Melódica: Jogo Digital Para Ensino Da Música

Bernardo Alves de Melo[1], **Eduardo Diconcili Costa**[1], **Milena Cristina França** [1]

[1]Instituto Federal de Santa Catarina (IFSC) - Campus Lages
R. Heitor Villa Lobos, 225 - São Francisco, Lages - SC, 88506-400
Curso de Ciência da Computação

bernardoalvesdemelo123@gmail.com, eduardodiconcilicosta@yahoo.com.br, milena@ifsc.edu.br

Abstract. *Digital games have been widely used as a teaching tool in the early years of school education, with their playful approach arousing the interest of their users in the content covered. Among the subjects present in the school curriculum, Music demonstrates characteristics similar to games in the way it is also used to spread knowledge and its study brings several benefits to cognitive and psychological development. This article contemplates the development of an educational digital game for teaching music called Melódica. The Melódica game belongs to the RPG genre and was developed using the Unity tool.*

Resumo. *Jogos digitais têm sido amplamente utilizados como ferramenta de ensino nos anos iniciais de formação escolar, com sua abordagem lúdica despertando o interesse de seus usuários nos conteúdos abordados. Dentre as disciplinas presentes no currículo escolar, a Música demonstra características semelhantes aos jogos no modo em que é também utilizada para propagação de conhecimento e seu estudo traz diversos benefícios para o desenvolvimento cognitivo e psicológico. Esse artigo contempla o desenvolvimento de um jogo digital educacional para o ensino da música chamado Melódica. O jogo Melódica pertence ao gênero RPG e foi desenvolvido utilizando a ferramenta Unity.*

1. Introdução

Segundo Durkheim (1977), Educação pode ser definida como um processo em que uma habilidade se desenvolve através de seu exercício contínuo, assim como o efeito de educar, aperfeiçoar as capacidades intelectuais e morais de um indivíduo. O processo educativo se aproveita de diversas técnicas, tais como a educação através do estudo contínuo de um conteúdo em específico, a educação através do consumo de mídias audiovisuais, até mesmo o uso de jogos como ferramenta educativa (Almeida Reis e Cavichiolli, 2008).

Enquanto jogos são utilizados primariamente como mídias de entretenimento, em sua concepção, possuem o objetivo de ensinar conceitos em um ambiente seguro e de forma lúdica, e podem ainda ser utilizados como recursos de ensino. A atividade de jogar precisa ser considerada em seu todo como uma forma de apoio dentro da educação para o desenvolvimento de muitas competências e habilidades do aluno, lhe servindo como estímulo enquanto este se encontra em fase de aprendizagem (Grando et al., 2008).

As instituições de ensino estão ampliando o uso das tecnologias de informação e comunicação para oferecer aos alunos mídias interativas que possam enriquecer as aulas.

Os jogos digitais aparecem nesse contexto como um recurso didático que contém características que podem trazer uma série de benefícios para as práticas de ensino e aprendizagem, segundo Savi et al. (2008). Jogos, dentro do contexto explorado neste trabalho, são atividades, que podem ser sociais ou não, e possuem objetivos finais, seja vencer um adversário, ultrapassar um obstáculo ou cumprir um objetivo. Normalmente, munidos de regras que definem uma lógica a ser seguida durante o desenvolvimento desta atividade, são definidos por ações e tomadas de decisões para o fim de sobressair em relação ao adversário ou apenas cumprir o objetivo estipulado dentro do sistema definido (Ludes et al., 2017).

Existem diversas formas de categorizar jogos, em relação ao meio em que utilizam, são considerados digitais ou analógicos. Jogos analógicos possuem uma variedade enorme de objetos de desenvolvimento de suas atividades, podem possuir cartas, dados, tabuleiros, peças de montagem, entre diversas outras ferramentas. Jogos eletrônicos se assemelham em proposta ao que é visto em jogos analógicos, aproveitando de recursos disponíveis em um contexto digital para expandir as possibilidades de interação. Assim como jogos analógicos, os jogos eletrônicos possuem uma variedade extensa em suas propostas e objetivos (Mendes, 2006). Os jogos também podem ser classificados de acordo com os seus objetivos podendo ser de ação, aventura, cassino, lógicos, estratégicos, esportivos, *roleplaying games* (RPGs), entre outros (Gebran, 2009).

Jogos como ferramentas de ensino, são extensivamente utilizados nos anos iniciais de formação devido a sua característica lúdica. Dentro desse contexto no currículo escolar, a disciplina de música demonstra características semelhantes aos jogos em como é utilizada como um meio de propagação de conhecimento, enquanto ao mesmo tempo, seu estudo traz diversos benefícios para o desenvolvimento cognitivo e psicológico. Segundo Rocha e Boggio (2013), muitos estudos têm apontado para o papel da música como ferramenta de intervenção em diferentes alterações neurológicas como afasia, autismo e dislexia.

Podem ser encontrados diversos estudos e relatos acerca do uso de jogos musicais para o aprendizado, crianças e adultos são expostos a jogos que os fazem trabalhar suas noções rítmicas, suas capacidades de coordenação motora, são expostos ao aprendizado de noções musicais, desenvolvem sua escuta orientada e seletiva, (Sousa, 2019). Esses estudos estão inseridos dentro do mundo dos jogos analógicos, onde a experiência necessita de ferramentas físicas, que por muitas vezes podem não estar disponíveis para o uso, ou serem de difícil acesso.

Jogos digitais nesse contexto, mostram-se como uma alternativa que pode ser facilmente acessada e reproduzida. Mas, apesar disso, há pouco destaque para trabalhos com jogos digitais que demonstrem a importância que a música possui ou que atuem como plataforma de propagação do ensino musical.

Na *internet* existem diversas ferramentas para o aprendizado da música, como sites educativos, vídeos com aulas explicativas de conceitos básicos e avançados. Porém, carece de ferramentas atrativas para pessoas que não estejam inseridas no contexto musical, que permitam formas de imersão e atividades com várias frentes de conexão como os jogos são capazes.

Esse trabalho possui como objetivo geral desenvolver um jogo educacional digital

para o ensino da música denominado Melódica. São objetivos específicos deste trabalho:

- Pesquisar trabalhos semelhantes relacionados ao ensino da música através de jogos.
- Realizar o levantamento de requisitos.
- Desenvolver o *design* e arquitetura da aplicação.
- Desenvolver a narrativa do jogo.
- Implementar o jogo digital.

Esse trabalho é composto por 5 etapas. Na primeira etapa será realizada uma pesquisa de trabalhos semelhantes, que abordem o tema de ensino da música através de jogos. Na segunda etapa serão levantados os requisitos para o desenvolvimento do *software*. Na terceira etapa será estabelecida a arquitetura e *design* da aplicação, utilizando diagrama de Linguagem de Modelagem Unificada (UML) para modelagem. Na quarta etapa será desenvolvida a narrativa da história do jogo. Por fim, na quinta etapa será realizado o desenvolvimento da aplicação utilizando a *engine Unit*, que utiliza a linguagem de programação C# como base.

Em relação à metodologia, este trabalho é considerado uma pesquisa aplicada em relação à sua natureza. Devido a forma em que o problema é abordado, esta pesquisa é qualitativa. Essa pesquisa é exploratória em função de seus objetivos. Quanto aos procedimentos técnicos, é de característica bibliográfica.

O documento é composto da seção de introdução além das seguintes seções: Na seção 2 é apresentado o referencial teórico. A seção 3 contém a definição de modelagem e requisitos. O desenvolvimento e implementação do trabalho são apresentados na seção 4. Por último, na seção 5, são apresentadas as conclusões finais em relação ao trabalho.

2. Referencial Teórico

Nessa seção são apresentados os tópicos essenciais para o entendimento deste trabalho, incluindo o conceito do que são jogos educacionais, a música no currículo escolar, transtornos cognitivos que podem ser auxiliados pela música e tecnologias que serão utilizadas. Essa seção também contempla uma revisão bibliográfica de trabalhos semelhantes.

2.1. Jogos Educacionais

Jogos educacionais são estratégias instrucionais que envolvem elementos, como regras, narrativas, desafios e *feedback* imediato, usados para atingir um objetivo educacional (Belarmino et al., 2021). Os jogos educacionais digitais, quando aplicados no processo de ensino-aprendizagem, buscam despertar o interesse dos alunos pelo uso de uma metodologia cativante, lúdica e desafiadora (Mayer et al., 2022).

Segundo Silva et al. (2021), não existe uma padronização oficial na categorização de jogos, mas a mais utilizada pelos pesquisadores é a categorização que separa jogos em jogos de ação, jogos de aventura, jogos de luta, jogos de quebra-cabeça, jogos de *Roleplaying Game* (RPG), simulações, jogos de esportes e jogos de estratégia.

Foi escolhido o gênero RPG para o desenvolvimento deste trabalho, fazendo uso de recursos como narrativas e exploração do ambiente para proporcionar uma experiência imersiva.

2.2. RPGs

A sigla RPG, que refere-se ao termo *Roleplaying Game*, é usada para definir um tipo de jogo, analógico ou digital, onde o jogador ou grupo de jogadores irá explorar uma história a partir do ponto de vista de um ou mais personagens.

No mundo dos jogos analógicos, sua origem pode ser traçada para o ano de 1974, com o lançamento oficial de *Dungeons & Dragons*, jogo de tabuleiro inspirado na literatura de J.R.R. Tolkien, autor de Senhor dos Anéis e Hobbit. Os jogadores, neste contexto, vivem em um mundo virtual imaginativo de fantasia com uma proposta do Mestre de Jogo, onde são instruídos por um livro de regras, que varia para cada jogo, a fim de criar suas personagens e viverem em um mundo fantástico, onde utiliza de recursos físicos e criativos para contar suas histórias (Jesus, 2022).

Dentro dos jogos eletrônicos, pode-se encontrar diversos *games* que utilizam os conceitos básicos de narrativa dos RPGs de tabuleiro para o desenvolvimento de suas dinâmicas. Jogos como *Tomb Raider* (2001) e *Uncharted* (2007) apresentam a dinâmica de inserção do jogador como um personagem dentro do mundo fictício proposto naquele contexto. De maneira mais direta, *Baldur's Gate 3* (2023) adapta fielmente o formato de RPGs de tabuleiro, trazendo para o mundo eletrônico as regras, mundo e ambientação daquilo que é proposto em *Dungeons & Dragons*.

