CodeQuest: Aprendendo a Programar com RPG Interativo

Gustavo Herpich[1] **, Rafael Vieira Coelho**[2]

[1]Instituto Federal do Rio Grande do Sul (IFRS) – Campus Farroupilha

Resumo. *Este artigo apresenta o desenvolvimento do jogo educacional CodeQuest, um serious game do gênero RPG interativo, voltado ao ensino de lógica de programação para iniciantes. A proposta busca mitigar as dificuldades recorrentes no ensino tradicional de algoritmos, como a abstração excessiva e a baixa motivação dos alunos, por meio da integração de mecânicas lúdicas, narrativa contextualizada e resolução progressiva de desafios. No jogo, o jogador interage com o ambiente e personagens utilizando comandos programados em Lua, os quais são convertidos automaticamente para GDScript a fim de executar ações no motor de jogo. O projeto foi desenvolvido utilizando a Godot Engine e a extensão GDExtension, permitindo a integração entre as linguagens e o controle dinâmico das mecânicas educacionais. O jogo é estruturado em fases que introduzem conceitos fundamentais de programação, como sequências lógicas, condicionais, laços de repetição e funções. Testes serão aplicados com alunos do IFRS – Farroupilha com o objetivo de avaliar o impacto do jogo no engajamento e na aprendizagem dos conceitos abordados.*

Abstract. *This paper presents the development of the educational game CodeQuest, an interactive RPG-style serious game designed to teach programming logic to beginners. The proposal aims to address common challenges in traditional algorithm teaching, such as high abstraction levels and low student engagement, by incorporating game mechanics, contextualized narrative, and progressive problem-solving activities. In the game, players interact with the environment and non-player characters through commands written in Lua, which are automatically converted into GDScript to be executed within the game engine. The project was developed using the Godot Engine and the GDExtension framework, enabling integration between scripting languages and educational mechanics. The game is organized into levels that gradually introduce core programming concepts, including logical sequencing, conditional statements, loops, and functions. Tests will be conducted with students from IFRS – Farroupilha to assess the game's impact on learning outcomes and student engagement.*

Palavras-chave: Educational Game. Programming Education. RPG. Godot. Lua.

1. Introdução

O ensino de programação ainda representa um grande desafio, sobretudo para iniciantes que enfrentam dificuldades em compreender conceitos como raciocínio lógico, estruturação de algoritmos e uso de linguagens. Muitas vezes, as metodologias tradicionais se apoiam em aulas expositivas e exercícios repetitivos, o que pode tornar o aprendizado pouco atraente e até desmotivador. Essa falta de engajamento contribui diretamente para os altos índices de evasão em cursos da área de informática Alonso e Figueiredo (2022).

De acordo com Qian e Lehman (2017), entre os principais obstáculos estão a compreensão de conceitos abstratos, a adaptação à sintaxe das linguagens e a falta de confiança na resolução de problemas por meio da programação. Além disso, fatores externos, como a necessidade de conciliar estudo e trabalho, também dificultam a permanência dos alunos em cursos técnicos e superiores, como evidenciado em Alonso e Figueiredo (2022).

Nesse cenário, os métodos tradicionais mostram-se insuficientes para atender a todos os perfis de estudantes. Muitos aprendem melhor com abordagens práticas e experimentais, que permitem vivenciar na prática o que seria apenas teórico. Essa realidade reforça a necessidade de buscar metodologias inovadoras que tornem o aprendizado mais acessível e motivador.

Uma das alternativas que vem ganhando destaque é o uso de jogos educacionais e estratégias de gamificação. Estudos como os de Muratet *et al.* (2009) e Wong e Yatim (2018) apontam que jogos sérios (*serious games*) favorecem a aprendizagem prática em ambientes simulados. Já Figueiredo e García-Peñalvo (2020) e Zhan *et al.* (2022) mostram que a gamificação pode transformar atividades monótonas em experiências dinâmicas e envolventes, aumentando a motivação dos estudantes.

Com base nessas ideias, este projeto tem como objetivo principal desenvolver um jogo educacional no estilo *Role-Playing Game* (RPG), em que o aprendizado de programação ocorre por meio de desafios interativos. Os jogadores escrevem pequenos trechos de código em Lua, que são traduzidos e executados no ambiente do jogo, permitindo acompanhar na prática os efeitos de seus comandos. A proposta busca, portanto, criar um espaço acessível e motivador para iniciantes.

Entre os objetivos específicos estão: criar um ambiente de programação dentro do jogo, desenvolver missões progressivas que ensinem conceitos fundamentais (como variáveis, estruturas de repetição e condicionais), além de avaliar a experiência dos usuários e analisar se o jogo contribui para a retenção de conhecimento.

A hipótese que orienta o estudo é que um RPG educacional pode tornar o aprendizado de programação mais envolvente e acessível, ajudando os alunos a desenvolver tanto habilidades lógicas quanto práticas. Para verificar isso, o jogo será aplicado em turmas do curso técnico em informática integrado ao ensino médio do IFRS – Farroupilha, onde um formulário será disponibilizado, após a finalização do jogo, para a coleta de dados. Além disso, o jogo será disponibilizado na plataforma *itch.io*, a fim de coletar feedbacks externos e ampliar a avaliação de seu potencial pedagógico.

Assim, este trabalho busca contribuir para o ensino de programação com uma abordagem inovadora e interativa, explorando o potencial dos *serious games* para tornar o aprendizado mais significativo e envolvente.

2. Revisão Bibliográfica

2.1. Desafios no Ensino de Programação

O ensino de programação tem sido historicamente desafiador, sobretudo para iniciantes, devido à necessidade de desenvolver raciocínio lógico, lidar com conceitos abstratos e compreender a sintaxe das linguagens. Segundo Qian e Lehman (2017), esses fatores representam obstáculos iniciais significativos. Enquanto Moreira *et al.* (2018) enfatiza a dificuldade de assimilação dos conteúdos teóricos, Queiroz, Rodrigues e Coutinho (2018)

ressalta que o excesso de métodos tradicionais pode gerar desmotivação e cansaço nos estudantes. Pesquisas mais recentes confirmam que tais problemas persistem: Pereira e Sousa (2024) observam que a falta de práticas em sala de aula e a ênfase excessiva em teoria levam os alunos a buscar materiais complementares na internet. Além das dificuldades técnicas, a evasão nos cursos de Computação também é influenciada por fatores socioeconômicos e motivacionais, como apontam Alonso e Figueiredo (2022) e Santos e Vieira (2023), que evidenciam a necessidade de ferramentas mais acessíveis e engajadoras, sobretudo no contexto brasileiro.

2.2. Jogos Educacionais e Gamificação

Nesse cenário, os jogos digitais aplicados à educação, conhecidos como *serious games*, surgem como alternativa promissora. O termo *serious games* foi originalmente cunhado por Clark Abt em 1970, que os definiu como jogos cujo propósito principal não é apenas o entretenimento, mas a promoção de aprendizado, treinamento ou mudança de comportamento, mantendo ainda características lúdicas próprias dos jogos. A partir dessa concepção, os *serious games* passaram a ser compreendidos como ambientes estruturados nos quais objetivos educacionais estão intrinsecamente integrados às mecânicas de jogo.

