INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO – CAMPUS CAMPOS DO JORDÃO

ANDREY DE SOUZA SETÚBAL DESTRO

CAVALEIRO DE CAMPOS

CAMPOS DO JORDÃO
2025

RESUMO

O presente relatório descreve o processo completo de desenvolvimento do jogo digital 2D "Cavaleiro de Campos", elaborado como projeto da disciplina de Tópicos Avançados I do curso de Análise e Desenvolvimento de Sistemas do IFSP - Campus Campos do Jordão. O jogo foi construído integralmente em JavaScript utilizando o framework Phaser 3, com suporte do ambiente de desenvolvimento Vite, abordando a concepção narrativa, a produção de assets visuais e sonoros, a implementação técnica modular, os testes e os resultados obtidos. A narrativa acompanha um cavaleiro guardião responsável por resgatar o cristal do sol, enfrentando górgonas que ameaçam as plantações da região. O projeto apresenta pipeline de carregamento dedicado (PreloadScene), sistema completo de cenas (título, lore e gameplay), mecânicas de ataque, defesa e manipulação de estados, inteligência artificial básica para inimigos, gestão de HUD, controle de pontuação, trilhas sonoras contextuais e tratamento de spritesheets. O relatório explicita escolhas técnicas, ferramentas empregadas, desafios enfrentados e perspectivas de evolução, evidenciando a consolidação das competências práticas e teóricas do estudante.

Palavras-chave: Desenvolvimento de jogos. Phaser 3. JavaScript. Game design. IFSP.

ABSTRACT

This report details the full development cycle of "Cavaleiro de Campos", a 2D digital game produced for the Advanced Topics I course in the Systems Analysis and Development program at IFSP - Campos do Jordão. Built entirely in JavaScript with the Phaser 3 framework and the Vite build environment, the project covers narrative conception, visual and audio asset production, modular technical implementation, testing, and outcomes. The storyline follows a guardian knight whose mission is to reclaim the sun crystal while facing Gorgon creatures that threaten local crops. The game implements a dedicated preload pipeline, a comprehensive scene system (title, lore, and gameplay), player mechanics for attack, defense, and state handling, basic enemy AI, HUD management, scoring system, contextual soundtracks, and spritesheet processing. This document discusses the technical decisions, tools, challenges, and future improvements, demonstrating the student's mastery of applied concepts.

Keywords: Game development. Phaser 3. JavaScript. Gameplay. IFSP.

SUMÁRIO

1 INTRODUÇÃO

A produção de jogos digitais configura-se como atividade interdisciplinar que integra lógica de programação, matemática, narrativa, design gráfico e experiência do usuário. No contexto acadêmico dos cursos de tecnologia, projetos de jogos permitem consolidar conhecimentos de maneira aplicada, estimulando o raciocínio criativo e o domínio de ferramentas modernas. O relatório aqui apresentado descreve minuciosamente o desenvolvimento do jogo "Cavaleiro de Campos", elaborado como atividade avaliativa da disciplina Tópicos Avançados I do curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP), campus Campos do Jordão.

1.1 Contextualização

O projeto foi motivado pela necessidade de produzir um jogo funcional que demonstrasse a integração entre narrativa temática, mecânicas de *gameplay* consistentes, arquitetura de software modular e uso adequado de assets audiovisuais. A escolha do framework Phaser 3 se deu pela robustez na criação de jogos 2D, ampla documentação e suporte a recursos como *scenes* independentes, sistema de física arcade, animações baseadas em *spritesheet*, gerenciamento de áudio e integrações com bibliotecas de terceiros. O ambiente de desenvolvimento Vite foi empregado para proporcionar servidor local rápido, suporte a ES modules e empacotamento eficiente.

1.2 Problema e Tema

O enredo de "Cavaleiro de Campos" parte de uma crise fictícia: o cristal do sol, responsável por garantir o clima propício às colheitas da região, é roubado por criaturas conhecidas como górgonas. Sem o cristal, a temperatura despenca e os campos se tornam improdutivos. O jogador assume o papel do cavaleiro guardião, incumbido de enfrentar os inimigos, resistir aos ataques e restituir o equilíbrio. O tema associa fantasia a elementos do cotidiano agrícola do interior paulista, estabelecendo paralelos com desafios reais enfrentados por comunidades rurais.

