FAR/AWAY: JOGO PARA AUXÍLIO NO DESENVOLVIMENTO DO RACIOCÍNIO E LÓGICA

Maria Vitória Macedo de Lima, Mário Luiz Schuster Henkels, Edesio Marcos Slomp, Msc – Orientador; **Roberto Luiz Debarba** - Coorientador

Curso Técnico em Informática
Centro de Educação Profissional de Timbó (CEDUP) – Timbó, SC – Brasil

501000976@estudante.sed.sc.gov.br, 4540047476@estudante.sed.sc.gov.br, 339201@profe.sed.sc.gov.br

Resumo: *Atualmente existe carência de ferramentas intuitivas que auxiliem no desenvolvimento de raciocínio lógico e matemático na educação. Uma forma de prover esse auxílio é por meio de jogos que promovam o desenvolvimento dessas habilidades. Com o objetivo de disponibilizar uma ferramenta intuitiva e divertida que possa auxiliar no desenvolvimento do raciocínio lógico e matemático de crianças e adolescentes, este trabalho apresenta o desenvolvimento de um protótipo de jogo eletrônico no estilo roguelite implementado através da plataforma Game Maker Studio 2. Este trabalho também descreve as ferramentas utilizadas no desenvolvimento deste protótipo para a criação de arte (por meio da ferramenta online Pixilart) e música (por meio da ferramenta FLStudio 20). Ao final da etapa de desenvolvimento, o protótipo desenvolvido acabou atendendo três dos quatro objetivos propostos e produziu bons resultados quanto ao engajamento dos usuários.*

Palavras-chave*: Jogo Eletrônico. Educação. Raciocínio Lógico.*

1 INTRODUÇÃO

Atualmente, existe uma carência de ferramentas educacionais que possibilitem a crianças e jovens desenvolverem raciocínio lógico e matemático. A matemática é comumente ensinada por meio da apresentação de fórmulas matemáticas rígidas e exercícios repetitivos, frequentemente limitando o desempenho de pessoas que têm dificuldade com esse método de ensino.

Os primeiros jogos eletrônicos surgiram na década de sessenta (jogos como Spacewar (1962), Tennis For Two (1958) e Pong (1970)). Inicialmente concebidos como formas pioneiras de entretenimento junto do desenvolvimento de hardware na época, vieram desde então a se tornar uma indústria mundial enorme e parte da vida de bilhões de pessoas. Jogos eletrônicos também foram mostrados para auxiliar no desenvolvimento cognitivo, social e emocional de jovens.

Alguns gêneros populares são jogos de tiro, RPGs (role-playing games), e principalmente no mercado *indie* (mercado de jogos independentes de grandes empresas ou publicadoras) os *Roguelites*. Jogos no estilo *roguelite* se tratam de jogos com fases intermináveis cujo objetivo principal do jogador é chegar o mais longe possível com uma vida única. Geralmente são pareados com elementos de aleatoriedade (de geração de fases e/ou inimigos) e habilidades desbloqueáveis que incentivam a rejogabilidade mesmo após uma derrota.

Tendo em vista o exposto, este trabalho descreve o planejamento e desenvolvimento de um protótipo de jogo eletrônico no estilo *roguelite* que busca ser uma maneira divertida e interativa de auxiliar no desenvolvimento e prática do raciocínio lógico e coordenação motora de crianças e jovens. Adicionalmente, a criação de arte e música para o protótipo como uma maneira diferenciada de complementar a experiência de jogo.

Foi utilizado o *Google Formulários* para a realização de pesquisa para levantamento de requisitos para o projeto, o *GameMaker Studio 2* para o desenvolvimento do protótipo, a ferramenta

online *Pixilart* para produção de sprites e assets e a ferramenta *FLStudio 20* para a produção da música.

2 OBJETIVOS

Os objetivos deste projeto consistem na definição e delimitação do que é planejado para alcançar o final do mesmo, definindo a direção do desenvolvimento do protótipo. Na sequência são listados os objetivos geral e específicos.

2.1 OBJETIVO GERAL

Desenvolver protótipo de aplicativo no segmento de jogos do tipo *indie* que possibilite a crianças e jovens desenvolverem o raciocínio e a lógica.

2.2 OBJETIVOS ESPECÍFICOS

- Possuir assets, efeitos sonoros e música que complementem a experiência.
- Estruturar o protótipo por fases intermináveis onde é necessário sobreviver pelo maior tempo possível.
- Possibilitar a escolha de diferentes níveis de dificuldade a fim de melhorar a experiência do usuário com o jogo.
- Garantir ao usuário desbloquear habilidades variadas à medida que o mesmo joga.

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos condizentes com jogos matemáticos e o que são *roguelites* (e sua diferenciação de *roguelikes*). Além disso, serão apresentados alguns benefícios decorrentes da utilização de jogos na educação.

3.1 JOGOS MATEMÁTICOS EDUCACIONAIS

Devido aos avanços da tecnologia de informação, especialmente da área da informática, os jogos vêm sendo tratados como maneiras de ensinar, já que estão presentes no dia a dia dos adultos e das crianças. Assim, o uso de jogos digitais na educação infantil torna-se uma oportunidade de usar tais habilidades para promover um aprendizado de alta performance.Como boa forma de uso na aprendizagem existem os jogos matemáticos digitais, que desenvolvem o raciocínio lógico e suas habilidades; levam-nas a entender a disciplina de matemática como prazerosa e não como um “dever”, permitindo um maior envolvimento dos alunos, por serem representativos, “atraentes” e “divertidos”, proporcionando também a criação de vínculos positivos na relação professor e aluno e alunos com a sua turma em geral.

Esta associação positiva é de extrema importância, visto que o ambiente escolar na contemporaneidade não se apresenta, em parte, o lugar mais propício para atender às necessidades dos alunos que têm acesso aos ambientes virtuais e que após a pandemia, os resultados pioraram em respeito à disciplina de matemática (KUHFELD et al., 2020, p. 2). Esses problemas começam logo no início da trajetória escolar, nos primeiros anos de ensino fundamental, continuam e se

aprofundam conforme os alunos vão passando de série, chegando ao ensino médio com indicadores preocupantes.

E uma proposta bem válida para tornar essa realidade diferente e melhorar o nível desses índices são os jogos digitais infantis, que possuem o objetivo de auxiliar nos aspectos mais importantes nas disciplinas de matemática, como soma, subtração e raciocínio lógico ou ainda desenvolver habilidades que ampliem, nos alunos, as capacidades intelectual e cognitiva.

Jogos como forma de aprendizagem da matemática e a lógica nas escolas, representa, uma mudança de postura do professor em relação ao o que é ensinar matemática, sendo assim, o papel do professor muda de comunicador de conhecimento para o de observador, organizador, consultor e incentivador da aprendizagem, do processo de construção.Um aspecto importante para incrementar as discussões sobre estratégias é o registro das jogadas através de rankings, pontuações e conquistas tanto as eficientes como as frustrantes. Tendo em mãos o jogo apropriadamente experimentado, torna-se mais fácil a análise do mesmo. Vale ressaltar que o sucesso não é imediato e o professor deve ter paciência para colher os frutos desse trabalho. Um cuidado metodológico que o professor deve considerar antes de levar os jogos digitais para a sala de aula, é o de estudar previamente o jogo que será apresentado em sala, o que só é possível jogando. Através da exploração e análise de suas próprias jogatinas e da reflexão sobre seus erros e acertos é que o professor terá condições de colocar questões que irão auxiliar seus alunos e ter noção das dificuldades que irão encontrar. O educador continua indispensável, é ele quem cria as situações, incentiva a evolução e o progresso nele e organiza exemplos que levem à reflexão e obriguem ao controle das soluções demasiado apressadas.

