Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Jogo Sério para Reforço de Conceitos Introdutórios de Programação e Pensamento Computacional

Igor Miranda Fam

JUIZ DE FORA
MARÇO, 2025

Jogo Sério para Reforço de Conceitos Introdutórios de Programação e Pensamento Computacional

Igor Miranda Fam

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Marcelo Caniato Renhe

JUIZ DE FORA

MARÇO, 2025

Jogo Sério para Reforço de Conceitos Introdutórios de Programação e Pensamento Computacional

Igor Miranda Fam

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Marcelo Caniato Renhe
Doutor em Engenharia de Sistemas e Computação

Bárbara de Melo Quintela
Doutora em Modelagem Computacional

Alessandreia Marta De Oliveira Julio
Doutora em Computação

JUIZ DE FORA
13 DE MARÇO, 2025

Resumo

A evasão em disciplinas introdutórias de programação é um problema comum em cursos de graduação na área de computação. Isto decorre, em grande parte, do desencorajamento dos estudantes perante os desafios iniciais enfrentados nestas disciplinas. Uma das formas de incentivar os iniciantes são jogos com propósito educativo, cuja eficácia depende de suas características. Para lidar com esse desafio, é proposto o desenvolvimento do jogo sério *Snake Case*, que busca integrar o aprendizado de programação com elementos lúdicos, visando aumentar o engajamento dos alunos e melhorar a qualidade da aprendizagem. A pesquisa analisou a diversão proporcionada pelo jogo e sua contribuição para o aprendizado de programação, segundo a percepção dos alunos. Por meio da aplicação de um questionário baseado no modelo MEEGA+ (PETRI; WANGENHEIM; BORGATTO, 2017), foi avaliada a influência do aspecto lúdico nesse processo, apresentando resultados favoráveis no geral.

Palavras-chave: Jogo sério, jogo educativo, ensino de programação, pensamento computacional.

Abstract

Academic evasion in introductory programming courses is a common issue in undergraduate computer science programs. This is largely due to students feeling discouraged by the initial challenges faced in these courses. One way to motivate beginners is through educational games, whose effectiveness depends on their characteristics. To address this challenge, the development of the serious game Snake Case is proposed, aiming to integrate programming learning with playful elements to increase student engagement and improve the quality of learning. The research analyzed the fun provided by the game and its contribution to programming learning, based on students' perceptions. By using a questionnaire based on the MEEGA+ model (PETRI; WANGENHEIM; BORGATTO, 2017), the influence of the game's playful aspects on this process was evaluated, with favorable results in general.

Keywords: Serious game, educational game, programming education, computational thinking.

Agradecimentos

A meus familiares, principalmente meus pais, por sempre acreditarem em mim e me incentivarem a alcançar meus sonhos, além de proverem toda a estrutura necessária para o meu desenvolvimento.

A minha noiva por todo o amor e apoio, que tornam minha vida mais bonita, e sem os quais os momentos difíceis se tornariam impossíveis.

A meus amigos pelas discussões e reflexões enriquecedoras, e pelas risadas que tornaram a caminhada mais agradável.

Ao professor Marcelo Caniato pela orientação, amizade e por contribuir para o crescimento da comunidade de desenvolvimento de jogos local.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

Conteúdo

Lista de Figuras

Lista de Abreviações

DCC	Departamento de Ciência da Computução
UFJF	Universidade Federal de Juiz de Fora
MEEGA+	*Model for Evaluating Educational Games*
TRI	Teoria da Resposta ao Item

1 Introdução

A programação é uma habilidade cada vez mais valorizada no mundo contemporâneo, dada a sua crescente relevância no contexto tecnológico. A digitalização de quase todos os setores da sociedade moderna amplificou a necessidade de profissionais capazes de compreender e desenvolver tecnologias de ponta. Nesse contexto, as disciplinas introdutórias à programação assumem um papel fundamental ao estabelecer os fundamentos da lógica de programação e do pensamento computacional.

A compreensão desses conceitos iniciais é crucial para a formação de uma base sólida de habilidades necessárias para o desenvolvimento de software e a resolução de problemas complexos em diversas áreas. No entanto, os desafios enfrentados por muitos estudantes nessa etapa inicial frequentemente resultam em altas taxas de evasão e desistência em cursos de computação, representando um obstáculo significativo para a formação de profissionais competentes em tecnologia da informação, conforme alertado por Gomes (2010):

> Essas disciplinas exigem dos alunos habilidades como interpretação e resolução de problemas, raciocínio lógico, capacidade de abstrair soluções e aplicá-las com o uso de uma linguagem de programação, dentre outros. Diante dessas habilidades, muitos estudantes acabam apresentando um baixo desempenho nessas disciplinas, conduzindo-os à reprovação ou, até mesmo, à desistência do referido curso (GOMES, 2010).

O uso de tecnologias educacionais, como os jogos sérios, emergiu como uma estratégia promissora para abordar esse problema, combinando o aspecto lúdico com o processo de aprendizado. No entanto, segundo Oliveira (2016), a eficácia desses jogos no ensino de programação pode ser limitada quando não há uma integração equilibrada entre o conteúdo educacional e os elementos envolventes e divertidos do jogo. Ainda de acordo com o estudo de Oliveira (2016), poucos dos métodos de desenvolvimento de jogos educacionais mencionam a diversão como um dos principais objetivos, o que explica o caráter conteudista desses jogos.

Neste contexto, o presente trabalho se concentra no desenvolvimento do jogo

sério *Snake Case*[1], cujo propósito é introduzir o jogador ao pensamento computacional e apresentar conceitos básicos de programação de forma envolvente. A intenção é aumentar o interesse dos alunos nas disciplinas introdutórias de programação e, potencialmente, contribuir para a consolidação dos conceitos fundamentais desde o início de sua formação acadêmica. Contudo, ressalta-se que o trabalho não avalia especificamente a compreensão ou retenção desses conceitos ao longo do tempo — o foco está, sobretudo, na proposta de design e na experiência motivacional do jogo.

1.1 Descrição do Problema

É importante notar que há jogos educativos que falham em conquistar os usuários devido ao excessivo foco no conteúdo didático em detrimento do aspecto lúdico, como menciona Oliveira (2016). Um exemplo notório disso é a tentativa da Nintendo na década de 1990 com o jogo "Mario is Missing", que visava ensinar conteúdos de geografia. De acordo com o artigo de McFerran (2023), devido a sua jogabilidade lenta e falta de ação, o jogo foi amplamente mal recebido e considerado muito inferior aos títulos principais da franquia, que tinham como único propósito o entretenimento.

Em seu artigo sobre as deficiências dos jogos educativos em comparação com os de entretenimento, Costa (2009) afirma que "há uma tendência em utilizar, quase que aleatoriamente, estruturas de jogos de entretenimento já consagrados para relacionar o conteúdo dos objetos de conhecimento, sem levar em conta as estruturas destes". Em outras palavras, há uma tentativa de aplicar fórmulas de jogos populares ao conteúdo que se deseja ensinar, resultando frequentemente em um produto que falha em divertir e em educar. O exemplo de "Mario is Missing" tentou incorporar o tema da geografia ao gênero clássico de plataforma dos jogos do Mario, o que resultou em uma experiência tediosa sobreposta a uma narrativa mal estruturada que, de forma grosseira, tentou integrar elementos do jogo e do universo da franquia a localizações do mundo real.

Diante dessa conjuntura, este estudo busca explorar a viabilidade de um jogo sério como uma ferramenta de aprendizado de programação, com ênfase na reconciliação entre o aspecto educacional e o lúdico. A intenção é avaliar a importância da diversão e

[1]Disponível em ⟨https://myrand-kk.itch.io/snake-case⟩

do engajamento gerados pelo jogo para a qualidade do aprendizado proporcionado.

1.2 Justificativa

A justificativa para este estudo é respaldada pela necessidade urgente de melhorar a adesão dos estudantes às disciplinas introdutórias de programação e, ao mesmo tempo, assegurar que a aprendizagem seja eficaz. O alto índice de evasão em tais cursos representa um problema substancial para as instituições de ensino e, consequentemente, para o setor de tecnologia da informação. A introdução do jogo sério *Snake Case* se justifica como uma tentativa de abordagem inovadora que integra o aspecto lúdico e o ensino de programação, a fim de cativar os alunos desde o início de sua jornada acadêmica e, por conseguinte, melhorar a qualidade do aprendizado.

A tecnologia é uma parte integral de nossa sociedade, e os profissionais de TI desempenham um papel fundamental em seu desenvolvimento e manutenção. Portanto, é essencial que o processo de formação desses profissionais seja atrativo e eficaz, garantindo a formação de profissionais qualificados e motivados.

1.3 Objetivos

Este trabalho tem como objetivo principal desenvolver e avaliar a eficácia do jogo sério *Snake Case* como uma ferramenta de ensino de programação, visando aumentar o interesse dos alunos e melhorar a retenção de conceitos nas disciplinas introdutórias à programação. Como objetivos específicos, podemos citar:

- desenvolver o jogo sério *Snake Case* com foco na introdução de conceitos básicos de programação de forma lúdica e envolvente;
- analisar a aceitação e a adesão de estudantes ao jogo como uma ferramenta de aprendizado de programação;
- avaliar a usabilidade e experiência do usuário provida pelo jogo através do modelo MEEGA+ (PETRI; WANGENHEIM; BORGATTO, 2017).

1.4 Organização do Trabalho

O presente trabalho está estruturado da seguinte forma: no Capítulo 2, são discutidas as bases teóricas que fundamentam o ensino de programação e a aplicação de jogos sérios como ferramentas educacionais, evidenciando os benefícios do aspecto lúdico e as principais abordagens na literatura. Em seguida, o Capítulo 3 traz a revisão bibliográfica, na qual são apresentados estudos e exemplos práticos de jogos sérios aplicados ao ensino, bem como os critérios de qualidade e efetividade destes jogos. O Capítulo 4 descreve o desenvolvimento do jogo, detalhando a narrativa, as mecânicas, o design dos níveis e a estrutura do projeto, de forma a evidenciar como os conceitos de programação foram integrados à experiência lúdica. No Capítulo 5, é descrito o processo de avaliação do jogo por meio do modelo MEEGA+, com a apresentação e discussão dos resultados quantitativos e qualitativos, além das ameaças à validade dos achados. Por fim, o Capítulo 6 oferece a conclusão, na qual são sintetizados os principais resultados, discutidas as limitações do estudo e possíveis linhas de trabalho futuras.

