AlgoBot: jogo sério para o desenvolvimento do pensamento computacional

Vinicius Higuchi
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC
Santo André - SP, Brasil
vinicius.higuchi@aluno.ufabc.edu.br

Danilo dos Santos Bezerra
Pós-graduação em Ciência da Computação
Universidade Federal do ABC
Santo André - SP, Brasil
danilo.bezerra@ufabc.edu.br

Rafaela Vilela da Rocha
Pós-graduação em Ciência da Computação
Centro de Matemática, Computação e Cognição, UFABC
Santo André - SP, Brasil
rafaela.rocha@ufabc.edu.br

Denise Hideko Goya
Pós-graduação em Ciência da Computação
Centro de Matemática, Computação e Cognição, UFABC
Santo André - SP, Brasil
denise.goya@ufabc.edu.br

***Resumo*—O pensamento computacional é uma habilidade que auxilia a resolução de problemas por meio do uso da abstração, reconhecimento de padrões, decomposição de problemas e pensamento algorítmico. Ela é indispensável e deve ser desenvolvida desde a formação básica. Nesse contexto, o objetivo deste artigo é apresentar a produção do protótipo AlgoBot, um jogo sério digital do tipo quebra-cabeça de programação, com geração procedimental de desafios, para introdução e desenvolvimento do pensamento computacional. As atividades, baseadas no método AIMED (*Agile, Integrative and Open Method for Open Educational Resources Development*), contemplaram três principais etapas: (I) revisão de literatura e projeto do game, (II) implementação e testes com o protótipo (duas iterações) e (III) análise e divulgação dos resultados. Como a mecânica do desafio de ordenação dos blocos de comandos do jogo sério é controlada pela *interface*, a primeira iteração de desenvolvimento teve como prioridade planejar e desenvolver essa *interface*; e na segunda, o foco foi o desenvolvimento dessa mecânica, do *feedback* do progresso e de sua integração com a *interface*. Como resultado inicial, tem-se um protótipo funcional do jogo AlgoBot. A natureza ágil e iterativa do método AIMED permitiu o desenvolvimento de recursos modulares, que podem ser facilmente integrados e modificados em novas iterações, caso necessário.**

Palavras-chave*—Jogo sério, Pensamento computacional, Tecnologia educacional

I. Introdução

O Pensamento Computacional (PC) é uma habilidade que auxilia a resolução de problemas por meio do uso da abstração (concentração em detalhes pertinentes no momento), do reconhecimento de padrões (identificação de padrões, similaridades e conexões em comum), da decomposição de problemas (divisão em partes menores e mais simples de serem resolvidos) e do pensamento algorítmico (aplicação de passos específicos para criação de soluções reusáveis) [1][8][9]. No contexto das novas tecnologias de informação, é uma habilidade indispensável, uma vez que a computação vem transformando a maneira que as mais diversas áreas do conhecimento atuam. Além disso, o PC é considerado pela Sociedade Brasileira de Computação como um dos "pilares fundamentais do intelecto humano, junto com a leitura, a escrita e a aritmética pois, como estas, serve para descrever, explicar e modelar o universo e seus processos complexos" [10]. O PC é uma habilidade que deve fazer parte da formação básica [1], e, portanto, é necessário buscar meios para desenvolvê-la.

Existem jogos como o Lightbot (jogo de *puzzle* baseado em codificação)[11] que utilizam conceitos de programação como parte da mecânica básica de jogo. Fator que, quando bem aplicado, tem potencial para promover o desenvolvimento do pensamento computacional [2]. Como evidenciado por pesquisas que relatam o uso de jogos sérios e de oficinas que desenvolvem o PC em crianças, tais como, noções de abstração de problemas e sequenciamento otimizado de ações [2] [3]. Por exemplo, em [3] foi avaliado um *kit* didático (com ambiente de programação visual em blocos para a plataforma de prototipagem eletrônica Arduino) em uma oficina de aprendizado de programação com robótica feita com sete crianças do ensino fundamental I (3º e 4º anos). Como resultado, foi relatado o aumento das habilidades relacionadas ao PC, pois foi observado que o melhor aproveitamento pode estar ligado à etapa do período operatório concreto (7/8 a 11/12 anos), descrita por Jean Piaget, do desenvolvimento cognitivo [3].

