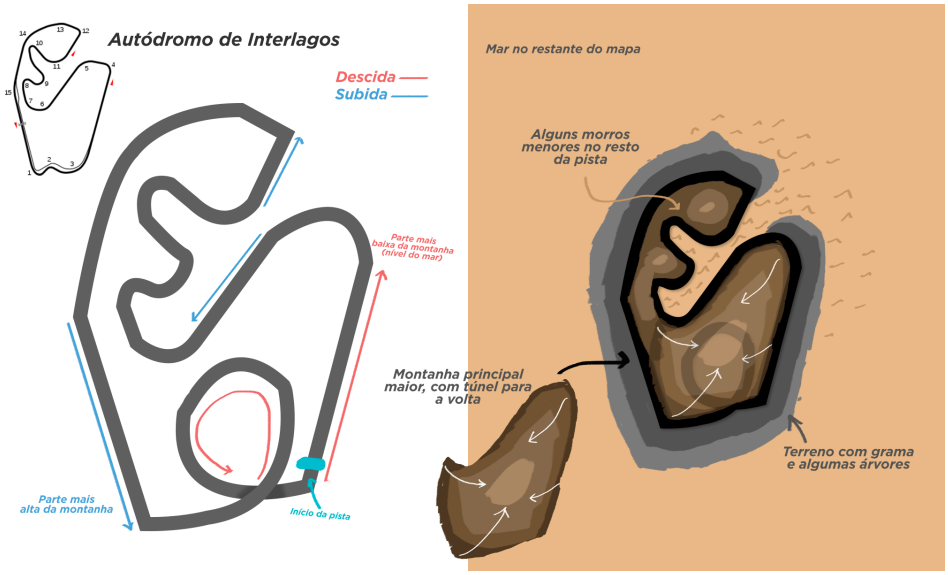

Figura 3. Concept Art - Pista

Depois disso, os modelos 3D dos veículos e pistas foram baixados da *Unity Store* e adaptados com o uso do *3DS Max*, possibilitando assim a criação de um cenário protótipo inicial. Logo após, com o uso da *game engine Unity* foram realizados testes de física dos veículos e a criação de componentes de *script* que utilizam Inteligência Artificial para o movimento dos veículos. O movimento do veículo será explicado na seção 4.4.3.

Com o veículo pondendo movimentar-se na pista de protótipo, o próximo passo foi melhorar e dar vida ao cenário, como adicionar texturas, povoar com vegetação, iluminação, névoa, entre outros. O cenário desenvolvido após esta etapa pode ser observado na Figura 5.

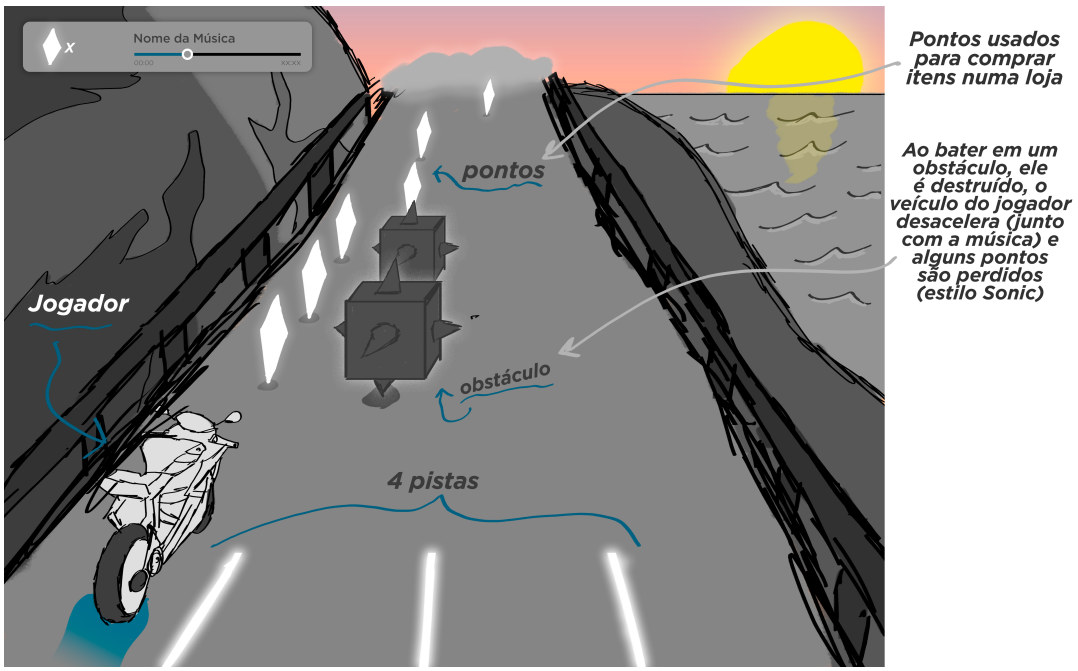

Figura 4. Concept Art - Hud Obstáculos

Com o avanço no desenvolvimento do cenário, a próxima etapa se concentrou nas telas de Menu Inicial, HUD (Informações da tela), entre outras. Essas telas foram criadas no *software Adobe Photoshop* e foram importadas para dentro do *Unity* na forma de texturas. Com a parte gráfica das telas montadas, foi feita a programação para seu funcionamento, utilizando transição entre cenas com a linguagem *C Sharp*. A tela de Menu Inicial pode ser observada na Figura 6.

Em paralelo ao desenvolvimento da tela de Menu Inicial, foram construídos os 4 caminhos possíveis na pista, sendo eles um em cada faixa, através de pontos invisíveis utilizando o *Unity*. Logo após, foi implementado um mecanismo para que o veículo seguisse automaticamente pontos na pista através de um *script* em *C Sharp*. Tal *script* foi programado de forma que a única interação que o jogador realiza é o pressionar das teclas Esquerda e Direita no teclado para alternar entre as faixas. O mapeamento dos pontos, bem como o *script* de movimento dos veículos pode ser observado na Figura 7.

Todas as etapas do desenvolvimento ocorreram no período de Outubro de 2021 até Fevereiro de 2022. As etapas iniciais de criação de terrenos, texturas e pista levaram cerca de 1 mês para serem finalizadas. Após isso, os componentes de *scripts* básicos de menu, movimento dos veículos e câmera levaram 1 mês e meio.

Com os componentes de *script* básicos finalizados, o mês de Janeiro foi inteiramente focado na criação de *scripts* mais complexos, como *Beat Detection*, geração de notas e obstáculos, *download* automático das músicas com o uso de *weblinks*, finalização dos *Pathfindings* (gatilhos que, quando o veículo os toca, o algoritmo faz com que ele seja mandado para o próximo, e assim por diante, fazendo com que o mesmo va se locomovendo para frente) e otimização geral. Todos esses componentes de *script* serão explicados nas próximas seções.