Por fim, em ambiente analógico ou eletrônico, o RPG utiliza da sua narrativa para auxiliar no desenvolvimento de noções sociais e comportamentos de grupo, além de ser uma prática direta de socialização através da reunião de pessoas, tendem a trazer pontos de reflexão e questionamento para os jogadores, a fim de gerar conscientização sobre temas importantes mas desconfortáveis que podem ser abordados de maneira mais familiar ou casual através do entretenimento.

2.3. Música no Currículo Escolar

Hoje, a educação infantil é a primeira etapa de educação básica, destinado de forma geral à criança até aproximadamente seus cinco anos de idade, considerando a extensão da creche até a pré-escola. Acredita-se, dentro da área escolar, que para as crianças de um a três anos o contato diário com a música é necessário para a introdução desta como um meio de comunicação para possibilitar o desenvolvimento da criança. Desta forma, hábitos e rotinas são criados, estes que vistos como essenciais para o desenvolvimento da memória, cognição e linguagem, além de conhecimentos gerais (Geronimo, 2018).

A trajetória da Música como componente curricular escolar é conturbada, com diversos vai-e-vem ao longo dos anos, onde passou por períodos de ausência do currículo, já foi disciplina própria para compor o currículo e também já foi vista como componente da disciplina de Artes. Atualmente, porém, a Música é compreendida como parte essencial da disciplina de Artes, junto dos outros componentes que compõem a disciplina, sendo eles dança, teatro e artes visuais (Filipak, 2020).

No tocante à aplicação da música para o currículo escolar de educação para jovens e adultos, em referência a instituições que utilizam a modalidade de ensino da Educação para Jovens e Adultos (EJA), é observado o uso em algumas instituições de ensino, como o CMET (Centro Municipal de Educação dos Trabalhadores) Paulo Freire, escola de Porto Alegre, capital do Rio Grande do Sul, do Centro Musical, que conta com oficinas de

instrumentos, técnicas vocais e canto livre, adjunto do canto coral e outras práticas musicais coletivas. Este trabalho também contribui com desenvolvimento cognitivo e emocional, além de ser visto como oportunidade para a construção de novos conhecimentos e ampliação de relações sociais (Fracasso, 2020).

Estudos no âmbito de desenvolvimento neurológico apontam que a música ativa áreas do cérebro vinculadas ao sistema límbico, responsável pelas emoções, memória, linguagem e aprendizado humano (Gouveia, 2022).

2.4. Música Em Tratamentos Neurológicos

Como alguns pacientes de tratamentos para o Transtorno do Espectro de Autismo possuem dificuldade para comunicação, a música pode ser utilizada como instrumento de intervenção para o auxílio do tratamento para lidar com os problemas sociais que acompanham a dificuldade da escrita, através da prática de repetição com letras de músicas, além da prática com instrumentos tornar mais envolvente o engajamento em atividades difíceis, e o uso da música como auxílio no controle de emoções (Barata et al., 2022).

2.5. Afasia

Segundo o Ministério da Saúde, afasia trata-se de uma disfunção cerebral que causa dificuldade de comunicação no indivíduo. Afeta a compreensão de imagens, sons e outros tipos de expressão. Normalmente, afeta a compreensão e capacidade da linguagem verbal, compreensão e capacidade de comunicação escrita. Comumente, sua causa mais vista é o Acidente Vascular Cerebral, popularmente conhecido como AVC, porém, suas causas são diversas, entre elas estão tumores cerebrais, encefalites e traumatismo crânio-encefálico.

Geralmente, após a identificação da afasia seja por sequela ou como sintoma de algum mal ao qual o indivíduo foi acometido, como aqueles supracitados, o paciente deve ser encaminhado para terapia através de um programa de reabilitação da linguagem que envolve a prática de habilidades linguísticas, podendo ensinar o paciente a suprir a capacidade linguística com outras formas de comunicação, onde há a alternativa do uso da Terapia de Entonação Melódica (Barbosa, 2018).

2.6. Autismo

O Ministério da Saúde classifica o autismo (Transtorno do Espectro Autista) como um problema no desenvolvimento neurológico que acaba por prejudicar a organização de pensamentos, emoções e sentimentos. Uma de suas principais características para iniciar um diagnóstico é a falta do domínio da linguagem e do uso da imaginação, além da dificuldade de socialização e a presença de comportamento limitado e repetitivo (Viana et al., 2020).

Alguns autistas podem apresentar excesso de hiperatividade, acessos de raiva, déficit de atenção, passividade, dificuldade para lidar com certos tipos de ruído e aumento ou diminuição na resposta à dor e temperaturas. O transtorno não possui cura e cada paciente poderá necessitar de um acompanhamento específico para seu caso (Monteiro et al., 2021).

2.7. Dislexia

Dislexia do desenvolvimento é um transtorno específico da área de aprendizagem, categorizado como de origem neurobiológica. O indivíduo que possui dislexia apresenta dificul-

dade para associar o símbolo gráfico com seus fonemas. Algumas características comuns são a dificuldade com a linguagem, ortografia, principalmente a escrita, e o aprendizado de variados temas através da prática da leitura (Signor, 2020).

O tratamento para dislexia requer uma atenção pontual para cada caso, o terapeuta profissional poderá utilizar a linha que for mais conveniente para seu paciente de acordo com seu nível de desenvolvimento e suas habilidades de contornar as dificuldades advindas da dislexia (Silva Lopes, 2021).

2.8. Revisão Bibliográfica

Para realizar a pesquisa de trabalhos relacionados, foi utilizado a ferramenta de pesquisa *Google Academic*, com as palavras chaves de pesquisa "jogo educacional", "música", filtrando por trabalhos dos 4 últimos anos. Foram escolhidos artigos que apresentaram objetivos semelhantes a esse trabalho.

O trabalho de (Freitas et al., 2023) *FoxMusic*, apresenta um jogo de plataforma, voltado para o ensino de música para a educação infantil. Através de resultados obtidos por questionários, o jogo demonstrou ser uma ferramenta capaz de engajar os alunos e incentivar o aprendizado musical. Embora o gênero do jogo seja plataforma e não RPG como proposto neste trabalho, a exploração de conceitos como notas musicais para o ensino também serão abordados.

Nascimento et al. (2023) desenvolveu um jogo educacional para facilitar o ensino de conteúdos de computação, demonstrando que jogos são capazes de conter ambientes para simulações de temas complexos. O jogo denominado *My Name* apresenta a abstração do Problema da Mochila através de fundamentos da Teoria da Carga Cognitiva e *Narrative Learning*, obtendo resultados positivos em suas análises de aprendizagem de usuários iniciais. A temática do jogo *My Name* é diferente da abordagem de ensino através da música proposta neste trabalho, mas a abstração de temas complexos também será relevante no jogo que será desenvolvido.

Pino et al. (2021) em razão das vantagens do uso da música para pessoas com DI (Deficiência Intelectual), desenvolveu um jogo sério a fim de contribuir para a iniciação de aprendizagem da teoria da música para esses alunos por meio de um ensino remoto. O jogo é chamado Melodia, e explora o universo musical, com o estímulo da imaginação, musicalidade, raciocínio, memória e concentração. Os benefícios que o ensino da música oferece é algo também exposto neste trabalho, e ambos utilizam de jogos como ferramentas de ensino, com o jogo Melodia adotando um estilo voltado ao minimalismo.

Ramos et al. (2024) realizou o desenvolvimento de um jogo para ensino de teste de *software*, com objetivo de facilitar a aprendizagem nesta área. O projeto desenvolvido, chamado de *Greatest Unity*, é uma adaptação de um jogo de cartas utilizando a *game engine Unity*. O jogo apresenta uma interface de interação por turnos que integra conceitos relacionados a testes de *software*, sendo essa interface utilizada como referência no planejamento da tela de encontros desenvolvida para esse projeto.

Aires et al. (2020) desenvolveu um jogo de plataforma em 3D chamado *Pharos*, com foco no ensino da matemática. O jogo é composto por *puzzles* e combate com inimigos, que integram conceitos matemáticos como obstáculos que devem ser enfrentados. No jogo que será desenvolvido a narrativa é um elemento essencial, o que também é uma

característica do jogo *Pharos*, onde a história é explorada conforme o avanço do jogador.

No Quadro 1 pode ser observada a relação dos trabalhos relacionados juntamente com os respectivos autores de cada trabalho.

Trabalho	Autor
FoxMusic: um jogo digital no auxílio do ensino-aprendizagem infantil da música	Freitas et al. (2023)
My Name: desenvolvimento de um conjunto de mecânicas para o Problema da Mochila em um jogo educacional	Nascimento et al. (2023)
Melodia: um jogo sério para o ensino inicial de teoria musical a pessoas com deficiência intelectual	Pino et al. (2021)
GREatest Unity - Jogo digital de cartas para o ensino de testes de software	Ramos et al. (2024)
Pharos: um jogo educacional digital inovador para auxiliar no ensino e aprendizagem da Matemática	Aires et al. (2020)

Quadro 1: Trabalhos relacionados

2.9. Tecnologias Utilizadas

Com a evolução e a inclusão de jogos na área de aprendizado eletrônico evoluíram também as ferramentas utilizadas para o desenvolvimento, destacando-se as *game engines*, (Pontes et al., 2020). Segundo Vohera et al. (2021), *game engines* aceleram o processo de desenvolvimento e facilitam a integração de módulos com funcionalidades como animação, gráficos, inteligência artificial e sistema de física, utilizando suas funcionalidades integradas.

A *Unity* é uma *game engine* que utiliza a linguagem de programação C#. Possui suporte a múltiplas plataformas e um ambiente de desenvolvimento bem estabelecido, com fácil acesso a materiais disponibilizados por desenvolvedores. Levando em conta tais aspectos, foi definido seu uso para o desenvolvimento deste trabalho.

Para o desenvolvimento dos protótipos, foi utilizado o *Draw.io*, *software* de desenho gráfico, com interface que suporta a criação de fluxogramas, diagramas UML, organogramas, entre outros.

Em relação ao *design* dos mapas e personagens, todos foram feitos através de desenhos no *Photoshop*, *software* de criação e edição de imagens.

Para captação dos sons de instrumentos musicais, foi utilizado o programa *FL Studio* (*Fruity Loops Studio*), *software* criado pela empresa *Image-Line* para produção musical.

3. Modelagem e Requisitos

A seção 3 apresenta as definições de requisitos funcionais e não funcionais, além do diagrama UML para apresentação dos estados das telas do jogo e arquitetura das classes utilizadas no código.

3.1. Definição do GDD (*Game Design Document*)

Segundo Rêgo et al. (2021), o GDD é um artefato que agrega todas as informações geradas na etapa de *design* a fim de aumentar o entendimento da equipe de desenvolvimento em relação aos requisitos do jogo. GDD será construído a partir das definições de *design* dos tópicos que serão abordados nessa seção de modelagem.