Neste contexto, o objetivo deste artigo é relatar o desenvolvimento e teste inicial com um protótipo de jogo sério digital, nomeado AlgoBot, com o objetivo de introduzir e desenvolver o pensamento computacional em crianças. O AlgoBot é um jogo do gênero quebra-cabeça para resolução de problemas de lógica de programação, com desafios gerados automaticamente de forma procedimental, para dispositivos móveis com *Android*.

Este artigo está organizado: na Seção II são apresentados os principais conceitos sobre jogos sérios e os trabalhos relacionados. Na Seção III é descrito o método de produção do jogo AlgoBot, que é detalhado na Seção IV. As considerações finais são apresentadas na Seção V.

II. Fundamentação e Trabalhos Relacionados

Nesta seção são apresentados os principais conceitos e trabalhos relacionados.

A. Fundamentação Teórica

A definição mais aceita de Jogos Sérios (JS) coloca que são jogos digitais ou analógicos que não tem entretenimento como o propósito primário, mas carregam os aspectos que definem um jogo convencional [6] [7]. Dada essa definição de JS, estes não podem estar desconectados de aspectos

próprios dos jogos. No trabalho de Plass, Homer e Kinzer [4] são explorados os mais diversos aspectos dos JS, e com base em estudos feitos com duas versões de JS, oferecem diversas perspectivas de como cada um destes aspectos pode influenciar no aprendizado. São estes elementos: mecânicas de jogo, identidade visual, narrativa, sistema de motivação, trilha sonora, conteúdo e habilidades [4]. Os autores argumentam que os JS não precisam adotar uma teoria da aprendizagem específica, especialmente por causa de como o *game design* é pensado e como os elementos facilitam o engajamento dos usuários.

Outro conceito importante do aprendizado baseado em jogos, trata-se do ciclo de desafio, resposta (do jogador) e *feedback* (do jogo) [4]. Que, segundo os autores, é completado quando o jogo oferece um desafio diferente baseado no desempenho, ou incentiva o jogador a providenciar uma resposta diferente ao desafio original.

B. Trabalhos Relacionados

O *Furbot* [12] é um jogo para o ensino de conceitos iniciais de programação de forma lúdica. No qual o jogador deve controlar o personagem, no nível fácil, com as setas do teclado; no nível médio, construir uma solução por meio da seleção do comando que se deseja usar; e no nível difícil, há a adição de lógica de repetição. O *Furbot* foi testado com alunos [2]. Eles foram apresentados a uma versão analógica para ambientação com mecânicas do jogo digital, e posteriormente introduzidos ao jogo sério digital. Os autores relataram um expressivo avanço na capacidade de abstração de problemas e sequenciamento otimizado de ações, habilidades características do PC.

No modelo comercial Lightbot [11], desenvolvido pela SpriteBox LLC., o jogador deve criar um algoritmo a partir de blocos para que o personagem percorra toda a fase e complete os objetivos. Diferentemente do jogo *Furbot* [12], a dificuldade é apenas adicionada como complexidade na solução do caminho, e juntamente com o aumento da dificuldade vem a necessidade de usar recursos, como definir funções e usar repetição "chamando" a função dentro da mesma, de uma maneira recursiva.

O Cargo-Bot [13], criado pelo estúdio Two Lives Left, é um jogo que auxilia no desenvolvimento de conceitos de programação. O jogo trabalha noções de algoritmos de ordenamento, no qual o jogador controla um guindaste para empilhar caixas em várias plataformas e busca um ordenamento final específico. Para isso, é necessário programar as ações do guindaste por meio de blocos de códigos simples, o jogador também pode definir funções, que possibilita repetir uma sequência específica de ações.

O RoboZZle [14], criado por Ostrovsky, é composto de um tabuleiro de tamanho variado com estrelas na sua superfície que devem ser coletadas para alcançar o objetivo. O jogo inclui uma maior diversidade de ações e mecânicas, o que possibilita que os jogadores possam se aprofundar em conceitos complexos e específicos da computação, como percorrer construtos parecidos à árvores binárias.