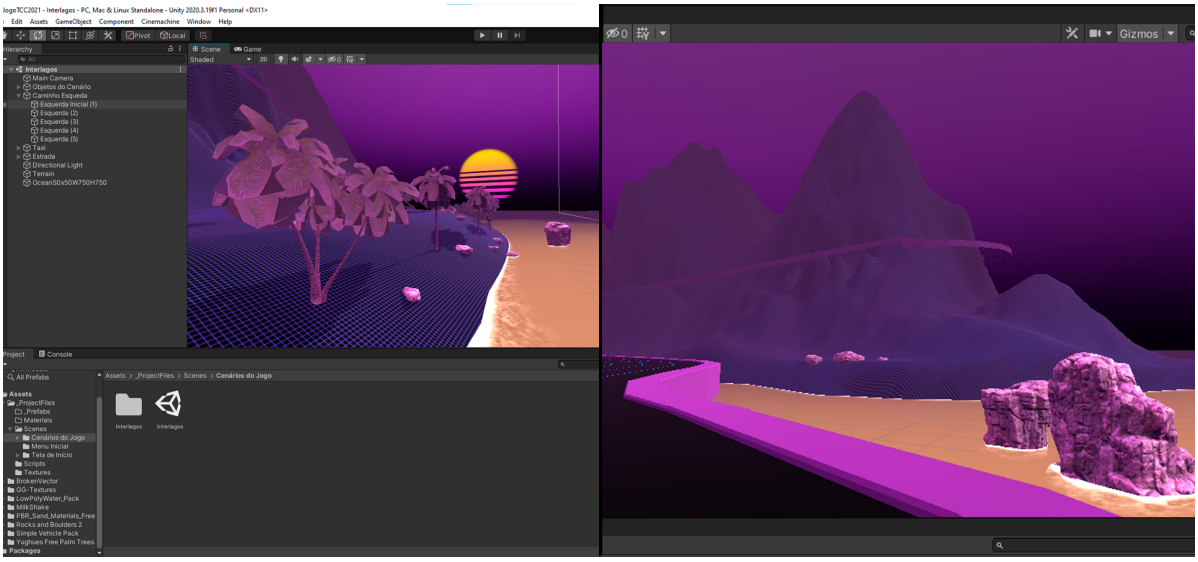

Figura 5. Cenários do jogo

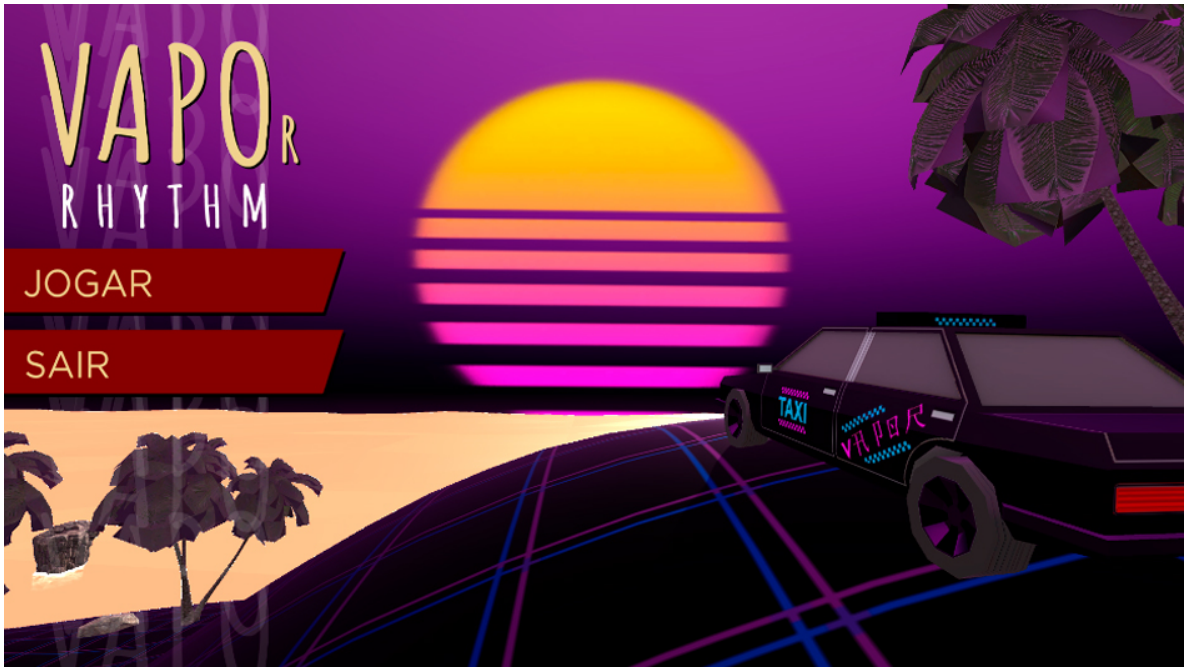

Figura 6. VaporRhythm (Referência a Vaporwave e Ritmo)

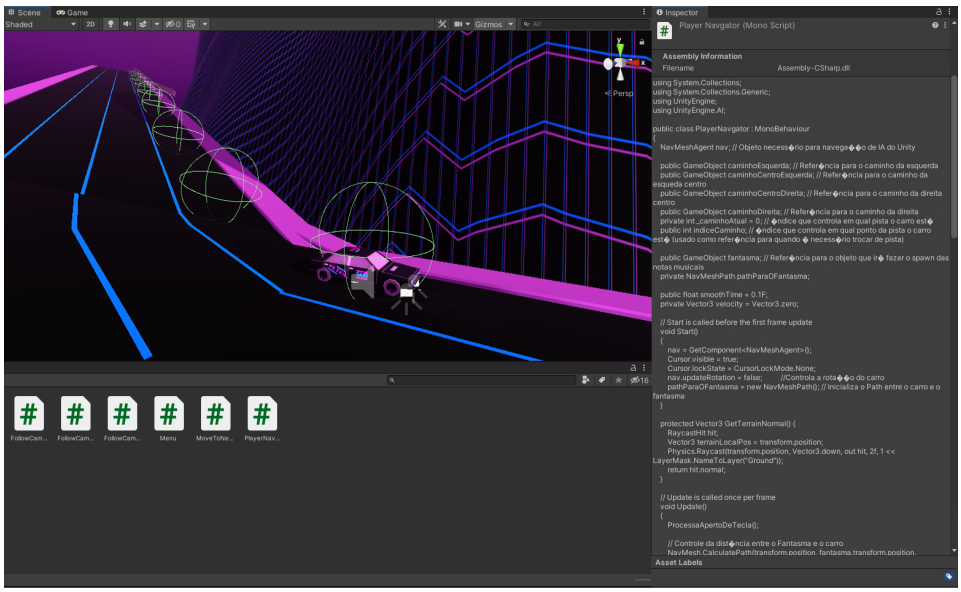

Figura 7. PlayerNavigator e Pathfindings

4.1. Requisitos do Jogo Digital

Os requisitos do jogo foram todos descritos em um arquivo na ferramenta *Google Drive*, chamado "Diário de Bordo". No Diário de Bordo, estão listados todos os passos a serem seguidos para a realização tanto do jogo quanto do artigo. Os requisitos foram, inicialmente, levantados e depois separados em *Sprints*, seguindo um modelo de desenvolvimento ágil baseado em *SCRUM*.