3.2 ROGUELITE

Roguelites são jogos com fases intermináveis cujo objetivo principal do jogador é chegar o mais longe possível com uma vida única. Geralmente são pareados com elementos de aleatoriedade (de geração de fases e/ou inimigos) e habilidades desbloqueáveis que incentivam a rejogabilidade mesmo após uma derrota.

Roguelites se diferenciam de Roguelikes, gênero que o precede, por preservar algum tipo de progresso entre as várias vezes que um usuário joga, seja por meio de habilidades desbloqueáveis ou upgrades incrementais, geralmente com a função de incentivar a rejogabilidade (STUART, 2021).

Uma das principais vantagens deste gênero é a eficiência do trabalho, já que possibilita a um desenvolvedor criar um jogo conciso e dinâmico que seja jogado várias e várias vezes. Jogos deste gênero costumam ser extremamente populares com seu público alvo, uma vez que sua estrutura cíclica interminável facilita o engajamento de jogadores por longos períodos de tempo. É simples visualizar este engajamento obtendo os números de jogadores concorrentes de diferentes jogos durante o mesmo período de tempo. Em agosto de 2022, The Binding Of Isaac: Rebirth (2014), um jogo indie, conseguiu quatro mil jogadores concorrentes a mais que Red Dead Redemption 2 (2019), jogo enorme lançado pela Rockstar, umas das desenvolvedoras de jogos líderes do mercado.

3.3 DESENVOLVIMENTO DE HABILIDADES POR MEIO DE JOGOS

Com a facilidade de acesso a internet e a tecnologia em geral, foram-se ampliando os conceitos da qualidade e os desafios que as mesmas possam trazer. Quando utilizados por crianças, esses suportes podem ainda ser ótimos aliados na descoberta e no desenvolvimento de habilidades diversas. Atualmente os jogos não são apenas entretenimento, mas recursos promotores para desenvolver habilidades. Dentre essas habilidades, jogar pode proporcionar o estímulo da

criatividade, pensamento abstrato, além de exercitar o raciocínio lógico como também a coordenação motora o que Diz respeito à destreza manual, onde os movimentos são mais específicos e envolvem pequenos grupos musculares como (mãos/dedos) através de uma pressão dos mesmos em um objeto (LOBO; VEGA, 2008).

Em relação ao estímulo do raciocínio lógico, sendo um dos mais importantes é possível com eles exercitar o cérebro e o aprendizado independente da idade. Esse tipo de recurso pode ser convertido para o ambiente escolar e fazer nas escolas, por meio de ferramentas que oferecem uma atitude positiva em relação ao interesse dos estudantes com qualquer disciplina, como nas física, matemática e química que costumam a ser as menos queridas pelos estudantes a gamificação pode contribuir para a melhoria do resultado do desempenho delas, como em jogos que exigem do jogador concentração e rapidez para que as decisões tomadas ao longo da brincadeira sejam acertadas.

Essa característica está presente na maioria dos jogos e é o que faz com que eles auxiliem no desenvolvimento do raciocínio lógico, dentre esses jogos temos como exemplo “Minecraft” que pode estimular a lógica e desenvolver a criatividade.

Assim que se tem em base que pode-se aprender com os games, além de ser uma maneira de lazer, mas também beneficiar na vida escolar ou profissional, tornou-se uma tendência da educação na era digital como estratégia pedagógica.

4 ESPECIFICAÇÃO

Na especificação deve ser apresentada toda a análise do sistema proposto, através de modelos e/ou diagramas que representem logicamente o trabalho desenvolvido, usando ferramentas de desenvolvimento e análise. Exemplos: diagramas de casos de uso, diagramas de classes, diagramas de atividades, etc. Exibir apenas os diagramas mais relevantes ou mais complexos, sempre com descrições textuais. Pseudo-códigos podem ser usados também.

4.1 REQUISITOS

Por meio de uma pesquisa de requisitos para o protótipo produzido ao decorrer deste projeto, foram definidos os requisitos funcionais (RF) apresentados no Quadro 1.

Quadro 1 – Requisitos funcionais (RF) e relação aos casos de uso

Requisito funcional	Caso de uso
RF01 – Manter Pontuações	UC01-UC04
RF02 – Possuir Menu Principal	UC01
RF03 – Manter Configuração (dificuldade)	UC05
RF04 – Ter fases intermináveis, com dificuldade que cresce à medida do tempo de jogo	UC01
RF05 – Possibilitar ao usuário desbloquear habilidades baseado em seu progresso	UC01-UC03
RF06 – Calcular pontuação baseado em um contador incrementável	UC01-UC02
RF07 – Apresentar ao usuário inimigos de cores distintas, que representam a operação que será aplicada no contador incrementável (Azul / Vermelho)	UC01
RF08 – Parametrizar sistema	UC06

Fonte: elaborado pela equipe.

Em complemento, o Quadro 2 apresenta as definições de disponibilidade, compatibilidade, forma de acesso e tecnologias envolvidas, através dos requisitos não funcionais (RNF)

Quadro 2 – Requisitos não funcionais (RNF)

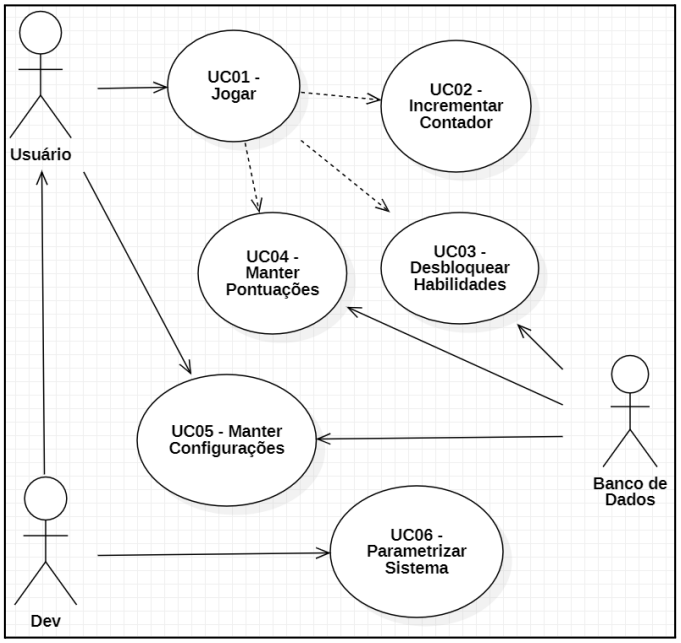
Requisito não funcional
RNF01 – Possuir trilha sonora original
RNF02 – Ter estilo de arte em pixel
RNF03 – Possibilitar ajuste dos volumes de música e efeitos
RNF04 – Possuir níveis de dificuldade mapeados pela idade do jogador
RNF05 – Possibilitar remapeamento de teclas
RNF06 – Ser desenvolvido como jogo roguelite

Fonte: elaborado pela equipe.

