UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

CARLOS HENRIQUE BRAVO SERRADO
FRANCISCO JOSÉ TAAM SANTOS DE OLIVEIRA
LUCAS ARAUJO CARVALHO

IMPLEMENTAÇÃO DE UM JOGO COMO FERRAMENTA DE AUXÍLIO AO DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL

RIO DE JANEIRO
2024

CARLOS HENRIQUE BRAVO SERRADO
FRANCISCO JOSÉ TAAM SANTOS DE OLIVEIRA
LUCAS ARAUJO CARVALHO

IMPLEMENTAÇÃO DE UM JOGO COMO FERRAMENTA DE AUXÍLIO AO DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profa. Valeria Menezes Bastos

RIO DE JANEIRO
2024

Serrado, Carlos Henrique Bravo
S487i Implementação de um jogo como ferramenta de auxílio ao desenvolvimento do pensamento computacional / Carlos Henrique Bravo Serrado, Francisco José Taam Santos de Oliveira e Lucas Araujo Carvalho. – 2024.

52 f.

Orientadora: Valeria Menezes Bastos.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação)- Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2024.

1. Jogos educacionais. 2. Pensamento computacional. 3. Ferramentas ludificadas. 4. Desenvolvimento de jogos. I. Oliveira, Francisco José Taam Santos de. II. Carvalho, Lucas Araujo. III. Bastos, Valeria Menezes (Orient.). IV. Universidade Federal do Rio de Janeiro, Instituto de Computação. V. Título.

CARLOS HENRIQUE BRAVO SERRADO
FRANCISCO JOSÉ TAAM SANTOS DE OLIVEIRA
LUCAS ARAUJO CARVALHO

IMPLEMENTAÇÃO DE UM JOGO COMO FERRAMENTA DE AUXÍLIO AO DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 26 de março de 2024.

BANCA EXAMINADORA:

Documento assinado digitalmente
gov.br
VALERIA MENEZES BASTOS
Data: 10/04/2024 18:03:11-0300
Verifique em https://validar.iti.gov.br

Valeria Menezes Bastos
Dr. (UFRJ)

Documento assinado digitalmente
gov.br
JOAO ANTONIO RECIO DA PAIXAO
Data: 11/04/2024 09:16:02-0300
Verifique em https://validar.iti.gov.br

João Antônio Recio da Paixão
Dr. (UFRJ)

Documento assinado digitalmente
gov.br
RONALD CHIESSE DE SOUZA
Data: 10/04/2024 22:17:59-0300
Verifique em https://validar.iti.gov.br

Ronald Chiesse de Souza
Dr. (UFRJ)

Documento assinado digitalmente
gov.br
SILVANA ROSSETTO
Data: 09/04/2024 17:18:14-0300
Verifique em https://validar.iti.gov.br

Silvana Rossetto
Dr. (UFRJ)

Dedicamos esse trabalho às nossas famílias e parceiros que nos apoiaram e motivaram durante nossa trajetória de vida. Aos nossos amigos, em especial o grupo Winx, que permaneceu unido durante toda nossa graduação e a diversos colegas de curso que nos inspiraram e ajudaram como Cássio Silva, Daniel Hashimoto, Letícia Freire e Luan Martins. Aos nossos animais de estimação por nos animarem em diversos momentos. À GDP, na qual discutimos diversos tópicos de desenvolvimento de jogos e nos animaram a começar esse projeto, além de nos abrigar na sala durante momentos de tédio entre aulas.

Em memória de Nina (2012-2023), uma felina laranjinha dengosa que aqueceu nossos corações diversas vezes e que partiu precocemente. Nunca será esquecida pelo seu mio rouquinho para pedir comida ou colo, seu andar exótico, e suas marcas de arranhão nas portas de casa quando queria entrar em algum dos quartos.

AGRADECIMENTOS

Gostaríamos de agradecer nossa orientadora Valeria Bastos que aceitou de prontidão a ideia do trabalho e nos ajudou durante o último ano, mesmo com prazos cada vez mais apertados. Também a Naomi Nitahara, que apesar de não ser orientadora, nos ajudou a revisar o texto diversas vezes.

Agradecemos também a Gustavo Araújo, Iago Rafael Lucas Martins, Lucas Moreno e Markson Arguello, integrantes originais de *Robots in Terveijan*. Apesar de diversas mudanças no projeto, esse trabalho não seria possível sem o apoio inicial do grupo.

Por fim, agradecemos também aos nossos orientadores acadêmicos João Paixão e Maria Luiza Campos e orientador de iniciação científica Daniel Sadoc por nos motivarem em momentos difíceis permitindo a conclusão do curso.

"A game is a problem-solving activity,
approached with a playful attitude."

Jesse Schell

RESUMO

Este estudo investiga a viabilidade do uso de um jogo eletrônico como ferramenta complementar no ensino do pensamento computacional em disciplinas introdutórias de computação. Uma revisão do estado da arte foi conduzida, examinando a literatura sobre ferramentas ludificadas e jogos educativos, culminando no desenvolvimento do jogo *Robots*, que foi projetado para desafiar os estudantes a resolver problemas desconhecidos por meio do pensamento computacional e da linguagem de blocos. A metodologia incluiu o desenvolvimento iterativo do jogo e sua avaliação por meio de um formulário aplicado a estudantes e ex-estudantes da Universidade Federal do Rio de Janeiro (UFRJ). Os participantes foram questionados sobre os aspectos lúdicos e educativos do jogo, bem como forneceram sugestões para melhorias. Com base nos resultados da avaliação, foram realizadas melhorias significativas no jogo. Os resultados indicaram uma receptividade positiva ao jogo, especialmente em fases mais avançadas, sugerindo engajamento e interesse crescente à medida que os conceitos se tornavam mais complexos. Este trabalho contribui para a compreensão do papel dos jogos eletrônicos no ensino de pensamento computacional e fornece reflexões valiosas para o desenvolvimento de ferramentas educacionais inovadoras.

Palavras-chave: jogos educacionais; pensamento computacional; ferramentas ludificadas; desenvolvimento de jogos.

ABSTRACT

This study investigates the feasibility of using an electronic game as a complementary tool in teaching computational thinking in introductory computer science courses. A state-of-the-art review was conducted, examining the literature on gamified tools and educational games, culminating in the development of the game *Robots*, which was designed to challenge students to solve unknown problems through computational thinking and block-based language. The methodology included the iterative development of the game and its evaluation through a questionnaire administered to students and alumni of the Federal University of Rio de Janeiro (UFRJ). Participants were asked about the game's ludic and educational aspects, as well as providing suggestions for improvements. Based on the evaluation results, significant improvements were made to the game. The findings indicated a positive receptivity to the game, especially in more advanced stages, suggesting increasing engagement and interest as concepts became more complex. This work contributes to understanding the role of electronic games in teaching computational thinking and provides valuable insights for the development of innovative educational tools.

Keywords: educational games; computational thinking; gamified tools; game development.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

UFRJ	Universidade Federal do Rio de Janeiro
GDD	Game Design Document

SUMÁRIO

1 INTRODUÇÃO

Uma das habilidades mais importantes para um estudante de ciência da computação é o Pensamento Computacional, que permite pensar de forma analítica e abstrata sobre problemas e sistemas (WING, 2006). Tal capacidade de decompor problemas em outros menores, perceber padrões e planejar um conjunto de passos simples para solucioná-los é imprescindível para assuntos avançados do curso, como desenvolvimento de algoritmos e sistemas. O primeiro contato de estudantes com Pensamento Computacional costuma ser paralelamente com o ensino de sua primeira linguagem de programação devido à sua fundamentalidade à disciplina. No entanto, o foco costuma ser na linguagem em si, deixando o aprendizado desse conceito de lado, apesar de estar se tornando cada vez mais necessário em diversas áreas (WING, 2008).

As disciplinas introdutórias de computação são de extrema importância e são base para o restante das matérias. Apesar de serem disciplinas com índices positivos de número de aprovações, costumam serem encarada com certa dificuldade por alguns estudantes. No curso de Ciência da Computação da Universidade Federal do Rio de Janeiro (UFRJ), cerca de um terço dos estudantes matriculados na disciplina de Computação I não são aprovados (FERREIRA; CANAANE, 2021), seja por reprovação ou abandono. Possíveis motivos que podem levar a esses números são a dificuldade de foco e organização pessoal ou dificuldade nas exigências da disciplina (SILVA et al., 2021). Por conta disso, mostram-se importantes novas abordagens para o ensino dessa disciplina, como ferramentas educativas ludificadas para auxiliar na didática e engajar os estudantes.

A ludificação, também conhecida como gamificação, refere-se ao uso de elementos e ideias comumente associados a jogos em diferentes contextos, com o intuito de tornar certas atividades mais interessantes. Elementos como placares, pontuação, objetivos e questionários despertam uma sensação de progresso e competitividade, e potencialmente fazem com que o engajamento nestas atividades se torne maior. Este potencial benefício é valioso no contexto de educação, e existem diversos *frameworks* na literatura propondo formas de utilizar esta técnica para tal (ALMEIDA et al., 2023).

No contexto de jogos, não existe uma definição formal para determinar o que pode ser considerado como tal. O livro *The Art of Game Design: A Book of Lenses* de Jesse Schell traz a seguinte definição, em tradução livre: "Um jogo é uma atividade de resolução de problemas, abordada com uma atitude lúdica." (SCHELL, 2008). Seguindo tal definição, é possível considerar plataformas como *Duolingo* (Duolingo Inc., 2011) e *Kahoot!* (Kahoot! ASA, 2012) como jogos tal qual *Human Resource Machine* (Tomorrow Corporation, 2015) e *Valiant Hearts: The Great War* (Ubisoft Entertainment, 2014), mas é importante distinguir os objetivos de cada um. Enquanto os dois primeiros têm como objetivo principal o aprendizado, e para isso utilizam de elementos de design de jogos, os

outros dois têm como foco o entretenimento, mas possuem um fator educativo presente.

Os impactos da ludificação ainda são amplamente discutidos. Por um lado, a prática pode ser considerada uma jogada de marketing oportunista para se beneficiar do caráter apelativo dos jogos (BOGOST, 2015), assim como podem trazer efeitos negativos, como aumento excessivo de competitividade ou estresse (ALMEIDA et al., 2023). Por outro lado, estudos indicam impactos positivos no aprendizado como melhora na retenção de informação por parte dos estudantes (PUTZ; HOFBAUER; TREIBLMAIER, 2020), que podem ser aproveitados para melhorar o aprendizado. Alguns jogos com o propósito de ensinar programação ou pensamento computacional seguiram a linha lúdica, sendo uma ideia comum abordada de diversas formas, normalmente apresentando um problema que deve ser resolvido com código (KAZIMOGLU et al., 2012) (LIU; CHENG; HUANG, 2011) (PARSONS; HADEN, 2006).