3.2. Requisitos

Foram utilizadas as recomendações propostas por Leite e de Mendonça (2013), para direcionar a descoberta inicial de requisitos, com a equipe de desenvolvimento realizando reuniões de *brainstorm* para definir quais requisitos serão necessários para esse projeto. Os requisitos dos jogo estão listados nos Quadros 2 e 3:

Nome	Descrição	Prioridade
[RF-001] Menu inicial	O sistema deve conter uma tela com um menu inicial com as opções de iniciar o jogo, acessar uma tela de configurações e consultar os créditos de desenvolvimento.	Essencial
[RF-002] Ambiente do jogo	O sistema deve conter uma definição do ambiente para visualização do jogador onde o personagem principal irá se movimentar.	Essencial
[RF-003] Interações com o ambiente	O sistema deve permitir que o personagem principal interaja com elementos do ambiente.	Essencial
[RF-004] Sistema de encontro ou combate	O sistema deve possuir uma tela para encontros, onde o protagonista e o seu grupo irão combater os vilões da narrativa do jogo através da execução de uma melodia.	Essencial
[RF-005] Tela de seleção para notas e acordes	O sistema deve permitir selecionar notas e acordes para a melodia durante os encontros.	Essencial
[RF-006] Tela de pausa	O sistema deve conter uma tela de pausa, onde o usuário poderá acessar as configurações de volume e sair do jogo.	Essencial

Quadro 2: Requisitos funcionais

Nome	Descrição	Prioridade
[RF-007] Consulta de objetivos	O sistema deve permitir consultar os objetivos seguintes através do menu de pausa.	Importante
[RF-008] Armazenamento do progresso	O sistema deve permitir que o usuário salve o seu progresso no jogo, e consiga carregar o jogo no menu inicial.	Importante
[RF-009] Tela de visualização do instrumento	O sistema deve permitir que o jogador possa acessar uma tela com informações acerca dos instrumentos de cada personagem.	Importante
[RF-010] Tela de configurações	O sistema deve permitir através do menu inicial assim como no menu de pausa, que o jogador possa acessar uma tela para ajustar as configurações de som, aumentando ou diminuindo o volume.	Desejável
[RF-011] Tela de visualização dos adversários	O sistema deve permitir, durante os encontros, que o jogador possa acessar uma tela com informações sobre os adversários.	Desejável
[RF-012] Configuração de preferências	O sistema deve permitir, dentro do menu de configurações, que o jogador possa alterar teclas de atalho do jogo para personalizar à própria preferência.	Desejável

Quadro 3: Requisitos funcionais

A partir dos requisitos funcionais levantados, foi identificado a necessidade de interfaces que mostram informações ou possuem componentes interativos, para questões mais voltadas à interação direta entre o usuário e o jogo, como efeitos sonoros e visuais, foram levantados os requisitos apresentados no Quadro 4.

Nome	Descrição	Prioridade
[RNF-001] Efeitos sonoros	O sistema deve conter efeitos sonoros condizentes com os elementos musicais que forem representados.	Essencial
[RNF-002] Visual e artes dos personagens	O jogo deve ser realizado em 2D, contendo imagens dos personagens e cenários destacando seus *designs*.	Essencial
[RNF-003] Plataforma	A aplicação deve ser capaz de ser executada em um dispositivo *desktop*.	Essencial

Quadro 4: Requisitos não funcionais

Para essa implementação será utilizado o *Unity UI (uGUI)*, que é um sistema de interface de usuário baseado em *GameObjects*.

A Figura 1 apresenta um diagrama de estados do padrão UML (*Unified Modeling Language*) detalhando a arquitetura das telas.

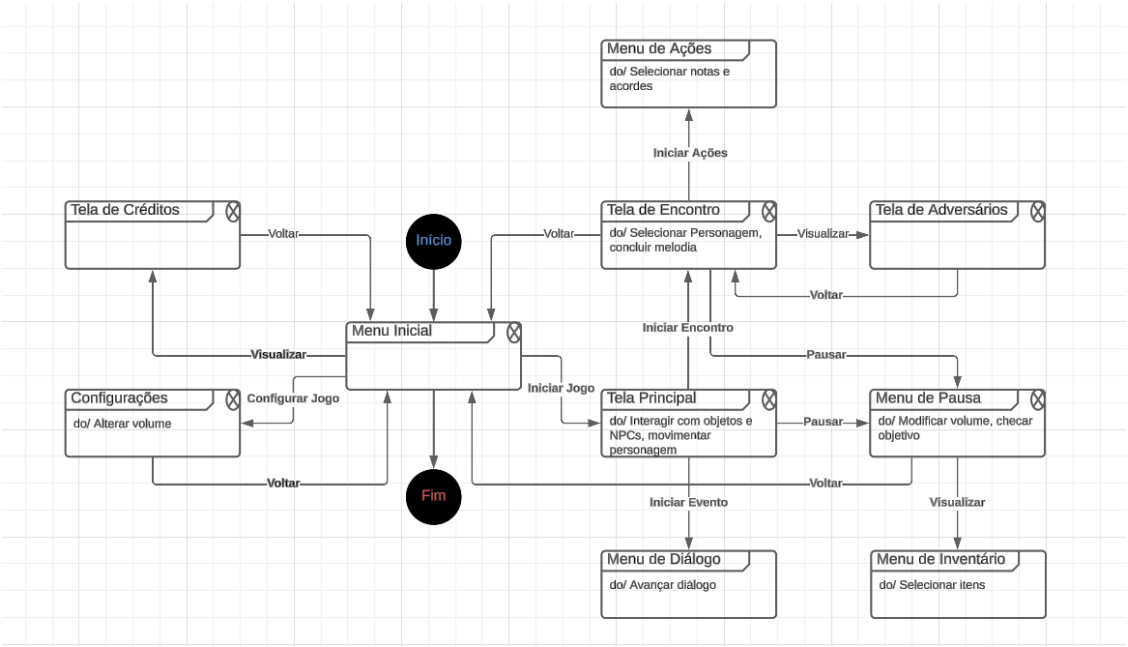


Figura 1. Diagrama de estados das telas do jogo

O usuário irá interagir com diferentes telas dependendo do estado da aplicação. Inicialmente será exibida a tela de menu inicial onde o usuário poderá iniciar o jogo ou ajustar configurações, onde também existirá uma tela para configurações de controles. Ao ser iniciado o jogo, ocorrerá uma transição para a tela principal, onde o usuário pode movimentar seu personagem no ambiente do jogo e interagir com objetos e NPCs.

O usuário também pode pausar o jogo, o que exibirá o menu de pausa, onde podem ser alteradas configurações de áudio e acessado um outro menu de inventário contendo os itens coletados. No menu de inventário é possível ver informações detalhadas dos conceitos musicais apresentados durante o progresso do jogo através dos itens coletados.

A narrativa do jogo é apresentada através de eventos que utilizam um menu de diálogo que é sobreposto a visualização atual do ambiente. Esse menu de diálogo também é apresentado durante interações com NPCs e objetos com interações.

Durante o jogo algumas ações podem provocar um evento de encontro, onde é exibida uma tela de combate com perfis selecionáveis dos personagens e pode ser acessado um menu de ações. No menu de ações é apresentado o instrumento respectivo do personagem selecionado, onde podem ser selecionadas notas musicais e acordes. Na tela de encontro é possível alternar para a tela de adversários, onde pode-se visualizar detalhes dos adversários.

3.3. Protótipo de Tela de Encontro

Na Figura 2, é possível visualizar uma imagem feita com auxílio do aplicativo *Draw.io* para ilustrar o protótipo de *design* das telas de encontro do jogo. Inicialmente, o jogador terá visualização dos personagens, onde ele poderá selecionar cada um dos personagens por vez e escolher qual a ação desejada para aquele personagem, se irá executar uma nota ou um acorde. Além disto, um botão de visualização para os adversários estará presente na interface.

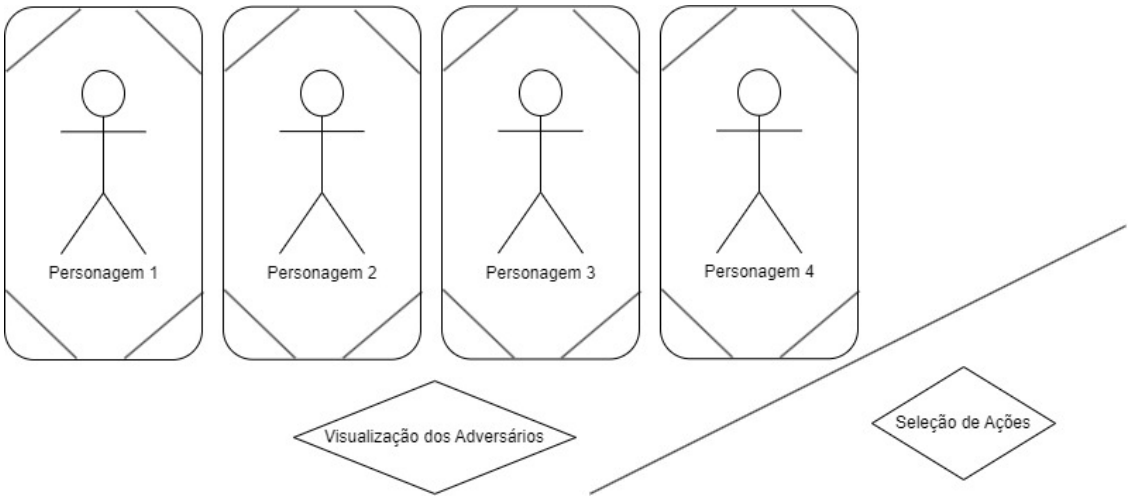


Figura 2. Protótipo de design da tela de encontro

O protótipo apresentado na Figura 3 ilustra o *design* da tela de visualização de adversários, tela secundária à tela de combate, para visualização dos adversários e o nível de melodia alcançado. Cada encontro terá um nível diferente de melodia necessário, tornando relevante a visualização da medida.