1.3 Organização do Relatório

Para oferecer visão abrangente, o documento está estruturado em oito capítulos: introdução, objetivos, justificativa, referencial teórico, metodologia, descrição detalhada da implementação, resultados e conclusão. Ao final, são listadas referências bibliográficas e apêndices contendo informações complementares (assets, estrutura de pastas, capturas). Cada seção destaca aspectos específicos do projeto, permitindo que o leitor compreenda tanto a concepção conceitual quanto a execução técnica.

OBJETIVOS

2.1 Objetivo Geral

Desenvolver um jogo digital 2D, intitulado "Cavaleiro de Campos", que combine narrativa imersiva, mecânicas de combate, progressão de dificuldade e estética retrô, utilizando o framework Phaser 3 e boas práticas de desenvolvimento web moderno.

2.2 Objetivos Específicos

- Organizar a aplicação em *scenes* distintas (*PreloadScene*, *TitleScene*, *LoreScene*, *GameScene*) garantindo fluidez de transição e isolamento de responsabilidades.
- Implementar o conjunto completo de animações, estados e controles do protagonista, incluindo ataque com espada, defesa ativa, movimentação lateral, salto, dano e morte.
- Desenvolver inimigos do tipo górgona com IA baseada em perseguição, ataques temporizados, janela de dano controlada e resposta a golpes do jogador.

- Criar HUD com exibição de pontuação, barra de vida dinâmica e mensagens situacionais (*game over*, reiniciar, menu).
- Integrar trilhas sonoras específicas para menu, narrativa, *gameplay* e tela de derrota, bem como efeitos de áudio sincronizados a eventos chave.
- Garantir consistência estética por meio de tipografia Press Start 2P, sombras e contornos uniformes, e paleta de cores inspirada em jogos retrô.
- Documentar o fluxo de desenvolvimento, as decisões técnicas e os desafios enfrentados, compondo relatório em formato acadêmico.

3 JUSTIFICATIVA

3.1 Relevância Acadêmica

A construção de um jogo completo permite revisar conteúdos trabalhados ao longo do curso, como modularização de código, manipulação de eventos, uso de APIs gráficas, estruturação de pastas, controle de dependências e, principalmente, desenvolvimento iterativo com testes sucessivos. Ao documentar cada etapa, o estudante consolida habilidades de comunicação técnica, importantes em ambientes profissionais e acadêmicos.

3.2 Relevância Tecnológica

Phaser 3 é ferramenta difundida na indústria independente e na comunidade de desenvolvedores web. Dominar seus recursos amplia possibilidades de atuação em áreas como *serious games*, simuladores e aplicações interativas. A integração com Vite, Node.js e ES modules reforça práticas modernas de desenvolvimento, alinhadas a *pipelines* utilizados em empresas de software.

3.3 Relevância Narrativa e Cultural

Situar a história em Campos do Jordão e associá-la ao conceito de "cristal do sol" agrega identidade local e valor simbólico, aproximando o público de temáticas rurais e agrícolas. A representação de górgonas como ameaças às colheitas permite discutir, de maneira lúdica, impactos de crises climáticas e a importância de protetores (representados pelo cavaleiro) na manutenção do equilíbrio.

4 REFERENCIAL TEÓRICO

4.1 Jogos Digitais como Ferramenta Educacional

Autores como Prensky (2001) e Paiva (2017) destacam o potencial dos jogos para promover aprendizado ativo. No contexto do IFSP, projetos aplicados ampliam a retenção de conceitos de programação ao relacioná-los com produtos tangíveis. "Cavaleiro de Campos" atua como laboratório em que diversas áreas convergem.

4.2 Arquitetura de Scenes no Phaser 3

Phaser implementa *scenes* como classes derivadas de Phaser.Scene, cada qual com métodos preload, create e update. Esse modelo reflete padrões de máquina de estados e facilita a divisão de responsabilidades, reduzindo acoplamento. O projeto utiliza *PreloadScene* para carregamento, *TitleScene* e *LoreScene* para narrativa, e *GameScene* para jogabilidade. As transições são controladas por this.scene.start, enquanto dados compartilhados são gerenciados por configurações passadas via construtor.