Diferentemente da gamificação, que aplica elementos de jogos a contextos externos, os *serious games* integram aprendizado e mecânica de jogo, permitindo que a prática ocorra diretamente dentro do ambiente lúdico Muratet *et al.* (2009). Diversos estudos demonstram que tais jogos aumentam o engajamento e favorecem a aprendizagem prática, embora sua eficácia dependa de um design pedagógico adequado e de sua integração com o papel do professor. Assim, o consenso na literatura é que os jogos não substituem o ensino formal, mas podem atuar como um importante recurso complementar.

A gamificação, por sua vez, refere-se à aplicação de elementos de jogos em contextos não lúdicos, buscando aumentar motivação e engajamento Figueiredo e García-Peñalvo (2020). Pesquisas recentes, como as de Zhan *et al.* (2022), demonstram ganhos significativos em motivação e desempenho acadêmico, sobretudo quando combinada a estratégias pedagógicas bem estruturadas, como ressalta Castro e Santos (2024). A definição clássica de gamificação, consolidada por Deterding *et al.* (2011), diferencia-a do design de jogos e evidencia seu potencial em cenários educacionais. Apesar dos avanços, ainda existem poucas iniciativas que unem gamificação a atividades de programação textual com execução real de código, representando uma lacuna que o CodeQuest visa preencher.

2.3. Jogos Digitais no Ensino de Programação e Contribuições do CodeQuest

Diversos projetos têm buscado aproximar o ensino de programação do universo dos jogos digitais. Um dos exemplos mais conhecidos é o *CodeCombat*, que utiliza mecânicas de RPG para ensinar lógica e algoritmos por meio de código textual, conforme ilustrado na Figura 1. Nele, o jogador controla um personagem cujas ações dependem do código escrito, proporcionando uma experiência imersiva e interativa no aprendizado de linguagens de programação.

Outro caso amplamente utilizado em contextos educacionais é o *Scratch*, apresentado na Figura 2, que adota uma interface baseada em blocos visuais para facilitar a introdução de conceitos de lógica e controle de fluxo. Embora essa abordagem seja eficaz para iniciantes, ela abstrai a sintaxe textual, o que pode dificultar a transição para linguagens de programação tradicionais utilizadas em cursos de Computação.

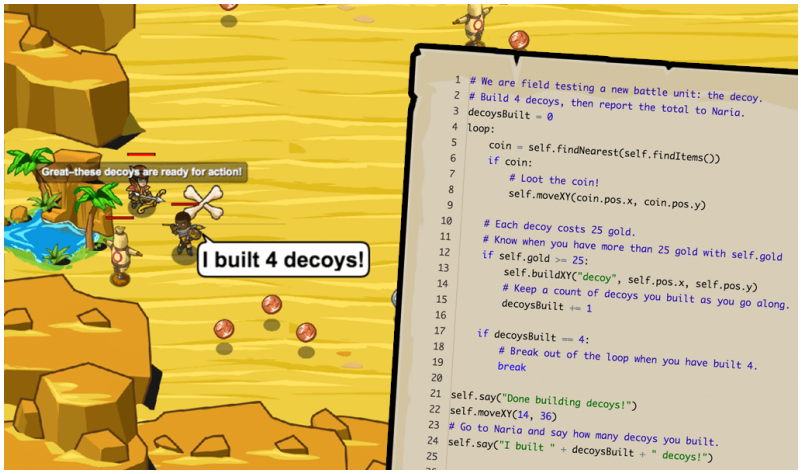

Figura 1. Interface do jogo *CodeCombat*, um exemplo de *serious game* que ensina lógica de programação por meio da escrita de código em linguagens textuais. O jogador controla um personagem através de comandos programados, promovendo o aprendizado ativo de estruturas de controle e algoritmos.
Fonte: Adaptado de CodeCombat (2025).

Estudos, como os de Kroustalli e Xinogalos (2021) e Wong e Yatim (2018), demonstram o potencial desses ambientes para aumentar o engajamento e a compreensão de conceitos fundamentais. No entanto, observa-se que grande parte das soluções atuais ainda privilegia linguagens visuais e simplificadas, oferecendo menor contato com a prática da codificação textual.

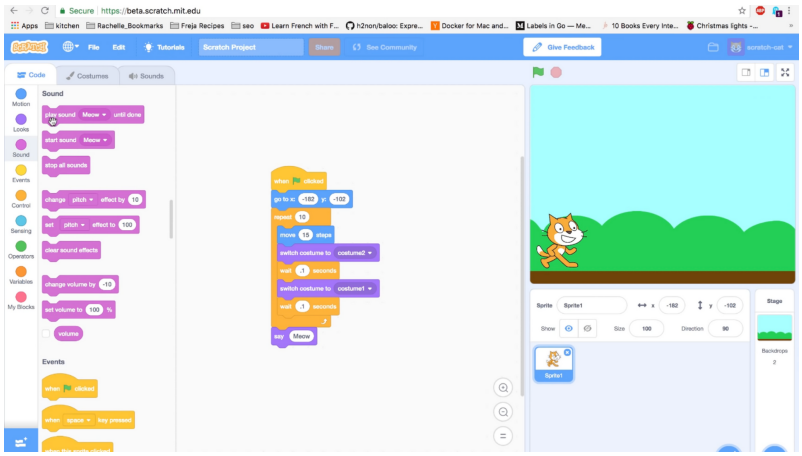

Figura 2. Ambiente visual do *Scratch*, ferramenta de ensino baseada em blocos lógicos que permite a criação de programas de forma intuitiva, voltada principalmente à introdução de conceitos de lógica e pensamento computacional para iniciantes.
Fonte: Adaptado de Scratch (MIT Media Lab, 2025).

Diante desse panorama, identifica-se uma lacuna no uso de jogos que integrem gamificação, programação textual e feedback imediato da execução do código. O *CodeQuest* propõe-se a preencher essa lacuna ao oferecer um ambiente gamificado no qual os estudantes escrevem código em *Lua* e observam seus efeitos diretamente dentro do jogo,

por meio do motor Godot. Essa abordagem combina interatividade, motivação e prática experimental, promovendo um aprendizado mais envolvente e alinhado às demandas atuais do ensino de Computação.

2.4. Inteligência Artificial e Desafios Cognitivos na Educação Contemporânea

O avanço da Inteligência Artificial (IA) tem transformado profundamente o cenário educacional, oferecendo novas possibilidades de personalização do ensino e automação de tarefas pedagógicas. Contudo, como destaca Azambuja e Silva (2024), essa revolução tecnológica também impõe a necessidade de repensar as práticas docentes e os modelos de aprendizagem, especialmente em um contexto em que o uso de sistemas inteligentes tende a substituir, e não a complementar, o esforço cognitivo humano. A formação de professores e o papel das instituições de ensino tornam-se, assim, centrais para garantir que a tecnologia seja usada como ferramenta de ampliação do pensamento crítico e não de sua redução.

