RFPG - Game Design e Avaliação Preliminar de um Jogo para Apoiar o Ensino de Algoritmos e Programação

Eduardo Leal de Carvalho, Jean Felipe Patikowski Cheiran

[1] Universidade Federal do Pampa (Unipampa) – Campus Alegrete
Av. Tiarajú, 810, Ibirapuitã — Alegrete, RS – Brasil

{eduardocarvalho.aluno, jeancheiran}@unipampa.edu.br

Abstract. *The present work describes the game design and evaluation process of the Red Fox Programming Game (RFPG), a serious game to support programming teaching. RFPG's gameplay is based on the souls-like genre and turn-based action. The player controls a warrior fox who must defeat challenging enemies that attack using patterns. The player must identify these patterns and code in a command prompt the actions to counterattack. The game design was represented by paper prototyping and three professors of Algorithms and Programming courses participated in usability tests with the prototype. The technique made it possible to detect improvements that could be implemented in the final version of the software.*

Resumo. *O presente trabalho descreve o processo de game design e avaliação do jogo Red Fox Programming Game (RFPG), um jogo sério para apoiar o ensino de programação. A jogabilidade do RFPG tem como base o gênero souls-like e a ação baseada em turnos. O jogador controla uma raposa guerreira que deve derrotar inimigos desafiadores com padrões de ataques definidos. O jogador deve identificar esses padrões e codificar em um prompt de comando ações para contra-atacar. O game design do jogo foi representado através da prototipação em papel e três professores de Algoritmos e Programação participaram de testes de usabilidade com o protótipo. A técnica permitiu detectar melhorias a serem implementadas na versão final do software.*

1. Introdução

O mercado de programação é um dos setores com maior crescimento na atualidade e traz consigo a demanda por novos profissionais [Screpanti 2024]. Entretanto, muitos estudantes desistem dessa área devido a dificuldades em conteúdos fundamentais como, por exemplo, resolução de problemas e matemática [Jenkins 2002]. Qian e Lehman também destacam a enorme dificuldade enfrentada por estudantes nos tópicos iniciais, como pensamento computacional, variáveis e estruturas condicionais [Qian and Lehman 2017].

Existem diversas práticas capazes de aumentar a eficiência do ensino de programação. Entre elas, o aprendizado baseado em jogos destaca-se por integrar teoria e prática de forma divertida e motivadora [Mayer 2019]. Essa metodologia utiliza jogos nos quais o conteúdo educacional está camuflado de forma lúdica nas características do jogo [Pivec et al. 2003]. Essas aplicações podem ser caracterizadas como 'jogos sérios', uma vez que são utilizadas como propósito de ensino-aprendizagem e não apenas entretenimento [Rocha et al. 2015]. Ainda, para garantir a efetividade dessa abordagem, é

crucial que esses jogos sérios sejam motivadores e mantenham a atenção de estudantes [Pivec et al. 2003].

Malliarakis e colegas mencionam diversos jogos criados para reforçar tópicos específicos da programação como, por exemplo, Catacombs, Saving Princess Sera e LightBot [Malliarakis et al. 2014]. Embora cada um desses jogos possua mecânicas e conteúdos educacionais específicos, o objetivo de todos é o mesmo: estimular e apoiar o aprendizado. Particularmente, o jogo Lightbot foi utilizado para avaliar o impacto do aprendizado baseado em jogos em grupos com e sem conhecimentos prévios de programação, relatando *feedbacks* positivos e aumento da confiança para programar em alguns casos [Mathrani et al. 2016].

O objetivo desse trabalho é desenvolver um jogo sério baseado em comandos para o apoio no aprendizado de lógica de programação.

Esse artigo apresenta os resultados de *game design* e avaliações preliminares do jogo Red Fox Programming Game (RFPG). Nesse jogo, o jogador escreve comandos para o personagem principal de forma similar a uma linguagem de programação. Influenciado pelos gêneros *soulslike* e estratégia por turnos, o objetivo do jogador é identificar os padrões de ação dos inimigos e codificar comandos para derrotá-los em diversas sequências de turnos alternados (jogador-inimigo), incentivando assim o pensamento computacional do estudante.