Considerando a importância do pensamento computacional, os impactos positivos de ferramentas ludificadas no processo de aprendizagem e a forma como costuma ser abordado, esse trabalho se propõe a criar um jogo, *Robots*, para o ensino ludificado de Pensamento Computacional. O jogador, em posse de um robô lutador, deve observar o funcionamento de um código pertencente a um robô inimigo e identificar como o mesmo é formado, descobrindo assim o problema e permitindo o desenvolvimento de uma solução com uma linguagem de programação em blocos para derrotá-lo. Mecânicas auxiliares para incentivo da solução e otimização, como a interface de depuração e um sistema de medalhas por algoritmos curtos e eficientes, serão implementados. Essa aplicação pode ser utilizada em salas de aulas para auxílio da docência do conteúdo, que também poderá contar com um criador de fases para desenvolver desafios voltados para as necessidades dos alunos. Além dos autores deste trabalho, o projeto foi concebido e inicializado na disciplina Introdução ao Desenvolvimento de Jogos Eletrônicos sob docência de Luiz Ribeiro, com participação de Gustavo Araújo (arte e animação), Iago Rafael Lucas Martins (*UX/UI design*), Lucas Moreno (programação) e Markson Arguello (programação).

Um formulário, com o intuito dos estudantes avaliarem a sua experiência, foi distribuído junto com o jogo e as respostas foram analisadas para trazer uma melhor compreensão das qualidades e defeitos da aplicação criada. Algumas das sugestões obtidas foram implementadas e outras foram consideradas para versões futuras do projeto. Com essas melhorias, foi realizada uma atividade com estudantes novos de ciência da computação para averiguar o impacto das alterações.

O capítulo 2 apresenta trabalhos relacionados e suas similaridades e diferenças. O capítulo 3 descreve o processo de criação do jogo, com sua concepção, visão geral e implementação. O capítulo 4 traz os resultados levantados com o formulário inicial, melhorias realizadas e reflexões sobre a avaliação em laboratório. Por fim, o capítulo 5 conclui o trabalho e levanta problemas que serão abordados futuramente.

2 TRABALHOS RELACIONADOS

Dentro do âmbito educacional, a ludificação ganha destaque como uma abordagem inovadora, explorando elementos e dinâmicas típicas de jogos para enriquecer experiências de aprendizagem. A integração de mecânicas de jogos, como desafios, recompensas e narrativas, visa não apenas tornar as atividades mais atraentes, mas também promover o engajamento dos aprendizes de maneira significativa. Ao adentrar o universo das ferramentas ludificadas e dos jogos educativos, abre-se um vasto leque de possibilidades para enriquecer o processo de ensino e aprendizagem, oferecendo ambientes interativos e estimulantes que favorecem o desenvolvimento de habilidades cognitivas, sociais e emocionais. Nesse contexto, explorar e compreender as nuances dessas ferramentas torna-se crucial para a construção de práticas pedagógicas mais dinâmicas e eficazes, capazes de atender às demandas e expectativas dos estudantes contemporâneos.

Os jogos educativos e as ferramentas ludificadas representam abordagens distintas, embora relacionadas, no contexto da aprendizagem. O primeiro é projetado com o propósito principal de ensinar conceitos, habilidades ou conteúdos específicos de forma indireta e implícita. Eles geralmente seguem uma estrutura de jogo clássica, com objetivos claros, desafios progressivos e feedback imediato, tudo isso com o objetivo de promover a aquisição de conhecimento de maneira lúdica e envolvente. Por outro lado, as ferramentas ludificadas referem-se a elementos e estratégias de ludificação incorporados em atividades educacionais ou em ambientes de aprendizagem para aumentar o engajamento, a motivação e a participação dos alunos. Estas ferramentas podem incluir elementos como sistemas de pontuação, recompensas, níveis de dificuldade adaptativos e narrativas envolventes, que são integrados em contextos não necessariamente ligados a jogos, como aplicativos de aprendizagem, plataformas online e até mesmo salas de aula físicas. Enquanto os jogos educativos trazem uma experiência divertida com tom didático, as ferramentas ludificadas visam transformar a prática de aprendizagem tradicional, tornando-a mais atrativa e estimulante para os alunos.

Duolingo e *Kahoot!* são exemplos proeminentes de ferramentas ludificadas que revolucionaram a educação de idiomas e a avaliação interativa, respectivamente. *Duolingo*, uma aplicação de aprendizado de idiomas amplamente reconhecida, adota uma abordagem inovadora ao transformar o processo de aquisição de línguas em uma jornada envolvente e interativa. A plataforma utiliza uma variedade de elementos lúdicos, como conquistas, pontos, níveis e desafios, para manter os usuários motivados e comprometidos com seu progresso linguístico (Figura 1). Além disso, o *Duolingo* incorpora técnicas de aprendizagem, como repetição espaçada, e adaptação de dificuldade com base no desempenho do usuário, oferecendo uma experiência personalizada e eficaz de aprendizado de idiomas.

Figura 1 – Tela de pergunta de *Duolingo*

Fonte: Página da Google Play [1]

Por outro lado, o *Kahoot!* é uma ferramenta que transforma avaliações tradicionais em experiências de aprendizagem colaborativa e empolgante. Com o *Kahoot!*, educadores podem criar questionários, pesquisas e discussões interativas que os alunos podem participar em tempo real utilizando dispositivos móveis ou computadores (Figura 2). A plataforma apresenta uma interface amigável e vibrante, incentivando a competição saudável e a participação ativa dos alunos por meio de placares de pontuação em tempo real e recompensas virtuais. A combinação de elementos de ludificação, como temporizadores e música de fundo cativante, transforma avaliações monótonas em experiências educacionais dinâmicas e memoráveis.

O *Scratch* (Scratch Foundation, 2014) é uma plataforma de programação visual projetada especialmente para crianças e jovens, permitindo que eles criem histórias interativas, animações e jogos de forma intuitiva e divertida. Foi criado em 2008, no Media Lab, do Instituto de Tecnologia de Massachusetts (MIT) e teve como uma de suas principais inspirações a linguagem LOGO, também criada no MIT no final da década de 1960, que

[1] Disponível em: https://play.google.com/store/apps/details?id=com.duolingo&hl=pt_BR&gl=US. Acesso em: 26 fev. 2024

[2] Disponível em: https://play.google.com/store/apps/details?id=no.mobitroll.kahoot.android&hl=pt_BR&gl=US. Acesso em: 26 fev. 2024

Figura 2 – Tela de pergunta de *Kahoot!*

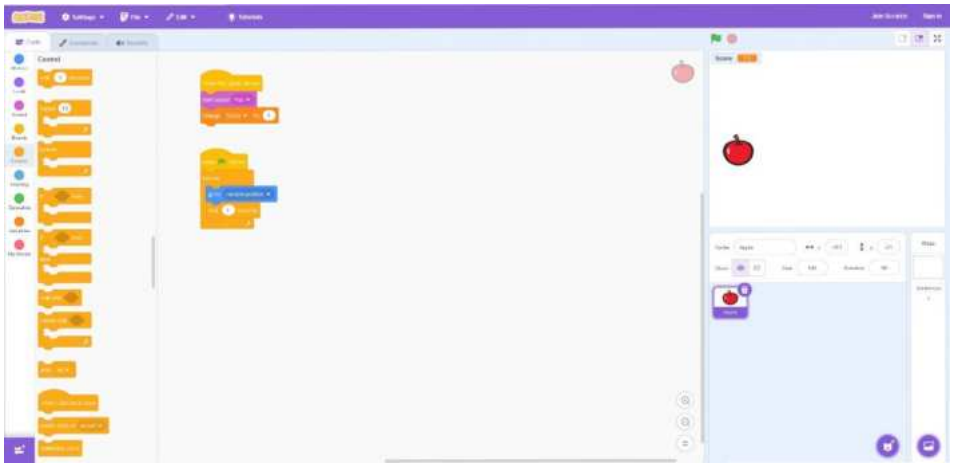
Fonte: Página da Google Play [2]

tinha como um de seus objetivos ser uma linguagem simples e acessível capaz de estimular a criatividade e o pensamento computacional em crianças (CAMARGO; FORTUNATO, 2018). Utilizando uma interface amigável baseada em blocos de código, os usuários podem criar programas combinando diferentes comandos e ações sem a necessidade de conhecimento prévio de linguagens de programação tradicionais (Figura 3). A simplicidade do *Scratch* torna a programação acessível a um público amplo, incentivando a criatividade, o pensamento lógico e a resolução de problemas. No contexto deste trabalho, a escolha de basear-se no *Scratch* ressalta a importância dos aspectos de programação em blocos, que facilitam a implementação de ideias complexas de maneira estruturada e visualmente compreensível, permitindo que os participantes explorem conceitos de computação de forma prática e envolvente.

Apesar das semelhanças com relação a alguns títulos citados, já que também utilizam programação em blocos, é essencial destacar o diferencial do projeto: o depurador. De todos os exemplos citados, ou o processo de depuração é introduzido de uma forma implícita e indireta ou não há um. Este trabalho, por outro lado, oferece uma interface para depuração clara e objetiva acerca de sua finalidade e aplicação, além de estar inserida na progressão do jogo como uma ferramenta de inestimável apoio na solução dos desafios propostos. Portanto, há destaque para uma etapa de grande importância no percurso de aprendizagem do pensamento computacional e que não é apresentada de forma evidente

em outras alternativas lúdicas conhecidas.

Figura 3 – Jogo programado em *Scratch*



Fonte: Página do *Scratch* [3]

Embora o *Scratch* e outras ferramentas ludificadas sejam fundamentais para introduzir conceitos de forma acessível, este trabalho se concentra na criação de um jogo educativo que combine entretenimento e aprendizado. Nesse sentido, há exemplos de jogos que incorporam elementos educativos de maneira eficaz. *while True: learn()* (Luden.io, 2019) é um jogo que desafia os jogadores a assumir o papel de um desenvolvedor de IA, resolvendo quebra-cabeças de programação para treinar algoritmos de aprendizado de máquina (Figura 4). A maneira como as fases são pensadas, com o jogador tendo que construir sistemas que liguem nós de entrada para obter as saídas desejadas, e a adição gradual de nós intermediários cada vez mais complexos oferece uma visão lúdica do processo de modelagem de redes neurais, fazendo com que o jogador pratique tais habilidades implicitamente.

