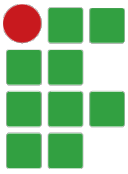


INSTITUTO FEDERAL
Rio Grande do Sul
Campus Osório

Leonardo Luz Fachel

Desenvolvimento de Jogo Sério para Auxílio no Ensino de Conceitos de Estruturas de Dados

Osório

2025

Leonardo Luz Fachel

Desenvolvimento de Jogo Sério para Auxílio no Ensino de Conceitos de Estruturas de Dados

Orientador: Bruno Chagas Alves Fernandes

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – IFRS
campus Osório
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Osório
2025

AGRADECIMENTOS

Agredecimentos...

Resumo

FIX: Este trabalho tem como objetivo apresentar os conceitos de estruturas de dados por meio de um jogo sério. O projeto busca facilitar o aprendizado dos alunos de forma lúdica e interativa. A metodologia utilizada foi a metodologia de desenvolvimento de jogos educacionais ENgAGed de forma adaptada para jogos sérios, com foco em testes práticos. Os resultados demonstraram que a gamificação pode ser uma ferramenta eficaz no ensino de estruturas de dados.

Palavras-chave: Jogo Sério. Estruturas de Dados. Ensino. Gamificação.

Abstract

FIX: This work aims to present the concepts of data structures through a serious game. The project seeks to facilitate student learning in a playful and interactive way. The methodology used was the ENgAGed educational game development methodology, adapted for serious games, with a focus on practical testing. The results demonstrated that gamification can be an effective tool in teaching data structures.

Keywords: Serious Game. Data Structures. Education. Gamification.

Lista de ilustrações

Lista de tabelas

LISTA DE ABREVIATURAS E SIGLAS

GIMP	*GNU Image Manipulation Program*
GNU	*GNU's Not Unix*
ILPC	Introdução Linguagem de Programação C
ENgAGED	*EducatioNAl GamEs Development*
LIFO	*Last In First Out*
FIFO	*First In First Out*
GBL	*Game-Based Learning*

Sumário

1 INTRODUÇÃO

O ensino e a aprendizagem de conceitos fundamentais da área de computação, como estruturas de dados, constituem um desafio recorrente para educadores e estudantes. De acordo com o autor Mtaho e Mselle (2024), a disciplina de estruturas de dados é altamente exigente para estudantes de ciência da computação, sendo frequentemente associada a uma elevada carga cognitiva e, consequentemente, a altas taxas de reprovação e evasão do curso. Entre os principais fatores que contribuem para essas dificuldades estão a natureza abstrata dos conceitos envolvidos e a baixa motivação dos alunos. Esse cenário se agrava pelo fato de que, tradicionalmente, o ensino desses conteúdos ocorre por meio de aulas expositivas e exercícios de codificação, o que tende a gerar baixa retenção do conteúdo e desinteresse por parte dos estudantes (CHILWANT, 2012).

Além disso, as novas gerações de estudantes estão cada vez mais habituadas a um fluxo constante de informações e experiências interativas, desenvolvendo um comportamento que valoriza respostas rápidas e estímulos visuais (HA; IM, 2020). Isso torna o ensino convencional ainda menos atrativo. A partir de sua pesquisa, o autor Mtaho e Mselle (2024) recomenda a adoção de novas estratégias de ensino para mitigar as dificuldades e melhorar a experiência de aprendizagem de estrutura de dados.

Nesse cenário, os jogos sérios surgem como uma estratégia educacional promissora, ao promover o aprendizado ativo e engajado (MOUAHEB et al., 2012). Diferentemente de abordagens instrucionais diretas, os jogos sérios podem ser projetados para que a aprendizagem ocorra como consequência da interação do jogador com o ambiente, desafios e regras do jogo. Essa perspectiva está alinhada à teoria do construcionismo, proposta por Papert (1993), segundo a qual o conhecimento é construído ativamente pelos alunos quando estes se envolvem com a criação, exploração e manipulação de artefatos significativos.

Jogos sérios são definidos como uma aplicação de videogames cujo objetivo principal é educar, treinar ou sensibilizar, sem abrir mão do entretenimento (MOUAHEB et al., 2012). Contudo, uma crítica recorrente a essa abordagem é que muitos desses jogos sérios falham como jogos, priorizam o conteúdo educativo de forma explícita, relegando a experiência lúdica a segundo plano, quando, na verdade, ensino e entretenimento deveriam caminhar lado a lado (MOUAHEB et al., 2012).

De acordo com Araujo e Silva (2025), atualmente, grande parte dos jogos sérios se utilizam os conceitos de programação apenas como tema, sem integrá-los verdadeiramente às suas mecânicas. Essa limitação evidencia um modelo que tende a transformar o jogo em um pretexto para ensinar diretamente, por meio de mecânicas expositivas como questionários ou simulações superficiais.

O presente trabalho propõe uma abordagem alternativa: utilizar mecânicas de jogo que representem, de forma implícita e interativa, conceitos fundamentais de estruturas de dados. Em vez de apresentar diretamente listas, pilhas ou filas, o jogo incorporará esses elementos em sua lógica e estrutura interna, permitindo que o jogador interaja com tais conceitos de forma intuitiva e contextualizada. Dessa maneira, o aprendizado ocorre como consequência da resolução de problemas e da exploração do sistema, e não como resultado de instruções explícitas ou desafios de programação.

Diferentemente de jogos educativos que simulam exercícios de codificação, o objetivo deste trabalho é projetar um jogo no qual os conceitos ensinados estejam presentes nas ações tomadas pelo jogador, mesmo que ele não os reconheça explicitamente como tais. Na próxima seção, o objetivo geral deste trabalho será aprofundado.

1.1 OBJETIVO GERAL

Este trabalho tem como objetivo geral desenvolver um jogo sério que aborde conceitos fundamentais de estruturas de dados de forma implícita, por meio de mecânicas lúdicas e interativas. A proposta busca promover um processo de aprendizagem mais significativo, intuitivo e motivador.

1.2 OBJETIVOS ESPECÍFICOS

Com base no objetivo geral, este trabalho também visa alcançar os seguintes objetivos específicos:

FIX: Grande parte destes é a metodologia cientifica

- Investigar modelos de jogos sérios e sua aplicação no ensino de conteúdos relacionados à computação;
- Projetar e implementar um jogo educacional fundamentado na metodologia ENgAGED;
- Incorporar, nas mecânicas do jogo, representações implícitas de estruturas de dados, como listas, filas e pilhas, além de algoritmos de busca e ordenação;
- Avaliar a usabilidade e a eficácia do jogo no processo de ensino e aprendizagem;
- Coletar e analisar o *feedback* dos usuários com o intuito de orientar futuras melhorias da ferramenta desenvolvida.

1.3 JUSTIFICATIVA

A proposta deste trabalho encontra respaldo na demanda por tornar o ensino de estruturas de dados mais motivador (MTAHO; MSELLE, 2024) e alinhado às expectativas das novas gerações de aprendizes. O uso de jogos sérios como recurso educacional permite contextualizar os conceitos dentro de uma narrativa envolvente, aumentando o engajamento e favorecendo a construção do conhecimento de forma mais prática e intuitiva (MOUAHEB et al., 2012), contribuindo para mitigar as duas principais dificuldades enfrentadas pelos alunos na disciplina de estrutura de dados, conforme apresentado na Tabela 1.

Tabela 1 – Fatores que contribuem para as dificuldades no aprendizado da disciplina de estruturas de dados

Fator	**Frequência**	**Percentual**	**Rank**
Natureza abstrata dos conceitos de estruturas de dados	13	31.0	1
Baixa motivação dos alunos	10	23.8	2
Natureza multidimensional dos conceitos de estruturas de dados	9	21.4	3
Natureza dinâmica dos conceitos de estruturas de dados	8	19.0	4
Metodologia de ensino inadequada	7	16.7	5
Conhecimento prévio deficiente dos alunos	7	16.7	6
Modelo mental defeituoso dos alunos	7	16.7	7
Organização ineficaz dos materiais de aprendizagem	6	14.3	8
Dificuldades no planejamento da solução do programa	5	11.9	9
Organização e implementação ineficaz do currículo	4	9.5	10

Fonte: (MTAHO; MSELLE, 2024), editado pelo Autor

Além disso, a adoção da metodologia ENgAGED (BATTISTELLA; WANGENHEIM, 2016) no processo de desenvolvimento garante uma abordagem sistemática, permitindo que os objetivos educacionais sejam alcançados sem comprometer a experiência lúdica.

Dessa forma, este trabalho justifica-se por buscar uma alternativa para o ensino de estruturas de dados, conforme recomendado por Mtaho e Mselle (2024), pretendendo contribuir para a formação de profissionais mais preparados, criativos e capazes de aplicar o conhecimento de maneira prática e estratégica no mercado de trabalho. No capítulo seguinte, apresenta-se o referencial teórico que embasa esta pesquisa, fornecendo os fundamentos conceituais necessários para compreender o desenvolvimento do trabalho.

2 REFERENCIAL TEÓRICO

Neste capítulo, serão abordados os conceitos e as tecnologias fundamentais que embasam este trabalho, conectando-os diretamente aos desafios de ensino e aprendizagem de estruturas de dados e à proposta de um jogo sério com aprendizagem implícita.

2.1 ESTRUTURAS DE DADOS

Estruturas de dados como listas, pilhas, filas, árvores e grafos fornecem mecanismos distintos para organizar e acessar informações, cada uma com vantagens e limitações específicas em termos de complexidade de inserção, remoção e busca (CORMEN et al., 2022). A compreensão dessas diferenças é essencial para o desenvolvimento de algoritmos eficientes e para a resolução de problemas computacionais complexos. Entretanto, a abstração desses conceitos representa uma barreira significativa para estudantes, tornando necessário explorar métodos pedagógicos mais interativos e motivadores.

Conforme destacado por Mtaho e Mselle (2024), a disciplina de estruturas de dados é frequentemente associada a uma elevada carga cognitiva e a altas taxas de reprovação e evasão em cursos de computação. As dificuldades advêm, em grande parte, da natureza abstrata dos conceitos envolvidos e da baixa motivação dos alunos conforme indicado na Tabela 1. O modelo de ensino tradicional, que se baseia em aulas expositivas e exercícios de codificação, muitas vezes não consegue proporcionar a retenção efetiva do conteúdo nem despertar o interesse necessário para a compreensão aprofundada (CHILWANT, 2012). Essa lacuna pedagógica ressalta a urgência de explorar e implementar estratégias de ensino inovadoras que possam mitigar essas barreiras e enriquecer a experiência de aprendizagem.

2.2 JOGOS SÉRIOS

Jogos sérios são definidos como aplicações interativas que utilizam o design e a tecnologia dos videogames para fins que vão além do puro entretenimento, como educação, treinamento ou conscientização (MOUAHEB et al., 2012). Eles surgem como uma alternativa promissora às metodologias de ensino tradicionais, buscando engajar os alunos em ambientes de aprendizagem imersivos e motivadores.

Contudo, a eficácia de um jogo sério não reside apenas em seu conteúdo educacional, mas na sua capacidade de integrá-lo de forma coesa à experiência lúdica. Uma crítica recorrente a muitos jogos com propósito educacional é que eles falham em ser bons jogos, priorizando a instrução direta em detrimento da jogabilidade (MOUAHEB et al., 2012). De acordo com

Araujo e Silva (2025), muitos jogos sérios para o ensino de programação, por exemplo, utilizam os conceitos apenas como tema, sem incorporá-los profundamente em suas mecânicas centrais. Isso resulta em experiências que se assemelham mais a questionários interativos ou a exercícios de codificação disfarçados, perdendo o potencial transformador que a mídia dos jogos pode oferecer.

2.3 APRENDIZAGEM BASEADA EM JOGOS

A Aprendizagem Baseada em Jogos, do inglês *Game-Based Learning* (GBL), é uma estratégia pedagógica que emprega jogos completos como ferramentas para alcançar objetivos de aprendizagem específicos. Diferencia-se da gamificação, que apenas aplica elementos de jogos (como pontos, medalhas e *rankings*) a contextos não lúdicos. Na GBL, o aprendizado emerge da própria interação do jogador com as mecânicas, sistemas e narrativas do jogo (MALONE; LEPPER, 2021).

