Instituto Federal de São Paulo (IFSP)

Curso Técnico em Desenvolvimento de Sistemas

Integrantes: Arthur Fukuyama de Andrade; Heitor Akira Isatugo; Leonardo Cabrera Corrêa; Rafael Fujita Masuko

Fear’s Last Frame

São Paulo

2025

Instituto Federal de São Paulo (IFSP)

Curso Técnico em Desenvolvimento de Sistemas

Integrantes: Arthur Fukuyama de Andrade; Heitor Akira Isatugo; Leonardo Cabrera Corrêa; Rafael Fujita Masuko

Fear's Last Frame

Documentação de Projeto Interdisciplinar apresentado como requisito parcial para avaliação nas disciplinas de Lógica de Programação, Princípios de Desenvolvimento Web, Arquitetura de Computadores e Redes e Fundamentos do Desenvolvimento de Sistemas.

Professores (Banca): Ana Lucia Grici Zacarin Mamede; Claudete de Oliveira Alves; Claudia Miyuki Werhmuller; Antonio Ferreira Viana; Daniela dos Santos Santana; André Evandro Lourenço; João Antonio Temochko Andre; Paula Neves de Araújo; Paulo Henrique Netto de Alcantara; Gislene Pereira de Oliveira Martins

São Paulo

2025

1. INTRODUÇÃO

O desafio proposto pelos professores foi criar um jogo do zero, aprendendo uma nova linguagem de programação e colocando em prática o que foi aprendido ao longo do curso. O objetivo deste documento é registrar como foi o processo de criação e desenvolvimento do projeto.

No começo, pensamos em fazer um jogo sobre Pokémon, mas depois decidimos criar algo original. Assim surgiu o *Fear's Last Frame*, um jogo de ficção no estilo RPG, onde o protagonista Felipe, um menino autista, precisa derrotar monstros que aparecem em um parque misterioso. Cada vitória leva o personagem mais perto dos brinquedos e de um novo amigo, representando uma jornada de superação e coragem.

Escolhemos o gênero RPG porque ele permite mais liberdade para criar histórias, personagens e mecânicas diferentes. O público-alvo do jogo são principalmente crianças e jovens, já que é um estilo divertido e cheio de possibilidades.

Neste documento estão descritas as ideias que tivemos, as ferramentas que usamos, como nos organizamos e as etapas que seguimos até o jogo ficar pronto.

2. PLANEJAMENTO DO PROJETO

Não analisamos jogos semelhantes, mas assistimos a tutoriais no YouTube dos canais Game Code Library e Two Tv Games e realizamos pesquisas para aprender a nova linguagem.
O jogo é uma plataforma 2D em que o jogador derrota monstros e ganha níveis conforme progride.
As principais ferramentas utilizadas foram:

- **Linguagem:** C# (pela semelhança com C)
- **Engine:** Unity (por ser otimizada e amplamente usada)
- **Outros programas:** Tiled Map Editor, Piskel, RPG Maker MZ e Pixilart (para sprites); VS Code (para o site); e IA (para correção de erros).

Divisão da equipe:

- Leonardo — documentação e mapa
- Arthur — parte gráfica e site
- Heitor — storyboards e site
- Rafael — código do jogo

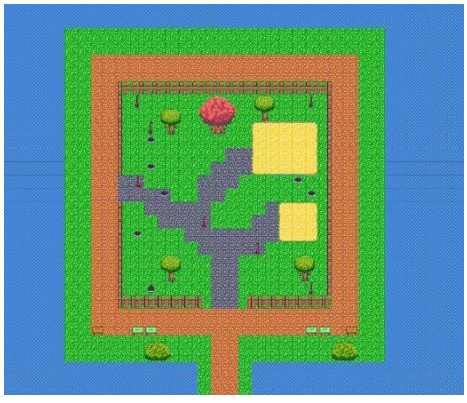
As principais decisões envolveram a divisão de tarefas conforme as habilidades de cada integrante e a escolha das ferramentas a serem usadas.

3. DESENVOLVIMENTO

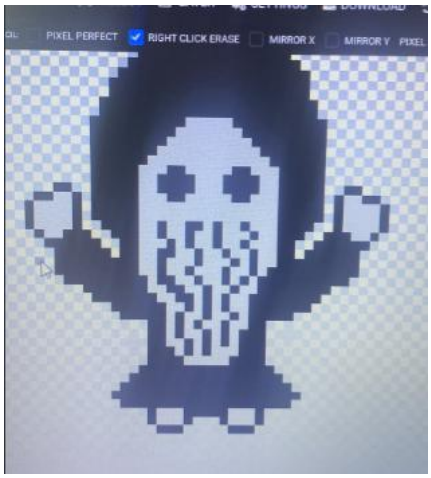
3.1 Design e Arte

O estilo visual escolhido foi pixel art**.**
Personagens:

- Felipe (protagonista)
- Cumacanga (inimigo comum)
- Mapinguari (mini boss)
- Bradador (boss)
- Curupira (NPC)

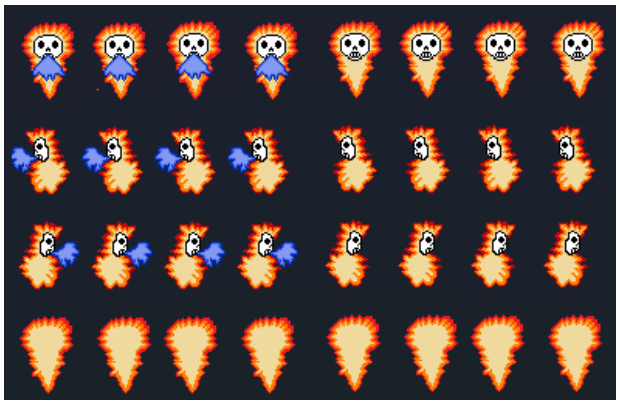


Mapa

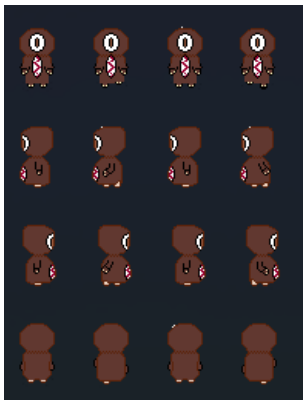


Inimigo 1 - Bradador

Felipe (protagonista)