Valiant Hearts: The Great War é outro exemplo notável, apresentando uma narrativa emocionante ambientada na Primeira Guerra Mundial. Esse jogo combina elementos de aventura com quebra-cabeças, e utiliza seu enredo e personagens para trazer uma perspectiva única e imersiva daquele momento. Além disso, a presença de "fatos históricos" (artefatos reais, documentos e curiosidades relacionadas à Primeira Guerra) como colecionáveis espalhados pelo jogo incita o jogador à descobrir mais sobre o contexto histórico da época.

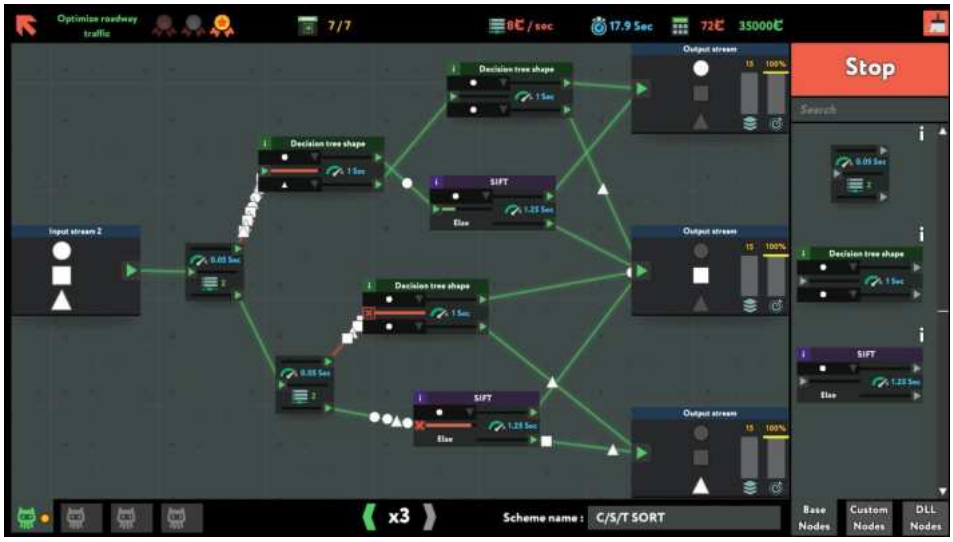
Human Resource Machine é um jogo de quebra-cabeça que desafia os jogadores a resolver problemas utilizando uma linguagem de programação de baixo nível. Ambientado

[3] Disponível em: https://scratch.mit.edu/projects/editor/?tutorial=getStarted. Acesso em: 26 fev. 2024

[4] Disponível em: https://store.steampowered.com/app/619150/while_True_learn/. Acesso em: 26 fev. 2024

[5] Disponível em: https://steamcommunity.com/sharedfiles/filedetails/?id=3098366901. Acesso em: 26 fev. 2024

Figura 4 – Fase de *while True: learn()*



Fonte: Página do jogo no *Steam* [4]

Figura 5 – Tela de *Valiant Hearts*

Fonte: Página do jogo no *Steam* [5]

em um escritório corporativo futurista, os jogadores assumem o papel de um funcionário encarregado de completar tarefas através da programação de um robô (Figura 6). Com sua abordagem desafiadora e educativa, *Human Resource Machine* oferece uma oportunidade para os jogadores desenvolverem habilidades de lógica e pensamento algorítmico.

Nesse capítulo, explorou-se a importância da ludificação e das ferramentas educativas no contexto da aprendizagem, reconhecendo o propósito das abordagens inovadoras que integram elementos de jogos para enriquecer a experiência educacional. Embora tenham sido analisadas diversas ferramentas, o foco principal deste trabalho recai sobre

[6] Disponível em: https://store.steampowered.com/app/375820/Human_Resource_Machine/. Acesso em: 26 fev. 2024

Figura 6 – Tela de *Human Resource Machine*

Fonte: Página do jogo no *Steam* [6]

o desenvolvimento de um jogo educativo que harmonize entretenimento e aprendizado. Inspirados em exemplos como *while True: learn()*, *Valiant Hearts* e *Human Resource Machine*, buscou-se criar uma experiência que ofereça não apenas diversão, mas também oportunidades significativas de se exercitar o pensamento computacional.

3 DESENVOLVIMENTO

A pré-produção demandou decisões fundamentais para escopo, ferramentas, estilo artístico e tecnologias a serem aplicadas. Para isso, foi elaborado e redigido um *Game Design Document* (GDD), um documento para nortear e descrever todos os aspectos do jogo (CARVALHO; SERRADO; OLIVEIRA, 2021). Em particular, o GDD deste projeto contém a descrição completa sobre o funcionamento da tela de depuração, a lógica das funções do robô do usuário, o sistema de medalhas por desempenho de solução, o sistema de criação de fases, entre outros. No entanto, é importante destacar que, durante a produção, é comum ocorrer desvios de planejamento em função de algum contexto (prazos, inviabilidades técnicas e etc.).

A produção, por sua vez, é o processo de desenvolvimento onde artistas, programadores, músicos e roteiristas materializam o conteúdo planejado durante a pré-produção. Nessa etapa, ferramentas específicas assumem um papel-chave para uma ou mais áreas de atuação. No caso de *Robots*, destacaram-se as ferramentas *Unity* (Unity Technologies, 2005), *GitHub* (Microsoft, 2008) e *Figma* (Figma, Inc., 2016). Para a programação, o *Unity* se fez presente para implementar todo o código acerca do projeto, desde o compilador da programação em blocos até os menus, sistema de criação de fases e sistema de combate e medalha, posteriormente publicados em repositório no *GitHub* (CARVALHO et al., 2021). Todos foram implementados usando a linguagem C#. Este *software* oferece uma ampla gama de recursos e funcionalidades, como mecanismos de física robusto, suporte gráfico avançado, ferramentas de animação, áudio e *networking*. O *Figma*, por outro lado, oferece um poderoso ambiente voltado para o *design* de interface de usuário e viabiliza prototipagem de qualquer grau de fidelidade. Este editor gráfico permite experimentar diferentes *layouts*, estilos visuais, esquemas de cores e fluxos de interação, catalizando o processo de teste de ideias antes da implementação. No contexto do projeto *Robots*, o *Figma* foi utilizado para a elaboração da identidade visual, além de ter sido usado na criação de todos os botões, interfaces e ícones.

A pós-produção é o período de intensivos testes e polimento. Isso envolve a identificação e correção de possíveis *bugs*, falhas de programação, problemas de desempenho e quaisquer questões relacionadas à jogabilidade que possam impactar a experiência do jogador de forma negativa. No contexto deste projeto, uma versão funcional do jogo foi gerada e distribuída para diversos grupos distintos. Um formulário sobre a experiência de *Robots* também foi distribuído para esses grupos, a fim de coletar *feedback* e direcionar o polimento.

3.1 CONCEPÇÃO

O jogador realiza todas as ações do jogo por meio do mouse. Na montagem do código, o usuário pode arrastar os blocos para o algoritmo. Entre elas, há estruturas de programação básicas (if, else, while, for) e funções específicas do robô (attack, defend, charge, heal). Com as variáveis, será possível observar exatamente o seu significado ao manter o cursor acima de cada ícone. Entre elas, há a quantidade de turnos e a vida atual e máxima dos robôs do usuário e do adversário (podendo ser o valor atual, dobrado ou metade), além de outras variáveis. Elas podem ser utilizadas em loops condicionais para aumentar o leque de possibilidades de solução. O objetivo do jogo é identificar o padrão de ações do oponente e programar uma solução capaz de superá-lo. Para auxiliar o usuário nesse desafio, haverá uma espécie de registro acerca de todo o combate (Tela de Depuração), que será aberto ao final de cada execução. Ao abrir, será possível verificar valores sobre as variáveis, rodada por rodada.

Todas as lutas possuem múltiplas soluções, contudo, algumas serão mais eficientes pela quantidade de passos necessários e/ou pelo tamanho de blocos utilizados. Por conta disso, cada solução alinhada perfeitamente com essas métricas receberá medalhas, sendo então uma para melhor desempenho possível (menor número de rodadas) e outra para menor quantidade de linhas possível utilizada. Assim, o jogador é incentivado a refazer as fases, em busca de um resultado mais otimizado.

Além do ensino do pensamento computacional, o projeto *Robots* também apresenta outros ganhos no ensino de programação para alunos da área. Entre eles, temos a introdução de um dos alicerces no cotidiano de um desenvolvedor: o processo de depuração, que consiste na investigação, localização e erradicação de falhas presentes em um algoritmo. Em paralelo com o projeto, é importante para o usuário sempre acompanhar o relatório de combate fornecido na interface de depuração para fornecer um mapeamento completo de toda a execução, assim catalisando o encontro da(s) falha(s) lógicas no combate do robô.

O terceiro principal ganho potencializado no projeto é o incentivo à dedução de como processos funcionam, ou seja, a análise suficientemente precisa de um sistema existente para entender seu funcionamento interno, sem necessariamente vê-lo. Muitos desafios reais são contemplados e solucionados por via da programação. No jogo, é necessário realizar essa análise no padrão do robô adversário para obter uma resposta possível em forma de código.

Outro componente de extrema importância foi a constituição do *Level Design*, isto é, a montagem de cada fase do jogo. (SCHELL, 2008) afirma, em tradução livre: "Uma das chaves para um bom *level design* é que os olhos do jogador os conduzam pelo nível, sem esforço. Isso faz com que o jogador se sinta no controle e imerso no mundo. Entender o que atrai o olhar do jogador pode lhe conferir um tremendo poder sobre as escolhas que

os jogadores desejam fazer.". Portanto, é imprescindível a montagem de cada fase em uma ordem de progressão intuitiva e que introduza o conteúdo de programação abordado em conformidade com programas de uma disciplina de computação do ensino superior.

Pensando nesta progressão, as 8 fases padrões do projeto foram projetadas de forma que o jogador possa se familiarizar gradualmente com as mecânicas do jogo e os conceitos apresentados (Apêndice A). As três primeiras fases tem baixa complexidade, e tem como principal função introduzir as ações básicas do jogo. A primeira apresenta o ataque e a cura, a segunda, a defesa e a terceira, a carga. A partir da quarta fase, conceitos básicos de programação começam a ser introduzidos. Na fase 4, o jogador é introduzido ao bloco *while* e à condição *true*, trazendo uma forma mais conveniente de fazer repetições e a ideia de escopo. As fases 5 e 6 tem como objetivo mostrar os blocos condicionais e os comparadores. E, por fim, as fases 7 e 8 introduzem o bloco *for* e oferecem inimigos com uma complexidade maior.

