INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA CAMPUS CAJAZEIRAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ALGOLAB: Facilitando a Aprendizagem de Algoritmos e Lógica de Forma Interativa

ANDRÉ LUCAS TAVARES DANTAS

Cajazeiras
2025

ANDRÉ LUCAS TAVARES DANTAS

AlgoLab: Facilitando a Aprendizagem de Algoritmos e Lógica de Forma Interativa

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras, como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador

Prof. Me. Francisco Paulo de Freitas Neto

Cajazeiras

2025

IFPB / Campus Cajazeiras
Coordenação de Biblioteca
Biblioteca Prof. Ribamar da Silva
Catalogação na fonte: Cícero Luciano Félix CRB-15/750

D192a Dantas, André Lucas Tavares.

Algolab : facilitando a aprendizagem de algoritmos e lógica de forma interativa / André Lucas Tavares Dantas. – 2025.
55f. : il.

Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Cajazeiras, 2025.

Orientador: Prof. Me. Francisco Paulo de Freitas Neto.

1. Engenharia de software. 2. Desenvolvimento de software. 3. Gameficação. 4. Ensino-aprendizagem. 5. Lógica de programação. I. Instituto Federal de Educação, Ciência e Tecnologia da Paraíba. II. Título.

IFPB/CZ CDU: 004.42:37(043.2)

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

ANDRÉ LUCAS TAVARES DANTAS

ALGOLAB: Facilitando a Aprendizagem de Algoritmos e Lógica de Forma Interativa

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras, como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador

Prof. Me. Francisco Paulo de Freitas Neto

Aprovada em: **10 de setembro de 2025.**

Prof. Me. Francisco Paulo de Freitas Neto - Orientador

Prof. Me. Afonso Serafim Jacinto - Avaliador

IFPB - Campus Cajazeiras

Prof. Me. Diogo Dantas Moreira - Avaliador

IFPB - Campus Cajazeiras

Documento assinado eletronicamente por:

- **Francisco Paulo de Freitas Neto**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/09/2025 16:39:05.
- **Afonso Serafim Jacinto**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 11/09/2025 09:52:00.
- **Diogo Dantas Moreira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 11/09/2025 10:26:58.

Este documento foi emitido pelo SUAP em 10/09/2025. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código 764418
Verificador: 16332cee60
Código de Autenticação:

RESUMO

Este trabalho propõe o desenvolvimento de um jogo educativo, controlado apenas pelo mouse, cujo objetivo é ensinar conceitos fundamentais de algoritmos e lógica de programação por meio de mecânicas visuais e desafios progressivos. A jogabilidade combina objetos abstratos (contêineres, transportadores, operadores, roteadores e ativadores) para que o usuário construa soluções práticas e receba feedback em tempo real, o que favorece a fixação conceitual e o raciocínio lógico. O desenvolvimento segue uma metodologia ágil e foi implementado na engine Godot, escolhida pela flexibilidade e bom suporte a jogos 2D. O design minimalista e a progressão gradual de dificuldade tornam a ferramenta acessível tanto a iniciantes quanto a usuários com maior experiência, sendo aplicável como recurso de ensino em escolas, cursos técnicos e superior, além de uso para autoaprendizado. O projeto busca integrar aprendizagem e entretenimento, oferecendo uma experiência educativa prática, escalável e passível de expansão para novas plataformas e métricas de avaliação.

Palavras-chave: algoritmos; lógica de programação; jogo educativo; Godot.

ABSTRACT

This final project presents an educational game designed to teach core algorithmic and programming concepts through a visual, puzzle-based approach controlled solely with the mouse. Players solve progressively challenging tasks by arranging and connecting abstract game elements (containers, transporters, operators, routers and activators), receiving immediate feedback that reinforces problem-solving and logical thinking. The development followed an agile workflow and was implemented with the Godot engine for its 2D capabilities and flexibility. A minimalist UI and staged difficulty progression make the game suitable for beginners and more advanced learners alike, with potential use in classrooms, technical courses, higher education and for self-study. The project aims to blend learning and play into a scalable educational tool, ready for expansion with performance metrics and multi-platform releases.

Keywords: algorithms; programming; educational game; Godot.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

2D – Ambiente composto por apenas duas dimenções
CD – Continuous Deployment (Desenvolvimento contínuo)
CI – Continuous Integration (Integração contínua)
C# – Linguagem de programação criada pela Microsoft
GDScript – Linguagem de programação utilizada no GODOT
GIT – Sistema de controle de versão de código
GITHUB – Plataforma de hospedagem de código baseada em Git
GODOT – Godot Engine (engine de desenvolvimento de jogos)
IEEE – Institute of Electrical and Electronics Engineers
IFPB – Instituto Federal da Paraíba
OBSIDIAN – Software de organização e anotações
RF – Requisitos Funcionais
RNF – Requisitos Não Funcionais
RPG – Role-Playing Game (jogo de interpretação de papéis)
SOLID – Princípios de design orientado a objetos (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion)
SRP – Single Responsibility Principle (Princípio da Responsabilidade Única)
TCC – Trabalho de Conclusão de Curso
UI – User Interface (Interface do Usuário)

SUMÁRIO

1 INTRODUÇÃO

Os jogos eletrônicos configuram-se como uma das formas de entretenimento mais valorizadas por diferentes faixas etárias. No Brasil, 82,8% da população afirma considerar os jogos digitais como uma de suas principais formas de lazer (GO GAMERS; SX GROUP; BLEND NEW RESEARCH; ESPM, 2025). Os videogames, assim como outras mídias, abrangem uma ampla gama de gêneros, incluindo jogos de ação e RPG(Role-Playing Game), jogos de quebra-cabeças e jogos de corrida, atendendo a diferentes perfis de jogadores. No entanto, além do entretenimento, jogos também ensinam: em aplicativos como *Duolingo*, o objetivo é transmitir diretamente conhecimentos de idiomas, enquanto em experiências indiretas, como *SHENZHEN I/O* — que simula circuitos eletrônicos — o próprio ato de jogar desenvolve noções de lógica, música, mecânicas e resolução de problemas.

O presente trabalho de conclusão de curso se dedica ao desenvolvimento de um jogo educativo, com o objetivo de ensinar conceitos fundamentais de algoritmos e programação de maneira prática e envolvente. Em um cenário cada vez mais digital e tecnológico, a demanda por habilidades de programação e lógica computacional é crescente. A programação proporciona um ambiente ideal para a experimentação e a exploração de ideias, permitindo que os alunos desenvolvam soluções personalizadas para problemas reais ou fictícios. Essa abordagem também os incentiva a pensar de maneira inovadora e a buscar novas maneiras de enfrentar desafios.

A análise curricular dessas disciplinas pode revelar que muitos estudantes ainda enfrentam dificuldades significativas na compreensão e aplicação dos conceitos iniciais de programação. Isso pode ser devido a diversos fatores, como a complexidade dos conteúdos, a metodologia de ensino utilizada, ou a falta de recursos adequados (Silva, E.P., Caceffo, R., & Azevedo, R. ,2023). Este projeto propõe-se a criar uma ferramenta interativa que, utilizando desafios de quebra-cabeça e mecânicas de jogo atraentes, visa tornar o aprendizado da

criação de algoritmos mais acessível e motivador.

1.1 Problema e Justificativa

O ensino de algoritmos e lógica de programação é uma etapa crítica na formação de qualquer profissional de tecnologia e, como observado por Sanches, Pontes e Rodrigues (2022), disciplinas introdutórias como Algoritmos e Programação e Álgebra Linear podem representar obstáculos significativos para estudantes sem experiência prévia, levando muitos iniciantes a desistirem ao se depararem com essas dificuldades. Estudos demonstram que uma das principais causas da alta taxa de desistência está relacionada à falta de familiaridade e habilidades dos alunos com a disciplina de algoritmos, cujo objetivo é desenvolver o pensamento lógico necessário para solucionar problemas em diversas áreas do conhecimento humano (Giraffa, M., & Mora, M. da costa. , 2016). Esta dificuldade é um entrave significativo, pois a compreensão de algoritmos é fundamental para o desenvolvimento de software eficiente e a resolução de problemas complexos. Assim, há uma necessidade clara de métodos de ensino que possam traduzir esses conceitos de forma mais prática e interativa. Segundo este artigo feito por professores que testaram o uso de educação gamificada:

> No geral, os resultados sugerem que a gamificação baseada em desafios melhora os resultados de aprendizagem em um curso de previsão. Esse tipo de gamificação apresenta a maior melhoria no desempenho dos alunos quando combinado com métodos de ensino tradicionais. No entanto, nossos resultados mostram que mesmo essa aplicação gamificada sozinha, integrada a uma palestra, pode ter um impacto positivo nos resultados de aprendizagem. (Nikoletta-Zampeta Legaki, Nannan Xi, Juho Hamari, Kostas Karpouzis, Vassilios Assimakopoulos. (2020, p. 10).)

Este estudo demonstrou que essa abordagem teve um impacto mais significativo em alunos de Engenharia Elétrica e de Computação. Isso reforça ainda mais a relevância de utilizar um jogo como ferramenta para ensinar esses conceitos, já que é uma forma eficaz de engajar e instruir esse público-alvo.

1.2 Proposta do Software

O software a ser desenvolvido é um jogo de quebra-cabeças, controlado

exclusivamente pelo mouse. O design visual será minimalista, mas futurista, a fim de criar uma atmosfera imersiva que mantenha o jogador focado nas tarefas propostas. A simplicidade do cenário permitirá que a atenção seja direcionada para as mecânicas de jogo, que são projetadas para ensinar conceitos fundamentais de algoritmos de forma intuitiva e prática.

O jogo utilizará uma série de mecânicas interativas, com as quais os jogadores devem resolver problemas utilizando elementos como *containers*, números, operações e *links*. Esses elementos representam abstrações de conceitos de algoritmos e estruturas de dados, proporcionando ao jogador uma compreensão visual e prática desses tópicos. Cada nível é projetado para introduzir um novo conceito, começando com problemas básicos como utilização de variáveis (*containers*) e progredindo para desafios mais complexos, como algoritmos de ordenação ou listas encadeadas. Essa escalada na dificuldade é intencional, permitindo que o jogador construa conhecimento de forma incremental, solidificando sua compreensão a cada etapa.

Embora o jogo atualmente não possua uma narrativa explícita, sua interface minimalista e os desafios propostos já fornecem o contexto necessário para a experiência do jogador. Caso o projeto venha a ser expandido futuramente, a adição de uma narrativa integrada poderia ser uma estratégia interessante para aumentar o engajamento e a imersão dos usuários, tornando a jornada de aprendizado ainda mais envolvente. No estágio atual, o foco permanece nas mecânicas de resolução de problemas, que constituem o núcleo da proposta educativa, transformando o estudo de algoritmos em uma atividade acessível e interativa.

A gamificação e o aprendizado baseado em jogos são abordagens reconhecidas por sua eficácia em tornar o aprendizado de disciplinas complexas, como algoritmos e programação, mais acessível e motivador. (Nikoletta-Zampeta Legaki, Nannan Xi, Juho Hamari, Kostas Karpouzis, Vassilios Assimakopoulos. , 2020). Ao integrar elementos lúdicos com desafios educacionais, o jogo oferece uma experiência que não só educa, mas também entretém. Essa abordagem é

particularmente valiosa em contextos em que o aprendizado tradicional pode ser percebido como tedioso ou difícil.