4.3 Programação Orientada a Eventos e Game Loop

O motor atualiza a física e renderização em cada *frame*, chamando update de todas as *scenes* ativas. Interações com teclado, mouse ou eventos temporizados são tratadas por *listeners* (this.input.keyboard, this.time.addEvent). Essa abordagem orientada a eventos simplifica a coordenação de estados, fundamental para o manejo dos ataques, defesas e *respawns*.

4.4 Spritesheets, Animações e Técnicas de Renderização

Spritesheets concentram múltiplos *frames* em uma única imagem, reduzindo requisições e permitindo animações fluidas. Funções como this.anims.create e generateFrameNumbers fatiam os *frames*. A estética retrô exige escala dupla (setScale(2)) e image-rendering: pixelated no CSS para preservar nitidez e sensação de jogos 16-bit.

4.5 Física Arcade e Gerenciamento de Colisões

Phaser Arcade Physics provê métodos simples para adicionar corpos físicos, aplicar gravidade e detectar sobreposições. No jogo, o cavaleiro utiliza this.physics.add.sprite, com gravidade de 600 no eixo Y e colisão com as bordas (setCollideWorldBounds). Górgonas rastreiam o jogador medindo distância e ajustando velocidade. Ataques controlam janela de dano para evitar múltiplos *hits* por animação.

4.6 Storytelling, HUD e Feedback ao Jogador

A narrativa é reforçada via textos estilizados. Funções utilitárias centralizam tipografia Press Start 2P, cor amarela (#fcee09) e sombras (*offset* 6 px). HUD inclui pontuação ("Pontuação: 0"), barras de vida e mensagens de fim. O feedback auditivo (trilhas e efeitos) reforça estados de jogo (menu, combate, derrota).

5 METODOLOGIA

5.1 Planejamento Narrativo

A narrativa foi concebida com foco na dualidade luz/escuridão e na defesa das colheitas. O cristal do sol representa estabilidade climática. Górgonas simbolizam

pragas que emergem da mata. O cavaleiro age como guardião ancestral dos campos. O fluxo narrativo se organiza em:

- Introdução (*TitleScene*) com chamada ao jogador.
- *LoreScene* contextualizando a crise.
- *GameScene* com ondas de inimigos e objetivos implícitos (sobrevivência, pontuação).
- *Game over* exibindo texto que destaca as consequências da derrota.

5.2 Pipeline de Assets Visuais

Spritesheets, *backgrounds* e ícones foram organizados em public/assets. Cavaleiro e górgona possuem dimensões 128x128 px, permitindo animações suaves em *Idle*, *Walk*, *Attack*, *Hurt* e *Dead*. A barra de vida é composta por retângulos (this.add.rectangle). O CSS (style.css) define *layout* responsivo, centraliza o *canvas* e aplica bordas que simulam moldura de arcade.

5.3 Implementação Técnica

A lógica central inicia em index.html, que importa main.js. Esse arquivo calcula dimensões responsivas (90% da *viewport*, com limites de 900x600), registra configurações compartilhadas e instancia Phaser.Game após garantir o carregamento da fonte Press Start 2P. A lista de *scenes* inclui *PreloadScene*, *TitleScene*, *LoreScene* e *GameScene*, criando *pipeline* previsível.

PreloadScene exibe barra de carregamento e utiliza this.load.setPath('assets') para reduzir repetição. Carrega *backgrounds*, *sprites* do cavaleiro e das górgonas, além das trilhas music_title, music_gameplay, music_gameover e efeitos sword_attack, gorgon_attack. Ao finalizar, inicia *TitleScene*.

TitleScene posiciona background_start, reinicia áudio, mostra título centralizado e botões interativos (Começar e Sair). Eventos pointerover aplicam

highlight via *tint*. A assinatura "2025 - Desenvolvido por Andrey Destro" ocupa o rodapé. Ao clicar em Começar, a cena migra para *LoreScene*.

LoreScene reaproveita a trilha do menu, exibe narrativa *multiline* com *word wrap* automático e orienta o jogador a pressionar ENTER ou clicar para continuar. O texto descreve o furto do cristal e convoca o cavaleiro. A função continueToGame interrompe o áudio e inicia *GameScene*.