Inimigo 2 – Cumacanga



Inimigo 3 - Mapinguari

Iminigo – Bradador

Curupira

3.2 Arquitetura e Código

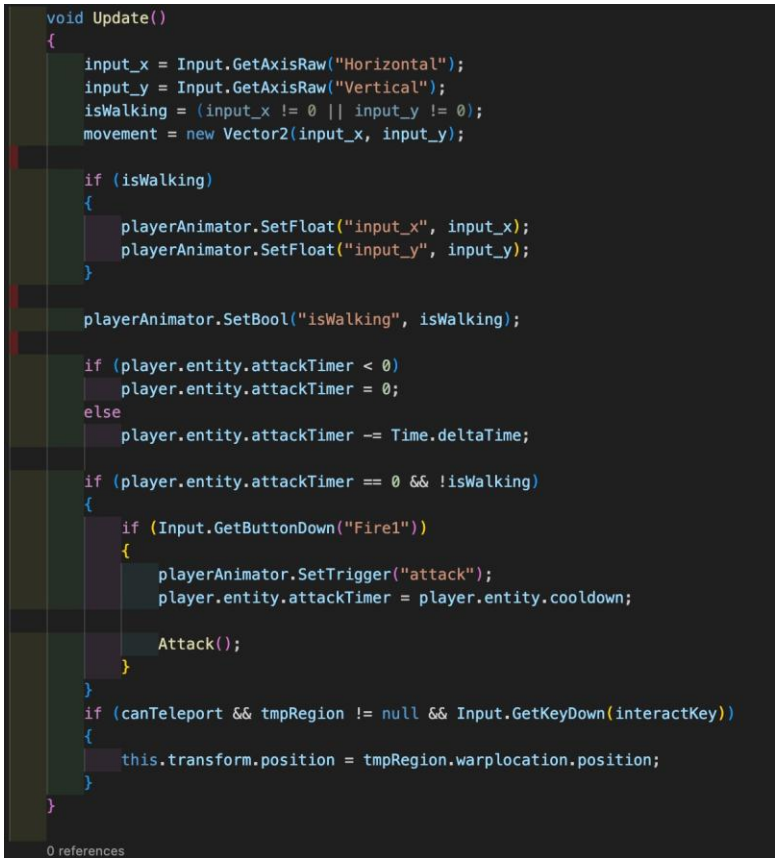
O jogo utiliza mecânicas básicas de movimentação, ataque e progressão de personagem. O objetivo principal é derrotar os inimigos e chegar ao parquinho no final do jogo. Toda a lógica foi construída em C# dentro da Unity, que gerencia a física, colisões e o comportamento dos personagens.

A lógica principal do jogo se baseia em sistemas que controlam vida, mana, estamina, experiência e level do jogador. Foram utilizadas estruturas condicionais (if/else) para verificar situações como a quantidade de vida do personagem, se ele pode regenerar status ou realizar ataques. Além disso, foram aplicados laços de repetição (loops) para manter o movimento contínuo do player e para atualizar as funções de regeneração de forma automática durante o jogo.

Um dos elementos mais importantes do código é a função de Level Up**,** responsável por evoluir o personagem conforme ele acumula experiência.

```
void Start()
{
    if (manager == null)
    {
        Debug.LogError("Você precisa anexar o game manager aqui no player");
        return;
    }

    entity.maxHealth = manager.CalculateHealth(entity);
    entity.maxMana = manager.CalculateMana(entity);
    entity.maxStamina = manager.CalculateStamina(entity);

    entity.currentHealth = entity.maxHealth;
    entity.currentMana = entity.maxMana;
    entity.currentStamina = entity.maxStamina;

    health.maxValue = entity.maxHealth;
    health.value = health.maxValue;

    mana.maxValue = entity.maxMana;
    mana.value = mana.maxValue;

    stamina.maxValue = entity.maxStamina;
    stamina.value = stamina.maxValue;

    exp.value = currentExp;
    exp.maxValue = expLeft;

    expText.text = String.Format("Exp: {0}/{1}", currentExp, expLeft);
    levelText.text = entity.level.ToString();

    // iniciar o regenhealth
    StartCoroutine(RegenHealth());
    StartCoroutine(RegenMana());
    StartCoroutine(RegenStamina());

    UpdatePoints();
    SetupUIButtons();
}
```

Essa parte é onde atribui os status calculados ao player e chama as funções de regeneração de vida, mana e estamina, logo depois chama as funções de alteração de status, com os botões adicionados para realizar essa alteração.



```
void Update()
{
    input_x = Input.GetAxisRaw("Horizontal");
    input_y = Input.GetAxisRaw("Vertical");
    isWalking = (input_x != 0 || input_y != 0);
    movement = new Vector2(input_x, input_y);

    if (isWalking)
    {
        playerAnimator.SetFloat("input_x", input_x);
        playerAnimator.SetFloat("input_y", input_y);
    }

    playerAnimator.SetBool("isWalking", isWalking);

    if (player.entity.attackTimer < 0)
        player.entity.attackTimer = 0;
    else
        player.entity.attackTimer -= Time.deltaTime;

    if (player.entity.attackTimer == 0 && !isWalking)
    {
        if (Input.GetButtonDown("Fire1"))
        {
            playerAnimator.SetTrigger("attack");
            player.entity.attackTimer = player.entity.cooldown;

            Attack();
        }
    }
    if (canTeleport && tmpRegion != null && Input.GetKeyDown(interactKey))
    {
        this.transform.position = tmpRegion.warplocation.position;
    }
}

0 references
```

Nessa parte está integrada a movimentação do personagem, mostrando também o ataque, chamando o botão esquerdo do mouse para realiza-lo e também declarando o botão, e para interagir com o teleporte.