O jogo foi projetado para ser acessível a uma ampla gama de usuários, desde estudantes de ensino médio, que estão começando a explorar o mundo da programação, até universitários e profissionais que desejam reforçar suas habilidades de lógica e algoritmos. O jogo oferecerá um espaço seguro e estimulante para o aprendizado. Com sua estrutura modular, o jogo permitirá que os usuários aprendam no seu próprio ritmo, revisitem conceitos conforme necessário, e se desafiem à medida que avançam. Além disso, a escolha de uma interface intuitiva e o controle simplificado pelo mouse garantem que o jogo seja acessível, independentemente do nível de experiência do usuário com jogos ou tecnologia. Essa acessibilidade é crucial para atingir um público diversificado, incluindo aqueles que podem não ter um interesse inicial em programação, mas que são atraídos pela natureza envolvente e lúdica do jogo.

Em resumo, o jogo a ser desenvolvido não apenas ensinará algoritmos de forma prática e progressiva, mas também criará uma experiência educativa diferenciada, onde o aprendizado é integrado de maneira fluida ao entretenimento. Essa combinação de ensino e diversão é o que diferencia o software, tornando-o uma ferramenta poderosa tanto para a educação formal quanto para o autoaprendizado.

1.3 Potencialidade de Comercialização e Áreas de Aplicação

O jogo educativo desenvolvido tem um potencial claro de aplicação em ambientes educacionais e de treinamento. Pode ser utilizado em escolas, universidades e cursos de programação como uma ferramenta complementar para o ensino de algoritmos. Além disso, pode ser comercializado como um produto para autoaprendizado, voltado para estudantes e entusiastas de programação. Sua simplicidade e acessibilidade tornam o jogo uma opção atrativa para diferentes públicos, oferecendo uma forma prática e lúdica de aprender conceitos complexos.

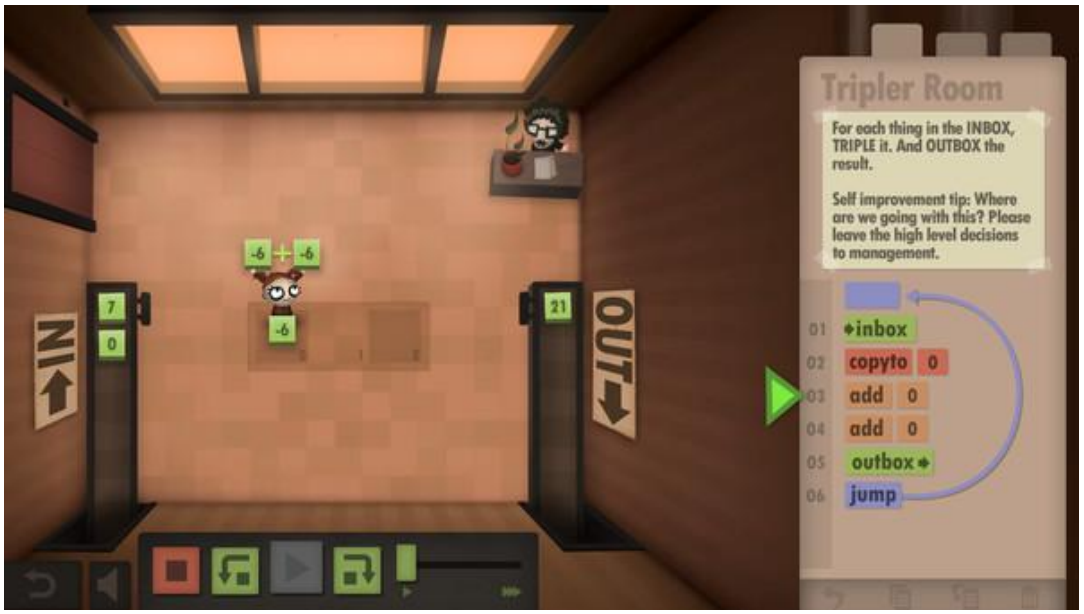
1.5 Referências e Inspirações

Durante o desenvolvimento deste projeto, foram utilizados como referência diversos jogos que apresentam mecânicas e abordagens similares à proposta do jogo proposto neste trabalho, tanto no aspecto educacional quanto no design e interatividade. Esses jogos serviram como referência para a elaboração de requisitos e o desenvolvimento de mecânicas funcionais. Abaixo, são destacados dois jogos que tiveram um papel importante na formulação das ideias para este projeto.

- ***Human Resource Machine***[1] é um jogo de quebra-cabeça baseado em programação desenvolvido pela Tomorrow Corporation, que foi uma das principais inspirações. Neste jogo, o jogador deve resolver problemas de processamento de dados por meio da criação de algoritmos visuais semelhantes a algoritmos. A maneira como ele utiliza uma interface simples para ensinar conceitos complexos de lógica e programação foi fundamental para a concepção das mecânicas do meu jogo, onde o objetivo é que o jogador manipule números e dados por meio de operações para solucionar problemas. A progressão gradual de dificuldade e a forma acessível como conceitos de programação são introduzidos e praticados também influenciaram a estrutura dos meus níveis, que buscam um equilíbrio entre ensino e desafio. A interface do jogo é mostrada na figura a seguir.

[1] Tomorrow Corporation : Human Resource Machine

Figura 1 – Inspiração Human Resource Machine

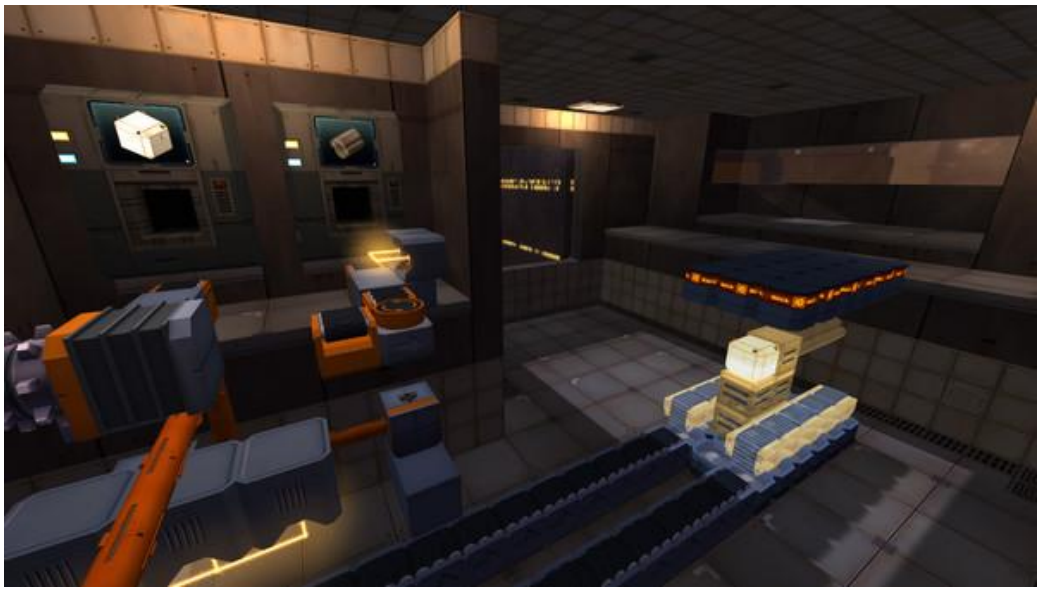


Fonte: steampowered.com

- ***Infinifactory***[2] é outro título que influenciou o design. Desenvolvido pela *Zachtronics.* Nele, o jogador constrói fábricas automatizadas para processar e montar produtos em um ambiente 3D, utilizando lógica e design industrial. O aspecto que mais inspirou o meu projeto foi o sistema de organização e o fluxo de processos. Assim como em **Infinifactory**, meu jogo envolve a criação de soluções lógicas para movimentar números e aplicar operações matemáticas, com foco na otimização e fluidez. A liberdade dada ao jogador para criar e organizar sua solução, junto com a variedade de desafios lógicos apresentados, também serviu como modelo para o meu design de nível. A interface do jogo é mostrada na figura seguinte.

[2] Zachtronics | Infinifactory

Figura 2 – Inspiração Infinifactory



Fonte: steampowered.com

Essas referências foram fundamentais para definir o estilo e a jogabilidade deste projeto, ao mesmo tempo que permitiram a criação de um ambiente único, adaptado às necessidades de aprendizagem de algoritmos.

1.4 Organização do trabalho

Este trabalho está estruturado para proporcionar uma leitura progressiva e compreensível, composta por quatro capítulos principais. Começamos com uma **Introdução**. Em seguida, detalhamos a **Metodologia** de Desenvolvimento, explicando as etapas e as ferramentas utilizadas. No capítulo de **Descrição do Sistema**, abordamos a arquitetura, as tecnologias usadas e as mecânicas de jogo do software desenvolvido. Por fim, falamos sobre as potenciais aplicações do projeto e como ele pode ser vendido nas **Considerações Finais**.

2 METODOLOGIA

Este capítulo apresenta as práticas e ferramentas adotadas para o desenvolvimento do jogo educativo, destacando o levantamento de requisitos e o fluxo de trabalho utilizado durante a implementação.

2.1 Levantamento de requisitos

O levantamento de requisitos para o desenvolvimento do jogo educativo foi realizado de maneira iterativa, levando em consideração as necessidades dos usuários-alvo e os objetivos pedagógicos do projeto.

Na engenharia de software, **requisitos** são descrições das funções, restrições e atributos de qualidade que um sistema deve possuir, servindo como base para orientar seu desenvolvimento e garantir que o produto final atenda às necessidades identificadas. Segundo a *IEEE Std 830-1998 – IEEE Recommended Practice for Software Requirements Specifications*, os requisitos podem ser classificados em duas categorias principais:

- **Requisitos Funcionais (RF)**: especificam **o que** o sistema deve fazer, descrevendo funcionalidades, comportamentos e serviços que ele deve oferecer.
- **Requisitos Não Funcionais (RNF)**: especificam **como** o sistema deve se comportar, abrangendo restrições e atributos de qualidade como desempenho, usabilidade, segurança, compatibilidade e manutenibilidade.

A identificação dos requisitos envolveu:

1. **Análise de Necessidades Educativas:** Para começar, foi realizada uma pesquisa de opinião para determinar os conceitos de algoritmos mais difíceis para os alunos e que o jogo deveria abordar. Consultas a materiais acadêmicos, como livros e artigos sobre ensino de algoritmos, além de revisões de jogos educativos semelhantes disponíveis no mercado, foram incluídos neste

levantamento.

2. **Definição de Objetivos Pedagógicos:** Os objetivos pedagógicos do jogo foram estabelecidos com base na análise de necessidades. Eles definiram quais conceitos de algoritmos seriam ensinados e como esses conceitos poderiam ser incorporados nas mecânicas do jogo.

Com base no levantamento de requisitos, foram elaborados quadros que organizam os principais requisitos funcionais e não funcionais do jogo, assim como as funcionalidades associadas que serão apresentadas na próxima seção. Com o Quadro 1 apresentando os requisitos funcionais, ajudando a visualizar as características essenciais do projeto e orientando o processo de desenvolvimento.

Quadro 1 - Requisitos Funcionais

Requisito	Objetivo	Descrição
RF01	*O jogo deve ensinar conceitos básicos de algoritmos*	Implementação de níveis iniciais focados em variáveis, operadores e estruturas de controle.
RF02	Deve haver progressão de dificuldade	Níveis com complexidade crescente, introduzindo novos conceitos de algoritmos a cada etapa.
RF03	Criação Simplificada de Objetos	O jogador deve ser capaz de criar e manipular objetos no jogo de forma intuitiva, com mínima complexidade.
RF04	Visualização da execução	O jogo deve mostrar a solução criada pelo jogador trabalhando em tempo real.

Requisito	Objetivo	Descrição
RF05	Sistema de dicas nos níveis	Mostrar dicas caso o usuário esteja com dificuldades de resolver o nível.

Fonte: Elaborado pelo autor.

E, a seguir, no Quadro 2, estão os requisitos não funcionais, garantindo que todas as demandas sejam atendidas de forma organizada e eficiente, tendo em vista o público-alvo do projeto.