GameScene concentra física, HUD e IA. A cena:

- Inicia variáveis de estado (maxHealth, score, *flags* de ataque, defesa, dano, morte).
- Mostra background_stage, reproduz music_gameplay em *loop* e cria texto de pontuação.
- Configura controles (*cursors*, espaço para ataque, shift para defesa).
- Registra animações de cavaleiro (*idle, walk, attack1, jump, knight_hurt, knight_defend, knight_dead*) e górgona (*idle, walk, attack1, hurt, dead*) com *helper* que evita duplicação quando a cena reinicia.
- Instancia *Knight* centralizado, cria barra de vida acima do herói e agenda *spawn* de górgonas: uma inicial na direita e novas a cada 6 segundos, alternando bordas.
- No update, ajusta barra de vida conforme dano, processa defesa (mantendo cavaleiro parado e animação *defend*), controla lista de inimigos ativos e verifica janela de dano inimiga (enemy.damageWindowActive). Quando o jogador é atingido sem defender, perde 10 pontos de vida, toca gorgon_attack, aciona animação *hurt* e, se vida chegar a zero, executa knight_dead seguido de handleGameOver.
- Ataques com espaço interrompem movimento, executam animação *attack1* e selecionam automaticamente o inimigo mais próximo dentro de 150 px, aplicando takeDamage(25).
- Movimentação horizontal usa velocidade 340 px/s; salto aplica -350 no eixo Y. Animações alternam entre *idle*, *walk* e *jump* conforme contexto.

- handleGameOver pausa física, exibe *overlay* translúcido, mostra pontuação final, narrativa de derrota e botões interativos para Reiniciar (scene.restart) ou Menu (scene.start TitleScene). A trilha music_gameover toca sem *loop*.
- playSfxQuiet garante reprodução de efeitos apenas quando carregados, evitando erros.

5.4 Organização de Código

Classes *Knight* e *Gorgon* residem em src/objects, reforçando separação entre entidades. O módulo src/utils/text.js concentra estilo tipográfico, aplicando sombra e *stroke* uniformes. A estrutura garante reuso sem duplicar código. Assets ficam em public/assets (*audio* e *images*), enquanto estilos residem em public/css.

5.5 Integração de Áudio

As trilhas são carregadas na *PreloadScene* e controladas em cada cena. music_title acompanha menu e lore, music_gameplay embala combate e music_gameover reforça derrota. Os efeitos sword_attack (golpe do cavaleiro) e gorgon_attack (dano recebido) sincronizam com animações. A função this.sound.stopAll() evita sobreposição ao trocar de cena.

5.6 Ferramentas Utilizadas

- Phaser 3 (framework principal).
- Vite (servidor e *bundler*).
- Node.js/npm (gerenciamento de dependências).
- VS Code (edição).
- Ferramentas de IA e editores gráficos (para gerar e lapidar *sprites*).
- MusicGPT para trilhas e efeitos.
- CraftPix.net para *assets* e *backgrounds*.

5.7 Estratégia de Testes e Validação

Foram executados testes manuais a cada módulo: verificação de carregamento de assets, resposta do cavaleiro, sincronização de barras de vida, *spawn* de górgonas, tratamento de *game over*, *loops* de áudio e transições de cena. Ajustes iterativos corrigiram distâncias de ataque, tempos de *cooldown* e alinhamento das barras relativas aos *sprites*.

6 DESCRIÇÃO DETALHADA DA IMPLEMENTAÇÃO

6.1 Estrutura Global do Projeto

O projeto utiliza index.html como ponto de entrada, carregando main.js. A pasta public armazena assets em subpastas audio e images. Scripts npm run dev, npm run build e npm run preview estão definidos em package.json. O arquivo style.css controla *layout* e *pixel art*.

6.2 Scene PreloadScene

Responsável por carregar o conteúdo a partir de assets/. A cena:

- Exibe barra de progresso com gráfico e texto "Carregando...".
- Define caminho raiz via this.load.setPath.
- Importa imagens de *background* (start, lore, stage) e *spritesheets* do cavaleiro e da górgona (*idle, walk, attack, hurt, dead*).
- Carrega músicas (title, gameplay, gameover) e efeitos (*gorgon_attack*, *sword_attack*).
- Remove barra ao concluir e inicia *TitleScene* automaticamente.