O potencial da GBL reside na sua capacidade de criar um ciclo de aprendizado motivado intrinsecamente. Um jogo bem projetado desafia o jogador, oferece *feedback* constante e permite a experimentação em um ambiente seguro, onde o erro é parte do processo de descoberta. Em vez de receber informações de forma passiva, o jogador aprende ativamente ao testar hipóteses, resolver problemas e superar obstáculos. Essa abordagem é particularmente relevante para conceitos abstratos como os de estruturas de dados, pois permite que os alunos visualizem e manipulem representações concretas desses conceitos dentro do universo do jogo, promovendo um engajamento que o ensino tradicional muitas vezes não consegue (MTAHO; MSELLE, 2024).

2.4 CONSTRUCIONISMO

FIX: Trocar para micromundos?

A fundamentação pedagógica deste trabalho está ancorada na teoria do Construcionismo de Seymour Papert. Derivada do construtivismo de Piaget, a teoria de Papert postula que a aprendizagem é mais eficaz quando o aprendiz está conscientemente engajado na construção de um artefato público e tangível, seja ele um castelo de areia, um poema, uma máquina ou um programa de computador (PAPERT, 1993). Essa perspectiva é crucial para o desenvolvimento de ambientes de aprendizagem que promovam a autonomia e a descoberta.

O Construcionismo defende que o conhecimento não é algo a ser simplesmente transmitido, mas algo a ser construído e reconstruído pelo indivíduo por meio de ações e interações com o mundo. Nesse contexto, o jogo proposto neste trabalho pode ser visto como um "micromundo" de aprendizagem, um ambiente onde os jogadores constroem seu entendimento sobre estruturas

de dados não por meio de instrução direta, mas ao manipular os sistemas do jogo para resolver problemas. As soluções que o jogador cria dentro do jogo são os artefatos que refletem e solidificam seu aprendizado. Essa abordagem se alinha perfeitamente à proposta de um aprendizado implícito, onde o conhecimento é uma consequência direta da experiência e da ação, e não o seu pré-requisito, conforme a metodologia de ensino que busca desacoplar o conceito ensinado de uma linguagem de programação específica, focando na compreensão conceitual através da interação lúdica.

2.5 UNITY

Para a concretização da proposta pedagógica deste trabalho, que visa o ensino implícito de estruturas de dados por meio de um jogo sério, a *game engine* Unity foi selecionada como a principal ferramenta de desenvolvimento. A Unity é um ambiente de desenvolvimento multiplataforma amplamente reconhecido na indústria de jogos, simulações e aplicações interativas, destacando-se por sua flexibilidade e robustez (Unity Technologies, 2025).

A escolha da Unity é fundamentada em suas características que se alinham diretamente aos objetivos do projeto. Primeiramente, sua arquitetura baseada em componentes e a utilização da linguagem C# permitem a criação de mecânicas de jogo sofisticadas e a representação abstrata de estruturas de dados de forma eficiente. Isso é crucial para a implementação de um aprendizado implícito, onde os conceitos são integrados à lógica do jogo sem serem explicitamente ensinados. Em segundo lugar, a capacidade multiplataforma da Unity garante que o jogo possa ser acessado por um público mais amplo de estudantes, independentemente do sistema operacional. Por fim, a vasta comunidade de desenvolvedores e a rica documentação disponível para a Unity aceleram o processo de desenvolvimento, permitindo uma maior concentração no design da experiência de aprendizagem (Unity Technologies, 2025).

3 METODOLOGIA

Segundo Creswell e Creswell (2021), um projeto de pesquisa é um plano estruturado que orienta todo o processo investigativo, abrangendo desde a formulação do problema e os objetivos do estudo até os procedimentos de coleta e análise de dados. A definição da metodologia é essencial, pois fornece ao pesquisador um caminho claro para desenvolver o estudo de maneira coerente e fundamentada. Diante disso, este capítulo apresenta os métodos de pesquisa e desenvolvimento utilizados, detalhando as escolhas metodológicas adotadas ao longo do trabalho.

3.1 METODOLOGIA CIENTÍFICA

A metodologia científica adotada neste trabalho é de natureza experimental, com abordagem mista, qualitativa e quantitativa, e classifica-se como uma pesquisa aplicada. A próxima seção detalha e justifica essas escolhas.

3.1.1 Classificação Metodológica

Este trabalho caracteriza-se como uma pesquisa aplicada, pois tem como objetivo solucionar um problema prático relacionado ao ensino de estruturas de dados por meio da utilização de um jogo sério (MOUAHEB et al., 2012). Diferentemente da pesquisa puramente teórica, a pesquisa aplicada visa gerar conhecimento com aplicação direta em contextos específicos. Neste caso, o foco está no ambiente educacional.

A natureza experimental da pesquisa se deve ao fato de propor uma intervenção concreta, que envolve o desenvolvimento e a aplicação de um protótipo funcional de jogo em um ambiente controlado com usuários reais. O objetivo é observar os efeitos dessa intervenção no processo de aprendizagem.

Adota-se uma abordagem mista, combinando métodos qualitativos e quantitativos com o intuito de proporcionar uma análise mais abrangente e precisa dos resultados. Os dados quantitativos são coletados por meio de instrumentos como questionários estruturados e testes de desempenho, permitindo uma avaliação objetiva. Já os dados qualitativos são obtidos por meio de observações, entrevistas e análise do comportamento dos participantes durante a interação com o jogo. Isso permite compreender de forma mais aprofundada a experiência dos usuários e a eficácia da ferramenta no processo de ensino e aprendizagem.

3.1.2 Etapas da Pesquisa

Esta pesquisa foi composta por diversas etapas e teve início com uma revisão bibliográfica, por meio da qual foram identificadas publicações acadêmicas e aplicações relacionadas ao uso de jogos sérios no ensino de conceitos de programação. Essa etapa proporcionou uma visão das abordagens já utilizadas, seus resultados e os conceitos mais frequentemente aplicados.

Em seguida, foi idealizado e desenvolvido um jogo sério educacional com base na metodologia ENgAGED (BATTISTELLA; WANGENHEIM, 2016). Esse processo envolveu diversas fases como a concepção, implementação e aplicação do jogo proposto, seguido da coleta de dados por meio de testes com usuários. Nessa etapa, foram empregadas técnicas qualitativas e quantitativas para avaliar a experiência dos participantes e a eficácia da ferramenta como recurso educacional.

Por fim, os resultados foram validados. Esta etapa consistiu no cruzamento dos dados obtidos com os objetivos da pesquisa, com o propósito de verificar se o jogo contribuiu de maneira significativa para o processo de ensino e aprendizagem de estruturas de dados.

3.2 METODOLOGIA DE DESENVOLVIMENTO

Para o desenvolvimento do jogo sério foi adotada, de forma adaptada, a metodologia ENgAGED, proposta por Battistella e Wangenheim (2016). O processo integra princípios de design instrucional com design de jogos, estruturando o desenvolvimento em fases sistemáticas que asseguram coerência pedagógica. Essa abordagem é particularmente adequada para projetos que buscam equilibrar objetivos educacionais com experiências lúdicas envolventes, evitando a falha comum de priorizar apenas um dos aspectos em detrimento do outro.

O processo ENgAGED estrutura-se em cinco fases principais, cada uma com objetivos e artefatos específicos. A seguir, cada fase é descrita e contextualizada para o desenvolvimento do jogo proposto.

3.2.1 Fase 1 - Análise da Unidade Instrucional

A primeira fase concentra-se na análise do contexto educacional, da definição do público-alvo e na especificação dos objetivos de aprendizagem que orientarão todo o desenvolvimento.

A1.1 - Especificação da Unidade Instrucional

A unidade instrucional foi definida como o ensino de conceitos fundamentais de estruturas de dados (pilha, fila e lista) no contexto de cursos de graduação em Análise e Desenvolvimento de Sistemas ou disciplinas correlatas de Computação. O foco está em permitir que os aprendizes

compreendam o funcionamento dessas estruturas e suas operações básicas (inserção, remoção e ordenação) de forma prática e contextualizada. Os objetivos de aprendizagem específicos incluem:

- Compreender o funcionamento de estruturas de dados lineares (pilha, fila e lista);
- Identificar as diferenças entre operações LIFO (último a entrar, primeiro a sair) e FIFO (primeiro a entrar, primeiro a sair);
- Aplicar conceitos de manipulação de estruturas em cenários de resolução de problemas;

A1.2 - Caracterização dos Aprendizes

O público-alvo deste estudo compreende estudantes de graduação em Análise e Desenvolvimento de Sistemas ou cursos correlatos na área de Computação, com idade entre 18 e 50 anos, que possuam acesso a computadores pessoais equipados com sistema operacional capaz de executar navegadores modernos, e que utilizem teclado e mouse como principais formas de interação.

Embora o jogo tenha sido projetado com esse público em mente, sua acessibilidade permite que qualquer pessoa interessada em jogos de forma geral possa participar, sem exigir conhecimento prévio explícito de estruturas de dados.

A1.3 - Definição dos Objetivos de Desempenho

O objetivo de desempenho do jogo é que, ao final da experiência, o jogador seja capaz de:

- Executar operações de inserção e remoção respeitando as regras de cada estrutura;
- Entender e aplicar conceitos de ordenação;
- Reconhecer padrões de funcionamento das estruturas através da interação implícita com o jogo.

3.2.2 Fase 2 - Projeto da Unidade Instrucional

A segunda fase dedica-se ao design instrucional, definindo como os conteúdos serão apresentados, como será realizada a avaliação e quais estratégias pedagógicas serão empregadas.

A2.1 - Definição da Avaliação do Aprendiz

A avaliação é incorporada diretamente nas mecânicas do jogo, funcionando como mecanismo de feedback contínuo:

- **Avaliação Formativa Implícita**: ao realizar combinações corretas de elementos alquímicos, o jogador recebe feedback positivo instantâneo (sucesso na ação);
- **Avaliação por Penalidade**: combinações incorretas resultam em penalidade (perda de pontos de vida e feedback visual), estimulando o jogador a refletir sobre suas ações e ajustar sua estratégia;
- **Métricas de Desempenho**: ao final de cada fase, são registradas métricas como tempo de conclusão, tentativas, erros e acertos de combinações.

Essa abordagem de avaliação está alinhada ao construcionismo (PAPERT, 1993), pois permite que o aprendiz construa seu conhecimento através da exploração, tentativa e erro, em um ambiente seguro onde o fracasso é parte do processo de descoberta.

A2.2 - Definição do Conteúdo e da Estratégia Instrucional

O conteúdo é introduzido de forma gradual na primeira fase, seguindo um nível de dificuldade crescente. Desde o início, o jogador tem acesso a todas as funcionalidades do jogo; contudo, essas são apresentadas de maneira progressiva ao longo dessa etapa inicial, permitindo uma assimilação natural dos conceitos e mecânicas.

A estratégia instrucional priorizará o **aprendizado implícito**: os conceitos não são apresentados verbalmente ou textualmente, mas descobertos através da interação com as mecânicas do jogo. Dessa forma, a aprendizagem emerge da necessidade prática de resolver desafios.

A2.3 - Decisão sobre Desenvolvimento ou Reutilização

Optou-se pelo **desenvolvimento original do jogo**. Essa decisão baseou-se em:

- Inexistência de jogos disponíveis que integrem a mecânica de estruturas de dados de forma implicita com temática de alquimia e estilo visual de pixel art inspirado em clássicos como Mario e Mega Man;
- Necessidade de controle total sobre a implementação das estruturas de dados para garantir precisão pedagógica;
- Oportunidade de customizar completamente a narrativa, visual e mecânicas para alinhar com os objetivos educacionais específicos;

A2.4 - Revisão do Modelo de Avaliação

O modelo de avaliação do jogo segue os princípios do **MEEGA+** (Modelo para Avaliação de Jogos Educacionais) de forma adaptada, abordando dimensões de:

- **Usabilidade**: facilidade de aprender e usar o jogo, clareza da interface;
- **Engajamento**: capacidade de manter o interesse do jogador, imersão, diversão;
- **Aprendizagem Percebida**: percepção do jogador sobre o aprendizado obtido, aplicabilidade dos conceitos;
- **Experiência do Usuário**: satisfação geral, disposição para recomendação do jogo.

