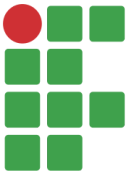


INSTITUTO FEDERAL DE SÃO PAULO

Angélica Francisconi de Souza

Arthur de Brito da Silva

Beatriz Aparecida Primos

Breno Oliveira Trigo

Elisabella Sabbah Maciel

Gabriel Rodrigues Valones

DOCUMENTAÇÃO DO JOGO DE CONCLUSÃO ANUAL

São Paulo

2025

Angélica Francisconi de Souza
Arthur de Brito da Silva
Beatriz Aparecida Primos
Breno Oliveira Trigo
Elisabella Sabbah Maciel
Gabriel Rodrigues Valones

DOCUMENTAÇÃO DO JOGO DE CONCLUSÃO ANUAL

Documentação de Projeto Interdisciplinar apresentado como requisito parcial para avaliação nas disciplinas de Lógica de Programação, Desenvolvimento WEB, Informática e Ferramentas para Desenvolvimento e Arquitetura de Computadores e Redes.
Banca: Claudete Alves, Cláudia Miyuki, Ana Mamede.

São Paulo

2025

SUMÁRIO DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Ao entrar no Curso Técnico de Desenvolvimento de Sistemas do Instituto Federal Câmpus São Paulo, foi proposto aos estudantes como trabalho de conclusão anual que um jogo que atendesse em nuance as quatro disciplinas do curso, estas são, Lógica de Programação, Desenvolvimento WEB, Informática e Ferramentas para Desenvolvimento e Arquitetura de Computadores e Redes, fosse feito.

Desta maneira, foi necessário que fossem concebidos: um jogo, um site e uma documentação do projeto. O objetivo deste documento em questão é documentar todo o processo da criação do jogo, desde sua idealização e concepção até sua final concepção, permeando o tipo de programação, arte adotada e afins, adotando sempre a linguagem técnica e os protótipos do jogo.

O jogo foi feito para ser apresentado defronte a uma banca de jurados e uma escola, onde se apresentaria o jogo jogável, a lógica por trás e seu site. Os alunos tiveram o ano todo, contudo o assunto se tornou mais persistente nos meses finais do ano devido às questões escolares.

2 PLANEJAMENTO

2.1 Idealização

Enquanto ponderava sobre o que poderia ser a temática do jogo, um dos integrantes sugeriu que o jogo poderia ser um **Jogo de Cartas** que visasse *features** envolvendo as cartas que seriam postas na mesa em seu desenrolar. O grupo de início aceitou como uma ideia a se pensar, contudo após um breve tempo, passou a crer que devido ao tempo e complexidade que poderiam ser demandadas ao desenvolvimento, passou-se a crer que a ideia poderia ser contraintuitiva e então preferiu-se optar por uma opção mais “sólida”.
Após conversas enquanto o assunto era dissertado em sala, chegou-se em mente um jogo já bem conhecido pela maior parte dos integrantes do grupo: o jogo de plataforma **Level Devil**. O jogo em questão faz parte do gênero “troll game”, jogos desse gênero visam atrapalhar e dificultar a gameplay do jogador, enquanto o mesmo tenta concluir um objetivo proposto, como chegar ao final de uma fase. É um estilo de jogo que ganhou força na última década

com youtubers que gravavam a si mesmos tentando concluir tais jogos mediante as dificuldades impostas. Os jogos do gênero troll game mais famosos são Shobon no Action, Trollface Quest e o próprio Level Devil.

Depois de pensar a respeito, o grupo concluiu que esse estilo de jogo se encaixa naquilo que desejavam manifestar em seu projeto: algo que fizesse o jogador ficar imerso no universo do jogo proposto.

2.2 Escopo do Jogo

Após organizar uma reunião na biblioteca da escola e algumas conversas subsequentes com a finalidade de decidir o escopo do jogo, o grupo chegou às seguintes conclusões:

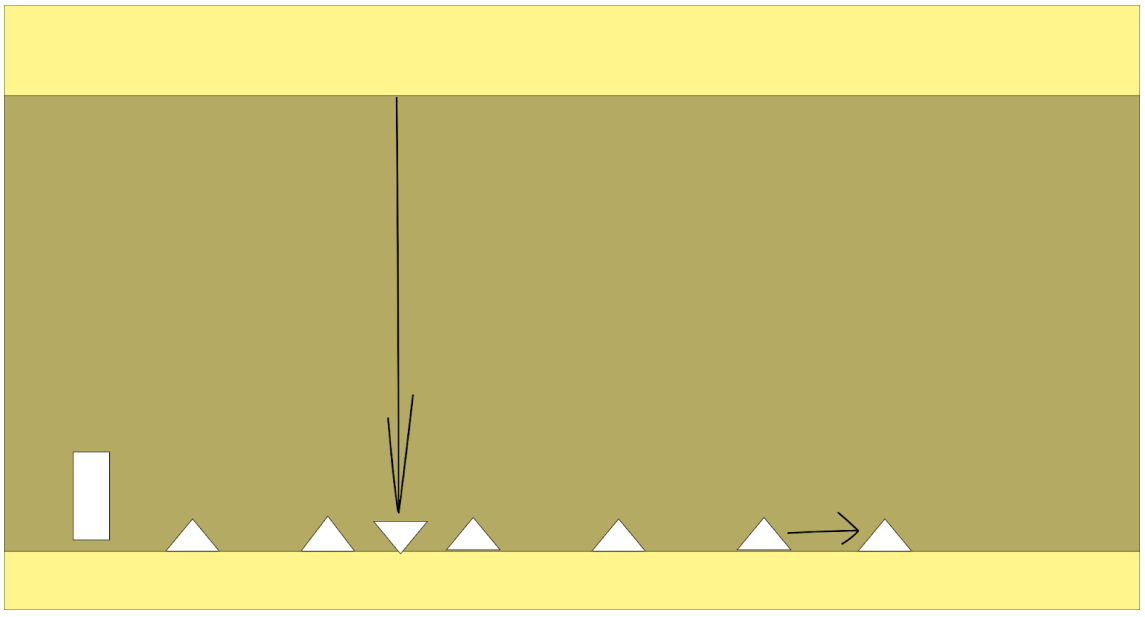
O jogo seria do tipo troll, 2D com inicialmente 14 fases, posteriormente, devido ao tempo o número de fases teve que ser reduzido para 10. o grupo ficou indeciso quanto à engine a ser usada, mas por fim acabou adotando a **Unity** como um motor gráfico. O jogo teria uma temática voltada para a escola, onde, o personagem principal é um formando que, no dia de sua formatura, acabou por ir à biblioteca e ser transportado para dentro de um livro, então tendo que resgatar professores do Instituto Federal que também haviam sido levados para dentro do mesmo.

No jogo, o jogador deveria encostar no professor, para que então passasse para outro nível. Sendo assim, haveria apenas duas formas de movimentação do jogador: o andar e o pular. Durante esse trajeto, o jogador se depararia com plataformas que caem, espinhos que somem e aparecem, pilastras esmagadoras,etc., sempre buscando uma sofisticação diferente em cada nível.

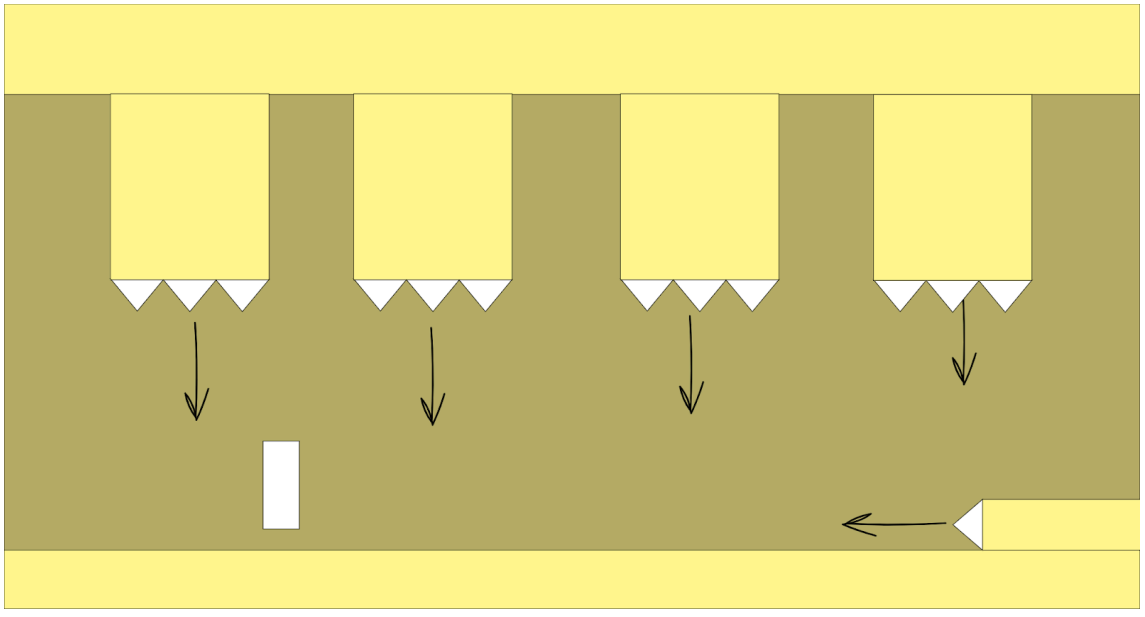
Haveriam dois mundos: um que se passaria numa pirâmide e outro que se passaria num navio. ambos os temas foram sugeridos por integrantes do grupo durante reuniões virtuais e melhor esboçados por meio de breves storyboards que demonstravam o conceito de cada fase.