Atualmente, o jogo possui mecânicas bem definidas [Schell 2014] e protótipos em papel [Snyder 2003] usados em estudos de usabilidade com professores de Algoritmos e Programação para coletar *feedback* sobre o jogo e seus aspectos pedagógicos.

2. Trabalhos Relacionados

Pesquisas foram realizadas em plataformas comerciais de jogos para computadores e celulares (Steam, Epic Games, Google Play, etc.), escolhendo a categoria "programação" para encontrar jogos que estimulem o aprendizado de programação e pensamento computacional. Paralelamente, foram realizadas buscas na biblioteca digital da Sociedade Brasileira de Computação[1] por artigos que abordem o desenvolvimento de jogos para apoio no ensino de programação criados no Brasil.

Code Monkey[2] é um jogo sério para crianças focado em resolução de quebra-cabeças através da codificação, trazendo como principal característica o ensino de linguagens de programação. Para solucionar os desafios propostos a cada nível, o jogador deve escrever códigos adequados com base em seus conhecimentos desenvolvidos nos cursos previamente definidos pela plataforma. O Code Monkey pode ser usado por professores que desejam incorporar o jogo em suas aulas ou diretamente pelo público infantil deseja aprender a programar.

Code Combat[3] é um jogo sério focado na solução de quebra-cabeças, similar ao Code Monkey, utilizando também a programação baseada em textos. Seu diferencial é disponibilizar fases de combate, gerando uma nova camada de ação e estratégia ao jogo. Além disso, o jogo torna-se mais atrativo para adolescentes e jovens adultos.

[1]Disponível em: `https://sol.sbc.org.br`.
[2]Disponível em: `https://www.codemonkey.com`.
[3]Disponível em: `https://codecombat.com`.

LightBot[4] é um jogo sério que promove o ensino de conceitos básicos de programação por meio da metáfora de programação em blocos. Novamente, observamos um jogo do gênero quebra-cabeças que utiliza desafios lógicos para ensinar conteúdos complexos de forma simplificada.

O jogo **7 Billion Humans**[5] destaca-se pelo equilíbrio entre ensino de programação e entretenimento, aliando a programação baseada em blocos e uma história cativante que se desenrola ao longo de fases com quebra-cabeças desafiadores. Nesse jogo, o jogador deve automatizar tarefas de funcionários em uma megacorporação através da codificação de suas ações por meio de blocos de quebra-cabeça, promovendo o aprendizado de lógica de programação. Os mesmos desenvolvedores desse jogo já haviam produzido outro jogo com as mesmas mecânicas anteriormente: **Human Resource Machine**[6].

No jogo **The Farmer Was Replaced**[7] o jogador desenvolve, através de programação em texto, a automação de ações de um drone que cuida de uma pequena plantação. O jogo propõe objetivos simples (e.g., ganhar moedas coletando recursos e expandir sua fazenda) para incentivar o jogador a aplicar conceitos de programação em um cenário contínuo e crescente, sem fases ou níveis definidos. Essa abordagem torna o jogo muito mais fluido em relação aos jogos anteriormente apresentados. Por outro lado, os tutoriais possuem muito texto para explicar comandos simples, tornando potencialmente o aprendizado cansativo e massante.

while True: learn()[8] é um jogo focado no ensino de Inteligência Artificial e *big data* através da solução de quebra-cabeças que envolvem a conexão e a calibragem de nodos de uma rede neural. Diferente de jogos como Code Combat e Code Monkey, while True: learn() foca-se no público jovem e adulto e não se propõe como prática de programação, limitando-se a estimular o pensamento computacional e à aprendizagem de conceitos de redes neurais e aprendizagem de máquina.

Nipo e colegas desenvolveram um jogo em Realidade Virtual que promove o pensamento computacional através da solução de quebra-cabeças utilizando blocos de comandos que podem ser encontrados ao longo do cenário [Nipo et al. 2023]. Ao explorar o local, o jogador se depara com robôs que solicitam ajuda e que podem ser controlados através de blocos de comando. Assim, o objetivo do jogo é organizar os comandos dos robôs de forma lógica em um painel para então executar esses comandos.