6.3 Scene TitleScene

Carrega background_start em tela cheia, inicia music_title e apresenta título centralizado com createStyledText. Os botões "Começar" e "Sair" utilizam interações pointerover para destaque. O primeiro leva a *LoreScene*; o segundo interrompe áudio e exibe alerta de agradecimento. O rodapé credita o autor.

6.4 Scene LoreScene

Exibe background_lore dimensionado, reaproveita music_title e apresenta narrativa *multiline* centralizada com *word wrap* de 80% da largura. O jogador avança com ENTER ou clique. A função continueToGame garante parada de áudio antes de iniciar *GameScene*.

6.5 Scene GameScene

Implementa a jogabilidade principal:

- Inicializa estados (vida, *score*, *flags*).
- Cria HUD com texto "Pontuação: 0".
- Configura teclas direcionais, espaço e shift.
- Cria animações de cavaleiro e górgona com *helper* maybeCreate.
- Instancia *Knight* no centro inferior e adiciona *container* com barra de vida (background cinza e preenchimento vermelho) sincronizada ao *sprite*.
- Spawna górgona inicial na direita e agenda evento this.time.addEvent com *delay* 6000 ms para novos inimigos nas bordas.
- No update, sincroniza barra do cavaleiro, processa defesa (mantendo animação *defend* enquanto tecla shift estiver pressionada), percorre tabelas de inimigos atualizando cada Gorgon.update.
- Detecta dano inimigo quando enemy.damageWindowActive e hasDealtDamage falso, com distância menor que 90 px. Sem defesa, reduz vida em 10, toca

gorgon_attack, aciona animação *hurt* e, se vida chegar a zero, executa knight_dead seguido de handleGameOver.

- Garantia de que estados *hurt*, *defend* ou *attack* bloqueiam novos comandos até suas conclusões.
- Ataque com espaço seleciona inimigo mais próximo (distância menor que 150 px), aplica takeDamage(25) e toca sword_attack.
- Movimento lateral controla *flipX* e velocidade; salto ocorre com setVelocityY(-350). Animações *idle*, *walk* e *jump* são alternadas automaticamente.
- handleGameOver pausa física, exibe *overlay*, pontuação e narrativa de derrota, criando botões interativos para reiniciar e retornar ao menu. A música de derrota toca sem *loop*.

6.6 Classe Knight (src/objects/Knight.js)

Instancia *sprite* com scene.physics.add.sprite, aplica escala 2, *depth* 1 e setCollideWorldBounds(true). A animação *idle* inicia imediatamente. O construtor recebe posição X/Y, permitindo centralização pela *GameScene*.

6.7 Classe Gorgon (src/objects/Gorgon.js)

Instancia *sprite* escalado, com *depth* 1 e colisões com limites. Define vida máxima 100, cria retângulos para barra de vida (fundo cinza e preenchimento vermelho) posicionados 90 px acima do inimigo. O comportamento inclui:

- Variáveis attackCooldown, isAttacking, hasDealtDamage, damageWindowActive e evento temporizado.
- Método takeDamage que reduz vida, atualiza barra, reproduz animação gorgon_hurt quando sobrevivente e gorgon_dead ao morrer. Desabilita corpo físico após morte e incrementa pontuação em 100.
- Método update que calcula distância ao cavaleiro. Se acima de 80 px, aproxima com velocidade 50 e animação *walk*; caso contrário, inicia animação de ataque

com delayedCall de 320 ms para abrir janela de dano de 180 ms. O *cooldown* impede ataques consecutivos.

- Sincronização da barra de vida com posição do *sprite* a cada *frame*.

6.8 Utilitários e Estilos Compartilhados

O módulo src/utils/text.js exporta applyTextEffects e createStyledText, padronizando fonte Press Start 2P, cor branca, sombra preta (*offset* 6 px) e *stroke*. O CSS em public/css/style.css centraliza *canvas*, define limites máximos (1100x740), aplica borda escura de 8 px e sombra para imitar gabinete arcade. O *canvas* usa image-rendering: pixelated.