Pesquisas recentes apontam que o uso excessivo de assistentes baseados em IA, como chatbots e geradores de texto, pode prejudicar o desenvolvimento cognitivo e a autonomia intelectual de jovens estudantes. Estudos como *Your Brain on ChatGPT* (2024) e matérias divulgadas por José Alfredo Carvalho (2024) e Jornal O Casarão (UFF) (2025) evidenciam que a dependência dessas ferramentas reduz o engajamento mental, a capacidade de memorização e o senso de autoria, gerando o que pesquisadores denominam "dívida cognitiva". Crianças e adolescentes, ao receberem respostas prontas, perdem o hábito de questionar e de construir conhecimento de forma ativa, o que acentua um problema já presente em métodos tradicionais de ensino que priorizam a memorização em detrimento da experimentação e da criatividade.

Nesse cenário, torna-se urgente adotar abordagens pedagógicas inovadoras que estimulem a aprendizagem ativa e o pensamento reflexivo. Os *serious games* e ambientes gamificados representam uma dessas alternativas, pois podem integrar elementos de IA de maneira equilibrada — utilizando algoritmos adaptativos e feedback dinâmico, sem eliminar o protagonismo do aluno. Dessa forma, o uso consciente da IA em jogos educativos pode contribuir para restaurar o papel central da curiosidade, da resolução de problemas e do raciocínio crítico no processo de aprendizagem, preparando estudantes e professores para um futuro no qual a tecnologia e a cognição humana evoluam de forma complementar.

3. Modelagem

A modelagem do *CodeQuest* teve como objetivo estruturar os componentes técnicos e pedagógicos do jogo, assegurando que os mecanismos de ensino de programação estivessem integrados de forma coerente à experiência lúdica proposta. Para isso, foram aplicados conceitos da Engenharia de Software, possibilitando a definição dos requisitos funcionais e não funcionais, das regras de negócio e da organização geral do sistema.

O *CodeQuest* foi concebido como um ambiente educacional interativo, no qual a progressão do jogador está diretamente vinculada à resolução de desafios baseados em lógica e programação. Cada fase apresenta um problema contextualizado na narrativa do jogo, cuja solução exige a escrita e execução de código, promovendo a aprendizagem por experimentação.

Os requisitos funcionais incluem a movimentação do personagem pelo cenário, a interação com objetos, terminais e personagens não jogáveis (NPCs), além do uso de um console de programação integrado ao jogo. Por meio desse console, o jogador escreve comandos na linguagem *Lua* para alterar o ambiente do jogo, como abrir portas, mover pontes ou ativar mecanismos. O console oferece suporte a estruturas fundamentais da linguagem, como variáveis, condicionais (`if`), laços de repetição (`for`, `while`) e funções, permitindo um aprendizado progressivo e contextualizado.

Como apoio ao processo de aprendizagem, o sistema inclui o Livro de Ajuda, que apresenta explicações teóricas, exemplos de código e métodos específicos do jogo. Novos conteúdos são desbloqueados conforme o avanço do jogador, estabelecendo uma relação direta entre progresso, desempenho e aquisição de conhecimento. O sistema também fornece feedback textual e visual imediato, simulando mensagens de erro e sucesso semelhantes às encontradas em ambientes reais de desenvolvimento.

Os requisitos não funcionais asseguram a qualidade técnica e a usabilidade do sistema. O jogo foi desenvolvido utilizando a *Godot Engine 4.4*, com suporte multiplataforma (Windows, Linux e Web), desempenho estável e arquitetura modular baseada em nós e cenas, facilitando a manutenção e a expansão futura do projeto.

3.1. Estados do Jogo

A dinâmica de funcionamento do *CodeQuest* pode ser compreendida por meio do seu diagrama de estados ilustrado na A Figura 3, que representa o fluxo principal da experiência do jogador desde o menu inicial até a conclusão das fases. O modelo descreve os principais estados do sistema e as transições entre eles, evidenciando como a progressão pedagógica está integrada à lógica do jogo.

Entre os estados centrais destacam-se a exploração do cenário, a interação com NPCs, o acesso ao terminal de programação, a execução de código e o retorno ao ambiente de jogo. As transições entre esses estados ocorrem de forma controlada, garantindo que o jogador só avance após a resolução correta dos desafios propostos.

Como exemplo, em uma das fases iniciais, o jogador encontra uma ponte desalinhada que bloqueia o acesso à próxima área. Para resolver o desafio, é necessário utilizar o terminal de programação e escrever um trecho de código que chame um método específico do jogo responsável por movimentar a ponte. Esse desafio está associado à introdução de conceitos básicos de funções e chamadas de métodos. Caso o código esteja incorreto, o sistema retorna mensagens de erro, impedindo a progressão até que a solução adequada seja encontrada.

Esse modelo exemplifica a proposta central do jogo: cada obstáculo no cenário corresponde a um conceito de programação que deve ser compreendido e aplicado pelo jogador. Outros desafios, envolvendo estruturas condicionais, laços de repetição e manipulação de variáveis, seguem a mesma lógica e são apresentados de forma progressiva ao longo das fases. Exemplos adicionais desses desafios são apresentados no Apêndice A e no Apêndice B.

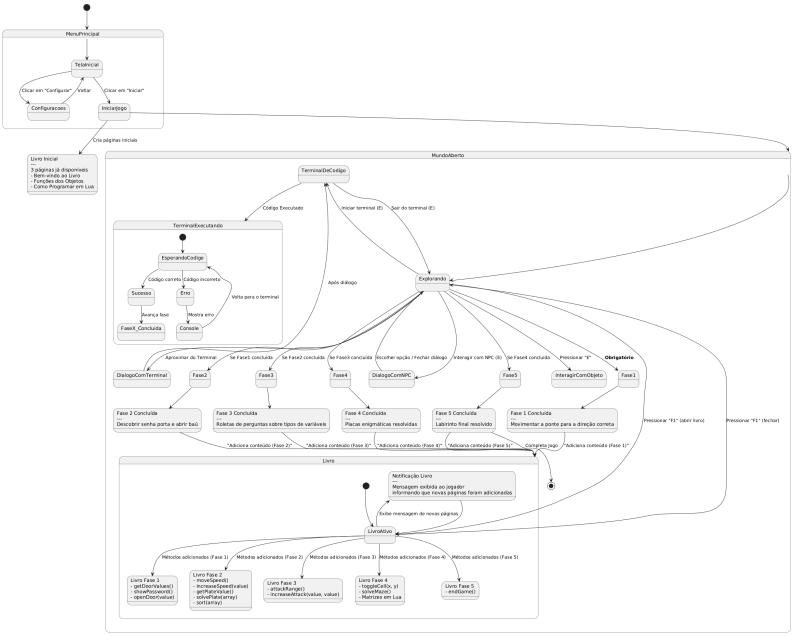

Figura 3. Diagrama de estados do jogo *CodeQuest*, representando o fluxo das fases, as interações com terminais e a progressão do jogador. O modelo destaca a integração entre os sistemas de diálogo, livro de ajuda e desafios lógicos presentes em cada fase.
Fonte: Elaborado pelo autor (2025).

3.2. Integração entre Lua e Godot

A execução dos códigos escritos pelo jogador no *CodeQuest* é viabilizada por meio da integração entre a *Godot Engine* e a linguagem *Lua*, realizada através de uma *GDExtension* personalizada denominada *LuaConsole*. Essa integração permite que os comandos inseridos no console do jogo sejam interpretados e executados em tempo real, produzindo efeitos imediatos no ambiente virtual.