Marques et al. desenvolveram um jogo para ensinar o pensamento computacional concorrente [Marques et al. 2021]. A proposta é que o jogador decida, utilizando blocos de comando, as ações realizadas simultaneamente por dois piratas em um mapa repleto de obstáculos. O objetivo final é chegar no quadrado com símbolo de "X" marcado e cavar o tesouro enterrado. O terreno acidentado e os objetos interativos no cenário impõem

[4]Disponível em: `https://lightbot.com`.

[5]Disponível em: `https://store.steampowered.com/app/792100/7_Billion_Humans`.

[6]Disponível em: `https://store.steampowered.com/app/375820/Human_Resource_Machine`.

[7]Disponível em: `https://store.steampowered.com/app/2060160/The_Farmer_Was_Replaced`.

[8]Disponível em: `https://store.steampowered.com/app/619150/while_True_learn`.

desafios ao jogador que precisa desenvolver o melhor algoritmo para avançar nas fases.

A literatura sobre jogos que promovem o desenvolvimento de habilidades de programação e pensamento computacional é bastante extensa. Entretanto, jogos que abordam o ensino da programação baseada em textos são escassos, especialmente ao observar artigos desenvolvidos por pesquisadores brasileiros. Essa programação baseada na escrita de comandos (e não apenas na escolha e na combinação de blocos) aproxima o estudante da representação mais comum de algoritmos, sendo uma alternativa de apoio para o ensino de programação encontrada nessa pesquisa apenas em jogos comerciais (por exemplo, **The Farmer Was Replaced**) e sem uma avaliação científica formal.

Ainda, a proposta do jogo descrito nesse artigo é apoiar a aprendizagem de programação e pensamento computacional por meio de um gênero diferente de jogo que combina dinâmicas *soulslike* e ação baseada em turnos. A maior parte dos trabalhos relacionados descritos nessa pesquisa estão categorizados como gênero quebra-cabeças (*puzzle*). Enquanto o foco de jogos de quebra-cabeças é montar uma solução para um cenário-problema, o foco do jogo desenvolvido nesse trabalho envolve somente cenários de combate difíceis nos quais é necessário reconhecer o padrão de ações do inimigo e implementar uma sequência de comandos que permitam derrotá-lo em um esquema de turnos alternados entre o jogador e o inimigo. Essa mudança de paradigma propicia aos estudantes uma nova forma de treinar a programação, incentivando o pensamento computacional e reconhecimento de padrões.

3. Game Design do Red Fox Programming Game

O jogador controla uma raposa guerreira que deve derrotar o inimigo no cenário. A Figura 1 apresenta o protótipo em papel do *gameplay* com a raposa e um inimigo no cenário. As ações da raposa são definidas pela escrita de comandos em um *prompt* à direita da tela e executadas após acionamento do botão “play”.

O jogo ocorre em turnos, sem tempo limite para finalizar, possibilitando que o jogador analise a melhor estratégia antes de agir. Depois de codificar as ações do personagem, o botão “play” deve ser acionado pelo jogador para ter início a animação com ações resultantes da execução do código na área de batalha no centro da tela.

Após o turno do jogador, o inimigo realizará suas ações obedecendo a um padrão previamente definido. É parte do desafio do jogador identificar o padrão de ações do inimigo e gerar o código adequado para o próximo turno do personagem. Os turnos do personagem e do inimigo ficam se alternando até que a vida de algum deles chegue a zero. Se a vida do inimigo chegar a zero, o inimigo é derrotado e o jogador avança para a próxima fase. Se a vida do protagonista for reduzida a zero, o personagem do jogador é derrotado e é necessário repetir o combate contra aquele inimigo a partir do início.

A quantidade de comandos que o jogador pode escrever no *prompt* é limitada, obrigando-o a desenvolver estratégias com estruturas de repetição para maximizar suas ações. Apesar de limitada, a quantidade de comandos pode ser aumentada em um menu de melhorias. Melhorias podem ser adquiridas ao coletar itens especiais obtidos ao derrotar inimigos ou encontrados a partir de objetos quebráveis no cenário (e.g., vasos e garrafas).