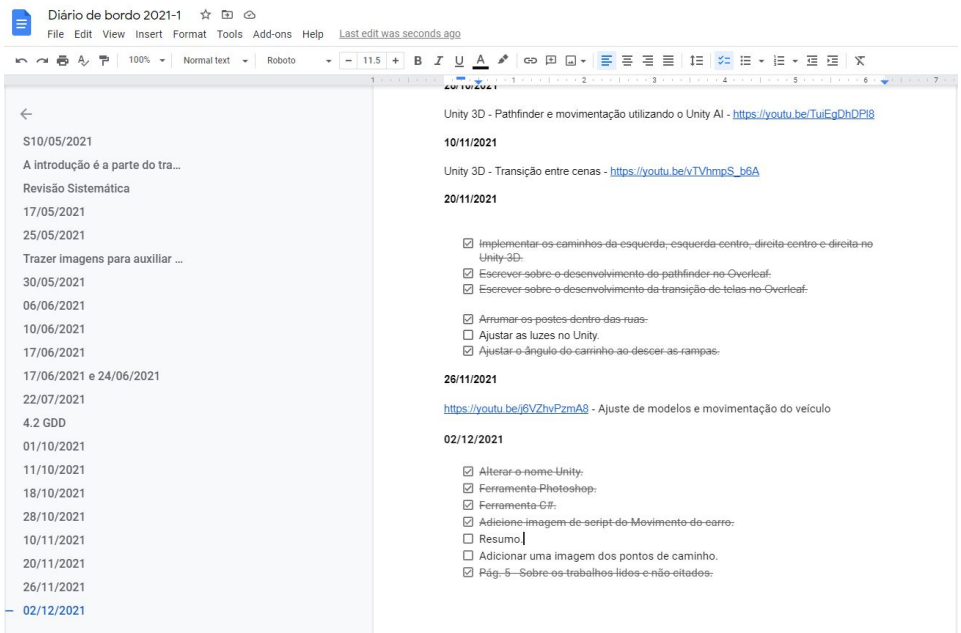

Figura 8. Diário de Bordo

O início do desenvolvimento dentro do Diário de Bordo é listado a partir do dia 01/10/2021. Desta data em diante, os requisitos para o jogo digital são descritos, como implementação dos caminhos, criação e ajuste de texturas, movimentação do veículo, entre outros. O Diário de Bordo pode ser observado na Figura 8.

4.2. Kanban de Produção

O kanban feito para o desenvolvimento do jogo utilizou o site Trello, sendo organizado em cartões divididos em ”A Fazer”, ”Em Andamento”e ”Concluídos”. Com a utilização desse sistema, foi possível uma maior organização nas tarefas de cada membro da equipe, de forma a manter todo o processo mais eficiente e dinamizado. O Kanban de Produção pode ser visto na Figura 9.

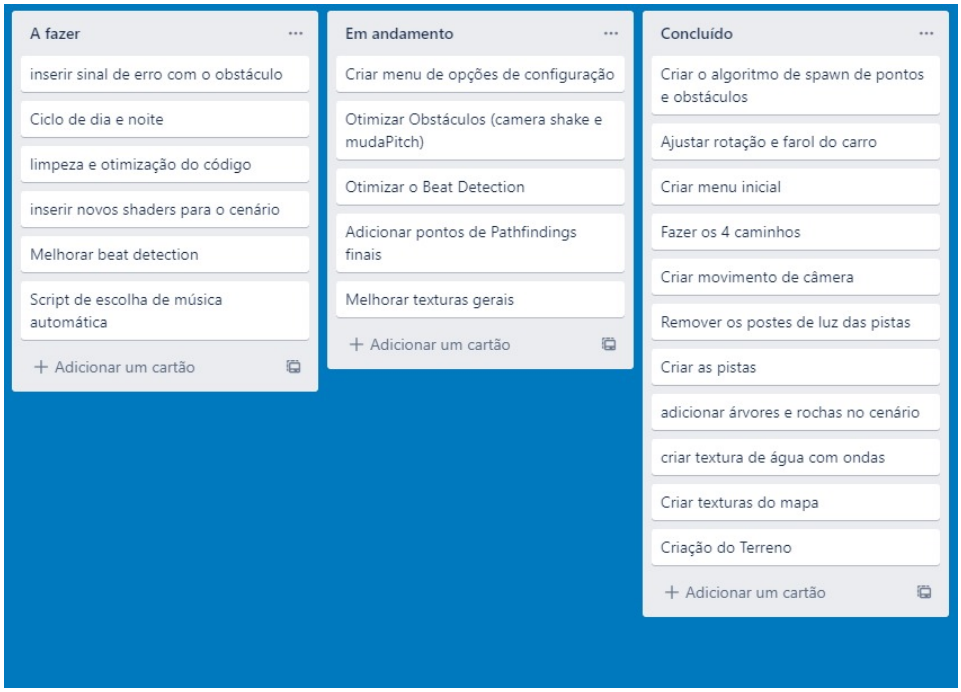

Figura 9. Kanban de Produção no Trello

4.2.1. Controle de Versões

O trabalho foi realizado com o auxílio da ferramente *Github Desktop*. Nela, os arquivos do *Unity* estavam conectados com os três computadores que trabalharam no jogo e, a cada atualização feita, um *Commit* era realizado para salvar as alterações em todos os outros computadores. Foram realizados 85 *commits*, no período de 40 dias (até a data do dia 31/01/2022).

A cada *Commit*, uma nova atualização era adicionada ao jogo, iniciando o ciclo de desenvolvimento a partir da criação do cenário e adição de texturas, até a finalização do *Beat Detection*, geração de pontos funcional, limpeza e otimização do código.

4.3. Arquitetura de Software

Na *game engine Unity*, existem componentes já programados com funções específicas, como é o caso de Colisores, Câmeras, *AudioSources*, entre outros. Esses componentes foram todos utilizados em conjunto a componentes programáveis, como *scripts* de movimentação do veículo, captação de pontos, gerador de notas, *triggers*, entre outros. Os componentes foram adicionados aos *Game Objects* criados, como é o caso do *Player*, Fantasma, *Pathfindings*, e outros, para dar o funcionamento correto à eles. A descrição de todos os componentes de *scripts* criados será feita nas próximas seções e o modelo de arquitetura de *software* utilizado pode ser visto na Figura 10.