Do ponto de vista da modelagem, essa arquitetura garante que a lógica de programação esteja desacoplada da lógica interna do jogo, preservando a integridade do sistema e permitindo a execução segura dos códigos. O detalhamento técnico dessa integração, bem como as motivações para a escolha da linguagem *Lua* e da tecnologia *GDExtension*, são apresentados na Seção de Desenvolvimento. A Figura 4 ilustra essa arquitetura de forma resumida.

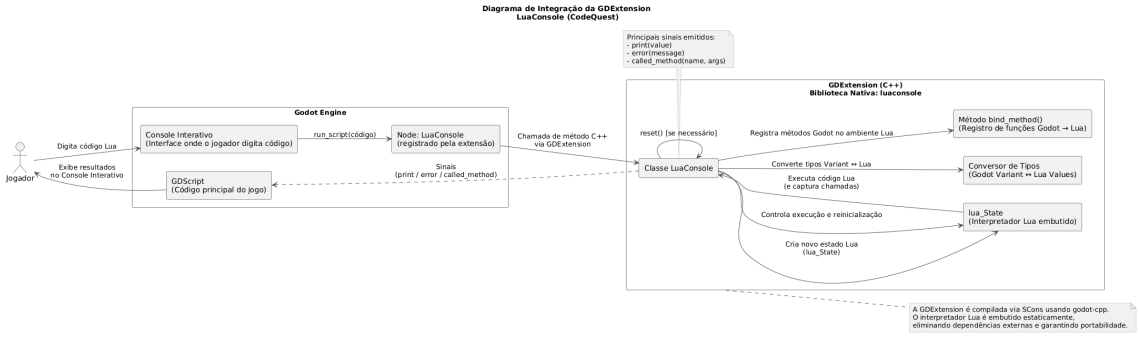

Figura 4. Diagrama da arquitetura de integração entre a *Godot Engine* e o interpretador *Lua* por meio da extensão *GDExtension LuaConsole*. O esquema ilustra a comunicação entre o console interativo, os scripts em *GDScript* e a biblioteca C++ responsável pela execução do código *Lua* dentro do jogo. Fonte: Elaborado pelo autor (2025).

4. Desenvolvimento

Esta seção descreve o processo de desenvolvimento do jogo educacional *CodeQuest*, abordando as tecnologias utilizadas, ferramentas de apoio, metodologias aplicadas e decisões de design que integraram o aprendizado de programação à experiência de jogo. Também são apresentados os principais elementos de gamificação, a estrutura das fases e os conteúdos pedagógicos trabalhados em cada uma, evidenciando o alinhamento entre as escolhas tecnológicas e os objetivos de ensino de lógica e pensamento computacional.

4.1. Metodologia de Desenvolvimento

A metodologia de desenvolvimento adotada no projeto *CodeQuest* foi de natureza incremental e iterativa, alinhada a práticas ágeis e ao uso de prototipagem contínua. Essa abordagem permitiu a construção gradual do sistema, com validações frequentes tanto dos aspectos técnicos quanto pedagógicos do jogo.

O desenvolvimento ocorreu por meio de ciclos sucessivos de planejamento, implementação, testes e refinamento. A cada ciclo, novas funcionalidades eram incorporadas e avaliadas com base na jogabilidade, na clareza dos desafios propostos e na efetividade das mecânicas educacionais, possibilitando ajustes rápidos e direcionados.

Foram priorizados protótipos funcionais das mecânicas centrais do jogo, como o sistema de interação, o console de programação e o Livro de Ajuda. Esses protótipos permitiram verificar a integração entre o código escrito pelo jogador e o comportamento do ambiente do jogo, assegurando que os conceitos de programação fossem aplicados de forma prática e significativa.

O versionamento do código-fonte foi realizado utilizando a plataforma *GitHub*, possibilitando o controle de versões, a organização do desenvolvimento e a rastreabilidade das alterações. Os testes foram conduzidos de forma contínua, abrangendo aspectos de jogabilidade, execução correta dos scripts em *Lua* e compatibilidade entre plataformas. O jogo foi testado nos sistemas operacionais Windows, Linux e em ambiente Web, por meio da plataforma *itch.io*, garantindo portabilidade, estabilidade e consistência na experiência do usuário.

4.2. Tecnologias de Desenvolvimento

4.3. Estrutura do Jogo e Mecânicas Pedagógicas

O *CodeQuest* foi projetado integrando elementos de gamificação como estratégia pedagógica central, conectando a resolução de desafios às práticas de programação textual. O principal instrumento de interação do jogador é o terminal de código (Figura 5), por meio do qual comandos escritos em *Lua* são executados e produzem efeitos imediatos no ambiente virtual. Essa abordagem permite associar diretamente o código ao comportamento do jogo, promovendo uma aprendizagem ativa e experimental.

Figura 5. Console interativo utilizado pelos estudantes para escrever e executar código em *Lua*, elemento central dos desafios das fases.
Fonte: Captura de tela do autor (2025).

A progressão do conteúdo foi estruturada de forma gradual. A cada fase, novos métodos e comandos são desbloqueados, reduzindo a sobrecarga cognitiva inicial e favorecendo a assimilação progressiva dos conceitos. Como apoio a esse processo, foi implementado o Livro de Ajuda (Figura 6), acessado por meio da tecla **F1**, que registra funções, exemplos de uso e explicações relacionadas aos desafios já enfrentados pelo jogador, funcionando como material de consulta contínua dentro do próprio jogo.

Figura 6. Livro de ajuda contendo métodos, exemplos e explicações liberadas conforme a progressão nas fases do jogo.
Fonte: Captura de tela do autor (2025).

O jogo foi organizado em fases que introduzem, de maneira progressiva, conceitos fundamentais de programação. Cada fase combina elementos narrativos, desafios lógicos e feedback visual, reforçando a relação entre o código escrito e o comportamento observado no ambiente virtual. A Tabela 1 apresenta os conteúdos pedagógicos trabalhados em cada fase, bem como os principais métodos disponibilizados ao jogador.

Tabela 1. Conteúdos de aprendizagem abordados em cada fase do *CodeQuest*, relacionando mecânicas do jogo aos conceitos de programação trabalhados. Fonte: Elaborado pelo autor (2025).

Fase	**Conceitos abordados**
Fase 1 — Ponte	Funções simples; sequenciamento lógico (`moveBridge`)
Fase 2 — Portas	Condicionais e laços (`getDoorValues`, `openDoor`)
Fase 3 — Roleta	Tipos de variáveis; efeitos em atributos (`increaseAttack`)
Fase 4 — Placas	Ordenação e manipulação de listas (`sort`, `solvePlate`)
Fase 5 — Labirinto	Matrizes e navegação bidimensional (`toggleCell`)

Essas escolhas de design visam alinhar as mecânicas do jogo aos objetivos pedagógicos do projeto, garantindo que cada desafio proposto esteja diretamente associado a conceitos de lógica e programação, reforçando a aprendizagem por meio da experimentação e da progressão controlada de complexidade.