Os feedbacks são implementados de forma imediata no jogo, reforçando a aprendizagem e orientando o jogador através de:

- Feedback visual (mudanças de cor, animações, efeitos particulares);
- Feedback sonoro (sons de sucesso ou erro);

3.2.3 Fase 3 - Desenvolvimento do Jogo Educacional

Esta fase abrange as etapas práticas de implementação: levantamento de requisitos, concepção visual e narrativa, design de mecânicas e implementação técnica. Esta etapa será mais aprofundada no Capítulo 5

3.2.4 Fase 4 - Execução da Unidade Instrucional

FIX: Na quarta fase, o jogo seria integrado em contextos educacionais reais, sendo utilizado em sala de aula como ferramenta complementar de ensino. Entretanto, por falta de tempo, este será proposto como trabalhos futuros.

3.2.5 Fase 5 - Avaliação da Unidade Instrucional

FIX: A avaliação final mede a efetividade global do jogo como ferramenta educacional e não será feita no trabalho proposto.

3.2.6 Síntese da Metodologia de Desenvolvimento

A adoção da metodologia ENgAGED garante que o desenvolvimento do jogo mantenha coerência sistemática, integrando continuamente princípios de design educacional com design de jogos. Ao estruturar o processo em fases bem definidas, com artefatos claros e objetivos específicos em cada etapa, a metodologia assegura que:

- O conteúdo educacional está solidamente fundamentado em objetivos de aprendizagem bem definidos;
- As mecânicas do jogo refletem os conceitos de estruturas de dados de forma implícita, promovendo aprendizado significativo;
- A experiência lúdica é priorizada, mantendo o jogo divertido e engajador;
- A avaliação é contínua e sistemática, fornecendo retroalimentação para iterações futuras;
- O resultado final é uma ferramenta educacional robusta, testada e validada para contribuir efetivamente ao processo de ensino e aprendizagem de estruturas de dados.

Dessa forma, este trabalho configura-se não apenas como um exercício acadêmico de desenvolvimento de software, mas também como uma contribuição metodologicamente estruturada para o campo da educação em computação, em especial, no que se refere à inovação de estratégias que tornem o ensino de estruturas de dados mais motivador, prático e efetivo.

4 TRABALHOS RELACIONADOS

Este capítulo apresenta uma revisão dos trabalhos existentes que combinam jogos sérios com o ensino de conceitos de programação. O conteúdo está organizado em duas partes: a primeira aborda pesquisas acadêmicas relacionadas ao tema, enquanto a segunda explora jogos já existentes com propostas semelhantes. Ao final, é apresentada uma síntese geral dos principais aspectos identificados nos trabalhos analisados.

4.1 ARTIGOS

Foi realizada uma pesquisa na plataforma *Google Scholar* com o objetivo de identificar trabalhos correlatos desenvolvidos nos últimos cinco anos (2021 a 2025). Utilizaram-se as palavras-chave “Estrutura de Dados”, “Jogo Sério” e “Desenvolvimento”, o que resultou inicialmente em um total de 45 artigos.

Após uma análise preliminar, 40 artigos foram descartados por não envolverem o desenvolvimento de um jogo ou por tratarem de jogos cuja temática não estava relacionada à área de programação. Com isso, restaram 5 artigos para uma avaliação mais aprofundada.

Os artigos selecionados para essa etapa de análise detalhada foram aqueles que atenderam simultaneamente aos seguintes critérios: aplicação de jogo sério no ensino de programação e apresentação de um protótipo funcional ou em desenvolvimento. Sendo estes:

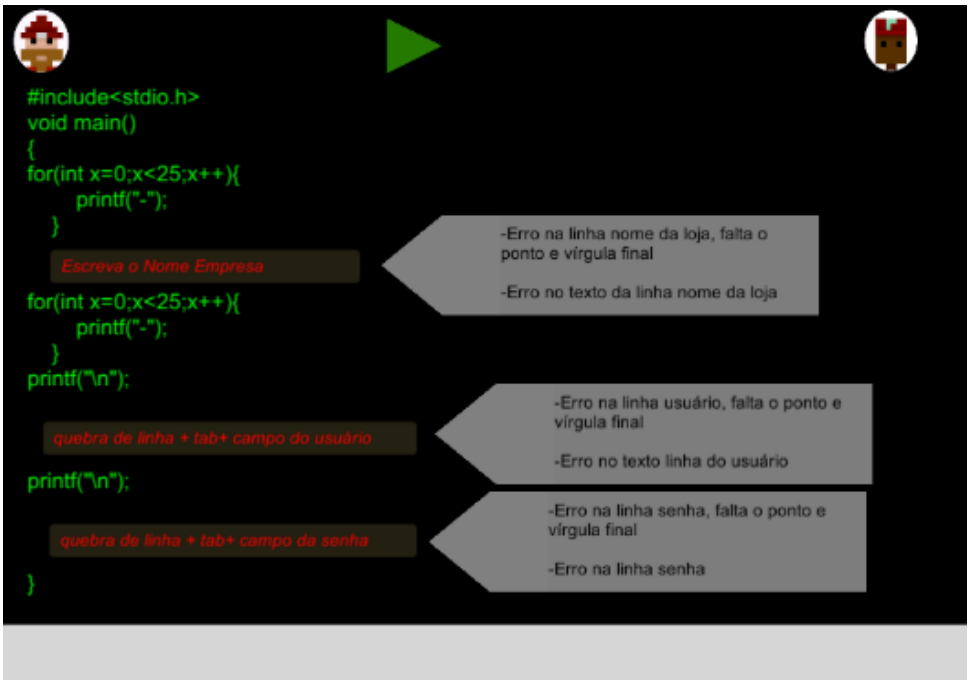
4.1.1 CodingJob: Um jogo sério para auxiliar nas disciplinas de Linguagem de Programação C

A dissertação CodingJob (COSTA et al., 2023) apresenta um jogo sério que fornece um ambiente para prática dos conhecimentos da disciplina de Linguagem de Programação I utilizando a linguagem C, não abrangindo nenhum conceito de Estruturas de Dados.

Este é um jogo *puzzle* que apresenta desafios de programação, simulando um ambiente de trabalho, o objetivo do jogador é arrumar as secções de códigos necessárias. Por se tratar de um simulador, o conteudo didático deste é ensinado de forma explicita.

Por fim, identificou-se que o projeto analisado apresentou boa aceitação por parte dos alunos participantes. O estudo também revelou que a narração exerce um impacto mais significativo do que o inicialmente previsto na motivação dos jogadores. Dessa forma, a construção de um enredo mais envolvente se mostra uma característica fundamental a ser considerada em futuras melhorias, com o objetivo de aumentar o engajamento dos usuários (COSTA et al., 2023).

Figura 1 – Captura de tela do jogo CodingJob



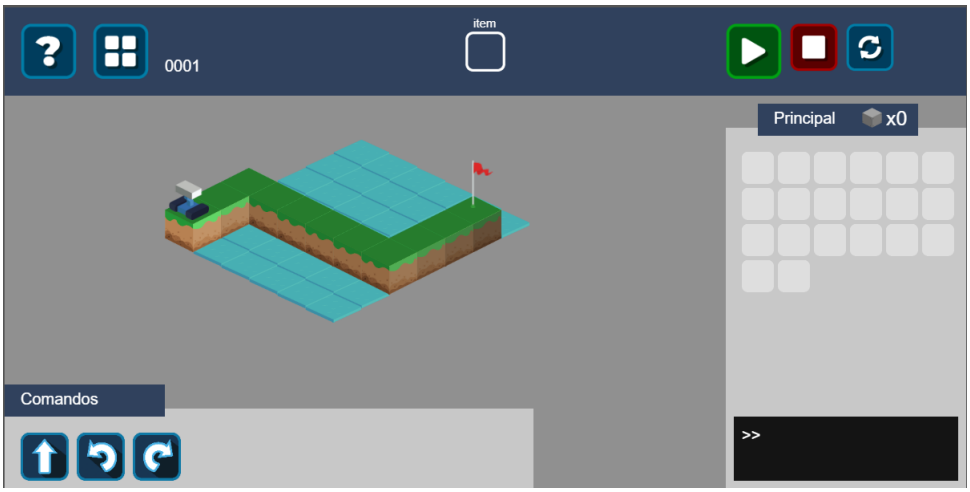
Fonte: (COSTA et al., 2023)

4.1.2 CodeBô: Design e avaliação de um puzzle game para o ensino de Estrutura de Dados

O intuito do projeto CodeBô (ARAUJO; SILVA, 2025) foi desenvolver um jogo digital isométrico de *puzzles* que se baseia na mecânica do *lightBot*, um jogo mobile educacional conceituado, mecânica esta que consiste em selecionar uma ordem de movimentos que o personagem deve executar para se mover até um local específico.

O jogo CodeBô utiliza essas mecânicas para ensinar conceitos como Pensamento Computacional, pilhas, filas e listas. (ARAUJO; SILVA, 2025)

Figura 2 – Captura de tela do jogo CodeBô

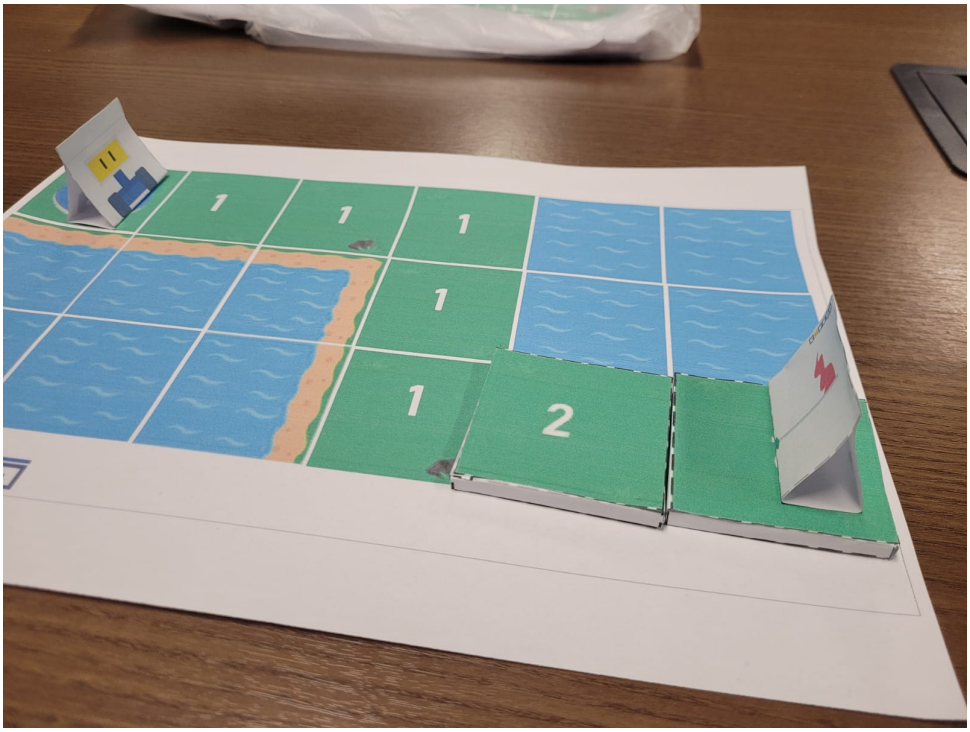


Fonte: (ARAUJO; SILVA, 2025)

4.1.3 CodeBo Unplugged: Um jogo desplugado para o ensino de Pilha

O jogo de tabuleiro CodeBo Unplugged (CERQUEIRA; SILVA; ARAUJO, 2023) foi desenvolvido com o intuito de ensinar a estrutura de dados Pilha a alunos do ensino fundamental de forma lúdica, utilizando elementos como robôs e mapas que aumentam em dificuldade de forma progressiva. (CERQUEIRA; SILVA; ARAUJO, 2023)

Figura 3 – Foto do tabuleiro de CodeBô Unplugged



Fonte: (CERQUEIRA; SILVA; ARAUJO, 2023)

4.1.4 Desenvolvimento de um Jogo para Auxílio no Ensino de Estruturas de Dados

Durante o trabalho de conclusão de curso intitulado “Desenvolvimento de um Jogo para Auxílio no Ensino de Estruturas de Dados”, foi desenvolvido um jogo digital mobile com o intuito de facilitar e auxiliar o ensino, a aprendizagem e a visualização dos conceitos de algoritmos de busca da disciplina de Estrutura de Dados.

A tecnologia utilizada para desenvolver este jogo foi a linguagem de programação Dart, em conjunto com o framework Flutter.