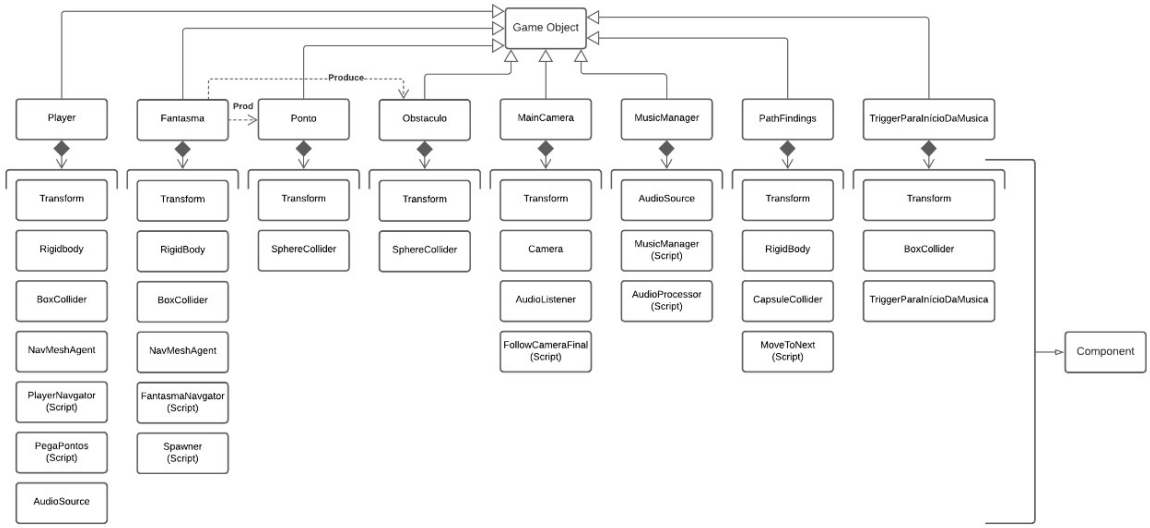

Figura 10. Arquitetura de Software

4.4. GDD

O *Game Design Document* descreve detalhadamente o processo de desenvolvimento do *software*, mostrando áreas da arte, programação, *design*, entre outras, e quais foram os esforços utilizados para cada uma dessas. Após a formulação das ideias, o GDD teve início, evoluindo juntamente com o artigo e o jogo.

Por se tratar de um ambiente dinâmico, o documento é muitas vezes alterado, atualizado, revisado e expandido, podendo ser chamado de *Living Document* (Documento Vivo), como é dito por Kitchin (2003).

O GDD para este artigo ficou organizado em subseções, cada uma explicando de forma detalhada o desenvolvimento e funcionamento de uma parte do *software*, sendo dividido em *Gameplay*, Personagens, Controles, Câmera, Interface, Trilha Sonora, Fantasma, Análise do Espectro da Música e Pontos e Obstáculos, respectivamente. As seções a seguir irão informar melhor o leitor sobre como o jogo foi desenvolvido.

4.4.1. Gameplay

A *gameplay* do jogo se baseia em jogos como *Guitar Hero* (Activision, 2005) e *Music Racer* (Abstract Art, 2018), onde um veículo é disposto em uma pista de corrida. A pista é dividida em 4 faixas, todas possuindo modelos esféricos invisíveis numerados, chamados de *Pathfindings*, que são gatilhos responsáveis por fazer o veículo se mover, sendo adicionados de forma manual na pista. Quando o jogador colide com um *pathfinding*, um contador aumenta, fazendo com que o veículo se mova para o *pathfinding* seguinte. Cada uma das 4 pistas possui o mesmo número de *Pathfindings* e, quando o jogador aperta as setas direita ou esquerda do teclado, o algoritmo muda a trajetória para o *Pathfinding* da próxima pista, fazendo com que o carro mude de faixa. Os Pathfindings podem ser vistos na Figura 7.

Esse movimento horizontal do veículo é essencial para que o jogador possa pegar os pontos que aparecem na pista assim como desviar de obstáculos que fazem o jogador perder pontos. Os pontos e os obstáculos aparecem na pista no ritmo da música tocada de fundo, através de um algoritmo inteligente que detecta os tons e *beats* principais da música e os transforma em sinais que são detectados pelo algoritmo de *Beat Detection* e, então, são transformados em objetos do jogo. O algoritmo de *Beat Detection* será apresentado e explicado na seção 4.4.8.

4.4.2. Personagens

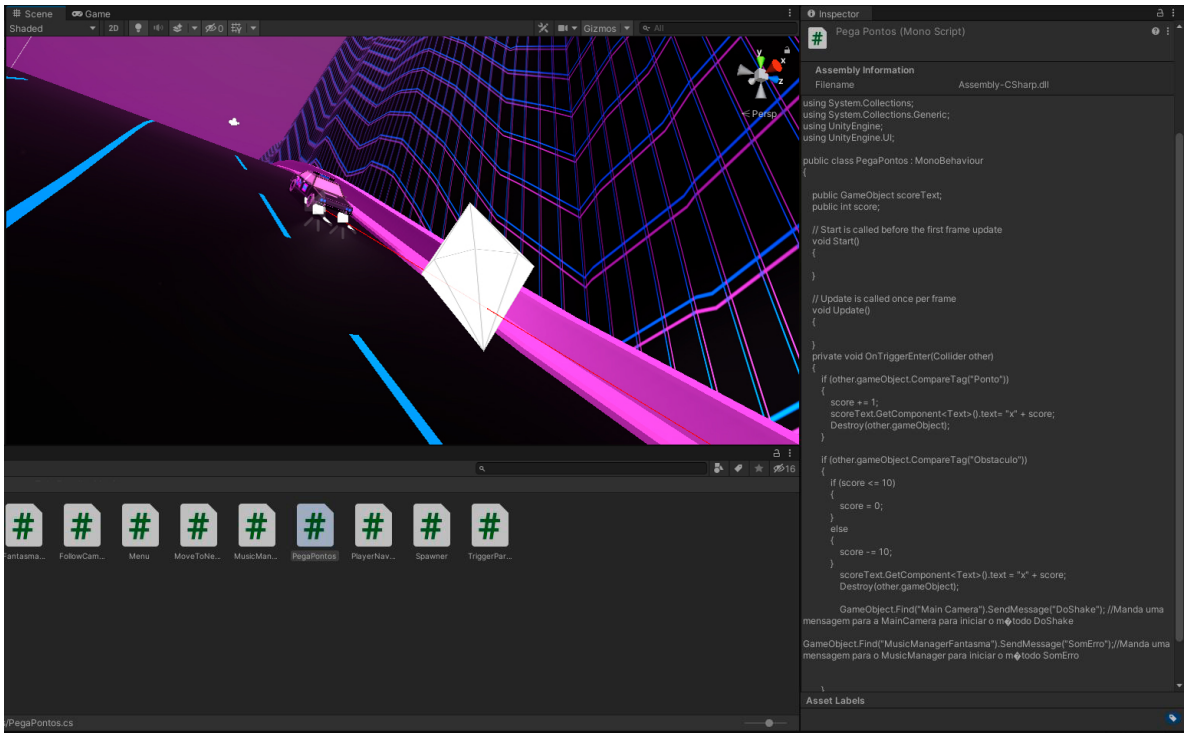

Figura 11. Player e PegaPontos

O personagem controlado pelo jogador se trata de um veículo. Esse que possui as

funções de andar pela pista, assim como trocar de faixa, com o objetivo de pegar pontos. O veículo utilizado pelo jogador possui 2 componentes, um sendo o *PlayerNavigator*, responsável por fazer a troca de faixas e detecção de *Pathfindings*, como mencionado anteriormente. O outro, PegaPontos, é resposável por detectar a colisão do jogador com os pontos e obstáculos espalhados pela pista, fazendo com que ele acumule pontos ao pegá-los ou perder pontos caso bata em um obstáculo. Personagem e componentes de *script* podem ser visualizados na Figura 11.