2.3 Storyboard

Os storyboards usados pelos designers para esboçar as ideias e conceitos principais de cada uma das fases são:



StoryBoard Exemplo 1. Elaborado pelo autor.



StoryBoard Exemplo 2. Elaborado pelo autor.

2.4 Ferramentas e Tecnologia

Da mesma maneira que qualquer jogo, foi necessário que a equipe se dividisse em áreas diferentes e consequentemente utilizasse softwares diferentes para determinadas finalidades do projeto.

A Engine usada para realização do projeto foi a **Unity Engine**, sendo ela escolhida devido a experiência prévia do programador e a facilidade na parte da exportação para plataformas e interface do jogo, coisas que seriam de suma importância graças ao tema do projeto. Além disso, a Unity é uma das mais completas engines do mercado, tendo há anos uma comunidade rebuscada oferece suporte para quaisquer dúvidas que os usuários possuam, além de sua completude técnica que a põe à frente de grande parte das demais engines.

Para a arte, um dos artistas optou por utilizar a ferramenta **Pixel Studio**, que possui um certo grau de afinidade com os projetos Unity em relação a sua exportação e formatos de imagem, já o outro (responsável pelos cenários) preferiu utilizar o software **Pixelart**, por seu prévio domínio.

Para a música, a musicista resolveu se utilizar da ferramenta **Ableton Live 12 Suite**, devido a já reputação da plataforma no mercado. Contudo, o musicista teve de se adaptar à música digital como um todo, necessitando de exercer domínio da plataforma com o desenrolar do jogo.

O grupo utilizou softwares de Inteligência Artificial para uma melhor compreensão sobre o ambiente, tirando dúvidas de sintaxe e sobre como manifestar as ideias já claras na plataforma.

2.5 Cronogramas e Atas

A principal reunião física ocorreu no dia 21 de Agosto, onde o grupo definiu os cargos de cada membro e consequentemente como a organização funcionaria. As conclusões da reunião foram:

Nome	**Cargo**
Angélica Francisconi	Level Designer.
Breno Trigo	Animador.
Elisabella Sabbah	Artista.
Gabriel Valones	Level Designer e Desenvolvedor web.
Beatriz Primos	Musicista.
Arthur Brito	Programador e desenvolvedor web.

Subsequentemente houveram reuniões digitais que serviram para tirar o molde dos storyboards antes vistos e estabelecer limites e temas a serem feitos. Como por exemplo, no dia 02 de Outubro, onde foi decidido que o tema das fases seriam relacionados a ambientes de pirâmide e de navio para complementar a narrativa relacionada ao cunho literário do jogo.

3 DESENVOLVIMENTO

3.1 Design e Arte

A presente seção tem como objetivo detalhar os aspectos visuais e estéticos do projeto, abordando desde a concepção das características do personagem principal até a construção dos cenários e elementos gráficos que compõem o universo do jogo. A escolha pelo estilo pixel art foi motivada pela sua capacidade de unir simplicidade técnica à expressividade visual, permitindo a criação de ambientes ricos e personagens marcantes. O design artístico

busca refletir a narrativa fantástica do jogo, que mescla referências egípcias e marítimas, ao mesmo tempo em que representa simbolicamente a jornada acadêmica dos estudantes do IFSP. Por meio de uma abordagem coerente entre arte e narrativa, o projeto visa proporcionar uma experiência imersiva, lúdica e visualmente significativa ao jogador.

3.1.1 Visão Geral Estética

O projeto apresenta uma estética baseada em *pixel art*, com resolução média e atenção minuciosa aos detalhes visuais. A escolha por esse estilo busca equilibrar simplicidade técnica com expressividade artística, permitindo a construção de cenários ricos e personagens marcantes. A ambientação combina elementos de civilizações antigas, como o Egito Antigo, com o universo da pirataria clássica, criando uma fusão visual única que reforça o caráter fantástico da narrativa.

A proposta estética visa transmitir sensações de mistério, desafio e descoberta, alinhando-se à jornada do protagonista dentro de um mundo mágico e imprevisível.

3.1.2 Personagem Principal: Biribo

Como personagem principal, o grupo em questão escolhera seu mascote, que teve seu design concebido no começo do ano: Biribo. Inspirado sobretudo no clássico Bomberman, Biribo em suas origens era uma bomba de terno e chapéu. Contudo, para melhor adaptação à ambientação do jogo, foram postas nele vestes de formando.

A caracterização do protagonista em relação ao jogo se mostra presente sobretudo em sua forma de morrer, gerando uma mini explosão, algo que remete à essência “bombástica” do personagem.

Tal escolha diz respeito a uma grande busca do grupo por sua identidade em meio ao projeto, evitando ao máximo temas e personagens muito universalizáveis.

3.1.3 Ambientação e Cenários

O jogo é dividido em dois núcleos visuais principais, cada um com identidade própria:

a) Fases Egípcias

- **Texturas predominantes:** Tijolos amarelos, areia, elementos cerimoniais como sarcófagos e tochas.
- **Paleta de cores:** Tons quentes – dourado, vermelho queimado, marrom.
- **Atmosfera:** Mística, com foco em enigmas e armadilhas

b) Fases Náuticas

- **Texturas predominantes:** Madeira envelhecida, velas rasgadas, céu aberto e baús de tesouro
- **Paleta de cores:** Tons frios – azul profundo, verde musgo, branco e dourado
- **Atmosfera:** Aventura, liberdade, tensão marítima

3.2 Arquitetura e Código

No que diz respeito a parte lógica do projeto, o principal desafio era programar features que fossem inovadoras e que pudessem ao mesmo tempo ser recicladas por outros códigos, garantindo eficiência e inovação durante o fluxo do jogo. Como mencionado anteriormente, o jogo foi programado em C#, linguagem da Microsoft que é a padrão da Unity e que trás consigo uma sintaxe levemente difícil, contudo que possui um amplo repertório. No desenvolvimento do projeto foi implementado o novo sistema de Inputs da Unity, que garante que uma determinada ação possa ocorrer por n teclas diferentes e que seja possível compatibilizar tanto o teclado ao projeto quanto o gamepad (controle) de maneira íntegra, tornando assim possível a execução do jogo no celular desde que o jogador em questão possua um gamepad que funcione no celular.

A Lógica do Jogo consiste em sua maioria na evocação de **eventos**, que nada mais são que “alarmes” produzidos por um código que evoca uma ação de outro código. No contexto do jogo, os eventos em sua maioria eram sensores que, ao colidirem com o jogador, disparavam as “ações de trollagem” dos níveis, como o levantar de plataformas, nascimento de espinhos, etc.

Como o jogo atua com diversas movimentações, seja do jogador seja do cenário, o principal componente contido na Unity foi o **Rigidbody2D**, componente esse responsável pela Física na Unity. Assim, quando algo no jogo se move de maneira dinâmica, ocorre por conta deste componente, que simula a velocidade do corpo em relação a sua posição na tela por meio de suas funções como MovePosition, Velocity, BodyType, etc.

Um outro princípio do desenvolvimento de um jogo como este é também a necessidade muito de colisores, pois são eles que garantem que o jogador faça determinada ação ao entrar em contato com outro objeto, como é o caso de morrer ou passar de fase. A Unity por si só já fornece uma robusta ferramenta quanto à colisores, sendo muito mais fácil de manuseá-los do que outras engines sem tanto suporte "externo" à programação.

3.3 Site de Divulgação

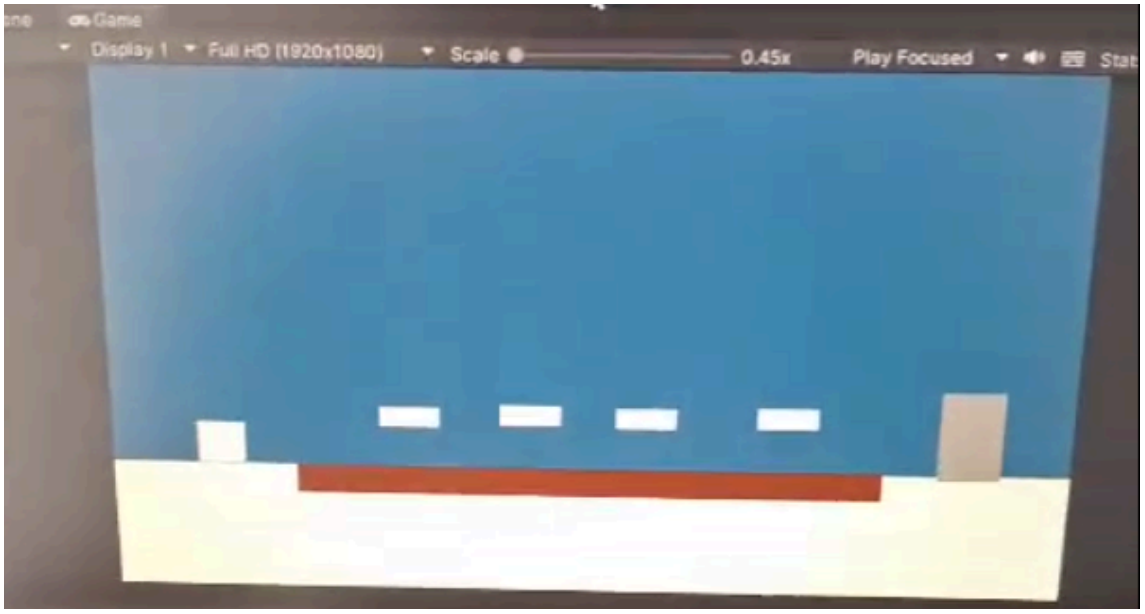
Além do jogo, uma das tarefas propostas pela banca foi a concepção de um **website** que tivesse por princípio divulgar e disponibilizar o jogo para sua execução. O site desenvolvido pelo grupo baseia-se em sites de jogos independentes como é o caso dos sites de jogos como Terraria e Undertale.