Por se tratar de um *quiz*, a abordagem de ensino é explicita. (GLATZ et al., 2023)

Figura 4 – Captura de tela do jogo para auxiliar em estrutura de dados

Fonte: (GLATZ et al., 2023)

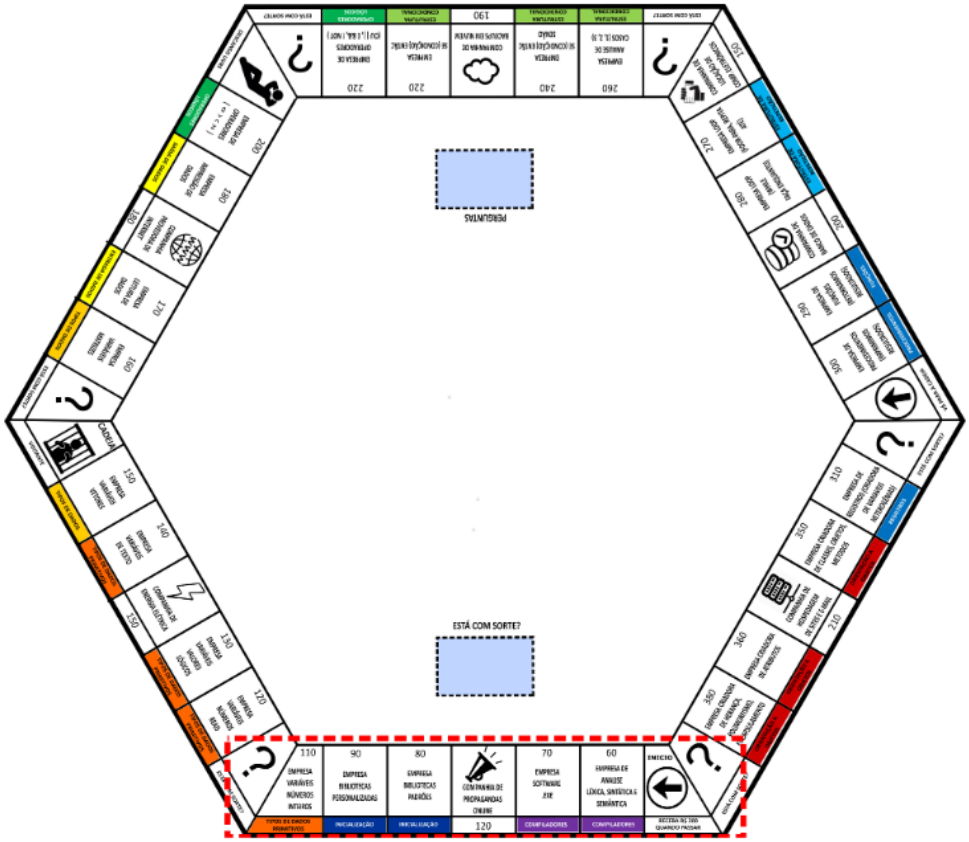
4.1.5 Prog-poly: Jogo de tabuleiro baseado no monopoly para ajudar nos estudos de linguagem de programação e engenharia de software

Durante o trabalho de pesquisa de mestrado Prog-poly (NASCIMENTO et al., 2022), foi desenvolvido um jogo de tabuleiro com o intuito de facilitar e auxiliar a aprendizagem de temas como linguagem de programação e engenharia de software.

Este jogo foi baseado na mecânica do clássico jogo de tabuleiro Monopoly, onde cada jogador deve comprar propriedades no tabuleiro. Entretanto, este jogo se diferencia pois para ter a oportunidade de comprar a propriedade, o jogador deve responder de forma correta perguntas a respeito de ILPC (Introdução Linguagem de Programação C) e, somente se acertar, poderá adquirir a propriedade, caso possua dinheiro suficiente. Ganha o jogador que possuir a maior quantidade de dinheiro e propriedades.

Este Jogo se caracteriza por ser um jogo com uma abordagem de ensino explicita, pois se trata de um *quiz*. (NASCIMENTO et al., 2022)

Figura 5 – Captura de tela do tabuleiro de Prog-poly



Fonte: (NASCIMENTO et al., 2022)

Por fim, foi efetuada uma comparação entre os trabalhos correlatos e este trabalho, demonstrada na Tabela 2.

4.1.6 Comparação dos artigos

Com base na análise dos cinco trabalhos selecionados, observa-se uma variedade de abordagens no uso de jogos sérios para o ensino de conceitos de programação. Cada proposta apresenta escolhas distintas quanto aos conceitos abordados, estilo visual, mecânicas de interação e a forma como os conteúdos educacionais são apresentados ao jogador.

A Tabela 2 a seguir apresenta uma síntese comparativa dos trabalhos analisados, destacando os aspectos centrais de cada proposta e os elementos recorrentes identificados entre eles.

Os critérios utilizados para a comparação incluem:

- **Jogo Digital (JD)** - Indica se o jogo é digital ou físico.
- **Conceitos de Estrutura de Dados (CED)** - Quais conceitos foram abordados:

Pilha (P)

Fila (F)

Lista (L)

Lista Encadeada (LE)

Lista Duplamente Encadeada (LDE)

Árvore Binária (AB)

Algoritmo de Busca (B)

Algoritmo de Ordenação (O)

- **Forma de Abordagem Educacional (Ensino)** - Define se o ensino é tratado de forma explícita ou implícita.
- **Estilo** - Representação visual do jogo.
- **Gênero** - Categoria do jogo.

Tabela 2 – Comparação entre os trabalhos relacionados

Trabalho	JD	CED	Ensino	Estilo	Gênero
CodingJob	Sim	-	Explícito	Simulador	*Puzzle*
CodeBô	Sim	F,L,P,B	Implícito	Isométrico	*Puzzle*
CodeBo Unplugged	Não	P	Implícito	Tabuleiro	*Puzzle*
AuxED	Sim	B	Explícito	P&C	*Puzzle*
Prog-poly	Não	-	Explícito	Tabuleiro	*Quiz*

Fonte: Autor

4.2 APLICATIVOS

Em uma busca realizada nas plataformas *Play Store* e *App Store*, para aplicativos móveis, e nas plataformas *Steam* e *Itch.io*, para jogos digitais, foram identificados jogos correlatos. Utilizaram-se as palavras-chave “Programação”, “Estrutura de Dados” e “Jogo Educacional”, resultando inicialmente em um total de 262 jogos encontrados.

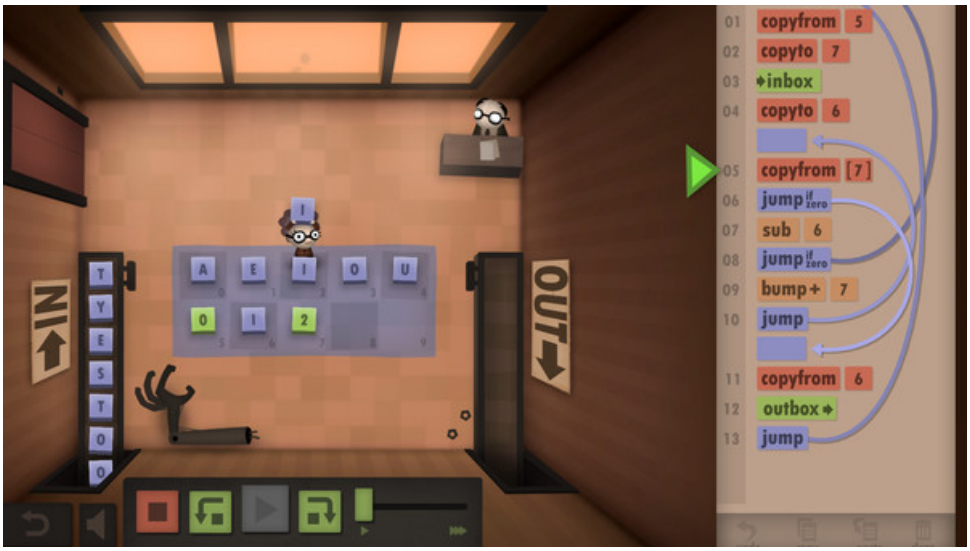
Após uma análise preliminar, 258 jogos foram descartados por não se enquadrarem nos critérios da pesquisa. A maioria consistia em questionários ou aplicativos de ensino que não se configuravam como jogos, enquanto outros apresentavam temáticas não relacionadas à programação ou não atendiam aos requisitos mínimos de qualidade. Com isso, restaram 4 jogos para a etapa de avaliação mais detalhada.

4.2.1 Human Resource Machine

Jogo de *puzzle* desenvolvido pela *Tomorrow Corporation*, tem como intuito ensinar lógica de programação de forma lúdica utilizando uma ambientação no meio corporativo, onde em cada fase o seu chefe lhe dá um trabalho e o seu objetivo é automatizar este trabalho programando pequenos trabalhadores para tal tarefa, e se você tiver sucesso nesta tarefa, será "promovido" para a próxima fase.

Com uma narrativa motivadora e envolvente, o jogo Human Resource Machine, se caracteriza como um jogo com ensino explicito. (Tomorrow Corporation, 2015)

Figura 6 – Captura de tela do jogo Human Resource Machine

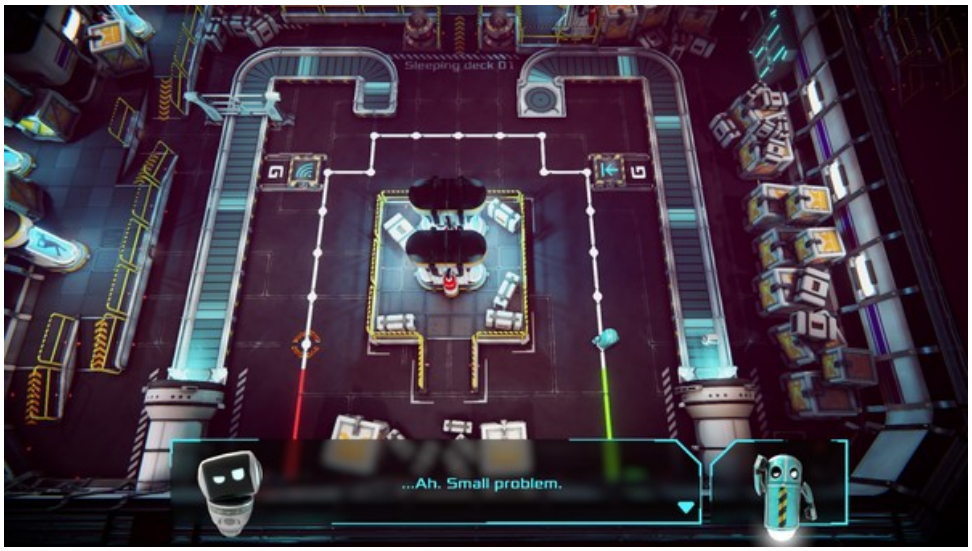


Fonte: (Tomorrow Corporation, 2015)

4.2.2 AlgoBot

Jogo de *puzzle* desenvolvido pela *Fishing Cactus*, tem como intuito ensinar algoritmos de forma lúdica utilizando uma ambientação futuristica, onde o jogador tera o papel de operador que deverá criar uma sequencia de commandos para o Algo Bot executar, seu objetivo é resolver puzzles para terminar a sua missão de reciclagem. (Fishing Cactus, 2018)

Figura 7 – Captura de tela do jogo AlgoBot



Fonte: (Fishing Cactus, 2018)

4.2.3 MOP'N SPARK

Jogo de *puzzle* desenvolvido pela *Omoplata Games*, tem como intuito ensinar algoritmos de forma lúdica utilizando uma ambientação fantasiosa, onde o jogador deverá criar uma sequencia de commandos para que os personagens Bepp e Gola alcancem os seus objetivos. (Omoplata Games, 2025)

Figura 8 – Captura de tela do jogo MOP'N SPARK

Fonte: (Omoplata Games, 2025)

4.2.4 Iron Ears: Data Structure

Iron Ears é um jogo do gênero *puzzle*, desenvolvido pela *NPC42 Games*, com o objetivo de ensinar conceitos de estruturas de dados de forma interativa. Ambientado em um universo fictício habitado por animais antropomorfizados, com inteligência e racionalidade comparáveis às humanas.

O jogo coloca o jogador no papel de um coelho humanóide encarregado de projetar e operar linhas de montagem de *Mechs*. Esses robôs são essenciais na resistência contra uma facção hostil que busca dominar o mundo.