4.2 DIAGRAMA DE USE-CASE

A partir da definição dos Requisitos Funcionais (RF) apresentados no Quadro 1, foi produzido o Diagrama Use Case (UC) apresentado abaixo na Figura 1. Nele, é possível visualizar os níveis de interação que diversos atores exercem sobre o sistema do protótipo. Por exemplo, vê-se que o Usuário pode jogar o jogo, e por meio de estar jogando o mesmo pode realizar as ações de Incrementar o contador ou Desbloquear Habilidades. Além disso, o Banco de Dados atua em todas as atividades onde se torna necessário o mantimento de dados.

Figura 1 – Diagrama de Use-Case



Fonte: elaborado pela equipe.

4.2.1 Notações de Use-Case

Assim que terminado o Diagrama de Use-Case, torna-se possível a elaboração das Notações de Use-Case (UC), buscando descrever cada Use-Case relevante, quais os requisitos que este atende, os atores que podem participar nele, além de quaisquer cenários principal e alternativos que devem ser tratados durante a implementação do mesmo.

O Quadro 3 apresenta os seguintes casos de uso: 'UC01 - Jogar', que descreve o principal *gameplay loop* (loop de gameplay, as características principais que o jogador deve ter em mente e que informam sua tomada de decisões momento-a-momento enquanto joga o jogo) do protótipo; 'UC02 - Incrementar Contador', que especifica como o jogador deverá interagir com o contador Incremental durante o jogo; 'UC05 - Manter Configurações', que define como o usuário poderá customizar as configurações do jogo por meio do menu principal.

Quadro 3 – Notações Use-Case (UC)

UC01 – Jogar	
REQUISITOS ATENDIDOS	RF01, RF02, RF04, RF05, RF06, RF07.
ATORES ENVOLVIDOS	Player, Dev, Banco de Dados.
PRÉ-CONDIÇÃO	Player deve ter clicado em "Start" no Menu Principal
CENÁRIO PRINCIPAL	Ao Player é apresentada a tela principal do jogo, que consiste em: ● Imagem de fundo representando o espaço sideral; ● Nave representando o jogador em si; ● Quaisquer naves inimigas ou asteroides que venham a aparecer no decorrer do jogo; ● Timer representando o tempo de jogo; ● Contador reativo no centro superior da tela; ● Barra vermelha no centro inferior da tela, representando a quantidade de vida do Player; ● Barra azul logo acima da barra de vida, representando a quantidade de escudo do Player; ● Música reativa que pode aumentar ou diminuir de intensidade baseado nas ações do Player;
CENÁRIO ALTERNATIVO 01	O Player pode controlar sua nave, movendo-a pela tela utilizando os botões padrão W, A, S e D, atirando com o botão padrão esquerdo do mouse ou utilizando uma habilidade com o botão padrão R, além de poder ativar um modo de tiro especial pela Barra de Espaço
CENÁRIO ALTERNATIVO 02	Caso um inimigo derrote o Player, este será levado à tela 'Game Over', onde verá e poderá salvar sua pontuação, além de poder reiniciar o jogo ou voltar ao Menu Principal.
CENÁRIO ALTERNATIVO 03	O objetivo do Player é sobreviver o maior tempo possível enquanto é atacado constantemente por ondas de inimigos, podendo aumentar sua pontuação atingindo o número-alvo do Contador o maior número de vezes possível.

UC02 – Incrementar Contador	
REQUISITOS ATENDIDOS	RF01, RF06, RF07
ATORES ENVOLVIDOS	Player, Dev
PRÉ-CONDIÇÃO	Usuário deve ter clicado em "Start" no Menu Principal
CENÁRIO PRINCIPAL	Durante o jogo, será apresentado ao Player um contador no centro superior da tela com o seguinte formato: <número atual> / <número alvo> Número Atual: iniciará como zero e poderá ser incrementado sempre que o Player derrotar um inimigo. Número Alvo: será gerado automaticamente baseado nas opções de dificuldade do

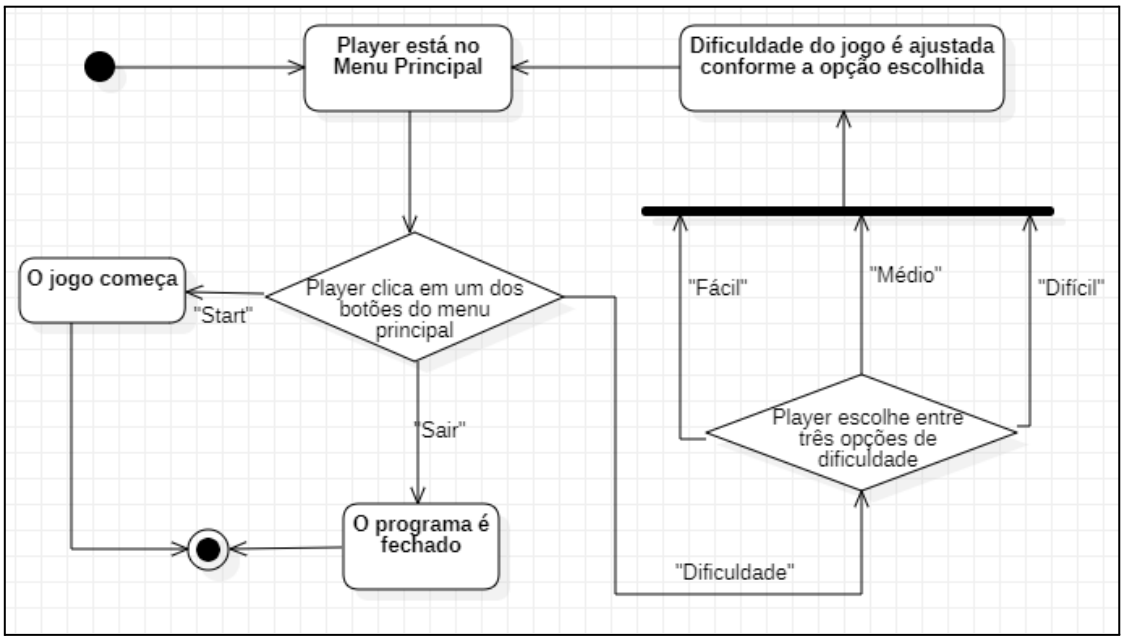
	jogo e no número atual.
CENÁRIO ALTERNATIVO 01	Cada inimigo possui um tamanho, que representa o seu valor numérico: ● Minúsculo: 1 ● Pequeno: 5 ● Médio: 10 ● Grande: 20 Cada inimigo também possui uma cor, que representa a operação que será aplicada ao contador com seu valor numérico caso o jogador o derrotar: ● Azul: Soma ● Vermelho: Subtração ● Cinza: Não realiza operação
CENÁRIO ALTERNATIVO 02	Caso o Player derrote um inimigo, este explodirá e será aplicada ao contador a operação definida pela cor do inimigo com o valor definido por seu tamanho. (Ex. um inimigo Azul de tamanho Médio, quando derrotado, adiciona +10 ao número do contador)
CENÁRIO ALTERNATIVO 03	Quando o Número Atual se iguala ao Número Alvo, um valor será incrementado à pontuação do Player e um novo Número Alvo será gerado.