4.3.1. Godot 4.4 e GDScript

O jogo foi desenvolvido na *Godot Engine* (versão 4.4), devido ao seu caráter *open-source*, suporte multiplataforma e praticidade na criação de jogos 2D. A arquitetura baseada em *nodes* e *scenes* possibilitou modularizar o desenvolvimento e facilitar a implementação de sistemas independentes, como o console, o livro de ajuda e as fases.

A Figura 7 ilustra o ambiente de desenvolvimento do projeto dentro da engine, onde foram criadas as cenas, scripts e componentes interativos que compõem o jogo.

A linguagem nativa *GDScript*, de sintaxe semelhante ao Python, foi utilizada para controlar o comportamento dos objetos, gerenciar animações, interações com NPCs e eventos do jogador. Entre os principais sistemas implementados via *GDScript*, destacam-se:

- **Sistema de Interação:** detecta ações do jogador e aciona componentes interativos do cenário;
- **Gerenciador Global:** centraliza informações de progresso, estados do jogo e atalhos de entrada;
- **Gestão de Fases:** controla desbloqueios de conteúdos pedagógicos conforme o avanço do jogador.

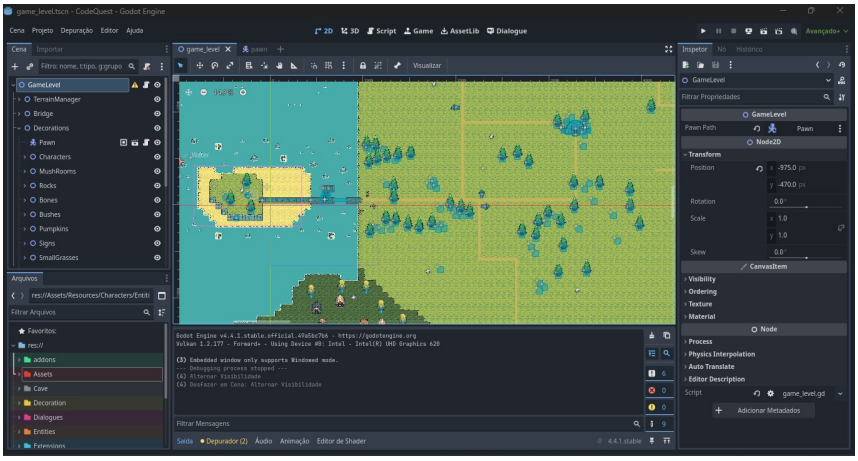

Figura 7. Ambiente de desenvolvimento do projeto no editor da *Godot Engine* (versão 4.4). A imagem ilustra a estrutura modular de *nodes* e *scenes*, utilizada na criação dos sistemas do jogo.
Fonte: Captura de tela do autor (2025).

4.3.2. Lua e Integração via GDExtension

A linguagem *Lua* foi incorporada ao *CodeQuest* para permitir que o jogador escrevesse e executasse códigos dentro do jogo, simulando um ambiente real de programação. A escolha do *Lua* se deu tanto por suas características técnicas quanto por sua ampla adoção na indústria de jogos digitais, especialmente em títulos que utilizam scripts para controle de lógica, comportamento de personagens e modificação dinâmica do ambiente.

Diversas empresas e motores de jogo adotam o *Lua* como linguagem embarcada devido à sua sintaxe simples, curva de aprendizado reduzida, alto desempenho e facilidade de integração com linguagens de baixo nível, como C e C++. Essas características tornam o *Lua* especialmente adequado para fins educacionais, permitindo que estudantes iniciantes compreendam conceitos fundamentais de programação sem a complexidade inicial de linguagens mais verbosas.

Para viabilizar a execução de código *Lua* dentro da *Godot Engine*, foi desenvolvida uma *GDExtension* personalizada denominada *LuaConsole*, implementada em C++. O uso da *GDExtension* permite estender as funcionalidades da engine sem a necessidade de recompilar o motor, mantendo o sistema modular e portável.

Essa integração foi realizada utilizando a API *godot-cpp* e compilada com a ferramenta *SCons*. Por meio dela, métodos do jogo podem ser expostos ao ambiente *Lua*, permitindo que o código escrito pelo jogador interaja diretamente com objetos e mecanismos do cenário. Além disso, o sistema realiza a conversão de dados entre os tipos da Godot e os tipos do *Lua*, bem como a emissão de sinais para retorno de mensagens de saída e erro.

Essa abordagem possibilitou a implementação de um terminal funcional dentro do jogo, com execução imediata e feedback visual, sem dependências externas, reforçando a proposta pedagógica do *CodeQuest*.

4.3.3. DialogueManager 3

O plugin *DialogueManager 3* foi utilizado para o gerenciamento de diálogos dinâmicos e instruções contextuais durante o aprendizado.

A ferramenta possibilita a criação de fluxos de diálogo condicionais, que podem ser personalizados conforme as variáveis do jogo e o progresso do jogador.

A Figura 8 apresenta um exemplo genérico de interface criada com o *DialogueManager 3*, ilustrando o funcionamento da árvore de diálogo e as opções de escolha configuráveis dentro da engine.

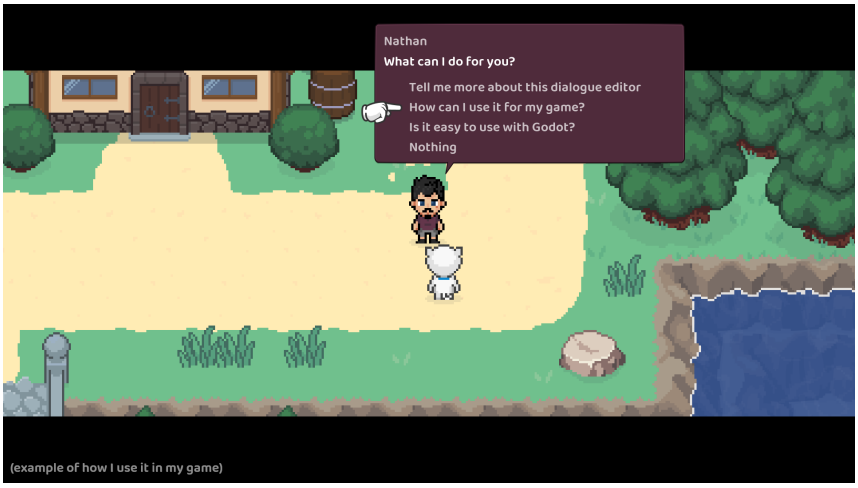

Figura 8. Exemplo de fluxo de diálogos criado com o plugin *DialogueManager 3* dentro da *Godot Engine*. O sistema permite construir árvores condicionais de conversas, utilizadas para orientar o jogador durante os desafios e transmitir instruções pedagógicas dinâmicas.
Fonte: Adaptado de DialogueManager 3 (2025).

4.3.4. Recursos Visuais e Sonoros

Os recursos gráficos utilizados no desenvolvimento foram obtidos em *assets* de domínio público disponíveis no *itch.io*, utilizados para compor o cenário, personagens e animações de teste durante o processo de prototipagem. A Figura 9 apresenta os principais sprites e animações do personagem principal desenvolvidos e adaptados para o jogo.