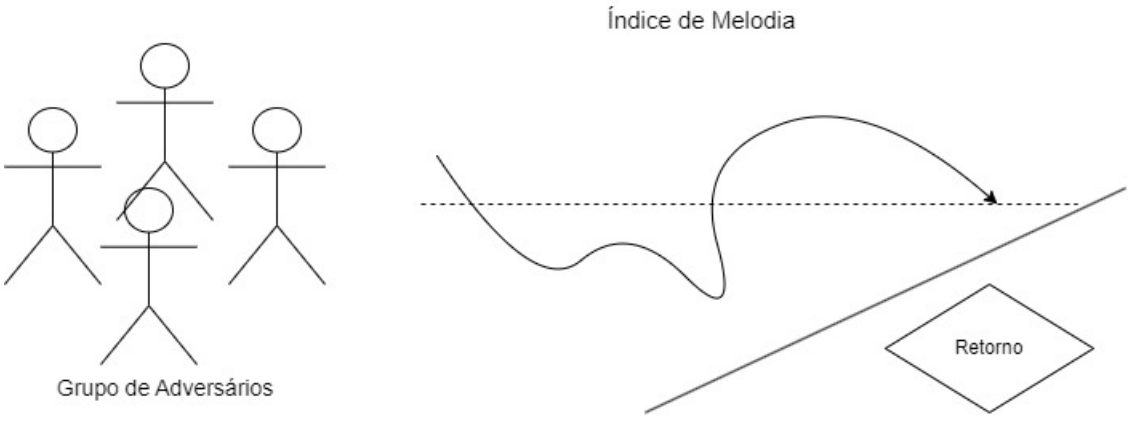


Figura 3. Protótipo de *design* da tela de visualização dos adversários

3.4. Arquitetura do *Software*

Os jogos desenvolvidos na *game engine Unity* são baseados em cenas, onde os objetos contidos são chamados *Game Objects*. Esses objetos são compostos por componentes,

que são estruturas modulares que possibilitam a extensão de funcionalidades através de *scripts*. Os *scripts* que serão implementados no jogo estão organizados em classes que controlam determinados aspectos como ações do jogador e movimentação. Essas classes serão explicadas durante a etapa de desenvolvimento.

Na Figura 4, apresenta-se uma imagem com a arquitetura das classes que serão utilizadas no jogo.

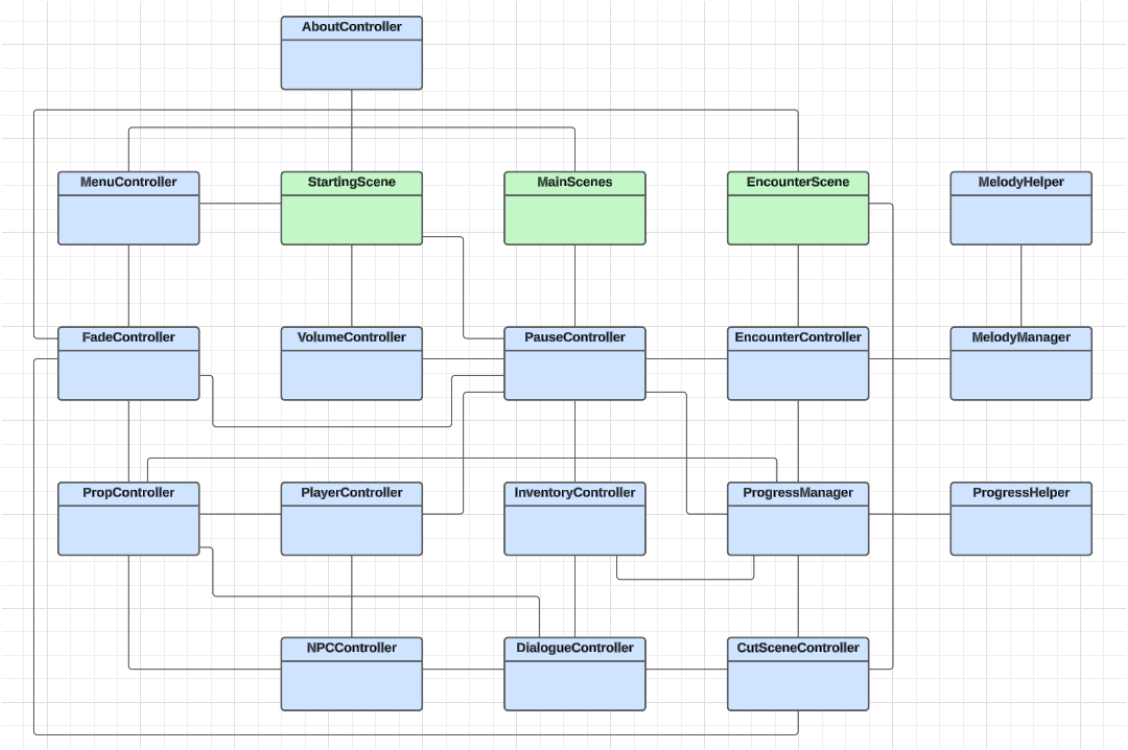


Figura 4. Arquitetura das classes do jogo

Conforme apresentado na Figura 4, os retângulos azuis representam as classes que serão utilizadas para a implementação das funcionalidades necessárias, enquanto que os retângulos verdes representam os tipos de cenas presentes no jogo. A *StartingScene* é a cena inicial apresentada durante o início da aplicação, enquanto que a *MainScenes* está representando as cenas principais do jogo onde o usuário irá movimentar o personagem em um determinado ambiente, com cada ambiente estando relacionado a uma cena específica. A *EncounterScene* trata-se da cena responsável por eventos de encontros onde o usuário irá combater adversários através da seleção de notas e acordes.

4. Desenvolvimento

A seção 4 engloba os detalhes voltados ao desenvolvimento do jogo como um todo, incluindo narrativa, *design* e implementação de código.

4.1. Narrativa

A subseção 4.1 apresenta a narrativa do jogo, que é compartilhada com o jogador conforme o progresso dentro do *game*.

4.1.1. Contexto Geral

Em uma realidade paralela, a música foi proibida, visando concentrar o máximo de tempo dos adultos nos seus devidos trabalhos. A Polícia Anti-Música (PAM) foi criada como órgão regulador para garantir que o trabalho com música e mesmo seu consumo sejam zerados, para que as pessoas se concentrem em "trabalhos sérios", usando de autoritarismo para focar toda a sociedade em trabalhos que foquem no desenvolvimento econômico.

Nesta situação, é responsabilidade da nova geração trazer a música de volta ao mundo, com incentivo e apoio das gerações anteriores que estão à sua volta.

4.2. *Design* e Artes Gráficas

A subseção 4.2 contém detalhes referentes ao *design* do jogo, incluindo mapas e estilo de personagens.

4.2.1. Mapas

O mapa da escola, onde o jogador se movimenta para alcançar as outras salas e seus objetivos, foi montado utilizando apenas o funcionamento de colisores 2D da *engine* do *Unity* por cima da imagem do mapa, ao invés de optado o uso de desenho dentro da plataforma.

Com texturas e filtros aplicados nas imagens, foi decidido como quesito de direção artística o uso de preto e branco ao longo da progressão da narrativa do *game*, as Figuras 5, 6 e 7 exemplificam o visual. O uso do filtro representa a falta de música, onde o mundo está preto e branco, conforme a música retorna a este mundo, as cores lentamente tornam a aparecer no ambiente.

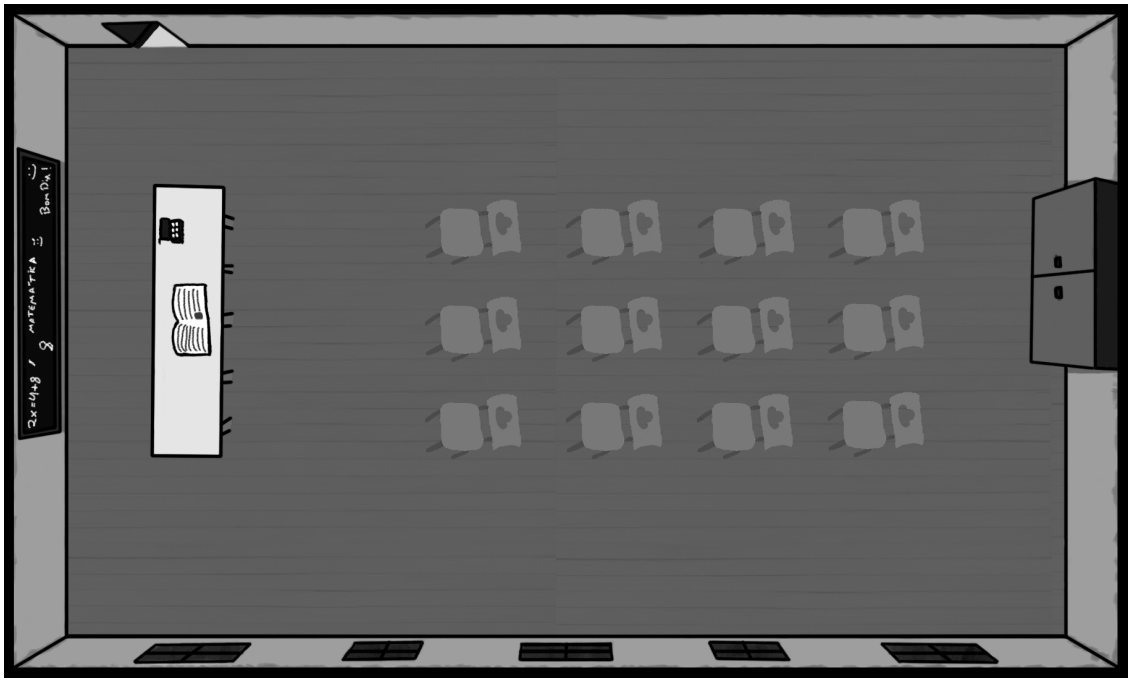


Figura 5. Mapa da Sala 1 no início do jogo, onde a música ainda há de ser descoberta