Além dos jogos descritos nesta subseção, existem outros com gêneros distintos voltados ao desenvolvimento do PC, conforme descrito na revisão feita por Costa et al. [15]. Os estudos [2] [3] mostram a possibilidade de trabalhar os fundamentos do pensamento computacional com crianças do ensino fundamental I.

Os trabalhos [11-14], apresentados nesta subseção, são JS do gênero de quebra-cabeças para desenvolvimento do PC, e o Furbot [12] foi avaliado por Mattos et al. [2]. Esses jogos relacionados exploram meios parecidos aos propostos neste artigo para desenvolver o PC, entretanto, a sua criação manual de fases limita a produção do jogo. Dessa forma, a adição da Geração Procedimental de Conteúdo (GPC) (requisito deste projeto para geração dos desafios/mapas) pode auxiliar no processo de desenvolvimento do jogo, deixando essas tecnologias mais disponíveis [16].

III. Método

Para o desenvolvimento do jogo sério foi usado o método AIMED (*Agile, Integrative and Open Method for Open Educational Resources Development*) [5]. A fase de pré-produção do método AIMED contempla o processo de: (1) *Planejamento inicial (concepção do projeto)*: definição dos aspectos pedagógicos de aprendizado, avaliação e *feedback*, e a definição dos requisitos técnicos, tais como, escopo do projeto, ferramentas e *interface*.

A fase de produção (realizada em ciclos iterativos) aborda os processos de: (2) *Análise e planejamento da iteração*: identificação e priorização do que será produzido nas iterações; (3) *Projeto iterativo*: desenvolvimento do projeto, que possui atividades que podem ser realizadas em paralelo, mas que por causa da limitação de pessoal para este projeto serão desenvolvidas de forma sequencial e incremental; (4) *Implementação incremental*: criação dos artefatos planejados, integração de todos os elementos, tais como, artes, áudios e *scripts*; e (5) *Integração, teste e revisão da iteração:* integração dos recursos no jogo, revisão do conteúdo, testes e avaliação da iteração.

IV. Produção do AlgoBot

A seguir é descrita a produção do jogo sério AlgoBot, conforme os processos usados do método AIMED [5].

A. Concepção do projeto

Baseado nas características e modelos descritas na seção de fundamentação e trabalhos relacionados, foi concebido o projeto do AlgoBot, cujo objetivo pedagógico é a introdução e o desenvolvimento do PC por meio de desafios de quebra-cabeça (abstração, reconhecimento de padrões, decomposição de problemas e pensamento algorítmico). O **desafio** proposto pelo jogo é programar o personagem com diversas ações (por exemplo, avançar, virar direita, virar esquerda, pular), para percorrer um mapa gerado proceduralmente, do ponto inicial ao ponto final. Dessa forma, o jogador deve providenciar uma **resposta**, ou seja, uma sequência correta de ações para esse personagem. A partir desta solução fornecida pelo jogador, o jogo oferece um ***feedback*** que avalia a resposta. A avaliação é baseada na eficiência do código, definida como a razão entre a quantidade de ações efetivas e a quantidade total de ações programadas. O diferencial do projeto está em receber os mapas do jogo de uma ferramenta de GPC, que facilita o desenvolvimento e permite maior disponibilidade de desafios. Dessa forma, completou-se o ciclo de *feedback* descrito em [4], por ter diferentes desafios (mapas), com uma ou mais possibilidade de resposta pelo jogador, que recebe *feedback* do seu desempenho. Já a parte estética (identidade visual) do jogo tem influências do título *Final Fantasy Tactics*, com o cenário renderizado em 3D com o personagem bidimensional.

O jogo AlgoBot é compatível com dispositivos móveis *Android*, com uso de *interface touchscreen* e *feedback* visual e sonoro. Para o desenvolvimento foram priorizadas ferramentas *open-source* e gratuitas. A parte visual foi desenvolvida com uso do GIMP, já o jogo foi implementado na *engine* Unity na linguagem de programação C#.

B. Primeira iteração

Conforme descrito na Seção III, cada iteração é separada em quatro processos, e as tarefas realizadas nesta iteração, em cada um dos processos, são descritas a seguir.