O design CSS e HTML baseiam-se em criar uma ambientação do jogo, fazendo com que o usuário adentre o site e acabe por sentir curiosidade a respeito do que o jogo se trata e como jogá-lo. Assim, a disposição de cores ajuda a fixar o plano de fundo do site (que por si só trata-se de um fundo do jogo).

Além disso, o site é **totalmente responsivo** com desktop, celular e tablet. Para tal, o grupo usou do recurso HTML denominado Media Queries, que ajuda a produzir responsividade em diferentes layouts, garantindo um acesso a todos aqueles que desejam utilizar.

4 PROTÓTIPO E TESTES

4.1 Desenvolvimento do Jogo



Primeira Versão de Testes. Elaborado pelo Autor.

A primeira versão de testes do jogo (finalizada em 30 de setembro) era extremamente simples, a principal mecânica residia na movimentação do jogador e das plataformas que o cercavam. Havia apenas um nível e poucas plataformas.

A principal finalidade desta versão residia em testar quais seriam os limites dos atributos do jogador (velocidade, força de pulo etc.) e quais as melhores adaptações de colisão. No protótipo em questão, o jogador poderia pular sobre as plataformas, mas, em uma delas, a plataforma despencava, fazendo com que o jogador caísse na Lava.

Dada a primeira versão, o grupo pôs-se a trabalhar todo dia no projeto. De modo que em uma semana havia sido desenvolvido:

- A troca de níveis de maneira dinâmica
- Um chão falso;
- Contador de mortes;
- O sprite do protagonista (que ainda não era o Biribo mas um estudante com um foco na universalidade já que o tema ainda não tinha sido definido por completo);

- um spawner de espinhos e um falso spawner de espinhos;
- o sprite de um professor que serviria de portal (que teve de ser trocado devido a autorização não-confirmada).

Ao término da semana, com os temas do jogo já firmados, um dos designers acabou por fazer os storyboards já vistos nesta documentação. Até então, os storyboards simbolizavam as sete primeiras fases do primeiro mundo do jogo, que, devido ao tempo, teve de ser reduzida, tornando-as posteriormente as cinco fases do primeiro mundo e duas últimas fases do segundo mundo.

Após a compreensão e explicação dos storyboards por parte do designer, o programador e os artistas partiram a melhor concepção da leitura que fosse possível dentro do prazo. Dia 08 de Outubro o animador por fim realizou as animações do mais novo personagem: Biribo, Havendo também o programador realizando a adaptação funcional das novas animações no mesmo dia.

Com o avançar do desenvolvimento das fases, foi concebido um **menu principal**, que contemplava a tela de créditos, tutoriais, sair e jogar. Por início, somente o botão de sair possuía uma aplicação 100% funcional, não sendo preciso alterá-lo em nenhum momento do jogo. Por outro lado, os outros botões realizavam suas funções de maneira correta contudo não possuíam artes ou gerenciamentos de código suficiente para sustentá-los, um exemplo disso é a tela de seleção de menu, que por mais que fosse planejada pelo programador desde o início, só teve sua criação realizada depois que o jogo já estava em uma versão muito avançada devido a complexidade do script em questão. As artes eram em suma limitadas, apesar de haverem todos os botões, as telas de créditos e tutoriais eram apenas imagens padrões de interface da Unity, havendo sido colocadas após o menu de seleção de níveis.

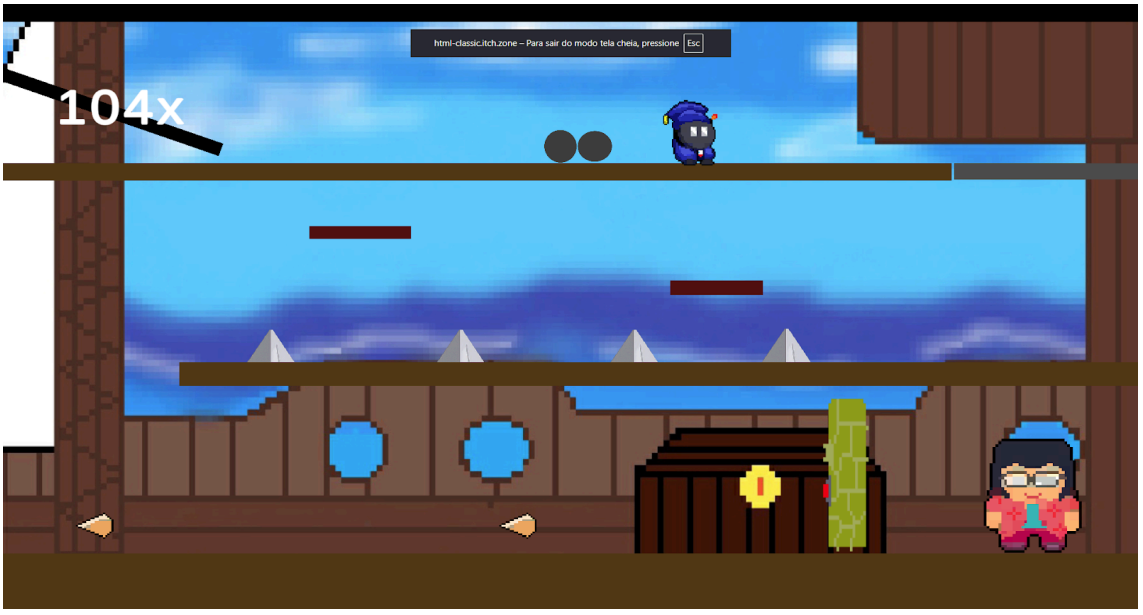
Assim como a maior parte das aplicações lógicas, grande parte do tempo investido na Lógica de Programação deste jogo foi dedicado a resolução de bugs e erros de sintaxe provenientes dos códigos. Isto possuiu uma carga especial graças ao tema escolhido, pois, como o jogo foca em enganar o jogador de uma grande quantidade de maneiras, foram desenvolvidas quase que uma ou duas features especiais novas para cada fase do jogo, o que totalizam de 10 a 20 features desenvolvidas durante este 1 mês de desenvolvimento, o que ajuda a explicar os mais de 20 scripts presentes no projeto.

Conforme o aumento das demandas e suas complexidades, no dia 21 de Outubro o programador adicionou o novo sistema de inputs da Unity, o que ajudara na melhor disposição das maneiras de jogar e compreensão do motor gráfico e dos scripts, que até então se comportavam segundo o limitado antigo sistema de inputs da Unity.

No dia 22 de Outubro, o design da última fase do jogo havia sido entregue, finalizando desta forma o escopo do que deveria ser feito na parte gráfica e lógica do projeto. Durante esta mesma semana a portabilidade do projeto à mouse foi propositalmente removida e uma leve cutscene (cena) foi adicionada ao clicar do jogador no botão “Jogar”, além do menu de pause (o que simbolizou uma certa liberdade da equipe a poder mexer em coisas não tão urgentes no projeto).

O processo de desenvolvimento do jogo está resumido no seguinte vídeo:
https://youtube.com/shorts/RVdxB9gajR4?si=cWgHlozmgOwlrBPK

4.2 Fase de Testes



Versão de Testes. Elaborado pelo autor.

Terminada a “essência” da última fase no projeto, o jogo entrou no período de testes, onde foi disponibilizado inicialmente por meio de um executável advindo do próprio sistema de Build na Unity. Um dos integrantes do grupo logo o instalou em sua máquina e pôs-se a jogar.