Elementos adicionais, como ícones e sprites de interface, foram criados manualmente no *Piskel*, preservando a identidade visual pixelada.

Os efeitos sonoros e trilhas foram selecionados a partir das bibliotecas *Mixkit.co* e *Pixabay*, priorizando ambiências e efeitos característicos de jogos de RPG 2D. As fontes tipográficas foram escolhidas no catálogo do *Google Fonts*, buscando legibilidade e harmonia visual com o estilo gráfico adotado.

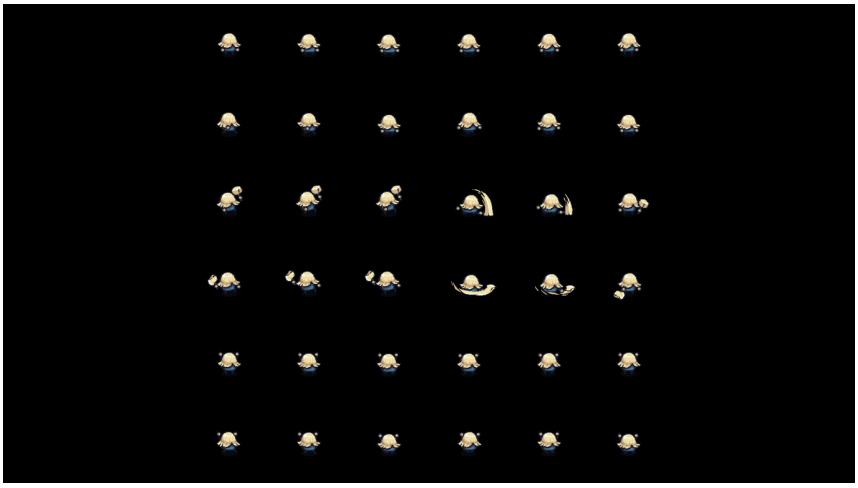

Figura 9. Sprites e animações do personagem principal do *CodeQuest*, desenvolvidos e adaptados a partir de *assets* de domínio público disponíveis na plataforma *itch.io*. As animações incluem estados de movimentação, interação e repouso, compondo a identidade visual do jogo.
Fonte: Adaptado pelo autor a partir de recursos públicos do *itch.io* (2025).

5. Resultados

Esta seção apresenta os resultados observados durante o desenvolvimento e a aplicação do jogo educacional *CodeQuest* em turmas do curso técnico em informática integrado ao ensino médio do IFRS – Campus Farroupilha. A aplicação ocorreu no final do período letivo de 2025, envolvendo três turmas distintas: uma turma do primeiro ano, uma do terceiro ano e uma do quarto ano do ensino médio. Cada turma teve, em média, aproximadamente duas horas de contato direto com o jogo, distribuídas em uma sessão prática em laboratório.

Os resultados descritos nesta seção baseiam-se nas percepções do autor durante a aplicação prática do jogo em sala de aula, considerando o comportamento dos estudantes, suas interações com o sistema, as dificuldades recorrentes observadas e os comentários espontâneos realizados ao longo da atividade. A análise contempla três dimensões principais: (i) aspectos técnicos e pedagógicos das mecânicas implementadas; (ii) o desempenho do jogo como ferramenta de apoio ao ensino de programação; e (iii) dificuldades e limitações identificadas durante a aplicação nas diferentes turmas.

5.1. Avaliação da experiência dos estudantes

5.1.1. Facilidade de uso e compreensão das fases

Durante a aplicação do *CodeQuest*, observou-se que a maioria dos estudantes apresentou um nível intermediário de facilidade para compreender o funcionamento geral do jogo e a progressão entre as fases. Estudantes do terceiro e do quarto ano do ensino médio demonstraram maior autonomia para explorar o mapa, interpretar as instruções e compreender os objetivos de cada desafio, conseguindo avançar pelas fases com menor necessidade de mediação do professor.

Em contrapartida, alunos do primeiro ano do ensino médio, especialmente aqueles sem contato prévio com programação, apresentaram maior dificuldade inicial. Essas dificuldades estiveram relacionadas principalmente à interpretação das instruções das

fases e à compreensão do fluxo de progressão dentro do cenário do jogo. Em todas as turmas, foi recorrente a manifestação de incerteza sobre o caminho a seguir no mapa, indicando a necessidade de aprimorar elementos de orientação visual e textual que auxiliem a navegação e o entendimento dos objetivos.

5.1.2. Clareza das instruções e suporte pedagógico

De modo geral, a clareza das instruções fornecidas pelos diálogos narrativos, pelo terminal de programação e pelo Livro de Ajuda foi percebida de forma positiva durante a aplicação do jogo. Observou-se que a combinação entre narrativa contextualizada, instruções textuais e documentação integrada contribuiu para reduzir a frustração inicial e apoiar os estudantes na resolução dos desafios propostos.

Ainda assim, especialmente entre os estudantes do primeiro ano, foi perceptível a necessidade de exemplos mais detalhados no Livro de Ajuda e de explicações adicionais sobre alguns comandos e conceitos. Em determinados momentos, os alunos demonstraram dependência maior da mediação do professor, o que indica oportunidades de aprimoramento no suporte pedagógico oferecido pelo sistema.

5.1.3. Engajamento e motivação

O engajamento dos estudantes manteve-se elevado ao longo da aplicação do *CodeQuest*, principalmente nas turmas do terceiro e do quarto ano do ensino médio. Observou-se interesse contínuo pela proposta do jogo, com destaque para o caráter lúdico, a narrativa integrada ao progresso das fases e a possibilidade de interação direta com o ambiente do jogo por meio do código.

Foi perceptível que a visualização imediata dos efeitos do código escrito no terminal sobre o cenário do jogo contribuiu significativamente para a motivação dos estudantes, incentivando a experimentação, a tentativa e erro e a permanência na atividade por períodos prolongados.

5.1.4. Compreensão de lógica e pensamento algorítmico

Durante a aplicação do jogo, observou-se avanço progressivo na compreensão de conceitos básicos de lógica de programação, como variáveis, estruturas condicionais e laços de repetição. Esse avanço foi mais evidente à medida que os estudantes progrediam pelas fases, especialmente quando os desafios apresentavam explicações conceituais seguidas de aplicação prática imediata.

Comentários espontâneos realizados durante a atividade indicaram que muitos estudantes passaram a compreender melhor a relação entre a estrutura do código e o comportamento do sistema no jogo. Entretanto, também foi recorrente a dificuldade na interpretação das mensagens de erro apresentadas no terminal, especialmente entre alunos iniciantes, apontando a necessidade de aprimorar a clareza e o caráter pedagógico desses retornos.

5.1.5. Relação entre código e efeitos visuais

Um dos aspectos mais evidentes observados durante a aplicação do *CodeQuest* foi a clareza da relação entre o código escrito no console e os efeitos visuais produzidos no ambiente do jogo. A possibilidade de observar imediatamente as consequências do código executado foi frequentemente destacada pelos estudantes como um dos principais diferenciais do jogo.