1) Análise do planejamento da iteração

Neste processo é feita a identificação do que será produzido, tais como artefatos, e o que será priorizado nesta iteração. Conforme o cronograma, foram definidos três meses para a realização das atividades desta iteração, portanto, foi necessário definir um escopo ideal que apoie a produção da segunda iteração, observando a natureza incremental do método, e que possa ser realizado de forma satisfatória nesse tempo. Como a *interface* é parte integrante e necessária para o jogo, e se encaixa na quantidade de tempo estipulada, foi planejado o desenvolvimento e integração da *interface*, tais como (1) projeto de interface, (2) criação de artefatos, (3) disposição dos artefatos na *engine* de jogo, e (4) desenvolvimento do *front-end* (lógica que rege a interação do usuário com a *interface*).

Como há a necessidade de definir os artefatos a serem desenvolvidos no próximo processo (projeto iterativo), isso foi feito, com um *software* de manipulação de imagem (GIMP). Dessa forma, o modelo da *interface* da tela de jogo e seus artefatos (como imagens de botões e painéis) foram projetados e produzidos nesse processo, pois é mais eficiente esse reúso.

Conforme é apresentado na Fig. 1, que ilustra a composição do modelo da *interface* criada no GIMP com mapa (cubos) gerados na Unity, no canto superior esquerdo há o botão que abre o menu de opções; do lado direito tem o componente da paleta de comandos e onde será construída a instrução para o personagem; na parte inferior, da esquerda para a direita, há o indicador de eficiência, o botão de executar instruções, e o indicador da dificuldade.

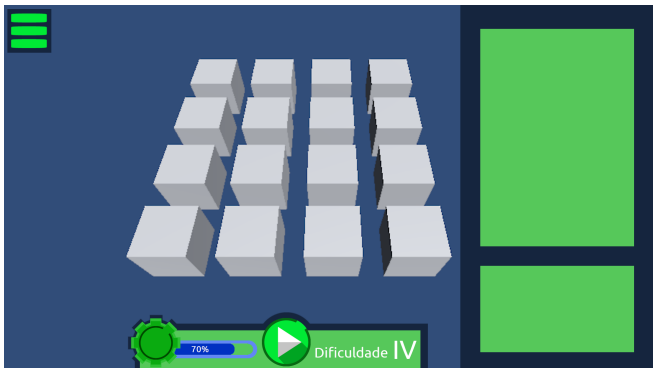

Fig. 1. Modelo da interface (criado pelos autores).

2) Projeto iterativo

Este processo engloba o desenvolvimento do projeto de acordo com o que foi descrito no subtópico 1. Além do que foi projetado inicialmente (modelo de *interface*), foi identificada a necessidade do desenvolvimento dos menus do jogo, de opções e principal.

3) Implementação incremental

No processo de implementação, foram priorizadas a estruturação da *interface* na game *engine* Unity e a programação da lógica do jogo em C#. A Fig. 2, referente a cena de jogo, ilustra o resultado da integração inicial do projeto na Unity.

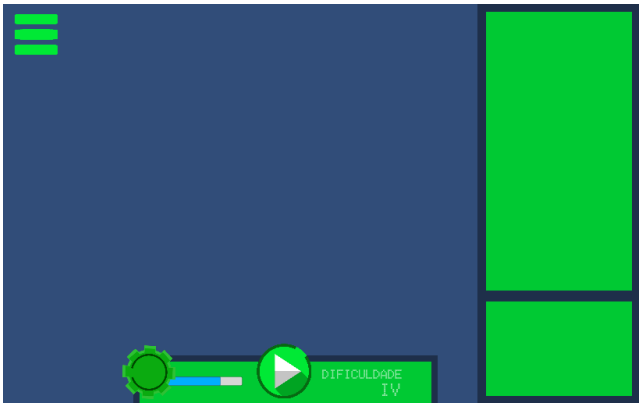

Fig. 2. Tela de jogo na 1a iteração (criada pelos autores).

Também foram implementados protótipos iniciais de *scripts* do sistema de jogo, para fins de teste de aspectos da *interface*, que foi usado como guia para a implementação do sistema na segunda iteração. O enfileiramento das ações por meio da *interface* não foi implementado pois essa *feature* têm uma lógica mais complexa e foi deixada para se trabalhar na segunda iteração, na qual a maior parte da lógica de programação foi desenvolvida.