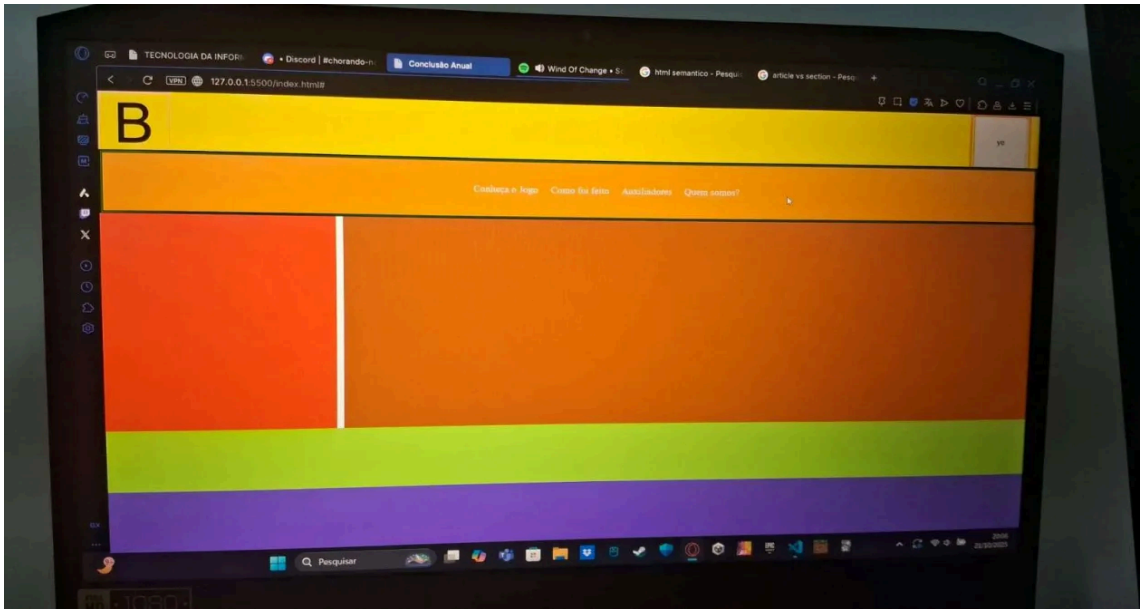
Os principais problemas reclamados pelo integrante foram a diminuição da última plataforma da segunda fase do primeiro mundo, a qual o mesmo julgava estranho com colisores quadrados e a diminuição do personagem, que acabava por ultrapassar vãos que deveriam ser propensos à sua queda.

Corrigido estes bugs, o jogo foi ganhando cada vez mais forma, de modo que se decidira que o jogo combinaria muito melhor com um estilo de jogos de navegador (HTML5), do que com jogos propriamente executáveis, fazendo com que o programador portasse e posteriormente adaptasse o jogo para o estilo em questão, publicando-o na plataforma **Itch.IO**, plataforma onde o programador já possuía uma leve experiência.

Assim, a nova versão do jogo, agora de navegador, foi lançada.

Após o teste de um usuário exterior ao grupo, houve uma reclamação referente a forma de recarregar a cena após a morte do jogador que, sendo imediata, resultava em diversas mortes acidentais. Feito isso, colocou-se um delay de cerca de 0.1 segundos na morte do jogador, o que por si só garantia um risco de mortes acidentais e depois foi melhor complementado com a animação de morte do jogador.

4.3 Desenvolvimento do WebSite

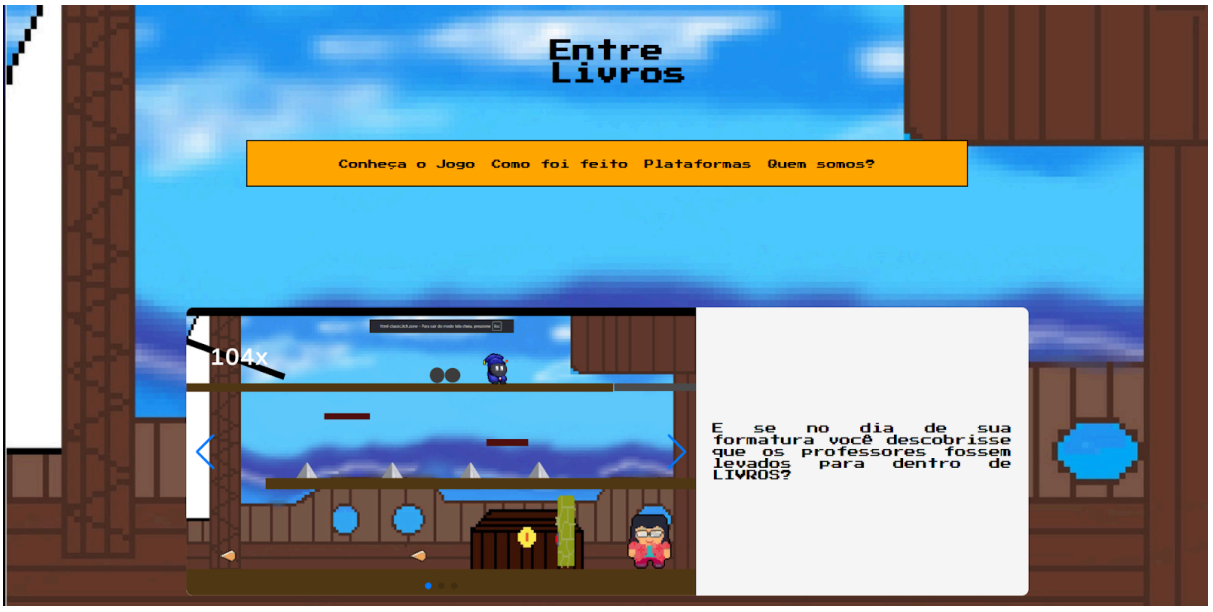


Primeira Versão do WebSite. Elaborado pelo autor.

No dia 21 de Outubro foi iniciado o website onde se divulgaria o jogo, até então o projeto não havia forma, baseava-se somente na disposição dos elementos por meio do CSS, sendo eles: o header, nav, aside, article, section e footer. Com o passar da semana o site foi ganhando melhores formas e identidade.

Durante a semana, o site foi ganhando mais forma e, com animações e imagens do próprio jogo, tornou-se um grande publicitário do projeto. Logo, aplicou-se os conceitos básicos de JavaScript e Media Queries para que se ganhasse interatividade e responsividade no site, chegando a uma versão agradável e funcional que porta o link do jogo para navegador.

O WebSite está disponível na pasta “Site” do repositório GitHub.



Última Versão do WebSite. Elaborado pelo autor.

Todo o processo de desenvolvimento foi amplamente catalogado no **GitHub** por cada membro, havendo primariamente os commits no que diziam respeito a animação e programação, e só posteriormente uma adição a outros tópicos como a música. Por tal, o grupo acabou não utilizando de muitas branches considerando o não-risco de sobreposição de arquivos que poderia ocorrer, considerando que ninguém de fato mexeu nos mesmos arquivos até o próximo fim iminente do projeto.

Link do Repositório: https://github.com/v3xxbr/DocumentosDoJogo
Link do Jogo: https://britao.itch.io/entrelivros

5 CONCLUSÃO FINAL

O Trabalho de Conclusão Anual proporcionou ao grupo uma experiência nunca antes vista até então no meio acadêmico, fazendo com que o mesmo possuísse uma completa experiência no mercado de trabalho.

Realizamos um escopo até então acessível ao tempo que tivemos, contudo reduzimos em prol da qualidade de cada nível, sendo isto algo que satisfez a todos do grupo, ainda que não seguisse a ideia inicial. Depois de pronto, o grupo observa que o jogo possuiu sim nuances que não estavam previstas no protótipo original. Apesar disto, o jogo saiu-se da maneira esperada, com a temática e jogabilidade prevista, proporcionando aos jogadores externos que o testaram a sensação determinada de cada fase.

As principais dificuldades em relação ao projeto foram sem dúvidas as plataformas envolvidas. Apesar da noção contida pelo grupo a respeito das principais plataformas usadas, foi necessário um amparo e repertório muito grande durante a finalização do trabalho, isso pois, o tema escolhido e a forma com que o projeto teve de ser manuseado tendeu a dificultar e requerer mais do que os estudantes haviam visto. Assim, usaram-se ferramentas de maneira externa que ajudaram o grupo, mas sem os substituir.

Durante a realização do Projeto, a parte mais agradável foi sem dúvidas poder olhar a um projeto que veio do zero e sentir que conforme aumentava-se o engajamento e paixão do grupo, o projeto começava a tomar uma forma identitária. Assim, a parte de maior regozijar foi poder observar um projeto que crescia dia após dia mediante o trabalho imposto.

O grupo pensa que caso lhes fosse dado mais um mês para se dedicar ao projeto, acabariam por polir mais as mecânicas já presentes e desenvolver ainda mais o universo que o compõe, possivelmente com mais animações e talvez uma história que o envolvesse.

Ademais, o grupo aprofundou ainda mais suas habilidades de pesquisa e desenvolvimento, aprimorando-se como desenvolvedores em aprendizagem e preparando-se para o ano subsequente do curso técnico.

6 REFERÊNCIAS BIBLIOGRÁFICAS

SHARPAX. *Pêndulo Bola de Espinhos / Lâmina na Unity*. YouTube, 18 out. 2021. Disponível em: https://youtu.be/EloJGIdDG_c. Acesso em: 8 nov. 2025.

Desenvolvedor Unity. Como usar o NOVO Sistema de Entradas da Unity | Teclado, Mouse, Touch, Gamepad. YouTube, 25 nov. 2021. Disponível em: https://youtu.be/zhi14HpKqnU. Acesso em: 8 nov. 2025.

BRACKEYS. *How to make AWESOME Scene Transitions in Unity*. YouTube, 12 jan. 2020. Disponível em: https://youtu.be/CE9VOZivb3I. Acesso em: 8 nov. 2025.

VALHALLA DSP. *ValhallaDSP Plugins*. Disponível em: https://valhalladsp.com/plugins/. Acesso em: 8 nov. 2025.

FABFILTER. *Quality Audio Plug-Ins for Mixing, Mastering and Recording*. Disponível em: https://www.fabfilter.com/. Acesso em: 8 nov. 2025.