Essa relação direta entre abstração algorítmica e transformação visual do ambiente mostrou-se eficaz para facilitar a compreensão de conceitos fundamentais de programação, especialmente para estudantes em estágios iniciais de aprendizagem.

5.2. Dificuldades observadas em turmas iniciantes

A aplicação do jogo na turma do primeiro ano do ensino médio revelou um padrão mais acentuado de dificuldades iniciais quando comparada às turmas mais avançadas. Entre os principais desafios observados, destacam-se:

- dificuldade em compreender claramente os objetivos de cada fase;
- incerteza quanto ao caminho correto a seguir no mapa do jogo;
- interpretação limitada das mensagens de erro apresentadas no terminal.

Essas observações indicam que, embora o jogo cumpra seu papel como ferramenta de apoio ao ensino, ajustes adicionais são necessários para melhorar a acessibilidade cognitiva, especialmente para estudantes em contato inicial com programação.

5.3. Considerações sobre acessibilidade e inclusão

Durante a aplicação do *CodeQuest*, também foi possível observar que estudantes com características do espectro autista ou com dificuldades cognitivas enfrentaram desafios significativamente maiores para progredir no jogo. Embora o projeto não tenha sido originalmente desenvolvido com diretrizes específicas de acessibilidade, essas observações evidenciam uma lacuna importante.

Entre os pontos que podem orientar trabalhos futuros, destacam-se a necessidade de instruções mais objetivas e visuais, navegação assistida, níveis ajustáveis de complexidade e diferentes formas de feedback, de modo a tornar o jogo mais inclusivo e adequado a diferentes perfis de aprendizagem.

5.4. Síntese dos resultados

De forma geral, as observações realizadas durante a aplicação do *CodeQuest* indicam que o jogo favorece o engajamento, a aprendizagem ativa e a compreensão prática de conceitos de lógica de programação. As dificuldades identificadas nas turmas iniciantes e em perfis com necessidades específicas apontam caminhos claros para aprimoramentos futuros, reforçando o potencial do jogo como ferramenta complementar eficaz no ensino introdutório de programação.

6. Conclusão

O desenvolvimento do *CodeQuest* permitiu investigar o uso de jogos digitais como recurso de apoio ao ensino introdutório de programação, explorando a integração entre narrativa,

desafios progressivos e execução real de código. A proposta do jogo foi concebida para atuar como um ambiente complementar às práticas docentes, buscando aproximar conceitos abstratos de programação de situações práticas e contextualizadas dentro de uma experiência lúdica.

A implementação do console em *Lua*, integrada à arquitetura modular da *Godot Engine*, possibilitou a criação de um ambiente interativo no qual os estudantes puderam experimentar comandos, testar hipóteses e observar os efeitos de suas ações no mundo virtual em tempo real. Essa característica favoreceu a compreensão da relação entre código e comportamento do sistema, aspecto considerado central no processo de aprendizagem de programação.

Os resultados obtidos com 55 estudantes do ensino médio integrado em informática indicam que o uso do *CodeQuest* contribuiu para a compreensão de conceitos fundamentais, como variáveis, laços de repetição e estruturas condicionais, especialmente pela visualização direta entre código e efeitos no cenário do jogo. Além disso, observou-se um aumento nos níveis de motivação e engajamento durante as atividades propostas, o que sugere que o jogo pode atuar como um elemento motivador no contexto educacional analisado.

Entretanto, os testes também evidenciaram limitações importantes. Estudantes iniciantes apresentaram dificuldades na interpretação das instruções e na navegação pelas fases, apontando a necessidade de aprimorar os feedbacks visuais, mensagens de erro e mecanismos de orientação. Ademais, a aplicação do jogo com estudantes do espectro autista e com dificuldades cognitivas revelou a necessidade de avanços significativos em termos de acessibilidade e adaptação a diferentes perfis de aprendizagem.

Dessa forma, conclui-se que o *CodeQuest*, no contexto em que foi avaliado, apresentou potencial como ferramenta de apoio ao ensino de lógica de programação, atendendo aos objetivos propostos para este trabalho. Contudo, os resultados obtidos não permitem generalizações amplas, indicando a necessidade de estudos adicionais, com amostras maiores e acompanhamentos de longo prazo, para avaliar de forma mais consistente o impacto do jogo no aprendizado e sua efetiva contribuição em relação aos métodos tradicionais.

Como trabalhos futuros, recomenda-se: (i) a ampliação do conjunto de fases e conteúdos abordados; (ii) o aprimoramento de mecanismos de acessibilidade e tutorialização; (iii) a incorporação de mensagens de erro mais próximas das apresentadas em ambientes reais de desenvolvimento; e (iv) a realização de avaliações longitudinais que permitam verificar a retenção do conhecimento adquirido e sua aplicação em contextos formais de ensino de programação.

Assim, este projeto contribui para a área de tecnologias educacionais ao apresentar uma proposta de integração entre programação textual e elementos de RPG, oferecendo evidências iniciais de que esse tipo de abordagem pode enriquecer experiências de aprendizagem, desde que associada a investigações contínuas e aprimoramentos futuros.

7. Referências

Referências

ALONSO, Élida Froes; FIGUEIREDO, Helenara Regina Sampaio. Evasão em cursos técnicos na área de informática: revisão de literatura de 2015 a 2019. **Educitec -**

Revista de Estudos e Pesquisas sobre Ensino Tecnológico, v. 8, e196022–e196022, 2022.

AZAMBUJA, Celso Cândido de; SILVA, Gabriel Ferreira da. Novos desafios para a educação na Era da Inteligência Artificial. **Filosofia Unisinos – Unisinos Journal of Philosophy**, Universidade do Vale do Rio dos Sinos (UNISINOS), v. 25, n. 1, p. 1–15, 2024. Acesso em: 3 nov. 2025. Disponível em: https://revistas.unisinos.br/index.php/filosofia/article/view/27063.

CASTRO, Maria Beatriz de Oliveira; SANTOS, Viviane Almeida dos. Gamificação como recurso para aprimorar o ensino de lógica de programação em cursos de computação no ensino superior: uma revisão sistemática. **Anais da Jornada Científica e Tecnológica do Campus Tucuruí**, v. 4, n. 1, p. 1–15, 2024. Acesso em: 6 ago. 2025. Disponível em: https://www.even3.com.br/anais/jctufpa2024/537716-gamificacao-como-recurso-para-aprimorar-o-ensino-de-logica-de-programacao-em-cursos-de-computacao-no-ensino-superior--uma-revisao-sistematica/.

DETERDING, Sebastian *et al.* From Game Design Elements to Gamefulness: Defining "Gamification". *In:* PROCEEDINGS of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments. [*S. l.*]: ACM, 2011. p. 9–15. DOI: 10.1145/2181037.2181040.

FIGUEIREDO, José; GARCÍA-PEÑALVO, Francisco José. Increasing student motivation in computer programming with gamification. *In:* 2020 IEEE Global Engineering Education Conference (EDUCON). [*S. l.*]: IEEE, 2020. p. 997–1000. DOI: 10.1109/EDUCON45650.2020.9125311.