4.4.3. Controles

Os controles do jogo são simples, tratando-se apenas do uso do mouse e setas do teclado. O componente responsável pela função de movimento do veículo é o *PlayerNavigator*. Esse algoritmo inicia-se criando referências pra cada uma das 4 pistas, além de indicar a velocidade do veículo e sua rotação inicial. Com o andamento do jogo, a rotação vai sendo variada dependendo da inclinação da pista e a velocidade vai sendo alterada conforme a distância do ponto mais afastado do jogador, sendo esses os movimentos básicos. Controles do jogo podem ser visualizados na Figura 7.

Seguindo para a troca de pistas, cada um delas é considerada um objeto diferente. As pistas são enumeradas de 0 à 3, indo da esquerda para direita. O jogo inicia com o veículo na pista 0. Nesse momento, a pista 0 está ativada e as outras 3 pistas estão desativadas, assim como seus *Pathfindings*. A partir do momento que o jogador pressiona as setas do teclado, o número da pista é aumentado ou diminuído, fazendo com que o veículo mude sua trajetória para o *Pathfinding* da pista desejada, desativando a pista anterior e ativando a nova.

4.4.4. Câmera

A câmera do jogo é fixada atrás do veículo, através de variáveis X e Y, que são posições referentes ao mundo 3D. Ela possui uma referência ao objeto *Target* (alvo) e estará sempre apontando para ele (no jogo, o *Target* é o veículo). A câmera viaja pelo cenário atrás do veículo respeitando sua velocidade e rotação.

A câmera possui ainda, dentro de seu componente de *script*, um algoritmo chamado *Camera Shake* (balançar da câmera). Esse algoritmo faz a câmera tremer quando o jogador bate em um obstáculo, como forma de ”punição”. O *script* de camera shake faz com que as variáveis X, Y e rotação mudem aleatoriamente durante alguns milésimos de segundo, servindo como informação ao jogador de que ele não conseguiu desviar de um obstáculo a tempo. *Script* de Câmera pode ser visualizado na Figura 12.

4.4.5. Interface

Quando o jogo é aberto, é apresentada uma tela de Menu, com opções de Iniciar o Jogo, Opções de Configuração e botão para Sair. Ao clicar no botão Jogar, o jogo é iniciado, com o veículo do jogador sendo mostrado na tela começando seu movimento. No botão Sair, o jogo

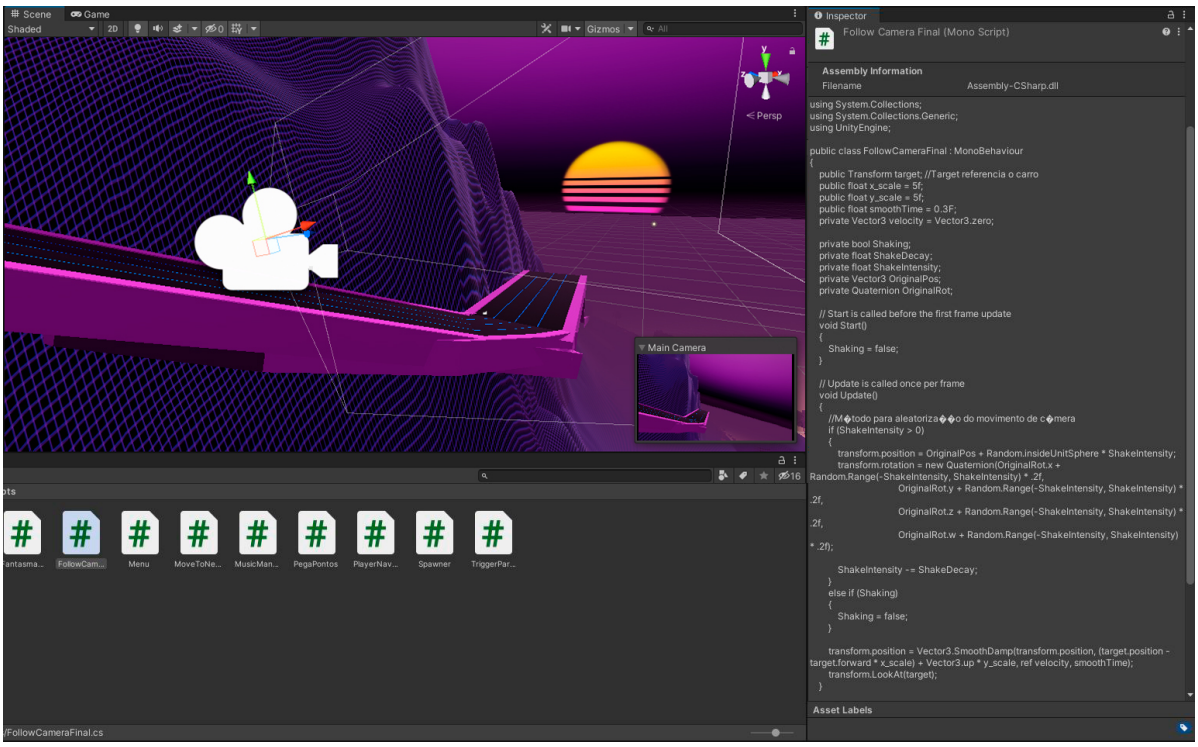

Figura 12. Script de Câmera

é fechado.

Dentro do jogo, a tela possui um *HUD*, que apresenta a informação de pontuação do jogador, que é alterada conforme a interação com os pontos e obstáculos da pista. A tela de Início do jogo pode ser vista na Figura 6.

4.4.6. Trilha Sonora

A trilha sonora do jogo se baseia em músicas escolhidas pelo próprio jogador. O componente de *script MusicManager* é o responsável por baixar a música de um servidor *web* através de um *link* que pode ser alterado dentro do jogo após a tela de Menu Inicial. Esse *link* é gerado através de uma url do Youtube, que é copiada e colada em um *site* de conversão de vídeo para MP3. Após o *download* ser finalizado, ele é transformado em um *AudioSource*, que é um componente de áudio, conectado a um objeto no jogo dentro do *Unity*. O áudio então é tocado através do comando *audioSource.Play()*.

Para que a música seja analisada pelo *Beat Detection*, outra função do Unity, a *audioSource.GetSpectrumData()*, é utilizada. Essa função retorna os dados de espectro da música, como volume, canais, *hertz*, entre outros. Esses dados são utilizados no algoritmo de *Beat Detection* para que a detecção em tempo real ocorra. O algoritmo de *Beat Detection* será apresentado e explicado na seção 4.4.8. O *script MusicManager* pode ser visto na Figura 13.