Quadro 2 - Requisitos Não Funcionais

Requisito	Objetivo	Descrição
RNF 01	O jogo deve ser possível jogar apenas com o mouse	Design de interface e mecânicas interativas simples, como cliques e arrastar e soltar (*drag and drop*).
RNF 02	O jogo deve ser acessível a iniciantes	Interface de usuário intuitiva, com tutoriais e dicas ao longo do jogo.

Fonte: Elaborado pelo autor.

2.1.1 Descrição dos requisitos

A seguir, será apresentada uma explicação detalhada de alguns dos requisitos mais relevantes, abordando tanto os requisitos funcionais, que descrevem o comportamento e as funcionalidades do sistema, quanto os não funcionais, que tratam de aspectos como desempenho, usabilidade e eficiência.

Requisitos Funcionais:

1. **RF01: O jogo deve ensinar conceitos básicos de algoritmos.**
    - Descrição: Implementação de níveis iniciais focados em variáveis, operadores e estruturas de controle.

- Objetivo: Facilitar o aprendizado dos conceitos fundamentais de algoritmos para iniciantes.
- Critérios de Aceitação: Níveis iniciais devem incluir tutoriais e exemplos práticos que expliquem variáveis, operadores e estruturas de controle.

2. **RF02: Deve haver progressão de dificuldade.**
    - Descrição: Níveis com complexidade crescente, introduzindo novos conceitos de algoritmos.
    - Objetivo: Desenvolver habilidades dos jogadores de forma gradual e contínua.
    - Critérios de Aceitação: Cada nível subsequente deve ser mais desafiador que o anterior, introduzindo novos conceitos e mecânicas. (Níveis que introduzem uma nova mecânica podem ser um pouco mais fáceis.)
3. **RF03: Criação Simplificada de Objetos.**
    - Descrição: O jogador deve ser capaz de criar e manipular objetos no jogo de forma intuitiva, com mínima complexidade.
    - Objetivo: Tornar a interação com o jogo fácil e acessível, mesmo para iniciantes.
    - Critérios de Aceitação: A interface deve permitir a criação e manipulação de objetos com poucos cliques e arrastar e soltar (*drag and drop*).

Requisitos Não Funcionais:

1. **RNF 01: Jogabilidade com o Mouse.**
    - Descrição: O jogo deve ser jogável apenas com o mouse, utilizando cliques e arrastar e soltar (*drag and drop*).
    - Objetivo: Garantir uma interface intuitiva e acessível, facilitando o uso por iniciantes e jogadores com pouca experiência em jogos.
    - Métricas: Feedback dos jogadores de teste sobre o uso do mouse.
2. **RNF 02: Acessibilidade para Iniciantes.**
    - Descrição: O jogo deve ser acessível a iniciantes, com tutoriais e

dicas ao longo do jogo.
    - Objetivo: Facilitar o aprendizado e a adaptação dos jogadores, aumentando a retenção e o engajamento.
    - Métricas: Aprovação por jogadores de teste iniciantes.

Esses requisitos foram documentados no OBSIDIAN[3] e versionados como GitHub Issues[4], permitindo rastreabilidade e refinamento à medida que o desenvolvimento avançava.

2.2 Ferramentas e Fluxo de Trabalho

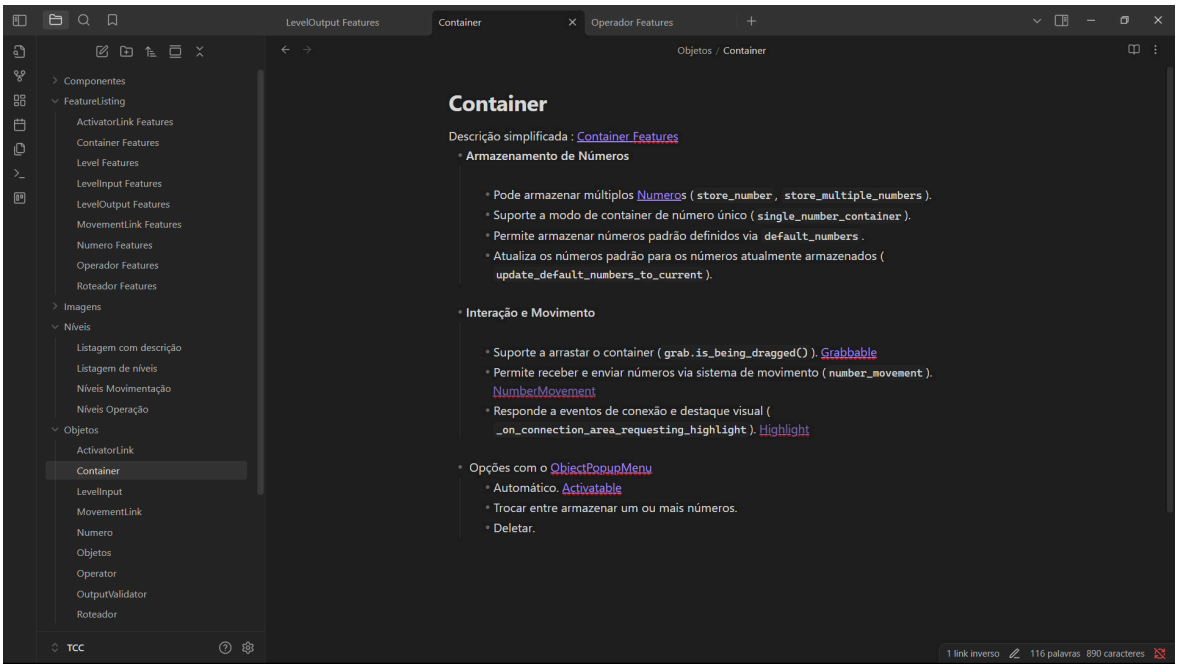
Nesta seção, detalha-se o papel de cada ferramenta adotada para organizar, documentar e acompanhar o desenvolvimento.

2.2.1 Obsidian

- **Notas Markdown**: criação de arquivos para cada objeto/recurso do jogo, por meio de uma interface simples, como visto na Figura 3 abaixo, permitindo:
    - Anotações estruturadas com títulos, listas, etc.
    - Controle de versões locais via Git.
    - Referências a outras notas ou partes do texto.

[3] OBSIDIAN, 2025 - Disponível em: Obsidian - Sharpen your thinking Acessado em: 04/07/2025.
[4] GITHUB ISSUES, 2025 - Disponível em: GitHub Issues · Project planning for developers Acessado em 04/07/2025.

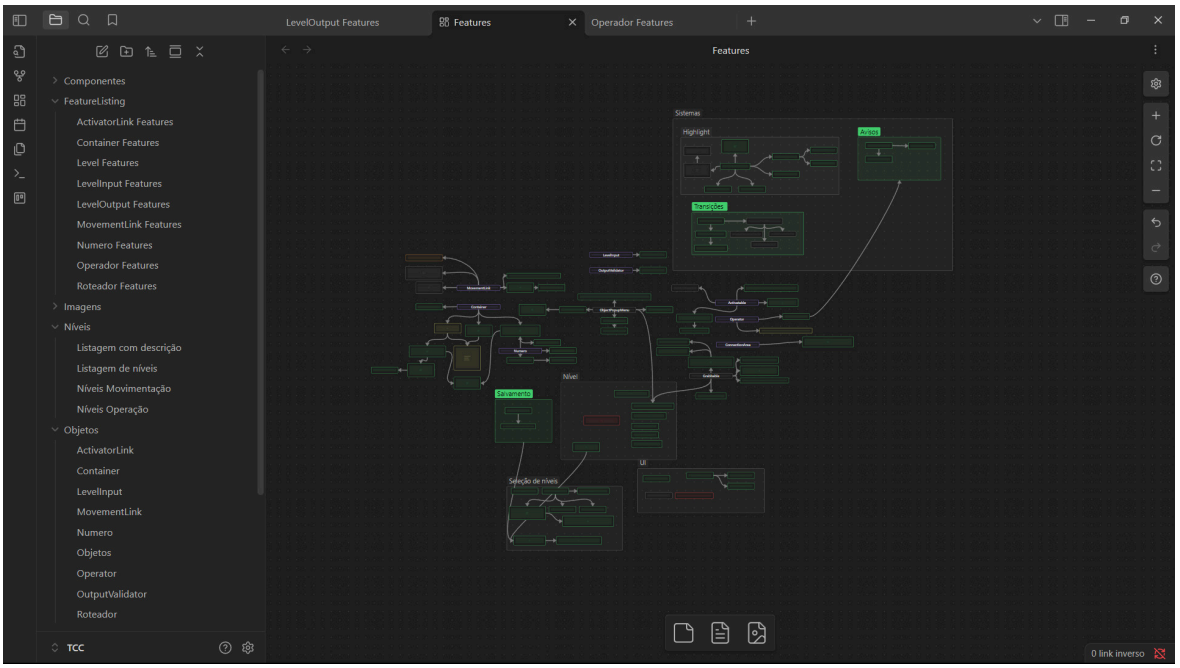
Figura 3 – Interface do editor de notas do Obsidian



Fonte: Autor.

- **Canvas**: quadro visual integrado às notas. A Figura 4 mostra a interface do Canvas do Obsidian, este modo possibilita:
    - Dispor notas livremente, conforme a lógica de implementação.
    - Destacar prioridades com cores (vermelho para crítico, verde para concluído, amarelo/laranja para adiável).
    - Conectar notas relacionadas por setas, ilustrando dependências e fluxos.
    - Agrupar notas em áreas específicas para features isoladas ou especiais.

Figura 4 – Interface de canvas do Obsidian.



Fonte: Autor.

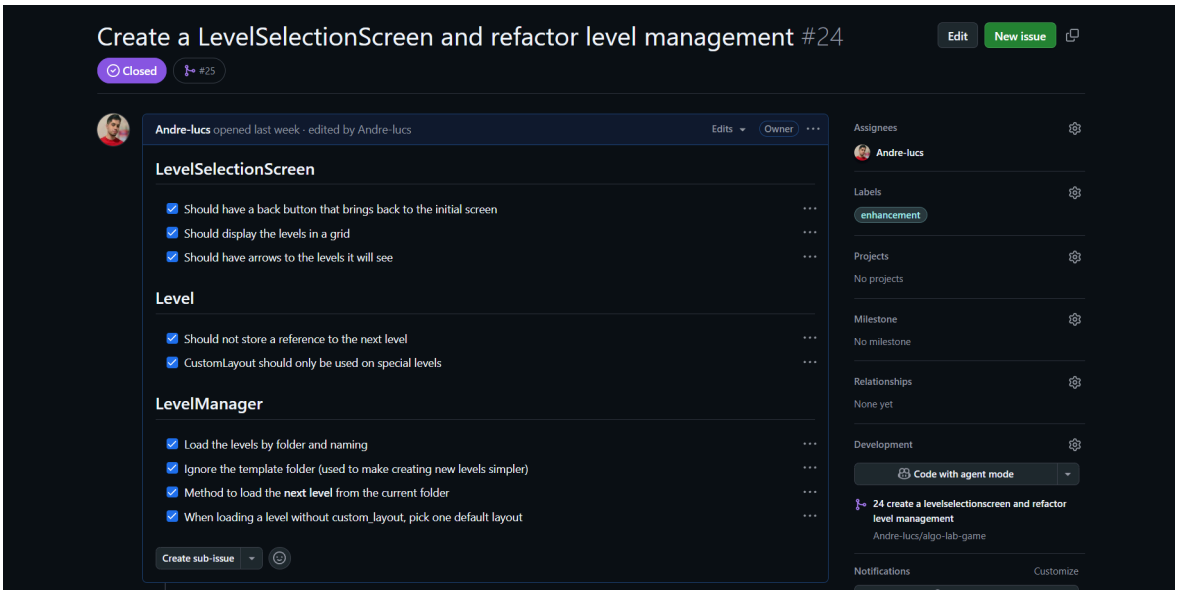
- **Linking & Backlinks**: rastreabilidade bidirecional entre notas, facilitando a navegação entre conceitos e requisitos.
- **Plugins Complementares**: uso de extensões para pré-visualização de tabelas, diagramas e checklists diretamente nas notas.