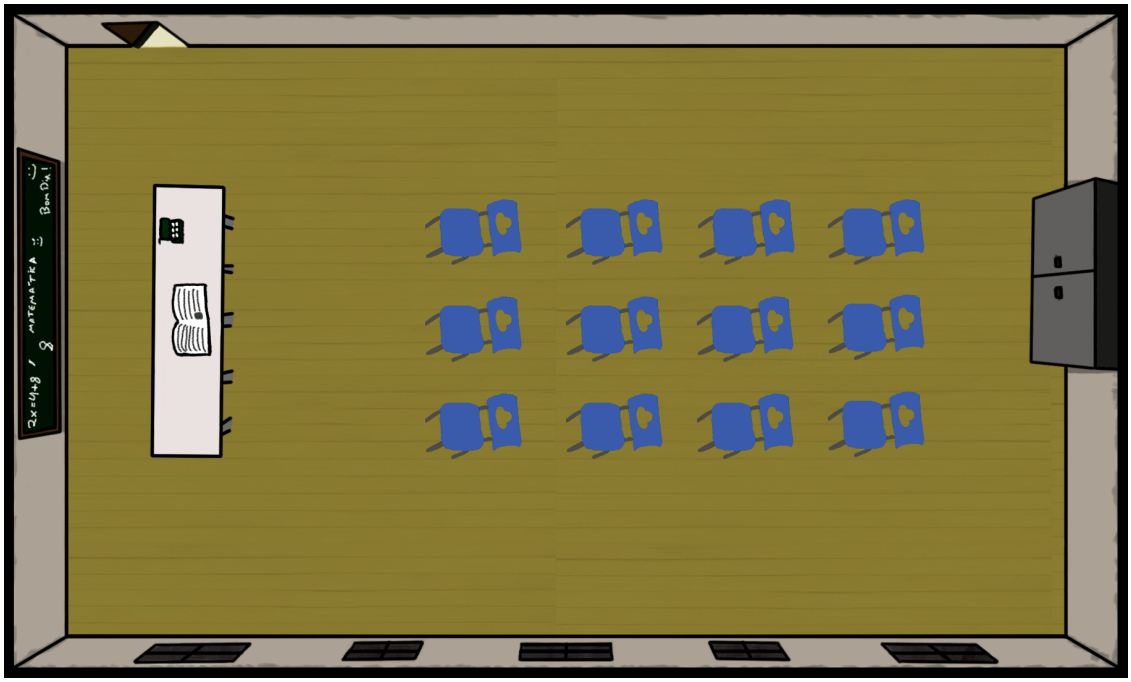


Figura 6. Mapa da Sala 1 no meio da progressão do jogo, após a música ser descoberta

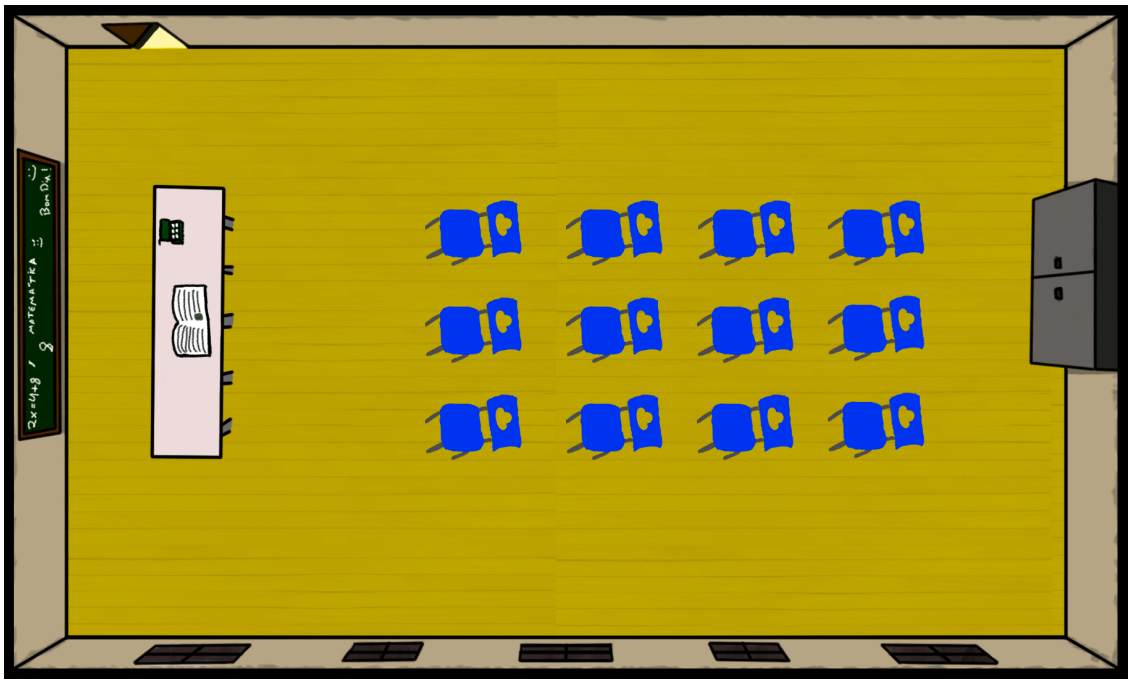


Figura 7. Mapa da Sala 1 ao final da progressão do jogo, com as cores completamente restauradas com a presença da música

Inspirado no estilo visto em muitos jogos *Flash* (apelido dado aos jogos *web* que utilizavam o *Adobe Flash Player*), o *design* dos mapas visa o minimalismo, evitando muitas sombras e detalhes que podem passar despercebidos aos olhos dos jogadores.

4.2.2. Arte de Personagem

O *design* dos personagens do jogo é gerado de forma autoral, por meio de desenhos à mão feitos com auxílio de mesa digitalizadora no *software* de imagens *Photoshop*, como apresentado na Figura 8.

Figura 8. Arte inicial do personagem principal do jogo

4.3. Implementação

A subseção 4.3 apresenta trechos de código utilizados na implementação dos *scripts* do jogo.

4.3.1. Interfaces

Para permitir comportamentos personalizados que possam ser tratados de forma genérica, foram implementadas interfaces com os métodos onde são esperadas implementações únicas em cada classe. As interfaces implementadas são apresentadas nas Figuras 9 e 10.

```
using UnityEngine;

namespace Assets.Shared.Scripts.Intefaces
{
    2 references
    public interface IColliderAction
    {
        2 references
        void ColliderEvent(GameObject collider);
    }
}
```

Figura 9. Interface de colisões

A interface *IColliderAction* demonstrada na Figura 9 é utilizada nas classes que possuem um comportamento de interação ao ocorrer uma colisão. Essas colisões são

ocasionadas durante tentativas de interação com algum elemento do ambiente, como ao interagir como uma porta ou para iniciar o diálogo com um NPC *(Non Playable Character)*.

```
namespace Assets.Shared.Scripts.Intefaces
{
    5 references
    public interface IEventAction
    {
        2 references
        void StartAction();
        2 references
        void EndAction();
    }
}
```

Figura 10. Interface de eventos

A Figura 10 exibe a interface *IEventAction* que foi implementada em razão da necessidade de execução de ações personalizadas durante o início e o fim de eventos ocorridos no jogo.

4.3.2. Gerenciador de Progresso

Foi implementada a classe *ProgressManager* para gerenciar e armazenar o progresso do usuário no jogo, a forma em que foi implementada está contida nas Figuras 11 e 12.

```
using Assets.Shared.Scripts.Flags;
using System.Collections.Generic;
using System.Linq;

5 references
public class ProgressManager
{
    1 reference
    public static int CurrentObjective {  get; set; } = 0;
    0 references
    public static int CurrentEncounter { get; set; } = 0;

    private static Dictionary<string, bool> _itenFlags = ProgressHelper.ItenFlags;
    private static Dictionary<string, bool> _eventFlags = ProgressHelper.EventFlags;

    2 references
    public static bool GetFlag(int key, FlagEnum flag)
    {
        var dictionary = GetFlagDictionary(flag);
        var keys = dictionary.Keys.ToList();

        return dictionary[keys[key]];
    }

    1 reference
    public static bool GetFlag(string key, FlagEnum flag)
    {
        var dictionary = GetFlagDictionary(flag);

        return dictionary[key];
    }
```

Figura 11. Gerenciador de progresso 1

```
    public static void SetFlag(string key, FlagEnum flag, bool value)
    {
        var dictionary = GetFlagDictionary(flag);

        dictionary[key] = value;
    }

    4 references
    private static Dictionary<string, bool> GetFlagDictionary(FlagEnum flag)
    {
        switch (flag)
        {
            case FlagEnum.Iten:
                return _itenFlags;
            case FlagEnum.Event:
                return _eventFlags;
            default: return null;
        }
    }
}
```

Figura 12. Gerenciador de progresso 2

A classe *ProgressManager* armazena o valor do objetivo atual e de evento de encontro atual para que outras classes utilizem esses valores para decidir a configuração de cenas e mensagens de ajuda.

Os dicionários armazenam valores referentes a itens coletados e de eventos, os valores são modificados através da execução dos métodos de interfaces que são executados durante a interação do usuário ou na execução de um evento.

4.3.3. Menu Inicial

O menu inicial é a primeira tela da aplicação e também a tela de retorno para quando o usuário decide sair do jogo. Suas funcionalidades são implementadas através da classe *MenuController* e sua execução pode ser observada na Figura 13.

Figura 13. Menu inicial

Na tela de menu inicial o usuário pode iniciar o jogo, ajustar o volume ou acessar a tela de créditos. O menu de volume é apresentado na Figura 14 e o menu de créditos na

Figura 16.

Figura 14. ***Menu de volume***

A opção de configurações realiza o acesso do menu de volume utilizando da classe *VolumeController* e do componente *mixer* apresentado na figura 15. No menu de volume podem ser ajustados os valores de volume para os sons de música, efeitos sonoros e o volume mestre.

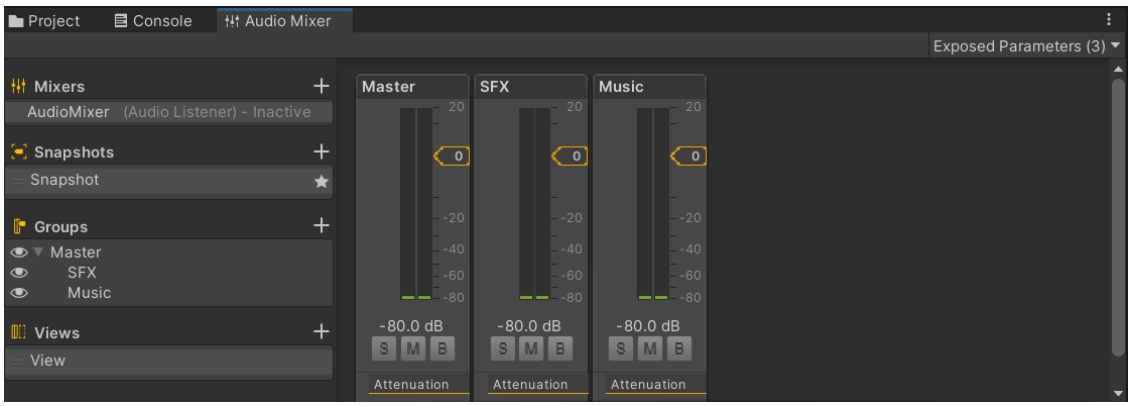


Figura 15. ***Audio mixer***

Para o gerenciamento interno do volume do jogo é utilizado o componente *mixer*, que possibilita a configuração de diferentes saídas de áudio que são relacionadas a objetos capazes de produzir som.