Por último, vale destacar o sistema de criação de fases e levantar suas possíveis aplicações. Com esse mecanismo, docentes podem projetar desafios personalizados de acordo com temas e aplicações específicas discutidas em aula, ou ainda reforçar o aprendizado de alguma das estruturas de programação abordadas com uma quantidade maior de desafios. Ademais, o próprio criador de fases pode ser um ambiente no qual alunos são engajados a desenvolver fases em conjunto como parte de algum método avaliativo. O criador de fases, além de permitir programar o algoritmo adversário em blocos, possibilita ainda a modificação de todos os atributos dos dois robôs, escolher quais blocos estarão à disposição para uso do jogador, definir as metas para as duas medalhas e, por fim, a mensagem de dica.

3.2 VISÃO GERAL

O jogo é apresentado com uma temática de robôs e uma estética fantasiosa. A interface segue uma abordagem simplificada, com botões em cores vibrantes para facilitar a visualização.

A partir da tela inicial (Figura 7) é possível acessar as duas principais funcionalidades do jogo: as fases e a tela de criação. Além disso, ela também possui um guia, representado por um botão em forma de interrogação, que oferece algumas explicações básicas sobre o objetivo do jogo e suas funcionalidades.

Na tela de seleção de fases (Figura 8), as fases disponíveis são representadas por botões numerados, e também possuem dois símbolos que representam as medalhas conquistadas naquela fase. Fases próprias do jogo são representadas com um contorno vermelho e fases customizadas, por um azul.

A tela de criação (Figura 9) é por onde o usuário pode criar suas próprias fases. Os blocos presentes no canto superior esquerdo podem ser arrastados para o painel à direita

Figura 7 – Tela inicial do jogo

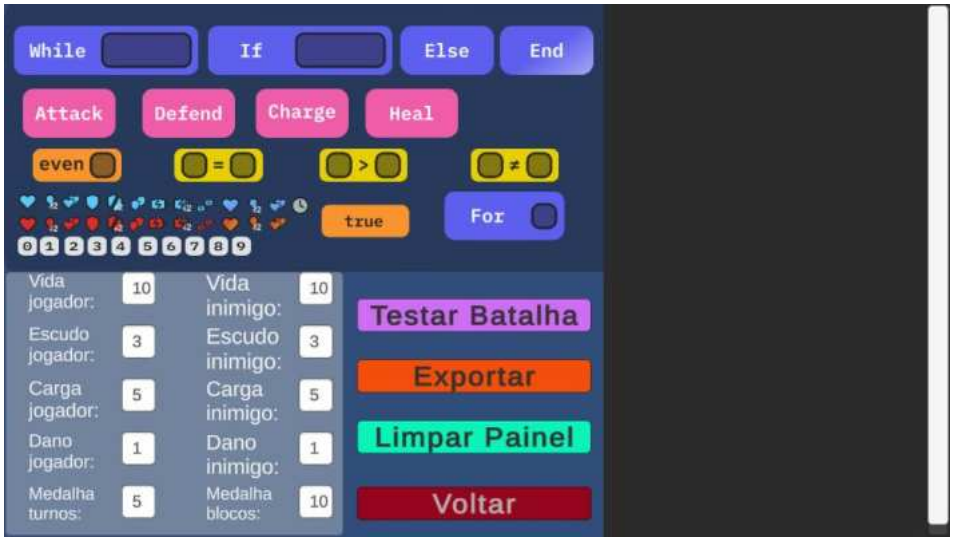
Figura 8 – Tela de seleção de fases

para montar o algoritmo do inimigo. Também é possível selecionar quais blocos estarão disponíveis para o jogador clicando com o botão direito do mouse sobre os blocos e definir os parâmetros da fase inserindo os valores desejados nos espaços apropriados.

O botão "Testar Batalha" pode ser usado caso o usuário queira ver o funcionamento do algoritmo durante a criação. Após ter a fase pronta, o botão exportar deve ser usado para que a fase criada fique disponível para jogar. Fases criadas pelos usuários ficam salvas na pasta *CustomMemories* nos arquivos do jogo e podem ser compartilhadas para outros jogadores. Já as fases padrões do jogo ficam na pasta *Memories*.

A tela de batalha (Figura 10) é a tela principal do jogo, acessada após escolher uma fase ou pelo botão de testar batalha da tela de criação. Assim como a tela de criação, essa tela também disponibiliza blocos que podem ser arrastados para o painel à direita para criar o algoritmo do robô jogador. No menu superior são mostradas as medalhas

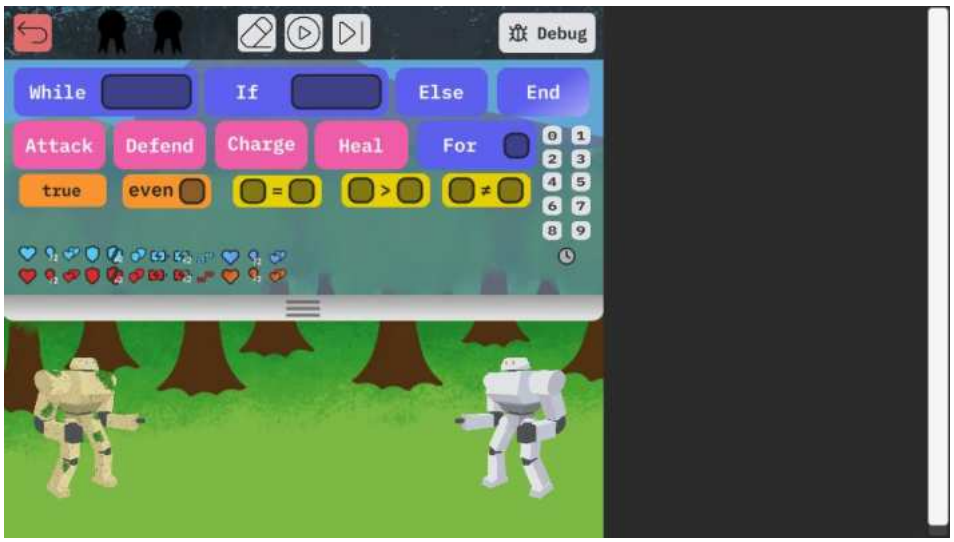
Figura 9 – Tela de criação de fases



adquiridas na fase e alguns botões de controle da batalha, bem como um botão de voltar para sair da tela. Com um algoritmo válido no painel, o jogador pode usar o botão de executar para compilar e iniciar a batalha.

Após executar uma batalha, o jogador pode ver uma representação visual dela pelas animações dos robôs no canto inferior da tela. Também está disponível no menu superior um botão para pular as animações. Além disso, o botão "*Debug*"pode ser utilizado para alternar entre o painel de blocos e o painel de depuração, que contém a execução passo a passo da batalha em forma de texto.

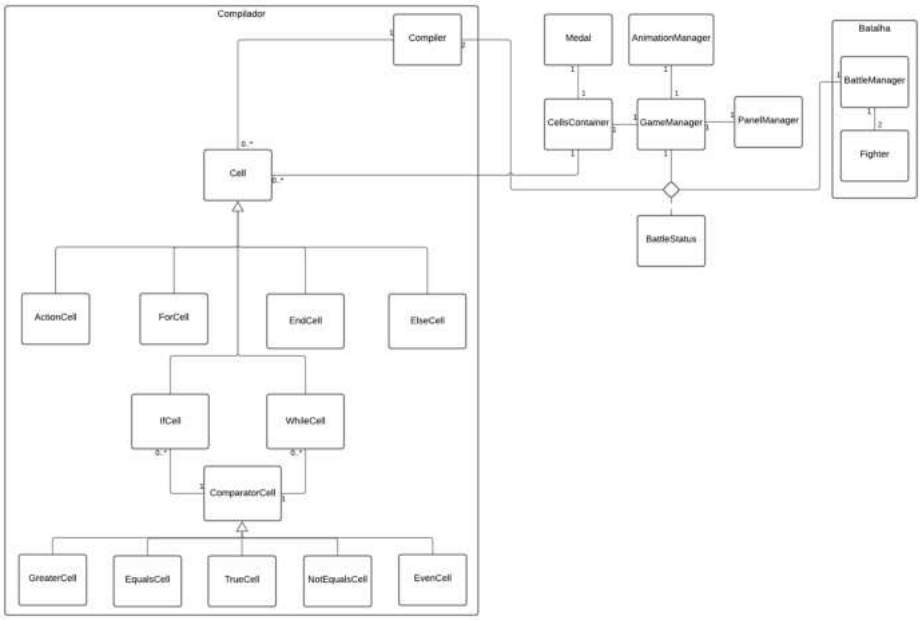
Figura 10 – Tela de batalha



3.3 IMPLEMENTAÇÃO

O código do projeto pode ser separado em 3 partes principais: compilador, batalha e interface. Cada uma dessas partes possui classes específicas para determinados problemas e são conectadas entre si através de uma classe principal chamada *GameManager*, responsável por realizar a comunicação entre os diversos componentes e realizar funções gerais que não podem ser delegadas a um componente específico. A Figura 11 mostra um diagrama de classes simplificado, dando foco nas relações entre os componentes.

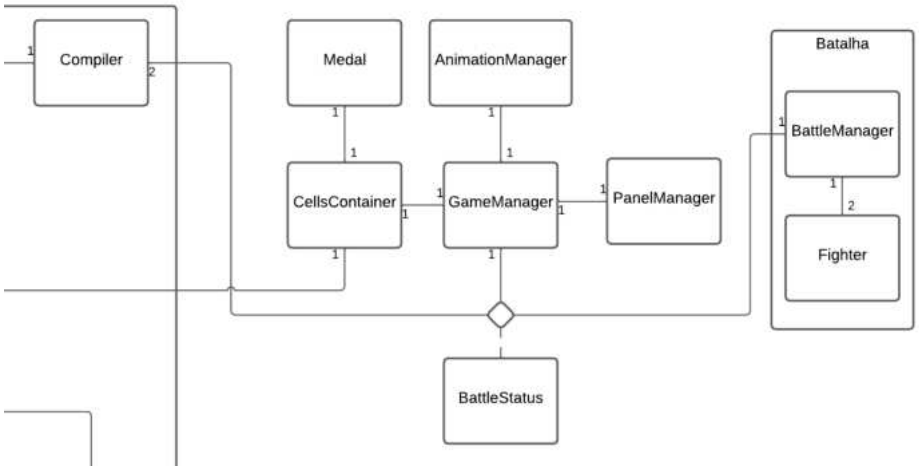
Figura 11 – Diagrama de classes



Quando uma batalha começa, é preciso permitir a criação de código arrastando os blocos de comando, determinar qual a ação de cada robô com base no código, determinar o resultado após essas ações e atualizar e salvar as medalhas de acordo com o resultado final. Essas funções estão delegadas às classes *PanelManager*, *Compiler*, *BattleManager* e *CellsContainer*, sendo intermediadas pelo *GameManager*. Esse ciclo pode ser melhor visualizado na Figura 12. Cada lutador é representado por uma classe *Fighter*, contendo os atributos e métodos necessários para a batalha, havendo uma referência para cada um no *BattleManager*, que realiza os turnos de acordo com as ações. Após realizar cada turno, é gerado um *BattleStatus*, contendo as informações de cada lutador em determinado turno da batalha. Além de gerenciar as medalhas, encapsuladas na classe *Medal*, a classe *CellsContainer* é responsável por conter o código da batalha atual, com cada batalha sendo uma instância serializada e salva em um arquivo binário.