2 Fundamentação Teórica

Este capítulo estabelece a base teórica do trabalho, oferecendo uma análise detalhada dos conceitos e metodologias que embasam o uso de jogos sérios no ensino de programação. Nele, são abordados os desafios inerentes ao ensino tradicional de programação e como a inserção de elementos lúdicos pode transformar o processo de aprendizagem, tornando-o mais envolvente e eficaz. A discussão se estende para a caracterização dos jogos sérios, diferenciando-os de estratégias de gamificação e ressaltando suas aplicações não apenas na educação, mas também em áreas como saúde e desenvolvimento de *soft skills*. Além disso, o capítulo apresenta o modelo MEEGA+ (PETRI; WANGENHEIM; BORGATTO, 2017), uma ferramenta validada e robusta que permite a avaliação sistemática da qualidade dos jogos educacionais, através da mensuração de aspectos como engajamento, usabilidade e aprendizado percebido. Com essa fundamentação teórica abrangente, o capítulo prepara o terreno para a análise crítica e a aplicação prática dos conceitos discutidos nos capítulos subsequentes.

2.1 Ensino de Programação

O ensino de programação é uma área de importância crescente, uma vez que as habilidades de codificação são cada vez mais necessárias em diversas profissões e setores da sociedade. No entanto, o ensino tradicional de programação pode ser desafiador e muitas vezes desmotivador para os alunos, especialmente os iniciantes. Nesse contexto, os jogos sérios surgem como uma solução inovadora para tornar o aprendizado de programação mais atraente, envolvente e eficaz, como sugere o estudo de Oliveira e Boff (2024).

O aspecto lúdico dos jogos sérios desempenha um papel fundamental na motivação dos alunos em relação ao aprendizado de programação. Ao proporcionar um ambiente de aprendizado que é, ao mesmo tempo, desafiador e divertido, os jogos educativos criam uma experiência envolvente. Os desafios apresentados, as recompensas a serem conquistadas e a possibilidade de competir com outros jogadores incentivam a participação

ativa e o aprofundamento no conteúdo. Isso não apenas mantém os alunos envolvidos, mas também pode inspirá-los a explorar e entender os conceitos de programação com entusiasmo.

O envolvimento lúdico promovido pelos jogos sérios facilita o aprendizado ativo, de acordo com Pacheco (2024). Quando os alunos podem experimentar a programação na prática, resolver problemas reais e observar resultados imediatos de suas ações, a compreensão conceitual e a retenção do conhecimento são fortalecidas. Esse tipo de aprendizado prático, no qual os alunos são incentivados a aplicar ativamente o que aprenderam, tem-se mostrado eficaz na criação de uma base sólida de habilidades de programação desde o início de sua formação acadêmica, como demonstra o estudo de Santos et al. (2017).

2.2 Jogos Sérios

Os jogos sérios (do inglês "*serious games*"), também conhecidos como jogos com propósito, são jogos projetados com a finalidade principal de ensinar, treinar ou informar os jogadores, em contraste com jogos puramente recreativos (BECKER, 2005).

O estudo realizado por Lara et al. (2023) analisa a produção acadêmica nacional acerca da utilização de jogos sérios na educação. Segundo os autores, a repetição do ato de jogar – que torna a atividade lúdica prazerosa – pode se configurar como um forte aliado no processo ensino-aprendizagem. Os experimentos apresentados indicam resultados promissores quanto à eficácia dos jogos sérios na assimilação de conteúdos, ao mesmo tempo em que proporcionam motivação e entretenimento aos alunos.

Os jogos sérios transcendem o âmbito educacional, encontrando aplicações significativas em diversas áreas estratégicas. No setor de saúde, por exemplo, podem ser empregados na reabilitação de pacientes. O trabalho de Filho e Jucá (2015) detalha o desenvolvimento de um jogo sério que utiliza o dispositivo Kinect para tornar o processo de fisioterapia mais envolvente e eficaz, beneficiando pacientes com espondilite anquilosante. No desenvolvimento de soft skills, esses jogos oferecem ambientes interativos para aprimorar habilidades como comunicação, trabalho em equipe e resolução de problemas, sendo utilizados em treinamentos corporativos e programas de capacitação (GUIMARãES et al., 2024). Dessa forma, os jogos sérios se consolidam como ferramentas versáteis e eficazes

em diferentes contextos, ampliando suas aplicações para além do aprendizado tradicional.

É importante salientar a diferença entre jogo sério e gamificação. De acordo com o artigo de Dantas et al. (2022), jogos sérios são projetados para fins educacionais ou instrucionais, mantendo características estruturais de um jogo tradicional, como objetivos claros, desafios, regras, interatividade e mecanismos de feedback. Já a gamificação não se trata de um jogo completo, mas sim da aplicação de elementos de jogos, como recompensas, rankings e desafios, em contextos não lúdicos, como a educação ou o ambiente corporativo, para engajar e motivar os participantes. Enquanto um jogo sério se assemelha a um jogo convencional com um propósito educativo explícito, a gamificação utiliza apenas algumas das mecânicas dos jogos para incentivar o aprendizado ou a realização de tarefas.

Os elementos lúdicos nos jogos educativos incluem desafios que estimulam a resolução de problemas, recompensas que incentivam o progresso e narrativas envolventes que prendem a atenção dos jogadores. Esses componentes criam uma experiência de aprendizado que envolve os jogadores, aumentando seu interesse e motivação, além de estimular a persistência e a superação de obstáculos (MEROTO et al., 2024).

2.3 Métodos de avaliação de jogos sérios

Para maximizar o potencial educativo de jogos sérios, é necessário estabelecer critérios para avaliar sua efetividade. Nesta seção, são citados dois modelos de avaliação de jogos sérios que destacam os critérios de qualidade a serem atingidos para que o jogo seja capaz de efetivamente engajar os jogadores e transmitir seu conteúdo.

2.3.1 Abdellatif, McCollum e McMullan (2018)

Abdellatif, McCollum e McMullan (2018) desenvolveram um *framework* para a avaliação da qualidade e efetividade de jogos sérios dividido em cinco dimensões: usabilidade, motivação, engajamento, experiência do usuário e compreensibilidade. A primeira mede a facilidade de aprendizado do conteúdo proposto pelo jogo, a jogabilidade e a presença de erros. A segunda trata do desafio, da diversão e curiosidade proporcionados ao jogador,

garantindo que, apesar do propósito primário do jogo ser o ensino, ele não falhe em divertir o usuário e aumentar seu interesse sobre o conteúdo. A terceira avalia a atratividade dos objetivos e cenários propostos, além do impacto das ações e decisões do jogador no resultado obtido no jogo. A quarta determina o grau de imersão dos jogadores e o quanto o jogo estimula a interação entre usuários e o compartilhamento de suas experiências. Por fim, a quinta mede a clareza e simplicidade do jogo, garantindo que ele possa ser jogado sem assistência ou suporte de terceiros. Esse *framework* não foi formalmente aplicado na avaliação do jogo *Snake Case*, mas foi levado em consideração durante o desenvolvimento para garantir a qualidade do produto final.

2.3.2 MEEGA+

Para avaliar a qualidade de jogos educacionais voltados especialmente para o ensino de conceitos de computação e engenharia de software, foi elaborado por Petri, Wangenheim e Borgatto (2017) o MEEGA+ (*Model for Evaluating Educational Games*). Esse modelo representa uma evolução do MEEGA, tendo sido refinado para superar limitações da versão anterior, principalmente em relação à validade de suas métricas. O MEEGA+ baseia-se em questionários estruturados aplicados aos jogadores e utiliza a Teoria de Resposta ao Item (TRI) para gerar escores que mensuram aspectos fundamentais da experiência do usuário, como engajamento, imersão, motivação e percepção de utilidade do jogo.

A estrutura do MEEGA+ fundamenta-se em dois grandes fatores de qualidade: experiência do jogador e usabilidade. O primeiro abrange elementos como atenção focada, diversão, desafio, interação social, confiança, relevância, satisfação e aprendizagem percebida. Já a usabilidade contempla estética, aprendibilidade, operabilidade e acessibilidade. Para garantir sua robustez, o modelo foi desenvolvido e validado a partir de 48 estudos de caso envolvendo 843 alunos, com avaliação de 18 jogos diferentes. A análise estatística demonstrou um alto nível de confiabilidade, com um coeficiente alfa de Cronbach de 0,928, evidenciando a consistência interna elevada dos itens avaliados.

O processo de aplicação do MEEGA+ inclui a coleta de dados por meio de um questionário padronizado, cujas respostas são analisadas em uma escala de cinco pontos, permitindo uma classificação que varia de “baixa qualidade” a “excelente qualidade”.

Esse método permite não apenas quantificar a qualidade do jogo, mas também identificar oportunidades de melhoria, fornecendo informações valiosas para pesquisadores e desenvolvedores ajustarem e refinarem seus jogos educacionais. Além disso, o modelo demonstrou validade convergente e discriminante, garantindo que suas métricas capturam efetivamente os aspectos críticos da experiência de jogo e usabilidade.

Dessa forma, o MEEGA+ se estabelece como um modelo confiável e sistemático para a avaliação de jogos educacionais, oferecendo uma estrutura bem definida e baseada em evidências empíricas para auxiliar na melhoria contínua desses recursos de ensino.

2.4 Considerações finais

Este capítulo forneceu uma base teórica sólida para o estudo sobre o uso de jogos sérios no ensino de programação, com ênfase no impacto do aspecto lúdico na adesão, interesse e qualidade do aprendizado. Os jogos sérios representam uma abordagem promissora para tornar o ensino de programação mais atraente e eficaz, motivando os alunos a se envolverem de maneira mais profunda e significativa com o conteúdo. No próximo capítulo, é apresentada uma revisão da literatura sobre trabalhos que aplicam os conceitos e ideias que foram abordados.

3 Revisão Bibliográfica

Este capítulo realiza uma revisão de trabalhos científicos que investigam a aplicação e os benefícios dos jogos sérios no ensino de programação. Nele, são explorados diversos exemplos práticos, comparando abordagens e soluções adotadas. Essa análise crítica não apenas evidencia as potencialidades dos jogos sérios como ferramentas de ensino, mas também identifica desafios e limitações, contribuindo para a construção de diretrizes e estratégias que visam aprimorar o aprendizado de programação por meio da ludicidade e da interatividade.