4) Integração, teste e revisão da iteração

Ao longo do desenvolvimento desta primeira iteração, foram realizados testes de usabilidade, verificação e reconhecimento da existência de *bugs* ao finalizar cada funcionalidade, e também correção dos mesmos.

C. Segunda iteração

1) Planejamento da segunda iteração

Considerando que na primeira iteração de três meses foi consolidada a *interface* de forma satisfatória, coube para a segunda iteração: (1) finalização do enfileiramento de ações na *interface*; (2) desenho ou pesquisa de *sprites* de personagem; (3) implementação das ações com o personagem; e (4) cálculo da eficiência para *feedback* do progresso. Além das tarefas pré-definidas e essenciais, o jogo também foi "polido": adicionado elementos gráficos que contribuem para harmonia visual; revisado experiência do usuário em todas as partes do jogo; testado *scripts* e arrumado eventuais *bugs*.

2) Projeto iterativo

Este processo de projeto tem como objetivo desenvolver o desenho e a animação do personagem. O personagem criado foi um robô. Para isso, foram consultadas diversas referências para guiar seu *design* geral e animação.

3) Implementação incremental

Foram desenvolvidos *scripts* que controlam as ações do personagem, o recebimento do mapa, e o enfileiramento de ações na *interface*, que permite ao jogador controlar o personagem por meio de blocos de código. O mapa do jogo é importado de uma ferramenta de GPC, que gera uma matriz 4x4 de *bits* e armazena-a em um arquivo de texto:

$$M_{4\times4} = \begin{bmatrix} 1000 & 0100 & 0000 & 0000 \\ 0000 & 0010 & 0010 & 0000 \\ 0000 & 0000 & 0100 & 0010 \\ 0000 & 0000 & 0000 & 0001 \end{bmatrix} \quad (1)$$

Cada sequência de *bits* representa diferentes objetos, conforme a Tabela I, que o jogo interpreta e organiza na tela.

TABELA I. CORRESPONDÊNCIA ENTRE BITS E ELEMENTOS.

Bits	Elemento
0000	Elemento vazio
0001	Bloco inicial
0010	Bloco regular
0100	Bloco deslocado para cima
1000	Bloco Final

O método AIMED possibilitou uma arquitetura modular dos recursos do jogo, de modo que os *scripts* desenvolvidos podem ser refatorados de forma individual, sem alterar o sistema como um todo.

4) Integração, teste e revisão da iteração

As funcionalidades planejadas foram integradas na cena de jogo com o uso de recursos desenvolvidos nos processos anteriores. Além do que foi inicialmente planejado, verificou-se a necessidade da implementação de um tutorial simples, que explica o fluxo do jogo.

A Fig. 3 mostra a cena de jogo no estado final da segunda iteração. Centralizado está um mapa de jogo (representado pela matriz de *bits (1)*), criado manualmente para testes. A direita há a paleta que permite que o jogador 'arraste' um bloco à uma posição. Quando o jogador coloca um bloco no espaço disponível, é criado outro espaço ao lado ou na próxima linha. Dessa forma, fica subentendida a ordem na qual os comandos serão reproduzidos.

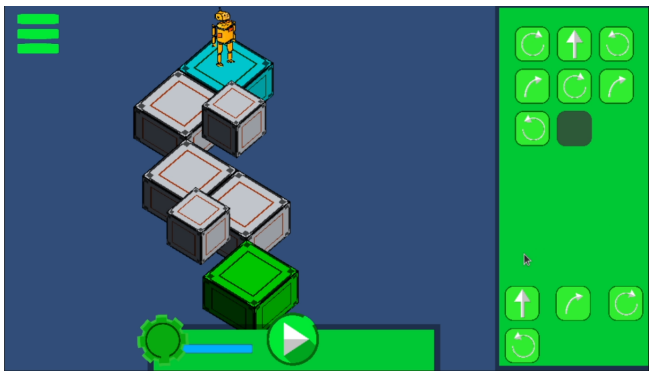

Fig. 3. Tela de jogo na 2a iteração (criada pelos autores).