Para que o discente possa executar um código e o docente possa criar um código para um inimigo, é preciso converter da programação em blocos para uma estrutura que possa

Figura 12 – Diagrama de classes - Principal



ser facilmente executada. Os blocos são representados por *GameObjects*[1] e, ao arrastar para o painel de programação, é criada uma lista desses *GameObjects*. Para representar de forma mais simples, os objetos são convertidos para enumerações, transformando assim a representação em inteiros para facilitar sua manipulação e armazenamento. Cada bloco é convertido em uma lista de uma a quatro enumerações, contendo informações como bloco, condição e variáveis, se houver, como mostra a Figura 13.

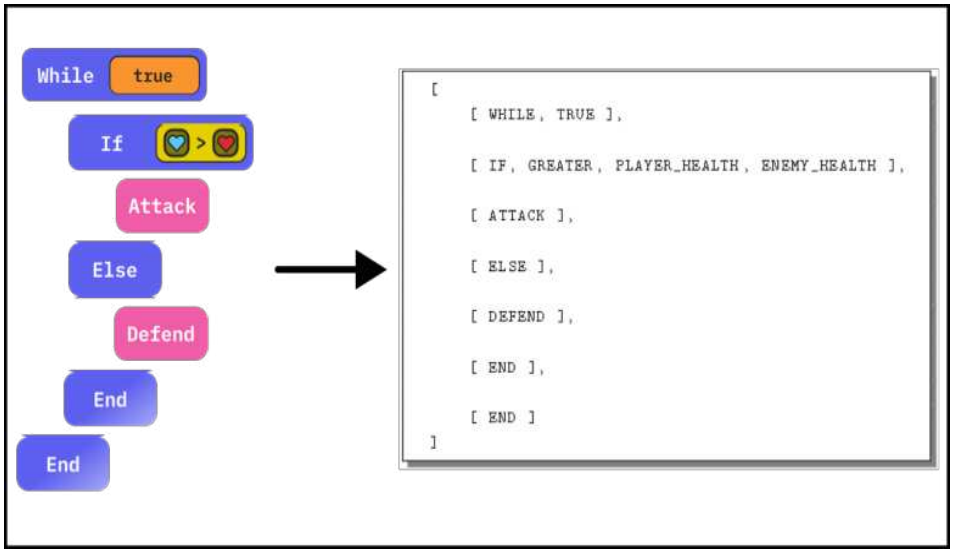


Figura 13 – Conversão de blocos para enumerações

Com a lista de enumerações, é possível compilar seguindo o Código 1, verificando se o algoritmo é válido[2] enquanto a lista é compilada. A compilação consiste em transformar de enumerações para *Cell*, uma classe contendo a ação correspondente e os saltos que devem

[1] Em Unity, um *GameObject* é um elemento fundamental que pode representar objetos, personagens, efeitos visuais, entre outros, dentro do ambiente de desenvolvimento.

[2] Um código é inválido se e somente se ultrapassou o limite de blocos, possui estruturas que são inválidas ou incompletas ou não há ações para realizar.

ser feitos a depender da condição. Para cada tipo de bloco há uma subclasse específica, como apresentado na Figura 14. As classes *IfCell* e *WhileCell* possuem uma condição definida através de um comparador, portanto possuem referência a um *ComparatorCell* com suas subclasses correspondentes, como mostrado na Figura 15. Com uma lista de *Cell*, é possível percorrer a lista com um apontador de instruções que pula para a próxima célula de acordo com a célula atual, descrito pelo Código 2.

Código 1 – Algoritmo de Compilação

```
listaComandos: Lista com os comandos do codigo do jogador
func compilaCodigo(listaComandos):
    celulas <- []
    Para cada comandos em listaComandos:
        comandoPrincipal <- Primeiro comando
        Se comandoPrincipal = {ataque, defesa, carga, cura}:
            celula <- Celula de acao correspondente
            celulas.add(celula)
        Se comandoPrincipal = end:
            Verifica se ha um inicio de bloco
            celula <- Celula de end
            celulas.add(celula)
            Atualiza as quantidades de saltos
        Se comandoPrincipal = else:
            Verifica se ha um if
            celula <- Celula de else
            celulas.add(celula)
            Atualiza as quantidades de saltos
        Se comandoPrincipal = for:
            valor <- Segundo comando
            celula <- Celula de for com valor
            celulas.add(celula)
        Se comandoPrincipal = {if, while}:
            comparador <- Segundo comando
            variavel1 <- Terceiro comando
            variavel2 <- Quarto comando
            celula <- Celula da estrutura com comparador e variaveis
            celulas.add(celula)
    Retorna celulas
```

Figura 14 – Diagrama de classes - Células

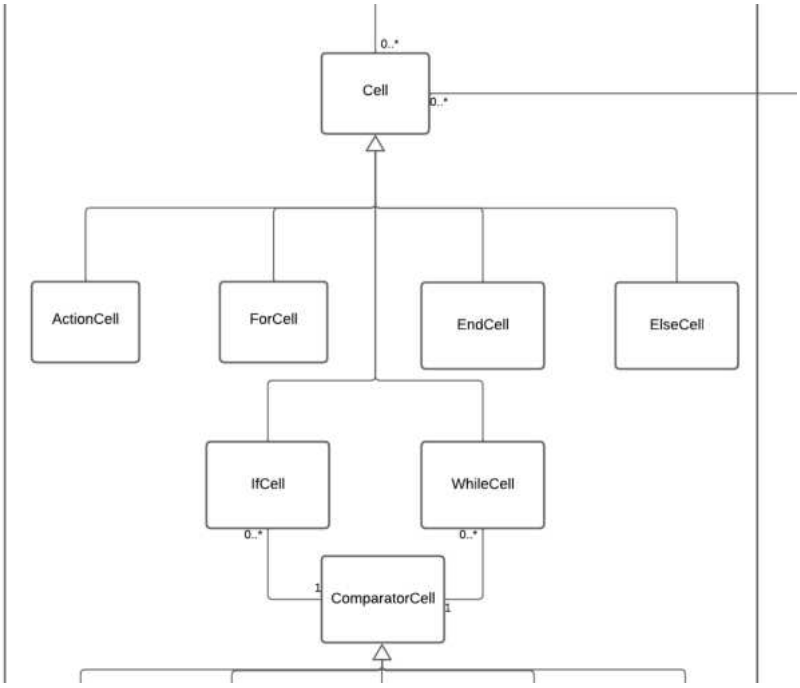
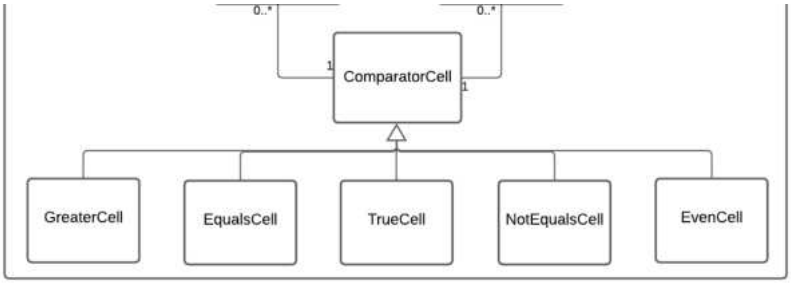


Figura 15 – Diagrama de classes - Comparadores



Código 2 – Algoritmo de Execução

```
listaCelulas: Lista de Celulas retornada pelo compilador
PC: Apontador de instrucoes atual
func executaCodigo(listaCelulas, PC):
    Enquanto PC nao superar tamanho da lista:
        celulaAtual <- memoria[PC]
        Se celulaAtual = acao:
            Retorna acao como atual
        Se celulaAtual = {end, else}:
            PC <- PC + salto da celula
        Se celulaAtual = {if, for, while}:
            Se condicao satisfeita:
                PC <- PC + salto correspondente
            Senao:
                PC <- PC + 1
```

4 RESULTADOS E DISCUSSÕES

O objetivo do trabalho é a criação de um jogo para uso de estudantes de computação. Portanto, se mostra necessário a avaliação por parte desse grupo de pessoas para averiguarmos seus impactos na educação. Com estes dados e opiniões, é possível melhorar problemas levantados para adaptar o jogo à situação dos respondentes.

4.1 AVALIAÇÃO DOS ALUNOS

O presente estudo utilizou um formulário direcionado a diversos estudantes e ex-estudantes da UFRJ, totalizando 10 respostas (Apêndice B). As indagações abordaram distintos aspectos, incluindo dados demográficos, conhecimento prévio, interesse, impactos no aprendizado de pensamento computacional e dificuldades enfrentadas durante a interação com o jogo. Na seção referente aos dados demográficos, os participantes foram questionados sobre sua faixa etária (Figura 16), cor (Figura 17) e gênero (Figura 18). Constatou-se que nove dos respondentes situavam-se na faixa etária entre 18 e 25 anos, nove possuíam autodeclaração de cor branca, e oito identificavam-se como do gênero masculino, evidenciando uma homogeneidade nos resultados obtidos.