JORNAL O CASARÃO (UFF). **Os impactos do uso da inteligência artificial no desenvolvimento de crianças e adolescentes**. Acesso em: 3 nov. 2025. 2025. Disponível em: https://jornalocasarao.uff.br/2025/06/11/os-impactos-do-uso-da-inteligencia-artificial-no-desenvolvimento-de-criancas-e-adolescentes/.

JOSÉ ALFREDO CARVALHO. **IA atrapalha o aprendizado? Estudo aponta riscos para jovens**. Acesso em: 3 nov. 2025. 2024. Disponível em: https://redept.com.br/blogosfera/ia-atrapalha-o-aprendizado-estudo-aponta-riscos-para-jovens/.

KROUSTALLI, Chrysoula; XINOGALOS, Stelios. Studying the effects of teaching programming to lower secondary school students with a serious game: a case study with Python and CodeCombat. **Education and Information Technologies**, v. 26, n. 5, p. 6069–6095, 2021. DOI: 10.1007/s10639-021-10596-y.

MOREIRA, Gabriel Luídy *et al.* Desafios na aprendizagem de programação introdutória em cursos de TI da UFERSA, campus Pau dos Ferros: um estudo exploratório. *In:*

ANAIS do Encontro de Computação do Oeste Potiguar (ECOP). Pau dos Ferros, RN: UFERSA, 2018. p. 90–96. Disponível em: https://periodicos.ufersa.edu.br/index.php/ecop/article/view/.

MURATET, Mathieu *et al.* Towards a Serious Game to Help Students Learn Computer Programming. **International Journal of Computer Games Technology**, v. 2009, n. 1, p. 1–7, 2009. DOI: 10.1155/2009/470590.

PEREIRA, Kleber Camara; SOUSA, Reudismam Rolim de. Dificuldades de aprendizagem e fatores que motivam para o estudo de programação: o que mudou desde os estudos de 2018 no curso de TI da UFERSA, Campus Pau dos Ferros? **RECIMA21 – Revista Científica Multidisciplinar**, v. 5, n. 4, p. 1–15, 2024. Acesso em: 3 set. 2025. DOI: 10.47820/recima21.v5i4.5151. Disponível em: https://doi.org/10.47820/recima21.v5i4.5151.

QIAN, Yizhou; LEHMAN, James. Students' misconceptions and other difficulties in introductory programming: A literature review. **ACM Transactions on Computing Education (TOCE)**, v. 18, n. 1, p. 1–24, 2017. DOI: 10.1145/3077618.

QUEIROZ, João Victor; RODRIGUES, Larissa Milena; COUTINHO, Jarbele C. Um relato dos fatores motivacionais na aprendizagem de programação na perspectiva de alunos iniciantes. *In:* ANAIS do ECOP 2018 - Encontro de Computação do Oeste Potiguar. Pau dos Ferros, RN: UFERSA, 2018. p. 162–168. Acesso em: 3 set. 2025. Disponível em: https://periodicos.ufersa.edu.br/index.php/ecop.

SANTOS, Dorlivete da Silva; VIEIRA, Maria Luiza dos Santos. A evasão escolar nos cursos técnicos integrados ao ensino médio: um olhar a partir de estudantes egressos. **Revista de Educação Profissional e Tecnológica em Debate**, v. 2, n. 1, p. 1–21, 2023. Acesso em: 6 ago. 2025. Disponível em: https://seer.ifba.edu.br/index.php/reptd/article/view/1823.

WONG, Yoke Seng; YATIM, Maizatul Hayati Mohamad. A Propriety Multiplatform Game-Based Learning Game to Learn Object-Oriented Programming. **Proceedings of IIAI-AAI**, p. 278–283, 2018. DOI: 10.1109/IIAI-AAI.2018.00060.

ZHAN, Zehui *et al.* The effectiveness of gamification in programming education: Evidence from a meta-analysis. **Computers and Education: Artificial Intelligence**, p. 100096, 2022. DOI: 10.1016/j.caeai.2022.100096.

A. Fase da Roleta de Perguntas — Tipos de Variáveis

Figura 10. Fase da roleta de perguntas do jogo *CodeQuest*, na qual o jogador responde a questões conceituais sobre fundamentos da programação, como tipos numéricos, valores booleanos, strings e regras de nomenclatura de variáveis. O avanço na fase ocorre apenas após a seleção correta das respostas, reforçando a assimilação conceitual antes da aplicação prática. Fonte: Captura de tela do autor (2025).

B. Fase de Manipulação e Ordenação de Arrays

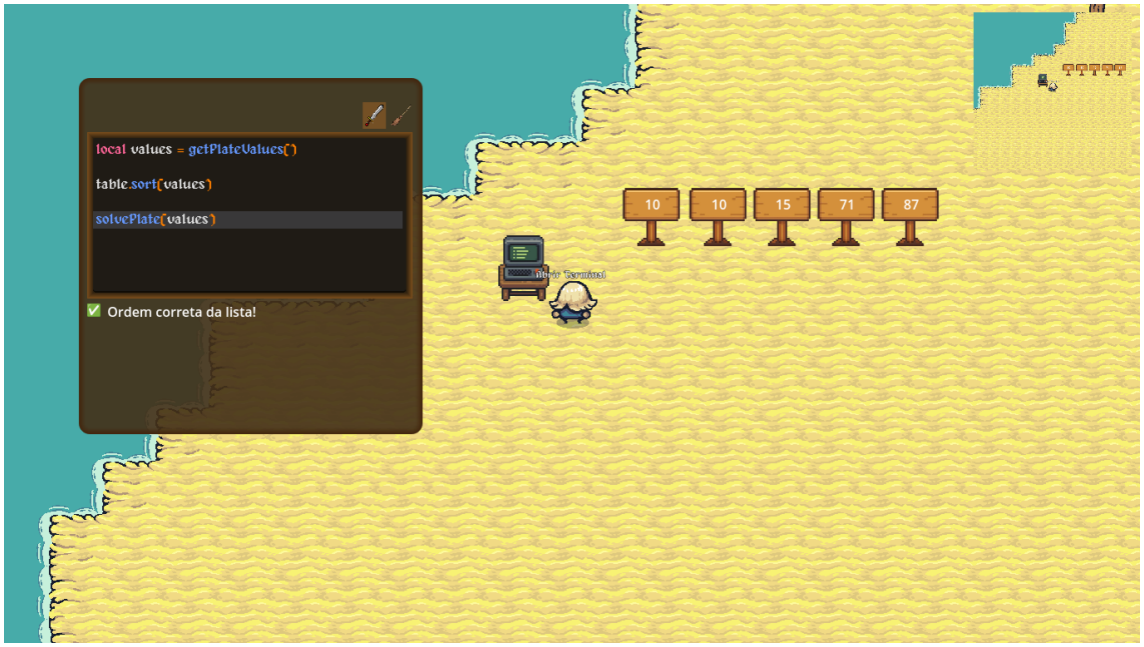

Figura 11. Fase do jogo *CodeQuest* dedicada à manipulação e ordenação de listas (arrays). O jogador deve aplicar operações sobre coleções de dados, como a ordenação de valores, para reorganizar visualmente os elementos do cenário e liberar a progressão na fase, estabelecendo a relação direta entre código e comportamento do ambiente.
Fonte: Captura de tela do autor (2025).