2.2.2 GitHub Issues

- **Registro de Tarefas**: cada requisito ou *bug* foi criado como uma *issue*, contendo:
    - Título descritivo e corpo com *checkboxes* para etapas ou critérios de aceitação.
    - Labels para categorizar ("*feature*", "*bug*", "documentação", "teste").
    - Assignees e milestones para definir responsabilidades e prazos.
- **Discussões e Atualizações**:
    - Comentários encadeados permitem registro de dúvidas, decisões de design e discussões de implementação.
    - Referência cruzada a *Pull Requests* e *Commits*.

- **Integrações & Automação**:
  - Status de CI/CD vinculado a merge de branches.
  - Templates de *issue* padronizam o formato de novos registros.
  - Solicitação de revisão automática de código utilizando o GitHub Copilot, que fornecia sugestões e validações para *Pull Requests*.

Figura 5 – Exemplo de issue do Github



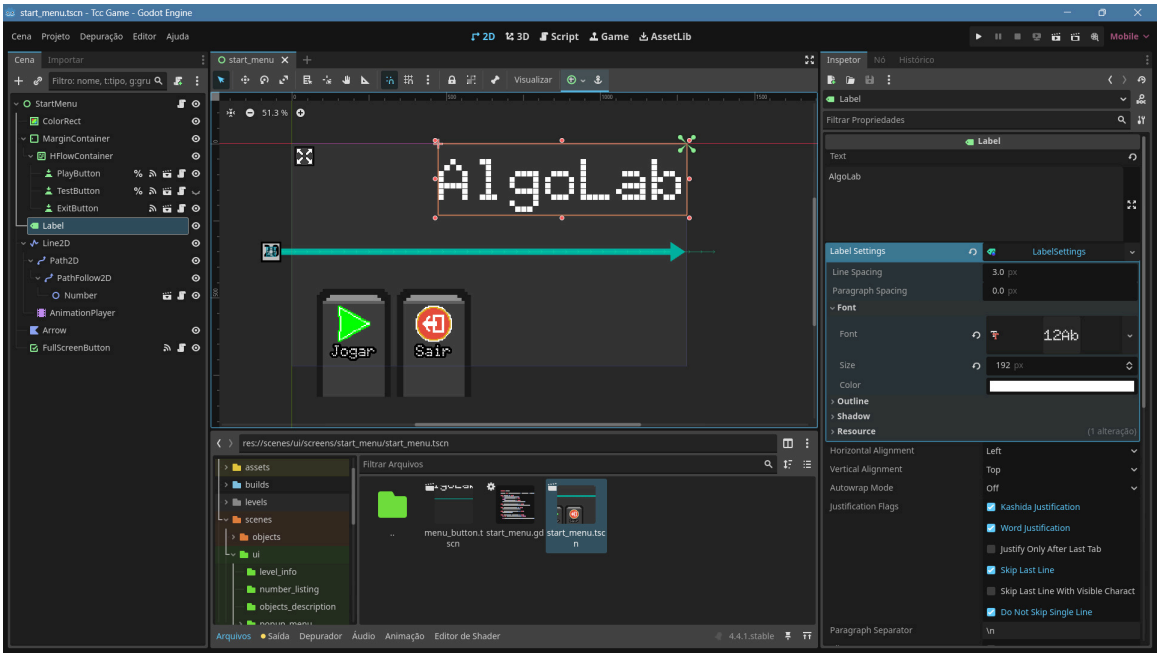
Fonte: Autor.

2.2.3 Engine de Desenvolvimento - Godot

O desenvolvimento do jogo foi realizado utilizando a engine **Godot**[5], escolhida por ser gratuita, de código aberto e altamente adequada ao desenvolvimento 2D. A Godot oferece uma interface visual integrada com sistema de nós hierárquicos e linguagem de script própria (*GDScript*), permitindo o desenvolvimento modular e reutilizável de componentes do jogo. Além disso, sua comunidade ativa e boa documentação facilitaram a implementação e os testes das mecânicas planejadas.

[5] GODOT, 2025 - Disponível em: Godot Engine - Free and open source 2D and 3D game engine Acessado em 04/07/2025.

Figura 6 – Interface do Godot



Fonte: Autor.

2.3 Estrutura do Processo de Desenvolvimento

As fases principais, mantidas e visualizadas no Canvas e no GitHub, foram:

1. **Levantamento de Requisitos** (conforme Seção 2.1).
2. **Design e Prototipagem**: esboço de níveis e mecânicas em notas do Obsidian, com *feedback* guiando ajustes.
3. **Desenvolvimento Incremental**: implementação via *Issues*, marcação de *checkboxes* e *pull requests* para controle de versão.
4. **Testes e Ajustes**: execução de testes baseados nos critérios de aceitação definidos nas Issues; fechamento de bugs como novas Issues.

2.4 Vantagens da Abordagem Utilizada

- **Flexibilidade Visual**: Canvas do Obsidian sem colunas fixas.
- **Documentação Unificada**: notas e *Canvas* integrados.

- **Rastreabilidade Detalhada**: *checkboxes* em *Issues* ligam planejamento e execução.
- **Ferramentas Barebones**: soluções gratuitas e adequadas a projeto solo.

2.5 Controle de Progresso e Adaptações

- Cores e agrupamentos no Canvas para rápida identificação de prioridades.
- Vinculação entre notas e issues assegurou histórico completo.
- Exportação periódica de snapshots do Canvas e Issues como evidência no repositório.

3 DESCRIÇÃO DO SISTEMA

Neste capítulo, será detalhado o desenvolvimento técnico e funcional do jogo educativo, cujo objetivo é ensinar algoritmos de maneira interativa.

O sistema será projetado em diferentes áreas que, em conjunto, proporcionarão uma experiência de aprendizado fluida e eficaz. A proposta deste trabalho é utilizar a integração entre aspectos visuais e mecânicos, a fim de garantir uma jogabilidade imersiva, fazendo uso de tecnologias modernas e boas práticas de desenvolvimento, de forma a otimizar o processo de criação do jogo e garantir a escalabilidade do produto.

A organização deste capítulo segue a seguinte estrutura: inicialmente, será apresentada uma visão geral do jogo. Em seguida, será feita uma descrição detalhada da arquitetura do sistema, das tecnologias empregadas, das principais mecânicas de jogo, da interface do usuário e decisões de *Gamedesign*. Cada um desses elementos será explorado de forma detalhada, incluindo exemplos visuais e práticos que demonstram como o jogo alcança seus objetivos principais, ao combinar aprendizado pedagógico com elementos de entretenimento para engajar e motivar os jogadores.

3.1 Visão geral

O jogo desenvolvido tem como principal objetivo auxiliar no ensino de lógica de programação e fundamentos de algoritmos de forma prática, acessível e interativa. Por meio de desafios baseados em raciocínio lógico, o jogador manipula abstrações visuais de variáveis, operações e estruturas de controle em um cenário funcional e minimalista.

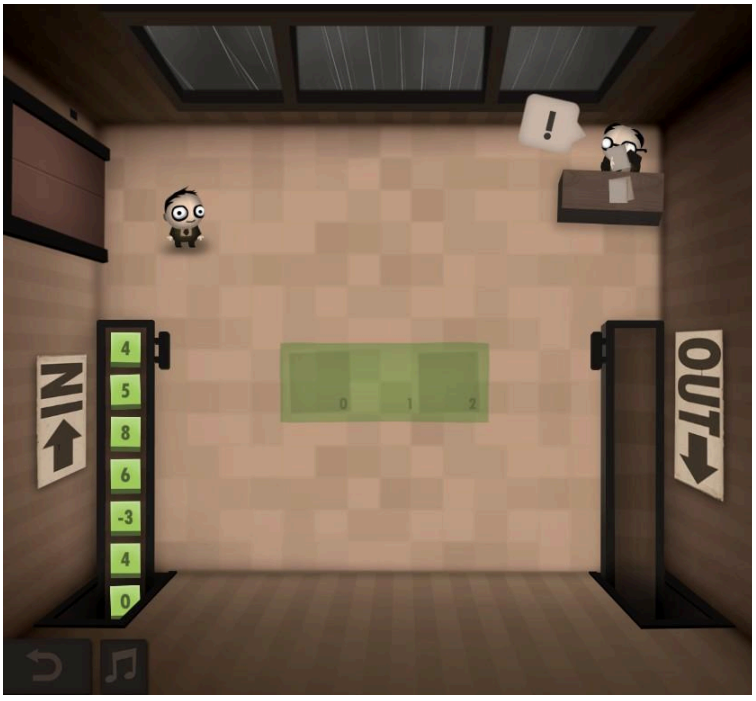
Para sua implementação, foi utilizada a engine Godot, que oferece ferramentas robustas para o desenvolvimento de jogos 2D com suporte a

scripting e modularização.

O jogo será apresentado por uma perspectiva *top-down* (de cima para baixo), frequentemente utilizada em jogos por proporcionar uma visão ampla e clara da área de interação. Essa escolha visa facilitar a organização espacial dos objetos e a visualização do fluxo de dados durante a construção das soluções.

A **Figura 7** exemplifica essa perspectiva no jogo *Human Resource Machine*, que foi uma das inspirações para este projeto. Nela, o jogador consegue observar claramente os componentes do jogo. De forma semelhante, a adoção dessa perspectiva no jogo desenvolvido neste trabalho permite que o jogador compreenda de forma mais intuitiva a lógica envolvida em cada desafio, reforçando a proposta pedagógica do sistema.

Figura 7 – Visão top-down do jogo Human Resource Machine



Fonte: steampowered.com

3.1.1 Público-Alvo

O jogo foi projetado para atender principalmente a três perfis de usuários:

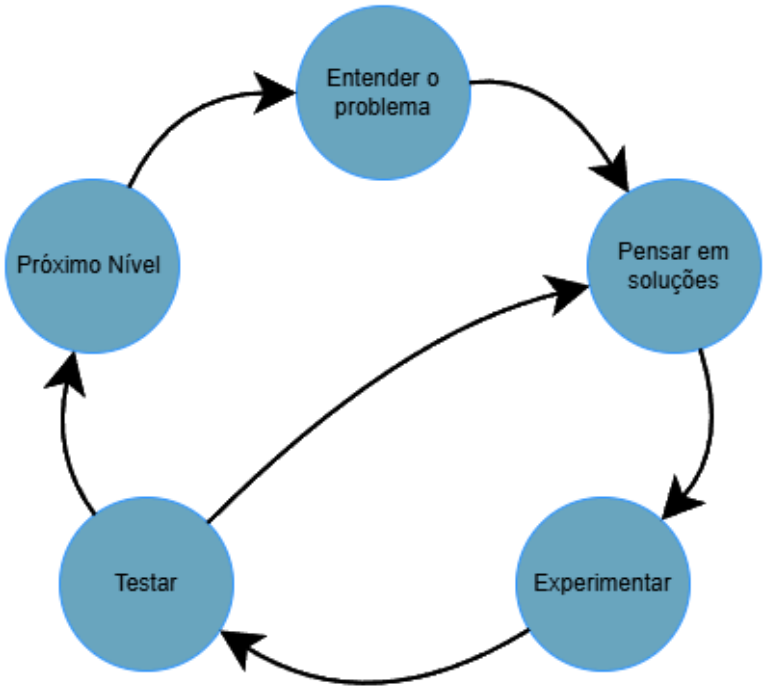
- **Estudantes do ensino médio e técnico**, que estão em processo de introdução aos conceitos de lógica e algoritmos, muitas vezes enfrentando essas temáticas pela primeira vez.
- **Acadêmicos de nível superior**, especialmente dos cursos de Análise e Desenvolvimento de Sistemas, Ciência da Computação e Engenharias, que desejam revisar ou consolidar conteúdos fundamentais de forma prática e interativa.
- **Docentes da área de computação**, que podem utilizar o jogo como ferramenta de apoio didático, complementando metodologias tradicionais e ampliando o engajamento em sala de aula.