A opção de créditos apresenta a informação dos desenvolvedores e orientadores que auxiliaram na realização do trabalho, como apresentado na Figura 16.

Figura 16. ***Menu de créditos***

4.3.4. Gerenciamento de Transições

Foi utilizada a classe *FadeController* incluída nas Figuras 17, 18 e 19 para gerenciar as transições de início e término de cenas e eventos, realizando gradualmente mudanças através da apresentação de uma imagem previamente definida.

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

Unity Script (1 asset reference) | 6 references
public class FadeController : MonoBehaviour
{
    4 references
    public static GameObject Instance { get; set; }

    public Image fadeImage;
    public float fadeSpeed = 3.0f;

    private Sprite _imageSprite;
    private bool _fadeOut = false;
    private bool _executing = false;
    private int _scene = 0;
    private float _alpha = 0.0f;

    Unity Message | 0 references
    private void Awake()
    {
        Instance = this.gameObject;
        _imageSprite = fadeImage.sprite;
        fadeImage.enabled = false;
    }
```

Figura 17. Controlador de transições 1

```
public void FadeOut(int scene, float speed = -1, Sprite sprite = null)
{
    _alpha = 0.0f;
    fadeSpeed = speed > 0 ? speed : fadeSpeed;
    fadeImage.sprite = sprite ?? _imageSprite;
    fadeImage.enabled = true;
    _fadeOut = true;
    _scene = scene;
    _executing = true;
}

2 references
public void FadeIn(float speed = -1, Sprite sprite = null)
{
    _alpha = 1f;
    fadeSpeed = speed > 0 ? speed : fadeSpeed;
    fadeImage.sprite = sprite ?? _imageSprite;
    fadeImage.enabled = true;
    _fadeOut = false;
    _executing = true;
}
```

Figura 18. Controlador de transições 2

```
private void Update()
{
    if (!_executing)
        return;

    if (_alpha > 1 || _alpha < 0)
    {
        _alpha = (_fadeOut ? 1f : 0.0f);
        fadeImage.color = new Color(fadeImage.color.r, fadeImage.color.g, fadeImage.color.b, _alpha);
        fadeImage.enabled = _fadeOut;
        _executing = false;

        if (_fadeOut)
        {
            var player = PlayerController.Instance.GetComponent<PlayerController>();
            player.talkAction.performed -= player.FindCollider;
            SceneManager.LoadScene(_scene);
        }

        return;
    }

    _alpha += Time.deltaTime * fadeSpeed * (_fadeOut ? 1 : -1);
    fadeImage.color = new Color(fadeImage.color.r, fadeImage.color.g, fadeImage.color.b, _alpha);
}

}
```

Figura 19. Controlador de transições 3

A classe *FadeController* possui valores padrões de atributos referentes a transição que podem ser definidos através dos métodos *FadeIn* e *FadeOut* pelo objeto que invocar a transição. Por padrão a imagem de transição é uma tela escura.

O método *FadeIn* realiza o processo de entrada da transição, onde a imagem definida é apresentada na tela e gradualmente desaparece. O método *FadeOut* está relacionado ao processo de saída da transição, com a imagem definida gradualmente surgindo na tela.

O método *Update* irá gradualmente aplicar o efeito do método de transição escolhido e por fim encerrar a transição ou trocar a cena atual no caso desta estar com o estado *_fadeOut* com valor *true*.

4.3.5. Menu de Pausa

Durante a execução do jogo, um menu de pausa pode ser acessado quando a tecla *Enter* é pressionada. A implementação desse menu é visível na Figura 20.

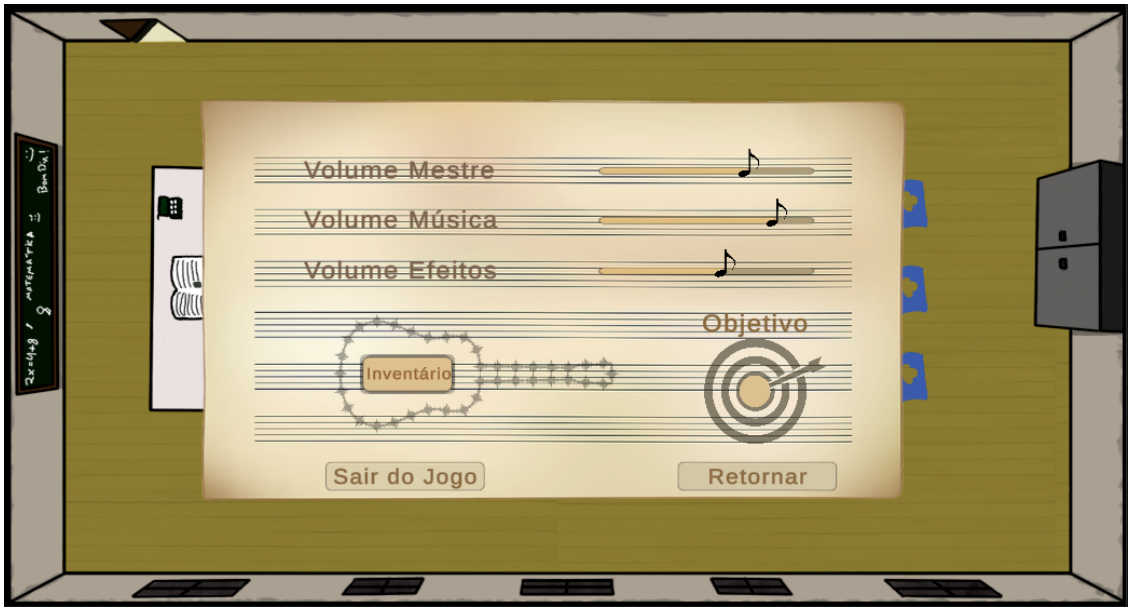


Figura 20. Menu de pausa

Enquanto o menu de pausa está ativo, os processos no jogo são parados e não é possível interagir com o ambiente. No menu de pausa é possível alterar as configurações de volume para as diferentes saídas disponibilizadas de volume mestre, de música e efeitos. Na parte inferior do menu encontra-se o botão de retornar que fecha o menu de pausa e o botão de sair do jogo que muda a cena para o menu inicial.

O botão de objetivo possui a função de apresentar o objetivo atual para o usuário, enquanto que o botão de inventário transaciona o menu atual para o menu de inventário apresentado na Figura 21.

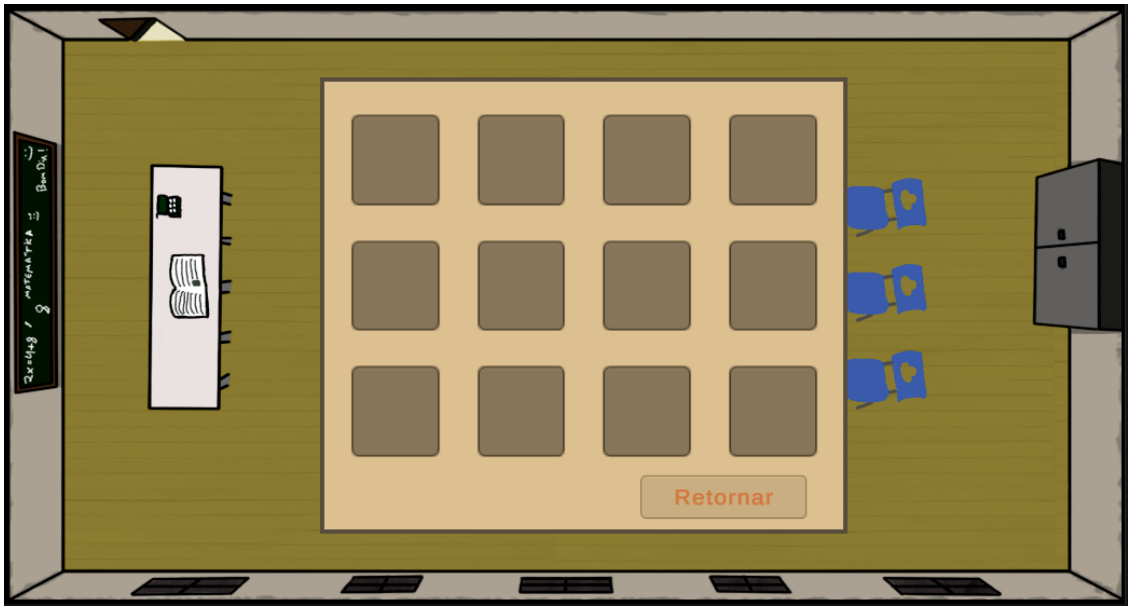


Figura 21. Menu de inventário

O menu de inventário contém os itens coletados pelo usuário durante o progresso no jogo, fornecendo a possibilidade de visualização de informações referentes a cada item.

4.3.6. Controlador de Diálogos

Durante conversas com NPCs ou eventos de narrativa, diálogos são apresentados para o usuário. Além do texto também podem ser apresentadas imagens ou modificada a câmera durante a execução do diálogo. Na Figura 22 está um exemplo de diálogo de um evento de narrativa.

Figura 22. Exemplo de diálogo

A classe *DialogueController* mostrada nas Figuras 23, 24 e 25 realiza o gerenciamento da apresentação dos diálogos incluindo os atributos e funcionalidades envolvidas.