UC05 - Manter Configurações	
REQUISITOS ATENDIDOS	RF03
ATORES ENVOLVIDOS	Player e Dev
PRÉ-CONDIÇÃO	Player deve estar no Menu Principal
CENÁRIO PRINCIPAL	No Menu Principal do jogo, haverá uma opção chamada “Dificuldade” com três opções: Fácil, Médio ou Difícil.
CENÁRIO ALTERNATIVO 01	Ao escolher uma das opções, alguns parâmetros do jogo serão afetados baseado na dificuldade escolhida, estes são: ● Frequência na qual novos inimigos são introduzidos no jogo (quanto maior a dificuldade, inimigos são criados mais rapidamente) ● Limite máximo de inimigos que podem estar ativos (vivos) a qualquer momento (Durante o jogo, existe um limite para quantos inimigos podem estar ativos ao mesmo tempo. Quanto maior a dificuldade do jogo, maior será esse limite)

4.3 DIAGRAMA DE ATIVIDADES

Na Figura 2 é apresentado o Diagrama de Atividades referente à navegação pelo Menu Principal. O mesmo busca descrever a atividade especificada em detalhe, mostrando quais as escolhas e opções que o usuário tem disponíveis dependendo de em qual tela do Menu Principal o mesmo se encontra.

Figura 2 – Diagrama de atividade



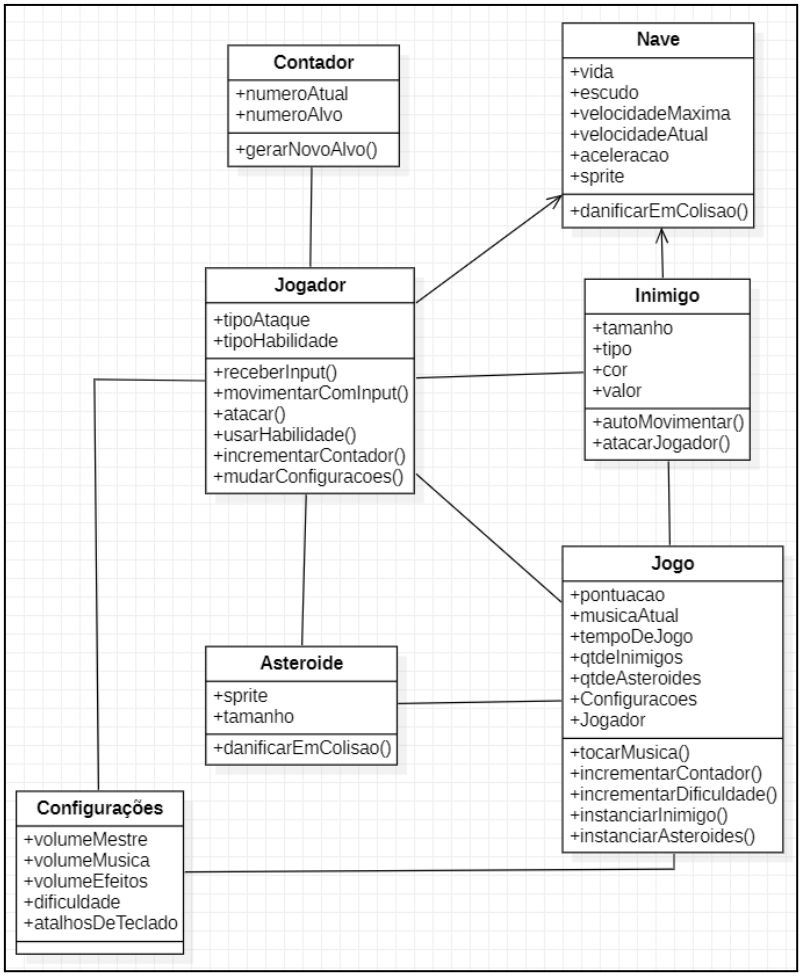
Fonte: elaborado pela equipe.

4.4 DIAGRAMA DE CLASSES

O Diagrama de Classes apresentado na Figura 3 busca nortear a implementação do código do protótipo, descrevendo as principais Classes a serem implementadas, seus diversos atributos e principais métodos. No decorrer do desenvolvimento do protótipo em si, as classes apresentadas foram aproximadamente codificadas como 'Objetos' dentro da plataforma Game Maker Studio 2.

Visualizando o diagrama, pode-se inferir que o mesmo foca em 'Jogador' como principal classe dentro do protótipo, já que a mesma interage com todas as outras de formas variadas. Por exemplo, o jogador pode se movimentar pela tela, incrementar o contador, atacar e ser atacado por inimigos ou modificar as configurações do sistema.

Figura 3 – Diagrama de classes



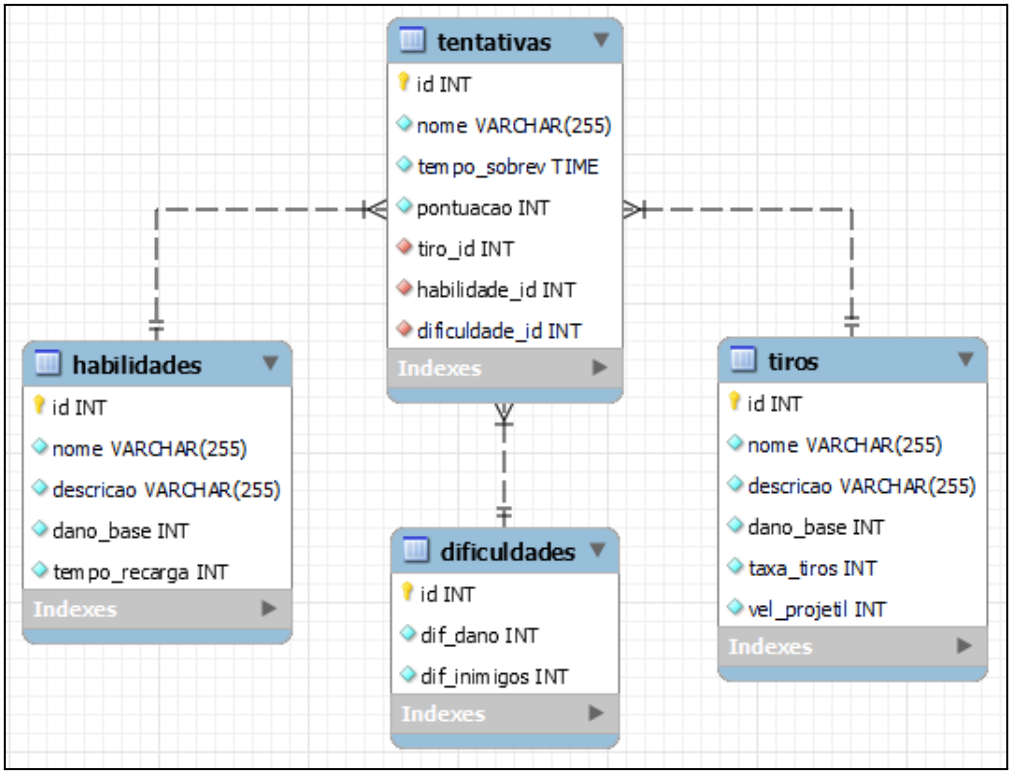
Fonte: elaborado pela equipe.

4.5 BANCO DE DADOS

O Modelo Físico do Banco de Dados busca descrever a estrutura do banco de dados do protótipo, as principais tabelas, suas colunas e relacionamentos entre tais tabelas.