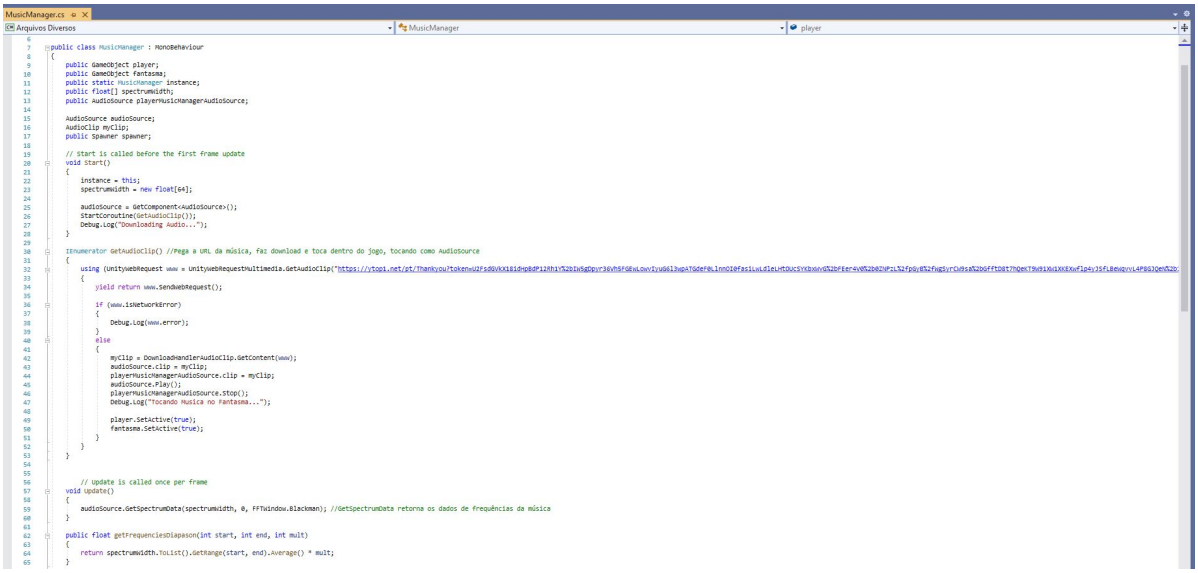

Figura 13. MusicManager

4.4.7. Fantasma

A forma encontrada para maior eficiência na geração de pontos foi a criação de um veículo igual ao do jogador, porém invisível, que segue a uma certa distância à frente e se move entre as faixas de forma independente, seguindo os *Pathfindings*. Esse veículo é chamado de Fantasma.

A função do Fantasma é de ser o responsável por gerar os pontos e obstáculos nas faixas e coordenadas corretas através do script *Spawner*, que será explicado na seção 4.4.9. Esse objeto possui o componente de *script FantasmaNavigator*, que funciona de maneira parecida ao *PlayerNavigator* comentado anteriormente, porém, com a diferença de não se mover através das setas do teclado, mas sim, de acordo com o tom da música. A função de troca de faixas de acordo com o tom da música será melhor explicada na seção 4.4.8. O Fantasma, assim como seu componente de *script* podem ser visualizados na Figura 14.

O *MusicManager*, mencionado na seção 4.4.6 controla o *AudioSource* do *Player* e do Fantasma, sendo usado tanto na análise do espectro para o *Beat Detection* quanto como música de fundo, porém, o Fantasma não pode analisar a música ao mesmo tempo que ela é tocada para o jogador pois os dois estão separados por uma certa distância e, se a música do jogador e a música do fantasma tocassem ao mesmo tempo, as notas geradas no Fantasma apareceriam antes do jogador ouvi-lás, tirando o ritmo do jogo. Para a resolução desse problema, foi implementado o *script TriggerParaInicioDaMusica*.

Esse *script* é conectado a um objeto invisível *Trigger* localizado ao lado do Fantasma no início do jogo. Quando o jogo se inicia, o Fantasma começa a tocar a música sem volume nenhum, servindo apenas para a detecção de *beats*. Quando o jogador começa a andar, ele toca no objeto *Trigger*, ativando o componente de *script TriggerParaInicioDaMusica*, fazendo com

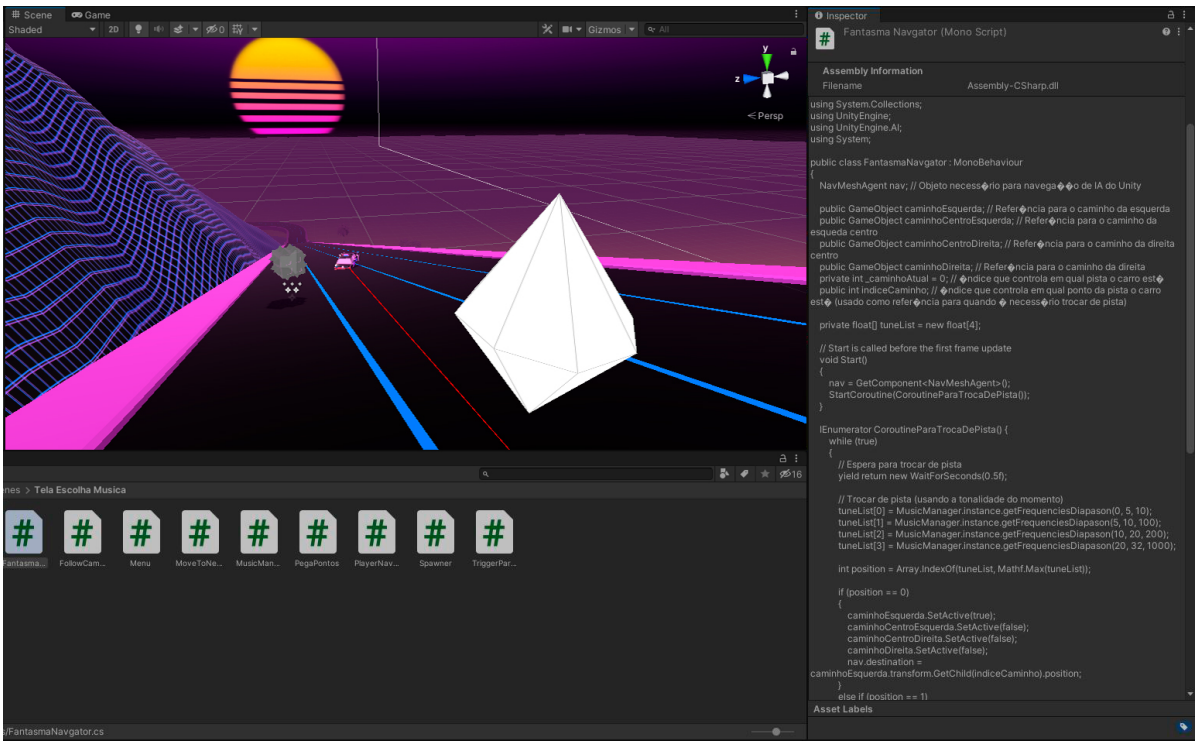

Figura 14. Fantasma

que a música comece a tocar de fato para que o jogador possa ouvi-lá no ritmo certo.