3.3 O Site de Divulgação

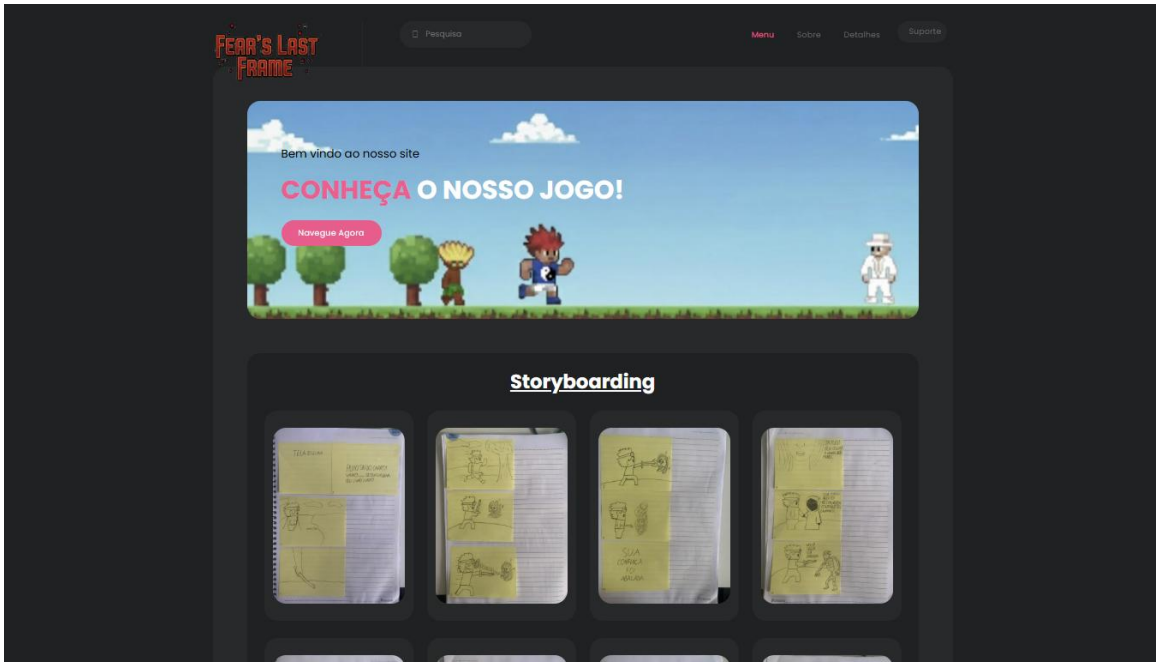
Nosso objetivo com o site de *Fear's Last Frame* é garantir que o jogador consiga acessar o jogo com facilidade e aproveitar ao máximo a experiência, de forma visual, informativa e interativa. O site foi desenvolvido com foco em apresentar o conteúdo principal do projeto, dividindo-se entre tutoriais visuais**,** gameplays e informações sobre o jogo**.**

Entre os conteúdos planejados, estão:

- **Tutorial de acesso**, com fotos e vídeos mostrando o passo a passo completo para download e instalação do jogo.
- **Gameplay e instruções**, com vídeos explicativos que mostram as ideias iniciais, os primeiros passos e as mecânicas principais do jogo.

- **Conteúdo informativo**, incluindo detalhes sobre a história (lore), o mapa e a organização do mundo do jogo.

Esses elementos foram pensados para que o visitante tenha uma visão clara do universo do jogo e entenda melhor suas mecânicas e narrativa.



Página inicial do site

A estrutura do site foi criada em **HTML5**, utilizando a tag como base para agrupar e organizar os elementos da página.
Foram usadas divisões como:

- div class="container" — define a largura central da página;
- div class="page-content" — envolve o conteúdo principal;
- div class="row" — organiza o layout em linhas, seguindo o padrão do Bootstrap.

A tag <img> foi utilizada para exibir imagens como o logo**,** storyboards**,** mapas e a classificação indicativa**.**
Outras tags semânticas usadas incluem <h1> a <h6> (títulos), <a> (links) e <p> (parágrafos).

3.3.1 Navegação (<nav> e <ul>)

A navegação foi construída com a tag <nav>, responsável pela área principal do menu (classe .main-nav).
Dentro dela, utilizamos listas não ordenadas (<ul>) com itens (<li>) para criar os links principais do site, como:

- Menu
- Sobre
- Detalhes
- Suporte

Essa estrutura torna a navegação simples, organizada e intuitiva para o usuário.

3.3.2 Frameworks e Estilo (Bootstrap e CSS)

O site utiliza o framework Bootstrap para definir sua estrutura e garantir a responsividade em diferentes tamanhos de tela (computador, tablet e celular).
Foram usadas classes como container, row, col-lg-12, col-lg-3 e col-sm-6 para montar o layout.

Outros componentes importantes incluem:

- header-sticky — para deixar o cabeçalho fixo;
- btn — usada em botões como o *“Navegue Agora”*.

Além disso, foram aplicados arquivos CSS externos, como fontawesome.css, templatemo-cyborg-gaming.css e owl.css, que servem para:

- Definir o estilo visual (cores, fontes e fundos);
- Adicionar ícones via Font Awesome**;**
- Controlar animações e carrosséis de imagens.

3.3.3 Interatividade (JavaScript/jQuery)

Os arquivos JavaScript foram carregados no final do <body> para melhorar o desempenho do site.
Entre eles estão jquery.min.js, bootstrap.min.js e custom.js.