A mecânica do jogo integra diretamente estruturas de dados clássicas, como listas, filas e pilhas, aos sistemas de produção, exigindo que o jogador aplique esses conceitos para resolver desafios e otimizar os processos de fabricação. (NPC42 Games, 2020)

Figura 9 – Captura de tela do jogo Iron Ears

Fonte: (NPC42 Games, 2020)

No tópico a seguir, é apresentada uma comparação entre os aplicativos selecionados.

4.2.5 Comparação dos aplicativos

A análise dos jogos selecionados revela uma variedade de abordagens no uso de jogos sérios voltados ao ensino de lógica de programação. No entanto, observa-se uma presença ainda limitada de propostas que exploram conceitos de estruturas de dados de maneira mais aprofundada.

A Tabela 3 apresenta uma síntese comparativa dessas iniciativas, destacando os principais elementos de cada proposta. Os critérios utilizados para esta comparação foram:

- **Conceitos de Estrutura de Dados (CED)** - Quais conceitos foram abordados:

  Pilha (P)

  Fila (F)

  Lista (L)

- **Forma de Abordagem Educacional (Ensino)** - Define se o ensino é tratado de forma explícita ou implícita.
- **Estilo** - Representação visual do jogo.
- **Gênero** - Categoria do jogo
- **Gratuito** - Indica se o jogo está disponível gratuitamente.
- **Avaliação** - Nota atribuída com base no sistema de avaliação da plataforma onde o jogo é disponibilizado.

Nos casos em que o jogo está disponível em plataformas que não possuem sistema de avaliação integrado (como o *itch.io*), ou em que não há avaliações registradas, a avaliação é considerada *indefinida*.

Para os jogos disponíveis na Steam, foi utilizada uma conversão aproximada baseada na porcentagem de avaliações positivas, conforme mostrado na Tabela 4.

Tabela 3 – Comparação entre os jogos relacionados

Trabalho	CED	Ensino	Estilo	Gênero	Gratuito	Avaliação
Human R.M.	-	Explícito	*Top Down*	*Puzzle*	Não	4.5
AlgoBot	-	Implícito	*Top Down*	*Puzzle*	Não	4.2
MOP'N SPARK	-	Implícito	Plataformer	*Puzzle*	Indefinido	Indefinido
Iron Ears	P,F,L	Implícito	*Drag & Drop*	*Puzzle*	Sim	Indefinido

Fonte: Autor

Tabela 4 – Conversão do Sistema de Avaliação da Steam para um sistema numeral de 1 a 5

Porcentagem de Avaliações Positivas	Avaliação Steam	Conversão Aproximada
90% - 100%	Extremamente positivas	4.5 - 5.0
70% - 89%	Muito positivas	3.5 - 4.4
40% - 69%	Neutras	2.0 - 3.4
10% - 39%	Muito Negativas	0.5 - 1.9
0% - 9%	Extremamente negativas	0.0 - 0.4

Fonte: Autor

O cálculo utilizado para a conversão é dado pela Equação 4.1

$$\text{Porcentagem de Avaliações Positivas} \times \frac{5}{100} \tag{4.1}$$

4.3 SÍNTESE DOS TRABALHOS RELACIONADOS

Com base na análise dos trabalhos e jogos relacionados, observa-se que, embora existam diversas iniciativas que utilizam jogos para o ensino de conceitos de programação, a maioria adota abordagens mais explícitas e centradas em gêneros como *puzzle* e *quizzes*.

Além disso, os conceitos de estruturas de dados são raramente abordados e quando são, costumam ser apresentados de forma segmentada, focando em apenas um ou dois tipos, como pilhas ou filas.

5 DESENVOLVIMENTO

Este capítulo descreve o processo de desenvolvimento do jogo sério seguindo a metodologia ENgAGED previamente descrita na subseção 3.2.3. O desenvolvimento foi estruturado em fases práticas, com foco no que foi efetivamente implementado durante este trabalho.

5.1 ANÁLISE DO JOGO E LEVANTAMENTO DE REQUISITOS

Os requisitos foram estruturados em duas categorias: funcionais e não-funcionais.

5.1.1 Requisitos Funcionais

Os requisitos funcionais definem as funcionalidades que o sistema deve implementar. Esses requisitos incluem aspectos como manipulação de inventários, sistema de combate, progresso do jogador, salvamento de partida e geração de feedback educacional.

Tabela 5 – Requisito Funcional 01

Identificador	**Nome**	**Prioridade**
FR-01	Ponto de controle	Médio
Descrição: O sistema deve possuir um ponto de controle no meio de cada fase, onde o usuário poderá retornar caso falhe.		

Fonte: Autor

Tabela 6 – Requisito Funcional 02

Identificador	**Nome**	**Prioridade**
FR-02	Manipular inventários	Alta
Descrição: O sistema deve permitir que os usuários manipulem seus inventários por meio de ações.		

Fonte: Autor

5.1.2 Requisitos Não-Funcionais

Os requisitos não-funcionais especificam características de qualidade e restrições técnicas. Esses requisitos abrangem aspectos como performance, compatibilidade, segurança, mantibilidade e acessibilidade.

Tabela 7 – Requisito Não Funcional 01

Identificador	**Nome**	**Categoria**
NFR-01	Frames por segundo estáveis	Performance
Descrição: O sistema deve se manter em um taxa de atualização constante e acima de 30 frames por segundo.		

Fonte: Autor

Tabela 8 – Requisito Não Funcional 02

Identificador	**Nome**	**Categoria**
NFR-02	Múltiplos idiomas	Usabilidade
Descrição: O sistema deve possuir tradução para o português e para o inglês.		

Fonte: Autor

5.2 CONCEPÇÃO DO JOGO

Esta seção apresenta o processo de concepção do jogo sério. Aqui são descritos os principais elementos que compõem o jogo, incluindo seus objetivos, narrativa, regras, mecânicas, elementos visuais, sistema de pontuação e feedback educacional, estabelecendo as bases conceituais para o desenvolvimento apresentado nas seções seguintes.

Objetivos do Jogo

O Jogador tem como objetivo narrativo recuperar a sua pesquisa a respeito da pedra filosofal

Gênero e Perspectiva

O jogo é um **platformer 2D**[1] com perspectiva *side-scrolling*[2]. Essa escolha baseia-se na familiaridade do público geral com clássicos como Super Mario Bros e Mega Man, facilitando a adoção do jogo e tornando a experiência intuitiva.

Temática e Narrativa

A narrativa é ambientada em um universo de **alquimia** mística e medieval. O protagonista é um alquimista caracterizado como um doutor da peste que teve sua pesquisa sobre a **pedra**

[1] Gênero de jogo plataforma em duas dimensões
[2] Estilo de *gameplay* onde o jogador se move horizontalmente e a tela o segue.

filosofal roubada por um alquimista rival enquanto viajava para a capital para apresentá-la ao Círculo dos Alquimistas (Grupo de estudiosos importantes). Seu objetivo é recuperar a pesquisa, navegando por diferentes regiões e enfrentando adversários durante a sua jornada.

Essa narrativa fornece contexto e motivação para as ações do jogador, aumentando o engajamento e criando imersão. O enredo também justifica a presença dos elementos alquímicos como mecanismo central de combate.

Regras

As regras do jogo definem as condições que orientam o comportamento do jogador e do sistema, estabelecendo os limites e critérios que estruturam a experiência interativa. Elas determinam as ações permitidas, as consequências de erros e acertos, e os parâmetros que conduzem à progressão ou ao término da fase.

No contexto deste projeto, as regras estão intrinsecamente relacionadas aos objetivos educacionais e narrativos do jogo. O jogador deve explorar o ambiente, realizar combinações de elementos alquímicos e superar desafios seguindo um conjunto de condições previamente definidas. Para progredir, é necessário alcançar o final de cada fase, vencendo os obstáculos e inimigos dispostos ao longo do percurso.

Entre as principais regras implementadas, destacam-se:

- O jogador perde pontos de vida ao realizar combinações incorretas de elementos alquímicos ou ao entrar em contato com inimigos e seus respectivos projéteis.
- Cada inventário possui um limite máximo de três elementos;
- A conclusão da fase ocorre mediante a interação do jogador com um objeto específico localizado ao final do nível;
- O tempo total de conclusão influencia diretamente a pontuação final obtida;
- O jogador é penalizado por mortes e erros recorrentes, refletindo os princípios de avaliação formativa e somativa;
- Ao atingir zero pontos de vida, o jogador perde uma tentativa e retorna ao ponto de controle anterior, recuperando integralmente sua vida;
- Caso o jogador fique sem tentativas restantes e volte a atingir zero pontos de vida, a fase é reiniciada, resultando na perda de todo o progresso obtido.

Mecânicas

A seguir, são aprofundadas as principais mecânicas desenvolvidas seguindo, majoritariamente, os requisitos funcionais previamente apresentados na subseção 5.1.1, sendo divididas entre mecânicas centrais e mecânicas genéricas.

Mecânicas Centrais

As mecânicas centrais correspondem aos elementos de jogabilidade diretamente relacionados aos objetivos principais do projeto e à aplicação dos conceitos de estrutura de dados. Elas representam o núcleo funcional do jogo, onde se concentram as interações que concretizam a proposta educacional do desenvolvimento. Na Tabela 9, a seguir, essas mecânicas são apresentadas e descritas, evidenciando como os princípios de estrutura de dados foram incorporados à dinâmica do jogo.

Tabela 9 – Mecânicas Centrais

Mecânica	**Descrição**
Manipulação de Inventários	Cada inventário é estruturado com base em uma estrutura de dados específica, permitindo, em casos específicos, operações como inserção, remoção, ordenação e movimentação de ponteiros. Essas operações preservam as propriedades de cada estrutura, proporcionando uma experiência prática de seus comportamentos.
Geração de Elementos	Permite ao jogador criar novos elementos mediante o consumo de mana. Os elementos gerados são automaticamente alocados no primeiro espaço livre de um dos inventários, seguindo a ordem da esquerda para a direita.
Consumíveis	Itens de uso estratégico que possibilitam realizar ações diretamente sobre os inventários, como inserir, remover ou ordenar elementos, servindo como ferramentas para reforçar o aprendizado sobre as operações das estruturas de dados.
Mistura de Elementos	Elementos do mesmo tipo podem ser combinados para gerar ataques. O dano causado varia conforme a quantidade de elementos iguais utilizados na combinação. A realização da mistura exige a remoção prévia dos elementos dos inventários correspondentes.
Fraquezas	Cada inimigo apresenta vulnerabilidades específicas a determinados elementos. O jogador deve identificar e combinar corretamente os elementos para maximizar o dano e avançar com eficiência.
Sistema de Pontuação	Calcula o desempenho do jogador com base em múltiplos fatores, como acertos e erros nas combinações, número de tentativas e tempo total para concluir a fase, promovendo uma avaliação quantitativa da performance.
Livro de Tutoriais	aqui são armazenados de forma explícita os tutoriais do jogo para que, caso o jogador se perca, ele possa revisar algo que não tenha entendido.

Fonte: Autor

Mecânicas Genéricas

As mecânicas genéricas compreendem o conjunto de funcionalidades fundamentais que sustentam a jogabilidade e garantem a coerência da experiência interativa. Elas formam a base sobre a qual as demais dinâmicas e desafios são construídos, assegurando que o jogador tenha controle responsivo e feedbacks claros de suas ações. Na Tabela 10, a seguir, essas mecânicas são apresentadas e descritas de forma a evidenciar seu papel na estrutura fundamental do jogo.

Tabela 10 – Mecânicas Genéricas

Mecânica	**Descrição**
Movimentação 2D	Permite ao jogador deslocar o personagem em um plano bidimensional, controlando direções horizontais, como andar para a esquerda e direita.
Pulo Responsivo	Define uma mecânica de salto com resposta imediata aos comandos do jogador, garantindo sensação de controle preciso e previsível.
Corrida	Possibilita ao personagem movimentar-se em velocidade aumentada mediante a sustentação de um comando específico, ampliando o dinamismo do deslocamento.
Pontos de Controle	Permitem salvar o progresso do jogador em determinados trechos do nível, reduzindo a frustração causada por reinícios completos e otimizando o fluxo de tentativa e erro.
Tentativas Limitadas	Estabelecem um número máximo de vidas ou tentativas antes do reinício total do nível, criando tensão adicional e incentivando o aprendizado a partir dos erros.
Pontos de Vida	Representam a resistência do personagem diante de danos. Quando os pontos de vida chegam a zero, o jogador sofre penalidades, como perda de progresso ou reinício do nível.
Mana	Recurso utilizado para a geração de elementos, cuja gestão estratégica é necessária para equilibrar poder ofensivo e conservação de energia durante os desafios.
Pulo Duplo Situacional	Permite ao personagem realizar um segundo salto no ar apenas após coletar um orbe específico durante o salto.
Capacidade de Aparar Certos Ataques	O jogador pode bloquear determinados projéteis inimigos utilizando um projétil próprio, neutralizando o dano e evitando impactos diretos.