4.4.8. Análise do Espectro da Música

A análise do espectro da música é a parte fundamental do jogo. Ela é responsável pelo funcionamento da geração dos pontos na pista. Essa análise em tempo real retorna dois tipos de dados, sendo primeiro o dado de tom da música.

O tom da música é dividido em grave, médio-grave, médio-agudo e agudo. Cada tom é respectivo a uma das 4 pistas da esquerda para direita. Dentro do jogo, o *script* de detecção de tom roda a cada meio segundo. Esse script é conectado ao objeto Fantasma, mencionado anteriormente na seção 4.4.7. Os pontos e obstáculos são gerados em cima desse objeto Fanstasma. Durante meio segundo, o fantasma andará em uma certa pista e, após isso, o algoritmo detecta o tom mais alto da música naquele instante e, dependendo de seu valor, o fantasma troca para a pista referente ao tom. Resumidamente, a detecção de tons mais graves faz com que o Fantasma se locomova na pista mais à esquerda e tons mais agudos, à direita.

O segundo dado retornado pela análise de espectro é o *Beat Detection*. O algoritmo de *Beat Detection* é o principal responsável pela geração dos pontos. Através da análise do espectro, ele é capaz de, em tempo real, detectar os beats principais da música, os transformando em sinais que são enviados para o *script* de geração das notas *Spawner*, fazendo com que elas apareçam a cada *beat*, ignorando *beats* mais fracos de acordo com a sensibilidade da

detecção, que pode ser alterada. O algoritmo de *Beat Detection* pode ser observado na Figura 15.

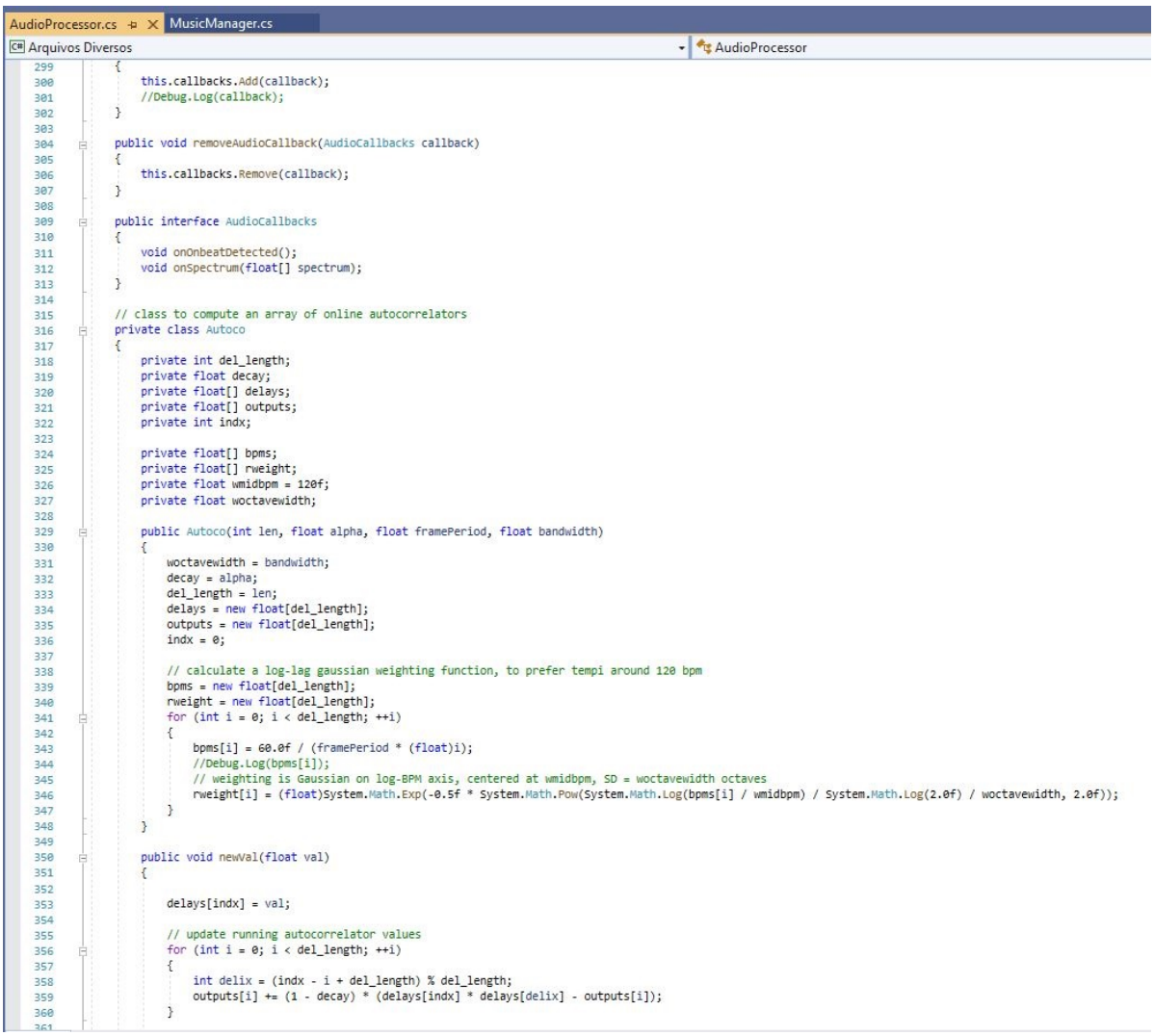

Figura 15. Algoritmo de Beat Detection

4.4.9. Pontos e Obstáculos

Os pontos e obstáculos são gerados pelo *script Spawner*, que é conectado a um carro invisível chamado Fantasma, mencionado na seção 4.4.7.

Esse script é conectado ao *BeatDetection* que, a cada *beat* principal da música, manda um sinal ao *Spawner* para que esse gere um objeto chamado Ponto. Os pontos coletados pelo jogador são somados no canto superior esquerdo da tela.

Além dos pontos, são gerados também obstáculos. Ao contrário dos pontos, esses fazem o jogador perder 5 pontos acumulados caso entrem em contato, além de ativar o *script*

de *CameraShake*, mencionado anteriormente, de forma a avisar o jogador que ele cometeu um erro. O *script* de geração é observado na Figura 16.