O jogo também conta com sistema de itens consumíveis (e.g., poções para recuperar vida) que podem ser encontrados em objetos quebráveis. Os itens coletados são

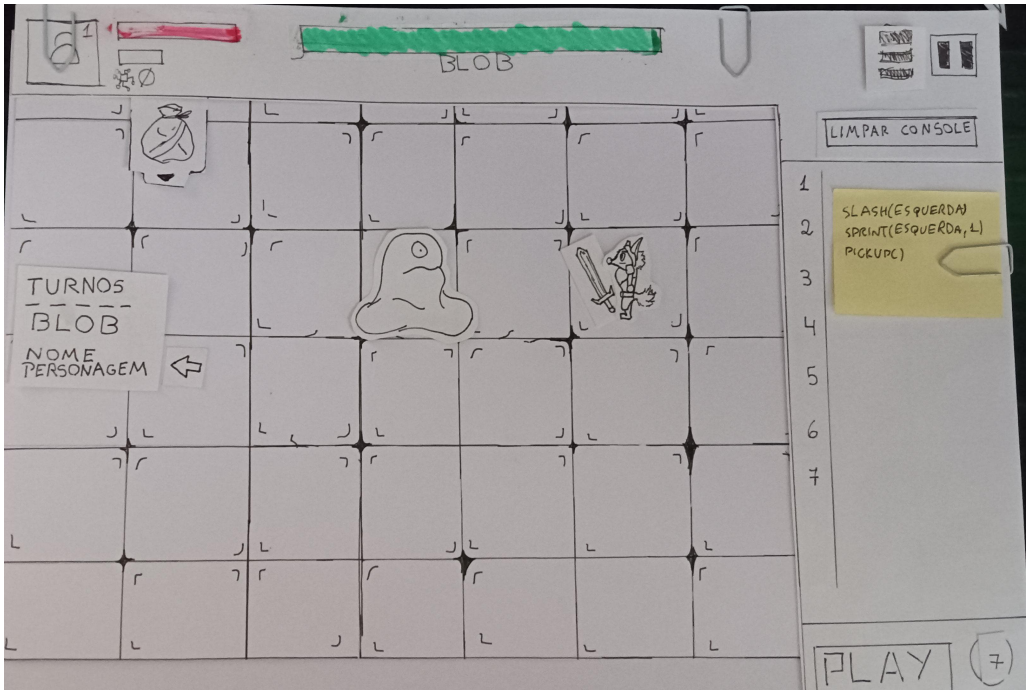
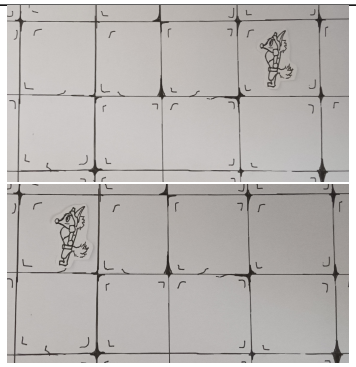
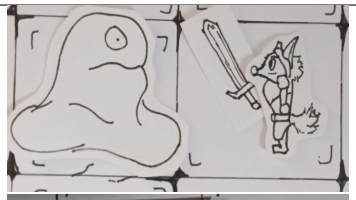
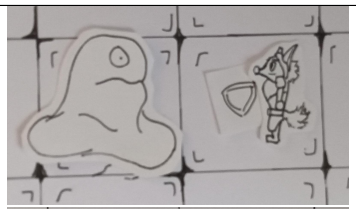
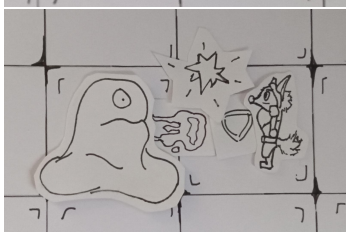


Figura 1. Protótipo em papel do *design* do jogo.

armazenados na mochila do jogador, que possui a estrutura de um vetor, e podem ser utilizados através do comando **inventory**(posição do item na mochila).