Fonte: Autor

Elementos do Jogo

Esta seção apresenta os principais elementos que compõem a estrutura e a experiência interativa do jogo. Cada componente foi projetado para contribuir com a jogabilidade, a narrativa e a ambientação, formando um ecossistema coeso entre mecânica e estética. A Tabela 11 descreve os objetos implementados e suas respectivas funções no contexto do jogo.

Tabela 11 – Elementos do jogo e suas descrições

Elemento	Descrição
Entidades Jogáveis e Interativas	
Jogador	Personagem principal, um alquimista caracterizado como um *plague doctor*, responsável por realizar combinações alquímicas e interagir com o ambiente.
Consumíveis	Itens coletáveis que restauram atributos ou concedem vantagens temporárias.
Fogueira	Ponto de controle utilizado para salvar o progresso e restabelecer o estado do jogador.
Gato	Personagem auxiliar que fornece dicas e interage em momentos específicos, contribuindo para o enredo e orientação do jogador.
Bandeira	Objeto que indica o término da fase, sendo necessário interagir com ele para que a pontuação final seja exibida.
Inimigos	
Gosma	Inimigo básico que se desloca lateralmente em patrulha constante.
Cogumelo	Inimigo que persegue o jogador ao entrar em seu alcance e libera esporos quando próximo.
Olho Alado	Inimigo aéreo que ataca à distância disparando projéteis.
Ambiente e Cenário	
Objetos de ambientação	Elementos decorativos como grama, arbustos, cercas e rochas.
Partículas	Efeitos visuais dinâmicos como poeira, fumaça, chuva e vagalumes, que enriquecem a atmosfera.
Blocos do cenário	Estruturas modulares que compõem o solo e as plataformas do ambiente de jogo.
Background	Representa o céu e o horizonte, compondo o plano mais distante do cenário.
Background Distante de Árvores	Camada intermediária que exibe árvores ao longe.
Background Próximo	Camada mais próxima ao jogador, contendo árvores e vegetação detalhada.
Camada Frontal de Árvores	Elementos que passam à frente do jogador, criando sensação de profundidade.
Camada Frontal de Rochas e Arbustos	Sombras e detalhes em primeiro plano que reforçam o volume do cenário.
Interface e Menus	
HUD	Interface principal que exibe informações como pontos de vida, tentativas, inventário, livro de tutoriais e botão de pausa.
Mensagens Temporários	Mensagens contextuais apresentadas de forma breve, utilizadas para instruções rápidas.
Menu Principal	Tela inicial que permite o acesso às demais seções do jogo.
Menu de Fases	Interface de seleção de níveis disponíveis.
Menu de Pause	Tela de pausa com acesso às opções e retorno à partida.
Menu de Opções	Acesso às opções de volume e idioma.

Continua na próxima página

Elemento	Descrição
Menu de Volume	Controle individual dos níveis sonoros do jogo.
Menu de Linguagem	Seleção de idioma.
Menu de Saída	Opção para encerrar o jogo.

Fonte: Autor

Cabe destacar que nem todos os elementos desenvolvidos nesta etapa estão contemplados na Tabela 11 acima. Alguns objetos secundários e recursos complementares, implementados com o objetivo de aprimorar a imersão e a fluidez da experiência, foram omitidos por não exercerem papel central na dinâmica do jogo.

Pontuação

O sistema de pontuação foi desenvolvido para oferecer uma forma objetiva de mensurar o desempenho do jogador ao longo das partidas. Ao término de cada fase, o jogo apresenta um **feedback somativo** por meio de um relatório de desempenho que sintetiza os resultados obtidos. A pontuação total é calculada a partir de múltiplos fatores, como o número de combinações corretas e incorretas, mortes e tempo total de conclusão, permitindo avaliar tanto a eficiência quanto a precisão das ações realizadas.

Esse cálculo é processado de forma automatizada, garantindo consistência na avaliação e eliminando subjetividades na análise do desempenho. A pontuação final é determinada a partir de uma combinação de fatores relacionados ao tempo de conclusão da fase, ao número de infusões corretas e incorretas, e à quantidade de mortes do jogador.

A fórmula utilizada para o cálculo da pontuação final é expressa pela Equação 5.1, em que cada componente contribui de forma ponderada para o resultado geral:

$$S = (1000 - 10\,t) + (100\,c - 25\,e) - 150\,d \tag{5.1}$$

em que:

- S representa a pontuação final obtida pelo jogador;
- t é o tempo total de conclusão da fase (em segundos);
- c é o número de infusões corretas realizadas;
- e é o número de infusões incorretas;
- d corresponde ao número de mortes ocorridas.

Feedback Educacional

O sistema de *feedback* educacional foi concebido para reforçar o aprendizado de forma contínua e integrada à experiência de jogo. Sua implementação busca promover uma aprendizagem ativa, na qual o jogador recebe respostas imediatas às suas ações, sem comprometer a fluidez do *gameplay*. Essa dinâmica permite que o processo de experimentação e correção ocorra de maneira natural, favorecendo a compreensão dos efeitos de cada escolha e das relações entre os conceitos de estrutura de dados representados nas mecânicas do jogo.

Os diferentes tipos de *feedback* foram projetados para atuar em múltiplos níveis de interação, variando de respostas visuais imediatas a avaliações formativas sutis, conforme sua função no processo de aprendizagem. Quando o jogador realiza uma combinação incorreta, o sistema apresenta efeitos sonoros e um **feedback visual de erro**, evidenciado por mudanças de cor no personagem e efeitos de penalidade, como a perda de pontos de vida. Esse retorno instantâneo sinaliza a falha sem interromper a continuidade da ação, permitindo que o jogador aprenda com o erro por meio da tentativa e erro.

Por outro lado, ao executar combinações corretas de elementos alquímicos, o jogador recebe uma **avaliação formativa implícita**, expressa por efeitos sonoros e visuais que indicam sucesso na ação. Esse tipo de retorno atua como reforço positivo, incentivando a experimentação e consolidando o aprendizado por meio da prática direta.

Complementando esses retornos imediatos, o **feedback somativo** apresentado ao final de cada fase, descrito na seção anterior, integra o processo avaliativo de forma global, permitindo ao jogador refletir sobre seu desempenho e identificar aspectos a melhorar. Assim, o sistema de *feedback* atua de maneira articulada, combinando respostas instantâneas e avaliações finais para sustentar um ciclo contínuo de aprendizado.

5.3 DESIGN DO JOGO

Esta seção apresenta o processo de concepção visual e estrutural do jogo, abrangendo desde a definição estética dos elementos gráficos até a organização espacial e narrativa das fases. O design do jogo foi orientado pelos requisitos funcionais estabelecidos na etapa de desenvolvimento, buscando aliar clareza visual, coerência temática e funcionalidade interativa. São descritos os principais componentes visuais, como personagens, inimigos, cenários e interfaces, bem como a modelagem do nível inicial e dos diálogos que compõem a experiência lúdica e pedagógica proposta.

5.3.1 Definir Game Engine

Dentre as diversas *game engines* disponíveis atualmente, realizou-se uma análise comparativa considerando critérios como facilidade de uso, documentação, suporte multiplataforma, performance, comunidade ativa e adequação aos objetivos educacionais do projeto. A partir desses critérios, foram selecionadas três alternativas principais para avaliação: **Godot**, **Unreal Engine** e **Unity**.

Tabela 12 – Comparativo entre Game Engines

Critério	Godot Engine	Unity	Unreal Engine
Licença	MIT (código aberto)	Proprietária (gratuita para uso educacional)	Proprietária (royalties comerciais)
Facilidade de uso	Alta	Alta	Moderada
Performance	Boa	Muito boa	Muito boa
Suporte a WebGL	Estável, mas limitado	Amplo e otimizado	Limitado e com alto custo de compilação
Comunidade e Documentação	Em crescimento	Ampla e consolidada	Consolidada, porém mais voltada a estúdios *Triple A*[3]

Fonte: Autor, (Godot Engine Community, 2025; Unity Technologies, 2025; Epic Games, 2025)

A **Godot** apresenta-se como uma ferramenta de código aberto, leve e altamente personalizável, com uma comunidade em crescimento constante. Seu principal atrativo está na licença permissiva e na facilidade de prototipagem rápida. No entanto, limitações no desempenho gráfico e na maturidade de certos recursos, especialmente para exportações WebGL, podem representar desafios em projetos que demandam maior desempenho. (Godot Engine Community, 2025)

A **Unreal Engine**, por outro lado, é reconhecida por seu poder gráfico e ferramentas robustas voltadas a produções de grande escala. Entretanto, sua curva de aprendizado mais acentuada, o tamanho considerável das compilações e a demanda por hardware mais potente a tornam menos adequada a projetos de caráter educacional ou voltados à execução em navegadores. (Epic Games, 2025)

Por fim, a **Unity** destaca-se como uma solução equilibrada entre flexibilidade e desempenho. Sua arquitetura modular, ampla documentação, integração multiplataforma e suporte consolidado ao WebGL permitem o desenvolvimento de aplicações interativas com boa performance e fácil implantação em ambientes educacionais. Além disso, a curva de aprendizado mais suave e o vasto ecossistema de recursos contribuem para a agilidade no processo de prototipagem e implementação. (Unity Technologies, 2025)

Dessa forma, a *Game Engine* selecionada para o desenvolvimento do projeto foi a **Unity**, por sua robustez, suporte multiplataforma e capacidade de criar jogos com desempenho eficiente, alinhando-se às necessidades tanto técnicas quanto pedagógicas desta pesquisa.

[3] Estúdios que desenvolvem jogos com os maiores orçamentos da indústria.

5.3.2 Produzir imagens dos elementos do jogo

Nesta etapa, são apresentadas as representações visuais dos principais elementos descritos anteriormente na Tabela 11. As imagens ilustram a aparência, o estilo artístico e o papel de cada componente dentro do ambiente do jogo, complementando as descrições textuais e auxiliando na compreensão da composição estética e funcional do projeto.

A Figura 10 apresenta o personagem principal, responsável pelas misturas alquímicas e pela interação com o ambiente. Seu design foi inspirado nos Médicos da Peste Negra.

Figura 10 – Personagem principal do jogo

Fonte: Autor

O personagem mostrado na Figura 11 atua como companheiro e guia do jogador, aparecendo de forma mística à sua frente. Sua concepção visual baseia-se no Gato de Cheshire da obra *Alice no País das Maravilhas* (CARROLL, 1865).

Figura 11 – Companheiro felino

Fonte: Autor

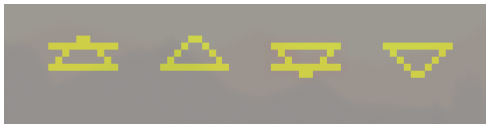
A fogueira da Figura 12, que funciona como ponto de controle e restauração, segue a estética das fogueiras presentes em jogos do gênero *soulslike*[4]. O conceito geral foi inspirado por obras como *Dark Souls* (FromSoftware, 2011).

Figura 12 – Fogueira, ponto de controle do jogador

Fonte: Autor

Os elementos alquímicos Ar, Fogo, Terra e Água, apresentados na Figura 13, foram baseados em símbolos tradicionais da alquimia medieval. (WIKIPEDIA, 2025)

Figura 13 – Elementos alquímicos utilizados nas combinações do jogo

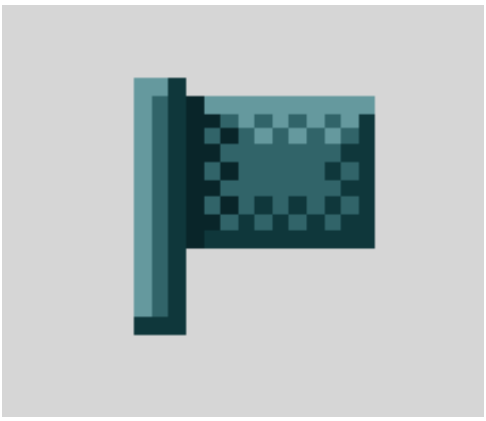


Fonte: Autor

O objeto que marca o fim da fase, representado na Figura 14, toma como referência elementos clássicos de jogos de plataforma, como a bandeira de *Super Mario Bros.* (Nintendo, 1985) e o letreiro de conclusão de fase de *Sonic the Hedgehog* (SEGA, 1991).