Os trabalhos citados foram encontrados através de buscas no IEEE Xplore e Google Scholar, com as palavras-chave "Serious game" e "Programming". Além disso, foi feita uma revisão manual das trilhas de educação das edições dos quatro anos anteriores do Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames). Dentre os trabalhos encontrados, aqueles discutidos na seção a seguir foram os que mais se aproximaram da presente proposta.

3.1 Jogos Sérios no Ensino de Programação

Um exemplo de jogo sério que se concentra no ensino de conceitos de programação é Maze Code, por Neto et al. (2021). No jogo, o jogador deve explorar a área disponível e coletar itens que representam fragmentos de código, e então organizá-los de forma a resolver um enigma e liberar a próxima área. Nesses enigmas, o jogador deve interagir com computadores para abrir interfaces nas quais, através de uma dinâmica *drag and drop*, pode utilizar os itens que coletou para formar códigos que satisfaçam ao desafio proposto e acionar um botão para executá-los, verificando o sucesso ou fracasso em cumprir o objetivo. Essa abordagem atende muito bem, principalmente, o critério de motivação proposto pelo framework de Abdellatif, McCollum e McMullan (2018), pois instiga a curiosidade e criatividade do usuário através da exploração do ambiente e da agregação dos itens coletados para solucionar desafios. Além disso, os problemas apresentados ao

jogador são divididos em cinco níveis, de forma a reforçar o aprendizado de forma gradativa e evitar a necessidade de auxílio externo para prosseguir nos desafios, como é proposto no critério de compreensibilidade de Abdellatif, McCollum e McMullan (2018). No entanto, o fato de os enigmas serem abertos em uma nova tela pode levar à quebra da imersão do usuário, pois interrompe o fluxo da jogabilidade proposta na fase de exploração do mapa.

Já em Hello Food, por Macena et al. (2022), o problema acima não pode ser encontrado, pois os desafios são apresentados dentro do ambiente do jogo, e resolvidos através da interação com os elementos do mundo. O jogo consiste em uma série de desafios de pensamento computacional integrados a uma ambientação ligada à culinária, de forma que os pratos são preparados através de algoritmos construídos pelo jogador, exercitando, principalmente, os conceitos de estruturas condicionais, laços de repetição e vetores. Hello Food se destaca por sua ambientação e integração de elementos de *puzzle* à temática proposta. Ao comparar estruturas da programação a elementos de culinária, o jogo facilita o entendimento do jogador sobre o conteúdo proposto ao abstraí-lo e inseri-lo em um contexto simples. O jogo também se compromete com o aprendizado do usuário, visto que, ao final de cada fase, disponibiliza feedback a respeito dos erros cometidos pelo jogador.

O jogo desenvolvido por Frosi e Silva (2019), intitulado CodeBots, tem como principais aspectos de qualidade o impacto das ações do jogador no resultado do jogo e a interação e competitividade entre jogadores. O jogo consiste em uma arena de combate entre robôs, no qual o jogador é responsável por programar as ações de seu robô (atirar, virar, andar ou parar) diante de determinados estímulos, como encontrar outro robô combatente, encontrar um item de cura ou se aproximar de uma parede. Para facilitar o processo de aprendizado, o jogo conta com um modo treino, no qual é possível customizar a programação de seu robô em tempo real sem que haja outras ameaças na arena. Dessa forma, é possível testar diferentes algoritmos para o comportamento do robô e decidir qual se sairá melhor em batalha. No entanto, é mencionado pelos autores que a ferramenta não possui a capacidade de proporcionar o aprendizado por si só: seu uso deve ser acompanhado por um docente que problematizará situações a serem resolvidas pelos alunos na plataforma. Além disso, como o jogador não possui informações sobre

os comportamentos dos inimigos, as partidas podem se tornar imprevisíveis e frustrantes, já que é necessário programar as ações de seu robô antes do início da batalha, e não é possível alterá-las no decorrer da partida.

Code Saga, de Tacouri e Nagowah (2021), é um jogo para ensino de programação com quatro modos. No primeiro, chamado *Treasure Hunt*, o jogador deve coletar livros, que contêm notas sobre conceitos básicos sobre programação, como variáveis e tipos de dados, e, ao encontrar pergaminhos, deverá responder a um quiz sobre os assuntos estudados. No segundo, *JavaWars*, o jogador usa loops e condições para ajudar o herói principal a derrotar inimigos. No terceiro, intitulado *Code Block*, o jogador precisa, através de uma mecânica *drag and drop*, organizar blocos de código para formar determinados algoritmos. *Sorting Algo*, o último modo, proporciona uma visualização, através de esferas coloridas, do funcionamento de algoritmos de ordenação. Apesar de proporcionar um aprendizado gradual através de quatro modos que tratam de assuntos de complexidades crescentes, o jogo não ensina o conteúdo necessário para que o usuário saiba resolver os desafios apresentados, o que fere os critérios de usabilidade e compreensibilidade propostos por Abdellatif, McCollum e McMullan (2018). Além disso, o primeiro e o último modo de jogo consistem, em grande parte, apenas na observação e leitura dos conteúdos disponibilizados, o que pode ter um impacto negativo no engajamento dos jogadores, segundo o framework de avaliação mencionado.

Program Your Robot, apresentado em (KAZIMOGLU, 2020), é um jogo de quebra-cabeça educativo que oferece uma experiência única para o desenvolvimento do pensamento computacional. Nele, o jogador deve programar um robô através de comandos simples, como mover para frente, girar à direita ou acender uma luz, para coletar os itens necessários e chegar ao final da fase. Esse processo ensina os conceitos fundamentais da lógica de programação, como sequenciamento, repetição e tomada de decisões. O jogo trabalha muito bem o ensino das mecânicas da jogabilidade através de um tutorial e níveis introdutórios simples e intuitivos. Além disso, a temática de programação de um robô combina muito bem com o conteúdo que se deseja ensinar (pensamento computacional), o que é positivo para o sucesso de um jogo educativo, segundo Oliveira (2016).

Em Oliveira e Boff (2024), os autores abordam as dificuldades que alunos de

cursos de tecnologia enfrentam no aprendizado de programação, fato que contribui para as elevadas taxas de evasão em disciplinas introdutórias. O estudo propõe uma abordagem baseada no desenvolvimento de um jogo sério que visa estimular o pensamento computacional e facilitar a compreensão dos conceitos básicos de programação – como abstração, lógica, funções e estruturas de repetição. Utilizando o método MEEGA+ (PETRI; WANGENHEIM; BORGATTO, 2017) para avaliação, os resultados apontam que a aplicação do jogo contribui para a criação de um ambiente de aprendizagem mais interativo e motivador, ajudando a mitigar as barreiras que dificultam o aprendizado nessa área.

Por fim, Bezerra et al. (2022) apresenta o desenvolvimento de um jogo sério denominado "Bora Jogar", cujo objetivo é auxiliar na aprendizagem de programação. A proposta envolve a criação de versões tanto físicas (por meio de jogos de tabuleiro) quanto digitais, de modo a proporcionar uma experiência lúdica e interativa que atenda às dificuldades dos alunos na disciplina de algoritmos e lógica de programação. O estudo descreve detalhadamente as etapas de criação do jogo, que se fundamentam em pesquisas prévias sobre os obstáculos enfrentados pelos estudantes, e destaca que a implementação da ferramenta contribuiu para tornar o aprendizado mais acessível e motivador. A avaliação, realizada com métodos quantitativos e qualitativos, indicou que a integração de elementos lúdicos efetivamente favorece a compreensão e a aplicação dos conceitos de programação.

3.2 Comparativo dos Estudos

Ao analisar os jogos apresentados na literatura, é possível identificar diferentes abordagens para o ensino de programação. A imersão e a interatividade são aspectos cruciais para o sucesso de um jogo educativo. O jogo *Snake Case*, desenvolvido neste trabalho, busca manter uma interação constante entre o jogador e os conceitos de programação ao longo de toda a experiência. Em contraste, alguns dos jogos mencionados, como Code Saga (TACOURI; NAGOWAH, 2021) e CodeBots (FROSI; SILVA, 2019), segmentam a interação com a programação em momentos específicos, o que pode reduzir a imersão e a aplicação contínua dos conceitos aprendidos.

Maze Code (NETO et al., 2021), por exemplo, embora integre exploração de am-

biente com resolução de *puzzles*, apresenta seus desafios em uma interface separada. Essa abordagem pode comprometer a fluidez da jogabilidade, pois interrompe a exploração para a resolução dos problemas. Em contrapartida, *Snake Case* integra os desafios diretamente ao cenário, garantindo uma transição mais natural entre o jogo e os conteúdos de programação, favorecendo a imersão do jogador e mantendo a continuidade da experiência de aprendizado.

Outro ponto de destaque é o escopo do conteúdo abordado. Enquanto alguns dos trabalhos revisados, como Program Your Robot (KAZIMOGLU, 2020) e Hello Food (MACENA et al., 2022), concentram-se em conceitos específicos da lógica computacional, como operações aritméticas, estruturação de algoritmos e sequencialidade de instruções, este trabalho adota uma abordagem mais abrangente. O jogo expõe o jogador a trechos de código estruturados de maneira similar à programação convencional, proporcionando um contato mais direto com práticas reais da área.

3.3 Considerações Finais

Em síntese, a revisão bibliográfica evidencia que os jogos sérios têm um potencial significativo para transformar o ensino de programação ao integrar ludicidade e interatividade, embora desafios relacionados à usabilidade, à clareza dos conteúdos e à imersão permaneçam. As diversas abordagens apresentadas apontam para a necessidade de estratégias que conciliem a diversão com o rigor pedagógico, servindo como base para o aprimoramento contínuo das ferramentas educacionais e incentivando futuras pesquisas que visem mitigar as limitações identificadas.

4 Desenvolvimento

Este capítulo apresenta, de forma detalhada e integrada, o processo de desenvolvimento do jogo *Snake Case*. Inicialmente, o capítulo mergulha na construção da narrativa, que acompanha um programador iniciante transportado para um universo digital repleto de desafios. Em seguida, são detalhadas as mecânicas do jogo, desde a coleta e manipulação de blocos de código até a execução de *scripts* que alteram o comportamento de entidades, oferecendo feedback imediato ao jogador. Por fim, é apresentada a estrutura do software, com uma explicação das classes, materiais e comportamentos que sustentam a lógica do jogo e possibilitam uma progressão gradual na aprendizagem dos conceitos de programação. O jogo foi implementado com o motor de jogos Godot 4.3, enquanto os gráficos foram produzidos com o editor de imagens Aseprite.