Os comandos básicos do *prompt* podem ser encontrados na Tabela 1.

Comando	**Ação realizada**	**Descrição**
sprint(direção, quantidade)		Movimenta o personagem uma determinada quantidade de blocos na direção selecionada (de 1 a 4 blocos).

slash(direção)	 	Ataca na direção selecionada atingindo três blocos paralelos ao personagem conforme area em vermelho. Caso atinja o inimigo o mesmo perde 1 ponto de vida.
block(direção)	 	Bloqueia ataques inimigos vindos direção selecionada, evitando dano ao personagem. O bloqueio começa apenas no início da rodada inimiga e termina no final da rodada inimiga.
pickUp()	 	Guarda na mochila do personagem o item que estiver no mesmo bloco que ele. Os itens coletados são adicionados à menor posição disponível do vetor inventário.

inventory(posição)		Consome o item do inventário que está na posição informada no comando.

Tabela 1. Comandos básicos da protagonista

4. Avaliação preliminar

A prototipação em papel foi aplicada para reduzir tempo e custo de desenvolvimento ao promover *feedbacks* de usuários no início da concepção de projetos [Snyder 2003]. A técnica também incentiva a criatividade dos desenvolvedores e garante que novos artefatos gráficos sejam criados sob demanda durante os testes e avaliados imediatamente pelos usuários em sessões experimentais. O método é dividido em duas partes [Snyder 2003]: (1) prototipação das telas e (2) testes de usabilidade do protótipo.

Na primeira etapa, toda a interface é concebida por meio de desenhos de lápis e caneta, colagens e recortes em papel, procurando comunicar da melhor maneira possível os elementos interativos do software. Posteriormente, testes de usabilidade [Rubin and Chisnell 2008] são conduzidos com usuários representativos da aplicação que utilizam a metodologia *thing aloud* [Ericsson and Simon 1993] para comunicar durante os testes suas impressões e dificuldades. Três atores estão presentes durante essa fase: jogador, facilitador e computador. O **jogador** simboliza um usuário final representativo da aplicação. O **facilitador**, por sua vez, indica as tarefas que o jogador deve realizar, observa as interações com o protótipo, faz perguntas sobre decisões do jogador e realiza anotações sobre problemas e possíveis soluções. Por fim, o **computador** representa um integrante do time de desenvolvimento que simula as respostas do jogo às ações do jogador (e.g., trocas de tela e mudanças de *sprites*) no protótipo em papel.

Três professores de Algoritmos e Programação da UNIPAMPA, campus Alegrete, foram convidados a testar o protótipo em papel do jogo e realizar sugestões (duas mulheres e um homem). Cada professor testou o protótipo separadamente em sessões experimentais de aproximadamente 30 minutos. A tarefa realizada foi a mesma para todos os participantes: completar o tutorial do jogo. O docente foi incentivado a expressar em voz alta seus pensamentos (dificuldades, críticas, preferências etc.) à medida que interagia com o protótipo, configurando o uso do protocolo *think aloud*.

Um vídeo[9] simulando uma sessão experimental com usuários no tutorial está disponível junto desse artigo. Os resultados e as melhorias são discutidos a seguir.

[9]Disponível em: `https://doi.org/10.6084/m9.figshare.27249165`

5. Resultados preliminares

Os três professores se mostraram favoráveis à proposta do jogo e à viabilidade do uso do software como apoio para estudantes de Algoritmos e Programação. Entretanto, problemas de interação ocorreram e melhorias referentes ao *game design* foram apontadas pelos participantes.

A **professora A** (pouco familiarizada com jogos digitais) citou a necessidade de um acesso rápido à lista de comandos do jogo para relembrar o jogador das possibilidades de ações disponíveis em uma determinada batalha. Além disso, ela indicou que, quando o personagem morre no tutorial, é importante uma opção para retornar ao ponto do desafio onde o personagem foi derrotado em vez de obrigar o jogador a refazer todo o tutorial.