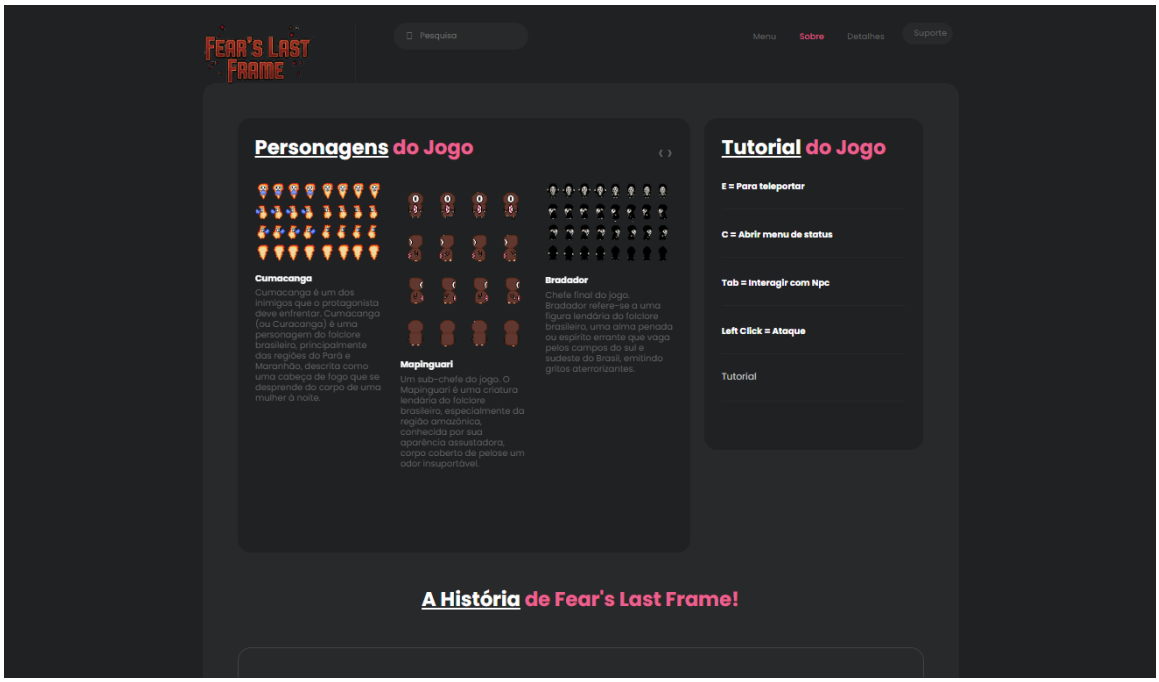
Esses scripts são responsáveis pelas funções dinâmicas do site, como:

- O pré-carregador (js-preloader);
- O menu sanfona (menu-trigger);

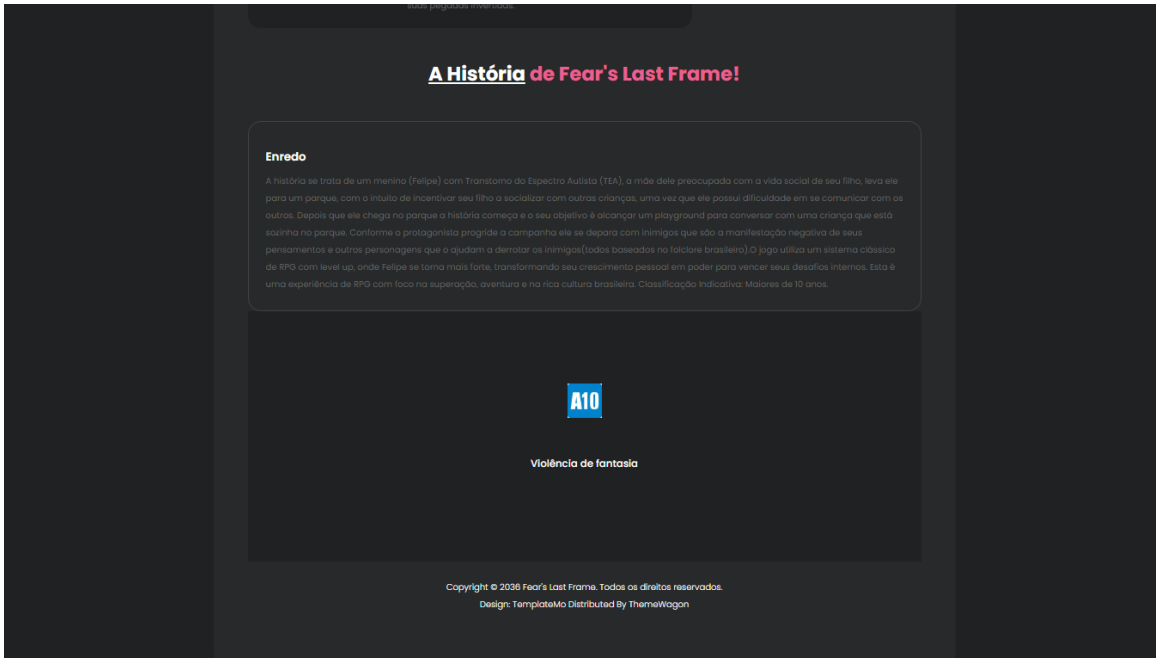
- Os carrosséis de imagens e animações interativas.

Essas funcionalidades deixam a navegação mais fluida e dão uma aparência mais profissional ao site de *Fear's Last Frame*.