XFER RECORDS. *Serum 2 – Advanced Wavetable Synthesizer*. Disponível em: https://xferrecords.com/products/serum. Acesso em: 8 nov. 2025.

ROLAND. *Zenology Software Synthesizer*. Disponível em: https://www.roland.com/global/products/rc_zenology/. Acesso em: 8 nov. 2025.

KSHMR. *KSHMR Sample Packs & Loops*. Disponível em: https://splice.com/sounds/kshmr. Acesso em: 8 nov. 2025.

7 APÊNDICE

7.1 Códigos

Os principais códigos do projeto são:

7.1.1 Código do Jogador

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
using UnityEngine.SceneManagement;
using UnityEngine.InputSystem;

public class Player : MonoBehaviour
{
    [Header("Componnents")]
    SpriteRenderer spr;
    Rigidbody2D rb;
    Animator anim;

    [Header("Attributes")]
    public static float speed = 3.5f;
    float jumpForce = 4f;

    [Header("StagesVars")]
    bool isJumping;

    public Vector2 moveInput;

    [Header("Foot")]
    [SerializeField]private Transform foot;
```

```
[SerializeField] float footRadius = 0.1f;
[SerializeField]private LayerMask ground;

public enum stages{
    Idle,
    Running,
    Jumping,
    Dead
}

public static stages currentStage;

// Start is called before the first frame update
void Start()
{
    currentStage = stages.Idle;

    rb = GetComponent<Rigidbody2D>();
    spr = GetComponent<SpriteRenderer>();
    anim = GetComponent<Animator>();
}

private void FixedUpdate()
{
    rb.velocity = new Vector2(moveInput.x * speed, rb.velocity.y);
    Animations();
}

public void Update()
{
    isJumping = !Physics2D.OverlapCircle(foot.position, footRadius, ground);
}

public void Move(InputAction.CallbackContext value)
```

```
{
    if (currentStage == stages.Dead) return;
    moveInput = value.ReadValue<Vector2>();
    currentStage = stages.Running;
}

public void Jump(InputAction.CallbackContext value)
{
    if (currentStage == stages.Dead) return;
    if (value.started && !isJumping)
    {
        rb.AddForce(Vector2.up * jumpForce, ForceMode2D.Impulse);
        currentStage = stages.Jumping;
        isJumping = true;
    }
}

private void Animations()
{
    if (currentStage == stages.Dead) return;
    if (isJumping)
      anim.SetInteger("transition", 2);

    else if (moveInput.x != 0)
      anim.SetInteger("transition", 1);

    else
      anim.SetInteger("transition", 0);

    flipSprite();
}

public IEnumerator Dead()
{
```

```
        currentStage = stages.Dead;
        gameObject.GetComponent<PlayerInput>().DeactivateInput();
        rb.velocity = Vector2.zero;
        rb.simulated = false;

        anim.SetInteger("transition", 3);
        anim.Update(0f);
        yield return new WaitForSeconds(0.5f);

        gameObject.SetActive(false);
        ++deathCount.deathTimes;
        deathCount.itsover = true;

        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }

    void OnCollisionEnter2D(Collision2D other)
    {
        if(other.gameObject.layer == LayerMask.NameToLayer("Hit") ||
other.gameObject.CompareTag("Hit"))
        {
            StartCoroutine(Dead());
        }
    }

    private void OnDrawGizmos()
    {
        if(foot != null)
        {
            Gizmos.color = Color.red;
            Gizmos.DrawWireSphere(foot.position, footRadius);
        }
    }
```

```
    private void flipSprite()
    {
        //caso o personagem ande para a esquerda
        if (moveInput.x<0)
            spr.flipX = true;

        //caso ande para a direita
        else if(moveInput.x>0)
            spr.flipX = false;
    }
}
```

7.1.2 Código do Sensor

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

public class Sensor : MonoBehaviour
{
    [Header("Vectors")]
    Vector2 posicao;
    Vector2 tamanho;

    public float sensorWidth = 1f;
    public float sensorHeight = 1f;

    LayerMask playerLayer;

    public event Action detection;

    // Start is called before the first frame update
    void Start()
```

```
    {
        playerLayer = LayerMask.GetMask("playerLayer");
    }

    // Update is called once per frame
    void Update()
    {
        posicao = (Vector2)transform.position + Vector2.up * sensorHeight;
        tamanho = new Vector2(sensorWidth, sensorHeight);

        Collider2D sensor = Physics2D.OverlapBox(posicao, tamanho, 0f, playerLayer);

        if (sensor != null)
        {
            if(detection != null)
            {
                detection.Invoke();
            }
        }
    }

    private void OnDrawGizmos()
    {
        posicao = (Vector2)transform.position + Vector2.up * sensorHeight;
        tamanho = new Vector2(sensorWidth, sensorHeight);

        Gizmos.color = Color.red;
        Gizmos.DrawWireCube(posicao, tamanho);
    }
}
```

7.1.3 Código do Menu

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.InputSystem;
using UnityEngine.EventSystems;

public class mainMenu : MonoBehaviour
{
    [Header("Panels")]
    public GameObject tutorialImage;
    public GameObject creditsImage;

    [Header("Objects")]
    GameObject Biribo;
    public GameObject target;
    public GameObject Lava;

    private void Start()
    {
        gameUI exisUI = FindObjectOfType<gameUI>();
        Cursor.visible = false;
        Cursor.lockState = CursorLockMode.Locked;

        if (exisUI != null)
        {
            Destroy(exisUI);
        }
    }

    public void Play()
    {
```

```
        StartCoroutine(shortCutscene());
    }

    IEnumerator shortCutscene()
    {
        Biribo = GameObject.FindGameObjectWithTag("Player");
        Biribo.GetComponent<Rigidbody2D>().gravityScale = 3;

        yield return new WaitForSeconds(0.9f);
        Destroy(Biribo);

        while(target.transform.position.y - Lava.transform.position.y > 0.1f)
        {
            Lava.transform.position = Vector2.Lerp(Lava.transform.position,
target.transform.position, 2f * Time.deltaTime);
            yield return null;
        }

        yield return new WaitForSeconds(0.8f);
        SceneManager.LoadScene("LevelSelection");
    }

    public void Tutorial()
    {
        if(!creditsImage.activeSelf)
            tutorialImage.SetActive(true);
    }

    public void Credits()
    {
        if(!tutorialImage.activeSelf)
            creditsImage.SetActive(true);
    }
```

```
    public void Panels()
    {
        tutorialImage.SetActive(false);
        creditsImage.SetActive(false);

        EventSystem ev = EventSystem.current;
        ev.SetSelectedGameObject(ev.firstSelectedGameObject);
    }

    public void NewGame()
    {
        PlayerPrefs.DeleteAll();
        PlayerPrefs.SetInt("Level1Unlocked", 1);
        PlayerPrefs.Save();

        Debug.Log("Progresso Resetado!");
    }

    public void Exit()
    {
        Application.Quit();
    }
}
```

7.1.4 Código da Interface do Jogo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.InputSystem;
using UnityEngine.SceneManagement;
using UnityEngine.EventSystems;

public class gameUI : MonoBehaviour
{
    public static gameUI objectt;
    public Animator animtransition;
    GameObject pauseMenu;

    public void Awake()
    {
        if (objectt != null && objectt != this)
        {
            Destroy(gameObject);
            return;
        }
        DontDestroyOnLoad(gameObject);

        objectt = this;
    }

    public static void createUI(GameObject h)
    {
        if (objectt != null)
            return;

        //verifica se realmente não há na cena
        objectt = FindObjectOfType<gameUI>();
```

```
    if (objectt != null)
    {
        DontDestroyOnLoad(objectt);
        return;
    }

    GameObject obj = Instantiate(h);
    objectt = obj.GetComponent<gameUI>();
}

private void OnEnable()
{
    SceneManager.sceneLoaded += OnSceneLoaded;
}

private void OnDisable()
{
    SceneManager.sceneLoaded -= OnSceneLoaded;
}

private void OnSceneLoaded(Scene scene, LoadSceneMode mode)
{
    if (scene.name == "MainMenu")
    {
        Hide();
    }
}

public void Hide()
{
    background bg = FindObjectOfType<background>();

    foreach (Transform childElements in gameObject.transform)
```

```
        {
            childElements.gameObject.SetActive(false);
        }
    }
}
```

7.1.5 Código do Plano de Fundo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class background : MonoBehaviour
{
    public GameObject[] backgrounds;
    int levelsQuant;
    int n=0;

    [Header("Songs")]
    AudioSource audioSource;
    public AudioClip[] songs;

    // Start is called before the first frame update
    void Start()
    {
        audioSource = GetComponent<AudioSource>();
        //caso o jogador entre direto pelo LevelSelection
        OnSceneLoad(SceneManager.GetActiveScene(), LoadSceneMode.Single);

        PlayerPrefs.Save();
        DontDestroyOnLoad(gameObject);
    }
```

```
private void OnEnable()
{
   SceneManager.sceneLoaded += OnSceneLoad;
}

//caso o jogador entre numa fase dinâmicamente (sem o LevelSelection)
void OnSceneLoad(Scene scene, LoadSceneMode mode)
{
   if (scene.name == "FinalGame" && audioSource!=null)
      audioSource.Stop();

   if (scene.name.StartsWith("Level")){

      if (audioSource == null || songs == null || songs.Length == 0) return;

      int currentNumLevel;
      if(int.TryParse(scene.name.Replace("Level", ""), out currentNumLevel))
      {
         n = (currentNumLevel - 1) / 5;

         if (audioSource.clip != songs[n] && scene.name != "FinalGame")
         {
            audioSource.clip = songs[n];
            audioSource.loop = true;
            audioSource.Play();
         }

         if (currentNumLevel >= 6)
         {
            PlayerPrefs.SetInt("world2Free", 1);
            PlayerPrefs.Save();
         }
      }
```