4.1 Narrativa

A narrativa desenvolvida em *Snake Case* foi desenvolvida ao redor de um programador iniciante, que tem muita dificuldade no início de seu aprendizado de programação. No início do jogo, o personagem principal aparece programando e jogando um jogo em seu computador. Devido à frustração de não conseguir progredir no desenvolvimento de seu programa, o personagem acaba dormindo sobre o teclado e, devido a um erro em seu computador, surge um portal na tela que o puxa para o mundo digital. Então, dentro do computador, o personagem encontra Py, uma cobra amigável que pede ajuda para encontrar seu irmão desaparecido e auxilia o protagonista em sua jornada durante a cena introdutória ao jogo, demonstrada na Figura 4.1.

4.2 Mecânicas do Jogo

No cenário do jogo, há trechos de código incompletos escritos no chão e pequenos blocos coletáveis que contêm valores (como variáveis, valores booleanos, e referências a outros

Figura 4.1: Diálogo com Py na *cutscene* inicial.

elementos do cenário) ou palavras-chave da programação (como condicionais e *loops*). A mecânica central do jogo consiste em completar e executar os códigos incompletos que existem no chão das fases. Para isso, o jogador deve coletar os blocos espalhados pelo cenário e posicioná-los em encaixes nos *scripts* que encontrar. Através desses *scripts*, o jogador pode mover, destruir e desativar os obstáculos que o impedem de chegar à saída da fase.

A principal ideia do jogo é que qualquer código construído pelo jogador deve ter algum impacto no cenário, mesmo que não seja o necessário para a solução do desafio proposto. Assim, é possível que o usuário associe o código a seu funcionamento, tendo um feedback preciso que demonstre o comportamento de cada estrutura da programação. Por exemplo, na primeira fase do jogo, demonstrada na Figura 4.2, é necessário que o jogador construa o código *"laser = false"* para que o laser seja desligado. Assim, ele pode realizar a operação *"exit = true"*, que faz com que a porta de saída seja aberta e a fase possa ser concluída.

Figura 4.2: Primeira fase do jogo.

4.3 Estrutura do projeto

Para elaborar o funcionamento dos códigos construídos pelo jogador e das entidades controladas por eles, foram criadas as classes descritas a seguir. A relação entre elas está ilustrada na Figura 4.3.

4.3.1 *Block* (Bloco)

Blocos são os objetos que podem ser utilizados pelo jogador para completar as lacunas existentes nos *scripts* das fases. Cada bloco pode representar um valor booleano (verdadeiro ou falso), um tipo de entidade (como Árvores, Sapos, Vírus, entre outros), um material (Madeira, Tijolo, Pedra ou Laser) ou uma variável. Os sprites de alguns blocos têm seu nome escrito de forma abreviada devido ao tamanho dos blocos e à baixa resolução do jogo. Na Figura 4.4, estão ilustrados todos os tipos de blocos presentes no jogo.

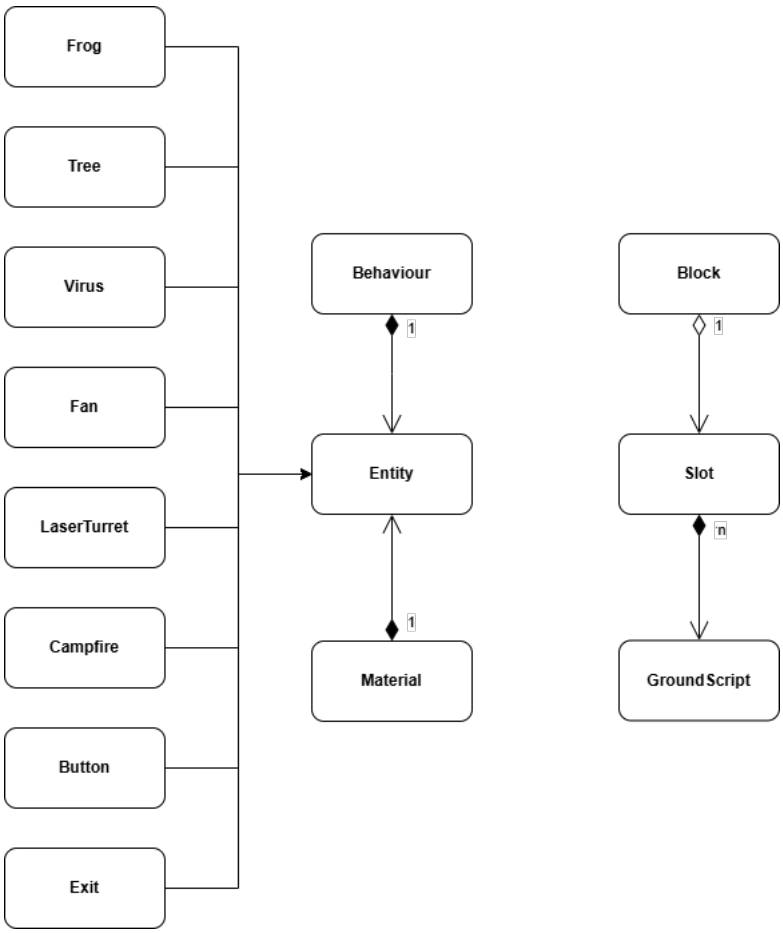

Figura 4.3: Diagrama de classes ilustrando os relacionamentos entre as principais estruturas do jogo.

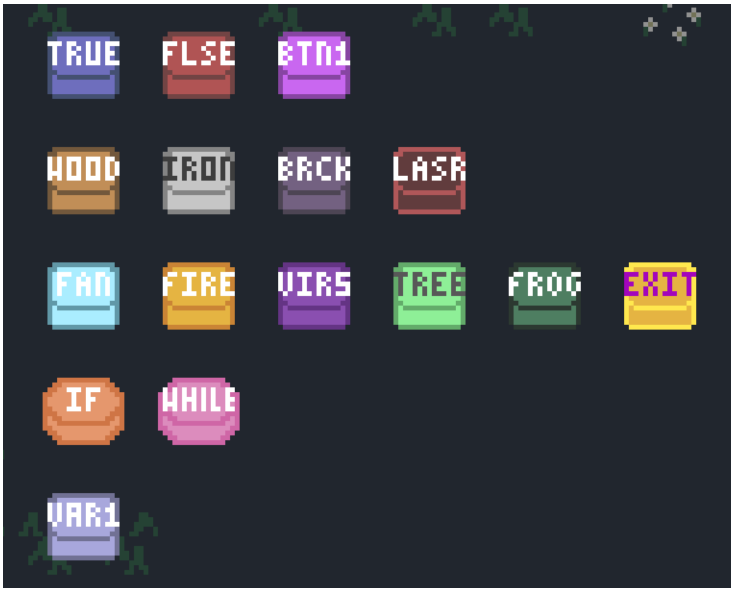

Figura 4.4: Tipos de blocos.

4.3.2 *Slot* (Encaixe)

Os *slots* indicam as posições dos *scripts* que podem ser preenchidas por blocos. Assim que todos os encaixes de um *script* forem preenchidos, a execução do código é iniciada. Então, as linhas se tornarão amarelas à medida que são executadas, para demonstrar ao jogador a sequencialidade da execução do código.

4.3.3 *Ground Script* (Código de Chão)

Os *Ground Scripts* são responsáveis por construir a lógica geral do código que o jogador deve executar para chegar à saída. Ele pode conter atribuições, chamadas de função, estruturas condicionais e loops. Eles contêm um nó filho do tipo *RichTextLabel* (nó da Godot que exibe um texto com parâmetros customizáveis, como cor e tamanho da fonte) que define o texto do código que é exibido para o jogador em uma linguagem simplificada baseada em *Python*. O atributo *code*, que armazena o código em *GDScript* (linguagem executada pelo motor Godot), que contém a lógica que, de fato, será executada.

As atribuições são a principal forma com a qual o jogador pode alterar os elementos do jogo. Nelas, o valor à esquerda sempre deve representar um conjunto de entidades do cenário. Caso esse valor seja "*Frog*", por exemplo, todos os sapos da fase atual receberão o valor à direita. Também é possível que o valor à esquerda seja um material, como "*Wood*". Assim, serão alteradas todas as entidades que tiverem o material "Madeira".

Quando o valor à direita representa um tipo de entidade, todas as entidades referenciadas pelo valor à esquerda da igualdade recebem o comportamento representado pelo valor à direita. No exemplo demonstrado na Figura 4.5, todas as árvores do cenário (*Tree*) passam a se comportar como sapos (*Frog*). Da mesma forma, se o valor à direita for um material, todas as entidades do valor à esquerda passarão a ser feitas desse material.

Já quando o valor à direita for um booleano, o efeito da atribuição dependerá do tipo de cada entidade afetada. Alguns desses tipos farão com que as entidades liguem ou desliguem, e outros farão com que apareçam ou desapareçam se o valor booleano for verdadeiro ou falso, respectivamente. Na Figura 4.2, por exemplo, a atribuição *"laser = false"* faz com que o laser seja desligado. Já na fase demonstrada na Figura 4.7, a atribuição *"tree = false"* faz com que as árvores desapareçam.

Figura 4.5: *Script* que atribui o comportamento de sapo às árvores.

Scripts também podem realizar chamadas de funções. Essas funções podem ser pré-definidas, como "*jump()*", ou definidas por outros scripts, como na fase da Figura 4.10. Quando um *script* que define uma função, todos os *scripts* que a chamam apenas são executados quando todos os encaixes, tanto deles quanto da função, são preenchidos.

4.3.4 *Entity* (Entidade)

O projeto foi organizado de forma que cada tipo de entidade que pudesse ser alterada pelo código construído pelo jogador tivesse sua própria classe, que herda da superclasse *Entity*. A superclasse *Entity* tem como principal finalidade armazenar os atributos que definem o comportamento e o material assumidos pela entidade. Esses atributos alteram as cores que compõem seu *sprite*, e também podem ser visualizados pelo jogador ao posicionar o cursor sobre a entidade, como mostra a Figura 4.6. Já as classes que herdam de *Entity* têm como finalidade definir configurações padrão de comportamento e material para cada tipo de entidade, além de lidar com elementos específicos de um tipo de entidade, como sprites animados, efeitos de partículas e interações específicas com outros elementos do jogo. A seguir são descritos os tipos de entidades e suas especificidades.