```
using TMPro;
using UnityEngine;
using UnityEngine.UI;

Unity Script (1 asset reference) | 7 references
public class DialogueController : MonoBehaviour
{
    public TextMeshProUGUI dialogueText;
    public TextMeshProUGUI nameText;
    public GameObject dialoguePanel;
    public Image characterImage;
    public Image backgroundImage;
    public Image dialogueImage;
    public Image nameImage;
    public Sprite defaultSprite;
    public Animator cameraAnimator;

    7 references
    public static DialogueController Instance { get; private set; }

    Unity Message | 0 references
    private void Awake()
    {
        dialoguePanel = this.gameObject;
        Instance = this;
        dialoguePanel.SetActive(false);
    }
```

Figura 23. ***DialogueController*** **1**

```
    1 reference
    public void ShowDialogue(string dialogue, string name, Sprite characterSprite, Sprite backgroundSprite, int camera)
    {
        dialogueImage.enabled = dialogue != "";
        nameImage.enabled = name != "";
        nameText.text = name;
        dialogueText.text = dialogue;
        characterImage.sprite = characterSprite;
        backgroundImage.sprite = backgroundSprite;
        cameraAnimator.SetInteger("Camera", camera);
        dialoguePanel.SetActive(true);
    }

    1 reference
    public void EndDialogue()
    {
        nameText.text = null;
        dialogueText.text = null;
        characterImage.sprite = null;
        backgroundImage.sprite = null;
        cameraAnimator.SetInteger("Camera", 0);
        dialoguePanel.SetActive(false);
    }
```

Figura 24. ***DialogueController*** **2**

```
    public void DialogueEvent(GameObject player,
                              DialogueAsset dialogueAsset,
                              string name,
                              ref int line,
                              Sprite characterSprite = null,
                              Sprite backgroundSprite = null,
                              int camera = 0)
    {
        var playerController = player.GetComponent<PlayerController>();
        characterSprite = characterSprite ?? defaultSprite;
        backgroundSprite = backgroundSprite ?? defaultSprite;
        int dialogueCount = dialogueAsset != null ? dialogueAsset.dialogue.Length : 1;

        if (line < dialogueCount)
        {
            var dialogue = dialogueAsset != null ? dialogueAsset.dialogue[line] : "";
            ShowDialogue(dialogue, name, characterSprite, backgroundSprite, camera);

             if (playerController != null)
                 playerController.inConversation = true;

             line++;
        }

        else
        {
            line = 0;

            if (playerController != null)
                playerController.inConversation = false;

            EndDialogue();
        }
    }
}
```

Figura 25. *DialogueController* 3

Durante um diálogo, outros processos no jogo são parados enquanto ele estiver ativo. Um diálogo é dividido em vários segmentos que são avançados pelo usuário ao pressionar a tecla de barra de espaço.

Para a execução de diálogo através da classe *DialogueController* também foi necessário a criação da classe *DialogueAsset* conforme a Figura 26.

```
using UnityEngine;

[CreateAssetMenu]
Unity Script | 4 references
public class DialogueAsset : ScriptableObject
{
    [TextArea]
    public string[] dialogue;
}
```

Figura 26. *DialogueAsset*

A classe *DialogueAsset* é um *ScriptableObject* que armazena o texto de diálogo que será apresentado por meio de em um vetor de *string*. Um *ScriptableObject* é uma classe que é salva como um *Asset* nas pastas do projeto ao invés de ser adicionado a um *GameObject*. Ela é uma classe capaz de armazenar dados de forma independente de instâncias de outras classes.

4.3.7. Eventos de Encontros

Durante o progresso do jogo o usuário será introduzido aos conceitos de notas musicais e acordes, esse conhecimento é posto em prática através de eventos de encontros onde o usuário confronta adversários utilizando da música.

Durante o evento de encontro, o usuário deve escolher quais notas musicais irá tocar e através dessa escolha uma pontuação é gerada em relação a harmonia da sequência de notas escolhidas. No decorrer da narrativa, o protagonista irá encontrar amigos que também participarão desse evento, com cada membro possuindo seu próprio instrumento e tela de escolha de notas referente.

Na Figura 27 é demonstrado a tela de escolhas de notas do protagonista cujo instrumento utilizado é o violão.

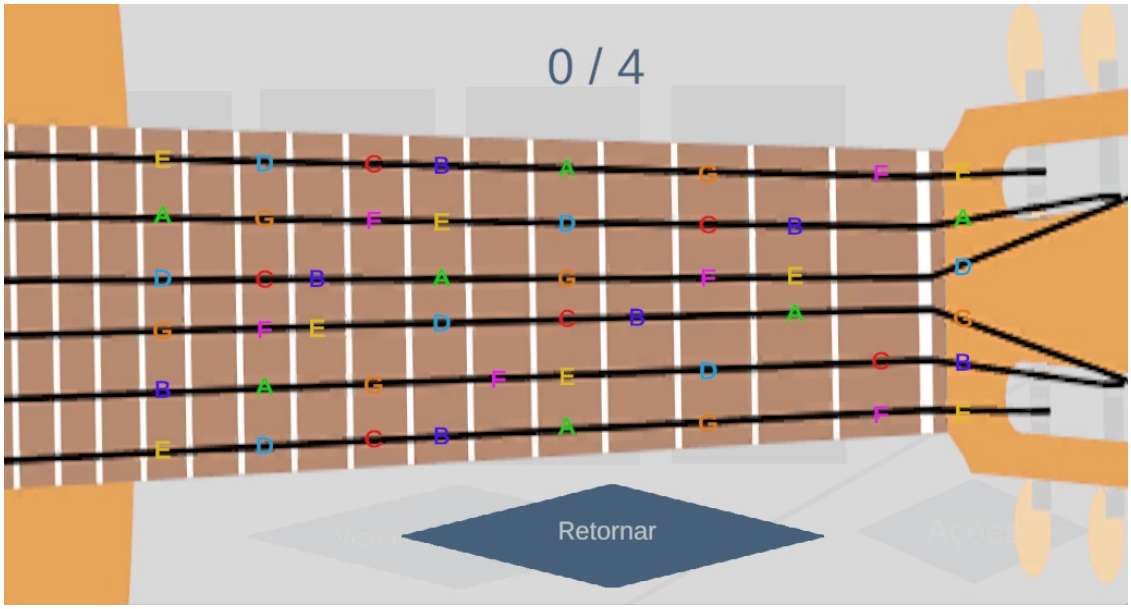
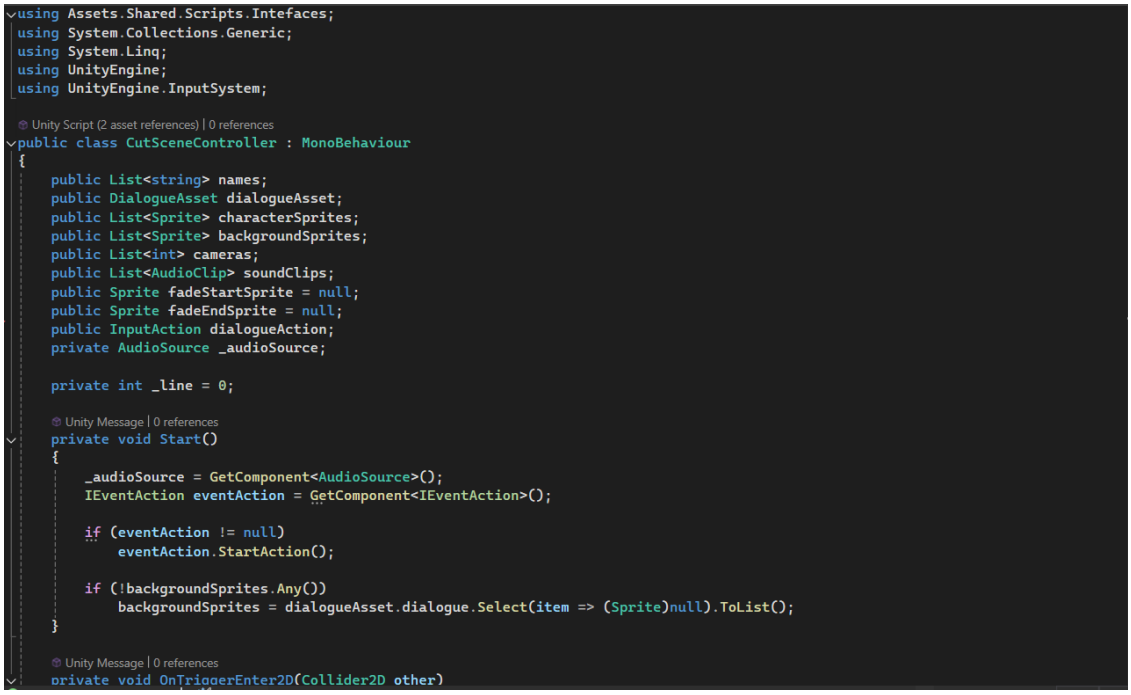


Figura 27. Tela de escolha de notas no violão

Outros instrumentos disponíveis por integrantes do grupo são a guitarra, baixo e tambor. As funcionalidades da cena de encontro são implementadas através da classe *EncounterController* que gerencia as interfaces gráficas e sonoras, além das regras para a formulação da pontuação da escolha de notas.

4.3.8. Controlador de Eventos de Narrativa

A narrativa do jogo é apresentada através de eventos que ocorrem quando o usuário entra em uma localidade do ambiente. Esses eventos são denominados internamente como *CutScenes*, e são gerenciados pela classe *CutSceneController* apresentada nas Figuras 28, 29 e 30 .



```
using Assets.Shared.Scripts.Intefaces;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.InputSystem;

Unity Script (2 asset references) | 0 references
public class CutSceneController : MonoBehaviour
{
    public List<string> names;
    public DialogueAsset dialogueAsset;
    public List<Sprite> characterSprites;
    public List<Sprite> backgroundSprites;
    public List<int> cameras;
    public List<AudioClip> soundClips;
    public Sprite fadeStartSprite = null;
    public Sprite fadeEndSprite = null;
    public InputAction dialogueAction;
    private AudioSource _audioSource;

    private int _line = 0;

    Unity Message | 0 references
    private void Start()
    {
        _audioSource = GetComponent<AudioSource>();
        IEventAction eventAction = GetComponent<IEventAction>();

        if (eventAction != null)
            eventAction.StartAction();

        if (!backgroundSprites.Any())
            backgroundSprites = dialogueAsset.dialogue.Select(item => (Sprite)null).ToList();
    }

    Unity Message | 0 references
    private void OnTriggerEnter2D(Collider2D other)
```

Figura 28. ***CutSceneController*** **1**

```
    private void OnTriggerEnter2D(Collider2D other)
    {
        PlayerController controller = other.GetComponent<PlayerController>();

        if (controller != null)
        {
            _audioSource.PlayOneShot(soundClips[_line]);

            if (fadeStartSprite != null)
                FadeController.Instance.GetComponent<FadeController>().FadeIn(5, fadeStartSprite);

            var playerController = PlayerController.Instance;

            dialogueAction.Enable();
            dialogueAction.performed += OnCollisionEvent;
            DialogueController.Instance.DialogueEvent(playerController,
                                                      dialogueAsset,
                                                      names[_line],
                                                      ref _line,
                                                      characterSprites[_line],
                                                      backgroundSprites[_line],
                                                      cameras[_line]);
        }
    }
```

Figura 29. ***CutSceneController*** **2**

```
public void OnCollisionEvent(InputAction.CallbackContext context)
{
    var index = _line < dialogueAsset.dialogue.Count() ? _line : _line - 1;

    _audioSource.PlayOneShot(soundClips[index]);

    GameObject player = PlayerController.Instance;