A proposta de acessibilidade — tanto na simplicidade da interface quanto na jogabilidade baseada em mouse — visa contemplar diferentes níveis de familiaridade com a tecnologia, tornando o jogo atrativo para iniciantes. Além disso, a estrutura modular permite ao usuário avançar no próprio ritmo, o que favorece tanto o **autoaprendizado** quanto **o uso em contextos formais de ensino**. Essa abordagem pedagógica inclusiva é especialmente pertinente diante das dificuldades frequentemente observadas na introdução ao ensino de algoritmos (cf. Seções 1.2 e 1.3).

3.1.2 Gameplay loop (Ciclo de jogabilidade)

Esse processo de enfrentamento de desafios e obtenção de *feedback* constitui o que se define como *gameplay loop*, ou ciclo de jogabilidade — uma sequência recorrente de ações que estrutura a experiência interativa do jogo. No contexto deste projeto, o ciclo é composto por etapas como a compreensão do problema proposto, o planejamento e montagem da solução, a experimentação e o teste, culminando na progressão para um novo desafio. A Figura 8 ilustra esse ciclo de forma esquemática, demonstrando como ele sustenta tanto o avanço pedagógico quanto o engajamento do jogador ao longo da experiência.

Figura 8 – Diagrama Loop de Gameplay



Fonte: Elaborado pelo autor

Conforme representado na Figura 8, o ciclo de jogabilidade pode ser resumido nas seguintes etapas:

1. **Entender o problema:** Nesta etapa, o jogador é introduzido ao desafio e deve compreender o que precisa ser resolvido.
2. **Pensar em soluções:** Já nessa etapa, o jogador formula possíveis estratégias para resolver o problema com as mecânicas do jogo.
3. **Experimentar:** O jogador coloca suas soluções em prática, testando-as dentro do jogo.
4. **Testar:** Caso a solução seja correta, o jogador avança para o próximo nível; se não, ele recebe um alerta mostrando qual problema teve, fazendo-o ajustar e refinar sua abordagem.
5. **Próximo nível:** Ao completar o desafio, o jogador é levado para o próximo nível, onde novos conceitos e dificuldades são

apresentados.

A progressão de dificuldade foi projetada para acompanhar o aprendizado gradual do jogador, introduzindo novos conceitos de maneira incremental. Além dos níveis principais, o jogo oferece também desafios opcionais com maior complexidade. Esses níveis, embora não obrigatórios para a conclusão do jogo, funcionam como caminhos alternativos para jogadores que busquem desafios mais complexos. Esta estrutura visa estimular a experimentação, a aplicação de conhecimentos prévios e o engajamento contínuo, oferecendo uma experiência mais personalizada de acordo com o ritmo e interesse do jogador.

Esse ciclo é repetido ao longo dos níveis, reforçando a aprendizagem por meio da prática contínua e permitindo ao jogador desenvolver fluência na aplicação de conceitos algorítmicos.

3.2 Mecânicas de jogo

O jogo é baseado em um conjunto de mecânicas de quebra-cabeça inspiradas em conceitos de algoritmos, divididas em ações que o jogador pode realizar e funções que os objetos desempenham para ajudar na resolução dos desafios. Essas mecânicas permitem ao jogador construir, manipular e interligar objetos, aplicando conceitos de algoritmos e lógica.

3.2.1 Mecânicas do Jogador

- **Criação de Objetos**: Através da interface de usuário, o jogador pode criar diferentes tipos de objetos que serão usados na solução dos desafios. Cada objeto possui uma função específica, abstraindo conceitos como variáveis, condicionais e operações matemáticas.
- **Manipulação de Objetos**: Ao clicar com o botão esquerdo do mouse, o jogador pode segurar e mover objetos pelo cenário, organizando sua solução de acordo com a lógica desejada. Isso permite ao jogador reorganizar sua estratégia enquanto resolve o desafio.

- **Configuração de Objetos:** Clicando com o botão direito sobre um objeto, o jogador pode mudar opções para customizar como o objeto deve se comportar no nível.
- **Conexão entre Objetos:** O jogador pode criar dois objetos que servem para criar uma conexão entre dois objetos, essas conexões servem para fazer os objetos interagirem entre si, sendo eles o Transportador e o Ativador.
- **Etiquetas**: O jogador pode atribuir etiquetas a objetos, dando nomes personalizados a eles. Isso melhora a organização e a clareza da solução, especialmente em quebra-cabeças mais complicados.

3.2.2 Sistema de ativação de Objetos

Grande parte dos objetos presentes no jogo compartilha um sistema comum de ativação, desenvolvido para permitir ao jogador total controle sobre como e quando cada objeto executa suas ações. Esse mecanismo é fundamental para assegurar previsibilidade e coerência nas soluções dos desafios propostos.

A ativação dos objetos ocorre por meio de duas abordagens principais:

1. **Seleção do modo de ativação**: Cada objeto ativável possui um botão de controle acessado no menu dele.
   Esse botão exibe o modo de ativação atual e permite alterná-lo entre as seguintes opções:

- **Manual**: O objeto só será ativado quando receber um sinal externo.
- **Uma vez**: Semelhante ao Manual, porém ele é ativado uma vez ao iniciar a execução do nível.
- **Automático**: O objeto será ativado automaticamente, podendo alternar entre o estado Ligado e Desligado.
- **Automático (Desligado)**: Igual ao modo automático, porém inicialmente em estado Desligado.

Essa flexibilidade permite que o jogador adapte a lógica de ativação conforme o contexto do nível e a estratégia adotada.

2. **Ativação por Ativador**: Outra forma de ativar um objeto é por meio da ligação entre dois componentes utilizando um *Ativador*. Quando o primeiro objeto emite um sinal visual (uma luz vermelha), um pulso é transmitido ao segundo objeto conectado, acionando sua ativação.
   O efeito dessa ativação depende do modo configurado no objeto de destino:

- Nos modos **Manuais** (*Manual* e *Uma vez*), o sinal simplesmente executa a ação prevista do objeto.
- Nos modos **Automáticos** (*Automático* e *Automático (Desligado)*), o sinal faz com que o objeto alterne entre os estados “ligado” e “desligado”.

Esse sistema de ativação modular e configurável é essencial para a resolução de problemas mais complexos, promovendo a construção de soluções lógicas e otimizadas por parte do jogador.

3.3 Objetos e Suas Funções

Nesta seção serão apresentados os objetos do jogo, suas funcionalidades e designs.

Transportador:

- **Função:** Conecta dois objetos e transporta números entre eles.
- **Ação principal (Ativação):** Move um número de um objeto de origem para o objeto de destino.
- **Evento disparador:** Acontece no momento em que o número começa a se mover.
- **Opções específicas:**
  - **Modo de transporte:** Define o comportamento da transferência. Pode ser:
    - **Mover instância:** Transfere o número original (remove do objeto de origem).

        - **Criar cópia:** Gera uma duplicata do número original no objeto de destino.

Contêiner:

- **Função:** Armazena números. Pode ser conectado a **transportadores** nos lados esquerdo e/ou direito, determinando a direção da interação.
- **Ação principal (Ativação):** Não possui reação direta à ativação externa.
- **Evento disparador:** Emitido quando o contêiner fica **vazio**.
- **Opções específicas:**
    - **Modo único:** Restringe o contêiner a armazenar apenas um número por vez.
    - **Exibição dos valores armazenados:**
        - Quando o contêiner armazena múltiplos números, **somente os valores visíveis nas saídas laterais são mostrados por padrão**.
        - Ao posicionar o cursor do mouse sobre o contêiner, uma **lista com todos os números armazenados** é exibida como tooltip, permitindo inspeção completa do conteúdo.

Roteador:

- **Função:** Atua como intermediário entre transportadores. Redireciona números recebidos para diferentes saídas, organizando o fluxo de dados.
- **Ação principal (Ativação):** Move um número recebido para um dos transportadores conectados, alternando ciclicamente entre eles a cada ativação, caso envie para um transportador no modo cópia, ele envia uma cópia e continua com o número.
- **Evento disparador:** Ocorre no momento em que um número é enviado por uma de suas saídas.

Ativador:

- **Função:** Conecta dois objetos. Quando o **objeto de origem** emite um **evento disparador**, o Ativador envia um **sinal de ativação** ao **objeto de destino**.
- **Ação principal (Ativação):** Não reage diretamente à ativação externa.
- **Evento disparador:** Não emite eventos disparadores próprios.
- **Observações:**
    - O Ativador é um componente **passivo**, utilizado para **encadear ações** entre objetos com base em eventos.
    - Como dito na Seção 3.2.2. O comportamento do objeto de destino dependerá do **modo de ativação** que ele estiver configurado (ex: Manual, Automático, etc.).

Operador:

- **Função:** Recebe dois números como entrada e realiza uma operação matemática para gerar um número como resultado.
- **Ação principal (Ativação):** Executa a operação se ambos os números estão presentes.
- **Evento disparador:** Emitido quando o resultado da operação estiver pronto.
- **Opções específicas:**
    - **Seleção de operação:** O jogador pode escolher qual operação será aplicada: **Adição**, **Subtração**, **Multiplicação** e **Divisão**. E se uma **divisão por zero** for realizada, a solução do nível falha, servindo como penalidade lógica.
    - **Orientação do objeto:** O operador pode ser girado verticalmente, permitindo melhor organização visual na construção da solução.
- **Observações:**
    - A ordem dos números pode influenciar o resultado em operações como subtração e divisão.

Número:

- **Função:** Representa um valor fixo que pode ser criado e editado pelo jogador. Serve como uma constante dentro da lógica da solução.
- **Interações:**
    - Seu valor pode ser **modificado** por meio de **clique duplo**, que abre um **popup de input numérico**.
    - Pode ser **inserido manualmente em contêineres** para ser utilizado na solução.
- **Observações:**
    - Este objeto é diferente dos números recebidos como entrada pelos níveis: **somente os números criados pelo jogador podem ser editados**.
    - Por sua natureza constante, o número é ideal para construir soluções que exigem valores fixos, como multiplicar outro número por 5.

Etiqueta:

- **Função:** Exibe um texto personalizado de até 20 caracteres, utilizado exclusivamente para organizar visualmente a solução criada pelo jogador.
- **Interações:**
    - O texto pode ser **editado** por meio de **clique duplo**, que abre um **popup de input** para modificação.
- **Observações:**
    - Não interage com outros objetos nem afeta a lógica da solução.
    - Serve como **elemento de anotação**, ideal para identificar partes específicas da montagem, como contêineres, operadores ou fluxos de dados.

Além dos objetos criáveis, cada nível inclui dois componentes internos fixos: um ponto de entrada dos dados e um verificador de saída. Por não serem manipuláveis pelo jogador, são descritos em detalhes na seção *Estrutura dos Níveis*.

3.4 Design de níveis

O design dos níveis do jogo foi estruturado para garantir uma curva de aprendizado gradual, onde o jogador é introduzido a conceitos de algoritmos e lógica de programação de forma simples e intuitiva. À medida que o jogador progride, a complexidade dos níveis aumenta de forma controlada, apresentando novos desafios e exigindo a aplicação de conceitos previamente aprendidos de maneiras mais sofisticadas.