Figura 4.6: Rótulo que mostra o comportamento e o material atuais de um sapo.

Tree **(Árvore)**

Árvores são inicializadas com o material Madeira e o comportamento *TreeBehaviour*. São entidades majoritariamente estáticas, que se movem apenas caso empurradas por ventiladores.

Frog **(Sapo)**

Sapos possuem, por padrão, o material Madeira e o comportamento *FrogBehaviour*. Além disso, possuem o atributo *target*, que permite que um alvo seja anexado ao sapo. Caso possua um alvo, o sapo alternará entre sua posição e a posição do alvo ao receber o comando *jump()* de um código feito pelo jogador.

LaserTurret **(Torre de Laser)**

Torres de Laser são entidades que iniciam com o material Ferro e o comportamento *TurretBehaviour*. Em seu *script* de inicialização, é atribuído a elas um raio Laser que pode ser ligado e desligado pelo jogador através de *scripts*. Os raios causam dano a entidades com materiais quebráveis e impedem a passagem do jogador, empurrando-o para trás.

Campfire (Fogueira)

As fogueiras iniciam com o material Madeira e o comportamento *CampfireBehaviour*. Possuem um nó filho do tipo *Area2D*, que representa uma chama, e são as únicas entidades que não desaparecem ao pegar fogo. Também podem ter suas chamas desligadas e ligadas através de códigos.

Button (Botão)

Os botões possuem, de início, o material Ferro, e são as únicas entidades que não possuem um comportamento associado, pois não podem ser controlados pelos códigos do jogador. Assim, toda sua lógica é construída dentro da própria classe. Os botões não se movem, e detectam colisões com o jogador e quaisquer outras entidades que sejam posicionadas sobre eles. Sua função é representar valores booleanos (verdadeiro ou falso) que podem ser acessados nos *scripts* e variam caso o botão esteja sendo pressionado ou não, respectivamente.

Vírus

Os Vírus são inicializados com o material Tijolo e o comportamento *VirusBehaviour*. São inimigos que, caso tocados pelo jogador, reiniciam o nível atual. Possuem três tipos de locomoção: estático, no qual não saem do lugar; linear, no qual se movem em linha reta e invertem o sentido do movimento ao colidir com algo; e perseguidor, no qual ativamente perseguem o jogador.

Fan (Ventilador)

Os ventiladores são feitos do material Ferro e possuem o comportamento *FanBehaviour*. Possuem como nó filho uma área que representa o vento emitido por eles. Ao entrar em contato com o vento, o jogador e entidades feitas de materiais não pesados são empurrados para longe do ventilador.

Saída

As portas de saída representam o objetivo final de cada nível. Possuem o material Ferro e o comportamento *ExitBehaviour*, e são as únicas entidades que não podem ter seu comportamento ou material alterados. Podem ser abertas ou fechadas via *script*.

4.3.5 Material

Os materiais determinam como as entidades interagem com outras entidades. Cada material pode ser ou não inflamável, quebrável e pesado. Entidades feitas de materiais inflamáveis pegam fogo ao tocar outras entidades que estejam em chamas. Após um determinado tempo em chamas, as entidades desaparecem, com exceção das fogueiras. Entidades feitas de materiais quebráveis recebem dano ao entrar em contato com Lasers, e podem ser quebradas dessa forma. Entidades pesadas não são movidas por ventiladores.

Os materiais definidos para o jogo são:

- **Madeira:** inflamável, quebrável e não pesada;
- **Tijolo:** não inflamável, quebrável e pesado;
- **Ferro:** não inflamável, não quebrável e pesado;
- **Laser:** não inflamável, não quebrável e não pesado.

4.3.6 *Behaviour* (Comportamento)

Comportamentos são classes que definem como as entidades agem ao receberem comandos ou interagirem com o jogador e outras entidades. Definem também a cor que as entidades que os assumirem recebem, além do nome e do ícone do comportamento que são exibidos ao posicionar o cursor sobre uma entidade. Herdam da superclasse *Behaviour*, que define as ações padrão de cada entidade, que podem ser sobrescritas por suas subclasses. São elas:

TreeBehaviour

Não altera o funcionamento de nenhuma ação do comportamento padrão, e é utilizado apenas para definir a cor, ícone e nome das árvores.

FrogBehaviour

Faz com que as entidades que o assumem, ao receber o comando *jump()*, pulem mais alto que o normal, permitindo que o jogador e outras entidades passem por baixo dela enquanto está no ar. Caso a entidade seja um Sapo e possua um alvo definido, o comportamento também é responsável pelo cálculo da trajetória do pulo do Sapo até seu alvo.

TurretBehaviour

Apenas pode ser assumido por entidades do tipo Laser, sendo responsável por ativar e desativar o raio emitido por elas através de atribuições booleanas. Ao ser atribuído a qualquer outro tipo de entidade, mantém o comportamento anterior da entidade e altera o material dela para Laser.

CampfireBehaviour

Apenas pode ser assumido por entidades do tipo Fogueira. Controla o estado atual das chamas da fogueira através de atribuições booleanas. Ao ser atribuído a qualquer outro tipo de entidade, mantém o comportamento anterior da entidade e simplesmente faz com que ela pegue fogo, caso seu material seja inflamável.

VirusBehaviour

Faz com que, caso o jogador toque a entidade, a fase atual seja reiniciada. Se esse comportamento for removido de um Vírus que se move, ele para o movimento imediatamente, e apenas o restaura caso o comportamento seja atribuído novamente.

FanBehaviour

É responsável por ligar e desligar o vento gerado por ventiladores através de atribuições booleanas. Caso seja atribuído a uma entidade que não seja do tipo Ventilador, cria uma

área de vento circular em volta da entidade, que também pode ser ligado e desligado. Ao perder esse comportamento, o vento emitido por uma entidade é desativado.

ExitBehaviour

Avança o jogo para a próxima fase caso qualquer entidade que assuma esse comportamento seja tocada pelo jogador.

4.4 *Level Design*

As fases do jogo foram elaboradas tendo em mente a ordem em que os conceitos de programação são introduzidos nas disciplinas iniciais do curso de computação. Inicialmente, as fases empregam apenas conceitos mais simples, como atribuições e variáveis. Então, à medida em que o jogador progride, as fases se tornam mais complexas, introduzindo novos obstáculos, como Vírus, e novos elementos de programação, como definições e chamadas de funções, estruturas condicionais e *loops* de repetição. Dessa forma, através da retórica procedural, o jogador aprende gradualmente sobre as mecânicas do jogo e, através do reconhecimento de padrões nos efeitos dos códigos construídos, se familiariza com o funcionamento de diferentes estruturas na programação.

O conteúdo a ser ensinado foi distribuído em duas partes, com o objetivo de que cada parte seja jogada pelos alunos pouco tempo após os conceitos aplicados em cada uma serem lecionados. A primeira parte abordou atribuições e variáveis. Na fase mostrada na Figura 4.7, por exemplo, é necessário atribuir o valor *true* ao ventilador (*"fan"*), para que ele empurre o bloco *"tree"* para o alcance do jogador. Então, é possível realizar a operação *"tree = false"*, que faz com que todas as árvores desapareçam e o caminho até a saída seja liberado.

Já na fase da Figura 4.8, é preciso alterar o valor da variável *"var1"* entre *true* e *false*, e atribuí-la à saída e ao laser de forma que a primeira seja aberta e o segundo desligado, tornando possível a conclusão da fase.

A segunda parte abordou, além do que foi tratado na primeira parte, condicionais, estruturas de repetição e funções. Na situação exibida na Figura 4.9, por exemplo, se o jogador apenas completar o código à esquerda com o bloco *"tree"*, as árvores irão pular

Figura 4.7: Terceira fase da primeira parte do jogo.

Figura 4.8: Quarta fase da primeira parte do jogo.

uma vez, pois a condição *"tree > 3"* foi atendida. No entanto, isso não contribuirá para a solução do problema, pois o jogador não consegue chegar à saída a tempo enquanto a árvore está no ar. Para que seja possível chegar até o final, é necessário trocar o bloco

"if" pelo *"while"*, fazendo com que as árvores pulem repetidamente. Isso torna possível que o jogador se aproxime da saída passando por baixo da árvore durante um de seus pulos.

Figura 4.9: Trecho da terceira fase da segunda parte do jogo.

Por fim, as funções são abordadas pela primeira vez na fase demonstrada na Figura 4.10. Nela, a função *"activate_frog_and_fan"*, definida no script superior, é chamada quando o script inferior é executado. Ela faz com que o sapo pule até seu alvo, mas seja empurrado de volta pelo ventilador que será ligado. Para manter o sapo em cima do botão, é preciso alterar seu material para tijolo, evitando que ele seja movido pelo vento e permitindo a abertura da porta através da atribuição *"exit = btn1"*.

Assim, a evolução da dificuldade dos níveis é feita através da construção de níveis gradativamente mais complexos, além da introdução de novos conceitos de programação na ordem em que são apresentados aos alunos na disciplina.

Figura 4.10: Quarta fase da segunda parte do jogo.

4.5 Considerações Finais

Este capítulo mostrou como o desenvolvimento do jogo *Snake Case* foi concebido para integrar uma narrativa envolvente, mecânicas de interação precisas e uma arquitetura de software cuidadosamente estruturada, de modo a proporcionar uma progressão gradual dos desafios que reflete os conceitos de programação ensinados em sala de aula. O capítulo a seguir detalha o processo de avaliação adotado para aferir a eficácia do jogo como ferramenta de apoio ao ensino de programação.

5 Avaliação

Nesta etapa do trabalho, foi realizada uma avaliação qualitativa com o objetivo de analisar a eficácia e a qualidade do jogo desenvolvido. Para tanto, o jogo foi disponibilizado a estudantes do primeiro período do curso de Ciência da Computação noturno da Universidade Federal de Juiz de Fora durante uma aula da disciplina Laboratório de Programação. Apesar de a aplicação ter sido parte da aula, os estudantes participaram de forma voluntária e as respostas foram coletadas de forma anônima, de acordo com o termo de consentimento apresentado no início do formulário da pesquisa.

Após a interação com o jogo, os participantes avaliaram sua experiência por meio do modelo MEEGA+, apresentado na Seção **??**. Esse modelo permitiu uma análise estruturada e detalhada das características do jogo, contribuindo para a compreensão de seu impacto e eficiência no contexto educacional.