Figura 16 – Faixa etária

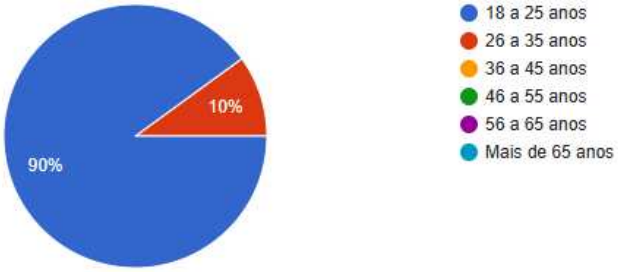


Figura 17 – Cor/Raça

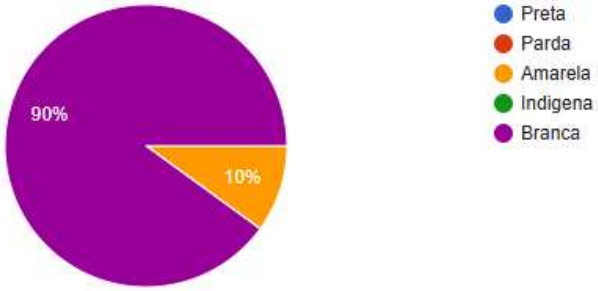
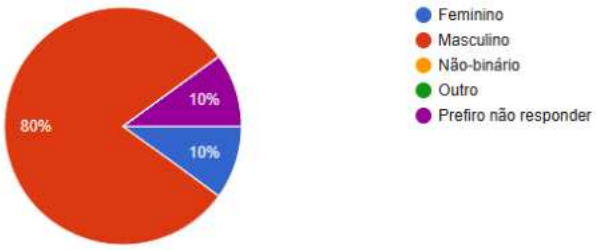


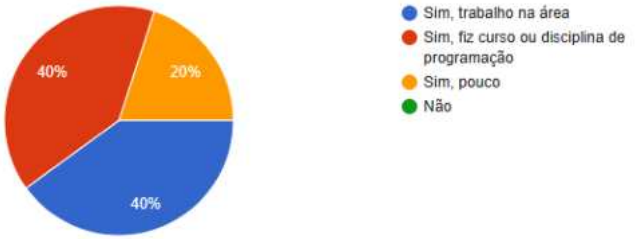
Figura 18 – Gênero



Em relação ao conhecimento prévio dos participantes, as perguntas abordaram a área de formação acadêmica (Figura 19) e o domínio em programação (Figura 20). Verificou-se que seis indivíduos cursavam graduação na área de computação, um proveniente de outra área das ciências exatas, um vinculado às ciências humanas ou sociais, e dois participantes não estavam matriculados em um curso de graduação. Apesar de apenas seis indivíduos serem estudantes da área de computação, todos apresentavam conhecimento prévio em programação, sendo que quatro possuíam experiência profissional na área, quatro haviam adquirido conhecimento em ambiente acadêmico, e dois possuíam conhecimento limitado. Tais resultados são corroborados por estudos anteriores, destacando uma tendência à homogeneidade na área de programação, majoritariamente composta por indivíduos do gênero masculino e autodeclarados como brancos (CASEIRA; MAGALHÃES, 2019) (FRANKLIN, 2013).

Figura 19 – Curso de graduação

Figura 20 – Conhecimento de programação



A seção sobre o interesse dos participantes abordou a frequência de jogos eletrônicos e físicos (Figura 21), o interesse em conquistar medalhas (Figura 22), a disposição em con-

tinuar o jogo após o teste (Figura 23) e o interesse em cada fase específica do jogo (Figura 24). Observou-se que metade dos participantes relatou jogar diariamente, indicando um interesse significativo por jogos. Quanto à busca por medalhas, constatou-se que metade dos participantes buscavam essa conquista sempre, enquanto outros três o faziam apenas quando a fase era considerada fácil, sugerindo que técnicas de ludificação podem despertar interesse por atividades que, de outra forma, não seriam exploradas. Seis participantes demonstraram interesse em continuar o jogo após o teste, evidenciando uma motivação não apenas educacional, mas também lúdica. Quanto ao interesse manifestado em relação a cada fase do jogo, observou-se uma distribuição equilibrada nas primeiras quatro fases, tendendo para um interesse moderado a baixo, com metade das respostas atribuindo nota 1 à primeira fase e as seguintes variando entre 2 e 4. Entretanto, nas últimas quatro fases, houve uma mudança significativa de cenário: embora um número considerável de participantes não tenha jogado, aqueles que o fizeram avaliaram as fases de forma mais equilibrada, com algumas recebendo notas baixas e outras altas, destacando-se como as únicas fases com um índice de interesse mais elevado.

Figura 21 – Frequência de jogo

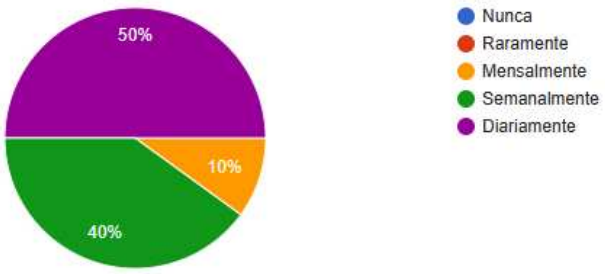


Figura 22 – Interesse em medalhas

No que se refere aos impactos do jogo na educação, o último tópico das perguntas objetivas, os participantes foram indagados sobre se o conhecimento em lógica de programação os auxiliou a completar as fases do jogo (Figura 25), bem como se estariam interessados em ter o jogo como parte da avaliação de uma disciplina (Figura 26). Constatou-se que cinco participantes atribuíram nota 4 ao benefício de ter conhecimentos em lógica de programação, com os demais resultados distribuídos de forma equilibrada. Quanto à aceitação em ter o jogo como parte da avaliação de uma disciplina, metade dos participantes

Figura 23 – Interesse em continuar

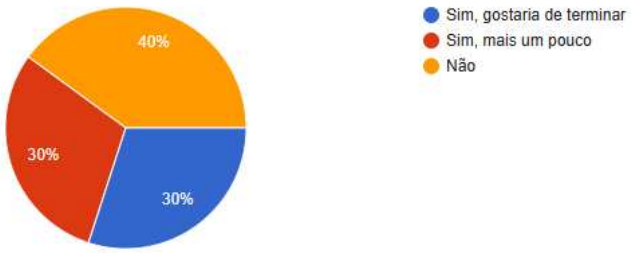
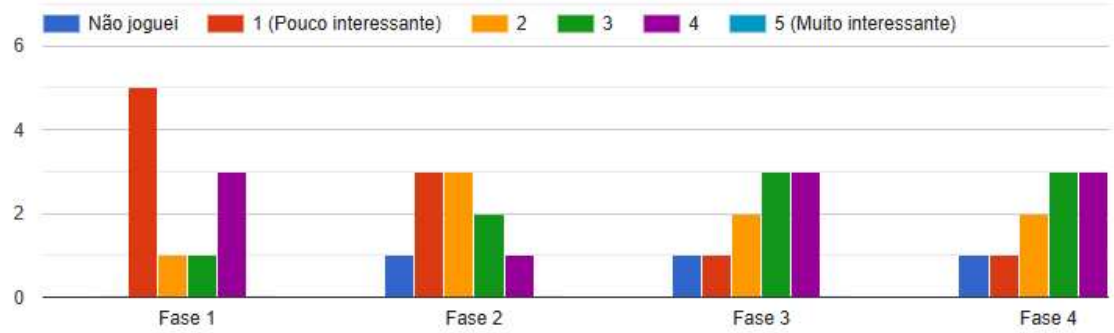
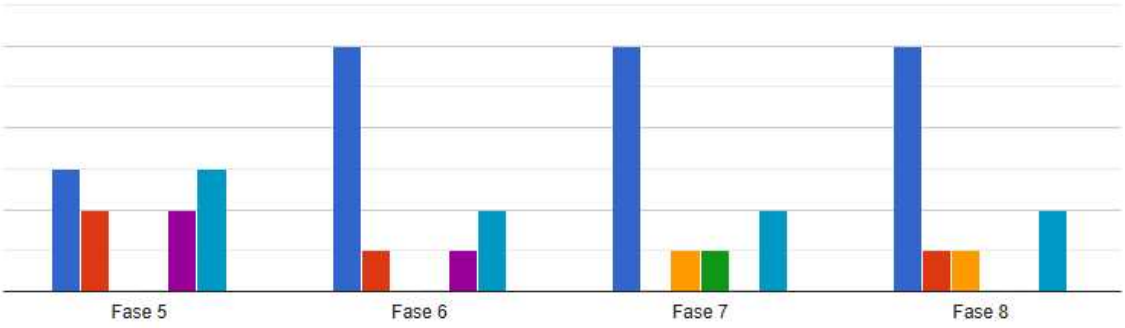


Figura 24 – Interesse nas fases

(a) Fases iniciais



(b) Fases finais



demonstraram interesse nessa proposta, enquanto apenas um discordou da ideia. Tais resultados sugerem que o jogo pode trazer benefícios quando utilizado em contexto educacional, auxiliando os docentes durante disciplinas introdutórias, ao despertar o interesse dos estudantes e promover o desenvolvimento do pensamento computacional necessário para o aprendizado na disciplina.

4.2 MUDANÇAS

Algumas melhorias foram implementadas posteriormente com base nas sugestões obtidas através do formulário de avaliação. Uma crítica recorrente foi a falta de percepção por

Figura 25 – Benefício de lógica de programação

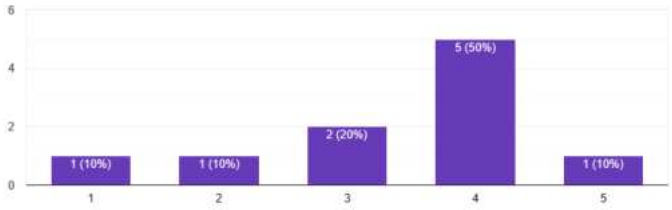
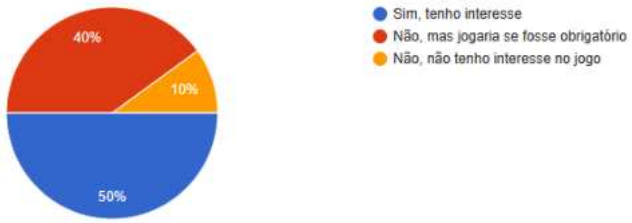


Figura 26 – Interesse em avaliação



parte dos jogadores em relação à existência de um botão de tutorial no menu principal. Diante disso, o botão foi substituído por um elemento mais explícito, visando facilitar sua identificação (Figura 27).

Figura 27 – Botão de tutorial melhorado

Outra crítica relevante diz respeito à confusão observada nos parâmetros de batalha a cada turno. Alguns usuários manifestaram dificuldade em compreender o significado dos valores apresentados no painel de depuração. Como solução, realizou-se uma modificação no texto exibido no painel, de modo a tornar mais claro que os valores ali representados correspondem aos parâmetros atuais dos robôs. Adicionalmente, foi desenvolvido um pequeno painel que exibe os valores iniciais da fase, contribuindo para uma compreensão mais clara e imediata da situação do jogo (Figura 28).

Com o intuito de proporcionar uma experiência mais enriquecedora e divertida, foi implementado um sistema de caixa de texto nas fases. Cada fase agora conta com um pequeno texto introdutório que apresenta o conceito central da fase e oferece dicas relevantes para sua resolução.

Além disso, outras melhorias relacionadas à qualidade de vida do jogador foram realizadas. Agora, mesmo as fases concluídas sem a obtenção de medalhas apresentam

Figura 28 – Interface de batalha melhorada

uma indicação visual de que foram vencidas (Figura 29). Adicionalmente, as soluções das fases concluídas são automaticamente guardadas, mesmo após o encerramento do jogo, garantindo assim uma experiência mais fluida e conveniente para o jogador.