    DialogueController.Instance.DialogueEvent(PlayerController.Instance,
                                              dialogueAsset,
                                              names[index],
                                              ref _line,
                                              characterSprites[index],
                                              backgroundSprites[index],
                                              cameras[index]);

    if (_line == 0)
    {
        if (fadeEndSprite != null)
            FadeController.Instance.GetComponent<FadeController>().FadeIn(5, fadeEndSprite);

        IEventAction eventAction = GetComponent<IEventAction>();

        if (eventAction != null)
            eventAction.EndAction();

        dialogueAction.performed -= OnCollisionEvent;
        Destroy(this.gameObject);
    }
}
}
```

Figura 30. *CutSceneController* 3

A classe *CutSceneController* gera eventos segmentados, onde cada passo é avançado quando o usuário pressiona a tecla de barra de espaço. Essa classe possui diversas listas de objetos, onde são utilizados os objetos de índice referente ao passo atual durante a execução do evento.

O método *Start* é executado quando a classe é inicializada. Deste modo, ela verifica se existe uma interface *IEventAction* relacionada ao objeto e no caso de existir, realiza uma chamada para o método *StartAction* que possui comportamentos únicos de inicialização referente ao evento atual.

O método *OnTriggerEnter2D* é invocado quando o usuário entra na área de colisão do objeto. Esse método inicialmente dispara um *clip* de áudio e invoca a classe *FadeController* se existir uma imagem de transição configurada. A propriedade *dialogueAction* referente a ação de pressionar a tecla de barra de espaço é então habilitada e relacionada ao método *OnCollisionEvent*. Por fim, é realizada uma chamada ao *DialogueController* com os objetos do passo inicial.

O método *OnCollisionEvent* realiza o progresso do evento, disparando um *clip* de áudio e invocando um novo evento de diálogo com os objetos do passo atual. Durante o fim da execução do evento é executado o método *EndAction* se existir alguma interface *IEventAction* e o objeto é então destruído.

4.4. Testes

Ao longo do desenvolvimento, pelas características de desenvolvimento de um *game*, testes unitários se aplicam majoritariamente a pequenos pedaços do projeto, desta forma, o método de teste de caixa-branca foi utilizado para realizar verificações quanto à lógica dentro do jogo, testes de colisores e verificação do funcionamento da mecânica de criação de melodias nos encontros.

5. Conclusões

Ao decorrer da narrativa é observável que esta primeira versão do jogo, que pode ser caracterizada como um protótipo expansível, tem como proposta a apresentação de conceitos básicos musicais, como o ensino das sete notas musicais e suas execuções nos instrumentos de corda apresentados no jogo, que são: o violão, guitarra e baixo, além dos acordes básicos encontrados no violão e guitarra. Além disto, noções rítmicas para instrumentos percussivos também são apresentados.

Foram efetuados testes de caixa-branca para testar o *game*, tendo em vista que outras validações necessárias demonstram carência de parceria com profissionais ligados à música e educação para verificar o valor educativo das informações e da forma de apresentá-las ao usuário. Demonstra ser de grande utilidade realizar validações de aprendizado com usuários através de formulários, levantando questões sobre o que foi aprendido e um índice de satisfação com a experiência. Da mesma forma, para validar a capacidade do software também como ferramenta de auxílio para tratamento terapêutico das condições mentais abordadas no trabalho, se faz necessária parceria para verificação prática com profissionais da saúde ligados diretamente ao tratamento terapêutico.

Como próximos passos, é necessária a verificação das informações supracitadas, expondo o *game* a jogadores interessados e solicitando que participem do questionário de qualidade que evidenciará posteriormente o quanto as informações estão sendo bem passadas aos usuários da aplicação.

Adiante, pode ser avaliada a possibilidade de expandir esta primeira versão do jogo, aumentando sua história conforme a narrativa estabelecida e abrangendo mais áreas do conhecimento musical e artístico.

Referências

Aires, S., Madeira, C., Santos, G., e Nascimento, N. (2020). Pharos: um jogo educacional digital inovador para auxiliar no ensino e aprendizagem da matemática. In *Anais dos Workshops do IX Congresso Brasileiro de Informática na Educação*, pages 232–239, Porto Alegre, RS, Brasil. SBC.

Almeida Reis, L. J. e Cavichiolli, F. R. (2008). Jogos eletrônicos e a busca da excitação. *Movimento*, 14(3):163–183.

Barata, R. A., Leão, L., Leão, L., e Júnior, Á. D. D. (2022). A música como intervenção neuropsicológica no tratamento do transtorno do espectro do autismo. *Nova Revista Amazônica*, 10(3):93–102.

Barbosa, T. N. (2018). Música e linguagem: aspectos atuais da terapia de entonação melódica na clínica das afasias.

Belarmino, G., Oliveira, R., Rodriguez, C., Goya, D., e Rocha, R. (2021). Descrição e análise dos processos de produção de um jogo educacional e seus impactos na sua qualidade. *Anais Estendidos do Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames)*, pages 407–416.

Durkheim, É. (1977). A educação como processo socializador: função homogeneizadora e função diferenciadora. *PEREIRA, L.; FORACCHI, MM Educação e sociedade*, 8.

Filipak, R. (2020). Arte (música) no currículo integrado: reflexões sobre seu papel diante das avaliações em larga escala e a base nacional comum curricular. *Anais do SIMPOM*, (6).

Fracasso, D. C. (2020). A música no currículo da educação de jovens e adultos: um estudo de caso. *Revista da ABEM*, 28.

Freitas, N. C., Castro, H. R., e Sousa, P. M. (2023). Foxmusic: Um jogo digital no auxílio do ensino-aprendizagem infantil da música. In *Anais Estendidos do XXII Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 695–705, Porto Alegre, RS, Brasil. SBC.

Gebran, M. P. (2009). Tecnologias educacionais. *Curitiba: Iesde Brasil Sa*, pages 189–190.

Geronimo, G. d. S. (2018). Utilização da música para o desenvolvimento da linguagem como parte do currículo escolar na creche.

Gouveia, C. (2022). A influência da música no neurodesenvolvimento infantil: Apontamentos neuropsicológicos. *Mosaico: Estudos em Psicologia*, 10(1):67–84.

Grando, A., Tarouco, L., e Rockenbach, M. (2008). O uso de jogos educacionais do tipo rpg na educação. *Revista Novas Tecnologias na Educação*, 6(1).

Jesus, I. A. d. (2022). Intertextualidades entre os universos de jrr tolkien e o de dungeons & dragons.

Leite, P. S. e de Mendonça, V. G. (2013). Catálogo para criação de jogos sérios para sistemas baseados em iot. In *Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital*. SBC.

Ludes, D., Xexéo, G., Carmo, A., Acioli, A., Taucei, B., Dipolitto, C., Mangeli, E., Kritz, J., Costa, L. F. C., Monclar, R., et al. (2017). O que são jogos. 6(1).

Mayer, R., Varela, P., Albonico, M., Rohling, A., e Steffen, V. (2022). Experiências de um jogo educacional digital para auxiliar no processo de ensino-aprendizagem de transformações químicas para o ensino médio. *Anais do Workshop de Informática na Escola (WIE)*, pages 59–67.

Mendes, C. L. (2006). Jogos eletrônicos, diversão, poder e subjetivação. Number 1. Papirus Editora.

Monteiro, C. G., de Araújo Batista, T. L., de Araújo, J. K., Rossi, R. R., Coelho, S. M., e Teixeira, L. M. S. (2021). A importância do uso de tecnologias no acompanhamento do autismo infantil. *BIUS-Boletim Informativo Unimotrisaúde em Sociogerontologia*, 27(21):1–7.

Nascimento, L., Honda, F., Melo, D., Pessoa, M., Oliveira, E., Fernandes, D., e Pires, F. (2023). My name: desenvolvimento de um conjunto de mecânicas para abordar o problema da mochila em um jogo educacional. *Anais do Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 888–899.

Pino, M., Dias, M., Matos, S., Borges, H., e Lopes, R. (2021). Melodia: Um jogo sério para o ensino inicial de teoria musical a pessoas com deficiência intelectual. *Anais Estendidos do Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames)*, pages 529–538.

Pontes, G., de Farias, A., Rocha, W., e Afonso, R. A. (2020). Mundo animal: Um jogo educacional livre para dispositivos móveis. *Anais da Escola Regional de Computação Bahia, Alagoas e Sergipe (ERBASE)*, pages 185–190.

Ramos, H., Araújo, A., Silva, J., Neto, J., Teixeira, F., Santos, I., e Andrade, R. (2024). Greatest unity - um jogo digital de cartas para o ensino de testes de software. In *Anais do IV Simpósio Brasileiro de Educação em Computação*, pages 357–366, Porto Alegre, RS, Brasil. SBC.

Rocha, V. C. e Boggio, P. S. (2013). A música por uma Óptica neurocientífica. *Per Musi*, 6(1).

Rêgo, I., Portela, C., e Oliveira, S. (2021). Análise de itens do game design document presentes em jogos da franquia mario da nintendo. In *Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 30–39, Porto Alegre, RS, Brasil. SBC.

Savi, Rafael, Ulbricht, e Ribas, V. (2008). Jogos digitais educacionais: BenefÍcios e desafios. *Revista Novas Tecnologias na Educação*, 6(1).

Signor, R. d. C. F. (2020). Dislexia do desenvolvimento em abordagem comparada: scoping review de pesquisas produzidas no brasil e na austrália. *Revista Psicopedagogia*, 37(112):74–96.

Silva, F., de Souza, B., e Werner, C. (2021). Catálogo para criação de jogos sérios para sistemas baseados em iot. In *Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 675–678, Porto Alegre, RS, Brasil. SBC.

Silva Lopes, E. (2021). *Dislexia: Do reconhecimento ao tratamento na percepção dos professores da educação básica*. PhD thesis.

Sousa, E. R. (2019). Adultos também se divertem: relatos de jogos musicais como recurso de aprendizagem para o ensino técnico-profissionalizante. In *Congresso Nacional da Associação Brasileira de Educação Musical (ABEM) - GT 2.2 – Ensino e aprendizagem de música em escolas especializadas de música*.

Viana, A. C. V., Martins, A. A. E., Tensol, I. K. V., Barbosa, K. I., Pimenta, N. M. R., e de Souza Lima, B. S. (2020). Autismo: uma revisão integrativa. *Saúde Dinâmica*, 2(3):1–18.

Vohera, C., Chheda, H., Chouhan, D., Desai, A., e Jain, V. (2021). Game engine architecture and comparative study of different game engines. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–6.