5.1 Resultados

A apresentação do jogo aos alunos foi realizada em duas etapas, com o intuito de introduzir cada seção do jogo após os alunos terem sido expostos, em sala de aula, aos conceitos de programação necessários para compreender e interagir com cada parte do jogo. Essa abordagem buscou alinhar o conteúdo do jogo com o aprendizado prévio dos estudantes, facilitando a assimilação dos conceitos e a aplicação prática dos conhecimentos adquiridos.

Após a aplicação da primeira parte do jogo e a coleta das respostas dos alunos por meio do formulário de avaliação, os dados obtidos foram processados seguindo o modelo MEEGA+ *(Model for Evaluating Educational Games)*. O processo de cálculo da pontuação do jogo envolveu as seguintes etapas:

1. Os dados coletados para os itens específicos do formulário foram organizados em um arquivo auxiliar no formato CSV, contendo as respostas dos participantes. Esse arquivo serviu como entrada para um *script* em R, que aplica a Teoria de Resposta ao Item (TRI) para calcular as pontuações.

2. Utilizando o software *RStudio*, o *script* fornecido pelo MEEGA+ foi executado. Esse *script* calcula as pontuações dos participantes com base na TRI, gerando uma pontuação para cada indivíduo em uma escala base identificada pela notação $(0, 1)$ (PETRI; WANGENHEIM; BORGATTO, 2017). Além disso, o *script* produz um arquivo de saída com as pontuações calculadas e os erros padrão associados.

3. Para classificar o jogo como um todo, foi calculada a média das pontuações individuais (coluna `SCORE_TRI`) de todos os participantes. Seguindo o protocolo de Petri, Wangenheim e Borgatto (2017), essa média foi então transformada para a escala $(50, 15)$ por meio da fórmula:

$$\theta_{50,15} = 50 + 15 \times \theta_{0,1}$$

onde $\theta_{0,1}$ é a pontuação média na escala original.

Com base nas respostas coletadas na primeira parte do jogo, que contou com a participação de 19 alunos, o jogo obteve uma pontuação de 60,497, enquadrando-se na categoria de "boa qualidade", que corresponde ao intervalo $42,5 \leq \theta < 65$. Na segunda parte, que foi respondida por 10 alunos, a pontuação final foi de 58,218, mantendo-se na mesma categoria de qualidade. Segundo o MEEGA+, um jogo classificado como de "boa qualidade" frequentemente apresenta atividades desafiadoras, oferecendo novos desafios aos alunos. Ele proporciona uma atenção moderadamente focada dos jogadores, embora os estudantes não se esqueçam completamente do ambiente ao seu redor. O jogo também costuma gerar sentimentos de confiança e satisfação nos jogadores, além de promover momentos de interação social e diversão. Frequentemente, o jogo é considerado relevante para os interesses dos alunos, e eles reconhecem que o conteúdo do jogo está relacionado ao curso. Em termos de usabilidade, o jogo geralmente possui regras claras e é fácil de jogar, embora nem sempre apresente um design totalmente atraente.

5.1.1 Discussão dos Resultados

Analisando as respostas obtidas na primeira parte da pesquisa, representadas nas Figuras 5.1 e 5.2, observa-se que 84,2% dos respondentes da Parte 1 indicaram facilidade para

aprender a jogar. O mesmo percentual afirmou que as regras eram claras e compreensíveis. No quesito design, a atratividade e organização visual receberam avaliações positivas de 94,7% dos participantes. Entretanto, 52,6% apontaram dificuldades relacionadas à legibilidade das fontes e à consistência dos textos, aspecto reforçado pelos comentários abertos. Quanto à progressão dos desafios, a totalidade dos participantes concordou que o jogo apresentou novos obstáculos em um ritmo adequado, evidenciando uma boa evolução na dificuldade.

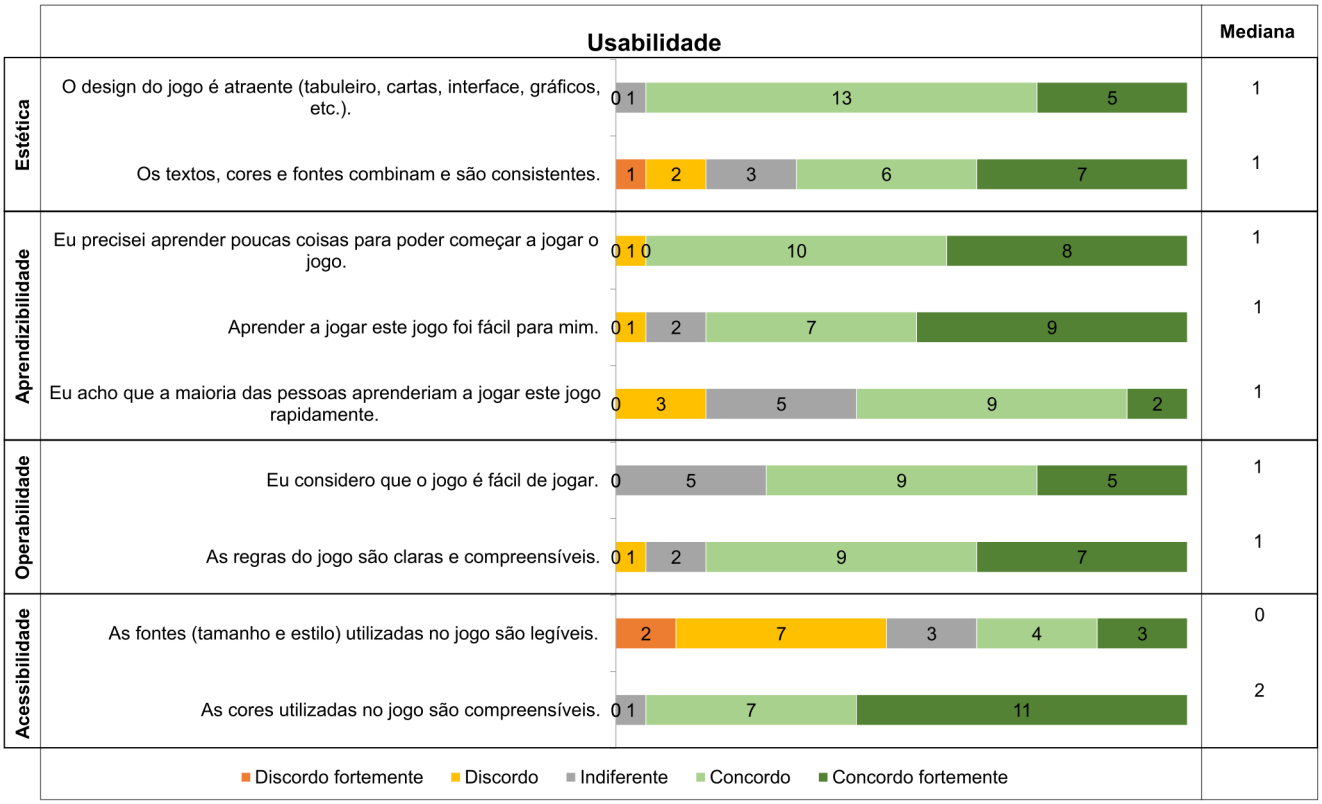

Figura 5.1: Respostas da parte 1 sobre usabilidade.

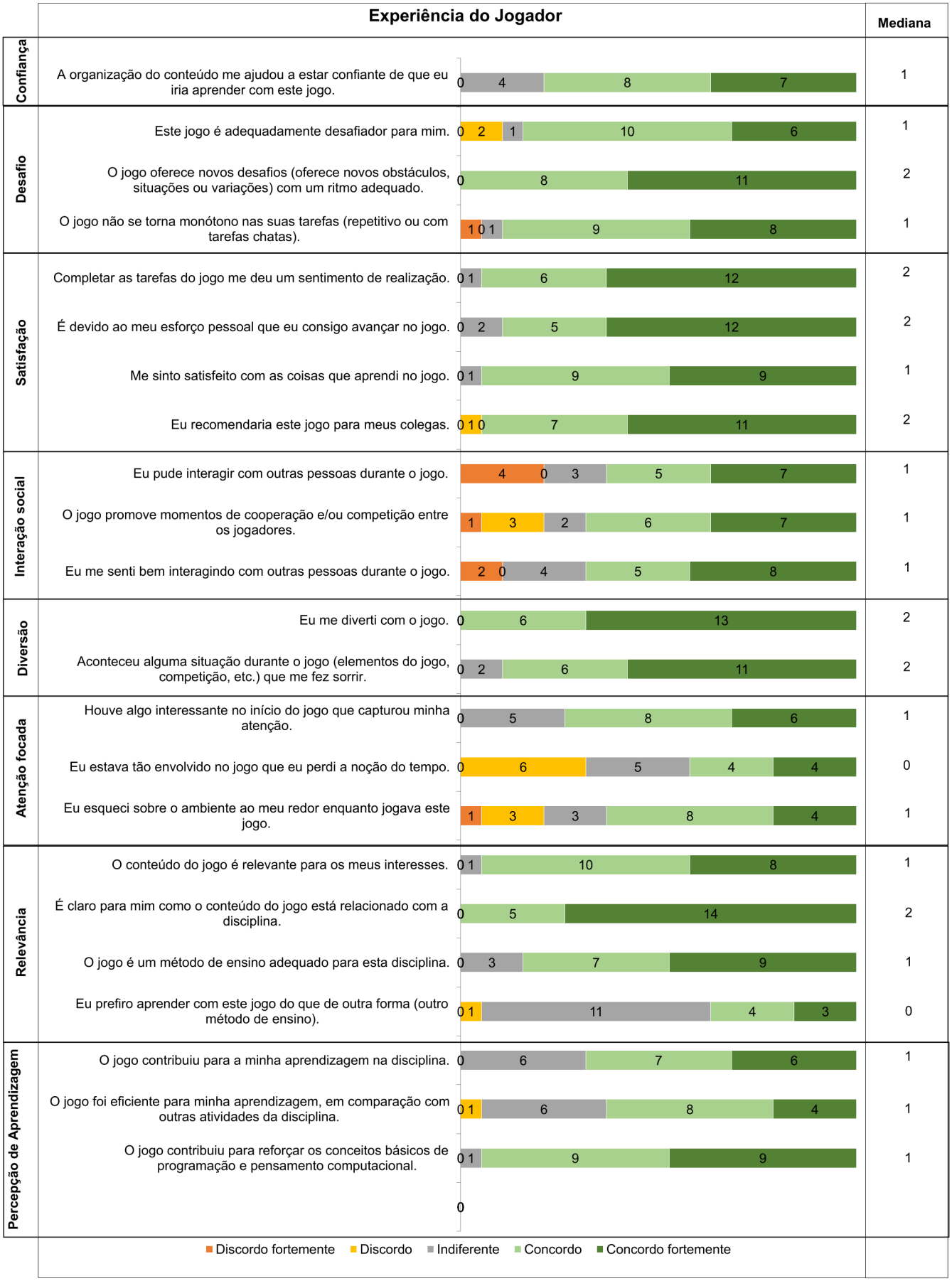

Figura 5.2: Respostas da parte 1 sobre experiência do jogador.