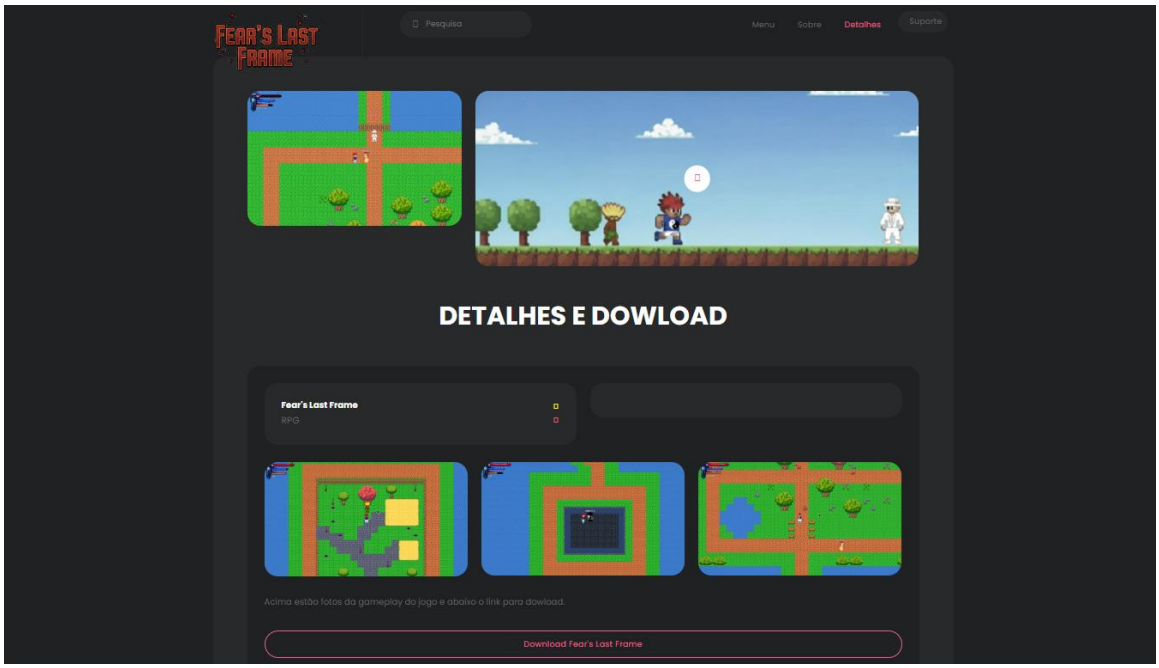
Site do jogo – imagem 1



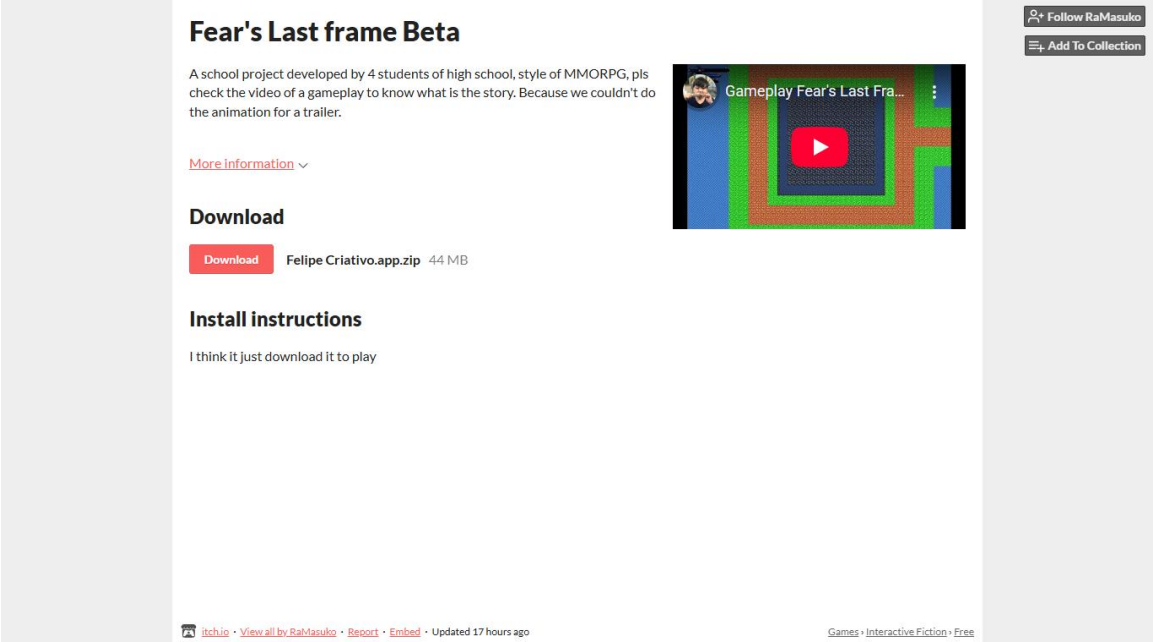
Site do jogo – imagem 2



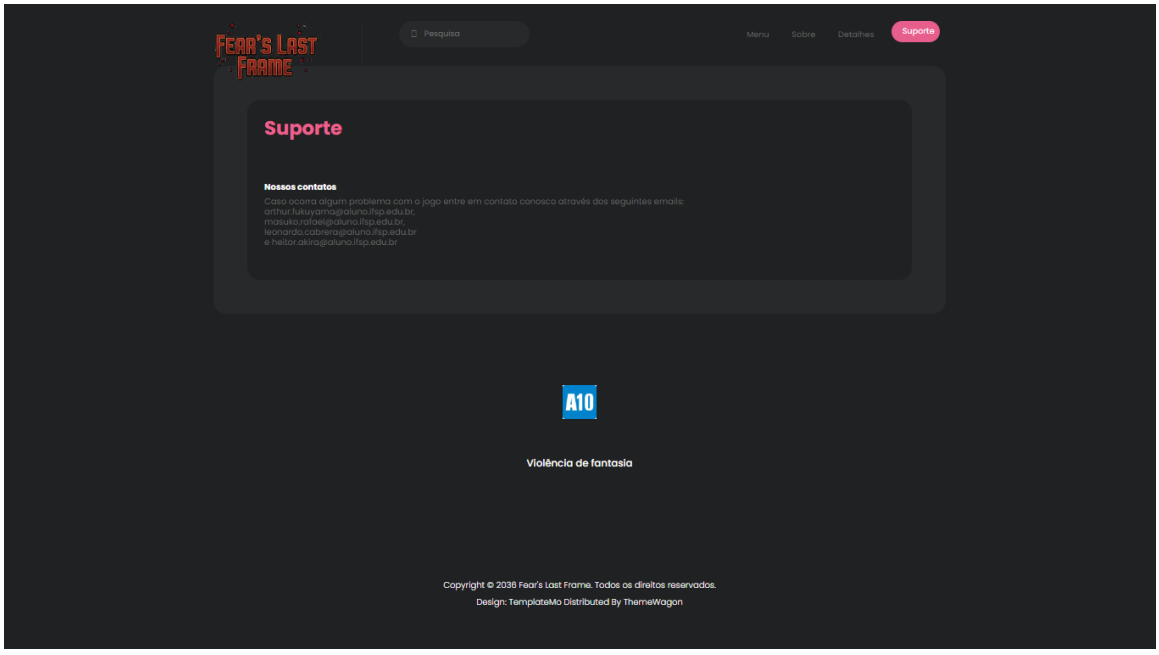
Site do jogo – imagem 4



Site do jogo – tela para download



Tela de download com gameplay



Site do jogo – Tela para contatos

Link do Site: https://arthurfuku.github.io/Site-oficial/

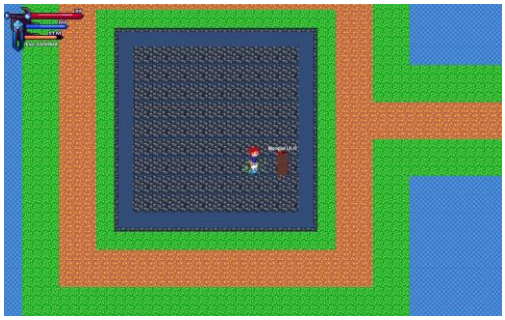
4. PROTÓTIPO E TESTES

Durante o desenvolvimento do *Fear's Last Frame*, realizamos os testes de forma dividida em etapas. A cada nova parte do código criada — como o sistema de movimentação, ataque ou colisão — fazíamos testes imediatos para ver se aquela parte estava funcionando corretamente antes de continuar o restante do projeto.

Esse método nos ajudou a identificar e corrigir erros mais rapidamente, evitando que eles se acumulassem e se tornassem mais difíceis de resolver depois. Quando algo não funcionava como esperado, analisávamos o código com calma e utilizávamos ferramentas de IA para entender o erro e encontrar uma forma de corrigir.

Durante os testes completos do jogo, ainda surgiram bugs mais complexos, principalmente em combates. Em alguns casos, os inimigos atravessavam paredes, andavam sobre a água ou interagiam de forma errada com o cenário. Esses erros foram sendo ajustados aos poucos, até o jogo alcançar um funcionamento estável.

A IA foi uma grande aliada nesse processo, ajudando a revisar trechos do código e a identificar falhas que passavam despercebidas. No final, os testes garantiram que o jogo rodasse de forma fluida e com as principais mecânicas funcionando como planejado.