A **professora B** (razoavelmente familiarizada com jogos digitais) comentou que muitos Ambientes de Desenvolvimento Integrados (IDEs, do inglês *Integrated Development Environment*) de software possuem opções de autocompletar comandos e que isso poderia ajudar o jogador ao mesmo tempo que o familiariza com funcionalidades que vai encontrar nesses ambientes. Adicionalmente, houve confusão com o nome dos comandos que nem sempre remetem a ações com resultados previsíveis e que são de difícil memorização. Por fim, ela também sugeriu utilizar os comandos em português para ampliar o acesso a estudantes sem familiaridade com a língua inglesa.

O **professor C** (bastante familiarizado com jogos do gênero RPG) reforçou a dificuldade em entender os nomes dos comandos. Paralelo a isso, o professor ainda mencionou que um pequeno *pop-up* que especificasse os parâmetros esperados durante a escrita dos comandos simplificaria significativamente sua utilização.

De forma geral, o tópico mais recorrente foi a falta de indicação de objetos interativos no cenário e tela. No tutorial do comando **pickUp**, por exemplo, existe um vaso que ao ser quebrado revela uma poção de vida que é o objetivo daquele tutorial em específico. Nenhum dos participantes identificou aquele objeto como quebrável e um participante pensou, inclusive, se tratar da poção de vida em si. Como outro exemplo, o inventário do personagem (ícone de sacola na Figura 1) não foi reconhecido por dois participantes. Isso ocorreu durante o tutorial do comando **inventory** no qual os jogadores são instruídos a utilizar a poção coletada e para isso precisam verificar sua posição no inventário.

Outro problema indicado foi a falta de um menu para consulta rápida da sintaxe dos comandos. Em todas as sessões, participantes esqueceram a sintaxe do código e precisaram consultar várias vezes os comandos de maneira informal pedindo para verem o pedaço de papel do protótipo referente ao tutorial que os descrevia. A própria opção da interface para rever as instruções do tutorial não foi reconhecida. Os docentes foram incentivados a desenhar um ícone para a opção de listar comandos, sendo que um tradicional menu hambúrguer mostrou-se adequado para exibir a lista dos comandos e, a partir dela, selecionar um comando para visualização de uso no jogo e detalhes de sintaxe.

Algumas instruções no tutorial também provocaram dúvidas nos participantes. A mensagem de explicação do comando **pickUp**, por exemplo, não deixava claro que o personagem e o item deveriam estar no mesmo bloco (quadrado da tela) para que o item pudesse ser coletado. Sugestões incluíram a possibilidade de coleta automática assim que a protagonista passasse pelo bloco contendo o item, embora isso removesse um comando interessante para ser explorado do ponto de vista pedagógico.

Os nomes dos comandos também geraram incerteza quanto a sua utilização, sendo os comandos **inventory** e **pickUp** os mais mencionados. Os participantes sugeriram então trocar **inventory** por **use**, e **pickUp** por **pick** ou **collect**.

Os resultados da análise das observações e das sugestões dos professores viabilizaram uma nova versão do protótipo[10] que será usada como base para testes com estudantes e, posteriormente, para desenvolvimento da primeira versão digital interativa do jogo.

6. Considerações finais

Um dos grandes desafios do ensino é proporcionar aprendizado atrativo e com qualidade. Os jogos sérios são uma proposta de solução dessa demanda, tornando o aprendizado divertido e interativo. No ensino de programação, jogos sérios têm sido criados para treinamento do pensamento computacional e aprendizado de programação.

Entretanto, a grande maioria desses jogos destina-se ao público infantil e tem como foco a solução de quebra-cabeças utilizando blocos de comando. Apesar de eficiente, a abordagem pode se tornar enjoativa e repetitiva, especialmente com estudantes mais velhos que estão iniciando seus estudos em computação. Além disso, essa estratégia é distante do código de programação que é alvo de aprendizado.