Na Parte 2, de acordo com as respostas obtidas, representadas nas Figuras 5.3 e 5.4, 90% dos respondentes continuaram a considerar o jogo fácil de aprender, mantendo a clareza das mecânicas mesmo com o aumento da complexidade. A adequação dos desafios foi reconhecida por 70% dos participantes, embora algumas sugestões tenham apontado a necessidade de mecanismos de suporte para auxiliar nas fases mais difíceis. A percepção sobre o design visual permaneceu consistente, com 90% dos jogadores destacando a atratividade dos gráficos. Além disso, após as avaliações da Parte 1, uma nova fonte foi aplicada aos textos, aprovação registrada por 70% dos alunos, enquanto 10% se mostraram indiferentes, evidenciando uma melhora significativa na legibilidade. No que se refere à interação social, houve maior variação nas respostas, com metade dos participantes satisfeitos com os momentos de cooperação e competição, enquanto os demais adotaram uma posição neutra, apontando um possível aprimoramento nessa área.

Em resumo, os dados quantitativos e qualitativos das duas etapas corroboram que o *Snake Case* parece ser eficaz em engajar os alunos, o que pode contribuir para reforçar conceitos fundamentais de programação, ao mesmo tempo em que apontam áreas específicas – como a padronização dos elementos visuais e o suporte em fases de maior dificuldade – para futuras melhorias. Esses *insights* são fundamentais para orientar refinamentos que possam potencializar o impacto educacional do jogo.

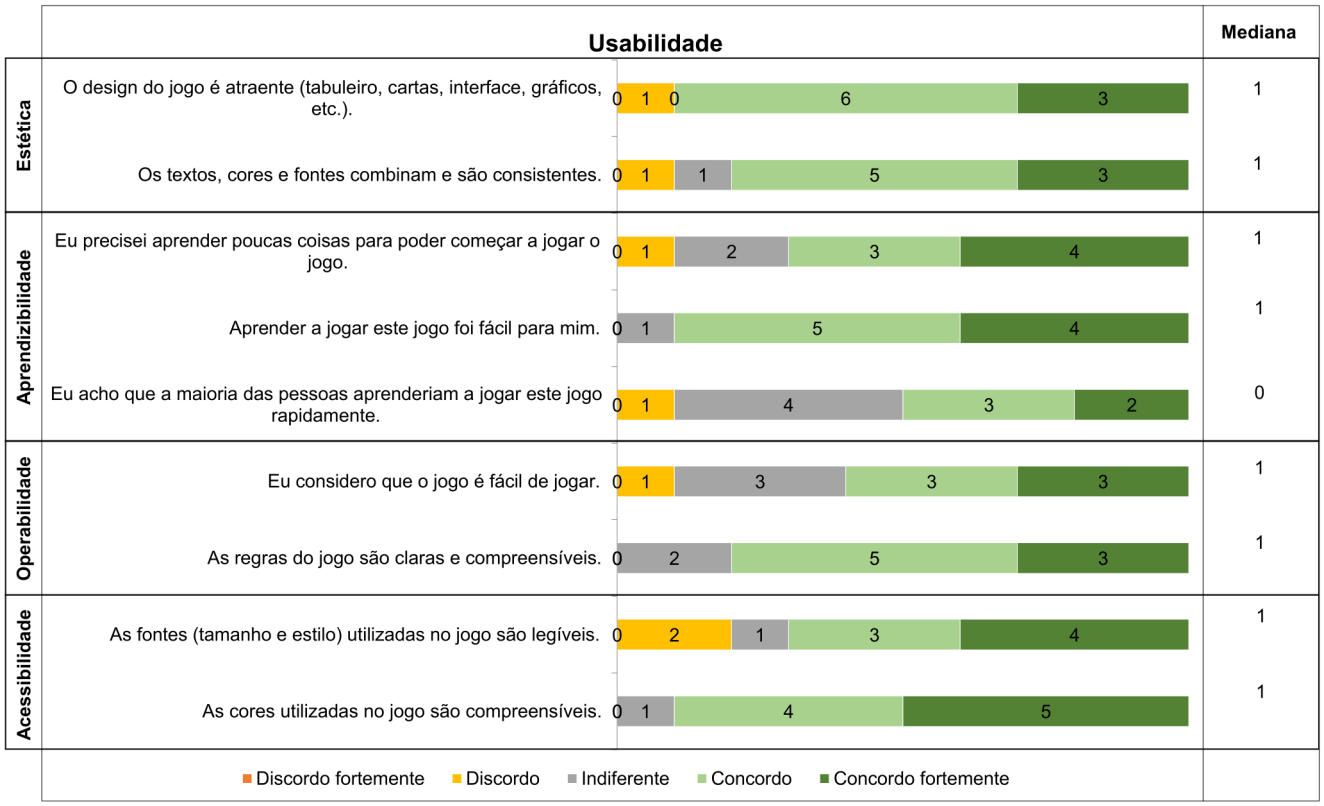

Figura 5.3: Respostas da parte 2 sobre usabilidade.

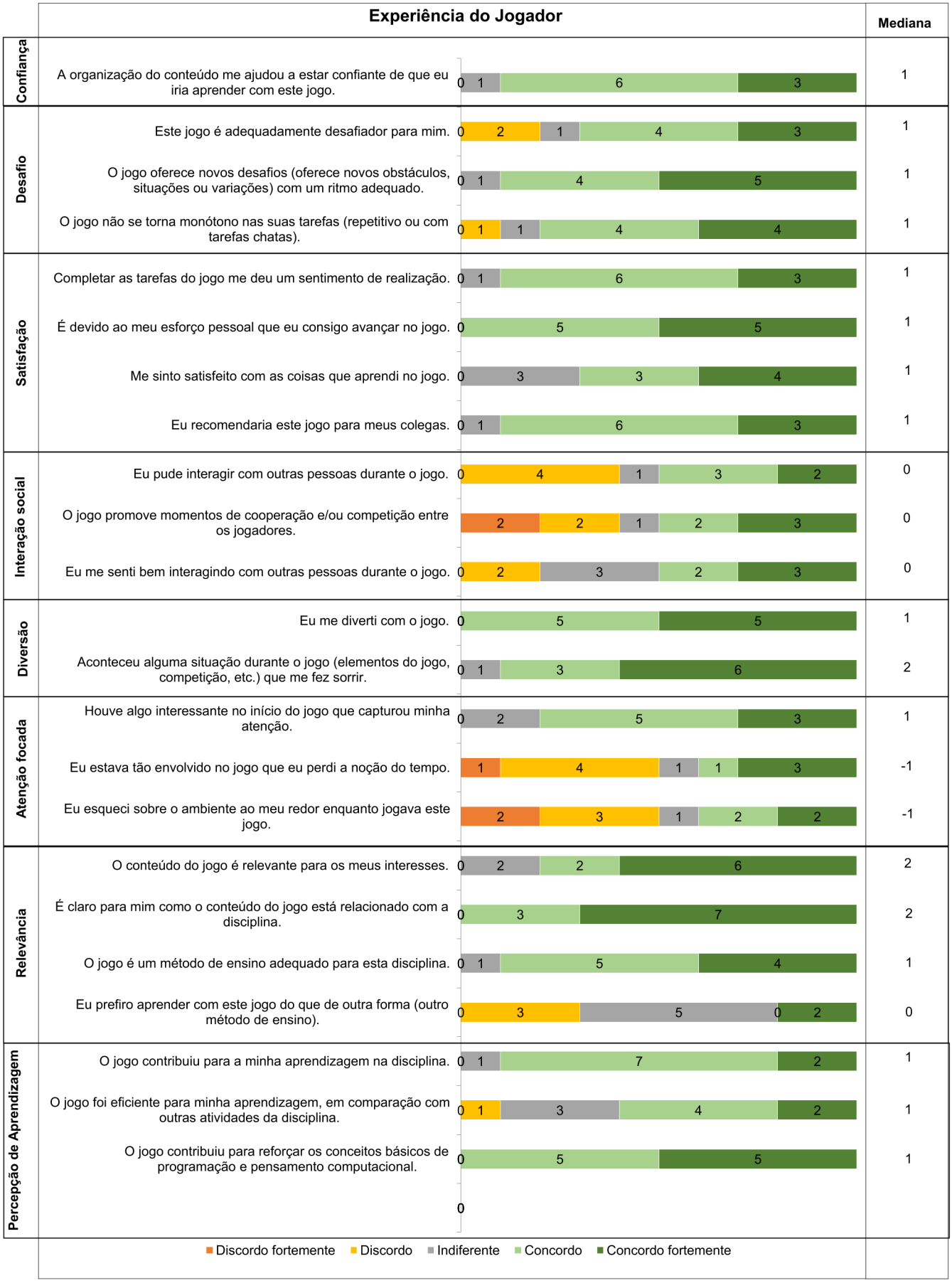

Figura 5.4: Respostas da parte 2 sobre experiência do jogador.

5.1.2 Ameaças à Validade

A interpretação dos resultados obtidos na avaliação deve ser realizada considerando diversas ameaças à validade, que podem impactar a generalização e a robustez das conclusões. Nesta seção, discutem-se as principais ameaças, organizadas em três categorias: validade interna, validade externa e validade de construto.