Dada a natureza do jogo eletrônico proposto, sendo um jogo *roguelite* independente de escopo pequeno, foi delimitado que as únicas informações interessantes a serem mantidas por um banco de dados seriam informações específicas relacionadas às tentativas, ou seja, a cada vez que um jogador joga o jogo, do momento em que o jogo inicia até o momento em que o jogador é inevitavelmente derrotado. Tais incluem o tempo sobrevivido, que habilidade o jogador tinha equipada durante esta tentativa, a dificuldade do jogo e, claro, sua pontuação final. É possível visualizar na Figura 4 o modelo físico do Banco de Dados, e no Quadro 4 o código SQL utilizado para a criação das tabelas.

Figura 4 – Modelo Físico do Banco de Dados



Fonte: elaborado pela equipe.

Quadro 4 – Código SQL do Banco de Dados

```
create database if not exists far_away_db;

use far_away_db;

create table if not exists dificuldades (
    id int not null auto_increment unique,
    dif_dano int not null,
    dif_inimigos int not null,
    primary key (id)
);
```

```
create table if not exists tiros (
    id int not null auto_increment unique,
    nome varchar(255) not null,
    descricao varchar(255) not null,
    dano_base int not null,
    taxa_tiros int not null,
    vel_projetil int not null,
    primary key (id)
);
create table if not exists habilidades (
    id int not null auto_increment unique,
    nome varchar(255) not null,
    descricao varchar(255) not null,
    dano_base int not null,
    tempo_recarga int not null,
    primary key (id)
);
create table if not exists tentativas (
    id int not null auto_increment unique,
    nome varchar(255) not null unique,
    tempo_sobrev time not null,
    pontuacao int not null,
    tiro_id int not null,
    habilidade_id int not null,
    dificuldade_id int not null,
    primary key (id),
    foreign key (tiro_id) references tiros(id),
    foreign key (habilidade_id) references habilidades(id),
    foreign key (dificuldade_id) references dificuldades(id)
);
```

5 DESENVOLVIMENTO

Buscando atender aos requisitos apresentados, foi desenvolvido um protótipo de jogo *roguelite* que mira disponibilizar um jeito divertido e interativo de se desenvolver habilidades matemáticas e raciocínio lógico.

Durante o jogo, o objetivo do usuário é sobreviver o maior tempo possível enquanto é atacado constantemente por ondas de inimigos, podendo aumentar sua pontuação atingindo o número-alvo do contador o maior número de vezes possível.

Para o desenvolvimento do protótipo, foram elencados as principais funcionalidades a serem implementadas:

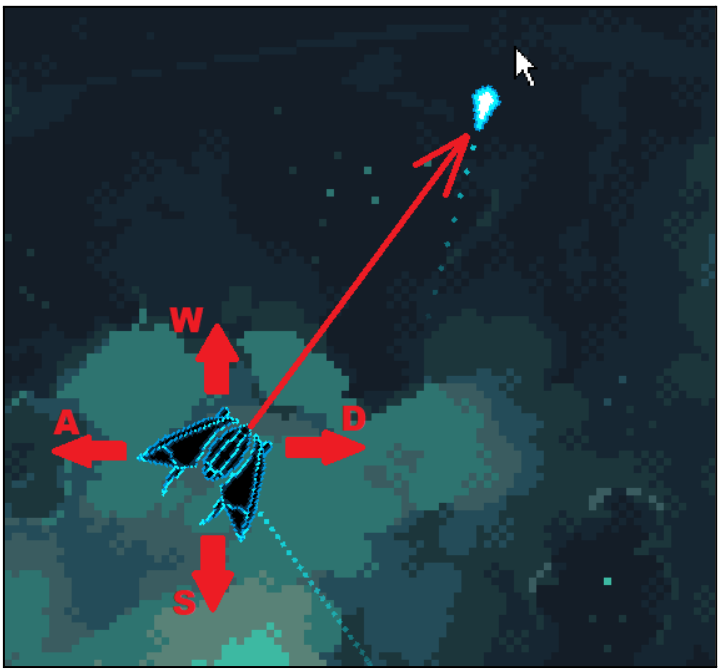
- Esquema de movimentação e mira simples e altamente responsivo.
- Inimigos sendo criados continuamente.
- Colorir sprites de inimigos por meio de shader, possibilitando a mudança de cores e consequentemente o valor numérico de inimigos a qualquer momento durante o jogo.
- Possibilitar ao usuário interagir com o contador derrotando inimigos.
- Música original, que aumenta ou diminui em intensidade dependendo de fatores do jogo.

Durante o jogo, a nave que representa o jogador sempre se orientará e atirará na direção do mouse e pode ser movimentada, na perspectiva da câmera, para cima, para a esquerda, para baixo ou para a direita respectivamente pelos botões W, A, S, D, como pode ser observado na Figura 5.

É importante mencionar que, para reproduzir a sensação de que a nave esteja flutuando pelo espaço, não existe fricção. Ou seja, quando o jogador está se movendo em uma direção, mesmo se o

mesmo não estiver tocando o teclado, ele continuará se movendo na mesma direção em velocidade contínua.

Figura 5 – Esquema de Movimentação do Jogador



Fonte: elaborado pela equipe.

Na função apresentada no Quadro 5, é possível visualizar como é efetuada a movimentação do jogador em nível de código. Utilizando parâmetros que descrevem a direção de movimento, além da aceleração e velocidade máxima que o jogador pode atingir, é utilizada a função nativa *motion_add()* do Game Maker Studio 2 para impulsionar o jogador na direção desejada. Adicionalmente, é feita uma validação para certificar que o jogador nunca pode exceder sua velocidade máxima de movimento. Esta função é chamada todo *frame* de execução do jogo, sessenta vezes por segundo.

Quadro 5 – Função que realiza a movimentação do jogador

```
/// @function                    handlePlayerMovement(moveX, moveY, accel, flightSpd)
/// @param    {int} moveX        1, -1 ou 0 caso direita, esquerda ou nada
/// @param    {int} moveY        1, -1 ou 0 caso baixo, cima ou nada
/// @param    {real} accel       Aceleração
/// @param    {real} flightSpd   Velocidade Máxima

function handlePlayerMovement(moveX, moveY, accel, flightSpd){

	switch(moveX) {
	    case 1:
	              motion_add(0, accel);
	              break;
	    case -1:
	              motion_add(180, accel);
	              break;
	}

	switch(moveY) {
	    case 1:
	           motion_add(270, accel);
	           break;
	    case -1:
```

```
                motion_add(90, accel);
                break;
        }

        if(speed > flightSpd){
            speed = flightSpd;
        } else if (speed < -flightSpd) {
            speed = -flightSpd
        }
}
```