```
        }

        updatingBackground();
    }

    public void updatingBackground()
    {
        if (backgrounds == null) return;
        //getbuildindex serve para pegar o index da cena só pelo nome
        for (int i = 0; i < backgrounds.Length; ++i)
            if(backgrounds[i]!=null)
                backgrounds[i].SetActive(i == n);
    }
}
```

7.1.6 Código do Final do Level

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

public class EndLevel : MonoBehaviour
{
    int currentMoment;
    [SerializeField] GameObject UIprefab;
    [SerializeField] float transitionTime = 1f;

    void Start()
    {
        gameUI.createUI(UIprefab);
        currentMoment = SceneManager.GetActiveScene().buildIndex;
    }
```

```
    private void OnTriggerEnter2D(Collider2D other)
    {
        if(other.CompareTag("Player"))
        {
            StartCoroutine(loadNewLevel(currentMoment + 1));
        }
    }

    IEnumerator loadNewLevel(int m)
    {
        background currentbg = FindObjectOfType<background>();
        Animator anim = gameUI.objectt.animtransition;

        int levelNumber = int.Parse(SceneManager.GetActiveScene().name.Replace("Level",
""));
        string keyNextLevel = "Level" + (levelNumber + 1) + "Unlocked";
        PlayerPrefs.SetInt(keyNextLevel, 1);
        PlayerPrefs.Save();

        anim.SetBool("start", true);
        yield return new WaitForSeconds(transitionTime);
        SceneManager.LoadScene(m);

        anim.SetBool("start", false);
    }}
```

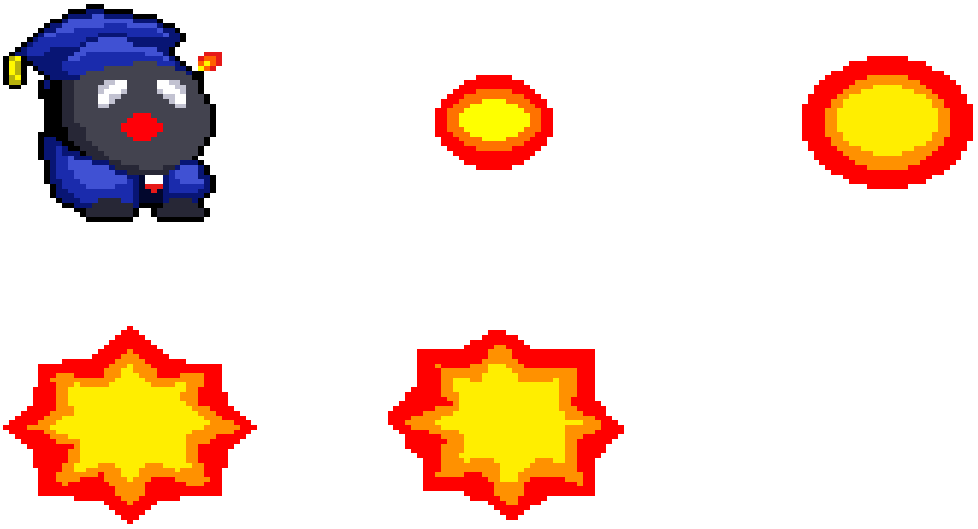
7.2 Sprites

Os principais Tile Sheets (conglomerado de sprites) utilizados durante o jogo foram:

7.2.1 Biribo Andando

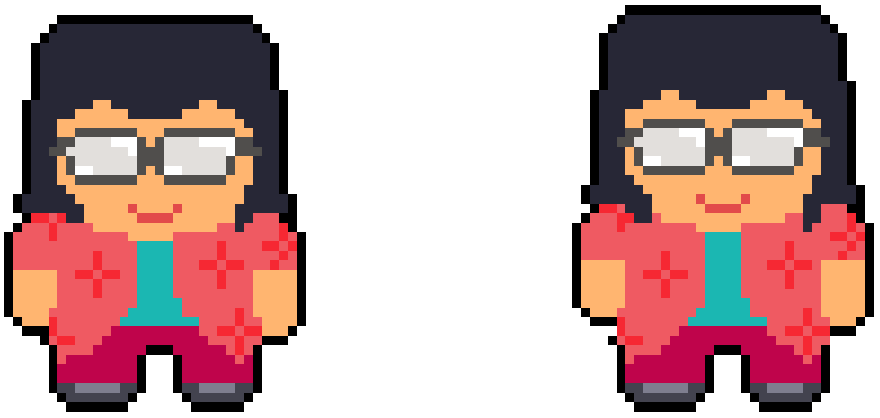
Biribo andando. Elaborado pelo autor.

7.2.1 Biribo Morrendo



Biribo Morrendo. Elaborado pelo autor.

7.2.3 Professora Miyuki

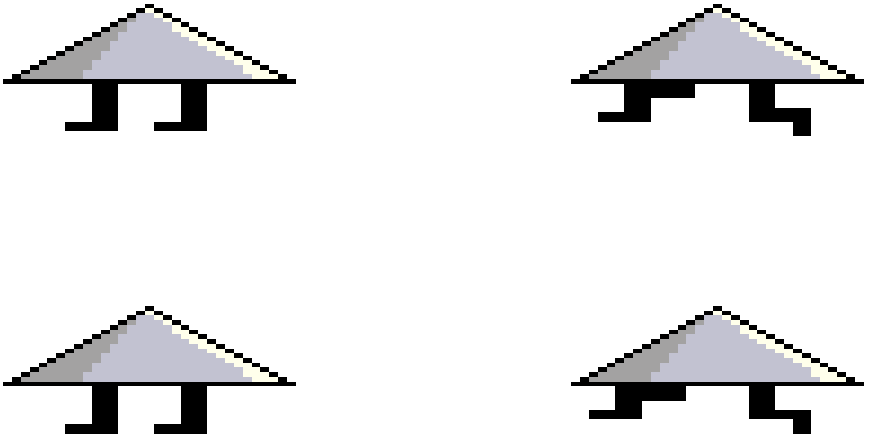


Professora Miyuki. Elaborado pelo autor.

7.2.4 Biribo Pulando

Biribo Pulando. Elaborado pelo autor.

7.2.5 Espinho com Pernas



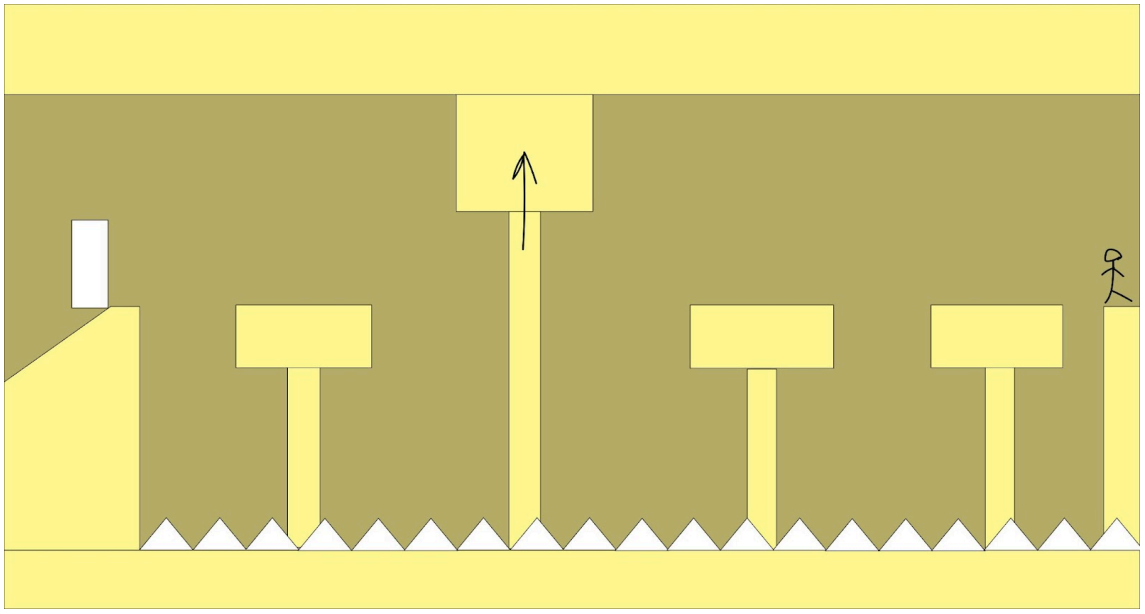
Espinho com Pernas. Elaborado pelo autor.