7 RESULTADOS OBTIDOS

7.1 Avaliação Funcional

O jogo apresenta fluxo completo: menu -> lore -> gameplay -> game over/menu. O cavaleiro responde imediatamente aos comandos, defesas cancelam dano recebido, ataques possuem feedback sonoro. Inimigos perseguem o jogador corretamente, realizam ataques com janela de dano coerente e sofrem animação de morte antes de serem removidos. Pontuação atualiza dinamicamente. Reinício retoma estado inicial sem recarregar a página.

7.2 Avaliação Estética e de Usabilidade

Tipografia retrô e sombras consistentes aumentam legibilidade. Fundo e *sprites* mantêm coesão cromática (verdes, amarelos, tons escuros). Barra de vida do cavaleiro acompanha *sprite*, destacando dano. Trilhas sonoras distintas proporcionam variação emocional (tensão no *gameplay*, melancolia no *game over*). Interface utiliza linguagem em português, alinhada ao público alvo.

7.3 Desafios e Soluções Encontradas

- **Sincronização de áudio:** implementado stopAll antes de iniciar novas trilhas, evitando sobreposição.
- **Controle de estado do cavaleiro:** *flags* isAttacking, isDefending, isHurt e isDead impediram conflitos de animação.
- **Janela de dano das górgonas:** uso de this.scene.time.delayedCall para ativar e desativar damageWindowActive com duração controlada.
- **Atualização de barras de vida:** posicionadas dentro do update do inimigo para acompanhar deslocamentos.
- **Responsividade:** cálculo de largura/altura baseado em 90% da *viewport* garantiu bom aproveitamento em diferentes resoluções.
-

7.4 Contribuições para a Formação Acadêmica

O projeto consolidou práticas de arquitetura modular, manipulação de estados, *timers*, eventos e colisões. Reforçou capacidade de depuração, planejamento narrativo, integração de áudio e documentação formal. Serviu como peça de portfólio e exercício de comunicação técnica.

8 CONCLUSÃO

8.1 Síntese do Projeto

"Cavaleiro de Campos" atingiu objetivos ao fornecer experiência jogável completa com estética própria, estrutura robusta e narrativa coerente. O aluno dominou recursos do Phaser 3, aplicou Vite e organizou código de maneira clara.

8.2 Recomendações e Trabalhos Futuros

- **Novas fases e cenários:** ampliar repertório visual e progressão de dificuldade.

- **Variedade de inimigos:** introduzir unidades com ataques à distância ou padrões diferenciados.
- **Sistema de *power-ups*:** adicionar *buff* temporário de defesa, velocidade ou cura.
- **Leaderboard:** armazenar pontuações em persistência local ou remota.
- **Cutscenes e cinematics:** reforçar narrativa com transições animadas.
- **Portabilidade mobile:** adaptar controles para toque e otimizar UI.
-

8.3 Considerações Finais

O jogo prova ser possível criar produto lúdico funcional em curto prazo utilizando tecnologias acessíveis. A documentação fortalece repertório de escrita técnica e pode servir de base para apresentação acadêmica, submissão em eventos de tecnologia educacional ou expansão como projeto de conclusão de curso.

REFERÊNCIAS

BRASIL. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo. **Slides da disciplina de Tópicos Avançados**. Curso de Análise e Desenvolvimento de Sistemas. Campos do Jordão, 2025.

CHROME DEVELOPERS. **Ferramentas de desenvolvimento (DevTools)**. Disponível em: https://developer.chrome.com/docs/devtools/. Acesso em: 24 nov. 2025.

CLIDEO. **Ferramenta de corte e edição de áudio online**. Disponível em: https://clideo.com/pt/cut-audio. Acesso em: 24 nov. 2025.

GOOGLE. **Gemini - IA generativa para imagens**. Disponível em: https://gemini.google.com/. Acesso em: 24 nov. 2025.

LUDO AI. **Ferramenta de criação de sprites e conceitos artísticos**. Disponível em: https://ludo.ai/. Acesso em: 24 nov. 2025.

MOZILLA DEVELOPER NETWORK. **Documentação JavaScript (ES6)**. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. Acesso em: 24 nov. 2025.

OPENAI. **ChatGPT - Ferramenta de apoio a conteúdo visual e textual**. Disponível em: https://chatgpt.com/. Acesso em: 24 nov. 2025.

PHASER. **Phaser 3 Framework**: documentação oficial. Disponível em: https://phaser.io/. Acesso em: 24 nov. 2025.