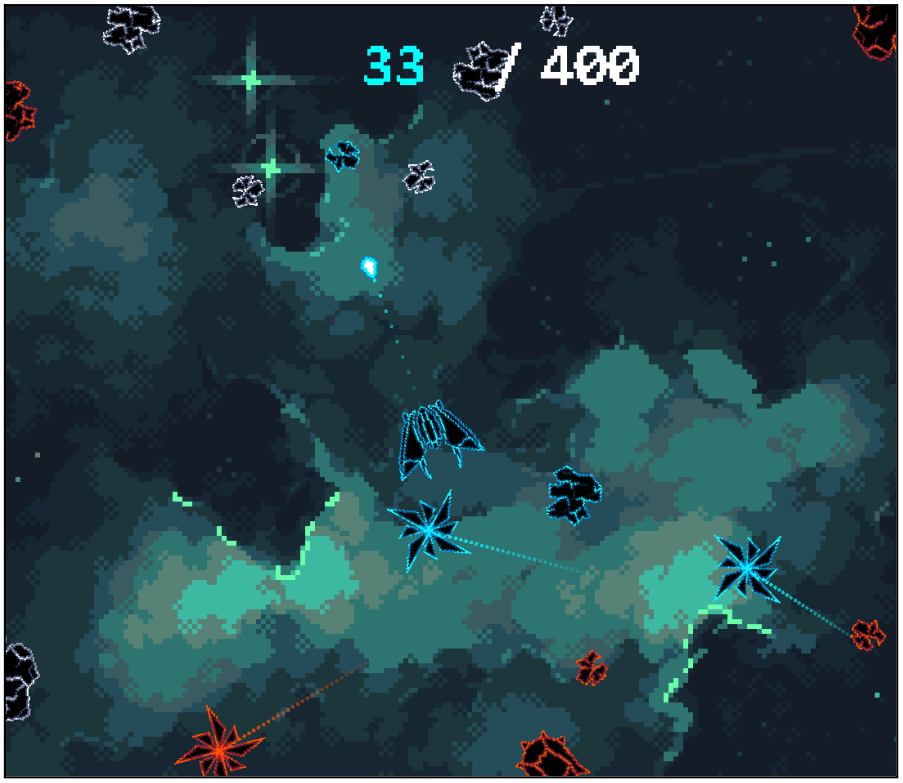
Fonte: elaborado pela equipe.

Como de costume nos jogos *roguelite,* o protótipo é estruturado como uma fase única contínua, que só termina quando o jogador é derrotado. Para que isso seja possível, é necessário que inimigos sejam constantemente instanciados nos cantos da tela à medida do tempo (visível na Figura 7), levando em conta a dificuldade escolhida para delimitar a frequência na qual novos inimigos poderão ser introduzidos no mundo.

Os inimigos que foram implementados no protótipo são:

- Rochas de diversos tamanhos, que quando atingidas criam duas outras rochas de tamanho menor.
- *Estrelas-ninja* rodopiantes que viajam em alta velocidade.
- *Naves-bomba* que quando atingidas explodem, atirando espinhos em um arco à sua frente.

Figura 6 – Inimigos implementados no protótipo



Fonte: capturado pela equipe.

Abaixo, o Quadro 6 apresenta a função utilizada para criar novos inimigos nos cantos da tela. Ela recebe o objeto (inimigo) a ser criado, a quantidade de inimigos deste tipo que devem ser criados, o *Layer* do GameMaker onde os objetos serão armazenados e calcula pontos aleatórios nas extremidades da tela nos quais cada objeto será finalmente instanciado. Esta função é chamada

periodicamente em um intervalo de tempo que depende da dificuldade escolhida.

Quadro 6 – Função que cria inimigos nos cantos da tela

```
///@function                          spawnRandomlyBorders(object, quantity, layer)
///@param {obj} object                O objeto a ser spawnado
///@param {int} quantity              Quantos deste objeto devem ser spawnados
///@param {string} layer              O Layer onde o objeto será colocado

function spawnRandomlyBorders(object, quantity, layer){
	var padding = 10;

	repeat(quantity){
		if(choose(true,false)){
			//lados
			var xx = choose(-padding, room_width + padding);
			var yy = irandom_range(0, room_height);
		} else {
			//acima / abaixo
			var xx = irandom_range(0, room_width);
			var yy = choose(-padding, room_height + padding);
		}

		instance_create_layer(xx, yy, layer, object)
	}
}
```

Fonte: elaborado pela equipe.

Durante a execução do protótipo, todos os inimigos possuem valores numéricos e, dependendo da cor de seus *sprites*, incrementarão seus respectivos valores ao contador por meio de adição (caso o inimigo seja azul) ou subtração (caso o mesmo seja vermelho). Inimigos cinza não interagem com o contador, servindo apenas como uma forma de aumentar a pontuação e aumentar um pouco a variedade de inimigos.

Dado o fato do objetivo do jogador ser atingir o valor numérico alvo do contador, em qualquer momento do jogo o mesmo geralmente procurará realizar apenas adição ou apenas subtração, o que significa que ele procurará atirar apenas nos inimigos azuis, ou apenas nos inimigos vermelhos. Para dar ao jogador mais controle sobre o ambiente do jogo e prevenir situações onde apenas inimigos de uma cor existem, foi introduzida uma habilidade que permite ao jogador trocar a cor de todos os inimigos na tela (inimigos azuis viram vermelhos e vice-versa).

No Quadro 7 é possível visualizar o *shader* utilizado por todos os objetos no protótipo. Através dele, é possível delimitar quais cores no sprite de um objeto serão substituídas por outras utilizando os valores R-G-B dos pixels individuais.

Quadro 7 – Shader utilizado por todos os objetos no protótipo

```
uniform float range;
uniform vec4 colorMatch1;
uniform vec4 colorMatch2;
uniform vec4 colorMatch3;
uniform vec4 colorMatch4;
uniform vec4 colorReplace1;
uniform vec4 colorReplace2;
uniform vec4 colorReplace3;
uniform vec4 colorReplace4;

void main()
{
	vec4 pixelColor = v_vColour * texture2D( gm_BaseTexture, v_vTexcoord );
```

```
	float newRange = range / 255.0;

	if(abs(pixelColor.r - colorMatch1.r) <= newRange
		&& abs(pixelColor.g - colorMatch1.g) <= newRange
		&& abs(pixelColor.b - colorMatch1.b) <= newRange){//substitui a cor luz

				pixelColor.rgb = colorReplace1.rgb;

	} else if(abs(pixelColor.r - colorMatch2.r) <= newRange
		&& abs(pixelColor.g - colorMatch2.g) <= newRange
		&& abs(pixelColor.b - colorMatch2.b) <= newRange){//substitui a cor base

				pixelColor.rgb = colorReplace2.rgb;

	} else if(abs(pixelColor.r - colorMatch3.r) <= newRange
		&& abs(pixelColor.g - colorMatch3.g) <= newRange
		&& abs(pixelColor.b - colorMatch3.b) <= newRange){//substitui a cor dark

				pixelColor.rgb = colorReplace3.rgb;

	} else if(abs(pixelColor.r - colorMatch4.r) <= newRange
		&& abs(pixelColor.g - colorMatch4.g) <= newRange
		&& abs(pixelColor.b - colorMatch4.b) <= newRange){//override red to white

				pixelColor.rgb = colorReplace4.rgb;
	}

    gl_FragColor = pixelColor;
}
```

Fonte: elaborado pela equipe.

Já o Quadro 8 apresenta parte da função que interage com esse *shader,* delimitando as novas cores do sprite. Essa função pode ser chamada a qualquer momento por um objeto e trocará a cor do mesmo pelas cores passadas a ela como parâmetros.

Quadro 8 – Função que altera as cores do shader de um objeto.