Link da gameplay: https://youtu.be/7-utVs1nyA0

Link do tutorial: https://youtu.be/ZQ09vB8Kp8I

5. CONCLUSÃO

Fazer o jogo foi uma experiência muito boa e cheia de aprendizados. Durante o processo, conseguimos melhorar nossa organização e o trabalho em equipe**,** já que cada um ficou responsável por uma parte diferente e todas precisavam se encaixar para o jogo funcionar.

Mesmo que o resultado final tenha ficado diferente da ideia inicial, que seria um jogo sobre Pokémon, achamos que essa nova versão ficou bem mais criativa e original. Criar o personagem Felipe e pensar em toda a história por trás dele foi algo que deixou o projeto mais interessante.

As maiores dificuldades foram aprender a usar as ferramentas certas para fazer o jogo, principalmente a Unity, que no começo parecia bem complicada. Também tivemos que lidar com vários erros e bugs, e às vezes usamos IA para nos ajudar a entender o que estava dando errado e como corrigir.

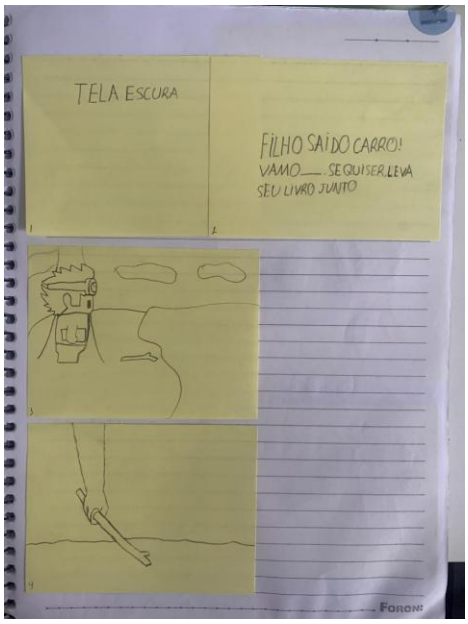
A parte gráfica foi uma das mais legais de fazer, porque deu pra usar bastante criatividade nos desenhos e nas ideias. Já a programação foi mais trabalhosa, mas ver o jogo funcionando depois de tanto esforço foi muito gratificante.

Se tivéssemos mais tempo, gostaríamos de colocar mais personagens, novos inimigos e um mapa maior, além de criar mais mecânicas para deixar o jogo mais divertido. No fim, ficamos bem satisfeitos com o que conseguimos fazer e com tudo o que aprendemos nesse projeto.