O jogo RFPG acrescenta um novo gênero aos jogos sérios para aprendizagem de programação, propiciando uma abordagem diferente da literatura atual ao exercitar o processo de codificação procedural de forma mais palpável e divertida. Apesar das vantagens citadas, a utilização do RFPG em sala de aula deve ser ponderada com base no nível de conhecimento de programação dos estudantes. Seu foco encontra-se em estudantes nos semestres iniciais de cursos de programação que procuram familiaridade com codificação baseada em textos. A abordagem do jogo pode não ser atrativa para crianças, por exemplo, onde a dinâmica predominante é a programação em blocos. Por outro lado, o RFPG se mostra como o próximo passo no ensino desse público.

A avaliação preliminar com três professores sugere benefícios significativos aos estudantes que experienciarem o jogo. Contudo, a avaliação com os próprios estudantes é uma etapa fundamental para identificar problemas de usabilidade e oportunidades de melhorar a experiência dos jogadores. Tendo em vista que estudantes são o público-alvo de nossa proposta, as impressões, sugestões e críticas desse grupo são essenciais [Schell 2014].

Os trabalhos futuros desse projeto incluem a avaliação por meio de protótipos em papel com estudantes de Algoritmos e Programação em uma perspectiva qualitativa, a construção da primeira versão do jogo utilizando a *engine* Godot em sua versão 4.2 (que proporciona diversos elementos pré-construídos que facilitam a inserção, coleta e interação do código-fonte desenvolvido pelo jogador ao longo do programa) e a avaliação da versão digital do jogo com professores e estudantes de Algoritmos e Programação em uma perspectiva quantitativa usando questionários padronizados para medir a experiência dos jogadores. A arte bidimensional do jogo será desenvolvida utilizando o software **libreSprite**, uma ferramenta gratuita para criação e animação de *sprites*, e os efeitos sonoros e músicas serão obtidos gratuitamente no catálogo da empresa Epidemic Sound.

[10]Disponível em: `https://doi.org/10.6084/m9.figshare.27249375`

Referências

Ericsson, A. and Simon, H. (1993). *Protocol Analysis, revised edition: Verbal Reports as Data*. MIT Press, USA.

Jenkins, T. (2002). On the difficulty of learning to program. In *3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, volume 4, pages 53–58. Loughborough University.

Malliarakis, C., Satratzemi, M., and Xinogalos, S. (2014). *Educational Games for Teaching Computer Programming*, pages 87–98. Springer New York, New York, NY, USA.

Marques, P., Mangeli, E., Monclar, R., and Xexéo, G. (2021). Desenvolvimento de um jogo digital educacional para o ensino de pensamento computacional concorrente. In *Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 68–75, Porto Alegre, RS, Brasil. SBC.

Mathrani, A., Christian, S., and Ponder-Sutton, A. (2016). Playit: Game based learning approach for teaching programming concepts. *Journal of Educational Technology & Society*, 19(2):5–17.

Mayer, R. E. (2019). Computer games in education. *Annual Review of Psychology*, 70(1):531 – 549.

Nipo, D., Rodrigues, R., França, R., Nascimento, J., and Pereira, M. (2023). Robo-think: Um jogo de realidade virtual para o ensino de habilidades de pensamento computacional. In *Anais Estendidos do XXII Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 915–924, Porto Alegre, RS, Brasil. SBC.

Pivec, M., Dziabenko, O., and Schinnerl, I. (2003). Aspects of game-based learning. In *3rd International Conference on Knowledge Management*, pages 216–225.

Qian, Y. and Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education*, 18(1):1–24.

Rocha, R., Bittencourt, I., and Isotani, S. (2015). Análise, projeto, desenvolvimento e avaliação de jogos sérios e afins: uma revisão de desafios e oportunidades. In *Anais do XXVI Simpósio Brasileiro de Informática na Educação - SBIE*, pages 692–701.

Rubin, J. and Chisnell, D. (2008). *Handbook of usability testing : how to plan, design, and conduct effective tests*. Wiley Publishers, Indianapolis, Ind., USA.

Schell, J. (2014). *The Art of Game Design: A Book of Lenses*. A. K. Peters, Ltd., USA, 2nd edition.

Screpanti, M. (2024). Crescimento do mercado de programação: veja o que especialista explica sobre o assunto. Revista Nacional da Tecnologia da Informação.

Snyder, C. (2003). *Paper Prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann, San Francisco, CA, USA.