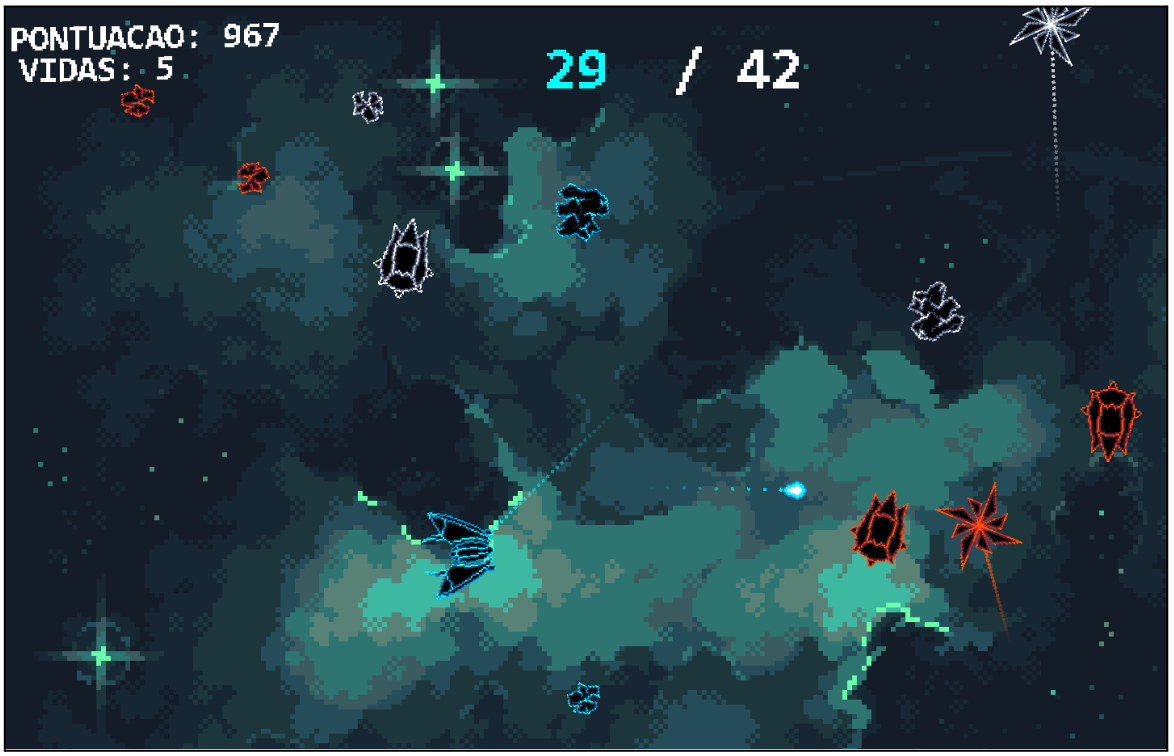
```
///@function                          replaceColor(colorLight,colorBase, colorDark)
///@param {Color} colorLight          nova cor clara      (macro no scr_color)
///@param {Color} colorBase           nova cor base       (macro no scr_color)
///@param {Color} colorDark           nova cor escura     (macro no scr_color)

function replaceColor(colorLight, colorBase, colorDark){

	colorMatch1 = grey_light;
	colorMatch2 = grey_base;
	colorMatch3 = grey_dark;
	colorMatch4 = white_override;

	colorReplace1 = colorLight;
	colorReplace2 = colorBase;
	colorReplace3 = colorDark;
	colorReplace4 = white;

	sh_handle_range = shader_get_uniform(sh_replaceColor, "range");
	sh_handle_match_1 = shader_get_uniform(sh_replaceColor, "colorMatch1");
	sh_handle_match_2 = shader_get_uniform(sh_replaceColor, "colorMatch2");
	sh_handle_match_3 = shader_get_uniform(sh_replaceColor, "colorMatch3");
	sh_handle_match_4 = shader_get_uniform(sh_replaceColor, "colorMatch4");
	sh_handle_replace_1 = shader_get_uniform(sh_replaceColor, "colorReplace1");
```

```
        sh_handle_replace_2 = shader_get_uniform(sh_replaceColor, "colorReplace2");
        sh_handle_replace_3 = shader_get_uniform(sh_replaceColor, "colorReplace3");
        sh_handle_replace_4 = shader_get_uniform(sh_replaceColor, "colorReplace4");

        shader_set(sh_replaceColor);

        shader_set_uniform_f(sh_handle_range, 1);

        shader_set_uniform_f(
               sh_handle_match_1,
               colorMatch1.toShaderVal(colorMatch1.red),
               colorMatch1.toShaderVal(colorMatch1.green),
               colorMatch1.toShaderVal(colorMatch1.blue)
        );
        shader_set_uniform_f(
               sh_handle_match_2,
               colorMatch2.toShaderVal(colorMatch2.red),
               colorMatch2.toShaderVal(colorMatch2.green),
               colorMatch2.toShaderVal(colorMatch2.blue)
        );
```

Fonte: elaborado pela equipe.

Na Figura 7 pode-se conferir a aparência do protótipo ao final da etapa de Desenvolvimento. A todo momento, é apresentado ao jogador os números atual e alvo do contador, além de sua pontuação e suas vidas. O usuário pode incrementar sua pontuação derrotando inimigos, sendo que derrotar inimigos coloridos também incrementará seus valores no contador. Quando o contador chegar em seu valor alvo, a pontuação será incrementada em cem pontos e o jogador receberá uma vida adicional.

Figura 7 – Imagem do protótipo em execução

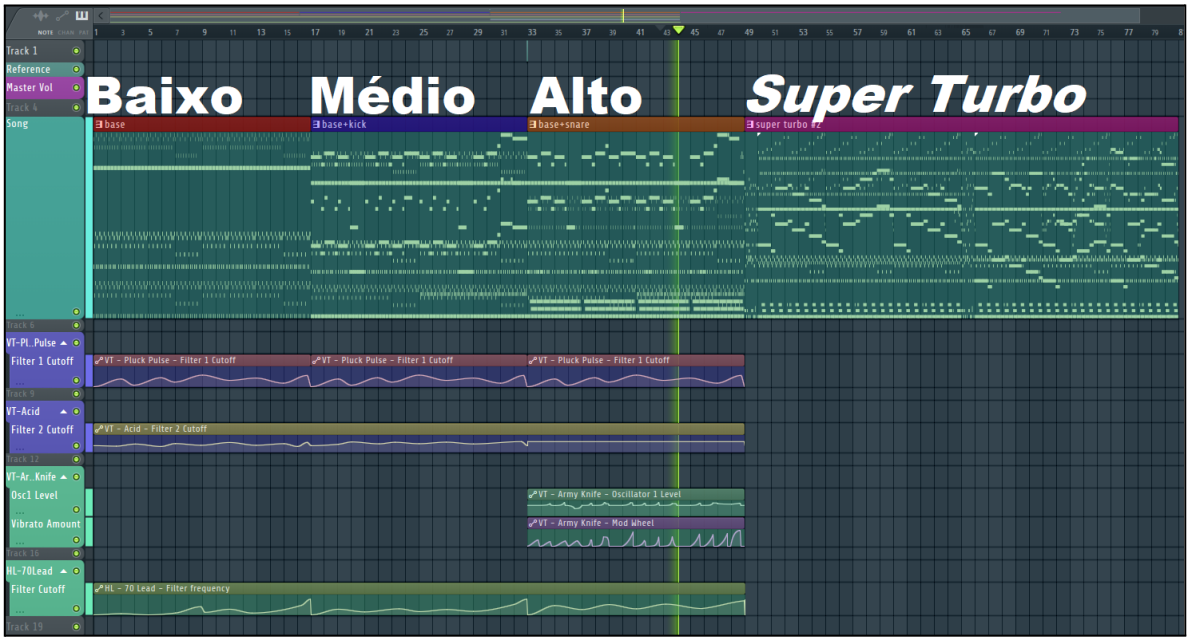


Fonte: capturado pela equipe.