7.3 Storyboards

Os StoryBoards usados durante o decorrer do projeto foram:

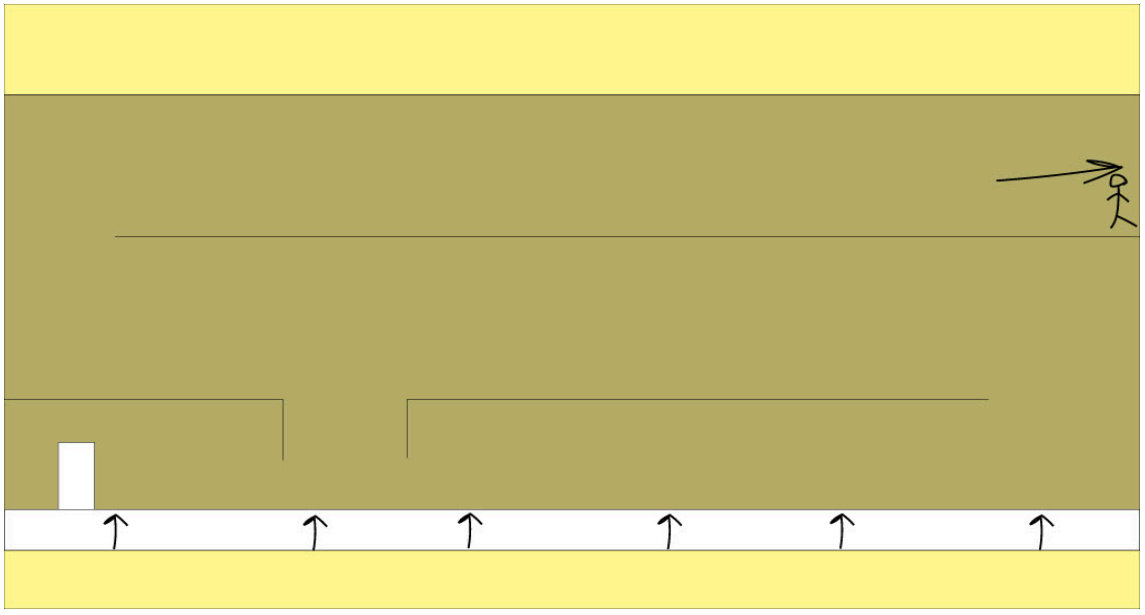
7.3.1 StoryBoard das Fases

7.3.1.1 StoryBoard Fase 1



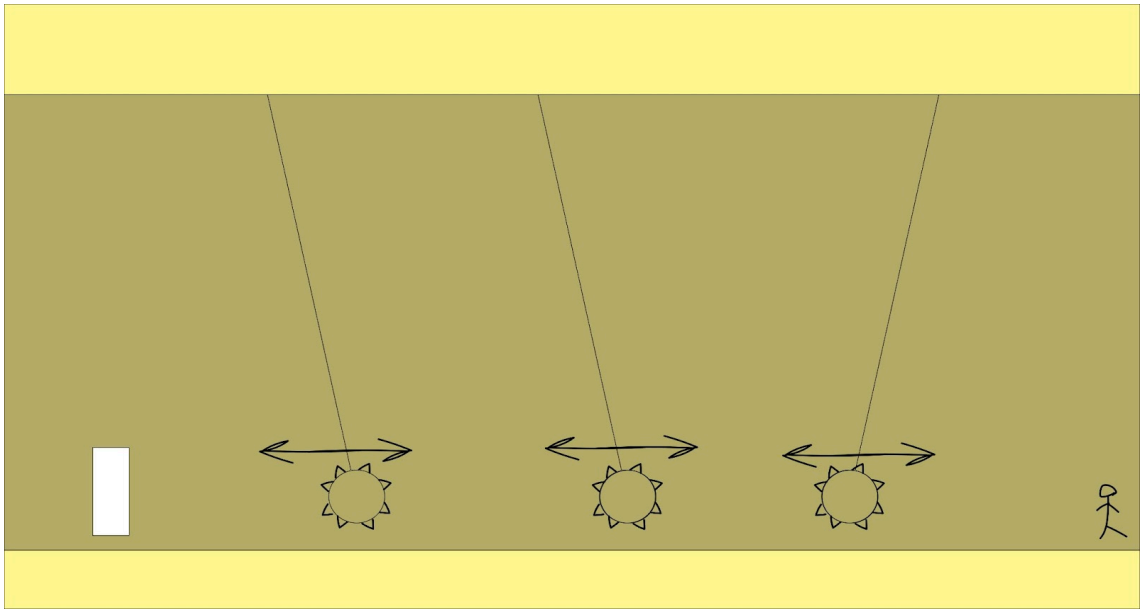
Storyboard da fase 1. Elaborado pelo autor.

7.3.1.2 StoryBoard Fase 2



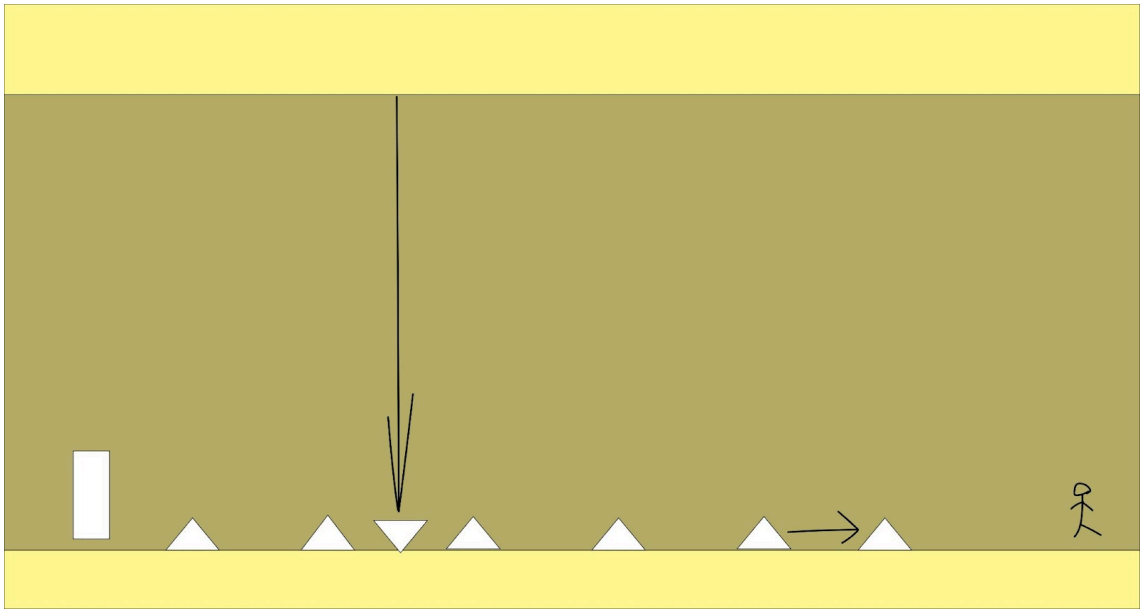
Storyboard da fase 2. Elaborado pelo autor.

7.3.1.3 StoryBoard Fase 3



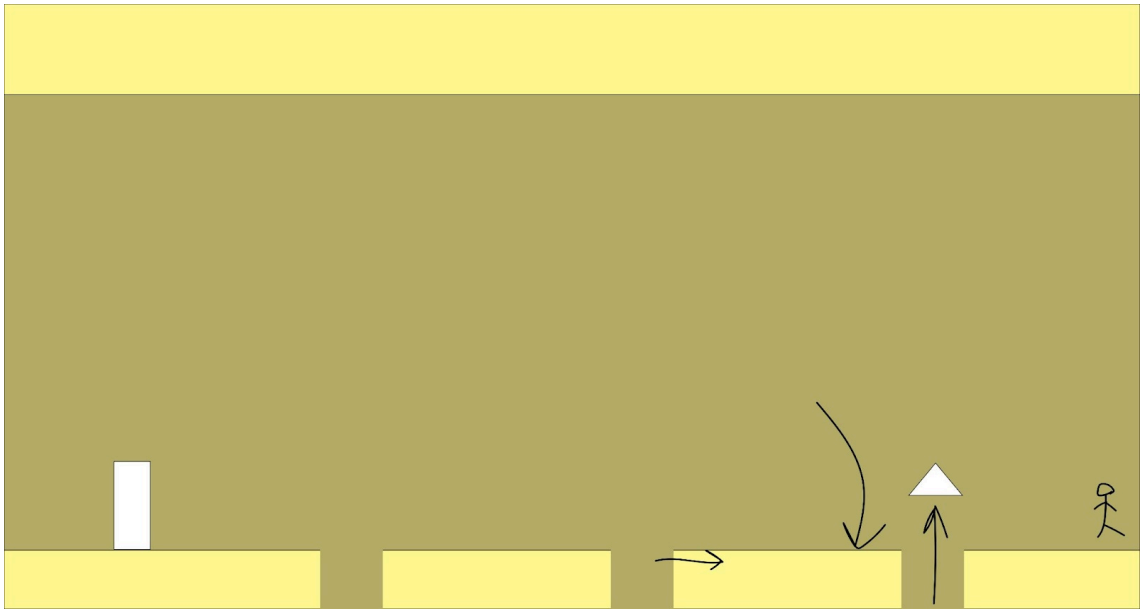
Storyboard da fase 3. Elaborado pelo autor.

7.3.1.4 StoryBoard Fase 4



Storyboard da fase 4. Elaborado pelo autor.