3.4.1 Estrutura dos Níveis

Os níveis são projetados em torno de uma lógica de progressão modular.

Cada nível é composto por:

- **Título**: Frase curta que caracteriza o objetivo do nível.
- **Descrição**: Um texto explicando o que o nível espera do jogador, podendo conter algumas dicas.
- **Entradas**: Objetos que contêm um conjunto de números, os quais o jogador deve utilizar para completar o objetivo do nível.
- **Saídas**: Objetos que esperam receber os números processados de acordo com o que é pedido pelo nível.
- **Objetos**: Uma lista com os objetos que o jogador pode criar no nível.

Cada nível apresenta ao jogador um ou vários conjuntos de números como entrada, e o objetivo é fazer com que os objetos processem os números para enviar à saída correta. Os níveis iniciais são tutoriais que introduzem os principais elementos e mecânicas, enquanto os níveis mais avançados aumentam a complexidade das entradas e requerem soluções mais elaboradas.

3.4.2 Progressão de dificuldade

A progressão de dificuldade segue três etapas principais:

1. **Introdução aos Conceitos Básicos**: Os primeiros níveis servem para

ensinar as mecânicas centrais, como a criação de objetos, a conexão entre eles e a manipulação de números.

2. **Aplicação de Múltiplos Conceitos**: Nos níveis intermediários, o jogador combina múltiplas operações e começa a entender como diferentes componentes interagem.
3. **Desafios Combinados e Lógica Condicional**: Nos níveis mais avançados, a solução dos problemas exige o uso da maioria dos objetos, fazendo com que o jogador tenha que planejar melhor suas soluções.

3.4.3 Exemplos de Níveis

Aqui estão exemplos de alguns níveis, ilustrando como novas mecânicas são introduzidas e a dificuldade é gradualmente aumentada:

- **Nível 1: Movendo números**
  - **Entrada:**
    - [5, 2, 4]
  - **Saída:**
    - [5, 2, 4]
  - **Objetivo:** Apresentar o jogador à mecânica principal do jogo.
  - **Solução:** O jogador cria uma conexão normal do contêiner inicial para o local de saída.
- **Nível 4: Alternando**
  - **Entrada**:
    - [4, 5, 6, -3, 2, -8]
  - **Saída:**
    - [4, 6, 2]
    - [5, -3, -8]
  - **Objetivo**: Ensinar o uso do roteador para alternar para qual objeto enviar os números.
  - **Solução**: O jogador cria um roteador, conecta a entrada a ele e conecta-o às saídas.
- **Nível 6: Copiando**

    - **Entrada**:
        - [2, 3, 7,-3]
        - [1]
    - **Saída:**
        - [2, 3, 7, -3]
        - [1, 1, 1, 1]
    - **Objetivo**: Mostrar como fazer cópias dos números.
    - **Solução**: O jogador conecta dois transportadores a suas respectivas saídas, um no modo de mover e o outro no modo de cópia.
- **Nível 8: Ativação Manual**
    - **Entrada**:
        - [2, 3, 7]
        - [5, -2, 11]
    - **Saída:**
        - [2, 5, 3, -2, 7, 11]
    - **Objetivo**: Mostrar como o jogador pode ativar manualmente um objeto como o transportador.
    - **Solução**: O jogador cria dois transportadores, um automático e outro manual, e utiliza um ativador para transmitir o sinal de ativação do automático para o manual.
- **Nível 14: Somando**
    - **Entrada**:
        - [7, -3, 2]
        - [2, 5, -22]
    - **Saída:**
        - [9, 2, -20]
    - **Objetivo**: Introduzir o jogador ao operador matemático.
    - **Solução**: Jogador cria o operador e move os números para fazer a operação desejada.
- **Nível 18: Em partes**
    - **Entrada**:

        - [4, 10, 8, 38]
    - **Saída:**
        - [14, 7, 46, 23]
    - **Objetivo**: Nível final que utiliza todas as mecânicas do jogo combinadas, pedindo para o jogador : “Envie a soma e a média de cada par de números”.

3.4.4 Testes de Validação

A validação das soluções propostas pelo jogador é uma etapa fundamental para garantir a integridade pedagógica do jogo. Conforme o jogador vai enviando os números, o nível verifica se os valores enviados às saídas estão corretos e seguem a ordem esperada, fazendo uma comparação com uma sequência de valores esperados.

Os valores utilizados nesses testes foram planejados para abranger uma ampla variedade de cenários possíveis. Essa abordagem assegura que o jogador não esteja apenas resolvendo o caso apresentado inicialmente, mas que sua lógica seja robusta o suficiente para lidar com variações, incluindo situações excepcionais como divisões por zero ou estruturas condicionais complexas.

Se todos os testes forem aprovados, o nível é marcado como concluído, liberando a progressão para os próximos desafios. Caso contrário, a execução é interrompida, e uma janela de aviso (popup) é exibida detalhando o erro identificado. Esse feedback imediato permite ao jogador revisar sua abordagem e refinar sua solução.

Esse método de validação não apenas assegura a correção das respostas, mas também reforça o entendimento da lógica aplicada, promovendo o aprendizado ativo e eficaz dos conceitos de algoritmos por meio da prática e da experimentação.

3.5 Arquitetura do Sistema

A arquitetura do jogo foi cuidadosamente projetada para garantir

modularidade, organização e escalabilidade. Cada componente funciona de forma isolada, permitindo atualizações e expansões sem afetar o funcionamento global da aplicação. Essa estrutura favorece a manutenção, evolução incremental e introdução de novas funcionalidades educacionais.

3.5.1 Padrões de Organização

3.5.1.1 Estrutura de Diretórios

O projeto adota uma divisão lógica e funcional dos diretórios, promovendo clareza na separação de responsabilidades:

- **Diretório scripts:** Contém os scripts responsáveis pela lógica do jogo, organizados em subpastas específicas:
    - level/: gerenciamento de níveis. (ex: level.gd, level_manager.gd, level_executor.gd)
    - resources/: definição de recursos personalizados e estruturas de dados. (LevelPropsResource, HighlightOptions, ToolBoxItem, etc.)
    - global.gd: Script singleton que mantém o estado da aplicação e fornece funcionalidades compartilhadas entre diferentes componentes, garantindo persistência de dados durante a execução.
- **Diretório scenes:** Reúne os elementos visuais e interativos:
    - objects/: implementação dos objetos do jogo. (containers, operadores, roteadores)
    - ui/: interface gráfica do usuário. (menus, controles e painéis informativos)
    - utils/: componentes reutilizáveis. (conexão entre objetos, animações, utilitários)
- **Diretório assets:** Armazena recursos visuais e arquivos de configuração:
    - Sprites dos objetos. (container.png, Router.png, etc.)
    - Ícones de interface. (Cart.png, Cross.png, Exit.png)
    - Shaders personalizados. (outline.gdshader para efeitos visuais)
    - Fontes tipográficas (UI/fonts/Doto/) para consistência na interface.

- Paleta de cores (color-palette.ase) que garante uniformidade estética.

3.5.1.2 Padrões Arquiteturais

A arquitetura do sistema adota práticas consolidadas para garantir eficiência e organização:

- **Separação de Responsabilidades:** As camadas de lógica (scripts), apresentação (scenes) e recursos (assets) operam de maneira independente, facilitando testes e manutenções, seguindo o ***Princípio de Responsabilidade Única (SRP) do SOLID***.
- **Sistema de Comunicação por Sinais:** Utilização do sistema de sinais da Godot para promover comunicação assíncrona entre componentes, sendo uma forma do **Padrão Observer,** ajuda minimizando o acoplamento direto entre objetos.
- **Arquitetura Orientada a Componentes:** Objetos do jogo são compostos por módulos reutilizáveis, como interação com mouse e gerenciamento de conexões, o que acelera o desenvolvimento de novas funcionalidades.

3.5.2 Organização de Objetos para Extensibilidade

3.5.2.1 Sistema de Herança e Composição

Para promover reusabilidade e adaptação, o sistema utiliza herança e composição de forma estratégica:

- **Classes Base:** Componentes genéricos como **MouseInteractionArea2D** e **ConnectionArea** fornecem funcionalidades essenciais que são estendidas por objetos específicos.
- **Padrão de Projeto Builder:** Objetos como **PathBuilder**, que herdam de **BaseLinkBuilder**, permitem a implementação de novos tipos de conexão com mínimo esforço, reaproveitando lógica existente.
- **Interface Uniforme para Conexão:** Todos os objetos que podem se conectar seguem uma interface padronizada, garantindo interoperabilidade

entre diferentes componentes.

3.5.2.2 Facilidades para Desenvolvimento

O sistema é otimizado para facilitar a criação e expansão:

- **Componentes Modulares Reutilizáveis:** Ações como edição de valores, reset de estado e interação com o cursor são implementadas como módulos reaproveitáveis.
- **Sistema de Templates:** A pasta _template/, na raiz dos níveis, fornece estruturas base para acelerar a criação de novos conteúdos.
- **Separação entre Lógica e Apresentação:** A lógica de processamento dos desafios é independente da representação gráfica, permitindo modificações visuais sem afetar o comportamento funcional dos níveis.
- **Scripts Singleton**: Scripts como SceneManager e Highlight que centralizam operações específicas, promovendo reutilização e consistência.

3.5.3 Sistema de Níveis

3.5.3.1 Estrutura e Organização

- **Definição por Recursos:** Cada nível é descrito por um recurso (LevelPropsResource), que especifica os dados de entrada, os valores esperados na saída, os objetos disponíveis para o jogador e metadados como título e descrição.
- **Instanciação Dinâmica:** Os níveis são gerados automaticamente com base nas configurações definidas, posicionando objetos de entrada e saída de maneira apropriada na interface.
- **Flexibilidade de Layout:** Suporte a layouts customizados por nível, permitindo variações visuais específicas quando necessário.

3.5.3.2 Gerenciamento e Validação

- **Controle de Progressão:** O componente LevelManager administra o acesso e a navegação entre os níveis.

- **Execução e Verificação:** O LevelExecutor é responsável pela coordenação da execução dos fluxos de dados e pela validação dos resultados, comparando com os objetivos definidos em cada nível.

3.5.3.3 Flexibilidade e Expansão

- **Configuração Declarativa:** Novos níveis podem ser criados por meio de arquivos de configuração simples, sem necessidade de alterações no código-fonte.
- **Expansão de Objetos:** A inclusão de novos objetos é feita por meio da configuração de ToolBoxItem e ObjectsDescription, promovendo evolução contínua da mecânica.
- **Organização Visual Coerente:** A estrutura da pasta assets possibilita a adição de novos elementos gráficos, mantendo consistência visual por meio de paletas e padrões predefinidos.

3.6 Tecnologias utilizadas

Uma Game Engine é uma ferramenta de desenvolvimento que fornece a base para criar jogos de forma mais rápida e eficiente. Elas oferecem uma série de funcionalidades prontas, como sistemas de renderização gráfica, física, som, e animações, permitindo que desenvolvedores concentrem seus esforços na lógica e no design do jogo. Além disso, as game engines facilitam a criação de ambientes interativos ao fornecer interfaces intuitivas e suporte para várias linguagens de programação, o que ajuda a reduzir o tempo e a complexidade do desenvolvimento.

Com uma game engine, os desenvolvedores não precisam construir cada elemento do zero, como a física dos objetos ou a exibição dos gráficos. Isso permite que mais tempo seja dedicado à criação de mecânicas, design visual e otimização da jogabilidade, sem se preocupar tanto com os aspectos técnicos de baixo nível. Exemplos de engines populares de jogos incluem *Unity*, *Unreal Engine* e *Godot*.