5.1 TRILHA SONORA

Para que a música fosse reativa durante a execução do protótipo, foi necessário produzi-la com uma estrutura específica, como se pode conferir na Figura 8. Qualquer música utilizada durante o combate teve de ser composta de quatro *loops* (repetições) distintos, com tamanhos iguais ou múltiplos, cada um podendo repetir-se indefinidamente. Tais *loops* representam a mesma música em diferentes níveis de intensidade que refletem o nível de perigo no qual o jogador se encontra, tais sendo baixo, médio, alto e altíssimo (ou *"super turbo"*).

Figura 8 – Música de combate estruturada em quatro seções distintas



Fonte: elaborado pela equipe.

Durante o jogo todos os quatro *loops* tocam simultaneamente, tendo seus volumes controlados para que o jogador ouça apenas um em cada momento. Assim, é possível mudar a intensidade da música aumentando e diminuindo os volumes de cada *loop* conforme o necessário. Algumas das funções utilizadas no controle da música podem ser observadas no Quadro 9.

Quadro 9 – Funções controladoras da música.

```
///@function                              musicIncreaseIntensity()

function musicIncreaseIntensity() {

       switch(global.currentMusic){

              case global.mscLow:
                     audio_sound_gain(global.currentMusic, 0, 500);
                     audio_sound_gain(global.mscMed, 1, 500);
                     global.currentMusic = global.mscMed;
                     break;

              case global.mscMed:
                     audio_sound_gain(global.currentMusic, 0, 500);
                     audio_sound_gain(global.mscHigh, 1, 500);
                     global.currentMusic = global.mscHigh;
                     break;
```

```
            case global.mscHigh:
                  audio_sound_gain(global.currentMusic, 0, 500);
                  audio_sound_gain(global.mscTurbo, 1, 500);
                  global.currentMusic = global.mscTurbo;
                  break;
      }
}

///@function                           musicGoTurbo()

function musicGoTurbo() {
      audio_sound_gain(global.currentMusic, 0, 500);
      audio_sound_gain(global.mscTurbo, 1, 500);
      global.currentMusic = global.mscTurbo;
}
```

Fonte: elaborado pela equipe.

6 RESULTADOS

Durante o desenvolvimento deste projeto, foi possível confirmar a ausência de ferramentas interativas e divertidas para o desenvolvimento do pensamento matemático nas escolas e a demanda por jogos que possam prover essa função por meio do questionário utilizado no levantamento de requisitos, onde 63,8% das respostas consideram jogos como uma forma de escapismo de atividades escolares e 93,6% das respostas foram positivas em relação à utilização de jogos no auxílio do aprendizado de pessoas que sentem dificuldade na escola.

A plataforma Game Maker Studio 2 provou-se muito eficaz para o desenvolvimento do protótipo. Mesmo sem experiência prévia com tal ferramenta ou com sua linguagem de programação própria, a linguagem GML, a equipe foi capaz de aprendê-la rápido e utilizar de sua arquitetura simples e de muitas funções próprias para rapidamente implementar funcionalidades sem se preocupar muito com a arquitetura básica do protótipo.

O resultado da etapa de desenvolvimento foi um jogo 2D do tipo *indie* onde o jogador toma controle de uma nave no espaço sideral, tendo o objetivo de sobreviver o maior tempo possível enquanto sendo constantemente atacado por naves inimigas e asteroides, podendo aumentar sua pontuação realizando operações de soma e subtração derrotando inimigos de cores distintas visando chegar em um número alvo no contador mostrado no centro superior da tela. O jogo também possui três níveis de dificuldade, fácil, médio e difícil, o que aumenta sua acessibilidade para usuários de diferentes idades e níveis de proficiência. Além disso, o jogo apresenta arte e música originais, o que proporciona ao mesmo uma identidade única, provendo ao usuário uma experiência diferenciada.

7 CONCLUSÕES

Através do desenvolvimento deste trabalho, permitiu-se verificar a capacidade do jogo criado de prover uma maneira interativa e divertida de auxiliar no desenvolvimento do raciocínio e lógica de crianças e jovens. A combinação do contador incremental com inimigos sendo criados continuamente encoraja o jogador a pensar matematicamente, realizando escolhas para chegar ao exato número alvo da maneira mais rápida e eficiente possível. Além disso, a presença de música produzida especificamente para o jogo auxilia na experiência, acentuando momentos de vitória ou tensão como for necessário.

Um dos primeiros objetivos a serem atendidos na fase de implementação foi a estruturação do protótipo por fases intermináveis. A criação contínua de inimigos nas extremidades das telas e a atualização do número alvo do contador assim que o jogador o atinge garantem ao mesmo sobreviver indefinidamente desde que seja proficiente o suficiente no jogo, assim, a fase só terminará quando o jogador for derrotado, atingindo o objetivo apresentado.

O objetivo de possibilitar a escolha de diferentes níveis de dificuldade foi atingido, visto que foi introduzida a opção de dificuldade no menu principal do protótipo, possibilitando ao usuário um nível de controle sobre como o mesmo interage com o jogo. Tal funcionalidade também auxilia na acessibilidade, permitindo que pessoas de variadas idades e níveis de proficiência possam vir a desfrutar do protótipo.

O objetivo de possuir assets, efeitos sonoros e música que venham a complementar a experiência do usuário foi atendido. Através da produção de *sprites* e música originais pelos membros da equipe e pela busca de efeitos sonoros apropriados, foi possível que o protótipo tenha um estilo único e distinto que auxilia na experiência do usuário e na memorabilidade do jogo em si.

O objetivo de garantir ao usuário desbloquear habilidades à medida que joga não foi atendido. Apesar de ter sido uma das primeiras ideias a serem sugeridas pela equipe na conceitualização do protótipo, o nível de complexidade da implementação de tal funcionalidade e a aproximação do prazo de entrega deste projeto levaram a equipe a priorizar o desenvolvimento de outras funções mais críticas para o funcionamento do protótipo, tais como o menu principal e o contador.

REFERÊNCIAS

KUHFELD, Megan *et al*. **Learning during COVID-19**: initial findings on students' reading and math achievement and growth. [S.I]: Nwea Research: Collaborative For Student Growth, 2020. 12 p. Disponível em: https://www.nwea.org/uploads/2020/11/Collaborative-brief-Learning-during-COVID-19.NOV2020.pdf. Acesso em: 08 nov. 2022.

LOBO, A. S.; VEGA, E. H. T. **Educação motora infantil**: orientações a partir das teorias construtivista, psicomotricista e desenvolvimentista motora. Caxias do Sul: Educs, 2008.

STUART, Keith. **Dungeon crawler or looter shooter?** nine video game genres explained. 2021. Disponível em: https://www.theguardian.com/games/2021/oct/11/modern-video-game-genres-explained-metroidvania-dungeon-crawler. Acesso em: 08 nov. 2022.