Figura 29 – Seleção de fases melhorada

4.3 AVALIAÇÃO NO LABORATÓRIO

Durante a primeira semana de aulas de calouros de ciência da computação da UFRJ acontece a Semana de Integração, onde são realizadas atividades dos professores com os estudantes novos. Foi realizada uma segunda avaliação de *Robots* com a turma do primeiro período de 2024, onde o jogo foi disponibilizado para que esses estudantes pudessem jogar com as melhorias implementadas e apontar novos problemas, dificuldades e soluções.

Cerca de 40 participantes estiveram presentes na atividade, onde foram divididos em grupos de uma a três pessoas a livre escolha, totalizando 26 equipes, com disponibilidade de uma hora e quarenta minutos com acesso ao jogo. Após esse tempo, as equipes foram individualmente perguntadas sobre a quantidade de fases concluídas, a quantidade de medalhas obtidas e observações apontadas.

Na segunda avaliação cerca de 96% equipes jogaram pelo menos quatro fases, com três dos 26 grupos concluindo todas as fases (Figura 30). A maioria dos grupos conseguiu oito ou nove medalhas, tendo três grupos com todas as medalhas, concluindo assim com perfeição o jogo (Figura 31). Por fim, foi analisado a quantidade de medalhas obtidas pelo total de medalhas disponíveis, considerando somente as fases concluídas, e cerca de

Figura 30 – Distribuição das equipes por fases concluídas

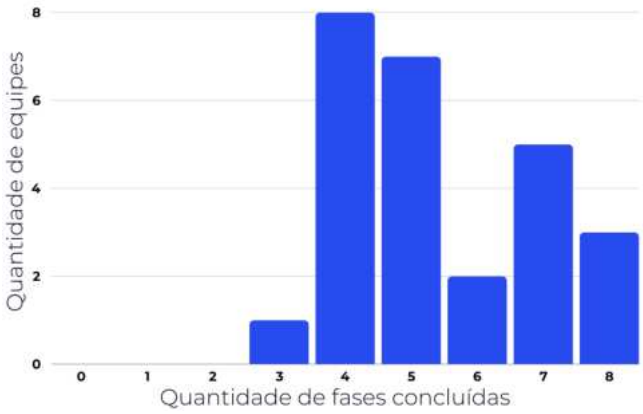
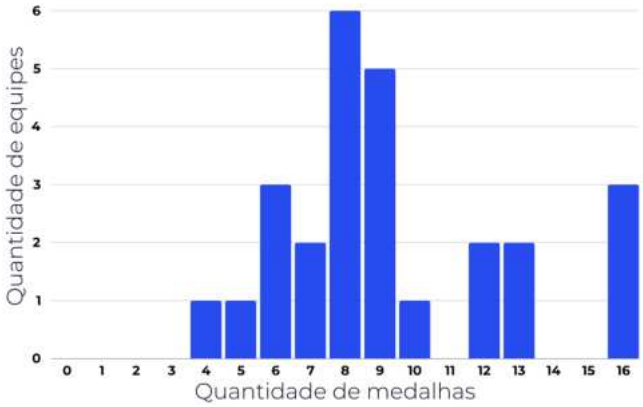


Figura 31 – Distribuição das equipes por medalhas adquiridas



72% das equipes conseguiram entre 90% e 100%, mostrando maior motivação em obter as medalhas com as mudanças realizadas (Figura 32).

Figura 32 – Distribuição das equipes por razão de soluções ótimas

5 CONCLUSÃO

Esse trabalho desenvolveu um jogo eletrônico como uma ferramenta complementar no ensino do pensamento computacional em disciplinas introdutórias de computação. A avaliação do jogo pelos estudantes abordou tanto seus aspectos educacionais quanto lúdicos, proporcionando uma análise abrangente de sua eficácia e aceitação. Embora as respostas tenham sido homogêneas, tendo sido predominantemente de indivíduos jovens, de cor branca e gênero masculino, esses dados refletem a demografia típica dos estudantes do curso de Ciência da Computação.

Foi observada uma maior receptividade às fases mais avançadas do jogo, algumas das quais receberam notas máximas. Possivelmente, os indivíduos com maior conhecimento em programação, que têm maior probabilidade de progredir para as fases mais complexas do jogo, podem representar uma parcela mais engajada e interessada na experiência, o que pode influenciar na maior receptividade observada nessas fases. Com as fases abordando conceitos-chave da disciplina de Programação de Computadores I da UFRJ, os professores têm a possibilidade de utilizar a ferramenta de criação de fases para adaptar o conteúdo de acordo com as preferências e dificuldades da turma em questão, enriquecendo a experiência de aprendizagem. Um possível exemplo de aplicação por parte do docente seria a criação de múltiplas fases contendo o mesmo algoritmo inimigo, com a diferença na disponibilidade dos blocos. A ideia é que o aluno, gradativamente, perceba as diferenças claras entre a aplicação de cada bloco, gerando resultados com desempenhos completamente distintos, em termos de otimização. Ao final, com todos os blocos disponíveis, o aluno teria condições viáveis de programar a solução merecedora de ambas as medalhas possíveis.

Após a avaliação inicial, diversas melhorias foram implementadas, incluindo a adição de dicas para as fases e ajustes na interface. Entretanto, durante a avaliação final, foram percebidos outros problemas que requerem atenção futura como:

- Melhorias na interface
  - Ajustar resolução do jogo para diversos monitores;
  - Dar destaque ao botão de regular a gaveta de blocos;
  - Permitir ver a tela de depuração e programação simultaneamente;
  - Exibir o painel de compilação por cima do painel de dica;
  - Reorganizar blocos e variáveis disponíveis no painel para facilitar visualização.
- Melhorias na experiência
  - Facilitar posicionar blocos no painel de programação;
  - Adicionar efeitos sonoros ao interagir e música de fundo;

  - Integrar o tutorial durante a batalha;
  - Disponibilizar o código fonte do adversário após a conclusão;
  - Facilitar a criação de fases.

- Melhorias do projeto
  - Exigir fornecer uma solução válida ao criar uma fase;
  - Disponibilizar uma estrela vermelha para soluções melhores que as definidas pelas medalhas da fase;
  - Melhorar progressão das fases;
  - Disponibilizar blocos novos posicionados para demonstrar o funcionamento;
  - Adicionar novas mecânicas referentes ao restante da ementa de Programação de Computadores I.

Para o desenvolvimento do compilador dos blocos, foram aplicados conhecimentos adquiridos em disciplinas como "Computadores e Programação" e "Compiladores", podendo futuramente também ser empregados conhecimentos de "Linguagens Formais" para aprimorar as mecânicas do jogo, como permitir que os usuários programem utilizando texto, o que proporcionaria a prática da escrita de programas e sintaxe. Outra sugestão levantada foi implementar conceitos de disciplinas como "Computação Concorrente" e realizar a execução de dois ou mais algoritmos simultaneamente, requerendo a coordenação e sincronicidade para a solução das fases.

Este trabalho concluiu que jogos educativos podem ter impactos positivos no ensino do pensamento computacional, tanto para pessoas com ou sem experiência prévia. Com esse auxílio, será possível facilitar docentes que precisam engajar estudantes e discentes que não se sentem motivados na disciplina. Acredita-se que essas melhorias possam contribuir para a utilização de jogos de código aberto em instituições de ensino brasileiras para melhorar o engajamento em diversas disciplinas de programação, oferecendo uma alternativa acessível e adaptável aos jogos comerciais de código fechado.

REFERÊNCIAS

ALMEIDA, C. et al. Negative effects of gamification in education software: Systematic mapping and practitioner perceptions. **Information and Software Technology**, Elsevier, v. 156, p. 107142, 2023.

BOGOST, I. Why gamification is bullshit. **The gameful world: Approaches, issues, applications**, Mit Press Cambridge, MA, v. 65, p. 65–79, 2015.

CAMARGO, I. R. de; FORTUNATO, I. O scratch como auxiliar no processo de ensino-aprendizagem de linguagem de programação: um balanço da pós-graduação nacional entre 2010 e 2016. **Revista on line de Política e Gestão Educacional**, Universidade Estadual Paulista Júlio de Mesquita Filho, v. 22, n. 2, p. 608–626, 2018.

CARVALHO, L. A.; SERRADO, C. H. B.; OLIVEIRA, F. J. T. S. de. **Game Design Document**. 2021. https://docs.google.com/document/d/16oO3Fok9DwSq22UDODujOJ17ki5UwxFkKaRQBnGEC2s.

CARVALHO, L. A. et al. **Robots**. 2021. https://github.com/CeHaga/robots.

CASEIRA, F. F.; MAGALHÃES, J. C. Meninas e jovens nas ciências exatas, engenharias e computação: Raça-etnia, gênero e ciência em alguns artefatos. **Diversidade e Educação**, p. 259–275, 2019.

Duolingo Inc. **Duolingo**. 2011. https://www.duolingo.com/.

FERREIRA, P. P. S. K.; CANAANE, I. d. O. Desempenho estudantil: uma análise da situação atual do bacharelado em ciência da computação. TCC (Bacharelado em Ciência da Computação) - Instituto de Computação, Universidade Federal do Rio de Janeiro, 2021.

Figma, Inc. **Figma**. 2016. https://www.figma.com/.

FRANKLIN, D. **A practical guide to gender diversity for computer science faculty**. [S.l.]: Morgan & Claypool Publishers, 2013.

Kahoot! ASA. **Kahoot!** 2012. https://kahoot.com/.

KAZIMOGLU, C. et al. A serious game for developing computational thinking and learning introductory computer programming. **Procedia-Social and Behavioral Sciences**, Elsevier, v. 47, p. 1991–1999, 2012.

LIU, C.-C.; CHENG, Y.-B.; HUANG, C.-W. The effect of simulation games on the learning of computational problem solving. **Computers & Education**, Elsevier, v. 57, n. 3, p. 1907–1918, 2011.

Luden.io. **while True: learn()**. 2019. https://store.steampowered.com/app/619150/while_True_learn/.

Microsoft. **GitHub**. 2008. https://github.com/.

PARSONS, D.; HADEN, P. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In: **Proceedings of the 8th Australasian Conference on Computing Education-Volume 52**. [S.l.: s.n.], 2006. p. 157–163.

PUTZ, L.-M.; HOFBAUER, F.; TREIBLMAIER, H. Can gamification help to improve education? findings from a longitudinal study. **Computers in Human Behavior**, Elsevier, v. 110, p. 106392, 2020.

SCHELL, J. **The Art of Game Design: A book of lenses**. [S.l.]: CRC press, 2008.

Scratch Foundation. **Scratch**. 2014. https://scratch.mit.edu/.