Spawner.cs

Arquivos Diversos — Spawner

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;
using System;

public class Spawner : MonoBehaviour, AudioProcessor.AudioCallbacks
{

    public GameObject ponto;
    public GameObject obstaculo;

    // Start is called before the first frame update
    void Start()
    {
        AudioProcessor processor = FindObjectOfType<AudioProcessor>();
        processor.addAudioCallback(this);
    }

    public void onOnbeatDetected()
    {
        float num = UnityEngine.Random.Range(1, 10);
        if (num <= 2)
        {
            SpawnObstaculo();
        }
        else
        {
            SpawnPonto();
        }

    }

    public void onSpectrum(float[] spectrum)
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    public void SpawnPonto()
    {
        Instantiate(ponto, transform.position, transform.rotation);
    }

    public void SpawnObstaculo()
    {
        Instantiate(obstaculo, transform.position, transform.rotation);
    }
}
```

Figura 16. Spawner de Pontos e Obstáculos

5. Resultados e Discussões

O resultado final do jogo foi satisfatório. A jogabilidade possui boa fluidez e dificuldade balanceada dependendo da música escolhida. O cenário foi bem construído, passando ao jogador uma boa sensação. Os algoritmos do jogo funcionam bem, cumprindo suas funções especificadas.

Algumas discussões surgiram, principalmente no que se diz respeito a movimentação do jogador com o *PlayerNavigator*, descrito na seção 4.4.3. Esse *script* funciona com base na utilização de *Pathfindings* espalhados pela pista, porém como esses possuem distâncias definidas uns dos outros, o movimento do veículo fica limitado, não permitindo o jogador fazer curvas e trocas de pista mais rapidamente, impossibilitando que alguns pontos sejam pegos. Com isso dito, uma das sugestões para possíveis trabalhos futuros seria a mudança no sistema de troca de pista.

6. Conclusão

A conclusão de tanto o artigo quando do jogo foram satisfatórias. Os principais objetivos propostos no início do desenvolvimento foram alcançados, com o jogo funcionando, agradável visualmente, bem otimizado e divertido de ser jogado. Alguns objetivos mais complexos, entretanto, não foram completamente atendidos, como o fato da escolha das músicas em servidores *web* ter que ser manual pelo jogador, por exemplo.

O problema inicial apresentado no artigo foi resolvido. O jogo apresenta uma forma inovadora de geração de notas em jogos rítmicos, diminuindo o esforço do desenvolvedor no momento da criação de fases, o *Level Design*. Algumas otimizações podem ser feitas na forma como o *Beat Detection* foi implementado, podendo assim ser melhorado em possíveis trabalhos futuros.

Algumas dificuldades surgiram na criação das pistas. Para tal, existia a possibilidade delas serem criadas do zero ou utilizando *assets* de pistas prontas, divididas em blocos para serem encaixadas como um quebra cabeça. A segunda opção foi escolhida, porém, dessa forma, a pista acabou com algumas falhas que, apesar de não alterarem a *gameplay*, diminuem a qualidade do cenário.

Outras dificuldades durante o desenvolvimento do jogo foram em torno do algoritmo *Beat Detection*. Esse possui uma grande complexidade por se tratar da análise em tempo real de batidas da música, quais batidas ignorar, em qual pista o Fantasma deve percorrer, como usar músicas de *webservers* sem precisar baixá-las manualmente, etc. Todas essas dúvidas foram estudadas para que a melhor forma de respondê-las fosse utilizada, porém, por se tratar de uma análise em tempo real, esta pode mudar dependendo do *hardware* utilizado pelo jogador, podendo a experiência de jogo ser diferente de acordo com o *FPS* (*Frames Per Second*) de cada um.

Para trabalhos futuros, como dito anteriormente, a implementação do algoritmo de *Beat Detection* pode ser melhorada, passando um sentimento de ritmo ainda maior ao jogador. O *script* para o uso de músicas de *webservers* funciona, porém, não da forma como foi imaginado, com a utilização de *links* copiados diretamente do Youtube. Essa é outra ferramenta que pode ser melhorada e otimizada com mais tempo de desenvolvimento.

Referências

Abreu, B. P. d. (2021). Aplicações de processamento de sinais em extração de informação musical.

Abstract Art (2018). Abstractart. music racer. [s. l.]: Abstractart, 20 jul. 2018. disponível em: https://store.steampowered.com/. acesso em: 7 fev. 2022.

Activision (2005). Activision. guitar hero. [s. l.]: Activision, 8 nov. 2005. cd-rom.

Almeida, M. (2007). *Desvendando o 3ds Max*. Universo dos Livros Editora.

Amorin, A. (2006). A origem dos jogos eletrônicos. *São Paulo: Edusp*.

Beat Games (2018). Beat games. beat saber. [s. l.]: Beat games, 1 maio 2018. disponível em: https://beatsaber.com/. acesso em: 7 fev. 2022.

Bégel, V., Di Loreto, I., Seilles, A., e Dalla Bella, S. (2017). Music games: potential application and considerations for rhythmic training. *Frontiers in human neuroscience*, 11:273.

Betti, L. C. N., da Silva, D. F., e de Almeida, F. F. (2013). A importância da música para o desenvolvimento cognitivo da criança. *CONSELHO EDITORIAL*, page 45.

Clua, E. W. G. e Bittencourt, J. R. (2005). Desenvolvimento de jogos 3d: concepção, design e programação. In *Anais da XXIV Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação*, pages 1313–1356.

Compton, K. e Mateas, M. (2006). Procedural level design for platform games.

Costikyan, G., Words, I. H. N., e Design, I. M. (1994). Interactive fantasy 2.

de Moraes, J. J. (2017). *O que é música*. Brasiliense.

Dean Herbert (2007). Dean herbert. osu!. [s. l.]: Dean herbert, 16 jul. 2007. disponível em: https://osu.ppy.sh/home. acesso em: 7 fev. 2022.

Herrera-Boyer, P., Klapuri, A., e Davy, M. (2006). Automatic classification of pitched musical instrument sounds. In *Signal processing methods for music transcription*, pages 163–200. Springer.

Kitchin, H. A. (2003). The tri-council policy statement and research in cyberspace: Research ethics, the internet, and revising a 'living document'. *Journal of Academic Ethics*, 1(4):397–418.

Lima, T. B. e Moreira, L. O. (2018). Uma abordagem para utilização musical na geração procedural de conteúdo em jogos digitais.

Lyra, R. e de Jesus, E. A. (2014). Jogo musical rítmico para auxílio em exercícios de execução rítmica. *Anais do Computer on the Beach*, pages 223–232.

Marconi, M. d. A. (2002). Lakatos, eva maria técnicas de pesquisa.

Moffat, D., Ronan, D., e Reiss, J. D. (2015). An evaluation of audio feature extraction toolboxes.

Oliveira, N. F. B. (2015). Music-based procedural content generation for games.

Peng, P. H. e Lane, S. H. (2011). Designing rhythm game interfaces for touchscreen devices. *University Of Pennsylvania, PA*.

PGB (2021). Pgb (são paulo) (ed.). pesquisa gamer brasil 2021. disponível em: https://www.pesquisagamebrasil.com.br/pt/pesquisa-game-brasil/. acesso em: 07 fev. 2022.

Pivec, M. e Kearney, P. (2007). Games for learning and learning from games. *Organizacija*, 40(6).

Prodanov, C. C. e De Freitas, E. C. (2013). *Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição*. Editora Feevale.

Schell, J. (2008). *The Art of Game Design: A book of lenses*. CRC press.

Tomczak, M. T. (2005). Beatlib: A general-purpose beat detection library.

Webster, G. C. e Mendes, T. G. (2014). AnÁlise da interatividade em dj hero: A jogabilidade dos jogos musicais de ritmo.