O protótipo do AlgoBot foi inicialmente testado por dois especialistas com experiência no desenvolvimento e avaliação de jogos sérios. A seleção dos especialistas foi por conveniência (experiência prévia dos avaliadores) e os testes foram executados de modo informal, sem participação dos pesquisadores ou coleta de dados no jogo. Entretanto, os especialistas avaliaram e forneceram *feedback* com captura das telas e sugestões detalhadas. Os testes apontaram possíveis melhorias na harmonia visual da *interface*, mudanças nos ícones que indicam cada ação, para melhorar a experiência do usuário final, e adição de uma funcionalidade que visa mostrar qual comando está sendo executado.

V. CONSIDERAÇÕES FINAIS

Por meio do uso do método AIMED, o protótipo do jogo AlgoBot foi produzido com duas iterações de três e quatro meses. Como a mecânica do jogo é controlada pela *interface*, na primeira iteração foram priorizados o planejamento e desenvolvimento da *interface*. Ao passo que na segunda iteração foram focados o desenvolvimento da mecânica e do *feedback* e a integração com a *interface*, além do desenvolvimento do tutorial.

Quanto aos processos e ferramentas usadas no projeto, é notável que a forma ágil e iterativa do método AIMED permitiu o desenvolvimento de *assets* modulares, que podem ser facilmente integrados e modificados caso necessário. Com o uso do GIMP foram facilmente criados os elementos visuais usados no projeto, e integrados na *engine* Unity. Por fim, o processo de desenvolvimento levou à consolidação de um protótipo funcional do jogo AlgoBot.

Futuramente, devem ser adicionados os mapas criados por um modelo de GPC em desenvolvimento no Laboratório de Informações em Redes e Tecnologias Educacionais (LIRTE) da UFABC, previsto no projeto do jogo. Novos testes com usuários finais deverão ser realizados para validação do jogo como ferramenta de desenvolvimento do pensamento computacional.

REFERÊNCIAS

[1] J. M. Wing, "Computational thinking", in IEEE Symposium on Visual Languages and Human-Centric Computing, 2011, p. 3.

[2] M. Mattos, et al., "Uma pesquisa-ação sobre o desenvolvimento do pensamento computacional com crianças." in WIE, 2018, p. 421-429.

[3] R. Queiroz, F. Sampaio and M. Santos, "DuinoBlocks4Kids: utilizando Tecnologia Livre e materiais de baixo custo para o exercício do Pensamento Computacional no Ensino Fundamental I por meio do aprendizado de programação aliado à Robótica Educacional", in: Workshop do CBIE, 2017, p. 25-34.

[4] J.L. Plass, B.D. Homer and C.K. Kinzer, Foundations of Game-Based Learning, Educational Psychologist, 50(4), 2015, p. 258-283.

[5] R. V. Rocha, et al., "AIMED: Agile, Integrative and Open Method for Open Educational Resources Development." in IEEE International Conference on Advanced Learning Technologies, 2017, p. 163-167.

[6] T. Susi, M. Johannesson and P. Backlund, Serious Games: an overview. 2007.

[7] A. De Gloria, et al., "Serious Games for Education and Training", in: International Journal of Serious Games, 1(1), 2014. p. 1-15.

[8] A. Csizmadia, Computational thinking-a guide for teachers. Computing at School, 2015.

[9] L. Liukas, Hello Ruby: adventures in coding. Macmillan, 2015.

[10] SBC, Diretrizes para ensino de Computação na Educação Básica, 2019.

[11] Spritebox LLC., LightBot Inc, 2017.

[12] L. Araújo, H. Silveira and M. Mattos, "Ensino do pensamento computacional em escola pública por meio de uma plataforma lúdica." in: Workshops CBIE, 2018, p. 589-5

[13] Two Lives Left, Cargo-Bot, 2012.

[14] Ostrovsky, RoboZZle, 2009.

[15] S. S. Costa, et al, "Um estudo exploratório dos games para introdução ao pensamento computacional", in: Congresso Nacional de Ambientes Hipermídia para Aprendizagem, 2015, p.1-15.

[16] D. S. Bezerra, F. I. Massetto and R. V. Rocha, "Design de Jogos Eletrônicos de Quebra-Cabeças usando Geração Procedural", in: SBGames, 2019, p. 331-334.