[4] O termo *soulslike* refere-se a jogos inspirados em características da série *Dark Souls*, como alta dificuldade, combate tático e progressão punitiva.

Figura 14 – Bandeira sinalizando o término da fase



Fonte: Autor

Os principais inimigos enfrentados na fase estão ilustrados na Figura 15.

Figura 15 – Inimigos presentes no jogo

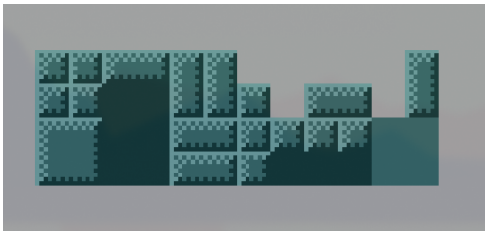
Fonte: Autor

Da esquerda para a direita, são apresentados:

1) Gosma, inspirada em criaturas similares presentes em franquias como *Final Fantasy* (Square Enix, 1987) e *Dragon Quest* (Square Enix, 1986);
2) Cogumelo, cuja estética remete a inimigos clássicos de *Super Mario Bros.* (Nintendo, 1985) e referências visuais de *Hollow Knight* (Team Cherry, 2017);
3) Olho Alado, baseado em criaturas recorrentes de séries como *The Legend of Zelda* (Nintendo, 1986) e *Final Fantasy* (Square Enix, 1987).

Os blocos modulares que compõem o terreno e as plataformas do cenário estão representados na Figura 16.

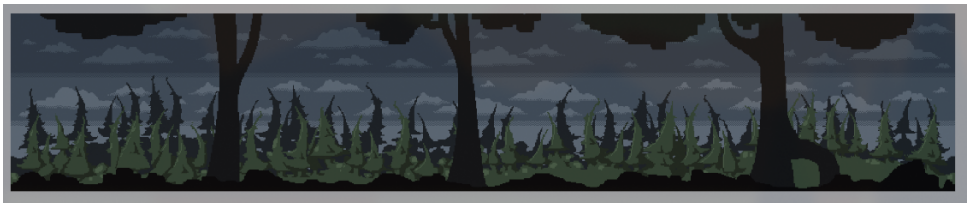
Figura 16 – Blocos do cenário utilizados na construção dos níveis



Fonte: Autor

As diferentes camadas de paralaxe[5] utilizadas para compor a profundidade visual do ambiente estão apresentadas na Figura 17.

Figura 17 – Camadas de Parallax que compõem a profundidade visual do cenário



Fonte: Autor

Os consumíveis disponíveis ao jogador, responsáveis por fornecer vantagens temporárias ou restaurar atributos, são mostrados na Figura 18.

Figura 18 – Consumíveis disponíveis ao jogador durante a partida

Fonte: Autor

Da esquerda para a direita, os consumíveis são:

1) Consumível de Inserção;
2) Consumível de Remoção;
3) Consumível de Ordenação;
4) Consumível de Cura;
5) Consumível de Mana;
6) Consumível de Vida Extra.

Por fim, a **HUD** apresenta as principais informações de jogabilidade, como inventário, vida, mana e botões de interação. A Figura 19 mostra a interface numerada.

[5] Paralaxe é uma técnica de design visual em que camadas do cenário se movem em velocidades distintas, criando a ilusão de profundidade.

Figura 19 – Interface de jogo com elementos numerados

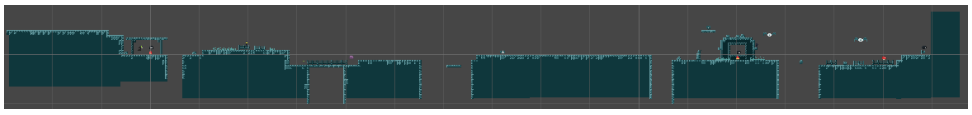
Fonte: Autor

1) Indicador de tentativas restantes;
2) Pontos de vida do jogador;
3) Barra de mana;
4) Botão de pausa;
5) Livro de tutoriais;
6) Espaço do consumível ativo;
7) Indicador da saída da pilha;
8) Indicador da saída da fila;
9) Indicador da saída da lista;
10) Indicador da próxima inserção.

5.3.3 Modelar o jogo

Nesta etapa, o processo de modelagem do jogo concentrou-se na definição do nível inicial e na estruturação dos diálogos que compõem a experiência narrativa e pedagógica. O objetivo foi estabelecer um ambiente introdutório que permitisse ao jogador compreender as mecânicas básicas, interagir com os principais elementos do cenário e familiarizar-se com a proposta do jogo.

Figura 20 – Design do Nível Inicial



Fonte: Autor

5.4 IMPLEMENTAÇÃO DO JOGO

Esta seção apresenta a implementação do jogo desenvolvido na *game engine* **Unity**. Inicialmente, são contextualizados os principais conceitos técnicos da *engine* utilizados como base para a construção dos sistemas implementados. Em seguida, **FIX: descrevem-se os principais elementos listados na Tabela 11 e discute-se como as mecânicas centrais e genéricas, definidas nas Tabela 9 e Tabela 10, foram convertidas em funcionalidades operacionais** . Dessa forma, evidencia-se a transição do design conceitual para sua materialização prática, demonstrando como cada decisão de design contribuiu para a composição do sistema final.

Conceitos Fundamentais da Unity

Antes de detalhar a implementação dos componentes do jogo, torna-se necessário contextualizar alguns conceitos fundamentais da Unity, os quais orientam a estruturação dos *scripts*, a organização dos objetos e o comportamento do sistema em tempo de execução. A Unity adota um modelo arquitetural baseado em componentes, no qual cada objeto do jogo é composto por módulos independentes responsáveis por sua lógica, aparência, física ou interação.

No centro desse modelo está a classe `MonoBehaviour`, da qual derivam a maioria dos scripts comportamentais. Ela define o ciclo de vida padrão utilizado pela *engine*, permitindo que o desenvolvedor implemente métodos executados automaticamente em momentos específicos da execução. Entre esses métodos, destacam-se:

`Start()`: Executado uma única vez na inicialização do objeto, sendo empregado para configurar estados iniciais, referências e variáveis de controle.

`Update()`: Chamado a cada quadro (frame), sendo adequado para lógicas dependentes da taxa de atualização, como leitura de entrada, animações e verificações contínuas.

`FixedUpdate()`: Executado em intervalos fixos, independentemente da taxa de quadros, sendo indicado para cálculos de física e interações que exigem maior estabilidade numérica.

A Unity também integra um sistema de física robusto, por meio do componente `Rigidbody2D`, que permite que objetos sejam afetados por forças, gravidade e colisões. A combinação entre `Rigidbody2D` e colisores define como cada objeto interage fisicamente com o ambiente, possibilitando comportamentos essenciais para movimentação, detecção de obstáculos e mecânicas de combate.

Esses conceitos estruturam toda a implementação do jogo e fundamentam a criação dos sistemas apresentados nas subseções seguintes.

Implementação do Jogador

A implementação teve início pelo componente central da experiência: o jogador. Foram desenvolvidos diversos *scripts* em **C#**, responsáveis pelas mecânicas de movimentação, combate e interação com os inventários.

A Código 1 apresenta o trecho referente à movimentação do jogador, no qual são aplicadas diretamente as funcionalidades do ciclo de vida da Unity e seus componentes físicos.

```
using UnityEngine;

public class PlayerMove : MonoBehaviour
{
    private Rigidbody2D rb;

    private void FixedUpdate()
    {
        // Atualiza o movimento horizontal do jogador
        Move(Input.GetAxis("Horizontal"));

        // Inicia o salto se o jogador estiver
        // no chão e pressionar o botão.
        if (Input.GetButtonDown("Jump") && isGrounded)
            Jump();
        // Reduz a força do salto caso o botão
        // seja solto durante a ascensão.
        else if (Input.GetButtonUp("Jump") && !isGrounded)
            CancelJump();
    }

    private void Jump()
    {
        // Aplica a velocidade vertical
        // correspondente ao salto.
        rb.linearVelocity = new Vector2(
          rb.linearVelocity.x,
          jumpForce
        );
    }

    private void CancelJump()
    {
        // Corta parte da velocidade vertical para
        // permitir variação na altura do salto
        // conforme o tempo de pressionamento do botão.
        rb.linearVelocity = new Vector2(
          rb.linearVelocity.x,
          rb.linearVelocity.y * 0.5f
        );
    }

    private void Move(float input)
    {
        // Ajusta a velocidade horizontal do jogador
        // de acordo com a entrada recebida.
        rb.linearVelocity = new Vector2(
          input * speed,
          rb.linearVelocity.y
        );
    }
}
```

Código 1 – Trecho Simplificado do Algoritmo de Movimentação do Jogador

Com o objetivo de estruturar e padronizar o funcionamento dos inventários, representados por diferentes estruturas de dados, foi desenvolvida a interface exibida no Código 2. Essa interface possibilita a manipulação genérica das estruturas, permitindo alterar tanto o tipo quanto a quantidade de inventários utilizados sem comprometer o restante do sistema.

```
1  public enum InventoryType { Stack, Queue, LinkedList }
2
3  public interface IInventory
4  {
5    public InventoryType Type { get; }
6    public int Count { get; }
7    public int MaxSize { get; }
8    public bool IsFull();
9    public void Clear();
10   public void Sort();
11   public Element[] ToArray();
12 }
```

Código 2 – Interface dos Inventários

Essa interface é utilizada no Código 3, responsável pela lógica de combate do jogador. Nesse módulo, as operações de ataque integram-se às operações de remoção e combinação dos inventários, conectando diretamente a mecânica de combate às representações das estruturas de dados.

```
using UnityEngine;

public class PlayerCombat : MonoBehaviour
{
  public IInventory[] inventories;

  // Lista temporária contendo os elementos selecionados pelo jogador
  // para compor uma combinação (conjuração).
  public List<Element> castingList = new List<Element>();

  private void Update()
  {
    HandleLinkedListSelection();
    HandleCastingInput();
  }

  private void HandleLinkedListSelection()
  {
    // Responsável por alterar o elemento atualmente selecionado
    // na Lista Encadeada.
  }

  private void HandleCastingInput()
  {
    // Adiciona elementos à conjuração conforme o inventário selecionado.
    if (Input.GetKeyDown(KeyCode.Alpha1))
      AddFromInventory(InventoryType.Stack);
    else if (Input.GetKeyDown(KeyCode.Alpha2))
      AddFromInventory(InventoryType.Queue);
    else if (Input.GetKeyDown(KeyCode.Alpha3))
      AddFromInventory(InventoryType.LinkedList);

    // Finaliza a conjuração quando o jogador confirma a combinação.
    else if (Input.GetKeyDown(KeyCode.Return)) FinishCasting();
  }

  private void AddFromInventory(InventoryType type)
  {
    // Remove um elemento da estrutura de dados correspondente
    // (Pilha, Fila ou Lista Encadeada) e o adiciona à lista de conjuração.
  }

  private void FinishCasting()
  {
    // Executa a conjuração utilizando os elementos selecionados,
  }
}
```

Código 3 – Trecho Simplificado do Algoritmo de Combate do Jogador

A lógica de geração e inserção de elementos nos inventários, essencial para o funcionamento das mecânicas centrais, é apresentada no Código 4.

```
using UnityEngine;

public class GenerateElement : MonoBehaviour
{
  // Variáveis e configuração inicial do script

  private void Update()
  {
    bool holdingDown = Input.GetAxis("Vertical") < 0;

    // Quando o jogador mantém o comando para baixo enquanto está no chão,
    // inicia o processo de carregamento da habilidade de geração.
    if (holdingDown && grounded.IsGrounded())
    {
      chargeTimer += Time.deltaTime;
      curMana -= Time.deltaTime;

      // Quando o tempo acumulado atinge o valor de recarga (cooldown),
      // tenta gerar um novo elemento.
      if (chargeTimer >= cooldown)
      {
        TryGenerateElement();
        chargeTimer = 0f;
      }
    }
    // Caso o jogador não esteja carregando,
    // a mana é regenerada até o limite máximo.
    else if (curMana < maxMana)
    {
      curMana += Time.deltaTime * ManaRegen;
    }
  }

  private bool TryGenerateElement()
  {
    // Tenta inserir um elemento aleatório
    // em um dos inventários disponíveis.
    // Se o inventário atual estiver cheio, tenta o próximo.
    // Caso todos estejam indisponíveis,
    // a geração é considerada malsucedida.
  }
}
```