7.3.1.5 StoryBoard Fase 5

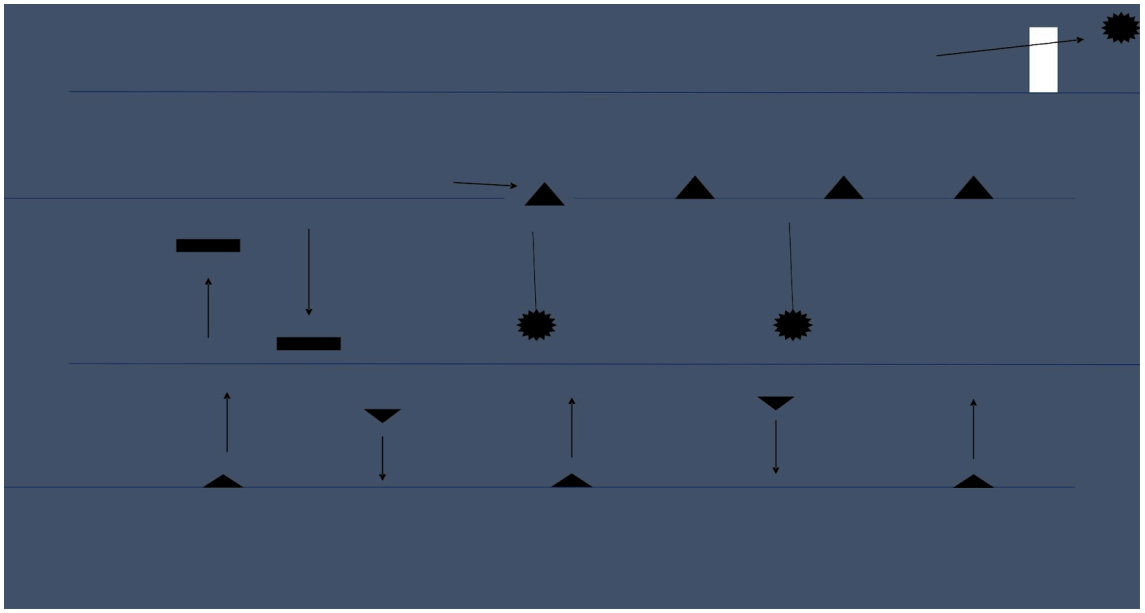


Storyboard da fase 5. Elaborado pelo autor.

7.3.1.6 StoryBoard Fase 6

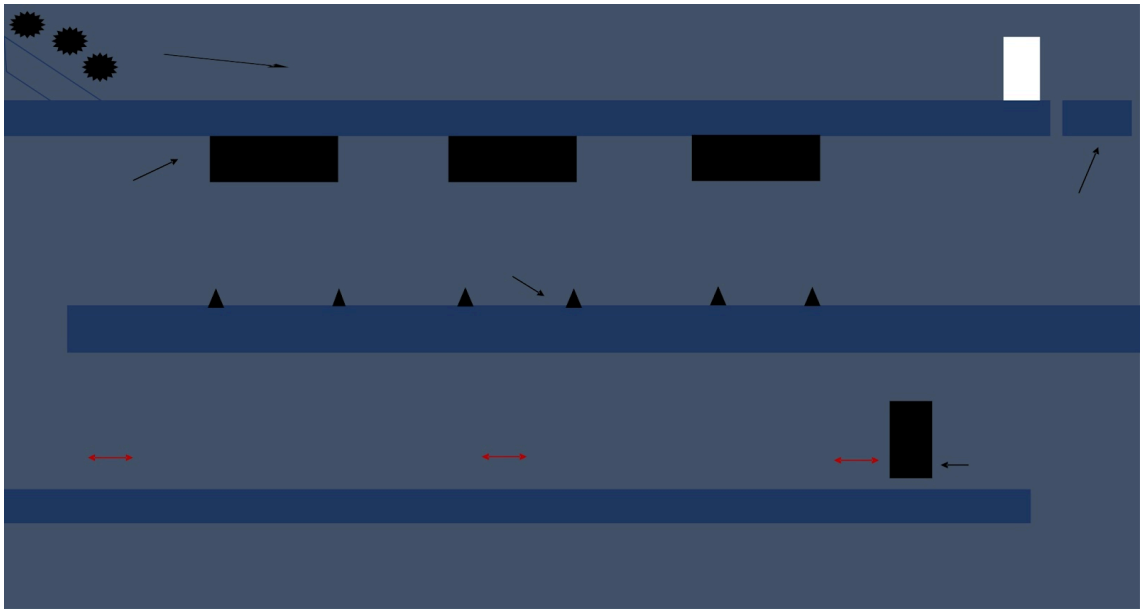
Storyboard da fase 6. Elaborado pelo autor.

7.3.1.7 StoryBoard Fase 7



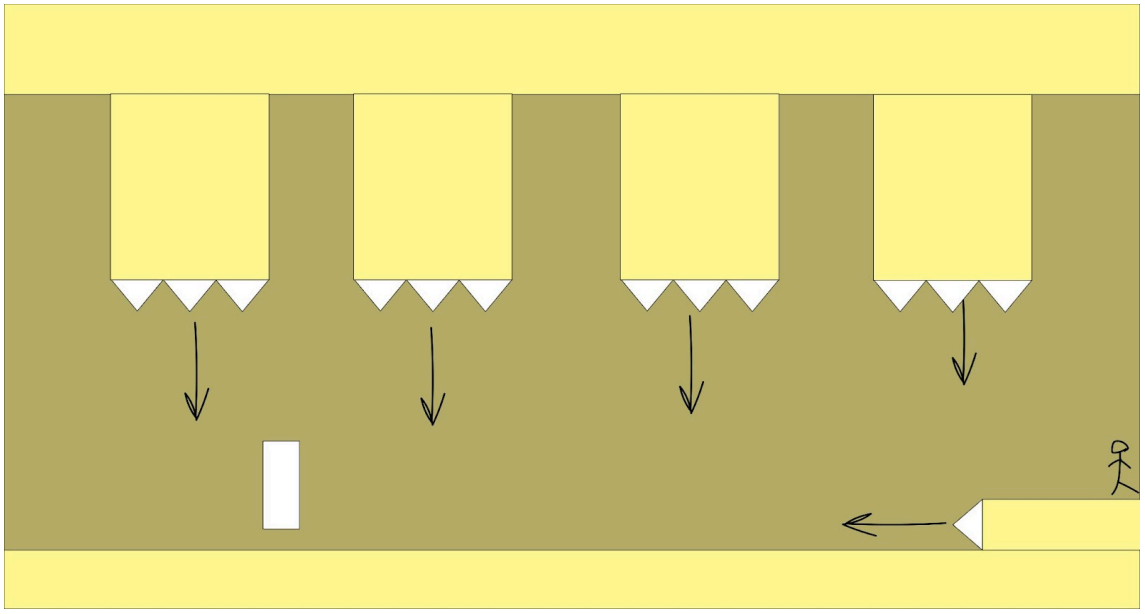
Storyboard da fase 7. Elaborado pelo autor.

7.3.2.8 StoryBoard Fase 8



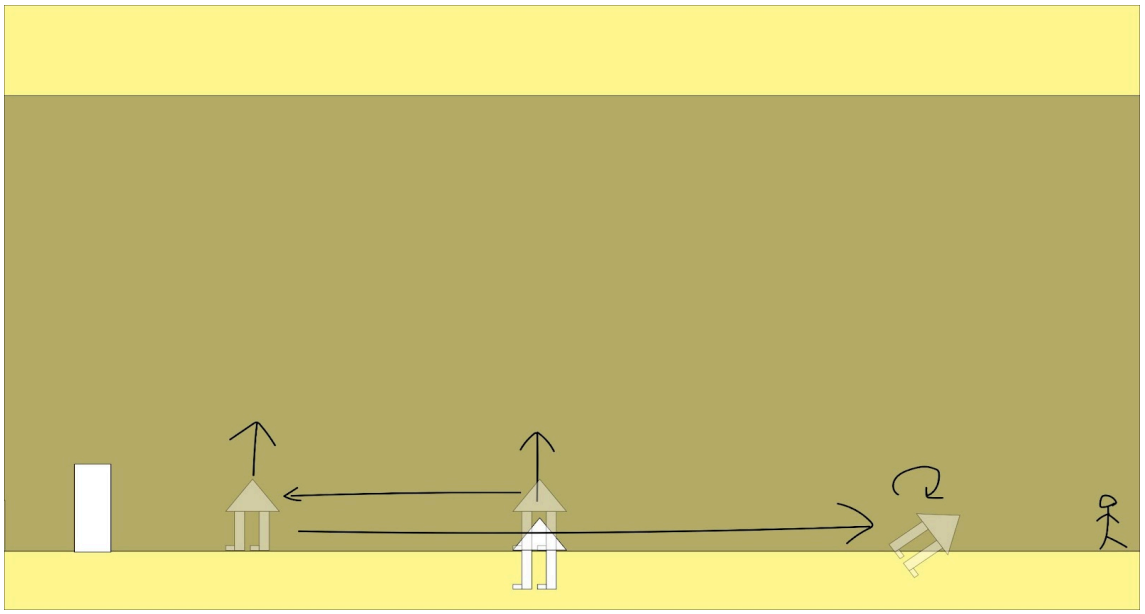
Storyboard da fase 8. Elaborado pelo autor.

7.3.2.9 StoryBoard Fase 9



Storyboard da fase 9. Elaborado pelo autor.

7.3.2.10 StoryBoard Fase 10



Storyboard da fase 10. Elaborado pelo autor.