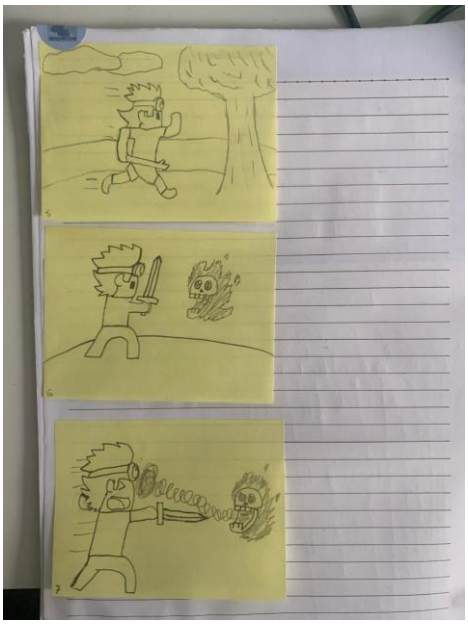
6. REFERÊNCIAS

- **DOCUMENTAÇÃO oficial da Unity.**
- **FERRAMENTAS utilizadas:** Unity, Tiled Map Editor, Piskel, RPG Maker MZ, Pixilart, Visual Studio Code e Inteligência Artificial (para identificar e corrigir erros).
- **GAME CODE LIBRARY.** *Tutoriais de desenvolvimento de jogos na Unity*. YouTube, 2024.
  Disponível em: https://www.youtube.com/watch?v=4-bu0YhbVE0&list=PLm43vXIKl0F-1uxmGW84NmlJh6dob296f.
  Acesso em: 7 nov. 2025.
- **SITES de pesquisa** - https://www.w3schools.com/cs/index.php
- **TUTORIAIS do YouTube** sobre Unity e C#.
- **TWO TV GAMES.** *Dicas e tutoriais de criação de jogos 2D na Unity*. YouTube, 2024.
  Disponível em: https://www.youtube.com.
  Acesso em: 7 nov. 2025.
- **UNITY TECHNOLOGIES.** *Criação e desenvolvimento de jogos com Unity*. Unity, 2025.
  Disponível em: https://unity.com/pt/games.
  Acesso em: 7 nov. 2025.