Para o desenvolvimento deste projeto, foi escolhida a engine *Godot*, uma

game engine open-source que tem se destacado por sua flexibilidade, facilidade de uso e por oferecer uma vasta gama de funcionalidades para jogos 2D e 3D. A Godot foi escolhida por diversos motivos:

Interface Intuitiva: A Godot possui uma interface visual limpa e de fácil aprendizado, o que acelera o desenvolvimento. Essa simplicidade é especialmente útil quando se trabalha com jogos educativos, pois permite concentrar-se nas mecânicas e no design pedagógico sem se perder em complexidades técnicas.

Open-Source e Leveza: Por ser open-source, a Godot é gratuita e permite que os desenvolvedores façam modificações no seu código, se necessário, garantindo flexibilidade e independência. Além disso, a engine é leve, o que facilita o desenvolvimento em máquinas menos potentes e a exportação de jogos para diferentes plataformas.

Suporte a Diversas Linguagens: A Godot suporta nativamente GDScript, sua linguagem nativa, e C#. Para este projeto, foi utilizado o GDScript por conta de sua integração com a Engine e sua versatilidade, podendo utilizar tipagem estática ou dinâmica dependendo da situação.

Especialização em 2D: Como o foco deste jogo é em um ambiente 2D, a Godot foi uma escolha ideal, já que possui uma das melhores implementações para jogos 2D entre as game engines disponíveis, garantindo bom desempenho e facilidade de controle sobre os elementos visuais.

Esses fatores foram determinantes para a escolha da Godot, garantindo uma solução equilibrada entre desempenho, flexibilidade e facilidade de uso.

3.7 Interface de usuário

A interface do software foi projetada para garantir interação intuitiva e feedback imediato ao usuário. O layout segue princípios de usabilidade que priorizam a consistência de padrões visuais e facilidade de navegação. Componentes como botões, barras de progresso e notificações foram dispostos

de forma a reduzir a carga cognitiva e acelerar o aprendizado de algoritmos por meio de pistas visuais claras.

3.7.1 Telas do jogo

- **Tela Inicial**:
  - A tela inicial mostrada na Figura 9 é exibida assim que o jogador inicia o aplicativo. Nela, estão disponíveis dois botões principais – **“Jogar”** e **“Sair”** – compostos por ícone e legenda, o que assegura clareza na comunicação das funções.
  - Botão responsável por alternar entre os modos de janela e tela cheia, visando melhorar a imersão do usuário.
  - No plano de fundo, posiciona-se o objeto “Transportador”, elemento visual que antecipa a ambientação e sugere as mecânicas centrais do jogo.

Figura 9 – Tela inicial.

Fonte: Elaborado pelo autor

- **Menu de Níveis**:
  - O menu de níveis apresentado na Figura 10 organiza os níveis em

forma de grade, exibindo todos os desafios disponíveis de modo organizado.

- Ao posicionar o cursor sobre cada nível, um painel emergente revela o título correspondente, antecipando a temática e os objetivos da fase.
- No canto da tela, uma legenda descreve o significado dos ícones sobrepostos aos botões, assegurando compreensão imediata das funcionalidades oferecidas.
- A navegação entre os conjuntos de níveis é realizada por meio de setas posicionadas lateralmente ou através do indicador inferior, o qual também permite a seleção direta do conjunto a ser exibido.

Figura 10 – Tela de seleção de níveis

Fonte: Elaborado pelo autor

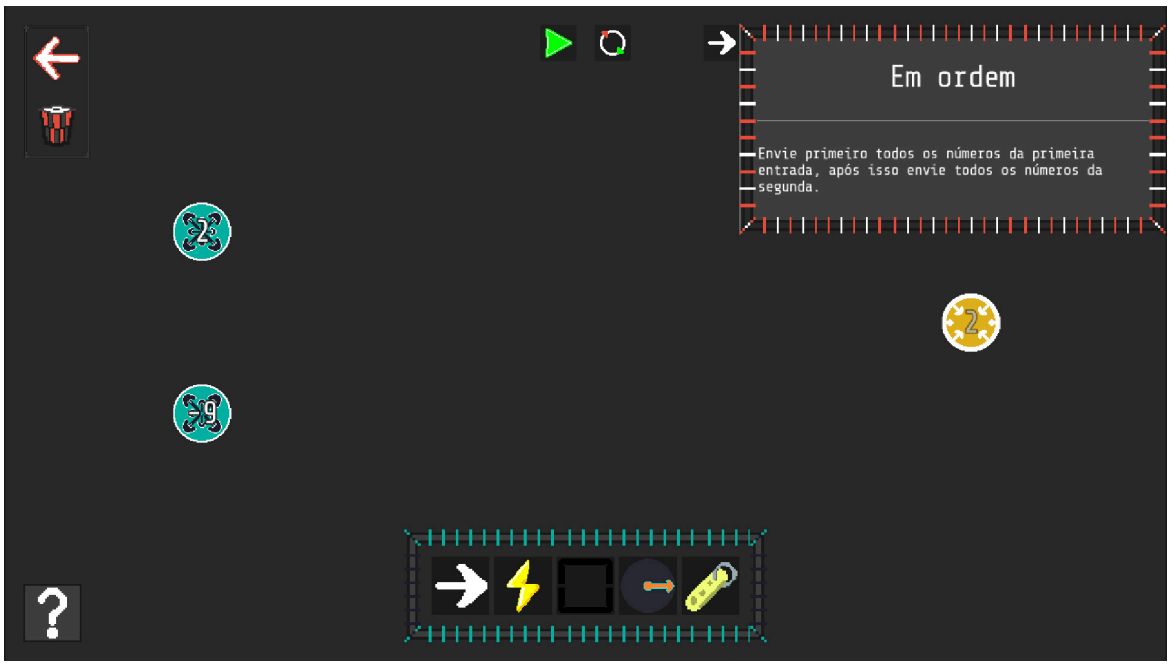
- **Tela do nível**: A tela do nível é exibida ao iniciar cada fase e organiza os elementos de controle, informação e criação de objetos de forma a otimizar a experiência do jogador.
  - **Controladores**: Na parte superior, estão dispostos os botões “Rodar”, “Pausar” e “Resetar”, identificados por ícones descritivos,

que permitem iniciar, interromper e reiniciar a execução da sequência de comandos.

- **Informação**: À direita, um painel retrátil e redimensionável apresenta as metas e instruções específicas do nível, garantindo clareza sobre os objetivos a serem alcançados.
- **Barra de objetos**: Na parte inferior, a barra de objetos reúne os itens que podem ser criados e posicionados no cenário, facilitando o desenvolvimento das soluções propostas.
- **Opções extras**: Por fim, no lado esquerdo, encontram-se os botões “Voltar”, “Apagar objetos” e “Ajuda”, igualmente ilustrados por ícones, que oferecem recursos auxiliares sem comprometer a área principal de interação.

Na figura 11 é mostrada a tela do nível na qual o jogador interage e monta sua solução:

Figura 11 – Tela do nível

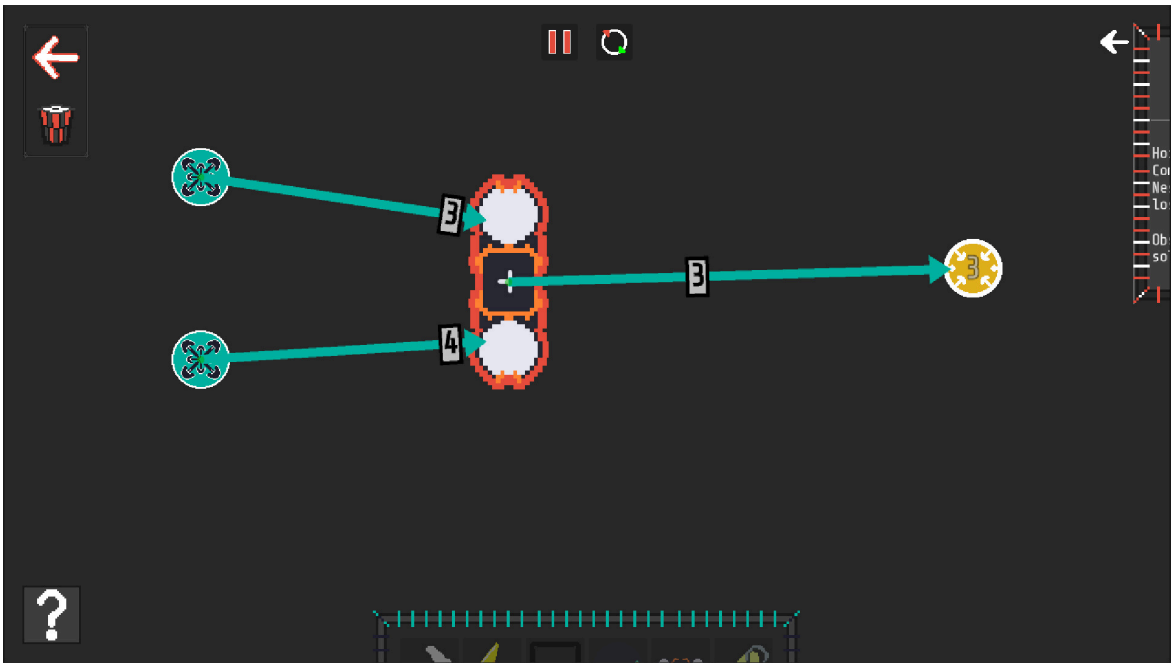


Fonte: Elaborado pelo autor.

Já na figura 12, mostra a tela de nível ao começar a execução da solução feita

pelo jogador, como visto abaixo:

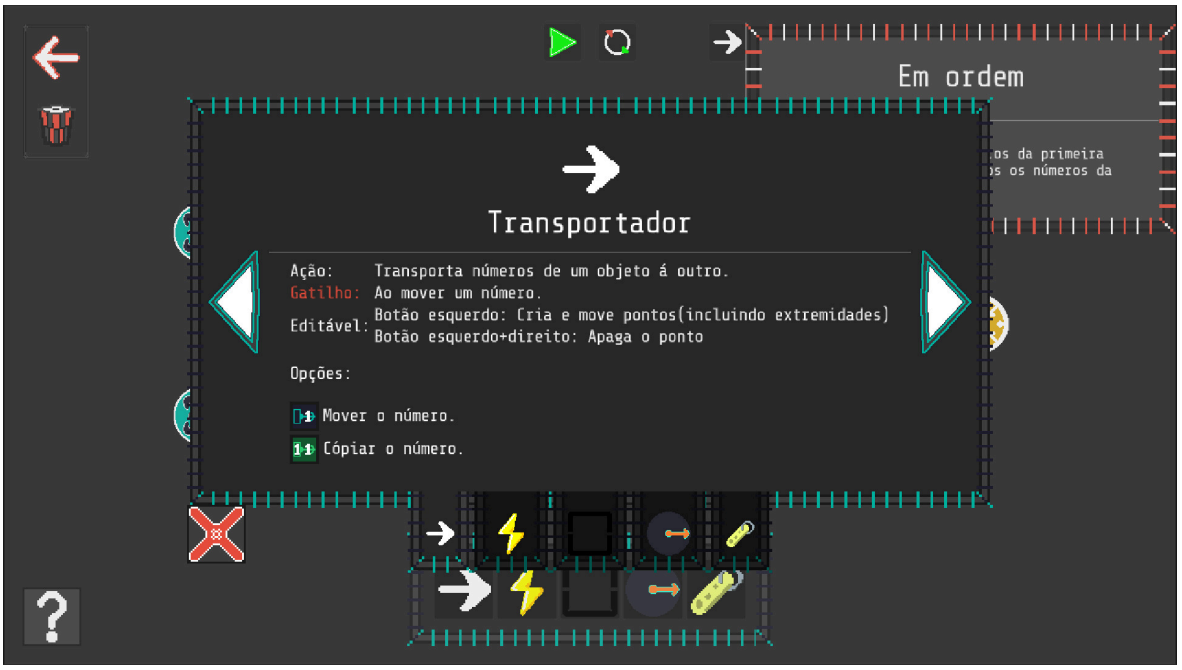
Figura 12 – Tela do nível em execução.