PIXABAY. **Biblioteca gratuita de músicas e efeitos sonoros**. Disponível em: https://pixabay.com/. Acesso em: 24 nov. 2025.

VITE. **Vite - Build tool para desenvolvimento Front-End**. Disponível em: https://vitejs.dev/. Acesso em: 24 nov. 2025.

APÊNDICE A – LISTA DE ASSETS UTILIZADOS

- **background_start.png:** imagem utilizada em *TitleScene*.
- **background_lore.png:** arte para *LoreScene*.
- **background_stage.png:** cenário principal do *gameplay*.
- **title.mp3:** trilha da tela inicial e narrativa.
- **gameplay.mp3:** trilha da cena principal.
- **gameover.mp3:** música de fim de jogo.
- **sword_attack.mp3:** efeito do ataque do cavaleiro.
- **gorgon_attack.mp3:** ataque do inimigo.
- **images/sprites/knight/Idle.png:** *spritesheet Idle* do protagonista.
- **images/sprites/knight/Walk.png:** *spritesheet* de corrida.
- **images/sprites/knight/Attack 1.png:** *spritesheet* do ataque principal.
- **images/sprites/knight/Jump.png:** *spritesheet* de salto.
- **images/sprites/knight/Defend.png:** *spritesheet* de defesa.
- **images/sprites/knight/Hurt.png:** *spritesheet* de dano.
- **images/sprites/knight/Dead.png:** *spritesheet* de morte.
- **images/sprites/knight/Protect.png:** *sprite* adicional para futuras habilidades.
- **images/sprites/gorgon/Idle.png:** *spritesheet idle* do inimigo.
- **images/sprites/gorgon/Walk.png:** animação de deslocamento.
- **images/sprites/gorgon/Attack_1.png:** animação principal de ataque.
- **images/sprites/gorgon/Attack_2.png** e **Attack_3.png:** recursos extras para expansão.

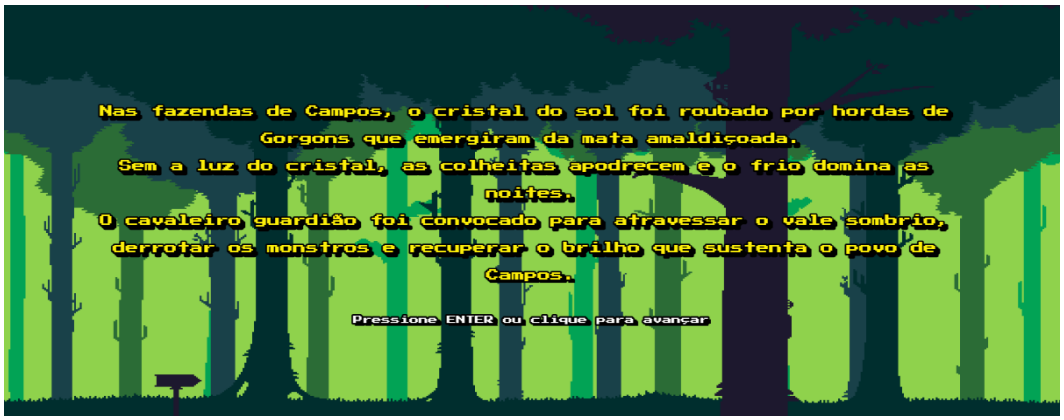
- **images/sprites/gorgon/Hurt.png:** animação de dano.
- **images/sprites/gorgon/Dead.png:** animação de morte.
- **images/sprites/gorgon/Idle_2.png**, **Run.png**, **Special.png:** *sprites* reservas para novos comportamentos.

APÊNDICE B – CAPTURAS DE TELA DO JOGO

1. Tela inicial com título "Cavaleiro de Campos" e opções "Começar" e "Sair".

Fonte: O autor (2025)

2. Cena de *lore* apresentando narrativa sobre o roubo do cristal do sol.



3. *Gameplay mostrando cavaleiro, barra de vida, pontuação e górgona em ataque.*

4.

Fonte: O autor (2025)

5. Tela de "Fim de jogo" com pontuação final, narrativa de derrota, botões de "Reiniciar" e "Menu".

Fonte: O autor (2025)