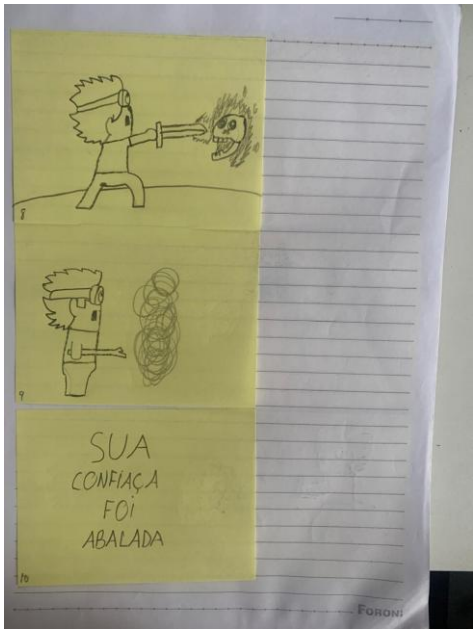
APÊNDICE A – STORYBOARDS DO JOGO



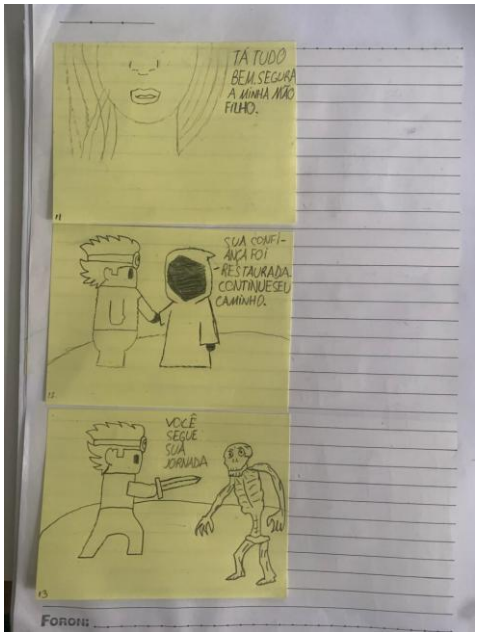
Cena 1



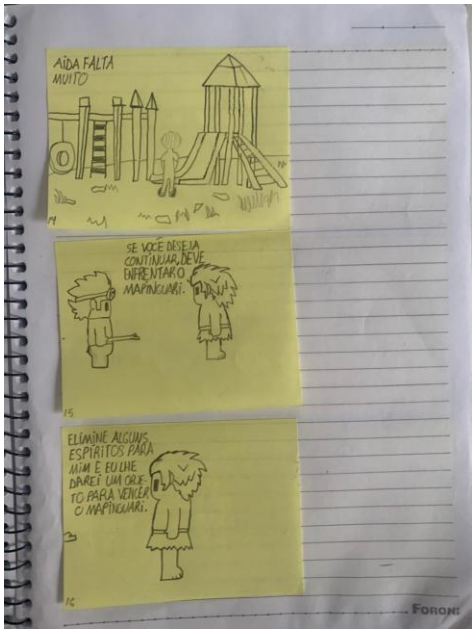
Cena 2



Cena 3

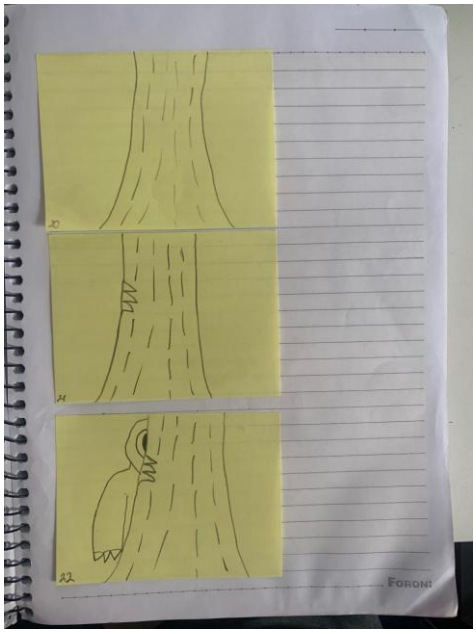


Cena 4

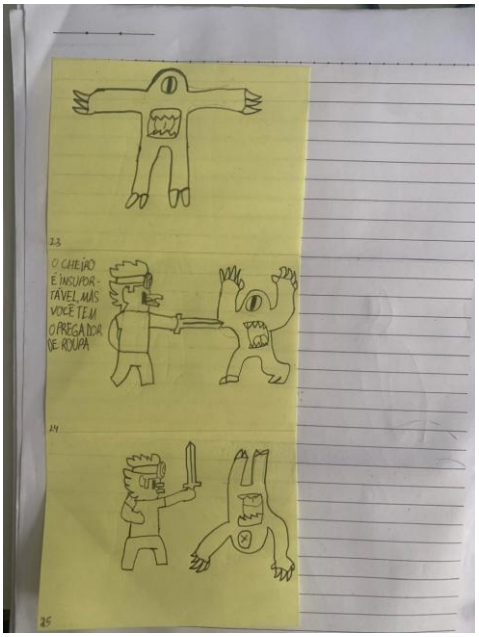


Cena 5

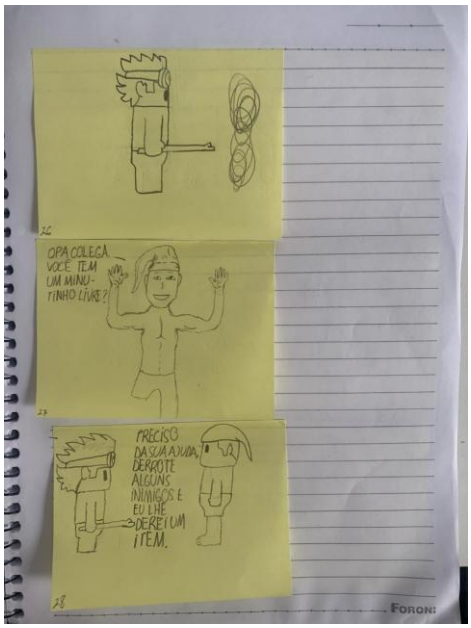
Cena 6



Cena 7

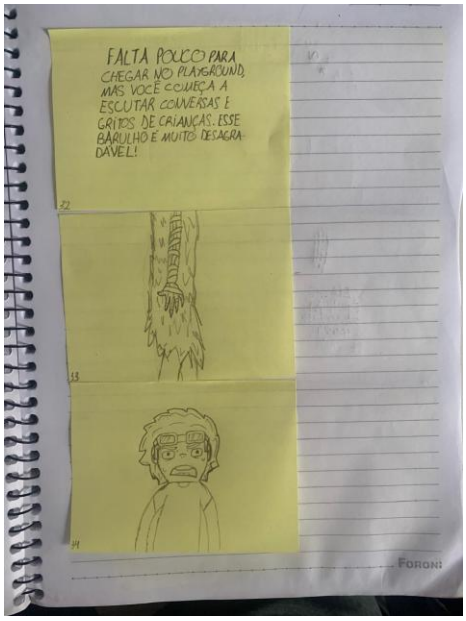


Cena 8

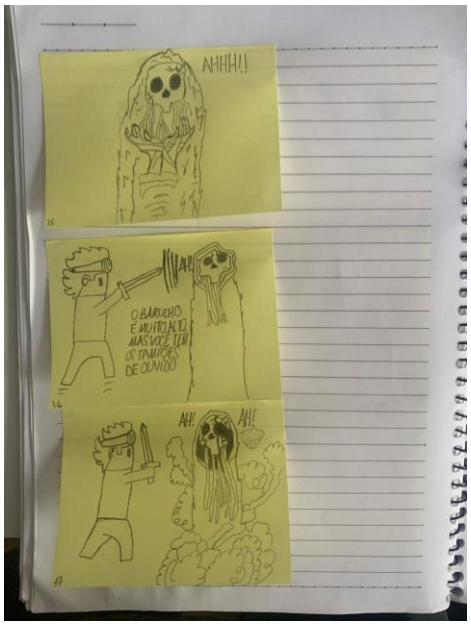


Cena 9

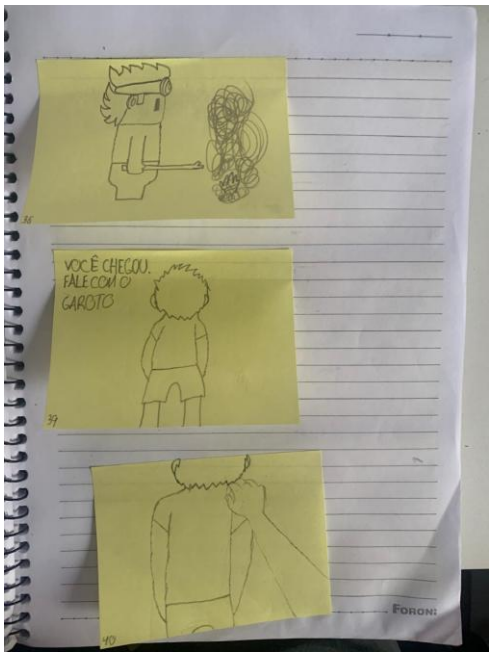
Cena 10



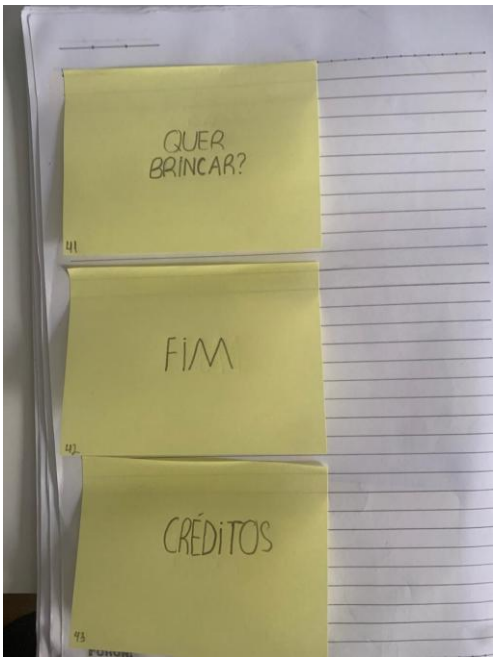
Cena 11



Cena 12



Cena 13



Cena 14