Fonte: Elaborado pelo autor.

- **Tela de ajuda:** A tela de ajuda, mostrada na Figura 13, fornece explicações detalhadas sobre o funcionamento de cada objeto disponível, acessível por meio do botão de ajuda, aparecendo como um painel sobreposto ao nível. Setas direcionais e botões com os ícones dos objetos permitem a navegação entre os diferentes objetos, assegurando que o usuário compreenda suas funcionalidades conforme necessário.

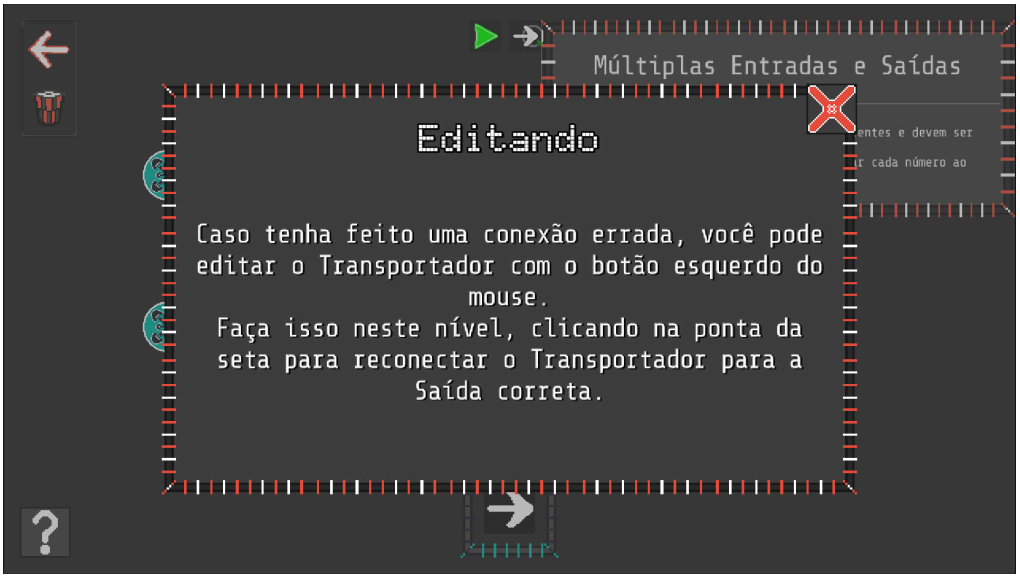
Figura 13 – Tela do ajuda



Fonte: Elaborado pelo autor

- **Tela de Aviso/Tutorial:** A tela de aviso, implementada como pop-up centralizado, apresenta informações críticas ao jogador antes ou durante o nível. Cada pop-up é composto por um título claro e um bloco de conteúdo explicativo, bloqueando temporariamente a interação com o restante da interface para garantir a assimilação das instruções antes de prosseguir. A tela de aviso padrão pode ser vista na Figura 14:

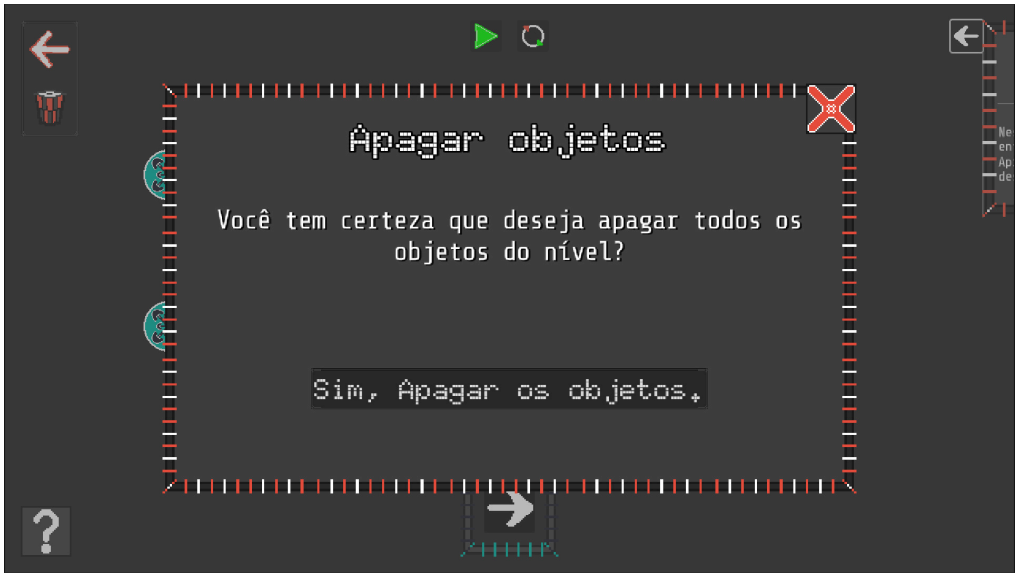
Figura 14 – Tela de aviso



Fonte: Elaborado pelo autor

Além disto, a tela de aviso também é utilizada como tela de confirmação, como, por exemplo, ao clicar no botão de apagar todos os objetos, como mostrado na Figura 15 abaixo:

Figura 15 – Tela de confirmação



Fonte: Elaborado pelo autor

3.7.2 Interação do Jogador com a Interface

- **Arrastar e Soltar**: O jogador pode arrastar e soltar objetos diretamente da barra de ferramentas para a área de trabalho, facilitando a organização e o processo de criar as soluções de forma dinâmica.
- **Zoom (interação do jogador)**: com o botão do meio, o jogador aproxima ou afasta a cena conforme sua preferência, direcionando o foco para as áreas com as quais deseja interagir com mais detalhes.
- **Conexão entre Objetos**: A interação entre os objetos é feita por meio de transportadores e ativadores, que representam a passagem de dados e disparo de eventos entre os objetos do jogo.
- **Feedback Imediato**: A interface proporciona feedback em tempo real para cada ação do jogador e de sua solução, mostrando o impacto de suas decisões. Isso inclui a ativação de conexões, gatilhos de ativação, o processamento de números pelos operadores e a validação final da solução proposta, mantendo o jogador sempre informado sobre o progresso.

3.7.3 Controles

A interação é realizada exclusivamente por meio do mouse, simplificando o esquema de comandos e mantendo o foco do jogador na resolução dos desafios. Os comandos disponíveis são os seguintes:

- **Botão Esquerdo**: Permite selecionar e manipular elementos da interface, bem como criar e posicionar objetos no cenário de jogo.
- **Botão Direito**: Aciona um menu contextual para configuração do objeto selecionado, viabilizando ajustes rápidos de propriedades sem interromper a dinâmica da fase.
- **Botão do meio (rolagem clicável):** a rolagem ajusta o nível de zoom da cena, enquanto o clique sustentado e o arraste permitem deslocar a câmera pelo ambiente de jogo.

4 CONSIDERAÇÕES FINAIS

O desenvolvimento do AlgoLab mostrou que ensinar conceitos de programação pode se tornar uma experiência envolvente e estimulante, capaz de engajar tanto iniciantes quanto quem já possui algum conhecimento prévio. Ao combinar desafios visuais com mecânicas de jogo progressivas e uma interface clara, o projeto permitiu que estruturas de repetição, tomadas de decisão e raciocínio lógico fossem assimilados de forma natural e divertida. A escolha da Godot Engine assegurou flexibilidade no desenvolvimento e suporte nativo a recursos 2D, garantindo fluidez na implementação de fases e animações.

Além dos ganhos práticos no domínio de algoritmos, o AlgoLab serviu como um laboratório para investigar como elementos lúdicos podem reforçar a motivação do estudante e ampliar sua autonomia no processo de estudo. A aplicação de testes iniciais indicou que níveis de dificuldade escalonados e feedback imediato são cruciais para manter o interesse ativo e promover a sensação de progresso contínuo. Esses aprendizados podem orientar futuras iniciativas em disciplinas de lógica de programação, incentivando educadores a explorarem abordagens similares em sala de aula.

Para evoluir o projeto, sugere-se a integração de métricas de desempenho que capturem o percurso do usuário em tempo real, bem como a expansão do repertório de desafios, incluindo estruturas de dados básicas e exercícios de complexidade crescente. A criação de versões para diferentes plataformas, aliada à construção de uma comunidade colaborativa de desenvolvimento, pode elevar o alcance e o impacto da ferramenta. Assim, o AlgoLab reafirma seu valor acadêmico ao oferecer um caminho promissor para o ensino de algoritmos, unindo aprendizado e entretenimento em um ambiente estimulante e acessível.

REFERÊNCIAS

ASSIMAKOPOULOS, V.; KARPOUZIS, K.; HAMARI, J.; XI, N.; LEGAKI, N.-Z. The effect of challenge-based gamification on learning: an experiment in the context of statistics education. International Journal of Human-Computer Studies, v. 144, p. 102496, 2020.

GIRAFFA, M.; MORA, M. da C. Evasão na disciplina de algoritmo e programação: um estudo a partir dos fatores intervenientes na perspectiva do aluno. In: CONGRESOS CLABES, 2016. (dados de publicação incompletos — verificar anais: local/organizador/páginas).

GODOT ENGINE. Godot Engine — free and open source 2D and 3D game engine. Disponível em: https://godotengine.org/
. Acesso em: 04 jul. 2025.

GO GAMERS; SX GROUP; BLEND NEW RESEARCH; ESPM. Pesquisa Game Brasil 2025: relatório gratuito. 2025. Disponível em: https://www.pesquisagamebrasil.com.br/
. Acesso em: 26 ago. 2025.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. IEEE Std 830-1998: IEEE recommended practice for software requirements specifications. New York: IEEE, 1998.

SANCHES, M. S.; PONTES, E. S.; RODRIGUES, R. S. Evasão em cursos de computação: uma revisão sistemática. Revista de Informática Teórica e Aplicada, v. 29, n. 2, 2022.

SILVA, E. P.; CACEFFO, R.; AZEVEDO, R. A syllabi analysis of CS1 courses from Brazilian public universities. Brazilian Journal of Computers in Education (Revista Brasileira de Informática na Educação – RBIE), v. 31, p. 407–436, 2023. DOI: 10.5753/rbie.2023.2870.

TOMORROW CORPORATION. Human Resource Machine. [S.l.]: Tomorrow Corporation, 2015. Disponível em: https://store.steampowered.com/app/375820/Human_Resource_Machine/
. Acesso em: 26 ago. 2025.

ZACHTRONICS. Infinifactory. [S.l.]: Zachtronics, 2015. Disponível em: https://www.zachtronics.com/infinifactory/
. Acesso em: 26 ago. 2025.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Cajazeiras - Código INEP: 25008978
Rua José Antônio da Silva, 300, Jardim Oásis, CEP 58.900-000, Cajazeiras (PB)
CNPJ: 10.783.898/0005-07 - Telefone: (83) 3532-4100

Documento Digitalizado Ostensivo (Público)

Algolab : facilitando a aprendizagem de algoritmos e lógica de forma interativa

Assunto:	Algolab : facilitando a aprendizagem de algoritmos e lógica de forma interativa
Assinado por:	Andre Lucas
Tipo do Documento:	Projeto
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Andre Lucas Tavares Dantas**, **DISCENTE (202212010009) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAJAZEIRAS**, em 17/09/2025 09:52:41.

Este documento foi armazenado no SUAP em 17/09/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1612009
Código de Autenticação: f8a18fd664