SILVA, R. A. d. S. et al. Evasão em computação na ufc sob a perspectiva dos alunos. In: SBC. **Anais do XXIX Workshop sobre Educação em Computação**. [S.l.], 2021. p. 338–347.

Tomorrow Corporation. **Human Resource Machine**. 2015. https://store.steampowered.com/app/375820/Human_Resource_Machine/.

Ubisoft Entertainment. **Valiant Hearts: The Great War™ / Soldats Inconnus : Mémoires de la Grande Guerre™**. 2014. https://store.steampowered.com/app/260230/Valiant_Hearts_The_Great_War__Soldats_Inconnus__Mmoires_de_la_Grande_Guerre/.

Unity Technologies. **Unity**. 2005. https://unity.com/.

WING, J. M. Computational thinking. **Communications of the ACM**, ACM New York, NY, USA, v. 49, n. 3, p. 33–35, 2006.

WING, J. M. Computational thinking and thinking about computing. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, The Royal Society London, v. 366, n. 1881, p. 3717–3725, 2008.

APÊNDICE A – FASES DO JOGO

FASE 1

Blocos disponíveis: Ataque e Cura

Frase:

```
O inimigo é mortal,
Porém sua defesa é banal.
Não basta apenas atacar,
Você vai precisar se curar.
```

Quadro 1 – Parâmetros da fase 1

PARÂMETRO	JOGADOR	INIMIGO
Vida	4	4
Escudos	0	0
Carga	0	0
Dano	1	3

Fonte: Elaboração própria

Quadro 2 – Medalhas da fase 1

TIPO	VALOR
Round	8
Tamanho	8

Fonte: Elaboração própria

Código 3 – Algoritmo do inimigo 1

```
while(true) {
    attack
    defend
    defend
}
```

Código 4 – Solução esperada da fase 1

```
attack
heal
heal
attack
heal
heal
attack
attack
```

FASE 2

Blocos disponíveis: Ataque, Defesa e Cura

Frase:

```
Na sombra do perigo, cuidado é essencial,
Seus golpes são perigosos, um risco letal.
Com pouca vida, defenda-se com coragem,
No jogo da sobrevivência, seja astuto na viagem.
```

Quadro 3 – Parâmetros da fase 2

PARÂMETRO	JOGADOR	INIMIGO
Vida	1	3
Escudos	5	0
Carga	0	0
Dano	1	1

Fonte: Elaboração própria

Quadro 4 – Medalhas da fase 2

TIPO	VALOR
Round	6
Tamanho	6

Fonte: Elaboração própria

Código 5 – Algoritmo do inimigo 2

```
while(true) {
    attack
    defend
}
```

Código 6 – Solução esperada da fase 2

```
defend
attack
defend
attack
defend
attack
```

FASE 3

Blocos disponíveis: Ataque, Defesa, Cura e Charge

Frase:

```
Em pé de igualdade,
Você e o inimigo estão.
Para vencer um ataque incessante,
Carregue duas vezes e vença de supetão.
```

Quadro 5 – Parâmetros da fase 3

PARÂMETRO	JOGADOR	INIMIGO
Vida	4	4
Escudos	3	0
Carga	5	0
Dano	1	1

Fonte: Elaboração própria

Quadro 6 – Medalhas da fase 3

TIPO	VALOR
Round	3
Tamanho	3

Fonte: Elaboração própria

Código 7 – Algoritmo do inimigo 3

```
while(true) {
    attack
}
```

Código 8 – Solução esperada da fase 3

```
charge
charge
attack
```

FASE 4

Blocos disponíveis: Comandos, while, end, true

Frase:

```
O inimigo sem escudos,
Retorna mais ardiloso.
Use bloqueios e ataques astutos,
Em um ritmo sempre viçoso.
```

Quadro 7 – Parâmetros da fase 4

PARÂMETRO	JOGADOR	INIMIGO
Vida	1	12
Escudos	10	0
Carga	3	5
Dano	1	3

Fonte: Elaboração própria

Quadro 8 – Medalhas da fase 4

TIPO	VALOR
Round	14
Tamanho	7

Fonte: Elaboração própria

Código 9 – Algoritmo do inimigo 4

```
while(true) {
    attack
    defend
    defend
    defend
    attack
}
```

Código 10 – Solução esperada da fase 4

```
while(true) {
    defend
    charge
    charge
    attack
    defend
}
```

FASE 5

Blocos disponíveis: Comandos, while, end, true, if, else, even
Variáveis disponíveis: Round
Frase:

```
Entre turnos pares e ímpares,
O inimigo faz uma separação.
Mas após muitos turnos,
Sua estratégia fica uma confusão.
```

Quadro 9 – Parâmetros da fase 5

PARÂMETRO	JOGADOR	INIMIGO
Vida	6	11
Escudos	6	0
Carga	5	5
Dano	1	1

Fonte: Elaboração própria

Quadro 10 – Medalhas da fase 5

TIPO	VALOR
Round	12
Tamanho	9

Fonte: Elaboração própria

Código 11 – Algoritmo do inimigo 5

```
while(true) {
    if(round > 9) {
        heal
        charge
        attack
    } else {
        if(even(round)) {
            heal
            heal
        } else {
            charge
            charge
        }
        attack
    }
}
```

Código 12 – Solução esperada da fase 5

```
while(true) {
    charge
    charge
    if(even(round)) {
        attack
    } else {
        defend
    }
}
```

FASE 6

Blocos disponíveis: Tudo menos for
Variáveis disponíveis: Todas
Frase:

```
O inimigo está se preparando,
Para poder revidar.
Ele irá te atacar,
Assim que a batalha estiver prestes a acabar.
```

Quadro 11 – Parâmetros da fase 6

PARÂMETRO	JOGADOR	INIMIGO
Vida	10	10
Escudos	3	0
Carga	3	10
Dano	1	1

Fonte: Elaboração própria

Quadro 12 – Medalhas da fase 6

TIPO	VALOR
Round	11
Tamanho	9

Fonte: Elaboração própria

Código 13 – Algoritmo do inimigo 6

```
while(true) {
    if ( VidaInimigo > MetadeVidaMaximaInimigo ) {
        charge
    } else {
        attack
    }
}
```

Código 14 – Solução esperada da fase 6

```
while(true) {
    while( VidaInimigo > MetadeVidaMaximaInimigo ) {
        attack
    }
    if ( CargaInimigo != 0 ) {
        defend
    }
    attack
}
```

FASE 7

Blocos disponíveis: Todos
Variáveis disponíveis: Todas
Frase:

```
Você está preparado,
Com bastante defesa.
Estude o inimigo,
E ataque com surpresa.
```

Quadro 13 – Parâmetros da fase 7

PARÂMETRO	JOGADOR	INIMIGO
Vida	1	10
Escudos	10	0
Carga	10	5
Dano	1	3

Fonte: Elaboração própria

Quadro 14 – Medalhas da fase 7

TIPO	VALOR
Round	15
Tamanho	13

Fonte: Elaboração própria

Código 15 – Algoritmo do inimigo 7

```
while(true) {
    for(4) {
        heal
    }
    defend()
    for(5) {
        attack
    }
}
```

Código 16 – Solução esperada da fase 7

```
while(true) {
    for(4) {
        charge
    }
    if( CargaJogador > 8 ) {
        attack
    } else {
        charge
    }
    for(5) {
        defend
    }
}
```

FASE 8

Blocos disponíveis: Todos
Variáveis disponíveis: Todas
Frase:

```
Sempre em prontidão,
De ataques fortes o inimigo defende.
Mas com ataques leves e certeiros,
Eventualmente o oponente se rende.
```

Quadro 15 – Parâmetros da fase 8

PARÂMETRO	JOGADOR	INIMIGO
Vida	1	5
Escudos	5	50
Carga	5	5
Dano	1	3

Fonte: Elaboração própria

Quadro 16 – Medalhas da fase 8

TIPO	VALOR
Round	20
Tamanho	11

Fonte: Elaboração própria

Código 17 – Algoritmo do inimigo 8

```
while(true) {
    for(3) {
        defend
        for(2) {
            defend
            if ( CargaJogador == 0 ) {
                charge
            }
            attack
        }
    }
}
```

Código 18 – Solução esperada da fase 8

```
while(true) {
    for(2) {
        defend
        defend
        attack
    }
    defend
    defend
    defend
    attack
}
```

APÊNDICE B – FORMULÁRIO DE AVALIAÇÃO

1. Você confirma que já jogou o jogo e concorda em participar da pesquisa?

    a) Sim

    b) Não

2. Qual a sua faixa etária?

    a) 18 a 25 anos

    b) 26 a 35 anos

    c) 36 a 45 anos

    d) 46 a 55 anos

    e) 56 a 65 anos

    f) Mais de 65 anos

3. Qual a sua cor?

    a) Preta

    b) Parda

    c) Amarela

    d) Indígena

    e) Branca

4. Qual o seu gênero?

    a) Feminino

    b) Masculino

    c) Não-binário

    d) Outro

    e) Prefiro não responder

5. Você cursou graduação?

    a) Sim, concluído

    b) Sim, cursando

    c) Sim, trancado

d) Não

6. Qual a área de seu curso de graduação?

   a) Computação
   b) Ciências Exatas ou Engenharia (Exceto cursos de computação)
   c) Linguagens
   d) Ciências Humanas ou Sociais
   e) Ciências Biológicas
   f) Não curso/cursei graduação

7. Você já aprendeu programação?

   a) Sim, trabalho na área
   b) Sim, fiz curso ou disciplina de programação
   c) Sim, pouco
   d) Não

8. Com que frequência você joga jogos?

   a) Nunca
   b) Raramente
   c) Mensalmente
   d) Semanalmente
   e) Diariamente

9. Você teve alguma dificuldade em entender como o jogo funciona? (Texto livre)
10. Você terminou quantas fases? (0 a 8)
11. O quão interessante foram as fases? (Para cada uma das 8 fases: Não joguei, 1 (Pouco interessante) a 5 (Muito interessante))
12. O quanto o jogo te ajudou a pensar com lógica de programação? (1 a 5)
13. Você se sentiu motivado a encontrar uma solução ótima e obter as medalhas?

    a) Sim, sempre
    b) Sim, mas somente nas fases fáceis
    c) Sim, mas só a medalha de quantidade de blocos

d) Sim, mas só a medalha de quantidade de turnos

e) Não busquei medalhas

14. Você teria interesse em jogar o jogo como parte de uma disciplina?

a) Sim, tenho interesse

b) Não, mas jogaria se fosse obrigatório

c) Não, não tenho interesse no jogo

15. Você tem interesse em jogar o jogo por conta própria?

a) Sim, gostaria de terminar

b) Sim, mais um pouco

c) Não