Validade Interna

A validade interna refere-se à extensão em que as mudanças observadas na variável dependente (desempenho e aceitação do jogo) podem ser atribuídas à intervenção (uso do jogo sério) e não a fatores externos. Entre as ameaças internas identificadas, destacam-se:

- **Amostra Limitada:** É possível que o fato de que muitos dos alunos da turma não terem participado das avaliações tenha impactado negativamente as perguntas sobre interação social;
- **Viés de Seleção:** A participação foi voluntária, o que pode ter levado a uma amostra não representativa da população de estudantes (alunos mais motivados ou com maior interesse em jogos e tecnologia podem ter se auto-selecionado para participar, influenciando positivamente os resultados);
- **Fatores Contextuais:** A avaliação foi realizada durante aulas de laboratório, onde os alunos estavam simultaneamente envolvidos em outras atividades, o que pode ter comprometido a imersão, a dedicação exclusiva à experiência do jogo e a possibilidade de interação entre os estudantes;
- **Efeito Novidade:** A novidade de utilizar um jogo como ferramenta de ensino pode ter aumentado temporariamente o engajamento dos alunos, sem necessariamente refletir uma melhoria duradoura na aprendizagem dos conceitos de programação.

Validade Externa

A validade externa diz respeito à capacidade de generalizar os resultados obtidos para outras populações, contextos ou momentos. Neste estudo, as principais ameaças externas são:

- **Amostra Limitada:** O número de participantes (19 na Parte 1 e 10 na Parte 2) pode não ser suficiente para extrapolar os resultados para toda a população de estudantes de cursos de tecnologia;

- **Contexto Específico:** O estudo foi realizado em uma única instituição de ensino (UFJF) e durante períodos específicos de aula. Dessa forma, os resultados podem não ser generalizáveis para outros contextos educacionais ou para alunos de diferentes níveis de formação.

Validade de Construto

A validade de construto avalia se os instrumentos de medida utilizados (como os formulários baseados no modelo MEEGA+) realmente capturam os aspectos que se pretende avaliar. As ameaças de validade de construto incluem:

- **Instrumentos de Medição:** A utilização de questionários pode estar sujeita a interpretações subjetivas dos participantes. A clareza dos itens e a escala de resposta adotada podem não refletir com precisão todas as dimensões de engajamento, usabilidade e imersão.

- **Fatores de Confusão:** A influência de variáveis não controladas, como o estado emocional dos alunos, suas experiências prévias com jogos ou a interação entre eles durante a avaliação, pode interferir na precisão dos dados coletados.

Considerações Finais

Para mitigar essas ameaças, recomenda-se a adoção de estratégias complementares, como a realização de estudos longitudinais, a aplicação de instrumentos de avaliação adicionais (por exemplo, testes de desempenho acadêmico) e a expansão da amostra para incluir diferentes instituições e contextos. Dessa forma, futuras pesquisas poderão validar de forma mais robusta os efeitos do jogo *Snake Case* no ensino de programação.

6 Conclusão

Ao longo deste trabalho, apresentou-se o desenvolvimento do jogo sério *Snake Case*, cujo objetivo é introduzir e reforçar conceitos de programação de forma lúdica. Iniciou-se com uma revisão sobre o ensino de programação, destacando as dificuldades e a alta evasão em disciplinas introdutórias, seguida do embasamento teórico acerca de jogos sérios e seu potencial na educação. Em seguida, discutiu-se a proposta do *Snake Case*, detalhando design, mecânicas, níveis e implementação. Por fim, foi conduzida uma avaliação qualitativa baseada no modelo MEEGA+.

A aplicação do jogo em duas partes, correspondentes a conteúdos de programação progressivamente mais complexos, obteve classificações de "boa qualidade" em ambas, conforme o MEEGA+. Os participantes apontaram facilidade de aprendizado das regras, clareza de objetivos e desafios crescentes. Esses resultados reforçam a aderência do jogo à sua proposta de servir como ferramenta de apoio inicial, com bom nível de motivação e relevância.

Dentre as limitações, destaca-se o contexto de aplicação do jogo, que foi jogado em curtos intervalos de aula, muitas vezes competindo com outras tarefas acadêmicas. Isso provavelmente reduziu a imersão dos estudantes, o que pode ter impactado negativamente as notas nesse quesito. Além disso, o jogo, em si, não avalia formalmente a retenção ou a compreensão profunda dos conceitos, sendo necessário que docentes ou outros instrumentos verifiquem tais aspectos. Também foram identificados pontos de melhoria na padronização visual, legibilidade e implementação de suporte adicional em fases complexas.

Como desdobramento, propõe-se o desenvolvimento de mais níveis, visto que foi necessário manter o jogo curto para que houvesse tempo de ser jogado durante a aula. Outra possibilidade é integrar métricas internas de desempenho (por exemplo, tempo gasto em cada fase, número de tentativas até a solução) que auxiliem professores a identificar dificuldades específicas dos alunos. Finalmente, novas avaliações, com maior amostragem e períodos de teste mais longos, podem ser conduzidas para investigar o

impacto do jogo na retenção de conceitos e desempenho acadêmico dos participantes.

Bibliografia

ABDELLATIF, A. J.; MCCOLLUM, B.; MCMULLAN, P. Serious games quality characteristics evaluation: The case study of optimizing robocode. In: *2018 International Symposium on Computers in Education (SIIE)*. [S.l.: s.n.], 2018. p. 1–4.

BECKER, K. Learning by doing, a comprehensive guide to simulations, computer games, and pedagogy in e-learning and other educational experiences, 2005. by clark aldrich. *The Canadian Journal of Learning and Technology*, v. 31, p. 105–108, 01 2005.

BEZERRA, A. et al. Bora jogar: desenvolvimento de jogos para auxiliar na aprendizagem de programação. *Research, Society and Development*, v. 11, p. e15511527668, 04 2022.

COSTA, L. D. *O que os jogos de entretenimento têm que os jogos educativos não têm*. [S.l.: s.n.], 2009.

DANTAS, G. A. F. et al. Aprenda a identificar e diferenciar gamificaÇÃo e jogo sÉrio. *Revista Conexão na Amazônia*, v. 3, n. Edição especial, p. 50–66, nov. 2022. Disponível em: ⟨https://periodicos.ifac.edu.br/index.php/revistarca/article/view/133⟩.

FILHO, S.; JUCá, P. Uso de jogos sérios para auxiliar na reabilitação motora de pacientes com espondilite anquilosante. In: . [S.l.: s.n.], 2015.

FROSI, F. O.; SILVA, I. C. S. da. Codebots: Ensino lúdico de conceitos introdutórios de programação para estudantes da educação básica. In: *Anais Estendidos do XVIII Simpósio Brasileiro de Jogos e Entretenimento Digital*. Porto Alegre, RS, Brasil: SBC, 2019. ISSN 2179-2259. Disponível em: ⟨https://www.sbgames.org/sbgames2019/files/papers/EducacaoFull/197906.pdf⟩.

GOMES, A. de J. Dificuldades de aprendizagem de programação de computadores: contributos para a sua compreensão e resolução. In: . [S.l.: s.n.], 2010.

GUIMARãES, A. B. C. et al. Jogos sÉrios como ferramentas para desenvolver soft skills, com base em revisÃo sistemÁtica da literatura. In: *XII Singep e 12a CIK - Proceedings*. [s.n.], 2024. ISSN 2317-8302. Disponível em: ⟨https://submissao.singep.org.br/12singep/proceedings/resumo?cod_trabalho=188⟩.

KAZIMOGLU, C. Enhancing confidence in using computational thinking skills via playing a serious game: A case study to increase motivation in learning computer programming. *IEEE Access*, v. 8, p. 221831–221851, 2020.

LARA, D. F. et al. A produção acadêmica sobre o uso de jogos sérios na educação: Avanços alcançados. *Temática*, v. 19, p. 206, 01 2023.

MACENA, J. et al. Hello food: um jogo para praticar conceitos de algoritmos para iniciantes na computação. In: *Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital*. Porto Alegre, RS, Brasil: SBC, 2022. p. 1066–1075. ISSN 0000-0000. Disponível em: ⟨https://sol.sbc.org.br/index.php/sbgames_estendido/article/view/23744⟩.

MCFERRAN, D. *The Making Of Mario Is Missing, The Plumber's Oddest Adventure*. [s.n.], 2023. Disponível em: ⟨https://www.timeextension.com/features/the-making-of-mario-is-missing-the-plumbers-oddest-adventure⟩. Acesso em: 19 fev. 2025.

MEROTO, M. B. d. N. et al. Jogando para aprender: Como a gamificaÇÃo estÁ mudando a educaÇÃo. *REVISTA FOCO*, v. 17, n. 1, p. e4122, jan. 2024. Disponível em: ⟨https://ojs.focopublicacoes.com.br/foco/article/view/4122⟩.

NETO, J. G. et al. Maze code: Retórica procedural aplicada ao ensino de lógica de programação. In: *Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital*. Porto Alegre, RS, Brasil: SBC, 2021. p. 519–528. ISSN 0000-0000. Disponível em: ⟨https://sol.sbc.org.br/index.php/sbgames_estendido/article/view/19685⟩.

OLIVEIRA, F. N. de. *Artigo: Por que jogos educativos são chatos?* [s.n.], 2016. Disponível em: ⟨https://www.fabricadejogos.net/posts/artigo-por-que-jogos-educativos-sao-chatos/⟩. Acesso em: 18 out. 2023.

OLIVEIRA, G. Carniel de; BOFF, E. Aprendizagem pelos jogos: uma abordagem para o ensino de programação. *Scientia cum Industria*, v. 13, n. 2, p. e241318, Dec. 2024. Disponível em: ⟨https://sou.ucs.br/etc/revistas/index.php/scientiacumindustria/article/view/13355⟩.

PACHECO, A. *Serious Game for Physics as an Active and Gamified Learning Strategy*. 2024. Disponível em: ⟨https://arxiv.org/abs/2407.10057⟩.

PETRI, G.; WANGENHEIM, C. Gresse von; BORGATTO, A. F. Meega+, systematic model to evaluate educational games. In: _____. *Encyclopedia of Computer Graphics and Games*. Cham: Springer International Publishing, 2017. p. 1–7. ISBN 978-3-319-08234-9. Disponível em: ⟨https://doi.org/10.1007/978-3-319-08234-9_214-1⟩.

SANTOS, C. S. et al. *Aprendendo Programacao Orientada a Objetos com uma Abordagem Ludica Baseada em Greenfoot e Robocode*. 2017. Disponível em: ⟨https://arxiv.org/abs/1710.04132⟩.

TACOURI, H.; NAGOWAH, L. Code saga – a mobile serious game for learning programming. In: *2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS)*. [S.l.: s.n.], 2021. p. 190–195.