Código 4 – Trecho Simplificado do Algoritmo de Geração de Elementos do Jogador

5.4.1 Implementação dos Inimigos

A implementação dos inimigos envolveu a criação de sistemas de movimentação, detecção de colisões, aplicação de dano e controle de comportamentos específicos para cada tipo de criatura. Além disso, foi incorporado o sistema de fraquezas, que relaciona cada inimigo a tipos específicos de elementos mais eficazes contra ele, conforme exemplificado no Código 5. Esse recurso integra o sistema de combate às estruturas de dados manipuladas pelo jogador.

```
using UnityEngine;

public class EnemyDamage : MonoBehaviour
{
  private void Start()
  {
    weaknesses = GenerateRandomWeaknesses();
  }

  private Element[] GenerateRandomWeaknesses()
  {
    // Gera um conjunto de elementos
    // que representam as fraquezas do inimigo.
    // A seleção ocorre de forma aleatória,
    // permitindo variação entre inimigos.
  }

  private void OnTriggerEnter2D(Collider2D collider)
  {
    // Verifica se o objeto que colidiu corresponde a algum elemento
    // ao qual o inimigo possui fraqueza. Caso corresponda, aplica dano.
    foreach (var weakness in weaknesses)
    {
      if ((weakness == Element.FIRE  && collider.CompareTag("Fire"))  ||
          (weakness == Element.WATER && collider.CompareTag("Water")) ||
          (weakness == Element.AIR   && collider.CompareTag("Air"))   ||
          (weakness == Element.EARTH && collider.CompareTag("Earth")))
      {
        // Obtém o valor de dano do projétil e aplica ao inimigo.
        int damage = collider.GetComponent<ProjectileController>().damage
        health.TakeDamage(damage);
        return;
      }
    }
  }
}
```

Código 5 – Trecho Simplificado do Algoritmo de Fraquezas dos Inimigos

5.4.2 Implementação da Pontuação

O cálculo da pontuação final ao concluir uma fase é apresentado no Código 6 e fundamenta-se na Equação 5.1. O algoritmo considera o tempo total, o número de combinações corretas, combinações incorretas e mortes, consolidando a avaliação da performance do jogador de forma objetiva e alinhada às mecânicas do jogo.

```
public class Score : MonoBehaviour
{
  // ...

  private double CalculateFinalScore(double time)
  {
    // Calcula o bônus de tempo:
    // começa em 1000 e reduz 10 pontos por segundo.
    // Nunca permite que o valor fique negativo.
    double timeBonus = Mathf.Max(0, 1000 - (float)time * 10);

    // Pontuação baseada no número de infusões corretas
    // (cada uma vale 100 pontos).
    double correctScore = (correctInfusions * 100);

    // Penalidade por infusões erradas (cada erro tira 25 pontos).
    double wrongScore = (wrongInfusions * 25);

    // Soma da pontuação líquida relacionada às infusões.
    double infusionScore = correctScore - wrongScore;

    // Penalidade por mortes, cada morte reduz 150 pontos.
    double deathPenalty = deaths * 150;

    // Calcula a pontuação final somando tudo
    // e garantindo que não fique negativa.
    return Mathf.Max(0, (float)(timeBonus + infusionScore - deathPenalty));
  }

  // ...
}
```

Código 6 – Algoritmo utilizado para calcular a pontuação

Outros Componentes

Além dos sistemas principais, diversos outros *scripts* foram implementados para garantir um *gameplay* fluido, estável e responsivo. Entre eles, incluem-se módulos para controle de efeitos sonoros, partículas, interface de usuário, indicadores visuais, carregamento de cenas e outras funcionalidades de suporte. Embora não sejam o foco central das mecânicas educacionais, esses componentes desempenham papel importante na polidez e na imersão da experiência.

5.5 TESTES DO JOGO

Os testes do jogo foram conduzidos com o objetivo de identificar erros, avaliar o funcionamento das mecânicas e aprimorar a experiência geral de jogabilidade. Esta etapa teve caráter iterativo, sendo realizada após a implementação de cada nova funcionalidade, tanto de forma unitária quanto integrada, garantindo a coerência entre os diferentes componentes do sistema.

Os testes foram executados pelo próprio desenvolvedor, permitindo validar a adequação pedagógica e técnica do jogo. Essa prática possibilitou identificar falhas de lógica, inconsistências visuais e problemas de desempenho, além de ajustar elementos de interface e equilíbrio de dificuldade.

A verificação contínua de interações e funcionalidades contribuiu para assegurar que o produto final estivesse livre de erros críticos e em conformidade com os objetivos estabelecidos. Assim, esta fase consolidou-se como um processo essencial para garantir a estabilidade e a fluidez do jogo sério desenvolvido.

6 CONSIDERAÇÕES FINAIS

REFERÊNCIAS

ARAUJO, L. G. de J.; SILVA, A. P. dos S. Codebô: Design e avaliação de um puzzle game para o ensino de estrutura de dados. In: SBC. *Simpósio Brasileiro de Educação em Computaç ão (EDUCOMP)*. [S.l.], 2025. p. 27–36. Citado 4 vezes nas páginas 9, 13, 22 e 23.

BATTISTELLA, P. E.; WANGENHEIM, C. G. von. *ENgAGED: Um Processo de Desenvolvimento de Jogos para Ensinar Computação*. Tese (Doutorado) — Universidade Federal de Santa Catarina, Florianópolis, SC, 2016. Disponível em: <https://www.researchgate.net/publication/309895607_ENgAGED_Um_Processo_de_Desenvolvimento_de_Jogos_para_Ensinar_Computacao>. Citado 2 vezes nas páginas 11 e 16.

CARROLL, L. *Alice's Adventures in Wonderland*. Macmillan, 1865. Acesso em: nov. 2025. Disponível em: <https://www.gutenberg.org/ebooks/11>. Citado na página 41.

CERQUEIRA, T. de O.; SILVA, A. P. S.; ARAUJO, L. G. de J. Codebo unplugged: Um jogo desplugado para o ensino de pilha. In: SBC. *Simpósio Brasileiro de Educação em Computaç ão (EDUCOMP)*. [S.l.], 2023. p. 04–05. Citado na página 23.

CHILWANT, K. Comparison of two teaching methods, structured interactive lectures and conventional lectures. *Biomedical Research*, v. 23, n. 3, p. 363–366, 2012. Citado 2 vezes nas páginas 9 e 12.

CORMEN, T. H. et al. *Introduction to algorithms*. MIT press, 2022. Accessed: 2025-07-04. Disponível em: <https://books.google.com.br/books?hl=en&lr=&id=RSMuEAAAQBAJ&oi=fnd&pg=PR13&ots=a3l1_W6FVM&sig=ws-06DHa7Xc06QXZKzh7Pb2fvsA&redir_esc=y#v=onepage&q&f=false>. Citado na página 12.

COSTA, G. S. d. et al. Condigjob: um jogo sério para auxiliar nas disciplinas de linguagem de programação c. 2023. Citado 2 vezes nas páginas 21 e 22.

CRESWELL, J. W.; CRESWELL, J. D. *Projeto de pesquisa-: Métodos qualitativo, quantitativo e misto*. Penso Editora, 2021. Disponível em: <https://books.google.com.br/books?hl=en&lr=&id=URclEAAAQBAJ&oi=fnd&pg=PT5&ots=9g5IlSG0Hy&sig=xJnONf44ZCNqOeEsuCHHZ5A0lq8&redir_esc=y#v=onepage&q&f=false>. Citado na página 15.

Epic Games. *Unreal Engine Documentation*. [S.l.], 2025. Acesso em: nov. 2025. Disponível em: <https://docs.unrealengine.com>. Citado na página 40.

Fishing Cactus. *Algo Bot on Steam*. 2018. Accessed: 2025-04-14. Disponível em: <https://store.steampowered.com/app/286300/Algo_Bot/>. Citado 2 vezes nas páginas 27 e 28.

FromSoftware. *Dark Souls*. Bandai Namco Entertainment, 2011. Acesso em: nov. 2025. Disponível em: <https://en.bandainamcoent.eu/dark-souls/dark-souls>. Citado na página 42.

GLATZ, I. et al. Desenvolvimento de um jogo para auxílio no ensino de estruturas de dados. Florianópolis, SC., 2023. Citado na página 24.

Godot Engine Community. *Godot Engine Documentation*. [S.l.], 2025. Acesso em: nov. 2025. Disponível em: <https://docs.godotengine.org>. Citado na página 40.

HA, Y.; IM, H. The role of an interactive visual learning tool and its personalizability in online learning: Flow experience. *Online Learning*, ERIC, v. 24, n. 1, p. 205–226, 2020. Citado na página 9.

MALONE, T. W.; LEPPER, M. R. Making learning fun: A taxonomy of intrinsic motivations for learning. In: *Aptitude, learning, and instruction*. [S.l.]: Routledge, 2021. p. 223–254. Citado na página 13.

MOUAHEB, H. et al. The serious game: what educational benefits? *Procedia-Social and Behavioral Sciences*, Elsevier, v. 46, p. 5502–5508, 2012. Citado 4 vezes nas páginas 9, 11, 12 e 15.

MTAHO, A. B.; MSELLE, L. J. Difficulties in learning the data structures course: Literature review. *The Journal of Informatics*, v. 4, n. 1, p. 26–55, 2024. Accessed: 2025-07-04. Disponível em: <https://www.ajol.info/index.php/tji/article/view/276299>. Citado 4 vezes nas páginas 9, 11, 12 e 13.

NASCIMENTO, L. R. d. et al. *Prog-Poly: jogo de tabuleiro baseado no monopoly para ajudar nos estudos de linguagem de programação e engenharia de software*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2022. Citado 2 vezes nas páginas 24 e 25.

Nintendo. *Super Mario Bros.* Nintendo, 1985. Acesso em: nov. 2025. Disponível em: <https://www.nintendo.com/pt-pt/Jogos/NES/Super-Mario-Bros-803853.html>. Citado 2 vezes nas páginas 42 e 43.

Nintendo. *The Legend of Zelda Series*. Nintendo, 1986. Franquia de jogos eletrônicos. Disponível em: <https://zelda.nintendo.com/>. Citado na página 43.

NPC42 Games. *Iron Ears: Data Structure*. 2020. Accessed: 2025-04-15. Disponível em: <https://npc42-games.itch.io/ironears>. Citado na página 29.

Omoplata Games. *MOP'N SPARK on Steam*. 2025. Accessed: 2025-04-14. Disponível em: <https://store.steampowered.com/app/3491720/MOPN_SPARK/>. Citado na página 28.

PAPERT, S. The children's machine. *Technology Review-Manchester NH-*, TECHNOLOGY REVIEW, v. 96, p. 28–28, 1993. Citado 3 vezes nas páginas 9, 13 e 18.

SEGA. *Sonic the Hedgehog*. SEGA, 1991. Acesso em: nov. 2025. Disponível em: <https://www.sega.com/sonic-the-hedgehog>. Citado na página 42.

Square Enix. *Dragon Quest Series*. Square Enix, 1986. Acesso em: nov. 2025. Disponível em: <https://dragonquest.square-enix-games.com/>. Citado na página 43.

Square Enix. *Final Fantasy Series*. Square Enix, 1987. Acesso em: nov. 2025. Disponível em: <https://finalfantasy.square-enix-games.com/>. Citado na página 43.

Team Cherry. *Hollow Knight*. Team Cherry, 2017. Acesso em: nov. 2025. Disponível em: <https://www.hollowknight.com/>. Citado na página 43.

Tomorrow Corporation. *Human Resource Machine on Steam*. 2015. Accessed: 2025-04-14. Disponível em: <https://store.steampowered.com/app/375820/Human_Resource_Machine/>. Citado na página 27.

Unity Technologies. *Unity Manual*. [S.l.], 2025. Acesso em: nov. 2025. Disponível em: <https://docs.unity3d.com/Manual>. Citado 2 vezes nas páginas 14 e 40.

WIKIPEDIA. *Alchemical Symbols*. 2025. Acesso em: nov. 2025. Disponível em: <https://en.wikipedia.org/wiki/Alchemical_symbol>. Citado na página 42.